Virginia Tech

Bradley Department of Electrical and Computer Engineering

ECE 5984 Data Engineering Project
Fall 2023

Assignment 2
Data Quality Assessment and Data Exploration

Please note the following:

- Solutions must be clear and presented in the order assigned. Solutions must show the work needed, as appropriate, to derive your answers.
- Data being processed in both the lab and homework should be in the correct format and location and any other deliverables should also be completed as mentioned in the modules.
- For the Homework, the submission process requires that all code files should be uploaded to Canvas before the deadline.
- Submit your Homework using the respective area of the class website by 11:55 p.m. on the due date.
- When a PDF file must be submitted, include at the top of the first page: your name (as recorded by the university), email address, and the assignment name (e.g., "**ECE 5984, Homework/Project 1**"). Submit a single file unless an additional file is explicitly requested.

Lab 2

2.1 Perform EDA (exploratory data analysis) on the data ingested from Lab 1.1

In this lab we will be performing exploratory data analysis (EDA). Exploratory data analysis is a quick look at your dataset to help you understand its structure, form, and size, as well as find patterns. EDA is often performed before tasks like data cleaning, transformations and feature extraction but tasks like those are only possible because of EDA. During EDA the top priority is to get a feel of the dataset and see what type of data needs to be cleaned, extracted or in general worked on. A large portion of this falls on the discretion of the task at hand.

For example, finding an outlier can be a bad thing if you are training a simple regression model and should be removed from the dataset before training the model but an outlier in a dataset that is to be used for an outlier detection model requires that the outliers stay in it. In this lab we will go through our financial dataset assuming we want to make a prediction model based on the 'adj close' price of different companies along with creating certain dashboards and metrics.

1. Open a new pycharm project on your local machine
2. Download the pickle data that was uploaded to your S3 bucket(data lake) from the Lab 1.1
3. Save the pickle data file to your local machine project folder
4. Save the provided EDA.py file in the same project location
5. Install the libraries needed to run the EDA.py script. That should include pandas
6. Load the variable raw_data in the EDA.py script with the pickle file downloaded in step 2 by putting the name of the pickle file in the load function inside the single quotes. If you have placed the pickle file in a different directory as the EDA.py file you need to use the path location of the file in the between the single quotes

The local machine will be used to perform EDA and figure out what kind of cleaning/transformation functions need to be performed and a different script will be run as a DAG later to perform the needed cleaning/transformation tasks on the pipeline.

7. Run the EDA.py script and follow along this lab and the EDA.py code file. You should see stuff like the screenshots below. The specific values that need to be changed will be different for each student however.
8. Display the raw data Dataframe. This can be done by the following function
   `print(df),` df being the dataframe you want to display

```
The Dataset looks like:
            Adj Close             ...     Volume
                 AAPL       AMZN  ...       AMZN       GOOGL
Date                              ...
2019-01-02   38.047047   76.956497 ... 159662000.0 31868000.0
2019-01-03   34.257278   75.014000 ... 139512000.0 41960000.0
2019-01-04   35.719696   78.769501 ... 183652000.0 46022000.0
2019-01-07   35.640198   81.475502 ... 159864000.0 47446000.0
2019-01-08   36.319611   82.829002 ...         NaN 35414000.0
...                ...         ... ...         ...        ...
2022-12-28  125.847855   81.820000 ...  58228600.0 19523200.0
2019-05-28   43.290482   91.821503 ...  64000000.0 20948000.0
2021-05-06  128.196884  165.318497 ...  88954000.0 25190000.0
2020-12-09  120.152000  155.210007 ...  82016000.0 31728000.0
2019-07-24   50.684097         NaN ...  52626000.0 27192000.0

[1109 rows x 18 columns] SIZE OF DATAFRAME
```

9. Size can be gotten separately by the following function as well
   `df.shape`
10. As seen above, since the dataframe is huge, therefore, in order to make it more manageable pandas automatically depreciates and hides a lot of the rows and columns of the dataframe. However sometimes you want the full information if you want to go through a big dataset. This can be done by setting certain pandas options by using functions like:
    ```
    pd.set_option('display.max_columns', None)
    pd.set_option('display.max_rows', None)
    ```
11. In order to view all the columns together you can also convert the dataframe to a string and then display it
    `df.to_string()`
12. In order to view the first n rows or last n rows of the dataframe the following functions can be used respectively
    ```
    df.head(n)
    df.tail(n)
    ```

