Süni İntellekt: Maşın öyrənmə üsulları

Nigar İsmayılova

Bakı, 2023

AZƏRBAYCAN DÖVLƏT NEFT VƏ SƏNAYE UNİVERSİTETİ

İSMAYILOVA N.T.

SÜNİ İNTELLEKT: MAŞIN ÖYRƏNMƏ ÜSULLARI

(Dərs vəsaiti)

Azərbaycan Dövlət Neft və Sənaye Universitetinin 2 noyabr 2023-cü il tarixli 3-29-50/3-1-3812/2023 №-li əmri ilə nəşr hüququ verlmişdir.

BAKI - 2023

Müəllif: İsmayılova Nigar Tahir qızı. *"Süni intellekt: maşın öyrənmə üsulları" adlı dərs vəsaiti.* Bakı, 2023, 118 səh.

Elmi Redaktor: Azərbaycan Dövlət Neft və Sənaye Universitetinin "Ümumi və tətbiqi riyaziyyat" kafedrasının müdiri, Əməkdar elm xadimi, r.e.d, prof. A.R.Əliyev

Rəy verənlər:
Azərbaycan Texniki Universitetinin "Kibertəhlükəsizlik" kafedrasının müdiri, t.e.d., dos. Y.N. İmamverdiyev

Azərbaycan Dövlət Neft və Sənaye Universitetinin "Sənayedə və iqtisadiyyatda intellektual idarəetmə və qərar qəbuletmə sistemləri" elmi-tədqiqat laboratoriyasının müdiri, t.e.d., dos. O.H.Hüseynov

Dərs vəsaiti ADNSU Azİİ Kitab Evi-nin biblioqrafı ***Vəliyeva Fidan Arif qızı*** *tərəfindən korrektə olunmuşdur.*

Dərs vəsaitində maşın öyrənmə üsullarının əsasları, onların qiymətləndirmə metrikaları, bu üsulların hiperparametrlərinin optimallaşdırılması təsvir olunmuşdur. Həmçinin klassifikasiya, proqnozlaşdırma və müəllimsiz öyrənmə üsulu olan klasterləşdirmə üsullarının real verilənlər çoxluğuna tətbiq olunması və Python mühitində realizasiyası öz əksini tapmışdır. Hər bir fəslin sonunda isə mövzuya uyğun olaraq çalışmalar verilmişdir.

ISBN 978 - 9952 - 39 - 103 - 9

Mündəricat

Giriş

1950-ci illərdə süni intellektin ataları sayılan Minski və Makkarti bu sahə üçün belə bir tərif vermişdilər: süni intellekt proqram və ya maşın tərəfindən icra olunan elə bir məsələyə deyilir ki, insan eyni məsələni həll edərkən intellekt istifadə etməli olur. Süni intellekt sistemləri insan intellekti ilə bağlı aşağıdakı davranışlardan heç olmasa birini özündə ehtiva edir: planlaşdırma, öyrənmə, əsaslandırma, problem həlli, biliklərin təsviri, qavrayış, hərəkət, manipulyasiya, sosial intellekt və kreativlik [1].

Con Makkarti və Marvin Minski tərəfindən 1956 - cı ildə təşkil olunan Süni İntellektdə Dartmouth Yay Tədqiqat Layihəsi (DSRPAI) tarixi konfransı müxtəlif sahələrdən ən yaxşı tədqiqatçıları bir araya topladı. 1957-ci ildən 1974-cü ilədək süni intellekt çiçəkləndi, kompüterlər daha çox informasiya saxlaya bilməklə yanaşı daha sürətli, daha ucuz və daha əlçatan oldu. Maşın öyrənmə alqoritmləri də inkişaf etdi və geniş tətbiq olunmağa başlandı. Hökumət də öz növbəsində xüsusilə danışıq dilini transkripsiya və tərcümə edə bilən, eləcə də yüksək ötürücülü məlumatların işlənməsini yerinə yetirilə bilən maşınların hazırlanması ilə maraqlanırdı. Optimizm və gözləntilər daha yüksək idi. 1970-ci ildə Marvin Minski Life Magazine jurnalına verdiyi müsahibədə demişdi: "Üç ildən səkkiz ilə qədər orta insanın ümumi zəkasına malik bir maşınımız olacaq." Bu prinsipin qismən sübutu mövcud olsa da, maşınlar tərəfindən təbii dilin işlənməsi, mücərrəd düşüncə və özünü tanıma kimi son məqsədlərə çatmaq üçün hələ çox zaman və yol vardı [2].

Süni intellektin mövcud olmasından bu günə qədər xeyli sayda uğurlu layihələrin hazırlanmasına baxmayaraq bu sahədə uğursuz layihələr də mövcuddur.

Uğurlu süni intellekt layihələri

Maşın tərcüməsi. Son illər ərzində Google Translate az sayda dillərdən inkişaf edərək hal-hazırda 100-dən çox dillərdə olan mətnləri, gün ərzində isə 140 milyarddan çox sözləri tərcümə edir [3].

Nitqin tanınması. Techtarget-ə əsasən nitqin tanınması danışıq dilindən sözlərin və söz birləşmələrinin təyin olunmasına və maşın tərəfindən oxuna bilən hala gətirilməsinə deyilir. Nitqin tanınması üçün cəhdlər 1950-ci ildən başlamışdır, lakin 1990-cı illərin sonunda təbii dilin tanınması yerinə yetirilmişdir. Cəmiyyəti cəzb edən ilk layihə insan nitqini tanıya bilən Apple şirkətinin Siri adlı virtual assistenti olmuşdur. Bu gün nitqin tanınması texnologiyası hər yerdədir. İstehlakçılar və texnoloqlar səsli əmrin faydalarını dərk edirlər, məsələn, öyrənmə fərqləri və ya

məhdudiyyətləri olan insanlara kömək etmək, sənədləşmə işlərini, vaxt və biznes əməliyyatları ilə bağlı xərcləri azaltmaq, və s. Süni intellekt artıq səsli əmrlərin imkanlarını sürətləndirməyə başlayıb. Google Home və Amazon Echo çox istifadə olunan istifadəçi gözləntilərini qarşılamaq və evdə gündəlik işləri avtomatlaşdırmaq üçün süni intellekt və nitqin tanınması əsasında səsli əmr texnologiyaları ilə işləyən nümunələrdir [4].

Üz tanınması şəxsin simasının cizgilərinə əsasən şəxsin kimliyinin təyin olunması prosesidir. 2014-cü ildə Facebook Research DeepFace adlı üz tanınma sistemini təsvir edən məqalə yayınlamışdır. Bu sistem adlandırılmış üzlər verilənlər bazasındakı (LFW) simaların 97.35 % - ni düzgün tanıya bilən 120 milyon parametrdən ibarət dərin neyron şəbəkədir. Məxfilik tərəfdarları geniş yayılmış üz tanıma sisteminin tətbiqinə qəti şəkildə qarşı çıxırlar, çünki bu, bəzi qurumlara (istər şirkət, istərsə də hökumət) izdihamın ixtiyari şəkillərini və videolarını çəkməyə və orada olan hər bir şəxsi müəyyən etməyə imkan verəcək, bu da anonim qalmaq imkanını aradan qaldıracaq [5].

Uğursuz süni intellekt layihələri

Onkologiya üçün Watson. IBM şirkətinin bu layihəsi 62 milyon ABŞ dolları investisiyadan və təhlükəli müalicə təkliflərindən sonra ləğv edildi. 2013-cü ildə Texas Universitetinin MD Anderson Xərçəng Mərkəzinin IBM ilə birgə xərçəng xəstəliyinin müalicəsi üçün koqnitiv hesablama əsaslı Watson sistemi yaradılmışdır. Bu sistem şiddətli qanaxması olan bir xərçəng xəstəsinə qanaxmanı pisləşdirə biləcək bir dərman verməsini təklif etdiyi bir hadisə də daxil olmaqla çoxsaylı təhlükəli və düzgün olmayan müalicə üsulları təklif etmişdir [6].

Microsoft Twitter çatbotu. 2016-cı ildə Microsoft şirkətinin "danışıq anlayışı"nda təcrübə adlandırdığı Tay - Twitter çatbotu yaradıldı. Microsoft-un dediyinə görə, Tay ilə nə qədər çox söhbət etsəniz, o, "təsadüfi və əyləncəli söhbət" vasitəsilə insanları cəlb etməyi öyrənərək bir o qədər ağıllı olur. Təəssüf ki, söhbətlərin keyfiyyəti uzun müddət davam etmədi. Tay işə salındıqdan qısa müddət sonra insanlar bu botla hər cür misoginist, irqçi və Donald Trumpist ifadələrlə dialoqa başladılar və Tay – mahiyyətcə internet bağlantısı olan robot tutuquşu olmaqla bu fikirləri istifadəçilərə geri qaytarmağa başladı [7].

Dərin neyron şəbəkələr üçün tək-piksel hücumları. Təsvirlərin tanınması sahəsində dərin neyron şəbəkələrə əsaslanan yanaşma təsvirlərin emalının ənənəvi üsullarını üstələyib, hətta insanla rəqabət edə bilən nəticələrə nail olmuşdur. Bununla belə, bir sıra tədqiqatlar göstərir ki, təbii təsvirlər üzərindəki süni dəyişikliklər asanlıqla süni neyron şəbəkəni yanlış təsnif etməyə yönəldə bilər və buna görə də "rəqib təsvirlər" adlanan belə nümunələrin yaradılması üçün effektiv alqoritmlər təklif edilir [8].

Süni intellekt nə qədər təhlükəli ola bilər?

Əksər tədqiqatçılar, super intellektli süni intellektin sevgi və ya nifrət kimi insan emosiyalarını nümayiş etdirmə ehtimalının az olduğu və süni intellektin qəsdən xeyirxah və ya pis niyyətli olmasını gözləmək üçün heç bir səbəb olmadığı

ilə razılaşırlar. Bununla yanaşı süni intellektin necə bir riskə çevrilə biləcəyini nəzərdən keçirərkən, mütəxəssislər əsasən iki ssenarini düşünürlər [9]:

Süni intellekt dağıdıcı bir ünsür üçün proqramlaşdırılmışdır. Avtonom silahlar öldürmək üçün proqramlaşdırılan süni intellekt sistemləridir. Yanlış adamın əlində bu silahlar asanlıqla kütləvi itkilərə səbəb ola bilər. Üstəlik, süni intellektlə silahlanma yarışı təsadüfən süni intellektlə müharibəyə səbəb ola bilər ki, bu da kütləvi itkilərlə nəticələnər.

Süni intellekt faydalı bir məsələ üçün proqramlaşdırılmışdır, lakin məqsədinə çatmaq üçün dağıdıcı bir üsul yaradır. Bu, süni intellektin məqsədlərini öz istəklərimizlə tam uyğunlaşdıra bilmədiyimiz zaman baş verə bilir. Əgər siz itaətkar bir ağıllı avtomobildən sizi hava limanına mümkün qədər tez çatdırmağı xahiş etsəniz, o, sizi vertolyotlarla təqib etdirilmə və qusma həddinə çatdırmaqla istədiyinizi deyil, sözün əsl mənasında dediyinizi edə bilər. Əgər super intellektli sistemə iddialı bir geomühəndislik layihəsi tapşırılarsa, bu, yan təsir kimi ekosistemimizə ziyan vura bilər və insanların onu dayandırmaq cəhdlərini qarşısı alınacaq təhlükə kimi görə bilər.

Bu nümunələrdən göründüyü kimi super ağıllı süni intellekt öz məqsədlərinə çatmaqda çox yaxşı olacaq və əgər bu məqsədlər bizim istəklərimizə uyğun gəlmirsə, bu halda böyük problemlə qarşılaşarıq. Yəqin ki, siz qarışqaları nifrətlə əzən bir insan deyilsiniz, amma əgər siz su elektrik stansiyasının yaşıl enerji layihəsinə cavabdehsinizsə və bölgədə su altında qalacaq qarışqa yuvası varsa, bu hal qarışqalar üçün çox pisdir. Süni intellektlə bağlı təhlükəsizlik tədqiqatlarının əsas məqsədi insanlığı heç vaxt bu qarışqaların yerinə qoymamaqdır.

Süni intellektin iki əsas mürəkkəbliyi mövcuddur. Bunlardan birinci hesablama mürəkkəbliyidir. Bir çox məsələləri polinom əsaslı zamanda yerinə yetirə bilərik, lakin əsasən süni intellekt məsələləri eksponensial kompleksliyə malik məsələlərdir.

İkinci mürəkkəblik mənbəyi informasiya kompleksliyidir. Tutaq ki, sonsuz imkanlı hesablama resursları olan bir otaqda bağlı qalsaq, hansısa bir cümləni tərcümə etmək və ya Yeni Zellandiyadan bir quşun tipini təyin etmək lazım gələrsə, hər iki halda hesablama gücünün artırılması kömək etməyəcək. Bu problemlərin həllində optimal qərarlar qəbul etmək üçün ya xarici dil haqqqında, ya da ornitologiya haqqında biliyə ehtiyacımız var.

Süni intellekt modellərinin əsasında hazırlandığı alqoritmlər əsasında bu sistemlər müəyyən mənada aşağı səviyyəli intellektdən yüksək səviyyəli zəkaya doğru irəliləyir, yalnız refleksiv qərar verən modellərdən məntiqi əsaslandırma ilə hazırlanan modellərə doğru inkişaf edir. Refleks əsaslı modellər sadəcə olaraq verilmiş giriş üzrə sabit hesablamalar ardıcıllığını yerinə yetirir. Bu modellərə sadə xətti təsnifat üsullarından dərin neyron şəbəkələrə qədər maşın öyrənməsinin əksər üsulları aiddir. Refleks əsaslı modellərin əsas xarakteristikası ondan ibarətdir ki, bu tip modellərdə hesablamalar yalnız irəliyə doğru aparılır, geri qayıtma olmur və alternativ hesablamalardan istifadə olunmur. Refleks əsaslı modellər daha yüksək intellekt tələb edən məsələlərin həlli üçün (məsələn, şahmat oyunu, yol xəritəsinin planlaşdırılması, və s.) çox primitivdir. Bu tip məsələlərin həllini yüksək səviyyədə həyata keçirmək üçün hal əsaslı, dəyişən əsaslı və ya məntiq

əsaslı modellərin tətbiqi daha məqsədəuyğundur.

Baxılan kitab süni intellektə giriş adlandığından burada daha sadə və yüksək intellekt tələb etməyən məsələlərin həllində istifadə olunan refleks əsaslı maşın öyrənmə modelləri təsvir olunmuşdur. İlk fəsildə verilənlərin tipləri, ölçülmə meyarları və mövcud olmayan verilənlərin bərpası istiqamətində verilən məlumatlarla başlayaraq, ikinci fəsildə maşın öyrənmə üsulları haqqında ümumi məlumatlar verilmişdir. Növbəti fəsillərdə proqnozlaşdırma, klassifikasiya üçün tətbiq olunan müxtəlif üsullar, onların nümunələr üzərində izahı və bu modellərin Python mühitində implementasiyası verilmişdir. Sonuncu fəsildə isə çıxışı məlum olmayan verilənlərin emalı üçün sadə klasterləşdirmə üsulları təsvir olunmuşdur. Kitab *Kompüter elmləri* ixtisasının *Süni intellekt* kursunun birinci hissəsi üçün nəzərdə tutulsa da, informasiya texnologiyaları, hətta digər texniki ixtisaslarda təhsil alan, verilənlərin emalı və maşın öyrənmə üzrə ilkin biliklər əldə etmək istəyən bakalavr və magistr tələbələrinə faydalı olacaqdır.

Fəsil 1

Verilənlərin emalı

Maşın öyrənmə metodlarını müzakirə etməzdən öncə süni intellektin əsas obyekti olan verilənlər, onların tipləri, emal zamanı yaranan problemlərin təyini və onların həlli məsələləri vacibdir. Verilənləri emal etməzdən öncə atributları və verilənlərin qiymətlərini təhlil etmək lazımdır. Real dünyada əldə etdiyimiz verilənlər adətən küylü, nəhəng ölçülərə malik (çox hallarda bir neçə Gbayt və daha artıq) olur və qarışıq, heterogen mənbələrdən əldə olunur.

Verilənlər çoxluqları obyektlərdən ibarət olur. Verilən obyekt hər hansı bir nəsnəni ifadə edir, məsələn satışlar haqqında verilənlər bazasında obyekt müştərilər, satılan əşyalar və ya satışlar; tibbi verilənlər bazasında xəstələr; universitet verilənlər bazasında tələbələr, müəllimlər və ya dərslər ola bilər. Verilən obyektlər adətən atributlarla ifadə olunur, həm də nümunələr, nöqtələr, hallar kimi də adlandırıla bilər. Əgər bu obyektlər verilənlər bazasında saxlanılırsa, onda onlar verilənlər dəstələri adlanır. Verilənlər bazasında sətirlər verilən obyektlərə, sütunlar isə atributlara uyğun gəlir.

1.1 Atributların təsnifatı

Atribut verilən obyektin xarakteristikasını və ya əlamətini təsvir edən qiymətdir. Bu termin üçün maşın öyrənmədə *əlamət*, statistikada isə *dəyişən* sözləri istifadə olunur. Verilmiş atribut üçün müşahidə olunan qiymətlər adətən müşahidələr adlanır. Verilmiş obyekti təsvir edən atributlar çoxluğu *atribut vektoru* və ya *əlamət vektoru* adlanır. Atributların tipləri mümkün qiymətlər çoxluğu ilə təyin olunur, nominal, ordinal, binar və ya ədədi ola bilər.

1.1.1 Nominal atributlar

Nominal sözünün mənası ada əsaslanan deməkdir. Nominal atributların qiymətləri simvollar və ya əşyaların adından ibarətdir. Hər bir qiymət hansısa kateqoriya, kodu göstərir, buna görə də nominal atributlara bəzən kateqoriya atributları da deyirlər. Bu qiymətlər hansısa bir düzülüşə malik olmur. **Məsələn**, *saç_ rəngi* və *ailə_ vəziyyəti* şəxsləri təsvir edən iki atributdur. Baxılan nümunədə *saç_ rəngi*

atributu qəhvəyi, qara, qırmızı, açıq qəhvəyi və ağ, *ailə_ vəziyyəti* atributu isə subay, ailəli və dul qiymətlərindən birini ala bilər.

Nominal atributların əşyaların adı ilə təsvir olunduğunu qeyd etdiyimizə baxmayaraq, nominal atributlar rəqəmlərlə də ifadə oluna bilər. Məsələn, yuxarıda qeyd olunan nümunədə *saç_ rəngi* atributunun qiymətləri üçün qara rəngi 0, qəhvəyini 1, və s. ilə ifadə etmək olar. Digər nümunə *müştəri_ identifikasiya_ nömrəsi* atributudur ki, burada bütün qiymətlər ədədidir. Lakin bu qiymətlər üzərində hesablama əməliyyatlarını yerinə yetirmək düzgün deyil. Nominal atributlar ədədi olmadığından və məntiqli düzülüşə malik olmadığından bu atributların orta qiyməti və ya medianının hesablanması məqsədəuyğun deyil. Burada maraq kəsb edən kəmiyyət ən çox rast gəlinən qiymətdir, buna görə də nominal atributlarda modanın hesablanması mərkəzi yayılma haqqında lazımi informasiyanı verir.

1.1.2 Binar atributlar

Binar atributlar yalnız iki qiymət: 0 və ya 1 alan nominal atributdur, burada 1 atributun mövcudluğunu, 0 - isə mövcud olmamasını bildirir. Əgər bu iki hal *doğru* və ya *yanlış* qiymətlərini alarsa onda bu atributlara bul dəyişəni də deyilir. **Məsələn**, xəstə obyekti üçün *siqaret_ çəkir* atributu üçün 1 qiyməti xəstənin siqaret çəkməsini, 0 qiyməti isə siqaret çəkməməsini bildirir və ya əgər *tibbi_ test* atributu binardırsa, ona 0 qiyməti testin neqativ, 1 qiyməti isə testin pozitiv olmasını bildirir.

Binar atributun hər iki qiyməti eyni çəkiyə malikdirsə, yəni hansı qiymətin 0 və ya 1 ilə ifadə olunmasının elə bir fərqi yoxdursa, onda bu atribut simmetrik adlanır. Nümunə olaraq, *cins* atributunun kişi və qadın qiymətlərini göstərmək olar.

Əgər binar atributun qiymətləri eyni vaciblik dərəcəsinə malik deyilsə onda bu atribut assimetrik adlanır. Məsələn, HIV virusunun test nəticəsinin kodlaşdırılması zamanı daha vacib olan və nadir olan pozitiv qiymətini 1, neqativ qiymətini isə 0 ilə ifadə edirik.

1.1.3 Ordinal atributlar

Ordinal atribut məntiqli ardıcıllığa malik olan və ya müəyyən reytinqi ifadə edən qiymətlərə malik olan atributlardır, lakin bu qiymətlər arasındakı fərqlərin ölçüsü haqqında məlumat verilmir. **Məsələn**, fastfud restoranlarında içkinin ölçüsünü bildirən *içki_ ölçüsü* atributu kiçik, orta və böyük qiymətlərini ala bilər. Bu qiymətlər içki ölçüsünün artmasını bildirən konkret ardıcıllığa malikdir, lakin böyük ölçünün ortadan nə qədər böyük olması haqqında bir söz deyə bilmirik. Ordinal atributlara digər nümunə olaraq *imtahan_ qiyməti* atributunu göstərmək olar, bu atributun qiymətləri A, B, C, D və s. olar.

Ordinal atributlar adətən obyektiv ölçülə bilməyən keyfiyyətlərin subyektiv qiymətləndirilməsi üçün istifadə olunur: məsələn, reytinqlərin hazırlanması üçün aparılan sorğularda. Sorğuda müştərilərin xidmətdən nə qədər məmnun qalmasını

qiymətləndirmək üçün aşağıdakı ordinal kateqoriyalardan istifadə olunur: 0: heç məmnun qalmadım, 1: az məmnun qaldım; 2: neytral; 3: məmnun qaldım; 4: çox məmnun qaldım.

Ordinal atributlar həmçinin ədədi kəmiyyətlərin diskretləşməsindən, qiymətlər intervalının sonlu sayda ardıcıl kateqoriyalara bölünməsindən də əldə oluna bilər. Ordinal atributun mərkəzi yayılması modanın və medianın (ardıcıllığın mərkəzindəki qiymət) hesablanması ilə təyin oluna bilər, ədədi ortanın təyin olunması isə mümkün deyil.

Qeyd edək ki, nominal, binar və ordinal atributlar keyfiyyət əsaslı atributlardır. Belə ki, bu atributlar obyektin əlamətini aktual ölçüsü və dəyəri haqqında məlumat vermədən təsvir edir. Bu cür keyfiyyət əsaslı atributların qiymətləri adətən kateqoriyaları təsvir edən sözlərlə və ya bu kateqoriyaları təsvir edən kompüter kodları ilə ifadə olunur.

1.1.4 Ədədi atributlar

Ədədi atributlar kəmiyyət əsaslı, tam və ya həqiqi qiymətlərlə ifadə olunan ölçülə bilən atributlardır. Ədədi atributlar interval miqyaslı və nisbət miqyaslı ola bilərlər.

İnterval miqyaslı atributlar bərabər ölçülü vahidlərin miqyasında ölçülür. Bu tip atributların qiymətləri ardıcıl düzülür, mənfi, 0 və ya müsbət ola bilir. Belə ki, qiymətlərin reytinqini verməklə yanaşı bu atributlar qiymətlər arasındakı fərqi müqayisə etməyə və hesablamağa imkan verir. **Məsələn**, temperatur atributu interval miqyaslıdır. Tutaq ki, müxtəlif günlər üçün havanın temperaturunu ölçmüşük, burada hər bir gün bir obyektdir. Qiymətləri düzərək biz temperatura əsasən obyektlərin reytinqini əldə edirik, həmçinin qiymətlər arasındakı fərqi də qiymətləndiririk. Belə ki, $20°C$ temperatur $15°C$ temperaturdan 5 dərəcə yüksəkdir.

Müxtəlif temperatur qiymətləri arasındakı fərqi hesablaya bilməyimizə baxmayaraq bir temperatur qiymətinin digərinin misli və ya nisbəti olduğunu deyə bilmərik. Temperatur qiyməti üçün mütləq sıfır olmadığına görə $5°C$ - nin $10°C$ - dən 2 dəfə sərin olduğunu demək olmaz. Belə ki, bu atribut qiymətləri üçün nisbətin mənası yoxdur. Uyğun olaraq, təqvim günləri üçün də mütləq sıfır qiyməti yoxdur (0-cı il zamanın başlaması demək deyil). Mütləq sıfır qiyməti olan atributları nisbət miqyaslı adlandırırıq.

Nisbət miqyaslı atributlar mütləq sıfır qiyməti olan ədədi atributlardır. Bu atributlar nisbət miqyaslı olduğundan bir qiymətin digərinin misli (və ya nisbəti) olması haqqında danışa bilərik. Əlavə olaraq qiymətlər ardıcıl düzüldüyündən onların arasındakı fərqi, modanı, medianı və orta qiyməti hesablaya bilərik. Nümunə olaraq *işçilər* obyektinin *təcrübə* atributu üçün illərin sayı və ya *sənəd* obyektləri üçün *sözlərin sayı* atributunu göstərə bilərik.

1.2 Mərkəzi yayılma ölçüləri

Bu bölmədə verilənlərin mərkəzi yayılmasının müxtəlif üsulları təsvir olunur. Tutaq ki, obyektlər çoxluğu üçün *maaş* atributu X verilmişdir. N sayda obyekt üçün X atributunun müşahidə olunan qiymətləri $x_1, x_2, ..., x_N$ şəklində ifadə olunur. *Maaş* üçün müşahidə olunan qiymətləri fəzada vizuallaşdırsaq nöqtələrin ən böyük hissəsi hansı hissəyə düşəcək? Bununla biz verilənlərin mərkəzi yayılması haqqında məlumat ala bilərik. Mərkəzi yayılmanın ölçüləri orta qiymət, moda və median ilə ölçülür.

1.2.1 Orta qiymət

Verilənlərin mərkəzi yayılmasının ölçülməsi üçün ən çox istifadə olunan ölçü kəmiyyəti verilənlərin orta qiymətidir. Yuxarıda qeyd olunan atributun orta qiyməti müşahidə olunan qiymətlərinin ədədi ortası kimi hesablanır:

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \frac{x_1 + x_2 + ... + x_N}{N} \tag{1.1}$$

Məsələn, tutaq ki, maaş atributu üçün müşahidə olunan qiymətlər artan sıra ilə düzülmüşdür: 30, 36, 47, 50, 52, 52, 56, 60, 63, 70, 70 və 110 (min manat), (1.1) tənliyindən istifadə edərək alarıq:

$$\bar{x} = \frac{30 + 36 + 47 + 50 + 52 + 52 + 56 + 60 + 63 + 70 + 70 + 110}{12} = \frac{696}{12} = 58$$

Beləliklə, verilənlərə əsasən illik orta maaşın orta qiyməti 58 min manat olar.

Bəzi hallarda x_i qiymətləri çəkilərlə $w_i, i = 1, ..., N$ ifadə olunur. Bu çəkilər aid olduğu qiymətlərin vaciblik, önəmlilik səviyyəsini və ya rast gəlinmə tezliyini ifadə edir. Bu halda

$$\bar{x} = \frac{\sum_{i=1}^{N} w_i x_i}{\sum_{i=1}^{N} w_i} = \frac{w_1 x_1 + w_2 x_2 + ... + w_N x_N}{w_1 + w_2 + ... + w_N} \tag{1.2}$$

düsturu ilə çəkili orta qiymət hesablanır.

Orta qiymətin verilənlər çoxluğunu təsvir edən əsas kəmiyyət olmasına baxmayaraq verilənlərin mərkəzinin ölçülməsi üçün həmişə ən yaxşı yol deyil. Bu kəmiyyətin ən böyük problemi ekstrim qiymətlərə olan həssaslığıdır. Az sayda ekstrim qiymətlər belə orta qiyməti korlaya bilər. Məsələn, şirkətdə maaşların orta qiyməti az sayda menecerlərin yüksək maaşları da əlavə edildikdə çox yüksək ola bilər. Oxşar olaraq sinfin qiymətlərinin orta qiyməti az sayda aşağı qiymət alan şagirdlərin nəticələri ilə çox enə bilər.

1.2.2 Median

Assimetrik verilənlər üçün mərkəzi yayılmanın hesablanmasının ən yaxşı yolu medianın, ardıcıl düzülmüş çoxluğun mərkəzi qiymətinin təyin olunmasıdır. Bu

qiymət verilənlərin aşağı qiymətli hissəsini yuxarı qiymətli hissəsindən ayırır. Adətən median ədədi atributlar üçün hesablanır, lakin ordinal verilənlər üçün də hesablana bilər. Tutaq ki, X atributunun N sayda qiyməti artan sıra ilə düzülmüşdür. Əgər N təkdirsə, median ortadakı qiymətdir, N cüt olarsa bu verilənlərin medianı yeganə deyil, ortadakı iki qiymətdən biri və ya bu qiymətlər arasındakı istənilən qiymət ola bilər. Əgər X ədədi atribut olarsa, onda median ortadakı iki qiymətin ədədi ortası kimi təyin olunur.

Yuxarıdakı nümunədə verilmiş qiymətləri nəzərdən keçirək. Verilənlər artan sıra ilə düzülmüşdür, sayı cüt olduğuna görə ortadakı iki qiymətə baxaq: bunlar 6-cı və 7-ci elementlər olan 52 və 56 ədədləridir. Bu halda median bu qiymətlərin ədədi ortası $\frac{52+56}{2} = 54$ olar. Deməli, *maaş* atributunun medianı 54 min manat olar.

1.2.3 Moda

Mərkəzi yayılmanın digər ölçüsü modadır. Moda verilənlər çoxluğunda ən çox rast gəlinən qiymətdir. Buna görə də həm kəmiyyət, həm də keyfiyyət bildirən atributlara tətbiq oluna bilər. Ən böyük tezlik bir deyil bir neçə qiymətə uyğun gələ bilər, bu halda verilənlər çoxluğunda bir deyil, bir neçə moda olur. Başqa halda isə, yəni bütün qiymətlər yalnız bir dəfə müşahidə olunursa, bu çoxluğun modası yoxdur.

Əvvəlki bölmədə verilmiş nümunədə verilənlər çoxluğu bimodal olub, iki modaya malikdir: 52000 və 70000 manat.

Bu paraqrafda qeyd olunan ölçülərdən əlavə verilənlərin dispersiyasını təyin etmək üçün orta kvadratik yayılma, standart yayınma, və s. kimi meyarları da hesablamaq mümkündür.

1.3 Mövcud olmayan verilənlərin doldurulması

Real dünyada verilənləri natamam, küylü və sistematik olmayan şəkildə əldə edirik. Verilənlərin təmizlənməsi mövcud olmayan verilənlərin doldurulması, küylərin təmizlənməsi və verilənlərin sistematik şəklə gətirilməsi proseslərindən ibarətdir.

Tutaq ki, verilmiş şirkətdə müştərilərin verilənləri təhlil olunur. Bu zaman bir sıra obyektlər üçün bəzi atributların, məsələn maaş atributunun bəzi qiymətləri bazada mövcud deyil. Bu atributun olmayan qiymətlərinin necə təxmin etmək olar? Bunun üçün müxtəlif üsullar mövcuddur:

- **Obyektin bazadan silinməsi** üsulu adətən klassifikasiya məsələsində obyektin sinfinin olmaması halında həyata keçirilir. Bu üsulu o qədər də effektiv deyil, çünki obyekt silindikdə mövcud olan atribut qiymətləri də silinir, halbuki onlar məsələnin həllində mühüm rol oynayır;

- **Mərkəzi yayılma meyarları** vasitəsilə mövcud olmayan atribut qiymətlərinin bərpası çox istifadə olunan üsuldur. Normal paylanan verilənlər üçün (simmetrik verilənlər) orta qiymət, simmetrik olmayan verilənlər üçün isə mediandan istifadə oluna bilər;

- **Ən çox ehtimala malik qiymətlə bərpa.** Bu üsul reqressiya, Bayes şəbəkəsi və ya qərar qəbuletmə ağacının tətbiqi ilə yerinə yetirilə bilər. Belə ki, məsələn, maaş atributunun mövcud olmayan qiymətlərini qərar qəbuletmə ağacını tərtib edib proqnozlaşdırmaq olar. Digər bir üsul isə mövcud olmayan qiymətlərin mövcud qiymətlərlə interpolyasiyası nəticəsində proqnozlaşdırılmasıdır.

Bu fəsildə maşın öyrənmə üsullarının tətbiq olunmasından öncə verilənlərin ilkin emalı üçün mövcud meyarlar və üsullar təsvir olunmuşdur. Verilənlərin ilkin emalının çox böyük sahə olmasını və süni intellekt məsələlərində mühüm əhəmiyyət kəsb etdiyini nəzərə alaraq qeyd edək ki, burada təsvir olunanlar ilkin emal üsullarının çox kiçik hissəsidir. Bunlardan əlavə küylərin təmizlənməsi, verilənlərin normallaşdırılması, transformasiyası, əsas komponentlərin təhlili, informativ atributların seçilməsi və s. kimi mərhələlər üçün çoxlu sayda yanaşmalar və alqoritmlər tədqiq və tətbiq olunmuşdur [10].

Çalışmalar

1. 12, 15, 18, 20, 22, 25, 25, 30, 35, 40 verilənləri üçün orta qiymət, median və modanı hesablayın.

2. 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 90 verilənləri üçün orta qiymət, median və modanı hesablayın. 90 ekstrim qiymətinin mərkəzi yayılma meyarlarının qiymətlərinə təsirini təhlil edin.

3. 8, 12, 15, 20, 25, ? nöqtələri verilmişdir. Əgər verilənlərin orta qiyməti 18 olarsa, mövcud olmayan qiyməti təyin edin.

4. Aşağıdakı şəkildə tezliklərlə paylanmış verilənlər verilmişdir:

Sinif	Tezlik
10-20	5
20-30	12
30-40	8
40-50	15

   Verilənlər çoxluğunun orta qiymətini hesablayın.

5. Tutaq ki, nəticəsi uyğun olaraq 5 bal, 10 bal və 15 bal olan 3 sual var. Tələbənin nəticəsi 7, ?, 12 şəklindədir. Çəkili orta qiymət 15 bal olarsa, olmayan qiyməti təyin edin.

6. Sorğuda iştirakçılara məmnuniyyətlərini 1-dən 5-ə kimi qiymətləndirmək tapşırılmışdır. Toplanan verilənlər 4, 5, 3, 2, ?, 4 şəklindədir. Mövcud olmayan verilənin modanı maksimallaşdıran qiymətini təyin edin.

Fəsil 2

Maşın öyrənmə üsulları

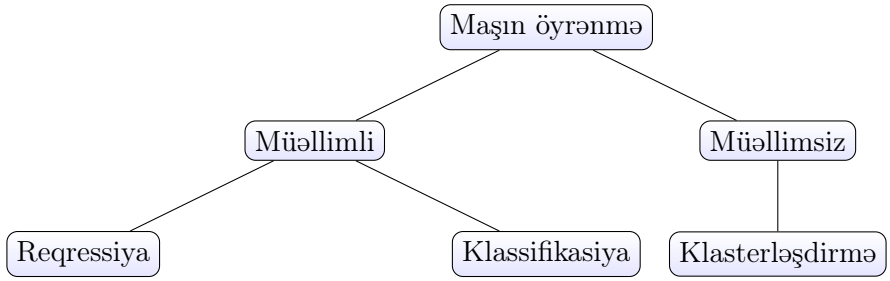
Kompüter dünyasında baş verən texnoloji irəliləyişlər böyük həcmdə verilənləri saxlamaq, həmçinin fiziki uzaq məsafələrdən onları emal etmək imkanı verir. Halhazırda istifadə olunan əksər cihazlar rəqəmsaldır və etibarlı informasiya mənbələridir. Nümunə olaraq bütün ölkədə yüzlərlə mağazası, milyonlarla müştərisi olan supermarketlər şəbəkəsini nəzərdən keçirək. Satış nöqtələrindəki terminallar hər bir əməliyyatın təfərrüatlarını (tarix, müştəri identifikasiya kodu, alınan mallar, onların qiyməti, xərclənən ümumi pulun miqdarı, və s.) qeyd edirlər. Beləliklə, gün ərzində qiqabaytlarla informasiya əldə olunur. Supermarketin bu suala cavab almaq istəyir. *Verilmiş məhsulu alma ehtimalı yüksək olan müştərilər kimlərdir?* Bu məsələ üçün qurulan alqoritm də aşkar deyil: zamana və məkana görə dəyişir. Saxlanılan verilənlər yalnız təhlil edilib faydalı informasiyaya (məsələn, proqnoz) çevrilə bildikdə yararlı olur [11].

Yeni hazırlanmış dondurmanı kim alacaq? Verilmiş müəllifin kitabını kim oxuyacaq? Yeni filmə kim baxacaq? Linkə kim keçid edəcək? Əgər bu sualların cavabını bilsəydik, onda heç bir informasiya təhlilinə ehtiyac olmazdı. Lakin, bu sualların cavabını bilmədiyimiz üçün yalnız verilənləri toplayıb emal etməklə bu və ya bənzər suallara cavab tapmağa ümid edirik.

Verilənlərin emalı müşahidə edilən məlumatları izah edən bir prosesin olması fərziyyəsi əsasında yerinə yetirilir. Məlumatların yaranmasının təfərrüatları (məsələn, istehlakçı davranışı) məlum olmasa da, verilənlərin yaranması tam təsadüfi deyil. İnsanlar supermarketə gedib təsadüfən əşyalar almırlar, pivə alanda çips alırlar, yayda dondurma, qışda isti içkilər alırlar, və s.

Prosesin tam təyini mümkün olmasa da, verilənlərin yaxşı və faydalı approksimasiyasının qurulması mümkündür. Bu approksimasiya prosesi tam təfərrüatı ilə izah etməsə də bəzi hissələri aydınlaşdıra bilər. Bu hissələr maşın öyrənməsinin nüvəsidir. Aşkarlanan örnəklər vasitəsilə proses izah oluna bilər və ya proqnozlaşdırmada istifadə oluna bilər.

Maşın öyrənməsi yalnız verilənlər bazası problemi deyil, həm də süni intellektin bir hissəsidir. Sistemin intellektual olması üçün dəyişən mühitdə öyrənmə qabiliyyətinə malik olmalıdır. Əgər sistem öyrənə bilirsə və baş verən dəyişikliklərə



Şəkil 2.1: Maşın öyrənmə üsullarının təsnifatı

adaptasiya ola bilirsə, onda sistemin müəllifi bütün mümkün halları öncədən görmək və nəzərə almaq məcburiyyətində deyil.

Maşın öyrənmə nümunə verilənlər və ya keçmiş təcrübə əsasında müvəffəqiyyət göstəricisinin optimallaşdırılması üçün kompüterlərin proqramlaşdırılmasıdır. Bir neçə parametrlərlə təyin olunan model verilir və öyrənmə nümunə verilənlər və ya keçmiş təcrübədən istifadə edərək bu parametrlərin optimallaşdırılması üçün proqram təminatının icrasıdır.

