DATA MINING ASSIGNMENT -2

KNN REPORT

1.Team Contribution:

Subhash, Varun Adit, and Supreetha communicated and contributed equally throughout the project.
Varun Adit gone through the preprocessing and cleansing of data and the techniques of visualization of data. He also contributed for better understanding of dataset.
Subhash was responsible for exploring python functions and pandas' libraries. He put forth her knowledge with us which helped us complete the python part of this assignment.

Supreetha has gone through site browsing which helped in completing the project and helped in accuracy and Report making.

2. Explaining the Nearest Neighbors method:

The k-nearest neighbors (KNN) algorithm is a data categorization technique for calculating the likelihood that a data point will join one group, or another based on the group to which the data points closest to it belong. A supervised machine learning approach known as the k-nearest neighbor algorithm is used to resolve classification and regression issues. But classification issues are where it's most frequently applied.

K-nearest neighbors' categorization is straightforward to comprehend and use. When the data points are well-defined or non-linear, it works well.

To decide the class of an unobserved observation, KNN essentially uses a voting method. This indicates that the class that receives the most votes will be the class for the relevant data point.

If K is equal to 1, we will only consider a data point's closest neighbor when determining its class. The ten closest neighbors will be used if K is equal to ten, and so on.

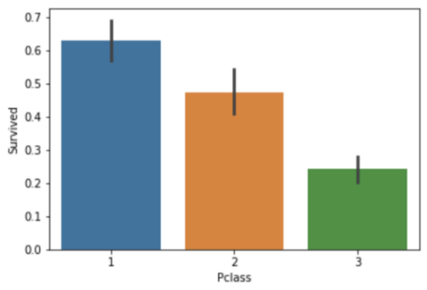
The Euclidean distance, Manhattan distance, hamming distance, and Minkowski distance are the four methods for calculating the distance between a data point and its closest neighbor. The Euclidean distance is the distance function or metric that is most frequently applied among the three.

```
knn = neighbors.KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print('Accuracy : %.3f' % (accuracy_score(y_test, y_pred)*100) +'%')
print(classification_report(y_test, y_pred))
```

3. Explain what your criteria was for selecting the three attributes

I have selected the 3 attributes Pclass, Sex, Age because:

1. Pclass

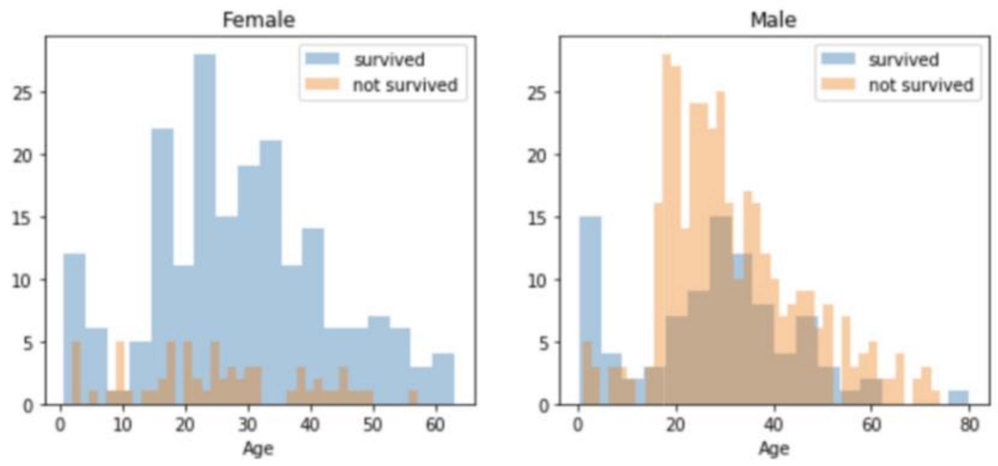


```
sns.barplot(x='Pclass', y='Survived', data=train_df)
```

```
Here, it is evident that Pclass affects a person's likelihood of
survival, especially if that individual is in class 1.
```

2. Age and Sex:

```
survived = 'survived'
not_survived = 'not survived'
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(10, 4))
women = train_df[train_df['Sex']=='female']
men = train_df[train_df['Sex']=='male']
ax = sns.distplot(women[women['Survived']==1].Age.dropna(), bins=18,
label = survived, ax = axes[0], kde =False)
ax = sns.distplot(women[women['Survived']==0].Age.dropna(), bins=40,
label = not_survived, ax = axes[0], kde =False)
ax.legend()
ax.set_title('Female')
ax = sns.distplot(men[men['Survived']==1].Age.dropna(), bins=18,
label = survived, ax = axes[1], kde = False)
ax = sns.distplot(men[men['Survived']==0].Age.dropna(), bins=40,
label = not_survived, ax = axes[1], kde = False)
ax.legend()
_ = ax.set_title('Male')
```



It is clear that males, and to a lesser extent women, have a higher chance of surviving when they are between the ages of 18 and 30. Between the ages of 14 and 40, women have a greater probability of surviving.

