**Titanic Survival Predictions**

The article should contain the following sub-topics :

1.Problem Definition  
2.Data Analysis  
3.EDA Concluding Remarks  
4.Pre-processing Pipeline  
5.Building Machine Learning Models  
6.Concluding Remarks

# Introduction of Data Set:

The Problem of Titanic is based on the sinking of the unsinkable ship” Titanic”. The data set will provide you the information like the ages of different people, their sexes, sibling counts, embarkment points and also about their survival because of the tragic disaster. Based on all these factors, you have to predict that a particular passenger on Titanic would survived the tragic disaster or not.

# Problem Definition:

The data set of Titanic gives insight for each and every passenger and their survival outcome. The problem statement entails predicting whether the passenger would survive or not the disaster with respect to the factors like class of passenger, age, fair, sex, number of siblings and also number of parents/children abroad.

# Data Analysis:

The data is about of 891 people who were there on the Titanic ship.

* Step by step Process:

I started with importing necessary modules

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** warnings

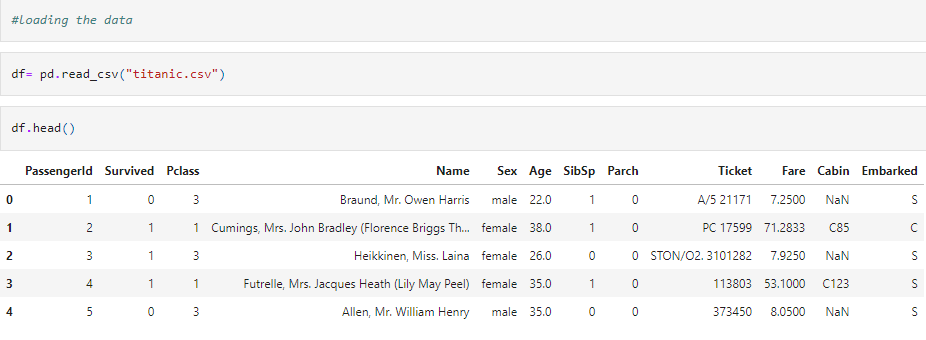
warnings**.**filterwarnings('ignore')

I loaded the data and display the data set

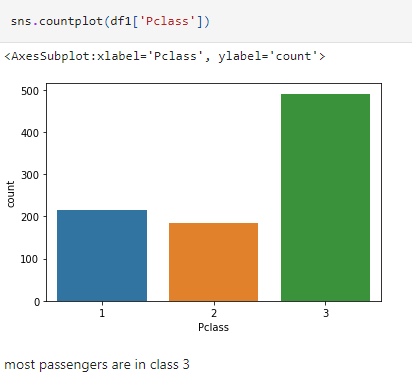
df**=** pd**.**read\_csv("titanic.csv")

In [5]:

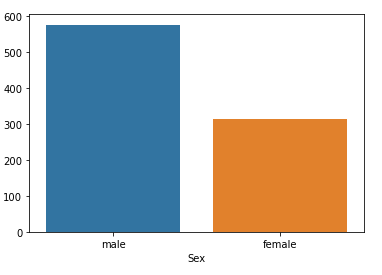
df**.**head()



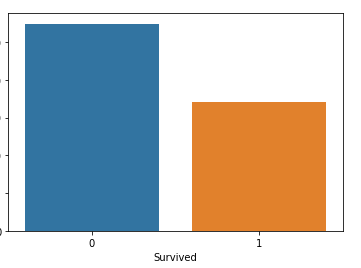
I found that most passengers are from class 3rd



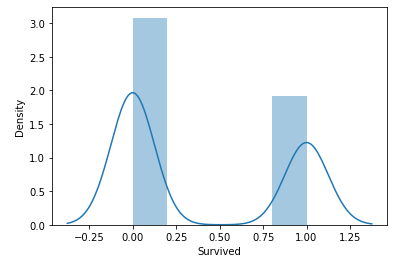
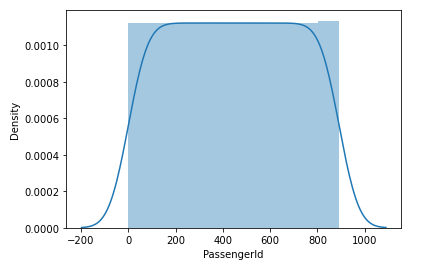
From the data set I found that male passenger are more in number than females.

