TITANIC SURVIVAL PREDICTION

Group members

Prakhar Shanker

Piyush Agrawal

Debasis Tripathy

6/20/20

EICT PROJECT

INTRODUCTION OF TITANIC PROBLEM

- 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.
- In this problem, we have to build a predictive model that answers the question: “what sorts of people were more likely to survive?” using passenger data (i.e. name, age, gender, socio-economic class, etc).

AIM OF THE PROJECT

- It is our job to predict if a passenger had survived the sinking of the Titanic or not.
- For each in the test set, we must predict a 0 or 1 value for the variable

REQUIREMENTS

- SOFTWARE REQUIREMENTS
  - Spyder
  - Jyupiter Notebook
- LIBRARIES USED
  - Analysis : NumPy, Pandas, Scikit Learn
- Visualization: Matplotlib, Graphviz

STEPS TO IMPLEMENT

- Importing the necessary Libraries
- Importing the Dataset
- Cleaning and analysing the Dataset
- Building the model
- Using different numbers of algorithms in classification techniques

IMPORTING NECESSARY LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import matplotlib.pyplot as plt
%matplotlib inline
import pydotplus
from sklearn import tree
from sklearn.tree import export_graphviz
from IPython.display import Image
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier as KNN
from sklearn.model_selection import cross_val_score
```

LOADING AND EXPLORING THE DATA

