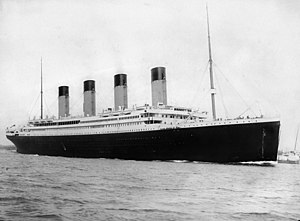
**Analysis on Famous Titanic Dataset**

* **Objective:**

In this blog, I am going to explain various machine learning model on the famous Titanic dataset, to explore multiple predictions and will save the best model. Through analysis, we will also try to get information on the fate of passengers on the Titanic, and summarized them according to class, sex, age and survival rate.

* **Let’s talk about RMS Titanic.**



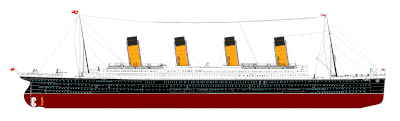
RMS Titanic was a British passenger liner operated by the White Star Line that sank in the North Atlantic Ocean on 15 April 1912, after striking an iceberg during her maiden voyage from Southampton to New York City. Of the estimated 2,224 passengers and crew aboard, more than 1,500 died, making the sinking at the time one of the deadliest of a single ship and the deadliest peacetime sinking of a superliner or cruise ship to date. With much public attention in the aftermath the disaster has since been the material of many artistic works and a founding material of the disaster film genre.

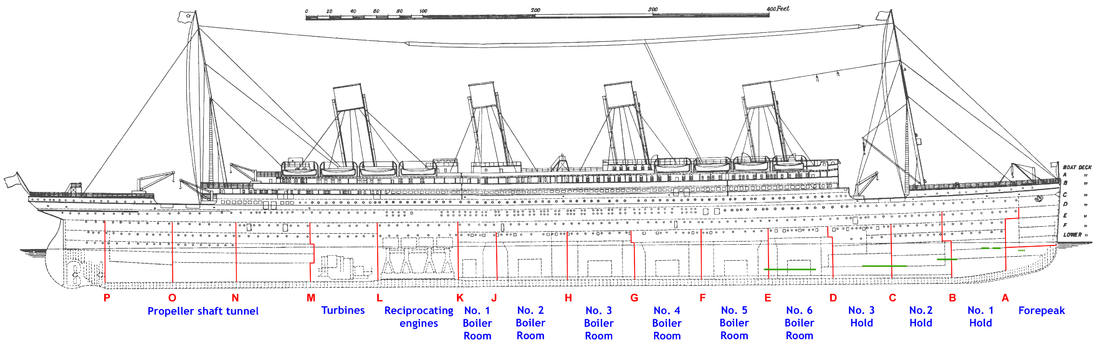
RMS Titanic was the largest ship afloat at the time she entered service and was the second of three Olympic-class ocean liners operated by the White Star Line. She was built by the Harland and Wolff shipyard in Belfast. Thomas Andrews, chief naval architect of the shipyard at the time, died in the disaster.

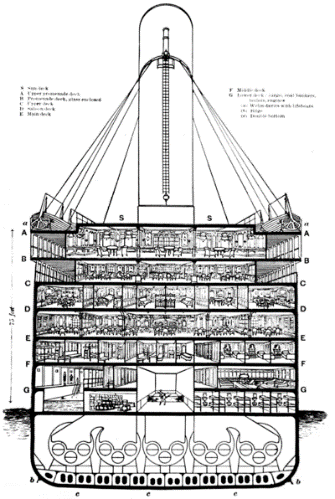
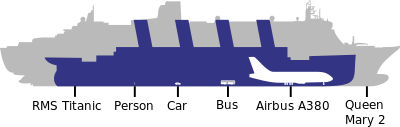
**Dimensions and Layout:**

Titanic was 882 feet 9 inches (269.06 m) long with a maximum breadth of 92 feet 6 inches (28.19 m). Her total height, measured from the base of the keel to the top of the bridge, was 104 feet (32 m).[25] She measured 46,328 gross register tons and with a draught of 34 feet 7 inches (10.54 m), she displaced 52,310 tons.

All three of the Olympic-class ships had ten decks (excluding the top of the officers' quarters), eight of which were for passenger use.





