Titanic: Machine Learning from Disaster

PROJECT BY: SHUBHNEET SANDHU

Introduction

- The sinking of the Titanic is one of the most infamous shipwrecks in history.

- On April 15, 1912, during her maiden voyage, the widely considered "unsinkable" RMS Titanic en route to New York City from Southampton, England, sank after colliding with an iceberg in Atlantic Ocean. Unfortunately, there weren't enough lifeboats for everyone onboard, resulting in the death of about 1,500 out of 2224 passengers and ship personnel.

Objective

- While there was some element of luck involved in surviving, it seems some groups of people were more likely to survive than others.

- The objective of this project is to build a classification model to successfully determine whether a passenger survived or not.

Steps

- Importing necessary libraries
- Importing the dataset
- Data Cleaning and Preprocessing
- Data Visualization
- Feature Selection
- Data Analysis Using Different Classification Models
- Result
- Conclusion

Importing necessary libraries

- Data Analyzing -
  : Pandas, NumPy
- Data Cleaning/Preprocessing -
  : Label Encoder, MinMax Scaler
- Data Splitting -: train_test_split
- Feature Selection -: Variance Threshold, SelectKBest, chi2
- Classification Models -: Logistic Regression, Decision Tree, Random Forest, KNN Classifier
- Model Testing -: Confusion matrix, Accuracy score

```
In [ ]:
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import VarianceThreshold
from sklearn.preprocessing import MinMaxScaler
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
```

Importing Dataset

```
In [2]: df = pd.read_csv("./Downloads/train.csv")
```

```
In [4]: df.head()
```

Out[4]:

	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 [5]: df.shape
Out[5]: (891, 12)
```

- The dataset consists of 891 rows and 12 columns
- Target Variable – Survived
- Features – PassengerId, PClass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked

Data Preprocessing

- Filled missing values with median in Age column
- Dropped missing values in Embarked Column
- Label Encoded columns 'Sex' and 'Embarked'
- Removed columns- Name, Cabin, Ticket and PassengerId as:

1. Name, PassengerId and Ticket are unique for each passenger so it won't contribute much to the target col.
2. Cabin has large missing values and substituting these values with mode value will imbalance the dataset.

```
In [6]: df.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
```

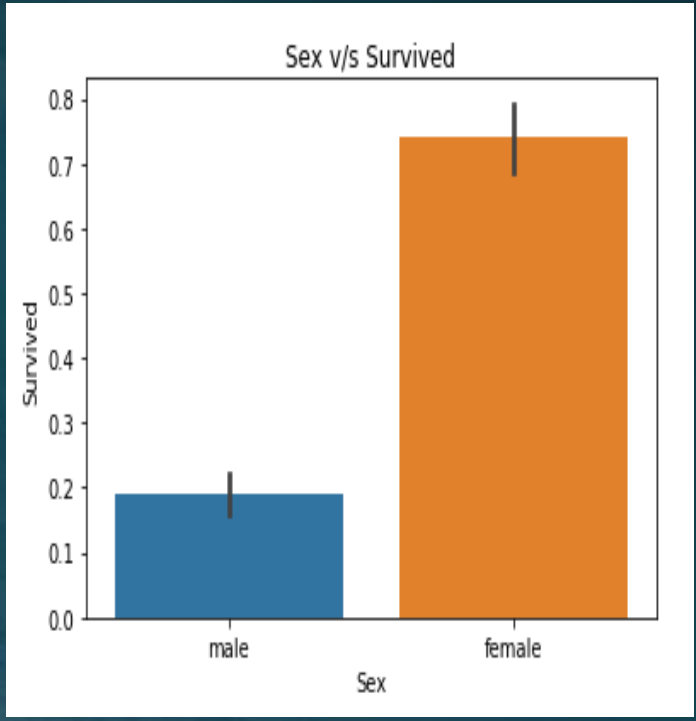
```
In [16]: df
```

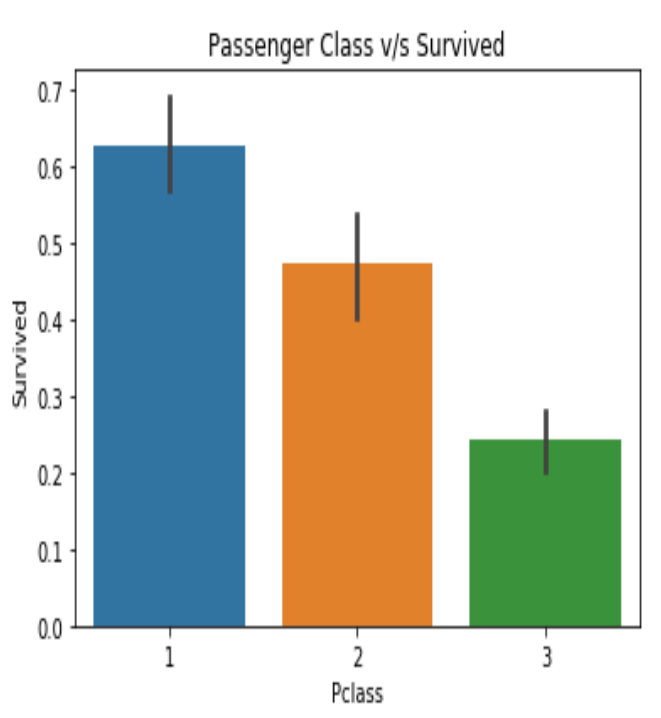
Out[16]:

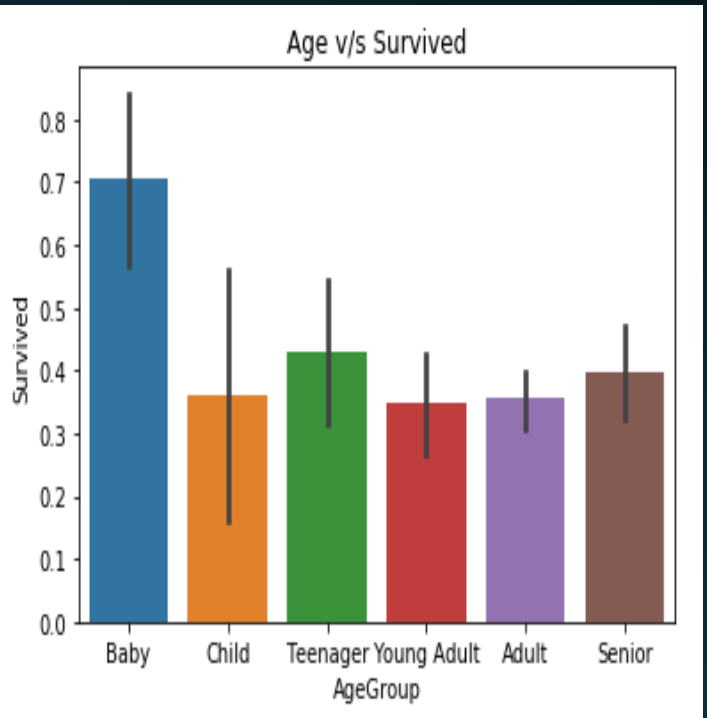
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	1	22.0	1	0	7.2500	2
1	1	1	0	38.0	1	0	71.2833	0
2	1	3	0	26.0	0	0	7.9250	2
3	1	1	0	35.0	1	0	53.1000	2
4	0	3	1	35.0	0	0	8.0500	2
...	...	...	...	...	...	...	...	...
886	0	2	1	27.0	0	0	13.0000	2
887	1	1	0	19.0	0	0	30.0000	2
888	0	3	0	28.0	1	2	23.4500	2
889	1	1	1	26.0	0	0	30.0000	0
890	0	3	1	32.0	0	0	7.7500	1

889 rows × 8 columns

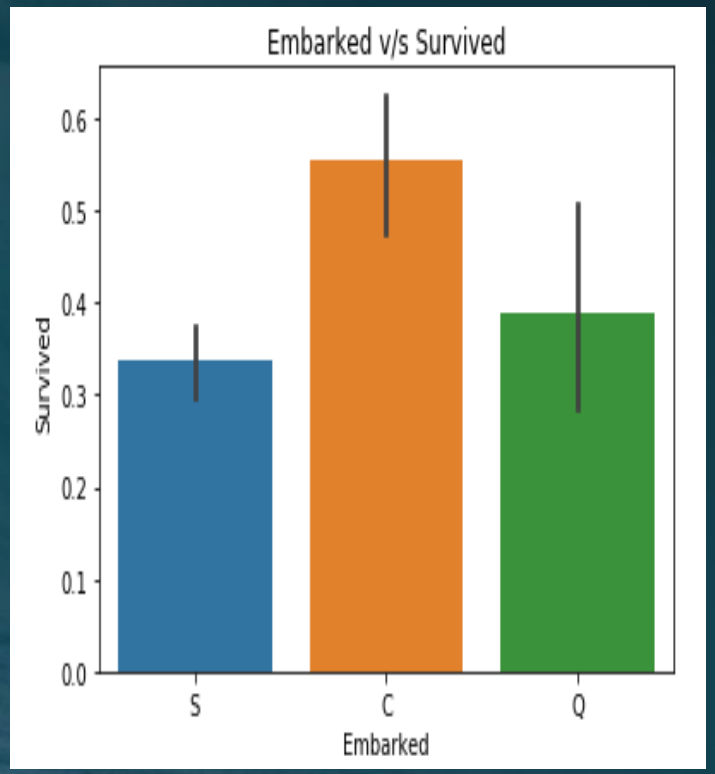
Data Visualization

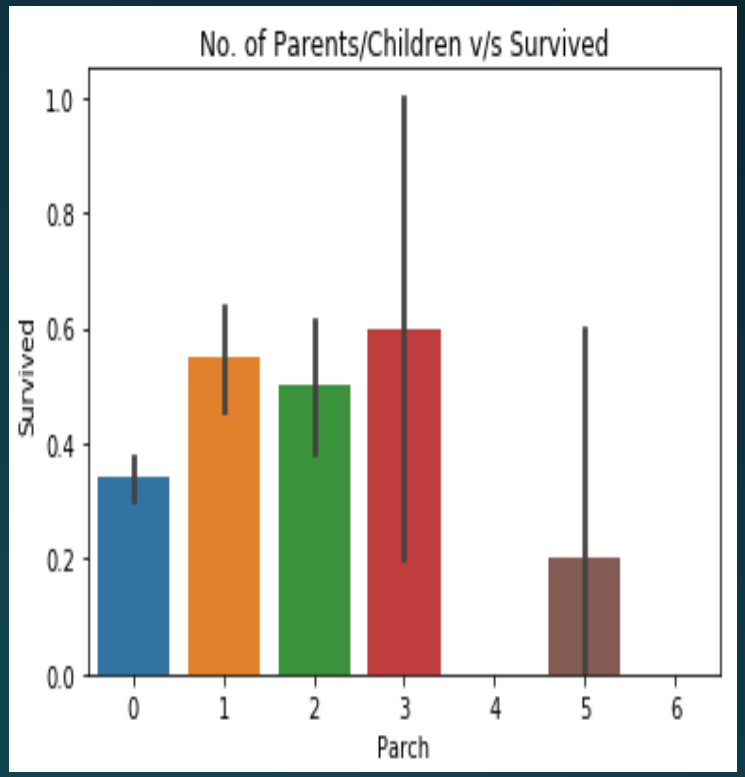


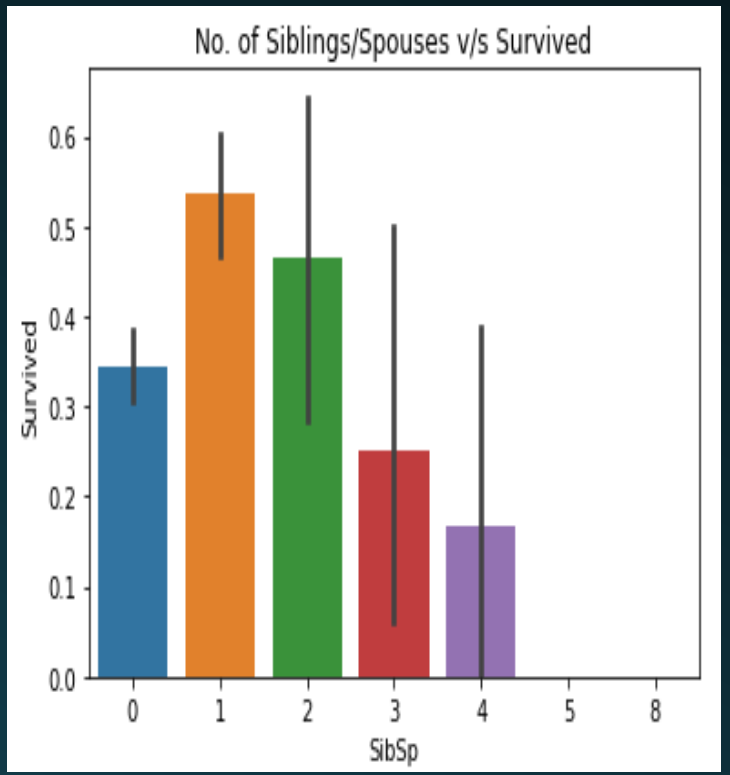




Data Visualization (Contd.)







Feature Selection

Before feature selection, the dataset is split into training and test set because we only select features based on the information from the training set and use test set to evaluate the performance of the feature selection and the model. Thus the information from the test set cannot be seen while we conduct feature selection and train the model.

In [17]:
```
X = df.iloc[:, 1:8]
Y = df.iloc[:, 0]
```

In [18]:
```
# Train Test Split

from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test =train_test_split(X, Y, test_size = 0.2, random_state = 0)
```

Feature Selection Methods

The feature selection methods used for this project are:

- Variance Threshold
- Chi- Square Test

Variance Threshold

This method removes features with variation below a certain cutoff i.e. features having low variance . Here the cutoff value is taken as 0.5. It is used here because if a feature doesn't vary much within itself, it generally has very little predictive power.

```
from sklearn.feature_selection import VarianceThreshold

mdlsel = VarianceThreshold(threshold=0.5)
mdlsel.fit(X)
ix = mdlsel.get_support()
data1 = pd.DataFrame(mdlsel.transform(X), columns = X.columns.values[ix])
data1.head()
```

]:

	Pclass	Age	SibSp	Parch	Fare	Embarked
0	3.0	22.0	1.0	0.0	7.2500	2.0
1	1.0	38.0	1.0	0.0	71.2833	0.0
2	3.0	26.0	0.0	0.0	7.9250	2.0
3	1.0	35.0	1.0	0.0	53.1000	2.0
4	3.0	35.0	0.0	0.0	8.0500	2.0

Chi – Square Test

The **Chi Square** statistic test is commonly used for **testing** relationships between categorical variables.

```
In [21]: from sklearn.feature_selection import SelectKBest, chi2

         bestfeatures = SelectKBest(chi2, k = 5)
         fit = bestfeatures.fit(X_train, Y_train)
```

From the chi2 test, it can be observed that Fare, Age, Sex and Pclass are the most important features for prediction.

So these 4 features will be used to train the model.

```
In [24]: featureScores
```

Out[24]:

	Specs	Score
0	Pclass	22.651692
1	Sex	82.415412
2	Age	45.473701
3	SibSp	0.529343
4	Parch	10.356638
5	Fare	4054.492164
6	Embarked	10.002081

Data Analysis Models

- Logistic Regression
- Decision Tree Classification
- Random Forest
- KNN Classification

Logistic Regression

```
In [35]: from sklearn.linear_model import LogisticRegression
         regressor = LogisticRegression()
         regressor.fit(X_train, Y_train)

Out[35]: LogisticRegression()
```

```
# Confusion Matrix

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, Y_pred)
```

```
cm
```

```
[40]: array([[83, 22],
             [31, 42]], dtype=int64)
```

```
# Accuracy Score

from sklearn.metrics import accuracy_score
acc = accuracy_score(Y_test, Y_pred)
```

```
acc
```

```
[42]: 0.702247191011236
```

Decision Tree

```
]:  # Decision Tree

    from sklearn.tree import DecisionTreeClassifier
    classifier = DecisionTreeClassifier()
    classifier.fit(X_train, Y_train)
```

```
[43]: DecisionTreeClassifier()
```

```
# Confusion Matrix

from sklearn.metrics import confusion_matrix
cm1 = confusion_matrix(Y_test, Y_pred1)
```

```
cm1
```

```
46]: array([[86, 19],
            [26, 47]], dtype=int64)
```

```
:  # Accuracy Score

   from sklearn.metrics import accuracy_score
   acc1 = accuracy_score(Y_test, Y_pred1)
   acc1
```

```
[47]: 0.7471910112359551
```

Random Forest

In [48]:
```
# RandomForest Classification
from sklearn.ensemble import RandomForestClassifier
randomClassifier = RandomForestClassifier()
randomClassifier.fit(X_train, Y_train)
```

Out[48]:
```
RandomForestClassifier()
```

```
# Confusion Matrix

from sklearn.metrics import confusion_matrix
cm2 = confusion_matrix(Y_test, Y_pred2)
cm2
```

]:
```
array([[90, 15],
       [21, 52]], dtype=int64)
```

```
# Accuracy Score

from sklearn.metrics import accuracy_score
acc2 = accuracy_score(Y_test, Y_pred2)
acc2
```

]:
```
0.797752808988764
```

KNN Classifier

The training dataset was scaled using MinMax Scaler before fitting it into KNN to get better results.

In [53]:
```
# KNN Classification

from sklearn.neighbors import KNeighborsClassifier
kClassifier = KNeighborsClassifier(n_neighbors = 14, metric = 'euclidean')
kClassifier.fit(X_train, Y_train)
```

Out[53]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=14)
```

```
# Confusion Matrix

from sklearn.metrics import confusion_matrix
cm3 = confusion_matrix(Y_test, Y_pred3)
cm3
```

]:
```
array([[97,  8],
       [37, 36]], dtype=int64)
```

```
# Accuracy Score

from sklearn.metrics import accuracy_score
acc3 = accuracy_score(Y_test, Y_pred3)
acc3
```

]:
```
0.7471910112359551
```

Model Result Matrix

Prediction Model	Accuracy Score
Logistic Regression	0.702247191011236
Decision Tree	0.747191011235955 1
Random Forest	0.797752808988764
KNN Classifier	0.747191011235955 1

Result

```
result
```

47]:

	PClass	Sex	Age	Fare	Survived
0	3.0	0.0	14.0	7.8542	1
1	3.0	1.0	28.0	69.5500	0
2	1.0	0.0	36.0	120.0000	1
3	1.0	1.0	36.0	78.8500	1
4	3.0	0.0	63.0	9.5875	0
...	...	...	...	...	...
173	2.0	1.0	27.0	13.0000	0
174	2.0	1.0	31.0	13.0000	0
175	3.0	1.0	9.0	31.3875	0
176	1.0	1.0	32.0	30.5000	0
177	3.0	0.0	22.0	7.7500	1

178 rows × 5 columns

INDEX:

Sex Column:

0 :- Female

1:- Male

Survived Column:

0:- Not Survived

1:- Survived

Conclusion

- Comparing the accuracy of each model, it can be deduced that Random Forest is giving the best prediction followed by Decision Tree and KNN Classifier.
- PClass, Age, Sex, and Fare are the features having major effect on survival of a person. Women, children and first class passengers as well as small-size families had a better chance at survival.
- The accuracy of Random Forest is `0.7978%.`

Acknowledgement

I would like to express my gratitude to EICT Academy, IIT Kanpur, for providing me an opportunity to do project work in Machine Learning/AI and giving me support and guidance which made me complete my project duly.

Secondly, I would like to thank my parents and my university- DCRUST, Murthal for bringing this course into my knowledge.

So with due regards, I express my sincere thanks to them.