TITANIC DATASET (FROM DISASTER)

Project1

ABSTRACT
Machine Learning On April 15, 1912, during her maiden voyage, the widely considered "unsinkable" RMS Titanic sank after colliding with an iceberg. Unfortunately, there weren't enough lifeboats for everyone onboard, resulting in the death of 1502 out of 2224 passengers and crew.

DATA 3402-001

Revan Thakkar
1001802099

Introduction

The goal of the project was to predict the survival of passengers based on a set of data.

We used Kaggle DataSet

"Titanic: Machine Learning from Disaster"

(see https://www.kaggle.com/c/titanic/data)

to retrieve necessary data and evaluate accuracy of our predictions.

The historical data has been split into two groups,

1. training set
2. test set

For the training set, dataset provides with the outcome (whether or not a passenger survived). this set is used to build our model to generate predictions for the test set.

For each passenger in the test set, we had to predict whether or not they survived the sinking.

score after the training dataset was the percentage of correctly predictions.

Goals Achieved during this project

- Programming language Python and its libraries NumPy (to perform matrix operations) and SciKit-Learn (to apply machine learning algorithms)
- Several machine learning algorithms (SVM,KNN,LR)
- Feature Engineering techniques

Tools used

- Google Colab
- Google Drive for DataSet
- *Python with* the libraries *numpy*, *sklearn*, and *matplotlib*

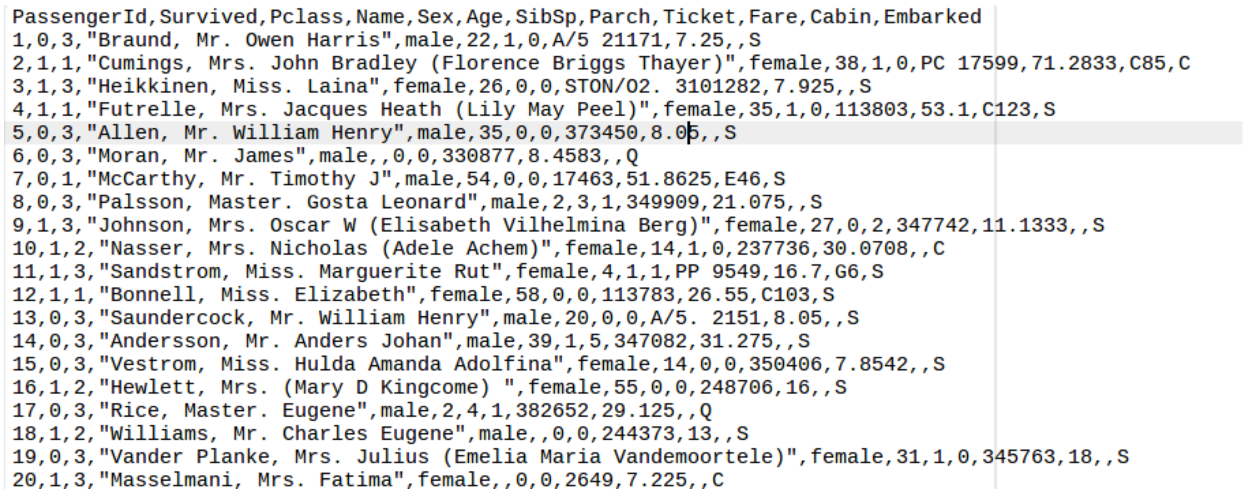
Dataset Details

Training and Test data come in CSV file and contain the following features:

- Passenger ID
- Passenger Class
- Name
- Sex
- Age
- Number of passenger's siblings and spouses on board
- Number of passenger's parents and children on board
- Ticket
- Fare
- Cabin
- City where passenger embarked

The Dataset is a labelled dataset so we use supervised learning techniques..

Sample Data in csv Format

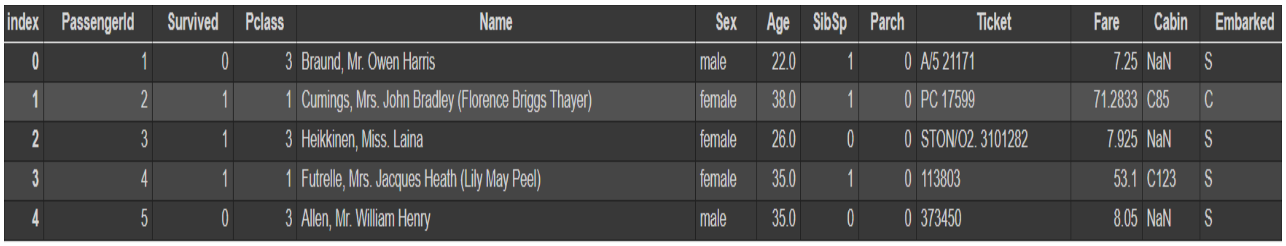


```
PassengerId,Survived,Pclass,Name,Sex,Age,SibSp,Parch,Ticket,Fare,Cabin,Embarked
1,0,3,"Braund, Mr. Owen Harris",male,22,1,0,A/5 21171,7.25,,S
2,1,1,"Cumings, Mrs. John Bradley (Florence Briggs Thayer)",female,38,1,0,PC 17599,71.2833,C85,C
3,1,3,"Heikkinen, Miss. Laina",female,26,0,0,STON/O2. 3101282,7.925,,S
4,1,1,"Futrelle, Mrs. Jacques Heath (Lily May Peel)",female,35,1,0,113803,53.1,C123,S
5,0,3,"Allen, Mr. William Henry",male,35,0,0,373450,8.05,,S
6,0,3,"Moran, Mr. James",male,,0,0,330877,8.4583,,Q
7,0,1,"McCarthy, Mr. Timothy J",male,54,0,0,17463,51.8625,E46,S
8,0,3,"Palsson, Master. Gosta Leonard",male,2,3,1,349909,21.075,,S
9,1,3,"Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)",female,27,0,2,347742,11.1333,,S
10,1,2,"Nasser, Mrs. Nicholas (Adele Achem)",female,14,1,0,237736,30.0708,,C
11,1,3,"Sandstrom, Miss. Marguerite Rut",female,4,1,1,PP 9549,16.7,G6,S
12,1,1,"Bonnell, Miss. Elizabeth",female,58,0,0,113783,26.55,C103,S
13,0,3,"Saundercock, Mr. William Henry",male,20,0,0,A/5. 2151,8.05,,S
14,0,3,"Andersson, Mr. Anders Johan",male,39,1,5,347082,31.275,,S
15,0,3,"Vestrom, Miss. Hulda Amanda Adolfina",female,14,0,0,350406,7.8542,,S
16,1,2,"Hewlett, Mrs. (Mary D Kingcome) ",female,55,0,0,248706,16,,S
17,0,3,"Rice, Master. Eugene",male,2,4,1,382652,29.125,,Q
18,1,2,"Williams, Mr. Charles Eugene",male,,0,0,244373,13,,S
19,0,3,"Vander Planke, Mrs. Julius (Emelia Maria Vandemoortele)",female,31,1,0,345763,18,,S
20,1,3,"Masselmani, Mrs. Fatima",female,,0,0,2649,7.225,,C
```

Preprocessing (Feature Engineering) Details

Data Reterived using

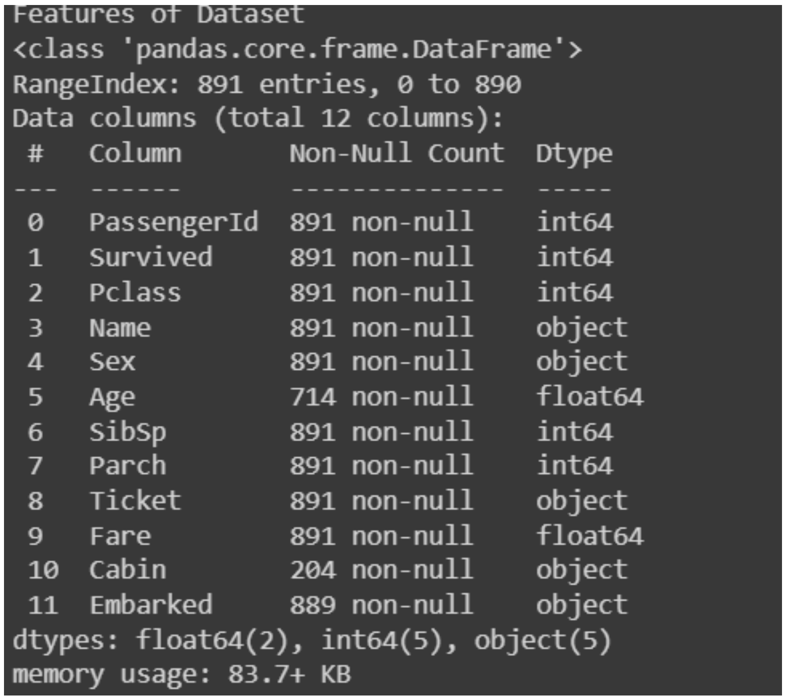
Train.head()



index	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	7.25	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	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	7.925	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S