**Timeline of RMS Titanic**

* 17 September 1908: Ship ordered.
* 31 May 1911: Ship launched.
* 1 April 1912: Trials completed.
* 10 April, noon: Maiden voyage starts. Leaves Southampton dock, narrowly escaping collision with American liner New York.
* 10 April, 19:00: Stops at Cherbourg for passengers.
* 10 April, 21:00: Leaves Cherbourg for Queenstown.
* 11 April, 12:30: Stops at Queenstown for passengers and mail.
* 11 April, 14:00: Leaves Queenstown for New York.
* 14 April, 23:40: Collision with iceberg (Latitude 41° 46′ N, Longitude 50° 14′ W).
* 15 April, 00:45: First boat, No. 7, lowered.
* 15 April, 02:05: Last boat, Collapsible D, lowered.
* 15 April, 02:20: Foundering.
* 15 April, 03:30–08:50: Rescue of survivors.
* 19 April – 25 May: US inquiry.
* 2 May – 3 July: British inquiry.
* 1 September 1985: Discovery of wreck.

For more information regarding same you may refer to below Wikipedia article.

<https://en.wikipedia.org/wiki/Titanic>

* **Titanic Dataset Overview.**

|  |  |  |
| --- | --- | --- |
| Column | Definition | Comments |
| PassengerId | Passenger ID |  |
| Survived | Survival | 0 = Died, 1 = Survived |
| Pclass | Ticket class | 1 = 1st class, 2 = 2nd class, 3 = 3rd class |
| Name | Name of the Passenger |  |
| Sex | Gender |  |
| Age | Age in years |  |
| SibSp | # of siblings / spouses aboard the Titanic |  |
| Parch | # of parents / children aboard the Titanic |  |
| Ticket | Ticket number |  |
| Fare | Passenger fare |  |
| Cabin | Cabin number |  |
| Embarked | Port of Embarkation | C = Cherbourg, Q = Queenstown, S = Southampton |

**Variable Notes:**

**pclass**: A proxy for socio-economic status (SES)

1st = Upper

2nd = Middle

3rd = Lower

**age**: Age is fractional if less than 1. If the age is estimated, is it in the form of xx.5

**sibsp**: The dataset defines family relations in this way...

Sibling = brother, sister, stepbrother, stepsister

Spouse = husband, wife (mistresses and fiancés were ignored)

**parch**: The dataset defines family relations in this way...

Parent = mother, father

Child = daughter, son, stepdaughter, stepson

Some children travelled only with a nanny, therefore parch=0 for them.

Let’s start Analysis on Titanic Dataset. Below you will find the way to do analysis on Titanic dataset using Python.

* **Importing the Libraries**

First, we need to import library to start analysis on data.

***# required Library***

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

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

**import** **sklearn** **as** **skl**

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

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

After importing libraries, we need to get data from csv and start doing analysis on it. We need to check various factor such as datatype, null values(missing values) to do more efficient analysis. If required, we also need to create additional columns for further analysis.

* **Getting the Data**

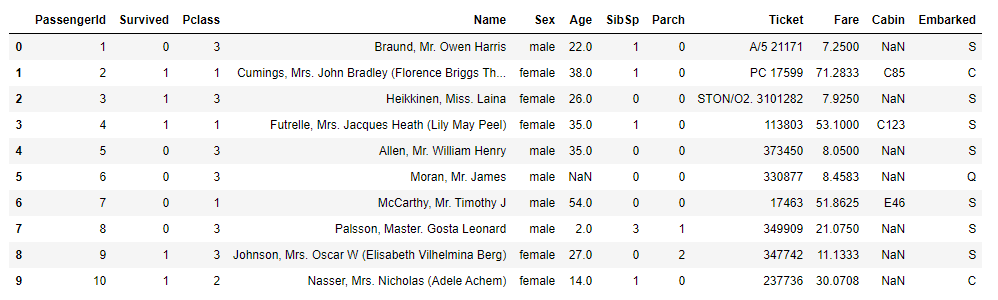
***# Load the data***

Titanic=pd.read\_csv(‘D:\DATA SC\_Practice Project\\p3\\titanic\_train.csv’)

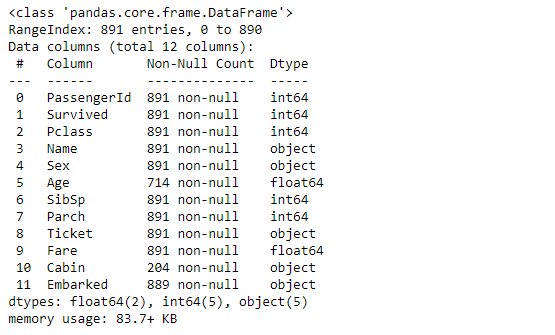
***# print the first 10 rows of titanic dataset***

Titanic.head(10)

Using above command, we can see the top 10 entries. It is always good to look on the data before start analysis. It is very helpful and easy to understand.



