Open in Colab

ENGS 108 Fall 2020 Assignment 3a

Due October 5, 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 (bultin) python libraries like random, os, sys, etc, (Standard Bultin 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 [ ]:
```
''' Import Statements '''
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import copy
import pickle
```

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 [ ]:
```
dataset_base_path = '/content/sample_data'

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

In [ ]:
```
#-- Load California Housing Data
calif_train_df = pd.read_csv(os.path.join(dataset_base_path, 'california_housing_train.csv'))
calif_test_df = pd.read_csv(os.path.join(dataset_base_path, 'california_housing_test.csv'))

#-- Load Wine Data
red_train_path = os.path.join(dataset_github_path, 'red_train.csv')
red_valid_path = os.path.join(dataset_github_path, 'red_valid.csv')
red_test_path = os.path.join(dataset_github_path, 'red_test.csv')
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)
```

Problem 1: Linear Regression

In this problem, you will be building a linear regression model to predict California Housing prices with the dataset that is provided with google colab. >

> **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)** Using the pandas DataFrame *calif_train_df* that is already loaded for you. Print a statistical summary of the data. *Hint: There is a nice pandas function to do this, research the documentation and find it.*

In [ ]:
```
#TODO: Your code goes here.
```

> > **(b)** Visualize the longitute and latitude features through a scatterplot and report on what the plot resembles. Maybe try playing with the graph opacity to yeild more sophisticated visualization.

In [ ]:
```
#TODO: Your code goes here.
```

> **Part 2** Now we will build our linear regression model.
>
> > **(a)** Using an appropriate library (such as scikit-learn), build a linear regression model on the training data so that you are using all available features to predict the median housing price of a new california district.

In [ ]:
```
from sklearn.linear_model import LinearRegression

#TODO: Seperate the data into X, y

def buildModelLinear(X, y):
  ''' Return a trained linear regression model on your dataset. '''
  return model

#TODO: Your code goes here.
```

> > **(b)** What is the largest coefficient of the linear model and which feature is associated with it?

In [ ]:
```
#TODO: Your code goes here.
```

> > **(c)** Choose an appropriate metric to calculate the accuracy/error of your linear regression model.

In [ ]:
```
from sklearn.metrics import mean_squared_error

#TODO: Your code goes here.
```

> > **(d)** Say instead of modeling every feature as linear, we think that median home price varies quadratically with median income. Build and train a model that uses this assumption and compare its accuracy/error to the pure linear model. Which model would you choose and why?

In [ ]:
```
#TODO: Your code goes here.
```

TODO: Write your explanation here.

Problem 2: Logistic Regression

In this problem, you will be building a classification model using linear functions to predict wine quality. >

> **Part 1** Let's begin this analysis by assuming we have talked to a wine sommelier about what attributes make a good wine. Our wine sommelier has told us that fixed acidity and alcohol concentration are the primary attributes that differentiate a "good wine" from a "bad wine."
>
> > **(a)** Let's assume that we can classify a "good wine" as a quality level of 7 or 8, and a "bad wine" has a quality of below 5. Choose an appropriate graph, plot fixed acidity and alcohol concentration of both "good wines" and "bad wines." Use the *red_train_df* dataset for these questions.

In [ ]:
```
#TODO: Your code goes here.
```

> > **(b)** Based on your plot in (a) could you use these features to perfectly delineate between "good" and "bad" wines? I.e. are these two features linearly separable?

TODO: Write your explanation here.

> > **(c)** Do you agree with the wine sommelier in their assessment about these two features? Explain your answer.

> **Part 2** Now use the sklearn linear classifier to build a linear function to classify wine quality based on all features of the dataset.
>
> > **(a)** Using a least squares loss function and stocastic gradient descent, fit a linear model to the training data and report accuracy on the test data. Trying running your code multiple times, what happens? Explain your observations.

In [ ]:
```
from sklearn.linear_model import SGDClassifier

def makeClassifier(dataset, loss_function='squared_loss'):
  '''Implement a function that builds your classifier and trains it on the dataset.'''
  return model

#TODO: Your code goes here.
```

TODO: Write your explanation here.

> > **(b)** Using a logistic regression loss function and stocastic gradient descent, fit a linear model to the training data and report accuracy on the test data. Do the observations you saw after running your code multiple times in (a) hold for logistic regression? Why or why not.

In [ ]:
```
#TODO: Your code goes here.
```

TODO: Write your explanation here.

> > **(c)** We learned in class that there are a couple hyperparameters that can be used in stocastic gradient descent. One very important one is learning rate. Using your logistic regression model in (b) vary the learning rate (i.e. *eta0* and set *learning_rate* argument to 'constant'), train the classifier on the training dataset, and use the validation set to choose the best model, and report the accuracy of this model on the testing dataset.

In [ ]:
```
def makeClassifierWithLearningRate(dataset, eta0, loss_function='log'):
  '''Implement a function that will build and train a logistic
  classifier with a specfied constant learning rate, and return that model.'''
  return model

#TODO: Your code goes here.
```