Machine Learning

Milestone 1

Project Name: Hotel Rating

Create a 3 files.py

1. preprocessing_Data.py: - to create all functions used in preprocessing and scaling and outliers.
2. main.py: - to read and split and preprocess and fit models about data.
3. Testing File.py: - to read test data and preprocessing and scaling and read models from file and predict output to all models.

preprocessing_Data.py

1. **Import Libraries**
2. Fill Null Values in Training Set: -

   Create a function **'fill_Null_train(data)',** which work as follow: -

   1) calculate **missing values in each feature 'mask_F'** this equal 'data. isnull (). any (axis=0)'.
   2) calculate **missing values in each row 'mask_R'** this equal 'data. Isnull (). any(axis=1)'.
   3) calculate **percentage of each row with na_values 'p_rows_with_nan'** this equal 'mask_R. sum()/len(data)'.
   4) calculate **percentage of each feature with na_values 'p_features_with_nan'** this equal 'mask_F.sum()/len(data)'.
   5) check **if** percentage of each row with na_values 'p_rows_with_nan' smaller than 0.07 then we deal with data without these rows 'in other hand we delete the rows have na_values.'
   6) **else** we first arrange these features by calculate mean to all of these ('lat', 'lng', 'avarage_score', 'Total_Numbers_of_Reviews') cause the 'Total_Numbers_of_Reviews' depends on Total number of valid reviews the hotel has, and the 'avarage_score' depends on the latest comment in the last year of this hotel.

7) use **fillna function** which take column name and take value to replace Na_values with it, replace Na_values in 'Hotel_Address' column with 'no address' , and 'Additional_Number_of_Scoring' column with '0.0', and 'Day' column with '00', and 'Month' column with '00', and 'Year' column with '0000', and 'Average_Score' column with mean, and 'Hotel_Name' column with 'no name', and 'Reviewer_Nationality' column with 'no info', and 'Negative_Review' column with 'Positive', and 'Review_Total_Negative_Word_Counts' column with '0.0', and 'Total_Number_of_Reviews' column with mean, and 'Positive_Review' column with 'Negative', and 'Review_Total_Positive_Word_Counts' column with '0.0', and 'Total_Number_of_Reviews_Reviewer_Has_Given' column with '0.0', and 'Tags' column with "['no trip', 'no Couple', 'no Room', 'Stayed 0', 'not Submitted']",and 'days_since_review' column with '0.0', and 'lat' column with mean, and 'lng' column with mean, and 'Reviewer_Score' column with '0.0'.

8) Defined data frame 'values' to restore these values in file ('lat'. mean, 'lng'. mean, 'avarage_score'. mean, 'Total_Numbers_of_Reviews'. mean),

after that return result.

3. Fill Null Values in Test Set

Create a function **'fill_Null_test (data)',** which work as function **'fill_Null_train(data)'** but it has a simple difference: - **open file values** to read the contain data in it (mean of columns ['lat', 'lng', 'avarage_score', 'Total_Numbers_of_Reviews']) after reading this data continue by this data.

4. Split Date

Create a function **'split_date(data)',** which work as follow: -

1) **split the column of 'Review_Date'** by any sign like "/, \, -", and put result in 'day & month & year' by creating data frame include 3

columns "'Day', 'Month', 'Year' ", and convert data from string to integer then we drop the column of 'Review_Date'.

2) **split the column of 'date_since_review'** by putting the column in a list called 'days' and make an empty list called 'day' then we start to iterate on first list to remove word 'day/s' by replacing it with empty space and then put the data of old list in 'day' list, after that start to convert the data of the 'day' list from string to integer and put it in another list 'res' then we make column 'date_since_review equals to 'res list',

after that return result.

5. Split Tags Columns

Create a function **'split_Tags (SplitDate)',** which work as follow: -

1) Then we **split the 'Tags'** column first create 5 empty lists called ('Trip' ,'Memories', 'Room_Kind', 'Nights', 'the_way_of_submission') and list called 'Tags' and put 'Tags' column in it .
2) after that we **iterate on 'Tags'** list and create another list called 'tag' and we replace each "[,],' " for list tags with empty space then we split each time we find ',' and append in 'tag' list then we loop on 'tag' list and start to create 5 Boolean variables called ('trip', 'mem', 'room', 'night', 'submission') with initial values False, then we create a loop start from k to iterator of outer loop **if** k (each part of list 'tag') contain 'trip' put this in list 'Trip' and make Boolean trip equals true, **else if** k contains 'room' or 'Room' put this in list 'Room_Kind' and make Boolean room equals true,

