Klasifikator pojačanja gradijenta kod procjene mogućnosti preživljavanja putnika na Titaniku

Lovrić, Irena

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Irena Lovrić

Zagreb, 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentor: Student:

Doc. dr. sc. Tomislav Stipančić, dipl. ing. Irena Lovrić

Zagreb, 2022.

Izjavljujem da sam ovaj rad izradila samostalno koristeći znanja stečena tijekom studija i navedenu literaturu.

Zahvaljujem se mentoru doc. dr. sc. Tomislavu Stipančiću na pristupačnosti i pruženoj pomoći tijekom izrade rada.

Irena Lovrić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i diplomske ispite studija strojarstva za smjerove.
proizvodno inženjerstvo, računalno inženjerstvo, industrijsko inženjerstvo i menadžment, inženjerstvo materijala i mehatronika i robotika

Sveučilište u Zagrebu Fakultet strojarstva i brodogradnje	
Datum	Prilog
Klasa: **602 – 04 / 22 – 6 / 1**	
Ur.broj: **15 - 1703 - 22 -**	

ZAVRŠNI ZADATAK

Student: **Irena Lovrić** JMBAG: **0035213799**

Naslov rada na hrvatskom jeziku: **Klasifikator pojačanja gradijenta kod procjene mogućnosti preživljavanja putnika na Titaniku**

Naslov rada na engleskom jeziku: **Gradient boost classifier in the assessment of the survival chance of passengers on Titanic**

Opis zadatka:

U svijetu podataka i velike količine informacija algoritmi umjetnu inteligenciju postaju sve značajniji. U ovisnosti o vrsti i strukturi podataka metode strojnog učenja koriste se prilikom predviđanja, odlučivanja, klasifikacije, prepoznavanja, itd.
Koristeći metodu pojačanja gradijenta strojnog učenja (eng. Gradient Boost Classifier) u radu je potrebno napraviti računalni model za predviđanje

U radu je potrebno:

- koristiti odgovarajući skup podataka tako da se on podjeli na podatke za trening i podatke za testiranje,
- koristeći tehnike rudarenja podataka iz skupa značajki koje opisuju putnike izdvojiti i povezati one koje značajnije utječu na mogućnost preživljavanja,
- usporediti rezultate klasifikacije više znanih klasifikatora na korištenom skupu podataka koristeći prikladnu evaluacijsku metriku,
- definirati parametre (eng. hyperparameters) klasifikatora pojačanja gradijenta, izvršiti klasifikaciju te napraviti kritički osvrt na rezultate.

U radu je potrebno navesti korištenu literaturu i eventualno dobivenu pomoć.

Zadatak zadan:
30. 11. 2021.

Zadatak zadao:
Doc. dr. sc. Tomislav Stipančić

Datum predaje rada:
1. rok: 24. 2. 2022.
2. rok (izvanredni): 6. 7. 2022.
3. rok: 22. 9. 2022.

Predviđeni datumi obrane:
1. rok: 28. 2. – 4. 3. 2022.
2. rok (izvanredni): 8. 7. 2022.
3. rok: 26. 9. – 30. 9. 2022.

Predsjednik Povjerenstva:
Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS OZNAKA

Oznaka	Jedinica	Opis
x		Input-i
y		Output-i
F*(***x***)		Funkcija koja povezuje input-e i output-e
$\boldsymbol{\psi}(y,F(\boldsymbol{x}))$		Funkcija gubitka
$h(\boldsymbol{x},\boldsymbol{a})$		Jednostavne funkcije (base learner)
$\boldsymbol{a}$		Parametri jednostavane funkcije
β_m		Koeficijent
M,m		Broj iteracije
R_{lm}		Razdvojne regije
$\bar{y}_{lm}$		Artimetička sredina u regiji
ν		Learning rate
TP		True Positive
TN		True Negative
FP		False Postive
FN		False Negative
TPR		True Positive Rate
FPR		False Positive Rate

SAŽETAK

U ovom radu je ukratko objašnjen algoritam Gradient Boost i prikazana matematika na kojoj je baziran i koju je objasnio Jerome H. Freidman. Gradient Boost je pokazao velik uspjeh u širokom spektru primjene u stvarnom svijetu. Također su definirani pojmovi funkcija gubitka (loss function), learning rate i overfitting. Funkcija gubitka i learning rate kao jedne od najznačajnijih faktora u Gradient Boost, te overfitting kao jedan od najvećih problema Gradient Boosta.

Također je objašnjen primjer klasifikacije baze podatka koristeći Gradient Boost „Titanic- Top 1% with Gradient Boost Clasifier“. Ovim primjerom se pokušava odrediti koji putnik će preživjeti i koji neće preživjeti potonuće Titanika. Primjer je uzet sa stranice Kaggle i napisan je u Python-u. Kroz objašnjenje primjera smo prošli kroz danu bazu podataka, prilagodili varijable (skupine podatka) našim potrebama, odredili suodnose varijabli, kako je odabran Gradinet Boost i prilagodili hiperparametre.

Ključne riječi: Freidman, Gradient Boost, funkcija gubitka (loss function), learning rate, overfitting, Python, Kaggle , Titanic

SUMMARY

In this pape Gradient Boost Algoritham is briefly explained and math by Jerome H. Freidman on witch it is based. Gradient Boost has shown great success in wide rage of applications. The concepts of loss function, learning rate and overfitting are also defined. The function of loss and learning rate are one of the most important factors in Gradient Boost, and overfitting is one of the biggest problems of Gradient Boost.

There i salso an explanation of an database classification using Gradiet Boost „Titanic- Top 1% with Gradient Boost Clasifier“. Attempt of this exaple is to determined wich passenger will survive and wich will not survive the sinking of the Titanic.This example was taken from th Kaggle page and it was writen in Python. Going through exaplme , we went over given datebase, adjusted the variables to our needs, determined the correlations of the varibale, how was Gradient Boost selesceted and adjusted the hyperparamethrs

Key words: Freidman, Gradient Boost, loss function, learning rate, overfitting, Python, Kaggle, Titanic

1. UVOD

Titanik je britanski prekooceanski putnički brod, u vlasništvu White Star Line, koji je prvi put isplovio iz Southamptona 10. travnja 1912. te pristao u Cherbourgu (Francuska) i Queenstown (Irska) prije nego što je zaplovio za New York. Bio je jedan od najvećih brodova na svijetu.

Jedna od najvećih pomorskih nesreća je potonuće Titanika te je od 2224 putnika i članova posade, od mogućih 3327 osoba, poginulo čak 1502 osobe. Razlog potonuća Titanika je bio sudar sa santom leda u 23:40h dana 14. 4. 1912. Kapetan nije smanjio brzinu plovidbe unatoč upozorenjima o santama leda. Brzina plovidbe je bila 22 čvora, što je pri sudaru uzrokovalo ozbiljno oštećenje broda te je voda počela prodirati u 5 od 16 vodonepropusnih odjeljaka dok brod može izdržati samo 4 poplavljena odjeljka . Titanik je u potpunosti potonuo u 2:20h dana 15.4.1912.

Glavni razlozi za pogiblju tolikog broja ljudi na Titaniku je nedovoljan broj čamaca za spašavanje. Na čamcima za spašavanje je bilo mjesta za oko pola putnika i članova posade tj. trećina mogućeg kapaciteta.

Iako je sreća imala ulogu kod preživljavanja, možemo primijetiti da su neke grupe ljudi imale veću šansu preživljavanja od drugih.

Kroz ovaj rad ćemo objasniti kod koji se zasniva na klasifikatoru pojačanja gradijenta kako bi odredili koji faktori najviše utječu na mogućnost preživljavanja.

2. Gradient boosting

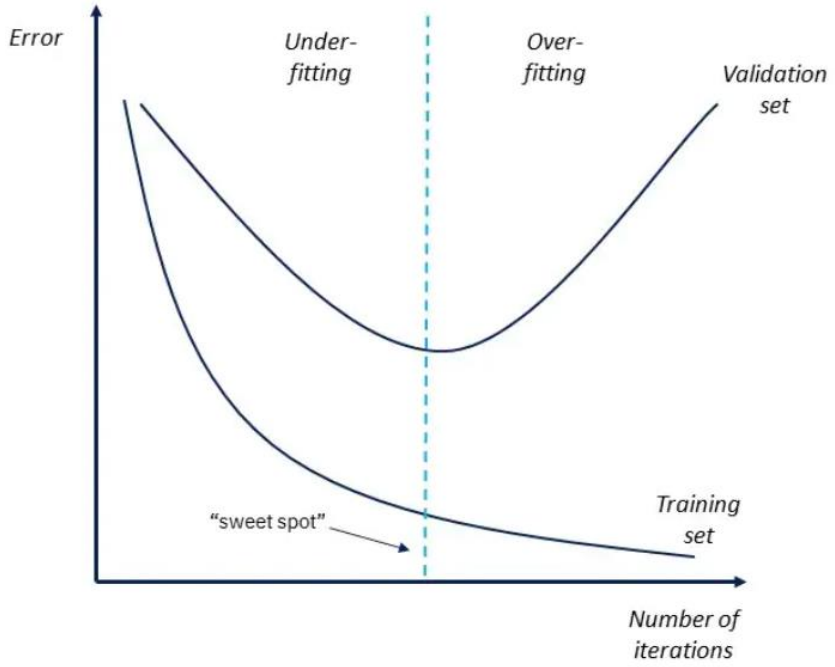
Gradient boosting je tehnika strojnog učenja koja se najčešće koristi kod regresije i klasifikacije i pokazala je znatan uspjeh u širokom spektru primjene u praksi. Prema Jerome H. Friedmanu, empirijski dokazi pokazuju da kretanjem u pravom smjeru sa puno malih koraka rezultira boljim predviđanjima sa testnom bazom podataka tj. manja je varijanca. Glavna ideja boostinga je uzastopno dodavanje novih slabih modela tj. temelji se na slijednom učenju. Slabi model je definiran kao svaki model koji ima bar malo bolju mogućnost predviđanja rezultata od nasumičnog odabira.. Svaki novi dodani slabi model uzima u obzir greške koje su napravljene od strane prijašnjih slabih modela. Učeći na greškama prethodnih modela potrebno je manje vremena i iteracija da se predvidi rezultat.

Razlika između predviđanja algoritma i stvarnog rezultata može se prikazati sa „loss function“ (funkcijom gubitka) i cilj je tu funkciju minimizirati. Funkcije gubitka su različite i razlikuju se između regresije i klasifikacije, ali postoje neke standardne funkcije koje se koriste.

Kao što je spomenuto bolja predviđanja dobivamo kada radimo puno malih koraka prema cilju, radi toga je uveden i parametar „learning rate“ koji utječe na brzinu predviđanja tako da smanjuje doprinos svakog stabla.

Gradient Boost daje prvo predviđanje iz slučajnih uzoraka koji je treniran na prvom slabom modelu. Pomoću funkcije gubitka određujemo reziduale (ostatke) i na njima treniramo novi model i to se ponavlja sve dok rezidual ne bude jako blizu nuli tj. dok greška između previđenih vrijednosti i stvarnog rezultate ne bude blizu 0. Ti reziduali su zapravo predviđanja greški. Na te greške utječemo sa „leraning rate-om“ kako bi utjecali na njihov doprinos. Na kraju se zbroji početno previđanje sa svim rezidualima, na taj način se prvo predviđanje modificira kako bi bilo bliže stvarnom rezultatu.

Jako je važno da ne dođemo u fazu gdje je model „overfitted“ (prenaučen) na podate iz treninga, što je jedan od problema Gradient Boost-a . Prenaučen model jako dobro radi na podacima za trening (traning set) ali jako loše na podacima za testiranje modela (test set) tj. radi jako loše na podacima koje nije vidio što je upravo suprotno od onog što je poželjno.



Slika 1 Under-fitting i Over-fitting modela

Kako bi uočili da je model prenaučen na training set vrši se unakrsna provjera valjanosti. Jedna od najčešćih unakrsnih provjera valjanosti je K-folds , koja je korištena u ovom radu i detaljnije je objašnjena u jednom od idućih poglavlja.

2.1. Matematička pozadina Gradient boosting-a (Jerome H. Freidman)

U problemu procjene funkcije postoji sustav koji se sastoji od nasumičnih „output-a" ili odzivne varijable y i skupa nasumičnih „input-a" ili polaznih varijabli $\boldsymbol{x} = \{x_1, \ldots, x_n\}$. Imajući „training" uzorak $\{y_i, \boldsymbol{x_i}\}_1^N$ poznatih y i x varijabli, cilj je pronaći funkciju $F*(\boldsymbol{x})$ koji povezuje varijable **x** i y, tako da je vrijednost specificirane funkcije gubitka $\boldsymbol{\psi}(y, F(\boldsymbol{x}))$ minimizirana u zajedničkoj raspodijeli svih vrijednosti varijabla (y,x.)

$$\boldsymbol{F*(x) = arg\ min\ \min_{F(x)} E_{y,x}\, \psi(y, F(x))} \qquad 1$$

Boosting aproksimira F*($\boldsymbol{x}$) koristeći „additive" proširenjem oblika

$$\boldsymbol{F(x) = \sum_{m=0}^{M} \beta_m\, h(x; a_m)} \qquad 2$$

Gdje su funkcije („base learner“) $h(\boldsymbol{x};\boldsymbol{a})$ odabrane kao jednostavne funkcije od $\boldsymbol{x}$ sa parametrima $\boldsymbol{a} = \{a_1, a_2, \dots\}$. Koeficijent $\{\beta_m\}_0^M$ i parametri $\{\boldsymbol{a}_m\}_0^M$ su prilagođeni podacima za trening na „stage-wise“ način . Počinje se sa početnim predviđanjem $F_0(\boldsymbol{x})$ i dalje za *m=1,2,..., M :*

$$(\boldsymbol{\beta}_m, \boldsymbol{a}_m) = \boldsymbol{arg}\ \min_{\boldsymbol{\beta},\boldsymbol{a}} \sum_{i=0}^{N} \boldsymbol{\psi}(\boldsymbol{y}_i, \boldsymbol{F}_{m-1}(\boldsymbol{x}_i) + \boldsymbol{\beta h}(\boldsymbol{x}_i; \boldsymbol{a})) \quad 3$$

$$\boldsymbol{F}_m(\boldsymbol{x}) = \boldsymbol{F}_{m-1}(\boldsymbol{x}) + \boldsymbol{\beta}_m \boldsymbol{h}(\boldsymbol{x}; \boldsymbol{a}_m) \quad 4$$

Gradient boosting približno rješava (4) za jednostavnu (derivatibilnu) funkciju gubitka s $\boldsymbol{\psi}(y, F(\boldsymbol{x}))$ u dva koraka. Prvo se funkcija $h(\boldsymbol{x};\boldsymbol{a})$ rješava metodom najmanjih kvadrata

$$\boldsymbol{a}_m = \boldsymbol{arg}\min_{\boldsymbol{a},\boldsymbol{\rho}} \sum_{\boldsymbol{i=0}}^{\boldsymbol{n}} [\tilde{\boldsymbol{y}}_{\boldsymbol{im}} - \boldsymbol{\rho h}(\boldsymbol{x}_i; \boldsymbol{a})]^2$$
$$\boldsymbol{5}$$

Trenutni „pseudo“ reziduali (ostatci)

$$\tilde{\boldsymbol{y}}_{\boldsymbol{im}} = -\left[\frac{\boldsymbol{\delta\psi}(\boldsymbol{y}_i, \boldsymbol{F}(\boldsymbol{x}_i))}{\boldsymbol{\delta F}(\boldsymbol{x}_i)}\right]_{\boldsymbol{F}(\boldsymbol{x})=\boldsymbol{F}_{m-1}(\boldsymbol{x})} \quad 6$$

Preko dane funkcije $h(\boldsymbol{x};\boldsymbol{a}_m)$ optimalna vrijednost koeficijenta β_m je određena:

$$\boldsymbol{\beta}_m = \boldsymbol{arg}\min_{\boldsymbol{\beta}} \sum_{i=1}^{N} \boldsymbol{\psi}(\boldsymbol{y}_i, \boldsymbol{F}_{m-1}(\boldsymbol{x}_i) + \boldsymbol{\beta H}(\boldsymbol{x}_i; \boldsymbol{a}_m)) \quad 7$$

Ova strategija zamjenjuje potencijalno tešku funkciju problema optimizacija (4) sa funkcijom baziranoj na metodi najmanjih kvadrata (5), zatim slijedi optimizacija jednog parametra (7) baziranog na kriteriju $\boldsymbol{\psi}$.