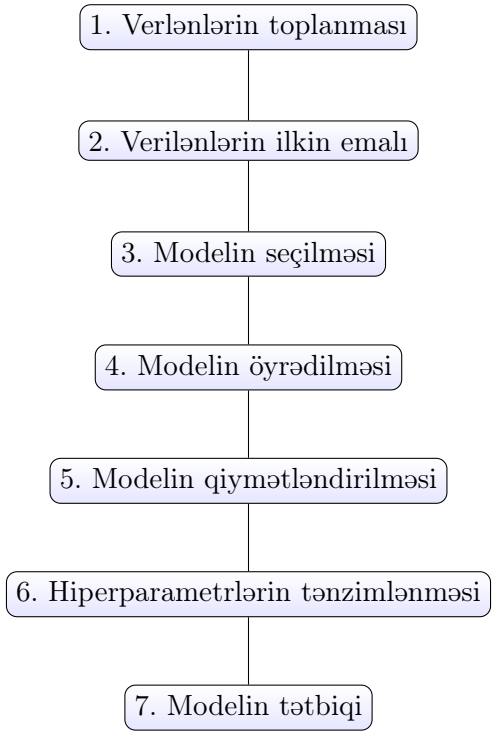
Maşın öyrənmə riyazi modellərin hazırlanması üçün statistika nəzəriyyəsindən istifadə edir, çünki burada əsas məqsəd nümunələrdən nəticənin əldə olunmasıdır. Maşın öyrənmədə kompüter elmlərinin rolu ikiqatdır: ilk növbədə öyrənmədə optimallaşdırma məsələsinin həlli və əldə olunan verilənlərin saxlanması və emalı üçün effektiv alqoritmlərə ehtiyac var; digər tərəfdən öyrənmiş modelin və nəticənin təsviri də effektiv olmalıdır. Müəyyən tətbiqlərdə öyrənmə və ya nəticə alqoritminin effektivliyi, yəni fəza və zaman kompleksliyi proqnozlaşdırma dəqiqliyi qədər vacib ola bilər.

Maşın öyrənmə üsullarını ümumi şəkildə müəllimli və müəllimsiz öyrənmə üsullarına ayırmaq olar (şəkil 2.1). Bu fəsildə və sonrakı fəsillərdə müəllimli maşın öyrənmə üsullarının qiymətləndirilməsi metrikaları, müxtəlif müəllimli maşın öyrənmə üsullarının tətbiqi təsvir olunmuşdur. Sonuncu fəsildə müəllimsiz öyrənmə üsullarına nümunə olaraq klasterləşdirmə üsulları haqqında məlumat verilmişdir.

2.1 Müəllimli öyrənmə

Maşın öyrənmə alqoritmləri verilənlər çoxluğunun hər bir nümunəsinə əlamətlər çoxluğu kimi baxır. Bu əlamətlər binar, keyfiyyət və ya kəmiyyət göstəriciləri şəklində ola bilər. Əgər verilənlər çoxluğunun nümunələri işarələnmişdirsə (çıxış qiymətləri məlumdursa), onda bu tip öyrənmə müəllimli öyrənmə üsulu adlanır. Müəllimli öyrənmə işarələnmiş verilənlərdə modelin öyrənməsinə və işarələnməmiş verilənlərdə test edilməsinə əsaslanır.

Müəllimli öyrənmə alqoritmi verilənlər çoxluğunun toplanması ilə başlayır, daha sonra verilənlər öyrədici və test bazalarına ayrılır və verilənlər ilkin emal olunur. Çıxarılmış əlamətlər alqoritmə verilir və model çıxış verilənləri ilə əlaqədar əlamətlərin öyrədilməsini həyata keçirir. Növbəti addımda model test bazasının



Şəkil 2.2: Müəllimli maşın öyrənmə modellərinin qurulma mərhələləri

elementlərinin çıxış verilənlərini proqnozlaşdırır və alınan nəticələr test bazasındakı çıxış verilənləri ilə müqayisə olunur. Nəticədə üsulun dəqiqliyi qiymətləndirilir (şəkil 2.2). Maşın öyrənmə modellərinin dəqiqliyində hiperparametrlərin düzgün seçilməsi mühüm rol oynayır. Buna görə də hiperparametrlərin tənzimlənməsi nəticəsində modelin dəqiqliyini artırmaq mümkündür. Maşın öyrənmə üsullarının qurulması zamanı arzuolunan nəticə alınana qədər 4 - 6-cı mərhələlər dövri şəkildə icra oluna bilər.

2.1.1 Klassifikasiya

Klassifikasiya və ya təsnifat həm insan, həm də maşın intellektinin əsasını təşkil edir. Hansı hərfin, sözün və ya təsvirin təqdim edildiyinə qərar vermək, səsləri, məktubları çeşidləmək, ev tapşırıqlarını qiymətləndirmək - bunların hamısı verilmiş girişə kateqoriyanın təyin edilməsinə, başqa sözlə verilənlərin klasifikasiyasına uyğun nümunələrdir.

Klassifikasiyanın əsas məqsədi vahid nümunənin təhlili, bir sıra vacib əlamətlərin təyin olunması və nəticədə bu nümunənin bir neçə diskret siniflərdən birinə aid edilməsidir. Müəllimli öyrətmə üsulu ilə klassifikasiya alqoritmi hər biri düzgün sinfə uyğunlaşdırılmış giriş verilənləri əsasında yeni müşahidənin düzgün sinfə aid edilməsinin təmin olunmasından ibarətdir.

Klassifikasiya üsullarını iki hissəyə ayırmaq olar: binar və çox sinifli klassi-

fikasiya. Binar klassifikasiya çıxış verilənləri binar olduqda və ya verilənlər yalnız iki sinfə aid olduqda tətbiq olunur. Məsələn, qeyri-müəyyənliyin təyin olunması zamanı model cümlənin qeyri-müəyyən olub-olmamasını proqnozlaşdırır, nəticə etibarilə yalnız iki çıxış/sinif mövcuddur. Çox sinifli klassifikasiya verilənlərin 2-dən artıq siniflərə bölünməsinə əsaslanır. Məsələn, psixi pozğunluqların proqnozlaşdırılmasının çoxlu sayda çıxışı ola bilər - depressiya, təşviş, şizofreniya, bipolyar pozğunluq və ya post-travmatik psixi pozğunluq. Beləliklə, nümunə bu 5 sinifdən birinə aid ola bilər.

Klassifikasiya üsullarının qiymətləndirilməsi

Klassifikasiya üsullarının qiymətləndirilməsində düzgün və yanlış proqnozlaşdırılan nümunələrin sayı və onların nisbəti istifadə olunur. Bu məqsədlə aşağıdakı metrikalardan istifadə edilir:

1. Binar klassifikasiya üçün qarışıqlıq matrisi (cədvəl 2.1)

		Proqnozlaşdırılan	
		Pozitiv (1)	Neqativ (0)
Real	Doğru (1)	DP	YN
	Yanlış (0)	YP	DN

Cədvəl 2.1: Binar klassifikasiya üçün qarışıqlıq matrisi

   Bu matrisdə aktual qiymətlər *Doğru (1)* və *Yanlış (0)* olaraq göstərilmişdir və *Pozitiv (1)* və ya *Neqativ (0)* olaraq proqnozlaşdırılır. Klassifikasiya modellərinin dəqiqliyinin qiymətlənidirlməsi qarışıqlıq matrisində verilən DP, YP, YN və DN qiymətləri əsasında yerinə yetirilir.
   DP (Doğru Pozitiv) - real çıxış qiyməti pozitiv olan və pozitiv olaraq proqnozlaşdırılan nümunələrin sayıdır;
   YP (Yanlış Pozitiv) - real çıxış qiyməti neqativ olan, lakin pozitiv olaraq proqnozlaşdırılan nümunələrin sayıdır;
   YP (Yanlış Neqativ) - real çıxış qiyməti neqativ olan, lakin pozitiv olaraq proqnozlaşdırılan nümunələrin sayıdır;
   DN (Doğru Neqativ) - real çıxış qiyməti neqativ olan və neqativ olaraq proqnozlaşdırılan nümunələrin sayıdır;

2. Dəqiqlik - doğru proqnozlaşdırılan nümunələrin sayının bütün nümunələrin sayına nisbəti ilə təyin olunur:

$$A = \frac{DP + DN}{DP + YP + YN + DN}; \tag{2.1}$$

3. DP nisbəti - doğru pozitiv nisbəti (həssaslıq) düzgün proqnozlaşdırılan pozitiv nümunələrin sayının bütün pozitiv nümunələrin sayına nisbətidir:

$$S = \frac{DP}{DP + YN}; \tag{2.2}$$

4. YP nisbəti - yanlış pozitiv nisbəti yanlış proqnozlaşdırılan neqativ nümunələrin sayının bütün neqativ nümunələrin sayına nisbətidir:

$$FPR = \frac{YP}{DN + YP}; \tag{2.3}$$

5. Səhihlik (Precision) - düzgün proqnozlaşdırılan pozitiv nümunələrin sayının bütün pozitiv proqnozlaşdırılan nümunələrin sayına nisbətidir:

$$PREC = \frac{DP}{DP + YP}; \tag{2.4}$$

6. Doğru neqativ nisbəti (Spesifiklik) - düzgün proqnozlaşdırılan neqativ nümunələrin sayının bütün neqativ nümunələrin sayına nisbətidir:

$$TNR = \frac{DN}{DN + YP}; \tag{2.5}$$

7. F-ölçü və ya F-qiymət - test mərhələsinin dəqiqliyini qiymətləndirir, aşağıdakı düsturla hesablanır:

$$F - score = \frac{2 \cdot PREC \cdot S}{PREC + S}; \tag{2.6}$$

8. Metyu (Matthew) Korrelyasiya Əmsalı - real qiymətlərlə proqnozlaşdırılan qiymətlər arasındakı korrelyasiyanı hesablayır:

$$MCC = \frac{DP}{\sqrt{(DP + YP)(DP + YN)(DN + YP)(DN + YN)}}; \tag{2.7}$$

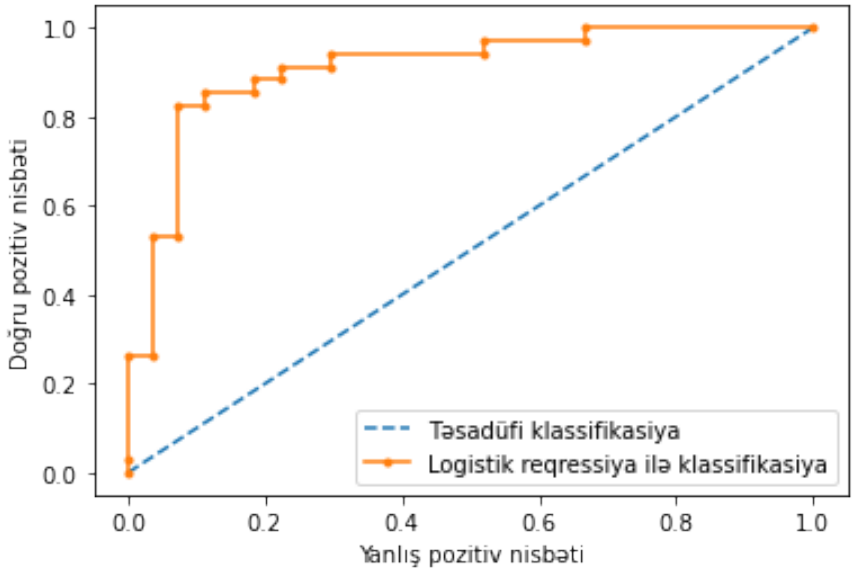
9. ROC[1] əyrisi - doğru pozitiv nisbət ilə yanlış pozitiv nisbət arasındakı qarşılıqlı əlaqəni vizuallaşdırır. X oxunda yanlış pozitiv nisbət - YP/AN, y oxunda isə doğru pozitiv nisbət - DP/AP göstərilən qrafikdir, burada $AN = YP+DN$ və $AP = DP+YN$ (şəkil 2.3). ROC qrafikinin öyrənlməsi ilə əsas iki müddəanı isbat edirik: y, x - ə nəzərən monotondur, pozitiv və neqativ nümunələrin balanslaşdırılmamış olduğu bazalarda y və ya x həssas deyil;

10. PRC[2] sahəsi - X oxunda doğru pozitiv nisbət - DP/AP, y oxunda isə pozitiv sinfin dəqiqlik nisbəti - DP/PP göstərilən qrafikdir, burada $AP = DP+YN$ və $PP = DP + YP$ (şəkil 2.4). Eyni qrafiki neqativ sinfi üçün də qurmaq mümkündür.

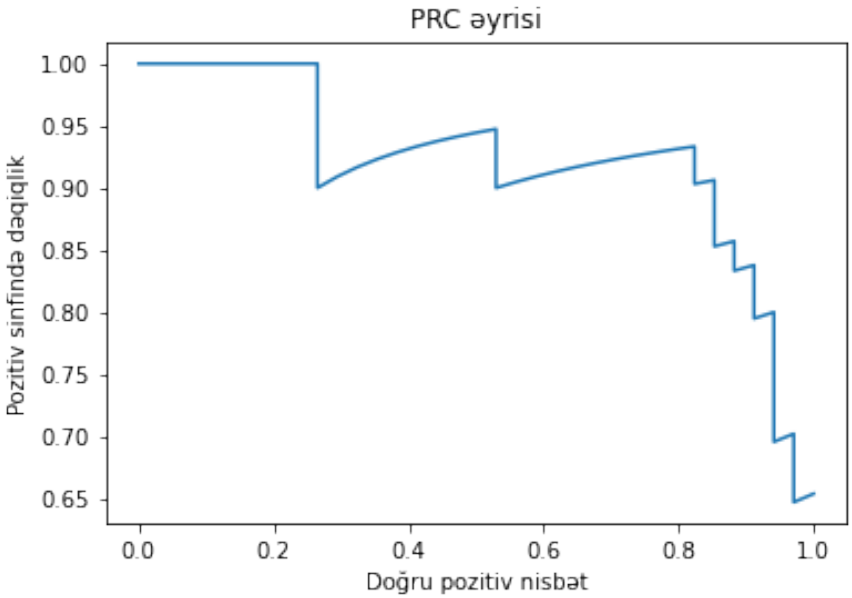
Yuxarıda təsvir olunan dəqiqlik, səhihlik, F1-qiyməti kimi metrikalar balanslaşdırılmış verilənlər çoxluğunda klassifikasiya modellərinin performansının qiymətləndirilməsi üçün əlverişlidir. Balanslaşdırılmamış verilənlər bazasında isə ROC əyrisi modellərin performansı haqqında daha çox məlumat verir.

1. ROC əyrisi və ya qəbuledici operator xüsusiyyəti əyrisi - binar klassifikatorun diaqnostika qabiliyyətini göstərir

2. PRC əyrisi və ya səhihlik-spesifiklik əyrisi - müxtəlif hədlər üçün səhihlik və spesifiklik arasındakı uzlaşmanı göstərir



Şəkil 2.3: *heart.csv* bazasının logistik reqressiya üsulu ilə proqnozlaşdırılmasının ROC qrafiki



Şəkil 2.4: *heart.csv* bazasının pozitiv sinfi üçün logistik reqressiya üsulu ilə proqnozlaşdırılmasının PRC qrafiki

2.1.2 Reqressiya

Reqressiya dəyişənlər arasında korrelyasiyanın aşkarlanmasına imkan verən və bu dəyişənlərin əsasında kəsilməz qiymətlərin proqnozlaşdırılmasını həyata keçirən maşın öyrənmə üsuludur. Əgər çıxış veriləni kəsilməz qiymət alırsa onda məsələ reqressiya məsələsi hesab olunur. Məsələn, insanın çəkisinin, yaşının, gəlirinin, havanın temperaturunun və ya evin qiymətinin proqnozlaşdırılması. Reqressiya məsələsində giriş verilənləri X çıxış verilənlərinə Y uyğunlaşdırılır. Klassifikasiya giriş verilənlərinin diskret çıxışlarının proqnozlaşdırılmasından ibarətdir. Reqressiya isə öz növbəsində kəsilməz qiymətlərin proqnozlaşdırılmasını həyata keçirir.

Reqressiya iki əsas kateqoriyaya ayrılır: sadə xətti və çoxdəyişənli. Sadə xətti reqressiya iki dəyişən (X və Y) arasında asılılığı təyin edən düz xəttin təyin olunmasından ibarətdir. Çox dəyişənli reqressiya çoxsaylı dəyişənlər arasında xətti və qeyri-xətti asılılıqları müəyyən edir.

Reqressiya üsullarının qiymətləndirilməsi

Reqressiya analizinin tətbiqi ilə alınan nəticələrin effektivliyini yoxlamaq üçün aşağıdakı metrikalardan istifadə edilir:

1. Xətanın orta mütləq qiyməti (MAE – mean absolute error) – test və ya öyrədici bazadakı x verilənlərinə əsasən real y qiymətləri və proqnozlaşdırılan qiymətlər $\hat{y}$ arasındakı fərqin mütləq qiymətinin ədədi ortası kimi hesablanır:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|; \tag{2.8}$$

2. Xətanın kvadratının orta qiyməti (MSE – mean squared error) – test və ya öyrədici bazadakı x verilənlərinə əsasən real y qiymətləri və proqnozlaşdırılan qiymətlər $\hat{y}$ arasındakı fərqin kvadratlarının ədədi ortası kimi hesablanır:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2; \tag{2.9}$$

3. Xətanın kvadratının orta qiymətinin kvadrat kökü (RMSE – Root mean squared error) - xətanın kvadratının orta qiymətinin kvadrat kökü kimi hesablanır:

$$RMSE = \sqrt{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}; \tag{2.10}$$

4. Kvadrat *R (R2)* – modelin mütləq mənada xətasını deyil, bütovlükdə modelin effektivliyini qiymətləndirir. R - kvadrat, asılı dəyişənin dəyişkənliyinin reqressiya modelində müstəqil dəyişən(lər) tərəfindən nə qədər izah edildiyini göstərən uyğunluğun statistik ölçüsüdür:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}, \tag{2.11}$$

burada $\overline{y}-$ real verilənlərin orta qiymətidir, $\overline{y} = \frac{\sum_{i=1}^{n} y_i}{n}$.

2.2 Sadə maşın öyrənmə üsulları

Bu bölmədə sadə maşın öyrənmə üsulları, onların klassifikasiya və ya reqressiya məsələlərinə tətbiqi, sadə nümunələr əsasında alqoritmlərin yerinə yetirilməsi, onların verilənlər bazaları mənbələrindən götürülmüş real verilənlər çoxluğunda tətbiqi və efektivliyinin qiymətləndirilməsi təsvir olunur. İlk öncə ən sadə maşın öyrənmə üsullarından olan və verilənləri atribut verilənlərinin tezliklərinə əsasən klassifikasiya edən Naive-Bayes üsulu, daha sonra isə "tənbəl" öyrənmə üsulu hesab olunan k ən yaxın qonşu üsulu izah olunur.

2.2.1 Naïve-Bayes klassifikatoru

Naïve-Bayes klassifikatoru Bayes düsturuna və əlamətlərin (atribut verilənlərin) asılı olmamasına əsaslanan klasifikasiya üsuludur [12]. Bu üsulun tətbiqi zamanı aşağıdakı şəkildə Bayes düsturundan istifadə olunur:

$$P(c = c_j|f_1, f_2, ..., f_n) = \frac{P(f_1, f_2, ..., f_n|c = c_j)P(c = c_j)}{P(f_1, f_2, ..., f_n)}, \tag{2.12}$$

burada $j = 1, 2, ..., M,\ f_1, f_2, ..., f_n$ – verilmiş nümunənin əlamətləri və ya atribut verilənləri, n - əlamətlərin sayı, $M-$ mövcud siniflərin sayı, c_j isə j -ci sinfi göstərir. Bu düstur vasitəsilə verilmiş əlamətlərə malik olan giriş nümunəsinin hər bir sinifə daxil olmasının ehtimalı hesablanır və ən yüksək ehtimala malik sinfin nömrəsi və ya adı çıxış olaraq qəbul olunur. Qeyd etdiyimiz kimi əlamətlər bir-birindən asılı olmadığına görə 2.12 düsturunun surətindəki birinci vuruğu aşağıdakı şəkildə yaza bilərik:

$$P(f_1, f_2, ..., f_n|c = c_j) = P(f_1|c = c_j)P(f_2|c = c_j)...P(f_n|c = c_j), \tag{2.13}$$

2.12 düsturunun məxrəcindəki ehtimal isə bütün verilənlər üçün sabit olduğundan qərar qəbul etməyə təsir göstərmir, buna görə də adətən düstur aşağıdakı şəkildə istifadə olunur:

$$P(f_1, f_2, ..., f_n|c = c_j) \propto P(f_1|c = c_j)P(f_2|c = c_j)...P(f_n|c = c_j). \tag{2.14}$$

Başqa sözlə desək Naïve-Bayes üsulu ilə klassifikasiya zamanı verilən girişə uyğun sinfin nömrəsi və ya adı

$$c_{NB} = \arg \max_{j=\overline{1,M}} \prod_{i=1}^{n} P(f_i|c = c_j)P(c = c_j) \tag{2.15}$$

şəklində təyin olunur. Dağılmanın qarşısının alınması və təsnifatın sürətinin artırılması üçün bu düsturda loqarifmdən istifadə olunur:

$$c_{NB} = \arg \max_{j=\overline{1,M}} (log(P(c = c_j)) + \sum_{i=1}^{n} log(P(c = c_j|f_i))). \tag{2.16}$$

Beləliklə, Naïve-Bayes üsulunda proqnozlaşdırılan sinif giriş əlamətlərinin xətti funksiyası kimi təyin olunur. $P(c = c_j)$ və $P(f_i|c = c_j)$ ehtimalları isə öyrədici verilənlər bazasından istifadə etməklə tezlik kimi hesablanır.

Verilənlərin təsnifatı. Nümunə

Naive-Bayes üsulunun işləmə mexanizmini daha aydın göstərmək üçün aşağıdakı verilənlər cədvəlini nəzərdən keçirək (Cədvəl 2.2).

İD	EV	MƏQSƏD	MƏBLƏĞ	STATUS
1	şəxsi	şəxsi	yüksək	təsdiq
2	şəxsi	təhsil	aşağı	imtina
3	ipoteka	tibbi	orta	təsdiq
4	kirayə	təhsil	yüksək	imtina
5	şəxsi	investisiya	aşağı	təsdiq
6	kirayə	investisiya	yüksək	imtina

Cədvəl 2.2: Credit_risk_dataset verilənlər bazasından bir neçə nümunə.

Təsvir olunan verilənlər kredit götürmək üçün müraciət edən şəxslərin xarakteristikaları və müraciətin statusu haqqında məlumatı özündə əks etdirir, burada *İD* - müraciət edənin identifikasiya nömrəsini göstərir və klassifikasiyanın nəticəsinə təsir göstərmir. *EV* - müraciət edənin hazırda yaşadığı evin statusu haqqında məlumatı, *MƏQSƏD* - kredit üçün müraciətin səbəbini, *MƏBLƏĞ* - kreditin məbləğinin kateqoriyasını göstərən atribut verilənləridir. Sadalanan əlamətlər keyfiyyət xarakterli əlamətlərdir, uyğun olaraq aşağıdakı qiymətlərdən birini alır:

- Ev - kirayə, ipoteka, şəxsi;
- Məqsəd - şəxsi, təhsil, tibbi, investisiya;
- Məbləğ - aşağı, orta, yuxarı

Burada proqnozlaşdırılan (asılı) parametr - *STATUS* isə müraciət edənlərin göstəricilərinə əsasən *təsdiq* və ya *imtina* qiymətlərindən birini alır. Tutaq ki, aşağıdakı parametrlərə malik yeni nümunə verilmişdir:

İD	EV	MƏQSƏD	MƏBLƏĞ	STATUS
7	şəxsi	şəxsi	aşağı	?

Cədvəl 2.3: Klassifikasiya olunması üçün verilmiş nümunə.

Öyrədici verilənlərdən istifadə edərək yeni nümunənin daxil olduğu sinfin hesablanmasına baxaq. Bunun üçün ilk növbədə 2.14 düsturunda iştirak edən ehtimalları hesablayaq (Cədvəl 2.4 - 2.6).
Öyrədici verilənlərin içərisində "təsdiq" sinfinə aid olan nümunələrin sayı 3, "imtina" tipinə aid olan nümunələrin sayı isə 3 olduğuna görə

EV

	Təsdiq	İmtina	P(c = təsdiq)	P(c = imtina)
Kirayə	0	2	0/3	2/3
İpoteka	1	0	1/3	0/3
Şəxsi	2	1	2/3	1/3
Cəmi	3	3	1	1

Cədvəl 2.4: *EV* əlaməti üçün şərti ehtimalların hesablanması

MƏQSƏD

	Təsdiq	İmtina	P(c = təsdiq)	P(c = imtina)
Şəxsi	1	0	1/3	0/3
Təhsil	0	2	0/3	2/3
Tibbi	1	0	1/3	0/3
İnvestisiya	1	1	1/3	1/3
Cəmi	3	3	1	1

Cədvəl 2.5: *MƏQSƏD* əlaməti üçün şərti ehtimalların hesablanması

MƏBLƏĞ

	Təsdiq	İmtina	P(c = təsdiq)	P(c = imtina)
aşağı	1	1	1/3	1/3
orta	1	0	1/3	0/3
yüksək	1	2	1/3	2/3
Cəmi	3	3	1	1

Cədvəl 2.6: *MƏBLƏĞ* əlaməti üçün şərti ehtimalların hesablanması

P(c=təsdiq)=3/6
P(c=imtina)=3/6
kimi hesablanır. Qeyd edək ki, hesablama zamanı şərti ehtimalların sıfır olmaması üçün müxtəlif hamarlama alqoritmlərindən istifadə edilir [13]. Bunlardan ən sadəsi sayı sıfır olan nümunnələrə vahid əlavə etməkdir, bu nümunədə də biz həmin yanaşmadan istifadə edəcəyik. İndi tutaq ki, aşağıdakı şəkildə əlamətlər verilib, Naïve-Bayes üsulu ilə bu nümunənin daxil olduğu sinfi müəyyən edək (Cədvəl 2.3). 2.14 düsturundan istifadə edərək Cədvəl 2.3 - də verilmiş nümunənin *təsdiq* və *imtina* siniflərinə daxil olma ehtimalını hesablayaq:

- $P(c = təsdiq \mid f_1 = şəxsi,\ f_2 = şəxsi,\ f_3 = aşağı) = P(c = təsdiq)P(\ f_1 = şəxsi \mid c = təsdiq\ )P(f_2 = şəxsi \mid c = təsdiq\ )P(f_3 = aşağı \mid c = təsdiq) = \frac{1}{2} \cdot \frac{2}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{27}$
- $P(c = imtina \mid f_1 = şəxsi,\ f_2 = şəxsi,\ f_3 = aşağı) = P(c = imtina)P(\ f_1 = şəxsi \mid c = imtina\ )P(f_2 = şəxsi \mid c = imtina\ )P(f_3 = aşağı \mid c = imtina)$

$$= \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} = \frac{1}{54}$$

Beləliklə, verilmiş nümunənin *təsdiq* sinfinə daxil olma ehtimalı daha böyük olduğuna görə, təsvir olunan parametrlərə malik kredit müraciəti edən şəxsin müraciəti üçün *təsdiq* qərarı verilə bilər.

Naive - Bayes üsulunun Python mühitində icra olunması

İndi isə Python *sklearn.naive_bayes* kitabxanasının *GaussianNB* funksiyasından istifadə edərək klasifikasiya məsələsini həll edək və üsulun dəqiqliyini müəyyən edək. Bu məqsədlə Avropanın müxtəlif xəstəxanalarından alınan məlumatlardan istifadə edərək insanların müayinə məlumatları əsasında onların ürək xəstəliyi risk qrupunda olub-olmaması haqqındakı verilənlərdən istifadə edəcəyik [14]. Qeyd olunan bazada 7 atribut verlənlərinə malik 303 nümunə mövcuddur, onlardan 20 faizini test üçün istifadə edək (listinq 1).

Listinq 1: Verilənlərin yüklənməsi, giriş - çıxış verilənlərinin təyin olunması, onların öyrədici və test bazalarına ayrılması

```
1 import numpy as np
2 import pandas as pd
3 data = pd.read_csv('heart.csv')
4 X = data.drop('target', axis = 1)
5 y = data['target']
6 from sklearn.model_selection import train_test_split
7 X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=1)
```

Bu nümunədə verilənlər daxil edilir, atribut və proqnozlaşdırılacaq əlamətlər təyin olunur və verilənlər öyrədici və test olaraq iki hissəyə ayrılır (test bazası bütün verilənlərin 20 % - ni əhatə edir. Test bazasının ölçüsünü dəyişmək üçün Listinq 1-də 7-ci sətirdə *test_size* dəyişəninin qiymətini dəyişmək lazımdır). Proqramın növbəti hissəsində *sklearn.naive_bayes* kitabxanasından *GaussianNB* funksiyası çağırılır, öyrədici bazasında öyrətmə aparılır, daha dəqiq ifadə etsək, 2.14 düsturunun sağ tərəfindəki şərti ehtimallar hesablanır. Daha sonra üsulun performansının qiymətləndirilməsi üçün qurulmuş model əsasında test bazasının nümunələrinin daxil olduğu siniflər proqnozlaşdırılır (Listinq 2). Klassifikiasiyanın dəqiqliyinin, test bazasındakı nümunələrin real çıxış qiymətləri ilə proqnozlaşdırılan qiymətlər arasındakı münasibətin qiymətləndirilməsi üçün *sklearn.metrics* kitabxanasından *classification_report* funksiyasını çağırmaq lazımdır. Klassifikasiya nəticəsində düzgün və yanlış proqnozlaşdırılan nümunələrin tezliklərinin müəyyən olunması üçün *sklearn.metrics* kitabxanasından *confusion_matrix* funksiyasını çağırmaqla qarışıqlıq matrisini tərtib etmək olar.

```
              precision    recall  f1-score   support

           0       0.79      0.73      0.76        30
           1       0.76      0.81      0.78        31

    accuracy                           0.77        61
   macro avg       0.77      0.77      0.77        61
weighted avg       0.77      0.77      0.77        61
```

Şəkil 2.5: *heart.csv* bazasının Naive - Bayes üsulu ilə klassifikasiyasının müvəffəqiyyət göstəriciləri

Listinq 2: Nümunələrin öyrədilməsi və üsulun performansının qiymətləndirilməsi.

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train, y_train)
y_pred = gnb.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

Nəticədə test bazasının nümunələrinin proqnozlaşdırılması əsasında aşağıdakı şəkildə qarışıqlıq matrisi alınır (Cədvəl 2.7).

		Proqnozlaşdırılan	
		y = 0	y = 1
Real	y = 0	22	8
	y = 1	6	25

Cədvəl 2.7: *heart.csv* verilənlər bazasının klassifikasiyası nəticəsində alınan qarışıqlıq matrisi

Qarışıqlıq matrisindən istifadə etməklə verilmiş nümunələrin klassifikasiyası üçün Naive - Bayes üsulunun tətbiqinin müvəffəqiyyət göstəricilərini asanlıqla hesablamaq olar (cədvəl 2.7). Python *sklearn.metrics* kitabxanasının *classification_ report* funksiyasının nəticələrinə əsasən *heart.csv* bazasının Naive - Bayes üsulu ilə klassifikasiyasının müvəffəqiyyət göstəriciləri şəkil 2.5 - də göstərilmişdir. Bu üsulla klassifikasiyanın dəqiqliyi 77 % - ə bərabərdir.

2.2.2 Mətnlərin təsnifatı

Bu bölmədə mətnlərin kateqoriyalara ayrılması, mətnlərdə duyğuların təhlili (*sentiment analysis*), yazıçının hər hansı bir obyekt haqqında müsbət və ya mənfi

fikirdə olduğunun aşkarlanmasına Naive - Bayes üsulunun tətbiqini nəzərdən keçirəcəyik. **Duyğuların təhlili**. Veb mühitində hər hansı bir kitab, film və ya obyekt haqqında şərh həmin məhsul haqqında istifadəçinin duyğularını ifadə edir, həmçinin siyasi bir mətn hər hansı bir namizəd və ya siyasi hadisə haqqında duyğuları ifadə edir. Müştərilərin və cəmiyyətin duyğularının aşkarlanması marketinqdən siyasətə kimi geniş istiqamətlərin ən vacib məsələlərindəndir.
Duyğuların aşkarlanmasının ən sadə versiyası binar klassifikasiya məsələsidir. Nümunə üçün tutaq ki, filmlər və restoranlar haqqında verilmiş mənfi və müsbət rəylərdən aşağıdakı sözlər və ifadələr çıxarılmışdır. Məsələn, *zəngin*, *möhtəşəm*, *heyif*, *bərbad*, *möhtəşəm*, *gülünc* kəlmələri çox informativ ifadələrdir:

- \+ ...zəngin personajlar, zəngin şəkildə tətbiq olunan satira və bəzi böyük süjet keçidləri
- \- Çox heyif. Bunun ən pis tərəfi boks səhnələri idi...
- \+ ...möhtəşəm karamel sousu və şirin qızardılmış badam. Mən bu yeri sevirəm!
- \- ...bərbad pizza və gülünc dərəcədə baha qiymətlər...

Spamların təyin olunması - elektron məktubun *spam* və ya *qeyri - spam* siniflərindən birinə aid edilməsi üçün binar klassifikasiya məsələsi mətn təsnifatının digər mühüm kommersiya tətbiqidir. Bu təsnifatı həyata keçirmək üçün müxtəlif leksik və digər əlamətlərdən istifadə etmək olar. Məsələn, *onlayn əczaçılıq* və ya *HEÇ BİR XƏRC ÖDƏMƏDƏN* və ya *Hörmətli qalib* ifadələrindən birinə malik olan elektron məktubdan şübhələnmək kifayət qədər ağlabatandır.

Mətnlər haqqında bilmək istəyəcəyimiz növbəti vacib məsələ onun yazıldığı dildir. Məsələn, sosial mediada yazılan mətnlər müxtəlif dillərdə yazıla bilər, onlara müxtəlif emal yanaşmaları tətbiq olunmalıdır. **Dil identifikasiyası** bir çox mətn emalı proseslərinin ilk addımıdır. **Müəllif identifikasiyası** - mətnin müəllifinin təyin olunması kimi oxşar mətn emalı prosesi də humanitar elmlərin rəqəmsallaşdırılması, sosial elmlər və məhkəmə linqvistikası kimi sahələrdə çox vacib məsələlərdəndir.

Sənədlərin təsnifatı. Nəhayət, mətn klassifikasiyasının ən qədim problemlərindən biri mətnə kitabxana kateqoriyası və ya mövzu başlığı təyin etməkdən ibarətdir. Tədqiqat məqaləsinin epidemologiyaya və ya embriologiyaya aid olmasının təyini informasiya axtarışının vacib komponentlərindəndir.
Mətnlərin klassifikasiyasında mətndən ibarət olan sənəd **söz boğçası** şəklində təsvir olunur. Burada sözlərin ardıcıllığı nəzərə alınmır, əsasən onların sənəddəki tezliyi əhəmiyyət daşıyır (Şəkil 2.6).

Mətn klassifikasiyası üçün 2.14 düsturundakı $P(c = c_j)$ *, burada j = 1, 2, ..., M,* M - mövcud siniflərin sayı və $P(f_i|c = c_j)$ *, burada i = 1, 2, ..., N,* N əlamətlərin sayıdır, ehtimallarını necə ifadə edə bilərik? Ən sadə üsul verilənlərin tezliklərindən istifadə etməkdir. Siniflərin ehtimalını $P(c = c_j)$ təyin etmək üçün öyrədici verilənlər çoxluğunda hər bir sinfə aid olan sənədlərin nisbətini hesablamaq lazımdır.

Şəkil 2.6: Söz boğçası və hər bir sözün işlənmə tezliyinin təsviri

Tutaq ki, S_{doc} - öyrədici bazadakı bütün sənədlərin sayı, $S_{c=c_j}$, c_j sinfindən olan sənədlərin sayıdır, onda c_j sinfinin ehtimalı aşağıdakı şəkildə hesablanır:

$$P(c = c_j) = \frac{S_{c=c_j}}{S_{doc}}. \tag{2.17}$$

$P(f_i|c = c_j)$ ehtimalını hesablamaq üçün hər hansı sözün sənədlərin söz boğçasında olub olmamasını əlamət kimi götürəcəyik. Beləliklə, $(P(w_k|c_j)$ ehtimalını c_j sinfinə aid olan bütün sənədlərin içərisində w_k sözünün işlənmə tezliyi kimi hesablayacağıq, burada w_k - söz boğçasındakı k - cı sözdür, $k = 1, 2, \ldots, L$, L - lüğətdə toplanmış bütün sözlərin ümumi sayıdır. İlk öncə c_j sinfinə daxil olan bütün sənədlərin sözlərindən ibarət böyük sözlər çoxluğu hazırlanır və w_k sözünün verilmiş sinfə aid olma ehtimalı hesablanır:

$$P(c = c_j|w_k) = \frac{count(w_k, c_j)}{\sum_{w \in V} count(w, c_j)}, \tag{2.18}$$

burada V yalnız c_j sinfində deyil, bütün siniflərə aid olan sənədlərdə olan bütün sözlər çoxluğudur. Burada da hasilin hesablanması zamanı ehtimalların sıfır alınmasının qarşısını almaq üçün hamarlaşdırma üsullarından istifadə olunur. Bunlardan ən sadəsi tezliklərə vahidin əlavə olunmasıdır:

$$P(w_k|c = c_j) = \frac{count(w_k, c_j) + 1}{\sum_{w \in V}(count(w, c_j) + 1)} + \frac{count(w_k, c_j) + 1}{(\sum_{w \in V} count(w, c_j)) + |V|)}. \tag{2.19}$$

Test verilənlərində istifadə olunan, lakin hər hansı bir sinfin öyrədici sənədlərinin içərisində olmadığına görə lüğətdə olmayan *bilinməyən sözlər* olduqda ən yaxşı həll yolu həmin sözlərə etinasız yanaşmaq və onlar üçün heç bir ehtimal hesablamamaqdır. Nəhayət, bir çox sistemlər bəzi sözləri ümumiyyətlə təsnifat zamanı istifadə etmirlər. Çox intensiv istifadə olunan *və*, *ki*, *da*, *də* tipli sözlər **stop sözlər**

kimi təyin olunur və öyrədici mətnlərdən çıxarılır. Bu məqsədlə lüğətdəki sözlər istifadə tezliyinə görə azalan sırada düzülür və ən çox istifadə olunan top 10 - 100 söz stop sözlər kimi qeyd olunur, həm öyrədici, həm də test bazasından çıxarılır.