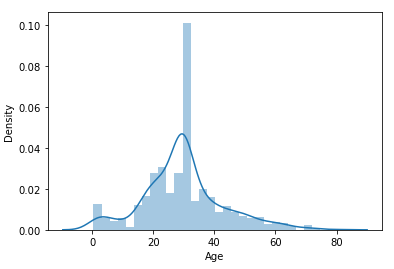
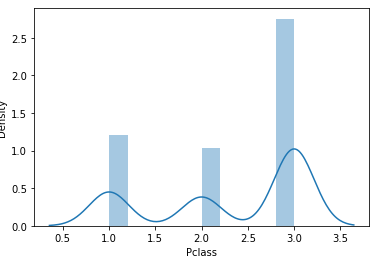


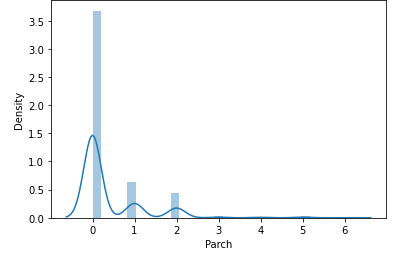
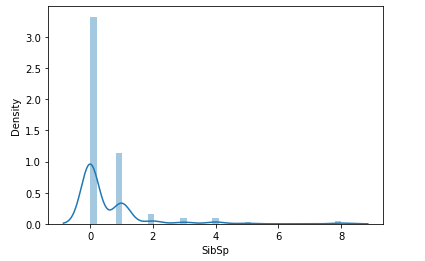
The passenger who survived the disaster are less in number. Out of 891 people, 549 people didn’t survived and the number of survival is 342. According to data, The survival rate of female is higher if age is between 18-80yrs old & from class 1. & Male survival rate in 2nd class with age 18-80 yrs old is worst.

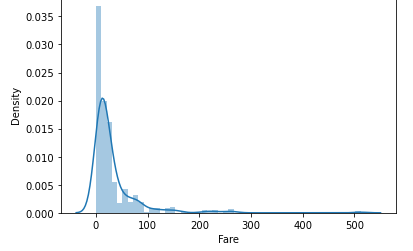


Let see the distribution of the data:









# EDA Concluding Remarks:

In the data set there are some missing values in the column in age, cabin and embark.

Code: df**.**isnull()**.**sum()

Out[13]:

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 177

SibSp 0

Parch 0

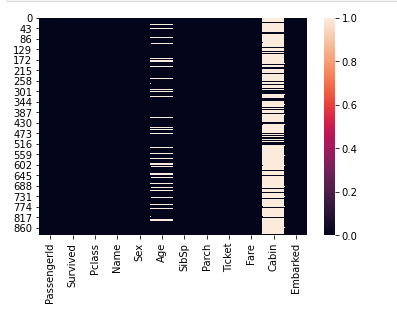
Ticket 0

Fare 0

Cabin 687

Embarked 2

dtype: int64



If we drop these column, 80% of the data will be lost. So I decided to fill the values with mean in column ‘age’. In Embarked column, I filled the missing values with ‘S’ as it has more data in it. I drop the cabin column as it is not necessary for the data.

Code: df**.**columns It will shows all the columns

Out[7]:

Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',

'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],

dtype='object')

# Pre-processing Pipeline:

I started with checking the skewness and outliers in the data to make the data set more precise.

