

PA4

- Due at 3:30 PM on Nov 29, 2023
- The 2 .py files (1 for Part 1 and 1 for Part 2) containing the code should be submitted on Blackboard in the Assignments/Projects section
- If there is any ambiguity or contradiction or confusion that you may have while working on the assignment, feel free to ask questions and clarifications in class or on Teams or discuss individually with Danish.
- Please do not post your code or code-snippet on Teams
- This is an individual assignment and focuses on data frames, web scraping, data transformation, data visualization, data analysis (including regression analysis) and natural language processing using Python.

PART 1

REGRESSION

- Create one .py file for PART 1:
 - *reg.py*
- Download the pa4_reg_data.txt from Teams for Part 1 of this assignment and read the data into a pandas dataframe. The delimiter/separator of the columns in pa4_reg_data.txt is tab ('\t')
- The data in the text file mentioned above has been created using data from https://raw.githubusercontent.com/rashida048/Datasets/master/home_data.csv

PART 1(REGRESSION): ADDING COLUMNS

- Add the following column in the data frame:
 - **age**
 - Holds the values of how old a house/real estate property is. It is calculated by subtracting the values in the columns `yr_built` from 2015. For instance if a house's `yr_built` value is 2005, its age value should be 10.

PART 2 (REGRESSION): DESCRIPTIVE STATISTICS

- Display the descriptive statistics of sqft_living, price, grade, floors, age, yr_built and view on the console as shown below:

	sqft_living	price	grade	floors	age	yr_built	view
count	6972.000000	6.972000e+03	6972.000000	6972.000000	6972.000000	6972.000000	6972.000000
mean	2041.913368	5.421775e+05	7.605995	1.477410	43.838497	1971.161503	0.235800
std	893.619914	3.629664e+05	1.167795	0.538463	29.144396	29.144396	0.764319
min	390.000000	7.500000e+04	3.000000	1.000000	0.000000	1900.000000	0.000000
25%	1400.000000	3.249500e+05	7.000000	1.000000	18.000000	1952.000000	0.000000
50%	1880.000000	4.508000e+05	7.000000	1.000000	41.000000	1974.000000	0.000000
75%	2500.000000	6.439625e+05	8.000000	2.000000	63.000000	1997.000000	0.000000
max	8000.000000	5.350000e+06	13.000000	3.500000	115.000000	2015.000000	4.000000

PART 1 (REGRESSION): ASSOCIATION BETWEEN PRICE AND OTHER VARIABLES

- Using the `ols` function from `statsmodels.formula.api` package, run the following regression equation:
 - `price ~ sqft_living + grade + floors + age + view`
- Add a column **price_pred** in the data frame. The `price_pred` column will hold the predicted price values that will be calculated using the coefficient estimates of variables and in the above-mentioned regression equation and intercept.

PART 1 (REGRESSION):