Duyğuların təhlili. Nümunə

Naive - Bayes üsulunun öyrədilməsi və test edilməsi üçün nümunəni, (+) müsbət və (-) mənfi siniflər üçün duyğuların təhlilini nəzərdən keçirək, aşağıdakı şəkildə real film şərhlərindən ibarət kiçik öyrədici və test bazasından istifadə edək (cədvəl 2.8):

	Sinif	Sənədlər
Öyrədici	-	çox darıxdırıcı
	-	tamamilə proqnozlaşdırılan və enerjidən məhrumdur
	-	sürprizlər yoxdur və az güldürən filmdir
	+	çox güclü
	+	yayın ən güldürən filmi
Test	?	az güldürən və proqnozlaşdırılan sonluqdur

Cədvəl 2.8: Mətnlərin təsnifatı üçün öyrədici və test bazasına nümunə

Hər bir sinfin ehtimalları 2.17 düsturu ilə asanlıqla hesablanır:

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}.$$

Təsnifata başlamazdan öncə *sonluqdur* sözünü öyrədici bazada rast gəlinmədiyinə görə bilinməyən söz kimi, *və* sözünü isə stop söz kimi siyahıdan silə bilərik. Ümumilikdə, (-) mənfi sinfindən olan cümlələrdə 11, (+) müsbət sinfində isə 6 fərqli söz işlənmişdir. Bütün şərhlərdə işlənən fərqli sözlərn sayı, yəni lüğətdə mövcud olan sözlərin sayı 15 -ə bərabərdir. Klassifikasiya üçün w_1 =*az*, w_2 =*güldürən*, w_3 =*proqnozlaşdırılan* sözlərinin öyrədici verilənlərə görə şərti ehtimallarını 2.19 düsturuna əsasən hesablayaq:

$$P(w_1|-) = \frac{1+1}{11+15}, \qquad P(w_2|-) = \frac{1+1}{11+15}, \qquad P(w_3|-) = \frac{1+1}{11+15},$$

$$P(w_1|+) = \frac{0+1}{6+15}, \qquad P(w_2|+) = \frac{1+1}{6+15}, \qquad P(w_3|+) = \frac{0+1}{6+15}.$$

T = *az güldürən və proqnozlaşdırılan sonluqdur* cümləsinin *sonluqdur* və *və* sözlərini sildikdən sonra siniflərə aid olması ehtimalları aşağıdakı şəkildə olur:

$$P(-)P(-|T) = \frac{3}{5} \times \frac{2^3}{26^3} \approx 0.0003,$$

$$P(+)P(+|T) = \frac{2}{5} \times \frac{2}{21^3} \approx 0.00009.$$

Beləliklə, model verilmiş cümləni *mənfi* sinfinə aid edir.

2.2.3 kNN - k ən yaxın qonşu alqoritmi

Ən yaxın qonşu alqoritmləri müəllimli maşın öyrətmə üsulları içərisində "ən sadə" alqoritmlərdən biridir, həm klassifikasiya, həm də proqnozlaşdırma məsələlərinə tətbiq oluna bilər. kNN verilmiş şərtlər daxilində approksimasiya edici funksiya olsa da, əsas yanaşma nisbətən sadədir. kNN öyrətmə zamanı öyrədici verilənləri

$$(x^i, y^i) \in D, (|D| = n)$$

burada n – öyrədici verilənlərin sayıdır, özündə saxlayan müəllimli öyrətmə üsuludur. Məhz buna görə kNN alqoritmini bəzən tənbəl öyrətmə üsulu da adlandırırlar, belə ki, öyrədici verilənlərin emalı yalnız proqnozlaşdırma zamanı baş verir, bu zamana qədər öyrətmə sadəcə olaraq öyrədici verilənlərin saxlanmasından ibarət olur. Daha sonra proqnozlaşdırmanın (sinfin adı və ya nömrəsi və ya ədədi çıxış veriləni) həyata keçirilməsi üçün kNN alqoritmi öyrədici verilənlər içərisində daxil olunan verilənin k sayda ən yaxın qonşusunu təyin edir və bu yaxın obyektlər əsasında ya sinfin adını, ya nömrəsini, ya da ədədi çıxış verilənini hesablayır.

kNN alqoritmində əsas problem verilənlər arasında məsafələrin hesablanması üçün düzgün metrikanın seçilməsidir, belə ki, müxtəlif xarakterli verilənlər üçün fərqli məsafə üsullarının seçilməsi daha məqsədəuyğundur. Məsələn, binar əlamətlərdən ibarət bazada nöqtələr arasındakı məsafənin hesablanması üçün kosinus oxşarlıq metrikası, ədədi verilənlərdə isə Evklid məsafəsinin tətbiq olunması daha effektiv olur.
kNN üsulunun alqoritmi aşağıdakı şəkildədir:

1. k - ən yaxın qonşuların sayı və məsafənin hesablanması üçün uyğun metrika seçilir;

2. Öyrədici bazadan $(x^i, y^i), i = \overline{1, n}$ verilənləri saxlanılır, burada x^i – öyrədici verilənlərin parametrlər vektorudur. Proqnozlaşdırılması üçün daxil edilən x^q ilə öyrədici bazadakı bütün verilənlər arasında məsafələr hesablanır, məsələn tutaq ki, Evklid metrikası ilə:

$$d_i = ||x^i - x^q||, i = \overline{1, n}; \tag{2.20}$$

3. Hesablanmış məsafələr artan sıra ilə düzülür və ilk sıradan k ədəd ən kiçik məsafəyə uyğun öyrədici verilənlər təyin olunur. Qərarın qəbul edilməsi zamanı çıxış veriləni olaraq əgər klasifikasiya məsələsidirsə, ən yaxın qonşuların çoxluğunun aid olduğu sinif götürülür, proqnozlaşdırma məsələsidirsə ən yaxın qonşuların çıxış verilənləri üzərində müəyyən əməliyyatlar aparmaqla, məsələn onların ədədi ortasını təyin etməklə nəticə müəyyən olunur.

kNN üsulu ilə təsnifat. Nümunə

Üsulun daha anlaşıqlı olması üçün Cədvəl 2.2 - də verilmiş credit_risk_dataset verilənlər çoxluğunu nəzərdən keçirək. İlk öncə kNN üsulunun tətbiqi zamanı

məsafələrin hesablanmasını təmin etmək üçün keyfiyyət göstəricilərini kəmiyyət göstəricilərinə çevirmək lazımdır.
EV veriləni üçün *şəxsi*, *ipoteka* və *kirayə* ketaqoriyalarını uyğun olaraq 1, 2, 3, *MƏQSƏD* parametrinin *şəxsi*, *təhsil*, *tibbi* və *investisiya* kateqoriyalarını 1, 2, 3, 4, *MƏBLƏĞ* veriləni üçün *aşağı*, *orta* və *yuxarı* ketaqoriyalarını 0, 1, 2 rəqəmləri ilə əvəz edək. Nəticədə atribut verilənləri rəqəmsal olan öyrədici nümunələr əldə etmiş oluruq. (Cədvəl 2.9).

İD	EV	MƏQSƏD	MƏBLƏĞ	STATUS
1	1	1	2	təsdiq
2	1	2	0	imtina
3	2	3	1	təsdiq
4	3	2	2	imtina
5	1	4	0	təsdiq
6	2	4	2	imtina

Cədvəl 2.9: Credit_risk_dataset verilənlər bazasının rəqəmsal atributlar ilə təsviri

Daha sonra verilmiş nümunənin də atributlarını rəqəmsal parametrlərə çevirib, yeni nümunə (Cədvəl 2.10) ilə öyrədici bazadakı bütün elementlər arasındakı məsafələri hesablayaq. Metrika olaraq Evklid məsafəsini götürək (Cədvəl 2.11).

İD	EV	MƏQSƏD	MƏBLƏĞ	STATUS
7	1	1	0	?

Cədvəl 2.10: Klassifikasiya olunması üçün verilmiş nümunə (rəqəmsal atributlarla).

N	UYĞUNLUQ D.	MƏSAFƏ
1	təsdiq	2.00
2	imtina	1.00
3	təsdiq	2.45
4	imtina	3.00
5	təsdiq	3.00
6	imtina	3.74

Cədvəl 2.11: Öyrədici nümunələrin çıxış qiymətləri və yeni verilmiş nümunə ilə aralarındakı məsafənin uzunluğu

k ən yaxın qonşu üsulunda $k = 3$ götürsək cədvəldən test üçün verilmiş nümunəyə ən yaxın 3 qonşunu aşağıdakı şəkildə təyin edə bilərik:

- 2 - məsafə = 1.00 (sinif = imtina)
- 1 - məsafə = 2.00 (sinif = təsdiq)
- 3 - məsafə = 2.45 (sinif = təsdiq)

Göründüyü kimi 3 ən yaxın qonşunun ikisi, yəni çox hissəsi *təsdiq* sinfinə aid olduğundan yeni nümunəni də bu sinfə aid edirik.

kNN üsulu ilə klassifikasiyanın Python mühitində icrası

İndi isə Python *sklearn.neighbors* kitabxanasının *KNeighborsClassifier* funksiyasından istifadə edərək klasifikasiya məsələsini həll edək və üsulun dəqiqliyini müəyyən edək. Bu məqsədlə 2.2.1-ci bölmədəki kimi ürək xəstəlikləri haqqında verilənlər bazasından istifadə edək. Eyni qayda ilə onlardan 20 faizini test üçün istifadə edək (listinq 3) və k = 3 götürək.

listinq 3: Python sklearn mühitində kNN üsulunun icrası

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('heart.csv')
X = data.drop('target', axis = 1)
y = data['target']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
from sklearn.neighbors import KNeighborsClassifier
neigh = KNeighborsClassifier(n_neighbors=3)
neigh.fit(X_train, y_train)
y_pred = neigh.predict(X_test)
from sklearn.metrics import classification_report,
    accuracy_score
print(classification_report(y_test, y_pred))
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

Nəticədə test bazasının nümunələrinin proqnozlaşdırılması əsasında aşağıdakı şəkildə qarışıqlıq matrisi alınır (Cədvəl 2.12).

		Proqnozlaşdırılan	
		y = 0	y = 1
Real	y = 0	20	9
	y = 1	12	20

Cədvəl 2.12: *heart.csv* verilənlər bazasının klassifikasiyası nəticəsində alınan qarışıqlıq matrisi

Qarışıqlıq matrisindən istifadə etməklə verilmiş nümunələrin klassifikasiyası üçün kNN ($k = 3$ olduqda) üsulunun tətbiqinin müvəffəqiyyət göstəricilərini asanlıqla hesablamaq olar. Python *sklearn.metrics* kitabxanasının *classification_report* funksiyasının nəticələrinə əsasən *heart.csv* bazasının kNN ($k = 3$ olduqda) üsulu

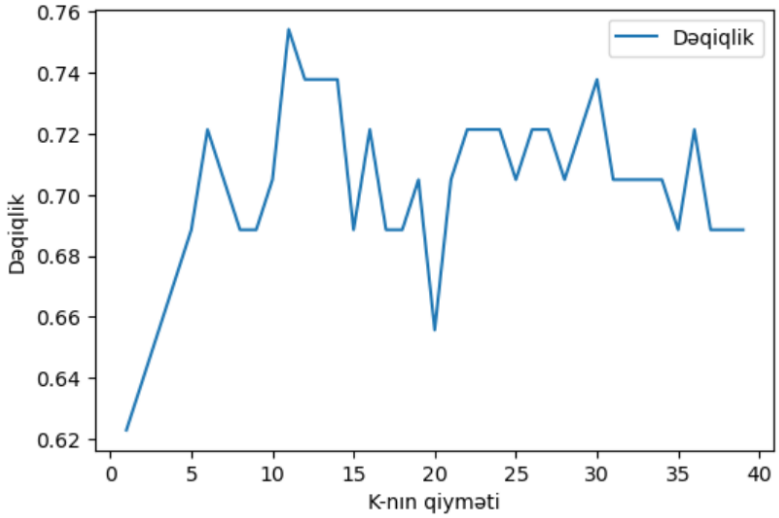
	precision	recall	f1-score	support
0	0.62	0.69	0.66	29
1	0.69	0.62	0.66	32
accuracy			0.66	61
macro avg	0.66	0.66	0.66	61
weighted avg	0.66	0.66	0.66	61

Şəkil 2.7: *heart.csv* bazasının kNN ($k = 3$ olduqda) üsulu ilə klassifikasiyasının müvəffəqiyyət göstəriciləri

	precision	recall	f1-score	support
0	0.77	0.69	0.73	29
1	0.74	0.81	0.78	32
accuracy			0.75	61
macro avg	0.76	0.75	0.75	61
weighted avg	0.76	0.75	0.75	61

Şəkil 2.8: *heart.csv* bazasının kNN ($k = 11$ olduqda) üsulu ilə klassifikasiyasının müvəffəqiyyət göstəriciləri

ilə klassifikasiyasının müvəffəqiyyət göstəriciləri şəkil 2.7 - də göstərilmişdir. Bu üsulla klassifikasiyanın dəqiqliyi 66 % - ə bərabərdir.
$k = 11$ götürək və nəticəni müşahidə edək (şəkil 2.8), bunun üçün listinq 4-ün 10-cu sətrində *"n_ neighbors"* parametrinin qiymətini dəyişmək lazımdır.
Ümumiyyətlə, k parametrinin optimal qiymətinin tapılması üçün müxtəlif yanaşmalar mövcuddur. Ən çox istifadə olunan üsullardan biri əyri üsuludur (elbow method). Bu zaman k-nın müxtəlif qiymətləri üçün dəqiqlik hesablanır və k-nın qiymətləri ilə uyğun dəqiqlik qiymətləri arasında əyri çəkilir (listinq 4). Bu əyriyə əsasən ən böyük dəqiqliyə uyğun k-nın qiyməti optimal olaraq götürülür (şəkil 2.9). Şəkildən göründüyü kimi ən yüksək dəqiqlik k=11 olduqda 75% kimi müşahidə olunur.



Şəkil 2.9: kNN üsulunda k - nın qiymətləri ilə modelin dəqiqliyinin asılılıq qrafiki

listinq 4: kNN üsulunun icrası zamanı k - nın optimal qiymətinin tapılması

```
accuracy_rates = []
for k in range(1,40):
    knn_model = KNeighborsClassifier(n_neighbors=k)
    knn_model.fit(X_train,y_train)
    y_pred = knn_model.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)
    accuracy_rates.append(accuracy)
plt.figure(figsize=(6,4),dpi=100)
plt.plot(range(1,40),accuracy_rates)
```

kNN üsulunda öyrətmə fazası digər klassifikasiya alqoritmlərinə nisbətən daha sürətlidir. Bu üsulda ümumiləşdirmək məqsədi ilə öyrətmə həyata keçirilmir, məhz bu səbəbdən kNN ən sadə və verilən əsaslı öyrətmə alqoritmi kimi qəbul olunur. Buna baxmayaraq kNN üsulu zamanı test fazası vaxt və xərc tələb edir. Proqnozlaşdırmanın həyata keçirilməsi üçün öyrədici bazanın bütövükdə yadda saxlanılması üçün böyük həcmdə yaddaş tələb olunur, həmçinin məsafələrin hesablanması üçün Evklid metrikasından istifadə olunduğundan verilənlərin normallaşdırılması tələb olunur.

Yuxarıda da qeyd olunduğu kimi kNN üsulu klassifikasiya ilə yanaşı kəsilməz çıxış verilənlərinin də proqnozlaşdırılmasına tətbiq oluna bilər.

kNN üsulu ilə reqressiya. Nümunə

Tutaq ki, bir şirkət tərəfindən müxtəlif aylarda COVID hallarının sayı və satılan maskaların sayı haqqında verilənlər mövcuddur (cədvəl 2.13). Bu verilənlərdən istifadə etməklə avqust ayı üçün halların sayı 1383 olduqda satılacaq maskaların sayını təxmin etmək lazımdır. kNN üsulu ilə bu məsələni həll edək. İlk öncə k hiper-

parametrinin qiymətini verək: $k = 3$. Daha sonra proqnozlaşdırılan nümunənin ən yaxın 3 qonşusunu təyin etmək üçün Evklid metrikası ilə bütün nümunələrin atribut veriləni (yoluxma hallarının sayı) ilə arasında olan məsafəni hesablayaq (Cədvəl 2.14).

Ay	Yoluxma hallarının sayı	Satılan maskaların sayı
Yanvar	40	14
Fevral	190	26
Mart	340	35
Aprel	680	130
May	720	450
İyun	900	700
İyul	1120	800
Avqust	1383	?

Cədvəl 2.13: Aylar üzrə satılan maskaların sayının COVID hallarından asılılığı

Ay	Satılan maskaların sayı	Proqnozlaşdırılan nümunə ilə məsafə
Yanvar	14	1343
Fevral	26	1193
Mart	35	1043
Aprel	130	703
May	450	663
İyun	700	483
İyul	800	263

Cədvəl 2.14: Öyrədici verilənlərlə proqnozlaşdırılan nümunə arasındakı məsafənin uzunluğu

Hesablanmış məsafələrə əsasən proqnozlaşdırmaq üçün verilmiş nümunəyə ən yaxın 3 qonşunu aşağıdakı şəkildə təyin edə bilərik:

- İyul - məsafə = 263 (satılan maskaların sayı = 800)
- İyun - məsafə = 483 (satılan maskaların sayı = 700)
- May - məsafə = 663 (satılan maskaların sayı = 450)

Avqust ayı üçün satılan maskaların sayını proqnozlaşdırmaq üçün seçilmiş 3 ən yaxın qonşunun çıxış verilənlərinin ədədi ortasını tapaq:

$$N = \frac{800 + 700 + 450}{3} = 650.$$

Beləliklə, avqust ayında yoluxma saylarına əsasən satılan maskaların sayı kNN üsulu ilə 650 kimi proqnozlaşdırılır.

kNN üsulu ilə proqnozlaşdırmanın Python mühitində icrası

İndi isə Python *sklearn.neighbors* kitabxanasının *KNeighborsRegressor* funksiyasından istifadə edərək proqnozlaşdırma məsələsini həll edək və üsulun dəqiqliyini müəyyən edək. Bu məqsədlə bir şirkətin müxtəlif mənbələrdə reklama xərclədiyi məbləği və satış həcmini göstərən *advertising.csv* verilənlər çoxluğundan istifadə edək [15]. Bu verilənlər bazasında televiziya, radio və qəzetlərdə verilmiş reklamlara xərclənən pul və satış miqdarı göstərilmişdir. Verilənlər çoxluğunun 20 faizini test üçün istifadə edək (listinq 5) və $k = 3$ götürək. Bu parametrlər üçün üsulun dəqiqliyini *R2* qiyməti ilə təyin edək, daha sonra k parametrinə müxtəlif qiymətlər verməklə ən yüksək dəqiqliyə malik olan modeli müəyyənləşdirək (listinq 6).

Listinq 5: Python sklearn mühitində kNN üsulu ilə proqnozlaşdırma

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
data = pd.read_csv("Advertising.csv")
X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.2, random_state = 42)
knn = KNeighborsRegressor(n_neighbors = 3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
r2 = r2_score(y_test, y_pred)
print('R2 = ', r2)
```

$k = 3$ üsulun dəqiqliyini qiymətləndirən R2 qiyməti 0.90 - a bərabər olur, bu isə o deməkdir ki, verilənlərin təxminən 90 % - i düzgün proqnozlaşdırılır. Buna baxmayaraq proqnozlaşdırmanın dəqiqliyini k hiperparametrinin optimal qiymətini verməklə daha da artırmaq olar. Listinq 6 - da göstərilmiş kodun icrası nəticəsində modelin ən yüksək dəqiqliyi $k = 9$ olduqda təxminən 0.92 - yə bərabər olur. k-nın optimal qiyməti üçün $r2\_rates$ vektorunun maksimum qiymətinin indeksi ilə təyin edilir. *Pythonnumpy* mühitində indeksləmə 0-dan başladığına görə listinq 6-nın 7-ci sətrində maksimum indeksin üzərinə vahid gəlmək lazımdır.

Listinq 6: k parametrinin optimal qiymətinin tapılması

```
r2_rates = []
for i in range(1, 40):
    knn = KNeighborsRegressor(n_neighbors = i)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    r2_rates.append(r2_score(y_test, y_pred))
print('Optimal k = ', np.argmax(r2_rates) + 1)
print('Optimal R2 = ', np.max(r2_rates))
```

Çalışmalar

1. Binar klassifikasiya məsələsinin test mərhələsində aşağıdakı nəticələr alınmışdır:
   Real qiymətlər: $[1, 0, 1, 0, 1, 1, 0, 1]$
   Hesablanan qiymətlər: $[1, 0, 0, 1, 1, 0, 0, 1]$
   Modelin dəqiqliyini hesablayın.

2. Binar klassifikasiya məsəsləsi üçün aşağıdakı qarışıqlıq matrisi alınmışdır: Həm pozitiv, həm tə neqativ siniflər üçün səhihliyi və spesifikliyi hesablayın.

		Proqnozlaşdırılan	
		Pozitiv	Neqativ
Real	Pozitiv	120	30
	Neqativ	20	130

3. Evlərin qiymətlərinin proqnozlaşdırılması məsələsinin test mərhələsində aşağıdakı nəticələr alınmışdır:
   Real qiymətlər: $[250000, 280000, 310000, 350000, 410000]$
   Proqnozlaşdırılan qiymətlər: $[255000, 275000, 305000, 340000, 400000]$
   Modelin mütləq orta xətasını hesablayın.

4. 3-cü tapşırıqdakı verilənlər üçün modelin orta kvadratik xətasının kvadrat kökünü hesablayın.

5. "Spam" və "Normal" kimi işarə olunan email siniflərinin klassifikasiya məsələsinə baxaq.
   $P(Spam) = 0.3$
   $P(Normal) = 0.7$
   $P(\text{"}ucuz\text{"}|Spam) = 0.8$
   $P(\text{"}ucuz\text{"}|Normal) = 0.2$
   $P(\text{"}lotoreya\text{"}|Spam) = 0.6$
   $P(\text{"}lotoreya\text{"}|Normal) = 0.1$
   ehtimallarından istifadə edərək *Ucuz lotoreya biletlərini indi alın!* email mətninin sinfini (Spam və ya Normal) proqnozlaşdırın.

6. (1, 2), (3, 4), (5, 6), (7, 8) nöqtələri və test nöqtəsi Q(4, 5) verilmişdir. Evklid metrikası ilə Q nöqtəsinin ən yaxın 2 qonşusunu təyin edin.

7. Müxtəlif məkanlardan gələn biliklər əsasında temperaturu təyin etmək lazımdır. Hər bir nöqtə məkanın enliyi və uzunluğu ilə və ölçülən temperatur qiyməti ilə verilmişdir:
   $X = [[40.71, -74.01], [34.05, -118.24], [51.51, -0.13]]$
   $y = [72, 80, 65]$

Evklid metrikasından istifadə edərək yeni məkanın $[[37.77, -122.42]]$ temperaturunu k ən yaxın qonşu üsulu ilə $k = 3$ və $k = 2$ olduqda proqnozlaşdırın.

8. Məhsulun satış miqdarının tarixi məlumatları verilmişdir:
$X = [[1], [2], [3], [4], [5]]$ və $y = [[100], [120], [130], [110], [150]]$. $X = [6]$ olduqda satış miqdarını k ən yaxın qonşu üsulu ilə proqnozlaşdırın.

9. *Python* mühitində *sklearn.datasets* kitabxanasının *load_iris* funksiyasından istifadə edərək süsən güllərinin klassifikasiyasın k ən yaxın qonşu üsulunu tətbiq edin. Verilənlərin 20%-ni test üçün ayırın və modelin dəqiqlik meyarı vasitəsilə k-nın optimal qiymətini təyin edin.

10. *Python* mühitində *sklearn.datasets* kitabxanasının *load_boston* funksiyasından istifadə edərək Bostonda evlərin qiymətinin proqnozlaşdırılması üçün k ən yaxın qonşu üsulunu tətbiq edin. Verilənlərin 20%-ni test üçün ayırın və xətanın orta kvadratik qiyməti vasitəsilə k-nın optimal qiymətini təyin edin.

Fəsil 3

Reqressiya təhlili

Reqressiya analizi sosial elmlərə 1870-ci illərdə Francis Galtonun işləri ilə daxil olmağa başlayıb, lakin *ən kiçik kvadratlar* üsulunun istifadəsinin tarixi 1800-cü illərin əvvəllərinə alman riyaziyyatçısı Karl Qaussun astronomik hadisələrin proqnozlaşdırılmasında istifadəsinə dayanır [16].
Yeni yarandığı zamanlardan başlayaraq reqressiya təhlili mürəkkəb, çətin və bahalı bir iş idi. Günümüzün kompüterləri isə reqressiya analizini çoxsaylı verilənlər üzərində çox qısa zamanda həyata keçirə bilir, həmçinin Python, Java, R, C# kimi proqramlaşdırma dillərinin imkanları isə mürəkkəb riyazi əməliyyatları kodlaşdırma ehtiyacı olmadan xüsusi funksiyaların köməyi ilə yerinə yetirməklə reqressiya analizini həyata keçirməyə imkan verir. Reqressiya təhlilinin məqsədi asılı dəyişəndəki dinamikanı bir və ya bir neçə müstəqil və ya idarəedici dəyişənlər vasitəsilə izah etməkdir. Reqressiya analizinin dörd əsas tətbiq sahəsi var:

1. Təsviri və ya izahedici: *Asılı dəyişəndəki dəyişkənliyə hansı amillər təsir edir?* Məsələn, şirkətin satış məlumatları arasında satışların artmasına səbəb olan amil;

2. Proqnozlaşdırıcı, məsələn, normal kvota və ya ilkin satışların müəyyən edilməsi. *Normal* və *anormal* və ya kənar müşahidələri müəyyən etmək üçün təxmin edilən tənlikdən istifadə bilərik;

3. Alternativ nəzəri izahların müqayisəsi;

4. Qərarların mənbəsi.

Reqressiya analizində parametrlərin təyin olunması üçün müxtəlif üsullardan istifadə olunur: ən kiçik kvadratlar üsulu, maksimum oxşarlıq ehtimalının təyini üsulu, çəkili ən kiçik kvadratlar üsulu, ən kiçik mütləq yayılmalar üsulu, və s.
Ən sadə reqressiya modeli aşağıdakı şəkildə yazıla bilər:

$$asılı\ dəyişən = sabit\ parametr + əmsal \times sərbəst\ dəyişən + xəta$$

$$y = w_0 + w_1 x + E \tag{3.1}$$

burada w_0 , w_1 və E naməlum parametrlərdir. Bu zaman N sayda asılı dəyişən və sərbəst dəyişən cütləri müşahidə olunur, reqressiya analizi nəticəsində sabit və əmsal parametrlərinin uyğun qiymətləri təyin olunur.

3.1 Ən kiçik kvadratlar üsulu

Tutaq ki, n sayda $(x_i, y_i), i = 1, 2, \ldots, n$ müşahidə cütləri məlumdur. Bu müşahidələr sadə reqressiya tənliyinə (3.1) uyğun olduğu üçün yaza bilərik:

$$y_i = w_0 + w_1 x_i + \epsilon_i, i = \overline{1, n} \tag{3.2}$$

ən kiçik kvadratlar üsulunun əsas prinsipi müşahidələr və (3.2) tənliyi ilə təyin olunan xətt arasındakı fərqlərin kvadratları cəmini minimumlaşdırmaqla w_0 və w_1 parametrlərinin qiymətlərinin tapılmasından ibarətdir.
Birbaşa reqressiya üsulu kvadratlar cəminin w_0 və w_1 parametrlərinə əsasən minimumlaşdırılmasına əsaslanır:

$$S(w_0, w_1) = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i)^2 \tag{3.3}$$

(3.3) funksiyasının minimum qiymətinin tapılması üçün parametrlərə əsasən birinci tərtib törəmələrini tapaq və sıfra bərabər edək:

$$\frac{\partial S(w_0, w_1)}{\partial w_0} = -2 \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i),$$

$$\frac{\partial S(w_0, w_1)}{\partial w_1} = -2 \sum_{i=1}^{n} (y_i - w_0 - w_1 x_i) x_i.$$

Alınmış törəmələri sıfra bərabər edib, $\overline{x} = \sum_{i=1}^{n} x_i$, $\overline{y} = \sum_{i=1}^{n} y_i$, $S_{xx} = \sum_{i=1}^{n} (x_i - \overline{x})^2$, $S_{xy} = \sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})$ əvəzləmələri aparsaq və alınmış tənlikləri həll etsək, w_0 və w_1 parametrlərinin optimal qiymətləri üçün aşağıdakı düsturları alarıq:

$$w_1 = \frac{S_{xx}}{S_{xy}} \tag{3.4}$$

$$w_0 = \overline{y} - w_1 \overline{x} \tag{3.5}$$

3.1.1 Xətti reqressiya üsuluna nümunə

Xətti reqressiya üsulunu izah etmək üçün cədvəl 2.14 - də təsvir olunmuş COVID hallarının sayı və satılan maskaların sayı haqqında verilənlər təyin olunmuşdur (cədvəl 3.1). Bu verilənlərdən istifadə etməklə avqust ayı üçün halların sayı 1383 olduqda satılacaq maskaların sayını təxmin etmək lazımdır. Ən kiçik kvadratlar üsulu vasitəsilə reqressiya analizi ilə bu məsələni həll edək. İlk növbədə bu dəyişənlərin korrelyasiyasını hesablayaq və onlar arasında xətti asılılıq olub-olmamasını təyin edək. Proqramın icrası nəticəsində korrelyasiya əmsalının qiyməti üçün 0.9122 alırıq (listinq 7, sətir 7), verilənlərin paylanmasını isə vizual olaraq şəkil 3.1 - də müşahidə edə bilərik. Daha sonra reqressiya xəttinin qurulması üçün Python *numpy* kitabxanasından istifadə etməklə dövri əmrlər yazmadan parametrlərin qiymətini (3.4) və (3.5) düsturları vasitəsilə tapaq (listinq 7, sətir 8 - 13).
Proqram kodundan göründüyü kimi növbəti addımda (3.4) və (3.5) düsturlarına əsasən parametrlərin optimal qiymətləri, başqa sözlə desək reqressiya xəttinin əmsalları təyin olunur:

$$w_0 = 1.27$$

$$w_1 = -416.31$$

Alınmış parametrlərdən istifadə etməklə *avqust* ayı üçün satılacaq maskların sayının təxminən 1341 olacağını proqnozlaşdıra bilərik. Alınan reqressiya xətti isə şəkil 3.2 - də təsvir olunmuşdur.

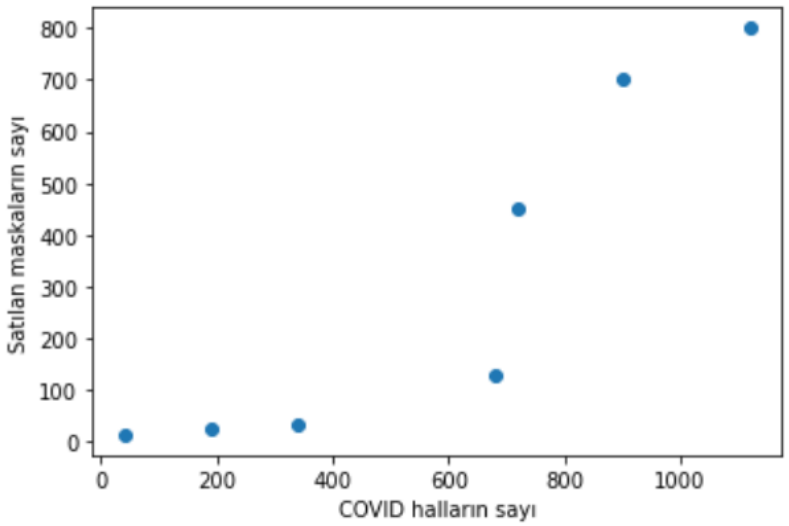
Beləliklə, alınan reqressiya xətti "COVID hallarının sayı" dəyişəninə qiymət verməklə "Satılan maskaların sayı" dəyişəninin qiymətini pronozlaşdırır. Avqustdan əvvəlki aylar üçün proqnozlaşdırılan qiymətlərlə real qiymətlər arasındakı xətanı isə R2 qiyməti və xətanın mütləq qiyməti meyarları ilə qiymətləndirə bilərik, baxılan misalda qiymətləndirmə metrikaları üçün uyğun olaraq aşağıdakı ədədlər alınır:

$$r2 = 0.78$$

$$mae = 171.36$$

3.2 Ən tez enmə üsulu

Ən kiçik kvadratlar üsulu iterativ olmayan sadə xətti reqressiya üsuludur. İterativ olaraq parametrləri təyin etmək üçün ən tez enmə qradient üsulundan istifadə olunur. Bunun üçün (3.3) funksiyasını ən tez enmə üsulu ilə minimallaşdırmaq, yəni iterativ olaraq bu funskiyanın minimum qiymət aldığı w_0 və w_1 parametrlərinin qiymətlərini təyin etmək üçün aşağıdakı alqoritmdən istifadə olunur:



Şəkil 3.1: Yoluxma hallarının sayının satılan maskaların sayından asılılığı

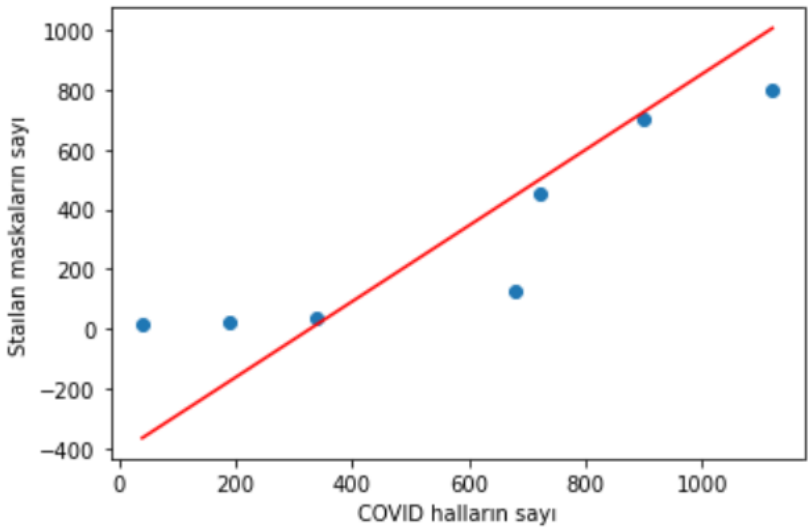
listinq 7: Ən kiçik kvadratlar üsulu ilə reqressiya xəttinin qurulması

```
import numpy as np
x = np.array([40, 190, 340, 680, 720, 900, 1120])
y = np.array([14, 26, 35, 130, 450, 700, 800])
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
corr, _ = pearsonr(x, y)
print(corr)
x_mean = np.average(x)
y_mean = np.average(y)
S_xx = np.sum(np.power(x - x_mean, 2))
S_xy = np.sum((x - x_mean)*(y - y_mean))
w1 = S_xx/S_xy
w0 = y_mean - w1*x_mean
print(w1)
print(w0)
x1 = 1383
y1 = w0+w1*x1
y1 = w0+w1*x
plt.plot(x, y1)
plt.scatter(x, y)
plt.show()
```

1. w_0 və w_1 parametrlərinin ilkin qiymətləri verilir;

2. Proqnozlaşdırılan qiymət k-cı iterasiyada

$$y_i = w_0^{(k)} + w_1^{(k)} x_i$$



Şəkil 3.2: Reqressiya xətii və nöqtələrin real paylanması

düsturu ilə, xəta funksiyasının qiyməti isə

$$S^{(k)}(w_0, w_1) = \sum_{i=1}^{n} (y_i - w_0^{(k)} - w_1^{(k)} x_i)^2$$

düsturu ilə hesablanır;

3. Parametrlərin yeni qiymətləri

$$w_0^{(k+1)} = w_0^{(k)} - \alpha \frac{\partial S^{(k)}(w_0^{(k)}, w_1^{(k)})}{\partial w_0^{(k)}}$$

$$w_1^{(k+1)} = w_1^{(k)} - \alpha \frac{\partial S^{(k)}(w_0^{(k)}, w_1^{(k)})}{\partial w_0^{(k)}}$$

düsturları ilə hesablanır, burada $\alpha-$ hiperparametrdir, maşın öyrənməsində bu əmsal öyrənmə addımı adlanır, adətən (0; 1) parçasında təyin olunur, modelin problemin həllinə nə qədər sürətlə yaxınlaşmasını göstərir. Belə ki, öyrənmə addımının çox kiçik qiyməti daha çox iterasiyaların hesablanmasını tələb edir, daha böyük qiymətlər isə həllin sürətlə dəyişməsinə və üsulun dağılmasına gətirib çıxarır;

4. 2-ci və 3-cü addımlar iterativ olaraq aşağıdakı kriteriyalardan biri ödənilənə qədər davam etdirilir:

   - Funksiyanın qiyməti verilmiş kiçik ϵ qiymətindən kiçikdir:

$$S^{(m)} < \epsilon, \quad m = 1, 2, ...$$

- İki ardıcıl iterasiyada funksiyanın qiymətlərinin fərqinin mütləq qiyməti verilmiş kiçik ϵ qiymətindən kiçikdir:

$$|S^{(m+1)} - S^{(m)}| < \epsilon, \quad m = 1, 2, ...$$

- İki ardıcıl iterasiyada hesablanan parametrlər arasındakı məsafə verilmiş kiçik ϵ qiymətindən kiçikdir:

$$||w^{(m+1)} - w^{(m)}|| < \epsilon, \quad m = 1, 2, ...$$

  burada məsafə Evklid norması ilə hesablana bilər.

- Verilmiş sayda iterasiyalar yerinə yetirilmişdir.

3.2.1 Python mühitində verilənlərin xətti reqressiya üsulu ilə təhlili

Xətti reqressiya üsulu ilə 2.2.3 - də təsvir olunmuş reklam verilənləri əsasında satış həcminin proqnozlaşdırılmasını nəzərdən keçirək. Python *sklearn.linear_model* kitabxanasından *LinearRegression* funksiyasından istifadə edək, verilənlər bazasının 80%-ni öyrətmə üçün, 20 %-ni test üçün istifadə edək və nəticənin effektivliyini qiymətləndirək. Nəticədə test bazasında reqressiyanın dəqiqliyinin qiymətləndirilməsi üçün aşağıdakı qiymətləri alarıq (Cədvəl 3.1). Beləliklə, verilmiş verilənlər bazasının test edilməsi nəticəsində $R2$ qiymətinə əsasən deyə bilərik ki, əldə olunan reqressiya xətti verilənlərin 90 %-ni düzgün təyin edir (listinq 9).

Metrikalar	**Qiymət**
MAE	1.46
MSE	3.17
RMSE	1.78
R2	0.90

Cədvəl 3.1: *Advertising* test bazasının proqnozlaşdırılmasının qiymətləndirilməsinin nəticələri

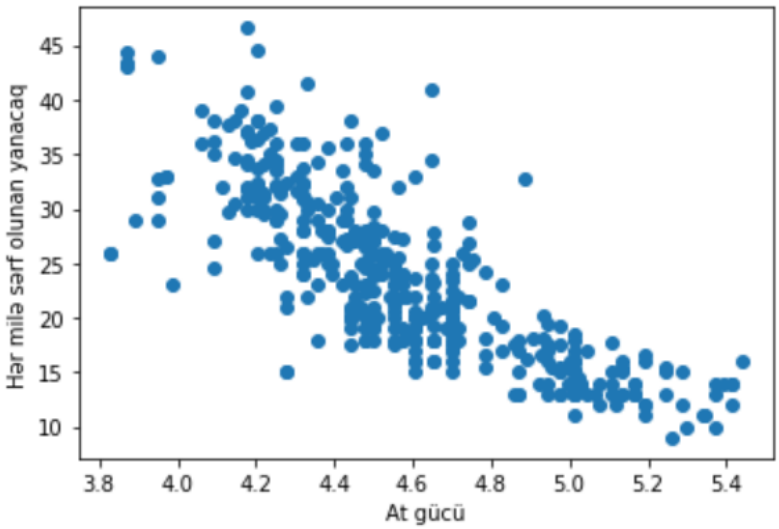
listinq 9: *LinearRegression* funksiyası ilə verilənlərin proqnozlaşdırılması

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
data = pd.read_csv("Advertising.csv")
X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.2, random_state = 42)
knn = KNeighborsRegressor(n_neighbors = 3)
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
lin_model = LinearRegression()
lin_model.fit(X_train, y_train)
y_pred = lin_model.predict(X_test)
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
print('R2 = ', r2)
print('MSE = ', mse)
print('MAE = ', mae)
print('RMSE = ', np.sqrt(mse))
print(lin_model.intercept_)
print(lin_model.coef_)
```

3.3 Stoxastik ən tez enmə üsulu

Maşın öyrənməsində ən çox istifadə olunan ən tez enmə üsulunun mənfi tərəfi ondan ibarətdir ki, alqoritmin hər bir iterasiyasında çoxlu sayda hesablamalar etmək lazım gəlir. Məsələn, əgər öyrədici bazada hər biri 10 əlamətlə təyin olunan 10000 nümunə varsa, onda düzgün qiymətlə hesablanan qiymətin fərqlərinin cəmi 10000 nümunə üçün hesablanacaq, eyni zamanda bu funksiyanın qradiyenti hər bir əlamət üçün hesablanmalıdır, yəni $10 \times 10000 = 100000$ sayda hesablama əməliyyatı yerinə yetirilməlidir. Adətən öyrənmə zamanı ən azı 1000 iterasiyadan istifadə olunduğunu nəzərə alsaq, alqoritmin yerinə yetirilməsi üçün 10^9 sayda əməliyyatın hesablanmasına ehtiyac var. Bu, kifayət qədər böyük hesablamalar tələb edir, buna görə də ən tez enmə üsulu böyük verilənlərin öyrədilməsi zamanı ləng işləyir. Bu halda çıxış yolu *stoxastik* qradiyent üsulunun tətbiqidir. Burada stoxastik *təsadüfi* mənada istifadə olunur.