Code: df1**.**skew()

Out[27]:

PassengerId 0.000000

Survived 0.478523

Pclass -0.630548

Age 0.434488

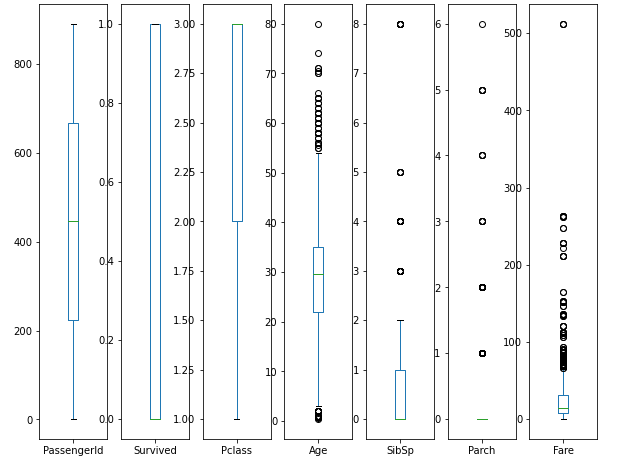
SibSp 3.695352

Parch 2.749117

Fare 4.787317

dtype: float64

Code: df1**.**plot(kind**=**'box',subplots**=True**,figsize**=**(10,8))



In the above figure, you can see that

* passengerId, & survived are normally distributed.
* Pclass is negatively skewed.
* SibSp,Parch and Fareare positively skewed.

When I was checking the types of data, ‘Sex’, ‘Name’ and ‘Embarked’ column have object values while all the other columns have float and integer values in the data.

Code: df**.**dtypes

Out[8]:

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

Before converting them I removed some unwanted columns like 'Fare', 'Name', 'Ticket', and 'PassengerId'. After that with the help of Label Encoding, all the object type columns are encoded to numerical forms.

**Code:**

**from** sklearn.preprocessing **import** LabelEncoder

le**=**LabelEncoder()

df2['Sex']**=**le**.**fit\_transform(df2['Sex'])

df2['Sex']**.**value\_counts()

Out[45]:

1 577

0 314

Name: Sex, dtype: int64

male=1 and female=0

**Code:**

**from** sklearn.preprocessing **import** LabelEncoder

le**=**LabelEncoder()

df2['Embarked']**=**le**.**fit\_transform(df2['Embarked'])

df2['Embarked']**.**value\_counts()

Out[47]:

2 646

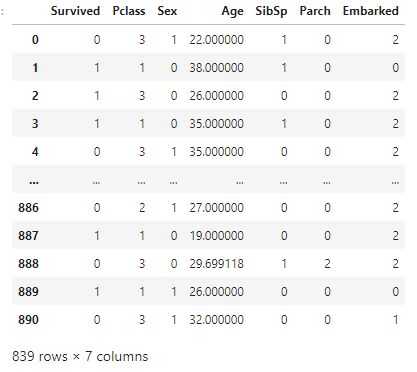
0 168

1 77

Name: Embarked, dtype: int64

S=2, C=0, & Q=1 (encoded)

You can now see the data after encoding.



Now, I checked the data for multicollinearity.

**Code:**

**import** statsmodels.api **as** sm

**from** scipy **import** stats

**from** statsmodels.stats.outliers\_influence **import** variance\_inflation\_factor

In [56]:

**def** calc\_vif(x1):

vif**=**pd**.**DataFrame()

vif['Variables']**=**x1**.**columns

vif['vif Factor']**=**[variance\_inflation\_factor(x1**.**values,i) **for** i **in** range(x1**.**shape[1])]

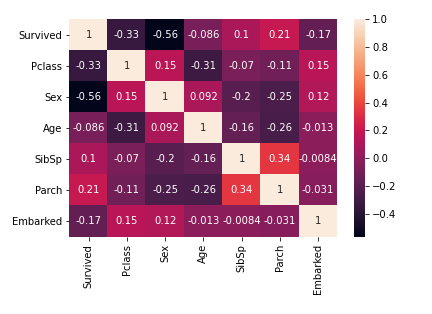
**return** (vif)

In [57]:

calc\_vif(df3)



Now, you can see the correlation of the data:



We can now remove the outliers with the help of Zscore.

