Lab 4 Report

Vehicle fleet management

Task 1 "A": Implement k-NN classifier:

It is also called lazy algorithm and nonparametric algorithm used for both classification and regression algorithms. KNN classification is used to classify the data into different categories like(true or false,1 or 0,yes or no).KNN -regression, the output is numerical value.

The machine learning algorithm is divided into two categories into different categories:

1)Fit(Training the data to the algorithm)

2)Predict(testing the algorithm based on unseen data or unused data)

In KNN-algorithm there is nothing called training .It just save the training data in the memory during training. During the testing state the KNN algorithm will find it's nearest neighbours and gives the result based on it's nearest neighbours.

In KNN-classification algorithm the result of the test data or sample will be classified based on k training sample nearest to the test sample. The commonly used distance metrics is Euclidean distance which is used to find the nearest neighbours.

In evaluation result I used confusion matrix it gives num[[Tp,Tn][Fp,Fn]]

Tp,Tn: True positive , True negative

Fp,Fn: False positive , False negative

B)The evaluation result of KNN algorithm is:

The number of correct prediction is 325 out of 400

Accuracy percentage is 81.25%

Confusion matrix of KNN-classification is:

```
In [19]:
1 print(cm)
2 print("****************************************")
3 print("accuracy of knn-classification is",accuracy_score(output,Y_test)*100)

[[104  27   1]
 [ 27  96  12]
 [  0   8 125]]
****************************************
accuracy of knn-classification is 81.25
```

```
In [11]:
1 print(accuracy_score(Y_test,output))
```

Task 1 “B”: Investigate different classification algorithms

1)Logistics Regression:

It takes the line equation y=c+m1x+m2x+m3x…….

Result:

The number of correct prediction is 325 out of 400

Accuracy of logistics regression is 93.5%

[11]:
```
#making confusion matrix
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
print(cm)
print("*************************")
print("accuracy of logistic regression is:",accuracy_score(y_test,y_pred))
```

```
[[137   3   0]
 [ 10  97   8]
 [  0   5 140]]
*************************
accuracy of logistic regression is: 0.935
```

2)Support vector machines(SVM) classifier:

The number of correct prediction is 293 out of 400

Accuracy of logistics regression is 73.25%

:
```
cm_svc=confusion_matrix(y_test,y_pred_svc)
print(cm_svc)
print("***************************************")
print("accuracy score og svm :",accuracy_score(y_pred_svc,y_test)*100)
```

```
[[120  20   0]
 [ 40  66   9]
 [ 28  10 107]]
***************************************
accuracy score og svm : 73.25
```

3) Gaussian naive bays:

The number of correct prediction is 374 out of 400

Accuracy of logistics regression is 95.25%

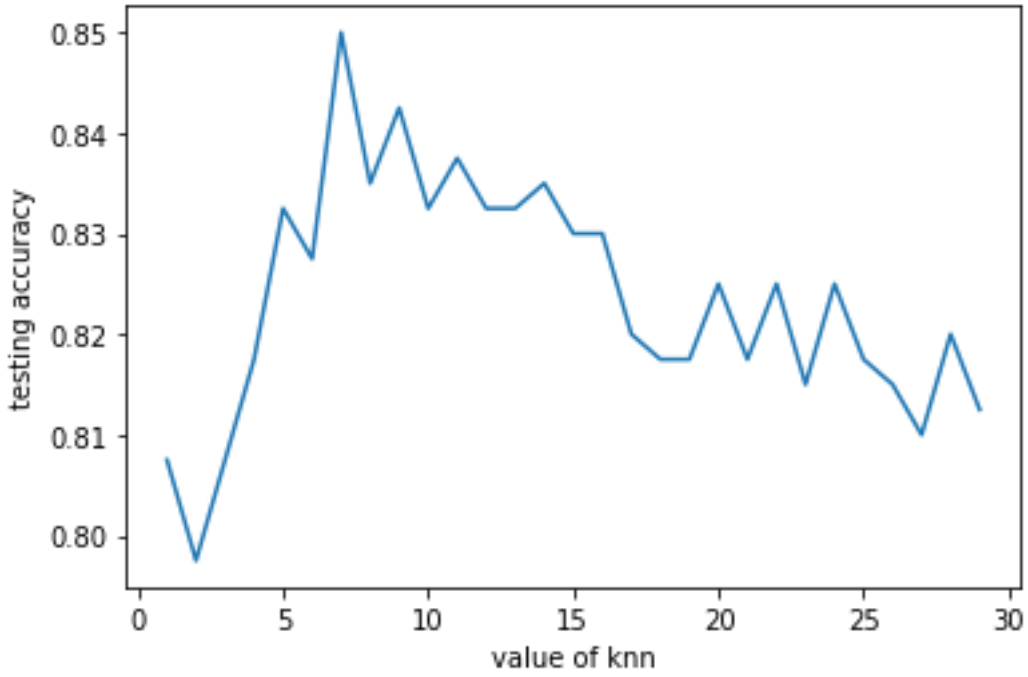
n [29]:
```
cm_gn=confusion_matrix(y_test,y_pred_gn)
print(cm)
print("*****************************")
print("accuracy score of Gaussian naive_bayes :",accuracy_score(y_pred_gn,y_test)*100)
```

```
[[137   3   0]
 [ 10  97   8]
 [  0   5 140]]
*****************************
accuracy score of Gaussian naive_bayes : 95.25
```