Stoxastik ən tez enmə üsulunda (Stochastic Gradient Descent - SGD) böyük həcmli hesablamaların azaldılması üçün təsadüfi olaraq hər iterasiyada bir nümunə seçilir və törəmələr hesablanır. Bir çox hallarda yalnız bir nümunə əvəzinə hər bir iterasiyada az sayda nümunələr çoxluğu seçilir, bu zaman üsul kiçik dəstə əsaslı



Şəkil 3.3: Maşınların parametrlərinin fəzada paylanması

ən tez enmə üsulu (mini - batch gradient descent) adlanır. Bu üsul ən tez enmə üsulunun performansını və stoxastik ən tez enmə üsulunun sürətini özündə birləşdirir. Stoxastik ən tez enmə üsulunu klassifikasiya və proqnozlaşdırma məsələlərinin həllinə tətbiqi üçün Python *sklearn* kitabxanasından klassifikasiya məsələləri üçün **SGDClassifier**, reqressiya məsələləri üçün **SGDRegressor** funksiyalarından istifadə etmək olar.

3.4 Qeyri-xətti reqressiya

Bir çox hallarda parametrlər və asılı dəyişən arasındakı əlaqənin xətti funksiya ilə deyil, qeyri-xətti funksiyalarla təyin olunması nəticəsində daha dəqiq nəticə əldə etmək olur. Bu zaman asılı dəyişən parametrlərin xətti kombinasiyası şəklində deyil, onlardan kvadratik, eksponensial, triqonometrik, və s. şəkildə asılı olan funksiyalarla təyin olunur.

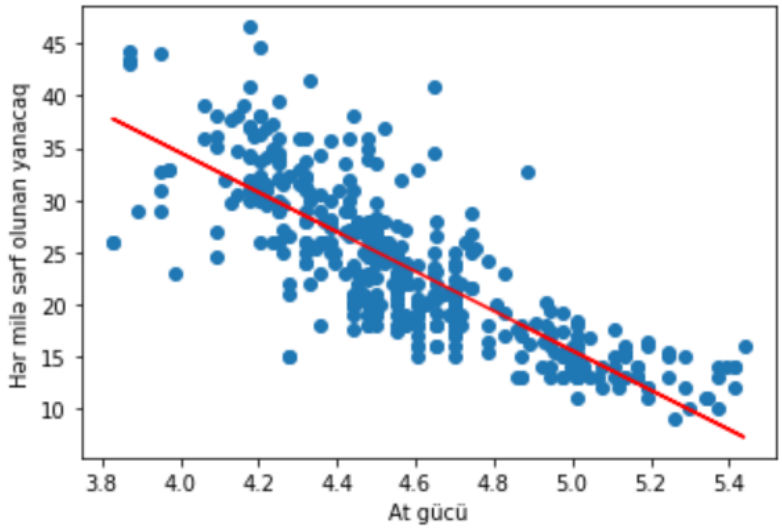
Nümunə kimi maşınlar haqqında verilənlər çoxluğunu nəzərdən keçirək, burada müxtəlif maşınların parametrləri haqqında məlumatlar toplanmışdır [17]. Qeyri-xətti reqressiyanı tətbiq etmək üçün maşının at gücü (*horsepower*) və hər mil üçün yandırdığı yanacağın miqdarı (*miles per gallon*) parametrlərinin ikiölçülü fəzada paylanmasına baxaq (şəkil 3.3).

İlk öncə verilənləri xətti reqressiya modeli ilə öyrədək (şəkil 3.4), bu zaman modelin qiymətləndirilməsinin nəticələri cədvəl 3.2 - də verilmişdir.

Verilənlərin paylanmasından da görünür ki, asılı dəyişənlə parametr arasındakı asılılıq xətti deyil. Asılı dəyişənin (*hər milə sərf olunan yanacaq*) parametrdən (*at gücü*) kvadratik asılılığına baxaq. Bu zaman parametri aşağıdakı şəkildə götürək:

Metrikalar	Qiymət
MAE	3.45
MSE	18.80
RMSE	4.33
R2	0.67

Cədvəl 3.2: Maşın verilənlərinin xətti reqressiya modelinin qiymətləndirilməsi



Şəkil 3.4: Verilənlərin xətti funksiya ilə approksimasiyası

```
X = data['horsepower']*data['horsepower']
```

Kvadratik modelin performansının qiymətləndirilməsi üçün alınan qiymətlər xətti modeldən daha zəifdir (Cədvəl 3.3).

Metrikalar	Qiymət
MAE	3.98
MSE	25.00
RMSE	5.00
R2	0.56

Cədvəl 3.3: Maşın verilənlərinin kvadratik funksiya ilə approksimasiyası modelinin qiymətləndirilməsi

Verilənləri üstlü funksiya ilə öyrədək:

```
X = data['horsepower']**3
```

Bu zaman modelin performansının nəticələri cədvəl 3.4 - də təsvir olunmuşdur.

Həmçinin alınan nəticələri daha da yaxşılaşdırmaq üçün verilənlərin müxtəlif dərəcəli

Metrikalar	**Qiymət**
MAE	3.08
MSE	14.52
RMSE	3.81
R2	0.74

Cədvəl 3.4: Maşın verilənlərinin üstlü funksiya ilə approksimasiyası modelinin qiymətləndirilməsi

polinom funksiyalarla öyrədilməsinə baxaq. Bu zaman *R2* metrikası üçün aşağıdakı nəticələr alınır (cədvəl 3.5).

Polinom dərəcələri	**R2**
2	0.671
3	0.671
4	0.672
5	0.677
6	0.680

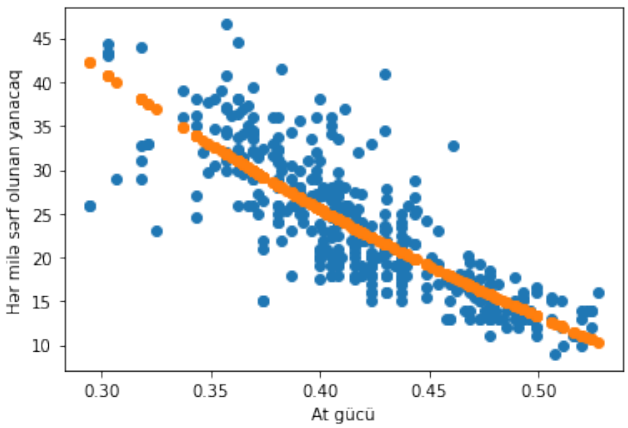
Cədvəl 3.5: Maşın verilənlərinin müxtəlif dərəcəli polinomlar ilə approksimasiya modellərinin qiymətləndirilməsi

Baxılan nümunədə verilənlərə tətbiq olunan müxtəlif xarakterli qeyri-xətti funksiyalarla öyrətmə zamanı ən yaxşı performans $y = \beta_0 + \beta_1 lnX$ funksiyası üçün alınır (cədvəl 3.6). Cədvəldən göründüyü kimi bu halda xətanın qiyməti minimum, *R2* əmsalının qiyməti isə 1-ə ən yaxındır. Alınan funskiyanın qrafiki şəkil 3.5 - də təsvir olunmuşdur.

Modelin adı	*MAE*	*MSE*	*RMSE*	*R2*
Xətti funksiya	3.448	18.804	4.336	0.669
Kvadratik funksiya	3.981	24.998	5.000	0.560
Loqarifmik funksiya	**3.028**	**14.063**	**3.750**	**0.752**
2 dərəcəli polinom	3.033	20.077	4.481	0.671
3 dərəcəli polinom	3.303	20.058.	4.4792	0.671
4 dərəcəli polinom	3.304	20.014	4.474	0.672
5 dərəcəli polinom	3.273	19.682	4.436	0.677
6 dərəcəli polinom	3.266	19.498	4.416	0.680

Cədvəl 3.6: Asılı dəyişənin parametrdən asılılığı üçün müxtəlif reqressiya modellərinin tətbiqi ilə alınan nəticələr

Qeyd edək ki, qeyri-xətti reqressiya zamanı asılı dəyişənlə parametrin asılılığını ən yüksək dəqiqliklə approksimasiya edən funksiyanın tapılması üçün optimal üsul mövcud deyil, bu məsələ eksperimental şəkildə həll olunur.



Şəkil 3.5: Verilənlərin loqarifmik funksiya ilə approksimasiyası

3.5 Logistik reqressiya

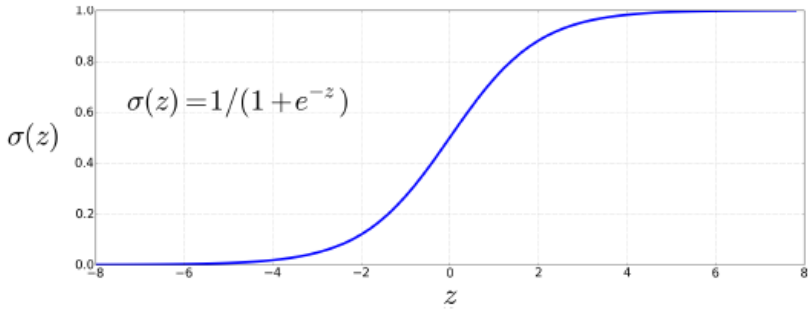
Logistik reqressiya verilənlərin klassifikasiyası üçün istifadə olunan üsuldur, Naïve Bayes üsulundan əsas fərqi onun fərqləndirici (discriminative) maşın öyrətmə üsulu olmasıdır, belə ki Naïve Bayes üsulu yaradıcı (generative) üsuldur [18]. Fərqi anlamaq üçün belə bir təsəvvürü nəzərdən keçirək. Fərz edək ki, it şəkillərini pişik şəkillərindən ayıran model qurmaq istəyirik. Bu zaman yaradıcı üsulun əsas məqsə-di pişiklərin və itlərin hansı şəkildə olmasını anlamaq olacaq. Daha dəqiq desək, bu model iti və ya pişiyi "yaratmağa", yəni şəklini çəkməyə çalışır. Qərar qəbuletmə zamanı isə verilən test şəklinin hansı sinfin şəklinə daha yaxın olduğunu müəyyən edir və çıxış olaraq həmin sinfi göstərir. Fərqləndirici model isə sinifləri öyrənmədən onları fərqləndirməyə çalışır. Məsələn, əgər öyrədici şəkillərin içərisində bütün itlərdə xalta varsa və pişiklərdə yoxdursa, bu əlamət belə iki sinfi bir-birindən ayırmağa kifayət edir. Əgər bu modellərdən siniflər haqqında nə bildiyini soruşsanız, onlar yalnızca pişiklərin xalta geyinmədiyini deyəcəklər.

Formal olaraq fərqi göstərmək üçün Naïve Bayes üsulunu yenidən nəzərdən keçirək. Bu üsul d verilənin c sinfinə aid olmasını birbaşa $P(c|d)$ ehtimalını hesablamaqla deyil, ilkin ehtimalı və uyğunluq ehtimalı vasitəsilə təyin edir:

$$\hat{c} = \arg\max_{c \in C} P(d|c)P(c).$$

Naïve Bayes üsulu kimi yaradıcı üsullar uyğunluq ehtimalı anlayışından istifadə edərək bu verilənin c sinfindən olduğunu bildiyimiz halda verilənin əlamətlərini müəyyən edir. Bundan fərqli olaraq fərqləndirici üsul $P(c|d)$ ehtimalını birbaşa hesablayır. Bu üsullar siniflərə uyğun nümunələr yaratmağa qadir olmasalar da, sinifləri bir-birindən ayırmağa kömək edən əlamətlərə daha çox çəki verməyi öyrənir.

Logistik reqressiya müəllimli maşın öyrənmədən istifadə edən ehtimal əsaslı klassifikatordur. Bu klassifikatorun aşağıdakı elementləri mövcuddur:



Şəkil 3.6: Siqmoidal funksiya

- Giriş verilənlərinin əlamət təsviri, hər bir $x^{(i)}$ veriləni üçün əlamətlər vektoru $[x_1, x_2, \ldots, x_m]$;
- $P(y|x)$ ehtimalı vasitəsilə proqnozlaşdırılan sinfi hesablayan $\hat{y}$ klassifikasiya funksiyası (məs. sigmoidal);
- Öyrətmə üçün öyrədici nümunələrdə xətanı minimallaşdıran funksiya, çarpaz-entropiya xəta funksiyası;
- Məqsəd funksiyasını minimallaşdıran optimallaşdırma üsulu (məs. stoxastik ən tez enmə üsulu).

Logistik reqressiya da iki mərhələdən ibarətdir:

1. Öyrətmə - çarpaz-entropiya xəta funksiyası və stoxastik ən tez enmə üsulu vasitəsilə sistemin (w və b çəkilərinin) öyrədilməsi;
2. Test – test nümunə üçün $P(y|x)$ ehtimalının hesablanması və $y = 0$ və ya $y = 1$ siniflərindən ən yüksək ehtimala malik olanının seçilməsi.

3.5.1 Siqmoidal funksiya

Binar logistik reqressiyanın əsas məqsədi yeni giriş verilənlərinin daxil olduğu sinif haqqında binar qərar qəbul edə bilən klassifikatoru öyrətməkdir. Bu qərarı qəbul edə bilən funksiyalardan biri siqmoidal funksiyadır.

Sadə reqressiya tənliyində olduğu kimi burada da nəticənin proqnozlaşdırılması üçün çəki vektoru ilə əlamətlər vektorunun skalyar hasili ilə b əmsalının cəmi hesablanır:

$$z = w \cdot x + b$$

Göründüyü kimi z $(-\infty; +\infty)$ aralığında istənilən qiyməti ala bilər, qərar isə ehtimal olduğundan z $[0; 1]$ aralığında qiymətlər almalıdır. Bunun üçün siqmoidal funksiyanın qiymətindən istifadə edəcəyik, bu funksiya logistik funksiya da adlanır (şəkil 3.6):

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.6}$$

Verilən x nümunəsinin hansı sinfə aid olmasına necə qərar veririk?

Əgər $P(y = 1|x)$ ehtimalı 0.5-dən böyük olarsa x "1" sinfinə, əks halda "0" sinfinə aid edilir, burada 0.5 qərar hüdudu adlanır:

$$f(x) = \begin{cases} 1, & P(y = 1|x) > 0.5 \quad olduqda, \\ 0, & P(y = 1|x) \leq 0.5 \quad olduqda. \end{cases} \tag{3.7}$$

Öyrətmə üçün elə bir funksiyaya ehtiyacımız var ki, x nümunəsi üçün proqnozlaşdırılan nəticənin real nəticəyə (y – 0 və ya 1 qiymətini alır) nə qədər yaxın olduğunu göstərsin. Logistik reqressiyada öyrətmə məqsədilə çarpaz entropiya xəta funksiyasından istifadə olunur:

$$L_{CE}(\hat{y}, y) = -logP(y|x) = -[ylog\hat{y} + (1 - y)log(1 - \hat{y})]. \tag{3.8}$$

Göründüyü kimi bu funksiyanın qiyməti model düzgün proqnozlaşdırıldıqda azalır, səhv proqnoz nəticəsində isə artır. Daha sonra isə ən tez enmə üsulunun tətbiqi ilə lazım olan parametrlər təyin olunur.

listinq 8: *LogisticRegression* funksiyası ilə verilənlərin klassifikasiyası

```
1  import pandas as pd
2  df = pd.read_csv('heart.csv')
3  from sklearn.linear_model import LogisticRegression
4  X = df.drop(['target'], axis = 1)
5  y = df['target']
6  from sklearn.model_selection import train_test_split
7  x_train, x_test, y_train, y_test = train_test_split(X, y,
       test_size=0.2, random_state=0)
8  model = LogisticRegression()
9  model.fit(x_train, y_train)
10 score = model.score(x_train, y_train)
11 print('Accuracy in train',score)
12 score = model.score(x_test, y_test)
13 print('Accuracy in test',score)
14 y_pred = model.predict(x_test)
15 from sklearn import metrics
16 cm = metrics.confusion_matrix(y_test, y_pred)
17 print(cm)
18 from sklearn.metrics import classification_report,
       accuracy_score
19 print(classification_report(y_test, y_pred))
```

3.5.2 Verilənlərin logistik reqressiya üsulu ilə klassifikasiyası

İndi isə Python *sklearn.linear_ model* kitabxanasının *LogisticRegression* funksiyasından istifadə edərək klasifikasiya məsələsini həll edək və üsulun dəqiqliyini müəyyən

	precision	recall	f1-score	support
0	0.85	0.81	0.83	27
1	0.86	0.88	0.87	34
accuracy			0.85	61
macro avg	0.85	0.85	0.85	61
weighted avg	0.85	0.85	0.85	61

Şəkil 3.7: Verilənlərin logistik reqressiya üsulu ilə klassifikasiyasının qiymətləndirilməsi

edək. Bu məqsədlə yenidən ürək xəstəliyi risk qrupunda olub-olmaması haqqındakı verilənlərdən istifadə edək və nəticələri digər modellərlə müqayisə edək (listinq 8). Öyrətmənin və test verilənlərinin klasifikasiyasının nəticəsində aşağıdakı qiymətləndirməni əldə edirik (Şəkil 3.7). Şəkildən göründüyü kimi modelin dəqiqliyi 85 %-dir. Bu isə öncə kitabda tətbiq olunan digər üsullardan (Naive Bayes, kNN) daha dəqiq nəticədir.

Heart.csv verilənlər çoxluğuna logistik reqressiya üsulunu tətbiq etdikdən sonra proqnozların paylanmasını göstərən qarışıqlıq matrisi cədvəl 3.7 - də göstərilmişdir.

		Proqnozlaşdırılan	
		y = 0	y = 1
Real	y = 0	22	5
	y = 1	4	30

Cədvəl 3.7: *heart.csv* verilənlər bazasının logistik reqressiya üsulu ilə klassifikasiyası nəticəsində alınan qarışıqlıq matrisi

3.5.3 Çox sinifli logistik reqressiya

Tutaq ki, verilənlər çoxluğunda 2 deyil daha çox sayda sinfə aid olan obyektlərin öyrədici verilənləri verilmişdir. Bu zaman aid olduğu sinfin proqnozlaşdırılması lazım olan nümunənin k sinfinə aid olması ehtimalının $P(y_k = 1|x)$ hesablanması üçün sigmoidal funksiyanın genişləndirilmiş forması olan softmax funksiyasından istifadə edilir. Softmax funksiyası k sayda qiymətlər vektorunun elementlərini $z = [z_1, z_2, ..., z_k]$, $(0, 1)$ intervalında qiymətlər alan və cəmi 1-ə bərabər olan ehtimal qiymətlərinə çevirir:

$$softmax(z_i) = \left[\frac{e^{z_1}}{\sum_{i=1}^{k} e^{z_i}}, \frac{e^{z_2}}{\sum_{i=1}^{k} e^{z_i}}, ..., \frac{e^{z_k}}{\sum_{i=1}^{k} e^{z_1}} \right].$$

Siqmoidal funksiyada olduğu kimi softmax funksiyasının da nəticələri binar şəklə çevrilir, belə ki, əgər elementlərdən biri digərlərin hamısından böyükdürsə o 1-ə, qalanları isə 0-a bərabər qəbul edilir.

Çox sinifli logistik reqressiya modelinin digər üsullara nisbətən performansını qiymətləndirmək üçün müxtəlif şüşə tiplərinin fərqləndirilməsi üçün toplanan verilənlər çoxluğunu emal edək [19]. Burada parametrlər olaraq şüşənin qırılma indeksi, tərkibində olan elementlərin çəki faizi, asılı dəyişən olaraq isə şüşənin tipi götürülür ki, bu parametr 1-dən 7-ə qədər şüşə tiplərini bildirən tam ədədlərin içərisindən qiymətlər alır, məsələn, 1 − ci sinif ev pəncərələrinin hazırlanması, 5 − ci sinif isə konteynerlərin hazırlanması üçün yararlı olan şüşə tipini bildirir. Adıçəkilən verilənləri klassifikasiya etmək üçün kNN, Naive-Bayes və logistik reqressiya üsullarını tətbiq edək və nəticələri müqayisə edək (cədvəl 3.8).

Üsulun adı	*Düzgün klassifikasiya olunan elementlərin sayı*	*Düzgün klassifikasiya olunmayan elementlərin sayı*	*Üsulun dəqiqliyi*
Naive - Bayes	18	47	0.277
kNN (k = 3)	44	21	0.677
Logistik reqressiya	46	19	0.708

Cədvəl 3.8: Müxtəlif üsullarla çoxsinifli klassifikasiya üsullarının performans göstəriciləri

Çalışmalar

1. (1, 3), (2, 5), (3, 7), (4, 8), (5, 10) nöqtələrindən ibarət verilənlər çoxluğu verilmişdir. Ən kiçik kvadratlar üsulu ilə bu nöqtələri approksimasiya edən düz xəttin parametrlərini təyin edin.

2. (1, 5), (2, 10), (3, 18), (4, 28), (5, 40) nöqtələrini 2-ci dərəcəli polinom ilə approksimasiya etmək üçün ən kiçik kvadratlar üsulunu tətbiq edin. Kvadratik funksiyanın əmsallarını hesablayın.

3. (1, 2.1), (2, 3.8), (3, 6.7), (4, 9.6), (5, 14.2) nöqtələrini qeyri-xətti $y = ae^{bx}$ funksiyası ilə approksimasiya etmək üçün ən kiçik kvadratlar üsulunu tətbiq edin. a və b əmsallarını təyin edin.

4. $f(x) = x^2 + 5x + 6$ kvadratik funksiyasının minimumunu ixtiyari başlanğıc qiymətlə öyrənmə addımı 0.1 olduqda ən tez enmə üsulu ilə hesablayın.

5. $f(x) = x^4 - 3x^3 + 2$ funksiyasının minimumunu $x = 2$ başlanğıc qiymətlə öyrənmə addımı 0.1, 0.3, 0.01 olduqda ən tez enmə üsulu ilə hesablayın.

6. *Python numpy* vasitəsilə [-10; 10] intervalında 1 vahid addım ilə sigmoid funksiyasının qiymətlər ardıcıllığını çıxarın.

7. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ boston* funksiyasından istifadə edərək Bostonda evlərin qiymətinin proqnozlaşdırılması üçün

xətti reqressiya üsulunu tətbiq edin. Real qiymətlərlə qarşılaşdırıb orta kvadratik xətanın qiymətini hesablayın.

8. *Python* mühitində *sklearn.datasets* kitabxanasının *load_iris* funksiyasından istifadə edərək süsən güllərinin klassifikasiyası üçün logistik reqressiya üsulunu tətbiq edin. Real qiymətlərlə qarşılaşdırıb modelin dəqiqliyini hesablayın.

Fəsil 4

Süni neyron şəbəkələr və dərin öyrənmə

Dərin öyrənmə, müxtəlif qeyri-xətti çevrilmələri birləşdirən mürəkkəb arxitektur ilə məlumatları modelləşdirməyə çalışan öyrənmə metodları toplusudur. Dərin öyrənmənin elementar kərpicləri dərin neyron şəbəkələri yaratmaq üçün birləşdirilən neyron şəbəkələrdir. Bu üsullar üz tanıma, nitqin tanınması, kompüter görməsi, avtomatlaşdırılmış dil emalı, mətnlərin təsnifatı (məsələn, spamların aşkarlanması) daxil olmaqla, səs və təsvirin emalı sahələrində əhəmiyyətli irəliləyişlərə imkan yaratmışdır. Dərin öyrənmənin potensial tətbiqləri çox saydadır [20]. Dərin öyrənmə üsulu ilə *Go* oyunu oynamağı öyrənən və 2016-cı ildə dünya çempionunu məğlub edən *AlphaGo* proqramı möhtəşəm bir nümunədir.
Süni neyron şəbəkələrin bir neçə tipi mövcuddur:

- Çoxlaylı perseptronlar, ən köhnə və ən sadə nümunələrdir;
- Qıvrılmış neyron şəbəkələr (Convolutional Neural Networks - CNN), xüsusilə təsvirlərin emalı üçün uyğunlaşdırılmışdır;
- Mətn və ya zaman seriyası kimi ardıcıl məlumatlar üçün istifadə olunan rekurrent neyron şəbəkələr (Recurrent Neural Networks - RNN).

4.1 Neyron şəbəkələr

Süni neyron şəbəkə θ parametrlərinə görə xətti olmayan, x girişini $y = f(x, \theta)$ çıxışı ilə əlaqələndirən proqramdır. Sadəlik üçün y çıxış verilənlərinin birölçülü olduğunu fərz edirik, lakin çoxölçülü də ola bilər. Neyron şəbəkələr həm reqressiya, həm də klassifikasiya üçün istifadə edilə bilər. Statistik öyrənmə üsullarına xas olaraq θ parametrləri öyrənmə nümunələrindən tapılır. Lokal minimumlar mövcud olduğuna görə minimallaşdırılan funksiya qabarıq deyil.

Cybenko (1989) və Hornik (1991) tərəfindən isbat olunan universal approksimasiya teoremi süni neyron şəbəkələrin öyrənmə üsulunun uğurunu xeyli artırmışdır [21]. Bundan əlavə Le Cun (1986) qradiyentin geri yayılması adlanan, asanlıqla kvadratik meyarın lokal minimumunu tapmağa imkan verən neyron şəbəkənin qradiyentinin hesablanması üçün effektiv üsul təklif etmişdir [22].

4.1.1 Süni neyron

Süni neyron rabitə çəkiləri vektoru $w_j = (w_{j,1}, w_{j,2}, ..., w_{j,d})$ ilə çəkiləndirilən, neyron meyli b_j ilə tamamlanan, aktivləşdirmə funksiyası ϕ ilə əlaqələndirilən giriş vektorunun $x = (x_1, x_2, ..., x_d)$ funksiyasından f_i ibarətdir:

$$y_j = f_j(x) = \phi((w_j, x) + b_j). \tag{4.1}$$

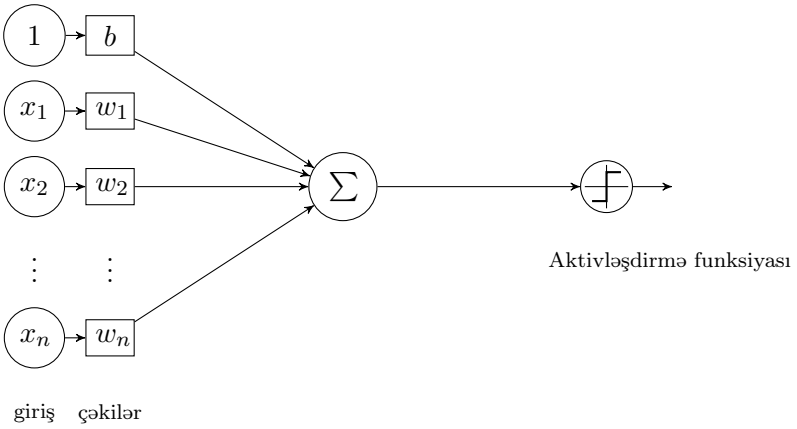
Süni neyron şəbəkələrin düyünlərində müxtəlif aktivləşdirmə funksiyalarından istifadə olunur:

- Eynilik funksiyası:
$$\phi(x) = x;$$
- Siqmoidal funksiya (və ya logistik):
$$\phi(x) = \frac{1}{1 + exp(-x)};$$
- Hiperbolik tangens funksiyası:
$$\phi(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} = \frac{exp(2x) - 1}{exp(2x) + 1};$$
- Sərt hədd funksiyası
$$\phi_\beta(x) = 1_{x \geq \beta};$$
- Düzləşdirilmiş xətti vahid (Rectified Linear Unit - ReLU) funksiyası:
$$\phi(x) = max(0, x).$$

Süni neyronun sxematik təsviri şəkil 4.1 - də təsvir olunmuşdur. Bu şəkildə verilmiş neyron şəbəkə 1 laydan və bu layda mövcud olan 1 neyrondan ibarətdir. Göründüyü kimi baxılan perseptronun n sayda giriş veriləni və 1 çıxış veriləni var. Gizli layın düyünündə giriş verilənlərinin və layın uyğun çəkilərinin hasillərinin cəmi hesablanır:

$$z = \sum_{i=1}^{n} x_i w_i + b,$$

çıxış layında isə z qiyməti aktivləşdirmə funksiyasına verilir və çıxış təyin olunur $y = \phi(z)$.



Şəkil 4.1: Süni neyronun sxematik təsviri

x_1	x_2	t
0	0	-1
0	1	-1
1	0	-1
1	1	$+1$

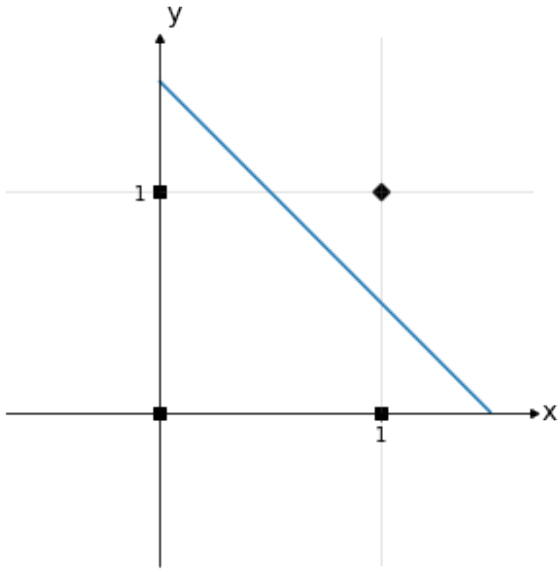
Cədvəl 4.1: AND funksiyasının qiymətlər cədvəli

Perspetronların tətbiqinə ən populyar nümunələrdən biri *bul* funksiyalarıdır. Bul funksiyaları N sayda giriş verilənlərinə vahid çıxış uyğunlaşdıran funksiyalardır. İki dəyişənli AND bul funksiyasının qiymətlər cədvəli cədvəl 4.1 - də verilmişdir. AND funksiyasının qrafik təsviri şəkil 4.2 - də verilmişdir (burada ■$-t = -1$, ♦$-t = +1$ hallarına uyğundur). Həmçinin verilənləri klassifikasiya etmək üçün ayırıcı düz xətt də təsvir olunmuşdur. Qeyd etmək lazımdır ki, ayırıcı düz xətt yeganə deyil, həmçinin bu düz xətti təyin edən w vektoru da sonsuz saydadır.

x_1	x_2	t
0	0	-1
0	1	$+1$
1	0	$+1$
1	1	-1

Cədvəl 4.2: XOR funksiyasının qiymətlər cədvəli

Cədvəl 4.2 - də XOR bul funksiyasının qiymətlər cədvəli Şəkil 4.3 - də isə bu qiymətlərin qrafik təsviri verilmişdir, göründüyü kimi burada $t = -1$ və $t = +1$ siniflərini düz xətt ilə ayırmaq mümkün deyil. Bu funksiyanın qiymətlər cədvəli şəklində paylanmış verilənləri klassifikasiya etmək üçün ən azı 2 düyündən ibarət neyron şəbəkə lazımdır. Beləliklə, xətti ayrıla bilən verilənləri ayırmaq üçün bir düyündən ibarət neyron şəbəkə - perseptron kifayət etdiyi halda xətti ayrılması



Şəkil 4.2: AND bul funksiyasının qrafik təsviri

mümkün olmayan verilənləri klassifikasiya etmək üçün çoxlaylı neyron şəbəkələrdən istifadə etmək daha məqsədəuyğundur.

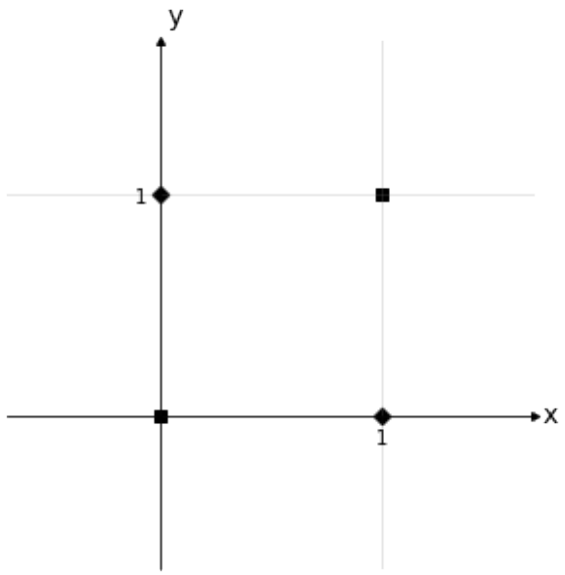
4.1.2 Çoxlaylı perseptron

Çox laylı perseptron (və ya neyron şəbəkə) bir neçə gizli neyron laylarından ibarət strukturdur, burada bir layın neyronunun çıxışı növbəti layın neyronunun girişinə çevrilir. Üstəlik, bir neyronun çıxışı eyni layın neyronunun və ya əvvəlki layların neyronunun girişi də ola bilər (rekursiv neyron şəbəkələr). Çıxış layı adlanan sonuncu layda, həll olunan problemlərin növündən asılı olaraq (reqressiya və ya klassifikasiya) gizli laylar üçün fərqli aktivləşdirmə funksiyaları tətbiq edilə bilər. Şəkil 4.4 - də n_0 sayda giriş dəyişənləri, bir çıxış dəyişəni və 2 gizli laydan (birinci gizli layda n_1, ikinci gizli layda n_2 düyün) ibarət neyron şəbəkə təsvir olunmuşdur.

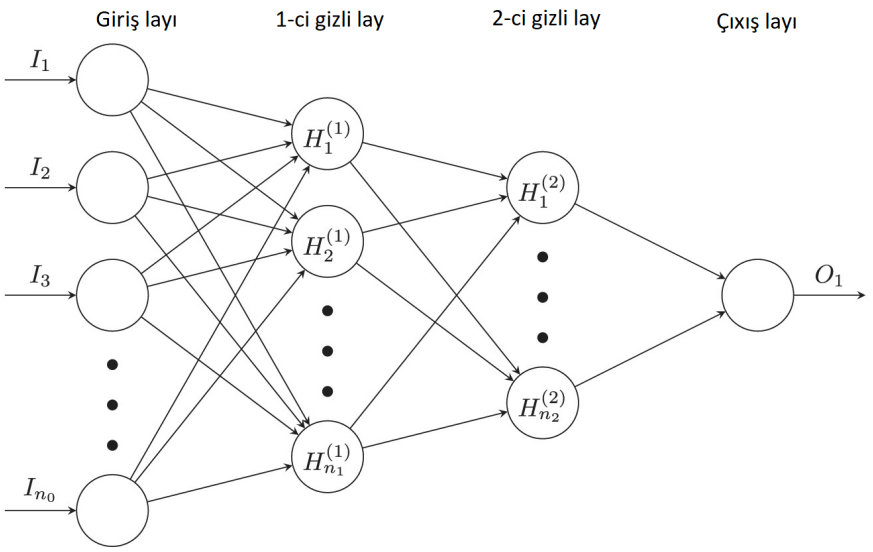
4.1.3 Xətanın geri yayılması üsulu

Xətanın geri yayılması üsulu ən tez enmə üsulu ilə süni neyron şəbəkələrin müəllimli öyrənmə üsuludur. Bu üsul vasitəsilə neyron şəbəkə və xəta funksiyası verildikdə neyron şəbəkənin çəkilərinə görə xəta funksiyasının qradiyenti hesablanır. Üsulun adında *geri yayılma* ifadəsinin olmasının səbəbi qradiyentin hesablanması zamanı prosesin geriyə istiqamətlənməsidir. Belə ki, son layın qradiyenti ilk növbədə, birinci layın qradiyenti isə ən sonda hesablanır.

