DSCI 503 – HW 07 Instructions

General Instructions

Create a new notebook named **`HW_07_YourLastName.ipynb`**. Download the files **`diamonds.txt`**, **`gapminder_data.txt`**, **`heart_disease.txt`**, and **`nyc.txt`** into the same directory as this notebook. Complete Parts 1 – 4 described below.

Any set of instructions you see in this document with an orange bar to the left will indicate a place where you should create a markdown cell. For each new problem, create a markdown cell that indicates the title of that problem as a level 2 header.

Any set of instructions you see with a blue bar to the left will provide instructions for creating a single code cell.

Read the instructions for each problem carefully. Each problem is worth 6 points. An additional 2 points are allocated for formatting and following general instructions.

Any time that you are asked to display a DataFrame in this assignment, you should do so without using the **`print()`** function.

Assignment Header

Create a markdown cell with a level 1 header that reads: "DSCI 503 - Homework 07". Add your name below that as a level 3 header

Import **`numpy`**, **`pandas`**, and **`matplotlib.pyplot`** using the standard aliases. Import **`LinearRegression`**, **`LogisticRegression`**, and **`train_test_split`** from **`sklearn`**.

In this assignment, you will be using scikit-learn to create linear regression and classification models.

Problem 1: NYC Restaurants Dataset

In Problem 1, you will be working with the NYC Restaurants Dataset. This dataset contains information about 168 Italian restaurants in New York City. You can find more information about this dataset here: Italian Restaurants Dataset

Load the data stored in the tab-delimited file **`nyc.txt`** into a DataFrame named **`nyc`**. Use **`head()`** to display the first 10 rows of this DataFrame.

The columns contained in this DataFrame are described below.

- **`Price`** – The average price in US dollars for a meal for two.
- **`Food`** – The Zagat customer rating of the quality of the food. On a scale from 1 – 30.
- **`Decor`** – The Zagat customer rating of the quality of the decor. On a scale from 1 – 30.
- **`Service`** – The Zagat customer rating of the quality of the service. On a scale from 1 – 30.
- **`Wait`** – The average wait time, in minutes, to be seated during dinner rush on a Friday evening.
- **`East`** – A binary variable indicating if the restaurant is East (1) or West (0) of 5th Avenue.

Our goal in this problem will be to create a linear regression model to predict the value of **`Price`** using the other five columns as features.

Perform the following steps in a single code cell:

- Create a 2D feature array named **`X1`** containing the relevant features, as well as a 1D label array named **`y1`** containing the labels. **(Note: These should be arrays, and not DataFrames or Series. See note below.)**
- Use **`train_test_split()`** to split the data into training and testing sets using an 80/20 split. Name the resulting arrays **`X_train_1`**, **`X_test_1`**, **`y_train_1`**, and **`y_test_1`**. Set **`random state=1`**.
- Print the shapes of **`X_train_1`** and **`X_test_1`**. Include text labeling the two results as shown below. Add spacing to ensure that the shape tuples are left-aligned.

```
Training Features Shape: xxxx
Test Features Shape:     xxxx
```

Note: You can extract a NumPy array from a pandas DataFrame or series object by adding **`.values`** to the end of it. For example, if **`df`** is a DataFrame object, if you run the statement **`X = df.iloc[:, some_columns].values`**, then **`X`** will be a 2D array containing the information for the selected columns.

We will now create a linear regression model that can be used to estimate the price at a similar restaurant.

Create a linear regression model named **`nyc_mod`** and then fit it to the training data. Display the intercepts and coefficients for the final model with text labels as shown below. Add spacing to ensure that the values replacing the **`xxxx`** characters are left-aligned. The intercept should appear as a single number, The coefficients should be in the form of an array and should be displayed on a single line.

```
Intercept:    xxxx
Coefficients: xxxx
```

We will now calculate the r-squared score for the model on both the training set and the test set.

Calculate and print the training and testing r-squared values for your model, rounded to four decimal places. Include the text labels explaining which value is which, as shown below. Add spacing to ensure that the scores are left-aligned.

```
Training r-Squared: xxxx
Testing r-Squared:  xxxx
```

We will now use the model to generate predictions for the restaurants in the test set.

