PROJECT 2 : TITANIC PASSENGER SURVIVAL – MACHINE LEARNING

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

1 . INTRODUCTION

In this application we will go through the whole process of creating a machine learning model on the famous **Titanic** dataset, which is used by many people all over the world. It provides information on the fate of passengers on the Titanic, summarized according to economic status (class), sex, age and survival.

The goal of the project is to predict the survival of passengers based off a set of data. We used the **Kaggle** for the dataset to retrieve necessary data and evaluate accuracy of our predictions.

2 . ABOUT THE DATASET

The historical data has been split into two groups, a 'training set' and a 'test set'. both of which are .csv files. The training set includes the response variable Survived and 11 other descriptive variables pertaining to 891 passengers. The test set does not include the response variable, but does contain the 11 other variables for 418 passengers. Note that the 418 passengers in the test dataset are different from the 891 passengers in the training dataset. It is to my knowledge that Kaggle did not specify the details as to how the training and test datasets were chosen. At least, one can assume that individuals were selected at random to form the training and test data sets.

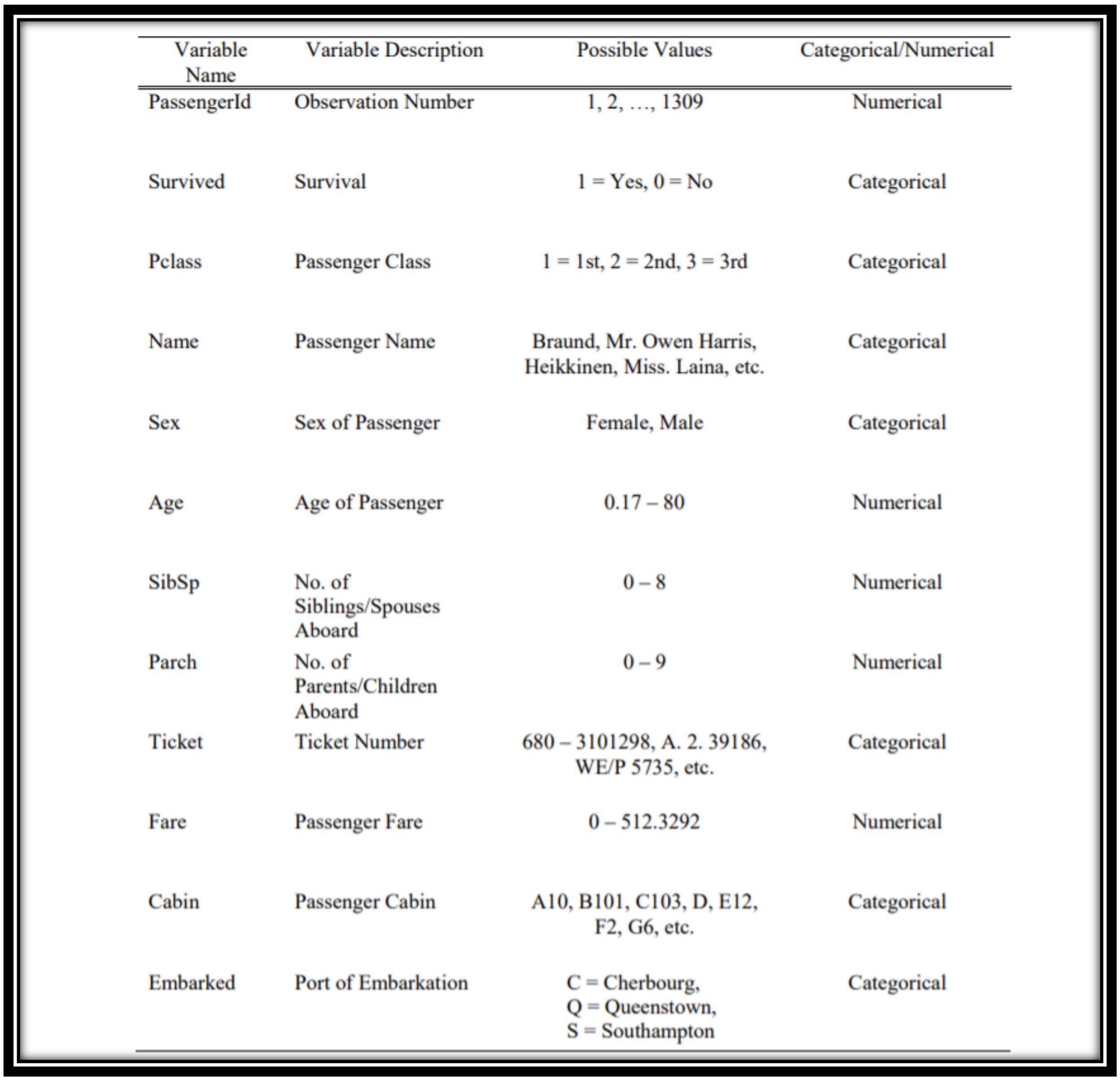
For the training set, we are provided with the outcome (whether or not a passenger survived). We used this set to build our model to generate predictions for the test set. For each passenger in the test set, we had to predict whether not they survived the sink.

3 . TRAIN AND TEST DATA

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

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

- Fare
- Cabin
- City where passenger embarked



Variable Name	Variable Description	Possible Values	Categorical/Numerical
PassengerId	Observation Number	1, 2, …, 1309	Numerical
Survived	Survival	1 = Yes, 0 = No	Categorical
Pclass	Passenger Class	1 = 1st, 2 = 2nd, 3 = 3rd	Categorical
Name	Passenger Name	Braund, Mr. Owen Harris, Heikkinen, Miss. Laina, etc.	Categorical
Sex	Sex of Passenger	Female, Male	Categorical
Age	Age of Passenger	0.17 – 80	Numerical
SibSp	No. of Siblings/Spouses Aboard	0 – 8	Numerical
Parch	No. of Parents/Children Aboard	0 – 9	Numerical
Ticket	Ticket Number	680 – 3101298, A. 2. 39186, WE/P 5735, etc.	Categorical
Fare	Passenger Fare	0 – 512.3292	Numerical
Cabin	Passenger Cabin	A10, B101, C103, D, E12, F2, G6, etc.	Categorical
Embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton	Categorical

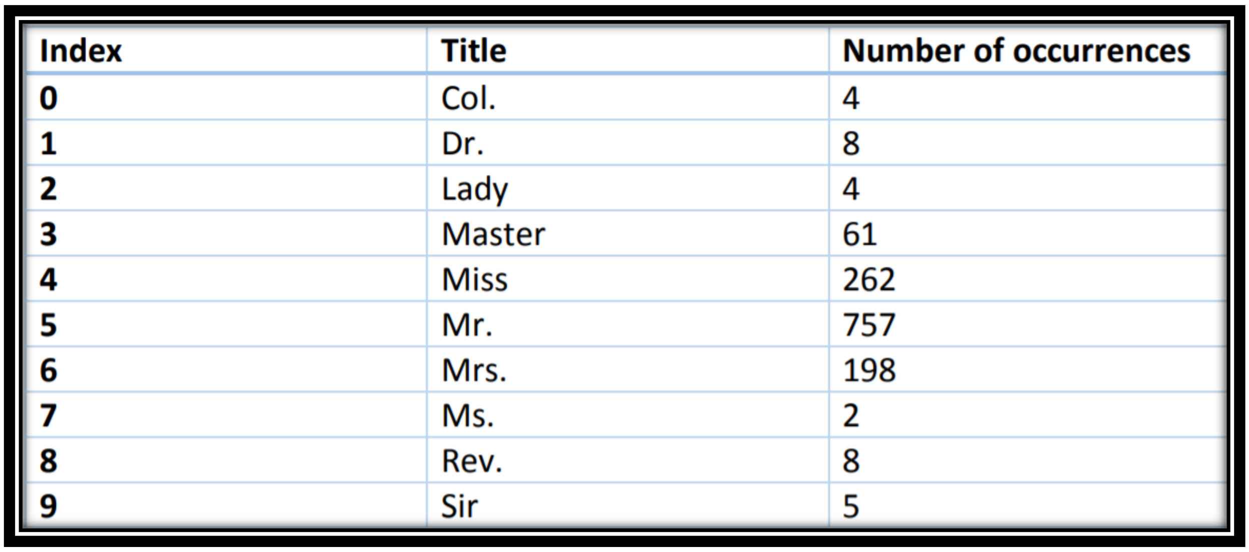
4 . CODE OF THE PROJECT

The GitHub repository of the project is:
https://github.com/Disha6/Titanic-survival

5 . FEATURE ENGINEERING

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.

