Machine Learning & Data Mining
T3 2019

COMP9417 – Homework-2

Chirag Panikkasseril Unni – z5241855
1/11/2019

Question 1

Part A

Decision Tree Results

Dataset	Default	0%	25%	50%	75%
australian	56.52% (2)	81.16% (7)	86.96% (2)	56.52% (2)	20.77% (7)
labor	66.67% (2)	94.44% (7)	44.44% (7)	66.67% (7)	50.00% (12)
diabetes	66.23% (2)	67.10% (7)	64.07% (12)	66.23% (2)	35.50% (27)
ionosphere	66.04% (2)	86.79% (7)	82.08% (27)	71.70% (7)	18.87% (12)

Part B

--- (4) increase overfitting by increasing max_depth of the decision tree

Part C

--- (5) yes, for 4/4 of the datasets

Question 2

Part A

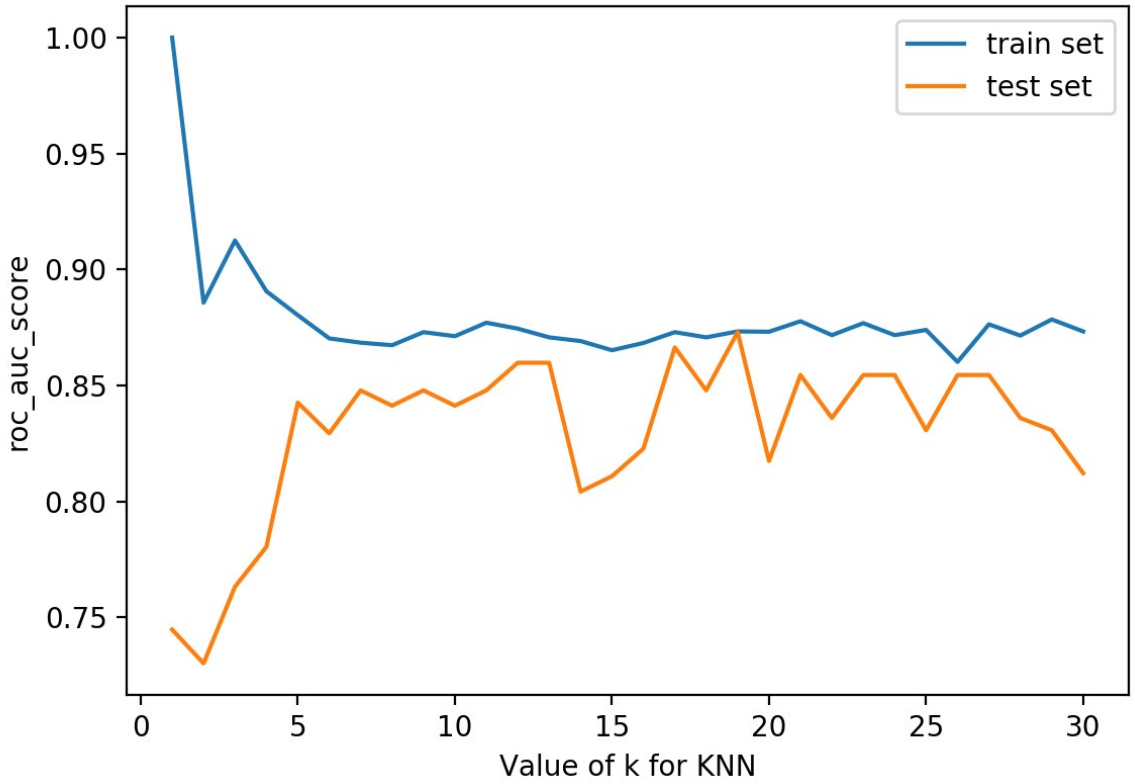
model A Train set
Train set accuracy: 0.8969404186795491

model A Test set
Test set accuracy: 0.7681159420289855

Part B

The optimal k : 19 with score 0.8121693121693122

Part C



Part D

Scores for Model with best k:19
Recall score: 0.8888888888888888
Precision score: 0.8