Use your model to generate price estimates based on the feature values in the test set. Store the results in a variable named **`test_pred_1`**. Print the first 10 observed y-values for the test set, and then the first 10 predictions, rounded to 2 decimal places. Include text labels with your output as shown below. Each price array should be displayed on a single line, and the two arrays should be left-aligned.

```
Observed Prices:  xxxx
Estimated Prices: xxxx
```

Suppose that you wish to use the model to estimate the price for three new restaurants that were not included in the original dataset. Assume that the feature values for these restaurants are as follows:

Food	Decor	Service	Wait	East
22	12	20	15	0
18	19	22	34	1
25	22	18	36	0

Create a DataFrame named **`nyc_new`** that contains the feature values for these 3 restaurants. Pass this DataFrame to the **`predict()`** method of your model, storing the results in a variable named **`new_pred_1`**. Print the price predictions stored in this variable, rounded to 2 decimal places, with a message as shown below.

```
Estimated Prices: xxxx
```

Problem 2: Diamonds Dataset

In Problem 2, you will be working with the Diamonds Dataset. This dataset contains information about several thousand diamonds sold in the United States. You can find more information about this dataset, including a description of its columns, here: Diamonds Dataset.

Load the data stored in the tab-delimited file **`diamonds.txt`** into a DataFrame named **`diamonds`**. Use **`head()`** to display the first 5 rows of this DataFrame.

Our goal in this problem will be to create a linear regression model to estimate the price of a diamond based only on its carat size. You will create a model that uses the qualitative variables cut, color, and clarity in a future assignment.

We have observed in a previous assignment that there is an approximately linear relationship between the natural logarithm of **`price`** and the natural logarithm of **`carat`**.

Add two new columns to **`diamonds`**. The new columns should be named **`ln_carat`** and **`ln_price`**, and should contain the natural logarithms of the **`carat`** and **`price`** columns. Use **`head()`** to display the first 5 rows of **`diamonds`**.

We will create scatterplots to confirm that **ln_carat** and **ln_price** have an approximately linear relationship.

Create two side-by-side scatterplots. The first scatter plot should display the relationship between **carat** and **price**, while the second scatterplot should display the relationship between the transformed variables. Create the figure according to the following specifications:

- Set a figure size of `[10,4]`.
- The plots should be titled "**Relationship between Price and Carat Size**" and "**Relationship between Log-Price and Log-Carat Size**".
- The axes should be labeled "**Carat Size**", "**Price**", "**Natural Log of Carat Size**", and "**Natural Log of Price**".
- Set the point size to 20, and the alpha level to 0.2. Do not include a border on the points.

Call **`plt.tight_layout()`** and then use **`plt.show()`** to display the figure.

Our goal is to create a linear regression model to estimate values of **`ln_price`** using **`ln_carat`** as the only feature. We will now prepare the feature and label arrays.

Perform the following steps in a single code cell:

- Create a 2D feature array named **X2** containing the relevant feature, as well as a 1D label array named **y2** containing the labels. **(Note: These should be arrays, and not DataFrames or Series.)**
- Use **`train_test_split()`** to split the data into training and testing sets using an 90/10 split. Name the resulting arrays **`X_train_2`**, **`X_test_2`**, **`y_train_2`**, and **`y_test_2`**. Set **`random state=1`**.
- Print the shapes of **`X_train_2`** and **`X_test_2`**. Include text labeling the two results as shown below. Add spacing to ensure that the shape tuples are left-aligned.

```
Training Features Shape: xxxx
Test Features Shape:     xxxx
```

We will now create a linear regression model that can be used to estimate the natural log of the price of a diamond.

Create a linear regression model named **`dia_mod`** and then fit it to the training data. Display the intercepts and coefficients for the final model with text labels as shown below. Add spacing to ensure that the values replacing the **`xxxx`** characters are left-aligned. The intercept should appear as a single number, and the coefficients should be in the form of an array and should be displayed on a single line.

```
Intercept:    xxxx
Coefficients: xxxx
```

We will now calculate the r-squared score for the model on both the training set and the test set.

Calculate and print the training and testing r-squared values for your model, rounded to four decimal places. Include the text labels explaining which value is which, as shown below. Add spacing to ensure that the scores are left-aligned.

```
Training r-Squared: xxxx
Testing r-Squared:  xxxx
```

We will now use the model to generate price estimates for the diamonds in the test set.

Use your model to generate estimates for the natural log of prices for diamonds in the test set. Store the results in a variable named **`test_pred_2`**. Print the first 10 observed prices for the test set, and then the first 10 estimated prices, rounded to the nearest whole number. Include text labels with your output as shown below. Each price array should be displayed on a single line, and the two arrays should be left-aligned.

```
Observed Prices:  xxxx
Estimated Prices: xxxx
```

Note that you are asked to display actual and estimated **prices**, and not the natural logarithms of these values. The arrays **`y_test_2`** and **`test_pred_2`** will contain natural logarithms of prices.