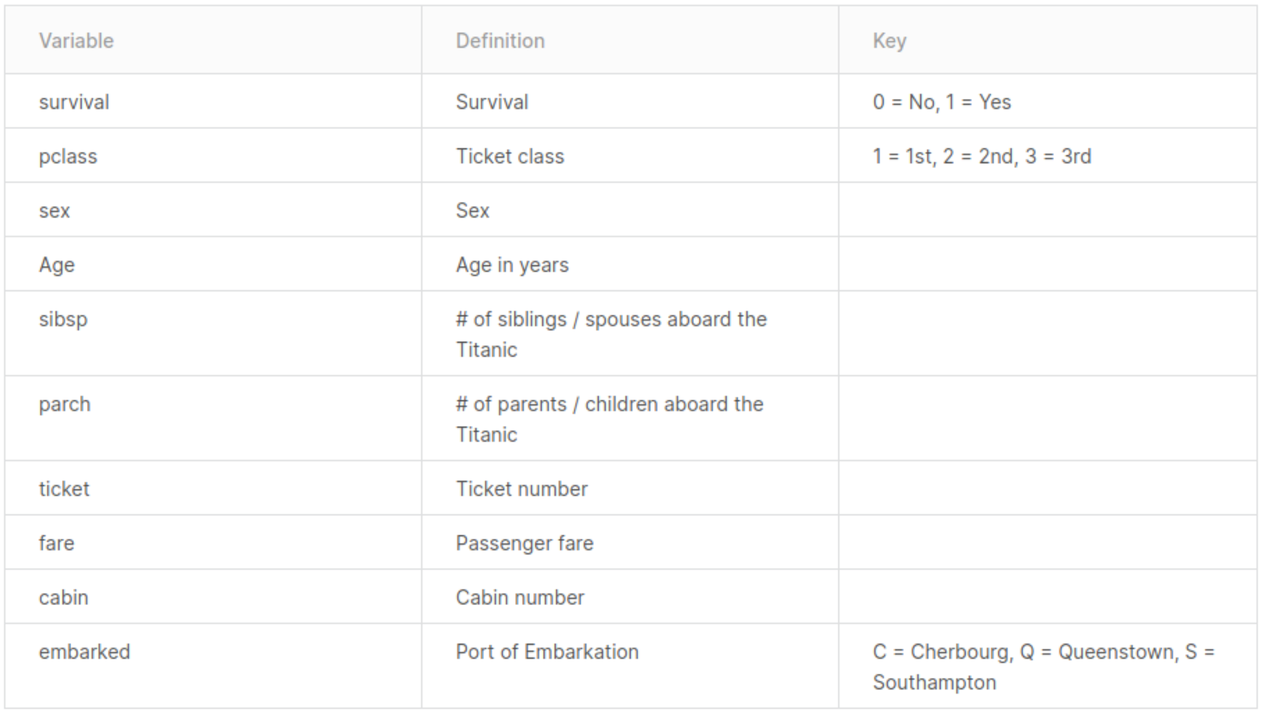
Features of DataSet

Train.info()



```
Features of Dataset
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Describing the data



Variable	Definition	Key
survival	Survival	0 = No, 1 = Yes
pclass	Ticket class	1 = 1st, 2 = 2nd, 3 = 3rd
sex	Sex	
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

Features Study Details

After getting a better perception of the different aspects of the dataset, exploring the features and the part they played in the survival or demise of a traveler.

1. Survived

The first feature reported if a traveler lived or died. A comparison revealed that **more than 60% of the passengers had died.**

2. Pclass

This feature renders the passenger division. The tourists could opt from three distinct sections, namely class-1, class-2, class-3. The third class had the highest number of commuters, followed by class-2 and class-1. The number of tourists in the third class was more than the number of passengers in the first and second class combined. The survival chances of a class-1 traveler were higher than a class-2 and class-3 traveler.

3. Sex

Approximately 65% of the tourists were male while the remaining 35% were female. Nonetheless, the percentage of female survivors was higher than the number of male survivors. **More than 80% of male commuters died, as compared to around 70% female commuters.**

4. Age

The youngest traveler onboard was aged around two months and the oldest traveler was 80 years. The average age of tourists onboard was just under 30 years. Clearly, **a larger fraction of children under 10 survived than died**. or every other age group, the number of casualties was higher than the number of survivors. More than 140 people within the age group 20 and 30 were dead as compared to just around 80 people of the same age range sustained.

5. SibSp

SibSp is the number of siblings or spouse of a person onboard. A maximum of 8 siblings and spouses traveled along with one of the traveler. **More than 90% of people traveled alone or with one of their sibling or spouse**. The **chances of survival dropped drastically if someone traveled with more than 2 siblings or spouse.**

6. Parch

Similar to the SibSp, this feature contained the number of parents or children each passenger was touring with. A maximum of 9 parents/children travelled along with one of the traveler.

7. Fare

By splitting the fare amount into four categories, it was obvious that there was a strong association between the charge and the survival. **The higher a tourist paid, the higher would be his chances to survive.**