   else if k contains 'Stayed' put this in list 'Nights' and make Boolean night equals true,

   else if k contains 'Submitted' put this in list 'the_way_of_submission' and make Boolean submission equals true,

else if k contains ‘Couple’ or ‘Group’ or ‘children’ or ‘traveler’ put this in list ‘Members and make Boolean mem equals true, and contains the outers loop to check **If** (not trip)
put in the list Trip ‘trip’,

else if (not room) put in the list Room_Kind ‘no Room’, else if (not night) put in the list Nights ‘Stayed 0’,

else if (not submission) put in the list the_way_of_submission ‘not Submitted’,

else if (not mem) put in the list Members ‘no Couple’.

3) Then we **create 5 columns** in the data called

   Trip to put in its ‘Trip’ list, **Member** to put in it ‘Members’ list, **Room** to put in it ‘room_kind’ list, **Submission** to put in it ‘the_way_of_submission’ list.

4) Then we create empty list called ‘numbers’ and **iterate on ‘Nights’** list until its length Split data and take only number from it and put this number on list called ‘converted number.’, then create a list called ‘convert’ to put in it the number from ‘converted number’ after converting it to integer, then append ‘convert’ list in ‘number’ list then **create a column called ‘Nights’** in data and make it equals ‘numbers’ list, after all of that we **drop the ‘Tags’** column,

after that return result.

6. Label Encoder to Convert Data to Numerical

Create a function **‘Feature_Encoder (data)’,** which work as follow: -

1) 1- **Defined object** from label encoder
2) start with ‘Hotel_Address’ column and make **fit transform** it , ‘Hotel_Name’ column and make fit transform it, ‘Reviewer_Nationality’ column and make fit transform it, ‘Members’ column and make fit transform it , ‘Room’ column and make fit transform it, ‘Submission’ column and make fit transform it,

'Negative_Review 'column and make fit transform it,
'Positive_Review' column and make fit transform it.

3) Start to **arrange our data** by creating data frame called 'data_num' and give it the numbers of columns as we need it to be arranged, to make data arrange as past and make the 'Reviwer_Score' column at the end, **defined 2 data frames 'X','Y'**, X read all the columns expect the 'Reviwer_Score' column, and Y read 'Reviwer_Score' column,

after that return result.

7. Feature Scaling

1. Normalizing Data

- Create a function called 'Normalization' which take (data)
  1- create object from 'MinMaxScaler'.
  2- fit the data and transform it and put it in data frame called 'Normalized_x'.
  3- return'Normalized_X'.

2. Standardize Data.

- Create a function called 'Standardization.' Which take (data)
  1- create object scaler from 'StandardScaler'.
  2- fit the data and transform it and put it in data frame called 'x_scaled'.
  3- return 'x_scaled'.

8. Box Plot

Create a function called **'plot_boxplot'** which take data and column which I want to apply on it, then we use df.boxpolt and give it the column I want draw on it.

9. Outliers

1- Create function **'delete_outlier (data, name, upper_limit, lower_limit)'** to delete outliers,

- create data frame called ‘update_data’ and put on it data of column if data of column less than or equals upper limit and bigger than or equals lower limit
- after that return update_data.

2- Create function **‘Norm_outlier (data, name, upper_limit, lower_limit)’** to arrange the ranges of outlier,

- create data frame called ‘new_data’ which copy data on it
- check if ‘new_data’ of name bigger than ‘upper_limit’ make it ‘upper_limit.’
- else make it ‘lower_limit’.
- After that, return ‘new_data’.

3- Create function **‘outliers(data)’** to detect outliers.

- calculate the length of data.
- iterate until length.
- create var called name carry columns name.
- calculate Q1 = data[name]. quantile (25%).
- Q3 = data[name]. quantile (75%).
- calculate IQR = Q3-Q1.
- make upper_limit =q3+1.5*IQR.
- lower_limit =q1-1.5*IQR

after all of this

-**If we Want to delete outliers**:

  - We call ‘delete_outlier’ and put the result of it in data frame called data.
  - Then return data.

-**If we want to arrange the ranges of outlier**:

- We call 'Norm_outlier' and put the result of it in data frame called data.
- Then return data.