EXTRACTING TITLE FROM NAME:



Index	Title	Number of occurrences
0	Col.	4
1	Dr.	8
2	Lady	4
3	Master	61
4	Miss	262
5	Mr.	757
6	Mrs.	198
7	Ms.	2
8	Rev.	8
9	Sir	5

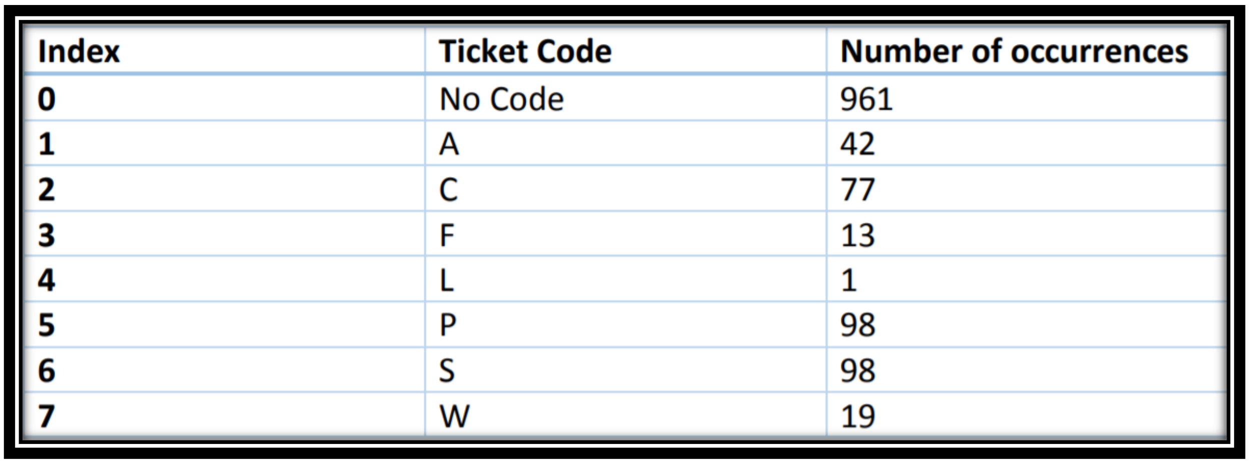
We can see that title may indicate passenger's sex (Mr. vs Mrs.), class (Lady vs Mrs.), age (Master vs Mr.), profession (Col., Dr., and Rev.).

CALCULATING FAMILY SIZE:

It seems advantageous to calculate family size as follows :

Family_Size = Parents_Children + Siblings_Spouses + 1

EXTRATING TICKET_CODE FROM TICKET:



Index	Ticket Code	Number of occurrences
0	No Code	961
1	A	42
2	C	77
3	F	13
4	L	1
5	P	98
6	S	98
7	W	19

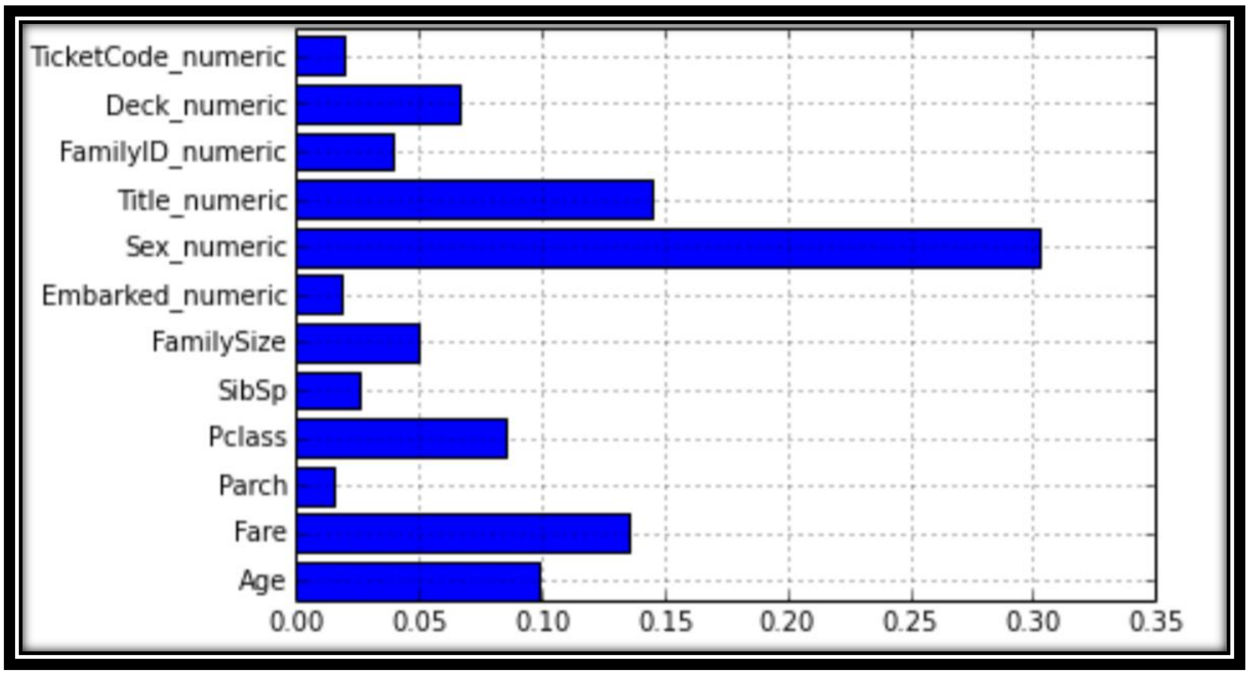
FILLING IN MISSING VALUE IN THE FIELDS:

Since the number of missing values was small, we used median of all Fare values to fill in missing Fare fields, and the letter 'S' (most frequent value) for the field Embarked. In the training and test data, there was significant amount of missing Ages. To fill in those, we used Linear Regression algorithm to predict Ages based on all other fields except Passenger_ID and Survived.

