ASSIGNMENT - 2

2228-CSE-5334-004-

DATA MINING

REPORT ON Classification - Nearest Neighbors

SUBMITTED BY

SANJANA KONDABATHINI (1001984312)

MEGHANA KADALI (1002027499)

MUCHARLA RAJASHEKAR (1002027966)

Under the guidance of PROF. Dr. Elizabeth D Diaz
Teaching Assistant Kunal Mehta

DEPARTMENT OF COMPUTER SCIENCE

Nearest Neighbors:

Introduction:

K-Nearest neighbors' algorithm is a supervised algorithm used to solve both classification and regression problem. It is a non-parametric algorithm method of classification. By computing the distance between the test and training points and locating the test point on which class the point closest to it belongs, the K-Nearest Neighbors algorithm attempts to forecast the output class for test data.

KNN Parameters:

n_neighbors: Determines Number of neighbors to use.
Weights {'uniform', 'distance'}**:** 'uniform' - uniform weights and 'distance' : weights by the inverse of their distance.
P: Power parameter for minkowski metric. P = 2 is for Euclidean distance.
Metric: Parameter used for distance.
n_jobs: Number of parallel jobs to run for search.

Data Set Description:

The name of the data set is dataset_NN (dataset_NN.csv). The data set consists of Survived, Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Embarked are the attributes in the data set.

Below is the glimpse of the data set. Here we used head function to get the information of 1st 5 rows and last 5 rows.

In [3]:
```
#First 5 Records
df.head()
```

Out[3]:

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

In [4]:
```
#Last 5 Records
df.tail()
```

Out[4]:

	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

Details of the data set

```
In [5]: #Details of dataset
        df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
Survived    891 non-null int64
Pclass      891 non-null int64
Name        891 non-null object
Sex         891 non-null object
Age         714 non-null float64
SibSp       891 non-null int64
Parch       891 non-null int64
Ticket      891 non-null object
Fare        891 non-null float64
Cabin       204 non-null object
Embarked    889 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 76.6+ KB
```

Detailed description of each attribute in the data set we used the describe function

```
In [6]: #Detailed description of dataset using describe()
        df.describe().transpose()
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Survived	891.0	0.383838	0.486592	0.00	0.0000	0.0000	1.0	1.0000
Pclass	891.0	2.308642	0.836071	1.00	2.0000	3.0000	3.0	3.0000
Age	714.0	29.699118	14.526497	0.42	20.1250	28.0000	38.0	80.0000
SibSp	891.0	0.523008	1.102743	0.00	0.0000	0.0000	1.0	8.0000
Parch	891.0	0.381594	0.806057	0.00	0.0000	0.0000	0.0	6.0000
Fare	891.0	32.204208	49.693429	0.00	7.9104	14.4542	31.0	512.3292

Data Cleaning and Pre-Processing:

Removing unwanted columns from the data set:

In [7]:
```
#Drop the following attributes from the table as there is no logical reason behind the following attributes:
##  1) Name
##  2) Ticket
##  3) Cabin
```

In [8]:
```
df = df.drop('Name', axis=1,)
df = df.drop('Ticket', axis=1,)
df = df.drop('Cabin', axis=1,)
```

Joining SibSp and Parch and 1(to count himself) to get the family count:

In [9]:
```
#Join SibSp, Parch
df['Family_Count'] = df['SibSp'] + df['Parch'] + 1
df = df.drop('SibSp', axis=1,)
df = df.drop('Parch', axis=1,)
```

Checking for null values with isnull() function.

In [10]:
```
#Checking Null values
df.isnull().sum()
```

Out[10]:
```
Survived         0
Pclass           0
Sex              0
Age            177
Fare             0
Embarked         2
Family_Count     0
dtype: int64
```

Handling the null values by filling the null values in Age by its median value and Embarked by its mode value.

In [11]:
```
#Handling Null Values
df["Age"] = df["Age"].fillna(df["Age"].median())
df["Embarked"].mode()
```

Out[11]:
```
0    S
dtype: object
```

In [12]:
```
df["Embarked"] = df["Embarked"].fillna("S")
```

Converting categorical values in Embarked as S – 0, C – 1, Q – 2 and in Sex as male – 0, female – 1.