```python
dataset=pd.read_csv("train.csv")
```

```python
dataset.head(10)
```

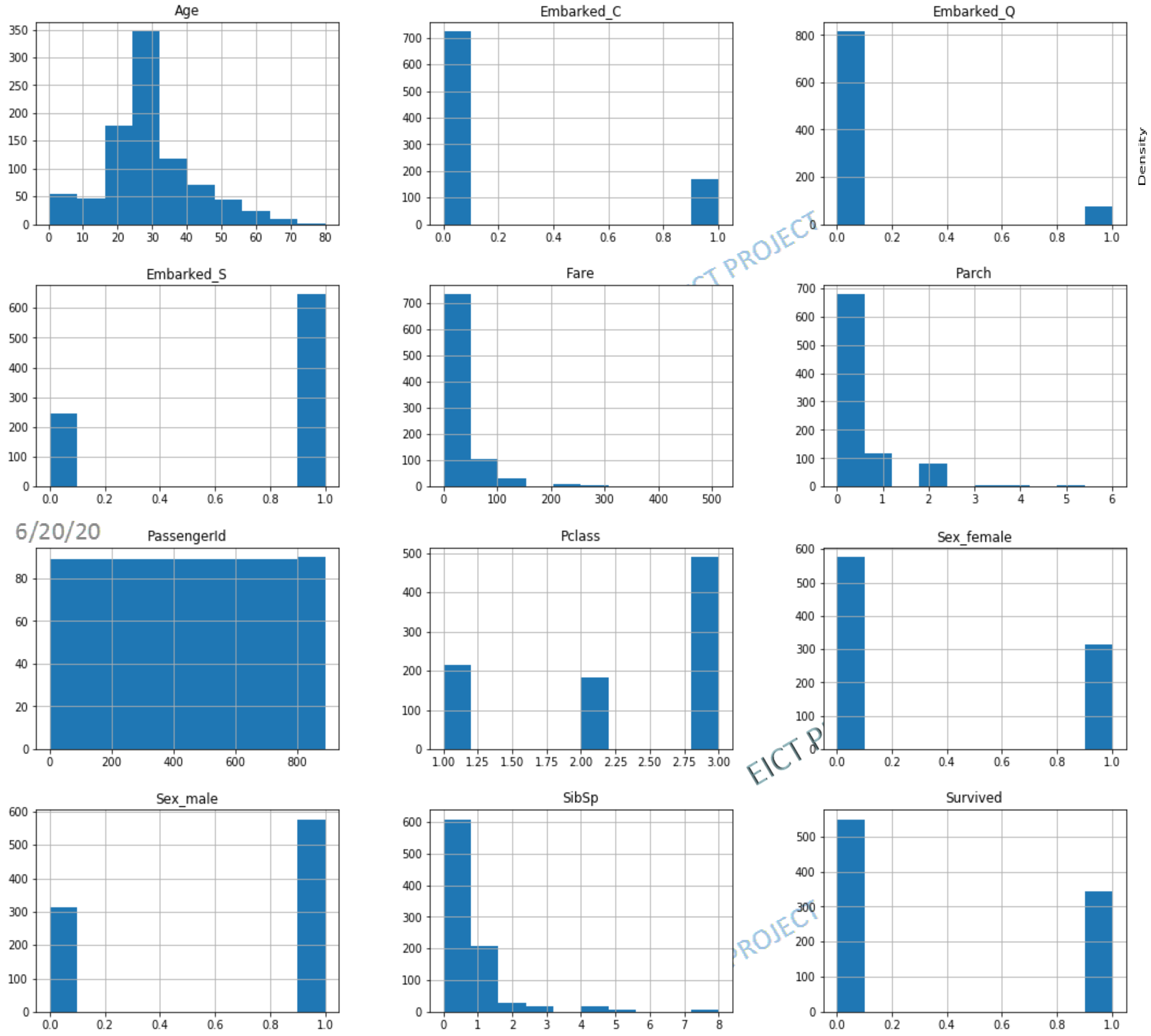
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C

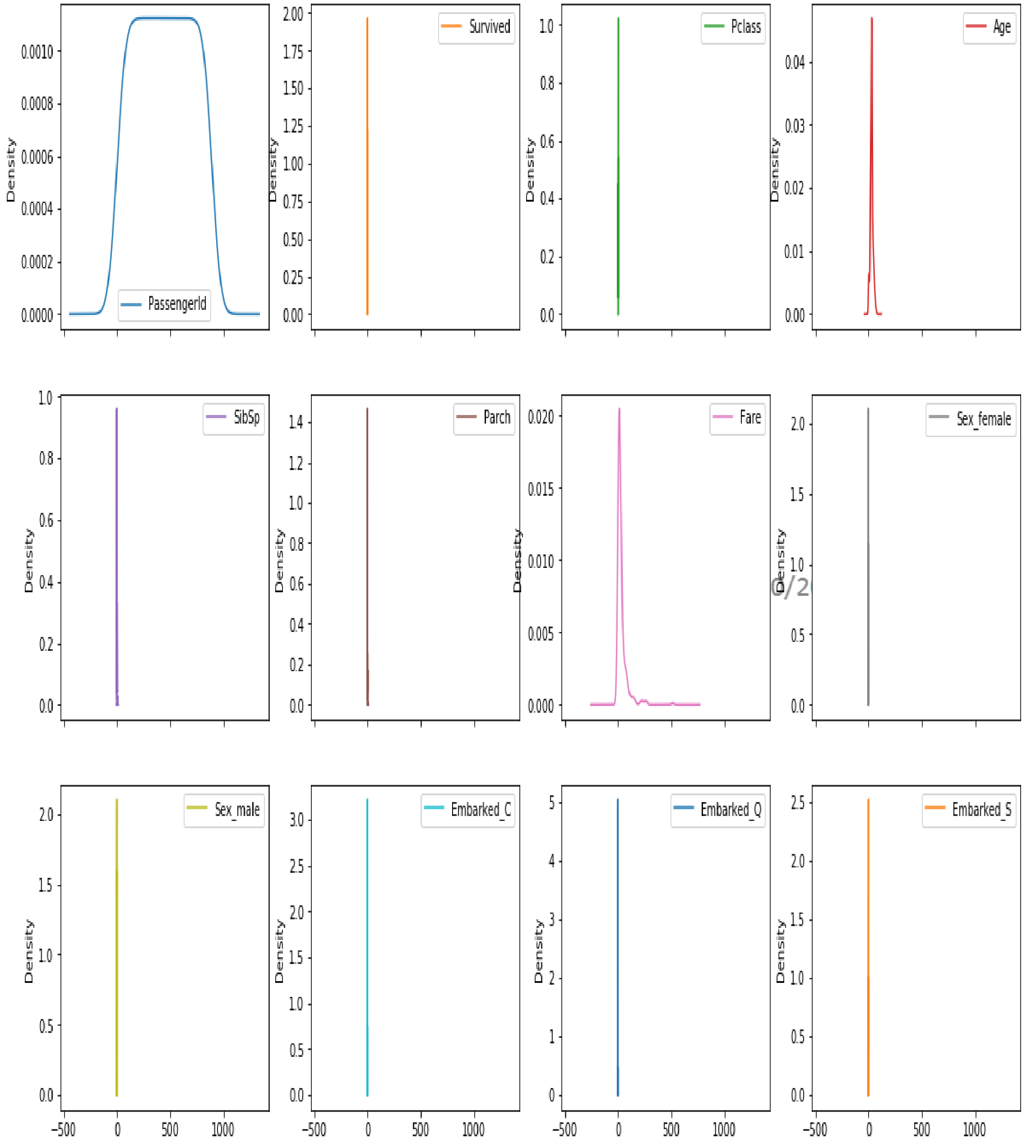
DATA ANALYSIS