IMPORTANCE OF FIELDS:

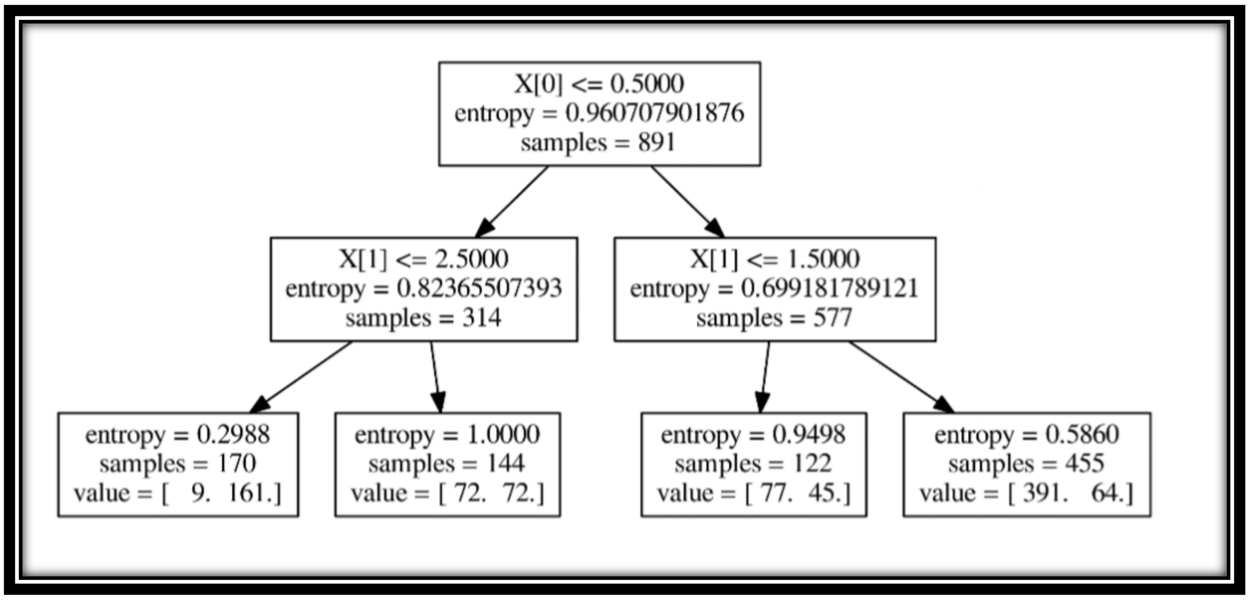
Decision Trees algorithm in the library SciKit-Learn allows to evaluate importance of each field used for prediction. Below is the chart displaying importance of each field.

We can see that the field Sex is the most important one for prediction, followed by Title, Fare, Age, Class, Deck, Family_Size, etc.



DECISION TREE:

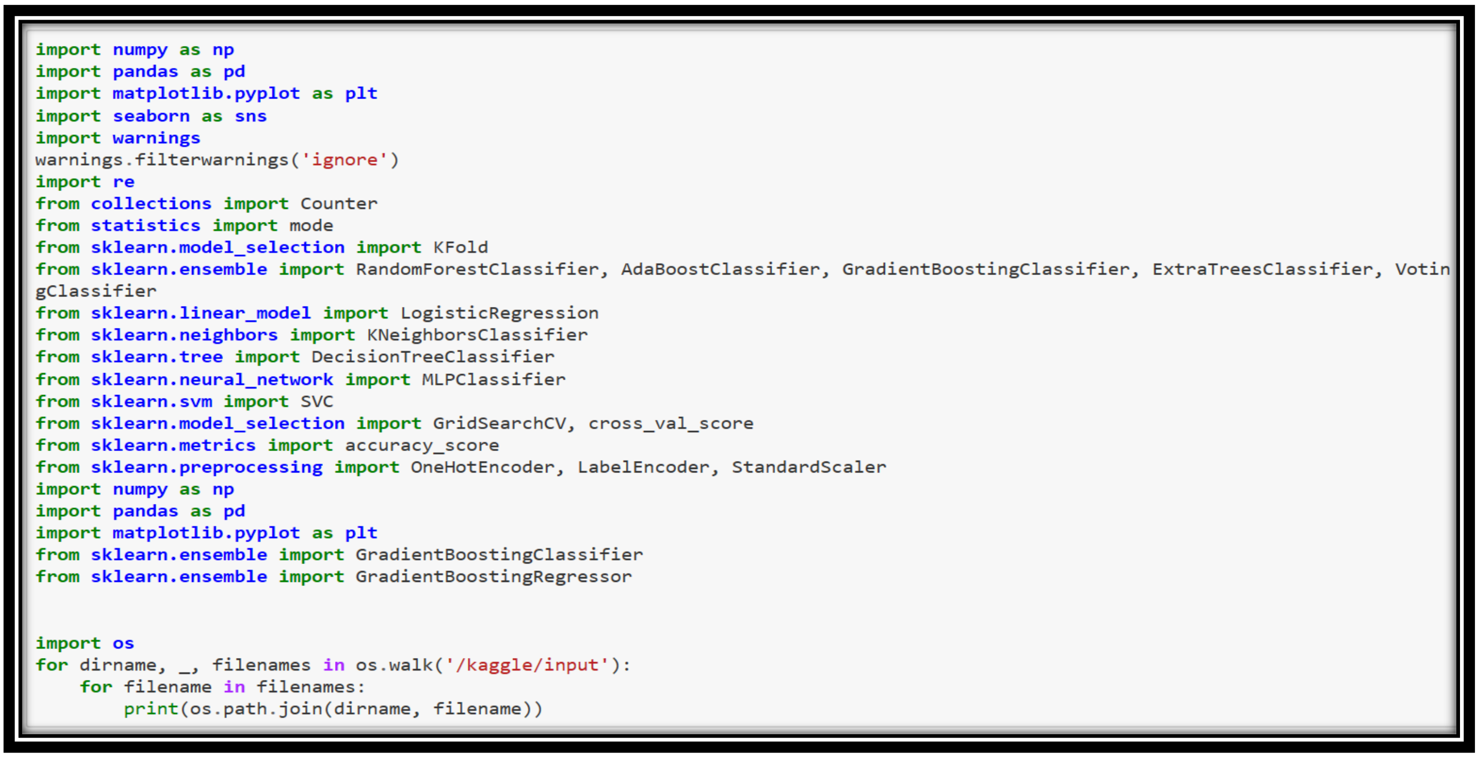
Our prediction system is based on growing Decision Trees to predict the survival status. A typical Decision Tree is pictured below:



6 . EXPLANATION OF CODE

a . Importing libraries:

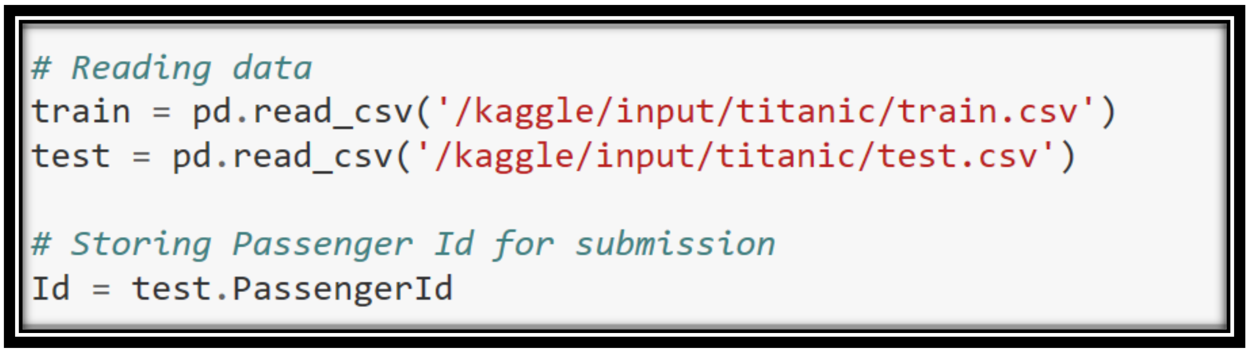
At the start of every notebook we must import necessary libraries in order to simply do Data Science.



