Предговор

Проблеми попут одређивања регуларности трансакција, препознавања да ли је новински чланак из области спорта, економије или политике или препознавање једне од скупа познатих личности чије се лице налази на фотографији спадају у проблеме *класификације.* Класификација, поред регресије, представља једну од две основне врсте проблема *надгледаног учења*. Тај вид машинског учења карактерише то да су подаци организовани у парове описа онога на основу чега се учи и онога што је из тога потребно научити, тј. уз вредности улаза дате су и вредности излаза. Поменути проблем разврставања трансакција на основу валидности би садржао поред трансакције и ознаку да ли је регуларна или не, што је аналогно процесу учења када наставник уз задатке приложи и решења како би ученик могао да провери своје одговоре. Код надгледаног учења треба уочити који однос важи између улаза и излаза да би се за будуће улазе на основу тог односа предвиђао излаз.

Након увођења појмова *атрибути* (енгл. *features*) за вредности улазних променљивих и *циљна променљива* (енгл. *target variable*) за излаз, може се рећи да је класификација проблем предвиђања категоричке циљне променљиве. Под категоричким променљивама подразумевају се оне променљиве које имају коначан број вредности које нису уређене ни на који начин. Насупрот класификацији, регресија се дефинише као проблем предвиђања непрекидне циљне променљиве. Такав проблем је предвиђање цена деоница на берзи на основу њихових цена у претходних неколико дана и глобалних квантитативних показатеља тржишта.

Постоје различите методе решавања проблема класификације. Ниједна од њих се не понаша најбоље у свим могућим ситуацијама. Нпр. алгоритам који је лак за разумевање и за који се верује да је један од најбржих метода за идентификацију најзначајнијег атрибута као и везе између једног или више атрибута јесте стабло одлучивања. Међутим, стабло одлучивања има и мане од којих је најуобичајнија преприлагођавање (енгл. *overfitting*). Могуће је ублажити или у потпуности уклонити недостатке стабла одлучивања применом више стабала. Размотримо следеће реалне ситуације. Када се особа одлучи за куповину аутомобила, неће одмах отићи у салон аутомобила и купити први који угледа, већ ће се консултовати са неколико особа у свом окружењу, узеће њихово мишљење у обзир и истраживаће пре него што донесе коначну одлуку. Слично, приликом планирања одмора људе интересује више туђих мишљења о дестинацијама. Наиме, једна особа може да буде пристрасна и њено мишљење о одређеној дестинацији ће се заснивати на сопственом искуству. Зато је потребно питати више људи како би се уклонила пристрасност једне особе чије је мишљење можда изузетак.

Други пример је ситуација с којом се сусрећемо на интервјуима за посао. Наиме, потребно је проћи више кругова испитивања иако су питања слична, а неретко и иста. Разлог је што у компанијама желе да чују мишљења више својих запослених. Ако више њих лоше оцени кандидата, велика је вероватноћа да ће тај кандидат бити одбијен.

У свету машинског учења овакав приступ се назива *ансамбл* (енгл. *ensemble*) *метода*. Ансамбл метода је тип надгледаног учења код ког се комбинује скуп модела који решавају оригинални проблем. Класификацију ансамблом метода карактерише формирање великог скупа слабих класификатора чији се излази комбинују у једно решење уместо коришћења једног јаког класификатора.

Насумична шума (енгл. *Random Forest*) је моћан алгоритам ансамбл машинског учења кога чине више стабала одлучивања. Због комплексности алгоритма, великог броја стабала који изграђују шуму и због насумичног одабира атрибута за обучавање сваког појединачног стабла, неизводљиво је испитивати свако појединачно стабло ради потпуног разумевања начина функционисања алгоритма, те се због тога често у литератури алгоритам насумичне шуме третира као *црна кутија*. Понекад је неопходна интерпретација модела као у ситуацији коју описује први пример у овом уводном делу. Наиме, нека је модел детекције сумњивих трансакција заснован на алгоритму насумичне шуме. Можда аналитичар, након што трансакција буде сврстана у сумњиве, пожели да зна зашто је модел донео ту одлуку, који атрибут је највише утицао на доношење такве одлуке. Претпоставимо да је модел очекивано радио на старом скупу података, али је променио понашање на новом скупу. Било би пожељно испитати који атрибут је довео до неочекиваних резултата. Једна од идеја овог рада јесте да покуша да помоћу испитивања основних параметара алгоритма, попут броја стабала који граде шуму, дубине стабла и сл., црну кутију претвори у белу.

У наставку рада ће бити речи о ансамблима и учењу ансамбала паковањем. Даље ће бити детаљно објашњена метода насумичне шуме и дати примери њене употребе како на једноставним и добро припремљеним скуповима података, тако и у студији случаја у завршном делу рада која потврђује описане теоретске концепте на једном реалном скупу података.

1. Ансамбл методе

Преприлагођавање поменуто у уводу карактерише се постојањем високе *варијансе* и ниског *бајаса* (енгл. *bias*, пристрасност, систематско одступање, померај) и односи се на ситуацију када модел савршено научи да врши предикцију на скупу података за тренинг, али има малу способност предвиђања на новом ненаученом скупу података. Са сложеношћу модела повећава се и варијанса. Модел би требало да одликује баланс поменутих грешака, познат као нагодба између систематског одступања и варијансе (енгл. *bias-variance tradeoff*). Како би се смањила грешка предвиђања, један од приступа је да се смањи варијанса, а при томе задржи исти бајас или се незнатно увећа. Ансамбл методама је то могуће учинити. Нека је $\mathcal{L}$ скуп података за учење. Основни принцип на ком се ансамбл заснива јесте да се из скупа $\mathcal{L}$ случајно одаберу скупови података за обучавање сваког појединачног модела. На тај начин се генерише више различитих модела, а затим се комбинацијом предикција појединачних модела формира предикција ансамбла.

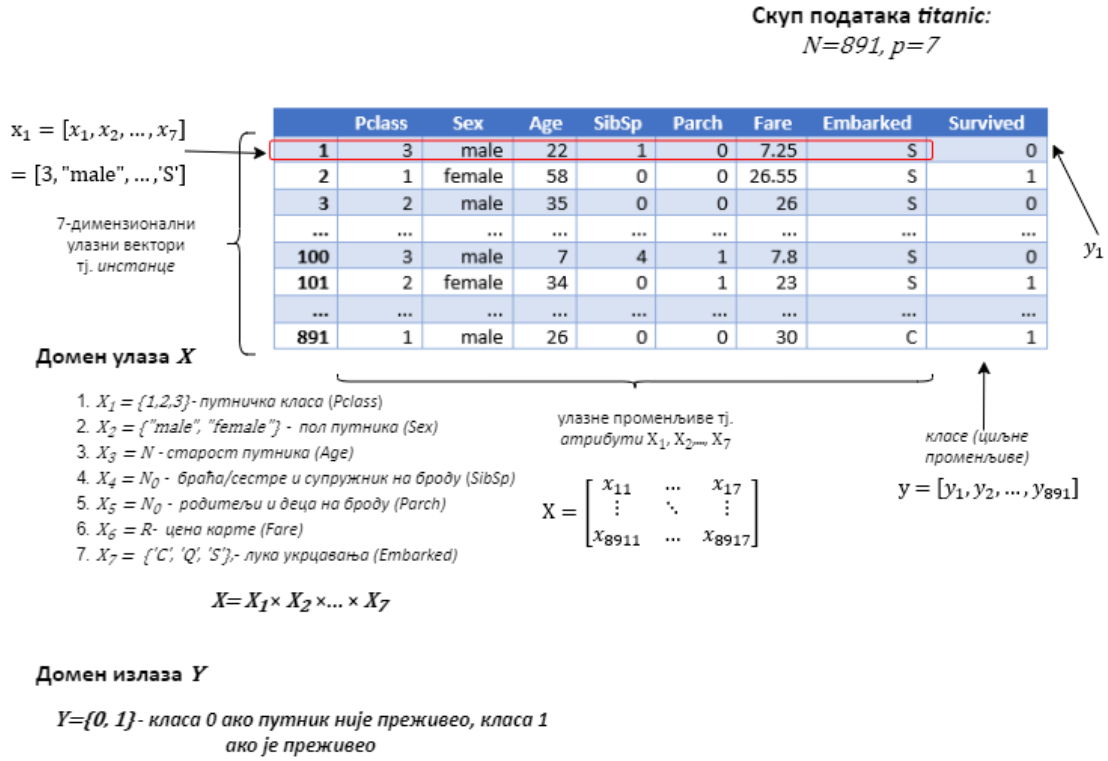
Најуобичајније методе ансамбл учења су *паковање* (енгл. *bagging*) и *појачавање* (енгл. *boosting*). Оба начина комбиновања подразумевају одабир хомогених базних модела, тј. модела истог типа. Главна разлика је у томе што се паковање користи за добијање ансамбла који карактерише нижа варијанса од варијансе компоненти, а појачавањем се добија ансамбл с мањим бајасом од бајаса базних модела. Како стабла одлучивања паковањем формирају насумичну шуму, у овом раду ће бити обрађено само паковање.

1.1. Формалне дефиниције

Да би се објаснио начин комбиновања предвиђања појединачних модела, прво је потребно увести додатне ознаке и дати формалну дефиницију класификације. Нека улазне вредности $x_1, x_2, \dots, x_p$, где је $x_j \in X_j$, ($j = 1, \dots, p$) одговарају улазној променљивој X_j. Улазне вредности заједно $[x_1, x_2, \dots, x_p]$ чине p-димензионални улазни вектор x који узима вредности из $X_1 \times \dots \times X_p = X$, где је X домен улаза. Улазни вектор се назива често и *инстанца* или примерак. Слично, са $y \in Y$ означава се вредност излазне променљиве тј. циљне променљиве Y, где је Y домен излаза[1].

Дефиниција 1.1.1. Скуп података за учење $\mathcal{L}$ је скуп N парова улазних вектора и излазних вредности $(\mathrm{x}_1, y_1), \dots, (\mathrm{x}_N, y_N)$ где су $\mathrm{x}_i \in X$ и $y_i \in Y$.

[1] Ако се не нагласи другачије, надгледано учење подразумева предвиђање једне излазне вредности.



Илустрација 1: Скуп података за учење – проблем предвиђања ко је преживео потонуће Титаника[2]

Посматрајмо пример који приказује илустрација *1*. Потребно је предвидети да ли је путник на Титанику преживео на основу његовог економског статуса, пола, старости, тога да ли је на броду био сам или с неким од чланова породице, колико је платио карту и у којој се луци укрцао на брод. Наведене карактеристике помоћу којих се врши предвиђање судбине путника јесу улазне променљиве или атрибути које смо означили са X_j, (j = 1, ... , p). Скуп вредности које може да има j-ти атрибут означен је са X_j. За скуп података *titanic*, на илустрацији *1*, дато је које вредности чине сваки од скупова вредности доступних атрибута. Декартовим производом скупова вредности свих атрибута добија се скуп уређених p-торки које су облика $[x_1, x_2, \dots, x_p]$. Тај скуп представља домен улаза X. Узмимо нпр. инстанцу под редним бројем 3 скупа података са илустрације *1*. Очигледно припада домену X, јер је можемо записати као уређену седморку $[2, \text{"male"}, 35, 0, 0, 26, 'S']$. Путник ком одговара тај улазни вектор није преживео, те је $y_3 = 0$. Како циљна променљива у овом примеру може, поред вредности 0, имати вредност 1 (ако је путник преживео), домен излаза је скуп $\{0, 1\}$.