```python
df.describe()
```

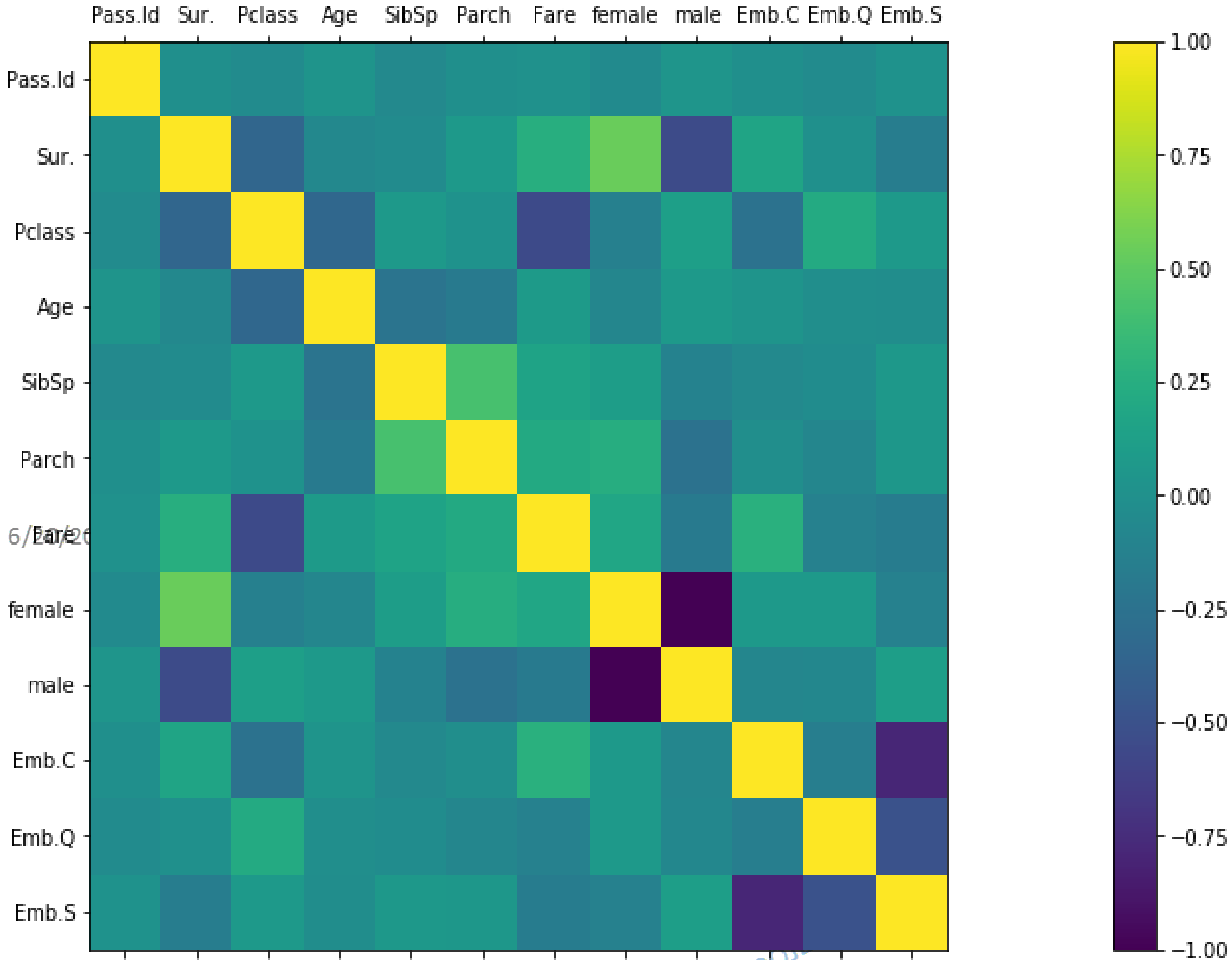
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208	0.352413	0.647587	0.188552	0.086420	0.725028
std	257.353842	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429	0.477990	0.477990	0.391372	0.281141	0.446751
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400	0.000000	0.000000	0.000000	0.000000	0.000000
50%	446.000000	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200	0.000000	1.000000	0.000000	0.000000	1.000000
75%	668.500000	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000	1.000000	1.000000	0.000000	0.000000	1.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200	1.000000	1.000000	1.000000	1.000000	1.000000