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import re
from collections import Counter
from statistics import mode
from sklearn.model_selection import KFold
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, ExtraTreesClassifier, Votin
gClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import OneHotEncoder, LabelEncoder, StandardScaler
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import GradientBoostingRegressor


import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

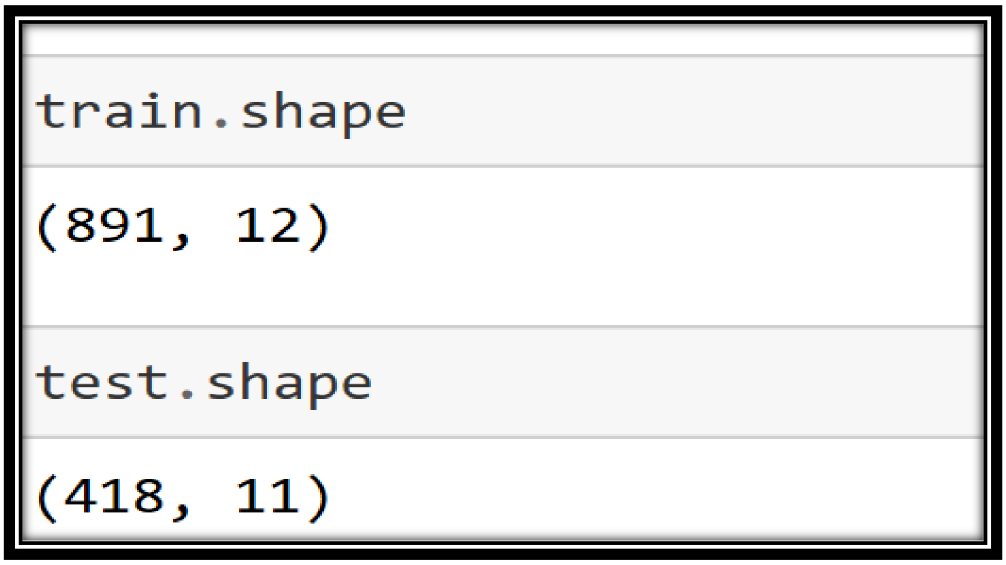
b . Reading Data:

In this section we read in the data and check the data we are working with. As we check the data we can clearly see the data's dimensions and it's features.



```
# Reading data
train = pd.read_csv('/kaggle/input/titanic/train.csv')
test = pd.read_csv('/kaggle/input/titanic/test.csv')

# Storing Passenger Id for submission
Id = test.PassengerId
```

For checking the shape of dataset.



```
train.shape
```

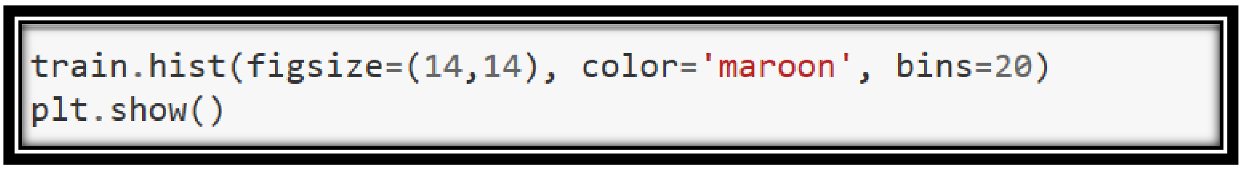
```
(891, 12)
```

```
test.shape
```

```
(418, 11)
```

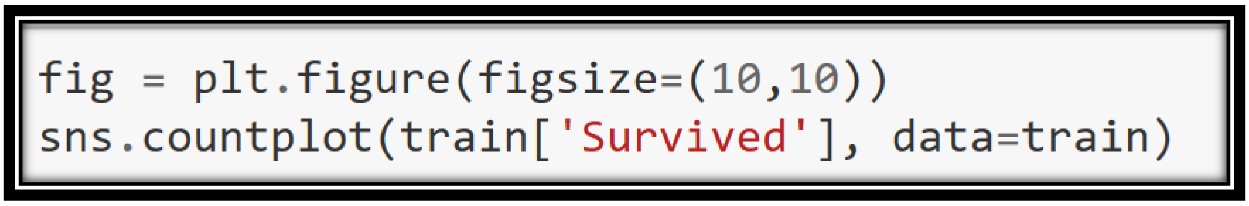
c . Data Visualisation:

In this section we view each feature's survival rate using barplots and factorplots. Using the visualizations we make assumptions and find correlations between different features.

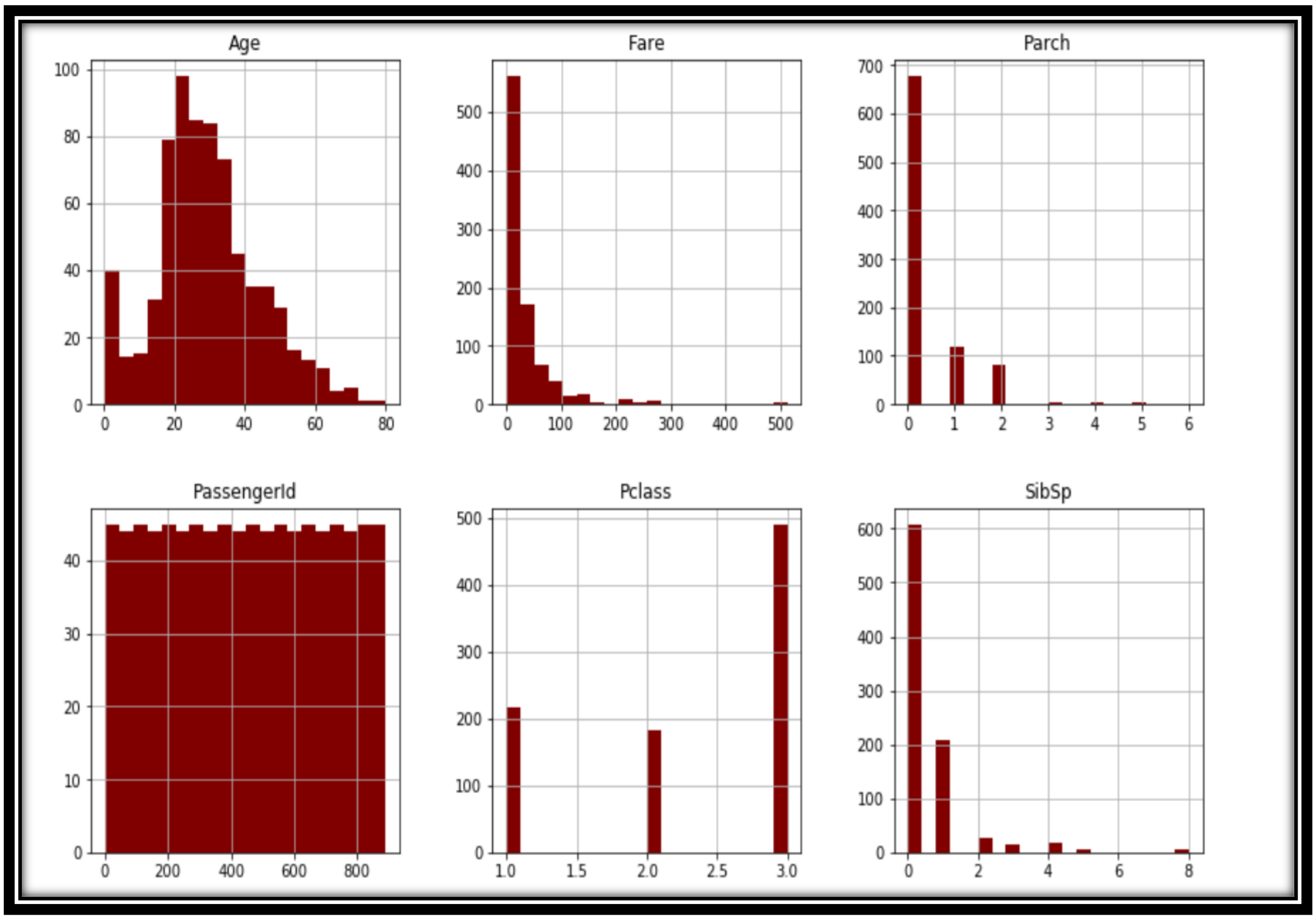


```
train.hist(figsize=(14,14), color='maroon', bins=20)
plt.show()
```

