JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.8, Issue.8, August 2020 E-ISSN: 2347-2693

Air Quality Index Prediction with the Implementation of Linear Regression - A Technical Paper

Soumyajit Chakraborty[1*], **Koustav Guha**[2]

[1,2]Dept. of Computer Science, University of Engineering and Management, Kolkata, India

**Corresponding Author: soumyajitchakraborty23@gmail.com*

DOI: https://doi.org/10.26438/ijcse/v8i8.3948 | Available online at: www.ijcseonline.org

Received: 16/Aug/2020, Accepted: 21/Aug/2020, Published: 31/Aug/2020

Abstract- Within the last few years, an intense curiosity has been progressed by the people in the daily air quality circumstances to which they are encountered. Directed by the growing consciousness of the physical state of air pollution exposure, especially by most sensitive sub–populations such as children and the elderly, short–term air pollution forecasts are being accentuated progressively by local authorities. The Air Quality Index (AQI) is the value implemented to estimate the quality of the air at a certain location. The components are estimated with the implementation of the covariance of the input data matrix.

Keywords- Air Quality Index, Linear Regression, Scikits Learn, Seaborn plot, Heat Map, Mean Absolute Error (MAE),Mean Squared Error(MSE),Root Mean Squared Error (RMSE),Pickle.

I. INTRODUCTION

Air is any of the greatest critical expected possessions for the endurance plus subsistence for the complete life on this globe. All forms of life comprising plant life plus wildlife depending on air for their basic endurance. Thus, all breathing creatures require better excellence of air which is free of dangerous fumes for continuing their lives. The increasing populace, vehicles and productions are poisoning all the air at an alarming rate. Dissimilar elements are related with metropolitan air contamination. In order to estimate the air impurity, contaminant constraints are considered in the lower altitudes of the troposphere, which are meticulous. Air excellence sensor devices extend the attention of particles that have an anthropogenic source and create hazardous effects during or after the gulp of air by human beings. Particles like PM2.5 disturbs the quality of air. Automobiles release enormous amounts of nitrogen oxides, carbon oxides, hydrocarbons and particulates when burning petrol and diesel. Therefore, we want to consider the added Parameters like number of vehicles along with the Pollutants data for prediction of pollution.

II. RELATED WORK

An error always leads to hazards within a machine learning interface. In the past, a simple machine learning was implemented for the estimation of Air Quality Index. However, the estimation proved to be inaccurate when was done mathematically. After using the Linear Regression to evaluate the AQI, the Air Quality Index proved to be more precise and accurate. Mean Absolute Error, Mean Squared Error and Root Mean Squared Error played a major role to estimate the accurate value of Air Quality Index.

III. METHODOLOGY

1) Analysing the data: Data has been collected from the AQI.csv file and implemented in the estimation of Air Quality Index.
2) Seaborn pairplot: Seaborn pairplot has been implemented to plot pairwise relationships from the dataset.The pairplot function generates a grid of axes in such a way that each variable in data will be shared in the y-axis across a single row and in the x-axis across a single column.
3) Generating Heat Map: Heat Map is generated to determine the features which are mostly familiar to the target variable
4) Generating Extra Trees Regressor:This module is imported and generated in the model to determine the significance of each and every particular feature of our dataset..
5) Implementation of linear regression: After bringing all the parameters into the learning circumstance, the linear regression is implemented to estimate the Air Quality Index.
6) Train and Test Split: 70% data of the dataset has been utilized for training and 30% has been enabled for the testing process.

7)Estimating the coefficients:Keeping the other features fixed,1 single unit increase in T is related with a decrease of 2.690 in AQI PM2.5.Similarly,keeping the other features fixed,1 single unit increase in TM is related with an increase of 0.46 in AQI PM2.5 .
8) Regression Evaluation Metrics:Mean Absolute Error(MAE),Mean Squared Error(MSE) and Root Mean Squared error (RSME) are calculated.
9) Generating byte stream file using pickle module:-For further evaluation,the object"regression_model.pkl"file is dumped into byte stream file and then again done "unpickling" to reconvert the analysed byte stream file into object file.The file is uploaded in the github and is accessible through this link "**https://github.com/Soumyajit567/Air-Quality-Index".**

IV. RESULTS AND DISCUSSION

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
pd.pandas.set_option('display.max_columns',None)
```

In [2]:

```
dataset=pd.read_csv(r"C:\Users\Soumya\Desktop\Air Quality Index\AQI.csv")

## print shape of dataset with rows and columns
print(dataset.shape)
(1093, 9)
```

In [4]:

```
dataset.head()
```

Out[4]:

Table 1

	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
0	7.4	9.8	4.8	1017.6	93.0	0.5	4.3	9.4	219.720833
1	7.8	12.7	4.4	1018.5	87.0	0.6	4.4	11.1	182.187500
2	6.7	13.4	2.4	1019.4	82.0	0.6	4.8	11.1	154.037500
3	8.6	15.5	3.3	1018.7	72.0	0.8	8.1	20.6	223.208333
4	12.4	20.9	4.4	1017.3	61.0	1.3	8.7	22.2	200.645833

In [6]:

```
## Check for null values

sns.heatmap(dataset.isnull(),yticklabels=False,cbar=False,cmap='viridis')
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e00af63608>
```

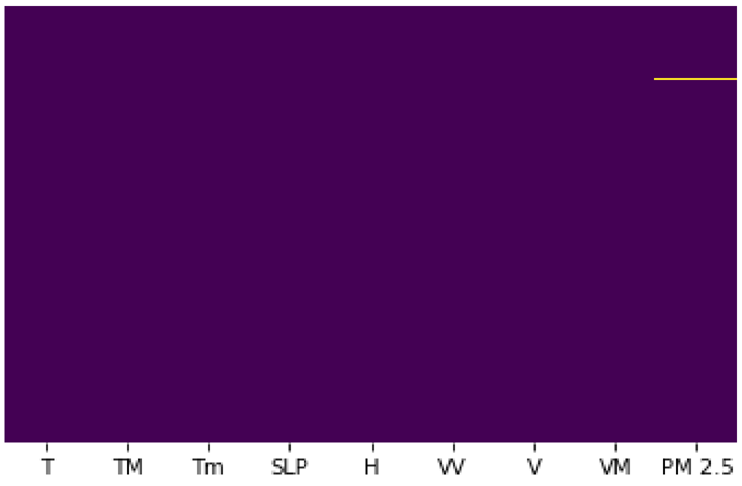

Fig. 1

In [8]:

```
dataset=dataset.dropna()
```

In [9]:

```
X=dataset.iloc[:,:-1] ## independent features
y=dataset.iloc[:,-1] ## dependent features
```

In [10]:

```
## check null values
X.isnull()
```

Out[10]:

Table 2

	T	**TM**	**Tm**	**SLP**	**H**	**VV**	**V**	**VM**
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
1088	False	False	False	False	False	False	False	False
1089	False	False	False	False	False	False	False	False
1090	False	False	False	False	False	False	False	False
1091	False	False	False	False	False	False	False	False
1092	False	False	False	False	False	False	False	False

1092 rows × 8 columns

In [11]:

```
y.isnull()
```

Out[11]:

```
0      False
1      False
2      False
3      False
4      False
        ...
1088   False
1089   False
1090   False
1091   False
1092   False
```

Name: PM 2.5, Length: 1092, dtype: bool

In [13]:

```
sns.pairplot(dataset)
```

Out[13]:

```
<seaborn.axisgrid.PairGrid at 0x1e00cb9abc8>
```

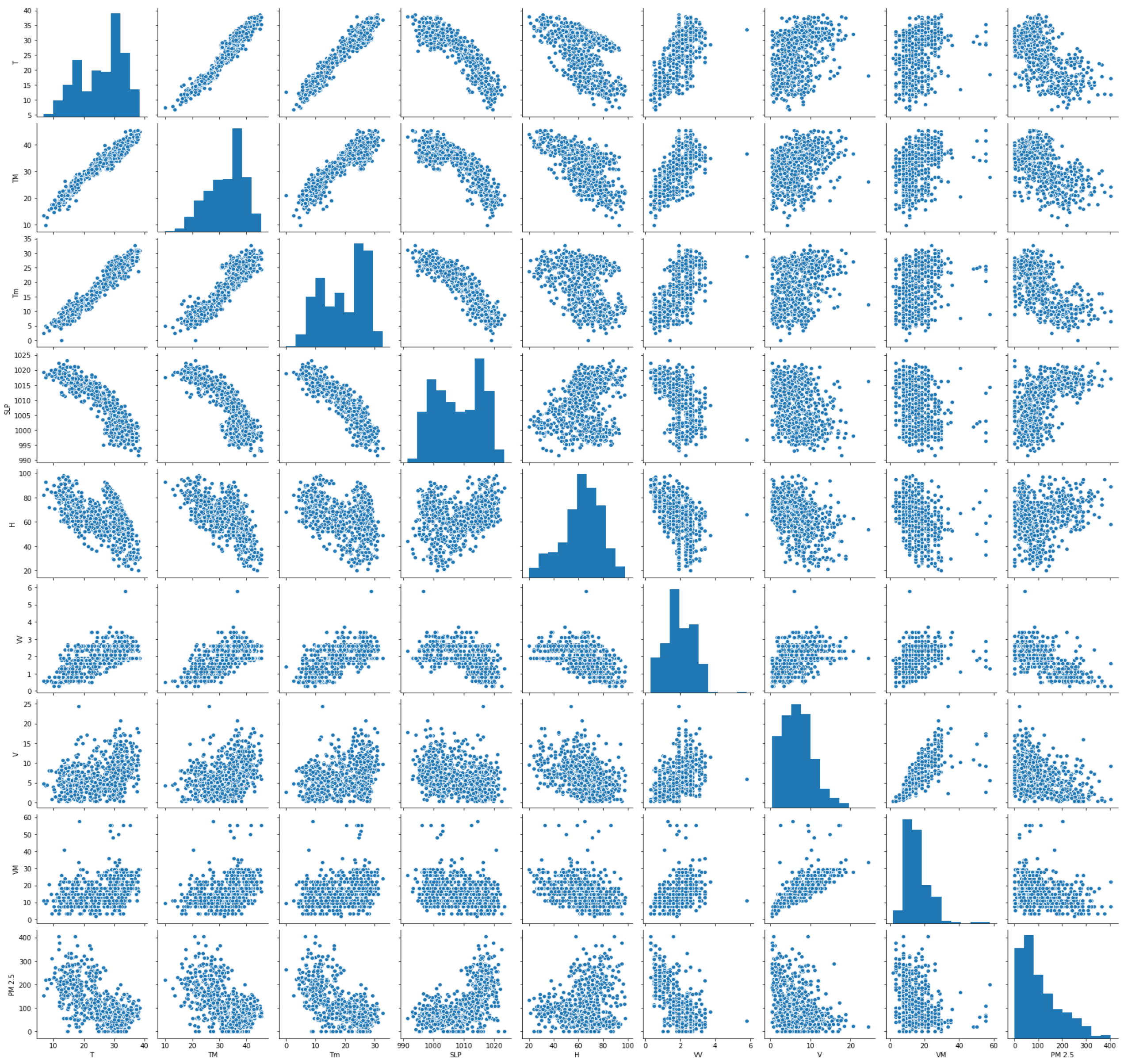

Fig. 2

In [14]:

```
dataset.corr()
```

Out[14]:

Table 3

	T	TM	Tm	SLP	H	VV	V	VM	PM 2.5
T	1.000000	0.967536	0.953719	-0.881409	-0.509299	0.640792	0.301994	0.287738	-0.631462
TM	0.967536	1.000000	0.892031	-0.822958	-0.586681	0.606945	0.292949	0.297011	-0.568409
Tm	0.953719	0.892031	1.000000	-0.917518	-0.287357	0.577240	0.296225	0.266782	-0.673824
SLP	-0.881409	-0.822958	-0.917518	1.000000	0.240256	-0.517915	-0.329838	-0.310704	0.623187
H	-0.509299	-0.586681	-0.287357	0.240256	1.000000	-0.465374	-0.380575	-0.362177	0.138005
VV	0.640792	0.606945	0.577240	-0.517915	-0.465374	1.000000	0.376873	0.342442	-0.573941
V	0.301994	0.292949	0.296225	-0.329838	-0.380575	0.376873	1.000000	0.775655	-0.268530
VM	0.287738	0.297011	0.266782	-0.310704	-0.362177	0.342442	0.775655	1.000000	-0.215854
PM 2.5	-0.631462	-0.568409	-0.673824	0.623187	0.138005	-0.573941	-0.268530	-0.215854	1.000000

Correlation Matrix with Heatmap

Correlation states how the features are related to each other or the target variable.

Correlation may be positive or negative .Heat Map implementation makes it convenient to predict which features are mostly familiar to the target variable, we will be plotting the heatmap of all the correlated features with the help of the seaborn library.

In [15]:

```
import seaborn as sns
#get all the correlations of each feature in dataset
corrmat = dataset.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plotting a heat map
g=sns.heatmap(dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

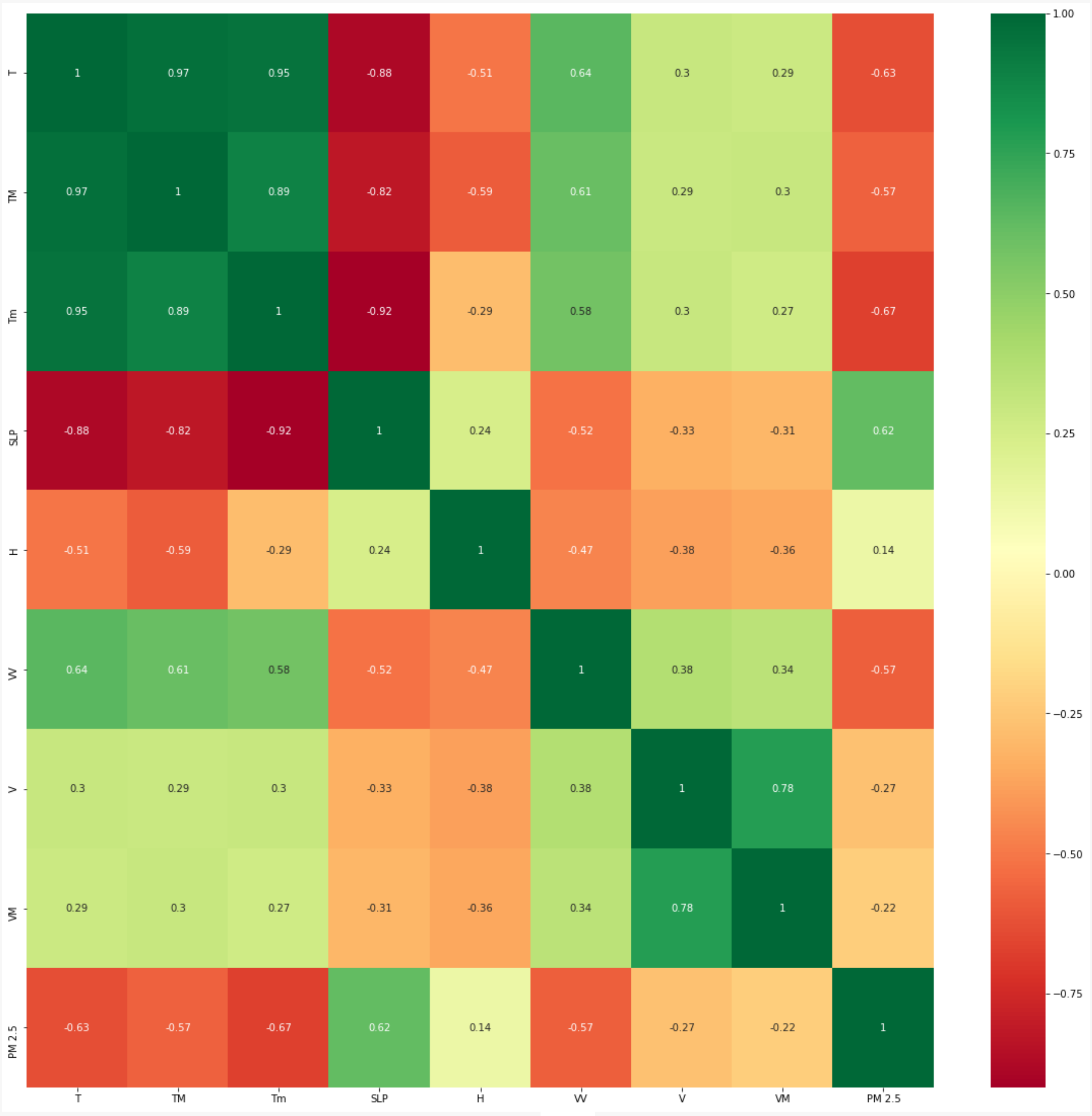

Fig. 3

In [16]:

```
corrmat.index
```

Out[16]:

```
Index(['T', 'TM', 'Tm', 'SLP', 'H', 'VV', 'V', 'VM', 'PM 2.5'], dtype='object')
```

Feature Importance

We may obtain the feature significance of each individual feature of our particular dataset by implementing the feature importance property of the observed model.

