DSCI 503 – HW 06 Instructions

General Instructions

Create a new notebook named **`HW_06_YourLastName.ipynb`**. Download the files **`ames_housing.txt`**, **`batting.csv`**, **`iris.txt`**, and **`titanic.txt`** into the same directory as this notebook. Complete Problems 1 – 8 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 06". Add your name below that as a level 3 header

Import the following packages using the standard aliases: **`numpy`**, **`pandas`**, and **`matplotlib.pyplot`**. No other packages should be used in this project.

In this assignment, you will be using Pandas to perform grouping operations on DataFrames. We will be working with the following four datasets:

- **Iris Dataset.** This dataset contains information about 150 iris flowers, split into three different species (setosa, versicolor, and virginica). Length and width measurements of sepals and petals are provided for each flower. You can find more information about this dataset here: Iris Dataset
- **Ames Housing Dataset.** This dataset contains information about 2930 houses sold in Ames, Iowa between 2006 and 2010. For each such house, 82 pieces of information are provided, including the sale price. You can find more information about this dataset here: Ames Housing Dataset
- **Titanic Dataset.** This dataset contains information about the 887 passengers on the first and only voyage of the HMS Titanic. We are provided with 10 pieces of information for each passenger, including their sex, their passenger class, and whether or not they survived. You can find more information about this dataset here: Titanic Dataset.
- **Lahman Batting Dataset.** This dataset contains batting data for Major League Baseball Players. It contains 105,861 records, with each record containing info about a single player, on a specific team, during a specific year. You can find more information about this dataset here: Lahman Batting Table.

Load the four data sets into DataFrames named **`iris`**, **`ames`**, **`titanic`**, and **`batting`**. This can be done using the following code:

```
iris = pd.read_csv('iris.txt', sep='\t')
ames = pd.read_csv('ames_housing.txt', sep='\t')
titanic = pd.read_csv('titanic.txt', sep='\t')
batting = pd.read_csv('batting.csv', sep=',')
```

Problem 1: Iris Dataset

In this problem, we will use grouping operations to determine the average sepal length, sepal width, petal length, and petal width for each of the three iris species.

Use the **`head()`** method to display the first 8 rows of the **`iris`** DataFrame.

We will now display some descriptive statistics for each of the columns in the iris DataFrame.

Use the **`describe()`** method to display information about the statistical distribution of each of the columns in **`iris`**.

Next, we will apply grouping operations to calculate column means for each of the three species.

Use **`groupby()`** and **`mean()`** to group the iris dataset by species, calculating the mean of each column for each of the three species. Store the results in a DataFrame named **`iris_means`**. Display the new DataFrame.

Next, we use a bar chart to graphically represent the information calculated above.

Create a list named **`iris_colors`** containing three named colors. Also create a list named **`iris_columns`** containing the following strings: `'Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width'`

Use a loop to create a figure containing four bar charts arranged in a 2x2 grid.

- Set the figure size to `[8,6]`.
- Each bar chart should contain three bars whose heights represent information from one of the four columns from the **`iris_means`** DataFrame.
- The bars should be labeled according to the iris species they represent. You can extract this information from **`iris_means`** using **`iris_means.index`**.
- The bars should have a black border. The **`color`** parameter should be set to the **`iris_colors`** list.
- Each subplot should have one of the following titles: "Mean Sepal Length", "Mean Sepal Width", "Mean Petal Length", or "Mean Petal Width". You can use the **`iris_columns`** list to help form these titles.
- Call **`plt.tight_layout()`** and display the figure using **`plt.plot().`**

Problem 2: Ames Housing - Neighborhoods

In this problem, we will use grouping operations to determine the mean sale price for houses in each of the 28 neighborhoods in the Ames Housing dataset.

Use the **`head()`** method to display the first 8 rows of the **`ames`** DataFrame.

This DataFrame has more columns than Pandas is able to display on the screen at one time. We can use the **`columns`** attribute of a DataFrame to view the names of the columns it contains.

Print **`ames.columns.values`**.

We will now calculate the average sale price for houses in for each neighborhood.

Create a DataFrame named **`ames_nbhd`** as follows:

1. Select the **`Neighborhood`** and **`SalePrice`** columns from **`ames`**.
2. Group by **`Neighborhood`**, and then calculate the mean of **`SalePrice`** for each neighborhood.
3. Sort the rows of the DataFrame according to the average sale price, in increasing order.

Use **`head()`** to display the first 5 rows of **`ames_nbhd`**.

A horizontal bar chart is one in which the bars are displayed horizontally rather than vertically. Such a chart can be created using **`plt.barh()`**. The syntax is identical to that of **`plt.bar()`**.