Gradient Boosting je specijalizirao ovaj slučaj do toga da su „base learner-i“ $h(\boldsymbol{x};\boldsymbol{a})$ L-terminalni čvorovi regresijskog stabla. Pri svakoj iteraciji *m*, regresijsko stablo pregradi **x**-space u razdvojene regije $\{R_{lm}\}_{l=1}^{L}$ i za svaku predviđa konstantnu vrijednost :

$$\boldsymbol{h(x;\{R_{lm}\}_1^L) = \sum_{l=1}^{L} \bar{y}_{lm} 1(x \epsilon R_{lm})} \quad 8$$

$\bar{y}_{lm} = mean_{x_i \in R_{lm}}(\tilde{y}_{im})$ označava aritmetičku sredinu (6) u svakoj regiji R_{lm}. Parametri ovog „base learner-a" su varijable cijepanja i odgovarajućih točaka cijepanja koje definiraju stablo, koji definira odgovarajuće regije $\{R_{lm}\}_1^L$ particije na m-toj iteraciji. Inducirani su od gore prema dolje tj. od prvog najboljeg, koristeći metodu najmanjih kvadrata za kriterij cijepanja. Sa regresijskim stablima, (7) se može riješiti odvojeno za svaku pojedinačnu regiju R_{lm} definiranu za svaki odgovarajući zadnji čvor l m-tog stabla. Zato što stablo (8) predviđa konstantnu vrijednost $\bar{y}_{lm}$ u svakoj regiji R_{lm} , rješenje (7) je reducirano na jednostavnu prosječnu „lokacije" temeljenu na kriteriju $\boldsymbol{\psi}$

$$\boldsymbol{\gamma_{lm} = arg \min_{\gamma} \sum_{x_i \epsilon R_{lm}} \psi(y_i, F_{m-1}(x_i) + \gamma)} \quad 9$$

Trenutna aproksimacija F_{m-1} je onda ažurirana u svakoj odgovarajućoj regiji

$$\boldsymbol{F_m(x) = F_{m-1}(x) + \nu \cdot \gamma_{lm} 1(x \epsilon R_{lm})} \quad 10$$

„Learning rate" je parametar $0 < \nu < 1$ kontrolira brzinu učenja procesa. Empirijski je uočeno da manje vrijednosti ($\nu \leq 0{,}1$) vode prema puno boljoj generalizaciji greške.

Algorithm 1: Gradient_TreeBoost

1. $F_0(\mathbf{x}) = \arg\min_{\gamma} \sum_{i=1}^{N} \Psi(y_i, \gamma)$
2. For $m = 1$ to M do:
3. $\quad \tilde{y}_{im} = -\left[\frac{\partial \Psi(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x}) = F_{m-1}(\mathbf{x})}, \; i = 1, N$
4. $\quad \{R_{lm}\}_1^L = L - \text{terminal node } tree(\{\tilde{y}_{im}, \mathbf{x}_i\}_1^N)$
5. $\quad \gamma_{lm} = \arg\min_{\gamma} \sum_{\mathbf{x}_i \in R_{lm}} \Psi(y_i, F_{m-1}(\mathbf{x}_i) + \gamma)$
6. $\quad F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \cdot \gamma_{lm} 1(\mathbf{x} \in R_{lm})$
7. endFor

Slika 2 Algoritam Gradietn Boost

3. Baza podataka

Baza podataka koja će se koristiti u ovom radu je dostupna na stranici Kaggle i pogodna je za početak strojnog učenja sa 10-15 numeričkih i kategoričkih varijabli (skupina podataka) . Polazna baza je podijeljena na dva seta podataka train set i test set. Train set sadrži 891 putnika, a test set koji sadrži 418 putnika, koji su različiti od putnika u train setu.

Train set se koristi za vježbanje programa, a pomoću test seta se provjerava točnost programa. U setovima imamo razne podatke o putnicima koji su raspoređeni u skupine. Zajedničke skupine podataka za setove su:

1. PassengerId
2. Pclass
3. Name
4. Sex
5. Age
6. SibSp
7. Parch
8. Ticket
9. Fare
10. Cabin
11. Embarked

Razlika kod train i test seta je što u train setu imamo i skupinu podataka „Surived“ o tome tko je preživio, a tko nije. „Survived“ je ujedno ciljana skupina podataka (varijabla) koju želimo dobiti iz podataka u test setu.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

Slika 3 Baza podatka- Train set #1

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

Slika 4 Baza podataka- Test set #1

Kao što se može primijetiti na slici 3[Slika 3] i slici 4 [Slika 4] u nekim ćelijama ne postoje podaci ("NaN“). Train set ima nepostojeće podatke u skupinama Age, Cabin i Embarked, dok test set ima nepostojeće podatke u Age, Cabin i Fare.

Train set

Survived	0
Pclass	0
Name	0
Sex	0
Age	177
SibSp	0
Parch	0
Ticket	0
Fare	0
Cabin	687
Embarked	2

Test set

Pclass	0
Name	0
Sex	0
Age	86
SibSp	0
Parch	0
Ticket	0
Fare	1
Cabin	327
Embarked	0

Slika 5 Količina nepoznatih podataka u set-ovima

Najviše nepoznatih podataka se nalazi u skupini „Cabin“ oko prilike 80%, dok su nepoznati podaci u ostalim skupina u dosta manjim postocima.

3.1. Kratki opis skupine podataka

- PassengerId pokazuje u kojem redu se nalazi putnik i podaci vezani uz tog putnika. Nema nikakav utjecaj na krajnji rezultat.
- Survived je ciljana varijabla koju želimo pretpostaviti i ima dvije vrijednosti koje iznose:

1 = putnik preživio

0 =putnik nije preživio

- Pclass (Passenger Class) predstavlja socio-ekonomski status putnika i ima tri vrijednosti koje iznose:

  1 =viša klasa

  2 =srednja klasa

  3 =niža klasa

- Name, Sex i Age su jasne same po sebi (ime, rod i godine)
- SibSp prikazuje broj putnikovih braće i sestara ili supružnika
- Parch prikazuje koliko je na brodu prisutno putnikove djece ili roditelja
- Ticket je broj putnikove karte
- Fare je putnička karta
- Cabin je broj putnikove kabine
- Embarked je luka u kojoj su se putnici ukrcali na Titanik i ima 3 vrijednost:

  C = Cherbourg

  Q = Queenstown

  S = Southampton

U tablici [Tablica 1] prikazuje u kojem rasponu se nalaze vrijednosti koje pojedina skupina poprima i vrstu podataka.

Tablica 1 Vrsta podataka po skupinana

Skupina	Vrijednost	Vrste podataka
PassengerId	1-891	Integer
Survived	0,1	Integer
Pclass	1-3	Integer
Name	Name of passengers	Object
Sex	Male, female	Object
Age	0-80	Real
SibSp	0-8	Integer
Parch	0-6	Integer
Ticket	Ticket number	Object
Fare	0-512	Real
Cabin	Cabin number	Object
Embarked	S, C, Q	Object

4. Feature Engineering

4.1. Passenger Title

Putnička imena u skupini „Name“ su zapisana u dobro definiranom uzorku pa će biti lako izvući titule iz skupine i kategorizirati ih.

```
import string

def substrings_in_string(big_string, substrings):
    for substring in substrings:
        if str(big_string).find(substring) != -1:
            return substring
    return np.nan

def replace_titles(x):
    title=x['Title']
    if title in ['Don', 'Major', 'Capt', 'Jonkheer', 'Rev', 'Col']:
        return 'Mr'
    elif title in ['Countess', 'Mme']:
        return 'Mrs'
    elif title in ['Mlle', 'Ms']:
        return 'Miss'
    elif title =='Dr':
        if x['Sex']=='Male':
            return 'Mr'
        else:
            return 'Mrs'
    else:
        return title

def create_title(df):
    title_list=['Mrs', 'Mr', 'Master', 'Miss', 'Major', 'Rev',
                        'Dr', 'Ms', 'Mlle','Col', 'Capt', 'Mme', 'Countess',
                        'Don', 'Jonkheer']
    df['Title']=df['Name'].map(lambda x: substrings_in_string(x, title_list))

    df['Title']=df.apply(replace_titles, axis=1)

create_title(df_train)
create_title(df_test)
```

Funkcija substrings_in_string se u ovom dijelu koristi kako bi pronašla titule u skupini podataka „Name“, koja je zapisana u obliku „Prezime, titula, ime“.

Funkcija replace_titles pomoću for petlje kategorizira titule u 3 skupine Mr, Mrs i Miss.

Funkcija create_title koristi substrings_in_string i replace_titles kako bi napravila novu skupinu podataka koja se zove „Title“. Pri tome koristi title_list sa popisom svih mogućih titula u skupini „Name“.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	Mr
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	Mrs
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	Miss
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	Mrs
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	Mr
...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	Mr
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	Miss
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	Miss
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	Mr
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	Mr

Slika 6 Train set #2

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q	Mr
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S	Mrs
894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q	Mr
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S	Mr
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S	Mrs
...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S	Mr
1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C	Mr
1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S	Mr
1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S	Mr
1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C	Master

Slika 7 Test set #2

4.2. Feature Adult

Koristeći novo napravljenu skupinu podataka „Title“ napravljena je još jedna skupina podataka „Adult“. Funkcija određuje svakog putnika kao odraslu osobu ako mu je titula Mr i Mrs, a sva ostala mjesta određuje kao djecu. Novu skupina podataka „Adult“ koja ima binarne vrijednosti 1 što znači odrasla osoba i 0 što znači dijete.

```
def create_adult(df):
    df['Adult'] = np.where((df['Title'] == 'Mr') | (df['Title'] =='Mrs'), 1, 0)

create_adult(df_train)
create_adult(df_test)
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	Adult
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	Mr	1
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	Mrs	1
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	Miss	0
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	Mrs	1
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	Mr	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	Mr	1
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	Miss	0
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	Miss	0
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	Mr	1
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	Mr	1

Slika 8 Train set #3

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	Adult
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q	Mr	1
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S	Mrs	1
894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q	Mr	1
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S	Mr	1
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S	Mrs	1
...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S	Mr	1
1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C	Mr	1
1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S	Mr	1
1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S	Mr	1
1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C	Master	0

Slika 9 Test set #3

4.3. Deck

Oznake kabina su napisane u obliku SLOVO+BROJ. Velik dio podataka „Cabin“ nedostaje što otežava postupak, ali se ne može zanemariti jer postoji mogućnost da neke kabine imaju veću stopu preživljavanja od drugih. Zato stvaramo novu skupinu podataka „Deck“ kako bi kategorizirali i organizirali podatke za pretpostavku preživljavanja. Slovo označava na kojoj se palubi nalazi kabina te ćemo to izdvojiti u „Deck“.

```
cabin_list = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'T']

def create_cabin(df):
    df['Deck'] = df['Cabin'].map(lambda x: substrings_in_string(x, cabin_list))

create_cabin(df_train)
create_cabin(df_test)
```

Koristimo vlastitu funkciju create_cabin kako bi stvorili novu skupinu podataka i to pomoću substrings_in_string ,koja je već korištena kod stvaranja skupine podatka „Title“, ćemo izdvojiti oznake palube.

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	Mr	1	NaN
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	Mrs	1	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	Miss	0	NaN
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	Mrs	1	C
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	Mr	1	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S	Mr	1	NaN
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S	Miss	0	B
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S	Miss	0	NaN
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C	Mr	1	C
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q	Mr	1	NaN

Slika 10 Train set #4

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q	Mr	1	NaN
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S	Mrs	1	NaN
894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q	Mr	1	NaN
895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S	Mr	1	NaN
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S	Mrs	1	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S	Mr	1	NaN
1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C	Mr	1	C
1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S	Mr	1	NaN
1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S	Mr	1	NaN
1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C	Master	0	NaN

Slika 11 Test set #4

4.4. Family

Nova skupina podataka „Family“ je spoj dvije skupine podataka SibSp i Parch, koja pokazuje koliko članova obitelji ima pojedini putnik na brodu uključujući i tog putnika. Također ćemo iz baze podataka izbaciti skupine SibSp i Parch jer ih dalje nećemo koristiti.

```
def create_family(df):
    df['Family'] = df['SibSp'] + df['Parch'] + 1
    df.drop(columns=[ "SibSp","Parch"], inplace=True)

create_family(df_train)
create_family(df_test)
```

PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family
1	0	3	Braund, Mr. Owen Harris	male	22.0	A/5 21171	7.2500	NaN	S	Mr	1	NaN	2
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	PC 17599	71.2833	C85	C	Mrs	1	C	2
3	1	3	Heikkinen, Miss. Laina	female	26.0	STON/O2. 3101282	7.9250	NaN	S	Miss	0	NaN	1
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	113803	53.1000	C123	S	Mrs	1	C	2
5	0	3	Allen, Mr. William Henry	male	35.0	373450	8.0500	NaN	S	Mr	1	NaN	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	211536	13.0000	NaN	S	Mr	1	NaN	1
888	1	1	Graham, Miss. Margaret Edith	female	19.0	112053	30.0000	B42	S	Miss	0	B	1
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	W./C. 6607	23.4500	NaN	S	Miss	0	NaN	4
890	1	1	Behr, Mr. Karl Howell	male	26.0	111369	30.0000	C148	C	Mr	1	C	1
891	0	3	Dooley, Mr. Patrick	male	32.0	370376	7.7500	NaN	Q	Mr	1	NaN	1

Slika 12 Train set #5

PassengerId	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family
892	3	Kelly, Mr. James	male	34.5	330911	7.8292	NaN	Q	Mr	1	NaN	1
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	363272	7.0000	NaN	S	Mrs	1	NaN	2
894	2	Myles, Mr. Thomas Francis	male	62.0	240276	9.6875	NaN	Q	Mr	1	NaN	1
895	3	Wirz, Mr. Albert	male	27.0	315154	8.6625	NaN	S	Mr	1	NaN	1
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	3101298	12.2875	NaN	S	Mrs	1	NaN	3
...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	NaN	A.5. 3236	8.0500	NaN	S	Mr	1	NaN	1
1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	PC 17758	108.9000	C105	C	Mr	1	C	1
1307	3	Saether, Mr. Simon Sivertsen	male	38.5	SOTON/O.Q. 3101262	7.2500	NaN	S	Mr	1	NaN	1
1308	3	Ware, Mr. Frederick	male	NaN	359309	8.0500	NaN	S	Mr	1	NaN	1
1309	3	Peter, Master. Michael J	male	NaN	2668	22.3583	NaN	C	Master	0	NaN	3

Slika 13 Test set #5

4.5. Surname

Skupina „Surname“ je napravljena kako bi se prikazala poveznica između putnika. U skupini podataka „Name“, koja je zapisana u obliku „Prezime, titula, ime“ se lako može izvući prezime i staviti ih u novu skupinu „Surname“ pomoću funkcije create_surname koja koristi for petlju.

```
def create_surname(df):
    data = df['Name']
    families = []

    for i in range(len(data)):
        name = data.iloc[i]

        if '(' in name:
            name_no_bracket = name.split('(')[0]
        else:
            name_no_bracket = name

        family = name_no_bracket.split(',')[0]
        title = name_no_bracket.split(',')[1].strip().split(' ')[0]

        for c in string.punctuation:
            family = family.replace(c, '').strip()

        families.append(family)

    df['Surname'] = families

create_surname(df_train)
create_surname(df_test)
```

PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	Surname
1	0	3	Braund, Mr. Owen Harris	male	22.0	A/5 21171	7.2500	NaN	S	Mr	1	NaN	2	Braund
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	PC 17599	71.2833	C85	C	Mrs	1	C	2	Cumings
3	1	3	Heikkinen, Miss. Laina	female	26.0	STON/O2. 3101282	7.9250	NaN	S	Miss	0	NaN	1	Heikkinen
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	113803	53.1000	C123	S	Mrs	1	C	2	Futrelle
5	0	3	Allen, Mr. William Henry	male	35.0	373450	8.0500	NaN	S	Mr	1	NaN	1	Allen
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	211536	13.0000	NaN	S	Mr	1	NaN	1	Montvila
888	1	1	Graham, Miss. Margaret Edith	female	19.0	112053	30.0000	B42	S	Miss	0	B	1	Graham
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	W./C. 6607	23.4500	NaN	S	Miss	0	NaN	4	Johnston
890	1	1	Behr, Mr. Karl Howell	male	26.0	111369	30.0000	C148	C	Mr	1	C	1	Behr
891	0	3	Dooley, Mr. Patrick	male	32.0	370376	7.7500	NaN	Q	Mr	1	NaN	1	Dooley

Slika 14 Train set #6

PassengerId	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	Surname
892	3	Kelly, Mr. James	male	34.5	330911	7.8292	NaN	Q	Mr	1	NaN	1	Kelly
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	363272	7.0000	NaN	S	Mrs	1	NaN	2	Wilkes
894	2	Myles, Mr. Thomas Francis	male	62.0	240276	9.6875	NaN	Q	Mr	1	NaN	1	Myles
895	3	Wirz, Mr. Albert	male	27.0	315154	8.6625	NaN	S	Mr	1	NaN	1	Wirz
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	3101298	12.2875	NaN	S	Mrs	1	NaN	3	Hirvonen
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	NaN	A.5. 3236	8.0500	NaN	S	Mr	1	NaN	1	Spector
1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	PC 17758	108.9000	C105	C	Mr	1	C	1	Oliva y Ocana
1307	3	Saether, Mr. Simon Sivertsen	male	38.5	SOTON/O.Q. 3101262	7.2500	NaN	S	Mr	1	NaN	1	Saether
1308	3	Ware, Mr. Frederick	male	NaN	359309	8.0500	NaN	S	Mr	1	NaN	1	Ware
1309	3	Peter, Master. Michael J	male	NaN	2668	22.3583	NaN	C	Master	0	NaN	3	Peter

Slika 15 Test set #6

4.6. SurvivalRate and SurvivalRateWeight

Iste vrijednosti u skupinama „Surname“, „Cabin“, i „Ticket“ ukazuju da postoji nekakva veza između putnika. Zato je potrebno napraviti SurvivalRate skupine podataka koje nam pokazuju prosječnu vrijednost preživjelih putnika koji imaju zajedničko prezime, kabinu ili kartu. SurvivalRate će imati različitu važnosti ako njena vrijednost dolazi od 1 ili od 3 vrijednost. Također dodana je nova skupina podataka koja se zove SurvivalRateWeight .

