ҚАЗАҚСТАН ҰЛТТЫҚ БАНКІ

МАШИНАЛЫҚ ОҚЫТУ АЛГОРИТМДЕРІНІҢ КӨМЕГІМЕН ТҰТЫНУШЫЛЫҚ КРЕДИТТЕР ТӘУЕКЕЛДЕРІН ТАЛДАУ

Ақша-кредит саясаты департаменті
№2021-4 экономикалық зерттеу

Шалқар Байқұлақов

Заңғар Белгібаев

Қазақстан Республикасы Ұлттық Банкінің (бұдан әрі – ҚРҰБ) экономикалық зерттеулері мен талдамалық жазбалары ҚРҰБ зерттеулерінің, сондай-ақ ҚРҰБ қызметкерлерінің басқа да ғылыми-зерттеу жұмыстарының нәтижелерін таратуға арналған. Экономикалық зерттеулер пікірталастарды ынталандыру үшін қолданылады. Құжатта айтылған пікірлер автордың жеке ұстанымын білдіреді және ҚРҰБ ресми ұстанымымен сәйкес келмеуі мүмкін.

Машиналық оқыту алгоритмдерінің көмегімен тұтынушылық кредит тәуекелдерін талдау

NBRK – WP – 2021 – 2

Машиналық оқыту алгоритмдерінің көмегімен тұтынушылық кредит тәуекелдерін талдау

Байқұлақов Шалқар[1]
Белгібаев Заңғар[2]

Аннотация

Осы зерттеу Қазақстан Республикасының Ұлттық Банкіне екінші деңгейдегі банктер ұсынатын деректер негізінде машиналық оқыту алгоритмдерінің көмегімен жеке тұлғалардың кредит төлеуге қабілеттілігін бағалау әрекетін білдіреді. Қарыз алушылардың кредит төлеу қабілеттілігін бағалау ҚРҰБ-ға екінші деңгейдегі банктер берген кредиттердің сапасын зерттеуге және әлеуетті жүйелі тәуекелдерді болжауға мүмкіндік береді.

Бұл зерттеуде екі сызықтық және алты сызықты емес сыныптау әдістері қолданылды (*сызықтық модельдер* - логистикалық регрессия, стохастикалық градиенттің түсуі, және *сызықты емес* - нейрондық желілер, k-жақын көршілер (kNN), шешімдер тізбегі (decision tree), кездейсоқ орман (кездейсоқ тізбек), XGBoost, қарапайым Байес сыныптауышы (Naïve Bayes) және сыныптаудың дұрыстығына (accuracy), дәлдікке (precision) және басқа да бірқатар көрсеткіштерге негізделген алгоритмдер салыстырылды. Сызықтық емес модельдер сызықтық модельдерге қарағанда дәлірек болжамдарды көрсетеді. Атап айтқанда, қайта дискредиттелген (oversampled) деректер бойынша кездейсоқ орман (random forest) және k-жақын көршілер (kNN) сызықтық емес модельдері ең перспективалы нәтижелерін көрсетті.

Түйінді сөздер: *тұтынушылық кредиттер, Машиналық оқыту, банктік реттеу, стохастикалық градиентті, төмен түсіру, логистикалық регрессия, k-жақын көршілер, кездейсоқ орман сынаптауышы, шешімдер тізбегі, gaussian NB (қарапайым Байес сыныптауышы), XGBoost, нейрондық желілер (көп қабатты перцептрон)*

JEL сыныптау: G21, G28, E37, E51.

---

[1]Байқұлақов Шалқар – Жобалау офисінің директоры, ҚРҰБ-ның Төлем және қаржылық технологияларын дамыту орталығы
E-mail: sh.baikulakov@payfintech.kz

[2]Заңғар Белгібаев – ҚРҰБ-ның Ақша-кредит саясаты департаменті, монетарлық талдау басқармасы, бас маман-талдаушы,
E-mail: zanggar.belgibayev@nationalbank.kz

Мазмұны

1. Кіріспе

Тұтынушылық кредиттердің кеңеюі соңғы жылдары жеке кредиттердің өсуінің негізгі факторы болып табылады. Тұтынушылық кредиттер беру елдегі экономикалық өсуге әкелуі мүмкін, бірақ банктердің шамадан тыс қаржыландыруы жүйелік тәуекелдердің пайда болуына әкелуі мүмкін екенін де атап өткен жөн. Тұтынушылық кредиттеудің тез өсуін бір жағынан жеке тұлғалар туралы жан-жақты мәліметтердің қалыптасуымен және кредиттік тәуекелді бағалаудың тиімді құралдарымен, екінші жағынан экономикалық жағдайдың тұрақтануымен және халықтың әл-ауқатының артуымен түсіндіруге болады.

Тұтынушылық кредиттеудің серпінді кеңеюі, ең алдымен, халықтың жекелеген топтарына борыштық жүктеме деңгейінің өсуіне байланысты бірқатар жүйелік тәуекелдерді көтереді. Халықтың нақты кірісі төмендеген жағдайда банк жүйесі тұтынушылық кредиттер бойынша жаппай дефолтқа тап болуы мүмкін. Нәтижесінде, бұл экономикадағы жиынтық сұраныстың төмендеуіне әкелуі мүмкін, нәтижесінде экономиканың жай-күйіне теріс әсер етуі мүмкін.

Машиналармен оқыту алгоритмдерін қолдану бойынша зерттеу жұмыстарын зерделеу негізінде деректердің ауқымды бөлігінде тұтынушылық кредиттердің кредиттік тәуекелдерін талдаудың ең танымал әдісі нейрондық желілер, k-жақын көршілер, шешімдер тізбегі, кездейсоқ орман, XGBoost, қарапайым Байес сыныптауышы, сондай-ақ логистикалық регрессия, стохастикалық градиенттің түсуі сияқты сызықтық модельдерді қолдану болып табылады деп қорытынды жасауға болады. Авторлардың көпшілігі кредиттік бюролар мен екінші деңгейдегі банктердің деректерін пайдаланды. Осы зерттеу орталық банк жинаған реттеуші деректер негізінде жүргізілгенін ескере отырып, нәтижелері әр түрлі тәсілдерде жекелеген айырмашылықтар болуы мүмкін.

Бірінші бөлім – басқа авторлардың осыған ұқсас жұмыстарын қарастыратын әдебиеттерге шолу. **Екінші бөлімде** зерттеу әдістемесі, сондай-ақ пайдаланылған өлшемдер тізімі сипатталған. **Кейіннен** авторлар модельдеу нәтижелерін сипаттайтын нәтижелерді **талқылау бөлімі** берілген. Жұмыстың **қорытынды бөлімінде** зерттеудің негізгі қорытындылары сипатталады

2. Әдебиетке шолу

Grier (2012) зерттеу нәтижелері бойынша тұтынушылық кредиттеуді талдау үшін 5 параметрді ескеру қажет екендігі анықталды. Бірінші өлшем – қарыз алушының беделі. Қазіргі уақытта кредиттік менеджерлер кредит беру туралы шешім қабылданғанға дейін тұтынушының кредиттік тарихын көрсететін кредиттік бюроның есептеріне қол жеткізе алады. Екінші фактор – бұл тұтынушының кірісі мен қолданыстағы қаржылық міндеттемелеріне байланысты әлеуеті/қабілеті. Капитал – қарыз алушы төлей алатын бастапқы жарнаны көрсететін үшінші өлшем. Төртінші фактор еңбек нарығының жағдайы және жалпы экономикалық жағдайлар сияқты сыртқы факторларды білдіреді. Соңғы параметр – кепіл.

Кредиттік скоринг әдістері кредиттік бюроның есебі мен кредиттік өтінімнің ақпаратын пайдалана отырып, тұтынушының кредитті өтеу ықтималдығын бағалайды (Grier, 2012). Қолданбалы Скоринг – бұл жаңа тұтынушылардың өтінімдерін капитал, қуат және т.б. сияқты өлшемдердің негізінде бағалайтын кредиттік скоринг моделінің бірінші түрі.

Кредиттік скоринг әдісін пайдалану мақсаттарының бірі – берілген кредиттер бойынша ақы төлемеу тәуекелдерін төмендету. Бұл үшін дәстүрлі әдістерден басқа, банктер жақында машиналық оқыту алгоритмдерін (ML) кредиттік скорингке біріктіре бастады. Мәселен, Henley and Hand (1996) k-жақын көршіні сыныптау әдісі (kNN) сияқты машиналық оқыту әдістерін сызықтық, логистикалық регрессия және шешімдер тізбегі сияқты дәстүрлі кредиттік скоринг әдістерімен салыстырды. Алгоритмдер үмітсіз қарыздардың тәуекелі тұрғысынан сынақтан өткізіледі және ең жақсы әдіс ең аз күтілетін тәуекел деңгейімен анықталды. K-ең жақын көршіні сыныптаудың өлшемдерге бағынбауына қарамастан, ол жақсы нәтиже берді. Сонымен қатар, тұтынушының кредит бойынша төлем қабілеттілігін бағалау үшін бірнеше секунд қажет (Henley and Hand, 1996).

Келесісі, Addo, Gueran, and Hassani (2018) логистикалық регрессияны, кездейсоқ орман сыныптауышын және градиентті жоғарылатуды, сондай-ақ компаниялардың кредиттік тәуекелін талдау үшін мұқият оқыту моделін қолданды. Әр түрлі мәліметтер жиынтығы қолданылды, содан кейін ең маңызды 10 өлшем таңдалды және нәтижелерді салыстыру үшін бірдей әдістер қолданылды. Ең жақсы алгоритм қисық астындағы ең үлкен ауданды (AUC) және ең кіші квадраттық қатені (RMSE) көрсетеді деп болжанады. Бинарлық сыныптауыштар мұқият оқыту модельдерінен асып түсті, ал ең жақсы өнімділік градиенттік бустинг сияқты сынытауышқа жатады.

Тұтынушылық кредит өтінімдерін бағалау үшін Машиналық оқытудың регрессиялық модельдері де қолданылады. Munkhdalai et al. (2019) машинаны оқыту алгоритмдерін Fico кредиттік рейтинг жүйесі сияқты модельдермен салыстырды. Survey of Consumer Finances (SCF) деректері Машиналық оқытудың әртүрлі регрессиялық әдістері қолданылатын мәліметтер жиынтығы ретінде пайдаланылды. Сонымен қатар, терең нейрондық желілер мен xgboost алгоритмдері жоғары дәлдікті көрсетті. Нәтижесінде, егер кредиттік

мекемелер 2001 жылдан бастап машиналық оқыту модельдерін қолдана бастаса, кредиттік шығындар төмен болатыны анықталды.