Task 1 “C”: Parameter tuning:

Euclidean distance:

In parameter tuning the x-axis is k-neighbours(different values of k) an Y-axis testing accuracy from t he below graph we can say that at k=7 we have the highest accuracy.
The distance measurement method is Euclidean distance.



The result at k=7 when the distance measure is Euclidean distance is:
The total number of correct predictions is 346 out of 400
The accuracy is 86.5%

In [62]:

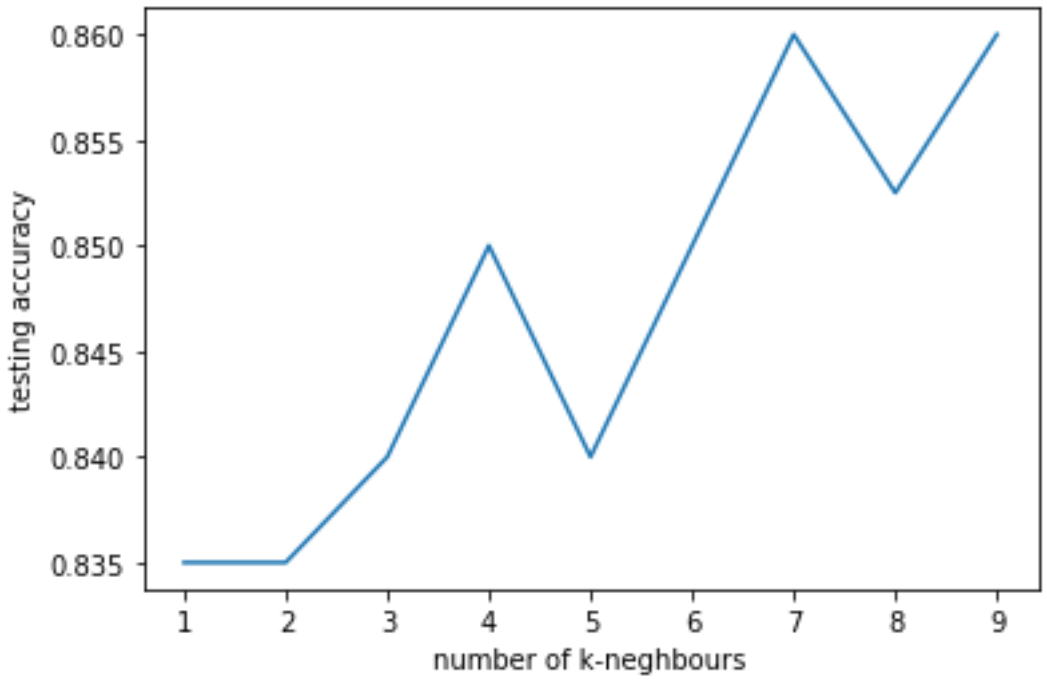
```
print(cm1)
print("****************************************")
print("accuracy of knn-classification is",accuracy_score(output,Y_test)*100)
```

```
[[110  10   1]
 [ 16 105  11]
 [  0  16 131]]
****************************************
accuracy of knn-classification is 86.5
```

In parameter tuning the x-axis is k-neighbours(different values of k) an Y-axis testing accuracy from t he below graph we can say that at k=4,7 we have the highest accuracy.