```
In [15]: df['Embarked']=df['Embarked'].map({'S' : 0, 'C' : 1, 'Q' : 2})
         df['Sex']=df['Sex'].map({'male' : 0, 'female' : 1})
```

```
In [16]: df.head()
```

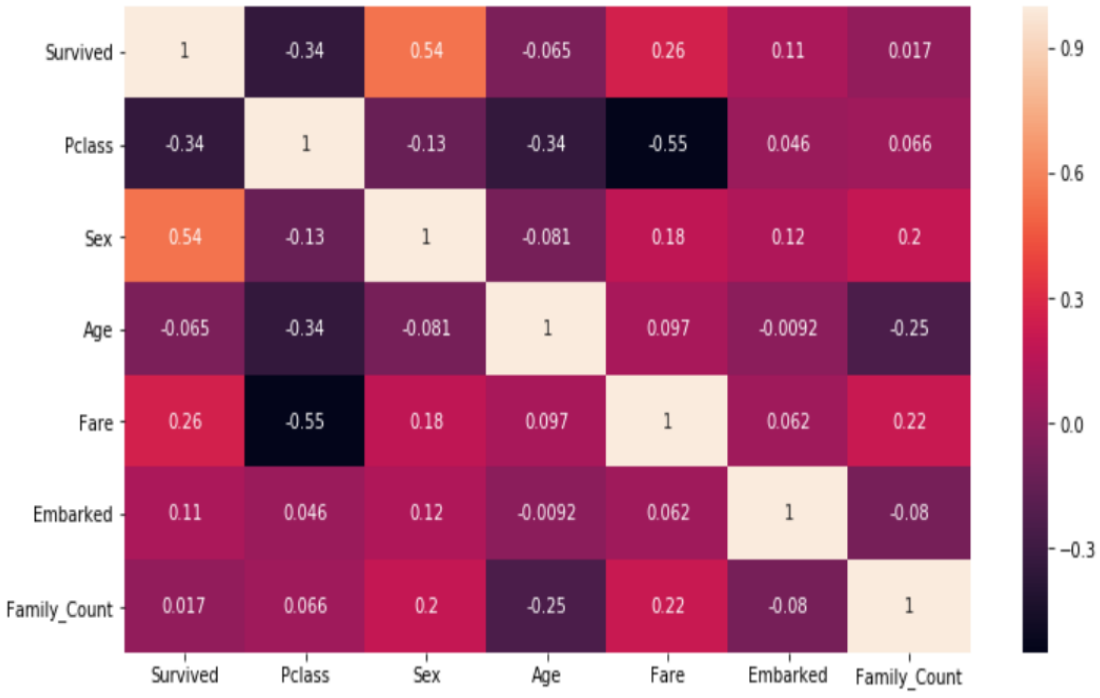
Out[16]:

	Survived	Pclass	Sex	Age	Fare	Embarked	Family_Count
0	0	3	0	22.0	7.2500	0	2
1	1	1	1	38.0	71.2833	1	2
2	1	3	1	26.0	7.9250	0	1
3	1	1	1	35.0	53.1000	0	2
4	0	3	0	35.0	8.0500	0	1

Finding correlation between variables using heatmap:

```
In [17]: #Finding correleation between the varabiles
         plt.figure(figsize=(12, 6))
         sns.heatmap(df.corr(), annot=True)
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x13662305748>



Finding the top most 3 attributes:

By using correlation function we can find the top most attributes with their index. As wee see the attributes Sex, Survived, Fare are the best attributes among all.

In [18]:
```
#Top 3 attributes
print('Important features:')
corr=df.corr()
corr.sort_values(['Embarked'],ascending=False,inplace=True)
corr.Embarked
```

```
Important features:
```

Out[18]:
```
Embarked        1.000000
Sex             0.116569
Survived        0.106811
Fare            0.062142
Pclass          0.045702
Age            -0.009165
Family_Count   -0.080281
Name: Embarked, dtype: float64
```

In [19]:
```
#Top 3 attributes for training and testing are Sex, Survived, Fare for Embarked
```

Training and testing the model:

Splitting the data set into 75% training and 25% testing.

