Name: Saurabh S. Ramteke
Roll No: 21-27-19
M.Tech: Data Science

Assignment No. 5

KNN Model - default

```
# KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier() # n = 5 default
# metric = 'minkowesi' - default

# Fitting of data
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

Metric

```
from sklearn.metrics import confusion_matrix, accuracy_score
CR = confusion_matrix(y_test, y_pred)
print(CR)
accuracy_score(y_test, y_pred)

# Diagonal are correct prediction of different classes
# elments to the side of 11 - sum of them is error
```

```
[[11  0  0]
 [ 0  8  5]
 [ 0  0  6]]

0.8333333333333334
```

Using Default values in sci-kit learn module for KNN, it uses ‘minoweski’ metric with 5 neihghbour we get fairly good accuracy of 83 %.

KNN - Neighbours n = 1

```
knn1 = KNeighborsClassifier(n_neighbors = 1) # n = 1
# metric = 'minkowesi' - default

# Fitting of data
knn1.fit(X_train, y_train)
y_pred = knn1.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score
CR = confusion_matrix(y_test, y_pred)
print(CR)
accuracy_score(y_test, y_pred)
```

```
[[11  0  0]
 [ 0  7  6]
 [ 0  0  6]]

0.8
```

For KNN model with neighbours n = 1, accuracy is reduced to 80% which is also not bad. Using default metric

Name: Saurabh S. Ramteke
Roll No: 21-27-19
M.Tech: Data Science

KNN - Neighbours n = 2

```
knn2 = KNeighborsClassifier(n_neighbors = 2)

# Fitting of data
knn2.fit(X_train, y_train)
y_pred = knn2.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score
CR = confusion_matrix(y_test, y_pred)
print(CR)
accuracy_score(y_test, y_pred)
```

```
[[11  0  0]
 [ 0 11  2]
 [ 0  0  6]]

0.9333333333333333
```

Now we take neighbour n = 2 our accuracy jumps up 93 % which is very good.

KNN - Neighbours n = 3

```
knn3 = KNeighborsClassifier(n_neighbors = 3)

# Fitting of data
knn3.fit(X_train, y_train)
y_pred = knn3.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score
CR = confusion_matrix(y_test, y_pred)
print(CR)
accuracy_score(y_test, y_pred)
```

```
[[11  0  0]
 [ 0  8  5]
 [ 0  0  6]]

0.8333333333333334
```

Taking n = 3 our accuracy again reaches to 83% which will be same until we take n =70-80, then the accuracy will reduce drastically.

Name: Saurabh S. Ramteke
Roll No: 21-27-19
M.Tech: Data Science

KNN Model - eucliedean metric

```
euc = KNeighborsClassifier(n_neighbors = 2, metric = 'euclidean')
euc.fit(X_train, y_train)
pred_euc = euc.predict(X_test)
pred_euc
```

```
array([0, 1, 1, 0, 2, 2, 2, 0, 0, 2, 1, 0, 2, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       2, 0, 2, 1, 0, 0, 1, 2])
```

Metric

```
CR_euc = confusion_matrix(y_test, pred_euc)
print(CR_euc)
accuracy_score(y_test, pred_euc)

# Diagonal are correct prediction of different classes
# elments to the side of 11 - sum of them is error
```

```
[[11  0  0]
 [ 0 11  2]
 [ 0  0  6]]

0.9333333333333333
```

Same model accuracy for Euclidena metric and minkoweski metric.