```
def create_frequencies_and_sr(df_train, df_test):

    # Create frequencies of same value for features Surname, Ticket and Cabin
    df_train_surname_fr = df_train.groupby('Surname')['Surname'].count()
    df_train_ticket_fr = df_train.groupby('Ticket')['Ticket'].count()
    df_train_cabin_fr = df_train.groupby('Cabin')['Cabin'].count()

    # Join in Train and in Test the frequencies of the Train
    df_train = df_train.join(df_train_surname_fr, on="Surname", rsuffix="Frequency")
    df_train = df_train.join(df_train_ticket_fr, on="Ticket", rsuffix="Frequency")
    df_train = df_train.join(df_train_cabin_fr, on="Cabin", rsuffix="Frequency")
    df_test = df_test.join(df_train_surname_fr, on="Surname", rsuffix="Frequency")
    df_test = df_test.join(df_train_ticket_fr, on="Ticket", rsuffix="Frequency")
    df_test = df_test.join(df_train_cabin_fr, on="Cabin", rsuffix="Frequency")
    #
    # Create SurvivalRateWeight, more is the value more is the weight.
    #

    # For the Train, when frequency is 1 we give 1 point to the weight, since the rate will be not usable, same where Cabin value is NaN.
    df_train["SurvivalRateWeight"] = 3 - (df_train[["CabinFrequency"]].isna().sum(axis=1) + \
    (df_train[[ "SurnameFrequency","TicketFrequency","CabinFrequency"]] == 1).sum(axis=1))

    # For the Test, when frequency is NaN we give 1 point to the weight, since the rate will be not usable.
    df_test["SurvivalRateWeight"] =  3 - df_test[[ "SurnameFrequency","TicketFrequency","CabinFrequency"]].isna().sum(axis=1)

    # Fill NaN frequencies with 1 for Train and Test
    df_train['CabinFrequency'].fillna(value=1, inplace=True)
    df_test['SurnameFrequency'].fillna(value=1, inplace=True)
    df_test['TicketFrequency'].fillna(value=1, inplace=True)
    df_test['CabinFrequency'].fillna(value=1, inplace=True)

    # Create SurvivalRate for the three features
    df_train_surname_sr = df_train.groupby('Surname')[['Survived', 'Surname']].mean().add_suffix("SurnameRate")
    df_train_ticket_sr = df_train.groupby('Ticket')[['Survived', 'Ticket',]].mean().add_suffix("TicketRate")
    df_train_cabin_sr = df_train.groupby('Cabin')[['Survived', 'Cabin']].mean().add_suffix("CabinRate")
```

```
    # Join in Train and in Test the rates of the Train
    df_train = df_train.join(df_train_surname_sr, on="Surname")
    df_train = df_train.join(df_train_ticket_sr, on="Ticket")
    df_train = df_train.join(df_train_cabin_sr, on="Cabin")
    df_test = df_test.join(df_train_surname_sr, on="Surname")
    df_test = df_test.join(df_train_ticket_sr, on="Ticket")
    df_test = df_test.join(df_train_cabin_sr, on="Cabin")

    # Correct rates of the Train with central value 0.5 for every passenger who have unique value (or empty) in the three features
    # (Otherwise we would have for this passengers a value equal to the target value)
    df_train['SurvivedSurnameRate'] = [0.5 if freq in [1] else (rate*freq-val)/(freq-1) for freq,val,rate in zip(df_train["SurnameFrequency"],df_train["Survived"],df_train['SurvivedSurnameRate'])]
    df_train['SurvivedTicketRate'] = [0.5 if freq in [1] else (rate*freq-val)/(freq-1) for freq,val,rate in zip(df_train["TicketFrequency"],df_train["Survived"],df_train['SurvivedTicketRate'])]
    df_train['SurvivedCabinRate'] = [0.5 if freq in [1] else (rate*freq-val)/(freq-1) for freq,val,rate in zip(df_train["CabinFrequency"],df_train["Survived"],df_train['SurvivedCabinRate'])]

    # Create final SurvivalRate and fill NaN values in the train with central value 0.5
    df_train['SurvivalRate'] = df_train[[ "SurvivedTicketRate","SurvivedCabinRate","SurvivedSurnameRate"]].mean(axis=1)
    df_test['SurvivalRate'] = df_test[["SurvivedTicketRate","SurvivedCabinRate","SurvivedSurnameRate"]].mean(axis=1)
    df_test['SurvivalRate'].fillna(value=0.5, inplace=True)


    drop_cols = ["SurvivedTicketRate","SurvivedCabinRate","SurvivedSurnameRate"]

    df_train.drop(columns=drop_cols, inplace=True)
    df_test.drop(columns=drop_cols, inplace=True)

    return (df_train, df_test)

df_train, df_test = create_frequencies_and_sr(df_train, df_test)

df_train = df_train.drop(columns=['Surname'])
df_test = df_test.drop(columns=['Surname'])
```

Frequency varijable prikazuju koliko se često ponavljaju pojedine vrijednosti u skupinama Surname, Ticket i Cabin iz train seta, te stvaramo nove skupine podataka SurnameFrequency,

TicktFrequency i CabinFrequency u oba seta. U test setu se u tim skupinama podataka nalaze samo podaci koji su zajednički i train i test setu, a na ostalim mjestima se nalazi prazna ćelija ili NaN.

SurvivalRateWeight pokazuje kolika je važnost novonastalih „Frequency“ varijabli i računa se na način:

```
    # For the Train, when frequency is 1 we give 1 point to the weight, since the rate will be not usable, same where Cabin valu
e is NaN.
    df_train["SurvivalRateWeight"] = 3 - (df_train[["CabinFrequency"]].isna().sum(axis=1) + \
    (df_train[[ "SurnameFrequency","TicketFrequency","CabinFrequency"]] == 1).sum(axis=1))

    # For the Test, when frequency is NaN we give 1 point to the weight, since the rate will be not usable.
    df_test["SurvivalRateWeight"] =  3 - df_test[[ "SurnameFrequency","TicketFrequency","CabinFrequency"]].isna().sum(axis=1)
```

PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate
1	0	3	Braund, Mr. Owen Harris	male	22.0	A/5 21171	7.2500	NaN	S	Mr	1	NaN	2	2	1	1.0	1	0.333333
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	PC 17599	71.2833	C85	C	Mrs	1	C	2	1	1	1.0	0	0.500000
3	1	3	Heikkinen, Miss. Laina	female	26.0	STON/O2. 3101282	7.9250	NaN	S	Miss	0	NaN	1	1	1	1.0	0	0.500000
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	113803	53.1000	C123	S	Mrs	1	C	2	2	2	2.0	3	0.000000
5	0	3	Allen, Mr. William Henry	male	35.0	373450	8.0500	NaN	S	Mr	1	NaN	1	2	1	1.0	1	0.666667
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	211536	13.0000	NaN	S	Mr	1	NaN	1	1	1	1.0	0	0.500000
888	1	1	Graham, Miss. Margaret Edith	female	19.0	112053	30.0000	B42	S	Miss	0	B	1	3	1	1.0	1	0.500000
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	W./C. 6607	23.4500	NaN	S	Miss	0	NaN	4	2	2	1.0	2	0.166667
890	1	1	Behr, Mr. Karl Howell	male	26.0	111369	30.0000	C148	C	Mr	1	C	1	1	1	1.0	0	0.500000
891	0	3	Dooley, Mr. Patrick	male	32.0	370376	7.7500	NaN	Q	Mr	1	NaN	1	1	1	1.0	0	0.500000

Slika 16 Train set #7

PassengerId	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate
892	3	Kelly, Mr. James	male	34.5	330911	7.8292	NaN	Q	Mr	1	NaN	1	4.0	1.0	1.0	1	0.75
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	363272	7.0000	NaN	S	Mrs	1	NaN	2	1.0	1.0	1.0	0	0.50
894	2	Myles, Mr. Thomas Francis	male	62.0	240276	9.6875	NaN	Q	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
895	3	Wirz, Mr. Albert	male	27.0	315154	8.6625	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	3101298	12.2875	NaN	S	Mrs	1	NaN	3	1.0	1.0	1.0	2	1.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	NaN	A.5. 3236	8.0500	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	PC 17758	108.9000	C105	C	Mr	1	C	1	1.0	2.0	1.0	1	0.50
1307	3	Saether, Mr. Simon Sivertsen	male	38.5	SOTON/O.Q. 3101262	7.2500	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1308	3	Ware, Mr. Frederick	male	NaN	359309	8.0500	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1309	3	Peter, Master. Michael J	male	NaN	2668	22.3583	NaN	C	Master	0	NaN	3	2.0	2.0	1.0	2	1.00

Slika 17 Test set #7

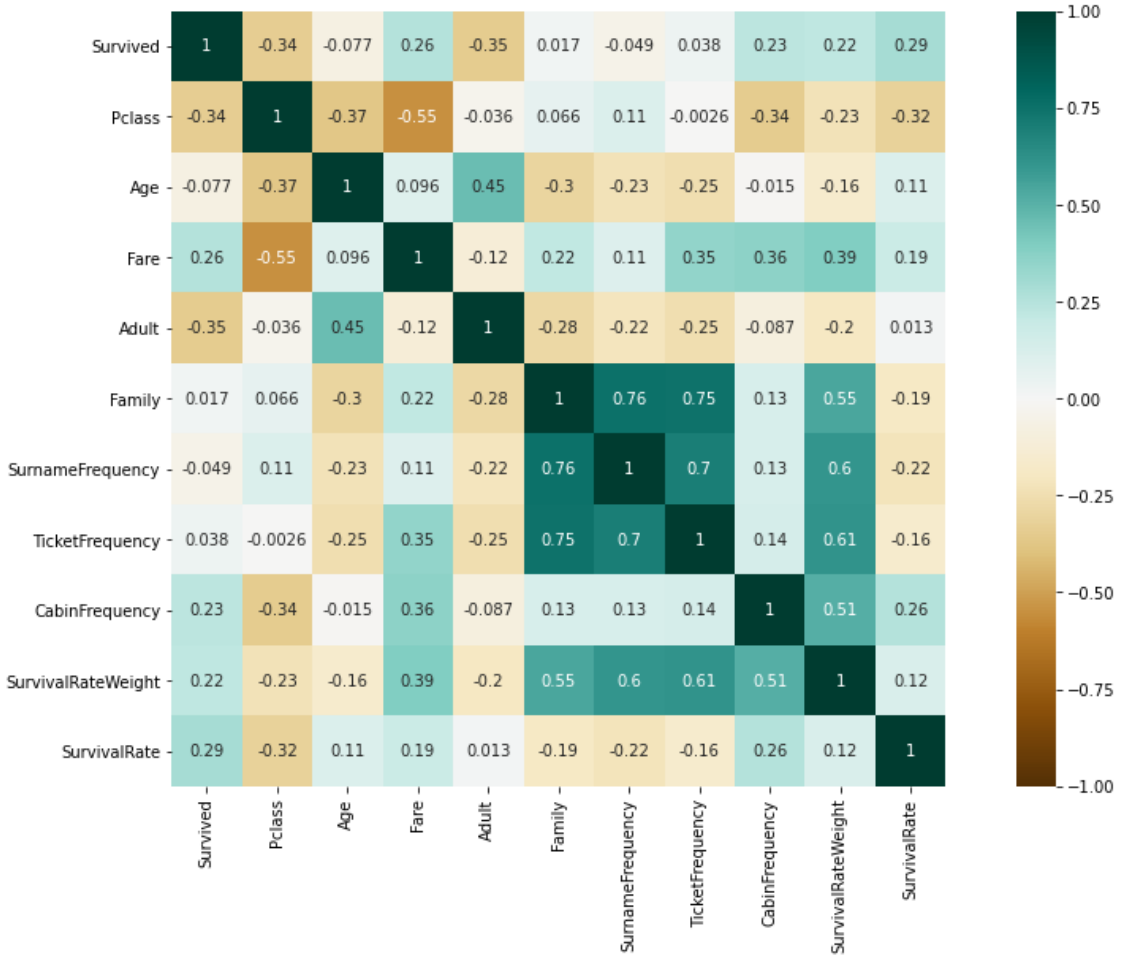
5. Features Correlation

Korelacija predstavlja suodnos ili međusobnu povezanost između dvije varijable, što znači da vrijednosti jedne varijable moguće odrediti poznavanjem vrijednosti druge.

```
from matplotlib import pyplot as plt
import seaborn as sns

a4_dims = (16,9)
fig, ax = plt.subplots(figsize=a4_dims)

sns.heatmap(df_train.corr(), annot = True, vmin=-1, vmax=1, center= 0,cmap='BrBG', square=True)
```



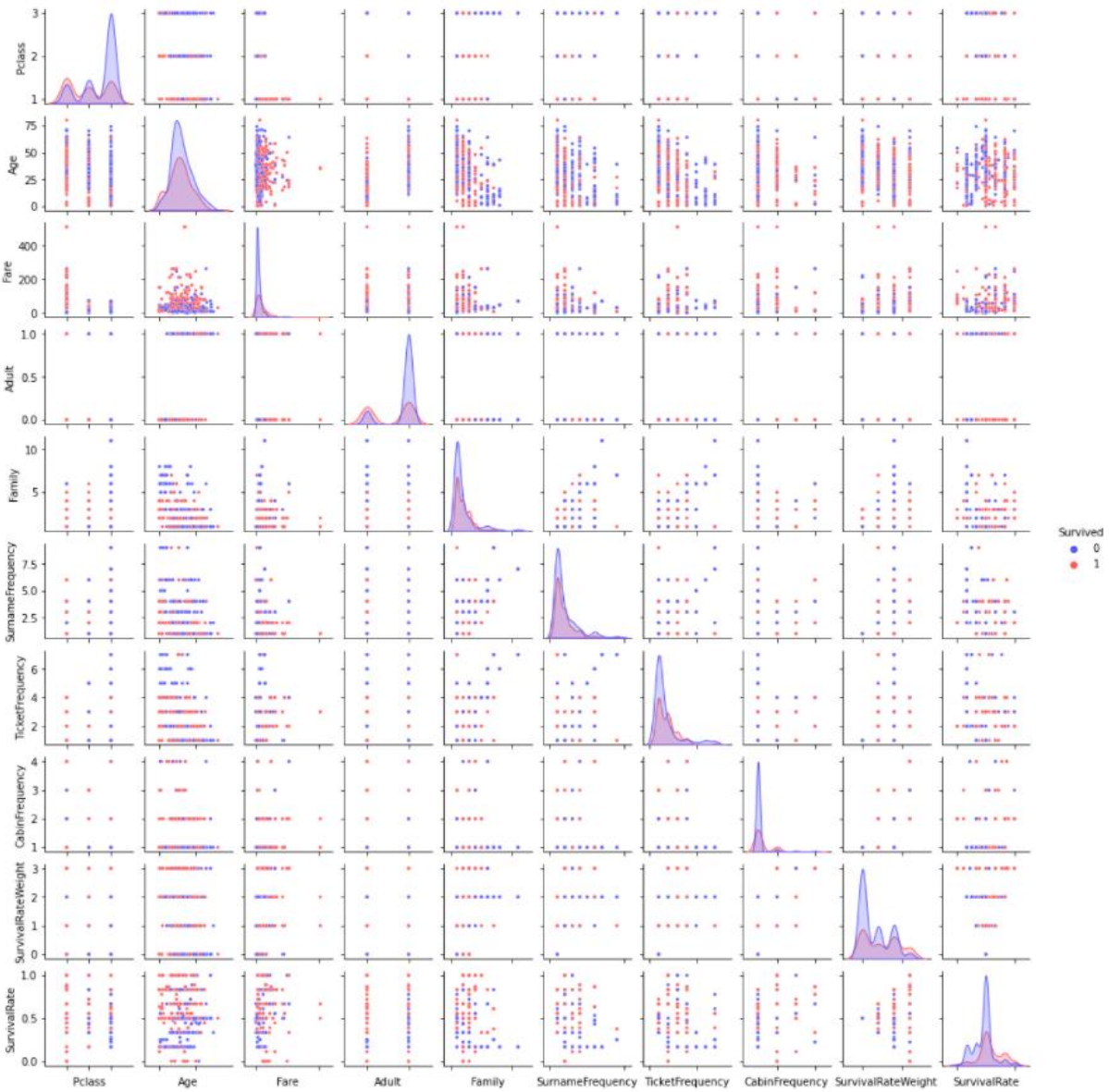
Slika 18 Heatmap - korelacije skupina podataka

Koristili smo funkciju corr() kako bi odredili korelacije između svih skupina podataka u bazi. To smo prikazali pomoću heatmap na slici 18 [Slika 18]. Funkcija corr() vraća jedinstvenu vrijednost koja predstavlja Pearsonov koeficijent korelacije, koji se koristi kada postoji linearna povezanost i neprekidna normalna distribucija između varijabli. Vrijednost Pearsonovog koeficijenta korelacije se kreće od -1 što označava savršenu negativnu korelacija i do +1 što

označava savršenu pozitivnu korelaciju. Predznak nas samo upućuje samo na smjer korelacija ali ne na snagu korelacije.

Također smo koristili funkciju pairplot kako bi stvorili mrežu osi u kojoj bi brojčane vrijednosti bile zajedničke po y-osi kroz cijeli red i x-osi po cijelom stupcu. Pairplot nam omogućava pregled distribucije jedne varijable (po glavnoj dijagonali) i veze između dvije varijable (gornji i donji trokuti).