Аналогно представљању једног улазног вектора, скуп од N p-димензионалних улазних вектора x_i $(i = 1, \dots, N)$ може се записати као матрица X димензија $N \times p$, чије врсте представљају улазне векторе x_i, $i = 1, \dots, N$, а колоне одговарају атрибутима X_j, $j = 1, \dots, p$. Слично, излазне вредности могу се представити вектором $\mathrm{y} = [y_1, y_2, \dots, y_N]$. Илустрација *1* показује на скупу података *titanic* како се добија матрица инстанци и вектор циљних класа. У овом контексту задатак надгледаног учења јесте учење функције φ: $X \to Y$ из скупа података за учење $\mathcal{L} = (\mathrm{X}, \mathrm{y})$. Циљ је пронаћи такав модел да су његова предвиђања $\varphi(\mathrm{x})$ најбоља могућа.

[2] Цео скуп података доступан на: https://www.kaggle.com/c/titanic/data.

Нека је Y категоричка променљива. Тада:

Дефиниција 1.1.2. *Класификатор* или правило класификације је функција φ: $X \to Y$, где је Y коначан скуп класа или лабела $\{c_1, c_2, \ldots, c_p\}$.

Правило регресије би била функција φ: $X \to Y$, где је $Y = \mathbb{R}$.

Нека је A алгоритам учења и нека је $A(\theta, \mathcal{L})$ модел који ћемо означавати са $\varphi_{\mathcal{L},\theta}$, где је θ параметар који контролише извршавање алгоритма A. Претпоставимо да је θ насумични параметар (енгл. *random seed*) задужен за симулацију стохастичког понашања A, те генерисања псеудонасумичних модела који се разликују један од другог у зависности од θ.

Нека је $\{\varphi_{\mathcal{L},\theta_m} | \, m = 1, \ldots, M\}$ скуп од M насумично одабраних модела, који су обучавани на истом скупу података $\mathcal{L}$, али је сваки од њих изграђен независним параметром θ_m. Нови ансамбл модел који предвиђа комбинацијом предикција ових појединачних модела означава се са $\psi_{\mathcal{L},\theta_1,\ldots,\theta_M}$.

2. Паковање

Паковање (енгл. *bagging*[3], *bootstrap aggregating*) представља једну од метода учења ансамбала и спада у паралелне методе тј. омогућава тренирање више модела који не зависе један од другог, тако да их је могуће тренирати конкурентно.

Идеја паковања проистиче из чињенице да модел који се добија као резултат тренирања зависи од тренинг података: да је коришћен други скуп података, добио би се други модел. Паковање се заснива на обучавању више независних модела и упросечавању њихових предикција како би резултујући модел имао што нижу варијансу. У пракси је немогуће обезбедити потпуно независне моделе, јер би то захтевало превише података. Да би се превазишао овај проблем, користи се бустрепинг техника која обезбеђује тренинг скупове који су приближно независни и репрезентативни.

2.1. Бутстрепинг

Бутстрепинг представља технику у статистици којом се из почетног скупа података величине N генеришу узорци величине B, тзв. бутстреп узорци (енгл. *bootstrap samples*) тако што се насумично бира B елемената из почетног скупа с понављањем.

Ако је $(1, 2, 3, 4, \dots, 11, 12)$ скуп података којим располажемо и ако је потребно креирати бутстреп узорак величине 5, могли бисмо добити скупове као на илустрацији *2*. Један од бустреп узорака је $(8, 2, 3, 9, 3)$ у ком примећујемо два појављивања елемента 3 јер су се елементи бирали с понављањем, док су појављивања неких елемената изостала.

Илустрација 2: Генерисање бутстреп узорака

[3] Назив *bagging* је Лео Брејман (1928 – 2005) 1996. дао техници ансамбл машинског учења која је до тад била позната као *bootstrap aggregating*. Брејман је био истакнути статистичар који је успео да превазиђе јаз између статистике и рачунарских наука, нарочито у области машинског учења. Такође је творац и алгоритма насумичне шуме (2001).

Под одређеним претпоставкама, тако добијени узорци имају добре статистичке карактеристике. Наиме, могу се сматрати репрезентативним и независним узорцима стварне расподеле података (енгл. *i. i. d. samples, independent identically distributed*). Репрезентативност је испуњена ако је N довољно велико да почетни скуп података може да обухвати скоро сву сложеност основне дистрибуције тако да узорковање иницијалног скупа података представља апрокцимацију узорковања стварне диструбуције. Да би узорци били независни, потребно је да N буде довољно велико у односу на B да би се што више избегла корелација између узорака. Код неких ансамбала који настају паковањем важи $N=B$, при чему обучавање модела таквих ансамбала које се одвија на насумично добијеном узорковању карактеришу и неки додатни случајни избори.

Бутстрепинг се често користи за процењивање варијансе статистичких прорачуна. Да би се проценила варијанса, потребно је израчунати варијансу на неколико независних узорака генерисаних неком интересном расподелом. У већини случајева потребно је много више података него што је доступно да би се узорци могли сматрати репрезентативним и независним. Тада се генеришу бутстреп узорци који се могу сматрати скоро репрезентативним и скоро независним (енгл. *almost i. i. d.*).

Ако би узорци били потпуно независни, идентично дистрибуирани и насумично конструисани, сваки са варијансом σ^2, просек од M таквих узорака би имао варијансу $\frac{1}{M}\sigma^2$. Међутим, како постоји нека корелација ρ између узорака, варијанса постаје $\rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$. Са порастом M други сабирак тежи нули, али први остаје и одатле се уочава да јачина корелације ограничава добити упросечавања, те отуда идеја да се насумично одабирају нови скупови за тренинг како би ρ било што мање.

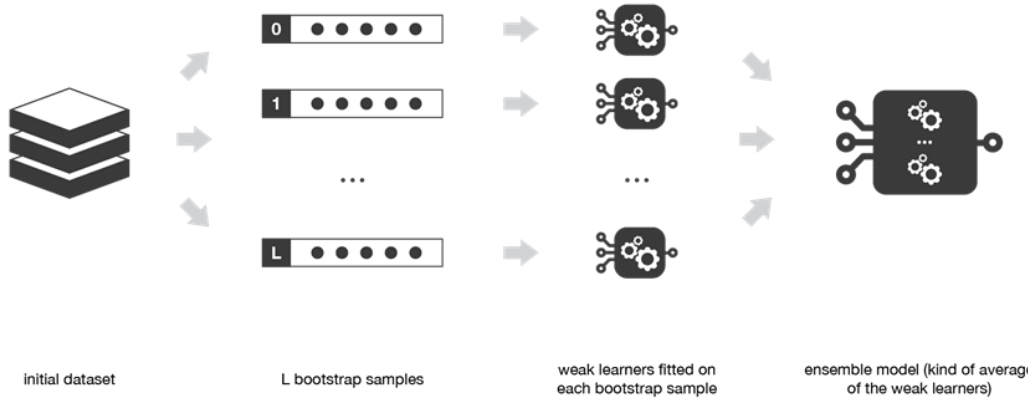
Приликом генерисања узорака тј. нових скупова за тренинг запажа се и следеће:

- Неке инстанце се могу појавити у више скупова.
- Неке инстанце се могу појавити више пута у једном скупу.
- За сваки нови скуп вероватноћа да буде изабрана било која инстанца је $\frac{1}{N}$, односно да не буде изабрана $1-\frac{1}{N}$. Пошто се инстанце бирају с понављањем вероватноћа да N примерака неће бити међу случајно одабраним постаје $\left(1-\frac{1}{N}\right)^N$. За довољно велико N вероватноћа појављивања инстанце у новом скупу конвергира ка $1-\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^N = 1-e^{-1} = 0{,}6322$, тј. вероватноћа $p(\mathrm{x} \in \mathcal{L}_i) \to 0{,}6322$, што значи да мање од $\frac{2}{3}$ примерака се појављује у једном бутстреп узорку.

2.2. Учење ансамбла паковањем

Први корак у учењу ансамбала паковањем је креирање бутстреп узорака тако да сваки од њих представља посебан, готово независан тренинг скуп (Илустрација *3*). Потом

се сваки од слабих базних модела обучава на једном бустреп узорку. Даље се неком техником агрегације и упросечавања излаза добија ансамбл модел.



Илустрација 3: Формирање ансамбла

Постоји више начина комбиновања предикција чланова ансамбла у коначан излаз. Код регресионих проблема, излаз ансамбла може бити просек излаза појединачних модела. У случају класификације, у ансамблу се спроводи гласање о чему ће више бити речено у наставку рада.

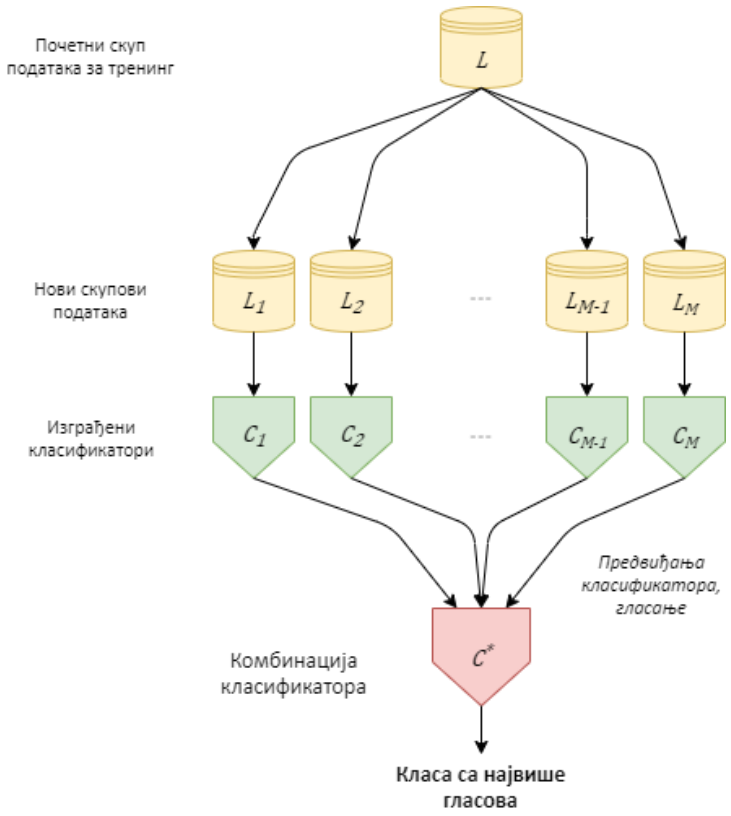
2.2.1. Гласање ансамбла

Приликом класификације модели у ансамблу се посматрају као чланови одбора и резултат ансамбла се добија гласањем. Упоредимо понашање ансамбла са следећом реалном ситуацијом. Претпоставимо да је предлог буџета прослеђен Влади на разматрање. Треба да се одржи седница на којој ће група министара одлучивати о усвајању предлога. Ако би председник или премијер самостално одлучивали, то би се могло протумачити као знак диктатуре, а донета одлука можда не би била у корист народа. Описана аналогија одговара одлучивању појединачног модела попут стабла одлучивања, логистичке регресије итд., чије ће перформансе увек бити за нијансу лошије од перформанси ансамбла. Нека седници присуствују четири министра. Сваки од њих износи и образлаже своје мишљење о датом предлогу. Коначна одлука се доноси на основу већинског гласања. Нека је један министар за одбијање предложеног буџета, а остала тројица за усвајање – предлог ће бити усвојен. Управо тако се врши предвиђање ансамбла. Описано већинско гласање, познато као чврсто гласање (енгл. *hard voting*) можемо записати и математички коришћењем функције *губитак нула-један* (енгл. *zero-one loss*), функције губитка уобичајене за класификацију. Губитку се додељује 0 у случају да је класификација тачна, односно 1 ако је класификација нетачна.