Neyron şəbəkələrin xətanın geri yayılması üsulu ilə öyrədilməsi üçün üç kəmiyyətə ehtiyac vardır:



Şəkil 4.3: XOR bul funksiyasının qrafik təsviri



Şəkil 4.4: Çoxlaylı neyron şəbəkənin təsviri

- Giriş - çıxış cütlərindən $(\vec{x_i}; \vec{y_i})$ ibarət verilənlər çoxluğu, burada $\vec{x_i}$ - giriş veriləni, $\vec{y_i}$ bu giriş veriləni üçün düzgün çıxış qiymətidir. N ölçülü giriş - çıxış cütləri çoxluğu $X = \{(\vec{x_1}; \vec{y_1}), ..., (\vec{x_N}; \vec{y_N})\}$ şəklində təsvir olunur;
- İrəliyə istiqamətlənmiş neyron şəbəkə, bu şəbəkənin parametrlər çoxluğu ümumi şəkildə θ ilə ifadə olunur. Bu parametrlər l_k layının $j-$ ci düyünü ilə l_{k-1} layının $i-$ci düyünü arasındakı tilin çəkisi w_{ij}^k və l_k layının $i-$ci düyününün meylindən b_i^k ibarətdir;
- Xəta funksiyası - θ parametrinin verilmiş qiymətində giriş - çıxış verilənlər çoxluğundan, $(\vec{x_i}; \vec{y_i}) \in X$, verilmiş $\vec{x_i}$ girişinə uyğun düzgün çıxış ilə $\vec{y_i}$ hesablanan $\hat{\vec{y_i}}$ çıxış arasındakı xətanı təyin edir $-E(X, \theta)$.

Ən tez enmə üsulu ilə neyron şəbəkənin öyrədilməsi xəta funksiyasının $E(X, \theta)$ şəbəkənin laylarının çəkilərinə w_{ij}^k və meyllərinə b_i^k görə qradiyentinin hesablanmasını tələb edir. Daha sonra öyrənmə addımına α, əsasən hər bir iterasiyada çəkilərin və meyllərin (toplu şəkildə θ kimi ifadə olunur) yeni qiymətləri hesablanır:

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta}, \tag{4.2}$$

burada θ^t ən tez enmə üsulunun $t-$ ci iterasiyasında parametrlərinin qiymətidir. Çoxlaylı irəliyə istiqamətlənmiş neyron şəbəkənin öyrədilməsində əsas məqsəd giriş verilənlərinə görə çıxış verilənlərini düzgün qiymətlərə ən yaxın hesablaya bilən gizli layların parametrlərinin (çəkilərin və meyllərin) tapılmasıdır. Xətanın geri yayılması üsulunun aşağıda verilmiş düsturları bir çıxışı olan neyron şəbəkə üçündür, amma zəncirvari qaydanı və hasil qaydasını tətbiq etməklə çox çıxışlı neyron şəbəkəyə uyğunlaşdırıla bilər. Beləliklə, bundan sonra giriş - çıxış verilənləri $(\vec{x}, y)$ şəklində ifadə edək, burada, y - çıxış veriləni vektor deyil. Bu öyrətmə alqoritmində istifadə olunan aşağıdakı kəmiyyətlərə nəzər yetirək:

- w_{ij}^k: l_{k-1} layının $i-$ci düyünündən l_k layının $j-$ci düyününə gələn tilin çəkisi;
- b_i^k: l_k layının $i-$ci düyününün meyli;
- a_i^k: l_k layının $i-$ci düyününə gələn tillərin çəkili xətti kombinasiyası ilə meylin cəmi;
- o_i^k: l_k layının $i-$ci düyününün çıxışı;
- r_k: l_k layının düyünlərinin sayı;
- g: gizli layların düyünlərinin aktivləşdirmə funksiyaları;
- g_0: çıxış layının düyünlərinin aktivləşdirmə funksiyaları.

Klassik xətanın geri yayılması üsulunda xəta funksiyası xətanın kvadratik ortası kimi hesablanır:

$$E(X, \theta) = \frac{1}{2N} \sum_{i=1}^{n} (\hat{y_i} - y_i)^2, \tag{4.3}$$

burada y_i, $(\vec{x_i}, y_i)$ - giriş - çıxış verilənlər cütünün düzgün çıxış qiyməti, $\hat{y_i}$ isə şəbəkənin hesabladığı çıxış qiymətidir. Əlbəttə başqa xəta funksiyalarından da istifadə etmək olar, lakin xətanın orta kvadratik qiymətinin xətanın geri yayılması üsulu ilə tarixi bağlılığı və uyğun riyazi xarakteristikaları bu funksiyanı öyrətmə üçün çox əlverişli edir.

Qradiyentlərin hesablanması. Xətanın geri yayılması üsulunda qradiyentlərin hesablanması üçün differensial analizin hasil və zəncir qaydalarından istifadə etmək lazımdır. Bu qaydaların tətbiqi aktivləşdirmə funksiyalarının uyğunluğundan bilavasitə asılıdır, məhz buna görə də hamar olmayan və differensiallanmayan funksiyalardan istifadə olunmur.
Növbəti hissələrdə sadəlik üçün k layının $i-$ci düyününün meylini (b_i^k), w_{0i}^k kimi işarə edəcəyik, bu zaman $k-1$ layının 0 saylı düyünü üçün çıxış $o_0^{k-1} = 1$ kimi qəbul edilir. Beləliklə,

$$w_{0i}^k = b_i^k.$$

Bu ifadənin orijinal düstura ekvivalent olmasını göstərmək üçün aşağıdakı kimi yazaq:

$$a_i^k = b_i^k + \sum_{j=1}^{r_{k-1}} w_{ji}^k o_j^{k-1} = \sum_{j=0}^{r_{k-1}} w_{ji}^k o_j^{k-1}, \tag{4.4}$$

burada sol tərəf orijinal düstura sağ tərəf isə yeni ifadəyə uyğun gəlir. Yuxarıdakı ifadələri nəzərə alsaq, geri yayılma alqoritmi neyron şəbəkənin parametrlərinə əsasən (4.3) funksiyasının minimallaşdırılmasından ibarətdir. Bunun üçün neyron şəbəkənin hər bir w_{ij}^k çəkisi üçün $\frac{\partial E}{\partial w_{ij}^k}$ xüsusi törəmələrinin hesablanması lazımdır:

$$\frac{\partial E(X,\theta)}{\partial w_{ij}^k} = \frac{1}{N}\sum_{d=1}^{N} \frac{\partial}{\partial w_{ij}^k}\left(\frac{1}{2}(\hat{y_i} - y_i)^2\right) = \frac{1}{N}\sum_{d=1}^{N} \frac{\partial E_d}{\partial w_{ij}^k}.$$

Xəta funksiyasının xüsusi törəməsinə zəncir qaydasını tətbiq edək:

$$\frac{\partial E}{\partial w_{ij}^k} = \frac{\partial E}{\partial a_j^k}\frac{\partial a_j^k}{\partial w_{ij}^k},$$

bu hasilin birinci həddi xəta adlanır və aşağıdakı kimi ifadə olunur:

$$\delta_j^k = \frac{\partial E}{\partial a_j^k},$$

ikinci həddin hesablanması üçün (3.4) düsturundan istifadə edək:

$$\frac{\partial a_j^k}{\partial w_{ij}^k} = \frac{\partial}{\partial w_{ij}^k}\left(\sum_{j=0}^{r_{k-1}} w_{ij}^k o_j^{k-1}\right) = o_j^{k-1}.$$

Beləliklə, xəta funksiyasının E, w_{ij}^k çəkilərinə görə xüsusi törəmələri:

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k o_j^{k-1}$$

şəklində hesablanır.
Sonuncu laydan başlayaraq xətanın geri yayılması üsulu δ_1^m qiymətinin hesablanmasını yerinə yetirir, burada $m-$sonuncu layı ifadə edir. Göründüyü kimi burada aşağı indeks j ilə deyil 1 ilə ifadə olunmuşdur, çünki sonuncu layda sadəcə bir düyün var, yəni $j = 1$. Xəta funksiyasını E, hesablanmış çıxışı isə a_1^m ilə ifadə etsək, aşağıdakı düsturu alarıq:

$$E = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2}(g_0(a_1^m) - y)^2.$$

Xüsusi törəmələrdən və zəncir qaydasından istifadə etməklə

$$\delta_1^m = (g_0(a_1^m) - y)g_0'(a_1^m) = (\hat{y} - y)g_0'(a_1^m) \tag{4.5}$$

alınır. Bütün bunları bir yerə cəmləşdirsək, xəta funksiyasının E, sonuncu layın çəkilərinə görə w_{i1}^m xüsusi törəmələri aşağıdakı şəkildə olar:

$$\frac{\partial E}{\partial w_{i1}^m} = \delta_1^m o_i^{m-1} = (\hat{y} - y)g_0'(a_1^m)o_i^{m-1}.$$

Gizli laylar. Sonuncu laydan başqa digər laylarda xüsusi törəmələrin hesablanması üçün $1 \leq k < m$ layında xəta həddinin δ_j^k hesablanması üçün aşağıdakı düsturu nəzərdən keçirək:

$$\delta_j^k = \frac{\partial E}{\partial a_j^k} = \sum_{l=1}^{r^{k+1}} \frac{\partial E}{\partial a_l^{k+1}} \frac{\partial a_l^{k+1}}{\partial a_j^k},$$

burada l, 1-dən r^{k+1}-ə kimi dəyişir. Qeyd edək ki, w_{0j}^{k+1} çəkisinə uyğun gələn o_0^k meyli əvvəlki layların çıxışından asılı deyil və l, 0 qiymətini almır. Xəta həddini δ_l^{k+1} düsturda yerinə qoysaq, alarıq:

$$\delta_j^k = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} \frac{\partial a_l^{k+1}}{\partial a_j^k}. \tag{4.6}$$

a_l^{k+1} həddinin tərifini nəzərə alsaq,

$$a_l^{k+1} = \sum_{l=1}^{r^k} w_{jl}^{k+1} g(a_j^k)$$

ifadəsini alarıq, burada $g(x)$ gizli layların aktivləşdirmə funksiyalarıdır,

$$\frac{\partial a_l^{k+1}}{\partial a_j^k} = w_{jl}^{k+1} g'(a_j^k).$$

Alınan ifadələri (4.6) düsturunda yerinə yazsaq, gizli layların xəta həddi δ_j^k üçün əsas düsturu alarıq, bu düstur geri yayılma düsturu adlanır:

$$\delta_j^k = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1} g'(a_j^k) = g'(a_j^k) \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1}. \tag{4.7}$$

Beləliklə, $1 \leq k < m$ layında xəta funksiyasının E, w_{ij}^k çəkilərinə görə xüsusi törəmələri aşağıdakı kimi hesablanacaq:

$$\frac{\partial E}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1} = g'(a_j^k) o_i^{k-1} \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1}. \tag{4.8}$$

Xətanın geri yayılması üsulu adını əsasən (4.8) düsturundan götürmüşdür, belə ki, k layının δ_j^k xətası növbəti $k+1$ layının δ_j^{k+1} xətasından asılıdır. Xəta qiymətləri sonuncu laydan birinci laya qədər əks istiqamətdə hesablanır. Burada tələb olunan hesablanan çıxış $\hat{y} = g_0(a_1^m)$ ilə real çıxış y arasındakı xətanın aşkarlanmasıdır. Daha sonra birinci laya çatana qədər öncəki layın xəta qiymətləri növbəti layın xəta qiymətlərinin (w_{ij}^{k+1} çəkilərinə vurulmuş) hasil cəmi ilə, $g'(a_j^k)$ funksiyasının hasili kimi hesablanır.

Bu xətanın geri istiqamətdə hesablanması neyron şəbəkənin çıxışının irəli istiqamətdə hesablanmasına anolojidir, məhz buna görə də şəbəkənin çıxışının hesablanması mərhələsi **irəli mərhələ**, xəta hədlərinin və törəmələrin hesablanması isə **geri mərhələ** adlanır. İrəli istiqamətdə hesablama zamanı giriş verilənləri w_{ij}^k çəkiləri ilə əmsallaşdırılmış hasil cəmi kimi və qeyri-xətti $g(x)$ və $g_0(x)$ aktivləşdirmə funksiyaları ilə dəyişdirilərək birinci laydan sonuncu laya qədər təkrarlanaraq hesablanır. Geri mərhələdə isə girişlər sonuncu layın w_{ij}^{k+1} çəkiləri ilə əmsalların hasil cəmi və qeyri-xətti $g'(a_j^m)$ və $g_0'(a_j^k)$ aktivləşdirmə funksiyaları ilə hesablanan xəta hədləri hesablanır.

Bundan başqa, geri mərhələnin hesablanması a_j^k aktivləşdirmə qiymətlərindən, əvvəlki və növbəti layların düyünlərinin çıxışlarından o_j^k asılıdır, bütün bu qiymətlər geri mərhələnin başlamasından öncə təyin olunmalıdırlar. Beləliklə, irəli mərhələ ən tez enmə üsulunun hər iterasiyasında geri mərhələni qabaqlayır. İrəli mərhələdə a_j^k aktivləşdirmə qiymətləri və o_j^k çıxışları geri mərhələdə istifadə olun-maq üçün yadda saxlanılır. Geri mərhələ tamamlandıqda və xüsusi törəmələr məlum olduqda, çəkilərin (və meyllərin $b_i^k = w_{0i}^k$) yeni qiymətləri ən tez enmə üsulu ilə hesablanır. Bu proses lokal minimum tapılana və ya yığılma kriteriyalarından biri ödənilənə qədər davam edir.

4.1.4 Geri yayılma üsulunun alqoritmi

1. Hər bir giriş - çıxış cütü $(\vec{x_d}, y)$ üçün irəli mərhələni hesablayın və 0 layından başlayaraq sonuncu m layına qədər k layının j düyünü üçün $\hat{y_d}, a_j^k$ və o_j^k nəticələrini yadda saxlayın;

2. Hər bir giriş - çıxış cütü $(\vec{x_d}, y)$ üçün geri mərhələni hesablayın və m çıxış layından 0 giriş layına qədər hər bir $k-1$ layının $i-$ci düyününü k layının $j-$ci düyünü ilə birləşdirən tilin w_{ij}^k çəkisi üçün $\frac{\partial E_d}{\partial w_{ij}^k}$ nəticələrini yadda saxlayın:

   - Aşağıdakı düstur vasitəsilə sonuncu layın xəta qiymətini δ_1^m hesablayın:

$$\delta_1^m = g_0'(a_1^m)(\hat{y_d} - y);$$

- Sonuncu $k = m - 1$ gizli laydan başlayaraq geri qayıtmaqla iterativ şəkildə gizli layların xəta hədlərini aşağıdakı düsturla hesablayın:

$$\delta_j^k = g'(a_j^k) \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{jl}^{k+1};$$

- Hər bir verilənin E_d xətasının w_{ij}^k çəkisinə görə xüsusi törəmələrini

$$\frac{\partial E_d}{\partial w_{ij}^k} = \delta_j^k o_i^{k-1}$$

düsturu ilə hesablayın.

3. Hər bir giriş - çıxış cütü üçün fərdi qradiyentləri $\frac{\partial E_d}{\partial w_{ij}^k}$ qruplaşdıraraq bütün $X = \{(\vec{x_1}; y_1), ..., (\vec{x_N}; y_N)\}$ giriş - çıxış cütləri üçün bütöv qradiyenti aşağıdakı düsturla hesablayın:

$$\frac{\partial E(X,\theta)}{\partial w_{ij}^k} = \frac{1}{N} \sum_{d=1}^{N} \frac{\partial}{\partial w_{ij}^k} \left(\frac{1}{2} (\hat{y_d} - y_d)^2 \right) = \frac{1}{N} \sum_{d=1}^{N} \frac{\partial E_d}{\partial w_{ij}^k}.$$

4. Öyrətmə addımı α və bütöv qradiyentə $\frac{\partial E(X,\theta)}{\partial w_{ij}^k}$ əsasən

$$\Delta w_{ij}^k = -\alpha \frac{\partial E(X,\theta)}{\partial w_{ij}^k}$$

düsturu ilə parametlərin yeni qiymətlərini hesablayın.

Klassifikasiya üçün neyron şəbəkələrin öyrədilməsi. Yuxarıda təqdim olunan neyron şəbəkənin xətanın geri yayılması üsulu ilə öyrədilməsi həm reqressiya, həm də klassifikasiya məsələləri üçün istifadə oluna bilər. Siqmoidal funksiyanın xarakteristikaları klassifikasiya üçün tərtib olunmuş neyron şəbəkələrdə aktivləşdirmə funksiyası kimi istifadə olunması üçün çox uyğundur. Beləliklə, bu tip şəbəkələrdə gizli layların düyünlərinin aktivləşdirmə funksiyaları $g(x) = \sigma(x)$, sonuncu layın aktivləşdirmə funksiyası isə identiklik funksiyasıdır $g_0(x) = x$. Siqmoidal aktivləşdirmə funksiyasının törəməsinin rahat düsturu onun neyron şəbəkələrdə tətbiqinin əsas səbəbidir:

$$g'(x) = \frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x)).$$

Beləliklə, siqmoidal funksiyanın törəməsinin hesablanması sadəcə $\sigma(x)$ çıxışının yadda saxlanılması və yuxarıdakı tənlikdə yerinə qoyulmasından ibarətdir. Bundan əlavə çıxış layının aktivləşdirmə funksiyası isə daha sadədir:

$$g_0'(x) = \frac{\partial g_0(x)}{\partial x} = \frac{\partial x}{\partial x} = 1$$

Listinq 9 - da gizli laylarının aktivləşdirmə funksiyaları siqmoidal, çıxış layının aktivləşdirmə funksiyası identiklik funksiyası olan 3 gizli laydan, giriş layında 3 düyündən və 1 çıxış düyünündən ibarət neyron şəbəkənin 6 öyrədici nümunə ilə öyrədilməsi üçün Python mühitində proqramın kodu verilmişdir.

	precision	recall	f1-score	support
0	0.90	0.73	0.81	26
1	0.82	0.94	0.88	35
accuracy			0.85	61
macro avg	0.86	0.84	0.84	61
weighted avg	0.86	0.85	0.85	61

Şəkil 4.5: Verilənlərin çoxlaylı neyron şəbəkə üsulu ilə klassifikasiyasının qiymətləndirilməsi

4.2 Python mühitində neyron şəbəkə modelləri

Python mühitində neyron şəbəkələrin qurulması, öyrədilməsi və proqnozlaşdırma və ya klassifikasiya üçün istifadə edilməsinin iki yolu vardır; ya listinq 8-də göstərildiyi kimi proqramı sıfırdan yazmaq, ya da hazır kitabxanalardan istifadə etmək. Əgər neyron şəbəkələrin işləmə prinsipi sizə məlumdursa neyron şəbəkənin hazırlanmasının ən sadə və ən tez üsulu *keras* kitabxanasından istifadə etməkdir.

Bundan başqa *sklearn.neural_ network* kitabxanasının klassifikasiya üçün *MLPClassifier* funksiyası, proqnozlaşdırma üçün isə *MLPRegressor* funksiyasından istifadə etmək olar.

4.2.1 *Sklearn.neural_ network* kitabxanası

MLPClassifier funksiyasının (həmçinin MLPRegressor funksiyasının) bir sıra parametrləri bu funksiya vasitəsilə hazırlanan və öyrədilən neyron şəbəkənin gizli laylarının, bu laylardakı düyünlərin, ən tez enmə üsulunda iterasiyaların sayı, aktivləşdirmə funksiyasının tipi, öyrətmə addımının qiyməti kimi hiperparametrləri öncədən sazlamağa imkan verir. *Sklearn.neural_ network* kitabxanasının klassifikasiya funksiyasının işləmə prinsipini göstərmək üçün əvvəlki fəsillərdə istifadə olunan *heart.csv* verilənlər bazasının klassifikasiyası üçün çoxlaylı neyron şəbəkə modelini quraq və performansını qiymətləndirək (listinq 10).

Üç gizli laydan, bu laylarda uyğun olaraq 150, 100 və 50 düyündən ibarət olan və siqmoidal aktivləşdirmə funksiyası tətbiq edilən neyron şəbəkə vasitəsilə qurulan model ilə təsnifatın test mərhələsinin qarışıqlıq matrisi cədvəl 4.3 - də verilmişdir. Şəkil 4.5 - dən göründüyü kimi bu model ilə təsnifatın dəqiqliyi 85 % - ə bərabərdir.

Öncəki misalda çoxlaylı perseptron vasitəsilə klassifikasiya məsələsinin həllini göstərdik. Halbuki *MLP* həm də regressiya məsələsi üçün tətbiq oluna bilər. Bunun üçün *sklearn.neural_ network* kitabxanasından *MLPRegressor* funksiyasını çağırmaq lazımdır. Neyron şəbəkəni evlərin qiymətini müxtəlif atributlarla təyin

Listinq 9: neyron şəbəkənin xətanın tərs yayılması üsulu ilə öyrədilməsinin proqramı

```
import numpy as np
def sigmoid(x, derivative=False):
    if (derivative == True):
        return sigmoid(x,derivative=False) * (1 - sigmoid(x
            ,derivative=False))
    else:
        return 1 / (1 + np.exp(-x))
np.random.seed(1)
alpha = .1
num_hidden = 3
X = np.array([
    [0, 0, 1],
    [0, 1, 1],
    [1, 0, 0],
    [1, 1, 0],
    [1, 0, 1],
    [1, 1, 1],])
y = np.array([[0, 1, 0, 1, 1, 0]]).T
hidden_weights = 2*np.random.random((X.shape[1] + 1,
    num_hidden)) - 1
output_weights = 2*np.random.random((num_hidden + 1, y.
    shape[1])) - 1
num_iterations = 10000
for i in range(num_iterations):
    # forward phase
    input_layer_outputs = np.hstack((np.ones((X.shape[0],
        1)), X))
    hidden_layer_outputs = np.hstack((np.ones((X.shape[0],
        1)), sigmoid(np.dot(input_layer_outputs,
        hidden_weights))))
    output_layer_outputs = np.dot(hidden_layer_outputs,
        output_weights)
    # backward phase
    output_error = output_layer_outputs - y
    hidden_error = hidden_layer_outputs[:, 1:] * (1 -
        hidden_layer_outputs[:, 1:]) * np.dot(output_error,
        output_weights.T[:, 1:])
    hidden_pd = input_layer_outputs[:, :, np.newaxis] *
        hidden_error[: , np.newaxis, :]
    output_pd = hidden_layer_outputs[:, :, np.newaxis] *
        output_error[:, np.newaxis, :]
    total_hidden_gradient = np.average(hidden_pd, axis=0)
    total_output_gradient = np.average(output_pd, axis=0)
    hidden_weights += - alpha * total_hidden_gradient
    output_weights += - alpha * total_output_gradient
print("Output After Training: \n{}".format(
    output_layer_outputs))
```

Listinq 10: heart.csv verilənlər bazasının MLPClassifier funksiyası ilə təsnifatı

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
df = pd.read_csv('heart.csv').dropna()
x = df.drop('target', axis=1)
y = df['target']
trainX, testX, trainY, testY = train_test_split(x, y,
    test_size = 0.2)
mlp_clf = MLPClassifier(hidden_layer_sizes=(150,100,50),
                        max_iter = 300,activation = '
                            logistic',
                        solver = 'adam')
mlp_clf.fit(trainX, trainY)
y_pred = mlp_clf.predict(testX)
print('Accuracy: {:.2f}'.format(accuracy_score(testY,
    y_pred)))
from sklearn.metrics import confusion_matrix
print(confusion_matrix(testY, y_pred))
```

		Proqnozlaşdırılan	
		y = 0	y = 1
Real	y = 0	19	7
	y = 1	2	33

Cədvəl 4.3: *heart.csv* verilənlər bazasının çoxlaylı neyron şəbəkə ilə klassifikasiyası nəticəsində alınan qarışıqlıq matrisi

edən *Real estate valuation dataset* verilənlər bazasından istifadə edək (listinq 11) [23]. Proqnozlaşdırma modelinin performansının qiymətləndirilməsi üçün verilənlərin 20 %-nin test üçün istifadə edək, nəticədə alınmış qiymətlər cədvəl 4.4 - də göstərilmişdir. Eyni verilənlər çoxluğunu xətti reqressiya, kNN üsulları ilə də öyrədib modellərin performansını müqayisə etmək olar.

4.2.2 *Keras* ilə dərin öyrənmə

Keras dərin öyrənmə modellərinin hazırlanması və öyrədilməsi üçün ən sadə və ən sürətli açıq qaynaqlı Python kitabxanasıdır. Bu kitabxana **Tensorflow** kitabxanasının bir hissəsidir və bir neçə sətir kodla neyron şəbəkə modellərini qurmağa və öyrətməyə imkan verir. Bu bölmədə çiçək şəkillərinin **tf.keras.Sequential** modeli vasitəsilə klassifikasiyasını nəzərdən keçirəcəyik. Modelin işləmə ardıcıllığı

Metrikalar	Qiymət
MAE	5.28
MSE	52.48
RMSE	7.24
R2	0.69

Cədvəl 4.4: *Real estate valuation dataset* test bazasının proqnozlaşdırılmasının qiymətləndirilməsinin nəticələri

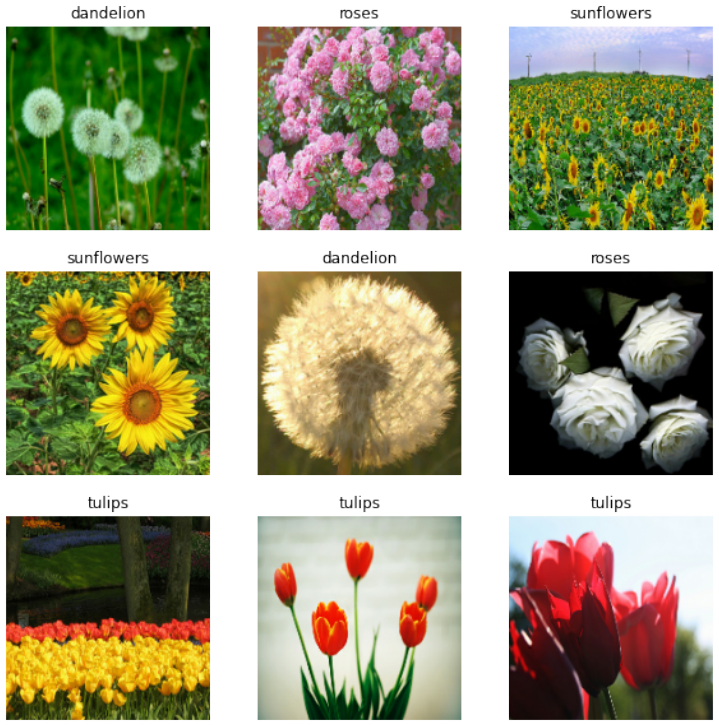
aşağıdakı addımlarla həyata keçirilir:

1. Verilənlərin daxil edilməsi;
2. Giriş layının hazırlanması;
3. Modelin qurulması;
4. Modelin öyrədilməsi;
5. Modelin test edilməsi;
6. Modelin təkmilləşdirilməsi və prosesin təkrarı.

Listinq 11: Real estate valuation dataset verilənlər bazasının MLPRegressor funksiyası ilə proqnozlaşdırılması

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
df = pd.read_excel('real_estate.xlsx').dropna()
X = df.drop(['No', 'Y house price of unit area'], axis = 1)
y = df['Y house price of unit area']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
from sklearn.neural_network import MLPRegressor
mlp_reg = MLPRegressor(hidden_layer_sizes=(150,100,50),
                       max_iter = 300,activation = 'relu',
                       solver = 'adam')

mlp_reg.fit(X_train, y_train)
y_pred = mlp_reg.predict(X_test)
from sklearn.metrics import mean_absolute_error
print(mean_absolute_error(y_test, y_pred))
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred))
print(pow(mean_squared_error(y_test, y_pred), 0.5))
from sklearn.metrics import r2_score
print(r2_score(y_test, y_pred))
```



Şəkil 4.6: Keras modeli ilə təsnif olunan çiçəklərin şəkillərinə nümunə

Təsnifat məsələsində 5 sinfə (qızılgül, çobanyastığı, lalə, günəbaxan və zəncirotu) aid 3700 şəkildən istifadə olunmuşdur (şəkil 4.6). Bunlardan 2936 şəkil öyrənmə üçün, 734 şəkil test üçün tətbiq olunmuşdur. Verilənlərin daxil edilməsi və nümunələrin nəzərdən keçirilməsi üçün kod listinq 12-də verilmişdir. Öyrətmə zamanı şəkillər 32 şəkildən ibarət dəstələrə (*batch*) bölünür, bu zaman şəkil dəstələrinin daxil edilməsi zamanı giriş verilənləri $32, 180, 180, 3$ şəklində olur, burada 180×180 şəklin ölçüsü 3 isə piksellərin RGB qiymətlərinə uyğun gəlir. Məlumdur ki, RGB qiymətləri $[0, 255]$ aralığında qiymətlər alır, lakin bu neyron şəbəkə üçün əlverişli olmadığından miqyaslaşdırma vasitəsilə (Rescaling) qiymətlər [0, 1] aralığına gətirilir.

Verilmiş şəkillərin təsnifatı üçün qurulmuş neyron şəbəkənin girişi ölçüləri $180 \times 180 \times 3$ olan üçölçülü matris şəklindədir. Daha sonra $16 \times 32 \times 64 \times 128$ ölçülü qıvrılmış neyron şəbəkə qurulur, birinci eksperimentdə hər bir layda aktivləşdirmə funksiyası kimi *relu* düzləşdirilmiş xətti funksiya istifadə olunmuşdur (listinq 13), bu zaman 10 iterasiya ilə öyrədilmə nəticəsində alınan nəticələr cədvəl 4.5 - də göstərilmişdir.

Cədvəldən göründüyü kimi yoxlamanın dəqiqliyi öyrənmənin dəqiqliyindən xeyli aşağıdır. Bunun səbəbini anlamaq üçün öyrətmənin dəqiqliyi ilə yoxlamanın dəqiqliyinin nə olduğunu nəzərdən keçirək. Öyrənmənin dəqiqliyini hesablamaq üçün həm öyrətmə, həm də test üçün eyni verilənlərdən istifadə olunur, yoxla-

Listinq 12: Çiçəklərin şəkillərindən ibarət bazanın daxil edilməsi, öyrədici və yoxlama bazasının təyini

```
import matplotlib.pyplot as plt
import numpy as np
import PIL
import tensorflow as tf
import pathlib
dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
data_dir = tf.keras.utils.get_file('flower_photos', origin=dataset_url, untar=True)
data_dir = pathlib.Path(data_dir)
batch_size = 32
img_height = 180
img_width = 180
train_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
val_ds = tf.keras.utils.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
class_names = train_ds.class_names
plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

Listinq 13: Şəkillərin təsnifatı üçün neyron şəbəkə modelinin qurulması

```
AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().shuffle(1000).prefetch(
    buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
normalization_layer = layers.Rescaling(1./255)
normalized_ds = train_ds.map(lambda x, y: (
    normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
# Notice the pixel values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
num_classes = len(class_names)
model = Sequential([
  layers.Rescaling(1./255, input_shape=(img_height,
      img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.
                  SparseCategoricalCrossentropy(from_logits=
                  True),
              metrics=['accuracy'])
model.summary()
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

ma zamanı isə öyrədici baza təsadüfi şəkildə altçoxluqlara bölünür (baxılan halda 1000 şəkildən ibarət). Bu altçoxluqlardan biri öyrətmə qalanları isə test üçün istifadə olunur. Qeyd edək ki, bu zaman öyrətmənin xətasının qiyməti 0.0574 - ə, yoxlanmanın xətasının qiyməti isə 1.7758 - ə bərabərdir. Sonuncu layda aktivləşdirmə funksiyasının tipini *siqmoidal* funksiyaya dəyişdikdə 10-cu iterasiyada alınan öyrənmə dəqiqliyi 0.9969, yoxlanma dəqiqliyi isə 0.6907 bərabərdir. Bu zaman öyrənmənin və yoxlanmanın xəta qiymətləri üçün uyğun olaraq 0.0251 və 1.1651 alınır. Daha yaxşı nəticələr almaq üçün modelin laylarının sayı, onların

ölçüləri, digər layların aktivləşdirmə funksiyalarının tipləri kimi hiperparametrləri dəyişmək və iterasiyaların sayını artırmaq lazımdır.

İterasiyanın nömrəsi	*Öyrənmənin (training) dəqiqliyi*	*Yoxlamanın (validation) dəqiqliyi*	*İcra müddəti (san)*
1	0.3835	0.5272	105
2	0.5433	0.5886	137
3	0.6359	0.6362	136
4	0.7234	0.6458	142
5	0.8178	0.6540	148
10	0.9850	0.6621	144

Cədvəl 4.5: Çiçək şəkillərinin *keras* modeli ilə təsnifatı zamanı alınan nəticələr (sonuncu layda aktivləşdirmə funksiyası xətti olduğu halda)

Çalışmalar

1. Aşağıdakı verilənlərə malik perseptron verilmişdir:
   Çəkilər: $w_1 = 0.5; w_2 = -0.7$
   Meyl: $b = 0.2$
   Aktivləşdirmə funksiyası: *tanh* funksiyası
   $x_1 = 0.8, x_2 = -0.3$ giriş verilənləri üçün çəkili orta kvadratik cəmi hesablayın.
   Aktivləşdirmə funksiyasını tətbiq edin və perseptronun çıxışını hesablayın.

2. İki əlamətdən ibarət verilənlər və onların çıxış sinifləri verilmişdir:
   $x_1 = [0.5, -0.3, 0.8, -0.5]$
   $x_2 = [-0.2, 0.9, 0.4, -0.7]$
   Siniflər: $[1, 0, 1, 0]$
   Başlanğıc çəkilər və meyl $w_1 = 0.1; w_2 = 0.2, b = 0.3$, öyrənmə addımı 0.1 olarsa perseptron öyrənmə alqoritmi ilə çəkilərin yeni qiymətlərini hesablayın.

3. MNIST verilənlər bazasının yükləyin, piksel qiymətlərini normallaşdırın və verilənlər bazasını öyrədici və test altçoxluqlarına bölün.

4. 3-cü tapşırıqdakı verilənlər bazası üçün sadə irəliyə doğru neyron şəbəkə qurun.

5. Eyni tapşırıq üçün qurulan neyron şəbəkəni öyrədin və iterasiyaları xətanı hesablyaraq qiymətləndirin.

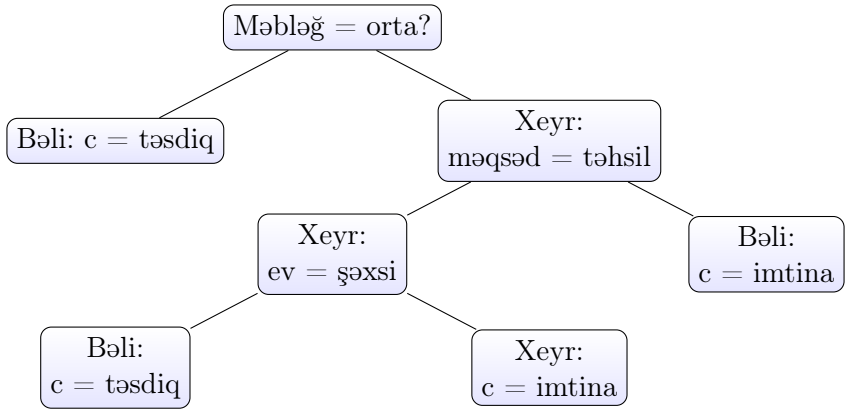
6. Test bazasında neyron şəbəkəni yoxlayın və nəticələrin dəqiqliyini, spesifikliyini və səhihliyini hesablayın.

Fəsil 5

Qərar qəbuletmə ağacı

Qərar qəbuletmə ağacının iş prinsipini praktiki olaraq anlamaq üçün paraqraf 2.2.1 - də təsvir olunmuş kredit risk verilənlər bazasında müraciət edənlərin xüsusiyyətlərinə əsasən onlar haqqında *təsdiq* və ya *imtina* şəklində klassifikasiyasından ibarət misalı nəzərdən keçirək (cədvəl 2.2). Tutaq ki, yeni müraciət daxil olmuşdur. Müraciət haqqında qərarın qəbul edilməsi üçün müraciət edənin xüsusiyyətləri haqqında suallar hazırlayıb onları cavablamaq lazımdır. Məsələn, birinci sual müraciət edənin evinin statusu haqqında ola bilər. Cədvələ əsasən əgər bu parametrin qiyməti *ipoteka* olarsa, onda qərar *təsdiq* olar, əks halda əlavə sual verilməlidir. Məsələn, kreditin məqsədi nədir? Əgər məqsəd *təhsil* olarsa, nəticə *imtina* olar, əks halda isə yenə əlavə sual verilməlidir.

Bu nümunə müxtəlif tipli suallar verməklə klassifikasiya problemini necə həll etmək mümkün olduğunu göstərir. Hər dəfə suala cavab verdikcə, növbəti sual gəlir və nümunənin aid olduğu sinif haqqında nəticənin alınmasına qədər suallara cavab verməyə davam etdirilir. Suallar ardıcıllığı və onlara mümkün cavablar çoxluğu düyünlər və istiqamətlənmiş tillərdən ibarət iyerarxik struktur olan qərar qəbuletmə ağacı ilə təsvir oluna bilər (şəkil 5.1).



Şəkil 5.1: Kredit üçün müraciət edənlərin qərar qəbuletmə ağacı ilə təsnifatı

Qərar qəbul etmə ağacında 3 tip düyün var:

1. Kök düyün - ona gələn tillər olmayan, ondan sıfır və ya daha çox tillər çıxan düyün;

2. Daxili düyün - ona gələn bir til olan və iki və daha çox çıxan til olan düyün;

3. Yarpaq düyün - ona gələn bir til olan və çıxan heç bir til olmayan düyün.

Şəkil 5.1 - də təsvir olunmuş qərar qəbuletmə ağacında *Məbləğ = orta* düyünü *kök* düyün, siniflərin təyin olunduğu düyünlər *yarpaq* düyünlər, digərləri isı *daxili* düyünlərdir.

Verilənlər çoxluğunun atribut parametrlərinə əsasən sonsuz sayda qərar qəbuletmə ağacı tərtib etmək olar. Bəzi ağaclar vasitəsilə təsnifat digərlərindən daha yüksək dəqiqliyə malik olur, lakin axtarış fəzası eksponensial ölçüyə malik olduğuna görə optimal ağacın tapılması qeyri - mümkündür. Buna baxmayaraq uyğun zaman müddətində kifayət qədər dəqiq suboptimal ağacın qurulması üçün effektiv alqoritmlər mövcuddur. Bu alqoritmlərdən biri **Hunt** alqoritmidir.

5.1 Hunt alqoritmi

Hunt alqoritmində qərar qəbuletmə ağacı öyrədici nümunələri rekursiv olaraq nisbətən kiçik altçoxluqlara bölməklə qurulur [24]. Tutaq ki, D_t, t düyününə uyğun nümunələri özündə saxlayan çoxluqdur və $y = \{y_1, y_2, ...y_c\}$ klassifikasiya sinifləridir. Aşağıda Hunt alqoritminin rekursiv icrasının addımları göstərilmişdir:

1. Əgər D_t çoxluğundakı bütün nümunələr eyni y_t sinfinə aiddirsə, onda t düyünü y_t kimi ifadə olunan yarpaq düyünüdür;

2. Əgər D_t çoxluğunda bir neçə sinfə aid olan nümunələr varsa, onda bu nümunələri daha kiçik çoxluqlara bölmək üçün yeni şərt əlavə olunur. Şərtin nəticələrinə uyğun olaraq yeni düyünlər əlavə olunur və D_t çoxluğunda nümunələr bu şərtin nəticələrinə uyğun altçoxluqlara bölünür. Daha sonra alqoritm rekursiv olaraq hər bir düyünə tətbiq olunur.

