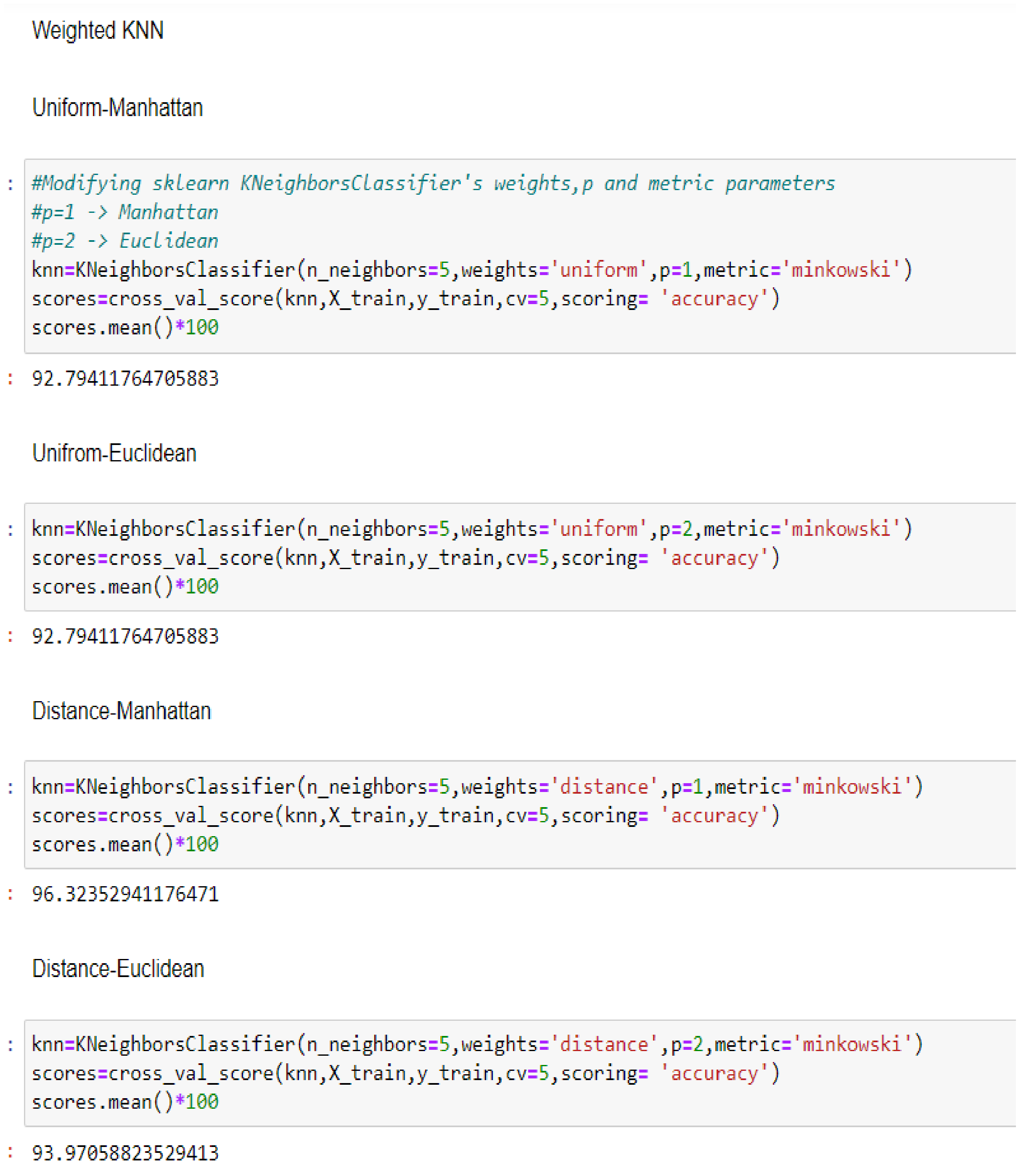
Improved Model

5) Weighted KNN – The KNeighborsClassifier’s weighting techniques (distance, uniform) and distance metrics (p=1->Manhattan, p=2->Euclidean) are utilized here and used on the validation test.

