HW 1

Please start working on this assignment as soon as possible. If you are a beginner in Python this might take a long time. The **objective** of this assignment is to help you familiarize w python packages related to machine learning, namely scikit-learn package.

DO NOT ERASE MARKDOWN CELLS AND INSTRUCTIONS IN YOUR HW submission

Instructions

This assignment covers several aspects of KNN Classifier and performence evaluation we have covered in introML module. eep the following in mind:

- Structure your notebook cells as sugested
- **Q** - QUESTION posted in a markdown cell
  - it explains the task in details
  - it is marked with **Q1**, ... **Q10** ...
- **A** - Marks the location where you need to enter your answer below
  - it can be `python code` (more often) or markdown cell (less often)
  - it is marked with **A1**, ... **A10** ... and you enter your answers **below**
  - make sure the cell is running and produces no errors
- Before you submit the HW:
  - Make sure your notebook can always be rerun from top to bottom.
- Follow instructions given in canvas for homework submission.

Tutorials

- KNN with sklearn
- Confusion Matrix
- Plot Confursion Matrix with Sklearn

1. CLASSIFICATION USING KNN ALGORITHM

Data is in the ../data/ folder, and datafile name is heart.dat **Keep** the relative path from **HW** folder to **data** folder in your submission e.g. you will access the file as ../data/heart.dat

Q1 use pandas to read ../data/heart.dat

- NOTE : use separator as space while reading this data
- Use column names from metadata in given order
- NOTE : YOU WON'T SEE 'PRESENCE' in metadata (in attribute information)

A1 Replace the ? mark with your answer

In [ ]:
```
import pandas as pd
columns = ['age', 'sex', 'type', 'pressure', 'cholestoral', 'blood sugar', 'ecg', 'hea
#columns = ?

df = pd.read_csv('../data/heart.dat', names=columns, delim_whitespace=True)
#df.head()
```

Q2

1. Have a look at head and tail of your data

- N.B: You can use .tail and .head methods
- N.B: Print both of them, if you just run `without printing` only output from last command will be printed

1. Let us view the size of dataset as well

- print data shape

1. Now let us see if there is some missing value
2. If there is any na values drop it

N.B You can add more cells as per your need.

A2 Replace ??? with code in the code cell below

In [ ]:
```
# Code goes below
df.head() #prints head

df.tail() #prints tail

df.shape    # shows shape

df.isnull() # shows if cells are null or not

df.dropna(axis=0) # axis=0 drops rows that have null values
```

Out[ ]:

	age	sex	type	pressure	cholestoral	blood sugar	ecg	heart rate	angina	oldpeak	slope	major vessels	thal
0	70.0	1.0	4.0	130.0	322.0	0.0	2.0	109.0	0.0	2.4	2.0	3.0	3.0
1	67.0	0.0	3.0	115.0	564.0	0.0	2.0	160.0	0.0	1.6	2.0	0.0	7.0
2	57.0	1.0	2.0	124.0	261.0	0.0	0.0	141.0	0.0	0.3	1.0	0.0	7.0
3	64.0	1.0	4.0	128.0	263.0	0.0	0.0	105.0	1.0	0.2	2.0	1.0	7.0
4	74.0	0.0	2.0	120.0	269.0	0.0	2.0	121.0	1.0	0.2	1.0	1.0	3.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
265	52.0	1.0	3.0	172.0	199.0	1.0	0.0	162.0	0.0	0.5	1.0	0.0	7.0
266	44.0	1.0	2.0	120.0	263.0	0.0	0.0	173.0	0.0	0.0	1.0	0.0	7.0
267	56.0	0.0	2.0	140.0	294.0	0.0	2.0	153.0	0.0	1.3	2.0	0.0	3.0
268	57.0	1.0	4.0	140.0	192.0	0.0	0.0	148.0	0.0	0.4	2.0	0.0	6.0
269	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0

270 rows × 14 columns

Q3 Now we will look deeper into the dataset

- Use pairplot from sns to plot this data frame
- See the statistics of the data by describing dataframe

