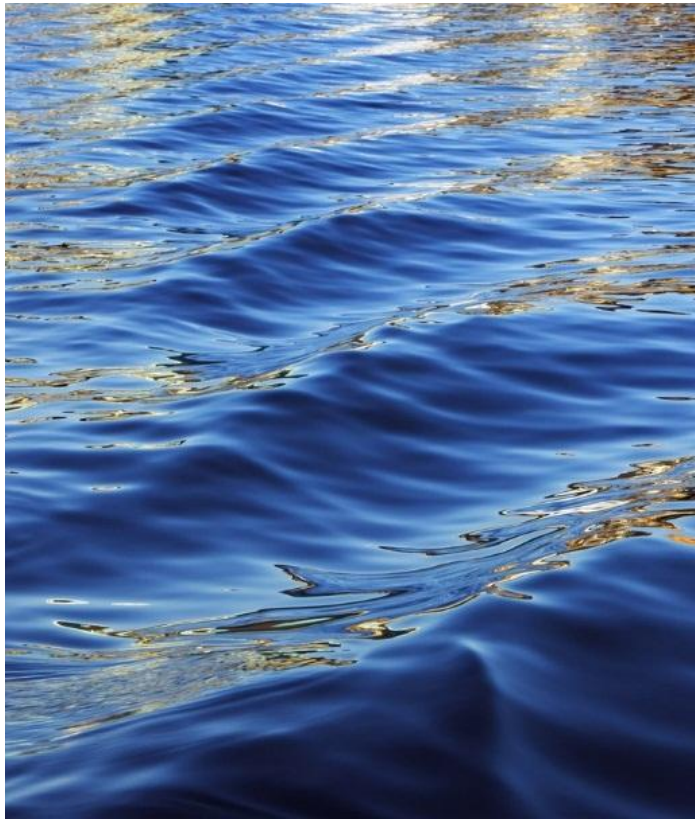
Calibrating PM 2.5 Values From Low-Cost Sensors using values from Reference Grade Monitors

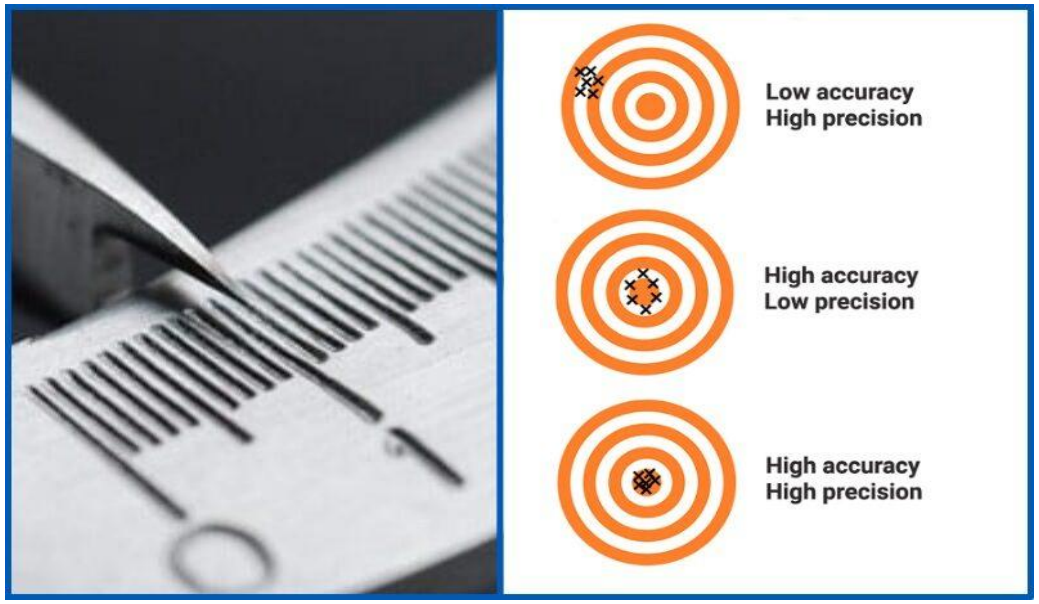
Using Machine Learning Methods



Name: Kunal Jalindar Gaikwad
Date: 12/13/2022

Calibration

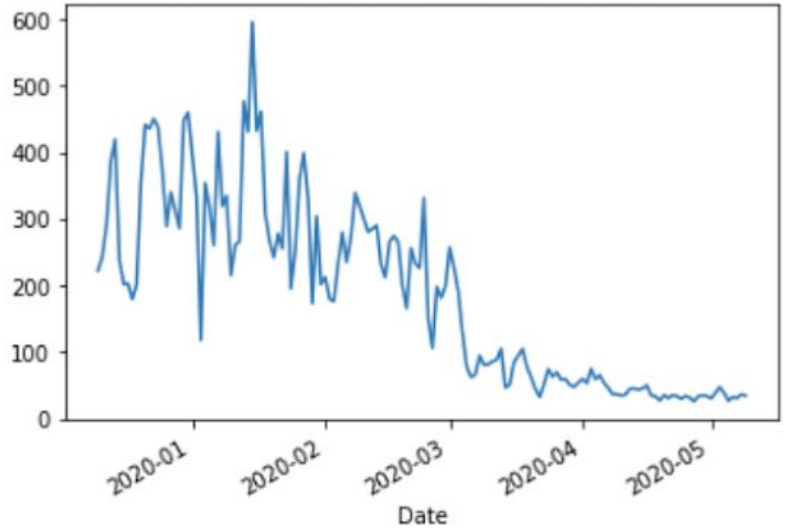
Calibration is the comparison of measurement values delivered by a device under test with those of a calibration standard of known accuracy. Such a standard could be another measurement device of known accuracy,