**Code:**

**from** scipy.stats **import** zscore

z **=** np**.**abs(zscore(df2))

z

df3**=**df2[(z**<**3)**.**all(axis**=**1)]

# Building Machine Learning Models:

As we know that all of our columns are in integer and floats we can split the data for model building.

df2**.**dtypes

Out[42]:

Survived int64

Pclass int64

Sex object

Age float64

SibSp int64

Parch int64

Embarked object

dtype: object

For splitting the data into x and y, we have to use below code:

Code:

x**=**df3**.**drop('Survived',axis**=**1)

y**=**df3['Survived']

We split our data as ‘Survived’ column is the target column.

Now, we have to scale the features in the data set.

**Code:**

**from** sklearn.preprocessing **import** StandardScaler

scale**=** StandardScaler()

x**=** scale**.**fit\_transform(x)

From the above code the data set is now scaled and we can send the data in to machine learning model for further process.

Now we have to import necessary libraries for sending data into the machine learning models.

**Code:**

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.metrics **import** accuracy\_score

We import train\_test\_split from sklearn and also I have also find the best random state to test the data. Finding the best random state will help the model to find the best accuracy of the model.

Code:

lr**=** LogisticRegression()

max**=** 0

rs**=** 0

**for** i **in** range(1001):

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.25,random\_state**=**i)

lr**.**fit(x\_train,y\_train)

pred**=** lr**.**predict(x\_test)

score**=** accuracy\_score(y\_test,pred)

**if** max**<**score:

max**=** score

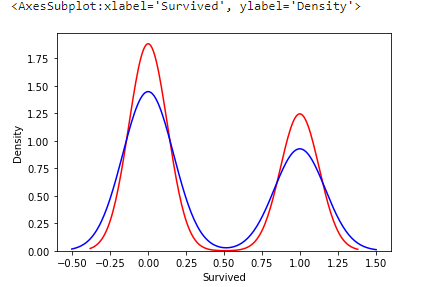
rs**=** i

print('Max score is: ',max,' at random state -',rs)

Max score is: 0.8666666666666667 at random state - 91

From the above code, we have found that we have to split the data by 91 percent. It gives the accuracy of approx 87 percent in Linear regression model.

After that i have visualize the fitting of the model which means that the model is over fit or the model is under fit.



Now I have use the train test split method with the best random state i.e. 91

Code:

x\_train,x\_test,y\_train,y\_test**=**train\_test\_split(x,y,test\_size**=**0.25,random\_state**=**91)

Now, I have tried to test the data with different other models like Decision tree classifier, Gaussian NB, K- Neighbour classifier, and SVC. For this, we have to import these models.

**Code:**

**from** sklearn.metrics **import** confusion\_matrix, classification\_report

**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.naive\_bayes **import** GaussianNB

**from** sklearn.svm **import** SVC

**from** sklearn.neighbors **import** KNeighborsClassifier

dtc**=**DecisionTreeClassifier()

gnb**=**GaussianNB()

svc**=**SVC()

knn**=**KNeighborsClassifier()

models**=**[lr,dtc,gnb,svc,knn]

**for** m **in** models:

m**.**fit(x\_train,y\_train)

predm**=**m**.**predict(x\_test)

print(f'The accuracy score of {m} is :',accuracy\_score(y\_test,predm))

print(f'The confusion matrix of {m} is:\n',confusion\_matrix(y\_test,predm))

print(f'The classification report of {m} is :\n',classification\_report(y\_test,predm))

print('\n')

