ECPS 211-Assignment 1

Due: January 24, 11:59 PM

This assignment helps you to set up the Python programming environment and guides you through a data exploration example.

First use the following link to install Anaconda on your computer:

https://docs.anaconda.com/anaconda/install/

Anaconda package gives you access to the latest version of Python and hundreds of its useful packages. If you have constraints on your disk space, you can go with Mini-conda, and whenever you need a specific package, install it individually.

The instruction of installing mini-conda is as follows:

https://docs.conda.io/projects/miniconda/en/latest/

It is recommended to use Jupyter Notebook as the environment to develop and run Python codes for the purposes of this course. Jupyter Notebook comes with Anaconda. If you go with Mini-conda then run

conda install **jupyter**

in your terminal to install it separately.

Submission: Prepare a report that includes the answers to the questions. If a question requires any code, it must be in the report or as a separate python file. Every graph should have axes labels. Start each question in a new page (i.e., if the answer to P1Q1 take only one line, leave rest of that page blank and start P1Q2 in the next page). Submit your report file on Canvas.

Problem 1: Loading and exploring a dataset in Python (50 points)

In this problem we load a public dataset and explore some of its basic statistics. After you install Anaconda, open a Jupyter Notebook and run this snippet (if you are using Mini-conda, you might need to first install some of these packages, Python will tell you whatever package it needs but doesn't find):

```
from matplotlib import pyplot as plt
#equivalent to:  "import matplotlib.pyplot as plt
from sklearn import datasets

iris = datasets.load_iris()
X = iris.data
y = iris.target
```

The **iris** dataset consists of four features (**X** values, features are information like sepal length, sepal width, etc.) of each flower, and a **y** value which indicates the type of flower. Three class of flowers are available which are mapped to {0,1,2}.

1) Find the shape of **X** and **y** variables (X.shape, y.shape). Explain the output. (5 points)

   In order to get the value of a certain feature (e.g. feature number 2) for a certain sample (e.g. sample number 39) you can address it as **X[39,2]**. Also, to get a vector of feature number 1 for all the samples, you can use **X[:, 1]**. To address a range, you can use **X[10:20, 1:3]**.

2) Print out **X[10:20, 1:3]**. What is dimension of the output? How does it relate to **X**? Answer the same question for **X[:40, 1:]** and **X[110:, :]** as well. (5 points)
3) Look up the *axis* hyperparameter in *np.mean()* method. Find the *mean, median* and *standard deviation* of each of the four features in X (Hint *X.mean(),X.median()* and *X.std()*). (10 points)
4) For each feature (separately), plot the histogram of the samples over all samples. (Hint: Use *plt.hist()*). (10 points)
5) For features 1 and 3 make a scatter plot of the samples, with each sample colored according to the class it belongs to (three different colors). See *plt.scatter()* . (10 points)

6) Repeat P1.5 for other combinations of features. Show one of the plots in your report (any combination of your choice except (1,3)). (10 points)

Problem 2: Logistic Regression (50 points)

1. Select part of the data that belongs to classes 0 and 1 of the iris flowers. Report how many samples are present to work with. (5 points)
2. Scale the X vectors using the *StandardScaler* class inside *preprocessing* package. Look up the function and briefly explain what it does. (5 points)
3. Split the dataset into Training and Test sections (80%-20%). Use the 8 digits number of your student ID number as the *random_state* (e.g. 00000000). Report the size of each set (test and train). (10 points)
4. Load and train the logistic regression classifier with the training data. Now you have a classifier that can map the features of each flower, to the class that flower belongs to. Print out the coefficients of the trained classifier. Explain what each of these coefficients are. (Hint: *<Classifier_name>.coef_* gives the list of coefficients). (10 points)
5. Apply the trained model on test data. Compare the output with the ground-truth (y_test). Note that the classifier has never seen the ground-truth (y_test) before. Also apply the classifier on X_train and compare the output with y_train. (Hint: Use classification_report() and accuracy_score()).(10 points)
6. Repeat steps 4-6 but this time for all three classes of flowers (do not exclude class '2' of flowers). Note that if there are more than 2 classes on the output (3 here), a single logistic regression function won't work. Now 3 nodes are present on the output (instead of 2). Then an extended version of logistic function (for more than binary classes) is used called *Softmax*. In *LogisticRegression* classifier you do not have to change any part of your code to address that, the module makes the necessary changes automatically. The important point to know here is there are three output nodes, hence three sets of coefficients.

   After you train the classifier, print out the coefficients and explain it. Use your trained classifier to predict the type of flowers in the test data and training data. Compare the predictions to the ground truth. (10 points)

```
1       from sklearn.preprocessing import StandardScaler
2       from sklearn.model_selection import train_test_split
3       from sklearn.linear_model import LogisticRegression
4       from sklearn.metrics import accuracy_score
5       from sklearn.metrics import classification_report
6       X = iris.data[iris.target < 2]
7       y = iris.target[iris.target < 2]
8       X_scaled = StandardScaler().fit_transform(X)
9       X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
                                                            test_size = 0.3,
random_state = <your id>)
10      LR_Classifier = LogisticRegression()
11      LR_Classifier.fit(X_train, y_train)
13      y_predicted = LR_Classifier.predict(X_test)
14      y_train_predicted = LR_Classifier.predict(X_train)
15      print( classification_report(y_test, y_predicted) )
16      print( accuracy_score(y_test, y_predicted))
```

Code 1. Code for training a binary logistic regression model on Iris dataset.