Brown and Mues (2012) теңгерімсіз кредит деректері үшін логистикалық регрессия, нейрондық желі, шешімдер тізбегі, градиенттік бустинг, квадраттық жоғалту функциясы (LS-SVM) және кездейсоқ орман сияқты сыныптау әдістерінің жарамдылығы мен дәлдігін тексерді. Мәліметтер жиынтығына жеткіліксіз іріктеу әдісі қолданылды, содан кейін машиналық оқытуды сыныптау әдістерін бағалау үшін мәліметтер жиынтығының теңгерімсіздігі біртіндеп өсті. Нәтиже кездейсоқ орман мен градиенттік бустинг сыныптауыштары теңгерілмеген мәліметтермен жақсы жұмыс істейтіндігін, ал шешімдер тізбегі, квадраттық дискриминантты талдау (QDA) және k-жақын көршілер (kNN) басқа әдістерге қарағанда нашар жұмыс істейтінін көрсетті.

Жоғарыда аталған модельдерден басқа, машиналық оқытуда бірнеше алгоритмдер бар. Baesens et al. (2003) ядро негізіндегі тірек векторларының әдісін (kernel-based SVM) және квадраттық шығын функциясы бар тірек векторларының әдісін (LS-SVM), сондай-ақ Бенилюкс пен Ұлыбританияның қаржы институттарының мәліметтер жиынтығын қоса алғанда, нақты кредиттік скоринг деректері үшін басқа да танымал машиналық оқыту сыныптауыштарын сынақтан өткізді. Нәтиже нейрондық желі сыныптаушы мен ядроға негізделген тірек векторларының машиналары өте жақсы жұмыс істегенін көрсетеді. Сонымен қатар автор сызықтық дискриминантты талдау мен логистикалық регрессияның жақсы жұмысын атап өтті. 41 сынытпауыш, соның ішінде бірдей мәліметтер жиынтығына қолданылатын кредиттік бағалаудың жаңа әдістері (Lesmann, Baesens, Seow, and Thomas, 2015). Зерттеу нәтижесі көрсеткендей, жасанды нейрондық желілер (ANN) экстремалды оқыту әдістеріне (ELM) қарағанда жақсы жұмыс істейді, ал кездейсоқ орман (RF) айналмалы орманға (RotFor) қарағанда жақсы. Алайда, бұл әдістер модельге экономикалық түсінік бере алмайды және осы мақсатқа жету үшін қосымша зерттеулер жүргізу қажет. Кездейсоқ орман сыныптауышы анықтама ретінде таңдалды, өйткені ол іргелі талдау үшін түсіндірме ақпарат бере алады.

Tsai and Chen (2010) қарапайым сыныптауыштармен салыстыру үшін төрт гибридті машиналық оқыту әдісін ойлап тапты. Гибридті алгоритм – бұл машиналық оқытудың екі әдісінің жиынтығы. Бұл зерттеу сыныптау және кластерлеу әдістерін, сондай-ақ төрт түрлі әдісті таңдады. Нәтижесі логистикалық регрессия (LR) мен нейрондық желілердің үйлесімі ең жоғары болжамды, ал «қос кластерлеу» ең нашар алгоритм екенін көрсетті.

2015 жылы қатысушылар Xgboost-ті Kaggle-дің 29 шешімінің 17-сіде жүзеге асырды. Алгоритм дүкеннің сатылымын болжау, веб-мәтінді сыныптау, клиенттердің іс-әрекетін болжау, зиянды бағдарламаларды (Chen & Guestrin, 2016) сыныптау үшін жүзеге асырылды. Бұл зерттеуде алгоритм кредиттік талдау үшін қолданылады және басқа әдістермен салыстырылады.

Бұл зерттеуде мәліметтер жиынтығына келесі бөлімде сипатталатын екі сызықтық және алты сызықты емес сыныптау әдістері қолданылды. Алгоритмдер дұрыс сыныптау (accuracy), дәлдік (precision) және бірнеше басқа көрсеткіштер бойынша салыстырылды.

3. Зерттеу әдіснамасы және бастапқы деректер

Қазақстандық банктердің тұтынушылық кредиттерінің портфельдерін талдау үшін машина арқылы оқыту алгоритмдері пайдаланылды. Деректер екінші деңгейдегі банктер Қазақстан Ұлттық Банкіне (ҚҰБ) ұсынатын ақпарат негізінде қалыптастырылатын Кредиттік тіркелімнен алынды. Деректерден жетіспейтін және сенімді емес өлшемдері бар кредиттер алынған. Сонымен қатар деректерден 90 күннен аспайтын мерзімі өткен төлемі бар несиелер алып тасталды. Егер кредит бойынша төлемнің мерзімін өткізіп алу 90 күннен асса, несие жұмыс істемейтін несие (NPL) болып танылады. Жұмыс істейтін кредиттердің тұтынушы тарапынан мерзімі өткен төлемі болмайды.

1-кесте

Зерттеу үшін деректер өлшемдері

Өлшемдер	**Түрлері**
Өңірлер	Нұр-Сұлтан, Алматы және 15 өңір
Валютасы	Теңге, Рубль, АҚШ доллары және Еуро
Қарыз түрі	Қолма-қол ақша қарыздары және кредиттік карта
Кредиттеу объектісі	Тұтынушылық және автокредиттер
Жынысы	Ер және әйел
Азаматтық	Резидент және бейрезидент
Пайыздық мөлшерлемесі	0%-дан 56%-ға дейін
Қарыз сомасы	10 000 теңгеден 15 млн теңгеге дейін
Жасы	18 жастан 99 жасқа дейін

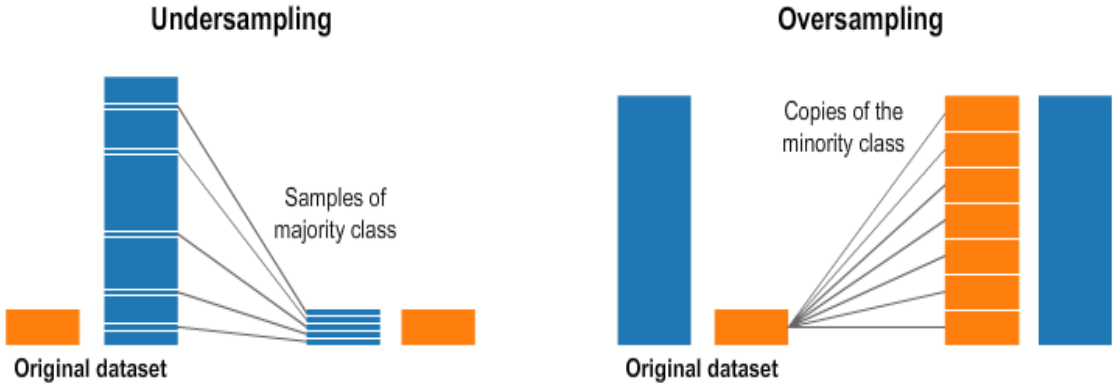
Дереккөзі: Қазақстан Республикасының Ұлттық Банкі

Теңгерімді емес деректерге арналған сэмплинг стратегиясы

Машина арқылы оқытудың қолданбалы алгоритмдерінің дәлдігін арттыру үшін мажоритарлық сынып мысалдарын жою (субдискредиттеу (undersampling)) және миноритарлық класс мысалдарының санын көбейту (қайта дискредиттеу (oversampling)) сияқты сэмплинг стратегиялары қолданылды. Жақсы несиелердің саны проблемалық несиелерден едәуір көп болғандықтан, деректер теңдестіріліп, екі әдіс салыстырылды.

1-сурет

Теңгерімді емес деректерге арналған қайта сэмплинг әдістері



Дереккөзі: Alencar, 2017

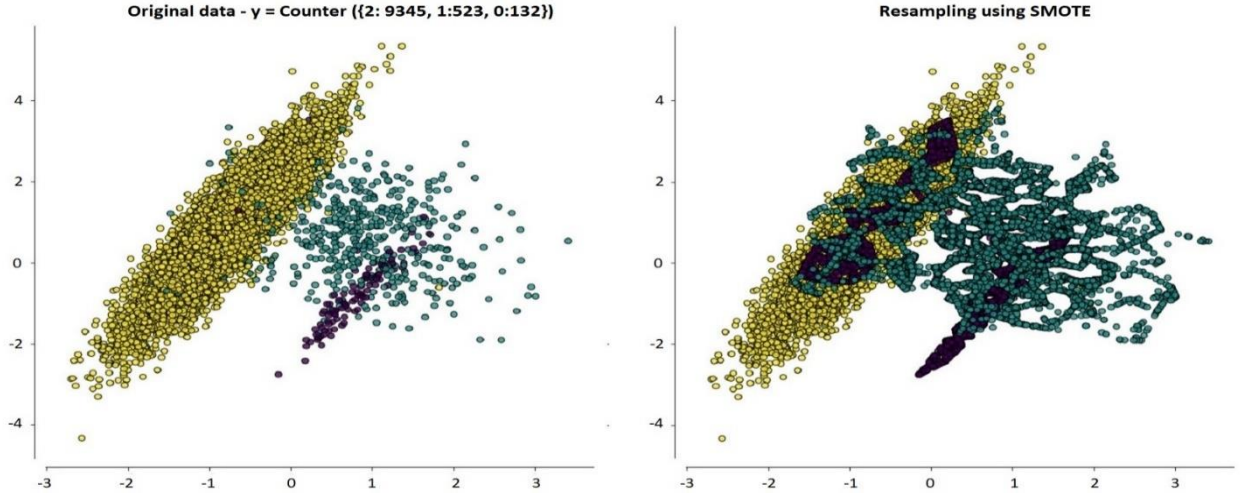
Мажоритарлық сынып мысалдарын кездейсоқ жою (кездейсоқ субдискредиттеу (random undersampling)) және миноритарлық сынып мысалдарының кездейсоқ қайталануы (кездейсоқ қайта дискредиттеу (random oversampling)) - ең танымал және қарапайым сэмплинг стратегиялары. Бірінші стратегия көпшілік сыныптан алынған азшылық сыныбы тең мөлшерімен жаңа сынып жасайды, ал екінші стратегия көпшілік пен азшылық сыныптарының ұзындығы тең болғанша азшылық сыныбын бірнеше рет қайталайды. Мажоритарлық сынып мысалдарын кездейсоқ жоюдың кемшілігі (random undersampling) ақпараттың жоғалуы болып табылады. Алайда, бұл сыныптық және сандық деректерден тұратын деректер жиынтығы үшін жеткіліксіз іріктеудің жалғыз қолайлы стратегиясы. Миноритарлық сынып мысалдарының кездейсоқ қайталануы (random oversampling) азшылық сыныбын көшіретіндіктен қайта оқытуға әкелетініне байланысты (Alencar, 2017) басқа қолайлы сэмплинг әдісі таңдалды.

