Lab 4

Artificial intelligence

Joseph Thankachan

961104-4877

- Task 1 A - kNN classifier
- Task 1 B - Other classification algorithms
- Task 1 C - Parameter tuning
- Task 1 E - kNN Regressor
- Task 1 F - Other regression algorithms
- Task 1 G - Parameter tuning
- Task 1 H - Comparison
- Task 2 A - More attributes
- Task 2 B - Learning algorithms
- Task 2 C - Cross validation

task 1

1.a.

In this task implement **k-nearest Neighbours algorithm** to classify the driver performance in the data set

In both cases, the input consists of the k closest training examples in the feature space. In the function “load Dataset” we have loaded “Lab4Data.csv”. Then we divided train and test data. We have predicted the accuracy 80.5%

steps

- split data set into training set and testing set
- calculate the distance using Euclidean or Manhattan method and sort the data based on the distance
- predict an output based on k value and maximum occurrence of a value from the neighbours.
- Compare predicted output and actual output then find percentage.

```
Length of Total Data: 2000

Train set: 1600

Test set: 400

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%

# predicted=2.0, actual=2.0

# predicted=3.0, actual=3.0

# predicted=1.0, actual=1.0

# predicted=3.0, actual=2.0

# predicted=1.0, actual=1.0

# predicted=3.0, actual=3.0
# predicted=3.0, actual=2.0

# predicted=1.0, actual=1.0

# predicted=2.0, actual=1.0

# predicted=2.0, actual=2.0

# predicted=1.0, actual=1.0

# predicted=3.0, actual=1.0

# predicted=2.0, actual=2.0

# predicted=1.0, actual=1.0

# predicted=3.0, actual=3.0

# predicted=1.0, actual=1.0

# predicted=2.0, actual=2.0

# predicted=3.0, actual=3.0

# predicted=3.0, actual=2.0

# predicted=2.0, actual=1.0

# predicted=1.0, actual=1.0

# predicted=1.0, actual=1.0

# predicted=2.0, actual=2.0

# predicted=1.0, actual=1.0

# predicted=2.0, actual=1.0

Accuracy: 80.5%
```

1.b

In this task implement 3 different classification algorithms support vector machine (SVM), MLP and **Decision Tree** using “scikit learn”

- Implement three classification algorithm support vector machine (SVM), decision tree and MLP.
- SVM perform more better than others.
- Decision tree better than KNN

```
Total length of Data =
2000%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Total length of Data = 1600length of Data =
400%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%S
VC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
kernel='linear', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001,
verbose=False)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%Number of  Predictions= 364 Out_of= 400 Test DataPercentage of Accuracy
Prediction =
91.0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%DecisionTr
eeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False,
random_state=None,
splitter='best')%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%MLPClassifier(activation='relu', alpha=1e-05, batch_size='auto',
beta_1=0.9,              beta_2=0.999, early_stopping=False, epsilon=1e-08,
hidden_layer_sizes=(5, 1), learning_rate='constant',
learning_rate_init=0.001, max_iter=200, momentum=0.9,
n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
random_state=1, shuffle=True, solver='adam', tol=0.0001,
validation_fraction=0.1, verbose=False,
warm_start=False)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%Number of Correct Predictions: 127 Out_of: 400 Test DataPercentage of
Accuracy Prediction =
31.75%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%Number of Correct Predictions: 345 Out_of: 400  Test DataPercentage of Accuracy
Prediction : 86.25
```

1.c

used distance and numbers of Neighbours parameters for tuning our KNN model.

```
Length of Train Data: 150Length of Test Data
50%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%KNeighbor
sClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=19, p=1,
weights='uniform')Accuracy of Prediction in Percent: [70.04487179487178,
62.08653846153847, 70.6698717948718, 72.00320512820512, 76.0448717948718,
75.96153846153847, 75.33653846153847, 74.00320512820512, 76.0448717948718,
74.71153846153845, 73.37820512820514, 72.71153846153847, 74.75320512820512,
72.71153846153847, 74.04487179487178, 74.08653846153845, 75.37820512820514,
73.46153846153845, 74.75320512820514]Highest Accuracy for manhattan:
76.0448717948718%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=19, p=2,
weights='uniform')Accuracy of Prediction in Percent: [69.4198717948718,
60.02564102564103, 70.00320512820511, 71.33653846153845, 74.0448717948718,
74.0448717948718, 76.00320512820514, 76.00320512820514, 75.37820512820512,
75.37820512820512, 74.0448717948718, 72.71153846153847, 74.0448717948718,
74.0448717948718, 74.71153846153847, 75.37820512820512, 76.08653846153845,
76.08653846153845, 76.75320512820512]Highest Accuracy for euclidean:
76.75320512820512
```

