Open in Colab

ENGS 108 Fall 2020 Assignment 2

Due September 28, 2020 at 11:59PM on Canvas

Instructors: George Cybenko

TAs: Chase Yakaboski

Rules and Requirements

1. You are only allowed to use Python packages that are explicity imported in the assignment notebook or are standard (builtin) python libraries like random, os, sys, etc. (Standard Builtin Python libraries will have a Python.org documentation). For this assignment you may use:
   - numpy
   - pandas
   - scikit-learn
   - matplotlib
2. All code must be fit into the designated code or text blocks in the assignment notebook. They are indentified by a **TODO** qualifier.
3. For analytical questions that don't require code, type your answer cleanly in Markdown. For help, see the Google Colab Markdown Guide.

In [1]:
```
''' Import Statements '''
import numpy as np
import pandas as pd
import sklearn
import matplotlib.pyplot as plt
```

Data Loading

Upload the red and synthetic datasets to your google colab session using Google Drive. Read the following tutorial for how to get setup.

In [2]:
```
#TODO: Set your base datasets path. This is my base path, you will need to change to match yours.
dataset_base_path = '/content/drive/My Drive/git/ENGS_108_Fall_2020/datasets'
```

In [6]:
```
#-- Everything else you should not need to change.
import os
import pickle

#-- Gather paths
synth_data_path = os.path.join(dataset_base_path, 'assign_2_synth_data.pk')
red_train_path = os.path.join(dataset_base_path, 'red_train.csv')
red_valid_path = os.path.join(dataset_base_path, 'red_valid.csv')
red_test_path = os.path.join(dataset_base_path, 'red_test.csv')
synth_train_path = os.path.join(dataset_base_path, 'synth_train.csv')
synth_valid_path = os.path.join(dataset_base_path, 'synth_valid.csv')
synth_test_path = os.path.join(dataset_base_path, 'synth_test.csv')

#-- Load Synth_Data
with open(synth_data_path, 'rb') as f_:
  synth_data = pickle.load(f_)

#-- Load Red Wine Data
red_train_df = pd.read_csv(red_train_path)
red_valid_df = pd.read_csv(red_valid_path)
red_test_df  = pd.read_csv(red_test_path)
synth_train_df = pd.read_csv(synth_train_path)
synth_valid_df = pd.read_csv(synth_valid_path)
synth_test_df = pd.read_csv(synth_test_path)

#-- Data is stored in a tuple of format (X, y) and are already converted to numpy arrays.
red_train = (red_train_df.drop('quality', axis=1).to_numpy(), red_train_df['quality'].to_numpy())
red_valid = (red_valid_df.drop('quality', axis=1).to_numpy(), red_valid_df['quality'].to_numpy())
red_test = (red_test_df.drop('quality', axis=1).to_numpy(), red_test_df['quality'].to_numpy())

#-- Load in Synth train, valid, test data with tuple format (X, y)
synth_train = (synth_train_df.drop('y', axis=1).to_numpy(), synth_train_df['y'].to_numpy())
synth_valid = (synth_valid_df.drop('y', axis=1).to_numpy(), synth_valid_df['y'].to_numpy())
synth_test = (synth_test_df.drop('y', axis=1).to_numpy(), synth_test_df['y'].to_numpy())
```

Problem 1: K-Means Clustering

In this problem, you will solve a clustering task using the k-means algorithm and an associated classification task using k nearest neighbors algorithm, both of which you learned in class. The dataset for this problem is a synthetic two-dimensional dataset *synth_data*. Each entry has two features (x_1, x_2). >

> **Part 1** A reasonable first step in every machine learning task is to understand the dataset at hand. Proceed to explore this problem's dataset by addressing the following:
>
> > **(a)** Choose a suitable type of plot and visualize the training data.

In [ ]:
```
#TODO: Write your code here. Use matplotlib for visualization.
```

> > **(b)** From your plot, how many clusters, k, would you estimate are represented in the dataset?