```
g = sns.pairplot(data=df_train, hue='Survived', palette = 'seismic',
                 height=1.2, diag_kind = 'kde',diag_kws=dict(shade=True),plot_kws=dict(s=10))
g.set(xticklabels=[])
g.fig.set_size_inches(15,15)
```



Slika 19 Pairplot

6. Features Encoding

6.1. Fare and Age binning

Age i Fare su skupine podataka koje imaju kontinuirane varijable. Zato bi bilo praktično podijeliti podatke u „bins“ tj diskretizirati ih, kako bi ih lakše razumjeli. „Bins“ su intervali podatka i između podataka u tim intervalima nema velike razlike u utjecaju na preživljavanje. Kako bi to napravili koristi se k-means algoritmi.

6.1.1. Fare

Količina „bin-ova“ za „Fare“ se određuje prema najvišoj korelaciji sa Survived.

```
from sklearn.preprocessing import KBinsDiscretizer

def corr_bin(val, feat):
    test_bin = pd.read_csv(Path+'/train.csv')
    disc = KBinsDiscretizer(n_bins=val, encode='ordinal', strategy='kmeans')
    test_bin[feat] = disc.fit_transform(df_train[[feat]])
    print(f'"{feat}" correlation with {val} bins: {test_bin.corr()["Survived"][feat]}')
```

```
for i in range(3,10):
    corr_bin(i, "Fare")
```

```
"Fare" correlation with 3 bins: 0.1847121462221732
"Fare" correlation with 4 bins: 0.27649253264235807
"Fare" correlation with 5 bins: 0.25531946735185657
"Fare" correlation with 6 bins: 0.24524336399962288
"Fare" correlation with 7 bins: 0.30136179197156937
"Fare" correlation with 8 bins: 0.29900624014131916
"Fare" correlation with 9 bins: 0.28606052407559984
```

Kao što se može vidjeti na slici iznad „Fare“ ima najveću korelaciju sa „Survived“ kada se koristi 7 bin-ova. Identifikator bin-a je integer, i svaka vrijednost u bin-u ima isti najbliži centroid.

Sva prazna mjesta u Fare u test setu popunjavamo sa srednjom vrijednosti.

```
df_test[["Fare"]] = df_test[["Fare"]].fillna(value=df_train["Fare"].median())
```

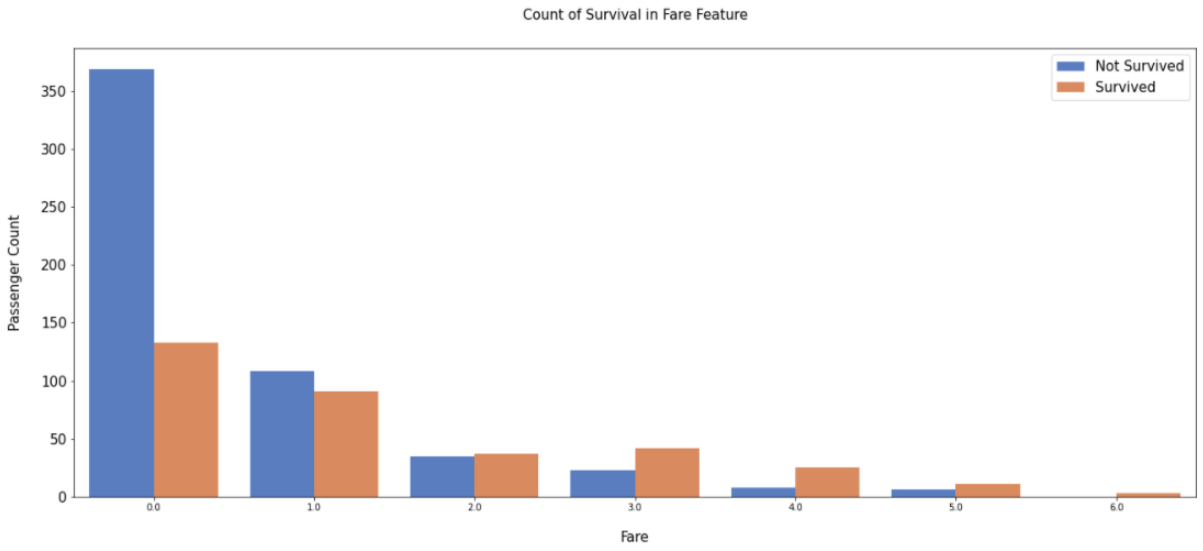
```
disc = KBinsDiscretizer(n_bins=7, encode='ordinal', strategy='kmeans')
disc.fit(df_train[["Fare"]])
df_train["Fare"] = disc.transform(df_train[["Fare"]])
df_test["Fare"] = disc.transform(df_test[["Fare"]])
```

PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate
1	0	3	Braund, Mr. Owen Harris	male	22.0	A/5 21171	0.0	NaN	S	Mr	1	NaN	2	2	1	1.0	1	0.333333
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	PC 17599	3.0	C85	C	Mrs	1	C	2	1	1	1.0	0	0.500000
3	1	3	Heikkinen, Miss. Laina	female	26.0	STON/O2. 3101282	0.0	NaN	S	Miss	0	NaN	1	1	1	1.0	0	0.500000
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	113803	2.0	C123	S	Mrs	1	C	2	2	2	2.0	3	0.000000
5	0	3	Allen, Mr. William Henry	male	35.0	373450	0.0	NaN	S	Mr	1	NaN	1	2	1	1.0	1	0.666667
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	211536	0.0	NaN	S	Mr	1	NaN	1	1	1	1.0	0	0.500000
888	1	1	Graham, Miss. Margaret Edith	female	19.0	112053	1.0	B42	S	Miss	0	B	1	3	1	1.0	1	0.500000
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	W./C. 6607	1.0	NaN	S	Miss	0	NaN	4	2	2	1.0	2	0.166667
890	1	1	Behr, Mr. Karl Howell	male	26.0	111369	1.0	C148	C	Mr	1	C	1	1	1	1.0	0	0.500000
891	0	3	Dooley, Mr. Patrick	male	32.0	370376	0.0	NaN	Q	Mr	1	NaN	1	1	1	1.0	0	0.500000

Slika 20 Train set #8

PassengerId	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate
892	3	Kelly, Mr. James	male	34.5	330911	0.0	NaN	Q	Mr	1	NaN	1	4.0	1.0	1.0	1	0.75
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	363272	0.0	NaN	S	Mrs	1	NaN	2	1.0	1.0	1.0	0	0.50
894	2	Myles, Mr. Thomas Francis	male	62.0	240276	0.0	NaN	Q	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
895	3	Wirz, Mr. Albert	male	27.0	315154	0.0	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	3101298	0.0	NaN	S	Mrs	1	NaN	3	1.0	1.0	1.0	2	1.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	NaN	A.5. 3236	0.0	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	PC 17758	4.0	C105	C	Mr	1	C	1	1.0	2.0	1.0	1	0.50
1307	3	Saether, Mr. Simon Sivertsen	male	38.5	SOTON/O.Q. 3101262	0.0	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1308	3	Ware, Mr. Frederick	male	NaN	359309	0.0	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1309	3	Peter, Master. Michael J	male	NaN	2668	1.0	NaN	C	Master	0	NaN	3	2.0	2.0	1.0	2	1.00

Slika 21 Test set #8

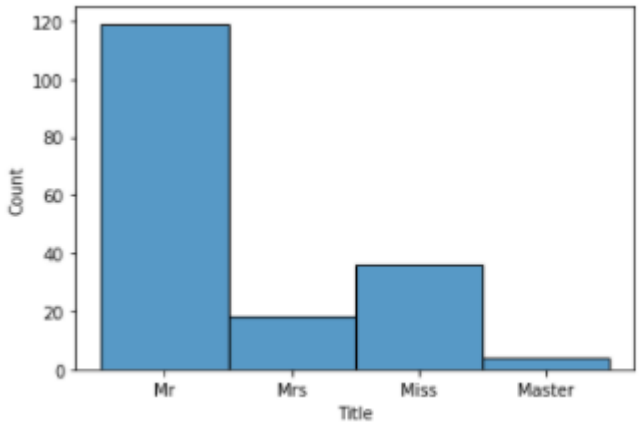


Slika 22 Broj preživjelih i poginulih po "Fare" bin-ovima

Na slici 22 [Slika 22] može se vidjeti broj preživjelih i umrlih kroz „Fare“ bin-ove. Primjećuje se da postotak poginulih opada, a postotak preživjelih polagano raste kroz bin-ove. U bin-ovima sa indikatorom 2 pa do 6 primjećujemo da je postotak preživjelih veći do postotka poginulih za razliku od bin-ova sa indikatorom 0 i 1. Kada usporedimo skupinu „Fare“ u početnoj bazi i trenutnoj , vidi se da ima manje putnika sa većim iznosima fare-a i oni se nalaze u bin-ovima sa višim indikatorima. Putnici koji se nalaze u bin-ovima s većim indikatorima su uglavnom u višoj klasi što upućuje da socio-ekonomski status ima utjecaj na preživljavanje.

6.1.2. Age

Slika 23 [Slika 23] prikazuje koliko putnika ima koju titulu, a da u bazi podatka nedostaje podatak u skupini „Age“. Zapisana je koja je aritmetička vrijednost godina putnika za te titulu.




```
Master     4.574167
Miss      21.835616
Mr        32.891990
Mrs       35.841667
```

Slika 23 Broj nepoznatih vrijednosti u „Age“ i srednja vrijednost godina po tituli

```
age_medians = df_train.groupby(["Title"]).mean()['Age']

df_test[["Age"]] = [age if age == age else age_medians[title] for title,age in zip(df_test['Title'], df_test['Age']) ]
df_train[["Age"]] = [age if age == age else age_medians[title] for title,age in zip(df_train['Title'], df_train['Age']) ]
```

U „Age“ popunjavamo sve prazne ćelije izračunatim aritmetičkim vrijednostima i u train i test setu.

Kao i za „Fare“ broj binova određujemo preko korelacije sa „Survided“. Za „Age“ gledamo najveću apsolutnu korelaciju, jer predznak korelacije nam ne govori o snazi korelacije nego samo o njenom smjeru.

```
"Age" correlation with 3 bins: -0.06347716986563916
"Age" correlation with 4 bins: -0.04158581821608
"Age" correlation with 5 bins: -0.09653908063769598
"Age" correlation with 6 bins: -0.06771711869259558
"Age" correlation with 7 bins: -0.04891425774014369
"Age" correlation with 8 bins: -0.07165243873363161
"Age" correlation with 9 bins: -0.07018412201597271
"Age" correlation with 10 bins: -0.10389521354919691
"Age" correlation with 11 bins: -0.10082092262565953
```

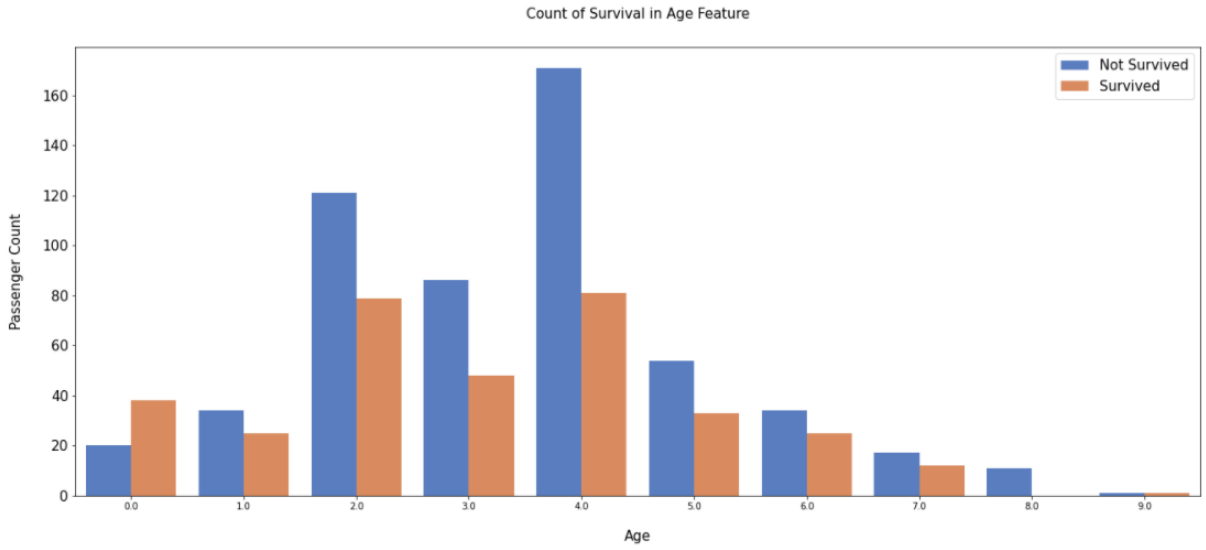
Kao što možemo vidjeti najveću apsolutnu korelaciju „Age“ ima sa „Survived“ je kada se koristi 10 bin-ova.

PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate
1	0	3	Braund, Mr. Owen Harris	male	2.0	A/5 21171	0.0	NaN	S	Mr	1	NaN	2	2	1	1.0	1	0.333333
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	5.0	PC 17599	3.0	C85	C	Mrs	1	C	2	1	1	1.0	0	0.500000
3	1	3	Heikkinen, Miss. Laina	female	3.0	STON/O2. 3101282	0.0	NaN	S	Miss	0	NaN	1	1	1	1.0	0	0.500000
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	4.0	113803	2.0	C123	S	Mrs	1	C	2	2	2	2.0	3	0.000000
5	0	3	Allen, Mr. William Henry	male	4.0	373450	0.0	NaN	S	Mr	1	NaN	1	2	1	1.0	1	0.666667
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	3.0	211536	0.0	NaN	S	Mr	1	NaN	1	1	1	1.0	0	0.500000
888	1	1	Graham, Miss. Margaret Edith	female	2.0	112053	1.0	B42	S	Miss	0	B	1	3	1	1.0	1	0.500000
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	2.0	W./C. 6607	1.0	NaN	S	Miss	0	NaN	4	2	2	1.0	2	0.166667
890	1	1	Behr, Mr. Karl Howell	male	3.0	111369	1.0	C148	C	Mr	1	C	1	1	1	1.0	0	0.500000
891	0	3	Dooley, Mr. Patrick	male	4.0	370376	0.0	NaN	Q	Mr	1	NaN	1	1	1	1.0	0	0.500000

Slika 24 Trein set #9

PassengerId	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate
892	3	Kelly, Mr. James	male	4.0	330911	0.0	NaN	Q	Mr	1	NaN	1	4.0	1.0	1.0	1	0.75
893	3	Wilkes, Mrs. James (Ellen Needs)	female	6.0	363272	0.0	NaN	S	Mrs	1	NaN	2	1.0	1.0	1.0	0	0.50
894	2	Myles, Mr. Thomas Francis	male	7.0	240276	0.0	NaN	Q	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
895	3	Wirz, Mr. Albert	male	3.0	315154	0.0	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	2.0	3101298	0.0	NaN	S	Mrs	1	NaN	3	1.0	1.0	1.0	2	1.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	4.0	A.5. 3236	0.0	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1306	1	Oliva y Ocana, Dona. Fermina	female	5.0	PC 17758	4.0	C105	C	Mr	1	C	1	1.0	2.0	1.0	1	0.50
1307	3	Saether, Mr. Simon Sivertsen	male	5.0	SOTON/O.Q. 3101262	0.0	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1308	3	Ware, Mr. Frederick	male	4.0	359309	0.0	NaN	S	Mr	1	NaN	1	1.0	1.0	1.0	0	0.50
1309	3	Peter, Master. Michael J	male	0.0	2668	1.0	NaN	C	Master	0	NaN	3	2.0	2.0	1.0	2	1.00

Slika 25 Test set #9

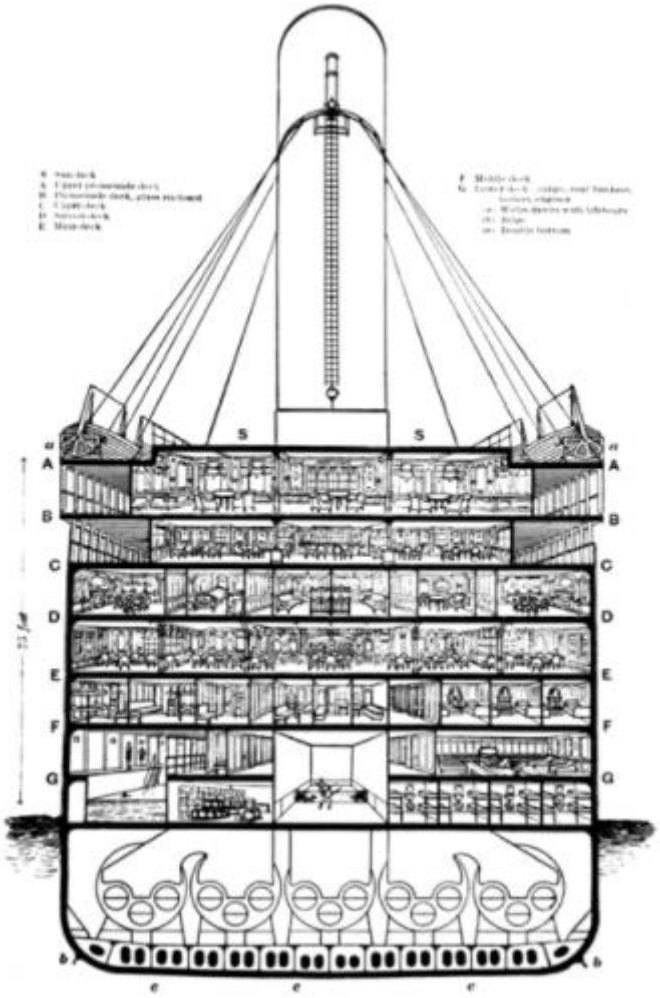


Slika 26 Broj preživjelih i poginulih po "Age"bin-ovima

Na slici 26 [Slika 26] može se vidjeti broj preživjelih i umrlih kroz „Age“ bin-ove. Može se primijetiti da u skoro svakom binu je više poginulih nego preživjelih za razliku od bin 0. Prema ovom vidimo da nijedna generacija nema neku veliku prednost za preživljavanje u odnosu na neku drugu generaciju.

6.2. Deck label encoding

Palube su označene po visini, te se „Deck“ može vrlo jednostavno normalizirati (pridružiti brojčane vrijednosti).



Slika 27 Prikaz paluba na Titaniku