Scores for Model A with k:2
Recall score: 0.5555555555555556
Precision score: 0.7894736842105263

Code

```
# Code for question 2

import numpy as np
import pandas as pd
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,
precision_score, roc_auc_score, recall_score
```

```
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

############################################################
# Code for Question 2

# Load Data
df =
pd.read_csv('/home/chirag/anaconda3/envs/Homework2/CreditCa
rds.csv')
data = np.array(df)

# Pre Processing
normalizer = preprocessing.MinMaxScaler()
data[:, :14] = normalizer.fit_transform(data[:, :14])
X = data[:, :14]
y = data[:, -1:]

# Split Train and Test Set
train_set = 621
Xtrain, Xtest = X[:train_set, :], X[train_set:, :]
ytrain, ytest = y[:train_set, 0], y[train_set:, 0]

# PART A

# Model A
modelA = KNeighborsClassifier(n_neighbors=2)
# train the model
modelA.fit(Xtrain, ytrain)
# get the predict value from Xtrain
yprediction = modelA.predict(Xtrain)
# print the accuracy
print("\nmodel A Train set")
print(f'Train set accuracy:  {accuracy_score(ytrain,
yprediction)}')

# get the predict value from Xtest
print("\nmodel A Test set")
yprediction = modelA.predict(Xtest)
# print the scores
print(f'Test set accuracy:  {accuracy_score(ytest,
yprediction)}')

print("\nScores for Model A with k:2")
print(f'Recall score:  {recall_score(ytest, yprediction)}')
print(f'Precision score:  {precision_score(ytest,
yprediction)}')
```

```
print()


# PART B
List_K = []
AUCscoreTrain = []
AUCscoreTest = []
optimal_score = 0
optimal_k = 0

diff = []

for k in range(1, 31):
    # Model with k neighbour
    List_K.append(k)
    model = KNeighborsClassifier(n_neighbors=k)

    # train the model
    model.fit(Xtrain, ytrain)

    # get the predict value from Xtrain
    yprediction = model.predict(Xtrain)
    scoreTrain = roc_auc_score(ytrain, yprediction)
    AUCscoreTrain.append(scoreTrain)

    # get the predict value from Xtest
    yprediction = model.predict(Xtest)
    scoreTest = roc_auc_score(ytest, yprediction)
    AUCscoreTest.append(scoreTest)

    diff.append(abs(scoreTrain - scoreTest))

# get the optimal k and score
optimal_k = List_K[diff.index(min(diff))]
optimal_score = scoreTest

print(f'The optimal k : {optimal_k} with score
{optimal_score}\n')


#PART C
# plot
plt.plot(List_K, AUCscoreTrain, label='training set')
plt.plot(List_K, AUCscoreTest, label='testing set')
plt.xlabel('k value of the model')
plt.ylabel('roc_auc_score')
plt.legend()
```

```
plt.show()


#PART D

# Model with optimal k

model = KNeighborsClassifier(n_neighbors=optimal_k)
# train the model
model.fit(Xtrain, ytrain)

# get the predict value from Xtest
print(f"Scores for Model with best k:{optimal_k}")
yprediction = model.predict(Xtest)
print(f'Recall score:  {recall_score(ytest, yprediction)}')
print(f'Precision score:  {precision_score(ytest,
yprediction)}')
```

Code Output

model A Train set
Train set accuracy: 0.8969404186795491

model A Test set
Test set accuracy: 0.7681159420289855

Scores for Model A
Recall score: 0.5555555555555556
Precision score: 0.7894736842105263

The optimal k : 19 with score 0.8121693121693122

Scores for Model with best k:19
Recall score: 0.8888888888888888
Precision score: 0.8

Process finished with exit code 0