Агрегација предикција ансамбла, у случају губитка нула-један, врши се третирањем модела у ансамблу као члановима одбора и прибегава се *већинском гласању* ради формирања предикције. m-ти ($m = 1, \dots, M$) модел даје један глас за класу c ако је $\varphi_{\mathcal{L},\theta_m}(\mathrm{x}) = c$, тј. ако инстанцу x класификује у класу c. Глас m-тог модела за класу c се

означава са $\mathbf{1}_{(\varphi_{\mathcal{L},\theta_m}(\mathrm{x})=c)}$. Класа која добије највише гласова је класа коју као резултат даје ансамбл:

$$\psi_{\mathcal{L},\theta_1,\ldots,\theta_M}(\mathrm{x}) = \underset{c \in y}{\operatorname{argmax}} \sum_{m=1}^{M} \mathbf{1}_{(\varphi_{\mathcal{L},\theta_m}(\mathrm{x})=c)} .$$



Илустрација 4: Класификација ансамбл методом - већинско гласање

Формирање ансамбла и технику предвиђања већинско гласање приказује илустрација *4*. Доступан скуп података за обучавање је L. Случајним бирањем примера из L добијени су скупови $L_1, L_2, \ldots, L_M$ за тренирање класификатора $C_1, C_2, \ldots, C_M$, респективно. Сваки од класификатора нуди своје предвиђање дајући глас за једну класу. Класа у коју је највише класификатора сврстало инстанцу са улаза представља предвиђање ансамбла. Претпоставимо да је инстанцу за коју је потребно дати предвиђање могуће класификовати у неку од следеће три класе: A, B или C. Да би пример био што једноставнији нека класификатора има 6 и нека је предикција првог класа A, другог и петог B и преосталих три C. Ансамбл C^*са илустрације *4* би у том случају донео одлуку да инстанца припада класи C.

Већинско гласање потиче из области политичких наука, тачније заснива се на Кондорсетовој теореми пороте. Претпоставимо да група од M гласача жели да донесе одлуку већинским гласањем. Теорема тврди да ако за сваког гласача постоји независна вероватноћа $p > \frac{1}{2}$ да ће да гласа за исправну одлуку, онда што је више гласача, то је већа вероватноћа да ће већинска одлука бити исправна. Када $M \to \infty$, вероватноћа да ће група донети исправну одлуку приближава се јединици. Обрнуто, ако $p < \frac{1}{2}$, онда је

вероватније да ће сваки гласач гласати погрешно и повећавање M ће погоршати ситуацију.

Претпоставимо да ансамбл чини двадесет и пет класификатора, тј. $M = 25$. Нека је за сваки класификатор проценат погрешно класификованих примерака $\varepsilon = 0{,}35$ и нека вероватноћа да ће класификатор погрешно предвидети не зависи од грешке предвиђања других класификатора. Ансамбл ће направити погрешно предвиђање ако већина класификатора предвиди погрешно. Вероватноћа да ће 13 или више класификатора дати нетачну предикцију износи:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} \approx 0{,}06 \ll \varepsilon.$$

Поред већинског гласања, постоји и тзв. меко гласање (енгл. *soft voting*). Меко гласање користи се пре свега када ансамбл сачињавају модели који као резултат класификације враћају вероватноћу да инстанца припада некој класи, попут логистичке регресије. Дакле, разматрају се вероватноће за сваку класу које су вратили сви модели, а као излаз ансамбла се задржава она класа чија је просечна вероватноћа највећа. m-ти ($m = 1, \dots, M$) модел процењује за свако $c \in \mathcal{Y}$ која је вероватноћа да пример x припада класи c у ознаци $\hat{p}_{\mathcal{L},\theta_m}(Y = c \mid X = \mathrm{x})$. Тада се коначна предикција добија на следећи начин:

$$\psi_{\mathcal{L},\theta_1,\dots,\theta_M}(\mathrm{x}) = \underset{c \in \mathcal{Y}}{\operatorname{argmax}} \frac{1}{M} \sum_{m=1}^{M} \hat{p}_{\mathcal{L},\theta_m}(Y = c \mid X = \mathrm{x}).$$

Наредни пример илуструје разлике између доношења одлуке већинским гласањем и техником *soft voting*. Претпоставимо да три различита класификатора граде ансамбл који изводи бинарну класификацију. Табела *1* показује да је сваки класификатор предвидео конкретну класу за неко x. У овом случају једноставним пребројавањем гласова се добија да је *класа 1* предвиђање ансамбла. Нешто комплексније је израчунати предвиђање ансамбла у ситуацији коју описује табела *2* – сва три класификатора су предвидела вероватноћу припадања инстанце x обема класама. У овом примеру први класификатор је 70% сигуран да је реч о инстанци *класе 1*, други процењује 50% за обе, а трећи се слаже са првим да x припада *класи 1*. Израчуна се просечна вероватноћа за сваку класу:

- За *класу 0*: $\frac{0{,}3 + 0{,}5 + 0{,}4}{3} = 0{,}4$ и
- За *класу 1*: $\frac{0{,}7 + 0{,}5 + 0{,}6}{3}$ = 0,6.

Како је већа просечна вероватноћа да x припада *класи 1*, ансамбл би као своје предвиђање дао *класу 1.*

Класификатор	Предикција
Класификатор 1	Класа 1
Класификатор 2	Класа 1
Класификатор 3	Класа 0

Табела 1: Hard voting

Класификатор	Класа 0	Класа 1
Класификатор 1	0,3	0,7
Класификатор 2	0,5	0,5
Класификатор 3	0,4	0,6

Табела 2: Soft voting

Уколико је реч о регресији, агрегација предикција се врши упросечавањем:

$$\psi_{\mathcal{L},\theta_1,\ldots,\theta_M}(\mathrm{x}) = \frac{1}{M}\sum_{m=1}^{M} \varphi_{\mathcal{L},\theta_m}(\mathrm{x}).$$

3. Насумична шума

Пре детаљног објашњења методе насумичне шуме потребно је дати уопштен опис базног модела чијом простом агрегацијом она настаје – стабала одлучивања.

3.1. Стабла одлучивања

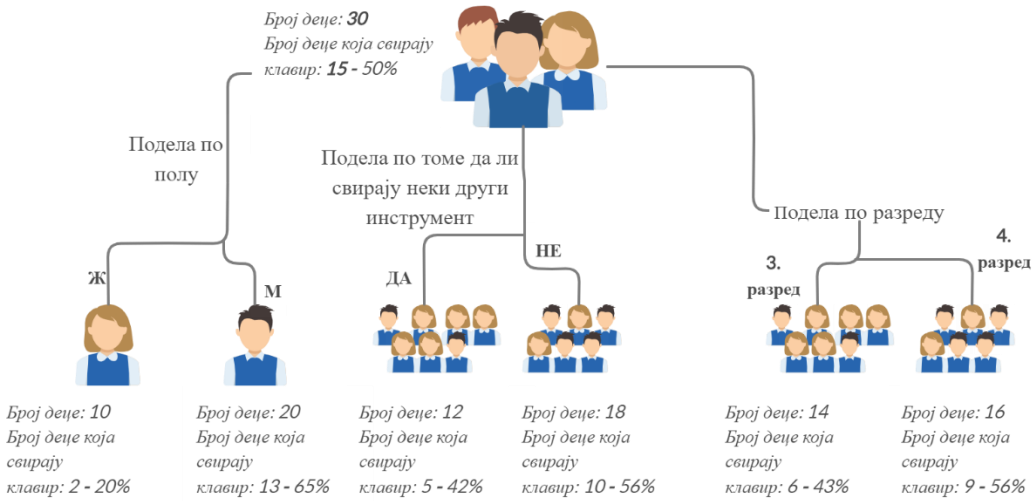
Стабла одлучивања представљају једноставан и интерпретабилан модел машинског учења који је применљив и на регресију и на класификацију. У сваком чвору стабла који није лист налази се по један тест. Сваки тест има више од једног исхода, а сваком исходу одговара грана која води до следећег чвора. Листови су означени вредностима које представљају предвиђања стабла. Тестови који се врше су углавном провере вредности појединачних атрибута. Ако је реч о категоричком атрибуту, ради се провера једнакости с неком конкретном вредношћу атрибута или се проверава припадност неком скупу вредности. У случају непрекидних атрибута, тест обично представља проверу да ли је вредност атрибута већа или мања од неке вредности.

Идеја је формулисати такав тест у корену стабла који би поделио инстанце скупа за обучавање на више група које су што хомогеније, а међусобно хетерогене. Затим рекурзивно креирати стабла над тим подскуповима и повезати их на корен укупног стабла. Дакле, потребно је дефинисати начин на који се формулише тест у неком чвору, критеријум заустављања и начин одређивања вредности у листовима. Ти избори се доносе у складу с неком хеуристиком. Неки од познатих приступа за квантификовање хомогености скупа инстанци, тј. за мерење чистоће (енгл. *purity*) добијених подгрупа су *Џини индекс* и *ентропија*. Оба приступа се заснивају на уделима инстанци различитих класа у укупном скупу.

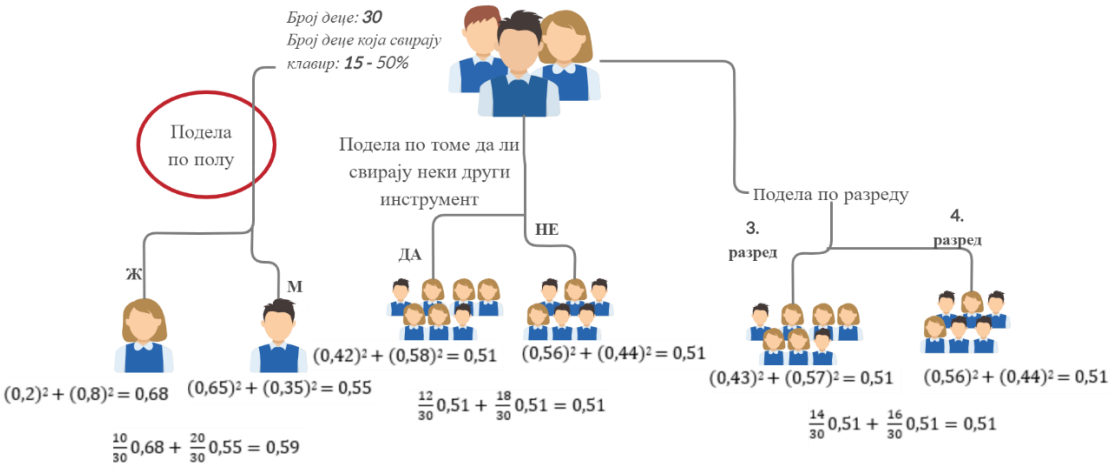
Џини индекс се рачуна тако што се од јединице одузме сума квадрата вероватноћа свих класа:

$$Gini = 1 - \sum_{i=1}^{c} p_i^2.$$

За атрибут који ће бити у неком чвору одлуке бира се онај чији је Џини индекс најмањи. Наиме, што је мањи Џини индекс, већа је хомогеност. Размотримо следећи пример са илустрације *5*. Од тридесеторо деце петнаесторо свира клавир у слободно време. Децу описују три атрибута: пол (М или Ж), разред (3. или 4. разред) и да ли свирају неки други инструмент (ДА или НЕ). Треба формирати стабло које ће да предвиђа ко свира клавир у слободно време. Илустрација *5* приказује како су деца подељена у групе на основу сваког атрибута. Израчунавањем Џини индекса за сваки атрибут одредићемо који је атрибут најзначајнији, самим тим који ће се налазити у корену стабла.



Илустрација 5: Предвиђање ко свира клавир – подела по атрибутима



Илустрација 6: Предвиђање ко свира клавир – рачунање Џини индекса

На основу израчунатих пондерисаних вредности Џини индекса за сваки атрибут, датих на илустрацији *6*, закључује се да је атрибут који треба да се нађе у корену стабла – пол.

Ентропија се дефинише као:

$$Entropy = \sum_{i=1}^{c} -p_i \log_2 p_i.$$

Представља заправо меру нехомогености и као и у случају Џини индекса, пожељније су мање вредности, а вредност нула је идеална.