8. Embarked

Embarked implies where the traveler mounted from. There are three possible values for Embark — **Southampton, Cherbourg, and Queenstown**. More than **70% of the people boarded from**

Southampton. Just under 20% boarded from Cherbourg and the rest boarded from Queenstown. **People who boarded from Cherbourg had a higher chance of survival than people who boarded from Southampton or Queenstown.**

Data Imputation

Data imputation is the practice of replacing missing data with some substituted values. There can be a multitude of substitution processes that can be used. I used some of them for the missing values.

Since the data can have missing fields, incomplete fields, or fields containing hidden information, a crucial step in building any prediction system is Feature Engineering. For instance, the fields Age, Fare, and Embarked in the training and test data, had missing values that had to be filled in. The field Name while being useless itself, contained passenger's Title (Mr., Mrs., etc.), we also used passenger's surname to distinguish families on board of Titanic. Below is the list of all changes that has been made to the data.

Filling in missing values in the fields Fare, Embarked, and Age

Since the number of missing values was small,

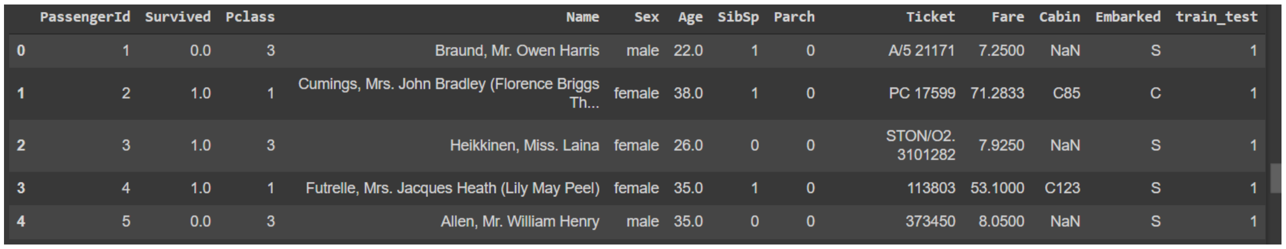
1. we used mean of all Age values to fill in the missing Age fields
2. we used 0 of all Fare values to fill in the missing Fare fields
3. we used mode for the field Embarked.

Transformation into a categorical column.

We have already noticed from the table, there are two columns that contain string-type values: The “Sex” column and the “Berth” column.

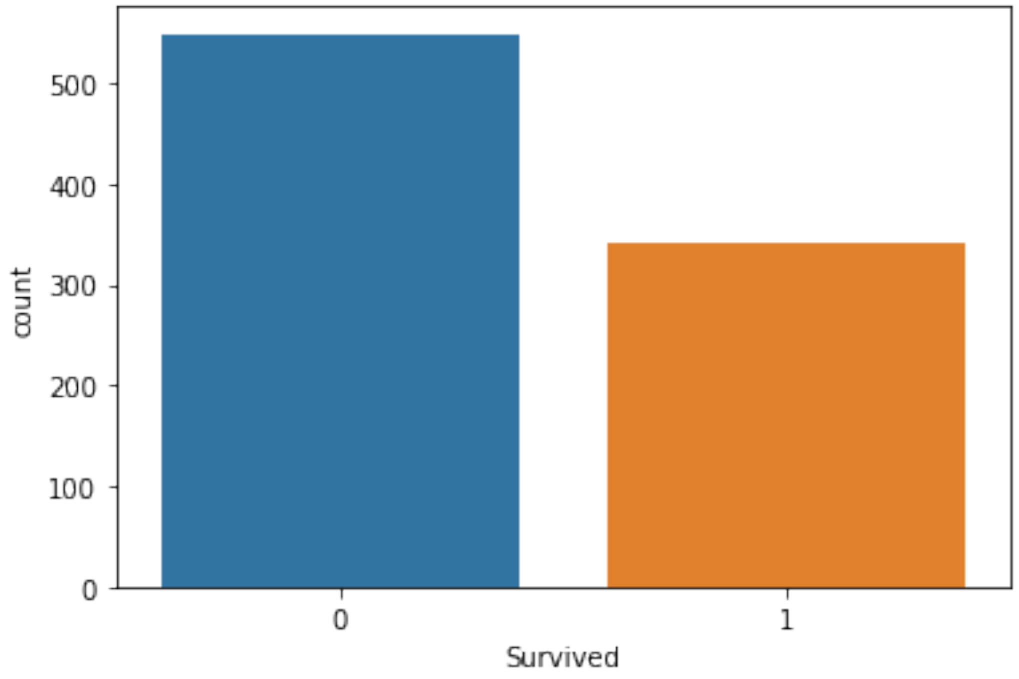
Let’s convert that into integer type values, and transform it into a categorical column:

After Missing Values handling

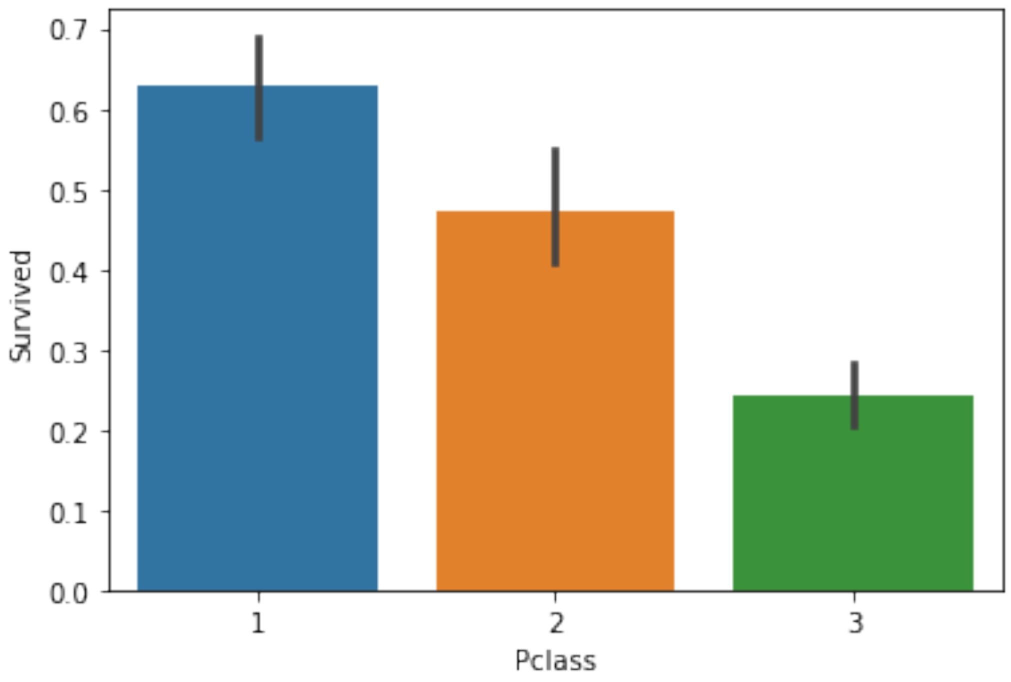


	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	train_test
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	1
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	1

Basic Exploratory on Dataset



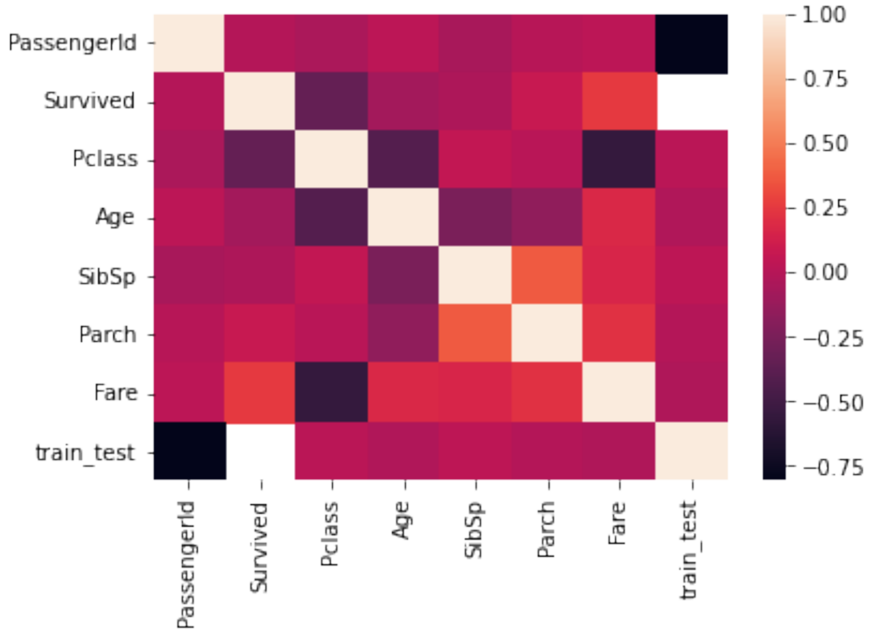
```
sns.countplot(train['Survived'],label="Count")
```



```
sns.barplot(x='Pclass', y='Survived', data=train)
```

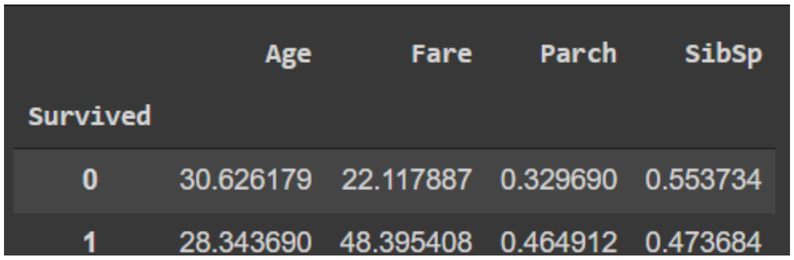
HeatMap for Correlation of Data

```

sns.heatmap(all_data.corr())
```



Pivot Table for Train data

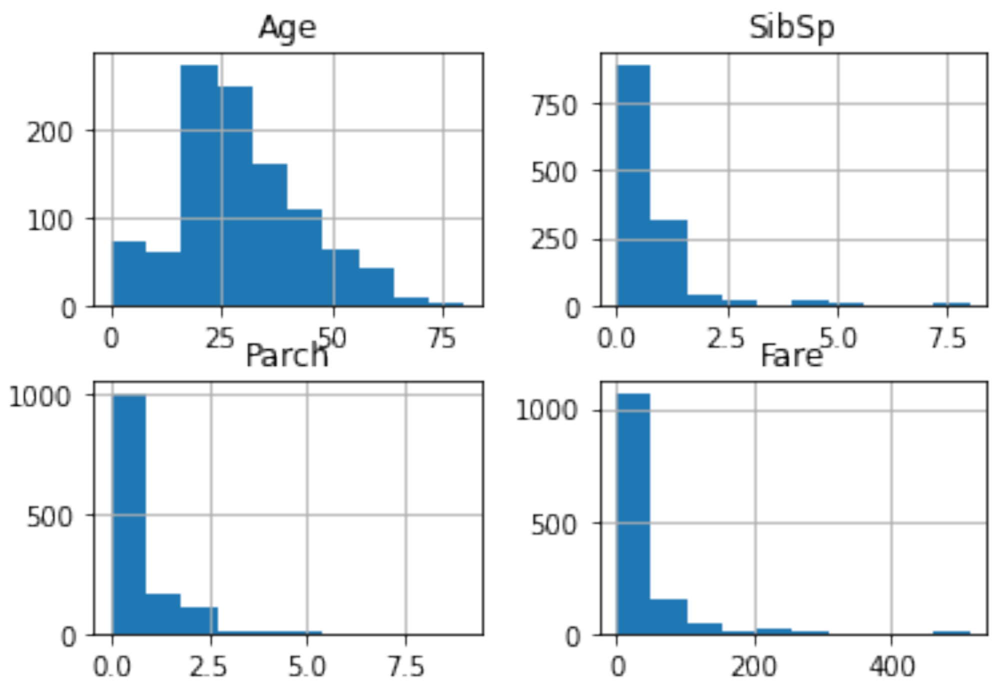
```
pd.pivot_table(train, index = 'Survived', values = ['Age','SibSp','Parch','Fare'])
```