a) For iris, with the chosen **k= 5**



Weighted KNN

Uniform-Manhattan

```
#Modifying sklearn KNeighborsClassifier's weights,p and metric parameters
#p=1 -> Manhattan
#p=2 -> Euclidean
knn=KNeighborsClassifier(n_neighbors=5,weights='uniform',p=1,metric='minkowski')
scores=cross_val_score(knn,X_train,y_train,cv=5,scoring= 'accuracy')
scores.mean()*100
```

```
92.79411764705883
```

Unifrom-Euclidean

```
knn=KNeighborsClassifier(n_neighbors=5,weights='uniform',p=2,metric='minkowski')
scores=cross_val_score(knn,X_train,y_train,cv=5,scoring= 'accuracy')
scores.mean()*100
```

```
92.79411764705883
```

Distance-Manhattan

```
knn=KNeighborsClassifier(n_neighbors=5,weights='distance',p=1,metric='minkowski')
scores=cross_val_score(knn,X_train,y_train,cv=5,scoring= 'accuracy')
scores.mean()*100
```

```
96.32352941176471
```

Distance-Euclidean

```
knn=KNeighborsClassifier(n_neighbors=5,weights='distance',p=2,metric='minkowski')
scores=cross_val_score(knn,X_train,y_train,cv=5,scoring= 'accuracy')
scores.mean()*100
```