A3 Replace ??? with code in the code cell below

In [ ]:

```
import seaborn as sns

sns.set(style="darkgrid", color_codes=True)
g=sns.pairplot(df)

#import matplotlib.pyplot as plt
#plt.?

#describe dataframe
df.describe()
```

```
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning:
use_inf_as_na option is deprecated and will be removed in a future version. Convert i
nf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Hunter\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The
figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[ ]:

	age	sex	type	pressure	cholestoral	blood sugar	ecg	heart rate
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	1.022222	149.677778
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	0.997891	23.165717
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000
25%	48.000000	0.000000	3.000000	120.000000	213.000000	0.000000	0.000000	133.000000
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	2.000000	153.500000
75%	61.000000	1.000000	4.000000	140.000000	280.000000	0.000000	2.000000	166.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000

Q4

1. From the above pairplot what kind of relationship we can derive between age and heartrate?

A4 Write your answer here.....

Q4 Answer

As age goes up, the heart rate also trends down, but with a lot of variance.

Q5 If you go through metadata (heart.doc) (Attribute Information:) you will see that all data in our dataframe are not of same types.

- So we should deal them accordingly.
- We don't have to do anything to 'real' data. However we have to deal with ordered data and nominal data
- We only need to convert all nominal and ordered data to dummy variables

A5 Replace ??? with code in the code cell below

In [ ]:
```
# 11, 7, 3, 13
# slope, ecg, type, thal
dummy_list = ['slope','ecg','type','thal']
df = pd.get_dummies(df, columns=dummy_list, prefix=dummy_list, prefix_sep='-')
df.head()
```

Out[ ]:

	age	sex	pressure	cholestoral	blood sugar	heart rate	angina	oldpeak	major vessels	presence	...	ecg-0.0	ecg-1.0
0	70.0	1.0	130.0	322.0	0.0	109.0	0.0	2.4	3.0	2	...	False	False
1	67.0	0.0	115.0	564.0	0.0	160.0	0.0	1.6	0.0	1	...	False	False
2	57.0	1.0	124.0	261.0	0.0	141.0	0.0	0.3	0.0	2	...	True	False
3	64.0	1.0	128.0	263.0	0.0	105.0	1.0	0.2	1.0	1	...	True	False
4	74.0	0.0	120.0	269.0	0.0	121.0	1.0	0.2	1.0	1	...	False	False

5 rows × 23 columns

KNN Model from sklearn

Q6 Get training data from the dataframe

1. Assign values of `presence` column to y, note you have to use .values method
2. Drop 'presence' column from data frame,
3. Assign df values to x

Split dataset into train and test data use train_test_split

1. Use stratify = y and test_size = 0.25 and random_state = 123
2. Create a KNN model using sklearn library, `Initialize n_neighbors = 4` , `(See the documenttaion for details)`
3. Fit the model with the train data

A6 Replace ??? with code in the code cell below

In [ ]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
# Assign values of ```presence``` column to y, note you have to use .values method
y = df['presence'].values # Documentation suggests using DataFrame.to_numpy() instead
# Drop 'presence' column from data frame,
df_presence_dropped = df.drop(columns=['presence']) # calling drop returns a new df wi

# Assign df values to x
x = df_presence_dropped.values
# View shape of x and y
x.shape, y.shape

# Use stratify = y and test_size = 0.25 and random_state = 123

#xtrain, xtest, ytrain, ytest = ?
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state=123
# Create a KNN model using sklearn library, k=4
knn = KNeighborsClassifier(n_neighbors=4)

# Fit the model with the train data
knn.fit(xtrain, ytrain)
```

Out[ ]:
```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

Q7 *Analysis*

- Predict xtest and view first 20 predicitons
- Compare prediction with real ytest 20 predictions
- Print the score with test data

The way we fit the dataset is not good

Normalization

- rescale only real value columns
- For each column normalize `df[col] as (x - mean) / standard_deviation`

A7 Replace ??? with code in the code cell below

In [ ]:
```
#  Predict xtest and view first 25 predicitons
print(knn.predict(xtest)[0:25])