Alqoritmin işləmə prinsipini göstərmək üçün daha öncə istifadə etdiyimiz verilənlər bazasını - kreditin nəticəsinin müraciət edənlərin xarakteristikaları ilə təsnifatını nəzərdən keçirək (cədvəl 5.1).

Bu məqsədlə ilk öncə kök düyün olaraq *Məbləğ = orta* götürək, bu zaman nümunələr 3-cü atributunun qiyməti *orta* olan (D_1) və fərqli olan (D_2) iki çoxluğa ayrılır. Məbləğ əlaməti orta olan nümunə *təsdiq* sinfinə aiddir. Məbləğ əlaməti yüksək və aşağı olan nümunələrin içərisində isə həm *təsdiq*, həm də *imtina* siniflərinə aid olan nümunələr mövcuddur. Bu o deməkdir ki, əlavə şərtə ehtiyac vardır. Bunun üçün növbəti daxili düyün üçün *məqsəd = təhsil* şərtini nəzərdən keçirək. Bu zaman D_2 çoxluğunu 2-ci atributu *təhsil* (D_3) və *təhsil olmayan* (D_4) iki sinfə ayrılır. D_3 çoxluğunun bütün elementləri *imtina* sinfinə aid olduğundan

İD	EV	MƏQSƏD	MƏBLƏĞ	STATUS
1	şəxsi	şəxsi	yüksək	təsdiq
2	şəxsi	təhsil	aşağı	imtina
3	ipoteka	tibbi	orta	təsdiq
4	kirayə	təhsil	yüksək	imtina
5	şəxsi	investisiya	aşağı	təsdiq
6	kirayə	investisiya	yüksək	imtina

Cədvəl 5.1: Credit_risk_dataset verilənlər bazasından bir neçə nümunə.

növbəti yarpaq düyün təyin olunur. D_4 çoxluğunda isə həm *təsdiq*, həm də *imtina* siniflərinə aid olan nümunələr mövcuddur. Buna görə də yeni bir şərt, $Ev = şəxsi$ olan düyün daxil edilir. Nəticədə D_4 çoxluğu 1-ci atributunun qiyməti *şəxsi* olan (D_5) və ondan fərqli olan (D_6) çoxluqlara ayrılır. D_5 çoxluğunun elementləri *təsdiq* sinfinə, D_6 çoxluğunda olan bir element isə *imtina* sinfinə aid olduğundan növbəti yarpaq düyünləri təyin olunur. Bu bölünmələrdə cədvəldəki bütün nümunələr iştirak etdiyinə görə alqoritm sona çatır (şəkil 5.1).

Qərar qəbuletmə ağacı ilə klassifikasiya üçün öyrənmə alqoritminin aşağıdakı əsas problemləri mövcuddur:

1. Öyrədici nümunələr necə ayrılmalıdır? Ağacın qurulmasının hər bir rekursiv addımı nümunələri daha kiçik çoxluqlara ayırmaq üçün atribut test şərtini seçməlidir. Bu addımı həyata keçirmək üçün müxtəlif tiplər üçün test şərtinin seçilməsi və hər bir test şərtinin qiymətləndirilməsi üçün obyektiv meyarın təyin edilməsi vacibdir;

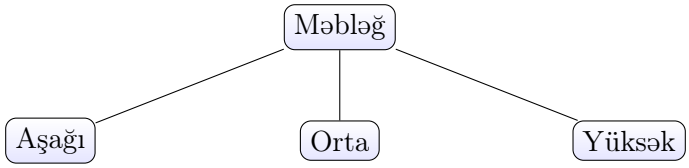
2. Nümunələrin ayrılması proseduru nə vaxt bitir? Ağacın qurulması prosesini sonlandıracaq bir meyar təyin olunmalıdır. Mümkün strategiya ondan ibarətdir ki, bütün nümunələr eyni sinifdən olana və ya bütün nümunələrin atribut qiymətləri eyni olana qədər davam etməkdir. Qərar ağacının qurulması prosedurunun sonlandırılması üçün hər iki strategiya uyğun olsa da prosesin daha tez sonlandırılması üçün başqa strategiyalar da mövcuddur.

5.2 Nümunələrin bölünməsi strategiyaları

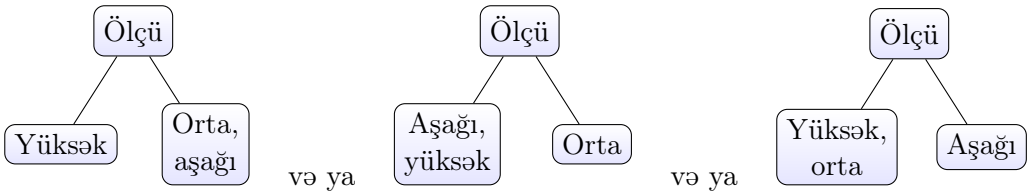
Qərar qəbuletmə ağacının qurulması alqoritmləri müxtəlif atribut tipləri üçün atribut test şərtləri və uyğun düyünləri təyin etmək üçün yolları özündə ehtiva etməlidir.

Binar atributlar - asanlıqla iki hissəyə ayrıla bilən atribut verilənləridir. Məsələn, heyvanların *soyuqqanlı* və *istiqanlı* kimi iki müxtəlif siniflərə ayrılması.

Nominal atributlar - çoxlu sayda müxtəlif qiymətlər ala bildiyinə görə test şərti iki yolla verilə bilər: ya baxılan atributun ala biləcəyi bütün qiymətlərə əsasən düyünlərin təyin olunması, ya da $2^{k-1}-1$ sayda mümkün üsullarla binar bölünmələrin həyata keçirilməsi, burada k - atributun qiymətlərinin sayıdır. Məsələn, kredit



Şəkil 5.2: Nominal atributların çoxsaylı düyünlərə ayrılması



Şəkil 5.3: Nominal atributların binar düyünlərə ayrılması

üçün müraicət edənlərin *{aşağı, orta, yüksək}* qiymətlərini alan *məbləğ* atributunu həm çoxlu olaraq (şəkil 5.2), həm də binar düyünlərə ayırmaq olar (şəkil 5.3).

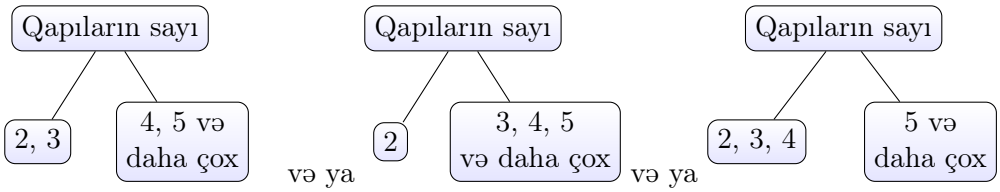
Ordinal atributlar - da eyni zamanda həm çoxsaylı, həm də binar düyünlərə ayrıla bilər. Ordinal atributlar düzülmə sırası pozulmadığı halda istənilən cür qruplaşdırıla bilər. Şəkil 5.4 - də maşın modellərinin təsnifatında istifadə olunan *Qapıların sayı* ordinal atributunun aldığı $\{2, 3, 4, 5$ və daha çox$\}$ qiymətlərin müxtəlif üsullarla binar düyünlərə ayrılması təsvir olunmuşdur.

Şəkil 5.5 - də verilmiş binar bölünmə düzgün deyil, çünki *2* və *5 və daha çox* qiymətləri eyni düyündə, *3* və *4* qiymətləri isə eyni düyündə yerləşmişdir, ardıcıllıq pozulmuşdur.

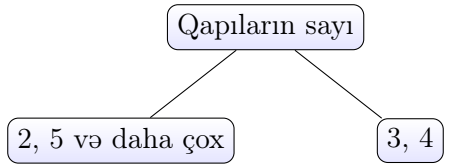
Rəqəmsal atributlar üçün test şərti $A < v$ və ya $A \geq v$ müqayisə testi şəklində və ya $i = 1, 2, ..., k$ olduqda $v_i \leq A < v_{i+1}$ qiymətlər intervalı şəklində seçilə bilər. Məsələn, $[0; 100K]$ intervalında qiymətlər alan *İllik qazanc* rəqəmsal atributu üçün hər iki üsulla bölünmənin fərqi şəkil 5.6 və 5.7 - də göstərilmişdir.

5.2.1 Ən yaxşı bölünmə meyarları

Nümunələrin müxtəlif atribut qiymətlərinə görə bölünməsi üçün müxtəlif meyarlar istifadə oluna bilər. Bu meyarlar bölünmədən əvvəl və sonra nümunələrin siniflərə paylanması əsasında təyin olunur. Tutaq ki, $p(i|t)$ verilmiş t düyünündə i sin-



Şəkil 5.4: Ordinal atributların binar düyünlərə ayrılması



Şəkil 5.5: Ordinal atributun düzgün olmayan binar düyünlərə ayrılması

Şəkil 5.6: Rqəməsal atributun binar düyünlərə ayrılması

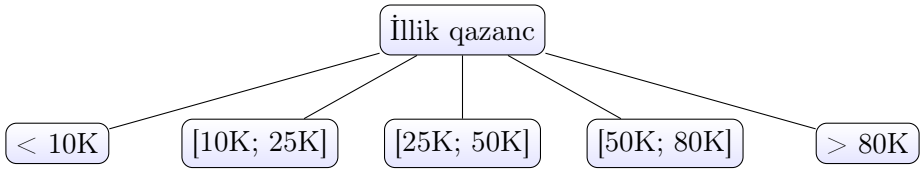
finə aid olan nümunələrin olma tezliyidir. Bir çox hallarda t düyünü istifadə olunmur və tezlik p_i kimi ifadə olunur. İki sinifli yəni binar klassifikasiya problemində nümunələrin hər hansı bir düyünə aid olması $(p_0; p_1)$ şəklində yazıla bilər, burada $p_1 = 1 - p_0$. Ən yaxşı bölünmənin seçilməsi adətən düyünlərin qarışıqlıq dərəcəsindən asılıdır. Qarışıqlıq dərəcəsi nə qədər kiçikdirsə, nümunələrin siniflərə paylanması bir o qədər dəqiqdir. Məsələn, əgər düyündə nümunələrin bölünməsi zamanı bir sinifdə 0 nümunə olduğu halda, qalan bütün nümunələr digər sinifdədirsə, qarışıqlıq dərəcəsi 0-a bərabərdir, bölünmə zamanı hər iki sinif də yarı-yarı bölünürsə, onda düyünün qarışıqlıq dərəcəsi maksimumdur. Qarışıqlıq dərəcəsi aşağıdakı üsullardan biri ilə hesablana bilər:

$$Entropy(t) = -\sum_{i=0}^{c-1} p(i|t) log_2 p(i|t), \tag{5.1}$$

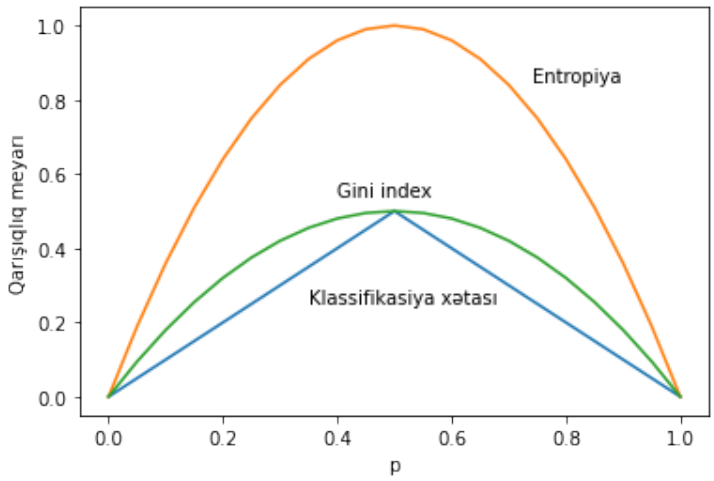
$$Gini(t) = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2, \tag{5.2}$$

$$CE(t) = 1 - max_i[p(i|t)], \tag{5.3}$$

burada CE - klassifikasiya xətası, c isə mövcud siniflərin sayıdır. Şəkil 5.8 - də binar klassifikasiya üçün qarışıqlıq meyarları arasındakı fərq vizual olaraq təsvir edilmişdir, burada p - iki sinifdən birinə daxil olan nümunələrin tezliyini göstərir.



Şəkil 5.7: Rqəməsal atributun çıxsaylı düyünlərə ayrılması



Şəkil 5.8: Binar klassifikasiya problemləri üçün qarışıqlıq meyarlarının fərqi

Qrafikdən göründüyü kimi hər bir meyar maksimum qiymətini $p = 0.5$ olduqda, yəni nümunələrin hər iki sinfə paylanması sayı eyni olduqda alınır. Minimum qiymətlər isə nümunələrin hamısının bir sinfə aid olduğunda ($p = 0$ və ya $p = 1$) olduqda alınır. Aşağıda qarışıqlıq meyarlarının hesablanması üçün bəzi nümunələr göstərilmişdir.
Tutaq ki, nümunələrin siniflər üzrə paylanması cədvəl 5.2 - də göstərildiyi kimidir. Bu halda qarşıqılıq meyarlarının qiyməti aşağıdakı kimi olar:

N_1 düyünü	Nümunələrin sayı
c (sinif) = 0	0
c (sinif) = 1	6

Cədvəl 5.2: Nümunələrin siniflər üzrə paylanması N_1

$$Gini = 1 - \left(\frac{0}{6}\right)^2 - \left(\frac{6}{6}\right)^2 = 0,$$

$$Entropiya = -\left(\frac{0}{6}\right) log_2\left(\frac{0}{6}\right) - \left(\frac{6}{6}\right) log_2\left(\frac{6}{6}\right) = 0,$$

$$CE = 1 - max\left[\frac{0}{6}, \frac{6}{6}\right] = 0.$$

Aşağıdakı düsturlarda nümunələrin cədvəl 5.3 və 5.4 - də göstərilən şəkildə nümunələrin paylanması üçün qarışıqlıq meyarlarının hesablanması göstərilmişdir.

$$Gini = 1 - \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2 = 0.278,$$

N_2 düyünü	Nümunələrin sayı
c (sinif) = 0	1
c (sinif) = 1	5

Cədvəl 5.3: Nümunələrin siniflər üzrə paylanması N_2

$$Entropiya = -\left(\frac{1}{6}\right) log_2\left(\frac{1}{6}\right) - \left(\frac{5}{6}\right) log_2\left(\frac{5}{6}\right) = 0.650,$$

$$CE = 1 - max\left[\frac{1}{6}, \frac{5}{6}\right] = 0.167.$$

N_3 düyünü	Nümunələrin sayı
c (sinif) = 0	3
c (sinif) = 1	3

Cədvəl 5.4: Nümunələrin siniflər üzrə paylanması N_3

$$Gini = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5,$$

$$Entropiya = -\left(\frac{3}{6}\right) log_2\left(\frac{3}{6}\right) - \left(\frac{3}{6}\right) log_2\left(\frac{3}{6}\right) = 1,$$

$$CE = 1 - max\left[\frac{3}{6}, \frac{3}{6}\right] = 0.5.$$

Yuxarıda təsvir olunmuş hesablamalara əsasən N_1 düyünü ən kiçik qarışıqlıq meyarına malikdir, daha yüksək qarışıqlıq meyarı N_2 düyünündə, ən yüksək qiymət isə N_3 düyünündə müşahidə olunur. Bölünmə üçün test şərtinin əlverişli olub-olmamasını yoxlamaq üçün *valideyn* düyünün qarışıqlıq meyarını hesablayıb (bölünmədən əvvəl), onu *uşaq* düyünlərinin qarışıqlıq meyarları ilə (bölünmədən sonra) müqayisə etmək lazımdır. Bu fərq nə qədər böyük olarsa, bölünmə o qədər əlverişli olar. Burada qazanc - Δ, bölünmənin əlverişliliyini ölçən meyardır:

$$\Delta = I(valideyn) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j) \tag{5.4}$$

burada $I(\cdot)$ - verilmiş düyünün qarışıqlıq meyarı, N - valideyn düyündəki bütün nümunələrin sayı, k - atribut verilənlərinin aldığı qiymətlərin sayı, $N(v_j)$ - v_j uşaq düyünü ilə bağlı olan nümunələrin sayıdır. Qərar qəbuletmə ağacının qurulması alqoritmləri adətən qazancın - Δ maksimumlaşdırılmasından ibarət olur.

Ev sahibidir	Ailə vəziyyəti	İllik qazanc	Gender	Qərar
Bəli	Subay	125K	Kişi	Xeyr
Xeyr	Evli	100K	Qadın	Xeyr
Xeyr	Subay	70K	Qadın	Xeyr
Bəli	Evli	120K	Qadın	Xeyr
Xeyr	Dul	95K	Qadın	Bəli
Xeyr	Evli	60K	Kişi	Xeyr
Bəli	Dul	220K	Kişi	Xeyr
Xeyr	Subay	85K	Kişi	Bəli
Xeyr	Evli	75K	Kişi	Xeyr
Xeyr	Subay	90K	Kişi	Bəli

Cədvəl 5.5: Bank müştərilərinin xarakteristikalarına əsasən borcların ödənməsinin gecikməsi haqqında məlumat

$I(valideyn)$ - bütün test şərtləri üçün eyni olduğundan qazancın maksimumlaşdırılması uşaq düyünlərinin qarışıqlıq meyarlarının orta qiymətinin minimallaşdırılmasına ekvivalentdir.

Qərar qəbuletmə ağacının qurulmasını nəzərdən keçirmək üçün cədvəl 5.5 - də verilmiş nümunələri nəzərdən keçirək. Bu cədvəldə bank müştərilərinin atribut parametrlərinə əsasən borclarını gecikdirməsi haqqında məlumatlar göstərilmişdir. İlk öncə binar atributların bölünməsinə baxaq. *Ev sahibliyi* və *Gender* atribut parametrlərinin bölünməsini nəzərdən keçirək. Bölünmədən öncə nümunələrin siniflərə paylanması cədvəl 5.6 - da göstərilmişdir. Adıçəkilən parametrlərə görə bölünmənin nəticələri cədvəl 5.7 və 5.8 - də təsvir olunmuşdur (şəkil 5.9).

Valideyn düyünü	
Qərar = Xeyr	7
Qərar = Bəli	3
Gini Index = 0.5	

Cədvəl 5.6: Nümunələrin paylanması

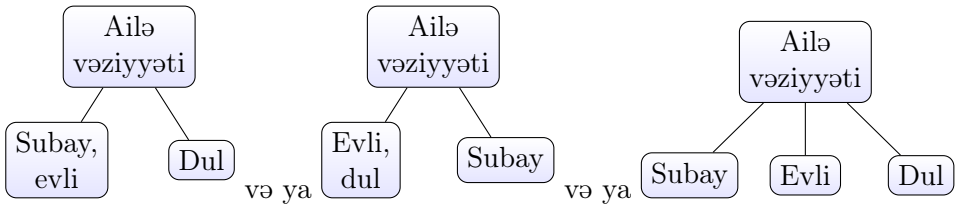
	N_1	N_2
Qərar = Xeyr	3	4
Qərar = Bəli	0	3
Gini Index	0	0.49
Gini Index = 0.286		

Cədvəl 5.7: *Ev sahibliyi* parametrinə görə bölünmə

Cədvəllərdən göründüyü kimi əgər test şərti olaraq *Ev sahibliyi* atributu götürülərsə, onda N_1 düyünü üçün (qiymət = Bəli) Gini Index 0, N_2 düyünü üçün (qiymət = Xeyr) Gini Index 0.49 olar, Gini Indexin orta qiyməti isə $0 \times 3/10 +$

	N_1	N_2
Qərar = Xeyr	4	3
Qərar = Bəli	2	1
Gini Index	0.444	0.375
Gini Index = 0.42		

Cədvəl 5.8: *Gender* parametrinə görə bölünmə



Şəkil 5.9: Nominal atributların binar düyünlərə ayrılması

$0.49 \times 7/10 = 0.286$ olar. Eyni qayda ilə *Gender* atributu üçün Gini Indexin orta qiyməti 0.42 olur. *Ev sahibliyi* atributunun Gini Indexi daha kiçik olduğuna görə test şərti olaraq seçilməsi daha məqsədəuyğundur.

İndi isə nominal atributların bölünməsini nəzərdən keçirək. Yuxarıda qeyd edildiyi kimi nominal atributları həm binar yolla, həm də çoxlu düyünlərə ayırmaq olar. Bunun üçün *Ailə vəziyyəti* atribut parametrini nəzərdən keçirək. Bu parametrin aldığı qiymətlərin sayı $k = 3$ olduğundan mümkün binar ayrılmalar $2^{k-1} - 1 = 3$ saydadır. Şəkil 5.9 - da *Ailə vəziyyəti* parametrinin binar bölünməsinə nümunələr və çoxsinifli ayrılma göstərilmişdir. Göstərilmiş bölünmələrə əsasən nümunələrin paylanması cədvəl 5.9, 5.10 və 5.11 - də göstərilmişdir. (5.2) düsturuna əsasən Gini Indexin qiymətlərini hesablasaq, minimum qiymətin çoxsinifli bölünmə üçün alındığını müşahidə etmiş oluruq. Beləliklə, daha dəqiq klassifikasiya əldə etmək üçün çoxsinifli bölünmədən, əgər binar ağacın qurulması tələb olunarsa isə cədvəl 5.10 - da göstərilən bölünmədən istifadə etmək məqsədəuyğundur. Ordinal atributların da bölünməsi nominal atributlara anolojidir, sadəcə burada atribut qiymətlərinin düzülüşünü nəzərdə saxlamaq lazımdır.

	$N_1 - Subay, Evli$	$N_2 - Dul$
Qərar = Xeyr	6	1
Qərar = Bəli	2	1
Gini Index	0.375	0.5
Gini Index = 0.4		

Cədvəl 5.9: *Ailə vəziyyəti* parametrinə görə binar bölünmə - 1-ci hal

Rəqəmsal atributların bölünməsini anlamaq üçün baxılan verilənlər çoxluğunda *İllik qazanc* atributunu nəzərdən keçirək. Tutaq ki, klassifikasiya problemini həll etmək üçün verilmiş nümunələri $A \leq v$ şərtinə əsasən bölmək lazımdır. Bu məsələni

	$N_1 - Evli, dul$	$N_2 - Subay$
Qərar = Xeyr	1	2
Qərar = Bəli	5	2
Gini Index	0.28	0.5

Gini Index = 0.368

Cədvəl 5.10: *Ailə vəziyyəti* parametrinə görə binar bölünmə - 2-ci hal

	$N_1 - Subay$	$N_2 - Evli$	$N_3 - Dul$
Qərar = Xeyr	2	4	1
Qərar = Bəli	2	0	1
Gini Index	0.5	0	0.5

Gini Index = 0.3

Cədvəl 5.11: *Ailə vəziyyəti* parametrinə görə çoxsinifli bölünmə

həll etmək üçün bütün nümunələrin bu atribut üçün aldığı bütün qiymətləri (N) nəzərə almaq lazımdır. Daha sonra hər bir v qiyməti üçün nümunələrin içərisindən *İllik qazanc* atributunu v - dən kiçik və v - dən böyük olmaqla siniflərə bölmək, hər bir hal üçün Gini indeksi hesablamaq və ən kiçik Gini İndexə malik olan bölünməni seçmək lazımdır. Bu proses hər bir bölünmə üçün Gini Indexin hesablanmasını tələb etdiyinə görə $O(N^2)$ sayda əməliyyatlara malik çox resurs tələb edən prosesdir. Tələb olunan əməliyyatların sayını azaltmaq üçün bütün nümunələr baxılan atributun qiymətlərinə əsasən artan sırada düzülür və bölünmə qiyməti olaraq iki qonşu qiymətin ədədi ortası götürülür. Bu zaman tələb olunan əməliyyatların sayı $O(NlogN)$ olar.

Cədvəl 5.5 - də verilən nümunələri *İllik qazanc* atributuna görə artan sırada düzək. Bu zaman baxılan atribut üçün $v = [60, 70, 75, 85, 90, 95, 100, 120, 125, 220]$ vektoru alınar. Daha sonra bölünmə şərtləri olaraq iki ardıcıl qiymətin ədədi ortasını götürək. Birinci bölünmə şərti $v = 55K$ olar. Bundan kiçik qiymət olmadığına görə bu düyünün Gini Indexi 0 - a bərabərdir. Digər tərəfdən $v \geq 55K$ olan nümunələrin sayı *Xeyr* sinfi üçün 7 - ə, *Bəli* sinfi üçün isə 3 - ə, hesablanan Gini Index isə 0.420 - ə bərabərdir. Beləliklə, bu bölünmənin Gini Indexinin orta qiyməti 0.420 olar. Növbəti şərt üçün birinci və ikinci nümunənin orta qiyməti olaraq $v = 65K$ qarışıqlıq meyarını əvvəlki paylanmanı dəyişməklə hesablamaq olar. Bu halda $v < 65K$ şərtini ödəyən nümunə *Xeyr* sinfinə aiddir, bu halda birinci düyün üçün bu sinifdən olan nümunələrin sayı 1 vahid artır, $v \geq 65K$ düyünü üçün bu sinifdən olan nümunələrin sayı 1 vahid azalır. Nəticədə bu bölünmə üçün Gini Indexin orta qiyməti 0.400 olar. Bu qayda ilə bütün qiymətlər üçün Gini Index hesablanır (cədvəl 5.12). Cədvəldən göründüyü kimi Gini Indexin ən kiçik qiyməti $v = 98$ olan hal üçün alınır. Təsvir olunan proses hər bölünmə şərtinin hesablanması üçün sabit vaxt tələb etdiyindən daha sərfəlidir. Eyni zamanda baxılan alqoritmdə iki qonşu test şərti üçün qarışıqlıq meyarı o vaxt hesablanır ki, bu bölünmə zamanı nümunələrin siniflərə paylanması fərqli olsun. Məsələn, düzülmüş

ilk 3 nümunə (60K, 70K, 75K) eyni sinifə aid olduğundan ən yaxşı bölünmə şərti 60K və 75K arasında olmayacaq. Eyni qayda ilə $v = 55, 65, 73, 88, 93, 110, 123, 173$ şərtləri nəzərə alınmır, çünki bu qiymətlər eyni sinfə aid olan iki nümunə arasında yerləşir. Bu yanaşma bölünmələrin sayını 11-dən 2-ə endirməyə imkan verir.

55		65		73		80		88		93		98		110		123		173	
$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$	$\leq$	$>$
0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
0	7	1	6	2	5	3	4	3	4	3	4	4	3	5	2	6	1	7	0
0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.420	

Cədvəl 5.12: *İllik qazanc* parametrinin bölünməsi

Reqressiya ağacında bölünmənin qiymətləndirilməsi

Qərar qəbuletmə ağacı modeli yalnız klassifikasiya deyil, proqnozlaşdırma üçün də istifadə oluna bilər. Reqressiya ağacının qurulması zamanı bölünmənin qiymətləndirilməsi üçün düyünün şərtlərini ödəyən nümunələr arasındakı yayılmanı ölçmək lazımdır. Klassifikasiya zamanı yarpaq düyünlərində nəticə olaraq adətən bu düyünün şərtlərini ödəyən öyrədici nümunələrin çox hissəsinin aid olduğu sinfin nömrəsi götürülürdüsə, reqressiya ağacında nəticə olaraq yarpaq düyünlərindəki nümunələrin nəticələrinin orta qiyməti götürülür.

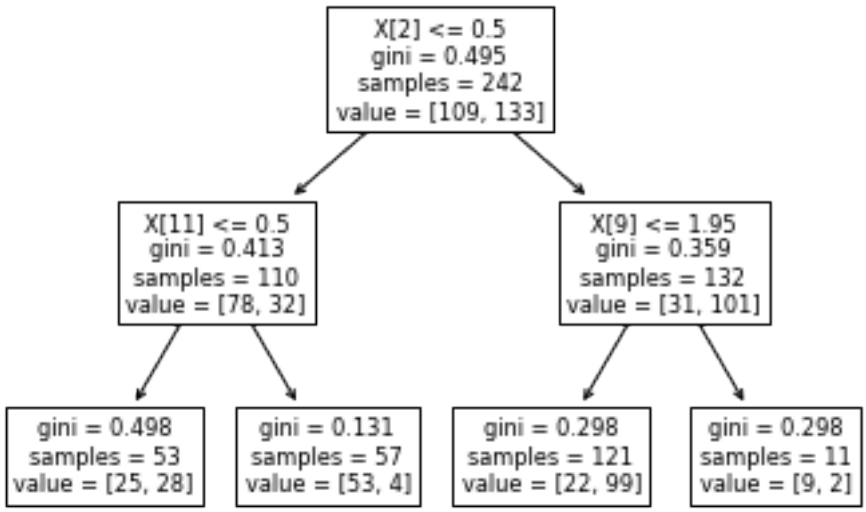
Düyünlər üçün ən yaxşı atributların seçilməsi məqsədilə əlamətlərin qiymətləndirilməsi üçün dispersiyanın qiyməti götürülür:

$$Var(D) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \mu)^2,$$

burada y_i - düyünün şərtlərini ödəyən nümunələrin çıxış qiymətləri, $\mu = \frac{1}{N} \sum_{i=1}^{N} y_i$.

5.3 Python mühitində qərar qəbuletmə ağacının qurulması

Verilənlərin klassifikasiyası üçün qərar qəbuletmə ağacının qurulması üçün Python *sklearn.tree.DecisionTreeClassifier* modulundan istifadə edək. Bunun üçün verilmiş funksiyanı əvvəlki paraqraflarda klassifikasiya üçün istifadə olunan *heart.csv* verilənlər çoxluğuna tətbiq edək.



Şəkil 5.10: heart.csv verilənlənlərinin klassifikasiyası üçün qərar qəbuletmə ağacı (dərinlik = 2)

Listinq 14: heart.csv verilənlər bazasının qərar qəbuletmə ağacı ilə klassifikasiyası

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_csv('heart.csv')
X = data.drop('target', axis = 1)
y = data['target']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
from sklearn import tree
clf = tree.DecisionTreeClassifier(max_depth =2)
DTclf = clf.fit(X_train,y_train)
y_pred = DTclf.predict(X_test)
from sklearn.metrics import classification_report,
    accuracy_score
print(classification_report(y_test, y_pred))
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, y_pred))
```

DecisionTreeClassifier funksiyasının müxtəlif parametrləri var, bunlardan biri qərar qəbuletmə ağacının maksimal dərinliyidir. Maksimal dərinlik ağacın kökündən yarpaq düyünlərə qədər olan tillərin maksimal uzunluğudur. Şəkil 5.10 - da maksimal dərinlik 2 olduqda alınan qərar qəbuletmə ağacı təsvir olunmuşdur. Bu zaman klassifikasiyanın qarışıqlıq matrisi cədvəl 5.13 - də, modelin dəqiqlik göstəriciləri

	precision	recall	f1-score	support
0	0.81	0.72	0.76	29
1	0.77	0.84	0.81	32
accuracy			0.79	61
macro avg	0.79	0.78	0.78	61
weighted avg	0.79	0.79	0.79	61

Şəkil 5.11: Verilənlərin qərar qəbuletmə ağacı ilə klassifikasiyasının qiymətləndirilməsi (max depth = 2 olduqda)

	precision	recall	f1-score	support
0	0.80	0.83	0.81	29
1	0.84	0.81	0.83	32
accuracy			0.82	61
macro avg	0.82	0.82	0.82	61
weighted avg	0.82	0.82	0.82	61

Şəkil 5.12: Verilənlərin qərar qəbuletmə ağacı ilə klassifikasiyasının qiymətləndirilməsi (max depth = 3 olduqda)

isə şəkil 5.11 - də göstərilmişdir.

		Proqnozlaşdırılan	
		y = 0	y = 1
Real	y = 0	21	8
	y = 1	5	27

Cədvəl 5.13: *heart.csv* verilənlər bazasının qərar qəbuletmə ağacı ilə (max depth = 2) klassifikasiyası nəticəsində alınan qarışıqlıq matrisi

Aydındır ki, qərar qəbuletmə ağacının həcmini, yəni maksimal dərinliyini artırsaq, modelin dəqiqliyi də artar. Listinq 14 - də 10 cu sətirdə *DecisionTreeClassifier* funksiyasının *max_ depth* parametrinin qiymətini 3 götürsək, həm qarışıqlıq matrisi, həm də modelin dəqiqliyində kifayət qədər yaxşılaşma müşahidə etmiş olarıq (şəkil 5.12 və cədvəl 5.14).

5.3.1 Reqressiya ağacı

Verilənlərin atributlarına əsasən asılı dəyişənin proqnozlaşdırılması üçün Python mühitində *DecisionTreeRegressor* funksiyasından istifadə etmək lazımdır. Məsələn,

		Proqnozlaşdırılan	
		y = 0	y = 1
Real	y = 0	24	5
	y = 1	6	26

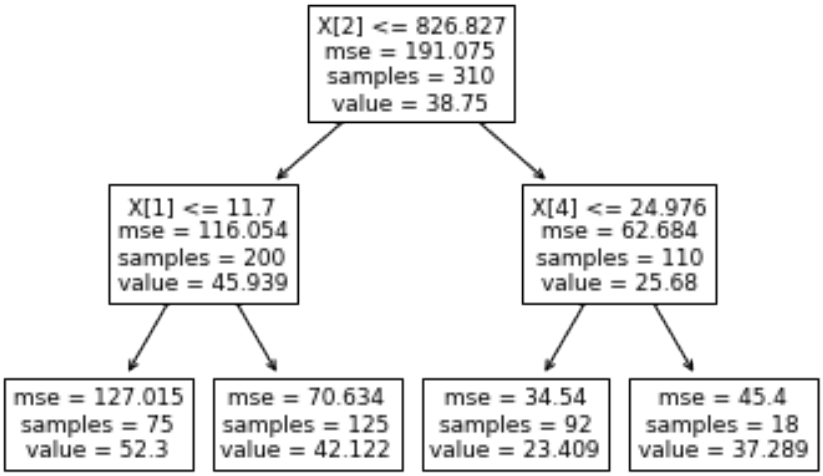
Cədvəl 5.14: *heart.csv* verilənlər bazasının qərar qəbuletmə ağacı ilə (max depth = 3) klassifikasiyası nəticəsində alınan qarışıqlıq matrisi

öncəki paraqraflarda nəzərdən keçirdiyimiz *Real Estate Valuation Dataset* verilənlərinin reqressiya ağacı ilə proqnozlaşdırılmasına baxaq (listinq 15). Bu zaman alınan reqressiya ağacı şəkil 5.13 - də təsvir olunmuşdur, test bazasında modelin qiymətləndirilməsinin nəticələri cədvəl 5.15 - də göstərilmişdir.

Reqressiya ağacının da dərinliyini artırmaqla nəticələri yaxşılaşdırmaq mümkündür, məsələn *DecisionTreeRegressor* funksiyasının *max_ depth* parametrinin qiymətini artırıb 3-ə bərabər etdikdə alınan nəticələrin dəqiqliyi kifayət qədər yaxşılaşır (cədvəl 5.16).

Listinq 15: Real estate.xlsx verilənlər bazasının reqressiya ağacı ilə proqnozlaşdırılması

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data = pd.read_excel('Real_estate.xlsx')
X = data.drop(['No', 'Y house price of unit area'], axis =
    1)
y = data['Y house price of unit area']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.25, random_state=42)
from sklearn import tree
clf = tree.DecisionTreeRegressor(max_depth =2)
DTclf = clf.fit(X_train,y_train)
y_pred = DTclf.predict(X_test)
from sklearn.metrics import mean_absolute_error
print(mean_absolute_error(y_test, y_pred))
from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test, y_pred))
print(pow(mean_squared_error(y_test, y_pred), 0.5))
from sklearn.metrics import r2_score
print(r2_score(y_test, y_pred))
tree.plot_tree(clf)
```



Şəkil 5.13: Verilənlərin proqnozlaşdırılması üçün reqressiya ağacı (max depth = 2 olduqda)

Metrikalar	**Qiymət**
MAE	6.00
MSE	63.21
RMSE	7.95
R2	0.60

Cədvəl 5.15: *Real estate valuation dataset* test bazasının reqressiya ağacı ilə (max depth = 2) proqnozlaşdırılmasının qiymətləndirilməsinin nəticələri

Metrikalar	**Qiymət**
MAE	5.10
MSE	50.78
RMSE	7.13
R2	0.68

Cədvəl 5.16: *Real estate valuation dataset* test bazasının reqressiya ağacı ilə (max depth = 3) proqnozlaşdırılmasının qiymətləndirilməsinin nəticələri

5.4 Ansambl öyrənmə üsulları

Adından da göründüyü kimi ansambl öyrənmə üsulları bir çox modellərin birləşdirilməsini nəzərdə tutur. Bu zaman proqnozlaşdırma üçün individual model deyil kollektiv modellərdən istifadə olunur. Ansambl öyrənmə əsasən iki tip modeldən istifadə edir:

- **Bagging** (qablaşdırma) - öyrədici verilənlərdən müxtəlif örədici nümunə çoxluqları yaradır, son nəticə səsəverməyə əsaslanır, məsələn təsadüfi meşə alqoritmi;

- **Boosting** (yüksəltmə) - Bu üsul ardıcıl modellər yaratmaqla zəif öyrənən modelləri güclü öyrənən modellərə çevirir, nəticədə son model ən yüksək dəqiqliyə malik olur, məsələn, ADABoost, XGBoost;

Təsadüfi meşə alqoritmi

2000-ci illərin əvvəllərində Breiman tərəfindən təklif olunan təsadüfi meşə alqoritmi (Random Forest) sadə və effektiv olan *"parçala və hökm sür"* prinsipinə əsaslanır: verilənlər çoxluğu kiçik hissələrə ayrılır, hər bir hissə üçün proqnozlaşdırıcı ağac qurulur və sonra bu ağaclar birləşdirilir [25].

Təsadüfi meşə alqoritminin populyarlaşması əsasən onların müxtəlif xarakterli proqnozlaşdırma məsələlərinə tətbiq oluna bilməsi və optimallaşdırılması vacib olan hiperparametrlərin az olması ilə bağlıdır. İstifadəsinin sadə olması ilə yanaşı bu üsul dəqiqliyinə görə də populyardır və yüksək ölçülü əlamət çoxluqları və kiçik ölçülü nümunələr bazalarında yaxşı nəticələr göstərir.