ASSOCIATION BETWEEN PRICE AND OTHER VARIABLES

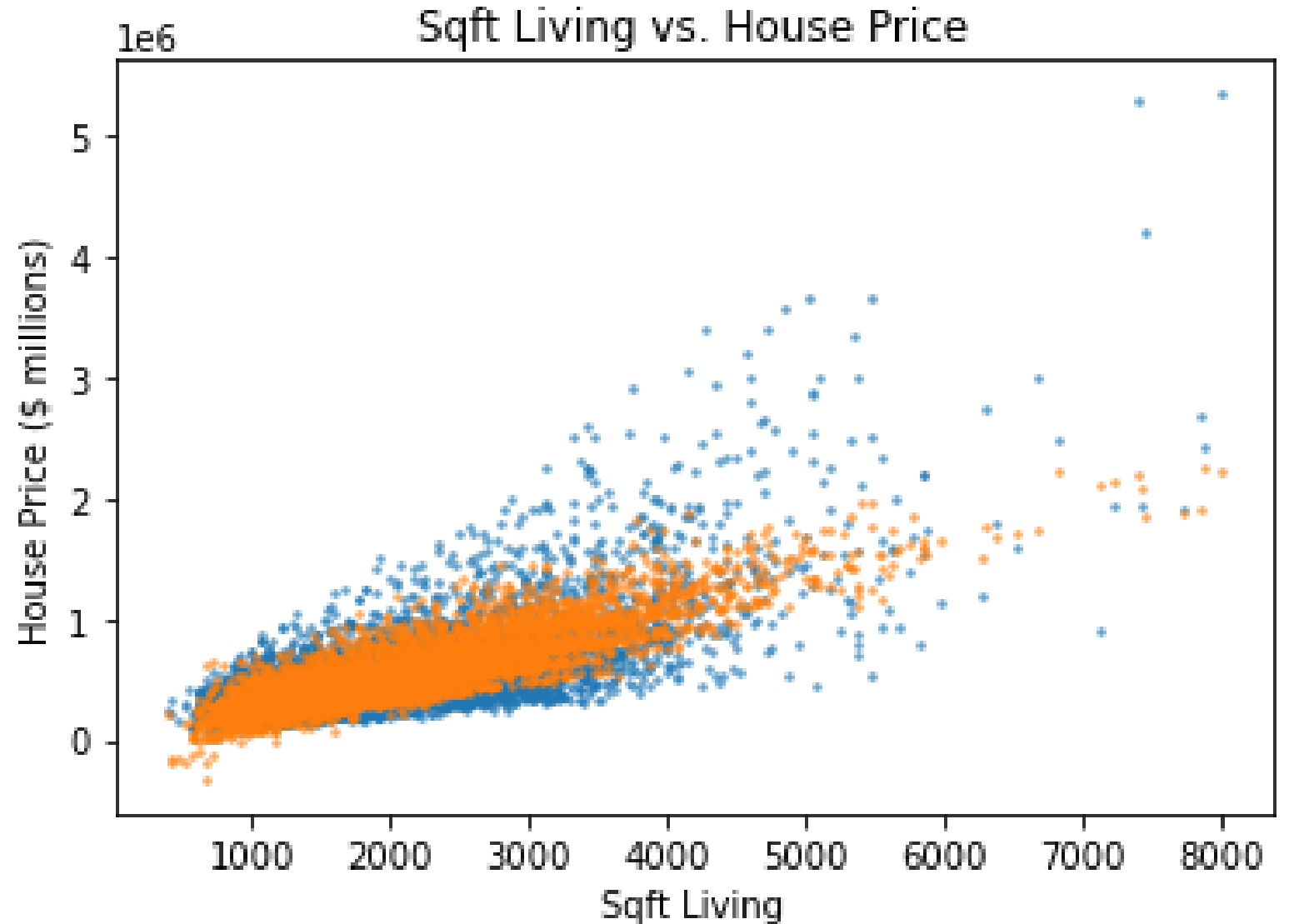
- Print the results of the above-mentioned regression model on the console as shown here

```
OLS Regression Results
=====
Dep. Variable:          price      R-squared:                0.611
Model:                  OLS        Adj. R-squared:           0.611
Method:                 Least Squares   F-statistic:             2190.
Date:                   Tue, 14 Nov 2023   Prob (F-statistic):       0.00
Time:                   21:58:11         Log-Likelihood:          -95855.
No. Observations:      6972           AIC:                    1.917e+05
Df Residuals:          6966           BIC:                    1.918e+05
Df Model:               5
Covariance Type:       nonrobust
=====
               coef      std err          t      P>|t|      [0.025      0.975]
-----
Intercept    -1.063e+06    2.65e+04    -40.164    0.000    -1.11e+06    -1.01e+06
sqft_living    150.4358         4.751     31.662    0.000     141.122     159.750
grade         1.403e+05    3913.668     35.854    0.000     1.33e+05     1.48e+05
floors         3.569e+04    6084.170      5.866    0.000     2.38e+04     4.76e+04
age           3639.1631     115.243     31.578    0.000     3413.252     3865.074
view          7.826e+04    3763.000     20.796    0.000     7.09e+04     8.56e+04
=====
Omnibus:            4426.268    Durbin-Watson:           1.995
Prob(Omnibus):      0.000    Jarque-Bera (JB):        139860.940
Skew:               2.542    Prob(JB):                 0.00
Kurtosis:           24.345    Cond. No.                 2.20e+04
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.2e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

PART 1 (REGRESSION): ASSOCIATION BETWEEN PRICE AND OTHER VARIABLES