Titanic.info()



***# Get a total number of Rows & Columns***

Titanic.shape

(891, 12)

From above, we can see that data contains 891 rows and 12 columns.

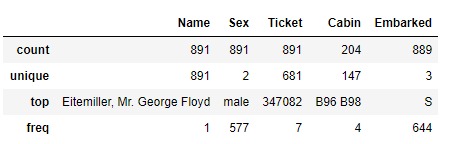
***# Get data in statistics form.***

Titanic.describe()



***# Lets Describe string values***

Titanic.describe(include=object)



***# Get a count of number of survivors on Titanic***

Titanic['Survived'].value\_counts()

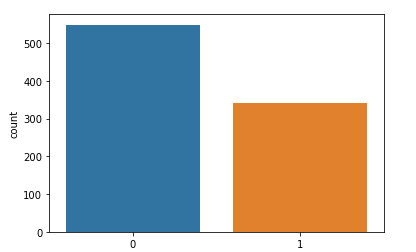
0 549

1 342

Name: Survived, dtype: int64

***# Visualize the count of number of survivors***

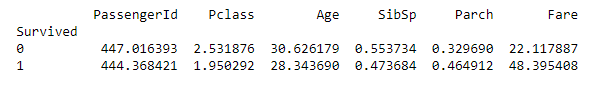
sns.countplot(Titanic['Survived'])



***# Let's find out the mean values of survivors with different group***

titanic=Titanic.groupby('Survived').mean()

print(titanic)



In the above outcome we can see that the fare cost of survived people was average 48.39

* **SOME KEY-OBSERVATIONS FROM THE ABOVE COLLECTED STATISTICAL DATA :**

1.Missing values found in the columns "Age" and "Cabin".

2.Around 38% of passengers have survived.

3.Also we need to change convert some objects to integers for a proper analysis.

4.Removing some Non-correlated data will help in efficient data analysis.

* **EXPLORATORY DATA ANALYSIS(EDA)/ DATA EXPLORATION**

plt.figure(figsize=(15,25))

sns.set\_style("whitegrid")

plt.subplot(3,3,1)

sns.countplot(Titanic["Survived"]).set\_title("survivors")

plt.subplot(3,3,2)

sns.set\_style("whitegrid")

sns.countplot(x="Survived",hue="Sex",data=Titanic,palette="rainbow").set\_title("Survivors among Male and Female")

plt.subplot(3,3,3)

sns.set\_style("whitegrid")

sns.countplot(x="Survived",hue="Pclass",data=Titanic,palette="rainbow").set\_title("survivors vs class")

plt.subplot(3,3,4)

sns.set\_style("whitegrid")

sns.countplot(x="Survived",hue="Embarked",data=Titanic,palette="rainbow").set\_title("Survivors vs Embarked")

plt.subplot(3,3,5)

sns.set\_style("whitegrid")

sns.countplot(x="Embarked",hue="Sex",data=Titanic,palette="rainbow").set\_title("survivors vs Sex")

plt.subplot(3,3,6)