The accuracy score of LogisticRegression() is : 0.8666666666666667

The confusion matrix of LogisticRegression() is:

[[109 10]

[ 18 73]]

The classification report of LogisticRegression() is :

precision recall f1-score support

0 0.86 0.92 0.89 119

1 0.88 0.80 0.84 91

accuracy 0.87 210

macro avg 0.87 0.86 0.86 210

weighted avg 0.87 0.87 0.87 210

The accuracy score of DecisionTreeClassifier() is : 0.8095238095238095

The confusion matrix of DecisionTreeClassifier() is:

[[105 14]

[ 26 65]]

The classification report of DecisionTreeClassifier() is :

precision recall f1-score support

0 0.80 0.88 0.84 119

1 0.82 0.71 0.76 91

accuracy 0.81 210

macro avg 0.81 0.80 0.80 210

weighted avg 0.81 0.81 0.81 210

The accuracy score of GaussianNB()is : 0.861904761904762

The confusion matrix of GaussianNB() is:

[[103 16]

[ 13 78]]

The classification report of GaussianNB() is :

precision recall f1-score support

0 0.89 0.87 0.88 119

1 0.83 0.86 0.84 91

accuracy 0.86 210

macro avg 0.86 0.86 0.86 210

weighted avg 0.86 0.86 0.86 210

The accuracy score of SVC() is : 0.8571428571428571

The confusion matrix of SVC() is:

[[112 7]

[ 23 68]]

The classification report of SVC() is :

precision recall f1-score support

0 0.83 0.94 0.88 119

1 0.91 0.75 0.82 91

accuracy 0.86 210

macro avg 0.87 0.84 0.85 210

weighted avg 0.86 0.86 0.85 210

The accuracy score of KNeighborsClassifier() is : 0.8285714285714286

The confusion matrix of KNeighborsClassifier() is:

[[105 14]

[ 22 69]]

The classification report of KNeighborsClassifier() is

precision recall f1-score support

0 0.83 0.88 0.85 119

1 0.83 0.76 0.79 91

accuracy 0.83 210

macro avg 0.83 0.82 0.82 210

weighted avg 0.83 0.83 0.83 210

From the above result, we find that Logistic regression is giving the best accuracy.

Now, we will check the cross validation score, so that it will help to improve the accuracy.

Code:

**from** sklearn.model\_selection **import** cross\_val\_score

**for** m **in** models:

cvscore**=**cross\_val\_score(m,x,y,cv**=**10)

print(f'Metrics of model {m}')

print(cvscore)

print(f'Mean cv score: ',cvscore**.**mean())

print(cvscore**.**std())

print('\n')

Metrics of model LogisticRegression()

[0.78571429 0.78571429 0.76190476 0.8452381 0.77380952 0.77380952

0.79761905 0.78571429 0.80952381 0.8313253 ]

Mean cv score: 0.7950372920252439

0.025126124021946913

Metrics of model DecisionTreeClassifier()

[0.66666667 0.75 0.73809524 0.78571429 0.82142857 0.79761905

0.8452381 0.76190476 0.83333333 0.78313253]

Mean cv score: 0.7783132530120481

0.04996910714930067

Metrics of model GaussianNB()

[0.72619048 0.76190476 0.73809524 0.83333333 0.79761905 0.77380952

0.82142857 0.75 0.80952381 0.80722892]

Mean cv score: 0.7819133677567413

0.03516375234440463

Metrics of model SVC()

[0.82142857 0.85714286 0.75 0.8452381 0.83333333 0.77380952

0.82142857 0.76190476 0.8452381 0.84337349]

Mean cv score: 0.8152897303499713

0.03683536988898986

Metrics of model KNeighborsClassifier()

[0.75 0.79761905 0.76190476 0.80952381 0.89285714 0.78571429

0.82142857 0.72619048 0.79761905 0.74698795]

Mean cv score: 0.7889845094664372

0.04514832777546134

By comparing accuracy score and cv score : The best model is – Knn

Now, I have done Hyper tunning to improve the accuracy of the model.

