In [1]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import OneHotEncoder
#try within same table encoding
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import accuracy_score
```

In [2]:
```
titanictrain = pd.read_csv('train.csv')
```

In [3]:
```
titanictrain.head(10)
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabi
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	Nal
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C8
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	Nal
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C12
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	Nal
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	Nal
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E4
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	Nal
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	Nal
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	Nal

In [4]:
```
titanictest = pd.read_csv('test.csv')
```

In [5]:
```
titanictest.head(10)
```

Out[5]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
5	897	3	Svensson, Mr. Johan Cervin	male	14.0	0	0	7538	9.2250	NaN	S
6	898	3	Connolly, Miss. Kate	female	30.0	0	0	330972	7.6292	NaN	Q
7	899	2	Caldwell, Mr. Albert Francis	male	26.0	1	1	248738	29.0000	NaN	S
8	900	3	Abrahim, Mrs. Joseph (Sophie Halaut Easu)	female	18.0	0	0	2657	7.2292	NaN	C
9	901	3	Davies, Mr. John Samuel	male	21.0	2	0	A/4 48871	24.1500	NaN	S

In [6]:

```
# I will check number of missing data in each attribute in trainset with 891 passer
titanictrain.isnull().sum()
```

Out[6]:
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [7]:
```
titanictest.isnull().sum()
```

Out[7]:
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [8]:
```
#we can see from results that 3 columns have missing data: age, cabin: a lot missi
#i will try to substitute the missing values with the mean of the data for age and
#cabin number can maybe be replaced starting with letter B is some sector of the sh
#in the same cell if there are more cabin names the all start with same letter, we

#i will drop column ticket number because i don't find it as having an infulence to
#inside the name column there are signs like 'the second' as nobel, but in train se
#treat it as a significant impact on surviving chances
#also, I will later remove PassingerID with no influence on surviving
```

In [9]:
```
#i will check on datatypes of columns
titanictrain.dtypes
```

Out[9]:
```
PassengerId      int64
Survived         int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [10]:
```
#titanictrain = titanictrain.drop(columns =['Name', 'Ticket'])

titanictrain = titanictrain.drop(['Name', 'Ticket'],axis=1)

#nisam stavila za minjat originalnu tabelu , axis=! javlja dole gresku ali radi! do
#inplace=True
# Remove two columns name is 'C' and 'D'
```

```
#df.drop(['C', 'D'], axis=1)
#https://www.geeksforgeeks.org/how-to-drop-one-or-multiple-columns-in-pandas-datafr
```

In [11]:
```
titanictrain.head(10)
```

Out[11]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	0	3	male	22.0	1	0	7.2500	NaN	S
1	2	1	1	female	38.0	1	0	71.2833	C85	C
2	3	1	3	female	26.0	0	0	7.9250	NaN	S
3	4	1	1	female	35.0	1	0	53.1000	C123	S
4	5	0	3	male	35.0	0	0	8.0500	NaN	S
5	6	0	3	male	NaN	0	0	8.4583	NaN	Q
6	7	0	1	male	54.0	0	0	51.8625	E46	S
7	8	0	3	male	2.0	3	1	21.0750	NaN	S
8	9	1	3	female	27.0	0	2	11.1333	NaN	S
9	10	1	2	female	14.0	1	0	30.0708	NaN	C

In [12]:
```
titanictrain['CabinSector'] = titanictrain['Cabin'].str.slice(0,1)

titanictrain.head(25)
#i took only the first letter of the cabin to create and add column cabin sector
#this is just to show how i would do that, but if i replace naan with any mode?? or
#afraid to affect on this feature too much, because more around
```

Out[12]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinS
0	1	0	3	male	22.0	1	0	7.2500	NaN	S	
1	2	1	1	female	38.0	1	0	71.2833	C85	C	
2	3	1	3	female	26.0	0	0	7.9250	NaN	S	
3	4	1	1	female	35.0	1	0	53.1000	C123	S	
4	5	0	3	male	35.0	0	0	8.0500	NaN	S	
5	6	0	3	male	NaN	0	0	8.4583	NaN	Q	
6	7	0	1	male	54.0	0	0	51.8625	E46	S	
7	8	0	3	male	2.0	3	1	21.0750	NaN	S	
8	9	1	3	female	27.0	0	2	11.1333	NaN	S	
9	10	1	2	female	14.0	1	0	30.0708	NaN	C	
10	11	1	3	female	4.0	1	1	16.7000	G6	S	
11	12	1	1	female	58.0	0	0	26.5500	C103	S	
12	13	0	3	male	20.0	0	0	8.0500	NaN	S	
13	14	0	3	male	39.0	1	5	31.2750	NaN	S	
14	15	0	3	female	14.0	0	0	7.8542	NaN	S	
15	16	1	2	female	55.0	0	0	16.0000	NaN	S	
16	17	0	3	male	2.0	4	1	29.1250	NaN	Q	
17	18	1	2	male	NaN	0	0	13.0000	NaN	S	
18	19	0	3	female	31.0	1	0	18.0000	NaN	S	
19	20	1	3	female	NaN	0	0	7.2250	NaN	C	
20	21	0	2	male	35.0	0	0	26.0000	NaN	S	
21	22	1	2	male	34.0	0	0	13.0000	D56	S	
22	23	1	3	female	15.0	0	0	8.0292	NaN	Q	
23	24	1	1	male	28.0	0	0	35.5000	A6	S	
24	25	0	3	female	8.0	3	1	21.0750	NaN	S	

In [13]:

```
# i will replace cabinsector column with random values of sectors i notices in tra
#USE cABINsECTOR COLUMN INSTEAD OF CABIN OR REPLACE VALUES OF CABIN WITH ACBINSECT
np.random.seed(123)
titanictrain.loc[titanictrain['CabinSector'].isna(), 'CabinSector'] = [i for i in
```

In [14]:

```
titanictrain.head(25)
```

Out[14]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinS
0	1	0	3	male	22.0	1	0	7.2500	NaN	S	
1	2	1	1	female	38.0	1	0	71.2833	C85	C	
2	3	1	3	female	26.0	0	0	7.9250	NaN	S	
3	4	1	1	female	35.0	1	0	53.1000	C123	S	
4	5	0	3	male	35.0	0	0	8.0500	NaN	S	
5	6	0	3	male	NaN	0	0	8.4583	NaN	Q	
6	7	0	1	male	54.0	0	0	51.8625	E46	S	
7	8	0	3	male	2.0	3	1	21.0750	NaN	S	
8	9	1	3	female	27.0	0	2	11.1333	NaN	S	
9	10	1	2	female	14.0	1	0	30.0708	NaN	C	
10	11	1	3	female	4.0	1	1	16.7000	G6	S	
11	12	1	1	female	58.0	0	0	26.5500	C103	S	
12	13	0	3	male	20.0	0	0	8.0500	NaN	S	
13	14	0	3	male	39.0	1	5	31.2750	NaN	S	
14	15	0	3	female	14.0	0	0	7.8542	NaN	S	
15	16	1	2	female	55.0	0	0	16.0000	NaN	S	
16	17	0	3	male	2.0	4	1	29.1250	NaN	Q	
17	18	1	2	male	NaN	0	0	13.0000	NaN	S	
18	19	0	3	female	31.0	1	0	18.0000	NaN	S	
19	20	1	3	female	NaN	0	0	7.2250	NaN	C	
20	21	0	2	male	35.0	0	0	26.0000	NaN	S	
21	22	1	2	male	34.0	0	0	13.0000	D56	S	
22	23	1	3	female	15.0	0	0	8.0292	NaN	Q	
23	24	1	1	male	28.0	0	0	35.5000	A6	S	
24	25	0	3	female	8.0	3	1	21.0750	NaN	S	

In [15]:
```
len(titanictrain['Cabin'])
```

Out[15]:
```
891
```

In [16]:
```
len(titanictrain['CabinSector'])
```

Out[16]:
```
891
```

In [17]:
```
#copies column CabinSector into new column Cabin