SMOTE (азшылықты синтетикалық қайта дискредиттеу әдісі) – бұл қолданыстағы деректерді пайдалана отырып, азшылық сыныптарының жаңа үлгілерін сызатын тағы бір танымал сэмплинг әдісі. Ол мысал мен жақын 5 көрші арасында сызықтар сызып, осы сызық бойымен жаңа жасанды үлгі жасайды. 2-график жаңа үлгілердің қалай жасалынғанын көрсетеді, сол арқылы машина арқылы оқыту моделіне қосымша ақпарат береді (Brownlee, 2020).

SMOTE жоғарыда аталған деректер жиынтығына қолданыла алмайды, өйткені ол санаттық айнымалыларға қолданылмайды. Сондықтан синтетикалық SMOTE-NC әдісі сэмплинг стратегиясы ретінде таңдалды. Ол үздіксіз деректер жиынтығы үшін SMOTE әдісі ретінде жұмыс істейді, ал жаңа үлгінің сыныптық айнымалысы k-жақын көршілер көпшілігінің мәні болып табылады (Chawla, Bowyer, Hall and Kegelmeyer, 2002).

2-сурет

SMOTE пайдалана отырып миноритарлық сынып оверсемплингі



Дереккөзі: Lemaitre, Nogueira, Oliveira and Aridas, n.d.

Алдын ала өңдеу (Preprocessing)

Алдын ала өңдеу – деректерді қолдануға мүмкіндік беретін маңызды міндет. Деректер жиынтығының алты параметрі категориялды деректер болып табылады және оларды кодтаушы арқылы сандық түрге айналдыру керек. Scikit-learn кітапханасы (Python) дискредиттік деректерді қарапайым сандық массивке түрлендіретін әдістерді ұсынады. Сонымен қатар деректерді оқыту және тестілеу үшін деректер жиынтығына бөлетін модельдер бар (du Boisberranger, n.d.).

Алдын ала өңдеудің келесі маңызды бөлігі – деректерді шкалалау. Машина арқылы оқыту алгоритмдері әдетте Евклид қашықтық өлшемін қолданады. Осылайша, олар параметр шамасына сезімтал болады. Мысалы, алгоритмдер кредит сомасының үлкен мәндеріне байланысты деректер жиынтығындағы басқа параметрлерді елемейді. Олай болса, деректер жиынтығын қалыпқа келтіру үшін функцияларды масштабтау қажет. Бұл сонымен қатар құнын азайтады: кредиттік скоринг әдістері қалыпқа келтірілген мәліметтер жиынтығын тезірек талдайды. Жоғарыда аталған Python кітапханасы мына формула арқылы деректер жиынтығын қалыпқа келтіретін стандартты шкалалау әдістерін ұсынады:

$$z = \frac{x-\mu}{\sigma}$$ (Boisberranger et al., n.d.)

мұнда, μ - орташа мәні және ϭ деректердің стандартты ауытқуы болып табылады.

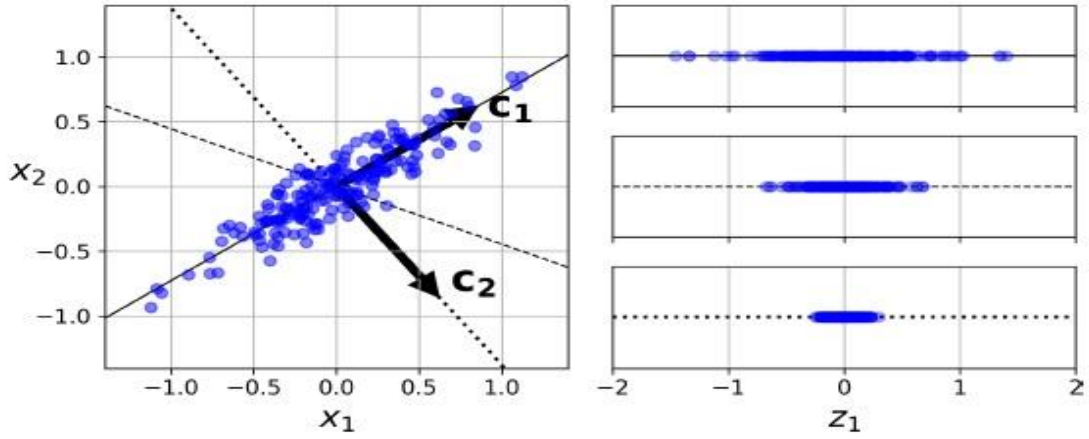
Басты компоненттер әдісі (Principal component analysis)

Басты компоненттер әдісі (PCA) – өлшемді азайтудың белгілі әдістерінің бірі. Бұл ақпаратты барынша аз жоғалтумен деректер жиынтығының мөлшерін азайтуға көмектеседі (Mueller & Guido, 2017). Демек, бұл алгоритмді есептеу уақытын азайтады.

Алгоритм дұрыс гипержазықтықты таңдап, осы гипержазықтыққа деректерді түсіреді. Болжамнан кейін деректердің дисперсиясы негізгі компонент деп аталатын әрбір жаңа ось негізінде есептеледі. Бірінші компонент барынша көп дисперсияны сақтайды, ал соңғы компонент барынша аз дисперсияны сақтайды.

3-сурет

Басты компоненттерді талдау

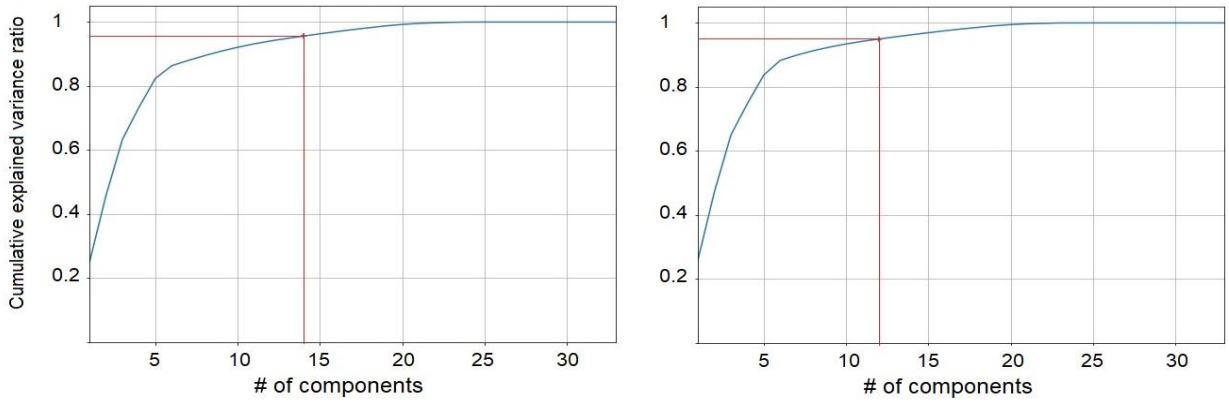


Дереккөзі: Geron, 2017

Басты компоненттер әдісі екі теңдестірілген деректер жиынтығы үшін де жасалады. Субдискредиттелген деректер жиынтығының нәтижесі алғашқы 14 компоненттің (бағандардың) деректер жиынтығының жалпы дисперсиясының 95%-ын қамту үшін жеткілікті екенін, ал қалған 19 компонентті жоюға болатындығын көрсетеді. Жалпы дисперсияның 95%-ын қамту қайта дискредиттелген деректер жиынтығының тек 12 компонентін пайдалану жеткілікті.

4-сурет

Басты компоненттерді талдаудың кумулятивтік дисперсиясының коэффициенті



Дереккөзі: авторлар жасаған

Бұл зерттеуде деректер жиынтығына машина арқылы оқытудың 2 сызықты және 6 сызықты емес әдісі қолданылды. Негізгі мақсат – қай алгоритмнің басқалардан асып түсетінін анықтау. Сонымен қатар бұл зерттеу

сызықтық және сызықтық емес алгоритмдерді салыстыруға және қарсы қойып салыстыруға көмектеседі.

Деректердің үлкен жиынтығын өңдеу мүмкіндігі алгоритмді таңдаудың басты өлшемшерты болып табылады. Сондықтан зерттеуде сызықтық сыныптауыш және тірек векторларының сыныптауышы (SVC) сияқты белгілі алгоритмдер қарастырылмады. Талдау мақсатында логистикалық регрессия, стохастикалық градиентті төмендеу сыныптауышы (SGD), Байес сыныптауышы, k-жақын көршілер (kNN), шешім жиынтығы (decision tree), кездейсоқ мәндер (random tree), көп қабатты персептрон сыныптауышы (MLP) (нейрондық желі сыныптауышы) және XGBoost қолданылып, нәтижелері талқыланды. Сызықтық емес алгоритмдердің кейбір гиперпараметрлері үлкен деректер жиынтығын өңдеудің мүмкін болмауына байланысты ескерілмейді.

Логистикалық регрессия (Logistic Regression)

Атына қарамастан жіктеу үшін логистикалық регрессия қолданылады. Алгоритм ықтималдылықты мына формула бойынша оқытушы деректер жиынтығы негізінде есептейді:

$$P(y=1|x) = \frac{1}{1+e^{-(w_0+\boldsymbol{W}^{\boldsymbol{T}}x)}}$$ (Baesens et al., 2003)

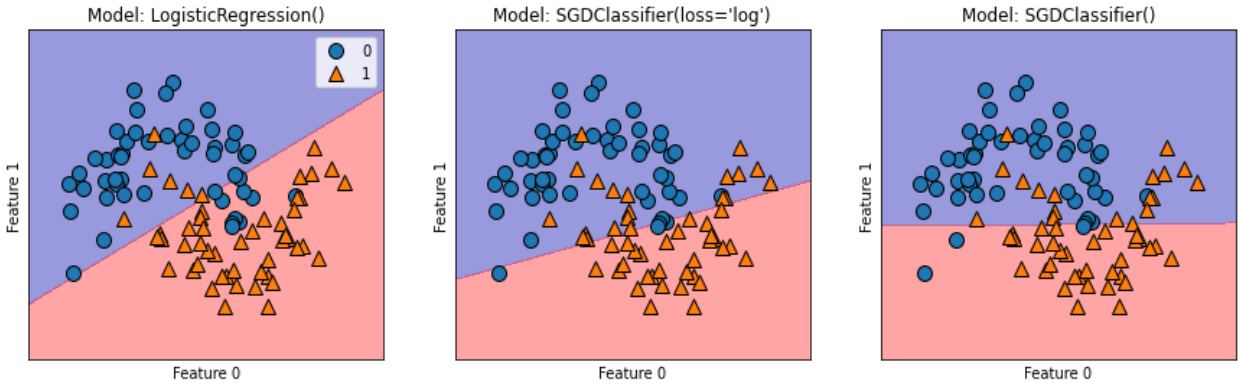
мұнда x – кіріс деректер, w_0 – скалярлық тоғысу векторы, ал W – параметрлер векторы. Егер ықтималдық 50%-дан көп болса, кіріс деректер оң деп жіктеледі.