```
deck_normalized = {'A':1, 'B':2, 'C':3, 'D':4, 'E':5, 'F':6, 'G':7, 'T':8}


def encode_cabin(df):
    df.replace({"Deck": deck_normalized}, inplace=True)
    df['Deck'].fillna(value=9, inplace=True)

encode_cabin(df_train)
encode_cabin(df_test)
```

Paluba A dobiva oznaku 1; paluba B dobiva oznaku 2 itd. Prazne ćelije u „Deck“ dobivaju zasebnu vrijednost 9.

PassengerId	Survived	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate
1	0	3	Braund, Mr. Owen Harris	male	2.0	A/5 21171	0.0	NaN	S	Mr	1	9.0	2	2	1	1.0	1	0.333333
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	5.0	PC 17599	3.0	C85	C	Mrs	1	3.0	2	1	1	1.0	0	0.500000
3	1	3	Heikkinen, Miss. Laina	female	3.0	STON/O2. 3101282	0.0	NaN	S	Miss	0	9.0	1	1	1	1.0	0	0.500000
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	4.0	113803	2.0	C123	S	Mrs	1	3.0	2	2	2	2.0	3	0.000000
5	0	3	Allen, Mr. William Henry	male	4.0	373450	0.0	NaN	S	Mr	1	9.0	1	2	1	1.0	1	0.666667
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	3.0	211536	0.0	NaN	S	Mr	1	9.0	1	1	1	1.0	0	0.500000
888	1	1	Graham, Miss. Margaret Edith	female	2.0	112053	1.0	B42	S	Miss	0	2.0	1	3	1	1.0	1	0.500000
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	2.0	W./C. 6607	1.0	NaN	S	Miss	0	9.0	4	2	2	1.0	2	0.166667
890	1	1	Behr, Mr. Karl Howell	male	3.0	111369	1.0	C148	C	Mr	1	3.0	1	1	1	1.0	0	0.500000
891	0	3	Dooley, Mr. Patrick	male	4.0	370376	0.0	NaN	Q	Mr	1	9.0	1	1	1	1.0	0	0.500000

Slika 28 Train set #10

PassengerId	Pclass	Name	Sex	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate
892	3	Kelly, Mr. James	male	4.0	330911	0.0	NaN	Q	Mr	1	9.0	1	4.0	1.0	1.0	1	0.75
893	3	Wilkes, Mrs. James (Ellen Needs)	female	6.0	363272	0.0	NaN	S	Mrs	1	9.0	2	1.0	1.0	1.0	0	0.50
894	2	Myles, Mr. Thomas Francis	male	7.0	240276	0.0	NaN	Q	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50
895	3	Wirz, Mr. Albert	male	3.0	315154	0.0	NaN	S	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	2.0	3101298	0.0	NaN	S	Mrs	1	9.0	3	1.0	1.0	1.0	2	1.00
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	male	4.0	A.5. 3236	0.0	NaN	S	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50
1306	1	Oliva y Ocana, Dona. Fermina	female	5.0	PC 17758	4.0	C105	C	Mr	1	3.0	1	1.0	2.0	1.0	1	0.50
1307	3	Saether, Mr. Simon Sivertsen	male	5.0	SOTON/O.Q. 3101262	0.0	NaN	S	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50
1308	3	Ware, Mr. Frederick	male	4.0	359309	0.0	NaN	S	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50
1309	3	Peter, Master. Michael J	male	0.0	2668	1.0	NaN	C	Master	0	9.0	3	2.0	2.0	1.0	2	1.00

Slika 29 Test set #10

6.3. IsMale binary encoding

Skupinu podataka „Sex“ se uklanja iz tablice i zamjenjuje se sa „IsMale“ koja ima binarne vrijednosti 1 i 0.

1 označuje male (muškarac)

0 označuje female (žena)

```
df_train['IsMale'] = np.where(df_train['Sex'] == 'male', 1, 0)
df_test['IsMale'] = np.where(df_test['Sex'] == 'male', 1, 0)
df_train = df_train.drop(columns=['Sex'])
df_test = df_test.drop(columns=['Sex'])
```

PassengerId	Survived	Pclass	Name	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate	IsMale
1	0	3	Braund, Mr. Owen Harris	2.0	A/5 21171	0.0	NaN	S	Mr	1	9.0	2	2	1	1.0	1	0.333333	1
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	5.0	PC 17599	3.0	C85	C	Mrs	1	3.0	2	1	1	1.0	0	0.500000	0
3	1	3	Heikkinen, Miss. Laina	3.0	STON/O2. 3101282	0.0	NaN	S	Miss	0	9.0	1	1	1	1.0	0	0.500000	0
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	4.0	113803	2.0	C123	S	Mrs	1	3.0	2	2	2	2.0	3	0.000000	0
5	0	3	Allen, Mr. William Henry	4.0	373450	0.0	NaN	S	Mr	1	9.0	1	2	1	1.0	1	0.666667	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	3.0	211536	0.0	NaN	S	Mr	1	9.0	1	1	1	1.0	0	0.500000	1
888	1	1	Graham, Miss. Margaret Edith	2.0	112053	1.0	B42	S	Miss	0	2.0	1	3	1	1.0	1	0.500000	0
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	2.0	W./C. 6607	1.0	NaN	S	Miss	0	9.0	4	2	2	1.0	2	0.166667	0
890	1	1	Behr, Mr. Karl Howell	3.0	111369	1.0	C148	C	Mr	1	3.0	1	1	1	1.0	0	0.500000	1
891	0	3	Dooley, Mr. Patrick	4.0	370376	0.0	NaN	Q	Mr	1	9.0	1	1	1	1.0	0	0.500000	1

Slika 30 Train set #11

PassengerId	Pclass	Name	Age	Ticket	Fare	Cabin	Embarked	Title	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate	IsMale
892	3	Kelly, Mr. James	4.0	330911	0.0	NaN	Q	Mr	1	9.0	1	4.0	1.0	1.0	1	0.75	1
893	3	Wilkes, Mrs. James (Ellen Needs)	6.0	363272	0.0	NaN	S	Mrs	1	9.0	2	1.0	1.0	1.0	0	0.50	0
894	2	Myles, Mr. Thomas Francis	7.0	240276	0.0	NaN	Q	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50	1
895	3	Wirz, Mr. Albert	3.0	315154	0.0	NaN	S	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50	1
896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	2.0	3101298	0.0	NaN	S	Mrs	1	9.0	3	1.0	1.0	1.0	2	1.00	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	Spector, Mr. Woolf	4.0	A.5. 3236	0.0	NaN	S	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50	1
1306	1	Oliva y Ocana, Dona. Fermina	5.0	PC 17758	4.0	C105	C	Mr	1	3.0	1	1.0	2.0	1.0	1	0.50	0
1307	3	Saether, Mr. Simon Sivertsen	5.0	SOTON/O.Q. 3101262	0.0	NaN	S	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50	1
1308	3	Ware, Mr. Frederick	4.0	359309	0.0	NaN	S	Mr	1	9.0	1	1.0	1.0	1.0	0	0.50	1
1309	3	Peter, Master. Michael J	0.0	2668	1.0	NaN	C	Master	0	9.0	3	2.0	2.0	1.0	2	1.00	1

Slika 31 Test set #11

6.4. Embarked and Title one hot encoding

One hot encoding se koristi za pretvorbu kategoričkih podataka u format koji se lako koristi u algoritmima za strojno učenje. Glavna ideja za one hot encoding je stvaranje novih skupina podatka sa vrijednostima 0 i 1 i da pri tome predstavljaju iste vrijednosti kao i skupine koje smo zamijenili.

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer

def encode_onehot(df_train, df_test, cols):

    df_train = df_train.reset_index()
    df_test = df_test.reset_index()


    to_be_encoded = [c for c in df_train.columns if c in cols]

    if len(to_be_encoded) > 0:
        enc = OneHotEncoder(handle_unknown='ignore')

        enc_df_train = pd.DataFrame(enc.fit_transform(df_train[to_be_encoded]).toarray())
        enc_df_train.columns = enc.get_feature_names(to_be_encoded)

        df_train = df_train.join(enc_df_train)
        df_train = df_train.drop(columns=[c for c in df_train.columns if c in to_be_encoded])


        enc_df_test = pd.DataFrame(enc.transform(df_test[to_be_encoded]).toarray())
        enc_df_test.columns = enc_df_train.columns
        df_test = df_test.join(enc_df_test)
        df_test = df_test.drop(columns=[c for c in df_test.columns if c in to_be_encoded])

    return (df_train.set_index("PassengerId"), df_test.set_index("PassengerId"))


df_train['Embarked'].fillna(value='others', inplace=True)
df_test['Embarked'].fillna(value='others', inplace=True)
df_train, df_test = encode_onehot(df_train, df_test, ['Embarked', 'Title'])

df_train = df_train.drop(columns=['Embarked_others'])
df_test = df_test.drop(columns=['Embarked_others'])
```

```
to_drop = ['Name', 'Ticket','Cabin']

df_train = df_train.drop(columns=to_drop)
df_test = df_test.drop(columns=to_drop)
```

Dodano je 7 novih stupaca (skupina podataka).

Embarked_C, Embarked_Q, Embarked_S su nastali iz skupine „Ebmarked“ , koju smo uklonili iz tablice jer više nije potrebna. U tim novim skupinama 1 označava da je putnik ukrcao na brod u toj luci, a 0 označava da se putnik nije ukrcao u toj luci nego u nekoj drugoj.

Title_Mister, Title-Miss, Title_Mr, Title_Mrs su nastali iz skupine „Title“, koju smo također uklonili jer više nije potrebna. U ovim skupinama 1 označava da osoba ima tu titulu, a 0 da nema tu titulu tj. da ima neku drugu.

Također smo uklonili skupine „Name“, „Ticket“ i „Cabin“ jer nam nisu potrebne za dalji rad.

Dobivena baza podatka ima isključivo brojčane vrijednosti, što će olakšati skaliranje.

PassengerId	Survived	Pclass	Age	Fare	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate	IsMale	Embarked_C	Embarked_Q	Embarked_S	Title_Master	Title_Miss	Title_Mr	Title_Mrs
1	0	3	2.0	0.0	1	9.0	2	2	1	1.0	1	0.333333	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
2	1	1	5.0	3.0	1	3.0	2	1	1	1.0	0	0.500000	0	1.0	0.0	0.0	0.0	0.0	0.0	1.0
3	1	3	3.0	0.0	0	9.0	1	1	1	1.0	0	0.500000	0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
4	1	1	4.0	2.0	1	3.0	2	2	2	2.0	3	0.000000	0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
5	0	3	4.0	0.0	1	9.0	1	2	1	1.0	1	0.666667	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	3.0	0.0	1	9.0	1	1	1	1.0	0	0.500000	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
888	1	1	2.0	1.0	0	2.0	1	3	1	1.0	1	0.500000	0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
889	0	3	2.0	1.0	0	9.0	4	2	2	1.0	2	0.166667	0	0.0	0.0	1.0	0.0	1.0	0.0	0.0
890	1	1	3.0	1.0	1	3.0	1	1	1	1.0	0	0.500000	1	1.0	0.0	0.0	0.0	0.0	1.0	0.0
891	0	3	4.0	0.0	1	9.0	1	1	1	1.0	0	0.500000	1	0.0	1.0	0.0	0.0	0.0	1.0	0.0

Slika 32 Train set #12

PassengerId	Pclass	Age	Fare	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate	IsMale	Embarked_C	Embarked_Q	Embarked_S	Title_Master	Title_Miss	Title_Mr	Title_Mrs
892	3	4.0	0.0	1	9.0	1	4.0	1.0	1.0	1	0.75	1	0.0	1.0	0.0	0.0	0.0	1.0	0.0
893	3	6.0	0.0	1	9.0	2	1.0	1.0	1.0	0	0.50	0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
894	2	7.0	0.0	1	9.0	1	1.0	1.0	1.0	0	0.50	1	0.0	1.0	0.0	0.0	0.0	1.0	0.0
895	3	3.0	0.0	1	9.0	1	1.0	1.0	1.0	0	0.50	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
896	3	2.0	0.0	1	9.0	3	1.0	1.0	1.0	2	1.00	0	0.0	0.0	1.0	0.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1305	3	4.0	0.0	1	9.0	1	1.0	1.0	1.0	0	0.50	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
1306	1	5.0	4.0	1	3.0	1	1.0	2.0	1.0	1	0.50	0	1.0	0.0	0.0	0.0	0.0	1.0	0.0
1307	3	5.0	0.0	1	9.0	1	1.0	1.0	1.0	0	0.50	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
1308	3	4.0	0.0	1	9.0	1	1.0	1.0	1.0	0	0.50	1	0.0	0.0	1.0	0.0	0.0	1.0	0.0
1309	3	0.0	1.0	0	9.0	3	2.0	2.0	1.0	2	1.00	1	1.0	0.0	0.0	1.0	0.0	0.0	0.0

Slika 33 Test set #12

6.5. Scaling dataset

Skaliramo pomoću RobustScaler kako bi poboljšali performanse modela. RobustScaler se zasniva na percentilima i zbog toga nije pod utjecajem nekolicine marginalnih odstupanja. Raspon koji se koristiti za skaliranje svake varijable je zadan kao IQR (Interquartile Range(Interkvartilni raspon)) koji je omeđen sa 25 i 75 percentilom.

Kako bi skalirali samo željene skupine koristimo ColumnTransformer, kako bi to radilo treba unaprijed definirati skupine koji će se transformirati tj. skalirati u ovom slučaju sve skupine su zapisane col_names. Također želimo zadržati sve ostale skupine, u bazi, koje se neće skalirati (skupine koji nisu u col_names) zato koristimo remainder='passthrough'.

```
from sklearn.preprocessing import RobustScaler
from sklearn.compose import ColumnTransformer

def scale(df_train, df_test):

    train_copy = df_train.drop(columns=['Survived'])
    col_names = [c for c in train_copy.columns if c not in  ['Survived']]

    ct = ColumnTransformer([('rsc', RobustScaler(), col_names)], remainder='passthrough')


    scal_train = ct.fit_transform(train_copy)
    scal_test = ct.transform(df_test)

    return (pd.DataFrame(scal_train, index=df_train.index, columns=train_copy.columns).join(df_train['Survived']),
            pd.DataFrame(scal_test, index=df_test.index, columns=df_test.columns))

df_train, df_test = scale(df_train, df_test)
```

Kako bi mogli trenirati model df_train rastavljamo na dva dijela X_train i y_train pomoću funkcije df_split . Y_train se sastoji samo od skupina „PassengerId“ i „Survived“ dok X-train također ima „PassengerId“ i sve ostale skupine osim „Survived“.

```
def df_split(df, col):
    y = df[col]
    X = df.loc[:, df.columns != col]
    return (X,y)

X_train, y_train = df_split(df_train, 'Survived')
```

```
PassengerId
1      0
2      1
3      1
4      1
5      0
      ..
887    0
888    1
889    0
890    1
891    0
Name: Survived,
```

Slika 34 y_train

PassengerId	Pclass	Age	Fare	Adult	Deck	Family	SurnameFrequency	TicketFrequency	CabinFrequency	SurvivalRateWeight	SurvivalRate	IsMale	Embarked_C	Embarked_Q	Embarked_S	Title_Master	Title_Miss	Title_Mr	Title_Mrs
1	0.0	-0.5	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	-0.166667	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	-2.0	1.0	3.0	0.0	-6.0	1.0	0.0	0.0	0.0	-0.5	0.000000	-1.0	1.0	0.0	-1.0	0.0	0.0	-1.0	1.0
3	0.0	0.0	0.0	-1.0	0.0	0.0	0.0	0.0	0.0	-0.5	0.000000	-1.0	0.0	0.0	0.0	0.0	1.0	-1.0	0.0
4	-2.0	0.5	2.0	0.0	-6.0	1.0	1.0	1.0	1.0	1.0	-0.500000	-1.0	0.0	0.0	0.0	0.0	0.0	-1.0	1.0
5	0.0	0.5	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.166667	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
887	-1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.5	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
888	-2.0	-0.5	1.0	-1.0	-7.0	0.0	2.0	0.0	0.0	0.0	0.000000	-1.0	0.0	0.0	0.0	0.0	1.0	-1.0	0.0
889	0.0	-0.5	1.0	-1.0	0.0	3.0	1.0	1.0	0.0	0.5	-0.333333	-1.0	0.0	0.0	0.0	0.0	1.0	-1.0	0.0
890	-2.0	0.0	1.0	0.0	-6.0	0.0	0.0	0.0	0.0	-0.5	0.000000	0.0	1.0	0.0	-1.0	0.0	0.0	0.0	0.0
891	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	-0.5	0.000000	0.0	0.0	1.0	-1.0	0.0	0.0	0.0	0.0

Slika 35 X_train

7. Model Selection

Stvorena je funkcija modelfit kako bi odabrali najbolji model. Modelfit će obavljati unakrsnu provjeru valjanosti i plotati grafove za lakši pregled. Unakrsna provjera valjanosti (eng. Cross Validation) je statistička metoda koja se koristi kako bi se odredila točnost metoda strojnog učenja, također koristi se kako bi se vidjelo je li model prenaučen na training set.