We will now use the model to estimate the price of diamonds with the following carat sizes: **0.5, 1.0, 1.5, 2.0, 2.5, 3.0**

Create a 2D column array named **`diamonds_new`** to store the natural logarithm of the carat sizes show above. Pass this array to the **`predict()`** method of your model, storing the results in a variable named **`new_pred_2`**. Print the predicted prices for diamonds with these carat sizes, rounded to the nearest dollar. Keep in mind that the model predicts the logarithms of prices, so you will have to transform the values stored in **`new_pred_3`** in order to get predicted prices. Display your results with a message as shown below.

```
Estimated Prices: xxxx
```

Problem 3: Heart Disease Dataset

In Problem 3, you will be working with the Statlog Heart Disease Dataset. This dataset contains medical information about 270 individuals, including a column that indicates whether or not the individual has heart disease. You can find more about this dataset, including descriptions of its columns, here: Statlog Heart Dataset.

Load the data stored in the tab-delimited file **`heart_disease.txt`** into a DataFrame named **hd**. Use **`head()`** to display the first 10 rows of this DataFrame.

Our goal in this problem will be to create a logistic regression model to predict the label **`heart_disease`** using the other columns as features. Note that a value of 1 in the **`heart_disease`** column indicates an absence of heart disease, while a value of 2 indicates the presence of heart disease.

Perform the following steps in a single code cell:

- Create a 2D feature array named **X3** containing the relevant features, as well as a 1D label array named **y3** containing the labels. **(Note: These should be arrays, and not DataFrames or Series.)**
- Use **`train_test_split()`** to split the data into training and testing sets using an 80/20 split. Name the resulting arrays **`X_train_3`**, **`X_test_3`**, **`y_train_3`**, and **`y_test_3`**. Set **`random state=1`**. **Use stratified sampling**.
- Print the shapes of **`X_train_3`** and **`X_test_3`**. Include text labeling the two results as shown below. Add spacing to ensure that the shape tuples are left-aligned.

```
Training Features Shape: xxxx
Test Features Shape:     xxxx
```

We will now create a logistic regression model that can be used to estimate the probability that an individual has heart disease based on the feature values.

Create a logistic regression model named **hd_mod** with **solver='lbfgs'** and **penalty='none'**. Then fit the model to the training data. If you get a warning message stating that the model failed to converge, then increase the **max_iter** parameter until it does converge.

Display the intercepts and coefficients for the final model with text labels as shown below. Note that the coefficients array will not fit on a single line, so please display it BENEATH the line containing the "`Coefficients:`" label.

```
Intercept:    xxxx
Coefficients:
xxxx
```

We will now calculate the accuracy score for the model on both the training set and the test set.

Calculate and print the training and testing accuracy scores for your model, rounded to four decimal places. Include the text labels explaining which value is which, as shown below. Add spacing to ensure that the scores are left-aligned.

```
Training Accuracy: xxxx
Testing Accuracy:  xxxx
```

We will now use the model to generate predictions regarding the presence or absence of heart disease for individuals in the test set.

Use your model to generate label predictions for observations in the test set. Store the results in a variable named **test_pred_3**. Print the first 20 observed labels for the test set, and then the first 20 predicted labels. Include text labels with your output as shown below. Each label array should be displayed on a single line, and the two arrays should be left-aligned.

```
Observed Labels:  xxxx
Predicted Labels: xxxx
```

As a final step, we will use our model to estimate the probability that each individual in the test set has heart disease.

Use the **predict_proba()** method of your model to estimate probabilities of being in each of the two classes for each individual in the test set. This function returns a 2D array. Display the predicted probabilities for the first 10 observations in the test set **as a DataFrame** with the columns named according to the labels that they represent (**1** and **2**).

Problem 4: Gapminder Dataset

In Problem 4, you will be working with the Gapminder Dataset. This dataset contains socioeconomic information for 184 countries for each year since 1800. You can find more information about this dataset in several places throughout the course, including in Lesson 18.

Load the data stored in the tab-delimited file **gapminder_data.txt** into a DataFrame named **gm**. Filter the DataFrame, keeping only the 2018 results, storing the results in a DataFrame named **gm18**. Use **head()** to display the first 10 rows of this DataFrame.

Note that the DataFrame **gm18** should only have 184 rows. You might want to check that before continuing.