Што се тиче критеријума заустављања рекурзивног поступка креирања стабла, постоје разни. Ако је реч о категоричким атрибутима као у примеру са илустрације *5*, очигледно је да дубина стабла не може да буде већа од броја атрибута. Некада се дубина експлицитно ограничава и рекурзивни поступак зауставља на некој унапред одређеној

дубини. Могуће је и унапред дефинисати неку ниску меру нехомогености и зауставити поступак када мера нехомогености подскупова инстанци достигне ту вредност.

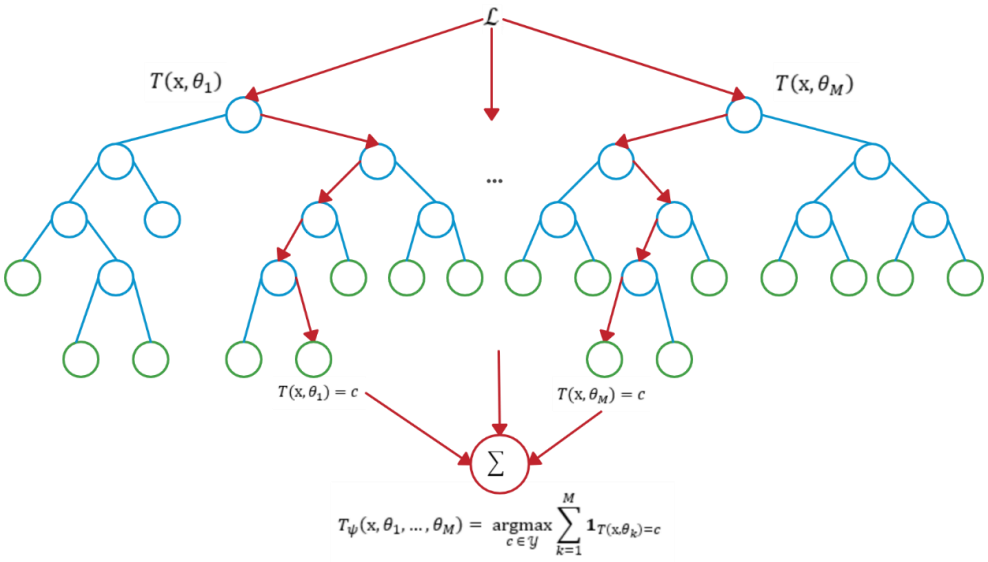
Вредности које одговарају листовима код регресије се израчунавају као просек вредности које су придружене датом листу, а у случају класификације је то већинска класа међу инстанцама. Могуће је добити и грубе процене поузданости предвиђања у виду стандардне девијације када је реч о регресији и вероватноће дате класе код класификације.

Стабла одлучивања се не сматрају моделима које карактерише висока прецизност. Дубока стабла су склона преприлагођавању, јер применом великог броја тестова могу да опишу и специфичности података на којима су обучаване које нису битне. Насупрот њима, плитка стабла карактерише проблем потприлагођавања. Иако се уочава да дубина стабла представља регулациони параметар и упркос постојању могућности регуларизације, стабла у пракси испољавају високу варијансу. Похлепна природа алгоритама њиховог обучавања највише утиче на такво понашање.

Као добра страна стабала највише се истиче њихова интерпретабилност. Наиме, ако је стабло мало, јасно се може закључити, на основу тестова које врши, како стабло доноси одлуке. Поред тога, стабла природно комбинују категоричке и непрекидне атрибуте и неосетљива су на монотоне трансформације атрибута. Под монотоном трансформацијом се подразумева свака трансформација једног скупа бројева у други таква да задржава редослед бројева из полазног скупа. Пример такве трансформације је $f(x) = 2x$.

3.2. Дефиниција насумичне шуме

Са основном идејом на којој се заснива насумична шума већ смо се упознали – ако једно стабло одлучивања даје добра предвиђања, више различитих стабала би требало да дају боља. Најинтересантније су технике захваљујући којима шума јесте *насумична*. Прва метода која уводи случајност у почетни скуп података је паковање, описано у делу *2*, а друга подразумева да се ограниче одлуке које стабло може да доноси. То се постиже тиме што се атрибути који ће се наћи у чворовима одлуке не бирају из скупа свих атрибута, већ његових случајно добијених подскупова. Детаљније објашњење даје поглавље *3.3*.



Илустрација 7: Насумична шума класификатор - поједностављен приказ

Илустрација *7* показује на поједностављен начин како се паковањем стабала одлучивања добија насумична шума и на који начин се врши класификација. Стабла која су означена са $T(\mathrm{x}, \theta_k), k = 1, \ldots, M$ разликују се јер је свако обучено на различитом бутсреп узорку скупа података $\mathcal{L}$. Резултат шуме се добија већинским гласањем, на исти начин као што је објашњено на илустрацији *4*. Класификаторима $C_1, C_2, \ldots, C_M$ одговарају стабла одлучивања $T(\mathrm{x}, \theta_k), k = 1, \ldots, M$, а ансамблу C^*одговара насумична шума $T_\psi(\mathrm{x}, \theta_1, \theta_2, \ldots, \theta_M)$.

Дефиниција 3.2.1. Насумична шума је класификатор који се састоји из колекције стабала одлучивања $\{T(\mathrm{x}, \theta_k), k = 1, \ldots, M\}$ где су $\{\theta_k\}$ независни идентично дистрибуирани насумични вектори, а свако стабло даје један глас за најпопуларнију класу којој припада инстанца x.[4]

Дакле, за k-то стабло се генерише насумични вектор θ_k, независан од претходних случајних вектора $\theta_1, \ldots, \theta_{k-1}$, али са истом расподелом, те се стабло конструише користећи скуп података за тренинг и θ_k да би се на крају добио класификатор $T(\mathrm{x}, \theta_k)$, где је x инстанца. Брејман је увео насумични вектор θ_k за све насумичне изборе који се праве при обучавању k-тог стабла, а функција стабла зависи не само од података, већ и од свих случајних избора. Код паковања коришћен је насумични параметар θ_m да би се добили случајни нови подскупови података за обучавање модела. При изградњи насумичне шуме поред насумичних тренинг подскупова, θ_k контролише и избор атрибута који ће се наћи у чворовима одлуке – разматрају се само атрибути из подскупова насумично изабраних атрибута. На тај начин се добијају још разноврснија стабла, што на крају резултира и тиме да су слабо корелисана.

[4] *Дефиницију 3.2.1* дао је Лео Брејман у свом раду у ком уводи појам насумична шума, доступном на: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf, при томе истичући њене класификационе способности, али наводећи даље у свом раду да се све идеје на којима почива насумична шума могу применити и на регресију.

3.3. Избор атрибута који ће се наћи у чворовима одлуке

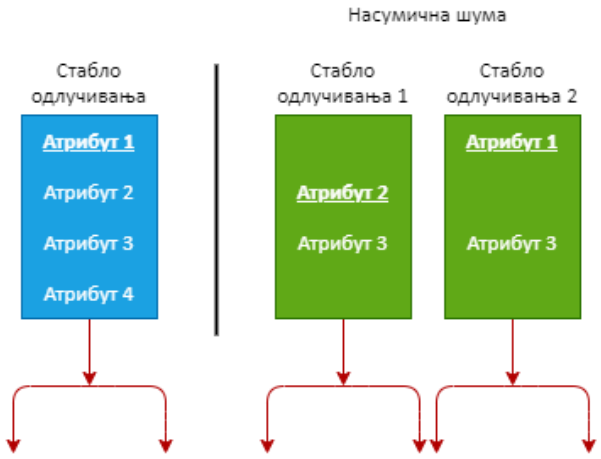
У делу *2.1* објашњен је однос корелације између подскупова података и варијансе. Насумично генерисање повећава независност, али, како је и поменуто тада, постоји још један параметар који смањује корелацију, а самим тим и варијансу – величина бутстреп узорка. Приликом изградње стабала одлучивања насумичне шуме користе се скупови настали насумичним узорковањем са понављањем иницијалног скупа података за обучавање, али ако је N број инстанци у почетном скупу, и скупови за обучавање стабала ће бројити исто N примерака. Запажање да се смањењем величине бутстреп узорка повећава некорелисаност података, па самим тим и стабала нашло је примену при одабиру кандидата из скупа свих атрибута који се могу наћи у чворовима одлуке. Ако атрибута има p, бира се неко $m \ll p$, тако да се за сваки чвор насумично бира m атрибута од p и у чвору ће се наћи најзначајнији атрибут од тих m. Све време током изградње шуме m има константну вредност.

У оригиналном раду Брејман је показао да на грешку предвиђања утичу две ствари:

1. *Веза* између било која два стабла у шуми; повећавањем повезаности расте и грешка предвиђања.
2. *Јачина* сваког индивидуалног стабла; стабло с малом грешком предвиђања је јак класификатор. Повећавањем јачине стабла опада грешка предвиђања.

Смањењем вредности параметра m смањује се и веза и јачина. Повећавањем расту оба. Потребно је пронаћи оптималну вредност параметра m. Препорука је да се за класификацију употребљава $m = \lfloor \sqrt{p} \rfloor$, а за регресију $m = \left\lfloor \frac{p}{3} \right\rfloor$.

У пракси најбоље вредности параметра m зависиће од конкретног проблема и требало би га сматрати *параметром за подешавање алгоритма*[5].



Илустрација 8: Избор атрибута који ће се наћи у чворовима одлуке стабала насумичне шуме

[5] Параметри модела су они фактори које је сам алгоритам учења у стању да оптимизује током процеса обучавања. Насупрот њима, понашање алгоритма може да зависи и од одређених параметара које алгоритам није у стању да оптимизује. Ти тзв. параметри за подешавање алгоритма се називају *хиперпараметрима* модела и њихове вредности је потребно подесити ручно пре почетка обучавања модела.

Размотримо како се бирају атрибути који ће бити у чворовима одлуке на визуелном примеру – на илустрацији *8* стабло одлучивања које није део насумичне шуме бира који ће се атрибут од четири могућа наћи у плавом чвору. Нека је *Атрибут 1* неком од метода одабира најзначајнијег атрибута изабран да се нађе у чвору одлуке као најзначајнији. С друге стране је насумична шума која се формира над истим скупом података као и стабло. За разлику од самосталног стабла, које на располагању има све атрибуте међу којима бира најзначајнији, стабла у шуми разматрају само подскупове насумично изабраних атрибута. *Стабло 1* бира који је значајнији атрибут између *Атрибута 2* и *Атрибута 3* (случајно одабраних). Иако *Атрибут 1* најбоље разврстава инстанце на хомогене групе, *Стабло 1* га не види, тако да је приморано да *Атрибут 2* прогласи за најзначајнији. *Стаблу 2* су доступни *Атрибут 1* и *Атрибут 3* и оно може да изабере *Атрибут 1*.

3.4. *Out of Bag* скуп

Подсетимо се да се приликом креирања нових подскупова скупа података за обучавање бутстрепинг техником поједине инстанце не појављују у неким новогенерисаним скуповима (Илустрација *2*). У поглављу *2.1* израчунато је да се нешто више од једне трећине података из полазног скупа за обучавање не појављује у бутстреп узорку. Прецизније, у идеалном случају, 36,8% почетног скупа података се не изабере насумичним одабирањем инстанци, јер за довољно велико N вероватноћа да се било која инстанца неће појавити у бутстреп узорку је $\lim_{N\to\infty}\left(1-\frac{1}{N}\right)^N = e^{-1} = 0{,}368$. Те инстанце чине *out of bag* (*OOB*) скуп.

Значај скупа неизабраних примерака се огледа у томе што се он користи за тестирање модела насумичне шуме. Иначе би се од оригиналног скупа података за обучавање морао одвојити посебан део података за тестирање пре самог обучавања. Неретко се дешава да није доступно довољно података и немогуће је приуштити њихову поделу. У таквим ситуацијама коришћење *OOB* података је одличан компромис. Такође, *OOB* скуп може да се користи и за валидацију, тако да поред тога што нема потребе за посебним скупом за тестирање, нема потребе ни за унакрсном валидацијом.