4- Finally, we use z-score technique:

We create function called **'z-score(data)',** which take data to work on as follow: -

1) First, we get the length of data columns and store it in variable called n.
2) start to iterate on this n and make variable name equals to data. columns[i] and upper_limit=data[name].mean() +3*data[name]. std (), and lower_limit=data[name]. mean()-3*data[name]. std (), then call Norm_outlire ()

   and give it data, name, upper_limit, lower_limit Store the output in data variable.
3) return data.

main.py

1. Import Libraries
2. Read Data

   - **Read our dataset** from the csv file, after that we concatenate 'No Negative' and 'No Positive' in values of 'na_values', read all this in data frame **'hotel_data'**.
3. Split Data

   1- defined 2 data frames 'X','Y',

   - X → read all the columns in 'hotel_data' expect the 'Reviwer_Score.'

column.

- Y → read 'Reviwer_Score' column.

2- Split X, Y by 'train_test_split' in **80% to train set** and **20% to test set** return splits in **(X_train, X_test, Y_train, Y_test).**

3- Create a 2 data frame: -

- Data frame called **'data'** to concatenate ('X_train', 'Y_train') in it.
- Data frame called **'data_test'** to concatenate ('X_test', 'Y_test') in it.

4. Split Date

- Defined data frame '**SplitDate**' to return output of function '**split_date(data)**'.

5. Split Tags Columns

-Defined data frame **'SplitTags'** to return output of function **'split_Tags (SplitDate)'**

6. Fill Null Values in Training Set

-Defined data frame **'data_fill_null'** to return output of function **'fill_Null_train (SplitTags)'.**

7. Label Encoder to Convert Data to Numerical

-Defined 2 data frame **'X', 'Y'** to return output of function **'Feature_Encoder (data_fill_null)**'

8. Feature Scaling

-Defined data frame called **'X'** equals called 'Standardization.' function and pass parameters to it(X).

-Then create 2 data frames called 'dff' and 'dff1.'
dff equals X after are scaling it, dff1 equals Y.

- merge the 2 data frames into **'df_merged'** data frame by concatenate dff and dff1.

9. Outliers

- Defined data frame **'data_cleaned'** to return output of function **'z_score (df_merged)**'.

- **defined 2 data frames 'X','Y',**

  - X → read all the columns in **'data_cleaned'** expect the 'Reviwer_Score.' column.
  - Y → read 'Reviwer_Score' column.

10. Feature Selection

- **F-classify Method**

  - use built in function called 'SelectKBest (score_func=f_clssif, k=7) 'It returns the best 7 features of data in variable called fs The we are applying fit on fs.
  - Using fit (X, Y. values. ravel ()) function Then we create variable selected_features and store in it the return of X. columns [fs.get_support ()]
  - After that print selected_features.
  - make X=X[selected_features]
  - open selected_features file and store on it the output.

11. Models

1). Multi Linear Regression Model

- Create object from 'linear_model. LinearRegression'. ○ Fit X, Y.
- open the file multi linear regression and store on it the 'model' object after applying fit data on it.

2). Polynomial Regression Model

- Called function 'PolynomialFeatures (degree 2)' in poly_features.
- Transform the exiting features to higher degree features.
  - X_train_poly = poly_features.fit_transform(X).
- Fit the transformed features to Linear Regression
  - poly_model = linear_model. LinearRegression ().
  - poly_model.fit (X_train_poly, y).
- predicting on training set
  - y_train_predicted = poly_model. predict(X_train_poly).
  - ypred = poly_model. predict (poly_features. transform(X)).
- We open the file polynomial regression and store on it the object poly_model.
- Then we open the file poly_features and store on it the object poly_features.

3). Lasso Regression Model

- Create object from 'Lasso(alpha=0.1)'
- Fit X, Y.
- We open the file lasso regression and store on it the object lasso.

- Create object from 'Ridge(alpha=0.1)'
- Fit X, Y.
- We open the file ridge regression and store on it the object ridge.

- Create object from 'ElasticNet (alpha=0.1, l1_ratio=0.5)' ○ Fit X, Y.
- We open the file ElasticNet regression and store on it the object elasticnet.

- Create object from 'RandomForestRegressor (n_estimators=100, max_depth=8, random_state=42)'

- Fit X, Y.
- We open the Random Forest object and store on it the object rf.

After all this print ‘train is done’

Testing File.py

1. Import Libraries
2. defined 2 data frames ‘X’,’Y’
   - X → read all the columns in ‘data-test’ expect the ‘Reviwer_Score.’ column.
   - Y → read ‘Reviwer_Score’ column.
3. Split Date

   - Defined data frame ‘**SplitDate**’ to return output of function ‘**split_date(data_test)**’.