Create a horizontal bar chart displaying the mean sale price for houses in each of the neighborhoods. The labels for the bars should correspond to the distinct neighborhoods, and can be set using the index of the **`ames_nbhd`** DataFrame. Set the figure size to be `[6,10]`, and choose a single named color to use for the bars. The x-axis should be labeled "Mean Sale Price" and the y-axis should be labeled "Neighborhood". The title of the chart should be "Mean Sale Price by Neighborhood".

The length of the bars in this chart should decrease as you move from top to bottom.

Problem 3: Ames Housing – House Styles

In this problem, we will use grouping operations to determine the number of houses sold for each of the eight house styles provided in the Ames Housing dataset.

Create a DataFrame named **`ames_style`** as follows:

1. Select the **`House Style`** and **`SalePrice`** columns from **`ames`**.
2. Group by **`House Style`**, and then calculate the count of **`SalePrice`** for each group.
3. Sort the rows of the DataFrame according to the count column, in increasing order.

By default, the count column in the resulting DataFrame will be named **`SalePrice`**. Change the name of this column to **`Count`** by using the following code:

```
ames_style.rename(columns={'SalePrice':'Count'}, inplace=True)
```

Display **`ames_style`**.

We will graphically represent the information above using a bar chart.

Create a standard (vertical) bar chart displaying the number of houses sold for each of the house styles. The labels for the bars should correspond to the different house styles, and can be set using the index of the **`ames_style`** DataFrame. Set the figure size to be `[6,4]`, and choose a single named color to use for the bars. The x-axis should be labeled "House Style" and the y-axis should be labeled "Count". The title of the chart should be "Houses Sold by House Style".

The height of the bars in this chart should increase as you move from left to right.

Problem 4: Titanic Dataset – Survival Rates

In this problem, we will determine how passenger class and sex affected the survival rates of passengers on the Titanic.

Use the **`head()`** method to display the first 8 rows of the **`titanic`** DataFrame.

The **`Survived`** column in this DataFrame indicates whether or not a particular passenger survived, with a value of 1 indicating survival and a value of 0 indicating death. Because of the 0/1 encoding, the average of the values in this column will be equal to the proportion of passengers who survived.

Calculate the average of the **`Survived`** column in the **`titanic`** DataFrame. You may use either **`np.mean()`** or the **`mean()`** DataFrame method. Print the results with a message in the format shown below. Round the numerical value to four decimal places.

```
Proportion of Passengers who survived: xxxx
```

Passengers aboard the Titanic were assigned one of three classes: First Class, Second Class, and Third Class. We will now use grouping to determine the survival rates within each of the six groups determined by the three classes and the two sexes.

Create a DataFrame named **`surv_rates`** by selecting the **`Pclass`**, **`Sex`**, and **`Survived`** columns from **`titanic`**, grouping the results by both **`Pclass`** and **`Sex`**, and then calculating the mean of each group. Display this DataFrame.

Problem 5: Titanic Dataset – Sex Distribution by Class and Outcome

In this problem, we will determine the distribution of males and females for each of the three passenger classes, and for each of the two survival outcomes. To accomplish this task, we will first need to add two new columns to the **titanic** DataFrame.

Add two new columns to **`titanic`**. The names of the new columns should be **`Female`** and **`Male`**. The values in the **`Female`** column should be equal to 1 for any record corresponding to a female, and 0 for any record corresponding to a male. The values in the **`Male`** column should be equal to 1 for any record corresponding to a male, and 0 for any record corresponding to a female.

This task can be accomplished in a number of ways, but I would recommend using **`np.where()`** along with the **`Sex`** column.

Use the **`head()`** method to display the first 8 rows of the new **`titanic`** DataFrame.

Create a DataFrame named **`sex_dist_by_class`** by selecting the **`Pclass`**, **`Female`**, and **`Male`** columns from **`titanic`**, grouping the results by **`Pclass`**, and then summing the remaining two columns within each group. Display this DataFrame.

Create a DataFrame named **`sex_dist_by_outcome`** by selecting the **`Survived`**, **`Female`**, and **`Male`** columns from **`titanic`**, grouping the results by **`Survived`**, and then summing the remaining two columns within each group. Display this DataFrame.

Problem 6: Batting Data – Home Runs

In this problem, we will determine which Major League Baseball teams have had the most home runs in a single season, and which teams have had the highest lifetime average number of home runs per season.

Use the **`head()`** method to display the first 5 rows of the **`batting`** DataFrame.

We will now print the names of the columns in this DataFrame.

Print **`batting.columns.values`**.

We will now determine which MLB teams have the most home runs in a single season.

Create a DataFrame named **`total_hr`** by selecting the **`teamID`**, **`yearID`**, and **`HR`** columns from **`batting`**, grouping the results by both **`teamID`** and **`yearID`**, and then summing the remaining column within each group. Sort the values by the total number of home runs, in **descending** order.