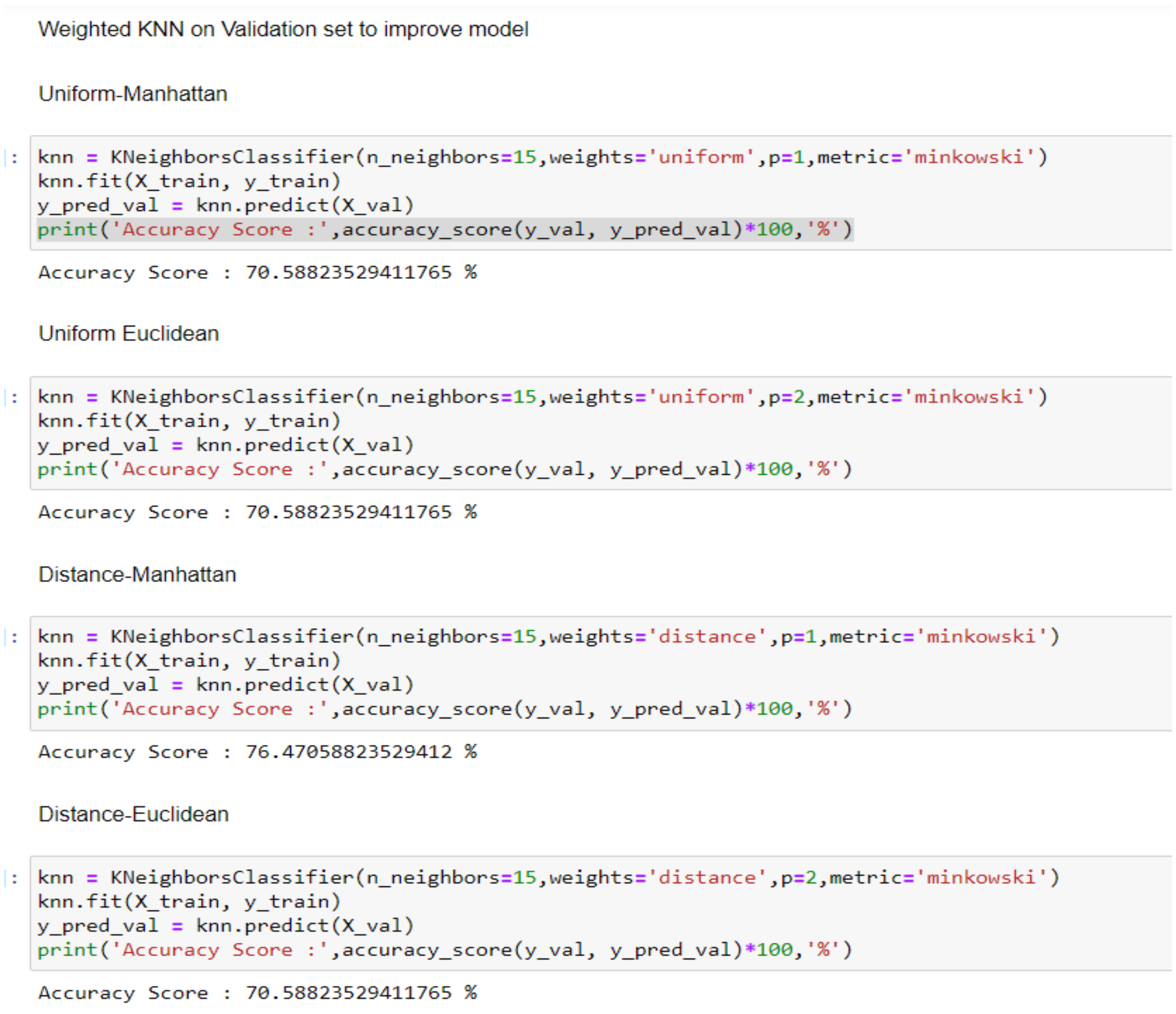
```
93.97058823529413
```

Weight	Distance	Accuracy
Uniform	Manhattan	92.79 %
Uniform	Euclidean	92.79 %
Distance	Manhattan	96.32 %
Distance	Euclidean	93.97 %

From the above results, we can conclude that among all the weights and distance metrics, **distance** weighted technique and **Manhattan** distance metric exhibit the highest accuracy.

Distance weight ensures that neighbours that are at the closest distance have more weight when classifying data points.

b) For heart disease , with the chosen **k= 15**



Weighted KNN on Validation set to improve model

Uniform-Manhattan

```
knn = KNeighborsClassifier(n_neighbors=15,weights='uniform',p=1,metric='minkowski')
knn.fit(X_train, y_train)
y_pred_val = knn.predict(X_val)
print('Accuracy Score :',accuracy_score(y_val, y_pred_val)*100,'%')
```

```
Accuracy Score : 70.58823529411765 %
```

Uniform Euclidean

```
knn = KNeighborsClassifier(n_neighbors=15,weights='uniform',p=2,metric='minkowski')
knn.fit(X_train, y_train)
y_pred_val = knn.predict(X_val)
print('Accuracy Score :',accuracy_score(y_val, y_pred_val)*100,'%')
```

```
Accuracy Score : 70.58823529411765 %
```

Distance-Manhattan

```
knn = KNeighborsClassifier(n_neighbors=15,weights='distance',p=1,metric='minkowski')
knn.fit(X_train, y_train)
y_pred_val = knn.predict(X_val)
print('Accuracy Score :',accuracy_score(y_val, y_pred_val)*100,'%')
```

```
Accuracy Score : 76.47058823529412 %
```

Distance-Euclidean

```
knn = KNeighborsClassifier(n_neighbors=15,weights='distance',p=2,metric='minkowski')
knn.fit(X_train, y_train)
y_pred_val = knn.predict(X_val)
print('Accuracy Score :',accuracy_score(y_val, y_pred_val)*100,'%')
```

```
Accuracy Score : 70.58823529411765 %
```

Weight	**Distance**	**Accuracy**
Uniform	Manhattan	70.58 %
Uniform	Euclidean	70.58 %
Distance	Manhattan	76.47 %
Distance	Euclidean	70.58 %