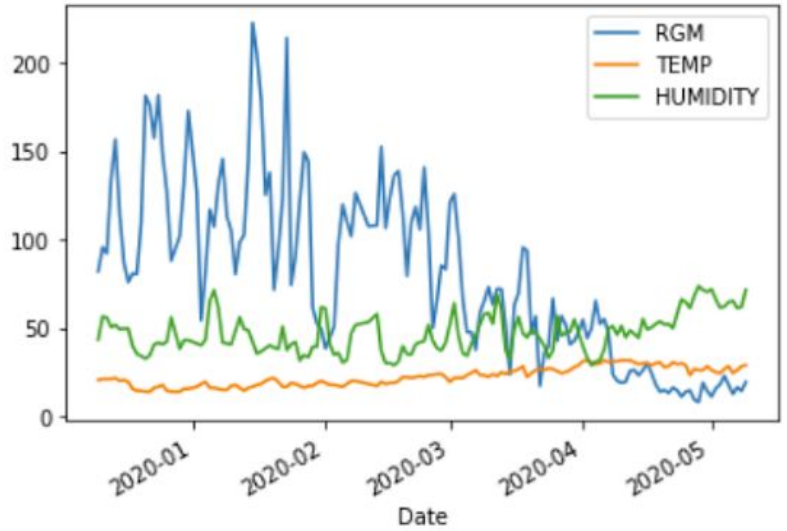


A reverse process to regression, where instead of a future dependent variable being predicted from known explanatory variables, a known observation of the dependent variables is used to predict a corresponding explanatory variable.

Image ref:-
https://blog.wika.com/knowhow/verification-or-calibration/?doing_wp_cron=1671143669.2394099235534667968750
https://soluzionesolare.com/guides/difference-between-calibration-adjustment-accuracy-and-precision/

Visualizing the Data





LCS - Predictant

Model

RGM Values, Temperature, Relative Humidity - Predictor

Time Period of Dataset:- Dec 12, 2019 to May 15 , 2020

Learning Calibration

- Datasets used :
  Low-Cost Sensor (LCS) Hourly Dataset
  Reference Grade Monitor (RGM) Daily Dataset
  For Location

- The problem of Calibration of Low-Cost Sensors is as follows

  Given 'n' days:

  (a) $\mathbf{W} \in \mathbf{R}^{n \times d1}$ ... d1 = features describing daily weather and RGM values.

  (b) $\mathbf{Z} \in \mathbf{R}^{n}$ ... column vector with LCS as features.

  Objective:

  At a given timestep t, predict the LCS value matrix $\mathbf{Z} \in \mathbf{R}^{n}$, given the temperature and RGM values matrix $\mathbf{W} \in \mathbf{R}^{n \times d1}$ at that time.

  Solution:

  Estimate a function, **F(Wi) = Zi**

Image ref:-
https://www.coleparmer.com/i/davis-instruments-6162-plus-wireless-weather-station-uv-solar/

Evaluating the Performance of Models

- The performance metrics used in the study are as follows:-

1. Root Mean Square Error & Bias-corrected mean-normalized RMSE

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}\left(c_{estimated,i} - c_{true,i}\right)^2}{n}} \qquad nRMSE = \frac{\sqrt{\sum_{i=1}^{n}\left(c_{estimated,i} - n_{bias} - c_{true,i}\right)^2}}{\sum_{i=1}^{n}\left(c_{true,i}\right)}$$

2. R2 score (Coefficient of determination)

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2 \qquad SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2 \qquad R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$$