Feature importance provides us the score for every individual feature of our data, the greater the score more important or familiar will be the feature towards our output variable.

Feature importance is the inbuilt class which is derived with Tree Based Regressor, and we will be implementing Extra Tree Regressor for the purpose of deriving the top ten features in respect to the dataset.

In [18]:

```
from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
```

Out[18]:

```
ExtraTreesRegressor(bootstrap=False, ccp_alpha=0.0, criterion='mse',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    max_samples=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=100, n_jobs=None, oob_score=False,
                   random_state=None, verbose=0, warm_start=False)
```

In [19]:

```
X.head()
```

Out[19]:

Table 4

	T	**TM**	**Tm**	**SLP**	**H**	**VV**	**V**	**VM**
0	7.4	9.8	4.8	1017.6	93.0	0.5	4.3	9.4
1	7.8	12.7	4.4	1018.5	87.0	0.6	4.4	11.1
2	6.7	13.4	2.4	1019.4	82.0	0.6	4.8	11.1
3	8.6	15.5	3.3	1018.7	72.0	0.8	8.1	20.6
4	12.4	20.9	4.4	1017.3	61.0	1.3	8.7	22.2

In [20]:

```
print(model.feature_importances_)
[0.16184242 0.08201152 0.21249824 0.17956508 0.08955295 0.17882091
 0.0543147  0.04139419]
```

In [21]:

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```

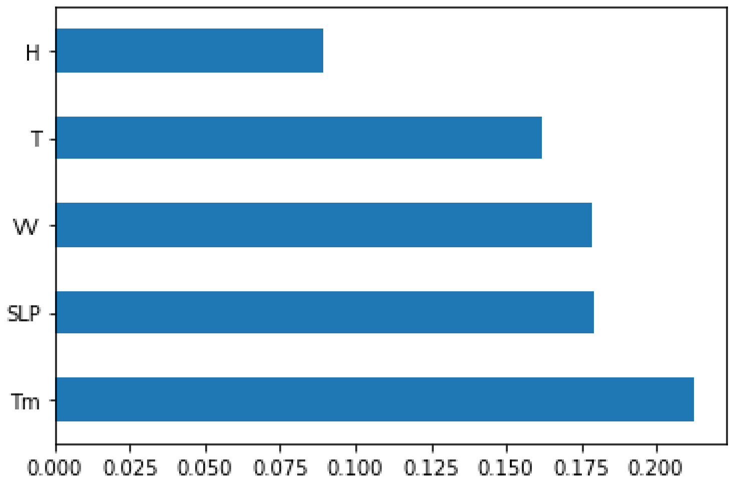

Fig. 4

Linear Regression

In [22]:

```
sns.distplot(y)
```

Out[22]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e011961288>
```

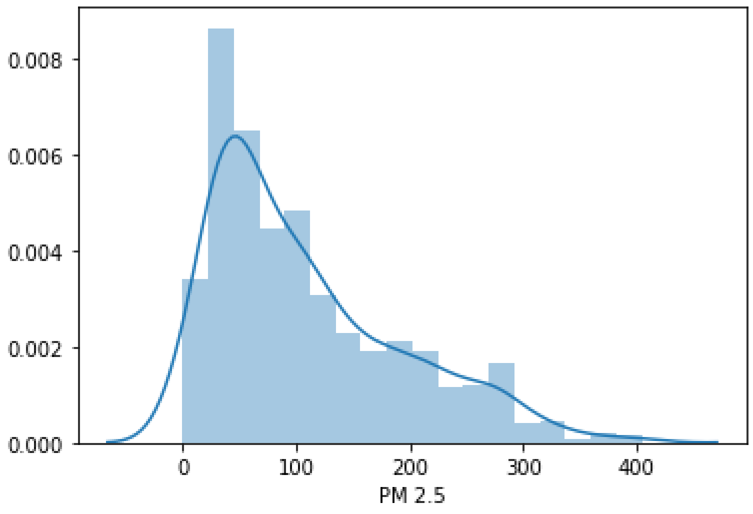

Fig. 5

Train Test Split

In [23]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)
```

In [24]:

```
from sklearn.linear_model import LinearRegression
```

In [25]:

```
regressor=LinearRegression()
regressor.fit(X_train,y_train)
```

Out[25]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [26]:

```
regressor.coef_
```