Дәлдікті арттыру үшін шешуші (solver) және реттеу параметрлері сияқты маңызды гиперпараметрлер орнатылды. «Liblinear» шешімі әдетте кішігірім деректер жиынтығы үшін қолданылады, ал «sag» және «saga» - ауқымды деректерді талдау үшін ең жақсы таңдау болып табылады, өйткені есептеу үшін аз уақыт қажет (du Boisberranger et al., N.d.).

Айыппұл (penalty) – бұл айыппұл салу кезінде қолданылатын реттеу гиперпараметрі және ол шешушімен тығыз байланысты. «Newton-cg», «sag» және «lbfgs» шешімдері тек «l2» айыппұлды қолдайды, ал «saga» шешімі «elasticnet» айыппұлын қолдайды. «C» - бұл реттеу күшіне кері шама, ол оң болуы тиіс. «C» ең аз мәні анағұрлым күшті реттеуді білдіреді (du Boisberranger et al., N.d.).

5-сурет

Сызықты сыныптау үлгілері



Дереккөзі: авторлар жасады

Стохастикалық градиенттік түсу (Stochastic Gradient Descent (SGD))

SGD сыныптауышы – үлкен деректер жиынтығына қолданылатын сызықтық сыныптауыштың тиімді әдістерінің бірі. Алгоритм бірінші қатардағы SGD оқыту рәсімін қолданады. Әдіс өлшемі берілген формула бойынша оқыту мысалдарымен итеративті түрде жаңартылады:

$$\omega = \omega - \eta[\alpha \frac{\partial R(\omega)}{\partial \omega} + \frac{\partial L(\omega^T x_i + b, y_i)}{\partial \omega}]$$ (du Boisberranger et al., n.d.)

Мұндағы α - нормалау күшін басқаратын гиперөлшем, R - үлгінің күрделілігін төмендететін нормалауды көрсету. L - үлгінің сәйкестігін өлшейтін залал функциясы, ŋ - оқу жылдамдығы, ал b - реттеусіз ұқсас түрде жаңартылатын қиылысу орны.

Градиенттік түсу - құн функциясын азайту үшін қолданылатын маңызды алгоритмдердің бірі (Fuchs, 2019). Жалпы, градиенттік түсудің үш танымал түрі бар:

1. Жалпы градиенттік түсуі (Batch gradient descent);
2. Стохастикалық градиенттік түсуі (Stochastic gradient descent);
3. «Mini-batch» градиенттік түсуі (Mini-batch gradient descent).

Жалпы градиенттік түсуі оның өлшемдерін ескере отырып, алгоритм құны функциясының жеке туындысын есептейді. Басқаша айтқанда, алгоритм үлгі өлшемдерінің әртүрлі мәндері алгоритм құнының функциясына қалай әсер ететінін анықтайды. Әр кезеңде есептеу үшін бүкіл оқу деректерін пайдаланатыны оның кемшілігі болып табылады. Сондықтан алгоритм үлкен деректер жиынтығын өңдей алмайды (Geron, 2019).

Стохастикалық градиенттік түсуі бұл мәселені шешеді, өйткені ол оқу деректерінің кездейсоқ даналарын таңдайды және осы жалғыз дананың негізінде градиенттік түсуді есептейді. Екінші жағынан, пакеттік градиенттік түсуі құн функциясын біртіндеп төмендетеді, ал алгоритм тоқтағанша, жоғары және төмен өзгереді. Алгоритм өлшемдердің оңтайлы мәндерін бере алмайды (Geron, 2019).

«Mini-batch» градиенттік түсуі оқу деректерінен аздаған іріктемені алады және жалпы градиенттік түсу алгоритмін есептейді. Сондықтан, құн функциясын есептеу нәтижесінде стохастикалық градиенттік түсуімен салыстырғанда қатесі аз (Geron, 2019).

Әдісте көптеген гиперөлшемдер бар, бұл оны өте күрделі етеді. Залалдар функциясы (loss function) сияқты маңызды өлшемдер және айыппұл (penalty) және альфа (alpha) сияқты реттеу өлшемдері үлгіні орнату процесінде ескеріледі. «Hinge» залал функциясы бар әдіс сызықтық SVM ретінде жұмыс істейді, «log» функциясымен логистикалық регрессия ретінде жұмыс істейді (du Boisberranger et al., N.d.).

Алгоритм шкала белгілеуге өте әсерлі. Алайда, оны оңай іске асыруға болады және әсіресе үлкен деректер жиынтығы үшін өте тиімді (du Boisberranger et al., N.d.). Сонымен қатар, ол ОЗУ-да жазбаны сақтамай жұмысын жалғастырады (Fuchs, 2019).

Қарапайым Байес сыныптауышы (Naïve Bayes classifier)

Жалпы, қарапайым Байес сыныптауышы (NB) әр өлшемнің тәуелсіздігін білдіретін Байес теоремасына негізделген:

$$P(y|x_1,\dots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1,\dots,x_n)}$$ (du Boisberranger et al., n.d.)

Байес сыныптауышының бірнеше түрлері бар. Бұл зерттеуде Гаусс қарапайым Байес сыныптауыштары (Gaussian Naïve Bayes (GaussianNB)) берілген формула негізінде белгілердің ықтималдығын бағалау үшін қолданылады:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}}$$ (du Boisberranger et al., n.d.)

мұндағы орташа және стандартты ауытқу ең жоғарғы ықтималдылық арқылы бағаланады.

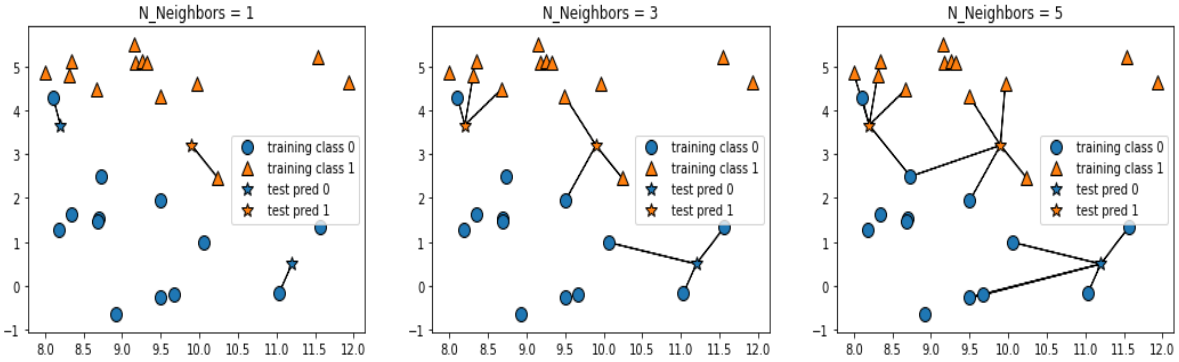
k-ближайшие соседи (kth Nearest Neighbors (kNN))

k-ең жақын көршілер – машиналық оқытудың қарапайым алгоритмдерінің бірі. Жақын көршілердің саны (k) осы алгоритмдегі негізгі өлшем болып табылады. Жаңа деректердің нәтижесі жақын көршілердің көптеген нәтижелері негізінде анықталады. Үлкен k мәні шудың әсерін басады, бірақ сыныптау шекарасында болады (du Boisberranger et al., N.d.). Қашықтық Евклид қашықтық формуласы бойынша өлшенеді:

$$d(x_i, x_j) = \|x_i - x_j\| = \left[(x_i - x_j)^T (x_i - x_j)\right]^{1/2}$$ (Brown & Mues, 2012)

6-сурет

Жақын көршілердің k сыныптау үлгісі



Дереккөзі: Mglearn library (Mueller & Guido, 2017)

Шешімдер тізбегі (Decision Tree classifier)

Шешім желісі бұл функцияларға негізделген шешім қабылдау ережелерін қолдану арқылы мақсатты нәтижені болжайтын қарапайым алгоритм. Сандық, категориялды сипаттамаларды және бірнеше шығару тәсілі бар есептерді өңдеу мүмкіндігі әдістің күшті жағы болып атылады. Тағы бір артықшылығы – үлгінің қарапайымдылығы. Бірақ үлгі шамадан тыс күрделі жүйелерді жасай алады, бұл үлгіні қайта оқытуға әкеледі (du Boisberranger et al., N.d.).

Үлгіні орнату процесінде жүйенің өлшемі мен ең жоғарғы тереңдігі ескеріледі. Бірінші өлшем – бөлу сапасын өлшейтін функция. «Gini» (Gini impurity) және «entropy» (information gain) - критерийлер функциясы үшін таңдаудың екі нұсқасы (du Boisberranger et al., N.d.).