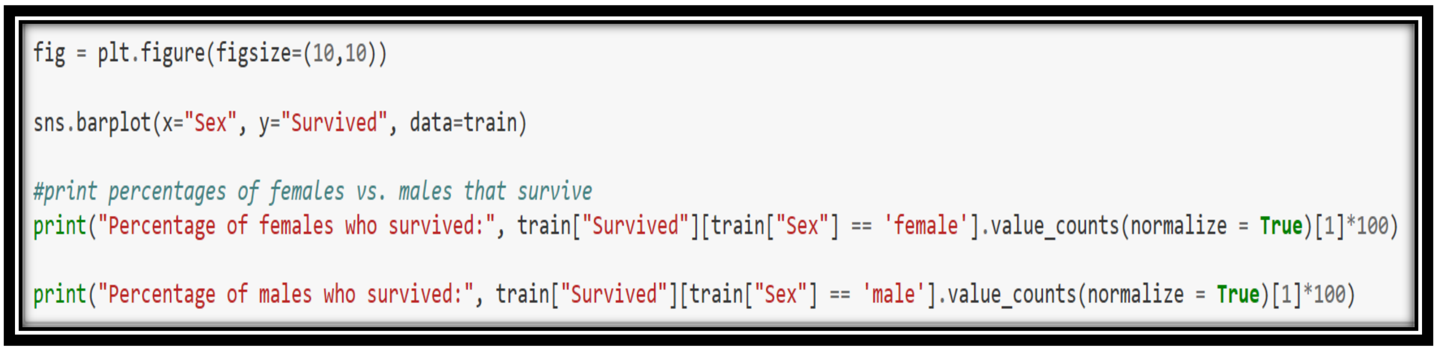
We can see that overall most passengers didn't survive.



```
fig = plt.figure(figsize=(10,10))
sns.countplot(train['Survived'], data=train)
```



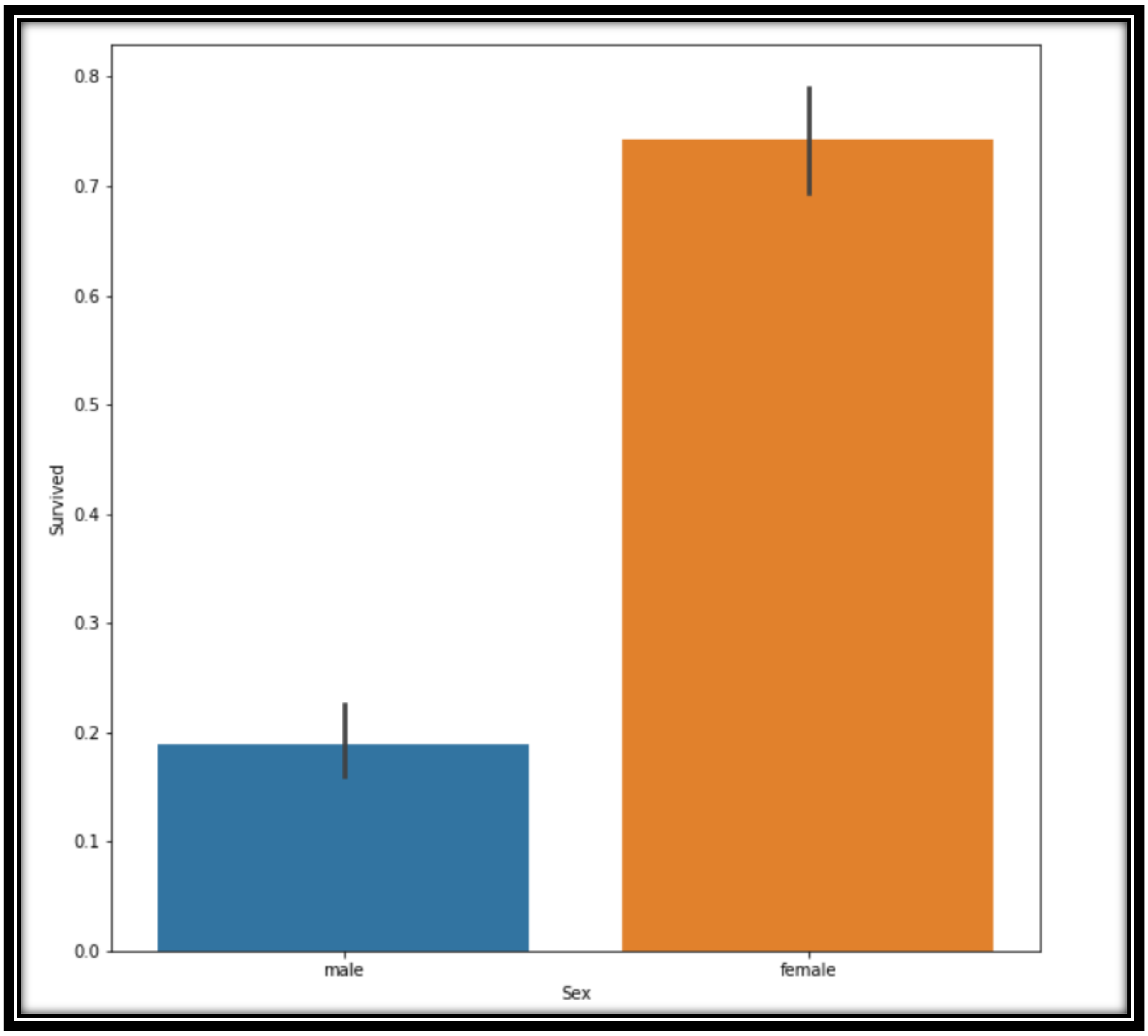
d . Sex Visualisation:



```
fig = plt.figure(figsize=(10,10))

sns.barplot(x="Sex", y="Survived", data=train)

#print percentages of females vs. males that survive
print("Percentage of females who survived:", train["Survived"][train["Sex"] == 'female'].value_counts(normalize = True)[1]*100)

print("Percentage of males who survived:", train["Survived"][train["Sex"] == 'male'].value_counts(normalize = True)[1]*100)
```

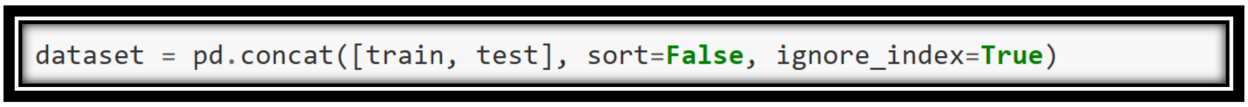
Females have a much higher chance of Survival than men.