Za unakrsnu provjeru valjanosti se koristi K-Fold Cross Validation. K-fold ima samo jedan parametar „*k*“ (u kodu FOLDS=5) i predstavlja na koliko će se grupa podijeliti dana baza (train set). „*k*“ se uzima kako bi svaka grupa imala isti broj podataka, ali najčešće vrijednost „*k*“ je 5 ili 10.

K-fold proces se općenito sastoji od sljedećih koraka.:

1. Dataset se nasumično izmiješa
2. Dataset se podjeli u k grupa
3. Za svaku grupu pojedinačno
    a. Uzeti tu grupu i koristiti je kao test grupu (validacijsku grupu)
    b. Ostale grupe koristiti kao training grupe
    c. Fitati model prema training grupama i evaluirati ga na test grupi
    d. Zadržati rezultat evaluacije i odbaciti model
4. Točnost modela je srednja vrijednost točnosti svake grupe

```
from sklearn import metrics
from sklearn.model_selection import GridSearchCV, cross_val_score, cross_val_predict

METRIC = 'f1'
FOLDS = 5

def modelfit(X, y, alg, cv_folds=FOLDS, plot=False):

    #Fit the algorithm on the data
    alg.fit(X, y)

    #Predict training set:
    dtrain_predictions = alg.predict(X)
    dtrain_predprob = alg.predict_proba(X)[:,1]
    #Perform cross-validation:
    cv_score = cross_val_score(alg, X, y, cv=cv_folds, scoring=METRIC, n_jobs = -1)
    cv_pred = cross_val_predict(alg, X, y, cv=cv_folds, n_jobs = -1)
```

Modelfit pravi dva grafa koji predstavljaju GradeintBoostingClassifier Model Report:

1. Cofusion matrix (Heatmap)
2. ROC krivulja (Reciver Operating Characteristic krivulja)

Confusion matrix se koristi za definiranje performanse klasifikacijskog algoritma tako da je vizualizira i sažme u obliku tablice. Confusion matrix se sastoji od 4 osnovne karakteristike:

1. TP- True Positive
2. TN- True Negative
3. FP- False Postive
4. FN- Falce Negative

ROC krivulja je graf koji prikazuje performansu modela na svim njegovim klasifikacijskim pragovima. Ima dva parametra:

1. True Positive Rate (TPR)
2. False Positiva Rate (FPR)

Ti parametri su definirani:

$$TPR = \frac{TP}{TP + FN} \qquad FPR = \frac{FP}{FP + TN}$$

Površina ispod ROC krivulja se naziva AUC (Area under the ROC Curve) i predstavlja koliko je model dobar u odvajanju klasa .AUC ima vrijednosti u rasponu od 0 do 1. Cilj je imati AUC što bliže vrijednosti 1 jer to pokazuje da model ima točne pretpostavke. Ako AUC jednaka nuli to znači da model uvijek ima pogrešne pretpostavke, tj. kada model pretpostavi 0 da je odgovor zapravo uvijek 1 i obrnuto. Ako AUC ima vrijednost 0,5 to znači da model nema kapacitet odvajanja, tj. nasumični odabir.

```
if(plot):                                                                                                    # ova

    #Print model report:
    desc = "Train Accuracy: %.4g | " % metrics.accuracy_score(y.values, dtrain_predictions)
    desc = desc + " Train AUC Score: %f\n" % metrics.roc_auc_score(y, dtrain_predprob)

    desc = desc + "CV Score : Mean - %.7g | Std - %.7g | Min - %.7g | Max - %.7g" % \
            (np.mean(cv_score),np.std(cv_score),np.min(cv_score),np.max(cv_score))

    #Print Summary:
    a4_dims = (16, 5)
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=a4_dims)
    fig.suptitle(f'{type(alg).__name__} Model Report')
    fig.subplots_adjust(bottom=0.2)
    fig.text(0.5, 0, desc, ha='center', va='center')

    ax1.title.set_text("Confusion matrix")
    cf_matrix = metrics.confusion_matrix(y.values, cv_pred)
    group_names = ['TN','FP','FN','TP']
    group_counts = ["{0:0.0f}".format(value) for value in cf_matrix.flatten()]
    group_percentages = ["{0:.2%}".format(value) for value in cf_matrix.flatten()/np.sum(cf_matrix)]
    labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in zip(group_names,group_counts,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cf_matrix, annot=labels, fmt="", cmap='Blues', xticklabels=False, yticklabels=False, ax=ax1)

    fpr, tpr, threshold = metrics.roc_curve(y.values, cv_pred)
    roc_auc = metrics.auc(fpr, tpr)

    ax2.title.set_text("Receiver Operating Characteristic")
    ax2.set_ylabel('True Positive Rate')
    ax2.set_xlabel('False Positive Rate')
    sns.lineplot(x=fpr, y = tpr, ax=ax2)
    sns.lineplot(x=[0, 1], y = [0, 1], ax=ax2)
    ax2.lines[1].set_linestyle("--")
    ax2.legend(['AUC: %0.2f' % roc_auc], loc="lower right")

return cv_score
```

7.1. Comparing baselines

Kako bi se odredilo koji algoritma ima najbolju performansu na dani dataset, vrši se unakrsna provjera valjanosti i ocjenjujemo rezultate.

Klasifikatore koji će se uspoređivati su:

- K-nearest Neighbrs
- Support Vector
- Nu-Support Vector
- Decision Tree
- Random Forests
- AdaBoost
- Gradinet Boosting
- Gaussian Naive Bayes
- Logistic Regresion

```
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier, AdaBoostClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC, LinearSVC, NuSVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression

SEED = 11

models = [
    ('kNN',KNeighborsClassifier()),
    ('SVC',SVC(random_state=SEED, probability=True)),
    ('NuSVC',NuSVC(random_state=SEED, probability=True)),
    ('DT',DecisionTreeClassifier(random_state=SEED)),
    ('RF',RandomForestClassifier(random_state=SEED)),
    ('AdaB',AdaBoostClassifier(random_state=SEED)),
    ('GB',GradientBoostingClassifier(random_state=SEED)),
    ('GauNB',GaussianNB()),
    ('LR',LogisticRegression(solver='liblinear'))
]
```

Algoritam sa najboljom performansom je onaj koji ima najveći mean (aritmetičku vrijednost), koji određujemo pomoću modelfit funkcije.

```
results = []
agg = []
names = []
for name, model in models:
    res = modelfit(X_train, y_train, model)
    agg.append([res.mean(), res.std()])
    results.append(res)
    names.append(name)

scores = pd.DataFrame(agg, index=names, columns=["Mean","Std"])
scores = scores.sort_values(by='Mean',ascending=False)

scores
```

Na slici 36 [Slika 36] imamo popis svih algoritama i vrijednosti njihovih mean-ova (artimetičkih sredina) i standardnih devijacija. Poredanih od algoritma sa najvišim mean-om do algoritma s najnižim mean-om. Najbolji algoritam za dani dana set je Gradient Boosting, a najgori je K-nearest Neighbrs.

	Mean	Std
GB	0.798672	0.033499
LR	0.785980	0.030031
AdaB	0.777109	0.026149
RF	0.767756	0.026936
NuSVC	0.766881	0.030412
SVC	0.766576	0.032085
GauNB	0.756196	0.028822
DT	0.746891	0.023528
kNN	0.726321	0.041467

Slika 36 Modeli i zračunati mean i Std

```
# boxplot algorithm comparison
fig = plt.figure(figsize=(12,7))
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
sns.boxplot(data=results)
ax.set_xticklabels(names)
plt.show()
```

Najbolji način za grafičku usporedbu algoritma koristimo boxplot dijagram. Boxplotovi su kompaktni u svom sažimanju podataka i upravo zato ih je dobro koristi za usporedbu.

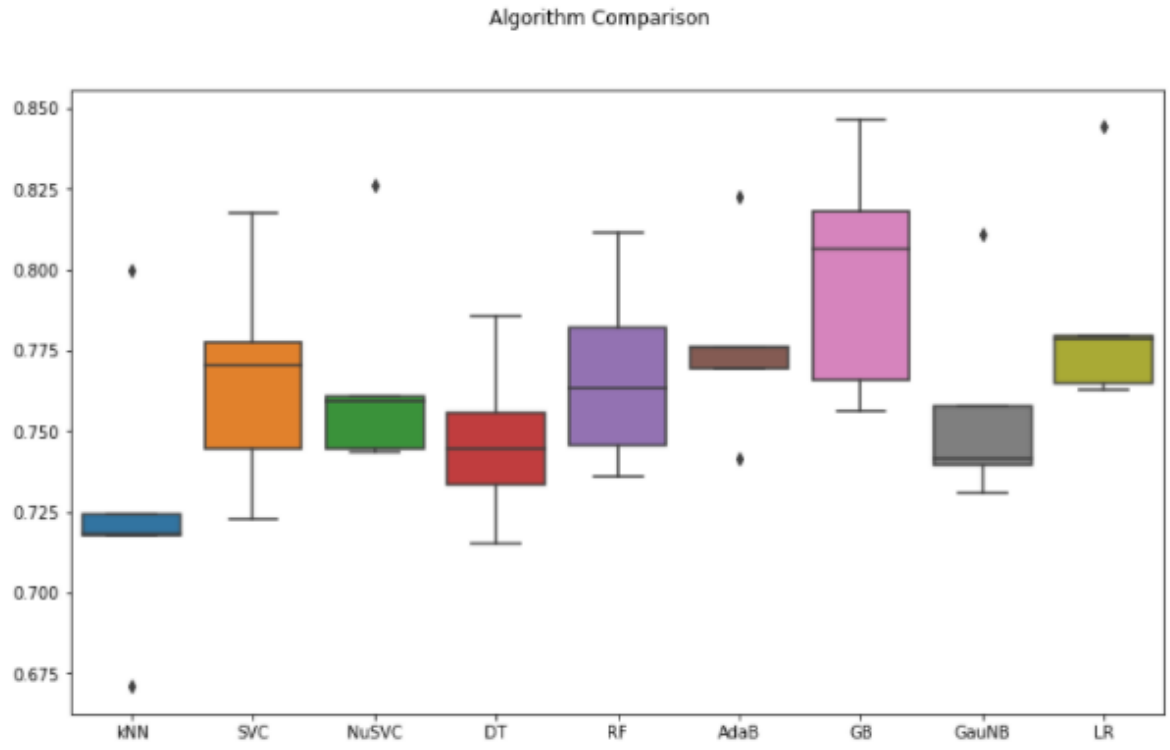
Box plot dijagram je standardni prikaz distribucije podataka na osnovi sažetka pomoću 5 brojčanih vrijednosti koje su:

1. Minimum- najmanja vrijednost isključujući „outliners“
2. Prvi kvartil (Q1)- srednja vrijednost između najmanjeg broja i medijana
3. Medijan -srednja vrijednost
4. Treći kvartil (Q3)- srednja vrijednost između medijana i najveće vrijednosti
5. Maksimum- najveća vrijednost isključujući „outliners“

Boxplot se obično sastoji od box i od para whiskers kako bi se prikazala distribucija jedne ili više grupa brojčanih podataka.

„Box“ u boxplotu označava interkvartilni raspon (Interquartile Range- IQR) i linija unutar box-a označava median (srednju vrijednost).

„Whiskers“ prikazuju rasponi preostalih podataka, te su omeđeni minimumom (lower whisker) i maksimumom (upper whisker). Svi podaci van granica whiskers-a su „outliner-i“.



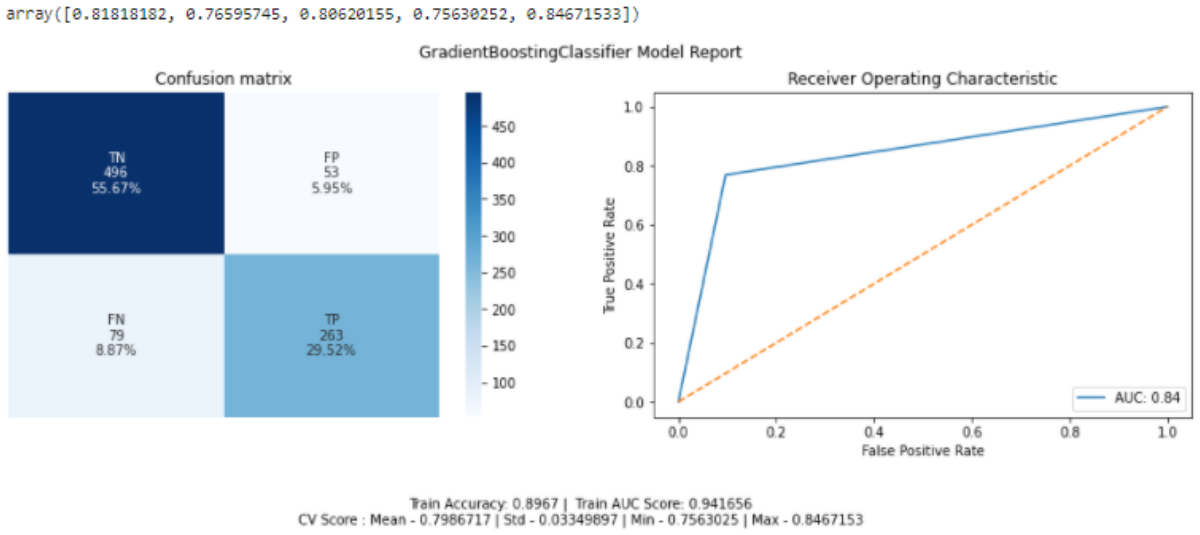
Slika 37 Boxplot prikaz modela

8. Gradient Boost Clasiffier

Algoritam Gradient Boost Clasification radi bolje od ostalih modela, stoga ćemo podesiti hiperparametre.

Podešavanjem hiperparametara ograničavaju se „weak learners“ na specifične načine kako bi osigurali da ostanu takvi.

Pomoću funkcije modelfit izrađeni su grafovi koji nam pokazuju izvješće o početnoj performansi Gradient Boost Classifier.



Slika 38 Izvješće #1

Podaci o performansi koji su prikazani na grafovima su: arry sa rezultatima cross validation za svaki fold, Train Accuracy, train AUC Score, CV Score (Mean, Std, Min, Max)

Hiperparametri se popravljaju i provodi se višestruko pretraživanje mreže kako bi uklopili parametre po važnosti.

```
param_test1 = {'n_estimators':range(10,70,1), 'learning_rate':np.linspace(0.05,0.15,10)}
gsearch1 = GridSearchCV(estimator = GradientBoostingClassifier(max_depth=8, min_samples_split=10, min_samples_leaf=5, max_features='sqrt', subsample=0.8, random_state=SEED),
    param_grid = param_test1, scoring=METRIC, n_jobs=-1,cv=FOLDS)
gsearch1.fit(X_train, y_train)

print(f'{gsearch1.best_score_, gsearch1.best_params_}')
```

Prvi hiperparametri koje ćemo ispravljati su n_estimators i learning_rate.

- n_stimators predstavlja broj faza pojačanja koje treba obaviti. Veći broj faza obično rezultira boljom performansom modela.
- learning_rate predstavlja broj za koliko se smanjuje doprinos svakog stabla, jer kretanjem u pravom smjeru sa puno malih koraka rezultira boljim predviđanjem tj. manja je varijanca.

Ti se parametri zajedno ispravljaju jer su teško uskladivi i teško je napraviti balans među njima.

Dok ispravljamo parametre pratimo i točnost model. Nakon što su ispravljeni n_estimators i learning rate dobili smo vrijednosti točnosti i parametara:

točnost: 0.7924102869494255

learning_rate: 0.10555555555555556,

n_estimators: 38

```
(0.7924102869494255, {'learning_rate': 0.10555555555555556, 'n_estimators': 38})
```

```
param_test2 = {'max_depth':range(1,40,1), 'min_samples_split':range(2,70,1)}
gsearch2 = GridSearchCV(estimator = GradientBoostingClassifier(min_samples_leaf=5, max_features='sqrt', subsample=0.8, random_state=SEED),
    param_grid = {**dict(map(lambda x: (x[0], [x[1]]), gsearch1.best_params_.items())), **param_test2}, scoring=METRIC, n_jobs=-1, cv=FOLDS)
gsearch2.fit(X_train, y_train)

print(f'{ gsearch2.best_score_, gsearch2.best_params_}')
```

Parametre koji se ispravljaju sljedeći su max_depth, min_samples_split.