**Code:**

**from** sklearn.model\_selection **import** GridSearchCV

par**=**{'n\_neighbors':[1,5,10,15,20,30,40,50],'weights':['uniform','distance'],'algorithm':['auto', 'ball\_tree', 'kd\_tree', 'brute'],

'leaf\_size':[10,30,50,80],'p':[1,2]}

grid**=**GridSearchCV(knn,param\_grid**=**par)

grid**.**fit(x\_train,y\_train)

print('Best score:',grid**.**best\_score\_)

print('Best Estimator:',grid**.**best\_estimator\_)

print('Best parameter:',grid**.**best\_params\_)

Best score: 0.8108444444444445

Best Estimator: KNeighborsClassifier(leaf\_size=10, n\_neighbors=10, p=1)

Best parameter: {'algorithm': 'auto', 'leaf\_size': 10, 'n\_neighbors': 10, 'p': 1, 'weights': 'uniform'}

best\_model**=**KNeighborsClassifier(algorithm**=**'auto', leaf\_size**=** 10, n\_neighbors**=** 15, p**=** 2, weights**=** 'uniform')

best\_model**.**fit(x\_train,y\_train)

predb**=**best\_model**.**predict(x\_test)

print('Accuracy Score',accuracy\_score(y\_test,predb))

print('Confusion matrix:\n',confusion\_matrix(y\_test,predb))

print('classification report:\n',classification\_report(y\_test,predb))

Accuracy Score 0.8571428571428571

Confusion matrix:

[[111 8]

[ 22 69]]

classification report:

precision recall f1-score support

0 0.83 0.93 0.88 119

1 0.90 0.76 0.82 91

accuracy 0.86 210

macro avg 0.87 0.85 0.85 210

weighted avg 0.86 0.86 0.86 210

accuracy is 86%

Now I have made the data set to check the actual and predicted model

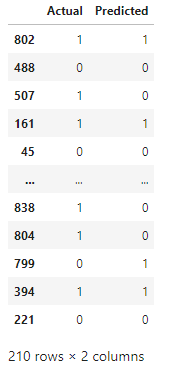
Code:

results**=**pd**.**DataFrame({})

results['Actual']**=**y\_test

results['Predicted']**=**predb

results



Now I have checked the roc curve

**from** sklearn.metrics **import** roc\_curve

fpr,tpr,thresholds**=**roc\_curve(y\_test,y\_pred\_prob)

In [120]:

plt**.**plot([0,1],[0,1],'k--')

plt**.**plot(fpr,tpr,label**=**'KNeighborsClassifier')

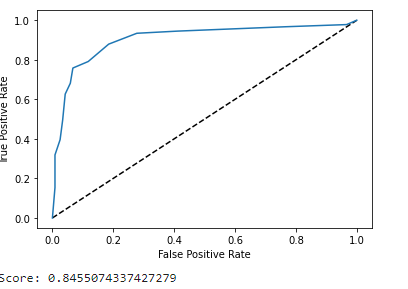
plt**.**xlabel('False Positive Rate')

plt**.**ylabel('True Positive Rate')

plt**.**show()

auc\_score**=**roc\_auc\_score(y\_test,best\_model**.**predict(x\_test))

print('Score:',auc\_score)



# Concluding Remarks:

This project aims to find the factors that may affect the probability of survival of individual passengers & crew when the tragic disaster occured. I have used 892 training data from Titanic case, while logistic regression, linear regression, decision tree & Random Forest Classifier and K- Neighbour are implemented in this project. K-Neighbour is the best model in this project with accuracy of 85 percent.

From this project, I have observed that in terms of each factor, sex & passenger class are the 2 major factors that may determine survival of individual passenger, e.g. female & higher class passengers will be more likely to survive in this case.