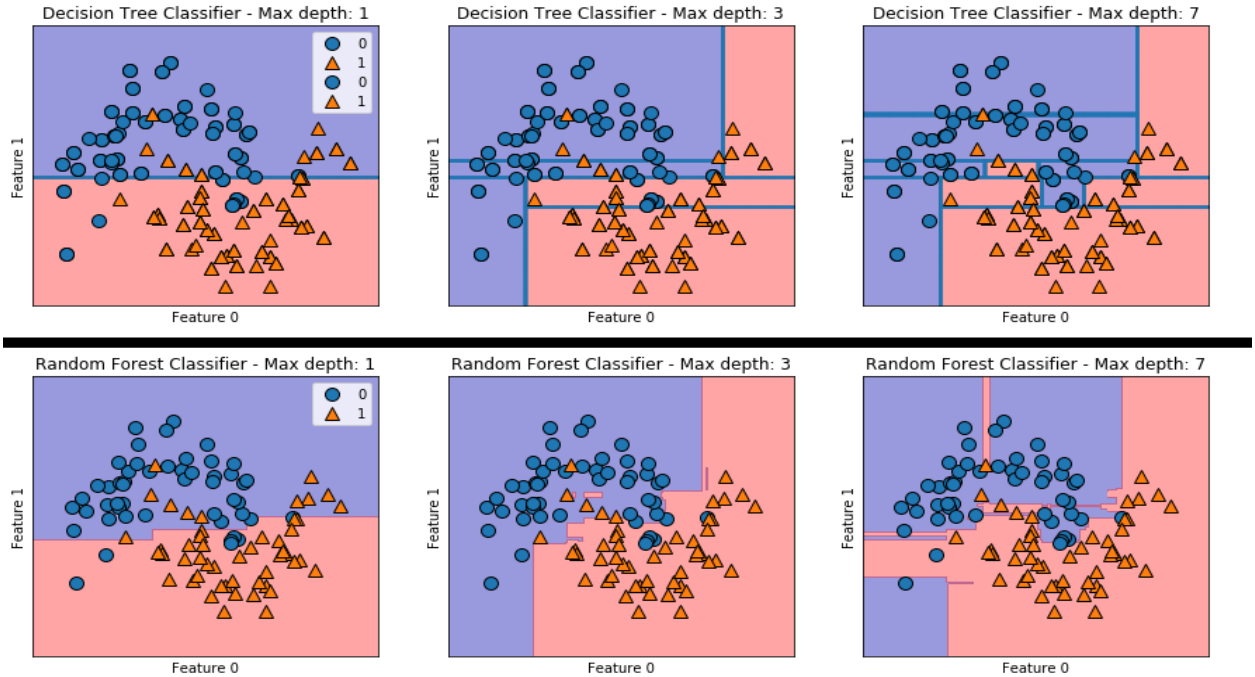
Кездейсоқ жиынтық (Random Forest Classifier)

Кездейсоқ жиынтық сыныптауышы – бұл шешім жүйесі сыныптауышының барлық гиперөлшемдерін қамтитын ансамбльді сыныптау әдісі. Екі үлгінің айырмашылығы біріншісі функциялар жиынтығы арасында ең жақсы функцияны іздейді, ал екіншісі түйінді бөлу кезінде ең жақсы функцияны іздейді. Осылайша, жиынтықтың кездейсоқ сыныптаушы үлкен жүйеге әкеледі (Geron, 2019).

Шешімдер жүйесінің сыныптауышы жоғары дисперсияны көрсетеді, бұл қайта оқытуға әкеледі. Кездейсоқ жиынтық сыныптауышы әртүрлі жүйелерді біріктіреді, осылайша дисперсияны азайтады. Кейіннен ол шешім жүйе сыныптауышы қарағанда жақсы үлгі жасайды (du Boisberranger et al., N.d.).

7-сурет

Шешімдер жүйесі және scikit-learn деректер жинағында кездейсоқ жиынтық сыныптауышы



Source: авторлар жасады

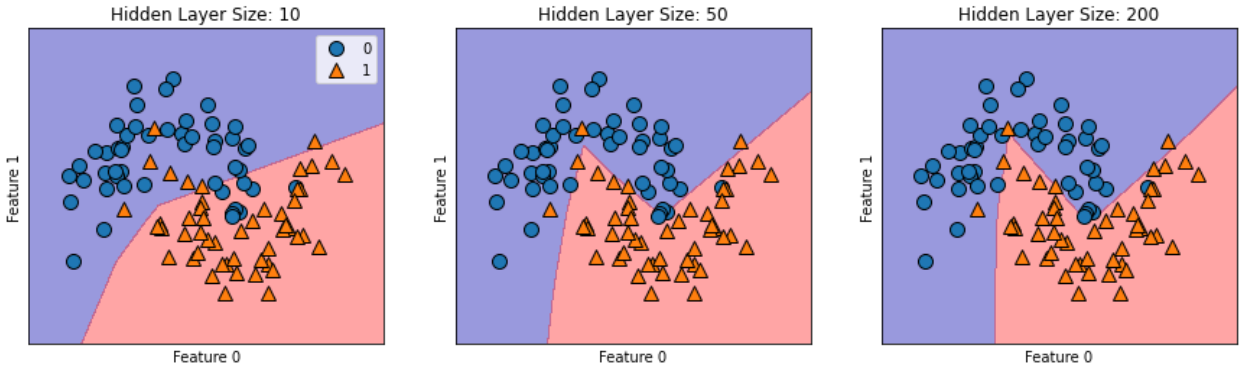
Нейрон желілері көп қатпарлы персептрон)

Нейрондық желі – бұл адам миының құрылымымен рухтандырылған үлгі. Алгоритмнің негізгі құрылымы кіріс, шығыс және жасырын қабаттардан тұрады. Жасырын қабаттардың мөлшері алгоритмнің маңызды гиперөлшем болып табылады. Әр деңгейде нөмірді қамтитын түйіндер бар және әр түйін алдыңғы деңгейдегі әр түйіннен сигнал алады және келесі деңгейдегі

түйіндерге сигнал жібереді. Әр сигналдың салмағы мен кіру әсер етпейтін орын ауыстыруы бар (Hansen, 2019).

8-сурет

Нейрон желісінің сыныптауышы



Source: авторлар жасады

Нейрон желісінің маңызды белгілерінің бірі градиенттік түсу алгоритмі болады. Ол қорытынды мен есептелген қорытынды мәні арасындағы ауытқуды азайту үшін қолданылады. Нейрон желісі градиенттерді есептеу үшін кері таралуды қолданады. Содан кейін ол барлық ығысуларды жаңартады және қорытындыда бастап барлық түйіндердің салмағын реттейді (Hansen, 2019).

XGBoost

Экстремалды градиенттік бустинг (XGBoost) – кеңінен қолданылатын алгоритмдердің бірі. Бұл градиенттік күшейткішпен шешім жүйелерін іске асыру. Модельдің орындау жылдамдығы мен өнімділігі градиентті жоғарылатудың басқа алгоритмдеріне қарағанда артықшылығы болып табылады (Brownlee, 2016).

Әдістің артықшылығы – масштабталуында. Ол сирек деректерді өңдеу үшін жаңа жүйелерді зерттеу алгоритмін қолданады және жетіспейтін мәндерді автоматты түрде өңдей алады. Әдістің тағы бір маңызды ерекшелігі – терең, оңтайландырылған жүйелер үшін жүйелерді кесу мүмкіндігі. Деректерді параллельді және бөліп өңдеу оны ең жылдам алгоритмдердің біріне айналдырады. Алгоритм деректерді өңдеуді жеңілдету және жылдамдату үшін жедел жадтан тыс есептеулерді қолданады. Сондықтан, бұл әдіс үлкен көлемдегі деректерді өңдеу үшін ең оңтайлы болып табылады. (Chen & Guestrin, 2016).

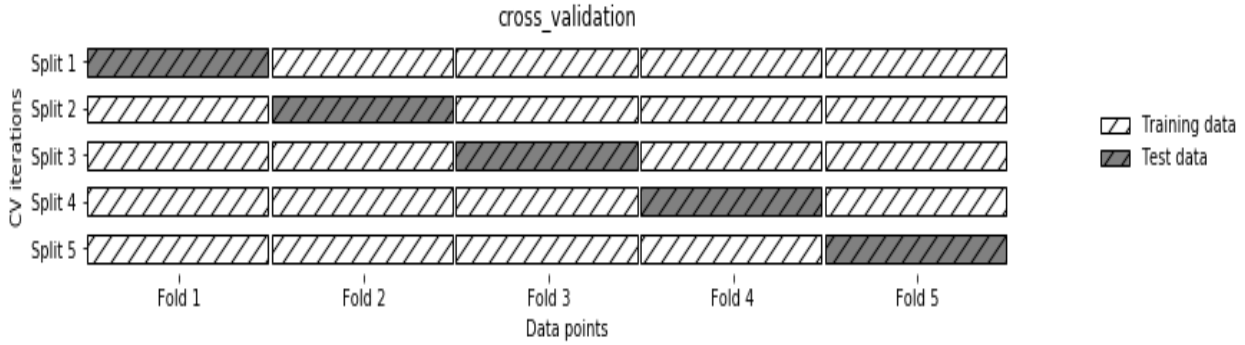
Кросс-валидация (Cross-validation)

Қиылыстырып тексеру – бұл оқыту және тестілеу деректері негізінде әдістердің өнімділігін өлшеу үшін қолданылатын статистикалық әдіс. Қиылыстырып тексерудің жиі қолданылатын нұсқасы k-x өлшемді қиылыстырып тексеру, мұнда қатпарлар саны 5 немесе 10 құрайды (Mueller & Guido, 2017). Зерттеуде 9-графикте көрсетілгендей 5-еселік қиылыстырып тексеру қолданылды. Сондықтан деректер жиынтығы бес тең бөлікке бөлінеді.

Егер деректердің бірінші бөлігі тестілік жиынтық ретінде пайдаланылса, қалғандары жаттығу іріктемесі болып табылады. Яғни, мақсат - деректер жиынтығының метрикаға негізінде үлгілердің дәлдігіне әсерін талдау.

9-сурет

Бес сатылы тоғыспалы валидация



Дереккөзі: Mglearn library (Mueller & Guido, 2017)

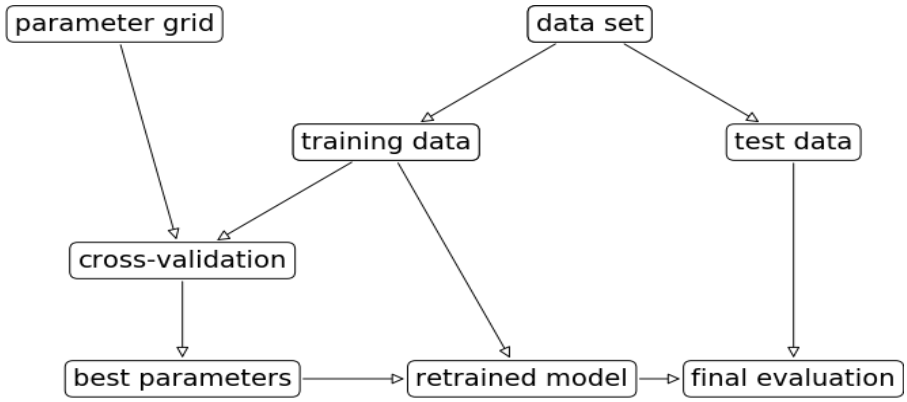
Тор бойынша іріктеу (Grid Search)

Тор бойынша іріктеу – бұл модель құрудың соңғы кезеңі. Осы кезеңде таңдалған машиналық оқыту моделі ретке келтіріледі. Параметрлер торы – бұл алгоритмдердің барлық мүмкін гиперпараметрлерінің жиынтығын қамтитын ретке келтіру процесінің алғашқы қадамы.