- max_depth predstavlja maksimalan broj „leaf node „ (zadnjih čvora u stablu odluke)
- min_samples_split predstavlja minimalan broj uzoraka potreban za cijepanje unutarnjeg čvora

Nakon što su ispravljeni max_depth, min_samples_split vrijednosti tih parametara i točnosti iznose:

max_depth: 6

min_samples_split: 23

točnost: 0.8006964125028043

Ostali hiperparametri, koji su se prije mijenjali , ostaju isti.

```
(0.8006964125028043, {'learning_rate': 0.10555555555555556, 'max_depth': 6, 'min_samples_split': 23, 'n_estimators': 38})
```

```
param_test3 = {'min_samples_split':range(2,100,1), 'min_samples_leaf':range(1,40,1)}
gsearch3 = GridSearchCV(estimator = GradientBoostingClassifier(max_features='sqrt', subsample=0.8, random_state=SEED),
    param_grid = {**dict(map(lambda x: (x[0], [x[1]]), gsearch2.best_params_.items())), **param_test3}, scoring=METRIC, n_jobs=-1, cv=FOLDS)
gsearch3.fit(X_train, y_train)

print(f'{gsearch3.best_score_, gsearch3.best_params_}')
```

Sljedeći parametar koji se ispravljaju su min_samples_leaf , te ponovo ispravljamo min_samples_split.

- min_samples_leaf predstavlja minimalni broj uzoraka koji mora biti u „leaf node-u“

Nakon što su ispravljeni min_samples_leaf, min_samples_split vrijednosti tih parametara i točnosti iznose:

min_samples_leaf: 23

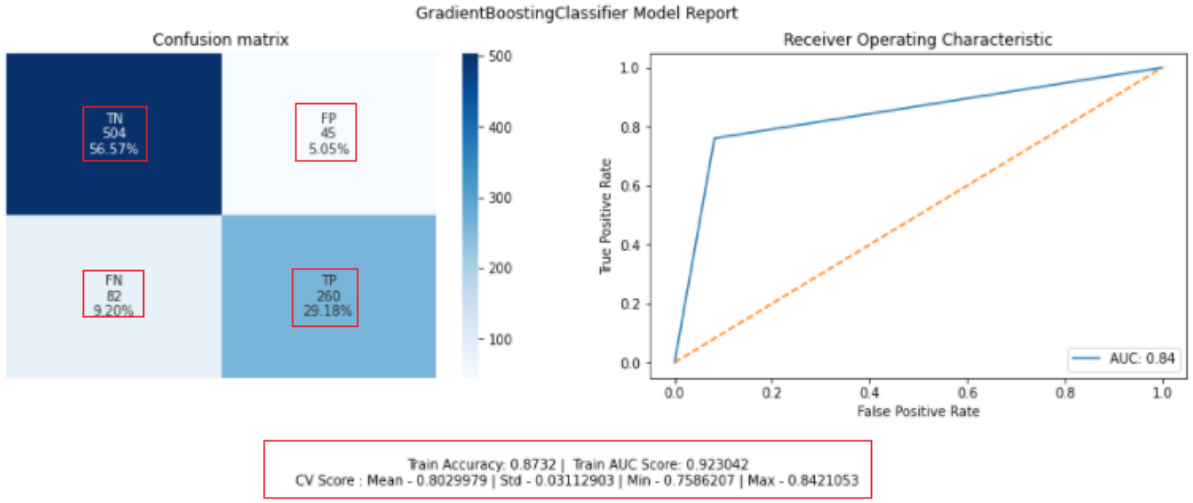
min_samples_split: 47

točnost: 0.8029979350845531

Ostali hiperparametri, koji su se prije mijenjali , ostaju isti.

```
(0.8029979350845531, {'learning_rate': 0.10555555555555556, 'max_depth': 6, 'min_samples_leaf': 23, 'min_samples_split': 47, 'n_estimators': 38})
```

```
array([0.828125  , 0.77697842, 0.80916031, 0.75862069, 0.84210526])
```



Slika 39 Izvješće #2

```
param_test4 = {'max_features':range(1,50,1)}
gsearch4 = GridSearchCV(estimator = GradientBoostingClassifier(subsample=0.8, random_state=SEED),
    param_grid = {**dict(map(lambda x: (x[0], [x[1]]), gsearch3.best_params_.items())), **param_test4}, scoring=METRIC, n_jobs=-1, cv=FOLDS)
gsearch4.fit(X_train, y_train)

print(f'{ gsearch4.best_score_, gsearch4.best_params_}')
```

Sljedeći parametar koji se ispravlja je max_features.

- max_features predstavlja broj značajki koje treba uzeti u obzir kada se traži najbolja podjela.

Nakon što je ispravljen max_features vrijednosti tog parametara iznosi:

max_features: 4

Ostali hiperparametri , koji su se prije mijenjali , i točnost ostaju isti

```
(0.8029979350845531, {'learning_rate': 0.10555555555555556, 'max_depth': 6, 'max_features': 4, 'min_samples_leaf': 23, 'min_samples_split': 47, 'n_estimators': 38})
```

```
param_test5 = {'subsample':np.linspace(0.1,2,58)}
gsearch5 = GridSearchCV(estimator = GradientBoostingClassifier(random_state=SEED),
    param_grid = {**dict(map(lambda x: (x[0], [x[1]]), gsearch4.best_params_.items())), **param_test5}, scoring=METRIC, n_jobs=-1, cv=FOLDS)
gsearch5.fit(X_train, y_train)

print(f'{ gsearch5.best_score_, gsearch5.best_params_}')
```

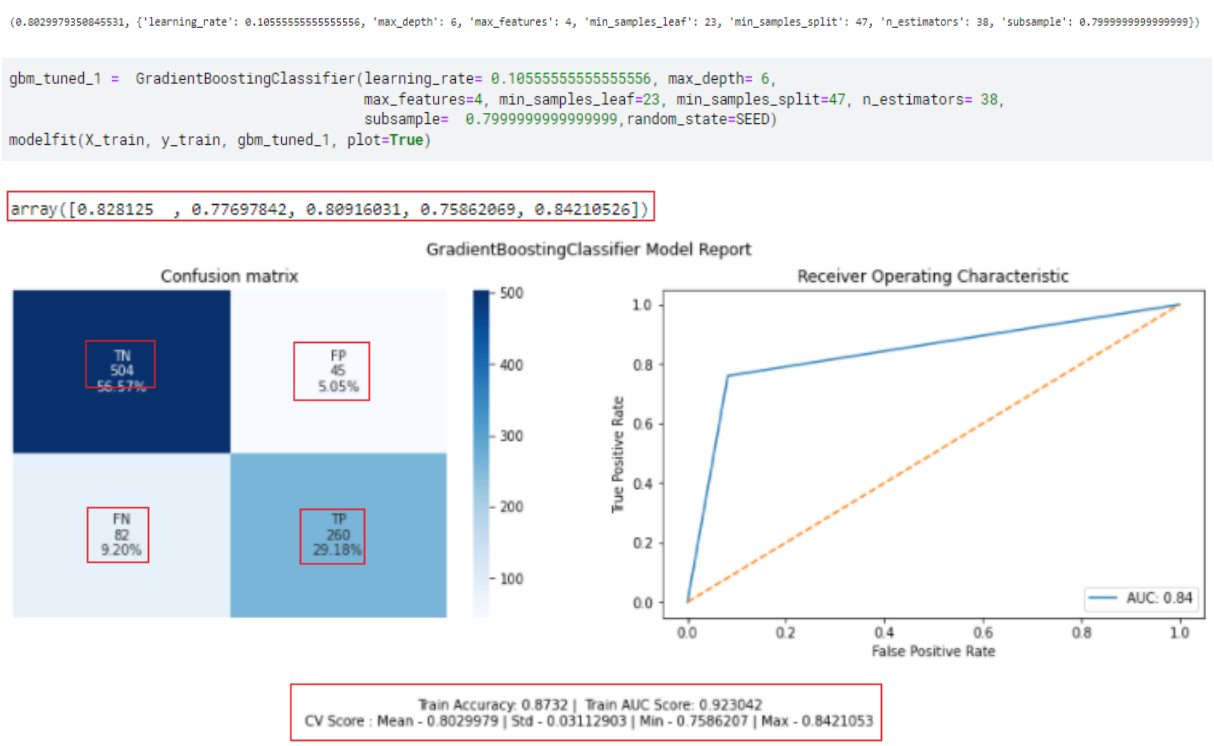
Sljedeći parametar koji se ispravlja je subsample.

- Subsample predstavlja dio uzorka koji se koristi za prilagođavanje pojedinih „base learners“

Nakon što je ispravljen subsample vrijednosti tog parametara iznosi:

subsamle: 0.7999999999999999

Ostali hiperparametri , koji su se prije mijenjali , i točnost ostaju isti.



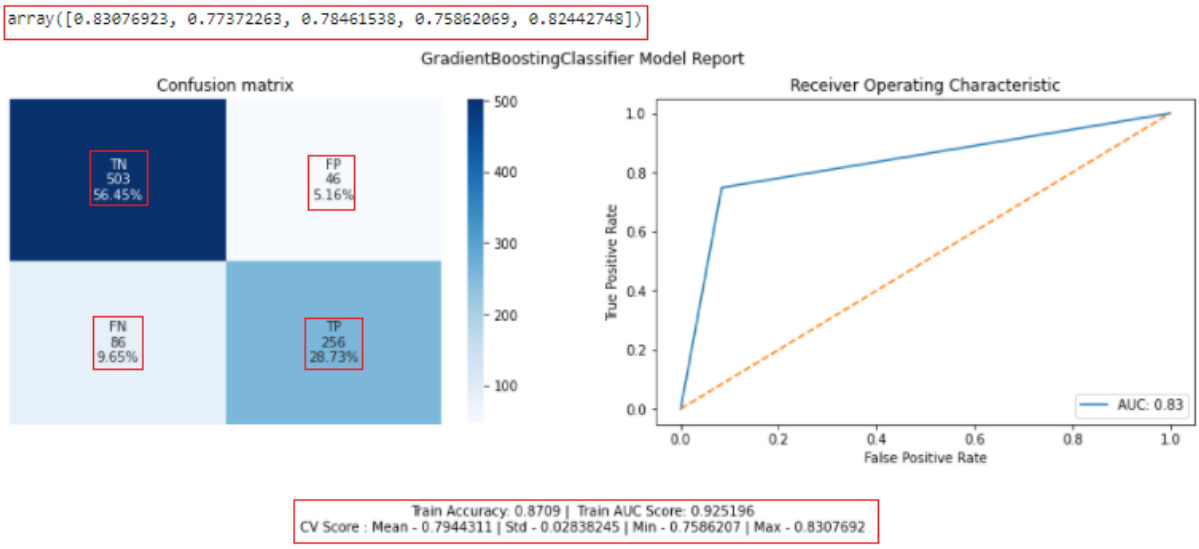
Slika 40 Izvješće #3

Nije došlo do promjena u izvješću što se moglo i pretpostaviti jer mijenjanjem hiperparametara točnost je ostajala nepromijenjena.

Rezultat javne predaje: 0,80622

Modificirat će se learning_rate i n_estimators tako da će learning_rate podijeliti sa istim brojem kojim ćemo pomnožiti n_estimators. U ovom slučaju to je broj 10.

```
gbm_tuned_2 = GradientBoostingClassifier(learning_rate= 0.010555555555555556, n_estimators= 380, max_depth= 6,
                                         max_features=4, min_samples_leaf=23, min_samples_split=47,
                                         subsample= 0.7999999999999999,random_state=SEED)
modelfit(X_train, y_train, gbm_tuned_2, plot=True)
```



Slika 41 Izvješće #4

Iako na prvi pogleda na izvješće izgleda koda je došlo do pogoršanje modela, smanjenje broja točnih predviđanja, train accurancy, train AUC score , rezultat javne predaje predviđanja je porastao.

Rezultat javne predaje: 0,80861

8.1. Fit Model

Ispišemo sve najvažnije hiperparametre koji su se mijenjali.

```
best_model = gbm_tuned_2
best_model.fit(X_train,y_train)
```

```
GradientBoostingClassifier(learning_rate=0.010555555555555556, max_depth=6,
                           max_features=4, min_samples_leaf=23,
                           min_samples_split=47, n_estimators=380,
                           random_state=11, subsample=0.7999999999999999)
```

Df_test se dijeli u 5 reprezentativnih fold-ova, tako da svaka grupa točno predstavlja cijelu bazu. U svakom fold-u nalazi se točan postotak svih podatka iz skupina kao što se nalazi i u cijeloj bazi i takav se fold zove strata. Postupak stvaranja ovakvih fold-ova se zove stratified sampling. Na te grupe ponovo izvodimo K-fold unakrsnu provjeru valjanosti.

```
from sklearn.model_selection import StratifiedKFold

probs = pd.DataFrame(np.zeros((len(df_test), FOLDS * 2)), columns=['Fold_{}_Prob_{}'.format(i, j) for i in range(1, FOLDS + 1) for j in range(2)])
importances = pd.DataFrame(np.zeros((X_train.shape[1], FOLDS)), columns=['Fold_{}'.format(i) for i in range(1, FOLDS + 1)], index=df_test.columns)

skf = StratifiedKFold(n_splits=FOLDS, random_state=SEED, shuffle=True)

for fold, (trn_idx, val_idx) in enumerate(skf.split(X_train, y_train), 1):
    # Fitting the model
    best_model.fit(X_train.iloc[trn_idx, :], y_train.iloc[trn_idx])

    # X_test probabilities
    probs.loc[:, 'Fold_{}_Prob_0'.format(fold)] = best_model.predict_proba(df_test)[:, 0]
    probs.loc[:, 'Fold_{}_Prob_1'.format(fold)] = best_model.predict_proba(df_test)[:, 1]
    importances.iloc[:, fold - 1] = best_model.feature_importances_
```

Kako bi fit-ali model stvaraju se dvije nove tablice probs i importances.

Probs prikazuje vjerojatnosti za preživljavanje i za umiranje u svakom fold-u, dok importances prikazuje važnost svake skupine podatka u svakom fold-u (strata).

	Fold_1_Prob_0	Fold_1_Prob_1	Fold_2_Prob_0	Fold_2_Prob_1	Fold_3_Prob_0	Fold_3_Prob_1	Fold_4_Prob_0	Fold_4_Prob_1	Fold_5_Prob_0	Fold_5_Prob_1
0	0.870701	0.129299	0.846910	0.153090	0.837400	0.162600	0.829655	0.170345	0.819723	0.180277
1	0.427679	0.572321	0.386968	0.613032	0.494353	0.505647	0.404498	0.595502	0.460016	0.539984
2	0.920821	0.079179	0.923542	0.076458	0.924826	0.075174	0.908981	0.091019	0.941155	0.058845
3	0.845494	0.154506	0.870035	0.129965	0.903587	0.096413	0.854909	0.145091	0.866122	0.133878
4	0.165572	0.834428	0.140493	0.859507	0.150179	0.849821	0.155049	0.844951	0.191192	0.808808
...	...	...	...	...	...	...	...	...	...	...
413	0.879608	0.120392	0.881515	0.118485	0.917413	0.082587	0.866532	0.133468	0.886335	0.113665
414	0.264627	0.735373	0.326865	0.673135	0.292261	0.707739	0.309755	0.690245	0.270235	0.729765
415	0.877680	0.122320	0.902853	0.097147	0.922443	0.077557	0.890370	0.109630	0.892636	0.107364
416	0.879608	0.120392	0.881515	0.118485	0.917413	0.082587	0.866532	0.133468	0.886335	0.113665
417	0.109380	0.890620	0.105434	0.894566	0.081516	0.918484	0.114241	0.885759	0.095005	0.904995

Slika 42 Probs tablica #1

	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5
Pclass	0.090327	0.096433	0.087320	0.070453	0.091244
Age	0.029204	0.034228	0.033577	0.032659	0.034455
Fare	0.041901	0.031202	0.031011	0.028513	0.050282
Adult	0.053738	0.055792	0.057202	0.063461	0.059996
Deck	0.046232	0.053811	0.051538	0.049529	0.055465
Family	0.046918	0.038881	0.037228	0.039082	0.028617
SurnameFrequency	0.024949	0.025607	0.027962	0.023971	0.018673
TicketFrequency	0.033111	0.026178	0.033032	0.025139	0.021619
CabinFrequency	0.013388	0.018718	0.017917	0.013166	0.013479
SurvivalRateWeight	0.016265	0.014955	0.012523	0.016029	0.014339
SurvivalRate	0.139599	0.123680	0.118932	0.147083	0.135926
IsMale	0.180451	0.199432	0.199192	0.204373	0.178849
Embarked_C	0.006954	0.006299	0.010163	0.005269	0.010040
Embarked_Q	0.003662	0.001220	0.000722	0.002619	0.002756
Embarked_S	0.011185	0.006392	0.011271	0.011368	0.014644
Title_Master	0.005655	0.002171	0.007011	0.004344	0.005580
Title_Miss	0.026017	0.023870	0.025004	0.039219	0.037568
Title_Mr	0.180186	0.186175	0.182614	0.180500	0.184988
Title_Mrs	0.050256	0.054957	0.055780	0.043222	0.041479

Slika 43 Importance tablica #1

```
importances['Mean_Importance'] = importances.mean(axis=1)
importances.sort_values(by='Mean_Importance', inplace=True, ascending=False)

plt.figure(figsize=(15, 20))
sns.barplot(x='Mean_Importance', y=importances.index, data=importances, palette="viridis")

plt.xlabel('')
plt.tick_params(axis='x', labelsize=15)
plt.tick_params(axis='y', labelsize=15)
plt.title('Mean Feature Importance Between Folds', size=15)

plt.show()
```

Kako bi odredili koja skupina podatka imaju najveći utjecaj na preživljavanje određujemo mean (artimetičku sredinu) svih novih fold-ova. Faktori će se preorganizirati tako da su poredani od faktora sa najvišim mean-om prema najmanjem.