In [20]:
```
#Splilting the data into 75% for training,25% for testing the classifier
y=df['Embarked'].values
x=df.loc[:,['Sex','Survived','Fare']]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=0)
```

In [21]:
```
print(x_train.shape), print(y_train.shape)
print(x_test.shape), print(y_test.shape)
```

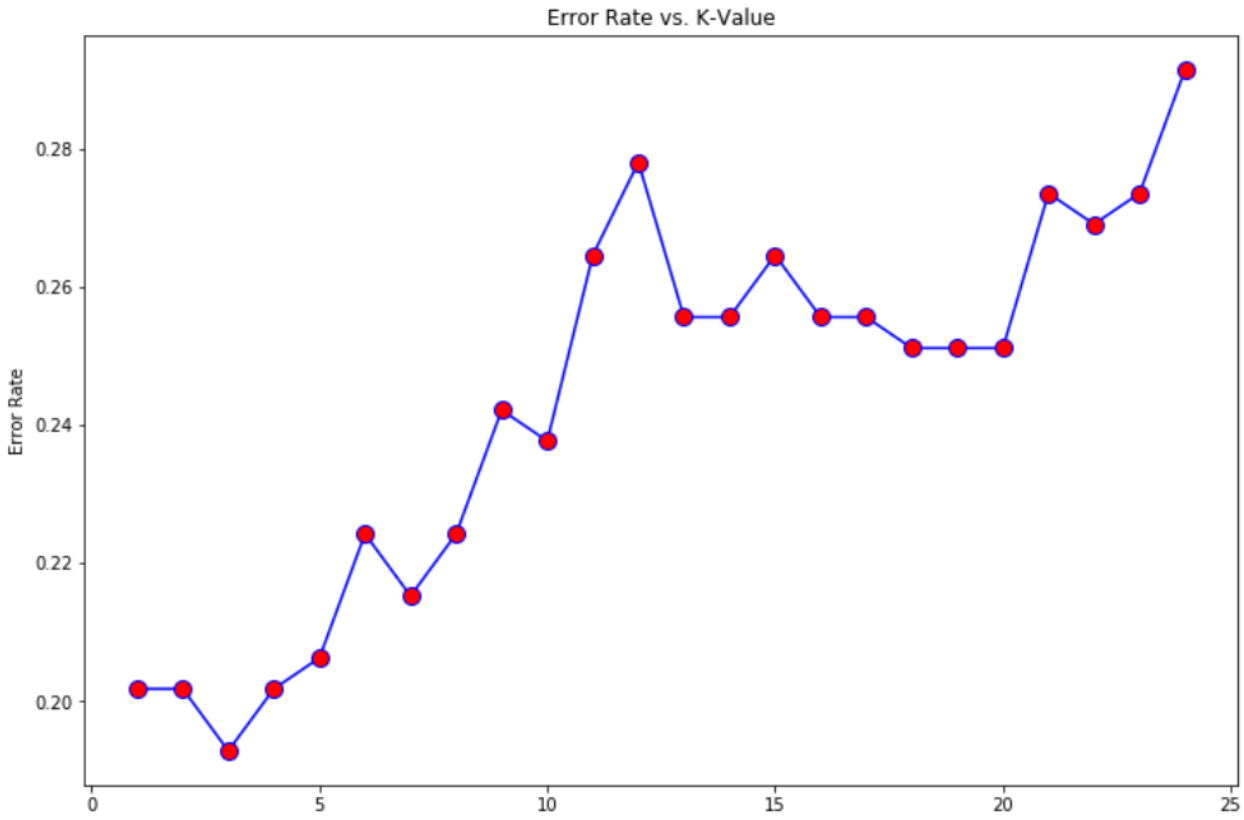
```
(668, 3)
(668,)
(223, 3)
(223,)
```

Out[21]:
```
(None, None)
```

Calculating K value with elbow method:

We select the value of K with the least error rate.

```
Min error -  0.19282511210762332 at K =  2
```



Applying KNN algorithm for the model:

In [26]:
```
#let k=3 with minkowski distance
knn = KNeighborsClassifier(n_neighbors = 3,metric = 'minkowski',p=2)
knn.fit(x_train,y_train)
```

Out[26]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

In [27]:
```
train_accuracy = knn.score(x_train, y_train)
print(train_accuracy)
```

```
0.8892215568862275
```

In [28]:
```
#let us get the predictions using the classifier we had fit above
y_pred = knn.predict(x_test)
confusion_matrix(y_test,y_pred)
```

Out[28]:
```
array([[146,   5,   3],
       [ 29,  21,   2],
       [  4,   0,  13]], dtype=int64)
```