- Create a scatter plot using the `sqft_living` and `price` columns and then draw the regression curve/line through the scatter plot using the `sqft_living` and `price` columns as shown here.



PART 1 (REGRESSION): ASSOCIATION BETWEEN PRICE AND OTHER VARIABLES

- Calculate the mean absolute error using price and price_pred variables and print it on the console as shown below:

Mean absolute error: 149029.55010674353


- Calculate the root mean squared error using price and price_pred variables and print it on the console as shown below:

Root mean squared error: 226303.43815400597

PART 2

NATURAL LANGUAGE PROCESSING

- Create one .py file for PART 2:
 - *nlp.py*



Beauty	5-core (198,502 reviews)
Apps for Android	5-core (752,937 reviews)
Office Products	5-core (53,258 reviews)
Pet Supplies	5-core (157,836 reviews)
Automotive	5-core (20,473 reviews)

- Download 5-core Office Products review website: <https://cseweb.ucsd.edu/~jmcauley/datasets/amazon/links.html>
- Read the data from the .gz file *reviews_Office_Products_5.json.gz* into a Pandas data frame.

References:

Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering

R. He, J. McAuley

WWW, 2016

Image-based recommendations on styles and substitutes

J. McAuley, C. Targett, J. Shi, A. van den Hengel

SIGIR, 2015

PART 2 (NLP): ADDING COLUMNS

- Add the following columns in the data frame:
 - **month**
 - Holds the values of the months extracted from the dates in the column reviewTime.
 - **year**
 - Holds the values of the years extracted from the dates in the column reviewTime.
 - **word_count**