COMPLETE DATA VISUALIZATION AND DECIDING IMPORTANT FACTORS :





VISUALIZING MORE CLEARLY BY GRAPH



Explanation of the graph

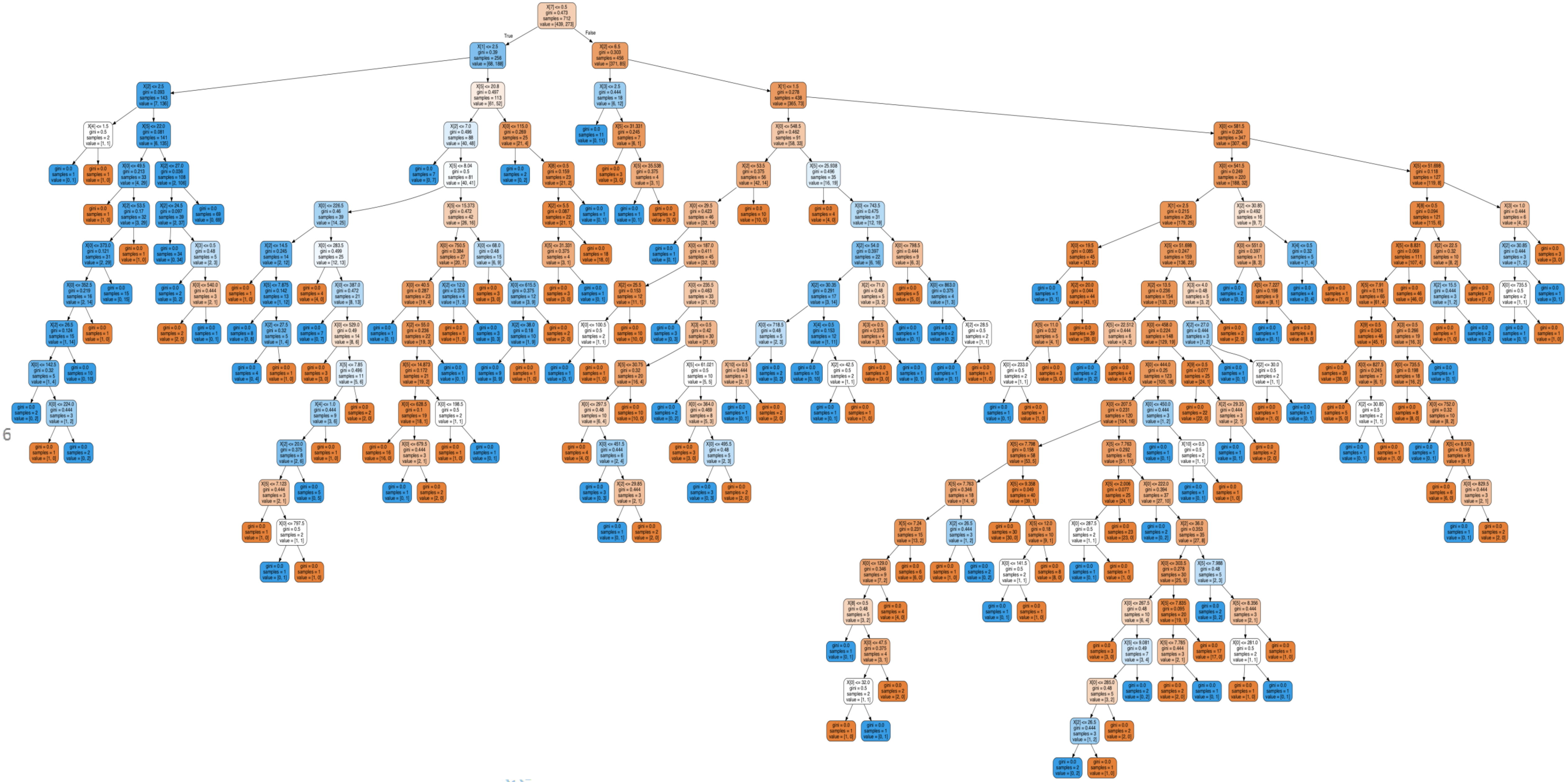
- From this visualization correlation of categories with one another is more positive towards yellow and most negative towards dark blue.
- This shows the correlation between our data. And we can find out categories which are corelated with survival rate most extremely.
- From this it is also very difficult to find out.

BUILDING A DECISION TREE CLASSIFIER MODEL

Building a Decision Tree Model