titanictrain.loc[:,'Cabin'] = (titanictrain['CabinSector'])
```

In [18]:
```
titanictrain.head(25)
```

Out[18]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinS
0	1	0	3	male	22.0	1	0	7.2500	G	S	
1	2	1	1	female	38.0	1	0	71.2833	C	C	
2	3	1	3	female	26.0	0	0	7.9250	F	S	
3	4	1	1	female	35.0	1	0	53.1000	C	S	
4	5	0	3	male	35.0	0	0	8.0500	G	S	
5	6	0	3	male	NaN	0	0	8.4583	C	Q	
6	7	0	1	male	54.0	0	0	51.8625	E	S	
7	8	0	3	male	2.0	3	1	21.0750	E	S	
8	9	1	3	female	27.0	0	2	11.1333	C	S	
9	10	1	2	female	14.0	1	0	30.0708	G	C	
10	11	1	3	female	4.0	1	1	16.7000	G	S	
11	12	1	1	female	58.0	0	0	26.5500	C	S	
12	13	0	3	male	20.0	0	0	8.0500	B	S	
13	14	0	3	male	39.0	1	5	31.2750	D	S	
14	15	0	3	female	14.0	0	0	7.8542	C	S	
15	16	1	2	female	55.0	0	0	16.0000	D	S	
16	17	0	3	male	2.0	4	1	29.1250	B	Q	
17	18	1	2	male	NaN	0	0	13.0000	G	S	
18	19	0	3	female	31.0	1	0	18.0000	B	S	
19	20	1	3	female	NaN	0	0	7.2250	A	C	
20	21	0	2	male	35.0	0	0	26.0000	B	S	
21	22	1	2	male	34.0	0	0	13.0000	D	S	
22	23	1	3	female	15.0	0	0	8.0292	G	Q	
23	24	1	1	male	28.0	0	0	35.5000	A	S	
24	25	0	3	female	8.0	3	1	21.0750	B	S	

In [19]:
```
#**********
#cabinsector nan solved with filling with random values and i will fill age and far
#and embarked with mode because of letter
```

In [20]:
```
medianagetrain = titanictrain['Age'].astype('float64').median(axis=0)
titanictrain['Age'].replace(np.nan, medianagetrain, inplace=True)
```

In [21]:
```
medianfaretrain = titanictrain['Fare'].astype('float64').median(axis=0)
titanictrain['Fare'].replace(np.nan, medianfaretrain, inplace=True)
```

In [22]:
```
modeembarkedtrain = titanictrain['Embarked'].astype('string').mode
titanictrain['Embarked'].replace(np.nan, modeembarkedtrain, inplace=True)
```

In [23]:
```
titanictrain.isnull().sum()
```

Out[23]:
```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Cabin          0
Embarked       0
CabinSector    0
dtype: int64
```

In [24]:
```
titanictrain.head(10)
```

Out[24]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSec
0	1	0	3	male	22.0	1	0	7.2500	G	S	
1	2	1	1	female	38.0	1	0	71.2833	C	C	
2	3	1	3	female	26.0	0	0	7.9250	F	S	
3	4	1	1	female	35.0	1	0	53.1000	C	S	
4	5	0	3	male	35.0	0	0	8.0500	G	S	
5	6	0	3	male	28.0	0	0	8.4583	C	Q	
6	7	0	1	male	54.0	0	0	51.8625	E	S	
7	8	0	3	male	2.0	3	1	21.0750	E	S	
8	9	1	3	female	27.0	0	2	11.1333	C	S	
9	10	1	2	female	14.0	1	0	30.0708	G	C	

In [25]:
```
titanictrainohe = pd.get_dummies(data=titanictrain, columns=['Sex','Cabin','Embark
#drop_first da 1 kolona,tj. 0 i 1, male i ako nije onda female ka ako odbacis prvu
#nastanu 2 kolone
#crated new table with encoded columns instead of exsisting, but i could now drop
#passinger id and CabinSector
```

In [26]:
```
titanictrainohe.head(5)
```

Out[26]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	CabinSector	Sex_female	Sex_male
0	1	0	3	22.0	1	0	7.2500	G	0	1
1	2	1	1	38.0	1	0	71.2833	C	1	0
2	3	1	3	26.0	0	0	7.9250	F	1	0
3	4	1	1	35.0	1	0	53.1000	C	1	0
4	5	0	3	35.0	0	0	8.0500	G	0	1

5 rows × 22 columns

In [27]:
```
titanictrain.head(5)
#titanictrain dataset stays the same after onehotencoding
```

Out[27]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSec
0	1	0	3	male	22.0	1	0	7.2500	G	S	
1	2	1	1	female	38.0	1	0	71.2833	C	C	
2	3	1	3	female	26.0	0	0	7.9250	F	S	
3	4	1	1	female	35.0	1	0	53.1000	C	S	
4	5	0	3	male	35.0	0	0	8.0500	G	S	

In [28]:
```
#**************
#there was after encoding en error saying embarked has to be string or numeric,med
#different types so i turned whole column in string
titanictrain = titanictrain.astype({'Embarked': str})
```

In [29]:
```
#another way
encodecols= ['Sex', 'Cabin', 'Embarked']

for col in encodecols:
    titanictrain[col]=le.fit_transform(titanictrain[col])
    print(le.classes_)