4. Split Tags Columns

   -Defined data frame **‘SplitTags’** to return output of function **‘split_Tags (SplitDate)’**

5. Fill Null Values in Training Set

   -Defined data frame **‘data_fill_null’** to return output of function **‘fill_Null_test (SplitTags)’.**

6. Label Encoder to Convert Data to Numerical

   -Defined 2 data frame **‘X’, ‘Y’** to return output of function **‘Feature_Encoder (data_fill_null)**’

7. Feature Scaling

-Defined data frame called **'X'** equals called 'Standardization.' function and pass parameters to it(X).

8. Feature Selection

- We open selected_features file to read data from it and load this data in a data frame called '**X'.**

9. Models

1). Multi Linear Regression Model

- open multi linear regression file to read fit data from it.
- Predict X in Y_pred.
- Calculate **mean square error** of Y, Y_pred and print it.
- Calculate score by using r2_score of Y, Y_pred in r2.
- Calculate **accuracy** by round score by 2 and multiply it in 100 and print this accuracy.

2). Polynomial Regression Model

- open polynomial regression file to read fit data from it.
- open poly_features file to read fit data from it.
- predicting on test set
  - prediction =poly_model. predict(poly_features.fit_transform(X)).
- Calculate **mean square error** → metrics. mean_squared_error (y_test, prediction) and print it.
- Calculate score by using r2_score of Y, prediction in r2.
- Calculate **accuracy** by round score by 2 and multiply it in 100 and print this accuracy.

3). Lasso Regression Model

- open lasso regression file to read data from it.
- Make predictions on the testing data.
  - y_pred = lasso. predict(X).
- Calculate **mean square error** → mean_squared_error (y, y_pred) and print it.

- Calculate score by using r2_score of Y, Y_pred in r2.
- Calculate **accuracy** by round score by 2 and multiply it in 100 and print this accuracy.

- open ridge regression file to read data from it.
- Make predictions on the testing data.
  - y_pred = ridge. predict(X).
- Calculate **mean square error** → mean_squared_error (y, y_pred) and print it.
- Calculate score by using r2_score of Y, Y_pred in r2.
- Calculate **accuracy** by round score by 2 and multiply it in 100 and print this accuracy.

- open elasticnet regression file to read data from it.
- Make predictions on the testing data.
  - y_pred = elasticnet. predict(X).
- Calculate mean square error ⟶ mean_squared_error (y, y_pred) and print it.
- Calculate score by using r2_score of Y, Y_pred in r2.
- Calculate accuracy by round score by 2 and multiply it in 100 and print this accuracy.

6). Random Forest Regression Model

- open Random Forest object file to read data from it.
- Make predictions on the testing data.
  - y_pred = elasticnet. predict(X).
- Calculate mean square error ⟶ mean_squared_error (y, y_pred) and print it.
- Calculate score by using r2_score of Y, Y_pred in r2.
- Calculate accuracy by round score by 2 and multiply it in 100 and print this accuracy.

Observations between all models

1. Multi Linear Regression Model

- Mean Square Error = 1.8969511550866585.
- Accuracy = 30.0%.

2. Polynomial Regression Model

- Mean Square Error = 2.8160270927167845.
- Accuracy = -4.0%.

3. Lasso Regression Model

- Mean Square Error = 1.8775502092720306.
- Accuracy = 31.0%.

4. Ridge Regression Model

- Mean Square Error = 1.8969507866661337.
- Accuracy = 30.0%.

5. The ElasticNet Regression Model

- Mean Square Error = 1.8536575365486878.
- Accuracy = 31.0%.

6. Random Forest Regression Model

- Mean Square Error = 1.5167753723566666.
- Accuracy = 44.0%.

This one is the best of all.