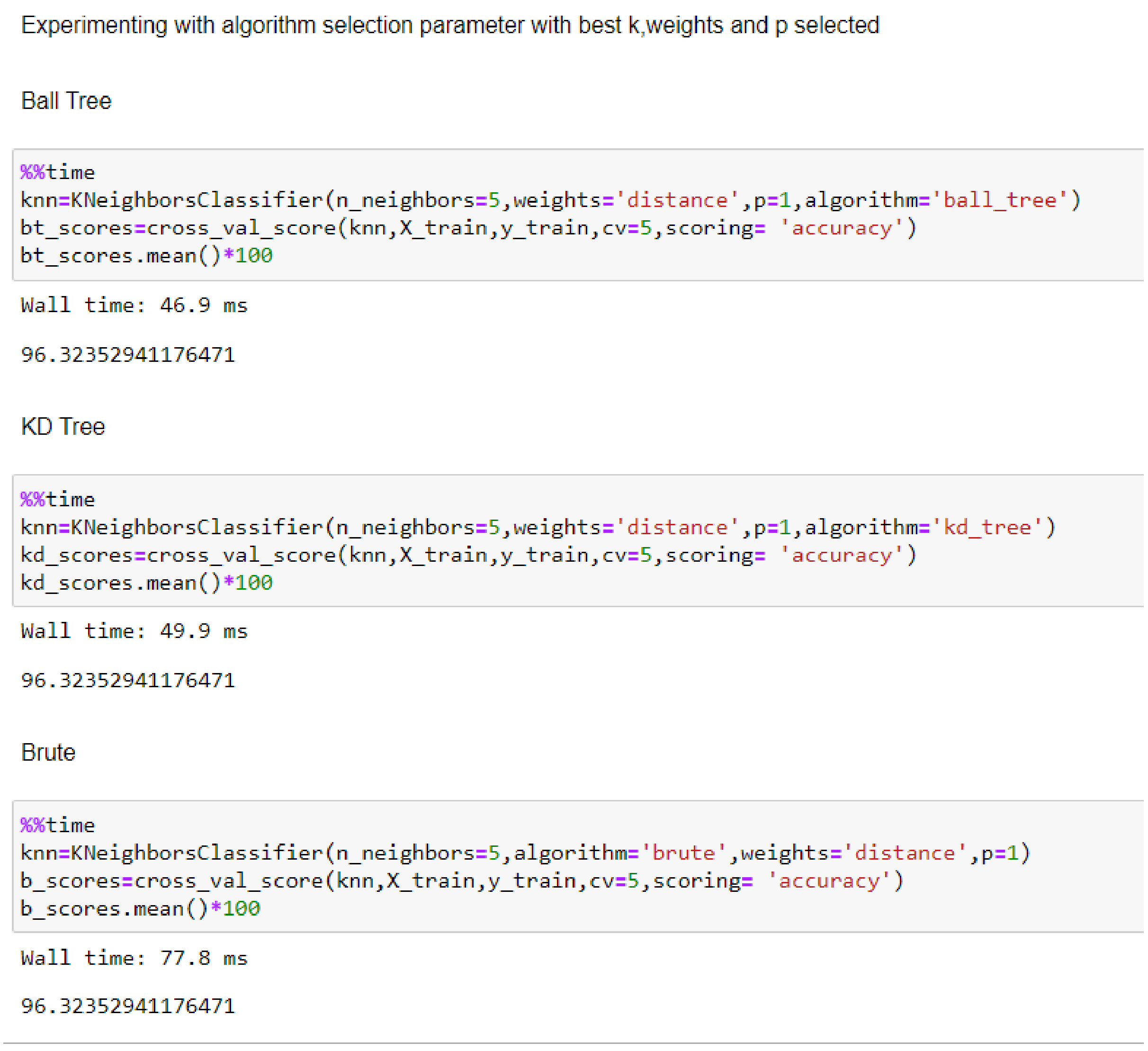
From the above results, we can conclude that using a **distance** weighted technique and **Manhattan** distance metric, a higher accuracy of 76.47 % is obtained.

Distance weight ensures that neighbours at a closer distance have more weight when classifying data points.