```
                Age       Fare     Parch     SibSp
Survived
       0  30.626179  22.117887  0.329690  0.553734
       1  28.343690  48.395408  0.464912  0.473684
```

Histogram on Numeric Features

```
df_num = all_data[['Age','SibSp','Parch','Fare']]
```



Data preprocessing for model

making our data, model-ready. The objectives we have to fulfill are listed below:

1. Drop the null values from the Embarked column
2. Include only relevant data
3. Categorically transform all of the data, using something called a transformer.
4. Impute data with the central tendencies for age and fare.
5. Normalize the *fare* column to have a more normal distribution.
6. using standard scaler scale data 0-1

Model Description

1. **Logistic regression**

Logistic regression, despite its name, is a classification model rather than regression model. Logistic regression is a simple and more efficient method for binary and linear classification problems. It is a classification model, which is very easy to realize and achieves very good performance with linearly separable classes. It is an extensively employed algorithm for classification in industry. The logistic regression model is a statistical method for binary classification that can be generalized to multiclass classification. Scikit-learn has a highly optimized version of logistic regression implementation, which supports multiclass classification

2. KNN (K Nearest Neighbour)

K Nearest Neighbour algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbour) suggests it considers K Nearest Neighbours (Data points) to predict the class or continuous value for the new Datapoint.

The algorithm's learning is:

1. Instance-based learning: Here we do not learn weights from training data to predict output (as in model-based algorithms) but use entire training instances to predict output for unseen data.

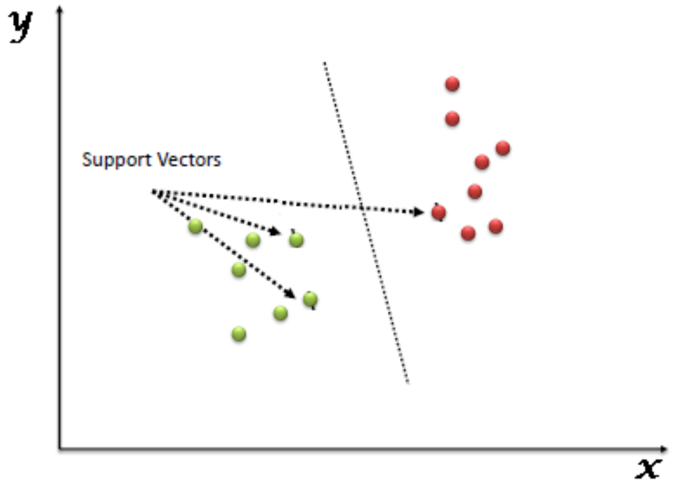
2. Lazy Learning: Model is not learned using training data prior and the learning process is postponed to a time when prediction is requested on the new instance.

3. Non -Parametric: In KNN, there is no predefined form of the mapping function.

For classification: A class label assigned to the majority of K Nearest Neighbours from the training dataset is considered as a predicted class for the new data point.

3. SVM (Support Vector Machine)

it is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well (look at the below snapshot).



Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line).

Splitting Dataset for Training and Testing

Training Data

The observations in the training set form the experience that the algorithm uses to learn. In supervised learning problems, each observation consists of an observed output variable and one or more observed input variables.

Test Data

The test set is a set of observations used to evaluate the performance of the model using some performance metric. It is important that no observations from the training set are included in the test set. If the test set does contain examples from the training set, it will be difficult to assess whether the algorithm has learned to generalize from the training set or has simply memorized it.

Underfitting and Overfitting

Splitting a dataset might also be important for detecting if your model suffers from one of two very common problems, called underfitting and overfitting:

1. **Underfitting** is usually the consequence of a model being unable to encapsulate the relations among data. For example, this can happen when trying to represent nonlinear relations with a linear model. Underfitted models will likely have poor performance with both training and test sets.
2. **Overfitting** usually takes place when a model has an excessively complex structure and learns both the existing relations among data and noise. Such models often have bad generalization capabilities. Although they work well with training data, they usually yield poor performance with unseen (test) data

What is Accuracy, Precision and Recall

Consider a classification task in which a machine learning system observes tumors and has to predict whether these tumors are benign or malignant. **Accuracy**, or the fraction of instances that were classified correctly, is an obvious measure of the program's performance. While accuracy does measure the program's performance, it does not make distinction between malignant tumors that were classified as being benign, and benign tumors that were classified as being malignant. In some applications, the costs incurred on all types of errors may be the same. In this problem, however, failing to identify malignant tumors is a more serious error than classifying benign tumors as being malignant by mistake.

We can measure each of the possible prediction outcomes to create different snapshots of the classifier's performance. When the system correctly classifies a tumor as being malignant, the prediction is called a **true positive**. When the system incorrectly classifies a benign tumor as being malignant, the prediction is a **false positive**. Similarly, a **false negative** is an incorrect prediction that the tumor is benign, and a **true negative** is a correct prediction that a tumor is benign. These four outcomes can be used to calculate several common measures of classification performance, like accuracy, precision, recall and so on.

Accuracy is calculated with the following formula −

ACC = (TP + TN)/(TP + TN + FP + FN)

Where, TP is the number of true positives

TN is the number of true negatives

FP is the number of false positives

FN is the number of false negatives.

Precision is the fraction of the tumors that were predicted to be malignant that are actually malignant. Precision is calculated with the following formula −

PREC = TP/(TP + FP)

Recall is the fraction of malignant tumors that the system identified. Recall is calculated with the following formula −

R = TP/(TP + FN)

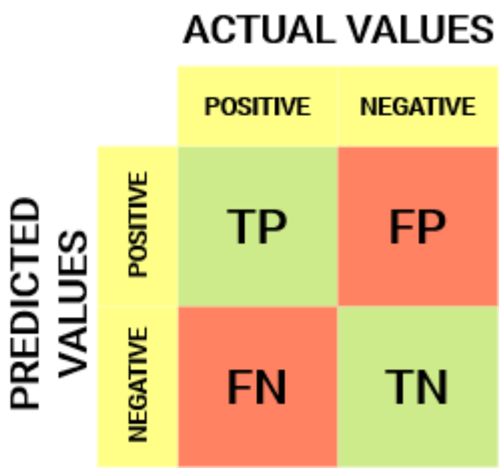
In this example, precision measures the fraction of tumors that were predicted to be malignant that are actually malignant. Recall measures the fraction of truly malignant tumors that were detected. The precision and recall measures could reveal that a classifier with impressive accuracy actually fails to detect most of the malignant tumors. If most tumors are benign, even a classifier that never predicts malignancy could have high accuracy. A different classifier with lower accuracy and higher recall might be better suited to the task, since it will detect more of the malignant tumors. Many other performance measures for classification can also be used.

What is a Confusion Matrix?

The million-dollar question – what, after all, is a confusion matrix?

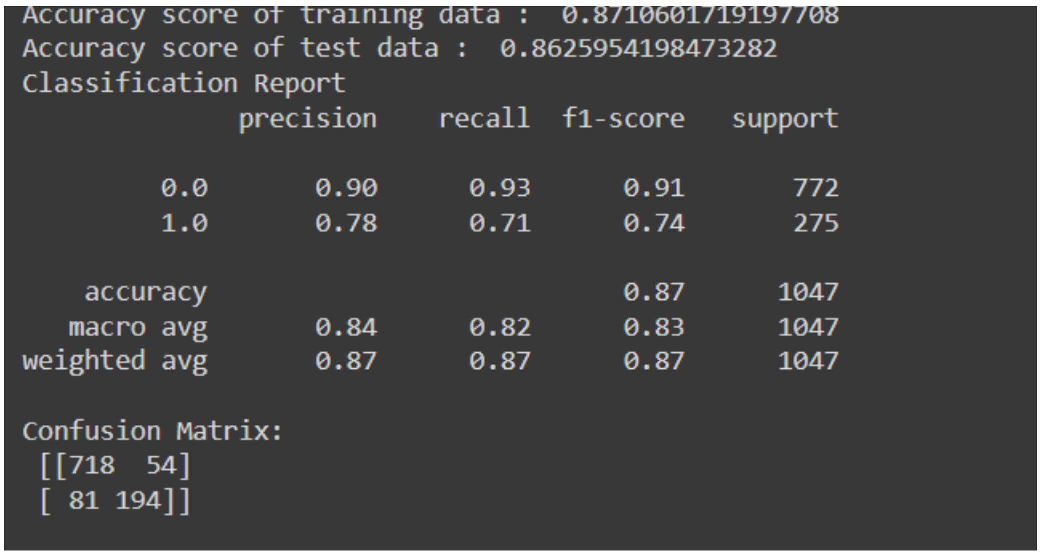
A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a 2 x 2 matrix as shown below with 4 values:



Model Evaluation

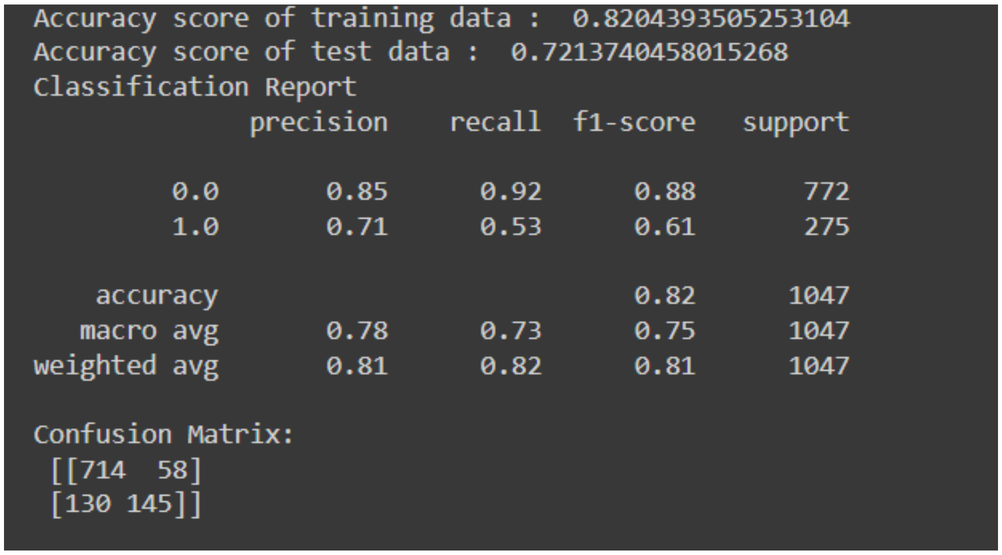
Linear Regression Results



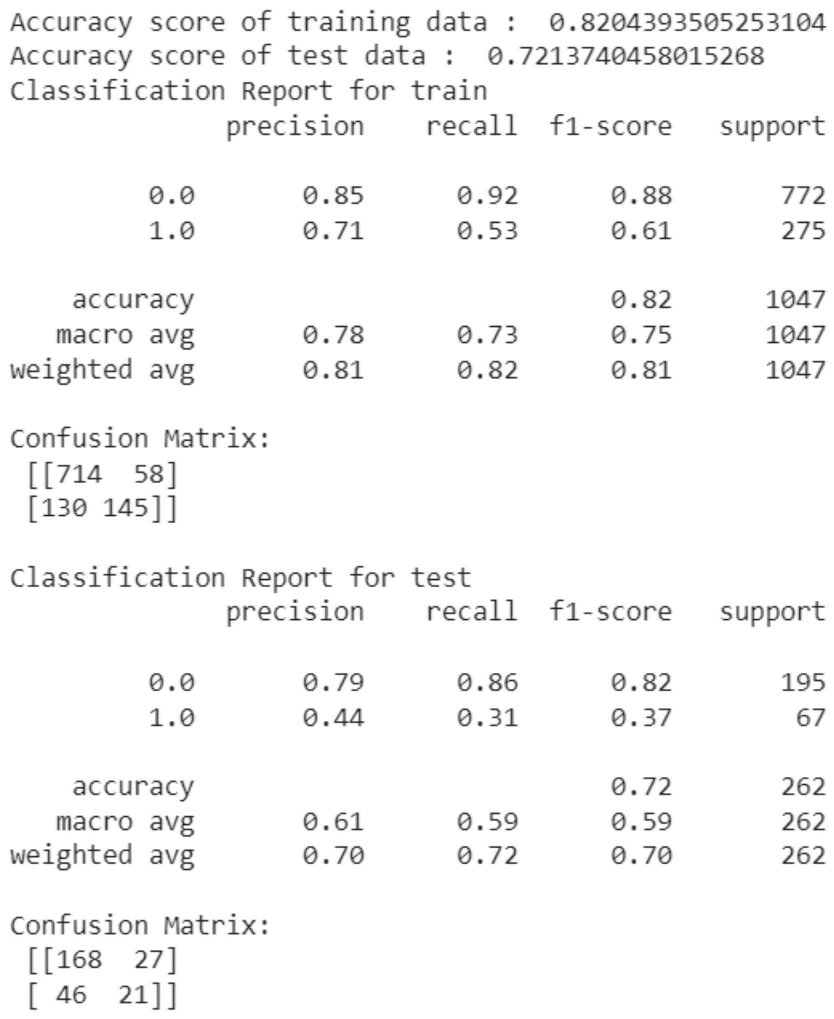
```
Accuracy score of training data :  0.8710601719197708
Accuracy score of test data :  0.8625954198473282
Classification Report
              precision    recall  f1-score   support