Graphs:

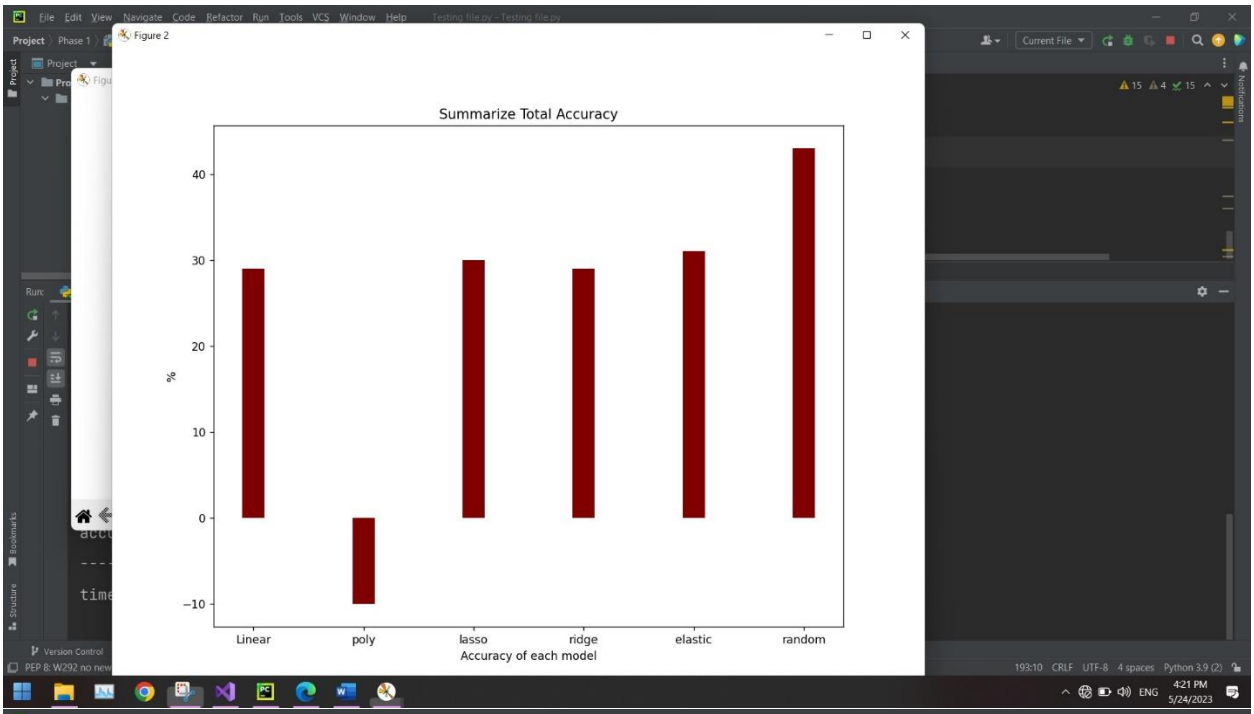
Figure 2
Summarize Total Accuracy
40
30
20
10
0
−10
%
Linear
poly
lasso
ridge
elastic
random
Accuracy of each model

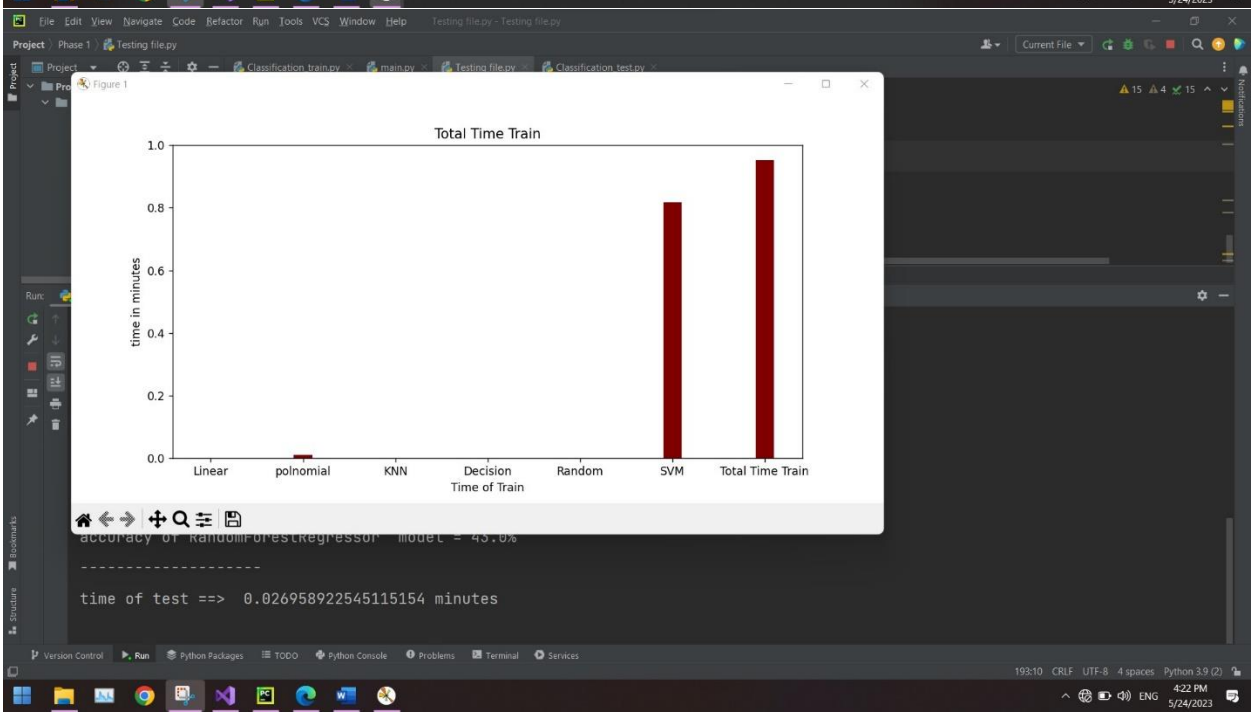
Figure 1
Total Time Train
1.0
0.8
0.6
0.4
0.2
0.0
time in minutes
Linear
polnomial
KNN
Decision
Random
SVM
Total Time Train
Time of Train
time of test ==> 0.026958922545115154 minutes