# Compare prediction with real ytest 25 predictions
```

```
print(ytest[0:25])

# Print the score with test data
print(knn.score(xtest, ytest))

#rescale only real value columns
realcols = ['age', 'pressure', 'cholestoral', 'heart rate', 'oldpeak', 'major vessels'
# For each column normalize ```df[col] as (x - mean) / standard_deviation```

for col in realcols:
    mean = df[col].mean()
    std = df[col].std()
    df[col] = (df[col] - mean) / std
```

```
[1 1 1 1 2 2 2 1 1 1 2 1 2 2 2 1 1 1 1 1 1 2 1 1 1]
[1 2 2 2 2 1 1 1 1 2 2 1 2 1 1 1 1 1 1 1 2 2 2 1 1]
0.6617647058823529
```

Q8 Write the code to train new model using KNN classifier, k=4 (same as above)

A8 Replace ??? with code in the code cell below

In [ ]:
```
# update x
x = df.drop(columns=['presence']).values
# Train test Split
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=0.25, random_state=123

# Model Initialization
knn = KNeighborsClassifier(n_neighbors=4)

# Model fitting with training data
knn.fit(xtrain, ytrain)
print(knn.predict(xtest))
print(ytest)
# Now print score on test data
knn.score(xtest, ytest)
```

Out[ ]:
```
[1 1 2 1 2 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 2 1 2 2 1 1 1
 1 1 2 1 1 2 1 1 2 1 1 2 2 2 2 2 1 1 1 1 1 1 1 2 2 2 1 1 2 2]
[1 2 2 2 2 1 1 1 1 2 2 1 2 1 1 1 1 1 1 2 2 2 1 1 1 2 1 1 2 2 2 2 2 1 2 1
 1 2 2 1 1 2 1 1 2 1 1 1 2 2 1 2 1 1 2 1 1 1 1 1 1 2 2 2 1 2 2]
0.7941176470588235
```

Q9 Lets analyze the difference between two modeling strategies (data normalization) Compare score with and without data normalization process and explain