The best result is 76.75%. It obtained for distance=1 (Manhattan) and neighbour =7

1.e Implement k-NN regression

Implemented regression and use the fuel consumption data (Lab4Data.csv) results are given below:

steps

- split data set into training set and testing set
- calculate the distance using Euclidean or Manhattan method and also sort the data based on the distance
- predict an output based on k value and maximum occurrence of a value from the neighbours.
- Compare predicted output and actual output then find percentage.

```
Data Length = 2000Train set: 1600Test set: 400# predicted=35.56433333333333,
actual=35.057# predicted=34.79633333333334, actual=33.671#
predicted=34.38966666666666, actual=34.126# predicted=33.318666666666665,
actual=33.793# predicted=33.370666666666665, actual=32.714#
predicted=34.51533333333334, actual=34.511# predicted=36.21766666666667,
actual=35.594# predicted=33.252, actual=32.363# predicted=32.544333333333334,
actual=31.182# predicted=35.16833333333333, actual=37.318#
predicted=30.883333333333336, actual=29.947# predicted=30.144333333333332,
actual=29.236# predicted=29.887, actual=30.029# predicted=34.63433333333333,
actual=33.251# predicted=34.24433333333334, actual=33.881#
predicted=31.175666666666668, actual=33.878# predicted=30.439666666666668,
actual=31.302# predicted=34.717666666666666, actual=34.054#
predicted=35.23666666666667, actual=36.21# predicted=34.27666666666667,
actual=33.104# predicted=33.169333333333334, actual=32.126#
predicted=32.02066666666666, actual=35.205# predicted=32.43366666666667,
actual=32.997# predicted=34.525000000000006, actual=35.196#
predicted=34.11066666666667, actual=34.845# predicted=32.11266666666666,
actual=32.173Accuracy: 44.0%
```

1.f

Implemented 3 different classification algorithms support vector machine (SVM), MLP and **Decision Tree** using "scikit learn"

- Implement three classification algorithm support vector machine (SVM), decision tree and MLP same as task 1b.
- SVM perform more better than others.
- Decision tree better than KNN

```
Length of Total Data: 
2000%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Len
gth of Train Data: 1600Length of Test Data 
400%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%SV
R(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, 
gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,    tol=0.001, 
verbose=False)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%Number of Correct Predictions: 371 Out_of: 400 Number of Test 
DataAccuracy of Prediction in Percent: 
92.75%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None, 
max_leaf_nodes=None, min_impurity_decrease=0.0, 
min_impurity_split=None, min_samples_leaf=1, 
min_samples_split=2, min_weight_fraction_leaf=0.0, 
presort=False, random_state=None, 
splitter='best')%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', 
beta_1=0.9,              beta_2=0.999, early_stopping=False, epsilon=1e-08, 
hidden_layer_sizes=(15,), learning_rate='constant', 
learning_rate_init=0.001, max_iter=1, momentum=0.9, 
n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5, 
random_state=1, shuffle=True, solver='adam', tol=0.0001, 
validation_fraction=0.1, verbose=False, 
warm_start=True)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%Number of Correct Predictions: 123 Out_of: 400 Number of Test 
DataAccuracy of Prediction in Percent: 
30.75%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%Number of Correct Predictions: 123 Out_of: 400 Number of Test DataAccuracy of 
Prediction in Percent: 30.75
```