         0.0       0.90      0.93      0.91       772
         1.0       0.78      0.71      0.74       275

    accuracy                           0.87      1047
   macro avg       0.84      0.82      0.83      1047
weighted avg       0.87      0.87      0.87      1047

Confusion Matrix:
 [[718  54]
 [ 81 194]]
```

```
Accuracy score of training data :  0.8710601719197708
Accuracy score of test data :  0.8625954198473282
Classification Report for Train
              precision    recall  f1-score   support

         0.0       0.90      0.93      0.91       772
         1.0       0.78      0.71      0.74       275

    accuracy                           0.87      1047
   macro avg       0.84      0.82      0.83      1047
weighted avg       0.87      0.87      0.87      1047

Confusion Matrix:
 [[718  54]
 [ 81 194]]

Classification Report for Test
              precision    recall  f1-score   support

         0.0       0.89      0.93      0.91       195
         1.0       0.76      0.67      0.71        67

    accuracy                           0.86       262
   macro avg       0.83      0.80      0.81       262
weighted avg       0.86      0.86      0.86       262

Confusion Matrix:
 [[181  14]
 [ 22  45]]
```

KNN Results

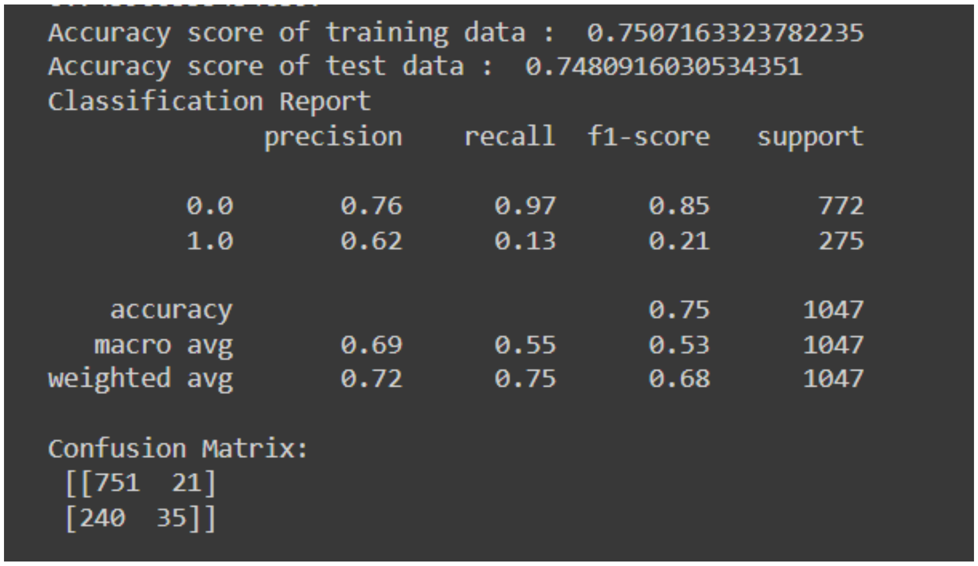