 - Holds the values of the number of words of the textual comments in the column reviewText. The number of words for each comment are calculated after all the characters except alphabets and blank spaces have been removed from the comments. For this, utilize regular expressions to first remove all characters except alphabets and white-spaces and then replace every group of contiguous white-spaces with just one white-space each. Next, remove the leading and trailing white-spaces. Finally utilize the number of white-spaces to calculate the word count. There may be some comments with no words. **Such comments will have word count of 0.**

PART 2 (NLP): ADDING COLUMNS

- Add the following columns in the data frame:
 - **sentiment_score**
 - Holds the values of the sentiment scores (polarities) of the textual comments in the column reviewText. The sentiment scores are calculated by using the textblob package (as done in the class).
 - **sentiment_type**
 - Holds the values of the sentiment types (positive, neutral and negative). These types are calculated using the values of the sentiment scores in the column in sentiment_score and the following criteria:
 - If sentiment score > 0 then sentiment type is positive
 - Else if sentiment score $== 0$ then sentiment type is neutral
 - Else sentiment type is negative (if sentiment score < 0)

PART 2 (NLP): DESCRIPTIVE STATISTICS

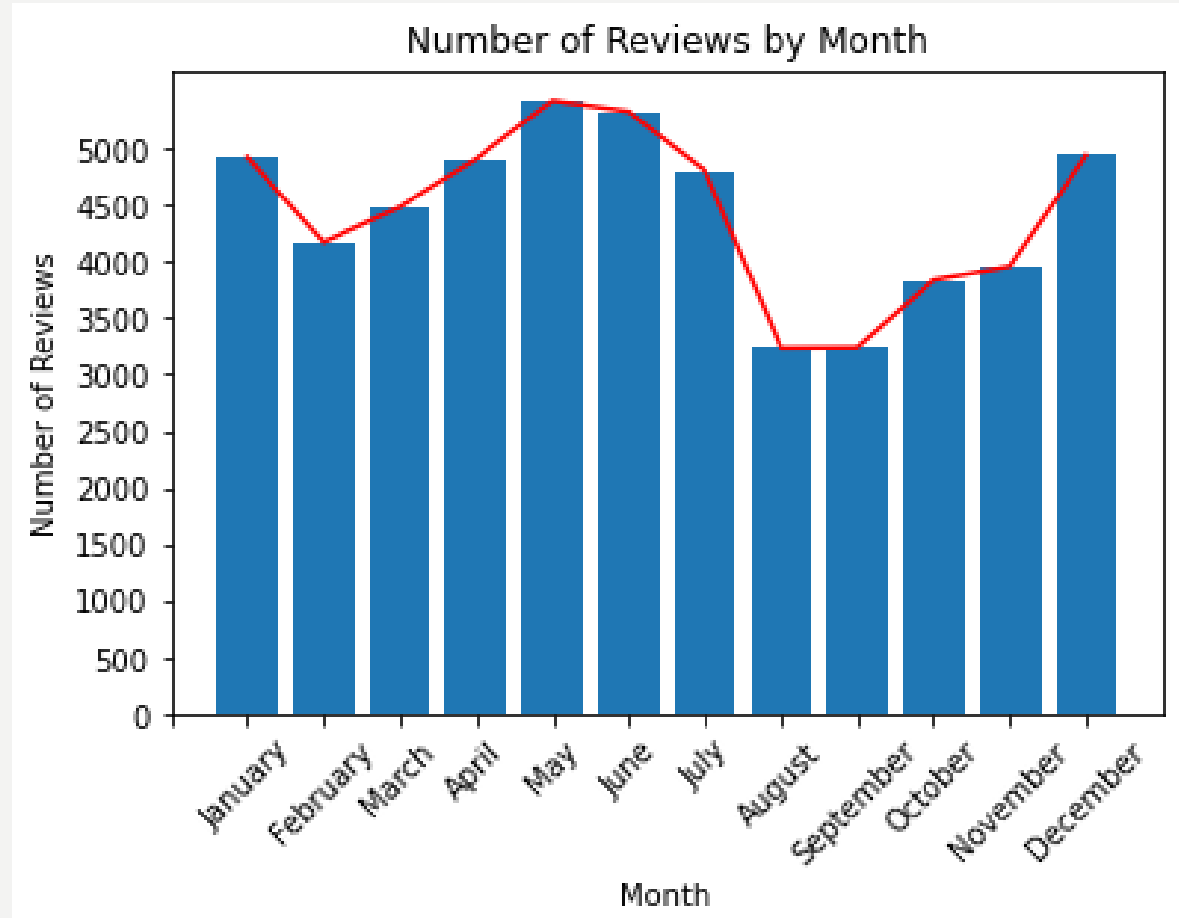
- Display the descriptive statistics of overall (numeric rating), word_count and sentiment_score on the console as shown below:

	overall	word_count	sentiment_score
count	53258.000000	53258.000000	53258.000000
mean	4.345957	143.758027	0.211688
std	0.930250	158.112360	0.162684
min	1.000000	0.000000	-1.000000
25%	4.000000	55.000000	0.112963
50%	5.000000	100.000000	0.194231
75%	5.000000	175.000000	0.293103
max	5.000000	5395.000000	1.000000

PART 2 (NLP): REVIEW VOLUME BY MONTH