Manhattan distance:

The distance measurement method is Manhattan distance



The result at k=7 when the distance measure is Manhattan distance is:
The total number of correct predictions is 344 out of 400
The accuracy is 86%

```
In [54]:
1 print(cm)
2 print("****************************************")
3 print("accuracy of knn-classification is",accuracy_score(y_pred_knn1,Y_test)*100)
```

```
[[107  13   1]
 [ 15 106  11]
 [  0  16 131]]
****************************************
accuracy of knn-classification is 86.0
```

Task 1 “D”:

The test case accuracy of
knn for n-neighbours=7 is 83.5%
Logistic Regression accuracy: 92.25%

Task 1 “E”: Implement k-NN regression:

The KNN regression algorithm is same as KNN-classification algorithm .In knn-classificatio n algorithm the output belongs to some class but in KNN-regression algorithm the output is v alue. The value assigned to the test sample is calculated is by taking the average of the K-near est neighbours.
The accuracy is calculated by finding the error(it is calculated by subtracting the original valu e to the predicted value at that test sample)
If the error should be low the algorithm is best algorithm.
The mean squared error we get when we use number of neighbours is 3.The error we got is 0.1402

```
In [12]:
1 output1=predictOutputNumeric(X_train,Y_train,X_test,3)

In [13]:
1 from sklearn.metrics import mean_squared_error

In [14]:
1 error=mean_squared_error(Y_test,output1)

In [15]:
1 error

Out[15]: 0.14027777777777778
```

Task 1 “F”: Investigate different regression algorithms :

1)Linear Regression:

It takes the line equation y=c+m1x+m2x+m3x…….

Error:

Root mean square error 0.45417471249350055

```
In [18]:
1 linear_pred=linear1.predict(x_test)

In [19]:
1 print("mean_absolute_error",mean_absolute_error(y_test,linear_pred))
2 print("Root mean square error",sqrt(mean_absolute_error(y_test,linear_pred)))
3 print("r2_score",r2_score(y_test,linear_pred))

mean_absolute_error 0.20627466946855388
Root mean square error 0.45417471249350055
r2_score 0.9004404043844749
```

2)SVR:

Error:

Root mean square error 0.7526308528123123

```
In [23]:
1 svr_pred=svr.predict(x_test)

In [24]:
1 print("mean_absolute_error",mean_absolute_error(y_test,svr_pred))
2 print("Root mean square error",sqrt(mean_absolute_error(y_test,svr_pred)))
3 print("r2_score",r2_score(y_test,svr_pred))

mean_absolute_error 0.5664532006049885
Root mean square error 0.7526308528123123
r2_score 0.31626426176990863
```

Task 1 “G”: Parameter tuning :

When n-neighbours is 1 we getting low root mean square value.

Root mean square error 1 : 0.4472135954999579

```
Root mean square error 1 : 0.4472135954999579
Root mean square error 2 : 0.4821825380496478
Root mean square error 3 : 0.49328828623162474
Root mean square error 4 : 0.4962358310319802
Root mean square error 5 : 0.5049752469181039
Root mean square error 6 : 0.5070338581725419
Root mean square error 7 : 0.5102520385624567
Root mean square error 8 : 0.5111262075065218
Root mean square error 9 : 0.51234753829798
Root mean square error 10 : 0.5200961449578337
Root mean square error 11 : 0.5252704930881716
Root mean square error 12 : 0.5271780217978237
Root mean square error 13 : 0.5322304301420868
Root mean square error 14 : 0.5331845030434721
Root mean square error 15 : 0.5389805191284746
Root mean square error 16 : 0.5432828453025182
```

Task 1 “H”:

When we are using Linear -regression we are getting best fit line when I compared with other algorithms like Gaussian or svr..etc

Task 2: Poker Project

Task 2 ”A”:

Few numeric attributes are introduced in the poker project They are:
1)Hand category: Player 1 & Player 2 hand categories varies from 0 to 9
2)Hand rank weights from 0 to 14
3)Coins of player 1 & player 2
4)Number of raises made by player 1 & player 2
5)Average amount of money raised by player 1 & player2

Task 2 “B”:

3 different learning algorithms are

1)Logistic regression:

The total correct predections are 9 out of 15
The accuracy is 60%

```
In [41]:  1  cm=confusion_matrix(y_pred,y_test)

In [42]:  1  cm

Out[42]: array([[6, 0, 2],
                [0, 2, 3],
                [1, 0, 1]], dtype=int64)

In [43]:  1  print('logistics regression accuracy:',accuracy_score(y_test,y_pred))

         logistics regression accuracy: 0.6
```