```
Accuracy score of training data :  0.8204393505253104
Accuracy score of test data :  0.7213740458015268
Classification Report
              precision    recall  f1-score   support

         0.0       0.85      0.92      0.88       772
         1.0       0.71      0.53      0.61       275

    accuracy                           0.82      1047
   macro avg       0.78      0.73      0.75      1047
weighted avg       0.81      0.82      0.81      1047

Confusion Matrix:
 [[714  58]
 [130 145]]
```

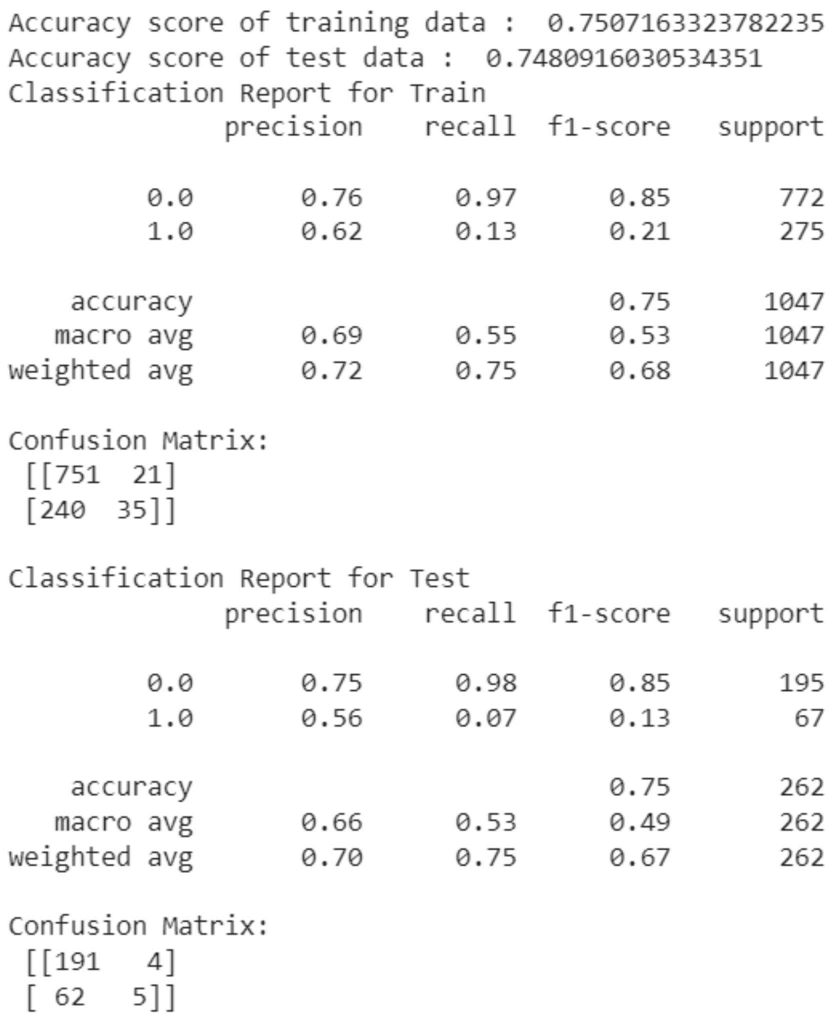


```
Accuracy score of training data :  0.8204393505253104
Accuracy score of test data :  0.7213740458015268
Classification Report for train
              precision    recall  f1-score   support

         0.0       0.85      0.92      0.88       772
         1.0       0.71      0.53      0.61       275

    accuracy                           0.82      1047
   macro avg       0.78      0.73      0.75      1047
weighted avg       0.81      0.82      0.81      1047

Confusion Matrix:
 [[714  58]
 [130 145]]

Classification Report for test
              precision    recall  f1-score   support

         0.0       0.79      0.86      0.82       195
         1.0       0.44      0.31      0.37        67

    accuracy                           0.72       262
   macro avg       0.61      0.59      0.59       262
weighted avg       0.70      0.72      0.70       262

Confusion Matrix:
 [[168  27]
 [ 46  21]]
```

SVM Results

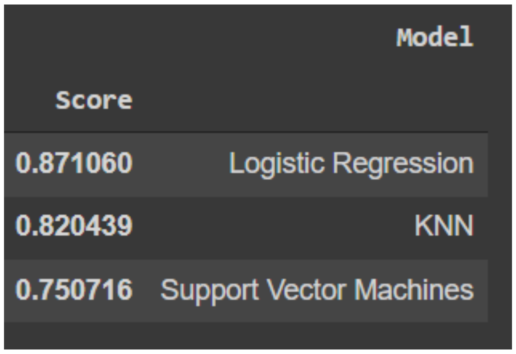


```
Accuracy score of training data :  0.7507163323782235
Accuracy score of test data :  0.7480916030534351
Classification Report
              precision    recall  f1-score   support

         0.0       0.76      0.97      0.85       772
         1.0       0.62      0.13      0.21       275

    accuracy                           0.75      1047
   macro avg       0.69      0.55      0.53      1047
weighted avg       0.72      0.75      0.68      1047

Confusion Matrix:
 [[751  21]
 [240  35]]
```

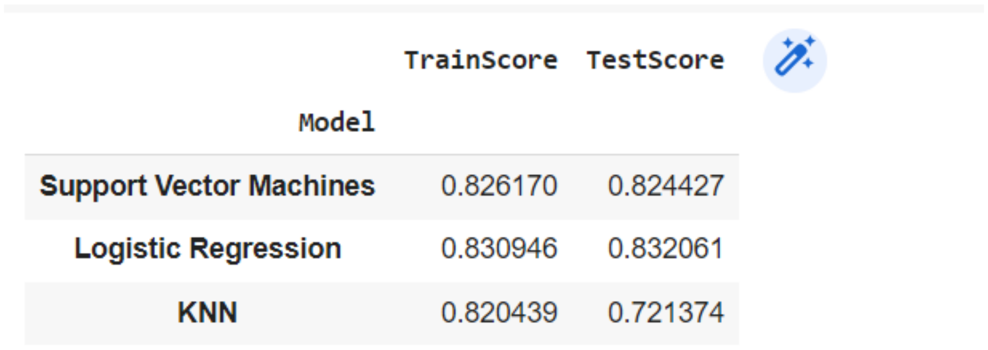


```
Accuracy score of training data :  0.7507163323782235
Accuracy score of test data :  0.7480916030534351
Classification Report for Train
              precision    recall  f1-score   support

         0.0       0.76      0.97      0.85       772
         1.0       0.62      0.13      0.21       275

    accuracy                           0.75      1047
   macro avg       0.69      0.55      0.53      1047
weighted avg       0.72      0.75      0.68      1047

Confusion Matrix:
 [[751  21]
 [240  35]]

Classification Report for Test
              precision    recall  f1-score   support

         0.0       0.75      0.98      0.85       195
         1.0       0.56      0.07      0.13        67

    accuracy                           0.75       262
   macro avg       0.66      0.53      0.49       262
weighted avg       0.70      0.75      0.67       262