Manhattan distance is a better metric in higher dimensions than Euclidean distance. Since heart disease data set has many categorical features, one hot encoding these features increases the number of columns in the dataset and thereby, increases the dimensionality. Because of this, Manhattan distance might be a better measure here.

6) Different NN Algorithms – The KNeighborsClassifier's algorithm selection parameter (kd tree, ball tree and brute) are utilized here and used on the validation test.

a) For iris , with the chosen **k= 5 ,weight=distance, distance=Manhattan**



Experimenting with algorithm selection parameter with best k,weights and p selected

Ball Tree

```
%%time
knn=KNeighborsClassifier(n_neighbors=5,weights='distance',p=1,algorithm='ball_tree')
bt_scores=cross_val_score(knn,X_train,y_train,cv=5,scoring= 'accuracy')
bt_scores.mean()*100
```

```
Wall time: 46.9 ms

96.32352941176471
```

KD Tree

```
%%time
knn=KNeighborsClassifier(n_neighbors=5,weights='distance',p=1,algorithm='kd_tree')
kd_scores=cross_val_score(knn,X_train,y_train,cv=5,scoring= 'accuracy')
kd_scores.mean()*100
```

```
Wall time: 49.9 ms

96.32352941176471
```

Brute

```
%%time
knn=KNeighborsClassifier(n_neighbors=5,algorithm='brute',weights='distance',p=1)
b_scores=cross_val_score(knn,X_train,y_train,cv=5,scoring= 'accuracy')
b_scores.mean()*100
```

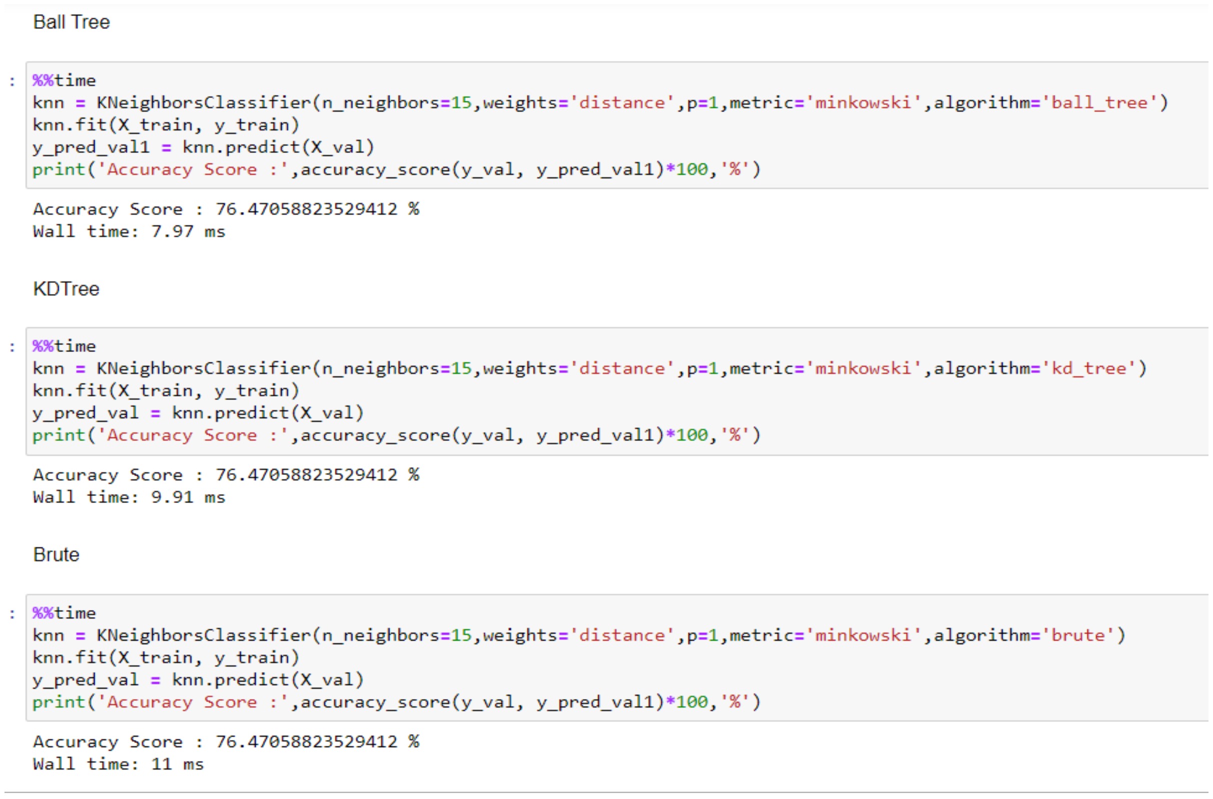
```
Wall time: 77.8 ms

96.32352941176471
```