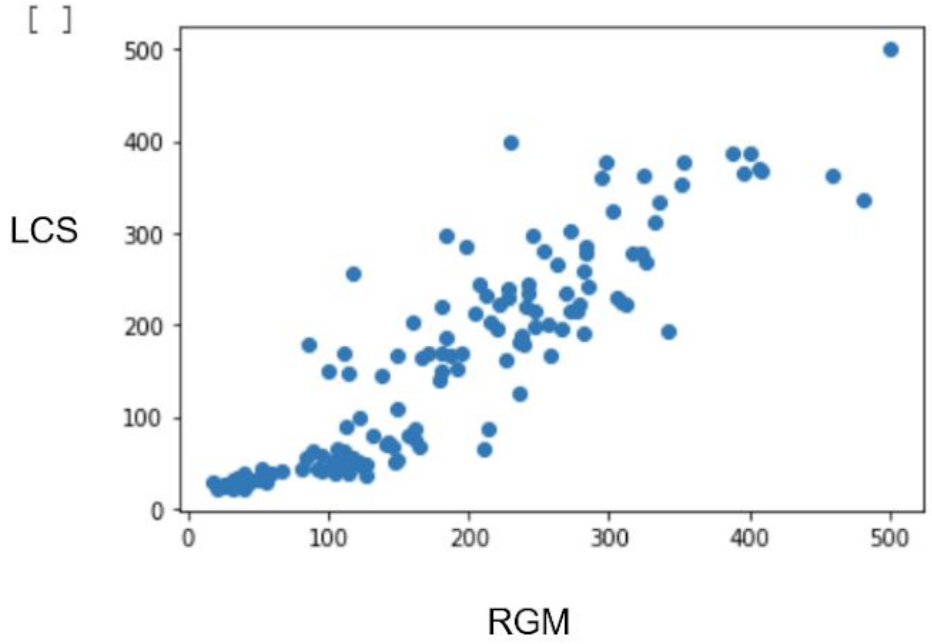
3. Mean Absolute Error & Bias-corrected mean-normalized MAE

$$MAE = \frac{\sum_{i=1}^{n}\left|c_{estimated,i} - c_{true,i}\right|}{n} \qquad CvMAE = \frac{\sum_{i=1}^{n}\left|c_{estimated,i} - n_{bias} - c_{true,i}\right|}{\sum_{i=1}^{n}\left(c_{true,i}\right)}$$

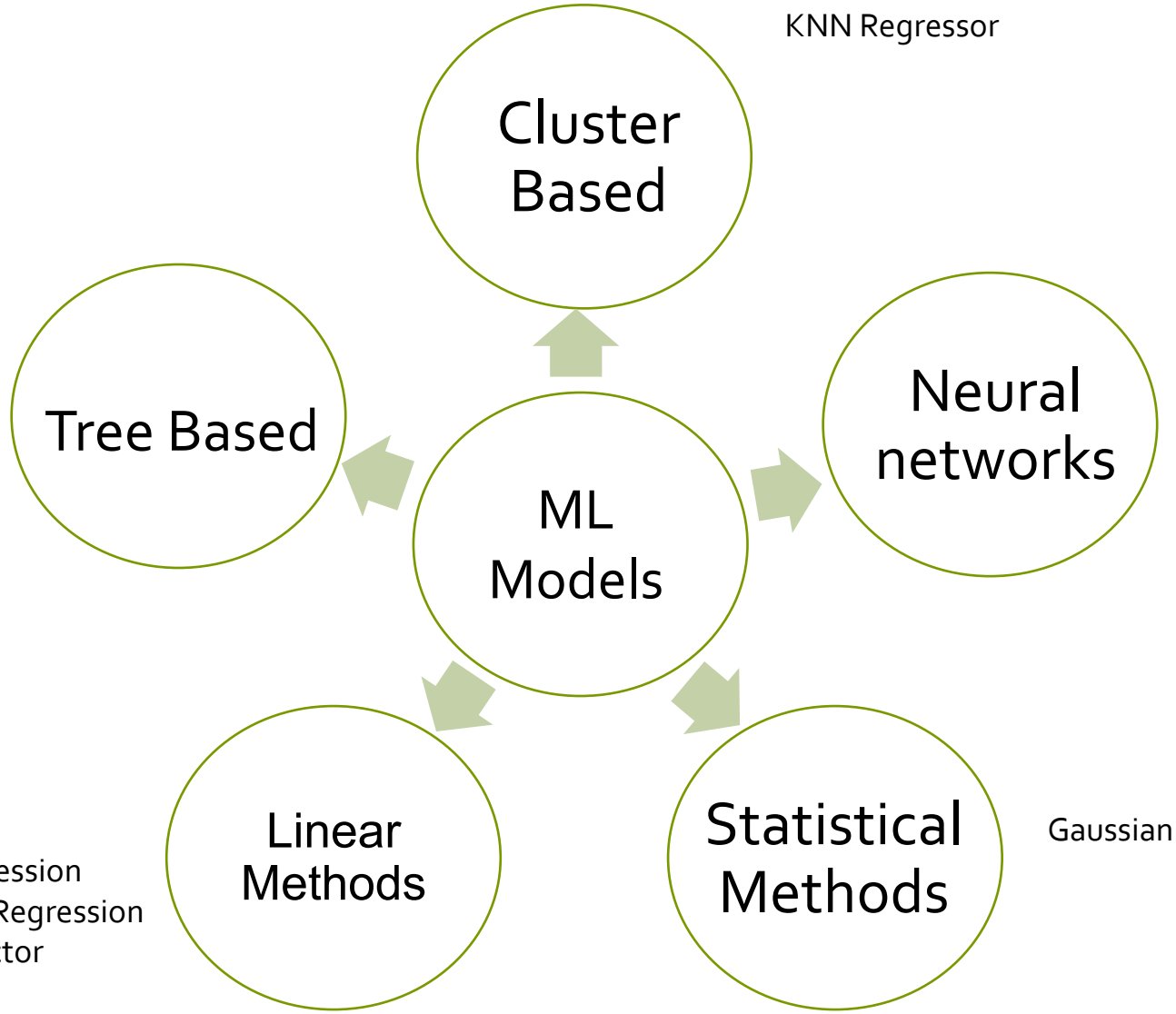
Image ref:-
https://www.sciencedirect.com/science/article/pii/S0021850221005644
https://en.wikipedia.org/wiki/Coefficient_of_determination