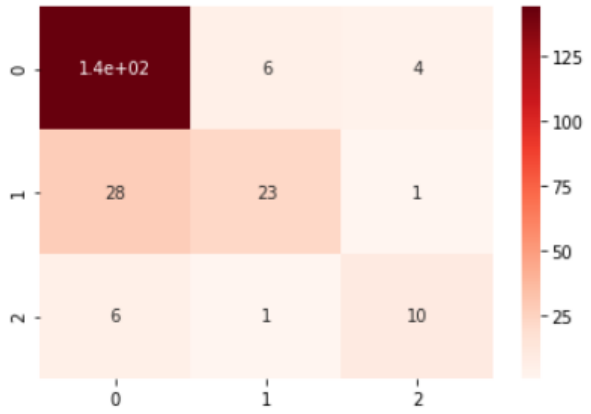
The below is the Confusion matrix, Classification report when **K = 5** for Euclidean distance.

```
accuracy_score :  0.7937219730941704
              precision    recall  f1-score   support

           0       0.81      0.94      0.87       154
           1       0.77      0.44      0.56        52
           2       0.67      0.59      0.62        17

    accuracy                           0.79       223
   macro avg       0.75      0.66      0.68       223
weighted avg       0.79      0.79      0.78       223
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x13662bd7f28>



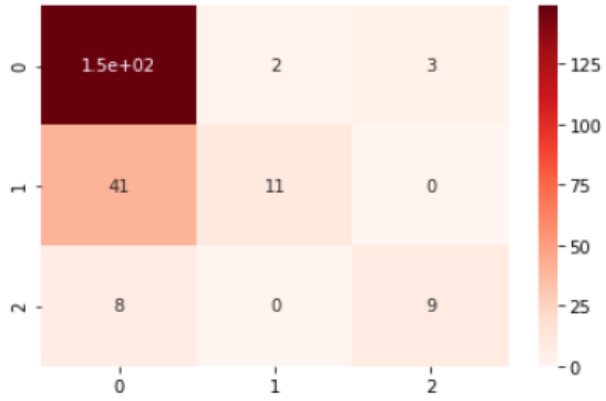
The below is the Confusion matrix, Classification report when **K = 9** for Euclidean distance.

```
accuracy_score :  0.757847533632287
              precision    recall  f1-score   support

           0       0.75      0.97      0.85       154
           1       0.85      0.21      0.34        52
           2       0.75      0.53      0.62        17

    accuracy                           0.76       223
   macro avg       0.78      0.57      0.60       223
weighted avg       0.77      0.76      0.71       223
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x13662c86240>



The below is the Confusion matrix, Classification report when **K = 2** for Euclidean distance.

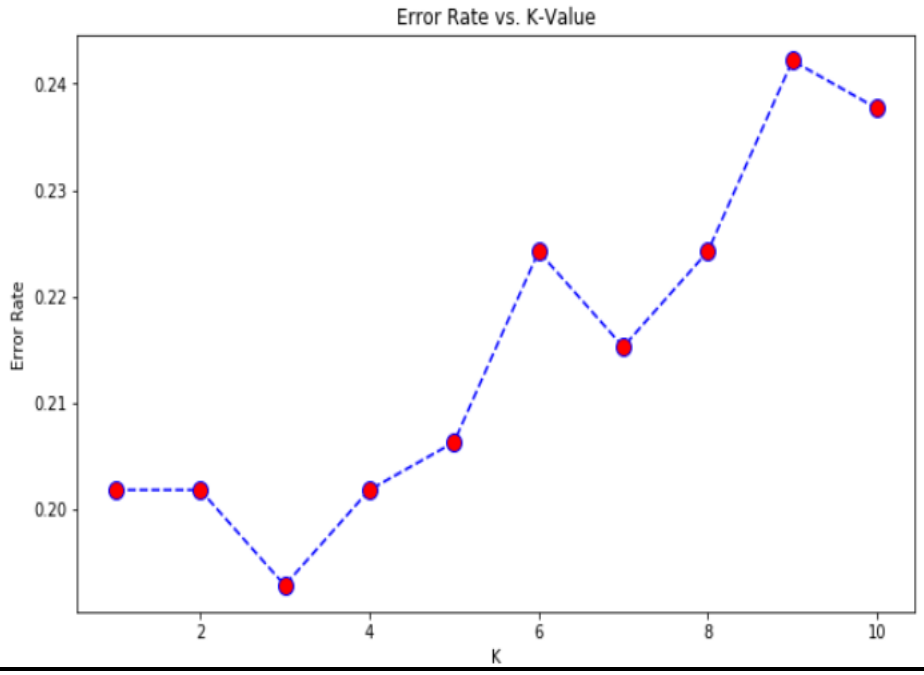
	precision	recall	f1-score	support
0	0.80	0.95	0.87	154
1	0.76	0.37	0.49	52
2	0.86	0.71	0.77	17
accuracy			0.80	223
macro avg	0.81	0.68	0.71	223
weighted avg	0.79	0.80	0.77	223

```
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x13662d1cdd8>
```