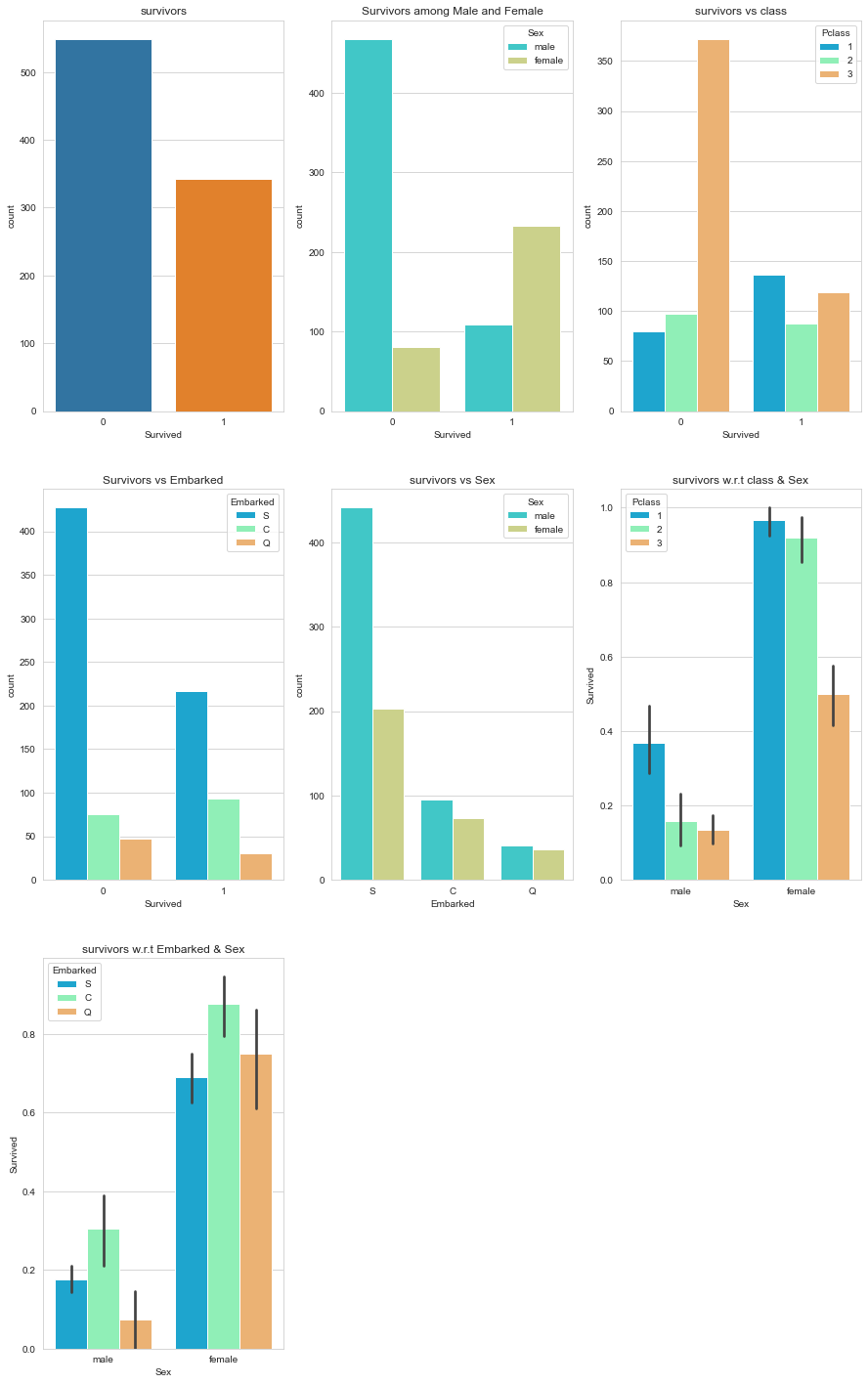
sns.set\_style("whitegrid")

sns.barplot(x="Sex",y="Survived",hue="Pclass",data=Titanic,palette="rainbow").set\_title("survivors w.r.t class & Sex")

plt.subplot(3,3,7)

sns.set\_style("whitegrid")

sns.barplot(x="Sex",y="Survived",hue="Embarked",data=Titanic,palette="rainbow").set\_title("survivors w.r.t Embarked & Sex")



**FROM THE ABOVE PLOTTED GRAPHS:**

1. We can see that the number of survivors is very less compared to the died.

2. Women have a higher survival rate compared to men.

3. Women from the 1st class seems to have survived a lot compared to all other classes.

4. The mortality rate for 3rd class passengers is high across all genders.

5. The passengers embarked from 'Port S' have a higher survival rate compared to all other ports.

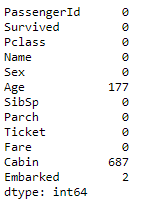
6. Also the mortality rate for the 'Port S' is high compared to all other ports.

7. Most of the passengers are men who have embarked from 'Port S'.

8. Most of the passengers form 1st class are women. Thus we observe that Sex have higher relation with survived passengers. But we can find that the attribute sex is object, so we will convert it into integer data type. So up next, we will change the sex data type and also replacing Null values with mean value in Age column will give us a solid Dataset to work with.