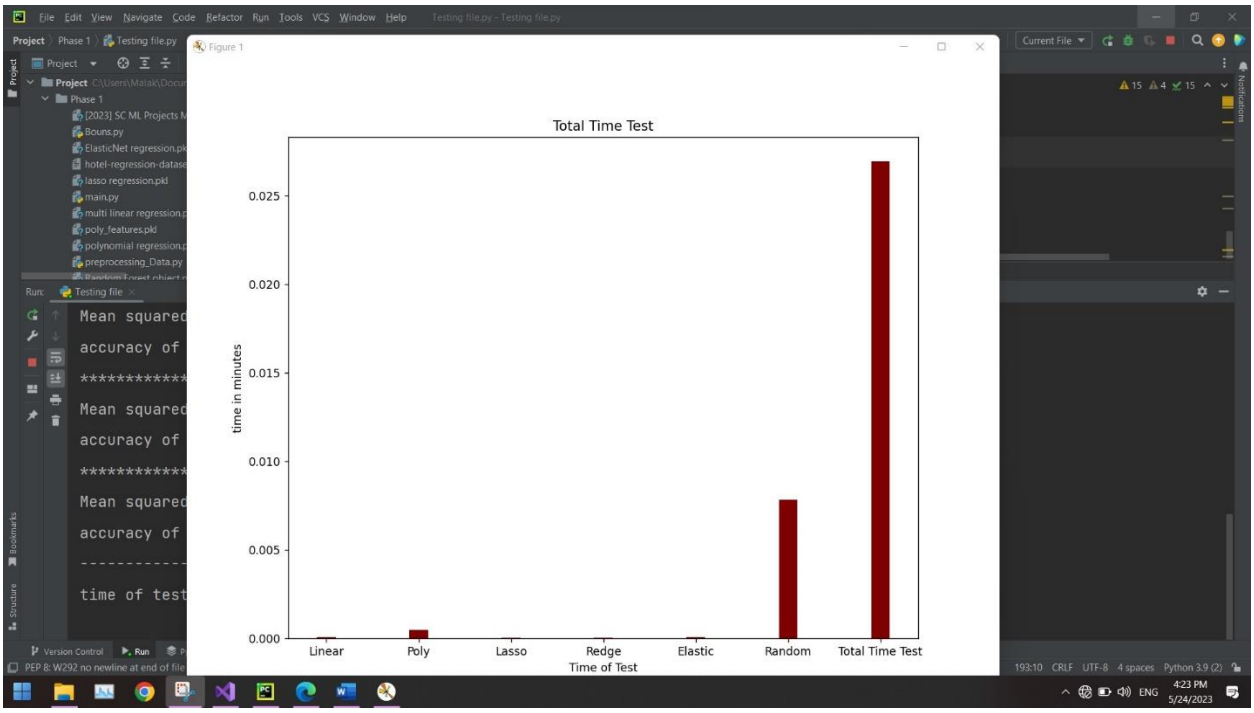
Figure 1
Total Time Test
time in minutes
0.025
0.020
0.015
0.010
0.005
0.000
Linear
Poly
Lasso
Redge
Elastic
Random
Total Time Test
Time of Test