Confusion Matrix:
 [[191   4]
 [ 62   5]]
```

Model Summary Chart for above 3 Models

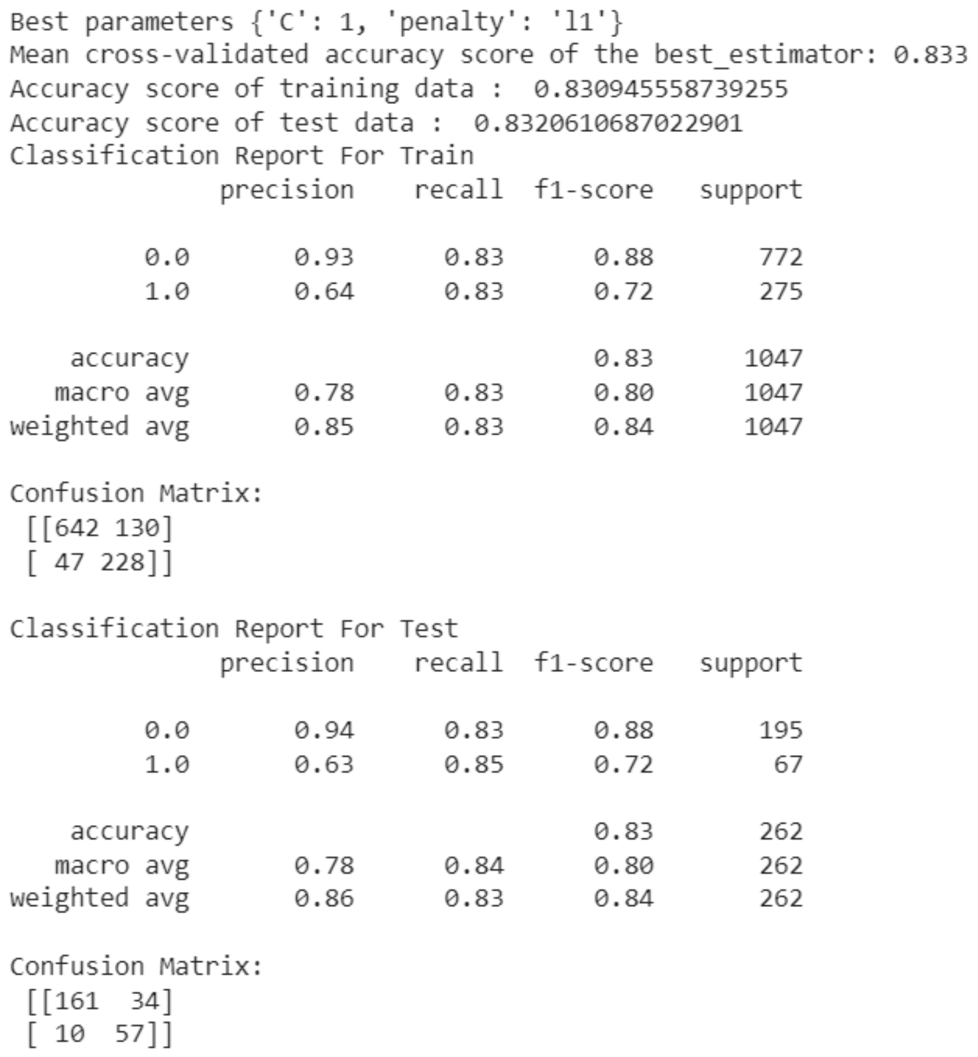


Score	Model
0.871060	Logistic Regression
0.820439	KNN
0.750716	Support Vector Machines



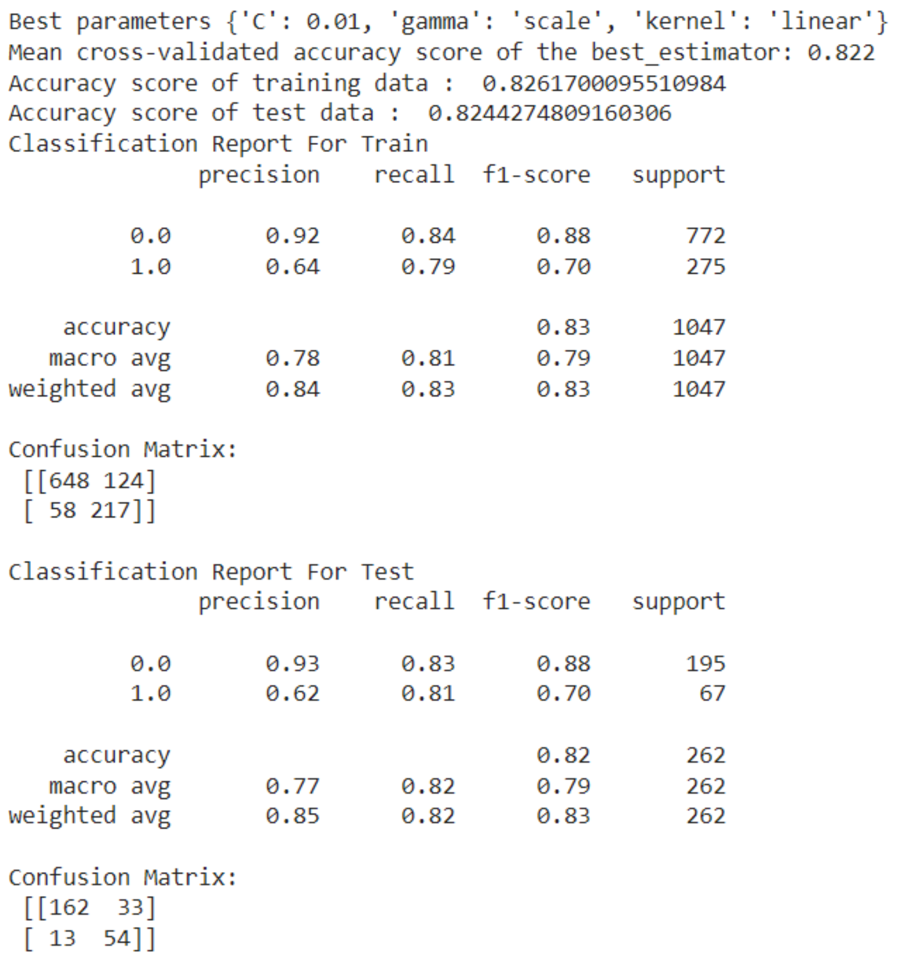
Model	TrainScore	TestScore
Support Vector Machines	0.826170	0.824427
Logistic Regression	0.830946	0.832061
KNN	0.820439	0.721374

Using GridSearchCV for HyperParameters in Linear Regression



```
Best parameters {'C': 1, 'penalty': 'l1'}
Mean cross-validated accuracy score of the best_estimator: 0.833
Accuracy score of training data :  0.830945558739255
Accuracy score of test data :  0.8320610687022901
Classification Report For Train
              precision    recall  f1-score   support

         0.0       0.93      0.83      0.88       772
         1.0       0.64      0.83      0.72       275

    accuracy                           0.83      1047
   macro avg       0.78      0.83      0.80      1047
weighted avg       0.85      0.83      0.84      1047

Confusion Matrix:
 [[642 130]
 [ 47 228]]

Classification Report For Test
              precision    recall  f1-score   support

         0.0       0.94      0.83      0.88       195
         1.0       0.63      0.85      0.72        67

    accuracy                           0.83       262
   macro avg       0.78      0.84      0.80       262
weighted avg       0.86      0.83      0.84       262

Confusion Matrix:
 [[161  34]
 [ 10  57]]
```

Using GridSearchCv for HyperParameters using SVC



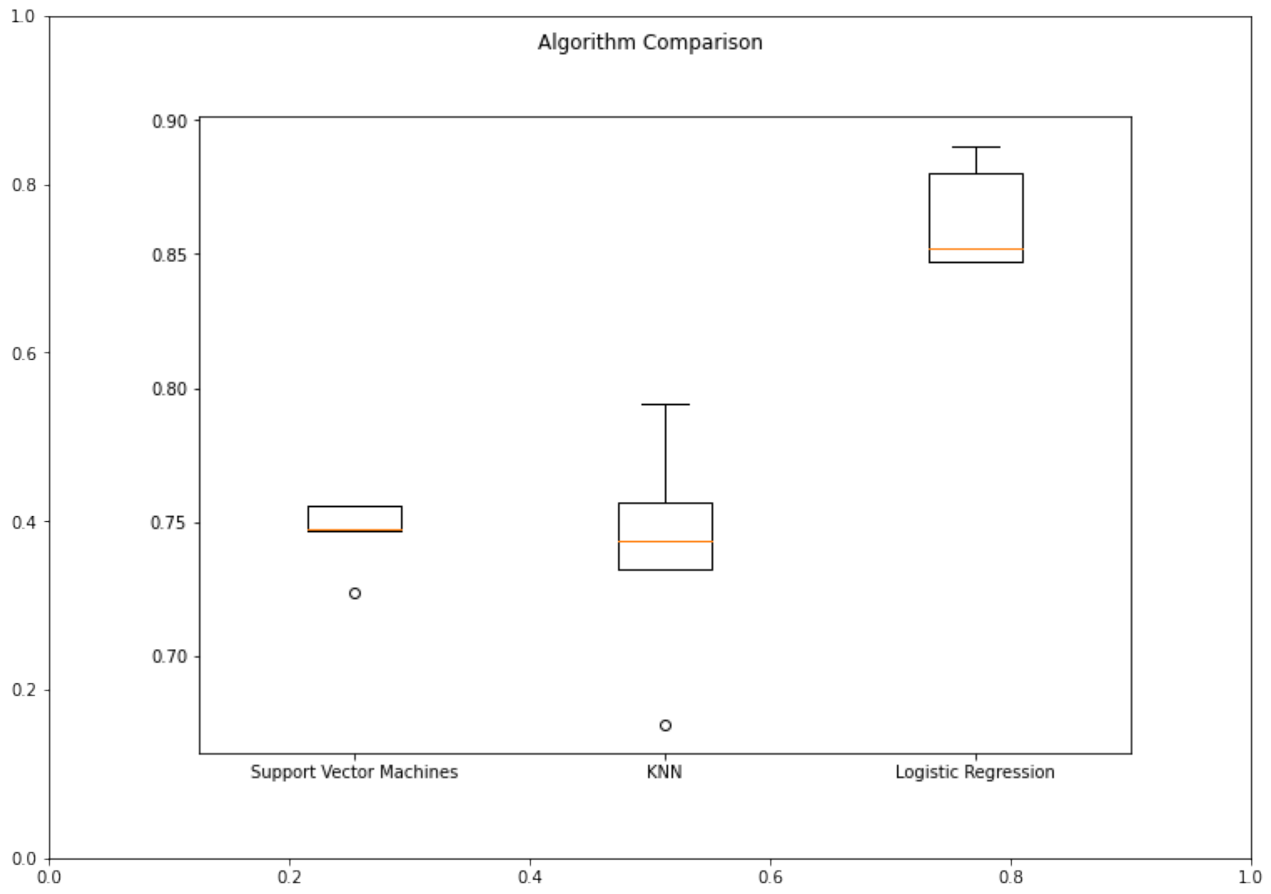
```
Best parameters {'C': 0.01, 'gamma': 'scale', 'kernel': 'linear'}
Mean cross-validated accuracy score of the best_estimator: 0.822
Accuracy score of training data :  0.8261700095510984
Accuracy score of test data :  0.8244274809160306
Classification Report For Train
              precision    recall  f1-score   support