1.g

used distance and numbers of Neighbours parameters for tuning our KNN model.

```
Length of Total Data: 
2000%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Length of 
Train Data: 1600Length of Test Data 
400%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%KNeigh
borsRegressor(algorithm='auto', leaf_size=30, metric='minkowski', 
metric_params=None, n_jobs=None, n_neighbors=19, p=1, 
weights='uniform')Accuracy of Prediction in Percent: [-0.2125, -
0.16593750000000002, -0.16090277777777778, -0.15492187500000001, -
0.14902500000000002, -0.14673611111111112, -0.1472576530612245, -0.146220703125, -
0.14680555555555558, -0.1483375, -0.14920971074380165, -0.1491927083333333, -
0.1513942307692308, -0.15214285714285714, -0.15433333333333332, -
0.15498291015625004, -0.15741782006920418, -0.1588695987654321, -
0.16038088642659282]Highest Accuracy
```

```
for manhattan:  -
0.146220703125%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski', 
metric_params=None, n_jobs=None, n_neighbors=19, p=2, 
weights='uniform')Accuracy of Prediction in Percent: [-0.21937500000000001, -
0.18125, -0.17381944444444444, -0.16746093750000002, -0.16417500000000002, -
0.16272569444444446, -0.16172193877551022, -0.160244140625, -0.16118055555555558,
-0.16165625, -0.16389979338842975, -0.16678819444444443, -0.1676812130177515, -
0.16954081632653062, -0.17144444444444445, -0.1734912109375, -0.17572880622837372,
-0.17904320987654324, -0.18148891966759]Highest Accuracy for euclidean:  -
0.160244140625
```

The best result is 83.12%. It obtained for distance=1 (Manhattan) and neighbour =7

Task 2

2.a

used “Hand strength”, “Call”, “Average Raise”, “Ratio”, “Stack1”, “Stack2” attributes.

2.b

Implemented 3 different classification algorithms support vector machine (SVM), MLP and **Decision Tree** using “scikit learn”

```
Length of Train Data: 150Length of Test Data
50%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%SV
C(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
kernel='linear', max_iter=-1, probability=False, random_state=None,
shrinking=True, tol=0.001,
verbose=False)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%Number of Correct Predictions: 37 Out_of: 50 Number of Test
DataAccuracy of Prediction in Percent:
74.0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False,
random_state=None,
splitter='best')%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto',
beta_1=0.9,              beta_2=0.999, early_stopping=False, epsilon=1e-08,
hidden_layer_sizes=(100,), learning_rate='constant',
learning_rate_init=0.001, max_iter=200, momentum=0.9,
n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
random_state=None, shuffle=True, solver='adam', tol=0.0001,
validation_fraction=0.1, verbose=False,
warm_start=False)%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%Number of Correct Predictions: 42 Out_of: 50 Number of Test
DataAccuracy of Prediction in Percent:
84.0%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Number of Correct Predictions: 42 Out_of: 50 Number of Test DataAccuracy of
Prediction in Percent: 84.0
```

2.c

used distance and numbers of Neighbours parameters for tuning our KNN model.

```
Length of Train Data: 150Length of Test Data
50%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%KNeighbor
sClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=19, p=1,
weights='uniform')Accuracy of Prediction in Percent: [70.04487179487178,
62.08653846153847, 70.6698717948718, 72.00320512820512, 76.0448717948718,
75.96153846153847, 75.33653846153847, 74.00320512820512, 76.0448717948718,
74.71153846153845, 73.37820512820514, 72.71153846153847, 74.75320512820512,
72.71153846153847, 74.04487179487178, 74.08653846153845, 75.37820512820514,
73.46153846153845, 74.75320512820514]Highest Accuracy for manhattan:
76.0448717948718%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
metric_params=None, n_jobs=None, n_neighbors=19, p=2,
weights='uniform')Accuracy of Prediction in Percent: [69.4198717948718,
60.02564102564103, 70.00320512820511, 71.33653846153845, 74.0448717948718,
74.0448717948718, 76.00320512820514, 76.00320512820514, 75.37820512820512,
75.37820512820512, 74.0448717948718, 72.71153846153847, 74.0448717948718,
74.0448717948718, 74.71153846153847, 75.37820512820512, 76.08653846153845,
76.08653846153845, 76.75320512820512]Highest Accuracy for euclidean:
76.75320512820512
```

The best result is 76.75%. It obtained for distance=1 (Manhattan) and neighbour =7