Out[26]:

```
array([ -2.69090829,   0.46219975,  -3.86705184,  -0.04494855,
        -1.21193616, -40.11490762,  -2.53563257,   0.56148181])
```

In [27]:

```
regressor.intercept_
```

Out[27]:

```
448.1161696758912
```

In [28]:

```
print("Coefficient of determination R^2 <-- on train set: {}".format(regressor.score(X_train, y_train)))
```

```
Coefficient of determination R^2 <-- on train set: 0.551516808175875
```

In [29]:

```
print("Coefficient of determination R^2 <-- on train set: {}".format(regressor.score(X_test, y_test)))
```

```
Coefficient of determination R^2 <-- on train set: 0.48525331308567876
```

In [30]:

```
from sklearn.model_selection import cross_val_score
score=cross_val_score(regressor,X,y,cv=5)
```

In [31]:

```
score.mean()
```

Out[31]:

```
0.4710569304807393
```

In [32]:

```
coeff_df = pd.DataFrame(regressor.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

Out[32]:

Table 5

	Coefficient
T	-2.690908
TM	0.462200
Tm	-3.867052
SLP	-0.044949
H	-1.211936
VV	-40.114908
V	-2.535633
VM	0.561482

Interpreting the coefficients:
Holding all other features fixed, a 1 unit increase in T is associated with an decrease of 2.690 in AQI PM2.5 . Holding all other features fixed, a 1 unit increase in TM is associated with an increase of 0.46 in AQI PM 2.5 .

In [34]:

```
prediction=regressor.predict(X_test)
sns.distplot(y_test-prediction)
```

Out[34]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1e0116f1048>
```

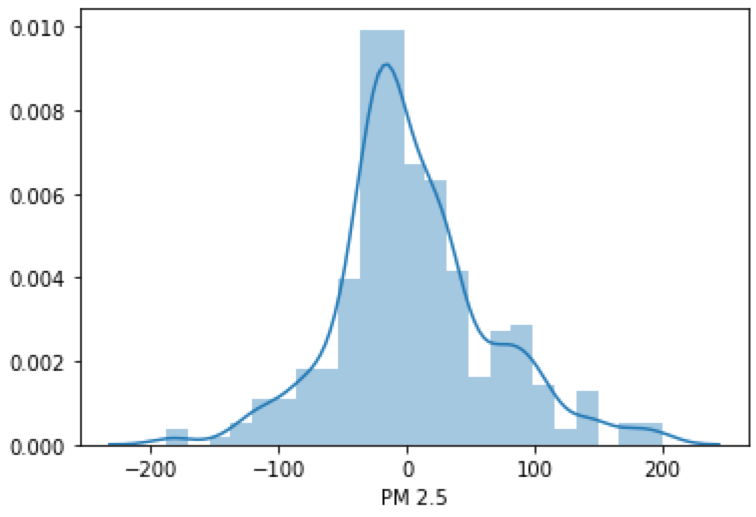

Fig. 6

In [35]:

```
plt.scatter(y_test,prediction)
```

Out[35]:

```
<matplotlib.collections.PathCollection at 0x1e0117e4c88>
```

In [35]:

```
plt.scatter(y_test,prediction)
```

Out[35]:

```
<matplotlib.collections.PathCollection at 0x1e0117e4c88>
```

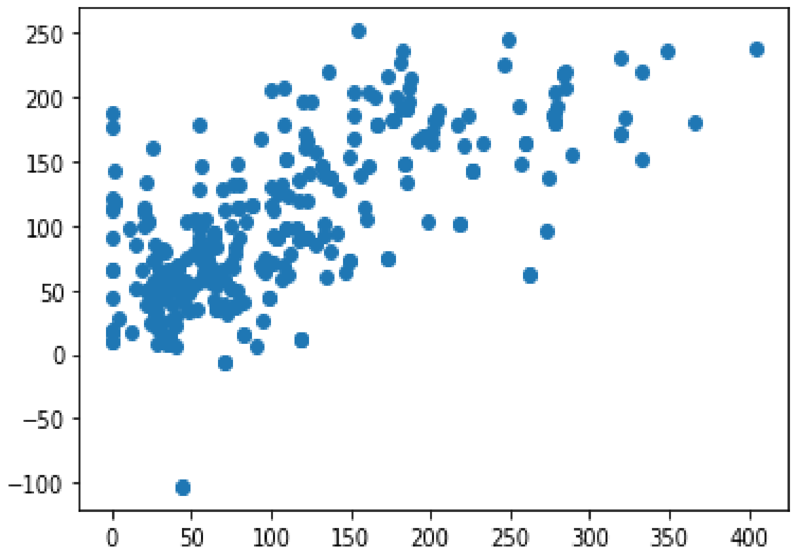

Fig. 7

Regression Evaluation Metrics

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

Mean Squared Error (MSE) is the mean of the squared errors: $$\frac 1n\sum_{i=1}^n(y_i-\hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