TODO: *Type your answer in Markdown here.*

> **Part 2** Build a model.
>
> > **(a)** Using the k-Means algorithm, implement a clustering model. *Hint: Use scikit-learn's K-means library.*

In [ ]:
```
#TODO: Write your code here. Hint: Just define a model, don't train yet.
```

> > **(b)** Train the clustering model on several reasonable values of k, taking into account your visual inspection from 1b. Plot the Bayesian information criterion (BIC) and Akaike information criterion (AIC) for each value of k.

In [ ]:
```
def train(k, dataset):
  ''' Using your model above, implement a function that will train your K-means
   for different values of k on your dataset and return the trained model'''

  return model
```

In [ ]:
```
def calculateBIC(model):
  ''' Using a trained model calculate the BIC for the model '''

  return bic
```

In [ ]:
```
def calcuateAIC(model):
  ''' Using a trained model calculate the AIC for the model '''

  return aic
```

In [ ]:
```
#TODO: Choose three different values of k based on your inspection and plot the BIC and AIC scores.
```

> > **(c)** What value of k is optimal? How does it compare to your visual inspection?

TODO: *Type your answer in Markdown here.*

In [ ]:
```
#TODO: Write code and plot a graph showing the optimal value of k.
```

Problem 2: k-NN Classification

In this problem, you will utilize data deriving from the same synthetic dataset as above. This time, the data has been separated into *synth_train*, *synth_valid* and *synth_test* arrays. Furthermore, each sample now includes a class label found in the *y* column. These class labels come from the set $\{1, 2, \ldots, 31\}$. *Note: These are not the same datasets as Problem 1.*

> **Part 1** Train an implementation of the k-Nearest Neighbors algorithm on the training dataset. Note that k here refers to the number of neighbors, not clusters.

In [ ]:
```
def train(k, dataset):
  ''' Implement a function that will train a k-NN
   for different values of k on your dataset and return the trained model'''

  return model
```

> **Part 2** Report the classification accuracy of this model on the validation set for different values for k. Plot these accuracies against k and report the optimal value for k.

In [ ]:
```
#TODO: Write your code here.
```

> **Part 2** Report the classification accuracy of this model on the data in synth test.csv using the optimal value of k that you found in Part 2.

In [ ]:
```
#TODO: Write your code here.
```

Problem 3: Decision Tree Classification

In this problem you will use decision trees to classify the quality of red vinho verde wine samples based on their physicochemical properties. The dataset has been separated into *red_train, red_valid and red_test* arrays. For all of these files, the rightmost column ("quality") is the target label for each datapoint. All other columns are features.

> **Part 1** First let's explore the datasets through the following exercises. Note that we cannot plot the data in a meaningful way given that number of features exceed the physical dimensions.
>
> > **(a)** How many datapoints are in the training, validation, and testing sets?

In [ ]:
```
#TODO: Write your code here.
```

> > **(b)** How many features are available for each datapoint?

In [ ]:
```
#TODO: Write your code here.
```

> > **(c)** What are the average *alcohol* and *pH* values for *training* samples?

In [ ]:
```
#TODO: Write your code here.
```

> **Part 2** Decision Trees.
>
> > **(a)** Implement a binary decision tree model for the training data. *Hint: Try looking at the scikit-learn decision tree library.*

In [ ]:
```
def train(dataset, max_depth=None):
  ''' Implement a function that will train a decision tree model
   on your dataset and return the trained model'''

  return model
```

> > **(b)** There are a number of hyperparameters that can be tuned to improve your model, one of which is the criteria for ending the splitting process. Two common ways of terminating the splitting process are *maximum depth* of the tree or *minimum number of samples* left. Tune the *maximum depth* of the tree by reporting the accuracy of the classifier in 2a on the validation set for different settings of *maximum depth*. Plot your findings.

In [ ]:
```
#TODO: Write your code here and plot your results.
```

> > **(c)** Use the optimum setting of *maximum depth* found in 2b to report the accuracy of the classifier on the *test* dataset.

In [ ]:
```
#TODO: Write your code here.
```