A9

Without normalization, the score is at 66%. With normalization, the score rises to 79%. Normalizations seems to help the ML model make better predictions.

Q10 Now we will write a function that will initialize, fit and return score on test data for given values of k and Plot result

1. Use values from 1 to 25(inclusive) and get score and plot as a line graph

- Hint : For advance method you can use map (recall functional programming from last exercise) or you can use simple loops

1. Finally you can print the best value of k by getting the index

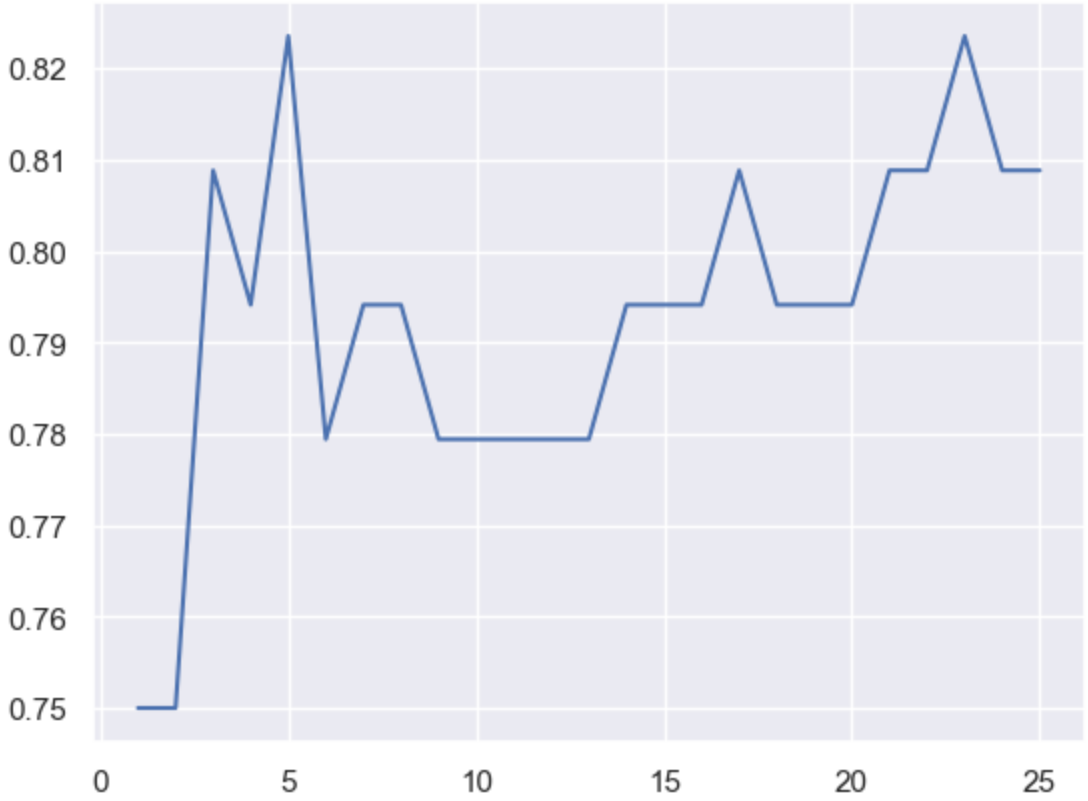
- N.B: Note index starts with 0 but values of k starts with 1 so actual value of k will be 1 more
- You can use `np.argmax()` function

1. Now define your best model as bestknn and print score

A10 Write the code below (replace??)

In [ ]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

def returnScore(k, xtrain, xtest, ytrain, ytest):
  knn = KNeighborsClassifier(n_neighbors=k)
  knn.fit(xtrain, ytrain)
  return knn.score(xtest, ytest)

result = [*map(lambda i: returnScore(i, xtrain, xtest, ytrain, ytest), range(1, 26))]
print(result)
plt.plot(range(1, 26), result) # I messed with the plot a little to make the x values

print('BEST VALUE OF K', np.argmax(result) + 1)

bestknn = KNeighborsClassifier(n_neighbors=(np.argmax(result) + 1))
bestknn.fit(xtrain, ytrain)
bestknn.score(xtest, ytest)


ypred = bestknn.predict(xtest)
print(ypred)
matrix = confusion_matrix(ytest, ypred)
print(matrix)
```