```
In [29]: #Importing Decision Tree Classifier
         from sklearn.tree import DecisionTreeClassifier

In [30]: #creating a decision tree instance
         clf = DecisionTreeClassifier(random_state=96)

In [31]: #training the model
         clf.fit(train_X,train_Y)

Out[31]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                                max_depth=None, max_features=None, max_leaf_nodes=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=96, splitter='best')

In [32]: #calculating score on training data
         clf.score(train_X, train_Y)

Out[32]: 0.922752808988764

In [33]: #score on test data
         clf.score(test_X, test_Y)

Out[33]: 0.8100558659217877

In [34]: #looking at the feature importance
         clf.feature_importances_

Out[34]: array([0.14591945, 0.05669768, 0.38729791, 0.37793077, 0.        ,
                0.01486118, 0.00549992, 0.01179308])
```

VISUALIZING THE DECISION TREE MODEL



BUILDING A RANDOM FOREST CLASSIFIER MODEL

Building a Random Forest

```
In [38]: #Importing random forest classifier
         from sklearn.ensemble import RandomForestClassifier

In [39]: #creating a random forest instance
         rclf = RandomForestClassifier(random_state=96)

In [40]: #train the model
         rclf.fit(train_X,train_Y)

Out[40]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, 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=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=96, verbose=0,
                                warm_start=False)
```



```
In [41]: #score on training data
         rclf.score(train_X, train_Y)

Out[41]: 0.922752808988764

In [42]: #score on test data
         rclf.score(test_X, test_Y)

Out[42]: 0.8156424581005587

In [43]: #looking at the feature importance
         clf.feature_importances_

Out[43]: array([0.14591945, 0.05669768, 0.38729791, 0.37793077, 0.        ,
                0.01486118, 0.00549992, 0.01179308])
```


BUILDING A KNN MODEL FOR A BETTER RESULT

```
In [50]: x.head()
```

Out[50]:

	PassengerId	Pclass	Parch	Fare	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	0.000000	1.0	0.0	0.014151	0.0	1.0	0.0	0.0	1.0
1	0.001124	0.0	0.0	0.139136	1.0	0.0	1.0	0.0	0.0
2	0.002247	1.0	0.0	0.015469	1.0	0.0	0.0	0.0	1.0
3	0.003371	0.0	0.0	0.103644	1.0	0.0	0.0	0.0	1.0
4	0.004494	1.0	0.0	0.015713	0.0	1.0	0.0	0.0	1.0

Deviding Total Dataset into Training And Testing Dataset

```
In [51]: from sklearn.model_selection import train_test_split
         train_x,test_x,train_y,test_y = train_test_split(x, y, random_state = 96, stratify=y)

In [52]: #importing KNN classifier and metric F1score

         from sklearn.neighbors import KNeighborsClassifier as KNN

In [53]: from sklearn.model_selection import cross_val_score
         score = cross_val_score( KNN(n_neighbors = 3), X = train_x, y = train_y, cv = 10)
         score