Error rate vs K values:

Below is the graph plotted for Error rate versus K values. Where we plotted with the plot() function.

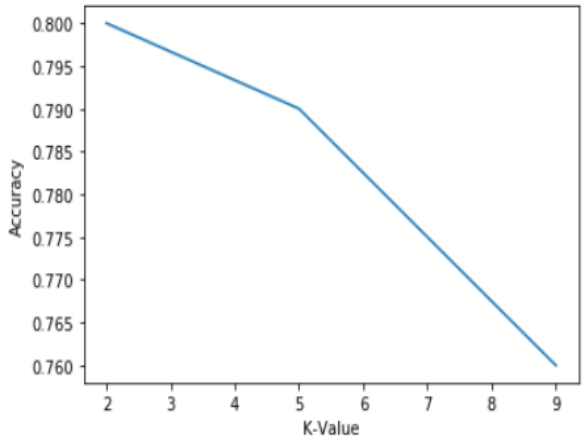
```
Out[35]: Text(0, 0.5, 'Error Rate')
```



Accuracy vs K values:

We plotted a graph which has accuracy on y axis and k values on x axis.

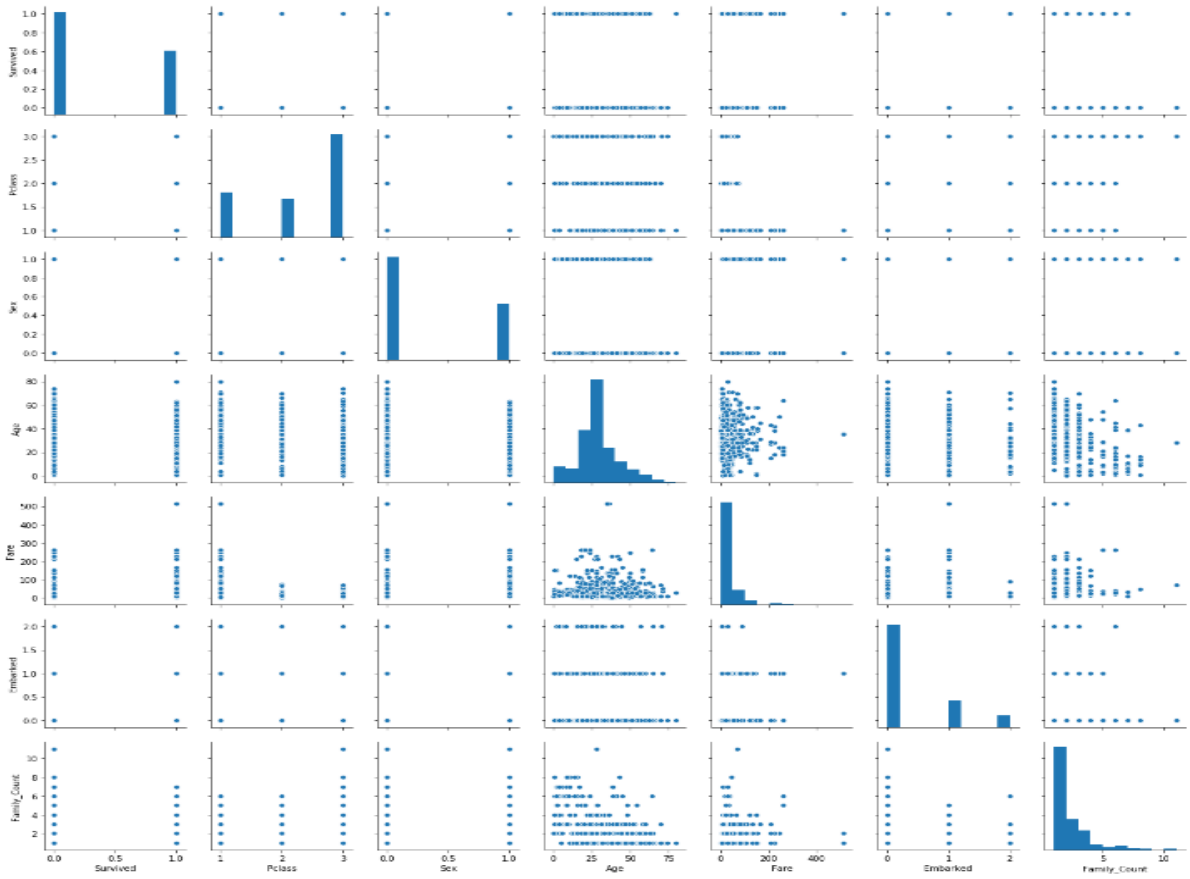
```
Out[36]: Text(0, 0.5, 'Accuracy')
```