.



e . Data Preparation:

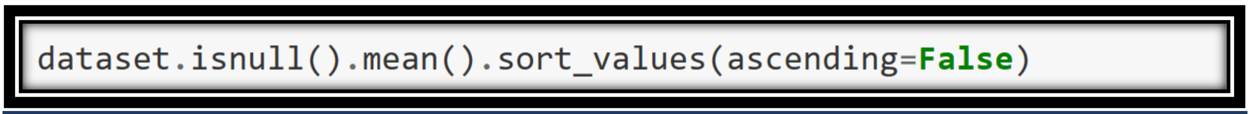
We find correlations between features and impute missing values using the correlations. For categorical features we map the feature so it becomes a numerical feature. Using Feature Engineering we are able to create new features 'Title' and 'FamSize' which will overall improve our model performance.



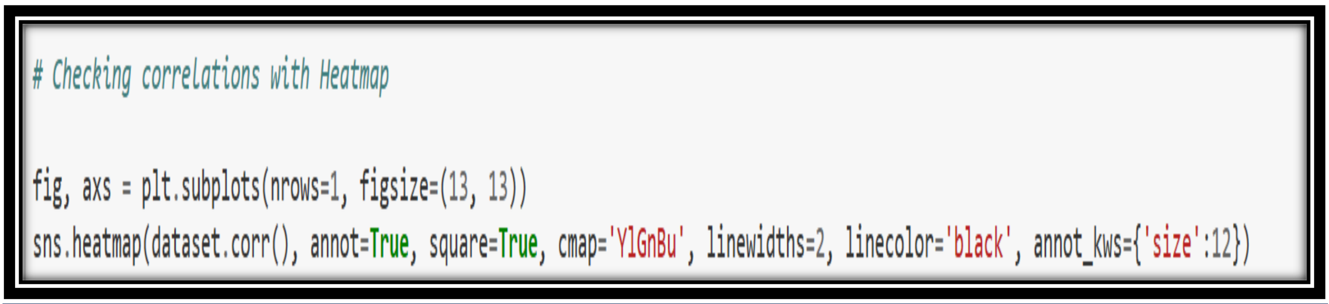
```python
dataset = pd.concat([train, test], sort=False, ignore_index=True)
```

Observations for Imputing Missing Values:

- Cabin is missing 77.4% of its values. We will remove this column
- Age is missing about 20% of its values. These are imputable therefore we probably will keep this column.
- Embarked and Fare are missing less than 1% of their values. We are definitely keeping these columns.

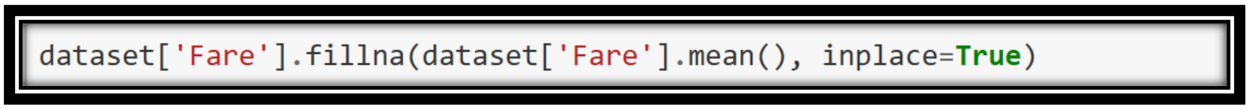


```python
dataset.isnull().mean().sort_values(ascending=False)
```

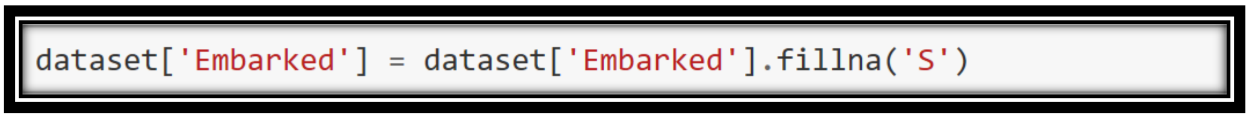


```python
# Checking correlations with Heatmap

fig, axs = plt.subplots(nrows=1, figsize=(13, 13))
sns.heatmap(dataset.corr(), annot=True, square=True, cmap='YlGnBu', linewidths=2, linecolor='black', annot_kws={'size':12})
```

f . Filling the missing values:



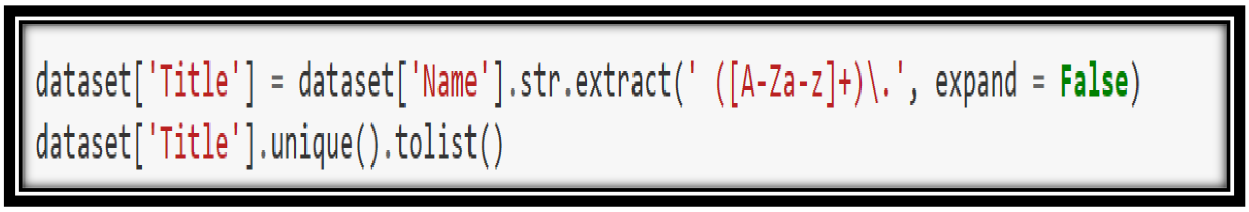
```python
dataset['Fare'].fillna(dataset['Fare'].mean(), inplace=True)
```



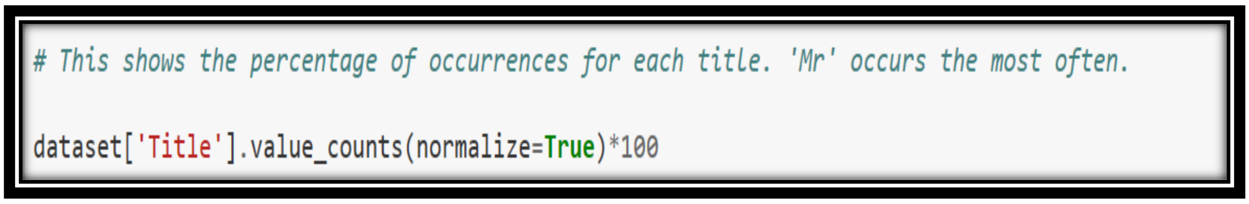
```python
dataset['Embarked'] = dataset['Embarked'].fillna('S')
```

g . Introducing new features:

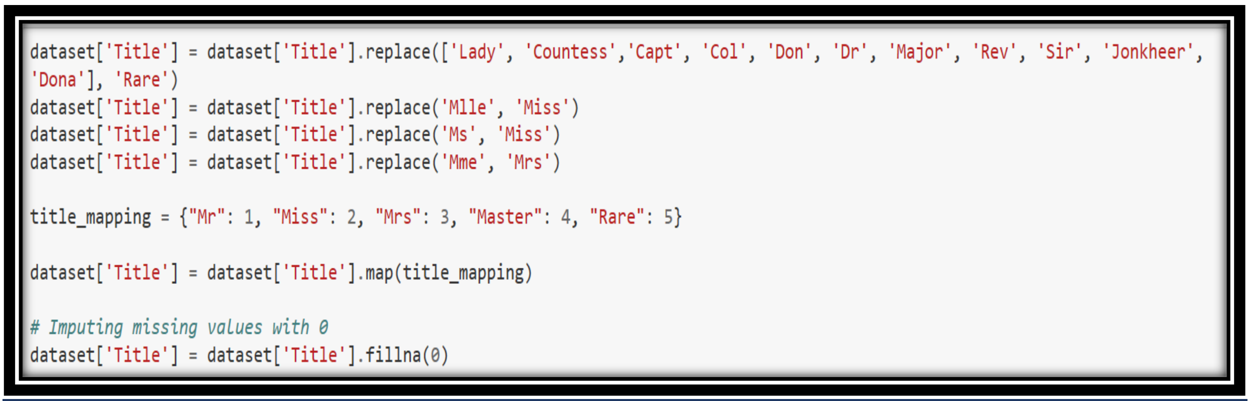
We are basically creating a new column 'Title' and extracting the Title from the 'Name' column.



```
dataset['Title'] = dataset['Name'].str.extract(' ([A-Za-z]+)\.', expand = False)
dataset['Title'].unique().tolist()
```