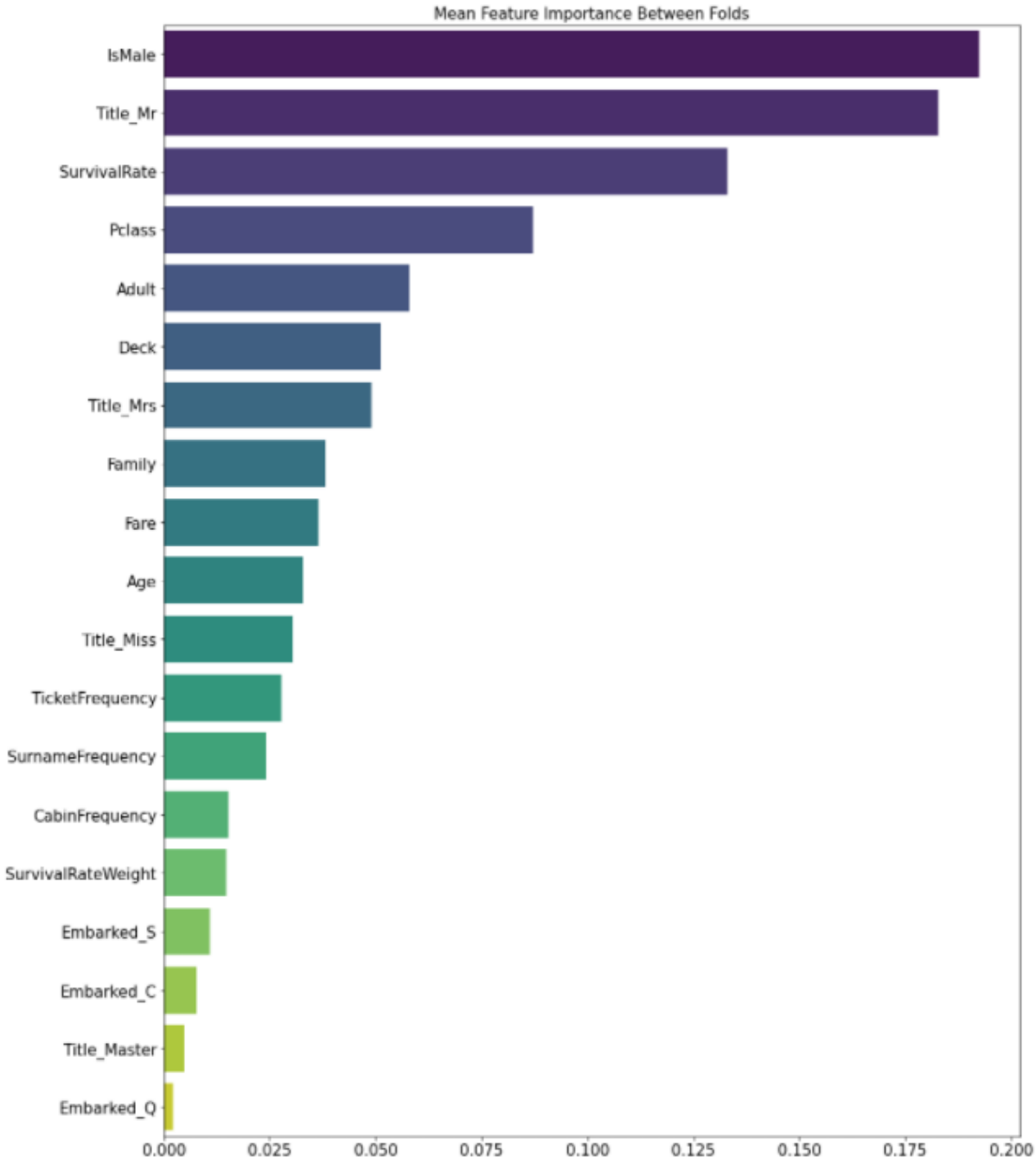
	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Mean_Importance
IsMale	0.180451	0.199432	0.199192	0.204373	0.178849	0.192460
Title_Mr	0.180186	0.186175	0.182614	0.180500	0.184988	0.182893
SurvivalRate	0.139599	0.123680	0.118932	0.147083	0.135926	0.133044
Pclass	0.090327	0.096433	0.087320	0.070453	0.091244	0.087156
Adult	0.053738	0.055792	0.057202	0.063461	0.059996	0.058038
Deck	0.046232	0.053811	0.051538	0.049529	0.055465	0.051315
Title_Mrs	0.050256	0.054957	0.055780	0.043222	0.041479	0.049139
Family	0.046918	0.038881	0.037228	0.039082	0.028617	0.038145
Fare	0.041901	0.031202	0.031011	0.028513	0.050282	0.036582
Age	0.029204	0.034228	0.033577	0.032659	0.034455	0.032825
Title_Miss	0.026017	0.023870	0.025004	0.039219	0.037568	0.030336
TicketFrequency	0.033111	0.026178	0.033032	0.025139	0.021619	0.027816
SurnameFrequency	0.024949	0.025607	0.027962	0.023971	0.018673	0.024233
CabinFrequency	0.013388	0.018718	0.017917	0.013166	0.013479	0.015334
SurvivalRateWeight	0.016265	0.014955	0.012523	0.016029	0.014339	0.014822
Embarked_S	0.011185	0.006392	0.011271	0.011368	0.014644	0.010972
Embarked_C	0.006954	0.006299	0.010163	0.005269	0.010040	0.007745
Title_Master	0.005655	0.002171	0.007011	0.004344	0.005580	0.004952
Embarked_Q	0.003662	0.001220	0.000722	0.002619	0.002756	0.002196

Slika 44 Importance tablica #2

Prema ažuriranoj importance tablice vidi se da najveći značaj na preživljavanje u fold-ovima imaju faktori:

1. IsMale
2. Title_Mr
3. SurvivalRate
4. PClass
5. Adult

To možemo prikazati i grafički pomoću bar-charta,



Slika 45 Bar chart prikaz importance tablice

```
class_survived = [col for col in probs.columns if col.endswith('Prob_1')]
probs['1'] = probs[class_survived].sum(axis=1) / FOLDS
probs['0'] = probs.drop(columns=class_survived).sum(axis=1) / FOLDS
probs['pred'] = 0
pos = probs[probs['1'] >= 0.5].index
probs.loc[pos, 'pred'] = 1

df_test.reset_index(inplace=True)
df_test['Survived']  = probs['pred'].astype(int)
```

Nova ažurirana probs tablica prikazuje mean (artimetičku sredinu) mogućnosti preživljvanja (stupac „1“) i umiranja (stupac „0“) za svakog putika u df-testu. U stupcu „pred“ u binarnom obliku zapisauje je li putnik preživio (vrijednsot 1) ili poginio (vrijednost 0). I to određuje na način da ako sutpac „1“ ima vrijedsnot veću ili jednaku 0,5 onda se u „pred“ zapisuje 1, u slučaju da ima vrijednost manju od 0,5 u „pred“ se zapisuje 0.

	Fold_1_Prob_0	Fold_1_Prob_1	Fold_2_Prob_0	Fold_2_Prob_1	Fold_3_Prob_0	Fold_3_Prob_1	Fold_4_Prob_0	Fold_4_Prob_1	Fold_5_Prob_0	Fold_5_Prob_1	1	0	pred
0	0.870701	0.129299	0.846910	0.153090	0.837400	0.162600	0.829655	0.170345	0.819723	0.180277	0.159122	0.872702	0
1	0.427679	0.572321	0.386968	0.613032	0.494353	0.505647	0.404498	0.595502	0.460016	0.539984	0.565297	0.547762	1
2	0.920821	0.079179	0.923542	0.076458	0.924826	0.075174	0.908981	0.091019	0.941155	0.058845	0.076135	0.939092	0
3	0.845494	0.154506	0.870035	0.129965	0.903587	0.096413	0.854909	0.145091	0.866122	0.133878	0.131970	0.894424	0
4	0.165572	0.834428	0.140493	0.859507	0.150179	0.849821	0.155049	0.844951	0.191192	0.808808	0.839503	0.328398	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
413	0.879608	0.120392	0.881515	0.118485	0.917413	0.082587	0.866532	0.133468	0.886335	0.113665	0.113719	0.909025	0
414	0.264627	0.735373	0.326865	0.673135	0.292261	0.707739	0.309755	0.690245	0.270235	0.729765	0.707251	0.434199	1
415	0.877680	0.122320	0.902853	0.097147	0.922443	0.077557	0.890370	0.109630	0.892636	0.107364	0.102804	0.917757	0
416	0.879608	0.120392	0.881515	0.118485	0.917413	0.082587	0.866532	0.133468	0.886335	0.113665	0.113719	0.909025	0
417	0.109380	0.890620	0.105434	0.894566	0.081516	0.918484	0.114241	0.885759	0.095005	0.904995	0.898885	0.280892	1

Slika 46 Probs tablica #2

```
_, sub = df_split(df_test.set_index("PassengerId"), 'Survived')
sub.to_csv('submission.csv', header=True, index=True)
```

Predviđanja dodajemo u submission.csv iz kojeg je prikazano prvih i zadnjih 10 redova na slici 47 [Slika 47]. Link za cijeli submission.csv :https://www.kaggle.com/dquadro/titanic-top-1-with-gradient-boost-classifier/data

PassengerId	Survived
892	0
893	1
894	0
895	0
896	1
897	0
898	1
899	0
900	1
901	0
...	
1131	1
1132	1
1133	1
1134	0
1135	0
1136	0
1137	0
1138	1
1139	0
1140	1

Slika 47 Prikaz submision.csv (prvih i zadnjih 10 redova)

9. ZAKLJUČAK

U ovom radu sažeto je objašnjen Gradient Boost kako bi se lakše razumio primjer „Titanic-Top 1% with Gradient Boost Clasifier “ koji je obrađen u detalje. Cilj ovog primjera je bilo predvidjeti koji od putnika prežive, a koji ne prežive potonuće Titanika. Rezultati predviđanja su prikazani u binarnom obliku (1-putnik je preživio, 0-putnik nije preživio).

Gradient Boost je odabrani model jer je u usporedni sa drugim modelima ima najviši mean-om od 0,798672.

Kako bi odredili koje varijable imaju najvećeg utjecaja na preživljavanje u test setu koristili smo metodu K-folds i varijable koje najviše utječu na preživljavanje su:

1. IsMale
2. Title_Mr
3. SurvivalRate
4. PClass
5. Adult

Prema tim varijablama vidi se da najveći utjecaj imaju rod (IsMale i Title_Mr), u kojoj generaciju pripada putnik (Adult) , socio-ekonomski status (PClass) i nalaze li se na brodu osobe s kojima se dijeli prezime, kabina ili karta (SurvivalRate).

LITERATURA

[1] Predicting_Survival_on_Titanic_by_Applying_Exploratory_Data_Analytics_and_Machine_Learning_Techniques/links/5c068f63a6fdcc315f9c0bb9/Predicting-Survival-on-Titanic-by-Applying-Exploratory-Data-Analytics-and-Machine-Learning-Techniques.pdf , dostupno dana 17.veljače.2022.

[2] Titanic Machine Learning Study from Disaster , dostupno dana 17.veljače.2022.

[3] Who Survived Titanic? A Logistic Regresion Analysis , dostupno dana 17.veljače.2022.

[4] titanic-advanced-feature-engineering-tutorial , dostupno dana 17.veljače.2022.

[5] https://rmstitanic456.wordpress.com/maiden-voyage/sinking/, dostupno dana 17.veljače.2022.

[6] encyclopedia-titanica , dostupno dana 17.veljače.2022.

[7] https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.31.1666&rep=rep1&type=pdf , dostupno dana 17.veljače.2022.

[8] Gradient boosting Wikipedia , dostupno dana 17.veljače. 2022.

[9] A_Comparative_Study_on_Machine_Learning_Techniques_Using_Titanic_Dataset, dostupno dana 17. veljače 2022.

[10] https://www.kaggle.com/dquadro/titanic-top-1-with-gradient-boost-classifier/notebook , dostupno dana 17.veljače 2022.

[11] https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d , dostupno dana 17.veljače 2022.

[12] https://blog.paperspace.com/gradient-boosting-for-classification/ , dostupno dana 17.veljače 2022.

[13] Korelacija Wikipedia , dostupno dana 17.veljače 2022.

[14] A_Comparative_Study_on_Machine_Learning_Techniques_Using_Titanic_Dataset , dostupno dana 17.veljače 2022.

[15] https://towardsdatascience.com/an-introduction-to-discretization-in-data-science-55ef8c9775a2 , dostupno dana 17. veljače 2022.

[16] https://www.askpython.com/python/examples/one-hot-encoding , dostupno dana 17.veljače 2022.

[17] https://www.statology.org/one-hot-encoding-in-python/ , dostupno dana 17. veljače 2022.

[18] https://scikit-learn.org/stable/auto_examples/preprocessing/plot_all_scaling.html#robustscaler , dostupno dana 17.veljače 2022.

[19] https://machinelearningmastery.com/robust-scaler-transforms-for-machine-learning/ , dostupno dana 17.veljače 2022.

[20] https://www.pythonforengineers.com/cross-validation-and-model-selection/ , dostupno dana 17.veljače 2022.

[21] https://machinelearningmastery.com/k-fold-cross-validation/, dostupno dana 17.veljače 2022.

[22] https://www.sciencedirect.com/topics/engineering/confusion-matrix , dostupno dana 17.veljače 2022.

[23] https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc , dostupno dana 17.veljače 2022.

[24] https://developers.google.com/machine-learning/glossary#AUC , dostupno dana 17.veljače 2022.

[25] https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51 , dostupno dana 17.veljače 2022.

[26] https://chartio.com/learn/charts/box-plot-complete-guide/ , dostupno dana 17.veljače 2022.

[27] https://en.wikipedia.org/wiki/Box_plot#Elements , dostupno dana 17.veljače 2022.

[28] https://chartio.com/resources/tutorials/what-is-a-box-plot/, dostupno dana 17.veljače 2022.

[29] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html , dostupno dana 17.veljače 2022.

[30] https://www.kaggle.com/gunesevitan/titanic-advanced-feature-engineering-tutorial , dostupno dana 17.veljače 2022.

[31] https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5 , dostupno dana 17.veljače 2022.

[32] gentle-introduction-gradient-boosting-algorithm-machine-learning , dostupno dana 17.veljače 2022.

[33] Download_diplosmi rad_Dominik Hrastic , dostupno dana 17.veljače 2022.

[34] Download_ Rad_ Jana Juroš, dostupno dana 17.veljače 2022.

[35] https://www.youtube.com/watch?v=9wDoiSo8trc&t=503s&ab_channel=SASUsers , dostupno dana 17.veljače 2022.

[36] https://www.ibm.com/cloud/learn/overfitting , dostupno dana 17.veljače 2022.

[37] RMS_Titanic Wikipedia , dostupno dana 17.veljače 2022.

[38] Stochastic Gradient Boosting, dostupno dana 17.veljače 2022.

[39] https://towardsdatascience.com/visualizing-data-with-pair-plots-in-python-f228cf529166 , dostupno dana 17.veljače 2022.

[40] https://algotrading101.com/learn/train-test-split/ , dostupno dana 17.veljače 2022.

[41] https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85 , dostupno dana 17.veljače 2022

[42] https://www.theanalysisfactor.com/what-is-an-roc-curve/ , dostupno dana 17.veljače 2022

[43] https://medium.com/analytics-vidhya/what-is-roc-curve-1f776103c998 , dostupno dana 17.veljače 2022.

[44] https://www.statology.org/interpret-roc-curve/ , dostupno dana 17.veljače 2022.

[45] https://www.analyticssteps.com/blogs/what-confusion-matrix , dostupno dana 17.veljače 2022.

[46] Box_plot Wikipedia , dostupno dana 17.veljače 2022

[47] https://cdn.fbsbx.com/v/t59.2708-21/274190613_922964188382848_5235660339426979961_n.pdf/diplomski.pdf?_nc_cat=101&ccb=1-5&_nc_sid=0cab14&_nc_eui2=AeFpBO_4Q0gYROV630-QPaIVXvj_nbscG1Fe-P-duxwbUQNk0lsF5WqSaoG5VSncRkpFcAnYlgPT2M3JAiCv-EQj&_nc_ohc=470SR_4AqLEAX8AfBQP&_nc_ht=cdn.fbsbx.com&oh=03_AVIEOUBQkTNNzPYWDasX71-7x4FPpkK1hQ1n-31K6ZMR9A&oe=620FD422&dl=1, dostupno dana 17.veljače 2022

PRILOZI

I. Link za pristup kodu koji je korišten kao primjer u ovom radu: https://www.kaggle.com/dquadro/titanic-top-1-with-gradient-boost-classifier/notebook