HW 2

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 m1 (../practice/m1/README.md) module. eep the following in mind:

- Structure your notebook (https://git.txstate.edu/ML/2022Spring/blob/master/tutorials/notebook-checklist.md) 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 README.md (README.md) for homework submission instructions

Tutorials

- KNN with sklearn (https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html)
- Confusion Matrix (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html)
- Plot Confursion Matrix with Sklearn (https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)

1. CLASSIFICATION USING KNN ALGORITHM

Data Get the exploratory data and the folowwing files:
`http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/heart/`

- Link contains the original data and the metadata both
- copy them in your HW folder

- If you are using command line: `>> wget http://archive.ics.uci.edu /ml/machine-learning-databases/statlog/heart/heart.dat`
  - If wget is not working
    - dowload it from link (https://eternallybored.org/misc/wget/)
    - follow steps (https://stackoverflow.com/questions/29113456/wget-not-recognized-as-internal-or-external-command)

Q1 use pandas to read 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 [55]:
```
import pandas as pd

columns = ["age","sex","chest paint type","resting blood pressure", "s
           ,"fasting blood sugar > 120 mg/dl","resting electrocardiogr
           ,"exercise induced angina","oldpeak = ST depression induced
           ,"the slope of the peak exercise ST segment","number of maj
           ,"thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

df =pd.read_csv('heart.dat', delim_whitespace=True, header = None, nam
```

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

2. Let us view the size of dataset as well

- print data shape

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

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

In [56]:
```
# Code goes below
df.head()

df.tail()

df.shape

df.isnull()
```

Out[56]:

	age	sex	chest paint type	resting blood pressure	serum cholestoral in mg/dl	fasting blood sugar > 120 mg/dl	resting electrocardiographic results	maximum heart rate achieved	exercise induced angina
0	70.0	1.0	4.0	130.0	322.0	0.0	2.0	109.0	0.0
1	67.0	0.0	3.0	115.0	564.0	0.0	2.0	160.0	0.0
2	57.0	1.0	2.0	124.0	261.0	0.0	0.0	141.0	0.0
3	64.0	1.0	4.0	128.0	263.0	0.0	0.0	105.0	1.0
4	74.0	0.0	2.0	120.0	269.0	0.0	2.0	121.0	1.0
...	...	...	...	...	...	...	...	...	...
265	52.0	1.0	3.0	172.0	199.0	1.0	0.0	162.0	0.0
266	44.0	1.0	2.0	120.0	263.0	0.0	0.0	173.0	0.0
267	56.0	0.0	2.0	140.0	294.0	0.0	2.0	153.0	0.0
268	57.0	1.0	4.0	140.0	192.0	0.0	0.0	148.0	0.0
269	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0

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 [67]:

```
import seaborn as sns

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

import matplotlib.pyplot as plt
plt.show()

#describe dataframe
```

Out[67]:

	age	sex	chest paint type	resting blood pressure	serum cholestoral in mg/dl	fasting blood sugar > 120 mg/dl	electrocardiog r
count	270.000000	270.000000	270.000000	270.000000	270.000000	270.000000	270.(
mean	54.433333	0.677778	3.174074	131.344444	249.659259	0.148148	1.(

	age	sex	chest paint type	resting blood pressure	serum cholestoral in mg/dl	fasting blood sugar > 120 mg/dl	r electrocardiog r
std	9.109067	0.468195	0.950090	17.861608	51.686237	0.355906	0.9

Q4 If you go through metadata (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

A4 Replace ??? with code in the code cell below

In [79]:
```
dummy_list = df
df = pd.get_dummies(dummy_list, columns=['sex'], prefix= "dmy", prefix
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_18036/2995807873.py in <module>
      1 dummy_list = df
----> 2 df = pd.get_dummies(dummy_list, columns=['sex'], prefix= "dmy", prefix_sep='-')
      3 df.head()

~\anaconda3\lib\site-packages\pandas\core\reshape\reshape.py in get_dummies(data, prefix, prefix_sep, dummy_na, columns, sparse, drop_first, dtype)
    888             raise TypeError("Input must be a list-like for parameter `columns`")
    889         else:
--> 890             data_to_encode = data[columns]
    891
    892         # validate prefixes and separator to avoid silently dropping cols

~\anaconda3\lib\site-packages\pandas\core\frame.py in __getitem__(sel
```

KNN Model from sklearn

Q5 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 = 0.2 and random_state = 1
2. Create a KNN model using sklearn library, `Initialize n_neighbors = 3, (See the documenttaion for details)`
3. Fit the model with the train data

A5 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 .v
y = ??
# Drop 'presence' column from data frame,
df.??
# Assign df values to x
x = ??
# View shape of x and y
x.shape, y.shape
```

```
# Use stratify = y and test_size = 0.2 and random_state = 1
xtrain, xtest, ytrain, ytest = ??

# Create a KNN model using sklearn library, k=3
knn = ??

# Fit the model with the train data
```

Q6 *Analysis*

- Predict xtest and view first 25 predicitons
- Compare prediction with real ytest 25 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`

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

In [ ]:

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

# Compare prediction with real ytest 25 predictions
print(?[?:?])

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

#rescale only real value columns
realcols = ??

# For each column normalize ```df[col] as (x - mean) / standard_deviat
for col in ??:
  mean = df[col].?
  std = ??
```

Q7 Write the code to train new model using KNN classifier, k=3 (same as above)

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

In [ ]:

```
# update x
x = ?

# Train test Split
xtrain, xtest, ytrain, ytest = ??

# Model Initialization
knn = ??

# Model fitting with training data
```

```
knn.fit(?, ?)

# Now print score on test data
```

Q8 Lets analyze the difference between two modeling strategies (data normalization) Compare score with and without data normalization process and explain

A8
Write your answer replacing this line

Q9 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 bar graph

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

2. 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

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

A9 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(??, ??)
  return knn.score(??, ??)


result = [*map(lambda i:returnScore(i,??, ??, ??, ??), range(?,?))]
print(result)
plt.plot(result)


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


bestknn = KNeighborsClassifier(n_neighbors=?)

bestknn.fit(??)
bestknn.score(??)


ypred = bestknn.?
matrix = confusion_matrix(?)
print(matrix)
```

Q10 Now we will plot the confusion matrix from the above matrix information

- Please review following examples in documentation plot confusion plots
- https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html (https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html)

A10 Replace ??? with code in the code cell below

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

cm = confusion_matrix(?)
plt.figure()
plot_confusion_matrix(?)
plt.title("Confusion Matrix")
plt.xticks(range(2), ["Class 1","Class 2"], fontsize=16)
plt.yticks(range(2), ["Class 1","Class 2"], fontsize=16)
```

Q11:

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

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

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

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

print(bestknn.score(?))


PrecisionRecallDisplay.from_estimator(?)
```

Further reading

- KNN model creation (https://towardsdatascience.com/building-a-k-nearest-neighbors-k-nn-model-with-scikit-learn-51209555453a)
- Example of KNN (https://github.com/a-martyn/ISL-python/blob/master/Notebooks/ch4_classification_applied.ipynb)

In [ ]:
```

```