Visualization:

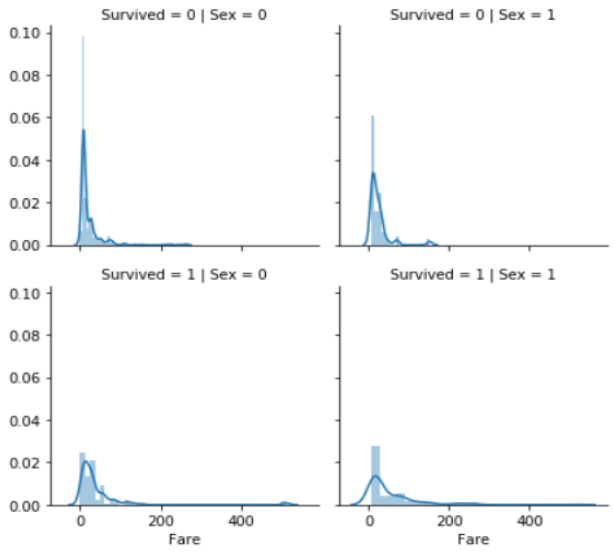
Using seaborn we plotted graphs using pairplot() function.
It gives pairwise relationships in the database.

```
Out[37]: <seaborn.axisgrid.PairGrid at 0x13662f7ffd0>
```



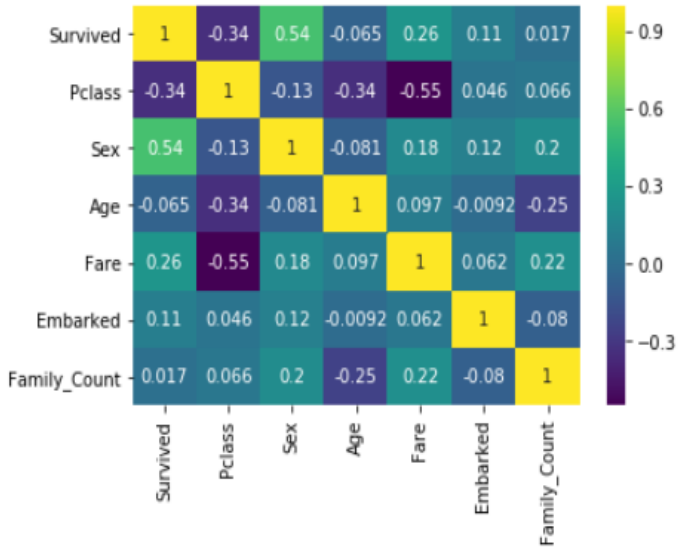
FacetGrid plot for the three top most found attributes.
It is a multi-plot grid for plotting conditional relationships:

```
Out[40]: <seaborn.axisgrid.FacetGrid at 0x1366638aef0>
```



Plotted Heatmap between all the attributes correlation values:

```
Out[39]: <matplotlib.axes._subplots.AxesSubplot at 0x136661ce438>
```



Contribution:

Sanjana Kondabathini	Worked on Classification – Decision Tree and Naive Bayes.
Meghana Kadali	Worked on Classification – Nearest Neighbors and report of knn.
Mucharla Rajashekar	Report on Classification – Decision Tree.

REFERENCES:

https://www.geeksforgeeks.org/k-nearest-neighbours/