Содан кейін басқа гиперпараметрлер жиынтығы бар модель оқу және тест деректеріне қолданылады. Бұл қадам модель ең жоғары көрсеткішті алатын параметрлер жиынтығын анықтауға көмектеседі. Модель ең жақсы гиперпараметрлер жиынтығын қолдана отырып қайта құрылады (Mueller & Guido, 2017).

10-сурет

Grid Search



Дереккөзі: Mglearn library (Mueller & Guido, 2017)

Модельді ретке келтіру үшін гиперпараметрлерді таңдау осы кезеңдегі маңызды мәселе болып табылады. Параметрлерді іріктеудің көп уақытты қажет ететінін ескере отырып, параметрлер торы ең маңызды параметрлер негізінде құрылды. Мысалы, жақын көршілердің жалғыз саны kNN моделін орнату үшін таңдалған параметр болды. Сондай-ақ, Kaggle-де қолданылатын

параметрлердің торлары қолданылды (деректерді өңдеу жөніндегі мамандарға арналған веб-сайт).

Метрикалар (Metrics)

Машиналық оқытуда регрессия, кластерлеу және жіктеу алгоритмдері әртүрлі өнімділік көрсеткіштерін қолданады. Жіктеу алгоритмдерінде екілік және мультикластық жіктеу үшін әртүрлі өлшемдер бар. Бұл зерттеуде модельдер тиімділігінің 6 көрсеткіші қолданылды:

1. Жіктеудің дұрыстығы (Accuracy score)
2. Дәлдік (Precision score)
3. Толықтық (Recall score)
4. F-өлшемі (F1 score)
5. Жаккар өлшемі (Jaccard score)
6. Қателердің қисық асты ауданы (The area under the receiving operating characteristic (ROC) curve)
7. Екі типтегі қателер үлесі (Type 2 error percentage)

Қателер матрицасы (Confusion matrix)

Қателер матрицасы – бұл сыныптауыштың тиімділігін бағалау үшін қолданылатын матрицалық кесте. Ол әдетте жіктеу алгоритмінің дұрыс және бұрыс нәтижелерінің санын көрсетеді. Сондай-ақ, ол осы бөлімде түсіндірілетін бірінші және екінші типтегі қателер туралы ақпарат береді.

2-кесте

Confusion matrix

	Actual class (observation)	
Predicted class (expectation)	TP (true positive) Correct result	FP (false positive) Unexpected result
	FN (false negative) Missing result	TN (true negative) Correct absence of result

Дереккөзі: Binary classification (du Boisberranger et al., n.d.)

1. Жіктеудің дұрыстығы (Accuracy score)

Accuracy – зерттеудегі ең маңызды көрсеткіш. Сондай-ақ, ол торды іздеу кезінде модельді ретке келтіру үшін бағалау көрсеткіші ретінде қолданылады. Метрика модельдің дұрыс нәтижелерінің үлесін көрсетеді:

$$Accuracy(y_{true}, y_{pred}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(if\ y_{true} = y_{pred})(du\ Boisberranger\ et\ al., n.d.)$$
$$= \frac{TP + TN}{TP + FP + FN + TN}\ (Geron, 2019;\ Mueller\ \&\ Guido, 2017)$$

2. Дәлдік (Precision score)

Дәлдік көрсеткіші – бұл жалпы оң болжамдар санына бөлінген дұрыс оң болжамдардың саны.

$$Precision = \frac{TP}{TP + FP}\ (Geron, 2019; Mueller\ \&\ Guido, 2017)$$

3. Толықтық (Recall score)

Толықтық – бұл нақты оң нәтижелер санына бөлінген дұрыс оң болжамдардың саны.

$$Recall = \frac{TP}{TP + FN}\ (Geron, 2019; Mueller\ \&\ Guido, 2017)$$

4. F-өлшемі (F1 score)

Дәлдік пен толықтық – маңызды көрсеткіштер. Бірақ, олардың ешбірі жеке алғанда толық сипатын бере алмайды. F-өлшемі – екілік жіктеу үшін қолданылатын тағы бір метрика және дәлдік пен толықтықтың гармоникалық орташа мәні (Geron, 2019; Mueller & Guido; 2017).

$$F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2 \times \frac{precision \times recall}{precision + recall} = \frac{TP}{TP + \frac{FN + FP}{2}}\ (Geron, 2019)$$

5. Жаккар өлшемі (Jaccard similarity coefficient score)

Жаккар өлшемі (JSC) – бұл оларды біріктіру үшін нақты және болжамды шешімнің қиылысуы.

$$JSC = \frac{TP}{TP + FP + FN} = \frac{F_1}{2 - F_1}\ (Labatut\ \&\ Cherifi, 2011)$$

6. Қателердің қисық асты ауданы (Area under ROC curve)

ROC қисығы екілік сыныптауыштардың өнімділігін талдаудың маңызды құралы болып табылады. Бұл FP мөлшерлемесіне қатысты TP пайыздық мөлшерлемесін көрсететін сызықтық қисық. Бұл қисық астындағы аудан – сыныптауыштарды салыстырудың тағы бір әдісі болып табылады (Герон, 2019).

Бұл көрсеткіштердің мәні 0-ден 1-ге дейін өзгереді: мәні неғұрлым жоғары болса, соғұрлым модель де тұтынушылық кредиттеуді талдауға сай болады. Сонымен қатар, бұл өлшемдер өте тығыз байланысты.

7. Екі типтегі қателер үлесі (Type 2 percentage)

Машиналық оқыту сыныптауыштарының статистикалық модельдер ретінде бірінші және екінші типтегі қателері болады. Бірінші типтегі қате жалған оң нәтиже ретінде де белгілі (FP) (Schmarzo, 2018). Мысалы, екінші деңгейдегі коммерциялық банктердің бірі кредиттік скоринг модельдерін енгізеді. Модель тұтынушыға несие беруден бас тартады, өйткені ол болашақта жұмыс істемейтін несие болады деп мәлімдейді. Бірақ, негізінде, тұтынушы несиені толығымен өтей алады. Бұл жағдайда банктер борышты шығарудан бас тартады, бірақ бұл банктің өтімділігі мен төлем қабілеттілігіне айтарлықтай әсер етпейді

Жалған теріс (FN) - қателердің тағы бір түрі, ол екінші типтегі қате ретінде де белгілі (Schmarzo, 2018). Банк тұтынушыға машиналық оқыту нәтижелері негізінде борыш береді делік, онда тұтынушы оны міндетті түрде төлейтіні бекітіледі. Шындығында, несие жұмыс істемейді, бұл банктің өтімділігіне теріс әсер етеді. 2-ші типтегі қатенің жоғары пайызы модельдің қаншалықты нашар екендігін көрсетеді.

1. Алынған нәтижелерді талқылау

Алдында айтып өткендей, мажоритарлық сыныпты кездейсоқ жою және миноритарлық сыныпты кездейсоқ қайталау әдістері ең жақсы сэмплинг стратегиясы емес. Біріншісі ақпараттың жоғалуына, екіншісі – қайта оқыту проблемаларына әкеледі. Одан басқа, мажоритарлық сыныпты кездейсоқ жою осы зерттеуде нормативтік деректерді бейімдеудің жалғыз тәсілі болды. Бұл бөлімде олар қолданылған мәліметтер негізінде барлық сыныптауыштардың нәтижесі талқыланады.

Субдискредиттелген деректер (*Undersampling data)*

Зерттеу нәтижелері, сызықтық сыныптауыштар және қарапайым Байес сыныптауышы кредиттік скорингті модельдеудің ең жақсы нұсқалары емес. 3-кесте барлық сыныптауыштардың жеткілікті түрде сәйкес келмеу проблемалары болғанын көрсетеді: алгоритмдер оқу деректерін нашар модельдеді және тестілеу деректеріндегі тапсырманы дұрыс орындамады.

3-кесте

Accuracy of classifiers applied to undersampled data

	Linear Models		Non-Linear Models					
	Logistic Regression	SGD	Naïve Bayes	kNN	Decision Tree	Random Forest	Neural Networks	XGB
Training	58,6%	58,7%	59,0%	68,6%	68,3%	64,9%	70,8%	69,6%
Testing	58,7%	58,9%	59,0%	67,1%	65,9%	64,6%	70,0%	67,2%

Дереккөзі: авторлар жасаған

Басқа метрикалар бойынша төмен көрсеткіштерге қарамастан, стохастикалық градиентті түсіру сыныптауышы (SGD) екінші типтегі қателердің ең төменгі үлестерінің бірін көрсетті. Осы уақытта нейрондық желілердің сыныптауышы көптеген метрикалар негізінде жеткіліксіз іріктелген деректерге қолданылатын модельдер арасында өте жақсы нәтиже көрсетті. Алайда, 2-ші типтегі қателердің жоғары пайызы, сондай-ақ деректерді өңдеуге көп уақыт жұмсау қажеттілігі оны басқаларынан артық жасамайды. Нәтижесінде, градиенттің экстремалды үдеткіш (XGB) сыныптауыш моделі оңтайлы метрикалық көрсеткіштері: көлемі жағынан жіктеудің дұрыстығының екінші көрсеткіші және көлемі жағынан екінші типтегі қателердің екінші пайызы бар кредиттік скорингті модельдеудің ең қолайлы нұсқасы болып табылады.

4-кесте

Models and performance results

		Metrics						
		Accuracy	*Precision*	*Recall*	*F1*	*JSC*	*AUC_ROC*	*Type* 2 error
Linear	Logistic Regression	58,7%	59,0%	58,5%	58,7%	41,6%	58,7%	20,8%
	SGD	58,9%	57,9%	66,5%	61,9%	44,8%	58,9%	16,8%
Non-Linear	Naïve Bayes	59,0%	60,1%	54,3%	57,0%	39,9%	59,0%	23,0%
	kNN	67,1%	68,4%	64,0%	66,1%	49,4%	67,1%	18,6%
	Decision Tree	65,9%	66,9%	63,5%	65,2%	48,3%	65,9%	18,3%
	Random Forest	64,6%	63,5%	69,4%	66,3%	49,6%	64,6%	15,4%
	Neural Networks	70,0%	72,8%	64,1%	68,2%	51,7%	70,0%	18,0%
	XGB	67,2%	67,4%	67,3%	67,4%	50,8%	67,2%	16,4%

Дереккөзі: авторлар құрастырған

Қайта дискредиттелген деректер (Oversampled data)

Жалпы алғанда, сыныптауыштар SMOTE қайта іріктеу әдісі негізінде құрылған артық іріктеме деректерімен әлдеқайда жақсы жұмыс істеді, бұл деректерге қосымша ақпарат берді. Бұл ретте, сызықтық сыныптауыштар мен қарапайым Байес сыныптауышы ең нашар нәтижелер көрсетті. Басқа сыныптауыштар оқуға арналған деректер жиынтығымен, сондай-ақ артық іріктеме деректерімен жұмыс жасауда оң нәтиже көрсетті.