Our goal in this problem will be to create a logistic regression model to predict the label **continent** using the columns **life_exp**, **gdp_per_cap**, and **gini** as features. **Note that only these three columns should be used as features.**

Perform the following steps in a single code cell:

- Create a 2D feature array named **X4** containing the relevant features, as well as a 1D label array named **y4** containing the labels. **(Note: These should be arrays, and not DataFrames or Series.)**
- Use **`train_test_split()`** to split the data into training and testing sets using an 70/30 split. Name the resulting arrays **`X_train_4`**, **`X_test_4`**, **`y_train_4`**, and **`y_test_4`**. Set **`random state=1`**. **Use stratified sampling**.
- Print the shapes of **`X_train_4`** and **`X_test_4`**. Include text labeling the two results as shown below. Add spacing to ensure that the shape tuples are left-aligned.

```
Training Features Shape: xxxx
Test Features Shape:     xxxx
```

We will now create a logistic regression model that can be used to estimate the probability that a country is in any particular continental region based on the feature values.

Create a logistic regression model named **`gm_mod`** with the following parameter vakues: **`solver='lbfgs'`**, **`penalty='none'`**, and **`multi_class='multinomial'`**. Then fit the model to the training data. If you get a warning message stating that the model failed to converge, then increase the **`max_iter`** parameter until it does converge.

Display the intercepts and coefficients for the final model with text labels as shown below. Note that the coefficients array will not fit on a single line, so please display it BENEATH the line containing the "`Coefficients:`" label.

```
Intercept:    xxxx
Coefficients:
xxxx
```

We will now calculate the accuracy score for the model on both the training set and the test set.

Calculate and print the training and testing accuracy scores for your model, rounded to four decimal places. Include the text labels explaining which value is which, as shown below. Add spacing to ensure that the scores are left-aligned.

```
Training Accuracy: xxxx
Testing Accuracy:  xxxx
```

We will now use the model to generate predictions the continental region of countries in the test set.

Use your model to generate label predictions for observations in the test set. Store the results in a variable named **`test_pred_4`**. Print the first 8 observed labels for the test set, and then the first 8 predicted labels. Include text labels with your output as shown below. Each label array should be displayed on a single line, and the two arrays should be left-aligned.

```
Observed Labels:  xxxx
Predicted Labels: xxxx
```

We will use our model to estimate the probability of countries in the test set being in each of the four continents based on their feature values.

Use the **`predict_proba()`** method of your model to estimate probabilities of being in each of the four classes for each individual in the test set. This function returns a 2D array. Display the predicted probabilities for the first 10 observations in the test set **as a DataFrame** with the columns named according to the associated labels: **`'africa'`**, **`'americas'`**, **`'asia'`**, and **`'europe'`**.

Recall that **gdp_per_cap** is a measure of average wealth and **gini** is a measure of wealth inequality. Suppose that we wish to explore the impact that these features have on the predictions generated by our model. To that end, we will consider six fictitious countries with the following feature values:

life_exp	gdp_per_cap	gini
75	5000	30
75	5000	40
75	5000	50
75	20000	30
75	20000	40
75	20000	50

Note that **life_exp** is the same for all six countries. The first three countries are relatively poor, with a per capita GDP of $5000, while the last three countries are relatively wealthy, with a per capita GDP of $20,000 (the per capita GDP of the world as a whole is about $15,000). Within each wealth group, we have three different **gini** scores ranging from 30 (relatively low inequality) to 50 (relatively high inequality).

Create a DataFrame named **gm_new** that contains the information shown in the table above. Pass this DataFrame to the **predict_proba()** method of your model. Round the estimated probabilities to 3 decimal places, and display the results in the form of a DataFrame. The title of the columns in the DataFrame should be the same as the set of possible labels: **'africa'**, **'americas'**, **'asia'**, and **'europe'**.

Add a markdown cell containing the message and bullet-pointed list show below. Fill in the blanks with the appropriate answers.

According to our model:

- Country 0 is most likely in ____.
- Country 1 is most likely in ____.
- Country 2 is most likely in ____.
- Country 3 is most likely in ____.
- Country 4 is most likely in ____.
- Country 5 is most likely in ____.

Submission Instructions

When you are done, click **Kernel > Restart and Run All**. If any cell produces an error, then manually run every cell after that one, in order. Save your notebook, and then export the notebook as an HTML file. Upload the HTML file to Canvas and upload the IPYNB file to CoCalc, placing it in the **Homework/HW 07** folder.