Comparing these metrics:

MAE is the most convenient to clarify, because it's the average error. MSE is much more in demand than MAE, because MSE rectifies larger errors, which reveals to be very useful in today's world. RMSE is much more popular than MSE, because RMSE is interpretable in the "y" units. All of these are loss functions, because we want to lower them.

In [36]:

```
from sklearn import metrics
```

In [37]:

```
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

```
MAE: 44.83624126628639
MSE: 3687.5430309324192
RMSE: 60.725143317512384
```

In [38]:

```
import pickle
```

In [39]:

```
file = open('regression_model.pkl', 'wb')

pickle.dump(regressor, file)
```

In [ ]:

```
##filename= fileread(r'C:\Users\Soumya')
pickle_out=open("regressor",'rb')
data = pickle.load(pickle_out)
print(data)
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

The Mean Absolute Error(MAE) was found to be 44.84 approximately,Mean Squared Error(MSE)was found to be 3687.54 approximately,Root Mean Squared error(RMSE) was found to be 60.73 approximately.The Heatmap has been plotted to predict the familiar features of the target variable using the seaborn library .Pickle module is implemented to convert the regressor file(object)into byte stream and dump function is used to dump the converted byte stream of the file"regressoion_model.pkl"into a file for further evaluation.After the processing and evaluation of the dumped file ,the byte stream file is again converted into object hierarchy using load function.

V. CONCLUSION

This paper provides a research analysis on the estimation of Air Quality Index.The heatmap is generated to predict the similar features of the target variable.Root Mean Squared error(RMSE) was found to be the highest priority for the learning of the model.Pickling and Unpickling played the role as an intermediate for the evaluation process of the Air Quality Index(AQI).

REFERENCES

[1] https://seaborn.pydata.org/generated/seaborn.heatmap.html. Heat Map correlation and plotting
[2] https://seaborn.pydata.org/introduction.html. Implementation of Seaborn Library module
[3] https://en.tutiempo.net/climate/india.html. Dataset has been downloaded from this website and implemented for Research
[4]https://www.datacamp.com/community/tutorials/pickle-python-tutorial?utm_source=adwords_ppc&utm_campaignid=10267161064&utm_adgroupid=102842301792&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=332602034358&utm_targetid=aud-299261629574:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=9061792&gclid=Cj0KCQjwo6D4BRDgARIsAA6uN1_TSGS3sO7w32Itj0gHy4FyQpjbTL54P0Qz-0rZ2y63NZnpT_PLFjIaAueoEALw_wcB. Imported Pickle module.
[5]https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html#:~:text=An%20extra-trees%20regressor.,accuracy%20and%20control%20over-fitting.&text=If%20int%2C%20then%20consider%20min_samples_split%20as%20the%20minimum%20number. Extra Regressor Classifier
[6] https://machinelearningmastery.com/feature-selection-machine-learning-python/Feature Selection
[7] https://towardsdatascience.com/interpreting-the-coefficients-of-linear-regression-cc31d4c6f235. Interpreting coefficients
[8] **https://scikit-learn.org/stable/modules/model_evaluation.html.** Regression evaluation Metric
[9] **Github link:- https://github.com/Soumyajit567/Air-Quality-Index.** This is my GitHub project link. The code is done in Jupyter Notebook and uploaded to GitHub.

AUTHORS PROFILE

Mr. Soumyajit Chakraborty currently pursuing Bachelors in Technology under the department of Computer Science and Engineering at University of Engineering and Management Kolkata. He is also working as a undergraduate research assistant at our university. His research interests are Machine Learning, Deep Learning and Data Science.

Mr. Koustav Guha is currently pursuing B.Tech in Computer Science and Engineering from University of Engineering and Management Kolkata. His research areas include Machine Learning, Deep Learning and Computer Vision. Currently working as a undergraduate research assistant at University of Engineering and Management Kolkata.