Нейрондық желілер перспективалы үлгі болды, бірақ жеткіліксіз және артық іріктеме деректері үшін үлгі көрсеткіші арасында шамалы айырмашылық бар. Артық іріктеме деректерінің дәлдігі жеткіліксіз іріктеме деректерімен салыстырғанда жоғарылағанына қарамастан, нейрондық желілер сызықты емес үлгілерден асып кете алмады.

5-кесте

Accuracy of classifiers applied to oversampled data

	Linear Models		Non-Linear Models					
	Logistic Regression	SGD	Naïve Bayes	kNN	Decision Tree	Random Forest	Neural Networks	XGB
Training	64,1%	64,1%	64,9%	99,9%	76,8%	99,6%	73,6%	74,6%
Testing	64,1%	64,1%	64,9%	83,5%	75,3%	84,8%	73,6%	74,4%

Дереккөзі: авторлар құрастырған

6-кестеге сәйкес k-жақын көршілер мен кездейсоқ ормандардың сыныптауыштары барлық көрсеткіштер бойынша басқа сыныптауыштардан асып түсті. Сонымен қатар, екі үлгі де 2-ші түрдегі қателіктердің ең төменгі пайызын көрсетті. Алайда, кездейсоқ орман сыныптауышы дәлдіктен басқа барлық көрсеткіштер бойынша k-жақын көршілердің сыныптауышынан асып түсті.

6-кесте

Models and performance results

		Metrics						
		Accuracy	*Precision*	*Recall*	*F1*	*JSC*	*AUC_ROC*	*Type* 2 error
Linear	Logistic Regression	64,1%	63,9%	65,1%	64,5%	47,6%	64,1%	17,5%
	SGD	64,1%	63,9%	65,1%	64,5%	47,6%	64,1%	17,5%
Non-Linear	Naïve Bayes	64,9%	62,6%	74,2%	67,9%	51,4%	64,9%	12,9%
	kNN	83,5%	85,2%	81,0%	83,0%	71,0%	83,5%	9,5%
	Decision Tree	75,3%	74,8%	76,2%	75,5%	60,7%	75,3%	11,9%
	Random Forest	84,8%	84,7%	85,0%	84,8%	73,6%	84,8%	7,5%
	Neural Networks	73,6%	71,2%	79,1%	75,0%	60,0%	73,6%	10,5%
	XGB	74,4%	73,5%	76,2%	74,9%	59,8%	74,4%	11,9%

Дереккөзі: авторлар құрастырған

5. Қорытынды

Зерттеу нәтижелері машиналық оқыту үлгілері орталық банк жинаған реттеуші деректер негізінде жақсы жұмыс істейтінін көрсетті. Сонымен қатар, машиналық оқыту алгоритмдері арқылы тұтынушылық кредиттерді талдау тұтынушылық қарыз алушылардың ерекшеліктерін (жаман және жақсы) бірегей түсінуді көрсетті, сондай-ақ банктер берген тұтынушылық кредиттердің дұрыстығын тексеру үшін ақпарат берді.

Бұл зерттеуде біз деректердің сапасын тексеру (қажет емес айнымалыларды жою үшін ретке келтіру және тазалау рәсімдері кезінде) және көптеген санаттардың пайдасына ығысуды болдырмау үшін теңгерімсіз оқыту деректерімен жұмыс істеу өте маңызды екенін көрсеттік.

Үлгілер болжамдарының дәлдігін бағалау бөлігінде SMOTE әдісімен түзетілген артық іріктеме деректері мажоритарлық сыныпты кездейсоқ жою стратегиясымен өңделген деректермен салыстырғанда анағұрлым перспективалы нәтижелер көрсетті. Басқаша айтқанда, SMOTE арқылы түзетілген (передискредитированные) алдын ала тексерілген деректер ақпараттың жоғалуын азайтуға және болжаудың тиімділігін арттыруға көмектесті. Жақсы таңдалған оқу деректері бар үлгілер субдискредиттелген деректермен салыстырғанда, қайта дискредиттелген (передискредитированные) деректермен жақсы жұмыс істеді.

Сонымен қатар, сызықты емес үлгілер сызықтық үлгілерге қарағанда дәлірек болжамдарды көрсетті. Атап айтқанда, кездейсоқ орман сыныптауыштары және k-жақын көршілер сияқты сызықтық емес үлгілер басқа сыныптауыштар үлгілерінен асып түсті. Екінші жағынан, логистикалық регрессия және SGD сыныптауышы сияқты сызықтық үлгілер салыстырылатын сегіз үлгінің ішіндегі ең нашар нәтижелерді көрсетті.

Қорытындылай келе, реттеуші деректерге негізделген үлгілер екінші деңгейдегі банктер берген тұтынушылық кредиттер бойынша кредиттік тәуекелді бағалау үшін барабар негіз бола алады, сондай-ақ орталық банкке ықтимал жүйелік тәуекелдерді болжауға көмектеседі деп қорытынды жасауға болады. Сонымен, зерттеу нәтижелері бойынша кредиттік тәуекелді машиналық оқыту арқылы бағалау тұтынушылық кредит беретін екінші деңгейлі банктерді реттеуге жақсы қосымша бола алады.

Осы зерттеуді одан әрі дамыту үшін бірнеше бағыт болжанады. Ең алдымен, Grier (2012) алгоритмдерінің өнімділігіне оң әсер етуі мүмкін қарыз алушылардың әлеуметтік-демографиялық сипаттамаларымен қолда бар деректер жиынтығын толықтыру қажет.

Үлгілеу бөлігінде үлгілердің тиімділігін арттыратын бірнеше тәсілдер қарастырылады:

а. Машиналық оқытудың гибридті әдістерін қолдану

б. Таңдамалы аралас болжау жүйесін құру

с. Параметрлер торына және сэмплинг стратегиясына қосымша параметрлерді қосу.

Әдебиеттер тізімі

Addo, P., Guegan, D., & Hassani, B. (2018). Credit Risk Analysis Using Machine and Deep Learning Models. *Risks*, *6*(2), 38. doi: 10.3390/risks6020038

Alencar, R. (2017). Resampling strategies for imbalanced datasets. Retrieved from https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets

Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J., & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of The Operational Research Society*, *54*(6), 627-635. doi: 10.1057/palgrave.jors.2601545

Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, *39*(3), 3446-3453. doi: 10.1016/j.eswa.2011.09.033

Brownlee, J. (2016). A Gentle Introduction to XGBoost for Applied Machine Learning. Retrieved from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Brownlee, J. (2016). Metrics to Evaluate Machine Learning Algorithms in Python. Retrieved from https://machinelearningmastery.com/metrics-evaluate-machine-learning-algorithms-python/

Brownlee, J. (2020). SMOTE for Imbalanced Classification with Python. Retrieved from https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16, 321-357. Doi: 10.1613/jair.953

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *Proceedings of The 22Nd ACM SIGKDD International Conference On Knowledge Discovery and Data Mining*, 785–794. doi: 10.1145/2939672.2939785

du Boisberranger, J., Van den Bossche, J., Esteve, L., Fan, T., Gramfort, A., & Grisel, O. et al. User guide: contents – scikit-learn 0.23.2 documentation. Retrieved from https://scikit-learn.org/stable/user_guide.html

Fuchs, M. (2019). Introduction to SGD Classifier - Michael Fuchs Python. Retrieved from https://michael-fuchs-python.netlify.app/2019/11/11/introduction-to-sgd-classifier/

Geron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). Sebastopol, California: O'Reilly Media, Incorporated.

Grier, W. (2012). *Credit analysis of financial institutions* (2nd ed., pp. 294-296). London: Euromoney Institutional Investor PLC.

Hansen, C. (2019). Neural Networks: Feedforward and Backpropagation Explained. Retrieved from https://mlfromscratch.com/neural-networks-explained/#overview

Henley, W., & Hand, D. (1996). A k-Nearest-Neighbour Classifier for Assessing Consumer Credit Risk. *The Statistician*, *45*(1), 77. doi: 10.2307/2348414

Labatut, V., & Cherifi, H. (2011). Evaluation of Performance Measures for Classifiers Comparison. *Ubiquitous Computing and Communication Journal*, *6*, 21-34. Retrieved from https://arxiv.org/abs/1112.4133

Lemaitre, G., Nogueira, F., Oliveira, D., & Aridas, C. 2. Over-sampling – imbalanced-learn 0.5.0 documentation. Retrieved from https://imbalanced-learn.readthedocs.io/en/stable/over_sampling.html#smote-adasyn

Lessmann, S., Baesens, B., Seow, H., & Thomas, L. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), 124-136. doi: 10.1016/j.ejor.2015.05.030

Mueller, A., & Guido, S. (2017). *Introduction to Machine Learning with Python* (1st ed.). Sebastopol, California: O'Reilly Media.