***#checking for Null values***

Titanic.isnull().sum()



Here we can see that column Age and Cabin have null values. Here we can replace null value of Age with mean of age. And drop the column as it is of no use.

***#findind mean value for Age***

Titanic["Age"].mean()

29.69911764705882

***#replacing the mean value with NaN values in Age attribute***

Titanic["Age"].fillna(Titanic["Age"].mean(),inplace=**True**)

Titanic.isnull().sum()

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 0

SibSp 0

Parch 0

Ticket 0

Fare 0

Cabin 687

Embarked 2

dtype: int64

***#dropping Cabin attribute since it has a lot of missing values***

Titanic.drop(["Cabin"],axis=1,inplace=**True**)

Titanic

Titanic.dropna(inplace=**True**)

Titanic.shape

(889, 11)

***#Checking Null Value again, if we have in any column***

Titanic.isnull().sum()

PassengerId 0

Survived 0

Pclass 0

Name 0

Sex 0

Age 0

SibSp 0

Parch 0

Ticket 0

Fare 0

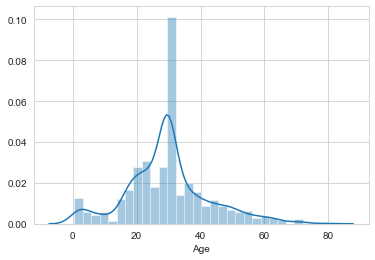
Embarked 0

dtype: int64

Since we have no Null values in any of the given attributes, we can proceed with further data analysis.

***#UNIVARIATE ANALYSIS,BIVARIATE AND MULTI-VARIATE ANALYSIS***

sns.distplot(Titanic["Age"])

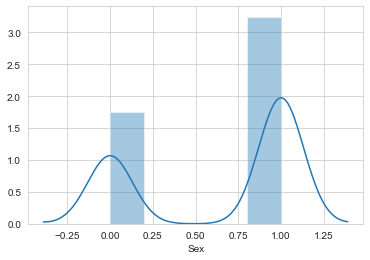


From the above data, we conclude that we have a normal distribution among Age.

***# Check the rate of survivors between male females and child***

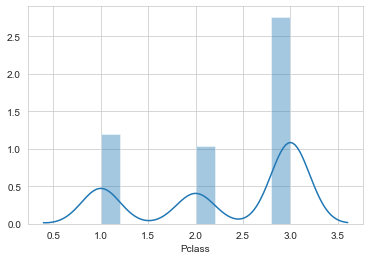
Titanic.Sex=Titanic.Sex.map({"male":1, "female":0})

sns.distplot(Titanic["Sex"])



Male are higher than female among all the passengers.

sns.distplot(Titanic["Pclass"])

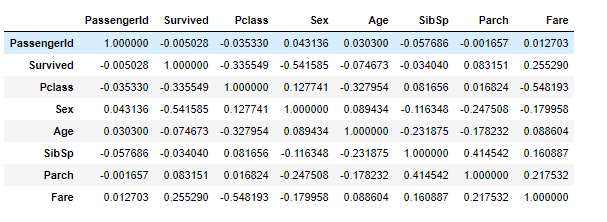


sns.pairplot(Titanic)



dfcor=Titanic.corr()

dfcor

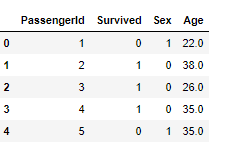


Many attributes such as Name, Fare, sibsb, tickets, pclass, embarked will have to be removed to check the accuracy of the given mode.

***#removing the unwanted attributes for this analysis.***

Titanic.drop(["Name","Fare","SibSp","Parch","Ticket","Pclass","Embarked"],axis=1,inplace=**True**)

Titanic.head()

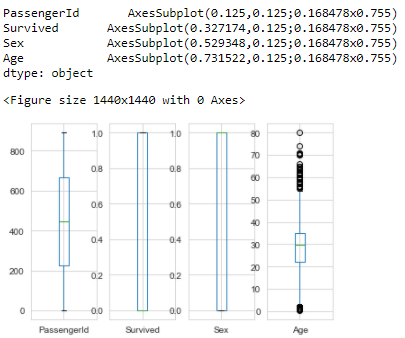


* **ANALYSING THE OUTLIERS.**

An outlier is an element of a data set that distinctly stands out from the rest of the data. In other words, outliers are those data points that lie outside the overall pattern of distribution. The easiest way to **detect outliers is to create a graph. Plots such as Box plots, Scatterplots and Histograms** **can help to detect outliers**. Alternatively, we can use **mean and standard deviation** to list out the outliers. Interquartile Range and Quartiles can also be used to detect outliers.

plt.figure(figsize=(20,20))

Titanic.plot(kind="box",subplots=**True**)



IT SEEMS "AGE" HAS LOT OF OUTLIERS PRESENT IN THEM. WE WILL HAVE TO REMOVE THE OUTLIERS FOR FURTHER PROCESS.

* **REMOVING THE OUTLIERS**

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

zs=np.abs(zscore(Titanic))

zs

array([[1.73250451, 0.78696114, 0.73534203, 0.59049493],

[1.72861124, 1.27071078, 1.35991138, 0.64397101],

[1.72471797, 1.27071078, 1.35991138, 0.28187844],

...,

[1.72471797, 0.78696114, 1.35991138, 0.00352373],

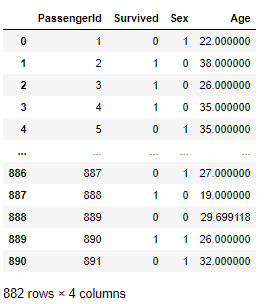
[1.72861124, 1.27071078, 0.73534203, 0.28187844],

[1.73250451, 0.78696114, 0.73534203, 0.18104628]])

threshold=3

Titanic\_new=Titanic[(zs<3).all(axis=1)]

Titanic\_new



print(Titanic.shape)

print(Titanic\_new.shape)

(889, 4)

(882, 4)

* **MODEL TRAINING**

In Machine Learning we create models to predict the outcome of certain events. To measure if the model is good enough, we can use a method called Train/Test.

**What is Train/Test?**

Train/Test is a method to measure the accuracy of your model. It is called Train/Test because you split the data set into two sets: a training set and a testing set.

* You train the model using the training set**.(Train the model means create the model.)**
* You test the model using the testing set.**(Test the model means test the accuracy of the model.)**

***#SEPERATING DEPENDENT AND IN-DEPENDENT VARIABLES***

x=Titanic\_new.drop("Survived",axis=1)

y=Titanic\_new["Survived"]

*#SPLITTING THE TESTING AND TRAINING DATAS*

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

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

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

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

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=.22,random\_state=42)

x\_train.shape

(687, 3)

y\_train.shape

(687,)

x\_test.shape

(195, 3)

y\_test.shape

(195,)

* **FINDING ACCURACY OF THE GIVEN DATASET**

**Logistic Regression** is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). In other words, the logistic regression model predicts P(Y=1) as a function of X.

***#Logical Regression***

lr=LogisticRegression()

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

lr\_pre=lr.predict(x\_test)*#testing*

print(lr\_pre)

print("**\n**Accuracy : ",accuracy\_score(y\_test,lr\_pre))

print("**\n**Confusion matrix : ",confusion\_matrix(y\_test,lr\_pre))

print("**\n**classification report : ",classification\_report(y\_test,lr\_pre))

[1 1 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 0 1 1 1 1 0 0 0 1 0 0 0 1 1 0 1 0

1 0 0 0 1 0 0 0 1 1 0 0 1 0 0 0 1 1 0 1 0 1 1 0 0 0 0 0 1 1 0 1 1 0 0 1 1

1 0 0 1 1 0 1 1 0 0 1 0 1 0 0 1 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 1 1 1

1 0 0 1 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0

1 1 1 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1 0 1 0 0 1 0 0 1 0 0 0 1 0 0 0 0 1 0

0 0 0 0 0 0 1 1 0 1]

Accuracy : 0.7948717948717948

Confusion matrix : [[99 23]

[17 56]]

classification report : precision recall f1-score support

0 0.85 0.81 0.83 122

1 0.71 0.77 0.74 73

accuracy 0.79 195

macro avg 0.78 0.79 0.78 195

weighted avg 0.80 0.79 0.80 195

**Random Forest** is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

***#Random Forest Regressor Method***

**from** **sklearn.ensemble** **import** RandomForestRegressor

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

**from** **sklearn.metrics** **import** r2\_score

rfr=RandomForestRegressor(criterion="mae",n\_estimators=200)

rfr.fit(x\_train,y\_train)

rfr.score(x\_train,y\_train)

pre=rfr.predict(x\_test)

pre

r2score=r2\_score(y\_test,pre)

cvs=cross\_val\_score(rfr,x\_train,y\_train,cv=5).mean()

print(f"Accuracy=**{**r2score\*100**}**,cross\_value\_score=**{**cvs\*100**}**,and difference=**{**(r2score\*100)-(cvs\*100)**}**")