titanictrain.head(5)
```

```
['female' 'male']
['A' 'B' 'C' 'D' 'E' 'F' 'G' 'T']
['<bound method Series.mode of 0      S\n1      C\n2      S\n3      S\n4      S\n
..\n886    S\n887    S\n888    S\n889    C\n890    Q\nName: Embarked, Length: 891,
dtype: string>'
 'C' 'Q' 'S']
```

Out[29]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSecto
0	1	0	3	1	22.0	1	0	7.2500	6	3	C
1	2	1	1	0	38.0	1	0	71.2833	2	1	C
2	3	1	3	0	26.0	0	0	7.9250	5	3	I
3	4	1	1	0	35.0	1	0	53.1000	2	3	C
4	5	0	3	1	35.0	0	0	8.0500	6	3	C

In [30]:
```
titanictrain = titanictrain.drop(['CabinSector'],axis=1)
```

In [31]:
```
titanictrain.head(10)
```

Out[31]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	1	0	3	1	22.0	1	0	7.2500	6	3
1	2	1	1	0	38.0	1	0	71.2833	2	1
2	3	1	3	0	26.0	0	0	7.9250	5	3
3	4	1	1	0	35.0	1	0	53.1000	2	3
4	5	0	3	1	35.0	0	0	8.0500	6	3
5	6	0	3	1	28.0	0	0	8.4583	2	2
6	7	0	1	1	54.0	0	0	51.8625	4	3
7	8	0	3	1	2.0	3	1	21.0750	4	3
8	9	1	3	0	27.0	0	2	11.1333	2	3
9	10	1	2	0	14.0	1	0	30.0708	6	1

In [32]:
```
titanictrain = titanictrain.drop(['PassengerId'],axis=1)
```

In [33]:
```
titanictrain.head(10)
```

Out[33]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	1	22.0	1	0	7.2500	6	3
1	1	1	0	38.0	1	0	71.2833	2	1
2	1	3	0	26.0	0	0	7.9250	5	3
3	1	1	0	35.0	1	0	53.1000	2	3
4	0	3	1	35.0	0	0	8.0500	6	3
5	0	3	1	28.0	0	0	8.4583	2	2
6	0	1	1	54.0	0	0	51.8625	4	3
7	0	3	1	2.0	3	1	21.0750	4	3
8	1	3	0	27.0	0	2	11.1333	2	3
9	1	2	0	14.0	1	0	30.0708	6	1

In [34]:
```
#  MinMaxScaler za age and fare
mms = MinMaxScaler()
```

In [35]:
```
# fit scaler on training data
titanictrain[['Age','Fare']] = mms.fit_transform(titanictrain[['Age','Fare']])
```

In [36]:
```
# fit & transform training data za age i fare
```

In [37]:
```
titanictrain.head(10)
```

Out[37]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	1	0.271174	1	0	0.014151	6	3
1	1	1	0	0.472229	1	0	0.139136	2	1
2	1	3	0	0.321438	0	0	0.015469	5	3
3	1	1	0	0.434531	1	0	0.103644	2	3
4	0	3	1	0.434531	0	0	0.015713	6	3
5	0	3	1	0.346569	0	0	0.016510	2	2
6	0	1	1	0.673285	0	0	0.101229	4	3
7	0	3	1	0.019854	3	1	0.041136	4	3
8	1	3	0	0.334004	0	2	0.021731	2	3
9	1	2	0	0.170646	1	0	0.058694	6	1

In [38]:
```
#titanictrain['Embarked'].values
#za provjerit je li dobro pretvorio, je
```

In [39]:
```
#titanictest[['Age','Fare']] = mms.transform(titanictest[['Age','Fare']])
#****
#probala i tocno, za titanictest samo transform koristis, jer se mms objekt primije
```

In [40]:
```
#******TEST**********
```

In [41]:
```
#i will check on datatypes of columns
titanictest.dtypes
```

Out[41]:
```
PassengerId      int64
Pclass           int64
Name            object
Sex             object
Age            float64
SibSp            int64
Parch            int64
Ticket          object
Fare           float64
Cabin           object
Embarked        object
dtype: object
```

In [42]:
```
#titanictrain = titanictrain.drop(columns =['Name', 'Ticket'])
titanictest = titanictest.drop(['Name', 'Ticket'],axis=1)

#nisam stavila za minjat originalnu tabelu , axis=! javlja dole gresku ali radi! do
#inplace=True
# Remove two columns name is 'C' and 'D'
```

```
#df.drop(['C', 'D'], axis=1)
#https://www.geeksforgeeks.org/how-to-drop-one-or-multiple-columns-in-pandas-datafr
```

In [43]:
```
titanictest.head(15)
```

Out[43]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	892	3	male	34.5	0	0	7.8292	NaN	Q
1	893	3	female	47.0	1	0	7.0000	NaN	S
2	894	2	male	62.0	0	0	9.6875	NaN	Q
3	895	3	male	27.0	0	0	8.6625	NaN	S
4	896	3	female	22.0	1	1	12.2875	NaN	S
5	897	3	male	14.0	0	0	9.2250	NaN	S
6	898	3	female	30.0	0	0	7.6292	NaN	Q
7	899	2	male	26.0	1	1	29.0000	NaN	S
8	900	3	female	18.0	0	0	7.2292	NaN	C
9	901	3	male	21.0	2	0	24.1500	NaN	S
10	902	3	male	NaN	0	0	7.8958	NaN	S
11	903	1	male	46.0	0	0	26.0000	NaN	S
12	904	1	female	23.0	1	0	82.2667	B45	S
13	905	2	male	63.0	1	0	26.0000	NaN	S
14	906	1	female	47.0	1	0	61.1750	E31	S

In [44]:
```
titanictest['CabinSector'] = titanictest['Cabin'].str.slice(0,1)

titanictest.head(15)
#i took only the first letter of the cabin to create and add column cabin sector
#this is just to show how i would do that, but if i replace naan with any mode?? or
#afraid to affect on this feature too much, because more around
```

Out[44]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSector
0	892	3	male	34.5	0	0	7.8292	NaN	Q	NaN
1	893	3	female	47.0	1	0	7.0000	NaN	S	NaN
2	894	2	male	62.0	0	0	9.6875	NaN	Q	NaN
3	895	3	male	27.0	0	0	8.6625	NaN	S	NaN
4	896	3	female	22.0	1	1	12.2875	NaN	S	NaN
5	897	3	male	14.0	0	0	9.2250	NaN	S	NaN
6	898	3	female	30.0	0	0	7.6292	NaN	Q	NaN
7	899	2	male	26.0	1	1	29.0000	NaN	S	NaN
8	900	3	female	18.0	0	0	7.2292	NaN	C	NaN
9	901	3	male	21.0	2	0	24.1500	NaN	S	NaN
10	902	3	male	NaN	0	0	7.8958	NaN	S	NaN
11	903	1	male	46.0	0	0	26.0000	NaN	S	NaN
12	904	1	female	23.0	1	0	82.2667	B45	S	B
13	905	2	male	63.0	1	0	26.0000	NaN	S	NaN
14	906	1	female	47.0	1	0	61.1750	E31	S	E

In [45]:

```
# i will replace cabinsector column with random values of sectors i notices in tra
#USE cABINsECTOR COLUMN INSTEAD OF CABIN OR REPLACE VALUES OF CABIN WITH ACBINSECT
np.random.seed(123)
titanictest.loc[titanictest['CabinSector'].isna(), 'CabinSector'] = [i for i in np
```

In [46]:

```
titanictest.head(25)
```

Out[46]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSector
0	892	3	male	34.5	0	0	7.8292	NaN	Q	G
1	893	3	female	47.0	1	0	7.0000	NaN	S	F
2	894	2	male	62.0	0	0	9.6875	NaN	Q	G
3	895	3	male	27.0	0	0	8.6625	NaN	S	C
4	896	3	female	22.0	1	1	12.2875	NaN	S	E
5	897	3	male	14.0	0	0	9.2250	NaN	S	C
6	898	3	female	30.0	0	0	7.6292	NaN	Q	G
7	899	2	male	26.0	1	1	29.0000	NaN	S	B
8	900	3	female	18.0	0	0	7.2292	NaN	C	D
9	901	3	male	21.0	2	0	24.1500	NaN	S	C
10	902	3	male	NaN	0	0	7.8958	NaN	S	D
11	903	1	male	46.0	0	0	26.0000	NaN	S	B
12	904	1	female	23.0	1	0	82.2667	B45	S	B
13	905	2	male	63.0	1	0	26.0000	NaN	S	G
14	906	1	female	47.0	1	0	61.1750	E31	S	E
15	907	2	female	24.0	1	0	27.7208	NaN	C	B
16	908	2	male	35.0	0	0	12.3500	NaN	Q	A
17	909	3	male	21.0	0	0	7.2250	NaN	C	B
18	910	3	female	27.0	1	0	7.9250	NaN	S	G
19	911	3	female	45.0	0	0	7.2250	NaN	C	B
20	912	1	male	55.0	1	0	59.4000	NaN	C	A
21	913	3	male	9.0	0	1	3.1708	NaN	S	G
22	914	1	female	NaN	0	0	31.6833	NaN	S	A
23	915	1	male	21.0	0	1	61.3792	NaN	C	B
24	916	1	female	48.0	1	3	262.3750	B57 B59 B63 B66	C	B

In [47]:
```
len(titanictest['Cabin'])
```

Out[47]: 418

In [48]:
```
len(titanictest['CabinSector'])
```

Out[48]: 418

In [49]:
```
#copies column CabinSector into new column Cabin

titanictest.loc[:,'Cabin'] = (titanictest['CabinSector'])
```

In [50]:
```
titanictest.head(15)
```

Out[50]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSector
0	892	3	male	34.5	0	0	7.8292	G	Q	G
1	893	3	female	47.0	1	0	7.0000	F	S	F
2	894	2	male	62.0	0	0	9.6875	G	Q	G
3	895	3	male	27.0	0	0	8.6625	C	S	C
4	896	3	female	22.0	1	1	12.2875	E	S	E
5	897	3	male	14.0	0	0	9.2250	C	S	C
6	898	3	female	30.0	0	0	7.6292	G	Q	G
7	899	2	male	26.0	1	1	29.0000	B	S	B
8	900	3	female	18.0	0	0	7.2292	D	C	D
9	901	3	male	21.0	2	0	24.1500	C	S	C
10	902	3	male	NaN	0	0	7.8958	D	S	D
11	903	1	male	46.0	0	0	26.0000	B	S	B
12	904	1	female	23.0	1	0	82.2667	B	S	B
13	905	2	male	63.0	1	0	26.0000	G	S	G
14	906	1	female	47.0	1	0	61.1750	E	S	E

In [51]:

```
print(titanictest)
```

```
     PassengerId  Pclass     Sex   Age  SibSp  Parch      Fare Cabin Embarked  \
0            892       3    male  34.5      0      0    7.8292     G        Q
1            893       3  female  47.0      1      0    7.0000     F        S
2            894       2    male  62.0      0      0    9.6875     G        Q
3            895       3    male  27.0      0      0    8.6625     C        S
4            896       3  female  22.0      1      1   12.2875     E        S
..           ...     ...     ...   ...    ...    ...       ...   ...      ...
413         1305       3    male   NaN      0      0    8.0500     G        S
414         1306       1  female  39.0      0      0  108.9000     C        C
415         1307       3    male  38.5      0      0    7.2500     A        S
416         1308       3    male   NaN      0      0    8.0500     G        S
417         1309       3    male   NaN      1      1   22.3583     B        C

    CabinSector
0             G
1             F
2             G
3             C
4             E
..          ...
413           G
414           C
415           A
416           G
417           B

[418 rows x 10 columns]
```

In [52]:

```
 #**********
#cabinsector nan solved with filling with random values and i will fill age and far
#and embarked with mode because of letter
```

In [53]:
```
medianagetest = titanictest['Age'].astype('float64').median(axis=0)
titanictest['Age'].replace(np.nan, medianagetest, inplace=True)
```

In [54]:
```
medianfaretest = titanictest['Fare'].astype('float64').median(axis=0)
titanictest['Fare'].replace(np.nan, medianfaretest, inplace=True)
```

In [55]:
```
modeembarkedtest = titanictest['Embarked'].astype('string').mode
titanictest['Embarked'].replace(np.nan, modeembarkedtest, inplace=True)
```

In [56]:
```
titanictest.isnull().sum()
```

Out[56]:
```
PassengerId    0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Cabin          0
Embarked       0
CabinSector    0
dtype: int64
```

In [57]:
```
titanictest.head(10)
```

Out[57]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSector
0	892	3	male	34.5	0	0	7.8292	G	Q	G
1	893	3	female	47.0	1	0	7.0000	F	S	F
2	894	2	male	62.0	0	0	9.6875	G	Q	G
3	895	3	male	27.0	0	0	8.6625	C	S	C
4	896	3	female	22.0	1	1	12.2875	E	S	E
5	897	3	male	14.0	0	0	9.2250	C	S	C
6	898	3	female	30.0	0	0	7.6292	G	Q	G
7	899	2	male	26.0	1	1	29.0000	B	S	B
8	900	3	female	18.0	0	0	7.2292	D	C	D
9	901	3	male	21.0	2	0	24.1500	C	S	C

In [58]:
```
titanictestohe = pd.get_dummies(data=titanictest, columns=['Sex','Cabin','Embarked
#drop_first da 1 kolona,tj. 0 i 1, male i ako nije onda female ka ako odbacis prvu
#nastanu 2 kolone
#one hot encoding
#crated new table with encoded columns instead of exsisting, but i could now drop t
#passinger id and CabinSector
```

In [59]:
```
titanictestohe.head(5)
```

Out[59]:

	PassengerId	Pclass	Age	SibSp	Parch	Fare	CabinSector	Sex_female	Sex_male	Cabin_A
0	892	3	34.5	0	0	7.8292	G	0	1	0
1	893	3	47.0	1	0	7.0000	F	1	0	0
2	894	2	62.0	0	0	9.6875	G	0	1	0
3	895	3	27.0	0	0	8.6625	C	0	1	0
4	896	3	22.0	1	1	12.2875	E	1	0	0

In [60]:

```
titanictest.head(5)
#titanictrain dataset stays the same after onehotencoding
```

Out[60]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSector
0	892	3	male	34.5	0	0	7.8292	G	Q	G
1	893	3	female	47.0	1	0	7.0000	F	S	F
2	894	2	male	62.0	0	0	9.6875	G	Q	G
3	895	3	male	27.0	0	0	8.6625	C	S	C
4	896	3	female	22.0	1	1	12.2875	E	S	E

In [61]:

```
#**************
#there was after encoding en error saying embarked has to be string or numeric,med
#different types so i turned whole column in string
titanictest = titanictest.astype({'Embarked': str})
```

In [62]:

```
#another way
encodecols= ['Sex', 'Cabin', 'Embarked']

for col in encodecols:
    titanictest[col]=le.fit_transform(titanictest[col])
    print(le.classes_)

titanictest.head(5)
```

```
['female' 'male']
['A' 'B' 'C' 'D' 'E' 'F' 'G']
['C' 'Q' 'S']
```

Out[62]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	CabinSector
0	892	3	1	34.5	0	0	7.8292	6	1	G
1	893	3	0	47.0	1	0	7.0000	5	2	F
2	894	2	1	62.0	0	0	9.6875	6	1	G
3	895	3	1	27.0	0	0	8.6625	2	2	C
4	896	3	0	22.0	1	1	12.2875	4	2	E

In [63]:

```
titanictest = titanictest.drop(['CabinSector'],axis=1)
```

In [64]:

```
titanictest.head(10)
```

Out[64]:

	PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	892	3	1	34.5	0	0	7.8292	6	1
1	893	3	0	47.0	1	0	7.0000	5	2
2	894	2	1	62.0	0	0	9.6875	6	1
3	895	3	1	27.0	0	0	8.6625	2	2
4	896	3	0	22.0	1	1	12.2875	4	2
5	897	3	1	14.0	0	0	9.2250	2	2
6	898	3	0	30.0	0	0	7.6292	6	1
7	899	2	1	26.0	1	1	29.0000	1	2
8	900	3	0	18.0	0	0	7.2292	3	0
9	901	3	1	21.0	2	0	24.1500	2	2

In [65]:
```
titanictest = titanictest.drop(['PassengerId'],axis=1)
```

In [66]:
```
titanictest.head(10)
```

Out[66]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	1	34.5	0	0	7.8292	6	1
1	3	0	47.0	1	0	7.0000	5	2
2	2	1	62.0	0	0	9.6875	6	1
3	3	1	27.0	0	0	8.6625	2	2
4	3	0	22.0	1	1	12.2875	4	2
5	3	1	14.0	0	0	9.2250	2	2
6	3	0	30.0	0	0	7.6292	6	1
7	2	1	26.0	1	1	29.0000	1	2
8	3	0	18.0	0	0	7.2292	3	0
9	3	1	21.0	2	0	24.1500	2	2

In [67]:
```
#  MinMaxScaler za age and fare
#mms = MinMaxScaler()
```

In [68]:
```
# fit scaler on training data
#titanictrain[['Age','Fare']] = mms.fit_transform(titanictrain[['Age','Fare']])
titanictest[['Age','Fare']] = mms.transform(titanictest[['Age','Fare']])
```

In [69]:
```
# fit & transform training data za age i fare ali fit po train opet pa onda primjer
#nije radilo do import train_test split
```

In [70]:
```
titanictest.head(10)
```

Out[70]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	3	1	0.428248	0	0	0.015282	6	1
1	3	0	0.585323	1	0	0.013663	5	2
2	2	1	0.773813	0	0	0.018909	6	1
3	3	1	0.334004	0	0	0.016908	2	2
4	3	0	0.271174	1	1	0.023984	4	2
5	3	1	0.170646	0	0	0.018006	2	2
6	3	0	0.371701	0	0	0.014891	6	1
7	2	1	0.321438	1	1	0.056604	1	2
8	3	0	0.220910	0	0	0.014110	3	0
9	3	1	0.258608	2	0	0.047138	2	2

In [71]:

```
#titanictest[['Age','Fare']] = mms.transform(titanictest[['Age','Fare']])
#****
#probala i tocno, za titanictest samo transform koristis, jer se mms objekt primije
```

In [72]:

```
titanictrain.head(10)
```

Out[72]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	1	0.271174	1	0	0.014151	6	3
1	1	1	0	0.472229	1	0	0.139136	2	1
2	1	3	0	0.321438	0	0	0.015469	5	3
3	1	1	0	0.434531	1	0	0.103644	2	3
4	0	3	1	0.434531	0	0	0.015713	6	3
5	0	3	1	0.346569	0	0	0.016510	2	2
6	0	1	1	0.673285	0	0	0.101229	4	3
7	0	3	1	0.019854	3	1	0.041136	4	3
8	1	3	0	0.334004	0	2	0.021731	2	3
9	1	2	0	0.170646	1	0	0.058694	6	1

In [73]:

```
titanicvisual=titanictrain
titanicvisual.head()
```

Out[73]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	1	0.271174	1	0	0.014151	6	3
1	1	1	0	0.472229	1	0	0.139136	2	1
2	1	3	0	0.321438	0	0	0.015469	5	3
3	1	1	0	0.434531	1	0	0.103644	2	3
4	0	3	1	0.434531	0	0	0.015713	6	3

In [74]:

```
#now I will split train data in train and validation data, because I dont have Y i
```

In [75]:
```
#iz analytics vidhya:
#https://www.analyticsvidhya.com/blog/2021/04/sklearn-objects-fit-vs-transform-vs-j
```

In [76]:
```
#prvo Y za izvuci survived kolonu,jer posli kad je izbacis, izbaci se iz svih gorn
#nema mi je odakle izvuc
```

In [77]:
```
#Y=titanictrain['Survived']; svi redovi i samo 1. kolona
Y=titanictrain.iloc[:,0]
#Y = titanictrain['Survived']
print(Y)
```

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [78]:
```
#; svi redovi i sve osim 1. kolone

#X= titanictrain.drop(['Survived'],axis=1)
X=titanictrain.iloc[:,1:]
print(X)
```

```
     Pclass  Sex       Age  SibSp  Parch      Fare  Cabin  Embarked
0         3    1  0.271174      1      0  0.014151      6         3
1         1    0  0.472229      1      0  0.139136      2         1
2         3    0  0.321438      0      0  0.015469      5         3
3         1    0  0.434531      1      0  0.103644      2         3
4         3    1  0.434531      0      0  0.015713      6         3
..      ...  ...       ...    ...    ...       ...    ...       ...
886       2    1  0.334004      0      0  0.025374      0         3
887       1    0  0.233476      0      0  0.058556      1         3
888       3    0  0.346569      1      2  0.045771      0         3
889       1    1  0.321438      0      0  0.058556      2         1
890       3    1  0.396833      0      0  0.015127      5         2