Outlook	Temperature	Humidity	Wind	Play Tennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Hot	High	Weak	Yes
Windy	Cold	Low	Weak	Yes

Табела 3: Скуп података play tennis

На *OOB* скупу се тачност предвиђања насумичне шуме процењује рачунањем вредности *OOB скор* (енгл. *OOB score*). Процену грешке предвиђања насумичне шуме даје метода *OOB грешка* (енгл. *out of bag error*). Искористићемо једноставан скуп података за обучавање који је дат у табели *3* ради илустрације поступка рачунања *OOB*

скора и *OOB грешке предвиђања*. Претпоставимо да насумичну шуму чини пет стабала одлучивања T_k $(k = 1, \dots, 5)$. Нека први бутстреп узорак чине прве три врсте скупа података *play tennis*. Тај подскуп ће се искористити за тренирање стабла T_1. Последња врста, изостављена из подскупа за тренирање, чини *OOB* скуп (Табела *4)*. У пракси ће бити изостављено више од једне врсте, али се овде због једноставности примера користи само једна.

Outlook	Temperature	Humidity	Wind	Play Tennis
Windy	Cold	Low	Weak	Yes

Табела 4: OOB скуп стабла T_1

Након што се стабла одлучивања истренирају, стабло T_1 предвиђа излаз за изостављену – до тада невиђену врсту. Нека предвиди тачно – *Yes*. Слично, сва стабла која су обучена на скупу који није садржао ову врсту, даће своје предвиђање. Претпоставимо да поред T_1, ни бустреп скупови за тренирање стабала T_3 и T_5 нису садржали последњу врсту табели *3*. Предвиђања стабала T_1, T_3 и T_5 за изостављену врсту дата су у табели *5.* Запажа се да је већинским гласањем насумична шума дала тачну предикцију за ову врсту.

Стабло одлучивања (*k*)	Предикција
1	YES
3	NO
5	YES
Већински глас: YES	

Табела 5: Предвиђања стабала T_1, T_3 и T_5 за изостављену врсту из скупова на којима су обучавана

Слично, за сваки *OOB* пример предикција ће се добити већинским гласањем у ком учествују стабла чији скуп за обучавање није садржао тај *OOB* пример. На крају, преброје се *OOB* примери за које су добијене тачне предикције. Та вредност представља *OOB скор.*

Обележимо сада са $\mathcal{L}_k$ $(k = 1, \dots, 5)$ бустреп узорке који су коришћени за тренирање стабала одлучивања T_k $(k = 1, \dots, 5)$, респективно. Пример из *OOB* скупа означимо са $(\mathrm{x}_1, \mathrm{y}_1)$, тако да у представља колону *Play Tennis* табеле *4* и нека је $\hat{\mathrm{y}}$ последња врста табеле *5.* У добијању предикције $\hat{\mathrm{y}}$ учествовала су стабла одлучивања која су истренирана на скупу који није садржао пример $(\mathrm{x}_1, \mathrm{y}_1)$, што математички записујемо $(\mathrm{x}_1, \mathrm{y}_1) \notin \mathcal{L}_k$. Нека је N укупан број *OOB* примера. Приликом рачунања *OOB скора* сабирају се тачне предикције, тј. проверава се за колико инстанци $(\mathrm{x}_i, \mathrm{y}_i)$ важи $\mathrm{y}_i = \widehat{\mathrm{y}_i}$ $(i = 1, \dots, N)$. Насупрот *OOB скору*, код рачунања *OOB грешке предвиђања* битне су нетачне предикције, и то проценат погрешно класификованих *OOB* примера.

Дакле, да би се израчунала грешка предвиђања k-то стабло класификује трећину података која није учествовала у изградњи k-тог стабла, тј. у предикцији y_i учествују само класификатори $T(\mathrm{x}, \theta_k)$ такви да важи $(\mathrm{x}_i, \mathrm{y}_i) \notin \mathcal{L}_k$. Нека је $T_\psi(\mathrm{x}_i, \theta_k) = \hat{\mathrm{y}}_i$. Тада се процењује да је *OOB грешка предвиђања* за класификацију:

$$\varepsilon_{oob} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{(y_i \neq \hat{y}_i)}.$$

Сабирак у ознаци $\mathbf{1}_{(y_i \neq \hat{y}_i)}$ утиче на број погрешно класификованих *OOB* инстанци ако је испуњено $y_i \neq \hat{y}_i$ (нетачна класификација) и тада износи 1, иначе 0.

Очигледно је да ће различита шума бити коришћена за предвиђање сваког y_i.

3.5. Важност атрибута

OOB подаци се могу користити и за одређивање важности атрибута, а овај поступак ће бити приказан коришћењем скупа података *titanic* познатог из поглавља *1.1* (Илустрација *1*). Важност атрибута се рачуна након што је завршено обучавање модела. Самим тим, неће се мењати нити модел ни предвиђања која се добијају за задату економску класу путника, године старости итд. Уместо тога се поставља питање – ако насумично пермутујемо вредности једног атрибута у *OOB* скупу, не мењајући остале атрибуте ни вредности циљне променљиве, шта се дешава са прецизношћу модела? Насумично прераспоређивање вредности једне колоне би требало да доведе до непрецизнијег предвиђања, јер подаци добијени пермутовањем нису реални. Предвиђање ће се нарочито погоршати ако пермутујемо вредности атрибута на који се модел највише ослања. Код проблема предвиђања који путник је преживео потонуће Титаника ако прераспоредимо вредности колоне *Pclass* (Илустрација *9*), добићемо значајно погоршање. Пермутовање колоне *Embarked* не би утицало ни приближно толико на предвиђање.

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked	Survived
1	3	male	22	1	0	7.25	S	0
2	1	female	58	0	0	26.55	S	1
3	2	male	35	0	0	26	S	0
...	...	...	...	...	...	...	...	...
100	3	male	7	4	1	7.8	S	0
101	2	female	34	0	1	23	S	1
...	...	...	...	...	...	...	...	...
891	1	male	26	0	0	30	C	1

Илустрација 9: Пермутација атрибута Pclass скупа података titanic

Имајући наведена запажања у виду, поступак одређивања најзначајнијег атрибута би подразумевао следеће:

1. Израчунати грешку предвиђања насумичне шуме помоћу *OOB* података на начин описан у поглављу *3.4*.
2. Пермутовати вредности у само једној колони у свим *OOB* примерима и искористити резултујући скуп података за рачунање грешке предвиђања. Затим је упоредити са грешком добијеном у 1. кораку. Разлика те две вредности показује колико се предвиђање погоршало и омогућава нам да одредимо меру важности атрибута.

3. Поништити пермутовање из 2. корака. У оригиналним *OOB* подацима пермутовати наредну колону (поновити корак 2 на другом атрибуту) све док се не израчунају мере важности за све атрибуте.

Дати поступак можемо и математички описати користећи ознаке уведене у делу *1.1*. Подсетимо се да смо са X_j (j = 1, ..., 7) означили атрибуте скупа *titanic*. Нека је $\varepsilon_{oob}^{(1)}$ грешка предвиђања добијена помоћу *OOB* примера у којима су пермутоване вредности атрибута *Pclass* (X_1). Тада се важност прве колоне скупа података *titanic* (Илустрација *9*) добија упросечавањем вредности $\varepsilon_{oob}^{(1)} - \varepsilon_{oob}$ над свим стаблима шуме. Отуда и назив методе – *Средње умањење прецизности* (енгл. *Mean Decrease in Accuracy – MDA*).

Нека је у општем случају број атрибута једнак p. Тада се за свако $j \in \{1, \dots, p\}$ у *out of bag* скупу насумично пермутују вредности j-тог атрибута и врши се рачунање грешке предвиђања у ознаци $\varepsilon_{oob}^{(j)}$. Средња вредност разлике $\varepsilon_{oob}^{(j)} - \varepsilon_{oob}$ на *out of bag* скупу представља смањење прецизности предвиђања и користи се као мера важности j-тог атрибута у насумичној шуми. Што је веће смањење прецизности, то је атрибут чије је пермутовање вредности довело до тог смањења значајнији.

Други начин за одређивање мере важности атрибута користи Џини индексе (поглавље *3.1*) и назива се *Средње умањење хетерогености* (енгл. *Mean Decrease in Impurity – MDI*). Мери се колико који атрибут доприноси хомогености група примера које се добију разврставањем на основу његове вредности. Атрибут је значајнији што су добијене групе хомогеније. Сваки пут када се одређени атрибут искористи у неком чвору одлуке, израчунају се вредности Џини индекса потомака и упореде се са Џини вредношћу оригиналног чвора: $Gini_{parent} - (Gini_{left} + Gini_{right})$. За сваки атрибут се промене у вредности Џини индекса саберу и израчуна се њихов просек над бројем стабала шуме. Атрибуте који разврставају инстанце у групе које се одликују већом хомогеношћу карактерише веће смањење Џини идекса низ стабло.

При избору између ове две методе треба узети у обзир то да је механизам који захтева пермутовање вредности атрибута рачунски захтевнији, али да даје поузданије резултате.

3.6. Алгоритам

Нека је скуп података за обучавање означен са $L = (\mathrm{x}_1, y_1), \dots, (\mathrm{x}_N, y_N)$, скуп атрибута са F и број стабала са M. Псеудокод *Алгоритам 1* илуструје рад алгоритма насумичне шуме.

Алгоритам ради на следећи начин: за свако стабло у шуми одабира се бутстреп узорак $L^{(i)}$ из скупа података L. Из скупа атрибута F за свако стабло се случајно бира подскуп атрибута $f \subseteq F$. У корену сваког стабла наћи ће се најзначајнији атрибут из f. Кардиналност скупа f је много мања од кардиналности скупа F. Следи обучавање стабла, на одговарајућем узорку, које подразумева насумично генерисање скупова $f \subseteq F$ за сваки његов чвор и одлучивање који ће атрибут из f бити искоришћен у чвору. Избор најзначајнијег атрибута је често најзахтевнија операција током обучавања стабла,

те сужавање скупа атрибута, поред описаних предности у делу *3.3*, и драстично убрзава обучавање стабла.

Алгоритам 1: Насумична шума

```
function RandomForest(L, F)
  T_ψ ← ∅
  for i ∈ 1, …, M do
    L^(i) ← бутстреп узорак скупа L
    T_i ← RandomizedTreeLearn(L^(i), F)
    T_ψ ← T_ψ ∪ {T_i}
  end for
  return T_ψ
end function

function RandomizedTreeLearn(L, F)
  За сваки чвор:
    f ← јако мали подскуп скупа F
    Искористи у чвору најзначајнији атрибут из f
    и разврстај примере на основу његове вредности
 return обучено стабло
end function
```

3.7. Параметри за подешавање алгоритма

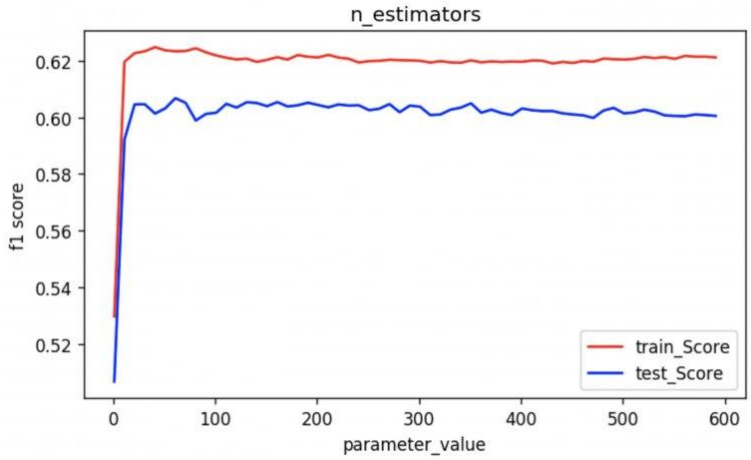
Постоји много *хиперпараметара насумичне шуме*[6]. Разликујемо две врсте – оне који утичу на брзину алгоритма и оне који утичу на прецизност модела. Неки од представника прве врсте су *n_jobs* (омогућава паралелно извршавање алгоритма на више процесора) и *oob_score* (омогућава употребу трећине доступних података за валидацију). У овом делу рада следи детаљније упознавање са хиперпараметрима насумичне шуме који утичу пре свега на прецизност:

1. *n_estimators*,
2. *max_depth*,
3. *min_samples_split*,
4. *max_leaf_nodes*,
5. *min_samples_leaf*,
6. *max_samples* (бутстреп узорак) и
7. *max_features*.