Preprocessing

- Preprocessing was done by converting both datasets to daily average values for LCS and RGM Values.
- Further, both datasets were scaled using Minmax scaling so that the values of both datasets range from 0 to 500.



Machine Learning Models



KNN Regressor

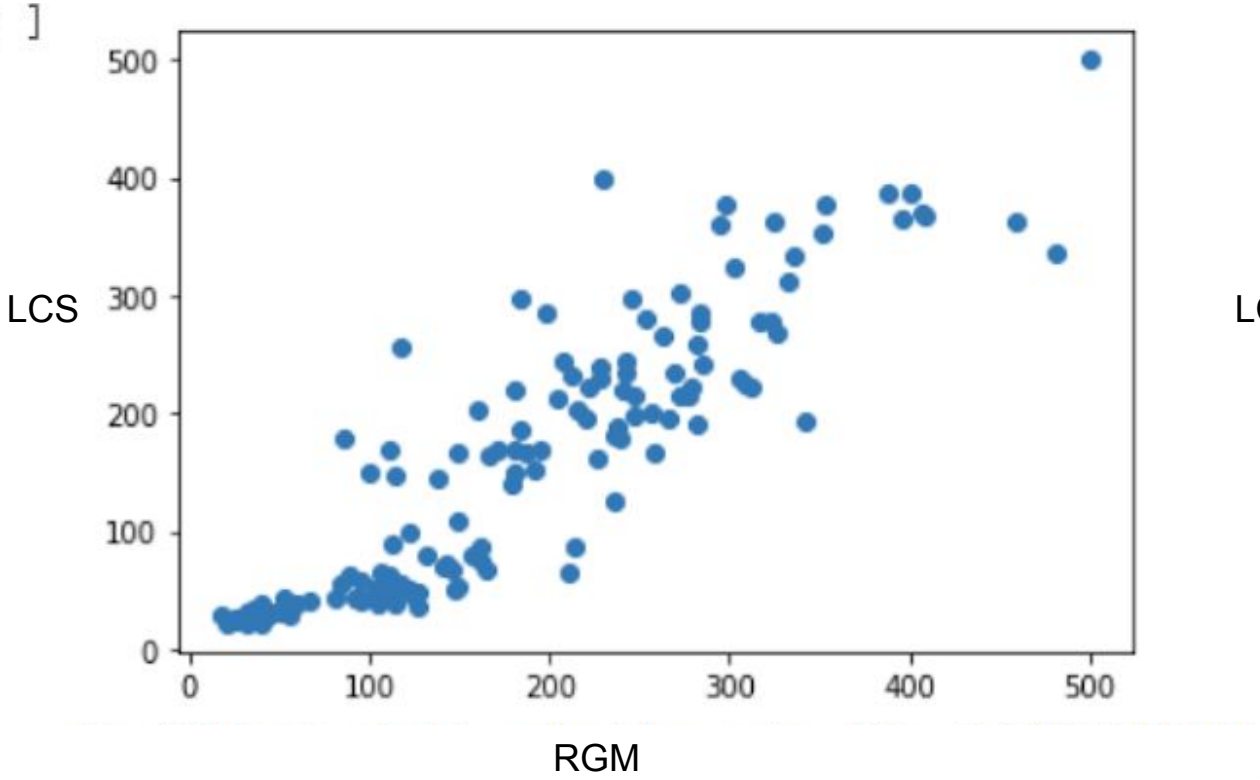
Decision Trees
Random Forests

Linear Regression
Elastic Net Regression
Support Vector Regression

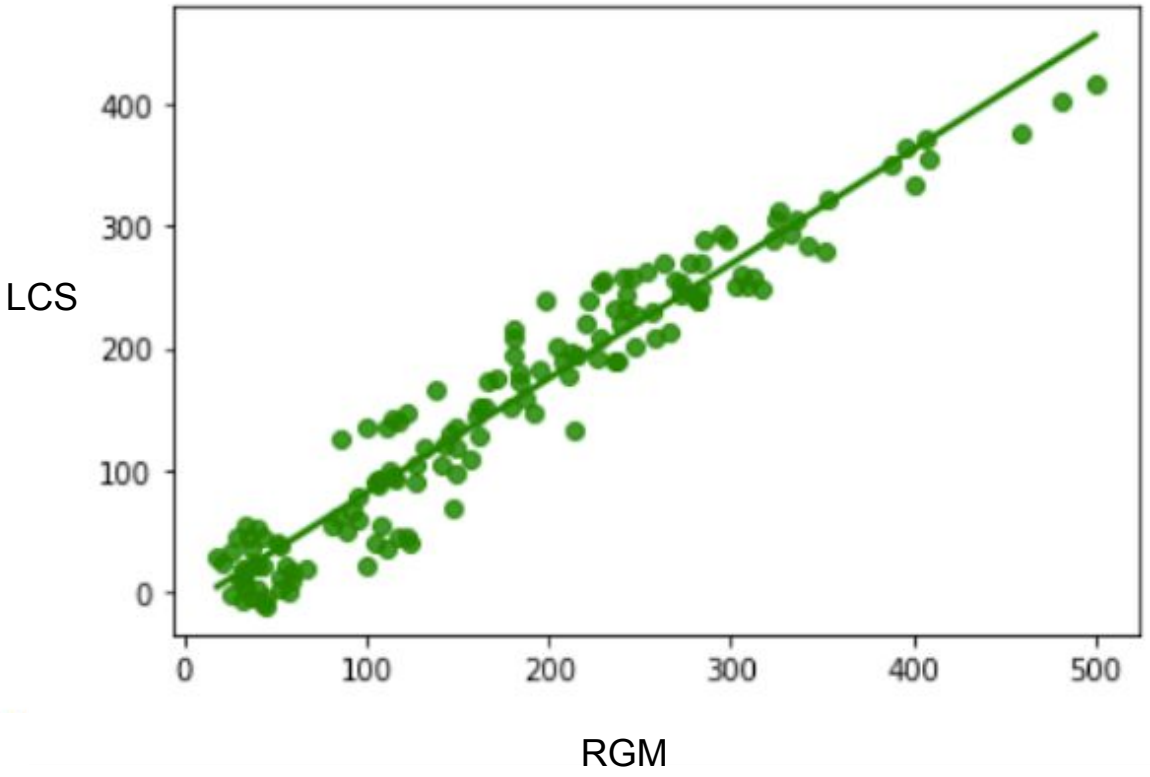
Gaussian process regressor

Multiple Linear Regression

[ ]