Basic EDA functions

1. `df.info()`

```
Basic Dataframe info
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1109 entries, 2019-01-02 to 2019-07-24
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   (Adj Close, AAPL)   997 non-null    float64
 1   (Adj Close, AMZN)   999 non-null    float64
 2   (Adj Close, GOOGL)  995 non-null    float64
 3   (Close, AAPL)       1001 non-null   float64
 4   (Close, AMZN)       998 non-null    float64
 5   (Close, GOOGL)      1001 non-null   float64
 6   (High, AAPL)        1001 non-null   float64
 7   (High, AMZN)        1000 non-null   float64
 8   (High, GOOGL)       1002 non-null   float64
 9   (Low, AAPL)         1000 non-null   float64
 10  (Low, AMZN)         998 non-null    float64
 11  (Low, GOOGL)        999 non-null    float64
 12  (Open, AAPL)        1000 non-null   float64
 13  (Open, AMZN)        993 non-null    float64
 14  (Open, GOOGL)       994 non-null    float64
 15  (Volume, AAPL)      1003 non-null   float64
 16  (Volume, AMZN)      1001 non-null   float64
 17  (Volume, GOOGL)     996 non-null    float64
```

Displays all the columns with a non-null count and data type for each column

2. `df.describe()`

```
More detailed Dataframe info
        Adj Close                                Close                                   High
             AAPL        AMZN       GOOGL        AAPL        AMZN       GOOGL        AAPL         AMZN        GOOGL
count  997.000000  999.000000  995.000000 1001.000000  998.000000 1001.000000 1001.000000 1000.000000 1002.000000
mean   114.748893  135.730941   96.950444  115.267785  133.394880   98.081052  115.442782  134.551054   98.587066
std     81.605879   85.038760   71.062936   81.290381   75.742708   76.328259   76.703083   70.617921   70.952436
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.000000
25%     65.655716   93.424252   64.809498   66.572502   93.496248   65.007004   67.000000   94.706375   65.474998
50%    122.515259  130.042999   88.083000  122.540001  127.380501   88.206497  124.180000  130.880501   89.118000
75%    146.895905  161.449745  117.247002  147.110001  161.180500  117.334000  148.449997  163.175507  119.197372
max   1000.000000 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000 1000.000000
```

Gives a more detailed description of the dataframe. Metrics like the following are calculated for each column:

count - The number of not-empty values.
mean - The average (mean) value.
std - The standard deviation.
min - the minimum value.
25% - The 25% percentile.
50% - The 50% percentile.
75% - The 75% percentile.
max - the maximum value.
Percentile meaning: how many of the values are less than the given percentile.

3. `df.isnull().sum().sort_values(ascending = False)`

```
Number of Empty values in each column:
Open       AMZN     116
           GOOGL    115
Adj Close  GOOGL    114
Volume     GOOGL    113
Adj Close  AAPL     112
Low        AMZN     111
Close      AMZN     111
Adj Close  AMZN     110
Low        GOOGL    110
High       AMZN     109
Open       AAPL     109
Low        AAPL     109
Close      GOOGL    108
High       AAPL     108
Close      AAPL     108
Volume     AMZN     108
High       GOOGL    107
Volume     AAPL     106
dtype: int64
```

Displays the number of empty values for each column in descending order

4. `df.apply(pd.Series.nunique)`

```
Number of Unique values in each column:
Adj Close  AAPL     895
           AMZN     893
           GOOGL    894
Close      AAPL     880
           AMZN     893
           GOOGL    895
High       AAPL     869
           AMZN     885
           GOOGL    894
Low        AAPL     881
           AMZN     890
           GOOGL    893
Open       AAPL     873
           AMZN     871
           GOOGL    889
Volume     AAPL     897
           AMZN     892
           GOOGL    888
dtype: int64
```

Counts the number of unique values in each column

5. `df.duplicated()`

```
Date
2019-01-02    False
2019-01-03    False
2019-01-04    False
2019-01-07    False
2019-01-08    False
              ...
2022-12-28     True
2019-05-28     True
2021-05-06     True
2020-12-09     True
2019-07-24     True
```

Return boolean Series denoting duplicate rows.

From looking at the results we can extrapolate a lot of information. Some key points that should be noted:

a. There are empty values in the dataset. This can be seen because the dataset has NaN elements and also the `df.isnull()` function shows that there are indeed NaN elements.
b. The dataset has outliers. This can be inferred because in `df.describe()` we can see the mean for most columns is around the 100 range but the max and min are a 1000 and 0 indicating outliers.
c. The dataset also has duplicate rows as seen by the `df.duplicated()` function.

For our task at hand which will eventually be to make a prediction model based on the 'adj close' value, we want to get rid of all the rows with the type of data mentioned above.