Прво питање које се поставља приликом конструкције насумичне шуме јесте колико стабала треба разматрати. Хиперпараметар који представља број стабала које

[6] Подешавање и оптимизација параметара насумичне шуме ће бити објашњена на параметрима класе *sklearn.ensemble.RandomForestClassifier*.

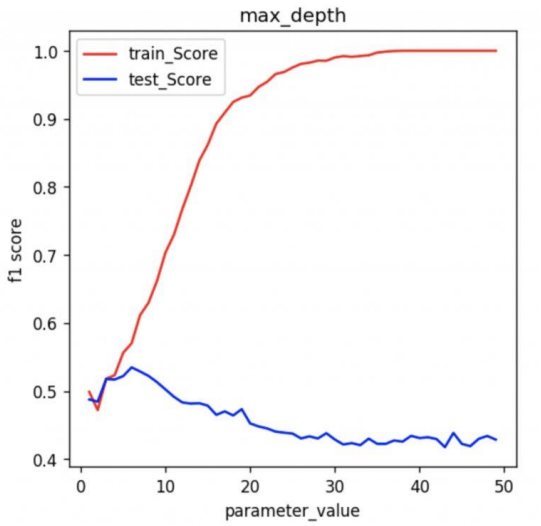
алгоритам обучи пре гласања или упросечавања предикција је *n_estimators*. С порастом броја стабала повећава се прецизност предвиђања ансамбла, али повећава се и комплексност модела. На илустрацији *10* приказано је побољшање перформанси модела с порастом броја стабала до неке тачке, након које веће вредности броја стабала престају да утичу на прецизност предвиђања. Може се уочити да није најбоља идеја користити веома велики број стабала у насумичној шуми. Иако неће доћи до деградације модела, значајно ће се успорити процес обучавања.



Илустрација 10: Прецизност предвиђања насумичне шуме у зависности од броја стабала[7]

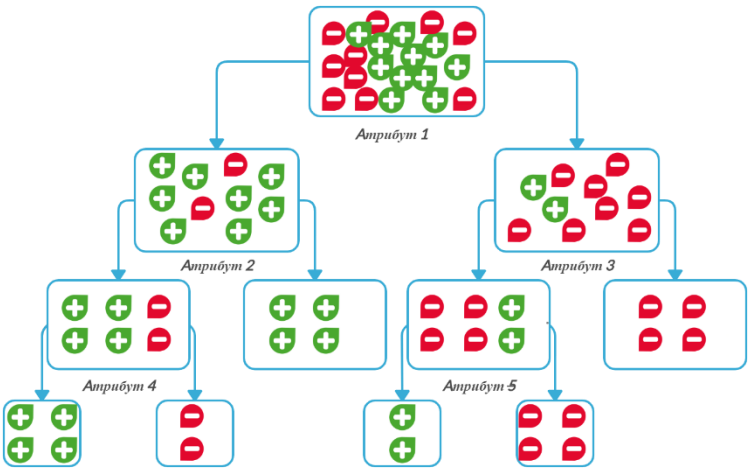
Следећи параметар *max_depth* дефинише се као најдужи пут од корена стабла до листа. Коришћењем тог параметра ограничава се до које дубине ће се стабло формирати. На илустрацији *11* се јасно уочава да како се повећава дубина стабла, побољшавају се и перформансе модела на подацима за обучавање. Насупрот томе, на подацима за тестирање примећује се пораст прецизности предвиђања до одређене тачке, након које перформансе нагло опадају. До тога долази због преприлагођавања стабла.

[7] График *Илустрација 10* као и сви графици зависности перформанси алгоритма насумичне шуме од хиперпараметара дати у поглављу *Параметри за подешавање алгоритма*, преузети су са https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/.

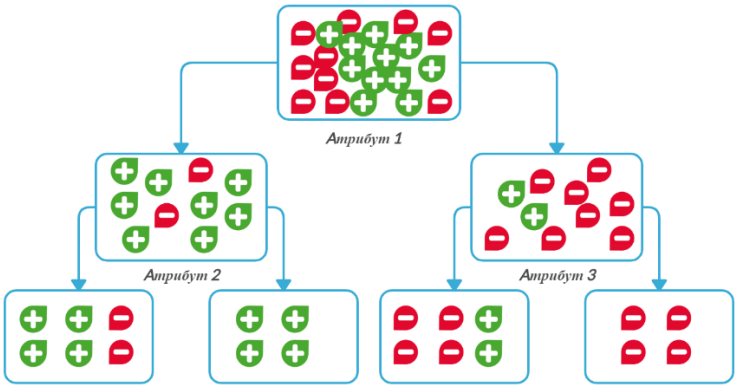


Илустрација 11: Зависност прецизности предвиђања од дубине стабла

Трећи параметар који се разматра такође може изазвати преприлагођавање. Ако је број инстанци које треба да се разврстају кроз неки чвор на хомогеније групе мањи од вредности хиперпараметра *min_samples_split*, тај чвор постаје лист и стаје се са разврставањем. Подразумевана вредност је 2. То значи да ако се након поделе инстанци у групе, добије група чија ентропија није нула, а садржи више од два члана, треба наставити са разврставањем примера (Илустрација *12*). Ако бисмо стаблу са илустрације *12* повећали вредност параметра *min_samples_split* са 2 на 6, онда би стабло изгледало као на илустрацији *13*. Очигледно је да разврставање примера за учење све док се не добију потпуно хомогене групе доводи до повећања дубине стабла, па самим тим до његовог преприлагођавања. Повећањем вредности параметра *min_samples_split* смањује се број гранања у стаблу одлучивања и спречава се преприлагођавање.

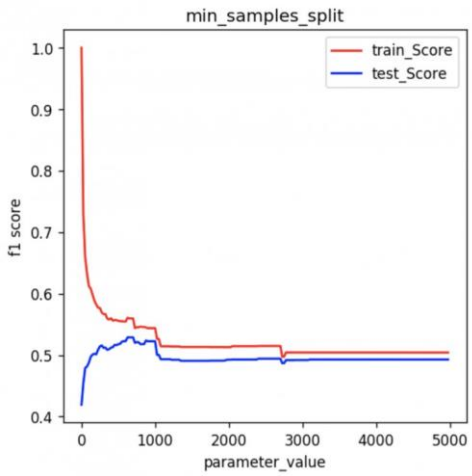


Илустрација 12: Стабло одлучивања с параметром min_samples_split = 2



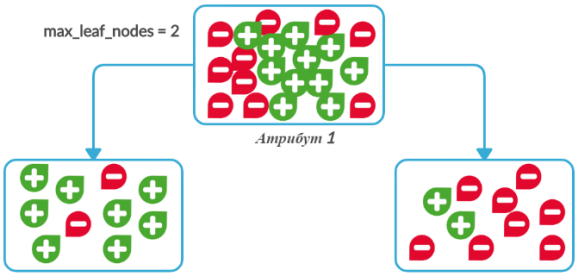
Илустрација 13: Стабло одлучивања с параметром min_samples_split = 6

На илустрацији *14* приказан је утицај параметра *min_samples_split* на перформансе модела подразумевајући да се остали параметри нису мењали. Запажа се да за мале вредности хиперпараметра постоји значајна разлика у резултатима које алгоритам постиже на скупу за обучавање у односу на резултате које постиже на подацима за тестирање, док за веће вредности *min_samples_split* разлика у перформансама опада. Треба имати у виду да ако се рано прекине са разврставањем инстанци, тј. ако је вредност *min_samples_split* велика, погоршаће се перформансе насумичне шуме и на скупу за обучавање и на скупу података за тестирање. Наиме, дођи ће до потприлагођавања, јер неће бити могуће уочити ниједну значајну поделу примера.



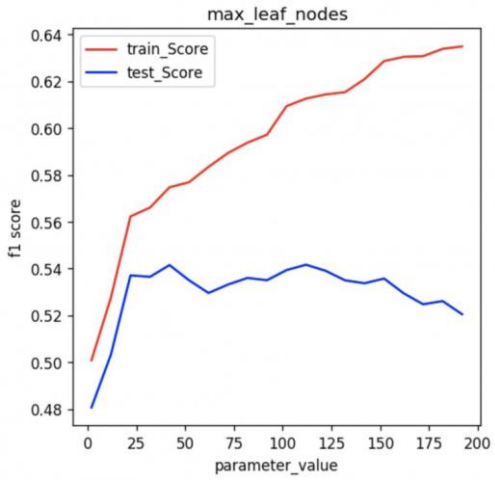
Илустрација 14: Зависност перформанси насумичне шуме од параметра min_samples_split

Хиперпараметар *max_leaf_nodes* спречава раст стабла тако што се прекида са разврставањем инстанци за учење кроз неки чвор када број листова стабла достигне максималну дозвољену вредност. Нека је максимални број листова стабла два (Илустрација *15*). Тада се већ након разврставања примера на основу вредности атрибута у корену стабла достигла највећа дозвољена вредност параметра *max_leaf_nodes*.



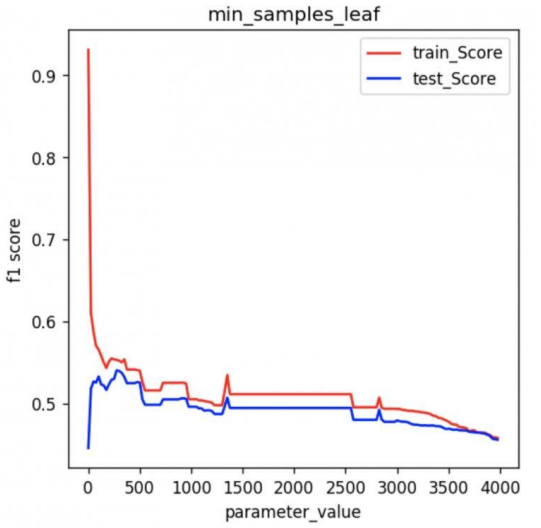
Илустрација 15: Стабло одлучивања с параметром max_leaf_nodes = 2

На тај начин је спречено преприлагођавање. С друге стране, ако је сувише мали максимални број листова, насумична шума је подложна потприлагођавању. Наведена запажања се дају уочити на графику зависности перформанси модела од хиперпараметра *max_leaf_nodes* (Илустрација *16*).



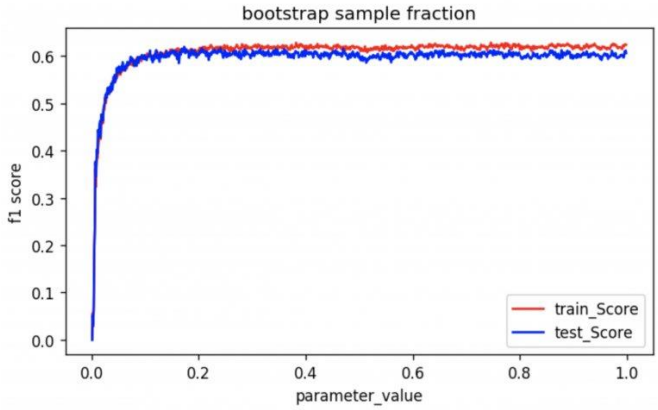
Илустрација 16: Зависност перформанси насумичне шуме од максималног броја листова

Раст стабла се контролише и параметром *min_samples_leaf* тако што се одређује најмањи могући број инстанци које се могу наћи у једном листу након поделе инстанци у листове на основу вредности атрибута који се тестира у родитељском чвору. Ако је број примерака мањи од дозвољеног, зауставља се подела и чвор у коме је дошло до нарушавања ограничења постаје лист. Очигледно, са повећањем вредности минималног броја инстанци у листу, параметар *min_samples_leaf* спречава појаву преприлагођавања. Са илустрације *17* се уочава да ако би се наставило с повећавањем минималног броја примерака у терминалном чвору, дошло би до потприлагођавања.