Between the ages of 5 and 18, men have an extremely low chance of surviving, while women do not. Furthermore, newborns have a slightly greater chance of surviving than adults.

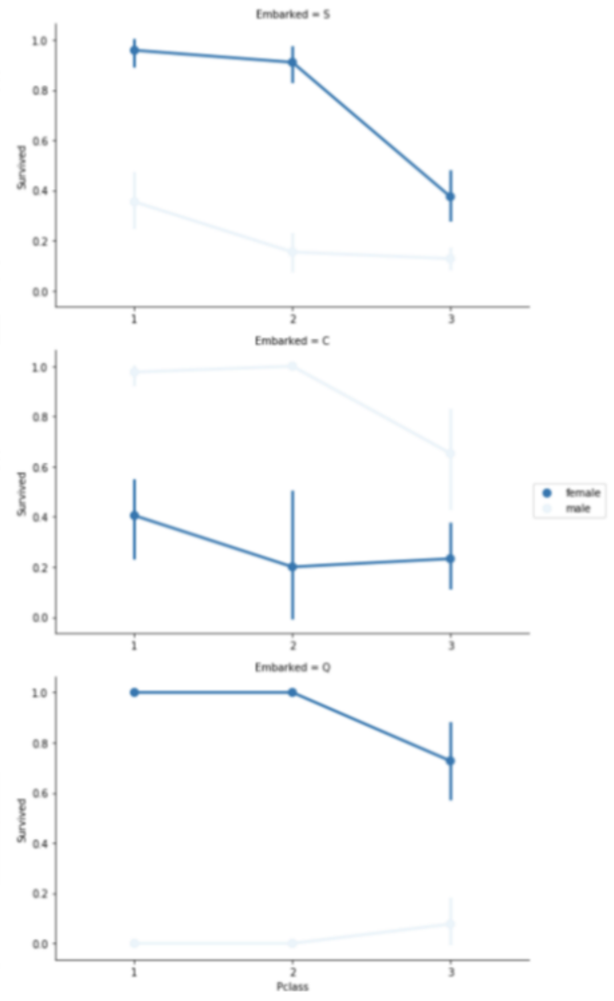
I'll develop age groups later because it seems that there are some ages that have higher survival rates and because I want every feature to be generally on the same scale.

What other 3 attribute can you choose?

I would have selected the 3 attributes Pclass, Sex, Embarked because:

```
FacetGrid = sns.FacetGrid(train_df, row='Embarked', size=4.5, aspect=1.6)
FacetGrid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette=None,
order=None, hue_order=None )
FacetGrid.add_legend()
```

```
<seaborn.axisgrid.FacetGrid at 0x10ba485c0>
```