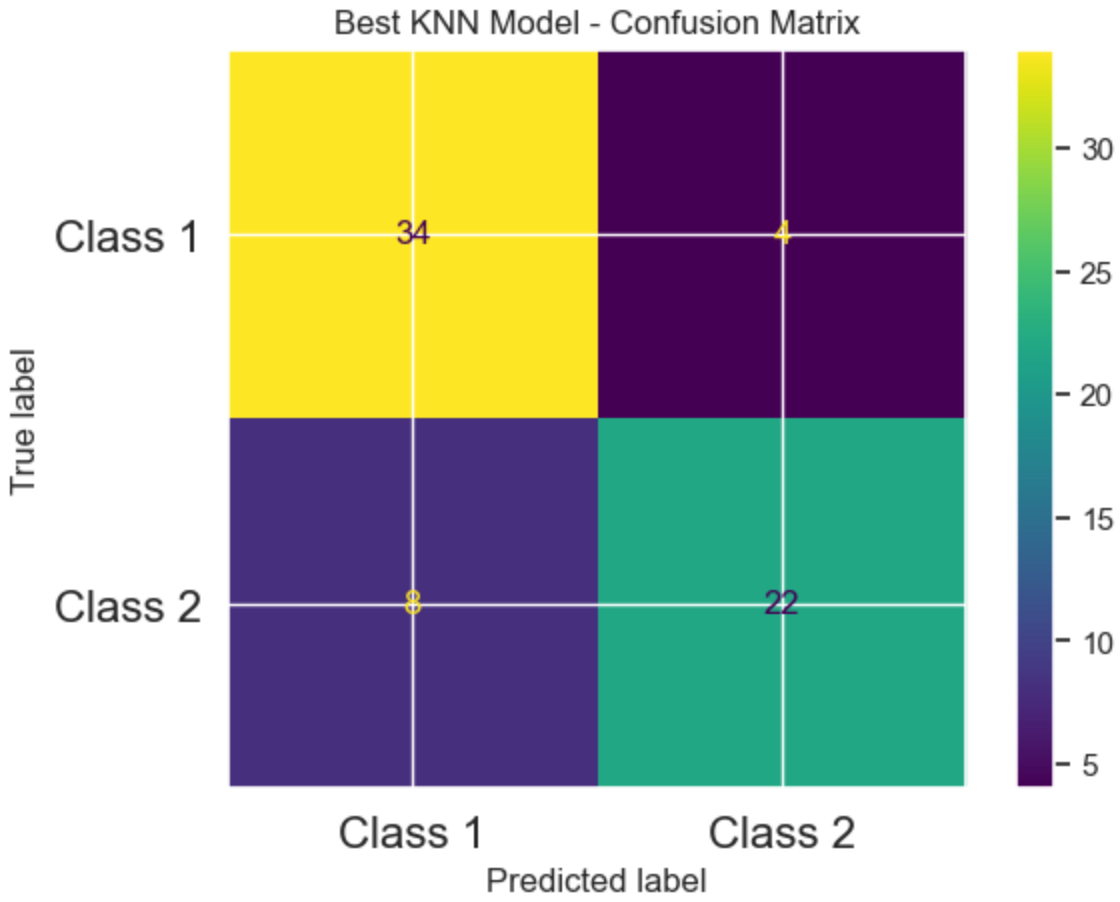
```
[0.75, 0.75, 0.8088235294117647, 0.7941176470588235, 0.8235294117647058, 0.7794117647
058824, 0.7941176470588235, 0.7941176470588235, 0.7794117647058824, 0.779411764705882
4, 0.7794117647058824, 0.7794117647058824, 0.7794117647058824, 0.7941176470588235, 0.
7941176470588235, 0.7941176470588235, 0.8088235294117647, 0.7941176470588235, 0.79411
76470588235, 0.7941176470588235, 0.8088235294117647, 0.8088235294117647, 0.8235294117
647058, 0.8088235294117647, 0.8088235294117647]
BEST VALUE OF K 5
[1 1 2 1 2 1 1 1 1 1 2 1 2 2 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 2 1 2 2 1 1 1
 1 2 2 1 1 2 1 1 2 1 1 2 2 2 2 2 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2]
[[34  4]
 [ 8 22]]
```