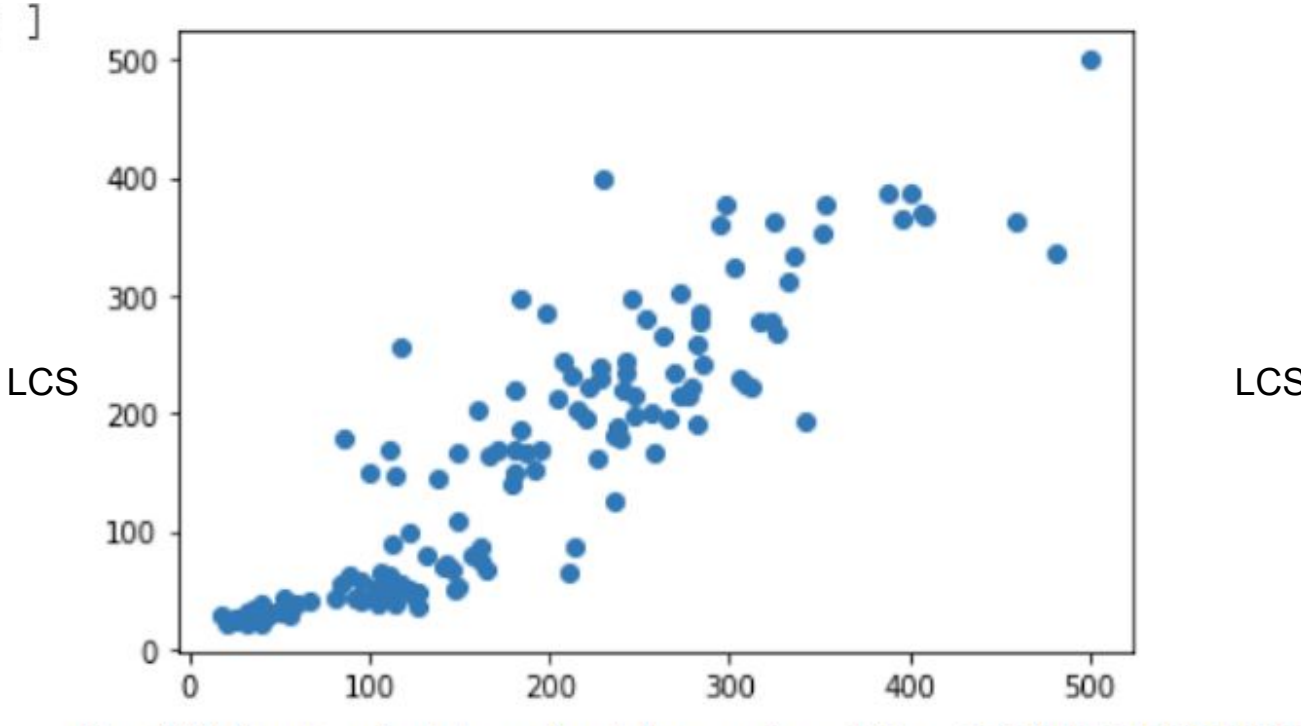
Uncalibrated data



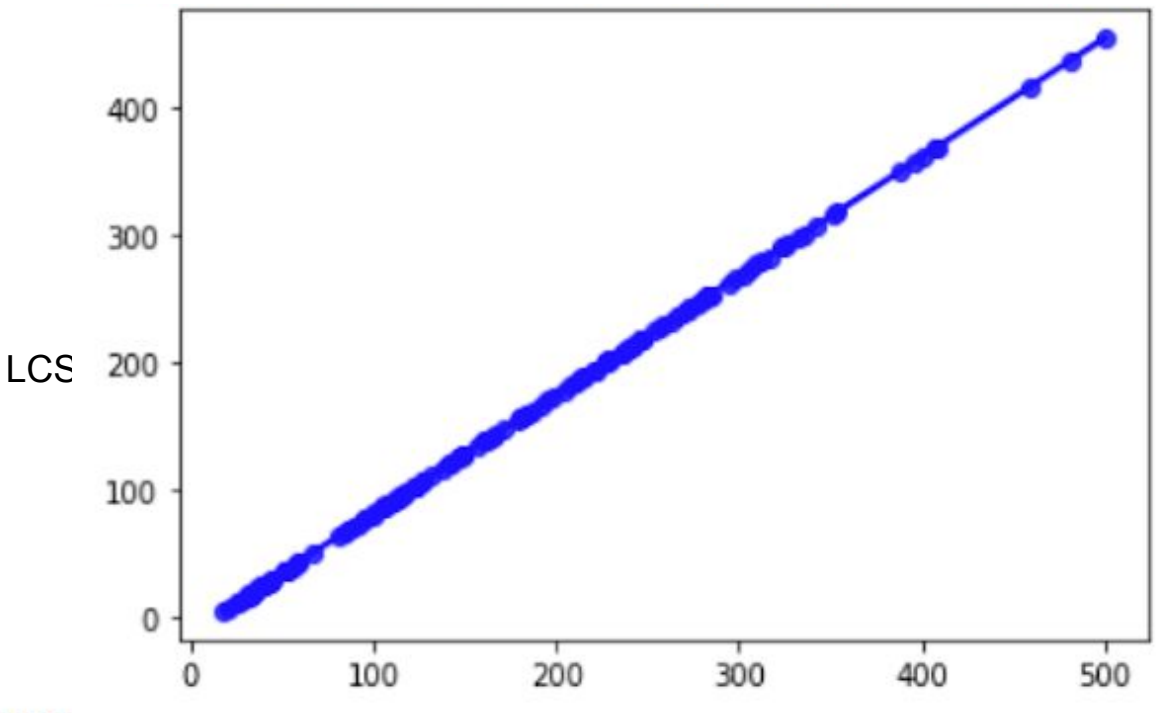
Calibrated using Multiple Linear Regression

Elastic Net Regression

[ ]