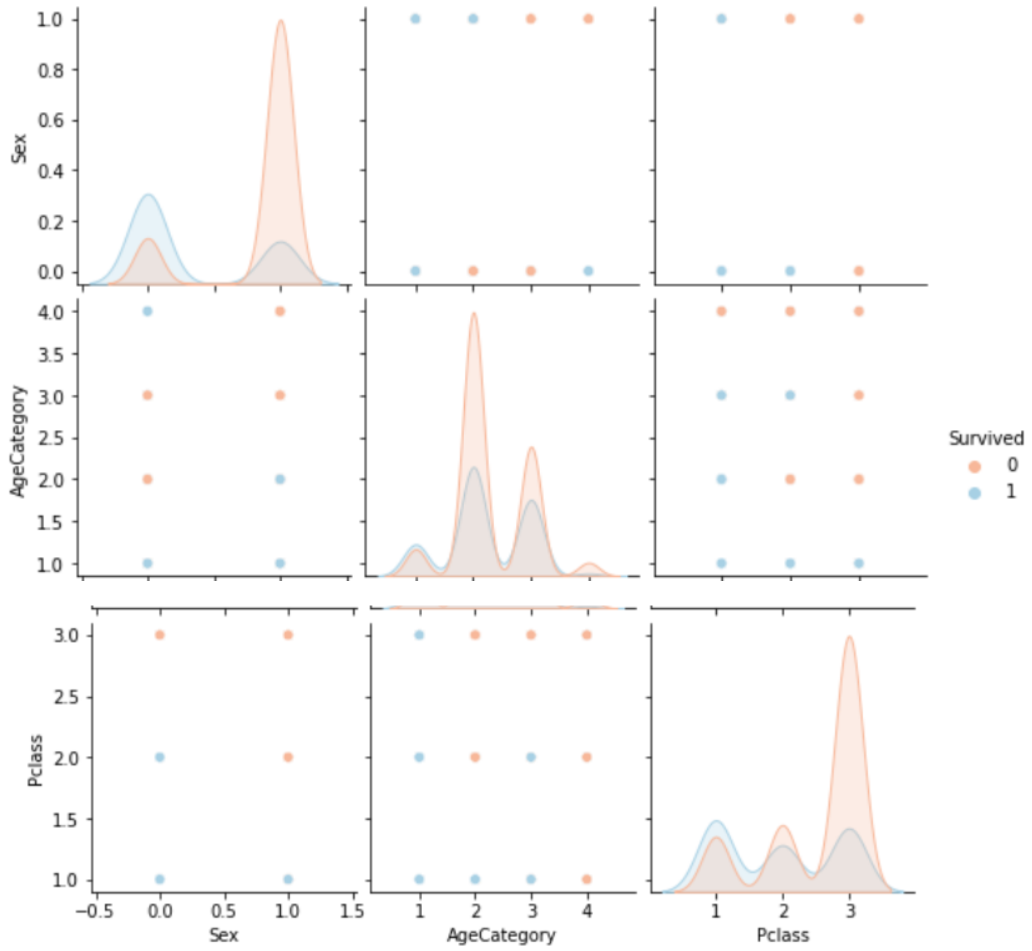


Depending on the gender, embarked seems to be connected with survival.

Women who are on ports Q and S are more likely to survive. If they are at port C, the opposite is true. If men are on port C, their chances of survival are good; nevertheless, if they are on port Q or S, their chances are poor.

4. Visualizations of the classifier in a 2D projection, and write your observations.

```
<seaborn.axisgrid.PairGrid at 0x7fd13ac4d550>
```



Here, it is evident that Pclass affects a person's likelihood of survival, especially if that individual is in class 1. It is clear that males, and to a lesser extent women, have a higher chance of surviving when they are between the ages of 18 and 30. Between the ages of 14 and 40, women have a greater probability of surviving. Between the ages of 5 and 18, men have an extremely low chance of surviving, while women do not. Furthermore, newborns have a slightly greater chance of surviving than adults. I'll develop age groups later because it seems that there are some ages that have higher survival rates and because I want every feature to be generally on the same scale.

5. Interpret and compare the results explain in detail

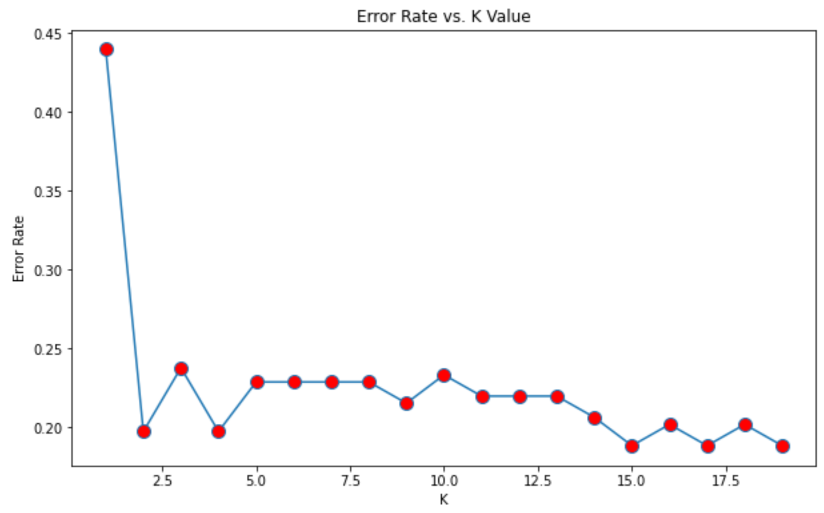
Here we can see that as the k value increases the error rate has decreased. The error rate was high when k value is in between 1 and 2. The error rate line is stable from 2 to 12.5 and the Accuracy is max at K= 15, 17,19.