Munkhdalai, L., Munkhdalai, T., Namsrai, O., Lee, J., & Ryu, K. (2019). An Empirical Comparison of Machine-Learning Methods on Bank Client Credit Assessments. *Sustainability*, *11*(3), 699. doi: 10.3390/su11030699

Schmarzo, B. (2018). Understanding Type 1 and Type 2 Errors [Blog]. Retrieved from https://www.datasciencecentral.com/profiles/blogs/understanding-type-i-and-type-ii-errors

Tsai, C., & Chen, M. (2010). Credit rating by hybrid machine learning techniques. *Applied Soft Computing*, *10*(2), 374-380. doi: 10.1016/j.asoc.2009.08.003

Қосымша

```
# Маңызды кітапханалар
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib
import os
path = 'E:\...'   # файл орналасқан құжатқа жолды таңдау
os.chdir(path)
df_ml = pd.read_csv('filename.csv')
X_df = df_ml.iloc[:,:-1].values
y_df = df_ml.iloc[:,-1].values

# Imblearn кітапханасынан мажоритарлық сыныпты кездейсоқ жою
from imblearn.under_sampling import RandomUnderSampler as rus
us = rus(random_state=42)
X, y = us.fit_resample(X_df, y_df)

# Imblearn кітапханасынан SMOTE-NC стратегиясы
from imblearn.over_sampling import SMOTENC
sm = SMOTENC(random_state=42, categorical_features=[0,1,2,3,4,5])  # бастапқы деректерде категориялды айнымалылар бар
X, y = sm.fit(X_df, y_df)

# Категориялды айнымалыларды кодтау
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

# категориялды айнымалылардың бірінші бағанында 18 түрлі айнымалы болды (үш қала мен аймақ (сонымен бірге бір аймақтың бұрынғы атауы)), осылайша 18 жол құрылды
ct0 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[0])], remainder='passthrough')
X=np.array(ct0.fit_transform(X))

# категориялды айнымалылардың екінші бағанында 4 түрлі айнымалы болды (қарыз берілген валютаның төрт түрі), осылайша 4 жол құрылды
ct1 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[18])], remainder='passthrough')
X=np.array(ct1.fit_transform(X))
```

```
# категориялды айнымалылардың үшінші бағанында 2 түрлі айнымалы болды (несие картасы немесе қарыз), осылайша 2 жол құрылды
ct2 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[18])], remainder='passthrough')
X=np.array(ct2.fit_transform(X))

#категориялды айнымалылардың төртінші бағанында 2 түрлі айнымалы (тұтынушылық немесе автокредиттеу) болды, осылайша 2 жол құрылды
ct3 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[18])], remainder='passthrough')
X=np.array(ct3.fit_transform(X))

# категориялды айнымалылардың бесінші бағанында 2 түрлі айнымалы (еркек немесе әйел) болды, осылайша 2 жол құрылды
ct4 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[18])], remainder='passthrough')
X=np.array(ct4.fit_transform(X))

# категориялды айнымалылардың алтыншы бағанында 2 түрлі айнымалы (резидент немесе бейрезидент) болды, осылайша 2 жол құрылды
ct5 = ColumnTransformer(transformers=[('encoder', OneHotEncoder(),[18])], remainder='passthrough')
X=np.array(ct5.fit_transform(X))

# y айнымалы Label Encoder
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
y = le.fit_transform(y)

# деректерді оқыту және тестілеу жиынтығына бөлу
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=1)

# шкалалау
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train[:,-3:] = sc.fit_transform(X_train[:,-3:])
X_test[:,-3:] = sc.transform(X_test[:,-3:])

# Негізгі компоненттер әдісі (Principal component analysis (PCA))
from sklearn.decomposition import PCA
pca = PCA(n_components=a)    # a is quantity where cumulative explained variance ratio > 95%
```

```
X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

# Логистикалық регрессия
from sklearn.linear_model import LogisticRegression
log = LogisticRegression().fit(X_train,y_train)
y_tr_log_pred = log.predict(X_train)
y_ts_log_pred = log.predict(X_test)

# Стохастикалық градиенттің түсуі
from sklearn.linear_model import SGDClassifier
sgd=SGDClassifier().fit(X_train,y_train)
y_tr_sgd_pred = sgd.predict(X_train)
y_ts_sgd_pred = sgd.predict(X_test)

# Гаусс қарапайым Байес сыныптауышы
from sklearn.naive_bayes import GaussianNB
nbc = GaussianNB().fit(X_train,y_train)
y_tr_nb_pred = nbc.predict(X_train)
y_ts_nb_pred = nbc.predict(X_test)

# k-жақын көршілер
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier().fit(X_train,y_train)
y_tr_knn_pred = knn.predict(X_train)
y_ts_knn_pred = knn.predict(X_test)

# Шешімдер тізбегі
from sklearn.tree import DecisionTreeClassifier
dtc=DecisionTreeClassifier(max_depth=14,
criterion='entropy').fit(X_train,y_train)
y_tr_dt_pred = dtc.predict(X_train)
y_ts_dt_pred = dtc.predict(X_test)

# Кездейсоқ орман
from sklearn.ensemble import RandomForestClassifier
rfc=RandomForestClassifier().fit(X_train,y_train)
y_tr_rf_pred = rfc.predict(X_train)
y_ts_rf_pred = rfc.predict(X_test)

# Көпқабатты персептрон
from sklearn.neural_network import MLPClassifier
nnc = MLPClassifier().fit(X_train,y_train)
y_tr_nnc_pred = nnc.predict(X_train)
```

```
y_ts_nnc_pred = nnc.predict(X_test)

# XGBoost
from xgboost import XGBClassifier
xgb = XGBClassifier().fit(X_train,y_train)
y_tr_xgb_pred = xgb.predict(X_train)
y_ts_xgb_pred = xgb.predict(X_test)

# Кросс-валидация
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = xgb, X = X_train, y = y_train, cv = 5)
print('Accuracy: {:.2f} %'.format(accuracies.mean()*100))
print('Standard deviation: {:.2f} %'.format(accuracies.std()*100))

### Ескерту: тоғыспалы тексеру нәтижесін талдау үшін XGB орнына басқа үлгілерді қоюға болады.

# Логистикалық регрессия үшін торды іріктеу
from sklearn.model_selection import GridSearchCV
parameters = [{'penalty': ['none'], 'solver':['newton-cg', 'sag', 'saga', 'lbfgs']},
              {'penalty': ['elasticnet'], 'C': [0.01, 0.1, 0.25, 0.5, 0.75, 1, 5, 10],
'solver':['saga']},
              {'penalty': ['l2'], 'C': [0.01, 0.1, 0.25, 0.5, 0.75, 1, 5, 10], 'solver':['newton-
cg', 'sag', 'saga', 'lbfgs']}]
grid_search = GridSearchCV(estimator = log,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           n_jobs = -1)
grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print('Best accuracy: {:.2f} %'.format(best_accuracy*100))
print('Best parameters: ',best_parameters)

# Түсудің стохастикалық градиенті үшін тор бойынша сұрыптау
from sklearn.model_selection import GridSearchCV
parameters = [{"loss" : ["hinge", "log", "squared_hinge", "modified_huber"],
               "alpha" : [0.0001, 0.001, 0.01, 0.1], "penalty" : ["l2", "l1", "none"]}]
grid_search = GridSearchCV(estimator = sgd,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           n_jobs = -1)
```

```
grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print('Best accuracy: {:.2f} %'.format(best_accuracy*100))
print('Best parameters: ',best_parameters)

# Гаусс қарапайым Байес сыныптауышы үшін тор бойынша сұрыптау
from sklearn.model_selection import GridSearchCV
parameters = [{'var_smoothing': [1e-12,1e-10,1e-7,1e-4,1e-3,1e-2,1e-1,1,10]}]
grid_search = GridSearchCV(estimator = nbc,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           n_jobs = -1)
grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print('Best accuracy: {:.2f} %'.format(best_accuracy*100))
print('Best parameters: ',best_parameters)

# К-жақын көршілер үшін тор бойынша сұрыптау
from sklearn.model_selection import GridSearchCV
parameters = [{'n_neighbors': list(range(1, 81))}]
grid_search = GridSearchCV(estimator = knn,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           n_jobs = -1)
grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print('Best accuracy: {:.2f} %'.format(best_accuracy*100))
print('Best parameters: ',best_parameters)

# Шешімдер тізбегі үшін тор бойынша сұрыптау
from sklearn.model_selection import GridSearchCV
parameters = [{'criterion':['gini', 'entropy'],'max_depth': list(range(1,21))}]
grid_search = GridSearchCV(estimator = dtc,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           n_jobs = -1)
grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
```

```
best_parameters = grid_search.best_params_
print('Best accuracy: {:.2f} %'.format(best_accuracy*100))
print('Best parameters: ',best_parameters)

# Кездейсоқ орман үшін тор бойынша сұрыптау
from sklearn.model_selection import GridSearchCV
parameters = [{'n_estimators': list(range(1,21)), 'max_features': ['auto', 'sqrt', 'log2'],
            'max_depth' : ['None',8], 'criterion' :['gini', 'entropy']}]
grid_search = GridSearchCV(estimator = rfc,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           n_jobs = -1)
grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print('Best accuracy: {:.2f} %'.format(best_accuracy*100))
print('Best parameters: ',best_parameters)

# Көп қабатты персептрон үшін тор бойынша сұрыптау
from sklearn.model_selection import GridSearchCV
parameters = [{'hidden_layer_sizes':[100,200,300,[200,50],[100,100],[200,100]],
            'activation':['identity','logistic','tanh','relu'],
            'solver': ['adam'],
            'learning_rate':['constant','invscaling','adaptive'],
            'max_iter': [1000,1500,2000 ]}]
grid_search = GridSearchCV(estimator = nnc,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           n_jobs = -1)
grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print('Best accuracy: {:.2f} %'.format(best_accuracy*100))
print('Best parameters: ',best_parameters)


# XGBoost үшін тор бойынша сұрыптау
from sklearn.model_selection import GridSearchCV
parameters = [{'n_estimators': [1000], #number of trees, change it to 1000 for better
results
            'max_depth': [6,7,8],
            'learning_rate': [0.05], #so called `eta` value
```

```
            'objective':['binary:logistic'],
            'tree_method':['exact'],
            'min_child_weight': [11],
            'subsample': [0.8],
            'colsample_bytree': [0.7]}]
grid_search = GridSearchCV(estimator = xgb,
                           param_grid = parameters,
                           scoring = 'accuracy',
                           cv = 5,
                           n_jobs = -1)
grid_search.fit(X_train, y_train)
best_accuracy = grid_search.best_score_
best_parameters = grid_search.best_params_
print('Best accuracy: {:.2f} %'.format(best_accuracy*100))
print('Best parameters: ',best_parameters)
```

Торды іріктегеннен кейін, ең жақсы дәлдікпен үлгі жоғары дәлдік беретініне сенімді болу үшін барлық жақсы параметрлерді қолдану керек, содан кейін басқа өлшемдерді талдау керек.

```
# Метрикалар
from sklearn.metrics import confusion_matrix, accuracy_score
knn_cm_tr = confusion_matrix(y_train,y_tr_knn_pred)
print(knn_cm_tr)
accuracy_score(y_train, y_tr_knn_pred)

knn_cm_ts = confusion_matrix(y_test,y_ts_knn_pred)
print(knn_cm_ts)
accuracy_score(y_test, y_ts_knn_pred)

from sklearn.metrics import roc_auc_score, jaccard_score, f1_score, precision_score, recall_score
print(roc_auc_score(y_test,y_ts_knn_pred))
print(jaccard_score(y_test,y_ts_knn_pred))
print(f1_score(y_test,y_ts_knn_pred))
print(precision_score(y_test,y_ts_knn_pred))
print(recall_score(y_test,y_ts_knn_pred))
```

Мұнда метрикалар k-жақын көршілер сыныптауышының жұмысын талдау үшін пайдаланылды, басқа үлгілердің нәтижесіне жету үшін айнымалылар өзгертілуі керек.