The accuracy results are as below:

Algorithm	**Accuracy**
Ball Tree	96.32 %
KD Tree	96.32 %
Brute	96.32 %

b) For heart disease dataset , with the chosen **k= 15, weight=distance, distance=Manhattan**



Ball Tree

```
%%time
knn = KNeighborsClassifier(n_neighbors=15,weights='distance',p=1,metric='minkowski',algorithm='ball_tree')
knn.fit(X_train, y_train)
y_pred_val1 = knn.predict(X_val)
print('Accuracy Score :',accuracy_score(y_val, y_pred_val1)*100,'%')
```

```
Accuracy Score : 76.47058823529412 %
Wall time: 7.97 ms
```

KDTree

```
%%time
knn = KNeighborsClassifier(n_neighbors=15,weights='distance',p=1,metric='minkowski',algorithm='kd_tree')
knn.fit(X_train, y_train)
y_pred_val = knn.predict(X_val)
print('Accuracy Score :',accuracy_score(y_val, y_pred_val1)*100,'%')
```

```
Accuracy Score : 76.47058823529412 %
Wall time: 9.91 ms
```

Brute

```
%%time
knn = KNeighborsClassifier(n_neighbors=15,weights='distance',p=1,metric='minkowski',algorithm='brute')
knn.fit(X_train, y_train)
y_pred_val = knn.predict(X_val)
print('Accuracy Score :',accuracy_score(y_val, y_pred_val1)*100,'%')
```

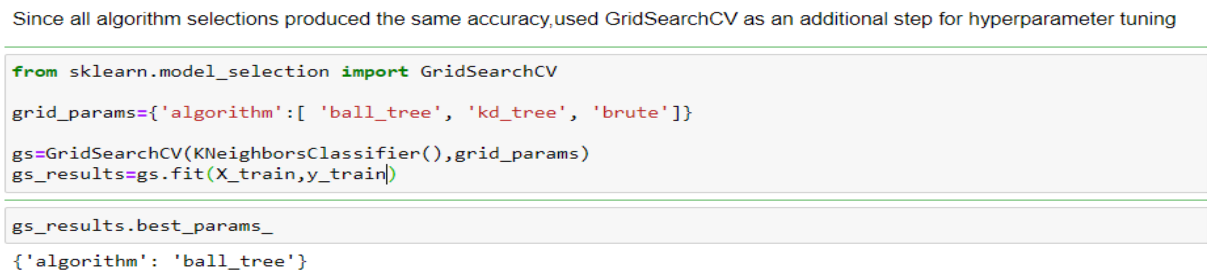
```
Accuracy Score : 76.47058823529412 %
Wall time: 11 ms
```

The accuracy results are tabulated as below:

Algorithm	**Accuracy**
Ball Tree	76.47 %
KD Tree	76.47 %
Brute	76.47 %

From the above results, **for both Iris and heart disease dataset**, we can observe that all the algorithms produce the same accuracy results. The computation times (calculated using %% time) vary in every successive code run.

Hence as an additional step, **sci-kit learn's GridSearchCV** was utilized on the training set for hyperparameter tuning of KNeighborsClassifier's algorithm selection parameter and for both datasets, **Ball tree** was observed as the best parameter for algorithm selection.