Accuracy at K= 4 : 80.269

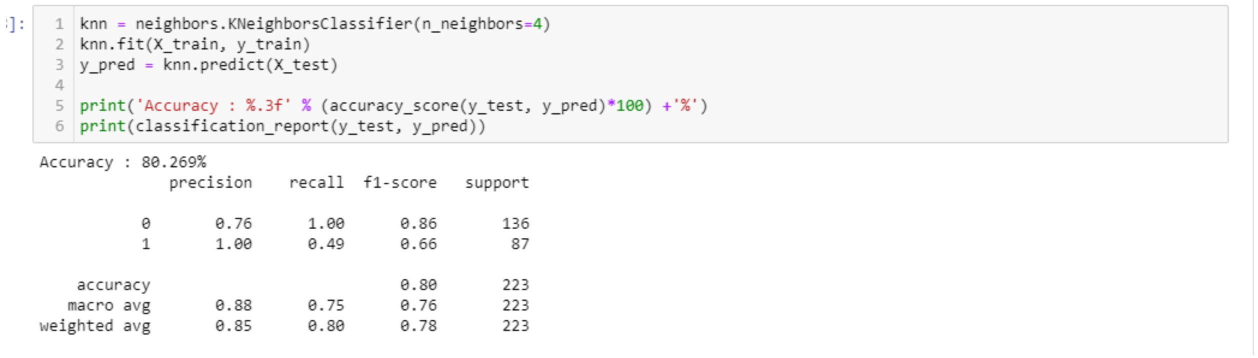
Accuracy at K= 15 : 81.166

Accuracy at K= 2 : 80.269

For all the values of K we are getting the accuracy more than 80 percent and for the k values at 15, 17, 19 getting the highest accuracy of 81.166



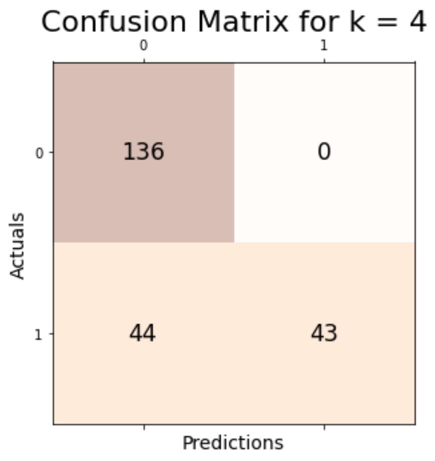
For k=4:



```
]:
knn = neighbors.KNeighborsClassifier(n_neighbors=4)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print('Accuracy : %.3f' % (accuracy_score(y_test, y_pred)*100) +'%')
print(classification_report(y_test, y_pred))
```

```
Accuracy : 80.269%
              precision    recall  f1-score   support

           0       0.76      1.00      0.86       136
           1       1.00      0.49      0.66        87

    accuracy                           0.80       223
   macro avg       0.88      0.75      0.76       223
weighted avg       0.85      0.80      0.78       223
```

```
plt.show()
```



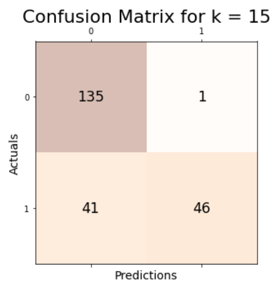
For k=15:

Predictions

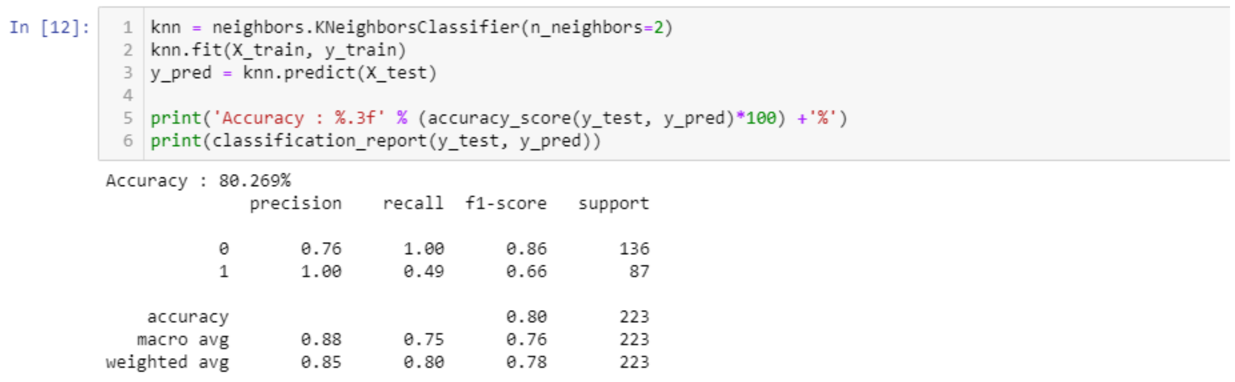
```
In [10]:
knn = neighbors.KNeighborsClassifier(n_neighbors=15)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print('Accuracy : %.3f' % (accuracy_score(y_test, y_pred)*100) +'%')
print(classification_report(y_test, y_pred))
```

```
Accuracy : 81.166%
              precision    recall  f1-score   support

           0       0.77      0.99      0.87       136
           1       0.98      0.53      0.69        87

    accuracy                           0.81       223
   macro avg       0.87      0.76      0.78       223
weighted avg       0.85      0.81      0.80       223
```

```
In [11]:
y_pred = knn.predict(X_test)

conf_matrix = confusion_matrix(y_true=y_test, y_pred=y_pred)
#
# Print the confusion matrix using Matplotlib
#
fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(conf_matrix, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=14)
plt.ylabel('Actuals', fontsize=14)
plt.title('Confusion Matrix for k = 15', fontsize=22)
plt.show()
```