         0.0       0.92      0.84      0.88       772
         1.0       0.64      0.79      0.70       275

    accuracy                           0.83      1047
   macro avg       0.78      0.81      0.79      1047
weighted avg       0.84      0.83      0.83      1047

Confusion Matrix:
 [[648 124]
 [ 58 217]]

Classification Report For Test
              precision    recall  f1-score   support

         0.0       0.93      0.83      0.88       195
         1.0       0.62      0.81      0.70        67

    accuracy                           0.82       262
   macro avg       0.77      0.82      0.79       262
weighted avg       0.85      0.82      0.83       262

Confusion Matrix:
 [[162  33]
 [ 13  54]]
```

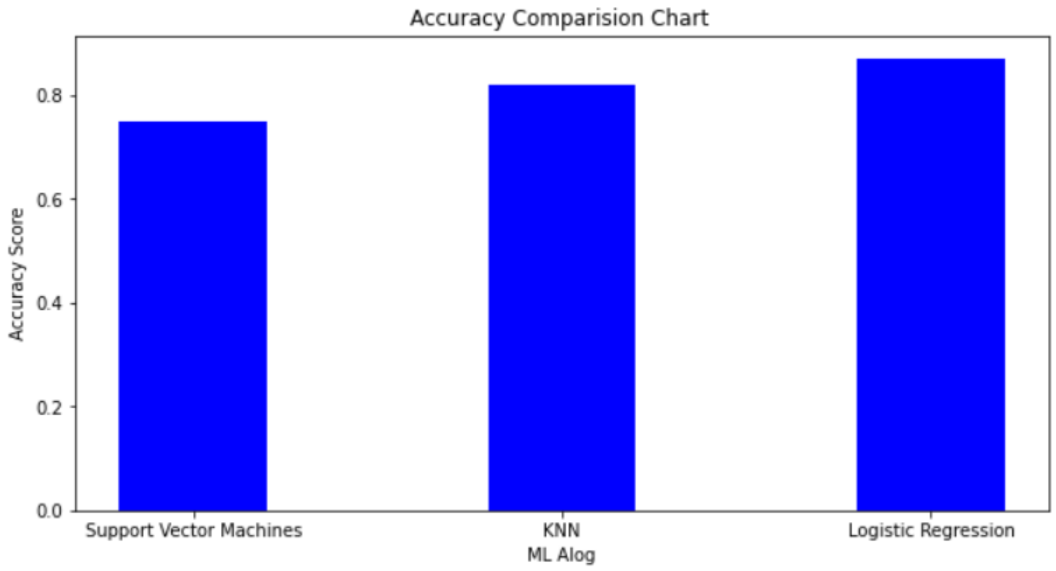
Using GridSearchCV for HyperParameters Using KNN

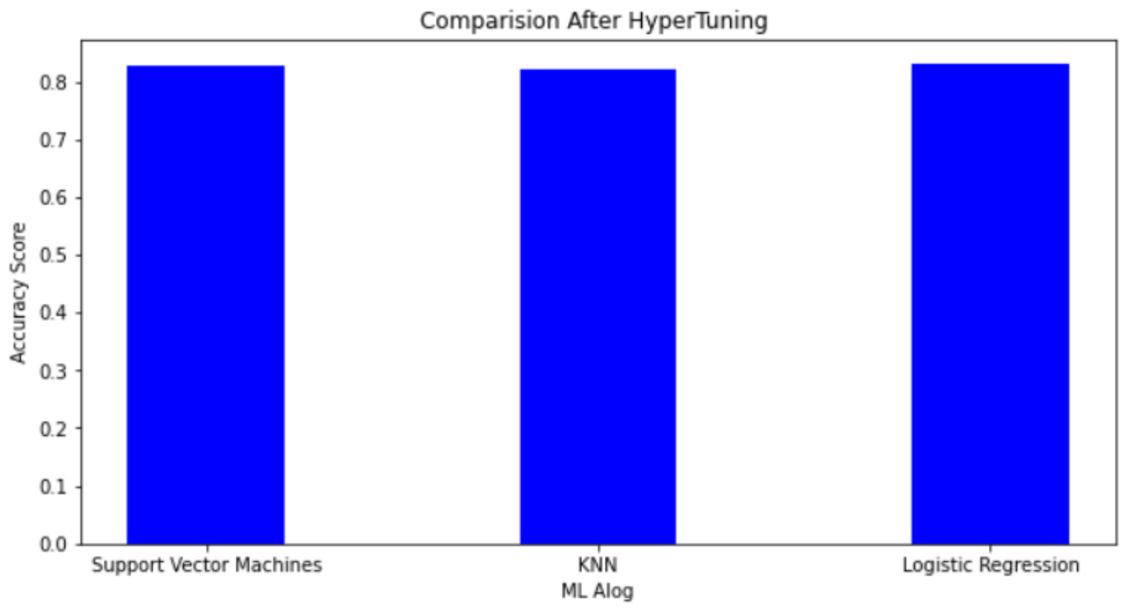
```
Best parameters {'weights': 'uniform'}
Mean cross-validated accuracy score of the best_estimator: 0.740
Accuracy score of training data :  0.8204393505253104
Accuracy score of test data :  0.7213740458015268
Classification Report For Train
              precision    recall  f1-score   support

         0.0       0.85      0.92      0.88       772
         1.0       0.71      0.53      0.61       275

    accuracy                           0.82      1047
   macro avg       0.78      0.73      0.75      1047
weighted avg       0.81      0.82      0.81      1047

Confusion Matrix:
 [[714  58]
 [130 145]]

Classification Report For Test
              precision    recall  f1-score   support

         0.0       0.79      0.86      0.82       195
         1.0       0.44      0.31      0.37        67

    accuracy                           0.72       262
   macro avg       0.61      0.59      0.59       262
weighted avg       0.70      0.72      0.70       262

Confusion Matrix:
 [[168  27]
 [ 46  21]]
```

Model Comparison After HyperParameters

	TrainScore	TestScore
Model		
Support Vector Machines	0.826170	0.824427
Logistic Regression	0.830946	0.832061
KNN	0.820439	0.721374


Algorithm Comparison
0.90
0.85
0.80
0.75
0.70
Support Vector Machines
KNN
Logistic Regression
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Accuracy Comparision Chart
Accuracy Score
0.0
0.2
0.4
0.6
0.8
Support Vector Machines
KNN
Logistic Regression
ML Alog


Comparision After HyperTuning
Accuracy Score
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Support Vector Machines
KNN
Logistic Regression
ML Alog

Conclusion

As a result of our work, we gained valuable experience of building prediction systems and achieved our best score on Kaggle:Titanic DataSet 82.04%

- We performed featured engineering techniques
    - Changed alphabetic values to numeric
    - Used linear regression algorithm to fill in missing ages
- We used several prediction algorithms in python
    - Logistic Regression
    - KNN
    - SVM
- We achieved our best score 82.04% correct predictions