[891 rows x 8 columns]
```

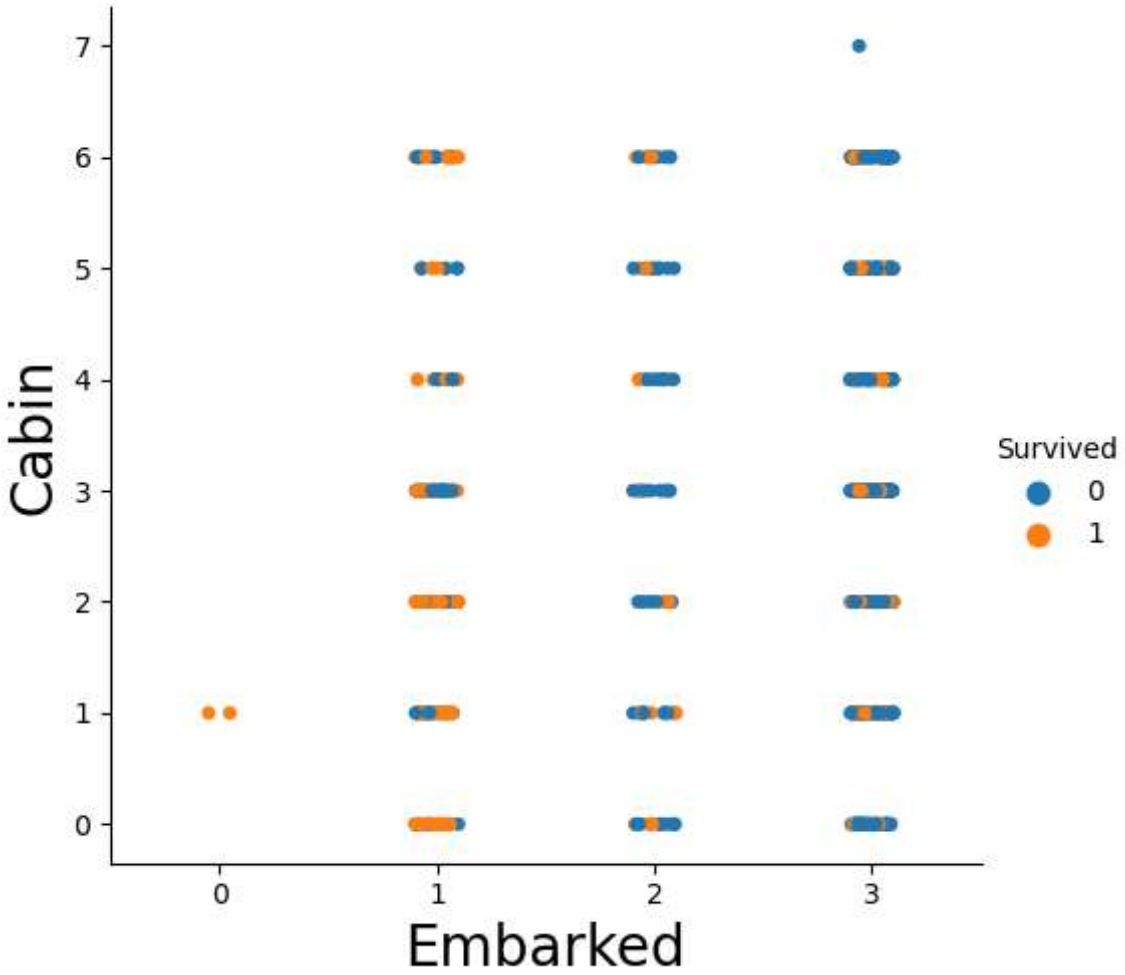
In [79]:
```
sns.catplot(y='Age', x='Sex', hue ='Survived', data = titanicvisual, aspect = 5)
plt.xlabel('Gender', fontsize = 20)
plt.ylabel('Age', fontsize = 20)
plt.pause(0.001)
plt.show
#1 is male 0 is female; 1 survived, 0 for deceased
```

Out[79]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

In [ ]:

In [80]:
```
sns.catplot(y='Cabin', x='Embarked', hue= 'Survived', data = titanicvisual)
plt.xlabel('Embarked', fontsize = 20)
plt.ylabel('Cabin', fontsize = 20)
plt.pause(0.001)
plt.show


# ['C' 'Q' 'S'] for embarked in C = Cherbourg, Q = Queenstown, S = Southampton
#1 survived, 0 for deceased
```



Out[80]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [81]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
#***from sklearn.model_selection import GridSearchCV
#now i will split train data in XTRAIN I XVAL
```

In [82]:
```
X_train, X_val, Y_train, Y_val = train_test_split(X,Y, test_size = 0.2, random_sta
#kao mini cross val
```

In [83]:
```
logreg = LogisticRegression(random_state=0, max_iter=1000)
logreg.fit(X_train, Y_train)
#primjenjuje nauceno iz traina i bez da na varijabli
```

Out[83]:
```
LogisticRegression(max_iter=1000, random_state=0)
```

In [84]:
```
yhatval = logreg.predict(X_val)
```

In [85]:
```
accuracy_score(Y_val, yhatval)
```

Out[85]:
```
0.7932960893854749
```

In [86]:
```
submission = pd.read_csv('gender_submission.csv')
submission.head(5)
```

Out[86]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1

In [87]:
```
yhatsubmission = logreg.predict(titanictest)

#za predaju submission
```

In [88]:
```
#accuracy_score(Y_given, yhatsubmission)

#Y_given radi accuracy pogledaj treba li numpy
#oba numpy

print(type(yhatsubmission))
#Y_val.to_numpy()
print(type(Y_val))
```

```
<class 'numpy.ndarray'>
<class 'pandas.core.series.Series'>
```

In [89]:
```
#za confusion matrix sam provjerila je li gore array i je u oba
def plot_confusion_matrix(Y_val, yhatval):

    cm = confusion_matrix(Y_val, yhatval)
    ax = plt.subplot()
    sns.heatmap(cm, annot=True, ax = ax);
    ax.set_xlabel('Predicted labels')
    ax.set_ylabel('True labels')
    ax.set_title('Confusion Matrix');
    ax.xaxis.set_ticklabels(['did not survive', 'survived'])
    ax.yaxis.set_ticklabels(['did not survive', 'survived'])

plot_confusion_matrix(Y_val, yhatval)
#y_val was made on 0,2*titanictrain (20%*891 na traintestsplit uzela)so it was 179
#it is a sum of 88+17+20+54
```

In [90]:
```
#od 74 prezivjela, predvidjeli smo 54 prezivjela, a 20 smo rekli da nece prezivit
#od 105 umrlih, za 88 smo rekli da nece prezivit i nisu, a za 17 smo rekli da ce pr
#neka sredina tocnosti izmedju te dvije od 0,83 i 0,77 je 0,80 tocnost
#confusion matrix mi iz rezultata y kolone jedinice cita kao true a nule kao false
```

In [91]:
```
#another way for confusion matrix

from sklearn import metrics
metrics.confusion_matrix(Y_val, yhatval)

from sklearn import metrics

#create confusion matrix
c_matrix = metrics.confusion_matrix(Y_val, yhatval)

#print confusion matrix
print(c_matrix)
```

```
[[88 17]
 [20 54]]
```

In [92]:
```
#my_array = np.array([[11,22,33],[44,55,66]])
#df = pd.DataFrame(my_array, columns = ['Column_A','Column_B','Column_C'])
#print(df)
#primjer: iz array u pandas



yhatsubmission = pd.DataFrame(yhatsubmission, columns = ['Yhatsubmission'])

print(yhatsubmission)
print(type(yhatsubmission))
```

```
     Yhatsubmission
0                 0
1                 1
2                 0
3                 0
4                 1
..              ...
413               0
414               1
415               0
416               0
417               0

[418 rows x 1 columns]
<class 'pandas.core.frame.DataFrame'>
```

In [93]:
```
#yhatsubmission_top = yhatsubmission.head()
#yhatsubmission_top
for col in yhatsubmission.columns:
    print(col)
    #ime kolone za svaki slucaj
```

```
Yhatsubmission
```

In [94]:
```
#submission = pd.read_csv('gender_submission.csv')
#ovo je tablica za predaju, isti brojevi putnika kao i u test dijelu excela
```

In [95]:
```
submission= submission.drop(['Survived'],axis=1)
print(submission)
```

```
#on 901 we see that in original submission table person didnt survive and down in r
#on 901 we predicted 901 to survive
```

```
     PassengerId
0            892
1            893
2            894
3            895
4            896
..           ...
413         1305
414         1306
415         1307
416         1308
417         1309

[418 rows x 1 columns]
```

In [96]:

```
extracted_col = yhatsubmission['Yhatsubmission']

display(extracted_col)
```

```
0      0
1      1
2      0
3      0
4      1
      ..
413    0
414    1
415    0
416    0
417    0
Name: Yhatsubmission, Length: 418, dtype: int64
```

In [97]:

```
submission.insert(1, 'Yhatsubmission', extracted_col)

display(submission)
```

	PassengerId	Yhatsubmission
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	0