2) Gaussian navie_bayes¶

The total correct predections are 3 out of 15
The accuracy is 20%

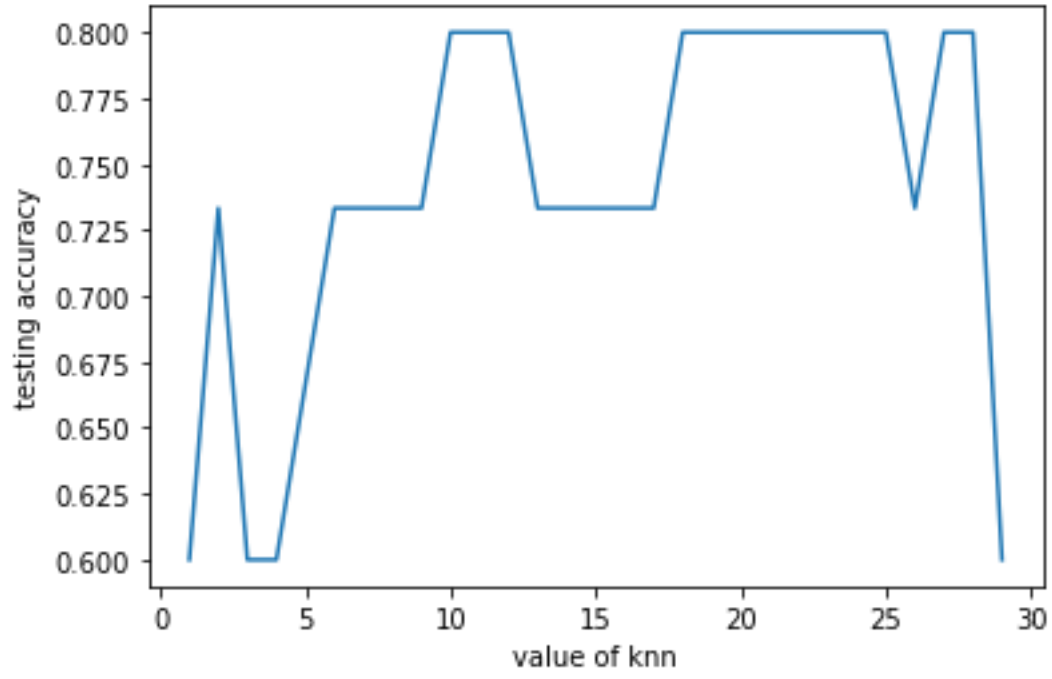
```
In [46]:  1  y_pred1=gn.predict(x_test)

In [47]:  1  cm1=confusion_matrix(y_pred1,y_test)

In [48]:  1  cm1

Out[48]: array([[0, 0, 1, 0],
                [2, 2, 0, 0],
                [0, 0, 1, 0],
                [5, 0, 4, 0]], dtype=int64)

In [49]:  1  print("accuracy score of Gaussian navie bayes:",accuracy_score(y_pred1,y_test))

         accuracy score of Gaussian navie bayes: 0.2
```

3)KNN:

The total correct predections are 7 out of 15
The accuracy is 46.67%

```
In [53]:
1 cm2=confusion_matrix(y_pred2,y_test)

In [54]:
1 cm2

Out[54]: array([[4, 0, 3, 0],
                [1, 2, 2, 0],
                [1, 0, 1, 0],
                [1, 0, 0, 0]], dtype=int64)

In [55]:
1 print("accuracy of knn:",accuracy_score(y_test,y_pred2))

accuracy of knn: 0.4666666666666667
```

Task 2 “C”:



From the above figure on x-axis(number of neighbours) ranges from 0 to 30 and on y-axis(tes t accuracy).
From the above figure we can say that when number of neighbours is 7 we get accuracy of 78%

```
In [60]:
1 y_pred2=knn.predict(x_test)
```

```
In [61]:
1 cm2=confusion_matrix(y_pred2,y_test)
```

```
In [62]:
1 cm2
```

```
Out[62]: array([[6, 1, 0],
                [2, 5, 0],
                [1, 0, 0]], dtype=int64)
```

```
In [63]:
1 print("accuracy of knn:",accuracy_score(y_test,y_pred2))
```

```
accuracy of knn: 0.7333333333333333
```