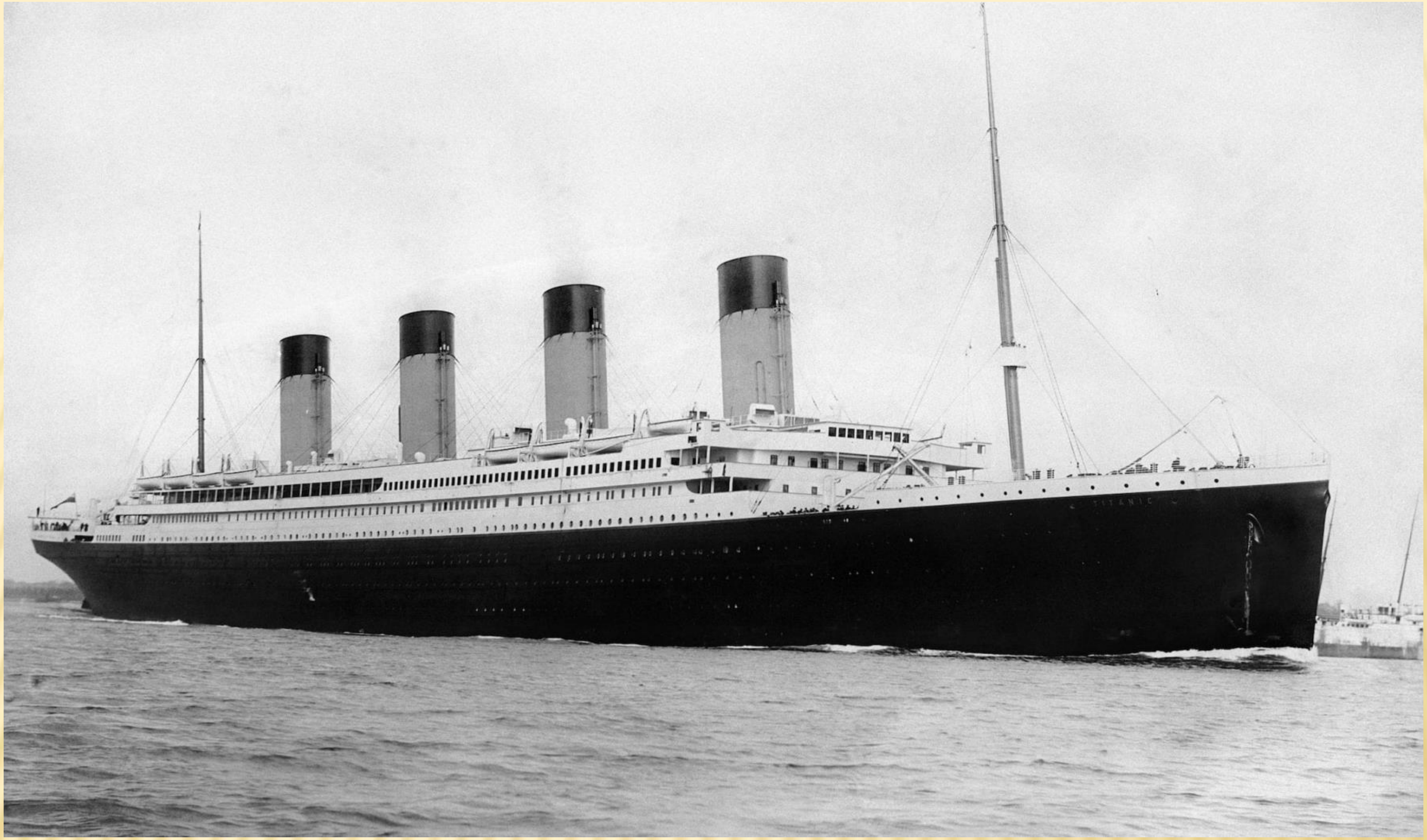


PREDICTING THE SURVIVAL OF TITANIC PASSENGERS

SUBMITTED BY-GURAZIZ SINGH BHATIA

TITANIC PROBLEM

- On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. Translated 32% survival rate.
- One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew.
- Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

AIM OF THE PROJECT

- To predict the number of surviving people from the disaster on basis of given dataset.
- To analyse various features which correlate to the surviving victims such as their Age, Sex, Fare they paid, etc.

REQUIREMENTS

SOFTWARE REQUIREMENTS

- •Jyupiter Notebook

LIBRARIES USED

- Analysis : NumPy, Pandas, Scikit Learn
- Visualization: Matplotlib

STEPS FOR IMPLEMENTATION

- Importing the necessary Libraries
- Importing the Dataset
- Cleaning and analyzing the Dataset
- Converting necessary columns to numeric data
- Building the model
- Using different numbers of algorithms in classification techniques

IMPORTING NECESSARY LIBRARIES

In [207]:

```
# data analysis and wrangling
import pandas as pd
import numpy as np
import random as rnd

# visualization
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

#data spliting
from sklearn.model_selection import train_test_split

# machine learning
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix,accuracy_score
```

IMPORTING DATASET

In [170]:
```
df=pd.read_csv('train.csv')
```

In [171]:
```
df.head()
```

Out[171]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [172]:
```
df.shape
```

Out[172]: (891, 12)

ANALYSING THE DATA BY DESCRIBING

In [208]:
```
df.describe()
```

Out[208]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	0.647587	29.699118	0.523008	0.381594	32.204208	1.536476
std	257.353842	0.486592	0.836071	0.477990	13.002015	1.102743	0.806057	49.693429	0.791503
min	1.000000	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	0.000000	22.000000	0.000000	0.000000	7.910400	1.000000
50%	446.000000	0.000000	3.000000	1.000000	29.699118	0.000000	0.000000	14.454200	2.000000
75%	668.500000	1.000000	3.000000	1.000000	35.000000	1.000000	0.000000	31.000000	2.000000
max	891.000000	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	512.329200	2.000000

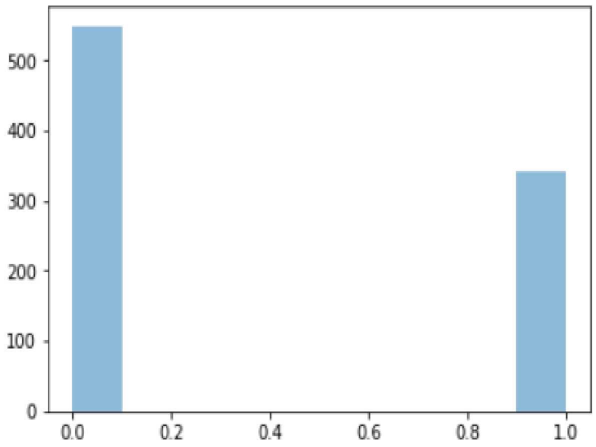
VISUALISING THE TARGET COLUMN

In [214]:
```
df.Survived.value_counts()
```

Out[214]:
```
0    549
1    342
Name: Survived, dtype: int64
```

In [215]:
```
plt.hist(df.Survived,alpha=0.5)
```

Out[215]:
```
(array([549.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0., 342.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 10 Patch objects>)
```



Approx. 38% survived the disaster according to the dataset

FINDING RELATION BETWEEN TARGET COLUMN AND OTHER FEATURES

- Sex, Pclass, Parch, Fare

In [221]:
```
df[['Sex','Survived']].groupby('Sex',as_index=False).mean()
```
Out[221]:

	Sex	Survived
0	female	0.742038
1	male	0.188908

In [222]:
```
df[['Parch','Survived']].groupby('Parch',as_index=False).mean()
```
Out[222]:

	Parch	Survived
0	0	0.343658
1	1	0.550847
2	2	0.500000
3	3	0.600000
4	4	0.000000
5	5	0.200000
6	6	0.000000

In [251]:
```
df[['Pclass','Survived']].groupby('Pclass',as_index=False).mean()
```
Out[251]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

In [223]:
```
df.Fare.agg(['min','max','mean','median','count'])
```
Out[223]:
```
min         0.000000
max       512.329200
mean       32.204208
median     14.454200
count     891.000000
Name: Fare, dtype: float64
```

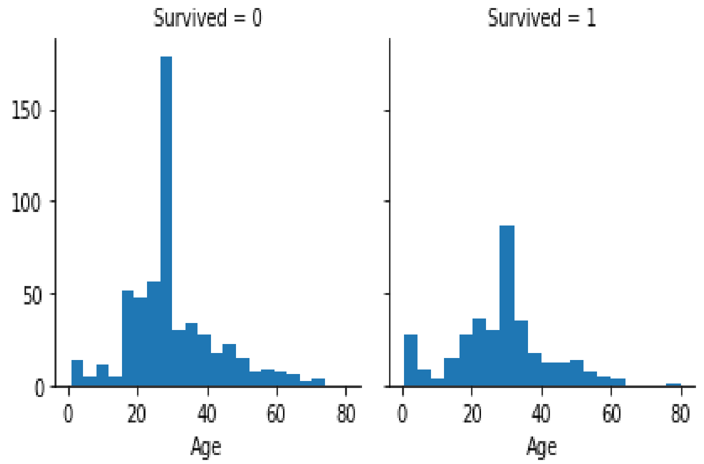
Age

```
In [224]: df.Age.agg(['min','max','mean','median','count'])

Out[224]: min         0.420000
          max        80.000000
          mean       29.699118
          median     29.699118
          count     891.000000
          Name: Age, dtype: float64
```

```
In [226]: g = sns.FacetGrid(df, col='Survived')
          g.map(plt.hist, 'Age', bins=20)

Out[226]: <seaborn.axisgrid.FacetGrid at 0x13e2f093860>
```



You can see that men have a high probability of survival when they are between 18 and 30 years old.

Embarked

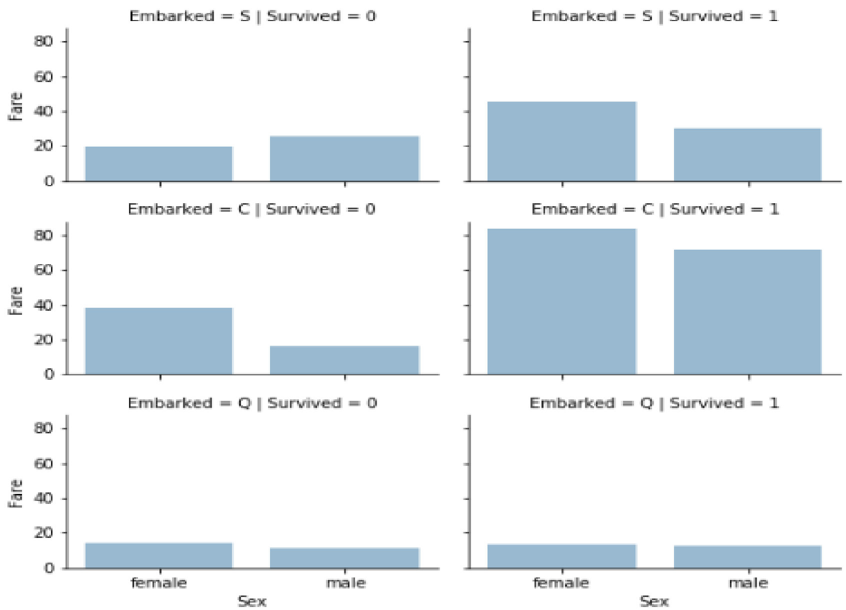
```
In [227]: df[['Embarked','Survived']].groupby('Embarked',as_index=False).sum()
Out[227]:
```

	Embarked	Survived
0	C	93
1	Q	30
2	S	219

```
In [228]: grid = sns.FacetGrid(df, row='Embarked', col='Survived', size=2.2, aspect=1.6)
          grid.map(sns.barplot, 'Sex', 'Fare', alpha=.5, ci=None)
          grid.add_legend()

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` paramter has been renamed to `heigh
t`; please update your code.
  warnings.warn(msg, UserWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:715: UserWarning: Using the barplot function without specifying
`order` is likely to produce an incorrect plot.
  warnings.warn(warning)

Out[228]: <seaborn.axisgrid.FacetGrid at 0x13e2f126438>
```



ENCODING DATA TO NUMERIC (FEATURE SCALING)

In [229]:
```
#Converting data to Numeric Using Label Encoding
```

In [230]:
```
from sklearn.preprocessing import LabelEncoder
labelEncoderoutput=LabelEncoder();
df.Sex=labelEncoderoutput.fit_transform(df.Sex)
df.Embarked=labelEncoderoutput.fit_transform(df.Embarked)
df.head()
```

Out[230]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	1	22.0	1	0	A/5 21171	7.2500	NaN	2
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	0	38.0	1	0	PC 17599	71.2833	C85	0
2	3	1	3	Heikkinen, Miss. Laina	0	26.0	0	0	STON/O2. 3101282	7.9250	NaN	2
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	0	35.0	1	0	113803	53.1000	C123	2
4	5	0	3	Allen, Mr. William Henry	1	35.0	0	0	373450	8.0500	NaN	2

In [231]:
```
Xt = df[['Pclass','Sex','Age','SibSp','Parch','Fare','Embarked']]
Yt = df["Survived"]
Xt.head()
```

Out[231]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	2
1	1	0	38.0	1	0	71.2833	0
2	3	0	26.0	0	0	7.9250	2
3	1	0	35.0	1	0	53.1000	2
4	3	1	35.0	0	0	8.0500	2

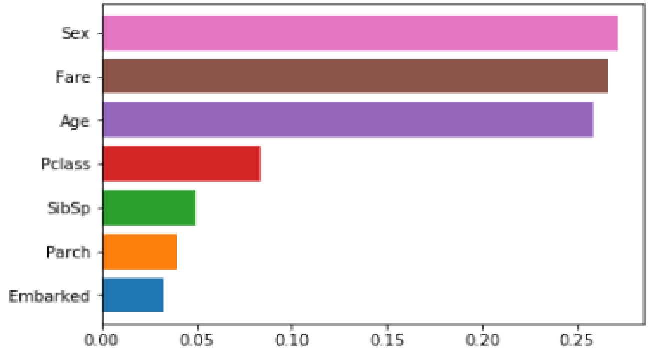
FEATURE SELECTION

In [ ]:
```
#Feature Selection
```

In [233]:
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
rf.fit(Xt,Yt)
feature_importance={}
for i in range(7):
 feature_importance[Xt.columns.values[i]]=rf.feature_importances_[i]
print(feature_importance)
pd.Series(rf.feature_importances_,Xt.columns).sort_values(ascending=True).plot.barh(width=0.8)
```

```
{'Pclass': 0.08374927199198436, 'Sex': 0.27115473688895664, 'Age': 0.25854197576281107, 'SibSp': 0.04919517737210158, 'Parch': 0.03893123544834381, 'Fare': 0.2659575772525442, 'Embarked': 0.03247002528325831}
```

Out[233]: <matplotlib.axes._subplots.AxesSubplot at 0x13e2f378d30>



In [234]:
```
xnew=Xt[['Fare','Sex','Age']]
xnew.head()
```

Out[234]:

	Fare	Sex	Age
0	7.2500	1	22.0
1	71.2833	0	38.0
2	7.9250	0	26.0
3	53.1000	0	35.0
4	8.0500	1	35.0

BUILDING MODEL AND CALCULATING PREDICTIONS

```
In [235]: xnew.ndim #checking demensionality
Out[235]: 2

In [236]: from sklearn.model_selection import train_test_split    #splitting data in training and testing data
          xtrain,xtest,ytrain,ytest = train_test_split(xnew.values,df.Survived.values,test_size=0.3,random_state=0)

In [ ]:

In [237]: print(xtrain)

          [[26.55       1.         51.        ]
           [76.7292     0.         49.        ]
           [46.9        1.          1.        ]
           ...
           [ 7.7333     1.         29.69911765]
           [17.4        0.         36.        ]
           [39.         1.         60.        ]]

In [238]: # Logistic Regression

In [239]: from sklearn.linear_model import LogisticRegression
          log = LogisticRegression(C=1,class_weight={1:5})
          log.fit(xtrain,ytrain)
          pred=log.predict(xtest)  #prediction using logistic regression model

          C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarning: Default solver will be changed
          to 'lbfgs' in 0.22. Specify a solver to silence this warning.
            FutureWarning)

In [240]: from sklearn.metrics import confusion_matrix,accuracy_score
          cmlog=confusion_matrix(ytest,pred)
          print(cmlog)
          acclog=accuracy_score(ytest,pred)
          print(acclog)

          [[ 64 104]
           [  4  96]]
          0.5970149253731343

In [241]: # Decision Tree Classification

In [242]: from sklearn.tree import DecisionTreeClassifier
          cls=DecisionTreeClassifier(criterion='entropy')
          cls.fit(xtrain,ytrain)
          predDecision=cls.predict(xtest) #prediction using Decision Tree Classication model

          cmDecision=confusion_matrix(ytest,predDecision)
          print(cmDecision)
          accDe=accuracy_score(ytest,predDecision)
          print(accDe)

          [[139  29]
           [ 30  70]]
          0.7798507462686567
```

```
In [243]: # Random Forest Classification
```

```
In [244]: from sklearn.ensemble import RandomForestClassifier
randomClassifier=RandomForestClassifier(n_estimators=100)
randomClassifier.fit(xtrain,ytrain)
predRandom=randomClassifier.predict(xtest) #prediction using Random Forest Classication model

cmRandom=confusion_matrix(ytest,predRandom)
print(cmRandom)
accRandom=accuracy_score(ytest,predRandom)
print(accRandom)
```

```
[[144  24]
 [ 30  70]]
0.7985074626865671
```

```
In [245]: # Support Vector Machines
from sklearn.svm import SVC
svc = SVC()
svc.fit(xtrain, ytrain)
predSVC = svc.predict(xtest)  #prediction using SVC model

cmSVC=confusion_matrix(ytest,predSVC)
print(cmSVC)
accSVC=accuracy_score(ytest,predSVC)
print(accSVC)
```

```
[[146  22]
 [ 60  40]]
0.6940298507462687
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid t
his warning.
  "avoid this warning.", FutureWarning)
```

```
In [246]: # KNN Classification
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(xtrain, ytrain)
predKNN = knn.predict(xtest) #prediction using KNN Classification  model

cmKNN=confusion_matrix(ytest,predKNN)
print(cmKNN)
accKNN=accuracy_score(ytest,predKNN)
print(accKNN)
```

```
[[127  41]
 [ 48  52]]
0.667910447761194
```

In [247]:

```
# Gaussian Naive Bayes

from sklearn.naive_bayes import GaussianNB
gaussian = GaussianNB()
gaussian.fit(xtrain, ytrain)
predNaive = gaussian.predict(xtest)  #prediction using Gausian Naive Bayes Classification  model

cmNaive=confusion_matrix(ytest,predNaive)
print(cmNaive)
accNaive=accuracy_score(ytest,predNaive)
print(accNaive)
```

```
[[137  31]
 [ 28  72]]
0.7798507462686567
```

MAPPING PREDICTIONS AND ANALYSING

In [248]:
```
models = pd.DataFrame({
    'Model': ['Support Vector Machines', 'KNN', 'Logistic Regression',
              'Random Forest','Decision Tree','Naive Bayes'],
    'Score': [accSVC, accKNN, acclog,
              accRandom,accDe,accNaive]})
models.sort_values(by='Score', ascending=False)
```

Out[248]:

	Model	Score
3	Random Forest	0.798507
4	Decision Tree	0.779851
5	Naive Bayes	0.779851
0	Support Vector Machines	0.694030
1	KNN	0.667910
2	Logistic Regression	0.597015

CONCLUSION

- We can conclude that Random forest classification model gives us best prediction of 79.85 %or 80% approx, whereas naïve bayes and decision tree classification models have same prediction accuracy and less than random forrest.
- Here prediction is made on the features fare ,sex and age wrt to survived which are derived from feature selection.