418 rows × 2 columns

In [98]:
```
#Feature importance shuffle
```

In [99]:
```
#my_array = np.array(X_val)
```

In [100…
```
X_val.head()
#pandas dataframe
```

Out[100]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.346569	1	1	0.029758	6	1
439	2	1	0.384267	0	0	0.020495	3	3
840	3	1	0.246042	0	0	0.015469	2	3
720	2	0	0.070118	0	1	0.064412	2	3
39	3	0	0.170646	1	0	0.021942	3	1

In [101…
```
print(type(X_val))
#Y_val je array(pandas core series provjerila je array)
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [102…
```
print(type(Y_val))
```

```
<class 'pandas.core.series.Series'>
```

In [103…
```
#submission.to_csv('TitanicpredajafeatKate2023.csv')
```

In [104…
```
#***first column permutation
X_val1 = X_val.copy()
#panda for permutation
```

In [105…
```
X_val1.head()
```

Out[105]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.346569	1	1	0.029758	6	1
439	2	1	0.384267	0	0	0.020495	3	3
840	3	1	0.246042	0	0	0.015469	2	3
720	2	0	0.070118	0	1	0.064412	2	3
39	3	0	0.170646	1	0	0.021942	3	1

In [106…
```
X_val1['Pclass'] = np.random.permutation(X_val1['Pclass'])
```

In [107…
```
X_val1.head()
#i see permutation in the first column
```

Out[107]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.346569	1	1	0.029758	6	1
439	1	1	0.384267	0	0	0.020495	3	3
840	1	1	0.246042	0	0	0.015469	2	3
720	1	0	0.070118	0	1	0.064412	2	3
39	1	0	0.170646	1	0	0.021942	3	1

In [108…

```
X_val1.to_numpy()
```

Out[108]:

```
array([[3.        , 1.        , 0.34656949, ..., 0.02975782, 6.        ,
        1.        ],
       [1.        , 1.        , 0.3842674 , ..., 0.02049464, 3.        ,
        3.        ],
       [1.        , 1.        , 0.24604172, ..., 0.01546857, 2.        ,
        3.        ],
       ...,
       [2.        , 0.        , 0.4722292 , ..., 0.06126432, 0.        ,
        3.        ],
       [2.        , 0.        , 0.2083438 , ..., 0.02049464, 4.        ,
        3.        ],
       [3.        , 0.        , 0.04498618, ..., 0.03259623, 6.        ,
        3.        ]])
```

In [109…

```
yX_val1 = logreg.predict(X_val1)
```

In [110…

```
a1 = accuracy_score(Y_val, yX_val1)
```

In [111…

```
#importancePclass= accuracy score X_Val-accuracy score X_val1
importancePclass = 0.7932960893854749-a1
print (importancePclass)
```

```
0.022346368715083886
```

In [112…

```
#***second column permutation
X_val2 = X_val.copy()
X_val2.head()
#panda for permutation
X_val2['Sex'] = np.random.permutation(X_val2['Sex'])
X_val2.head()
```

Out[112]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	0	0.346569	1	1	0.029758	6	1
439	2	0	0.384267	0	0	0.020495	3	3
840	3	1	0.246042	0	0	0.015469	2	3
720	2	0	0.070118	0	1	0.064412	2	3
39	3	1	0.170646	1	0	0.021942	3	1

In [113…

```
X_val2.to_numpy()
```

Out[113]:
```
array([[3.        , 0.        , 0.34656949, ..., 0.02975782, 6.        ,
        1.        ],
       [2.        , 0.        , 0.3842674 , ..., 0.02049464, 3.        ,
        3.        ],
       [3.        , 1.        , 0.24604172, ..., 0.01546857, 2.        ,
        3.        ],
       ...,
       [3.        , 1.        , 0.4722292 , ..., 0.06126432, 0.        ,
        3.        ],
       [2.        , 0.        , 0.2083438 , ..., 0.02049464, 4.        ,
        3.        ],
       [3.        , 1.        , 0.04498618, ..., 0.03259623, 6.        ,
        3.        ]])
```

In [114…
```
yX_val2 = logreg.predict(X_val2)
```

In [115…
```
a2 = accuracy_score(Y_val, yX_val2)
```

In [116…
```
#importancePclass= accuracy score X_Val-accuracy score X_val2
importanceSex = 0.7932960893854749-a2
print (importanceSex)
```

```
0.2234636871508381
```

In [117…
```
#***third column permutation
X_val3 = X_val.copy()
X_val3.head()
#panda for permutation
X_val3['Age'] = np.random.permutation(X_val3['Age'])
X_val3.head()
```

Out[117]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.195778	1	1	0.029758	6	1
439	2	1	0.824076	0	0	0.020495	3	3
840	3	1	0.346569	0	0	0.015469	2	3
720	2	0	0.246042	0	1	0.064412	2	3
39	3	0	0.057552	1	0	0.021942	3	1

In [118…
```
X_val3.to_numpy()
```

Out[118]:
```
array([[3.        , 1.        , 0.19577783, ..., 0.02975782, 6.        ,
        1.        ],
       [2.        , 1.        , 0.8240764 , ..., 0.02049464, 3.        ,
        3.        ],
       [3.        , 1.        , 0.34656949, ..., 0.01546857, 2.        ,
        3.        ],
       ...,
       [3.        , 0.        , 0.34656949, ..., 0.06126432, 0.        ,
        3.        ],
       [2.        , 0.        , 0.33400352, ..., 0.02049464, 4.        ,
        3.        ],
       [3.        , 0.        , 0.3842674 , ..., 0.03259623, 6.        ,
        3.        ]])
```

In [119…
```
yX_val3 = logreg.predict(X_val3)
```

In [120…
```
a3 = accuracy_score(Y_val, yX_val3)
```

In [121…
```
#importancePclass= accuracy score X_Val-accuracy score X_val3
importanceAge = 0.7932960893854749-a3
print (importanceAge)
```

```
0.011173184357541999
```

In [122…
```
#***fourth column permutation
X_val4 = X_val.copy()
X_val4.head()
#panda for permutation
X_val4['SibSp'] = np.random.permutation(X_val4['SibSp'])
X_val4.head()
```

Out[122]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.346569	1	1	0.029758	6	1
439	2	1	0.384267	0	0	0.020495	3	3
840	3	1	0.246042	1	0	0.015469	2	3
720	2	0	0.070118	0	1	0.064412	2	3
39	3	0	0.170646	0	0	0.021942	3	1

In [123…
```
X_val4.to_numpy()
```

Out[123]:
```
array([[3.        , 1.        , 0.34656949, ..., 0.02975782, 6.        ,
        1.        ],
       [2.        , 1.        , 0.3842674 , ..., 0.02049464, 3.        ,
        3.        ],
       [3.        , 1.        , 0.24604172, ..., 0.01546857, 2.        ,
        3.        ],
       ...,
       [3.        , 0.        , 0.4722292 , ..., 0.06126432, 0.        ,
        3.        ],
       [2.        , 0.        , 0.2083438 , ..., 0.02049464, 4.        ,
        3.        ],
       [3.        , 0.        , 0.04498618, ..., 0.03259623, 6.        ,
        3.        ]])
```

In [124…
```
yX_val4 = logreg.predict(X_val4)
```

In [125…
```
a4 = accuracy_score(Y_val, yX_val4)
```

In [126…
```
#importancePclass= accuracy score X_Val-accuracy score X_val4
importanceSibSp = 0.7932960893854749-a4
print (importanceSibSp)
```