- Plot a combination of bar and line chart depicting the total number of reviews for each of the twelve months across all the years along with printing the number of reviews for each month on the console as shown below. For this, you can begin by creating a data frame with two columns, one storing the values of month numbers (1 to 12) and the other storing the total number of reviews for each month (hint: you can import calendar package to create names for the corresponding month numbers).

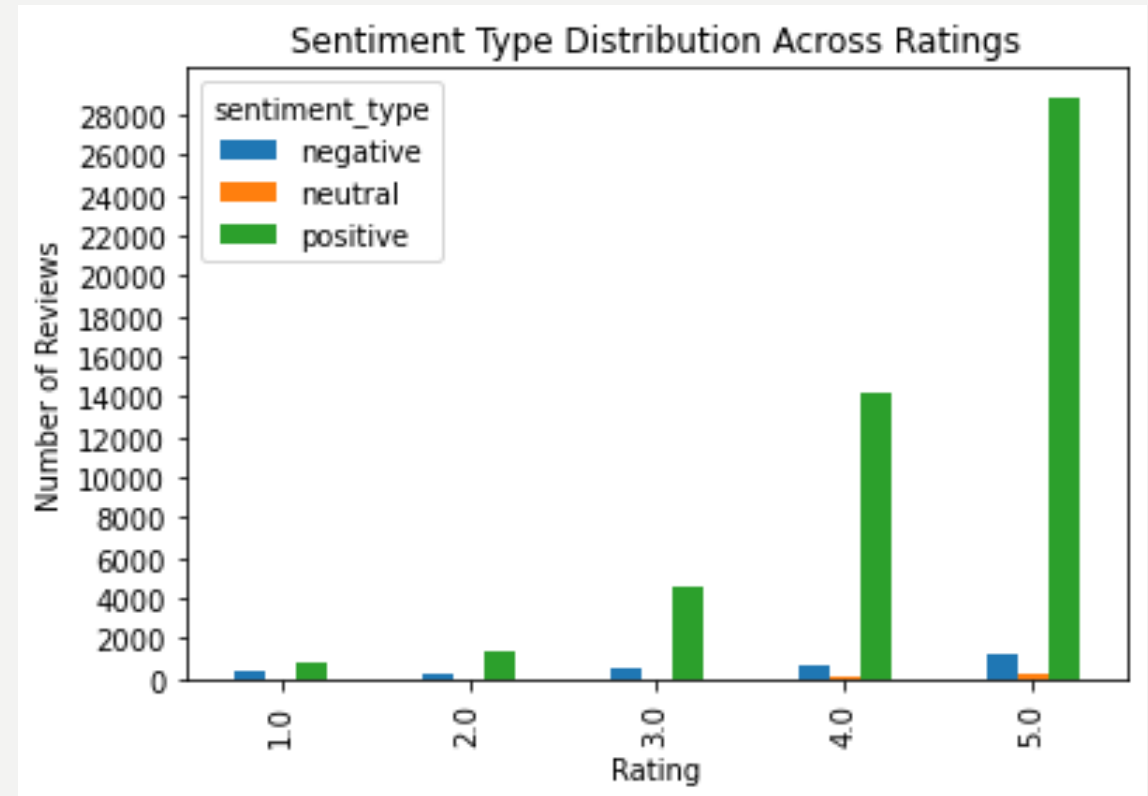
	month	count
3	1	4923
7	2	4172
6	3	4483
4	4	4909
0	5	5424
1	6	5327
5	7	4804
11	8	3239
10	9	3243
9	10	3835
8	11	3959
2	12	4940



PART 2 (NLP): SENTIMENT SCORES & RATINGS COMPARISON

- Plot and print on the console, the distribution of the sentiment types across the ratings as shown below. For this, first create a data frame with three columns, one storing the values of the ratings (1 to 5), another storing the sentiment types (positive, neutral and negative) and the third containing the number of reviews of each sentiment type for each rating. Then create a pivot dataframe where the index values are the ratings, has three columns: positive, neutral and negative and the values are taken from the third column of the aforementioned data frame.

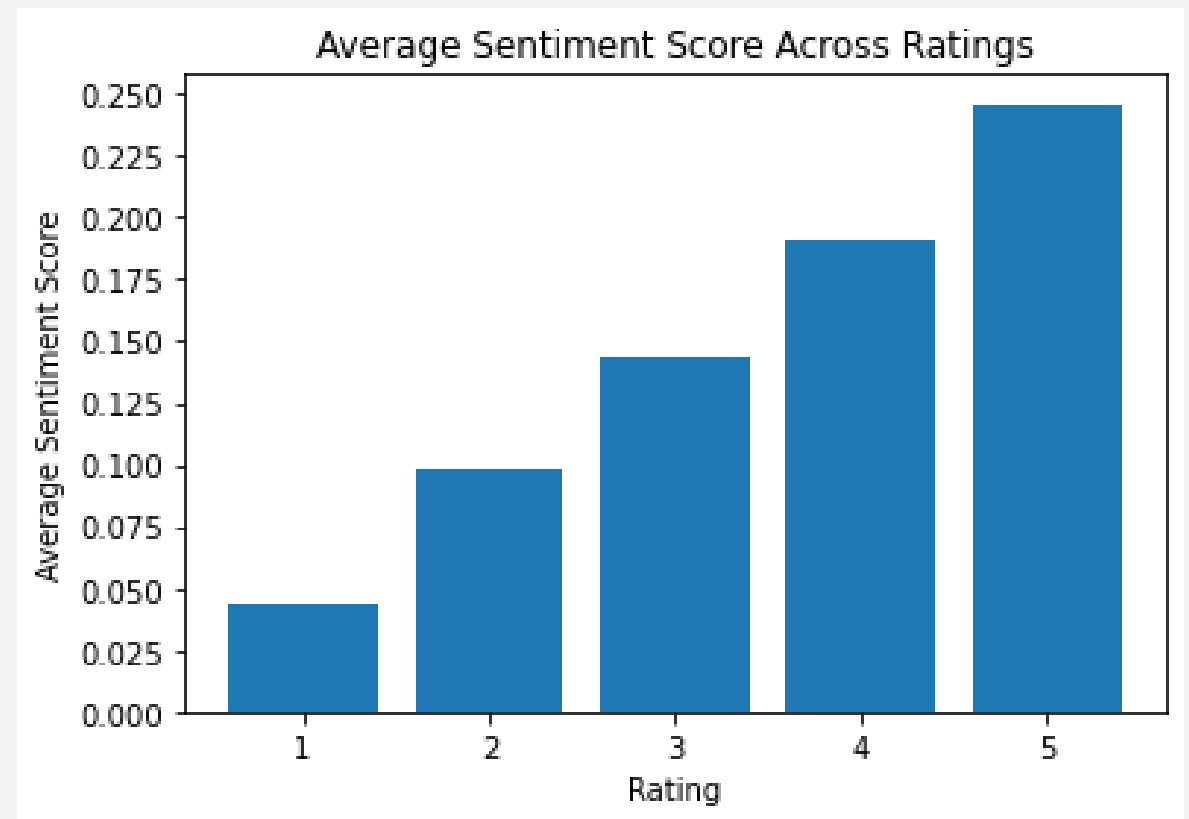
sentiment_type	negative	neutral	positive
overall			
1.0	364	25	741
2.0	320	20	1386
3.0	461	35	4564
4.0	691	89	14235
5.0	1172	263	28892



PART 2 (NLP): SENTIMENT SCORES & RATINGS COMPARISON

- Plot and print on the console, the distribution of the average sentiment scores across the ratings as shown below. For this, first create a data frame with two columns, one storing the values of the ratings (1 to 5) and the other storing the average sentiment scores of the reviews with each of the five ratings.