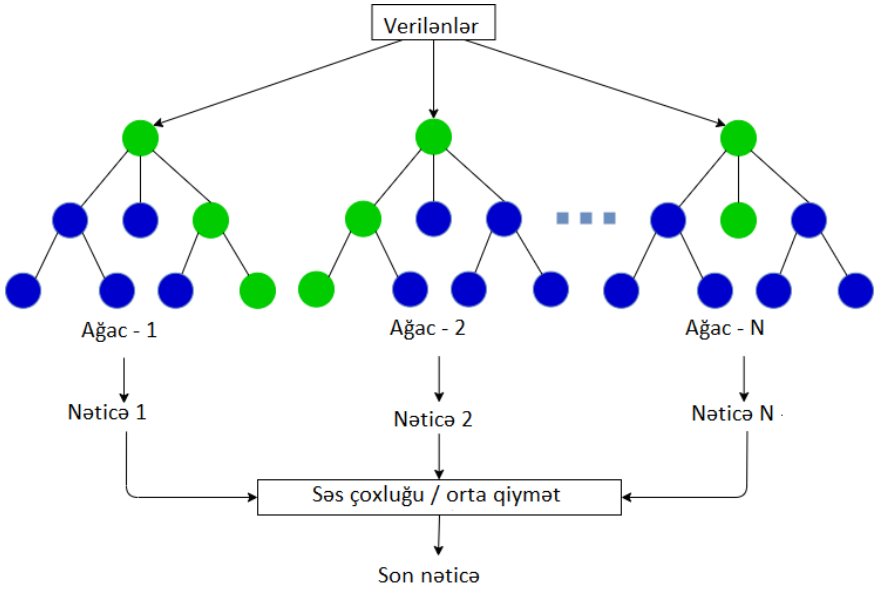
Yuxarıda da qeyd olunduğu kimi təsadüfi meşə alqoritmi həm klassifikasiya, həm də reqressiya məsələlərinə tətbiq oluna bilər. Baxılan alqoritmin iş prinsipini öyrənməzdən öncə *ansambl öyrənmə* texnologiyasını nəzərdən keçirək.

Təsadüfi meşə alqoritmi bagging prinsipi ilə işləyir. Bagging üsulunda hər bir model əvəzlənən üsulla bütöv öyrədici bazadan təsadüfən seçilən öyrədici nümunələr altçoxluqları üçün qurulur. Burada əvəzlənən üsul anlayışı seçilən hər bir altçoxluqda təkrarlanan elementlərin olmasının mümkünlüyünü ifadə edir. Hər bir model bir-birindən asılı olmadan öyrədilir və nəticələri proqnozlaşdırır. Son qərar bütün modellərin qərarları əsasında səsvermə ilə təyin olunur. Aşağıdakı nümunəyə baxaq:

Nümunə: Tutaq ki, verilənlərin emalı üçün n sayda ağacdan ibarət təsadüfi meşə alqoritmi qurulmuşdur. Bu zaman test nümunənin klassifikasiyası və ya proqnozlaşdırılması üçün ilk növbədə verilənlər çoxluğundan seçilən altçoxluqların hər biri üçün n sayda ağacın nəticələri təyin olunur. Əgər klassifikasiya məsələsidirsə, son nəticə səs çoxluğu nəticəsində təyin olunur, yəni ağacların nəticələri içərisində ən çox hansı sinif varsa, verilmiş nümunə həmin sinfə aid olunur. Reqressiya məsələsi olduqda isə son nəticə bütün nəticələrin orta qiyməti kimi təyin olunur (şəkil 5.14).

XGBoost alqoritmi

XGBoost alqoritmi qərar qəbuletmə ağaclarından ibarət dayanıqlı maşın öyrənmə alqoritmidir. Bu alqoritm Vaşinqton Universitetində tədqiqat layihəsi kimi hazırlanmışdır və 2016-cı ildə Tianqi Chen və Carlos Guestrin tərəfindən konfransda təqdim olunmuşdur [26]. XGBoost ekstrim gradiyent yüksəltmə (extrim



Şəkil 5.14: Təsadüfi meşə alqoritmi ilə verilənlərin emalı

	precision	recall	f1-score	support
0	0.86	0.86	0.86	29
1	0.88	0.88	0.88	32
accuracy			0.87	61
macro avg	0.87	0.87	0.87	61
weighted avg	0.87	0.87	0.87	61

Şəkil 5.15: *heart.csv* verilənlər bazasının təsadüfi meşə alqoritmi ilə klassifikasiyasının qiymətləndirilməsi (ağacların sayı 50 olduqda)

gradient boosting) termininin qısaldılmış formasıdır, hər bir iterasiyada performansın qiymətləndirilməsi metrikalarının optimallaşdırılması üçün ən tez enmə və ya digər üsullardan istifadə edərək nəticənin dəqiqliyinin artırılmasına xidmət edir. Əvvəlki fəsillərdə həll etdiyimiz klassifikasiya və reqressiya məsələlərini ansambl öyrənmə üsulları (təsadüfi meşə və XGBoost) ilə öyrədək və nəticələri müqayisə edək (listinq 16, 17).

	precision	recall	f1-score	support
0	0.83	0.86	0.85	29
1	0.87	0.84	0.86	32
accuracy			0.85	61
macro avg	0.85	0.85	0.85	61
weighted avg	0.85	0.85	0.85	61

Şəkil 5.16: *heart.csv* verilənlər bazasının XGBoost alqoritmi ilə klassifikasiyasının qiymətləndirilməsi (iterasiyaların sayı 25 olduqda)

Listinq 16: heart.csv verilənlər bazasının təsadüfi meşə və XGBoost alqoritmi ilə klassifikasiyası

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
data = pd.read_csv('heart.csv')
X = data.drop('target', axis = 1)
y = data['target']
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
rf = RandomForestClassifier(n_estimators = 50, random_state
    = 42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
from sklearn.metrics import classification_report
print (classification_report(y_test, y_pred))
from xgboost import XGBClassifier
xgb = XGBClassifier(n_estimators=25, random_state = 42)
xgb.fit(X_train, y_train)
y_pred = xgb.predict(X_test)
print (classification_report(y_test, y_pred))
```

Listinq 17: Real_ estate.xlsx verilənlər bazasının təsadüfi meşə və XGBoost alqoritmi ilə reqressiyası

```
import pandas as pd
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
data = pd.read_excel('Real_estate.xlsx')
X = data.drop(['No', 'Y house price of unit area', 'X1
    transaction date'], axis = 1)
y = data['Y house price of unit area']
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
rf = RandomForestRegressor(n_estimators = 3, random_state =
     42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
print(mean_absolute_error(y_test, y_pred))
print(mean_squared_error(y_test, y_pred))
print(pow(mean_squared_error(y_test, y_pred), 0.5))
print(r2_score(y_test, y_pred))
xgb = XGBRegressor(n_estimators=30, random_state = 42)
xgb.fit(X_train, y_train)
y_pred = xgb.predict(X_test)
print(mean_absolute_error(y_test, y_pred))
print(mean_squared_error(y_test, y_pred))
print(pow(mean_squared_error(y_test, y_pred), 0.5))
print(r2_score(y_test, y_pred))
```

Metrikalar	**Qiymət**
MAE	4.39
MSE	37.46
RMSE	6.12
R2	0.78

Cədvəl 5.17: *Real estate valuation dataset* test bazasının təsadüfi meşə alqoritmi ilə (ağacların sayı = 50) proqnozlaşdırılmasının qiymətləndirilməsinin nəticələri

Şəkil 5.15 və 5.16-dan, həmçinin cədvəl 5.17 və 5.18-dən göründüyü kimi təsadüfi meşə və XGBoost alqoritmlərinin tətbiq olunması klassifikasiya və reqressiya məsələlərinin keyfiyyətini xeyli artırır. İstər klassifikasiyanın dəqiqliyi üçün alınan yüksək qiymətlər, istərsə də reqressiyanın xətasının kiçik olması ansambl öyrənmə üsullarının praktiki üstünlüyünü sübut edir. Onu da qeyd edək ki, bu paraqrafda tətbiq olunan üsullarda alınan nəticələr sadəcə ağacların sayı üçün uyğun qiymətlər verməklə alınmışdır, lakin həm ansambl öyrənmə üsullarının,

Metrikalar	Qiymət
MAE	4.10
MSE	36.13
RMSE	6.01
R2	0.78

Cədvəl 5.18: *Real estate valuation dataset* test bazasının XGBoost alqoritmi ilə (iterasiyaların sayı = 20) proqnozlaşdırılmasının qiymətləndirilməsinin nəticələri

həm də kitabda bəhs olunan digər üsulların çoxsaylı hiperparametrlərinə müxtəlif qiymətlər verməklə nəticələrin dəqiqliyini artırmaq mümkündür. Bu maşın öyrənmə üsullarının implementasiyadan öncəki son mərhələsidir. Hiperparametrlərin optimallaşdırılması üsulları və onların praktiki tətbiqi növbəti paraqrafda izah olunmuşdur.

5.5 Hiperparametrlərin optimallaşdırılması

Maşın öyrənmə alqoritmləri öyrədici verilənlər əsasında öyrənir, başqa sözlə alqoritmin daxili parametrləri bu verilənlərə əsasən tənzimlənir. Bu tip parametrlər *model parametrləri* və ya sadəcə *parametrlər* adlanır. Buna baxmayaraq maşın öyrənmə alqoritmlərində elə parametrlər var ki, onlar öyrənmə zamanı dəyişmir, daha doğrusu öyrənmə başlamazdan öncə bu parametrlər təyin olunmalıdır. Bu parametrlərə *hiperparametrlər* deyirlər. Model parametrləri giriş verilənlərinin arzuolunan nəticəyə çevrilməsini ifadə edərkən hiperparametrlər modelin strukturunu təyin edir. Maşın öyrənmə modelinin performansı hiperparametrlərin seçilməsinə və onların qiymətinə əsasən kəskin şəkildə dəyişə bilər [27].

Məsələn, qərar qəbuletmə ağacında *maksimum dərinlik* hiperparametri mövcuddur ki, bu hiperparametrə orta qiymət verildikdə çox yaxşı nəticələr alına bilər, lakin bu hiperparametrin yüksək qiyməti modelin performansını azalda bilər. Məhz buna görə də hiperparametrlər ehtiyatla seçilməlidir. Baxılan verilənlər bazası üçün hiperparametrlərin optimallaşdırılması müxtəlif üsullarla həyata keçirilə bilər. Bunlardan biri hiperparametrlərin manual olaraq seçilməsi və uyğun dəqiqliyin hesablanmasıdır. Sonra bu hiperparametrlərin digər qiymətləri təyin olunur və hər bir dəyişiklik üçün uyğun dəqiqlik hesablanır. Hiperparametrlərin bu cür sınama - xəta üsulu ilə təyini hədsiz dərəcədə böyük və vaxt aparan məsələdir. Hiperparametrlərin uyğun qiymətlərini təyin etməyin digər yolu proqram təminatlarının tətbiqində, ədəbiyyatlarda və ya təcrübələrin nəticəsində tövsiyə olunan qiymətlərin istifadəsidir. Bəzən bu qiymətlər bir verilənlər bazası üçün yaxşı nəticələr versə də, bu heç də onların hər zaman yüksək dəqiqlik verməsi demək deyil.

Alternativ olaraq biz hiperparametr optimallaşdırma strategiyalarından istifadə edə bilərik. Bu strategiyalar maşın öyrənmə alqoritminin ümumi xətasını hiperparametrlər axtarış fəzası üzərində minimallaşdırmağa çalışan veriləndən asılı optimallaşdırma üsullarıdır.

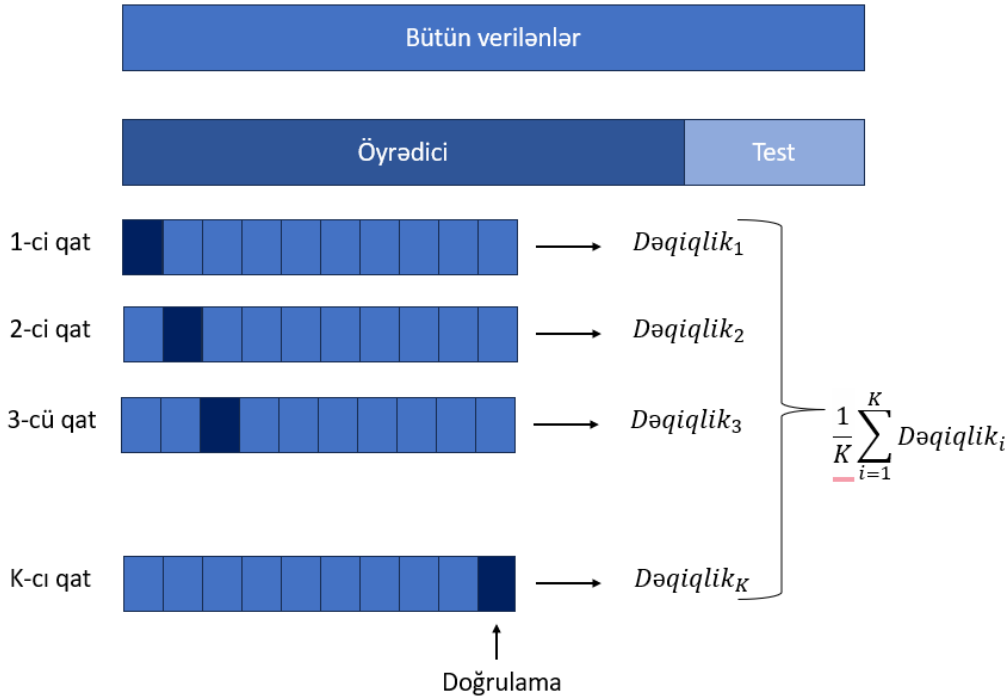
Hiperparametrlərin optimallaşdırılması üsullarından danışmazdan öncə bu prosesin əsas mərhələlərindən biri olan *çarpaz - doğrulama* məsələsinə baxaq. *Çarpaz - doğrulama (cross - validation - CV)* maşın öyrənmə üsullarının dəqiqliyini qiymətləndirmək üçün statik üsuldur. Model öyrədildikdən sonra bu modelin məlum olmayan verilənlər üçün necə işləyəcəyi barədə dəqiq məlumatımız olmur. Lakin modelin implementasiyasından öncə proqnozlaşdırmanın dəqiqliyindən əmin olmaq lazımdır. Maşın öyrənmə üsullarının performansının qiymətləndirilməsi zamanı test üçün məlum olmayan verilənlərdən ibarət çoxluğa ehtiyacımız var. Məlum olmayan verilənlər üçün modelin performansına əsasən biz bu üsulun tam öyrədilmiş, həddindən artıq öyrədilmiş və ya yaxşı ümumiləşdirilmiş olduğunu təyin etmiş olarıq. Əldə olan verilənlərin kifayət qədər olmadığı halda çarpaz doğrulama metodu maşın öyrənmə üsulunun nə dərəcədə effektiv olduğunu test etmək üçün ən əlverişli üsuldur. Çarpaz - doğrulamanın yerinə yetirilməsi üçün verilənlərin altçoxluğu test və doğrulama üçün nəzərdə tutulmalıdır; bu altçoxluq modelin öyrədilməsində istifadə olunmur, növbəti istifadə üçün kənarda saxlanılır. K-qat çarpaz doğrulamanın ən çox istifadə olunan texnikalarından biridir.

K-qat çarpaz doğrulama üsulunda K parametri verilmiş verilənlər bazasının bölündüyü qatların və ya hissələrin sayını göstərir. Qatlardan biri doğrulama üçün saxlanılır, yerdə qalan *K-1* sayda verilən altçoxluğu isə maşın öyrənmə modelinin öyrədilməsi üçün istifadə olunur. K-qatların hər bir qatı doğrulama üçün istifadə olunur və hər biri üçün K sayda dəqiqlik qiyməti alınır. Sonda şəkil 5.17-də göstərildiyi kimi hər bir qat üçün dəqiqliklərin orta qiyməti hesablanır.

5.5.1 Grid axtarış üsulu

Hiperparametrlərin optimallaşdırılması üçün ən çox istifadə olunan ənənəvi üsul grid axtarış üsuludur. Bu üsul hiperparametrlərin bütün mümkün qiymətlərinin Dekart hasilini yaradır. Grid axtarış alqoritmi hiperparametrlərin bütün kombinasiyaları üçün öyrədir; bu proses zamanı performansın ölçülməsi metrikasından istifadə olunur, öyrədici bazada "çarpaz - doğrulama" texnikasından istifadə edilərək üsul qiymətləndirilir. Bu doğrulama texnikasına əsasən öyrənən model verilənlər bazasından bütün məlumatları özündə ehtiva edir. Grid axtarış alqoritminin icra olunması üçün hiperparametrlərin mümkün qiymətlərinin sonlu çoxluğu hazırlanır; daha sonra grid axtarış alqoritmi hiperparametrlərin Dekart hasilinin hər bir kombinasiyası üçün maşın öyrənmə modelini öyrədir. Hər bir kombinasiyanın performansı ya əvvəldən təyin olunmuş doğrulama çoxluğu üzərində, ya da öyrədici bazada daxili çarpaz - doğrulama çoxluqları üzərində qiymətləndirilir. Sonda grid axtarış alqoritminin nəticəsi olaraq doğrulama prosesində ən yüksək performansa malik olan konfiqurasiya təyin olunur.

Grid axtarış alqoritmi ilə təyin olunan hiperparametrlərin optimal qiymətləri modeldə istifadə olunur. Grid axtarış alqoritmləri hiperparametrlərin optimal qiymətlərinin tapılmasına zəmanət verir. Buna baxmayaraq sürətli yığılma və böyük ölçülü verilənlərə gəldikdə bu üsul müəyyən problemlər yaradır. Belə ki, əgər axtarış zamanı k sayda parametr n sayda diskret qiymətə malikdirsə, onda grid



Şəkil 5.17: K-qat çarpaz - doğrulama üsulu

axtarış alqoritminin kompleksliyi exponensial olaraq $O(n^k)$ ölçüsündə artır.

Grid axtarış alqoritminin maşın öyrənmə modellərinin dəqiqliyini artırdığını göstərmək üçün *heart.csv* verilənlər bazasının təsadüfi meşə alqoritmi ilə öyrədilməsi zamanı hiperparametrlərinin optimallaşdırılmasına baxaq. Hiperparametr verilənləri üçün aşağıdakı uyğun qiymətləri götürək:

- maksimum dərinlik üçün - max_depth: [7, 8, 9, 10, 11];
- ən yaxşı bölünmələrin təyin olunması üçün istifadə olunan əlamətlərin maksimum sayı - max_features: [2, 3];
- yarpaq düyün üçün tələb olunan nümunələrin minimal sayı - min_samples_leaf: [3, 4, 5];
- düyünün bölünməsi üçün tələb olunan nümunələrin minimum sayı - min_samples_split: [8, 10, 12];
- meşədəki ağacların sayı - n_estimators: [5, 10, 20, 30, 100].

Daha sonra grid axtarış alqoritminin tətbiqi üçün *sklearn.model_selection* kitabxanasından *GridSearchCV* funksiyasını çağırmaq lazımdır. Listinq 17-nin 18-ci sətrində bu funksiya verilmiş parametrlər əsasında çağırılır və öyrədici bazaya tətbiq edilir. Burada maşın öyrənmə modeli olaraq təsadüfi meşə alqoritmi, hiperparametrlərin uyğun qiymətləri üçün *param_grid* vektoru, çarpaz doğrulama üsulundakı

qatların sayı 3 (cv = 3) götürülür. Bunlardan əlavə sistemdə mövcud olan bütün prosessorlar üzərində paralelləşmənin həyata keçirilməsi üçün *n_jobs = -1* təyin olunur. *verbosity = 2* olması isə proqramın icrasının müfəssəlliyinin 2-ci səviyyəsinə uyğun gəlir, bu səviyyədə hər bir qat üçün hesablama müddəti və hiperparametrin qiyməti ilə yanaşı üsulun dəqiqliyi haqqında da məlumat verilir.

Listinq 17: heart.csv verilənlər bazasının təsadüfi meşə alqoritmi ilə klassifikasiyasının hiperparametrlərinin optimallaşdırılması

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV,
    train_test_split
import pandas as pd
from sklearn.metrics import classification_report
data = pd.read_csv('heart.csv')
X = data.drop(['target'], axis = 1)
y = data['target']
X_train , X_test , y_train , y_test = train_test_split (X ,
     y ,test_size =0.2 , random_state =42)
param_grid = {
    'bootstrap': [True],
    'max_depth': [7, 8, 9, 10, 11],
    'max_features': [2, 3, 4, 5],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12, 14, 16],
    'n_estimators': [5, 10, 20, 30, 100]
}
rf = RandomForestClassifier()
grid_search = GridSearchCV(estimator = rf, param_grid =
    param_grid,
                           cv = 3, n_jobs = -1, verbose = 2)
grid_search.fit(X_train, y_train)
grid_search.best_params_
best_grid = grid_search.best_estimator_
y_pred = best_grid.predict(X_test)
print(classification_report(y_test, y_pred))
```

Grid axtarış üsulunun tətbiqindən sonra hiperparametrlərin optimal qiymətləri üçün aşağıdakı qiymətlər alınmışdır:

- 'max_depth': 8;
- 'max_features': 2;
- 'min_samples_leaf': 4;
- 'min_samples_split': 12;
- 'n_estimators': 30.

	precision	recall	f1-score	support
0	0.90	0.90	0.90	29
1	0.91	0.91	0.91	32
accuracy			0.90	61
macro avg	0.90	0.90	0.90	61
weighted avg	0.90	0.90	0.90	61

Şəkil 5.18: Grid axtarış alqoritmi ilə optimallaşdırma nəticəsində *heart.csv* verilənlər çoxluğunun klassifikasiyasının nəticələri

Nəticədə üsulun dəqiqliyi də əvvəlki paraqraflarda alınan dəqiqliyə nəzərən daha yüksək olmuşdur (şəkil 5.18).

Eyni qayda ilə XGBoost alqoritminin hiperparametrlərinin optimallaşdırılmasına baxaq, bunun üçün metodu əvvəlki fəsillərdə də istifadə olunan *Real_ estate_ validation* verilənlər çoxluğunda sınaqdan keçirək (listinq 18). Burada hiperparametrlərin uyğun qiymətləri aşağıdakı şəkildə verilmişdir:

- maksimum dərinlik üçün max_depth: [4, 5, 6, 7];
- iterasiyaların sayı n_estimators: [20, 25, 30].

Grid axtarış alqoritminin yerinə yetirilməsindən sonra hiperparametrlərin optimal qiymətləri üçün aşağıdakı qiymətlər alınmışdır:

- 'max_depth': 4;
- 'n_estimators': 20.

Grid axtarış üsulunun tətbiqi ilə alınan hiperparametrlərin qiyməti ilə öyrətməni yerinə yetirdikdə test mərhələsində qiymətləndirmə meyarları üçün daha dəqiq qiymətlər alınır (cədvəl 5.19).

Metrikalar	**Qiymət**
MAE	3.93
MSE	30.41
RMSE	5.52
R2	0.82

Cədvəl 5.19: *Real estate valuation dataset* test bazasının hiperparametrlərin optimal qiymətləri ilə XGBoost alqoritmi vasitəsilə proqnozlaşdırılmasının qiymətləndirilməsinin nəticələri

Listinq 18: Real estate values verilənlər bazasının XGBoost alqoritmi ilə proqnozlaşdırılmasının hiperparametrlərinin optimallaşdırılması

```
import pandas as pd
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import GridSearchCV,
    train_test_split
from xgboost import XGBRegressor
data = pd.read_excel('Real_estate.xlsx')
X = data.drop(['No', 'Y house price of unit area', 'X1
    transaction date'], axis = 1)
y = data['Y house price of unit area']
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
reg = XGBRegressor(random_state = 42)
param_grid = {
        'n_estimators': [20, 25, 30],
        'max_depth': [4, 5, 6, 7]
}
grid_search = GridSearchCV(estimator = reg, param_grid =
    param_grid,
                           cv = 10, n_jobs = -1, verbose =
                              2)
grid_search.fit(X_train, y_train)
grid_search.best_params_
best_grid = grid_search.best_estimator_
y_pred = best_grid.predict(X_test)
print(mean_absolute_error(y_test, y_pred))
print(mean_squared_error(y_test, y_pred))
print(pow(mean_squared_error(y_test, y_pred), 0.5))
print(r2_score(y_test, y_pred))
```

5.5.2 Təsadüfi axtarış üsulu

Əvvəlki paraqrafda göstərildiyi kimi grid axtarış alqoritminin tətbiqi zamanı hiperparametrlərin qiymətlərinin bütün kombinasiyaları üçün model öyrədilir və test və ya doğrulayıcı çoxluq üçün model qiymətləndirilir. Bir çox hallarda bu üsul effektiv olmur. Məsələn, 5 parametr üçün 10 fərqli qiymətin yoxlanması 100,000 cəhd tələb edir. Əgər 10 qat doğrulayıcı çoxluqdan istifadə olunursa, bu zaman 1,000,000 öyrətmə və 1,000,000 testə ehtiyac var. Bu da öz növbəsində həm zamana, həm də resurslara görə xərcləri artırır. Bunun əksinə olaraq təsadüfi axtarış alqoritmi axtarış fəzasındakı bütün nümunələr üçün deyil, müəyyən ehtimal paylanması əsasında qiymətləndirməni yerinə yetirir. Qısa olaraq bu üsulda modelin optimal variantını tapmaq üçün hiperparametrlərin təsadüfi kombinasiyalarından istifadə olunur. Məsələn, bütün mümkün 100,000 kombinasiyanın içərisindən 1000 nümunə üçün modelin keyfiyyəti qiymətləndirilir.

	precision	recall	f1-score	support
0	0.89	0.86	0.88	29
1	0.88	0.91	0.89	32
accuracy			0.89	61
macro avg	0.89	0.88	0.88	61
weighted avg	0.89	0.89	0.89	61

Şəkil 5.19: Təsadüfi axtarış alqoritmi ilə hiperparametrlərin optimallaşdırılması nəticəsində *heart.csv* verilənlər bazasının klassifikasiyasının nəticəsi

Təsadüfi axtarış üsulunda hesablamaların sayı əvvəldə qeyd olunmalıdır, buna görə də n qiymətləndirmədən ibarət təsadüfi axtarış alqoritminin kompleksliyi $O(n)$ olar. İndi isə təsadüfi axtarış alqoritmini klassifikasiya və reqressiya məsələlərinə tətbiq edək (listinq 19 və listinq 20).

Listinq 19: heart.csv verilənlər bazasının təsadüfi meşə alqoritmi ilə proqnozlaşdırılması zamanı hiperparametrlərinin təsadüfi axtarış üsulu ilə optimallaşdırılması

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import RandomizedSearchCV,
    train_test_split
import pandas as pd
from sklearn.metrics import classification_report
data = pd.read_csv('heart.csv')
X = data.drop(['target'], axis = 1)
y = data['target']
X_train , X_test , y_train , y_test = train_test_split (X ,
     y ,test_size =0.2 , random_state =42)
parameters = {
     'bootstrap': [True],
    'max_depth': [7, 8, 9, 10, 11],
    'max_features': [2, 3, 4, 5],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12, 14, 16],
    'n_estimators': [5, 10, 20, 30, 100]
}
random_search = RandomizedSearchCV(RandomForestClassifier()
    , parameters, cv = 3, n_jobs = -1, verbose = 2)
random_search.fit(X_train, y_train)
random_search.best_params_
best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)
print(classification_report(y_test, y_pred))
```

Nəticədə təsadüfi meşə alqoritminin hiperparametrləri üçün aşağıdakı optimal qiymətlər

- max_depth=11,
- max_features=3,
- min_samples_leaf=4,
- min_samples_split=12,
- n_estimators=30,

XGBoost alqoritminin hiperparametrləri üçün isə

- max_depth = 4,
- n_estimators = 25

qiymətləri alınır. Üsulların effektivliyinin ölçülməsi meyarları üçün alınan qiymətlər isə şəkil 5.19 və cədvəl 5.20-də göstərilmişdir. Metrikalar üçün alınan qiymətlərdən də görünür ki, təsadüfi axtarış alqoritmi ilə hiperparametrlərin optimallaşdırılması nəticəsində alınan qiymətlər grid axtarış alqoritminin tətbiqi nəticəsində alınan qiymətlərlə eynidir. Klassifikasiyanın nəticələrinin nisbətən aşağı olmasına baxmayaraq təsadüfi axtarış alqoritmi proqramın icra müddəti baxımından xeyli irəlidədir. Belə ki, orta statistik parametrlərə malik kompüterdə eyni parametrlərlə klassifikasiya məsləsi üçün grid axtarış alqoritminin icrası 42 saniyə, təsadüfi axtarış alqoritminin icrası isə 0.2 saniyə ərzində yekunlaşmışdır. Həmçinin eyni kompüterdə reqressiya məsələsinin həlli zamanı grid axtarış alqoritminin icrasına 3.5 saniyə, təsadüfi axtarış üsulunun icrasına isə 1 saniyə sərf olunmuşdur.

Metrikalar	**Qiymət**
MAE	3.92
MSE	30.98
RMSE	5.57
R2	0.82

Cədvəl 5.20: Təsadüfi axtarış üsulu ilə hiperparametrlərin optimallaşdırılması nəticəsində *Real estate valuation dataset* test bazasının proqnozlaşdırılmasının nəticəsi

Qeyd etmək lazımdır ki, hiperparametrlərin optimallaşdırılması üçün grid axtarış alqoritmi və ya təsadüfi axtarış alqoritmi yalnız ansambl öyrənmə üsullarına deyil, digər maşın öyrənmə üsullarına da tətbiq oluna bilər.

Listinq 20: Real estate values verilənlər bazasının XGBoost alqoritmi ilə proqnozlaşdırılmasının hiperparametrlərinin təsadüfi axtarış üsulu ilə optimallaşdırılması