Out[53]: array([0.82089552, 0.73134328, 0.79104478, 0.70149254, 0.70149254,
                0.80597015, 0.86567164, 0.79104478, 0.8030303 , 0.8030303 ])

In [54]: # Consistency using Mean and standard deviation in percentage
         score.mean()*100, score.std()*100

Out[54]: (78.15015829941203, 5.0658719306624)
```

DECIDING THE VALUE OF K USING ELBOW GRAPH WITH THE HELP OF A AUTO ITERATING LOOP

```
n [56]: from sklearn.preprocessing import StandardScaler

n [57]: ss=StandardScaler()
        x=ss.fit_transform(X)

n [58]: y=df['Survived']

n [59]: from sklearn.model_selection import train_test_split
        #divide into train and test sets
        train_X,test_X,train_Y,test_Y = train_test_split(X,Y, random_state = 5, stratify=Y)
        from sklearn.neighbors import KNeighborsClassifier as KNN
        from sklearn.metrics import f1_score

n [60]: KNN()

ut[60]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                             metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                             weights='uniform')
```

Text

```
n [61]: #Creating instance
        clf=KNN(n_neighbors=5,metric='euclidean')
        clf.fit(train_X,train_Y)

ut[61]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                             metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                             weights='uniform')

n [62]: test_predict=clf.predict(test_X)
        K=f1_score(test_predict,test_Y)

In [ ]:

n [63]: print('test f1 score',K)

        test f1 score 0.6900584795321637
```

6/

LOOP CONSTRUCTION FOR K VALUE

```
[64]: K=list
      train_f1=[]
      train_f2=[]

[ ]:

[65]: def Elbow(K):
          test_Error=[]
          for i in K:
           clf=KNN(n_neighbors=i)
           clf.fit(train_X,train_Y)
          tmp=clf.predict(train_X)
          tmp=f1_score(tmp,train_Y)
          train_f1.append(tmp)
          tmp=clf.predict(test_X)
          tmp=f1_score(tmp,test_Y)
          test_f1.append(tmp)
          return train_f1,test_f1

[66]: K=range(1,150)
```

BY CHECKING THE ACCURACY AND COMPARING WITH RESULTS :

RF AND DT ACCURACY:

```
print(  "Decission tree Accuracy: %.3f (%.3f)" % ( results.mean()*100,
                                                   results.std()*100 )   )
```

```
Random forest Accuracy: 80.471 (4.183)
Decission tree Accuracy: 80.469 (4.127)
```


KNN SCORE WITHOUT ELBOW:

```
In [53]: from sklearn.model_selection import cross_val_score
         score = cross_val_score( KNN(n_neighbors = 3), X = train_x, y = train_y, cv = 10)
         score

Out[53]: array([0.82089552, 0.73134328, 0.79104478, 0.70149254, 0.70149254,
                0.80597015, 0.86567164, 0.79104478, 0.8030303 , 0.8030303 ])

In [54]: # Consistency using Mean and standard deviation in percentage
         score.mean()*100, score.std()*100

Out[54]: (78.15015829941203, 5.0658719306624)
```

KNN USING ELBOW SCORE:

```
n [62]: test_predict=clf.predict(test_X)
        K=f1_score(test_predict,test_Y)

In [ ]:

n [63]: print('test f1 score',K)

        test f1 score 0.6900584795321637
```

FINAL CONCLUSION :

- From the above models we can observe that RF and DT are showing similar results while KNN and KNN (using elbow) showing different results.
- This is due to the fact of scaling the input data. In a distance based algorithm where Euclidean distance is used to point out the neighboring points misbehaves if we do not scale the input data correctly.
- But in RF and DT input data is not being scaled and results may not be accurate.
- From using elbow curve with an iterative process we can reach at a conclusion that KNN is giving least better prediction above all other algorithms with a score of 69% of non-survival rate. So the survival rate of passengers on the titanic according to this model is 31%. However we can use Feature Engineering to get better prediction too.
- Random Forest and Decision Tree models giving more importance to Age followed by Sex, Pclass, Fare. Similar pattern is also followed by other models as well. But KNN is taking account and additional feature SbSP.

Thank You