Since all algorithm selections produced the same accuracy,used GridSearchCV as an additional step for hyperparameter tuning

```
from sklearn.model_selection import GridSearchCV

grid_params={'algorithm':[ 'ball_tree', 'kd_tree', 'brute']}

gs=GridSearchCV(KNeighborsClassifier(),grid_params)
gs_results=gs.fit(X_train,y_train)
```

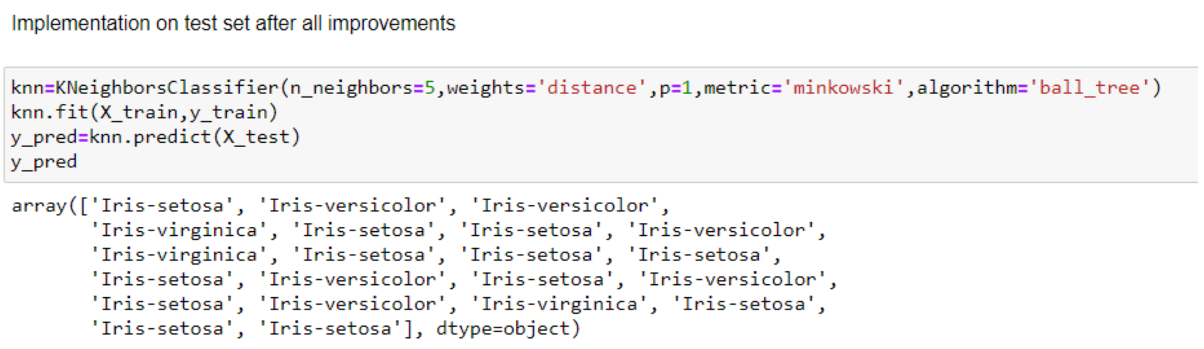
```
gs_results.best_params_
```

```
{'algorithm': 'ball_tree'}
```

7) [CM7] Evaluating improved model on the test set

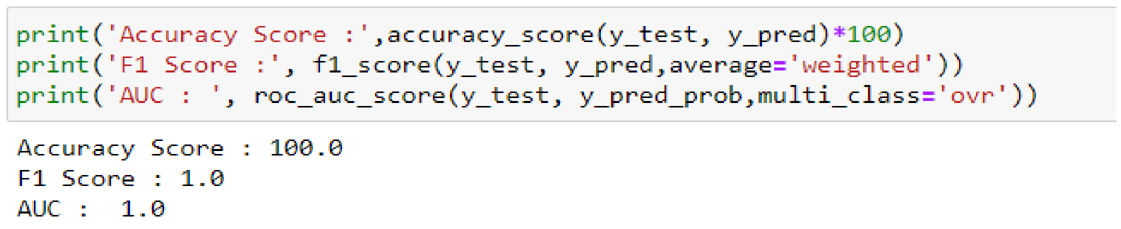
a) For iris, on test set,

k= 5, weight=distance, distance=Manhattan, Algorithm=Ball Tree


Implementation on test set after all improvements

```
knn=KNeighborsClassifier(n_neighbors=5,weights='distance',p=1,metric='minkowski',algorithm='ball_tree')
knn.fit(X_train,y_train)
y_pred=knn.predict(X_test)
y_pred
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-versicolor',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-versicolor',
       'Iris-setosa', 'Iris-versicolor', 'Iris-virginica', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa'], dtype=object)
```



```
print('Accuracy Score :',accuracy_score(y_test, y_pred)*100)
print('F1 Score :', f1_score(y_test, y_pred,average='weighted'))
print('AUC : ', roc_auc_score(y_test, y_pred_prob,multi_class='ovr'))
```

```
Accuracy Score : 100.0
F1 Score : 1.0
AUC :  1.0
```

The basic model and Improved Model results can be compared as below:

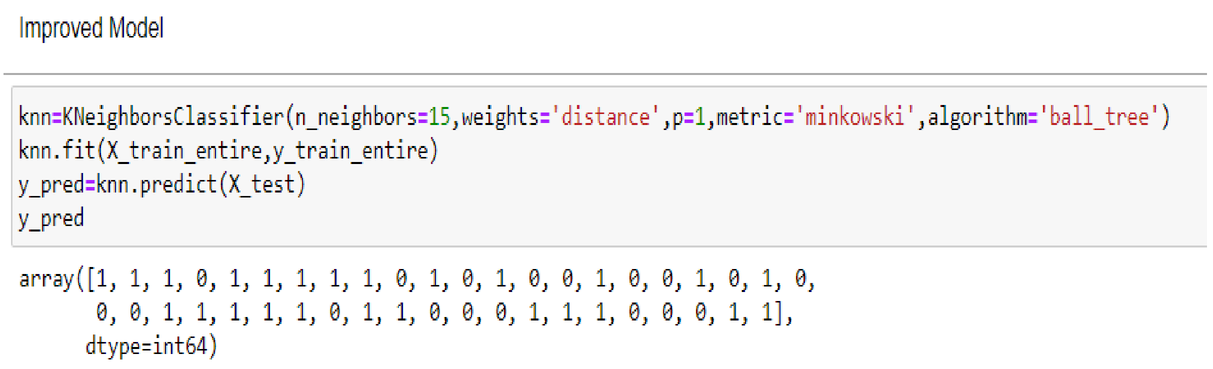
Model	k	Accuracy	f-score	AUC
Basic Model	5	95.23 %	0.94	1.0
Improved Model	5	100%	1.0	1.0

It can be observed from the above results that, with improvements in the model,

- Accuracy has increased from 95.23% to 100%
- F1-score is improved from 0.94 to 1.0
- Since, Area under the Curve (AUC) was already at maximum of 1.0, the same is retained in the improved model.

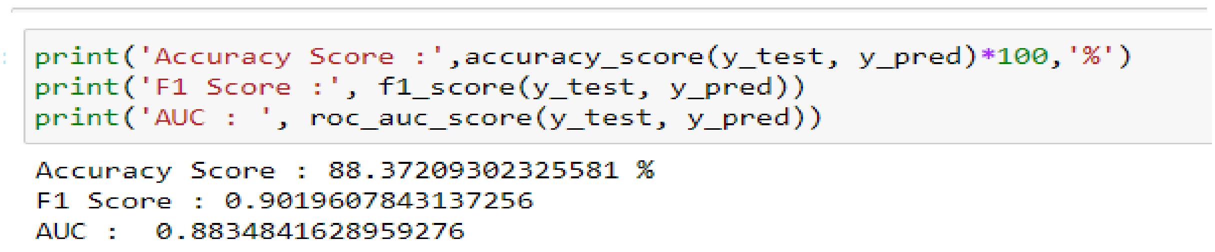
b) For Heart Disease, on the test set,

k= 15, weight=distance, distance=Manhattan, Algorithm=Ball Tree


Improved Model

```
knn=KNeighborsClassifier(n_neighbors=15,weights='distance',p=1,metric='minkowski',algorithm='ball_tree')
knn.fit(X_train_entire,y_train_entire)
y_pred=knn.predict(X_test)
y_pred
```

```
array([1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1],
      dtype=int64)
```



```
print('Accuracy Score :',accuracy_score(y_test, y_pred)*100,'%')
print('F1 Score :', f1_score(y_test, y_pred))
print('AUC : ', roc_auc_score(y_test, y_pred))
```

```
Accuracy Score : 88.37209302325581 %
F1 Score : 0.9019607843137256
AUC :  0.8834841628959276
```

The basic model and Improved Model results can be compared as below:

Model	k	Accuracy	f-score	AUC
Basic Model	15	83.72%	0.86	0.85
Improved Model	15	88.37 %	0.90	0.88

It can be observed from the above results that, with improvements in the model,

- Accuracy has increased from 83.72% to 88.37%.
- F1-score is improved from 0.86 to 0.90
- Area Under the curve has increased from 0.85 to 0.88