When you call **`groupby()`** in this example, please set the optional **`as_index`** parameter to **`False.`** An explanation of this request is provided below.

Use **`head()`** to display the first 10 rows of this DataFrame.

When we perform grouping operations, the default behavior is for the columns that we are grouping on to be converted to indices, rather than actual columns. Setting **`as_index=False`** changes this behavior, and causes the columns on which we grouped to be added as regular columns. In the next cell, we will perform additional grouping steps that will require **`teamID`** and **`yearID`** to exist as regular columns.

We will now determine which MLB teams have the highest average number of home runs per season.

Create a DataFrame named **`avg_hr`** by selecting the **`teamID`** and **HR** columns from the **`total_hr`** DataFrame, grouping the results by **`teamID`**, and then averaging the remaining column within each group. Sort the values by the average number of home runs per season, in **descending** order. Use **`head()`** to display the first 10 rows of this DataFrame.

Problem 7: Batting Data – Batting Averages

In this problem, we will determine which Major League Baseball teams have had the highest batting average during a single season.

A team's batting average for a season is defined to be the number of hits obtained by the team during the season divided by by the number of at-bats that the team has had during the season. The number of hits for an individual player in the Lahman dataset is stored in the **H** column and the number of at-bats is stored in the **AB** column.

Create a DataFrame named **`batting_avg`** by selecting the **`teamID`**, **`yearID`**, **H**, and **AB** columns from the **`batting`** DataFrame, grouping the results by **`teamID`** and **`yearID`**, and calculating the sum of the remaining two columns within each group.

Then add a new column named **BA** to the **`batting_avg`** DataFrame. This new column should be calculated by dividing the values in the **H** column by the values in the **AB** column. **This should be done without using a loop.**

Finally, sort the DataFrame by **BA**, in descending order. Use **`head()`** to display the first 10 rows of this DataFrame.

Problem 8: Batting Data – Cardinals vs. Cubs

In this problem, we will calculate the batting average and the number of home runs for the St. Louis Cardinals and the Chicago Cubs during every season since 1900. We will then compare the results.

Note that within the Lahman Dataset, the **`teamID`** for the St. Louis Cardinals is listed as **`'SLN'`** and the **`teamID`** for the Chicago Cubs is listed as **`'CHN'`**.

Create a DataFrame named **`stl_batting`** that contains the number of hits, at-bats, and home runs for every season of the St. Louis Cardinals since 1900. You can do this as follows:

- Use **`loc`** to filter the **`batting`** DataFrame, keeping only the records for which **`teamID`** is equal to **`'SLN'`** and for which the **`yearID`** is greater than or equal to 1900.
- Use **`loc`** to select the **`yearID`**, **H**, **AB**, and **HR** columns. You can do this at the same time as when you are selecting the rows, or with a second use of **`loc`**.
- Group the results by **`yearID`**, and then calculate grouped sums for the remaining columns.

Add a new column named **BA** to the **`stl_batting`** DataFrame. This new column should be calculated by dividing the values in the **H** column by the values in the **AB** column. **This should be done without using a loop.**

Create a DataFrame named **`chi_batting`** that contains the number of hits, at-bats, home runs, and batting average for every season of the Chicago Cubs since 1900. The process is the same as above, except that you will use the **`teamID`** of **`'CHN'`** to select records corresponding to the Cubs.

If you want to *temporarily* display **`stl_batting`** and **`chi_batting`** to check your work, you can check that in 1900 the batting average for the Cardinals was 0.291163 and the batting average for the Cubs was 0.260037. Please remove the code for displaying these DataFrames prior to submitting your work.

Create a figure with two side-by-side line plots. Both plots should display two lines. The plot on the left should display the batting averages for the two teams for each year since 1900, and the one on the right should display the total number of home runs for the two teams for each year since 1900. Create the figure according to the following specifications:

- Set the figure size to `[12,4]`.
- Select a single named color to use for the Cardinals in both plots. Select a different named color to use for the Cubs in both plots.
- The x-axis should be labeled "Year", and should show tick marks corresponding to years since 1900.
- The y-axes of the two plots should be labeled "Batting Average" and "Home Runs".
- The titles should be "Batting Average By Year" and "Home Runs by Year".
- Both plots should include a legend with two items: "Cardinals" and "Cubs".

Display the figure using **`plt.show()`**.

Use **`np.mean()`** along with an array comparison between two columns of **`stl_batting`** and **`chi_batting`** to determine the proportion of years since 1900 in which the Cardinals had a higher batting average than the cubs. Display the result rounded to four decimal places.

Use **`np.mean()`** along with an array comparison between two columns of **`stl_batting`** and **`chi_batting`** to determine the proportion of years since 1900 in which the Cardinals had more home runs than the cubs. Display the result rounded to four decimal places.

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 06** folder.