2.2 Perform basic data cleaning and basic data transformations on the data ingested from Lab 1.1 using pandas and push the cleaned data onto S3 bucket (data warehouse)

After performing our EDA we have to make certain decisions depending on the task at hand. Since our goal is to make a predictive model based on the 'adj close' price of the 3 companies (Apple, Amazon, Google) that we have selected, as discussed earlier, we remove the rows with NaN values, remove rows with outlier values and remove any duplicated rows. To do so the following functions are used.

Basic Data cleaning functions

1. `df.dropna()`
   Removes missing values from a dataframe.
2. `df.drop()`
   Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names. Can also drop rows on columns based on certain parameters.
3. `df.drop_duplicates()`
   Return DataFrame with duplicate rows removed.

Alongside cleaning the data we also transform the data to make it more readable and manageable before we push it onto our data warehouse(S3 bucket). One important task we do is divide the data from the 3 companies to 3 separate tables for each company and then push the data forward according to a database schema. All of these tasks are done by the function called transform_data() in the transform.py script. You can go through the script and understand the functions and how they work. You can also copy the functional parts of the code from the transform.py script and paste it towards the end of the EDA.py script and print out into the console the changes that are made to the raw data instead of pushing the transformed data to your S3 bucket after transformation in order to better understand the order of operations.

Now in order to use the transform.py script in our pipeline we need to push the transform.py code onto our cloud infrastructure along with changing the dag.py file from the earlier Lab 1.1 to actually use the new code we are pushing. Even Though a different batch_ingest.py file is provided it should be noted that it is exactly the same as the one in lab 1.1 since we do not need to make any changes in batch_ingest.py because we are still ingesting data in exactly the same way. In order to run your code on airflow follow the following:

1. Open the batch_ingest.py file, the dag.py file and the transform.py file on your local machine
2. Edit and fill in the needed details in the 3 files, key being the S3 bucket location where you want to retrieve your pickle data file in batch_ingest.py and the transform.py file, The S3 bucket folder location where you want your final cleaned and transformed data to end up at in the transform.py file.
3. Save the dag.py, batch_ingest.py and transform.py scripts to your local machine
4. Spin up a docker container as shown in Step 2.2: Spin up and exit out of a docker container (lab 0)
5. Install the needed packages for the dataset running the following commands
   a. `pip install pandas-datareader`
   b. `pip install yfinance`
6. Navigate to the airflow/dags folder by typing the following command:
   a. `cd airflow/dags`
7. Create a new dag.py and copy all the code from your local machine dag.py onto it. To do so enter the command:
   a. `sudo nano dag.py`

8. This will open a command line based text editor. Copy the contents of your local dag.py file to the newly created one inside your container (Tip: use ctrl+shift+V to paste content directly if using the web browser)
9. If you already have a dag.py that exists the command should open that file. Replace all the contents of your old dag.py file with new content from your local machine
10. Save by pressing ctrl+x , then press y to confirm changes and then hit Enter
11. Similarly create/replace the batch_ingest.py and create a new transform.py file and copy code from local machine batch_ingest.py and transform.py onto it using the same steps
12. Access your airflow GUI (Step 2.4: Launching and access the airflow GUI (lab 0))
13. You should see the same **batch_ingest_dag**. Click it and go to graph. On the right side click the play button and press trigger DAG
14. This triggers the dag we just uploaded which intern runs the ingest_data function from batch_ingest.py and the transform_data() function from transform.py and you are able to see the whole process
15. After the DAG finishes running you should be able to navigate to your S3 bucket directory and check to see the data now there in a pickle file format.
16. After confirming your data arrived at the correct location in your S3 bucket, close airflow by pressing ctrl-C on the command line where airflow is running
17. Exit out of the container you have been working from using the command `exit` and double check if the container actually stopped by using the command `docker ps`. To manually stop the container if it had not done so automatically run the command `docker stop <pid>`

Homework 2

Perform the same EDA (Exploratory Data Analysis) process as done in Lab 2.1 using a different ingested data source. This can be the same data source as used in Lab 1.1

Some example datasets that can be used:

1. https://www.kaggle.com/datasets/borismarjanovic/price-volume-data-for-all-us-stocks-etfs
2. https://www.kaggle.com/datasets/bharatnatrayn/movies-dataset-for-feature-extracion-prediction

Deliverables

1. The dag.py file, batch_ingest.py and the transform.py file for the homework should be uploaded to Canvas.
2. Stock data from homework should be in the appropriate S3 bucket locations (data lake and data warehouse).
3. Data from the lab should be in the appropriate S3 bucket location (data lake and data warehouse).