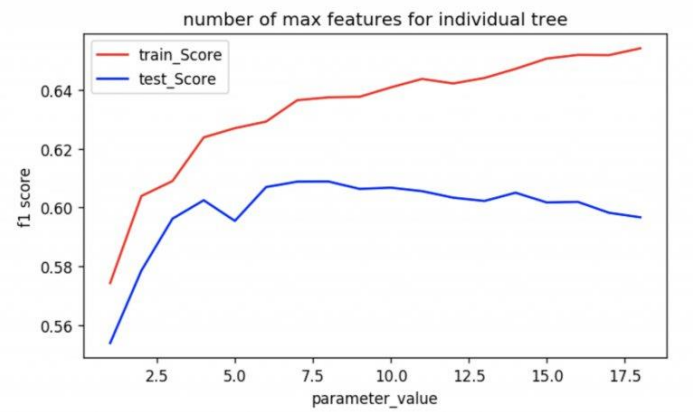
Илустрација 17: Зависност перформанси насумичне шуме од минималног броја инстанци у листу

Следеће што треба размотрити јесте колики део оригиналног скупа података се користи за обучавање сваког појединачног стабла. То одређује параметар *max_samples*. Наиме, испоставља се да није неопходно доделити насумичној шуми цео скуп података. За пример са графика *(*Илустрација *18*) довољно је мање од 20% оригиналног скупа, јер даљим повећавањем удела, перформансе се не побољшавају. Иако вредност параметра *max_samples* зависи од конкретног скупа података, примећујемо да се за обучавање сваког стабла може користити део скупа података уместо целог скупа и тиме значајно убрзати процес обучавања.



Илустрација 18: Зависност перформанси насумичне шуме од величине удела скупа података на ком се појединачно стабло обучава у оригиналном скупу

Параметар *max_features* је поменут у делу *3.3* где је означаван са m. Представља број насумично одабраних атрибута које разматра свако појединачно стабло. Са илустрације *19* се уочава да повећавање броја атрибута позитивно утиче на перформансе модела. Међутим, ако се повећавање настави, и након неке одређене вредности, перформансе ће остати исте или ће се погоршавати на скупу података за тестирање, док ће скуп за обучавање давати све боље резултате, што је очигледни показатељ да је модел почео да се преприлагођава. Подразумевана вредност *max_features* је квадратни корен укупног броја атрибута.



Илустрација 19: Зависност перформанси насумичне шуме од максималног броја случајно одабраних атрибута за свако појединачно стабло

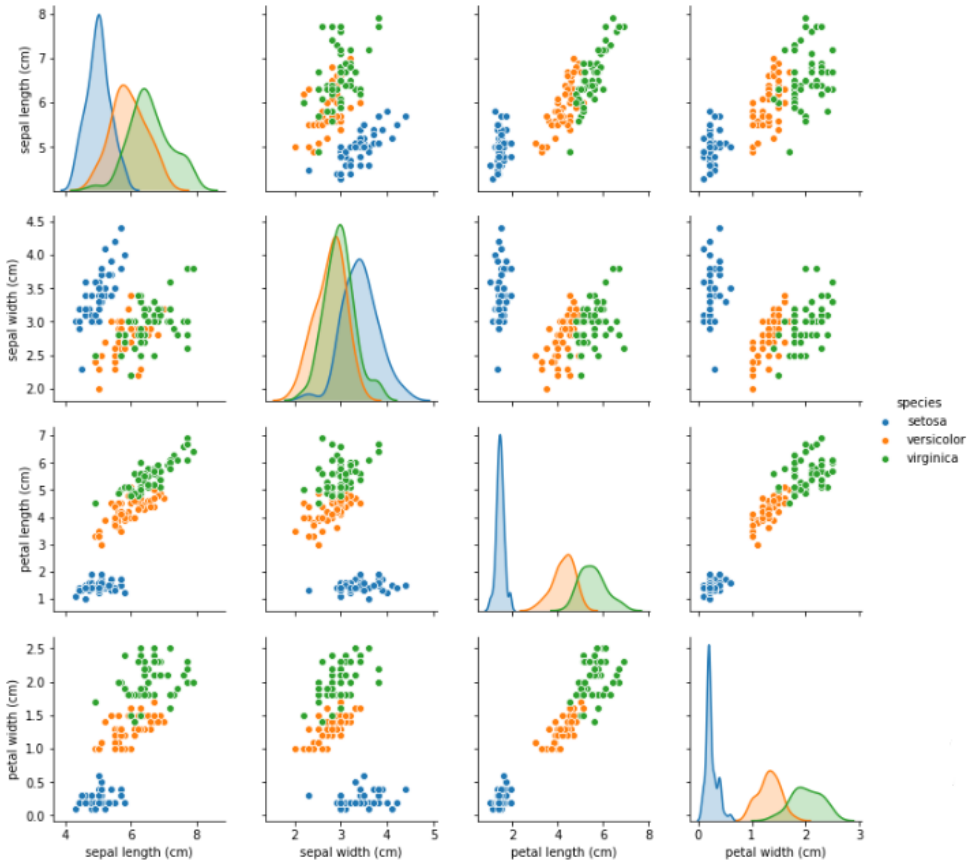
4. Креирање модела насумичне шуме за тренинг скуп *Iris*

Погодан скуп података за илустрацију проблема класификације када поједине класе нису јасно раздвојене од других и вероватно најкоришћенији скуп је *iris* (перуника).

Код 1: Учитавање скупа iris и приказ учитаних података помоћу матрице расипања

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import datasets
iris = datasets.load_iris()

df = pd.DataFrame(iris.data, columns=iris.feature_names)

# sklearn je svakoj sorti dodelio numericku vrednost; to je potrebno zbog
# klasifikacije
# generisanje matrice rasipanja
df['species'] = np.array([iris.target_names[i] for i in iris.target])

sns.pairplot(df, hue='species')
```

Илустрација *20* даје приказ матрице расипања (енгл. *scatter matrix*) учитаног скупа података. Уочава се да се сорта *setosa* лако идентификује на основу дужине латице (енгл. *petal length*), и ширине латице (енгл. *petal width*) док је остале две сорте перуника теже разликовати. Због једноставности самог скупа, претпроцесирање је изостављено.



Илустрација 20:Матрица расипања скупа iris

Прво је извршена подела података на скуп за обучавање и скуп за тестирање и то у односу 75:25. Пошто *sklearn* захтева да сви атрибути и циљне променљиве буду нумеричке, три класе су представљене целобројним вредностима (0, 1, 2).

Код 2: Подела података на скупове за обучавање и тестирање

```
from sklearn.model_selection import train_test_split

# upotreba stratify obezbedjuje da oba skupa sadrze reprezentativne instance sve
# tri klase
X_train, X_test, y_train, y_test = train_test_split(df[iris.feature_names],
iris.target, test_size=0.25, stratify=iris.target, random_state=123456)
```

Добијени скуп података за обучавање искоришћен је за формирање насумичне шуме (поглавље *4.1*) и за формирање самосталног стабла одлучивања (поглавље *4.2*). За оба модела је израчуната тачност предвиђања на истим примерима за тестирање ради упоређивања перформанси. Циљ је показати да насумична шума заиста тачније предвиђа од свог базног модела.

4.1. Формирање предикција применом насумичне шуме

У овом примеру коришћене су подразумеване вредности параметара насумичне шуме. Изузетак је рачунање *OOB скора* – подразумевана вредност параметра *oob_score* је *false*, тако да је потребно да класификатору кажемо да желимо да се процени тачност предвиђања употребом *OOB* примера. То смо учинили постављањем параметра *oob_score* на *true*.

Код 3: Креирање објекта класе RandomForestClassifier и обучавање

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, oob_score=True,
random_state=123456)

rf.fit(X_train, y_train)
```

Потребно је погледати сада какве су перформансе креираног модела на непознатом скупу података за тестирање. Упоредићемо процену тачности израчунату коришћењем *OOB* скупа и стварне тачности предвиђања израчунате на подацима скупа за тестирање. Добијене вредности су приказане на илустрацији *21*. За *OOB* процену тачности предвиђања добили смо 94,6%. То је прецизност коју можемо да очекујемо на новим подацима. На тестном скупу постигнута прецизност је 92,1%. Дакле, резултати нису лоши, али очекивали смо да ће наш модел класификовати с мало већом прецизношћу.

Код 4: Рачунање тачности модела

```
from sklearn.metrics import accuracy_score

predicted = rf.predict(X_test)
accuracy = accuracy_score(y_test, predicted)
print(f'Out-of-bag score estimate: {rf.oob_score_:.3}')
print(f'Mean accuracy score: {accuracy:.3}')
```

```
Out-of-bag score estimate: 0.946
Mean accuracy score: 0.921
```

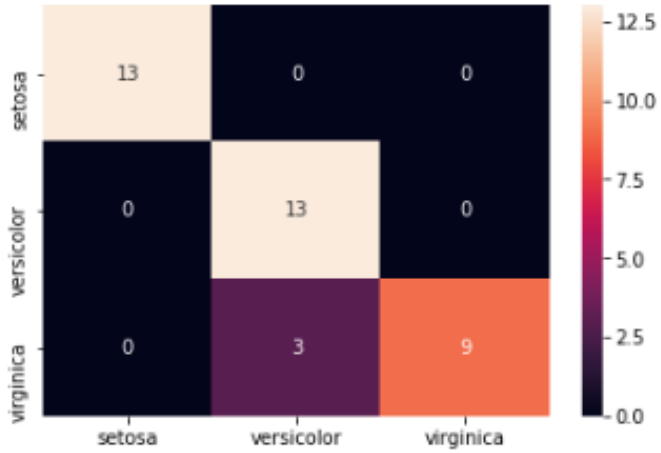
Илустрација 21: Резултат кода 4

Корисна техника која даје бољи увид у перформансе модела је матрица конфузије (енгл. *confusion matrix*). На главној дијагонали матрице налази се број инстанци за које се предвиђене класе поклапају с правим класама (енгл. *true positive*), док се ван дијагонале налазе инстанце које је модел погрешно сврстао у неку другу класу (енгл. *false positive* и *false negative*).

Код 5: Генерисање матрице конфузије

```
from sklearn.metrics import confusion_matrix

cm = pd.DataFrame(confusion_matrix(y_test, predicted),
columns=iris.target_names, index=iris.target_names)
```

```
sns.heatmap(cm, annot=True)
```



Илустрација 22: Матрица конфузије насумичне шуме над скупом iris

Тумачењем матрице конфузије (Илустрација *22*) изводи се закључак да модел насумичне шуме тачно класификује сорту *setosa*, али показује извесну меру конфузије приликом разликовања сорти *versicolor* и *virginica* – 3 примера класе *virginica* је погрешно сврстано у класу *versicolor*.

4.2. Формирање предикција применом самосталног стабла одлучивања

На исти начин као за насумичну шуму, израчуната је тачност предвиђања стабла одлучивања на истом тестном скупу и добијено је да износи 89,5% (Илустрација *23*). Јасно је да се постиже побољшање применом ансамбла стабала – 92,1% (Илустрација *21*).