Uncalibrated data

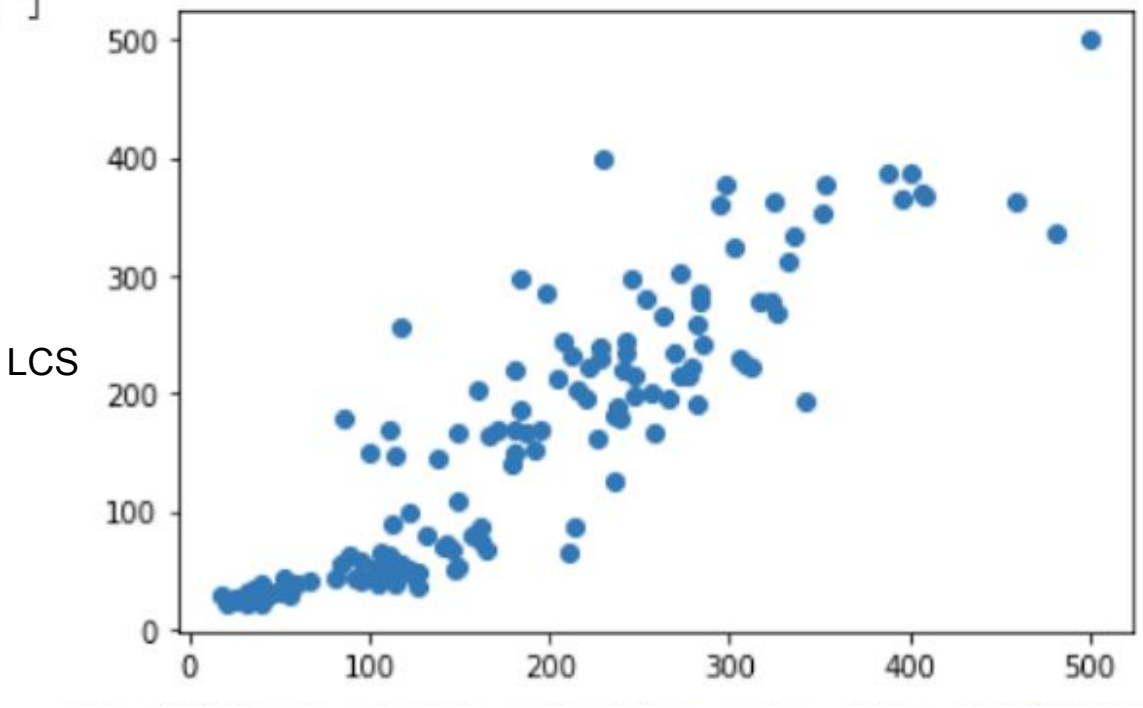


Calibrated using Elastic Net Regression

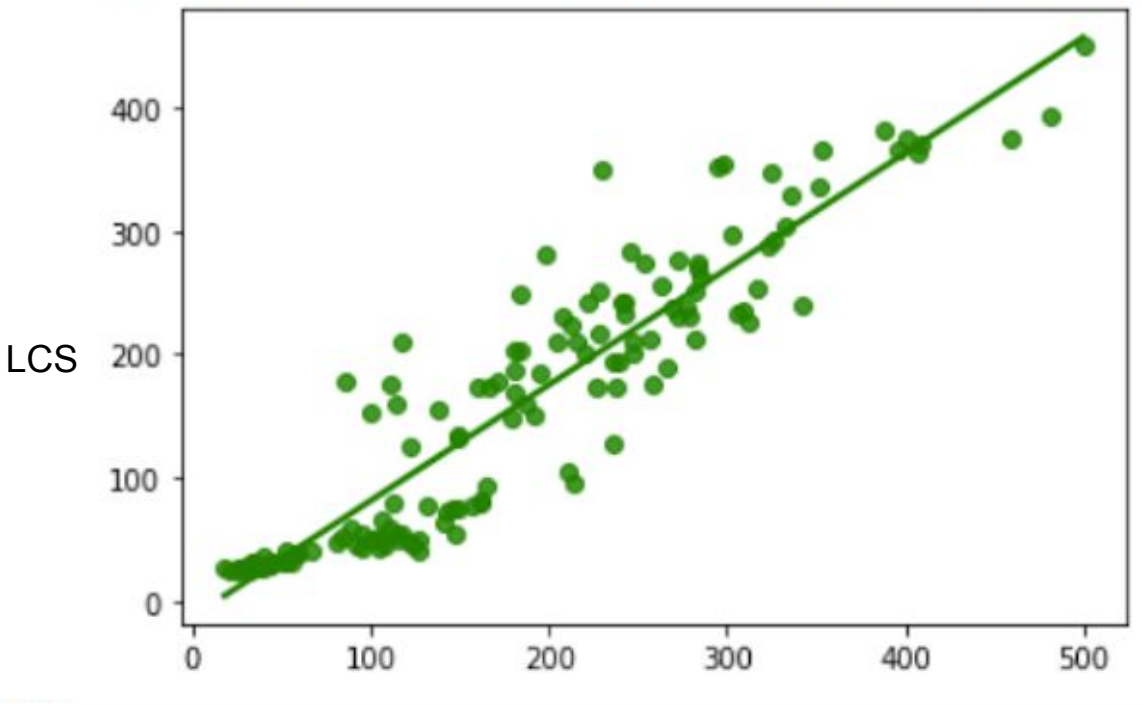
Hyperparameters -
```
{"max_iter": [1, 5, 10,25,100,200],
 "alpha": [0.0001, 0.001, 0.01, 0.1, 1, 10, 100],
 "l1_ratio": np.arange(0.0, 1.0, 0.01)}
```

Random Forest Regression

[ ]