```
0.022346368715083886
```

In [127…
```
#***fifth column permutation
X_val5 = X_val.copy()
X_val5.head()
#panda for permutation
X_val5['Parch'] = np.random.permutation(X_val5['Parch'])
X_val5.head()
```

Out[127]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.346569	1	0	0.029758	6	1
439	2	1	0.384267	0	1	0.020495	3	3
840	3	1	0.246042	0	0	0.015469	2	3
720	2	0	0.070118	0	0	0.064412	2	3
39	3	0	0.170646	1	0	0.021942	3	1

In [128…

```
X_val5.to_numpy()
```

Out[128]:

```
array([[3.        , 1.        , 0.34656949, ..., 0.02975782, 6.        ,
        1.        ],
       [2.        , 1.        , 0.3842674 , ..., 0.02049464, 3.        ,
        3.        ],
       [3.        , 1.        , 0.24604172, ..., 0.01546857, 2.        ,
        3.        ],
       ...,
       [3.        , 0.        , 0.4722292 , ..., 0.06126432, 0.        ,
        3.        ],
       [2.        , 0.        , 0.2083438 , ..., 0.02049464, 4.        ,
        3.        ],
       [3.        , 0.        , 0.04498618, ..., 0.03259623, 6.        ,
        3.        ]])
```

In [129…

```
yX_val5 = logreg.predict(X_val5)
```

In [130…

```
a5 = accuracy_score(Y_val, yX_val5)
```

In [131…

```
#importancePclass= accuracy score X_Val-accuracy score X_val5
importanceParch = 0.7932960893854749-a5
print (importanceParch)
```

```
-0.005586592178770888
```

In [132…

```
#***sixth column permutation
X_val6 = X_val.copy()
X_val6.head()
#panda for permutation
X_val6['Fare'] = np.random.permutation(X_val6['Fare'])
X_val6.head()
```

Out[132]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.346569	1	1	0.020495	6	1
439	2	1	0.384267	0	0	0.040062	3	3
840	3	1	0.246042	0	0	0.015412	2	3
720	2	0	0.070118	0	1	0.014737	2	3
39	3	0	0.170646	1	0	0.153905	3	1

In [133…

```
X_val6.to_numpy()
```

Out[133]:
```
array([[3.        , 1.        , 0.34656949, ..., 0.02049464, 6.        ,
        1.        ],
       [2.        , 1.        , 0.3842674 , ..., 0.04006213, 3.        ,
        3.        ],
       [3.        , 1.        , 0.24604172, ..., 0.01541158, 2.        ,
        3.        ],
       ...,
       [3.        , 0.        , 0.4722292 , ..., 0.01541158, 0.        ,
        3.        ],
       [2.        , 0.        , 0.2083438 , ..., 0.15216447, 4.        ,
        3.        ],
       [3.        , 0.        , 0.04498618, ..., 0.1545881 , 6.        ,
        3.        ]])
```

In [134…
```
yX_val6 = logreg.predict(X_val6)
```

In [135…
```
a6 = accuracy_score(Y_val, yX_val6)
```

In [136…
```
#importancePclass= accuracy score X_Val-accuracy score X_val6
importanceFare = 0.7932960893854749-a6
print (importanceFare)
```

```
0.0
```

In [137…
```
#***seventh column permutation
X_val7 = X_val.copy()
X_val7.head()
#panda for permutation
X_val7['Cabin'] = np.random.permutation(X_val7['Cabin'])
X_val7.head()
```

Out[137]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.346569	1	1	0.029758	2	1
439	2	1	0.384267	0	0	0.020495	3	3
840	3	1	0.246042	0	0	0.015469	4	3
720	2	0	0.070118	0	1	0.064412	6	3
39	3	0	0.170646	1	0	0.021942	0	1

In [138…
```
X_val7.to_numpy()
```

Out[138]:
```
array([[3.        , 1.        , 0.34656949, ..., 0.02975782, 2.        ,
        1.        ],
       [2.        , 1.        , 0.3842674 , ..., 0.02049464, 3.        ,
        3.        ],
       [3.        , 1.        , 0.24604172, ..., 0.01546857, 4.        ,
        3.        ],
       ...,
       [3.        , 0.        , 0.4722292 , ..., 0.06126432, 6.        ,
        3.        ],
       [2.        , 0.        , 0.2083438 , ..., 0.02049464, 3.        ,
        3.        ],
       [3.        , 0.        , 0.04498618, ..., 0.03259623, 3.        ,
        3.        ]])
```

In [139…
```
yX_val7 = logreg.predict(X_val7)
```

In [140…
```
a7 = accuracy_score(Y_val, yX_val7)
```

In [141…
```
#importancePclass= accuracy score X_Val-accuracy score X_val7
importanceCabin = 0.7932960893854749-a7
print (importanceCabin)
```

```
-0.005586592178770888
```

In [142…
```
#***eighth column permutation
X_val8 = X_val.copy()
X_val8.head()
#panda for permutation
X_val8['Embarked'] = np.random.permutation(X_val8['Embarked'])
X_val8.head()
```

Out[142]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
709	3	1	0.346569	1	1	0.029758	6	2
439	2	1	0.384267	0	0	0.020495	3	1
840	3	1	0.246042	0	0	0.015469	2	1
720	2	0	0.070118	0	1	0.064412	2	1
39	3	0	0.170646	1	0	0.021942	3	1

In [143…
```
X_val8.to_numpy()
```

Out[143]:
```
array([[3.        , 1.        , 0.34656949, ..., 0.02975782, 6.        ,
        2.        ],
       [2.        , 1.        , 0.3842674 , ..., 0.02049464, 3.        ,
        1.        ],
       [3.        , 1.        , 0.24604172, ..., 0.01546857, 2.        ,
        1.        ],
       ...,
       [3.        , 0.        , 0.4722292 , ..., 0.06126432, 0.        ,
        3.        ],
       [2.        , 0.        , 0.2083438 , ..., 0.02049464, 4.        ,
        3.        ],
       [3.        , 0.        , 0.04498618, ..., 0.03259623, 6.        ,
        2.        ]])
```

In [144…
```
yX_val8 = logreg.predict(X_val8)
```

In [145…
```
a8 = accuracy_score(Y_val, yX_val8)
```

In [146…
```
#importancePclass= accuracy score X_Val-accuracy score X_val8
importanceEmbarked = 0.7932960893854749-a8
print (importanceEmbarked)
```

```
-0.03351955307262566
```

In [147…
```
dictionary_feature_importance = {'Pclass': [importancePclass], 'Sex':[importanceSe
```

In [148…
```
feature_importance = pd.DataFrame(dictionary_feature_importance)
feature_importance.head()
```

Out[148]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0.022346	0.223464	0.011173	0.022346	-0.005587	0.0	-0.005587	-0.03352

In [149…
```
#We can see that the strongest impact has variable Sex
#spremala sam varijable accuracy u posebne jer iz svakog novog refresha mala razli
```

```
#u prenesenom oduzimanju niže
```

In [150…

```
#submission.to_csv('TitanicKate2023visual1.csv')
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: