



WOULD YOU SURVIVE THE TITANIC?

Machine Learning Has The Answer

Final Project Report

Data Analytics

Team Members:

Tigmanshu Chaudhary(TIC48)

Charu Sreedharan(CHS263)

Varun Nair(VAN17)

Anirban Sen(ANS331)

OBJECTIVE

To predict and analyze which people were likely to survive the titanic tragedy based on the given features by employing a machine learning technique.

PROGRAMMING LANGUAGE

Python

DATASET

- The dataset is provided on Kaggle and like most datasets it is divided into train and test data.
- The training data is to be used to train the model and contains 11 features as well as the outcome of the passengers(the Target variable).
- The test data is to finally test our final model on this unseen data as.This data doesn't contain passenger outcomes and it is the job of the model to predict these outcomes.

Description of the dataset:

- The training set consists of records of 891 passengers
- The test set consists of records of 418 passengers
- There are 11 attributes for each passenger which are:

Feature Set:

- PassengerId→this is the rowID of the passenger details.
- Pclass→indicates the economic class or status of the passenger(where 1 refers to first class, 2 refers to second class and so on).
- Name→this is the name of the passenger.
- Sex→ this is the gender of the passenger.
- Age→Age of the passenger.
- SibSp→this indicates if there is a sibling or spouse for a passenger.
- Parch→indicates number of parents plus number of children.
- Ticket→this is the serial number of the ticket.
- Fare→indicates the value of the ticket.
- Cabin→indicates the cabin number of the passenger.
- Embarked→indicates the port from which the passenger embarked
C(Cherbourg),S(Southampton),Q(Queenstown)

Target Variable:

- Survived→indicates if the passenger survived(0 indicates that the passenger died and 1 indicates that the passenger survived)

DATA ENGINEERING

We had to make some changes to the dataset as the original dataset contains missing values for rows and also because not all features might be important for prediction.

A sample of the training dataset:

Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	3	Braund, M	male	22	1	0	A/5 21171	7.25		S
1	1	Cumings, M	female	38	1	0	PC 17599	71.2833	C85	C
1	3	Heikkinen, M	female	26	0	0	STON/O2.	7.925		S
1	1	Futrelle, M	female	35	1	0	113803	53.1	C123	S
0	3	Allen, Mr.	male	35	0	0	373450	8.05		S
0	3	Moran, Mr	male		0	0	330877	8.4583		Q
0	1	McCarthy, M	male	54	0	0	17463	51.8625	E46	S
0	3	Palsson, M	male	2	3	1	349909	21.075		S
1	3	Johnson, M	female	27	0	2	347742	11.1333		S
1	2	Nasser, Mr	female	14	1	0	237736	30.0708		C
1	3	Sandstrom, M	female	4	1	1	PP 9549	16.7	G6	S
1	1	Bonnell, M	female	58	0	0	113783	26.55	C103	S
0	3	Saunders, M	male	20	0	0	A/5. 2151	8.05		S

Pre-processing of data:

Out of the given features we omitted the following features for our predictions:

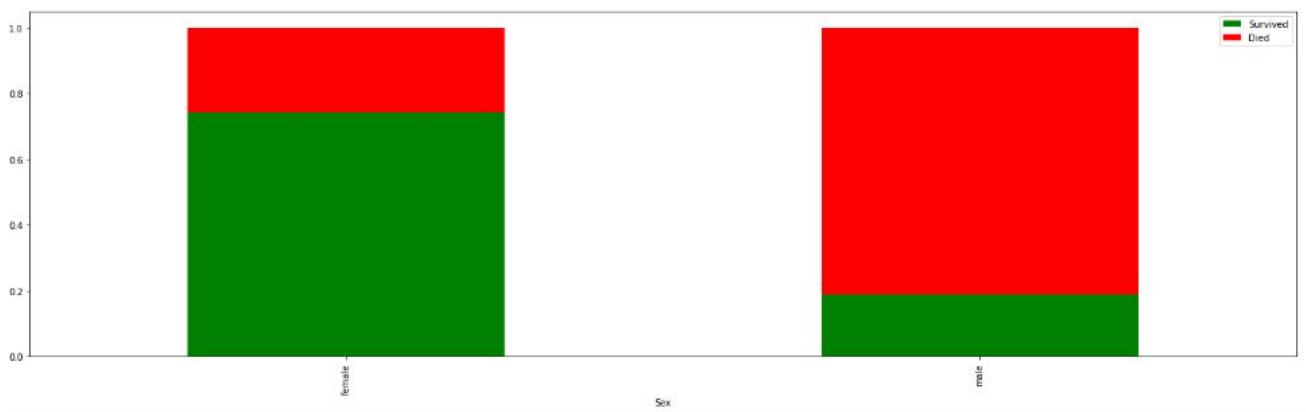
- Name: We assumed that name of a person doesn't help in predicting the survival of a passenger.
- Ticket: We removed ticket as this is just the ticket number of the passenger and doesn't help in predicting survival. Also this is just a string value which can't be encoded to any meaningful representation.
- Cabin: We removed this feature as it doesn't help in predicting the survival of passengers and also because a majority of data records have missing cabin values.
- Embarked: The port of embarking doesn't help in predicting the survival of passengers.
- We found that age feature had many missing values. So we replaced the null values of age with the mean of the age column.
- We also found that fare column had missing values. So, we replace null fare values with the mean of the column.
- We label encode sex column as 0 and 1 for male and female.

Feature Engineering:

FamilySize: We created this new feature from existing features SibSp and Parch. The logic behind this feature was that the number of siblings/spouses and parent/children would matter in the survival of the passenger, hence the family size would matter.

$$\text{FamilySize} = \text{SibSp} + \text{Parch} + 1.$$

Title: We are extracting title (salutation) from the name of the passenger. Then we are encoding these titles into numerical values. While encoding, we assign maximum weight to the most respected social title like Captain, Duchess, Count, etc. Then, we assign the next highest weights to children, women and men respectively. This is because in our analysis shown below, we found that men are more likely to die than women. Also, we assume that respectable titles will be given the maximum preference during evacuation.



Data Split

After deciding on our final features, our first job was to split the training dataset for training, validation and testing purposes

We split the training dataset into two datasets :

- Trainval dataset: to train and validate different models using cross validations.
- Test dataset: to check the accuracy of the model.

Scaling

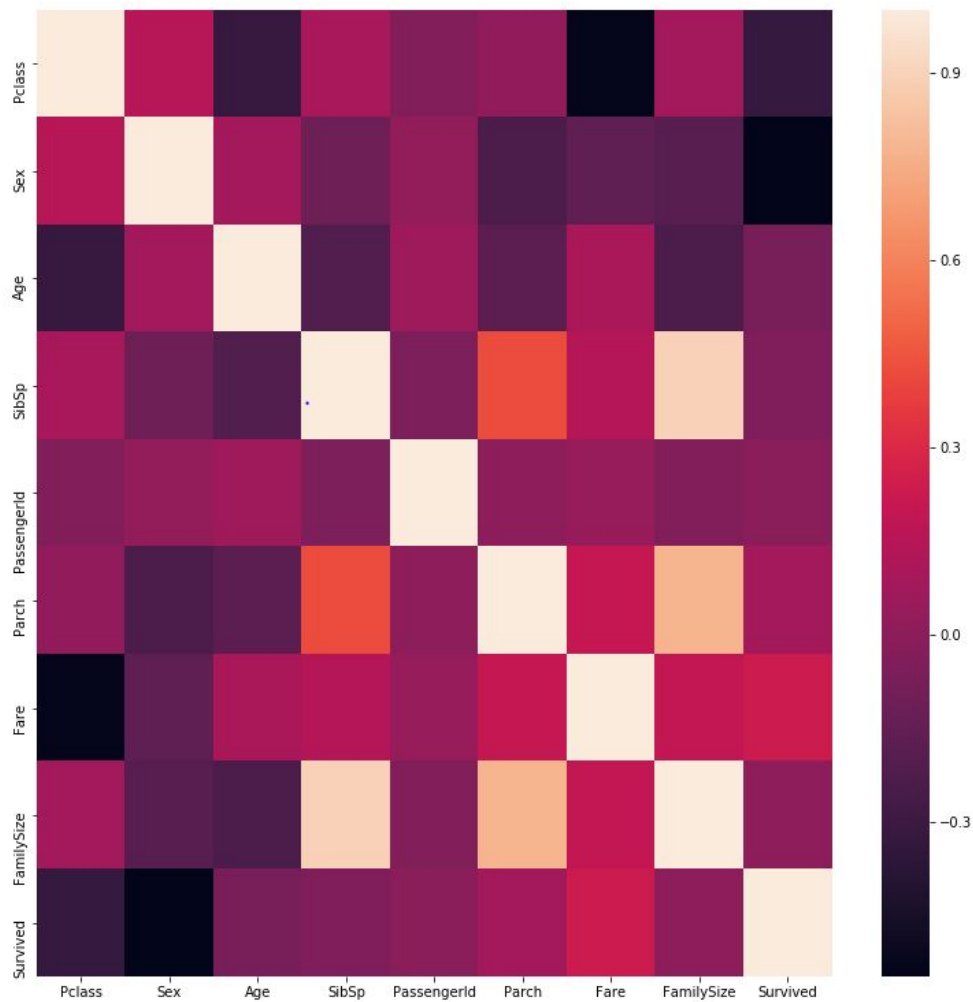
Standard Scaler was used for scaling the dataset for better accuracy and efficiency.

Data Analysis:

Before diving into training models we would first like to visualize some features and see how they relate to the survival of passengers as this might help us in our prediction analysis and understanding of features.

We start by making the correlation matrix:

```
Out[66]: <matplotlib.axes._subplots.AxesSubplot at 0x20363cdea20>
```



- We can see that sex is negatively correlated with Survival.
- We can see that Pclass is negatively correlated with Survival.
- We can see that FamilySize is negatively correlated with Survival which means that smaller the family size, better is the chance of survival of the passenger.

We further thought of checking these results by visualizing these features individually with the Target variable.

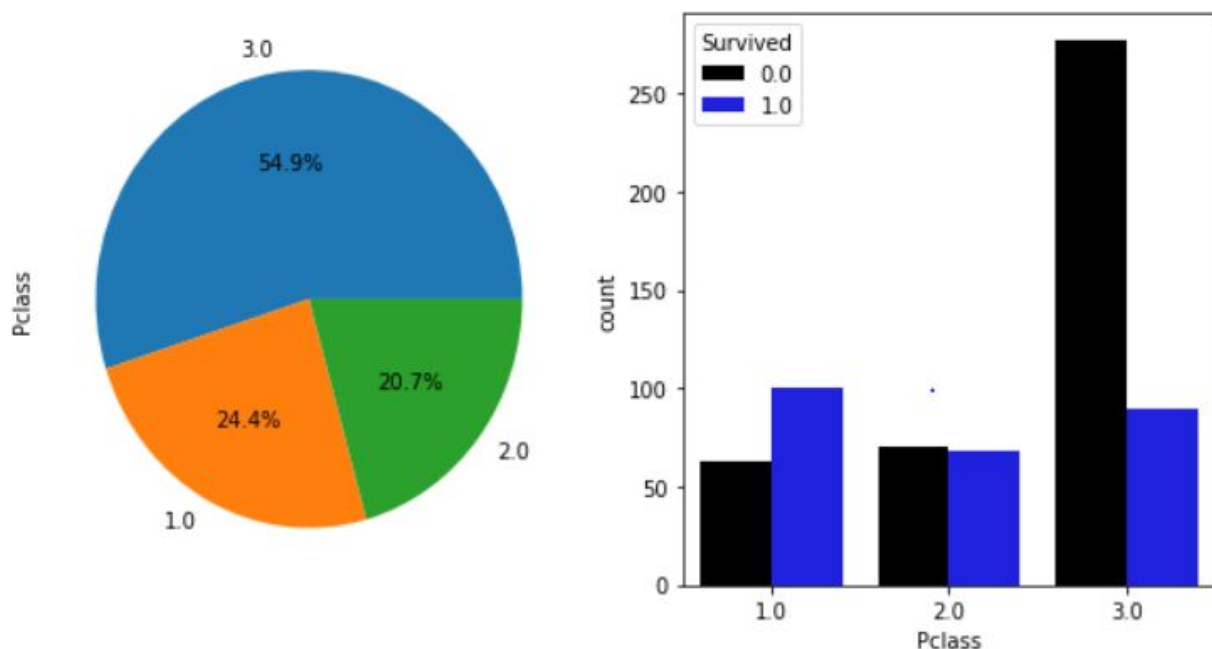
We used seaborn library of python for our visualisations:

```
import matplotlib.pyplot as plt
import seaborn as sns
def cat_plot(df, feature_name, target_name, palettemap):
    fig, [axis0,axis1] = plt.subplots(1,2,figsize=(10,5))
    df[feature_name].value_counts().plot.pie(autopct='%1.1f%%',ax=axis0)
    sns.countplot(x=feature_name, hue=target_name, data=df,
                  palette=palettemap,ax=axis1)
    plt.show()

survival_palette = {0: "black", 1: "blue"}
cat_plot(dataset, 'Pclass', 'Survived', survival_palette)
```

⇒ PClass:

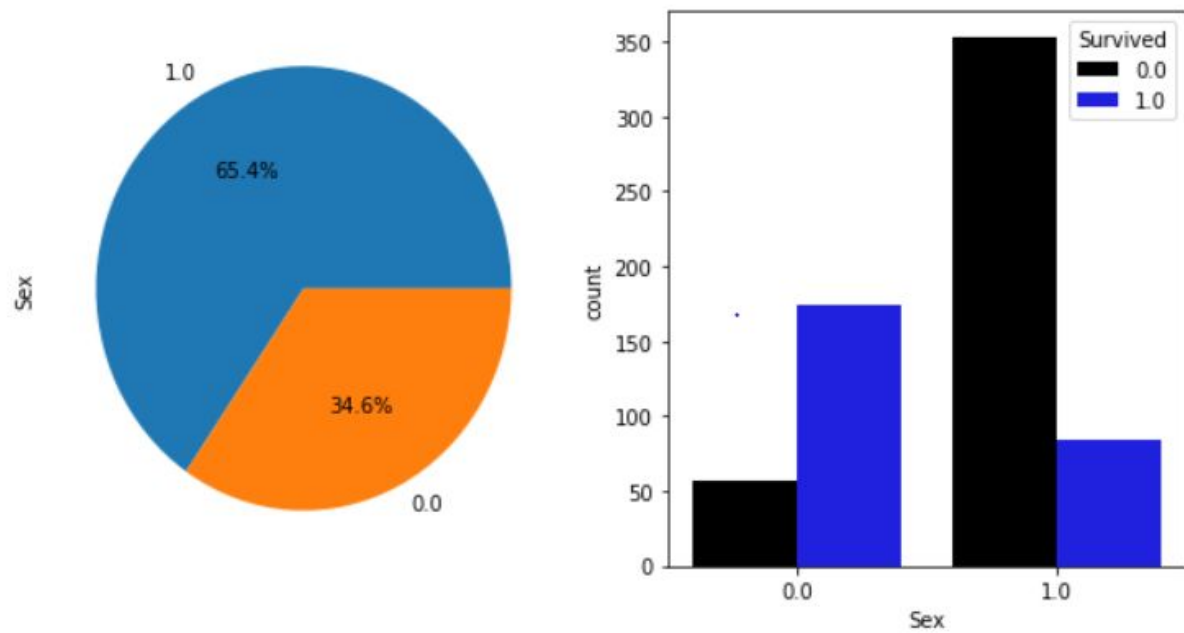
```
cat_plot(dataset, 'Pclass', 'Survived', survival_palette)
```



We can see clearly from the graph that amongst all the classes, only the first class people have more survivors than deaths. This indicates that a passenger has more chance of survival if he/she belongs to the first class.

⇒ Sex:

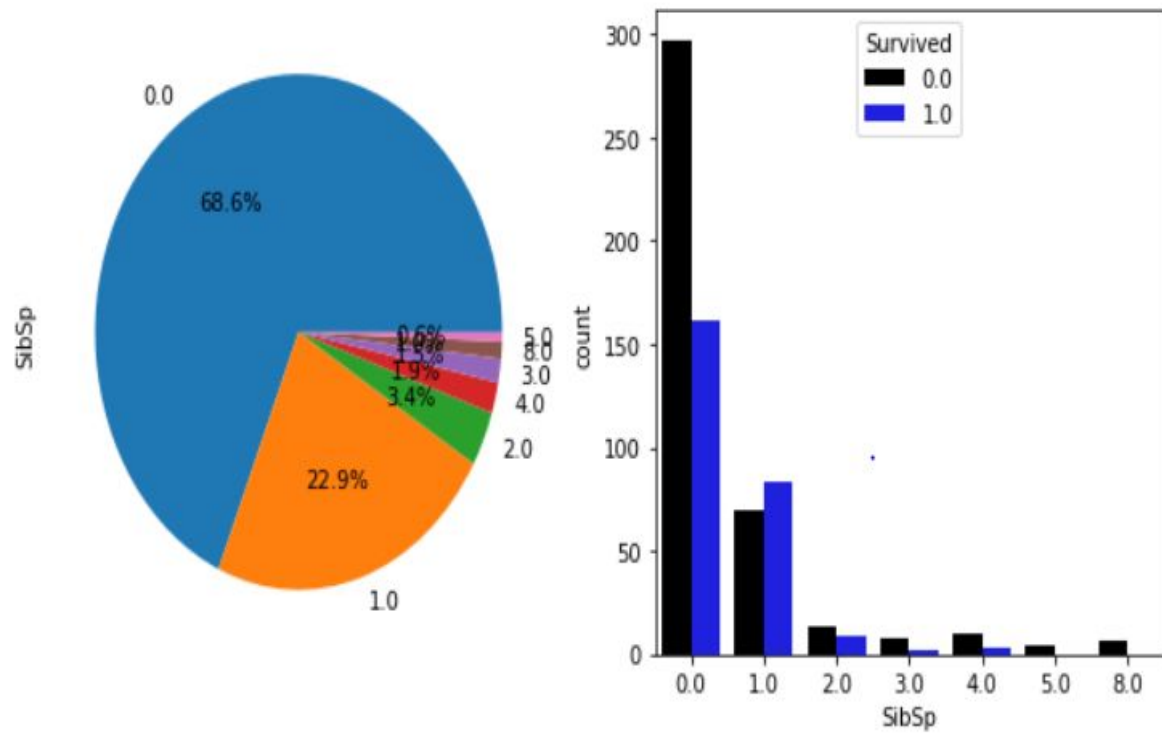
```
cat_plot(dataset, 'Sex', 'Survived', survival_palette)
```



Here 1 indicates a female passenger and 0 indicates a male passenger which means that females are more likely to survive than male passengers.

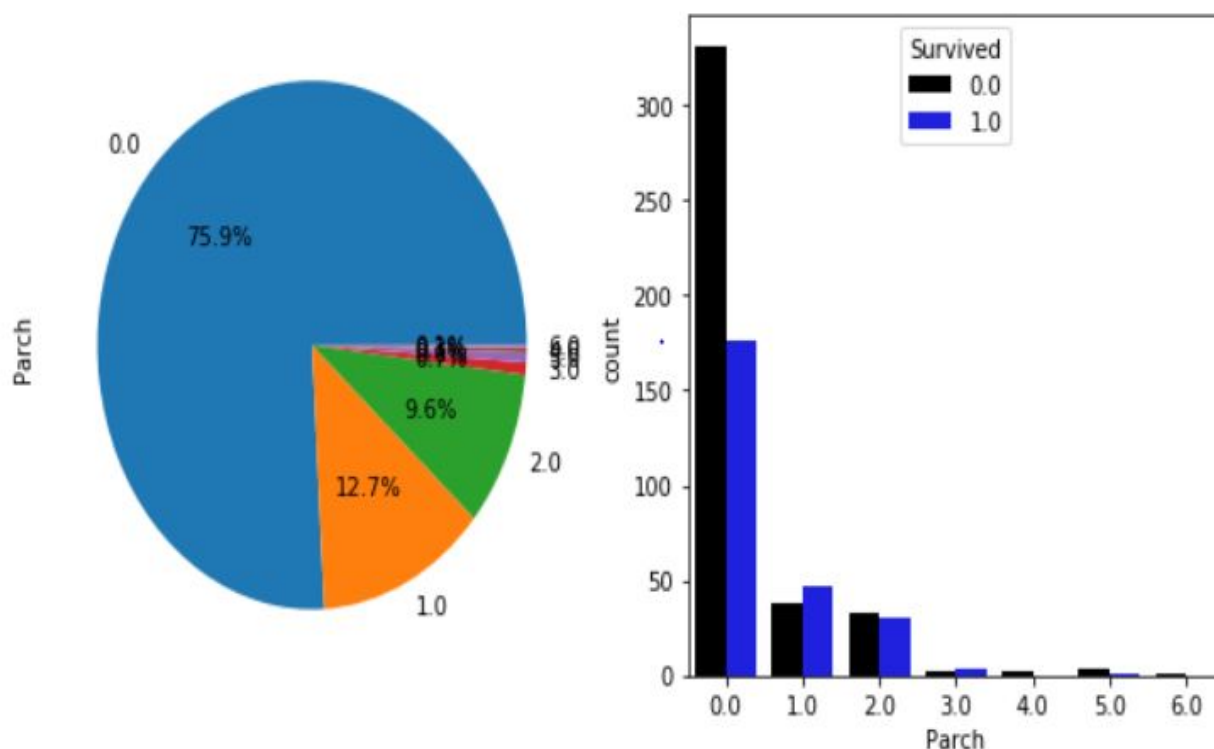
⇒ SibSp:


```
cat_plot(dataset, 'SibSp', 'Survived', survival_palette)
```



➤Parch:

```
cat_plot(dataset, 'Parch', 'Survived', survival_palette)
```



In the above two cases it seems that a passenger has more chances of survival if the family members are comparatively less. Hence we decided to create a new feature FamilySize using Parch and SibSp.

Selecting Prediction Model:

We started by trying out some simple models for baseline comparisons.

We got the following accuracies on the validation set with the below models:

Model Name	Accuracy on Train_TestDataset
KNN Classification	0.758
Logistic Regression	0.815
Linear Discriminant Analysis	0.780
Quadratic Discriminant Analysis	0.741

Since Logistic Regression got highest accuracy among the four models, we investigated further to find **p values** for this particular model and found that the following features ('Pclass', 'Sex', 'Age', 'SibSp') have the **lowest p values(<0.05)** out of all the features. After retraining the Logistic regression model with only these features the accuracy rose to **0.826**.

But these results were only with the validation dataset and therefore when we tested the above models with the test data on Kaggle Kernel, the accuracies dropped significantly and hence we decided to include other features and finally resorted to the following features :

Pclass , Sex , Age , SibSp , PassengerId , Parch, Fare , FamilySize

We then moved on to other models like Random Forest, AdaBoost and SVC and got best result with Random Forest model.

Adaptive Boosting Algorithm:

Boosting is basically the process of creating a strong classifier from a number of weak classifiers. And AdaBoost is a boosting algorithm that works very well with binary classification hence we expected good results from this algorithm.

For this algorithm we used 5 fold cross validation and the number of estimators for predicting the tree was used from the Range 1 to 202:

```
from sklearn.ensemble import AdaBoostClassifier

num_folds= 5
best_score=0
estimators = [n for n in range(1, 202, 1)]
for val in estimators:
    BoostModel= AdaBoostClassifier(n_estimators=val)
    scores= cross_val_score(BoostModel, X_trainval_transformed, Y_trainval, cv=num_folds)
    score= np.mean(scores)
    if(score > best_score):
        best_score= score
        best_estimate= val

print("best_estimate", best_estimate)
BoostModel= AdaBoostClassifier(n_estimators=best_estimate).fit(X_trainval_transformed, Y_trainval)
print("The classification accuracy of the Adaptive Boosting Classifier is: ", BoostModel.score(X_test_transformed, Y_test))
Y_pred_AB = BoostModel.predict(X_testset_transformed)
acc_AB = round(BoostModel.score(X_test_transformed, Y_test) * 100, 2)
print(acc_AB)
print(Y_pred_AB)
result= pd.concat([X_testset.PassengerId, pd.Series(Y_pred_AB.reshape(len(Y_pred_AB)))], axis=1)
result.rename(columns={'PassengerId': 'PassengerId', 0: 'Survived'}, inplace=True)
result.to_csv("submission_logreg.csv", index=False)
```

With validation dataset, we got the accuracy as 0.79 and got an accuracy of 0.78 on the test set.

```

best_estimate 4
The classification accuracy of the Adaptive Boosting Classifier is: 0.7982062780269058
79.82
[0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 0 0 0 1 1 0 0 1
 1 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 1 1 0 1 0
 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0
 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0
 0 0 1 0 0 1 0 0 1 1 0 1 1 0 1 0 0 1 0 0 1 1 0 0 0 1 1 0 1 1 0 0 1 0 1
 0 1 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0
 1 0 1 1 0 1 0 0 1 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1
 0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 0 0 0 0
 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0
 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0
 0 1 0 0 1 1 1 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0
 0 1 1 1 1 1 0 1 0 0 0]

```

SVC Model:

The objective of Support vector machine (SVM) model is to find a hyperplane (decision boundary) that best separates the target classes. In SVM, we only consider observation points closest to the decision boundary. These points are called support vectors. There are many possible kernels which can be used to measure the similarity between the new observation and support vectors like Linear, Polynomial, Radial Basis Function (RBF). We used RBF kernel for fitting the SVM model.

For SVC model we iterated over [0.001, 0.01, 0.1, 1, 5, 10, 100] for value of C and [0.001, 0.01, 0.1, 1, 5, 10, 100] for value of Gamma. We used 5 fold cross validation for finding the best parameters for our model from the above mentioned C and Gamma values.

```

C = [0.001, 0.01, 0.1, 1, 5, 10, 100]
Gamma = [0.001, 0.01, 0.1, 1, 5, 10, 100]

num_folds= 5
best_score=0
for val in C:
    for gam in Gamma:
        SvmModel= SVC(kernel='rbf', gamma=gam, C=val)
        scores= cross_val_score(SvmModel, X_trainval_transformed, Y_trainval, cv=num_folds)
        score= np.mean(scores)
        if(score > best_score):
            best_score= score
            best_regparameter= val
            best_gamma=gam

FinalSvmModel= SVC(kernel='rbf', gamma=best_gamma, C=best_regparameter).fit(X_trainval_transformed, Y_trainval)
print("The best tuning parameter for regularization in this dataset is: ", best_regparameter)
print("The best RBF parameter Gamma value in this dataset is: ", best_gamma)

print("R-squared metric of the SVM model after 5-fold cross validation when tuning parameter = ",best_regparameter,
      "and gamma = ",best_gamma,"is:", FinalSvmModel.score(X_test_transformed, Y_test))
Y_pred_SVM = FinalSvmModel.predict(X_testset_transformed)
acc_SVM = round(FinalSvmModel.score(X_test_transformed, Y_test) * 100, 2)
print(acc_SVM)
print(Y_pred_SVM)

```

With validation dataset, we got accuracy as 0.816 and on test set, we got an accuracy of 0.794

The best tuning parameter for regularization in this dataset is: 1
The best RBF parameter Gamma value in this dataset is: 0.1
R-squared metric of the SVM model after 5-fold cross validation when tuning parameter = 1 and gamma = 0.1 is: 0.816
1434977578476
81.61
[0 1 0 0 1 0 1 0 1 0 0 0 1 0 1 1 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1
1 0 0 0 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0 0 1 0 1 0
1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0
1 1 1 1 0 0 1 0 1 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0
0 0 1 0 0 1 0 0 1 1 0 1 1 1 1 0 0 1 0 0 1 1 0 0 0 0 0 1 1 0 1 1 1 0 0 1 0 1
0 1 0 0 0 0 0 0 0 1 0 1 1 0 1 1 1 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0
1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1
0 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0
1 0 0 0 0 0 0 0 1 1 0 1 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 0
0 1 0 0 1 1 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 0 1 0 1 0 0 0 0
0 1 1 1 1 1 0 1 0 0 0 0]

Random Forest Model:

Random Forest is an ensemble bagging decision tree algorithm. We build a number of decision trees on training samples. And in each split in a tree, we randomly select subset of features (d features $<$ number of features), then find the best one out of these d features for splitting. This is instead of considering all features and finding the best one. Then, we aggregate the results through voting (classification) or averaging (regression).

Here, we do an exhaustive search over specified parameter values for an estimator using the function GridSearchCV. We also pass the number of folds to be done in the cross-validation as a parameter(5). Here,

n_estimators : The number of trees in the forest,

max_depth : The maximum depth of the tree.

min_samples_split : The minimum number of samples required to split an internal node.

```
parameter_grid = {
    'max_depth' : [4, 6, 8],
    'n_estimators': [50, 10],
    'max_features': ['sqrt', 'auto', 'log2'],
    'min_samples_split': [2, 3, 10],
    'min_samples_leaf': [1, 3, 10],
    'bootstrap': [True, False],
}

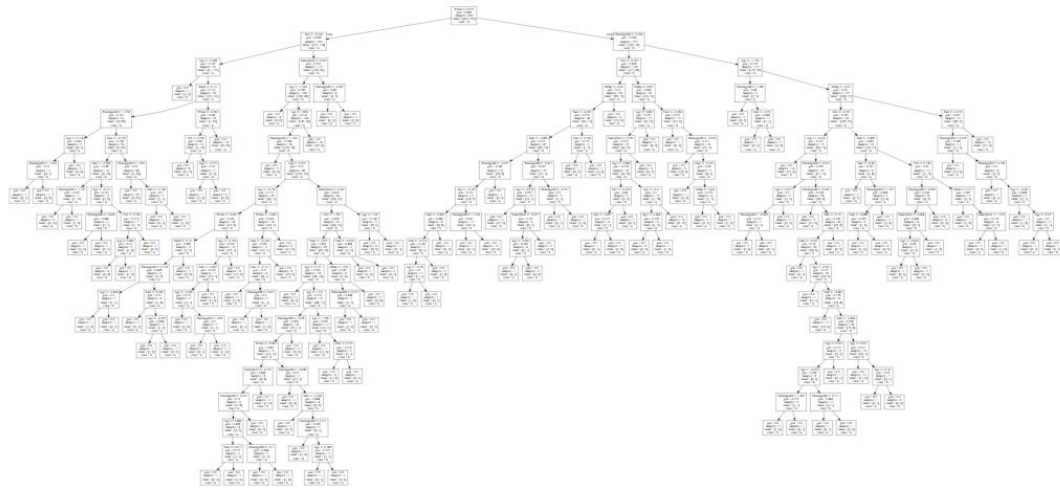
forest = RandomForestClassifier()
cross_validation = StratifiedKFold(n_splits=5)

random_forest = GridSearchCV(forest,
                              scoring='accuracy',
                              param_grid=parameter_grid,
                              cv=cross_validation,
                              verbose=1
                              )

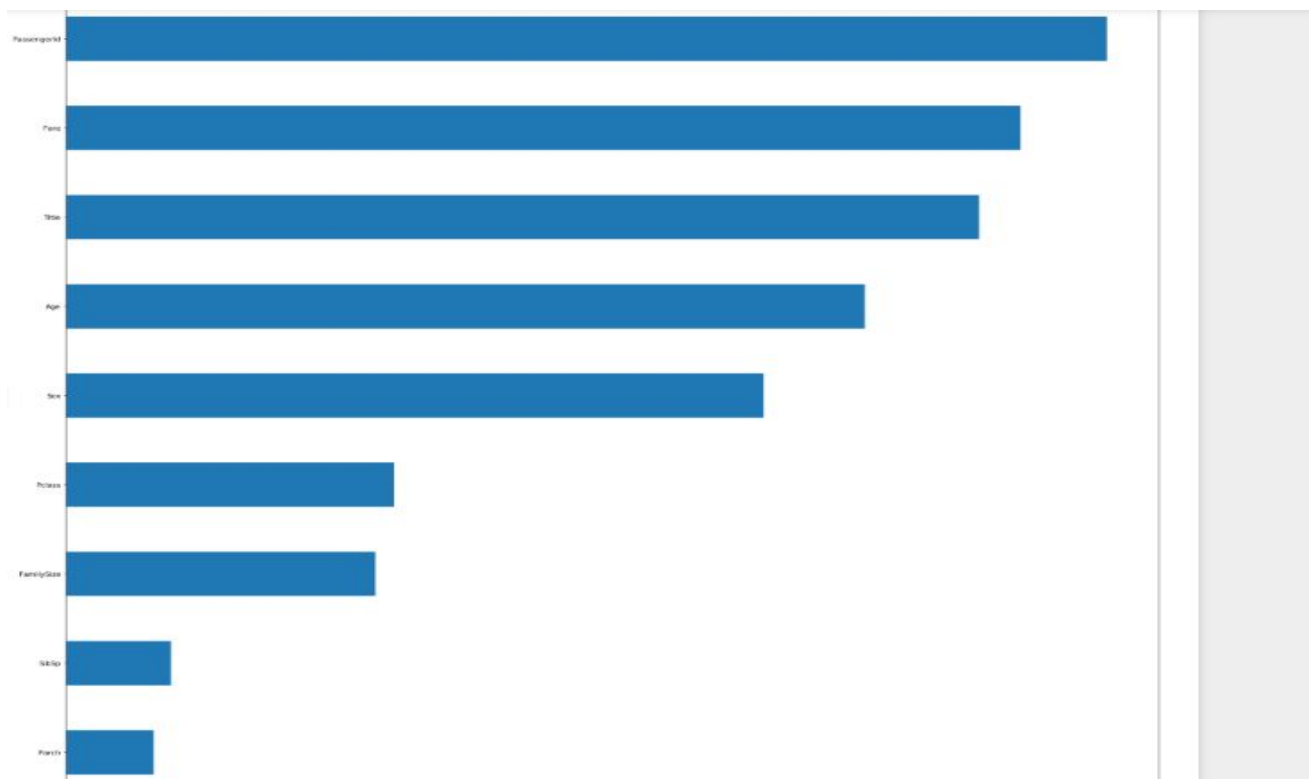
rf = random_forest.fit(X_trainval_transformed, Y_trainval)
Y_pred_rf = random_forest.predict(X_testset_transformed)
```

Using Random Forest, we got an accuracy of 0.845 on the validation set and 0.799 on the test set.











Out [8] :



Below is the importance of each feature using Random Forest model:



Furthermore, we tested our final models with the test data on Kaggle Kernel and got the best accuracy with Random Forest model of 0.799.

1782	new	being lost		0.79904	3	10h
1783	 305	TigmanshuGroupTitanic	   	0.79904	18	8h
Your Best Entry  Your submission scored 0.79904, which is not an improvement of your best score. Keep trying!						
1784	 3415	НиколайДружинин		0.79904	39	6h
1785	new	Olivier Flamand		0.79904	15	7h

Conclusion and future improvements:

Random Forest was the most successful model in predicting the survival. We got an accuracy of 79.9% on the test data. So, in this project we explored the titanic data, cleaned it and performed data engineering as per our analysis and understanding. We applied various models and compared their performances.

There is scope for improvement to dig more into the data and eventually build new features.

We could also try different models like Neural Networks.