Accuracy=28.521968335953307,cross\_value\_score=15.16563016636826,and difference=13.356338169585047

The [**Decision Trees**](https://scikit-learn.org/stable/modules/tree.html#tree)is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.

*#DecisionTreeRegressor*

**from** **sklearn.tree** **import** DecisionTreeRegressor

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

**from** **sklearn.metrics** **import** r2\_score

dcr=DecisionTreeRegressor(criterion="friedman\_mse")

dcr.fit(x\_train,y\_train)

dcr.score(x\_train,y\_train)

pre=dcr.predict(x\_test)

pre

r2score=r2\_score(y\_test,pre)

cvs=cross\_val\_score(dcr,x\_train,y\_train,cv=5).mean()

print(f"Accuracy=**{**r2score\*100**}**,cross\_value\_score=**{**cvs\*100**}**,and difference=**{**(r2score\*100)-(cvs\*100)**}**")

Accuracy=-35.75117898046258,cross\_value\_score=-36.937458264908564,and difference=1.1862792844459804

The **K** in the name of this classifier represents the **k nearest neighbors**, where **k is an integer** value specified by the user. Hence as the name suggests, this classifier implements learning based on the k nearest neighbors. The choice of the value of k is dependent on data.

***#KNeighbors Classifier***

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

knn=KNeighborsClassifier()

knn.fit(x\_train,y\_train)

knn\_y=knn.predict(x\_test)

print("Accuracy Score",accuracy\_score(y\_test,knn\_y))

print("Confusion Matrix**\n**",confusion\_matrix(y\_test,knn\_y))

print("Classification Report**\n**", classification\_report(y\_test,knn\_y))

Accuracy Score 0.5435897435897435

Confusion Matrix

[[83 39]

[50 23]]

Classification Report

precision recall f1-score support

0 0.62 0.68 0.65 122

1 0.37 0.32 0.34 73

accuracy 0.54 195

macro avg 0.50 0.50 0.50 195

weighted avg 0.53 0.54 0.53 195

**Support vector machines (SVMs)** are a set of supervised learning methods used for [classification](https://scikit-learn.org/stable/modules/svm.html#svm-classification), [regression](https://scikit-learn.org/stable/modules/svm.html#svm-regression) and [outliers detection](https://scikit-learn.org/stable/modules/svm.html#svm-outlier-detection).

The support vector machines in scikit-learn support both dense (numpy.ndarray and convertible to that by numpy.asarray) and sparse (any scipy.sparse) sample vectors as input. However, to use an SVM to make predictions for sparse data, it must have been fit on such data.

***#Support Vector Machine***

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

**def** svmkernel(ker):

svc=SVC(kernel=ker)

svc.fit(x\_train,y\_train)

svc.score(x\_train,y\_train)

svc\_y=svc.predict(x\_test)

print("Accuracy Score",accuracy\_score(y\_test,svc\_y))

print("Confusion Matrix**\n**",confusion\_matrix(y\_test,svc\_y))

print("Classification Report**\n**", classification\_report(y\_test,svc\_y))

svmkernel('rbf')

Accuracy Score 0.6256410256410256

Confusion Matrix

[[122 0]

[ 73 0]]

Classification Report

precision recall f1-score support

0 0.63 1.00 0.77 122

1 0.00 0.00 0.00 73

accuracy 0.63 195

macro avg 0.31 0.50 0.38 195

weighted avg 0.39 0.63 0.48 195

An **AdaBoost**  **classifier** is a meta-estimator that begins by fitting a **classifier** on the original dataset and then fits additional copies of the **classifier** on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent **classifiers** focus more on difficult cases.