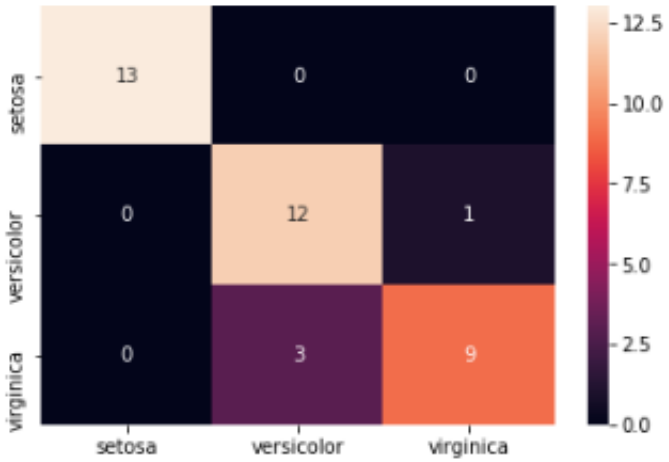
Код 6: Обучавање стабла одлучивања и рачунање тачности модела

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

clf = DecisionTreeClassifier()

clf.fit(X_train, y_train)

predicted = clf.predict(X_test)
accuracy = accuracy_score(y_test, predicted)
print(f'Mean accuracy score: {accuracy:.3}')
```

```
Mean accuracy score: 0.895
```

Илустрација 23: Резултат кода 6

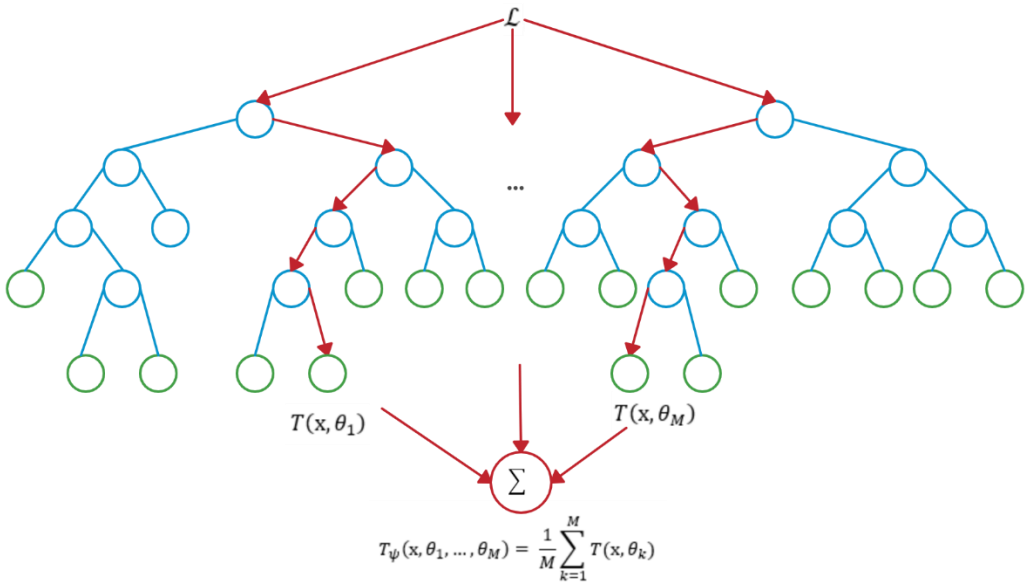
Постигнути резултати су приказани и у виду матрице конфузије са илустрације *24*. Стабло одлучивања је погрешно класификовало 4 примера из скупа за тестирање (1 пример класе *versicolor* је сврстан у класу *virginica* и 3 примера класе *virginica* је погрешно сврстано у класу *versicolor*). И на основу матрица конфузије насумичне шуме и самосталног стабла, изводи се исти закључак – насумична шума класификује боље. Циљ постављен пре обучавања оба модела је постигнут.



Илустрација 24: Матрица конфузије стабла одлучивања над скупом iris

5. Регресија алгоритмом насумичне шуме

Насумичну шуму за регресију формирају стабла изграђена у зависности од насумичног вектора θ тако да функција предвиђања стабла $T(\mathrm{x},\ \theta)$, за разлику од класификације, вредности узима из скупа реалних бројева. Излазна вредност је реалан број и претпостављамо да су скупови за обучавање стабала слабо корелисани. Уместо гласања, упросечавањем појединачних предикција добија се излазна вредност насумичне шуме (Илустрација *25*).



Илустрација 25: Насумична шума за регресију

У случају регресије атрибут који ће бити искоришћен у неком чвору одлуке се бира из подскупа насумично одабраних атрибута скупа свих атрибута, идентично као код класификације. Принципи на којима почива значај *OOB* скупа података и мерење важности атрибута такође су заједнички класификацији и регресији. Како је природа проблема из области регресије другачија од класификационих, те је циљна променљива континуална, рачунање *OOB* грешке предвиђања разликује се од дефиниције дате у поглављу *3.4* и изводи се на следећи начин:

$\varepsilon_{oob} = \frac{1}{N}\sum_{i=1}^{N}(\mathrm{y}_i - \widehat{\mathrm{y}}_i)^2$, при чему важе све претпоставке наведене у делу *3.4*.

У поглављу *3.7* су обрађени неки од хиперпараметара алгоритма насумичне шуме користећи класу *sklearn.ensemble.RandomForestClassifier*. За регресионе проблеме се користи друга класа библиотеке *scikit-learn – sklearn.ensemble.RandomForestRegressor.* Обема класама су заједнички параметри из дела *3.7*, стога све наведено у том поглављу важи и за подешавање параметара приликом формирања регресионе насумичне шуме.

5.1. Пример регресионог модела – предвиђање плата запослених на основу позиције

За демонстрацију рада алгоритма насумичне шуме за решавање регресионих проблема коришћен је скуп података са илустрације *26*. Садржи три колоне – позицију на којој запослени ради (*Position*), ниво квалификације запосленог (*Level*) и плату (*Salary*). Скуп података даје за сваког запосленог приближну плату на основу нивоа квалификације. На пример, ако запослени ради на позицији пословног аналитичара (*Business Analyst*) и сматра се да је ниво квалификације пословног аналитичара 1, онда би требало да запослени прима плату око 45000 долара. Модел који креирамо треба да предвиђа коју плату би требало понудити новом запосленом на основу његовог нивоа квалификације.

perpecvja.ipynb ● | pozicija_plata.csv ×

Delimiter: ,

	Position	Level	Salary
1	Business Analyst	1	45000
2	Junior Consultant	2	50000
3	Senior Consultant	3	60000
4	Manager	4	80000
5	Country Manager	5	110000
6	Region Manager	6	150000
7	Partner	7	200000
8	Senior Partner	8	300000
9	C-level	9	500000
10	CEO	10	1000000

Илустрација 26: Скуп података за обучавање[8]

На основу примера у претходном параграфу закључује се да се нивоом квалификације описује позиција на којој запослени ради (нпр. Правилником о запошљавању је прописано који је ниво квалификације потребан за коју позицију.), тако да се предвиђање плата практично врши само на основу нивоа квалификације. За предвиђање се не користи колона *Position*. Доделом `X = df.iloc[:,1].values` издвојили смо све врсте и прву колону *Level* на основу које вршимо предвиђање колоне *Salary* – `y`. Циљној променљивој додељена је последња колона и све врсте са `y = df.iloc[:, 2].values`. Резултујуће вредности `X` и `y` приказане су на илустрацији *27*.

Код 7: Насумична шума за регресију у Python-у – учитавање скупа података

```
# importovanje potrebnih biblioteka
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# ucitavanje skupa podataka
```

[8] Подаци су доступни на https://www.kaggle.com/akram24/position-salaries.

```
df = pd.read_csv('pozicija_plata.csv')

# selektovanje svih vrsta i prve kolone za X, a sve vrste i drugu kolonu
# dodeliti y
X = df.iloc[:, 1].values
y = df.iloc[:, 2].values
```

```
[2]: X

[2]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10], dtype=int64)

[3]: y

[3]: array([  45000,   50000,   60000,   80000,  110000,  150000,  200000,
         300000,  500000, 1000000], dtype=int64)
```

Илустрација 27: Приказ скупа података за обучавање генерисан у Python-у

Пошто је скуп података јако мали, цео скуп се користи за обучавање. Коришћена је класа sklearn.ensemble.RandomForestRegressor. Прво је креиран објекат и наведено је да 10 стабала чини шуму. Због једноставности скупа података није коришћено више. Затим следи обучавање модела прослеђујући X и y у нешто измењеном облику (Илустрација 28).

Код 8: Насумична шума за регресију у Python-у – обучавање модела

```
# obuciti model
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
# -1 – nepoznat broj vrsta, 1 – broj kolona
regressor.fit(X.reshape(-1,1), y.reshape(-1,1))
```

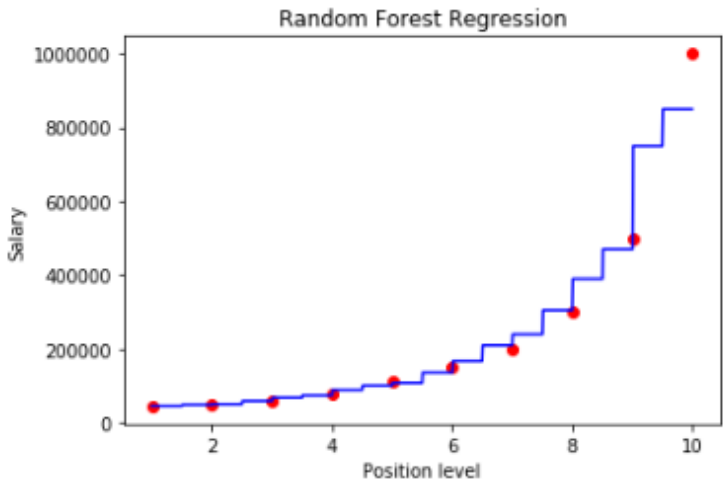
```
[7]: X.reshape(-1,1)

[7]: array([[ 1],
            [ 2],
            [ 3],
            [ 4],
            [ 5],
            [ 6],
            [ 7],
            [ 8],
            [ 9],
            [10]], dtype=int64)

[8]: y.reshape(-1,1)

[8]: array([[  45000],
            [  50000],
            [  60000],
            [  80000],
            [ 110000],
            [ 150000],
            [ 200000],
            [ 300000],
            [ 500000],
            [1000000]], dtype=int64)
```

Илустрација 28: Параметри који се прослеђују методи за обучавање - fit

Резултате за регресиону насумичну шуму можемо погледати на графику зависности плата од позиције (Илустрација *29*). Предвиђања насумичне шуме су представљена континуалним тачкама (плава боја на графику). Стварне вредности представљене су црвеним тачкама. Уочава се највеће одступање за нивое квалификације близу вредности 10.

Код 9: Насумична шума за регресију у Python-у – приказ резултата

```
# prikaz rezultata
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))

# stvarni podaci
plt.scatter(X, y, color = 'red')

# predvidjeni podaci
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Random Forest Regression')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```



Илустрација 29: Резултат кода 9 – упоредни приказ предвиђених и правих плата у зависности од позиције

На крају, претпоставимо да је кандидат за посао конкурисао за позицију регионалног менаџера и да има двогодишње искуство на тој позицији. На основу табеле са илустрације *26* ниво квалификације је између 6 и 7, нека буде 6,5. Наш модел је предвидео да том кандидату треба предложити плату од 167000 долара (Илустрација *30*).

```
y_pred = regressor.predict([[6.5]])
print('Plata zaposlenog ciji je nivo kvalifikacije 6,5 treba da bude oko ',y_pred)
```

```
Plata zaposlenog ciji je nivo kvalifikacije 6,5 treba da bude oko  [167000.]
```

Илустрација 30: Предвиђање плате за нову вредност нивоа квалификације