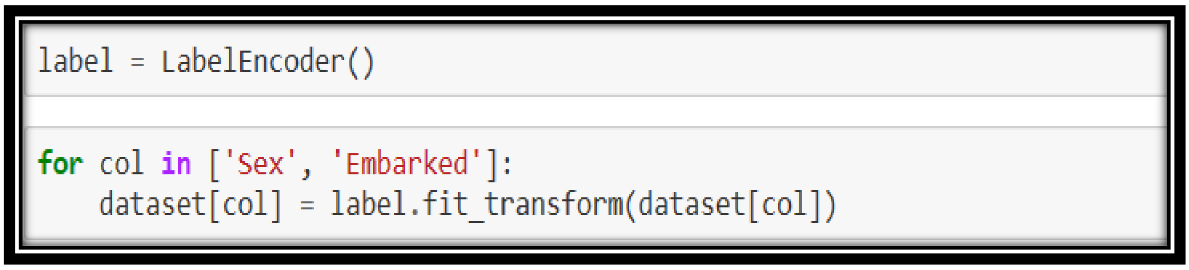
```
# This shows the percentage of occurrences for each title. 'Mr' occurs the most often.

dataset['Title'].value_counts(normalize=True)*100
```



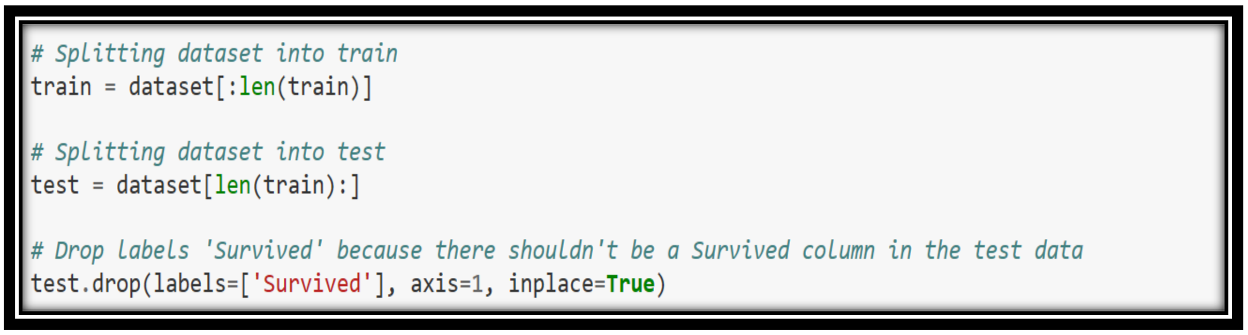
```
dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col', 'Don', 'Dr', 'Major', 'Rev', 'Sir', 'Jonkheer',
'Dona'], 'Rare')
dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}

dataset['Title'] = dataset['Title'].map(title_mapping)

# Imputing missing values with 0
dataset['Title'] = dataset['Title'].fillna(0)
```

h .Modeling:

Splitting dataset into train and test.



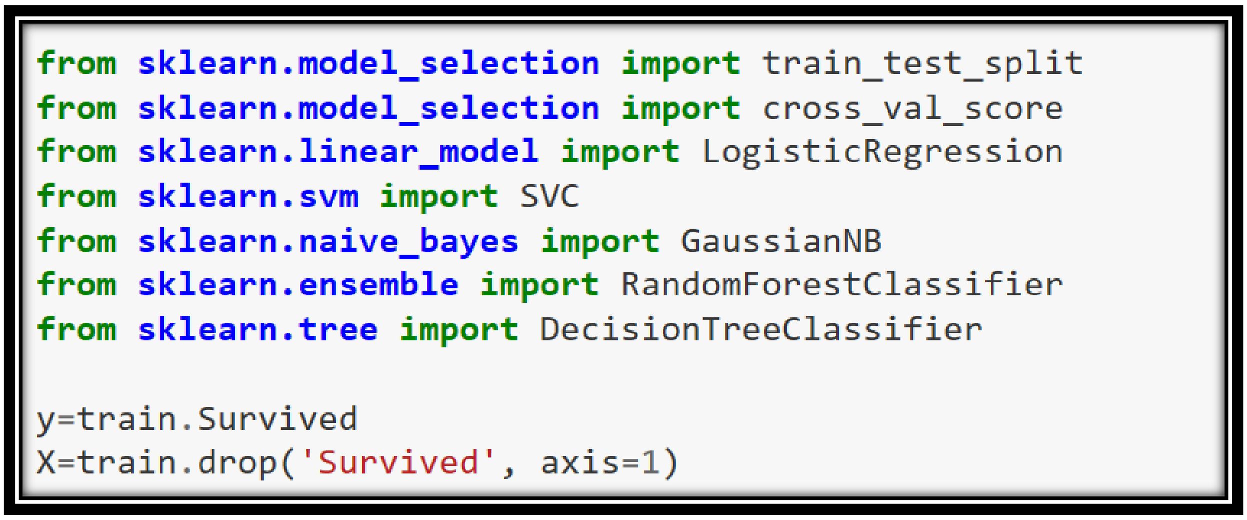
```
label = LabelEncoder()

for col in ['Sex', 'Embarked']:
    dataset[col] = label.fit_transform(dataset[col])
```



```
# Splitting dataset into train
train = dataset[:len(train)]

# Splitting dataset into test
test = dataset[len(train):]

# Drop labels 'Survived' because there shouldn't be a Survived column in the test data
test.drop(labels=['Survived'], axis=1, inplace=True)
```