***#AdaBoostClassifier***

**from** **sklearn.ensemble** **import** AdaBoostClassifier

abc=AdaBoostClassifier()

abc.fit(x\_train,y\_train)

abc\_y=abc.predict(x\_test)

print("Accuracy Score",accuracy\_score(y\_test,abc\_y))

print("Confusion Matrix**\n**",confusion\_matrix(y\_test,abc\_y))

print("Classification Report**\n**", classification\_report(y\_test,abc\_y))

Accuracy Score 0.8102564102564103

Confusion Matrix

[[99 23]

[14 59]]

Classification Report

precision recall f1-score support

0 0.88 0.81 0.84 122

1 0.72 0.81 0.76 73

accuracy 0.81 195

macro avg 0.80 0.81 0.80 195

weighted avg 0.82 0.81 0.81 195

**Gradient boosting** is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees

***#GradientBoostingClassifier***

**from** **sklearn.ensemble** **import** GradientBoostingClassifier

gbc=AdaBoostClassifier()

gbc.fit(x\_train,y\_train)

gbc\_y=gbc.predict(x\_test)

print("Accuracy Score",accuracy\_score(y\_test,gbc\_y))

print("Confusion Matrix**\n**",confusion\_matrix(y\_test,gbc\_y))

print("Classification Report**\n**", classification\_report(y\_test,gbc\_y))

Accuracy Score 0.8102564102564103

Confusion Matrix

[[99 23]

[14 59]]

Classification Report

precision recall f1-score support

0 0.88 0.81 0.84 122

1 0.72 0.81 0.76 73

accuracy 0.81 195

macro avg 0.80 0.81 0.80 195

weighted avg 0.82 0.81 0.81 195

On the basis of Accuracy score, the best model is done by Logical Regression, AdaBoostClassifier, GradientBoostingClassifier having accuracy b/w 75% -85%.

**GridSearchCV** implements a “fit” and a “score” method. It also implements **“score\_samples”, “predict”, “predict\_proba”, “decision\_function”, “transform” and “inverse\_transform”** if they are implemented in the estimator used.

The parameters of the estimator used to apply these methods are optimized by cross-validated grid-search over a parameter grid.

***#creating parameter list to pass in gridsearch CV***

param\_grid\_lr=[{"C":[0.001, 0.01, 0.05, 0.1, 0.5, 1.0, 10.0]}]

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

GCV=GridSearchCV(lr,param\_grid=param\_grid\_lr,scoring="accuracy",cv=10,refit=**True**,n\_jobs=1)

GCV.fit(x\_train,y\_train)

GCV.best\_estimator\_

GCV\_pred=GCV.best\_estimator\_.predict(x\_test)

print("accuracy = ",r2\_score(y\_test,GCV\_pred)\*100)

accuracy = 12.418594206153166

**Joblib** is a set of tools to provide lightweight pipelining in Python. In particular:

1. transparent disk-caching of functions and lazy re-evaluation (memoize pattern)
2. easy simple parallel computing

**import** **joblib**

joblib.dump(lr,"titanic\_survivor.pkl")

['titanic\_survivor.pkl']

***#PREDICTING THE SURVIVAL RATE WITH A SAMPLE***

td=np.array([15,0,3.500])

td.shape

(3,)

td=td.reshape(1,-1)

td

array([[15. , 0. , 3.5]])

lr.predict(td)

array([1], dtype=int64)

* **TESTING THE MODEL WITH THE GIVEN DATA-SET**

i=np.array(y\_test)

i

array([1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,

0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,

0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,

0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0,

0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,

1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0,

0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1,

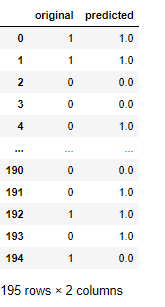
0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0,

0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1],

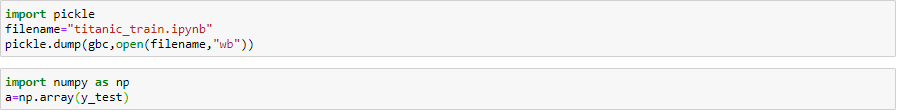
dtype=int64)

df\_com=pd.DataFrame({"original":i,"predicted":pre},index=range(len(i)))

df\_com



* **SAVING THE BEST MODEL**



* **CONCLUSION:**

THE ABOVE OBSERVATIONS SHOW THAT THE MODEL HAS PREDICTED THE VALUES WITH AN ACCURACY OF 75-85%

AND, THE SURVIVAL RATE OF WOMEN’S ARE MORE THAN MEN’S.