Q11 Now plot confusion matrix using ConfusionMatrixDisplay, for xtest data. Use the Best KNN model from the above question as the estimator. See Visualization with Display Objects example.

A11 Replace ??? with code in the code cell below

In [ ]:
```
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt

cm = ConfusionMatrixDisplay(confusion_matrix=matrix)
cm.plot()
plt.title("Best KNN Model - Confusion Matrix")
plt.xticks(range(2), ["Class 1","Class 2"], fontsize=16)
plt.yticks(range(2), ["Class 1","Class 2"], fontsize=16)
plt.show()
```



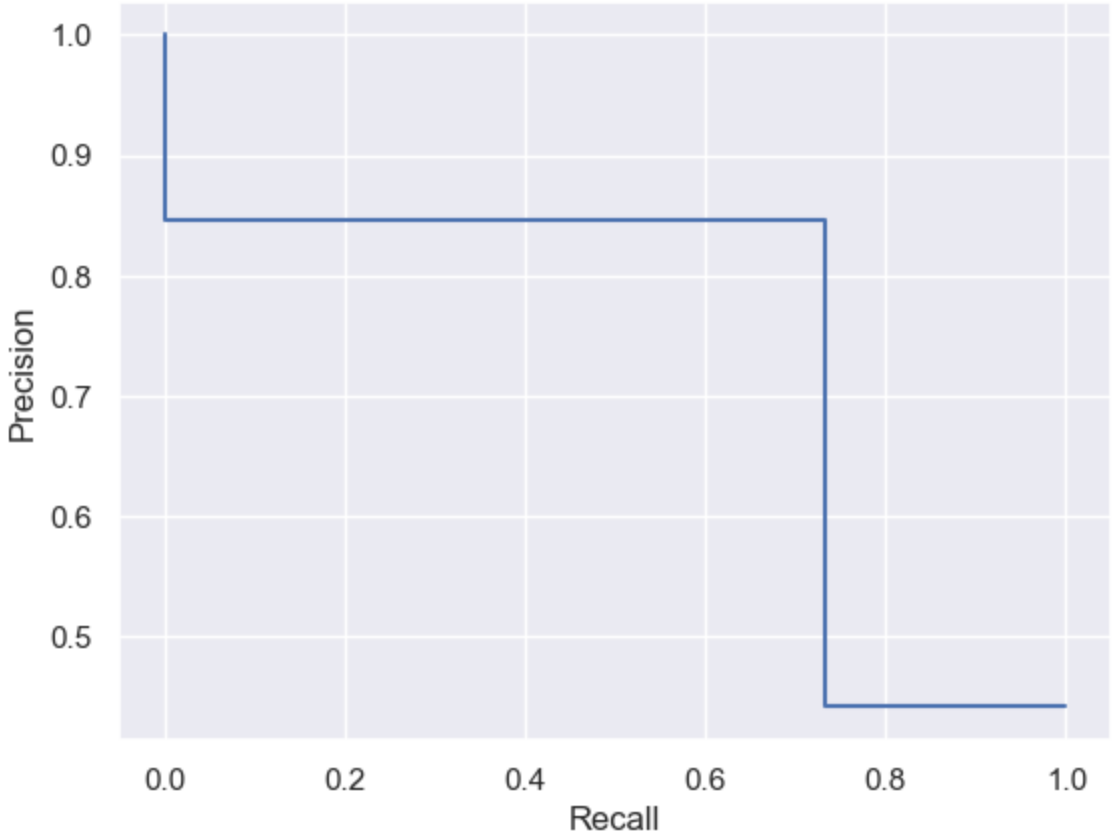
Q12:

1. Calculate the test MSE
2. Get the score from the model using test data
3. Plot Precision-Recall Curve from the true & predicted test data (Use sklearn PrecisionRecallDisplay)

A12 Replace ??? with code in the code cell below

In [ ]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import PrecisionRecallDisplay, precision_recall_curve
import matplotlib.pyplot as plt

# Calculate the test MSE
mse = mean_squared_error(ytest, ypred)
print("Test mean squared error (MSE): {:.2f}".format(mse))

print(bestknn.score(xtest, ytest))
ytest_binary = np.array(ytest) - 1
ypred_binary = np.array(ypred) - 1
# precision = tp / (tp + fp)
# recall = tp / (tp + fn)
precision, recall, _ = precision_recall_curve(ytest_binary, ypred_binary)
PrecisionRecallDisplay(precision=precision, recall=recall).plot()
plt.show()
```

```
Test mean squared error (MSE): 0.18
0.8235294117647058
```



Further reading

- KNN model creation
- Example of KNN

Submission Instructions

1. Run all cells in `HW1.ipynb` and make sure there are no errors
2. Print `HW1.ipynb` to pdf file
3. Create a Folder HW0 and Upload `HW1.ipynb` and `HW1.pdf` files to your git repo allocated for this course e.g: https://git.txstate.edu/NetID/netid before the deadline. Make Sure Instructor and TA has access for the repo.