```
import pandas as pd
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import RandomizedSearchCV,
    train_test_split
from xgboost import XGBRegressor
data = pd.read_excel('Real_estate.xlsx')
X = data.drop(['No', 'Y house price of unit area', 'X1
    transaction date'], axis = 1)
y = data['Y house price of unit area']
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
param_grid = {
        'n_estimators': [20, 25, 30],
        'max_depth': [4, 5, 6, 7]
}
random_search = RandomizedSearchCV(XGBRegressor(
    random_state = 42), param_grid,
                           cv = 10, n_jobs = -1, verbose =
                              2)
random_search.fit(X_train, y_train)
random_search.best_params_
best_model = random_search.best_estimator_
y_pred = best_model.predict(X_test)
print(mean_absolute_error(y_test, y_pred))
print(mean_squared_error(y_test, y_pred))
print(pow(mean_squared_error(y_test, y_pred), 0.5))
print(r2_score(y_test, y_pred))
```

Çalışmalar

1. Aşağıda verilmiş cədvələ əsasən *Yaş həddi* atributu üçün informasiya qazancını hesablayın:

Yaş həddi	**Gender**	**Sinif**	**Nəticə**
Gənc	kişi	1	Xeyr
Yaşlı	qadın	3	Bəli
Gənc	qadın	2	Bəli
Yaşlı	kişi	3	Bəli
Gənc	kişi	2	Xeyr
Yaşlı	qadın	1	Bəli

2. 1-ci tapşırıqda verilmiş müşahidələrə əsasən qərar qəbuletmə ağacını qurun.

3. Aşağıda verilmiş cədvələ əsasən *Yaş* atributu üçün informasiya qazancını hesablayın:

Şəxs	**Yaş**	**Gəlir**	**Nəticə**
1	25	40	Bəli
2	30	60	Bəli
3	35	30	Xeyr
4	20	25	Xeyr
5	28	50	Bəli
6	45	70	Bəli

4. 3-cü tapşırıqdakı verilənlər üçün Hunt alqoritmi ilə maksimum dərinliyi 2 olan qərar qəbuletmə ağacını qurun.

5. *Rəng* atributunun (*Qırmızı, Yaşıl, Göy*) qiymətləri və asılı parametrin (*Bəli, Xeyr*) qiymətləri üçün aşağıdakı altçoxluqlar verilmişdir:
(Qırmızı) altçoxluğu üçün: Bəli: 4; Xeyr: 2;
(Yaşıl) altçoxluğu üçün: Bəli: 1; Xeyr: 3;
(Göy) altçoxluğu üçün: Bəli: 2; Xeyr: 1;
Rəng atributu üçün informasiya qazancını hesablayın.

6. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ iris* funksiyasından istifadə edərək süsən güllərinin klassifikasiyası üçün qərar qəbuletmə ağacını qurun. Real qiymətlərlə qarşılaşdırıb modelin dəqiqliyini hesablayın.

7. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ diabetes* funksiyasından istifadə edərək xəstələrin verilənlərini yükləyin və bu obyektlərin klassifikasiyası üçün qərar qəbuletmə ağacını qurun. Real qiymətlərlə qarşılaşdırıb modelin dəqiqliyini hesablayın.

8. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ boston* funksiyasından istifadə edərək evlərin qiymətlərinin proqnozlaşdırılması üçün qərar qəbuletmə ağacını qurun. Real qiymətlərlə qarşılaşdırıb modelin dəqiqliyini hesablayın.

9. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ diabetes* funksiyasından istifadə edərək xəstələrin verilənlərini yükləyin və bu obyektlərin klassifikasiyası üçün təsadüfi meşə alqoritmini tətbiq edin. Modelin dəqiqliyinə görə hiperparametrlərin optimallaşdırılmasını həyata keçirin.

10. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ boston* funksiyasından istifadə edərək evlərin qiymətlərinin proqnozlaşdırılması üçün XGBoost alqoritmini tətbiq edin. Orta kvadratik xtaya sasn hiperparametrlrin optimallaşdırılmasını Grid axtarış alqoritmi ilə yerin yetirin.

11. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ boston* funksiyasından istifadə edərək evlərin qiymətlərinin proqnozlaşdırılması üçün XGBoost alqoritmini tətbiq edin. Xətanın mütləq qiymətinə əsasən hiperparametrlərin optimallaşdırılmasını təsadüfi axtarış alqoritmi ilə yerin yetirin.

12. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ diabetes* funksiyasından istifadə edərək xəstələrin verilənlərini yükləyin və bu obyektlərin klassifikasiyası üçün k ən yaxın qonşu alqoritmini tətbiq edin. Modelin dəqiqliyinə görə $k = \{1, 2, 3, 4, 5, 6, 7, 8\}$ olduqda Grid axtarış və təsadüfi axtarış üsulları ilə hiperparametrlərin optimallaşdırılmasını həyata keçirin.

13. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ iris* funksiyasından istifadə edərək süsən güllərinin klassifikasiyası üçün logistik reqressiya üsulunu tətbiq edin. Modelin dəqiqliyinə əsasən üsulun hiperparametrlərini Grid axtarış və təsadüfi axtarış üsulları ilə optimallaşdırın.

14. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ diabetes* funksiyasından istifadə edərək xəstələrin verilənlərini yükləyin və bu obyektlərin klassifikasiyası üçün k ən yaxın qonşu alqoritmini tətbiq edin. Modelin F1 qiymətinə görə $k = \{1, 2, 3, 4, 5, 6, 7, 8\}$ olduqda Grid axtarış və təsadüfi axtarış üsulları ilə hiperparametrlərin optimallaşdırılmasını həyata keçirin.

15. *Python* mühitində *sklearn.datasets* kitabxanasının *load_ iris* funksiyasından istifadə edərək süsən güllərinin klassifikasiyası üçün logistik reqressiya üsulunu tətbiq edin. Modelin F1 qiymətinə əsasən üsulun hiperparametrlərini Grid axtarış və təsadüfi axtarış üsulları ilə optimallaşdırın.

Fəsil 6

Klasterləşdirmə

Müəllimsiz maşın öyrənməsi kimi də tanınan müəllimsiz öyrənmə, adlandırılmamış yəni təsnif edilməmiş məlumat dəstələrini təhlil etmək və qruplaşdırmaq üçün maşın öyrənməsi alqoritmlərindən istifadə edir. Bu alqoritmlər insan müdaxiləsinə ehtiyac olmadan gizli nümunələri və ya məlumat qruplarını aşkar edir. Məlumatlardakı oxşarlıqları və fərqləri aşkar etmək qabiliyyəti bu üsulları məlumatların kəşfiyyat xarakterli təhlili, çarpaz satış strategiyaları, müştərilərin seqmentasiyası və təsvirlərin tanınması məsələləri üçün ideal həll yolu edir.

Müəllimsiz öyrənmə modelləri üç əsas məqsəd üçün istifadə olunur - qruplaşma, əlaqələndirmə və ölçülərin azaldılması. Bu kitabda əsasən sadə klasterləşdirmə üsulları təsvir və tətbiq olunmuşdur.

Klasterləşdirmə obyektlər və onlar arasında əlaqəni təsvir edən informasiyaya əsasən obyektləri qruplara ayırır. Burada məqsəd ondan ibarətdir ki, eyni qrupda olan obyektlər ya oxşar, ya da əlaqəli olmalı, digər qruplardakı obeyktlərdən isə mümkün qədər fərqli olmalıdırlar. Qruplarda oxşarlıq və ya bircinslik və qruplararası fərq nə qədər çox olarsa, klasterləşdirmə o qədər yaxşı sayılır [28].

Klasterləşdirmə verilən obyektləri qruplara bölən digər üsullarla oxşardır. Məsələn, klasterləşdirmə obyektlərə adlar mənimsədərək siniflərə bölən klassifikasiya alqoritmi kimi düşünülə bilər. Lakin klasterləşdirmə zamanı bu qrupların adları yalnız atribut verilənlərindən alınır. Əksinə olaraq klassifikasiya müəllimli öyrətmə üsuludur, belə ki, yeni, adlandırılmamış verilənlərin sinifləri əvvəldən siniflərə ayrılmış öyrədici obyektlər vasitəsilə qurulmuş model əsasında təyin olunur. Məhz bu səbəbdən, klasterləşdirmə **müəllimsiz öyrətmə** üsullarına aid edilir.

Xüsusilə multimedia verilənlərinin emal olunmasında müəllimsiz öyrətmə çox önəmlidir, belə ki, bu hallarda aid olduğu siniflərin adları mövcud olmayan verilənlərin qruplaşdırılması və ya hissələrə ayrılması mühüm əhəmiyyət kəsb edir.

6.1 k-means klasterləşdirmə

Ümumiyyətlə, müxtəlif tip klasterləşdirmə üsullarını fərqləndirirlər. Bu bölmədə əsasən verilənləri hissələrə ayıran klasterləşdirmə üsullarından bəhs edilir. Bu

tip klasterləşdirmə üsulları verilənlər çoxluğunu $C = \{C_1, C_2, ..., C_k\}$ kimi işarə olunan k sayda klasterlərə ayırır. Belə üsullar başlanğıc həllin iterativ şəkildə optimallaşdırılması ilə qlobal məqsəd funksiyasının lokal approksimasiyasının təyin olunması məqsədini güdür.

k-means klasterləşdirmə üsulu ən çox istifadə olunan verilənləri hissələrə ayırma üsuludur. Bu üsul verilən obyektlərin və k klaster nümayəndələri arasındakı məsafəni lokal olaraq minimallaşdıraraq iterativ sxemi həyata keçirir. *Mərkəz (centroid)* adlanan hər bir nümayəndə verilmiş klasterə aid edilən bütün obyektlərin vektorlarının orta qiymətindən ibarət vektorla təyin olunur.

Alqoritmin klassik versiyasında məsafə Evklid metrikası ilə ifadə olunduğundan klasterləşdirmə prosesinin məqsədi klaster mərkəzləri $\{\mu_1, \mu_2, ..., \mu_k\}$ ilə obyektlər arasındakı orta kvadratik xətanın minimallaşdırılmasından ibarətdir:

$$SSE(C) = \sum_{c=1}^{k} \sum_{x_i \in C_c} \|x_i - \mu_c\|^2, \tag{6.1}$$

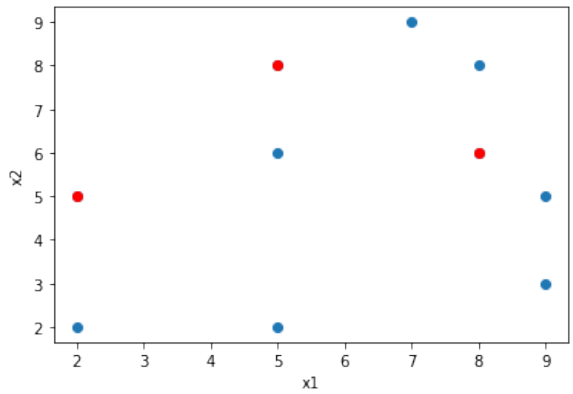
burada $\mu_c = \frac{\sum_{x_i \in C_c} x_i}{|C_c|}$. Əsas alqoritmin çoxlu sayda modifikasiyalarının olmasına baxmayaraq klasterləşdirmədə ən çox istifadə olunan metod (6.1)-də göstərilən iki addımlı prosesdən ibarətdir. Birinci addımda hər bir obyekt mərkəzinə ən yaxın olduğu klasterə aid edilir. Bütün obyektlər emal olunduqdan sonra mərkəz vektorları yenilənir, nəticədə obyektlərin klasterlərə yenidən aid edilməsi birinci addımdakı kimi həyata keçirilir. Bu iterativ proses verilmiş sonlandırma meyarı ödənilənə qədər davam etdirilir. Çox hallarda təsvir olunmuş üsul iki ardıcıl iterasiyada klasterlərdəki obyektlərin tərkibinin dəyişmədiyi halda sonlandırılır. Alternativ olaraq alqoritm (6.1) düsturunda verilmiş orta kvadratik xətanın qiymətinin verilmiş xətanın qəbul olunan qiymətindən kiçik olduqda da sonlandırıla bilər.

6.1.1 Obyektlərin k-means üsulu ilə klasterləşdirilməsi. Nümunə

İlk öncə k-means klasterləşdirmə üsulunu anlamaq üçün iki ölçülü fəzada verilmiş aşağıdakı nöqtələrin qruplaşdırılması məsələsini həll edək:

$$(5;8), (8;2), (2;2), (5;6), (9;3), (5;2), (8;6), (2;5), (8;8), (9;5), (7;9).$$

Hesablamalara başlamazdan öncə $k = 3$ qəbul edək və təsadüfi olaraq 3 klasterə mərkəz nöqtə seçək. Birinci mərhələdə (5, 8), (8, 6) və (5, 2) nöqtələrini klasterlərin mərkəzi kimi götürək (şəkil 6.1).
Daha sonra seçilmiş mərkəzlər və obyektlər arasındakı Evklid məsafəsini hesablayaq (Cədvəl 6.1). Cədvəldən istifadə edərək hər bir sətirdə ən kiçik məsafələri xüsusi qeyd edək, nöqtələri onlara ən yaxın olan mərkəzlərin klasterlərinə aid edək və yeni mərkəzləri obyektlərin orta qiyməti kimi təyin edək. Cədvəldən göründüyü kimi 1-ci klasterə 1, 3, 7 və 10, 2-ci klasterə 4, 6, 8 və 9, 3-cü klasterə isə 2 və 5 nöqtələri



Şəkil 6.1: Birinci iterasiyada təsadüfi seçilən klaster mərkəzləri

N	x_1	x_2	Mərkəz 1 ilə məsafə	Mərkəz 2 ilə məsafə	Mərkəz 3 ilə məsafə
1	5	8	**0**	3.6	6.9
2	2	2	6.7	7.2	**3**
3	5	6	**2**	3	4.9
4	9	3	6.4	**3.2**	4.4
5	5	2	6	5	**0**
6	8	6	3.6	**0**	5.7
7	2	5	**4.2**	6.1	4.9
8	8	8	3	**2**	7.6
9	9	5	5	**1.4**	5.6
10	7	9	**2.2**	3.2	8.2

Cədvəl 6.1: Verilmiş nöqtələrin klasterləşdirilməsi (1-ci iterasiya)

aiddir. Bunlar nəzərə alsaq yeni klaster mərkəzlərini aşağıdakı şəkildə hesablaya bilərik:

$$centroid1 = (\frac{5+5+2+7}{4}; \frac{8+6+5+9}{4}) = (4.8; 5.8)$$

$$centroid2 = (\frac{9+8+8+9}{4}; \frac{3+6+8+5}{4}) = (8.5; 5.5)$$

$$centroid3 = (\frac{2+5}{2}; \frac{2+2}{2}) = (3.5; 2)$$

Yenidən tapılmış mərkəzlərlə obyektlər arasındakı Evklid məsafəsini taparaq obyektləri yenidən klasterləşdirək (cədvəl 6.2). Cədvəldən göründüyü kimi artıq 2-ci klasterə daxil olan obyektlərin sayı artmış, 1-ci klasterə daxil olan obyektlərin sayı isə azalmışdır, 3-cü klasterin tərkibi isə dəyişməmişdir. Növbəti addımda isə yeni mərkəzlər klasterlərə daxil olan obyektlərin orta qiymət vektoru şəklində hesablanır və bu iki addım iterativ olaraq davam etdirilir.

N	x_1	x_2	Mərkəz 1 ilə məsafə	Mərkəz 2 ilə məsafə	Mərkəz 3 ilə məsafə
1	5	8	**2.2**	4.3	7.1
2	2	2	4.7	7.4	**1.5**
3	5	6	**0.3**	3.5	5.1
4	9	3	5.1	**2.5**	5.8
5	5	2	3.8	4.9	**1.5**
6	8	6	3.2	**0.7**	6.7
7	2	5	**2.9**	6.5	4.2
8	8	8	3.9	**2.5**	8.3
9	9	5	4.3	**0.7**	6.7
10	7	9	3.9	**3.8**	8.7

Cədvəl 6.2: Verilmiş nöqtələrin yeni mərkəzlərə görə klasterləşdirilməsi (2-ci iterasiya)

K-means klasterləşdirmə üsulunda modeli qiymətləndirmək üçün düstur (6.1) - də verilən məqsəd funksiyasının qiymətini hesablamaq lazımdır. Göstərilən nümunənin 2-ci iterasiyasında orta kvadratik xətanı hesablayaq, bu iterasiyanın nəticələrinə əsasən 1-ci klasterə 1, 3 və 7 nöqtələri, 2-ci klasterə 4, 6, 8, 9 və 10 nöqtələri, 3-cü klasterə isə 2 və 5 nöqtələri aiddir (şəkil 6.2). Yeni klaster mərkəzlərinə uyğun vektorları aşağıdakı şəkildə hesablayaq:

$$centroid1 = (\frac{5+5+2}{3}; \frac{8+6+5}{3}) = (4; 6.3)$$

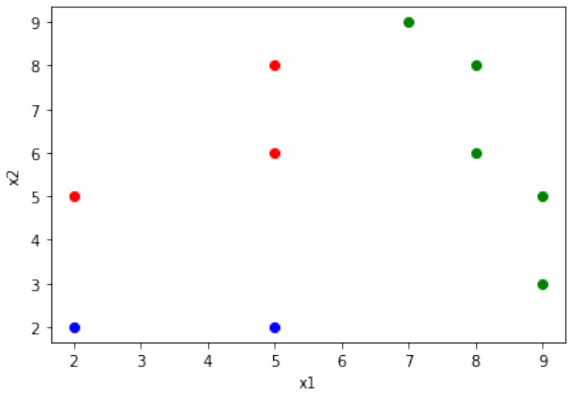
$$centroid2 = (\frac{9+8+8+9+7}{5}; \frac{3+6+8+5+9}{5}) = (8.2; 6.2)$$

$$centroid3 = (\frac{2+5}{2}; \frac{2+2}{2}) = (3.5; 2)$$

Cədvəl 6.2-də qalın şriftlərlə verilmiş obyektlərin aid olduqları klaster mərkəzlərinə olan məsafələri toplasaq aşağıdakı xəta qiymətini alarıq:

$$SSE = 5.4 + 10.2 + 3 = 18.6$$

K-means klasterləşdirmə üsulunun alqoritmi çox sadə olduğundan yalnız *Python numpy* kitabxanasının köməyi ilə alqoritmin proqramını tərtib etmək olar. Tutaq ki, verilmiş sahədə müxtəlif şirkətlərin verilənləri emal edilir (cədvəl 6.3).
Cədvəl 6.3-də verilmiş 4 parametrə (*müştərilərin sayı*, *qaytarma faizi*, *qazanc*, *yaş*) əsasən 3 klasterə ayrılmasını nəzərdən keçirək (verilənlərin hamısını verilmiş linkdən yükləyə bilərisiniz [29]).



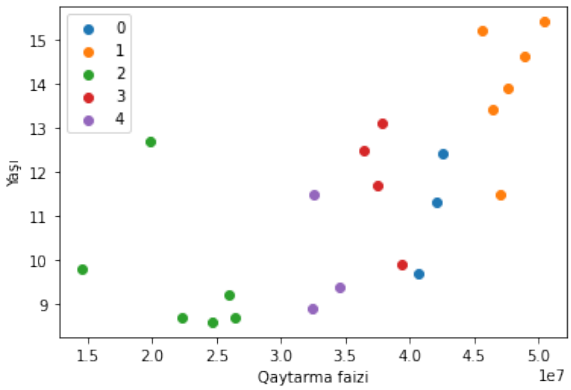
Şəkil 6.2: İkinci iterasiyadan sonra obyektlərin qruplaşdırılması

Şirkət	Müştərilərin sayı	Qaytarma faizi	Qazanc	Yaşı
A	150	15.4	50400200	18
B	144	11.3	42100650	15
C	120	9.9	39440420	12
D	110	12.5	36500520	16
E	100	9.7	40650005	10
F	99	15.2	45660230	12
G	56	9.2	25978080	8

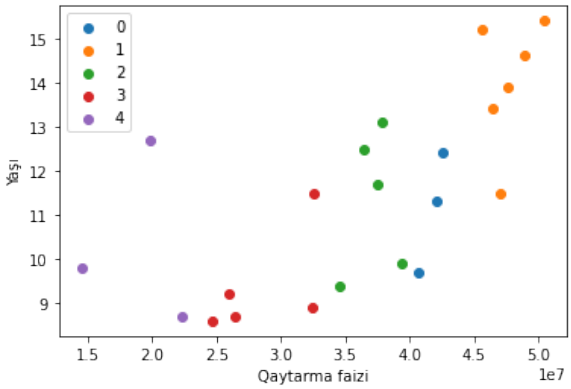
Cədvəl 6.3: Araşdırılan şirkətlər haqqında məlumat

Listinq 21: k-means klasterləşdirmə üsulu ilə verilənlərin qruplaşdırılması üçün funksiya və nəticələrin vizuallaşdırılması

```
import numpy as np
from scipy.spatial.distance import cdist
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import pandas as pd
data = pd.read_csv('KMeansClustering.csv')
X = data.drop(['Cluster', 'Company'], axis = 1)
def kmeans(x,k, no_of_iterations):
    idx = np.random.choice(len(x), k, replace=False)
    centroids = x[idx, :]
    distances = cdist(x, centroids ,'euclidean')
    points = np.array([np.argmin(i) for i in distances])
    for _ in range(no_of_iterations):
        centroids = []
        for idx in range(k):
            temp_cent = x[points==idx].mean(axis=0)
            centroids.append(temp_cent)
        centroids = np.vstack(centroids)
        distances = cdist(x, centroids ,'euclidean')
        points = np.array([np.argmin(i) for i in distances
            ])
    return points
label = kmeans(X,5,10000)
u_labels = np.unique(label)
```



Şəkil 6.3: Klasterləşmənin nəticələri, $k = 5$, *iterasiyaların sayı* $= 100$ olduqda

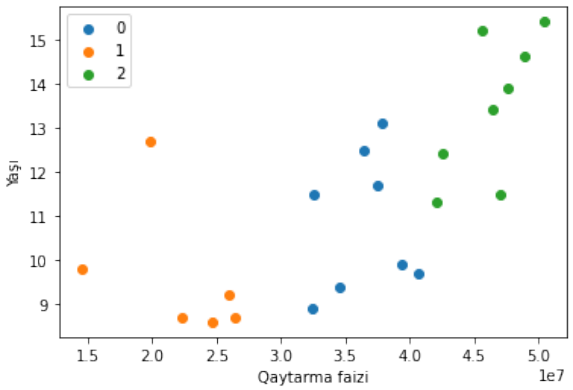


Şəkil 6.4: Klasterləşmənin nəticələri, $k = 5$, *iterasiyaların sayı* $= 20000$ olduqda

İlk olaraq lazım olan kitabxanaları çağıraq və verilənləri yükləyək. k-means klasterləşdirməni həyata keçirmək üçün verilənlər, klasterlərin sayı və iterasiyaların sayı ilə təyin olunan funksiyanı quraq (listinq 21). Bu funksiya ilk öncə ilkin mərkəzləri təyin edir, obyektlərlə mərkəzlər arasındakı məsafələri hesablayır və obyektləri uyğun klasterlərə mənimsədir. Daha sonra bu iki addım iterativ olaraq təkrarlanır, sadə-cə burada mərkəzlər təsadüfi olaraq deyil, hər iterasiyada klaster obyektlərinin orta qiymət vektoru kimi təyin olunur (Listinq 21, sətir 8 - 21).

K-means klasterləşdirmə funksiyasını göstərilmiş verilənlər üçün müxtəlif klaster qiymətləri və iterasiya sayları ilə tətbiq edib eksperiment aparaq və nəticələri vizuallaşdıraq (şəkil 6.3 - 6.5). Şəkillərdən göründüyü kimi iterasiyaların sayı artdıqca klasterləşmənin dəqiqliyi də artır, amma klasterlərin sayı haqqında belə bir fikir yürütmək qeyri-mümkündür. Belə ki, baxılan verilənlər üçün klasterlərin optimal sayını dəqiq təyin etmək mümkün deyil. Bunu yenə də k ən yaxın qonşu üsulunda olduğu kimi müxtəlif sayda klasterlər üçün alqoritmi yerinə yetirib xətanı hesablayaraq hansı qiymətin optimal olduğunu təyin etmək olar.

Qeyd edək ki, şəkillərdə klasterləşmənin nəticəsi yalnız iki parametrə görə ve-



Şəkil 6.5: Klasterləşmənin nəticələri, $k = 3$, *iterasiyaların sayı* $= 20000$ olduqda

rilsə də, alqoritm verilənlərin bütün parametrlərinə görə qruplaşdırmanı həyata keçirir. Baxılan proqram koduna orta kvadratik xətanın hesablanması üçün lazım olan əmrləri da daxil etməklə eksperimentlər zamanı xətanın qiymətini hesablamaq mümkündür. Həmçinin k-means klasterləşdirmə alqoritmini *sklearn.cluster* kitabxanasının *KMeans* funksiyası ilə də həyata keçirmək olar.

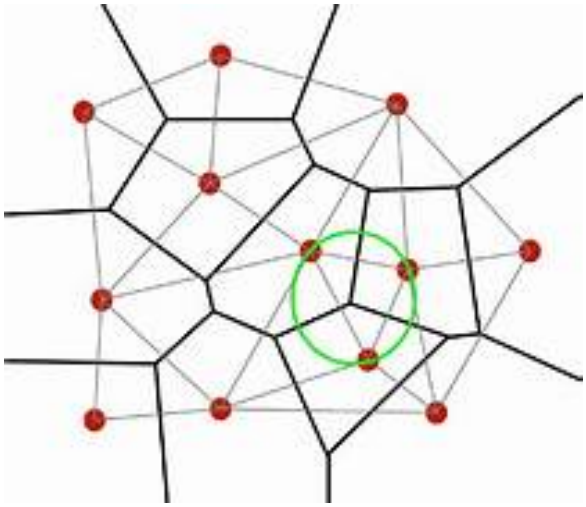
Voronoi diaqramı

Tutaq ki, m - ölçülü Evklid fəzasında n sayda nöqtələr verilmişdir: $S = \{p_1, p_2, ..., p_n\}$. Bu halda S Voronoi diaqramı bu fəzanı hər bir xanasında yalnız bir nöqtə olan n hissəyə bölünməsi kimi təyin olunur. S fəzasının Voronoi diaqramı *V(S)* kimi işarə olunur. Tutaq ki, fəzada a və b nöqtələri arasındakı məsafə *d(a, b)* şəklində təyin olunur. Onda tərifə əsasən u nöqtəsi S fəzasından istənilən p_j nöqtəsi və $j \neq i$ üçün yalnız və yalnız $d(u, p_i) < d(u, p_j)$ olarsa, p_i nöqtəsinə uyğun xanada olar. Voronoi diaqramına nümunə şəkil 6.6 - da göstərilmişdir. ν Voronoi təpəsi və S fəzası üçün ən böyük boş dairə təyin edək, belə ki, bu dairənin içərisində S fəzasının heç bir nöqtəsi yoxdur. Bu dairəni $CirS(\nu)$ ilə işarə edək. Voronoi təpələrinin belə bir xassəsi var, q təpəsi yalnız o vaxt *Vor(S)* Voronoi diaqramının təpəsi hesab olunur ki, $CirS(q)$ dairəsinin sərhəddində S fəzasının üç və ya daha çox nöqtəsi olsun. n nöqtəli Voronoi diaqramının maksimum $2n - 5$ sayda Voronoi təpələri olar [30].

6.2 İyerarxik klasterləşdirmə

İyerarxik klasterləşdirmə obyektlərin ağacşəkilli təsviri - dendroqram əsasında həyata keçirilir. Bunlara nümunə olaraq toplayıcı klasterləşdirmə (agglomerative clustering), bölücü klasterləşdirmə (divisive clustering) və hissəli ən kiçik kvadratlar üsulunu göstərmək olar [31].

İyeraxik klasterləşdirmənin hissələrə bölən klasterləşdirməyə nisbətən müəyyən üstünlükləri vardır. Məsələn, burada klasterlərin sayının təyin olunması kimi tələb



Şəkil 6.6: Voronoi diaqramı

yoxdur. Buna baxmayaraq iyerarxik klasterləşdirməni yerinə yetirərkən müxtəlif tipli suallar yaranır. Məsələn, çoxlu sayda mövcud klasterləşdirmə üsullarının içərisində hansı üsulu seçmək lazımdır? Klasterləşdirmə metodu seçildikdən sonra mövcud əlamətlər dendroqrama necə təsir göstərir? Dendroqramdakı hər bir nümunənin yeri əlamətlərlə necə ifadə oluna bilər?

İyerarxik klasterləşdirmə verilənlərin üzərindən ağac, adətən binar ağac qurur. Bu ağacın yarpaqları individual verilənlər, kökü isə bütün verilənləri özündə saxlayan yeganə klasterdən ibarət olur. Kök və yarpaqlar arasında verilənlərin alt-çoxluqlarını saxlayan aralıq klasterlər yerləşir. İyerarxik klasterləşdirmənin əsas ideyası ondan ibarətdir ki, "klasterlərin klasterlərini" təyin edərək yuxarıya doğru ağacı qurmaqdır. Bu cür ağacın qurulmasının iki konseptual əsası mövcuddur. İyerarxik toplayıcı klasterləşdirmə (Hierarchical agglomerative clustering) aşağıdan, hər bir obyektin tək klasterdə yerləşdiyi hissədən başlayır və qrupları bir-biri ilə birləşdirir. Bölücü klasterləşdirmə (Divisive clustering) bütün verilənlərin yerləşdiyi bir klasterlə başlayır və hər bir obyektin tək klasterdə yerləşməsinə qədər klasterləri bölür.

6.2.1 Toplayıcı klasterləşdirmə

Toplayıcı klasterləşdirmənin əsas alqoritmi klasterlərin aktiv çoxluğu, bütün elementlərin ayrı-ayrı klasterlərdə olduğu hal ilə başlayır və hər bir mərhələdə birləşdirəcəyi qrupları təyin edir. İki klaster birləşdirildikdə hər biri aktiv çoxluqdan silinir və onların vəhdəti aktiv çoxluğa daxil edilir. Bu proses aktiv çoxluqda yalnız bir klaster qalana qədər davam etdirilir.

Toplayıcı klasterləşdirmədə tək birləşdirici meyar olaraq bütün mümkün cütlər

içərisində seçilmiş metrika ilə ən kiçik məsafə götürülür:

$$Dist(\{x_n\}_{n=1}^{N}, \{y_m\}_{m=1}^{M}) = \min_{n,m} \|x_n - y_m\| \tag{6.2}$$

Bu meyar qrupu onun ən yaxın qonşusu ilə birləşdirir, bunu aşağıdakı şəkildə izah etməyə çalışaq. Fərz edək ki, verilənlər qrafın təpələridir, düsturda verilən birləşdirmə meyarını tətbiq edərkən 2 təpə arasında bu düsturu minimallaşdıran tili qrafa daxil edirik. Qrupun mövcud olan elementlərini bir də daxil etmədiyimizə görə bu qrafda ilgəklər və qapalı zəncirlər heç vaxt olmur. Beləliklə, alqoritmin sonunda ağac əldə edirik.

İyerarxik klasterləşdirmədə ən böyük problemlərdən biri budur ki, bu üsul bizə nə qədər klaster olduğu və ya klasterləri təyin etmək üçün dendroqramı harada kəsmək lazım olduğu haqqında məlumat vermir. Verilənləri qruplara bölmək üçün iyerarxik ağacı verilmiş hündürlükdə kəsmək olar. Digər klasterləşdirmə üsullarında olduğu kimi burada da klasterlərin sayı ya eksperimentlərin, ya da vizual təhlilin nəticəsində təyin olunur.

Nümunə Paraqraf 6.1.1 - də verilmiş 5 nöqtənin toplayıcı klasterləşdirmə üsulu ilə qurplaşdırılmasını nəzərdən keçirək:

$$1 : (5; 8), 2 : (8; 2), 3 : (2; 2), 4 : (5; 6), 5 : (9; 3)$$

İlk öncə bu nöqtələrin hər birini ayrı-ayrı klasterlərə yerləşdirək. Daha sonra nöqtələr arasındakı məsafələri təyin edək (cədvəl 6.4).

	0	**1**	**2**	**3**	**4**
0	0				
1	6.7	0			
2	6.7	6	0		
3	2	5	5	0	
4	6.4	**1.4**	7.1	5	0

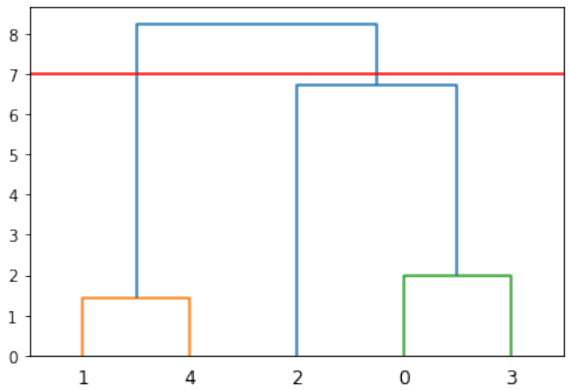
Cədvəl 6.4: Klasterləşdirilən nöqtələr arasındakı məsafə

İndi klasterləşdirməyə başlayaq. Göründüyü kimi ən kiçik məsafə 1 və 4 nöqtələri arasındadır, beləliklə, ən birinci onlar bir qrupda və ya klasterdə birləşdirilir $\{1, 4\}$. Yeni məsafə matirisini qurmaq üçün 1 və 4 nöqtələri klasterdən çixarılır və $\{1, 4\}$ klasteri aktiv çoxluğa daxil edilir. Bu klasterlə digər nöqtələr arasındakı məsafə bu nöqtələrin 1 və 4 nöqtələri ilə olan məsafələrinin maksimumu kimi götürülür (Cədvəl 6.5). Məsələn, $d(0, 1) = 6.7$ və $d(0, 4) = 6.4$ olduğu üçün $d(0, \{1, 4\} = 6.7)$ olar. Növbəti mərhələdə ən kiçik məsafəyə malik klasterlər qruplaşdırılır, bunlar 0 və 3 nöqtələridir.

Bu cür davam etdirilir və 5 iterasiyadan sonra bütün obyektlər bir klasterdə birləşdirlir. Nəticə şəkil 6.7 - də göstərilmişdir. Burada x oxu üzərində nöqtələr, y oxu üzərində isə klasterlər arasındakı məsafələr göstərilmişdir. Əgər bitmə meyarı olaraq hündürlüyü $y = 7$ götürsək və bu hündürlükdə ağacı kəssək $\{1, 4\}$ və $\{0, 2, 3\}$ klasterlərini əldə edərik.

	1, 4	**0**	**2**	**3**
1, 4	0			
0	6.7	0		
2	7.1	6.7	0	
3	5	**2**	5	0

Cədvəl 6.5: Məsafə matrisi, 2-ci iterasiya



Şəkil 6.7: Klasterləşdirmənin nəticəsində alınan dendrogram

6.2.2 Bölücü klasterləşdirmə

Bundan öncə nəzərdən keçirdiyimiz toplayıcı klasterləşdirmədə proses aşağıdan yuxarı həyata keçirilirdi, lakin klasterləşdirməni yuxarıdan aşağıya doğru da yerinə yetirmək olar. Bu üsulu bölücü klasterləşdirmə adlanır və bütün obyektlərin bir klasterdə birləşdiyi mərhələdən başlayır. Sonra rekursiv olaraq hər bir klasterdə tək obyekt qalana qədər yastı klasterləşdirmə üsulu ilə klasterlərə bölünür.

Yuxarıdan aşağıya doğru klasterləşdirmə aşağıdan yuxarıya doğru klasterləşdirmədən daha mürəkkəbdir. Belə ki, burda əlavə olaraq ikinci yastı klasterləşdirmə alqoritminə ehtiyac var. Bu zaman əgər tam olaraq ayrı-ayrı elementlərdən ibarət klasterlər qalana qədər iyerarxiya yaratmasaq bu üsul toplayıcı klasterləşdirməyə nəzərən üstünlüyə malikdir.

Nümunə. Paraqraf 6.2.1 - də verilən nöqtələri nəzərdən keçirək. İlk mərhələdə bütün nöqtələrdən ibarət bir klasterlə başlayırıq. Yenə də bu nöqtələr arasındakı məsafəni Evklid metrikası ilə hesablayaq. Hansı nöqtənin digərlərindən daha çox fərqləndiyini təyin etmək üçün bütün nöqtələrin digərləri ilə ortaq məsafəsini hesablayaq (Cədvəl 6.6).

Cədvəldən göründüyü kimi orta məsafəyə görə digərlərinə ən uzaq olan nöqtə 2-dir, beləliklə alınan klasterlər $A = \{ 0, 1, 3, 4\}$ və $B = \{ 2\}$ olar. Növbəti mərhələdə A klasterindəki nöqtələrin içərisində B klasterinə yəni 2 nöqtəsinə daha yaxın olanları seçmək lazımdır. Bunun üçün A klasterinin hər bir x nöqtəsi üçün $d(x, A-x)$ və $d(x, B)$ məsafələrini hesablayırıq. Məsələn, 0 nöqtəsi üçün aşağıdakı məsafəni

	0	**1**	**2**	**3**	**4**
0	0	6.7	6.7	2	6.4
1	6.7	0	6	5	1.4
2	6.7	6	0	5	7.1
3	2	5	5	0	5
4	6.4	1.4	7.1	5	0
Orta məsafə	5.45	4.78	6.2	4.25	4.98

Cədvəl 6.6: Bölücü klasterləşdirmə - birinci addım

hesablayaq:

$$[d(0,1) + d(0,3) + d(0,4)]/3 - d(0,2) = -1.7$$

Bütün məsafələrin qiymətləri cədvəl 6.7 - də göstərilmişdir. Cədvəldən göründüyü kimi bütün məsafələr mənfi olduğundan A klasterində B klasterinə daha yaxın olan nöqtələr yoxdur. Əvvəlki qayda ilə A klasterini hissələrə ayıraq (Cədvəl 6.8).

	0	**1**	**3**	**4**
Klasterlərarası məsafə	-1.7	-2.3	-1	-2.8

Cədvəl 6.7: Bölücü klasterləşdirmə - ikinci addım

Üçüncü addımın nəticələrindən göründüyü kimi digərlərindən ən uzaq nöqtə 0 nöqtəsidir. İkinci addımdakı kimi klasterlərarası məsafələri hesablayaq (cədvəl 6.9).

	0	**1**	**3**	**4**
0	0	6.7	2	6.4
1	6.7	0	5	1.4
3	2	5	0	5
4	6.4	1.4	5	0
Orta məsafə	5	4.4	4	4.3

Cədvəl 6.8: Bölücü klasterləşdirmə - üçüncü addım

Dördüncü addımdan göründüyü kimi 3 nöqtəsi üçün klasterlərarası məsafə müsbət olduğuna görə onu yeni klasterə əlavə edirik, beləliklə, aşağıdakı klasterləri alarıq: $A = \{0, 3\}$, $B = \{2\}$ və $C = \{1, 4\}$. A və C klasterlərindən hansını ilk olaraq klasterləşdirmək lazım olduğunu onların diametrlərini hesablamaqla təyin etmiş oluruq. A klasterinin diametri 0 və 3 nöqtələri arasındakə məsafəyə, yəni 2-yə, C klasterinin diametri isə 1.4 - ə bərabərdir. A klasterinin diametri daha böyük olduğuna görə ilk olaraq onu, sonra isə C klasterini qruplara ayıraq. Nəticədə yalnız bir elementdən ibarət klasterlər alınır.

K-means klasterləşdirmə üsulu ilə qruplaşdırdığımız şirkətlər haqqında verlənləri iyerarxik klasterləşdirmə üsulu olan toplayıcı klasterləşdirmə ilə qruplaşdıraq

	1	**3**	**4**
Klasterlərarası məsafə	-3.5	3	-3.2

Cədvəl 6.9: Bölücü klasterləşdirmə - dördüncü addım

(şəkil 6.8). Bunun üçün *sklearn.cluster* kitabxanasından *AgglomerativeClustering* funksiyasından istifadə etmək lazımdır.

Bu fəsildə təsvir olunan klasterləşdirmə üsullarını müqayisə edək. K-means klasterləşdirmə üsulu yığılma zəmanəti olmasına və müxtəlif ölçülü və formalı klasterlər üçün yaxşı nəticə göstərməsinə baxmayaraq klasterlərin sayının k proqnozlaşdırılması və başlanğıc nöqtələrin seçilməsindən asılı olması kimi problemlərə malikdir. İyerarxik klasterləşdirmə üsulları daha çox hesablama və yaddaş ($n \times n$ ölçülü məsafə matrisinin hesablanması və yadda saxlanılması) tələb etdiyinə görə mükəmməl olmasa da tətbiqinin asan olması və istənilən tip atributlara tətbiq edilə bilməsi kimi üstünlüklərə malikdir.

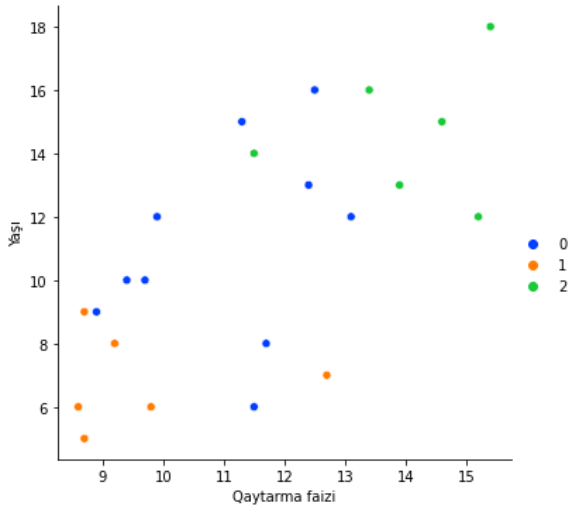
Hal - hazırda adlandırılmamış verilənlərin qruplaşdırılması, anomal halların aşkarlanması, əlamətlərin çıxarılması kimi sahələrdə müəllimsiz öyrənmə üsullarının tətbiq sahəsi genişlənir, yeni yanaşma və alqoritmlər təklif və tətbiq olunur [32].

Sonda qeyd edək ki, müəllimsiz və müəllimli öyrənməyə əlavə olaraq, gücləndirici öyrənmə (reinforcement learning) adlanan üçüncü növ maşın öyrənmə strategiyası də mövcuddur. Müəllimsiz öyrənmədə olduğu kimi gücləndirici öyrənmədə də adlandırılmış məlumat yoxdur. Bunun əvəzinə, bir model mövcud olduğu mühit ilə qarşılıqlı əlaqə quraraq zamanla öyrənir. Məsələn, bir robot yeriməyi öyrənirsə, o, fərqli ardıcıllıqlarla addımlar atmaq üçün müxtəlif strategiyalara cəhd edə bilər. Robot daha uzun müddət uğurlu şəkildə yeriyirsə, bu nəticəyə səbəb olan strategiyaya mükafat təyin edilir. Zamanla, gücləndirici öyrənmə modeli kəşfiyyatı (yeni strategiyaları sınamaq) və istismarı (məlum uğurlu üsullardan istifadə etməklə) balanslaşdırmaqla uşaq kimi öyrənir.

Çalışmalar

1. Aşağıdakı şəkildə nöqtələr verilmişdir:
   (2, 3), (3, 5), (6, 8), (8, 7), (10, 10)
   (1, 1), (4, 4), (7, 9), (9, 6), (12, 11)
   Bu nöqtələrin klasterlərə bölünməsi:
   klaster 1: 1, 2, 3
   klaster 2: 4, 5, 6
   klaster 3: 7, 8, 9, 10
   şəklindədir. Yeni klaster mərkəzlərini təyin edin və nöqtələrin paylanmasını yenidən hesablayın.

2. 1-ci tapşırıqda verilmiş nöqtələr üçün ixtiyari başlanğıc nöqtələrlə və Evklid metrikası ilə k-means üsulunun 3 iterasiyasını həyata keçirin və nöqtələri qruplaşdırın.



Şəkil 6.8: İyerarxik klasterləşdirmənin nəticəsi

3. A(2, 10), B(2, 5), C(8, 4), D(5, 8), E(7, 5), F(6, 4), G(1, 2), H(4, 9) nöqtələri verilmişdir. Toplayıcı klasterləşdirmə üsulundan istifadə edərək dendroqramı qurun.

4. 2-ci tapşırıqda verilmiş nöqtələrdən və bölücü klasterləşdirmə üsulundan istifadə edərək nöqtələri klasterləşdirin və dendroqramı qurun.

Ədəbiyyat

[1] Jyh-An Lee, Reto Hilty, Kung-Chung Liu. *Artificial Intelligence and Intellectual Property*. Oxford Academic, Oxford, 2021.

[2] The history of AI,
https://aiws.net/the-history-of-ai/

[3] Recent Advances in Google Translate,
https://blog.research.google/2020/06/recent-advances-in-google-translate.html

[4] Gaikwad Santosh K., Bharti W. Gawali, and Pravin Yannawar. *A review on speech recognition technique*. International Journal of Computer Applications, 10(3):16–24, 2010.

[5] Anil Jain, Stan Li. *Handbook of face recognition*. Springer, New York, 2011.

[6] Machine learning the hard way: IBM Watson's fatal misdiagnosis,
https://www.theregister.com/2022/01/31/machine_learning_the_hard_way/

[7] AI Gone Wrong: 5 Biggest AI Failures Of All Time,
https://www.jumpstartmag.com/ai-gone-wrong-5-biggest-ai-failures-of-all-time/

[8] Wang, Peng, et al. *Detection mechanisms of one-pixel attack*. Wireless Communications and Mobile Computing, 1–8, 2021.

[9] What is AI, is it dangerous and what jobs are at risk?
https://www.bbc.com/news/technology-65855333

[10] Han Jiawei, Micheline Kamber, Jian Pei. *Data mining concepts and techniques third edition*. Morgan Kaufmann, Elsevier, 2012.

[11] Ethem Alpaydin. *Machine Learning*. MIT Press, 2021.

[12] Daniel Jurafsky, James H. Martin. *Speech and Language Processing*. Pearson Education, 2014.

[13] He Feng, Xiaoqing Ding. *Improving naive bayes text classifier using smoothing methods.*. Advances in Information Retrieval: 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007. Proceedings 29. Springer Berlin Heidelberg, 2007.

[14] Heart disease
`https://archive.ics.uci.edu/dataset/45/heart+disease`

[15] Advertising.csv
`https://www.kaggle.com/datasets/bumba5341/advertisingcsv`

[16] Howard J. Seltman. *Experimental Design and Analysis*. Carnegie Mellon University, 2013.

[17] Auto-mpg dataset
`https://www.kaggle.com/datasets/uciml/autompg-dataset`

[18] Chenyu Zheng, Guoqiang Wu, Fan Bao, Yue Cao, Chongxuan Li,Jun Zhu *Revisiting Discriminative vs. Generative Classifiers: Theory and Implications*.arXiv preprint arXiv:2302.02334, 2023.

[19] Glass Classification
`https://www.kaggle.com/datasets/uciml/glass`

[20] Nielsen Michael. *Neural networks and deep learning*. Determination press, 2015.

[21] Kurt Hornik, Maxwell Stinchcombe, Halbert White. *Multilayer feedforward networks are universal approximators.*. Neural networks, 2(5):359–366, 1989.

[22] David E. Rumelhart, Richard Durbin, Richard Golden, Yves Chauvin. *Backpropagation: The Basic Theory*. Stanford University, 1996.

[23] Real estate valuation data set
`https://archive.ics.uci.edu/dataset/477/real+estate+valuation+data+set`

[24] Kantardzic Mehmed. *Data mining: concepts, models, methods, and algorithms*. John Wiley Sons, 2011.

[25] Biau Gérard. *Analysis of a random forests model*.The Journal of Machine Learning Research, 13, 1063-1095. 2012.

[26] Friedman Jerome. *Greedy function approximation: a gradient boosting machine*. Annals of statistics, 1189-1232. 2001.

[27] Hutter Frank, Lars Kotthoff, Joaquin Vanschoren *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[28] Tan Pang-Ning, Michael Steinbach, Vipin Kumar *Introduction to data mining.* Pearson Education India, 2016.

[29] What is K Means Clustering? With an Example
`https://statisticsbyjim.com/basics/k-means-clustering/`

[30] Aurenhammer Franz, Rolf Klein. *Voronoi Diagrams.* Handbook of computational geometry 5.10, 201-290, 2000.

[31] Murtagh Fionn, Pedro Contreras. *Algorithms for hierarchical clustering: an overview.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2.1, 86-97, 2012.

[32] Celebi M. Emre, Kemal Aydin. *Unsupervised learning algorithms.* Springer, 2016.