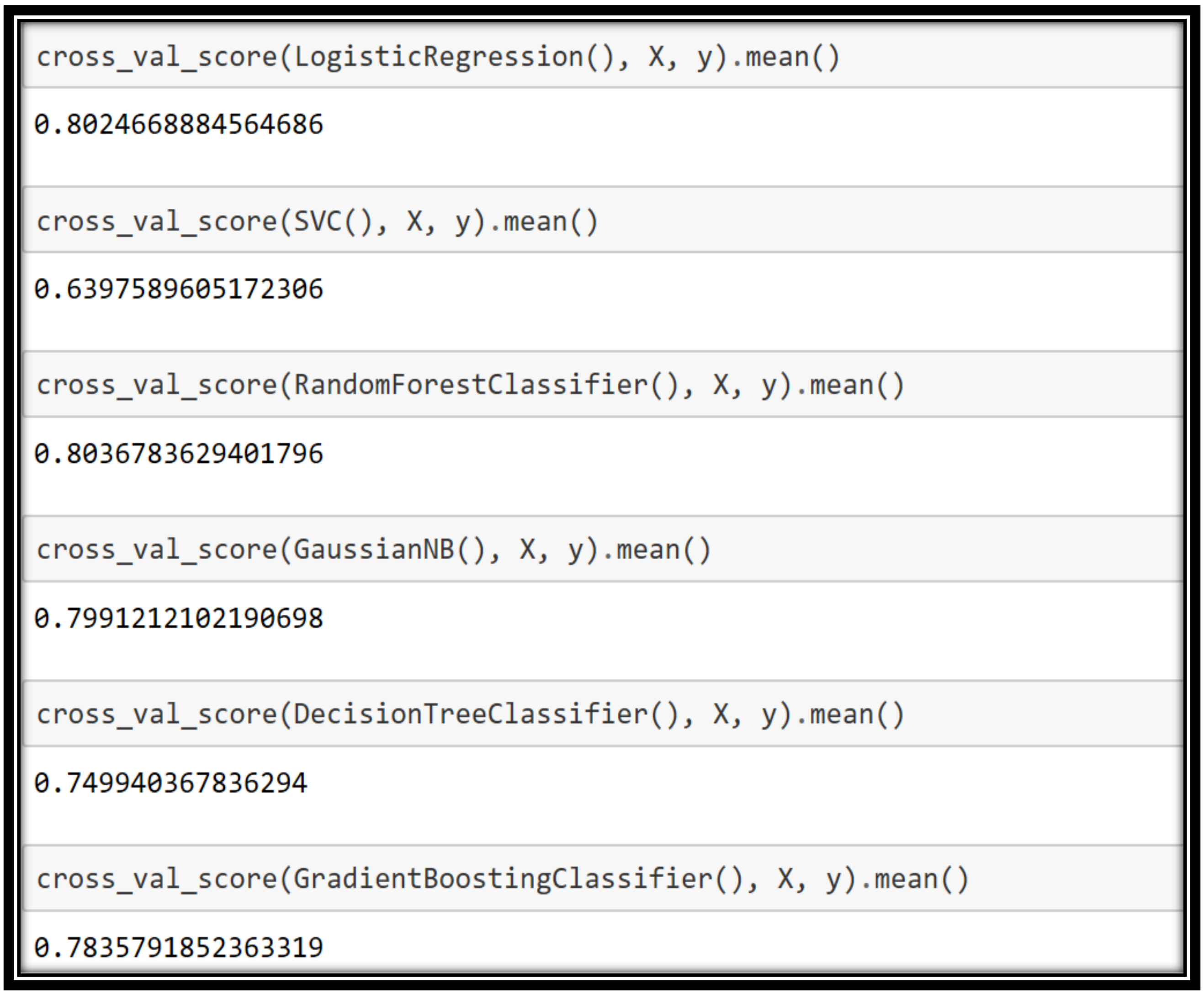
i . Checking the Percentage:

Firstly importing the libraries.



```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

y=train.Survived
X=train.drop('Survived', axis=1)
```

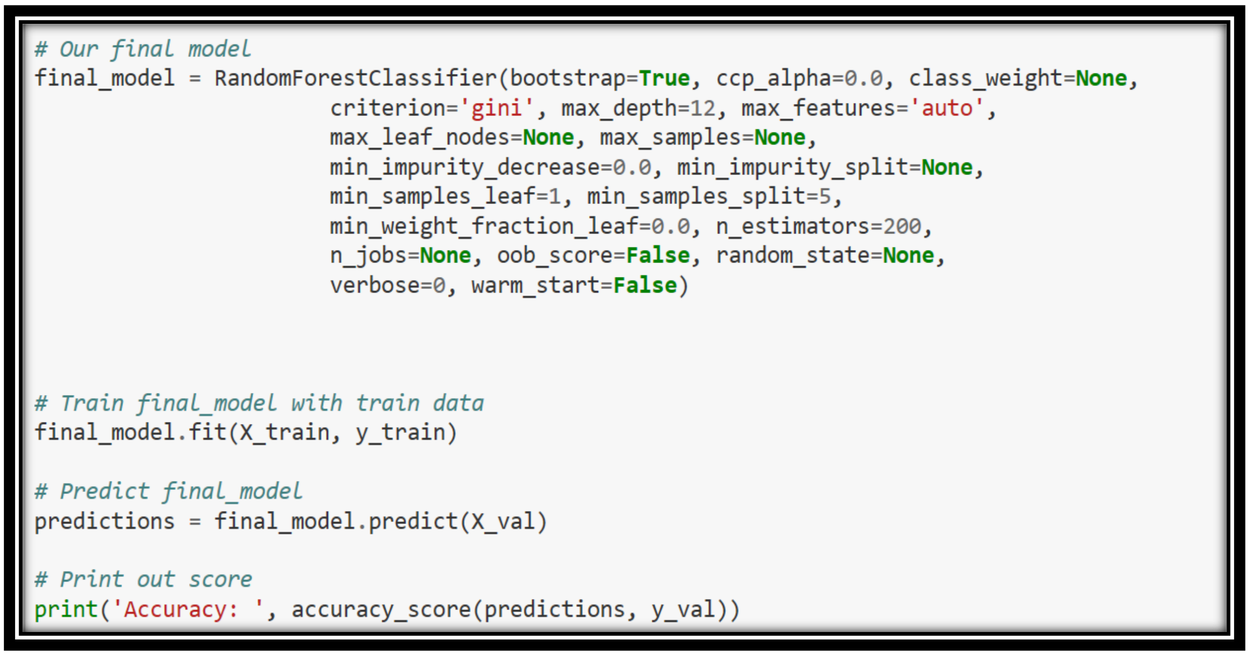
Checking randomly with different methods.



```
cross_val_score(LogisticRegression(), X, y).mean()

0.8024668884564686

cross_val_score(SVC(), X, y).mean()

0.6397589605172306

cross_val_score(RandomForestClassifier(), X, y).mean()

0.8036783629401796

cross_val_score(GaussianNB(), X, y).mean()

0.7991212102190698

cross_val_score(DecisionTreeClassifier(), X, y).mean()

0.749940367836294

cross_val_score(GradientBoostingClassifier(), X, y).mean()

0.7835791852363319
```

j . Final Model:

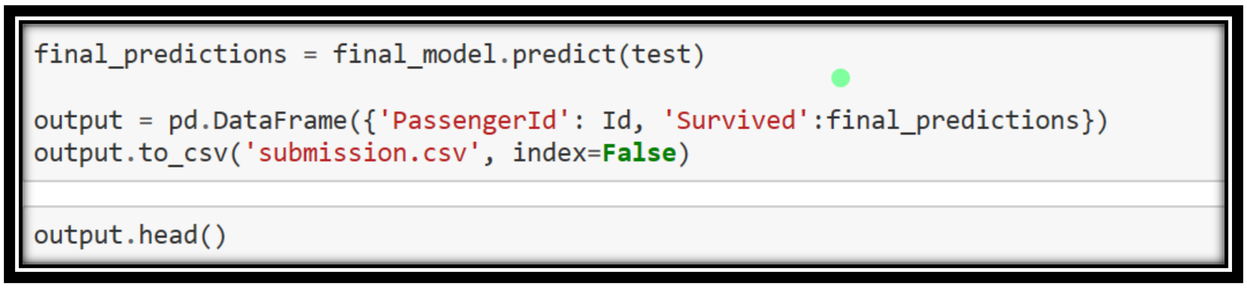
Checking for the accuracy percentage:



```
# Our final model
final_model = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=12, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=5,
                       min_weight_fraction_leaf=0.0, n_estimators=200,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)



# Train final_model with train data
final_model.fit(X_train, y_train)

# Predict final_model
predictions = final_model.predict(X_val)

# Print out score
print('Accuracy: ', accuracy_score(predictions, y_val))
```

k . Submission:

Giving the answer only that the person has survived or not.



```
final_predictions = final_model.predict(test)

output = pd.DataFrame({'PassengerId': Id, 'Survived':final_predictions})
output.to_csv('submission.csv', index=False)
```

```
output.head()
```