Part A

Part B

Question 1

from sklearn.model\_selection import train\_test\_split

import pandas as pd

# Read the data into a pandas dataframe

data = pd.read\_csv('pima-indians-diabetes.data.csv',delimiter=',')

#Remove the target variables for x. Y only has the target variables

x = data.drop("Class variable (0 or 1)", axis=1)

y = data["Class variable (0 or 1)"]

#Split it. Training has 70% of the records. Test has 30%

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size=0.3)

#Combine the target and features together

train = pd.concat([X\_train, y\_train], axis=1)

test = pd.concat([X\_test, y\_test], axis=1)

#save the two sets to csv files

train.to\_csv("train.csv")

test.to\_csv("test.csv")

The csv files were download and split. And then saved to two csv files. This is done to keep the data consistent, and therefore ours results comparable. Code shown above

Step 1

from sklearn.tree import DecisionTreeClassifier

for i in range(2, 20, 2):

    classifier = DecisionTreeClassifier(max\_depth=i)

    classifier.fit(X\_train, y\_train)

    y\_pred = classifier.predict(X\_test)

    print(str(i) + " " + str(accuracy\_score(y\_test, y\_pred)))

|  |  |
| --- | --- |
| max\_depth | Test accuracy |
| 2 | 0.696969696969697 |
| 4 | 0.6926406926406926 |
| 6 | 0.6623376623376623 |
| 8 | 0.670995670995671 |
| 10 | 0.6753246753246753 |
| 12 | 0.658008658008658 |
| 14 | 0.6666666666666666 |
| 16 | 0.6623376623376623 |
| 18 | 0.653679653679653 |

rom sklearn.neural\_network import MLPClassifier

x = np.arange(0.001, 0.01, 0.001)

for i in x:

    classifier = MLPClassifier(learning\_rate\_init=i)

    classifier.fit(X\_train, y\_train)

    y\_pred = classifier.predict(X\_test)

    print(str(i) + " " + str(accuracy\_score(y\_test, y\_pred)))

Step 2

|  |  |
| --- | --- |
| Learning rate | Test accuracy |
| 0.001 | 0.7272727272727273 |
| 0.003 | 0.7142857142857143 |
| 0.004 | 0.7402597402597403 |
| 0.005 | 0.7056277056277056 |
| 0.006 | 0.7186147186147186 |
| 0.007 | 0.7056277056277056 |
| 0.008 | 0.70995670995671 |
| 0.009 | 0.683982683982684 |

for k in range(2, 7, 1):

    fs = SelectKBest(chi2, k=k)

    fs.fit(X\_train, y\_train)

    X\_train\_fs = fs.transform(X\_train)

    X\_test\_fs = fs.transform(X\_test)

    classifier = DecisionTreeClassifier(max\_depth=2)

    classifier.fit(X\_train\_fs, y\_train)

    y\_pred = classifier.predict(X\_test\_fs)

    print(str(k) + " " + str(accuracy\_score(y\_test, y\_pred)))

Step 3

|  |  |
| --- | --- |
| k | Test accuracy |
| 2 | 0.6926406926406926 |
| 3 | 0.6883116883116883 |
| 4 | 0.7142857142857143 |
| 5 | 0.696969696969697 |
| 6 | 0.6623376623376623 |

for k in range(2, 7, 1):

    fs = SelectKBest(chi2, k=k)

    fs.fit(X\_train, y\_train)

    X\_train\_fs = fs.transform(X\_train)

    X\_test\_fs = fs.transform(X\_test)

    classifier = MLPClassifier(learning\_rate\_init=0.004)

    classifier.fit(X\_train\_fs, y\_train)

    y\_pred = classifier.predict(X\_test\_fs)

    print(str(k) + " " + str(accuracy\_score(y\_test, y\_pred)))

Step 4

|  |  |
| --- | --- |
| k | Test accuracy |
| 2 | 0.7402597402597403 |
| 3 | 0.7359307359307359 |
| 4 | 0.7359307359307359 |
| 5 | 0.7402597402597403 |
| 6 | 0.7489177489177489 |

Step 5

Question 2

1)

from sklearn.neural\_network import MLPClassifier

classifier = MLPClassifier(hidden\_layer\_sizes=[1, 20], max\_iter=150)

classifier.fit(X\_train, y\_train)

y\_pred = classifier.predict(X\_test)

print(str(accuracy\_score(y\_test, y\_pred)))

0.683982683982684

2)

for i in range(1, 20, 1):

    classifier = MLPClassifier(hidden\_layer\_sizes=(20-i, i), max\_iter=150)

    classifier.fit(X\_train, y\_train)

    y\_pred = classifier.predict(X\_test)

    print("(" + str(20-i)  + ", " + str(i) + ") " + str(accuracy\_score(y\_test, y\_pred)))

|  |  |
| --- | --- |
| Neuron combination | Test accuracy |
| (19, 1) | 0.645021645021645 |
| (18, 2) | 0.7186147186147186 |
| (17, 3) | 0.7272727272727273 |
| (16, 4) | 0.6493506493506493 |
| (15, 5) | 0.7229437229437229 |
| (14, 6) | 0.7272727272727273 |
| (13, 7) | 0.7316017316017316 |
| (12, 8) | 0.7489177489177489 |
| (11, 9) | 0.7186147186147186 |
| (10, 10) | 0.7229437229437229 |
| (9, 11) | 0.7359307359307359 |
| (8, 12) | 0.7532467532467533 |
| (7, 13) | 0.7229437229437229 |
| (6, 14) | 0.7575757575757576 |
| (5, 15) | 0.7359307359307359 |
| (4, 16) | 0.7402597402597403 |
| (3, 17) | 0.7229437229437229 |
| (2, 18) | 0.645021645021645 |
| (1, 19) | 0.645021645021645 |

3)

4)

data = pd.read\_csv('car.data.csv',delimiter=',')

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

for i in data.columns:

    data[i]=le.fit\_transform(data[i])

|  |  |
| --- | --- |
| Neuron combination | Test accuracy |
| (19, 1) | 0.7129094412331407 |
| (18, 2) | 0.7148362235067437 |
| (17, 3) | 0.7109826589595376 |
| (16, 4) | 0.7148362235067437 |
| (15, 5) | 0.7842003853564548 |
| (14, 6) | 0.7784200385356455 |
| (13, 7) | 0.7341040462427746 |
| (12, 8) | 0.7109826589595376 |
| (11, 9) | 0.7186897880539499 |
| (10, 10) | 0.7514450867052023 |
| (9, 11) | 0.791907514450867 |
| (8, 12) | 0.7649325626204239 |
| (7, 13) | 0.6763005780346821 |
| (6, 14) | 0.7533718689788054 |
| (5, 15) | 0.6685934489402697 |
| (4, 16) | 0.7129094412331407 |
| (3, 17) | 0.7244701348747592 |
| (2, 18) | 0.7129094412331407 |
| (1, 19) | 0.7129094412331407 |

5)