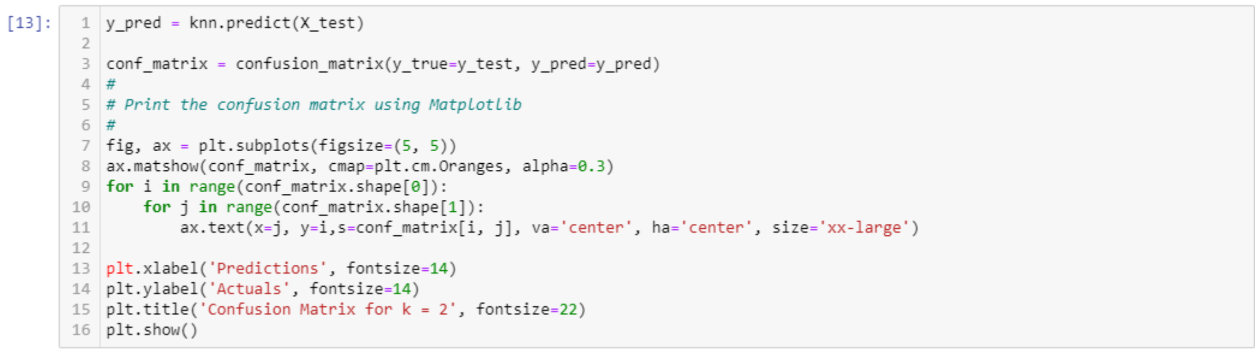


For k=2:

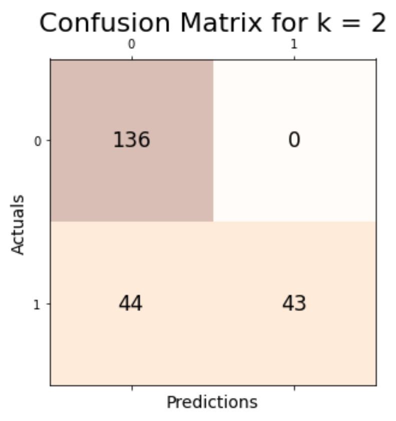


```
In [12]:
knn = neighbors.KNeighborsClassifier(n_neighbors=2)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

print('Accuracy : %.3f' % (accuracy_score(y_test, y_pred)*100) +'%')
print(classification_report(y_test, y_pred))
```

```
Accuracy : 80.269%
              precision    recall  f1-score   support

           0       0.76      1.00      0.86       136
           1       1.00      0.49      0.66        87

    accuracy                           0.80       223
   macro avg       0.88      0.75      0.76       223
weighted avg       0.85      0.80      0.78       223
```



```
[13]:
y_pred = knn.predict(X_test)

conf_matrix = confusion_matrix(y_true=y_test, y_pred=y_pred)
#
# Print the confusion matrix using Matplotlib
#
fig, ax = plt.subplots(figsize=(5, 5))
ax.matshow(conf_matrix, cmap=plt.cm.Oranges, alpha=0.3)
for i in range(conf_matrix.shape[0]):
    for j in range(conf_matrix.shape[1]):
        ax.text(x=j, y=i,s=conf_matrix[i, j], va='center', ha='center', size='xx-large')

plt.xlabel('Predictions', fontsize=14)
plt.ylabel('Actuals', fontsize=14)
plt.title('Confusion Matrix for k = 2', fontsize=22)
plt.show()
```



References:

https://learn.g2.com/k-nearest-neighbor

https://towardsdatascience.com/predicting-the-survival-of-titanic-passengers-30870ccc7e8

https://github.com/ElsitaK/Titanic_kNN

https://thedatamonk.com/kaggle-titanic-solution/

https://www.kaggle.com/competitions/titanic-dataset/data?select=titanic_train.csv