	overall	avg_sentiment
0	1.0	0.043293
1	2.0	0.098032
2	3.0	0.143741
3	4.0	0.191333
4	5.0	0.245845



IMPORTANT NOTE

- The format, values and content of your the console outputs and plots should match the format, values and content of the console outputs and content shown in the corresponding slides (don't worry about the spacing in the console outputs).
- Values in any of the dataframes or plots should not be hard-coded. They also should not be modified, deleted or added individually or collectively using hard-coded values in a list, dictionary or any other Python data structure.
- The code for the construction of dataframe and/or columns and analysis for each of the deliverables should follow the instructions given on the corresponding slides. For those parts of the deliverables for which the instructions are not given on the slides, you can utilize an appropriate logic to derive the expected console outputs and plots.

PA3 GRADING (150 POINTS TOTAL)

Deliverables		Points
Part 1	Adding the age column	10
	Descriptive Statistics of sqft_living, price, grade, floors, age and view	5
	Association between price and other variables	65
	Efficient Program Structure (e.g. Code breakdown in appropriate methods, comments)	10
Part 2	Adding the month, year, word_count, sentiment_score and sentiment_type columns	25
	Descriptive Statistics of Rating, Word Count and Sentiment Score	5
	Review Volume by Month	20
	Sentiment Scores & Ratings Comparison	30
	Efficient Program Structure (e.g. Code breakdown in appropriate methods, comments)	10

*20 potential points for additional analysis

*10 potential bonus points

Any additional/extra features should be described briefly in comments in the beginning of the code