Uncalibrated data



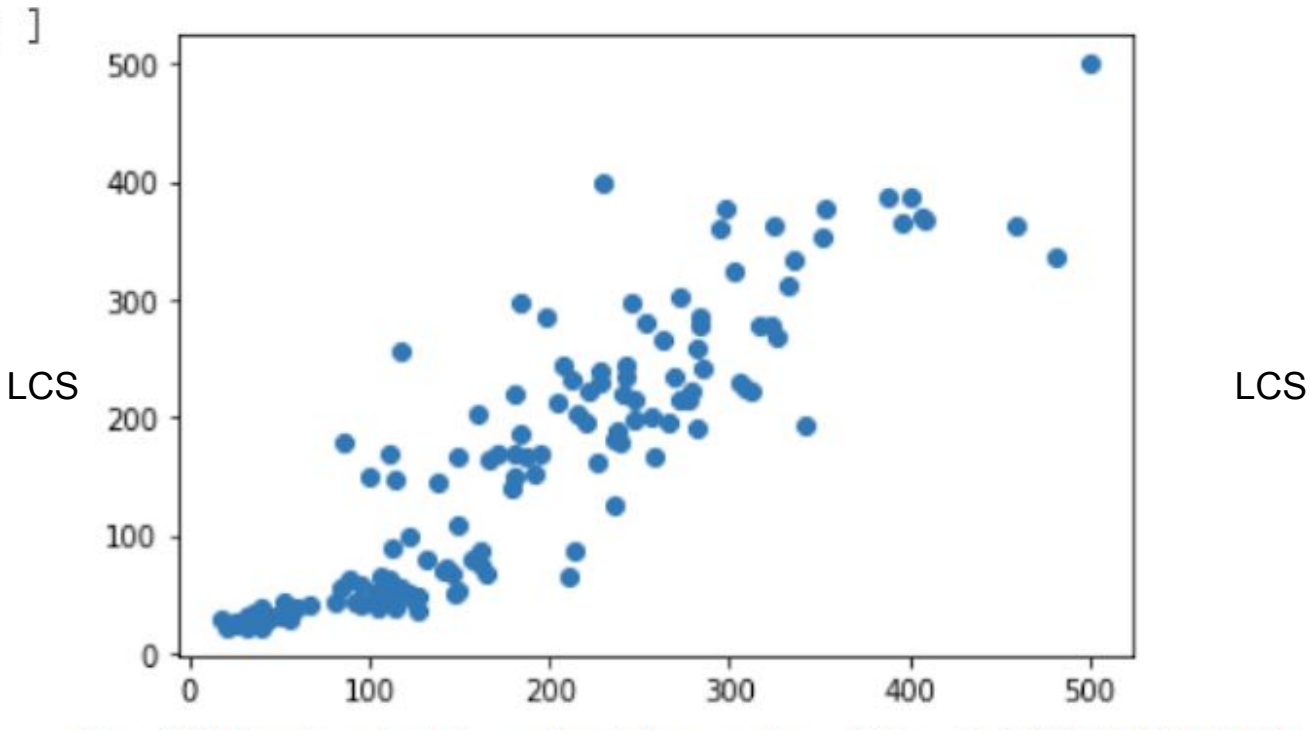
Calibrated using Random Forest Regression

Hyperparameters

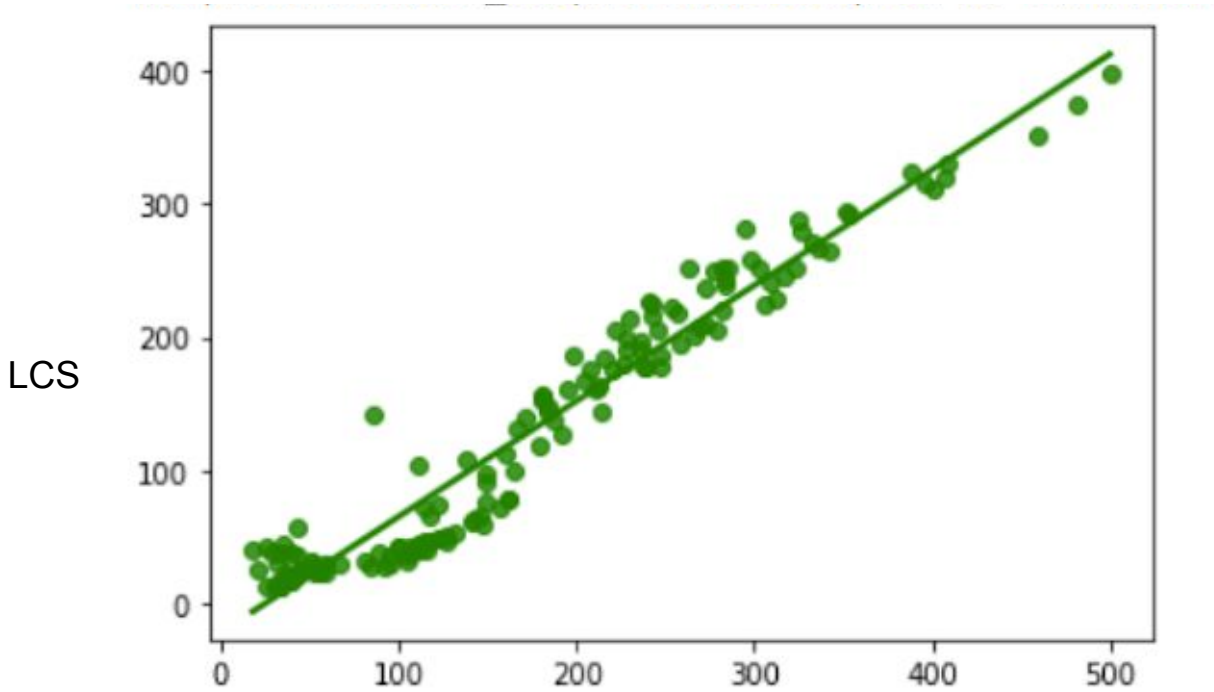
```
-,n_estimators1 = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)],max_features1 = ['auto', 'sqrt']
max_depth1 = [int(x) for x in np.linspace(10, 110, num = 11)],min_samples_split1 = [2, 5, 10],min_samples_leaf1 = [1, 2, 4]
,bootstrap1 = [True, False]
```

Neural Networks

[ ]



Uncalibrated data



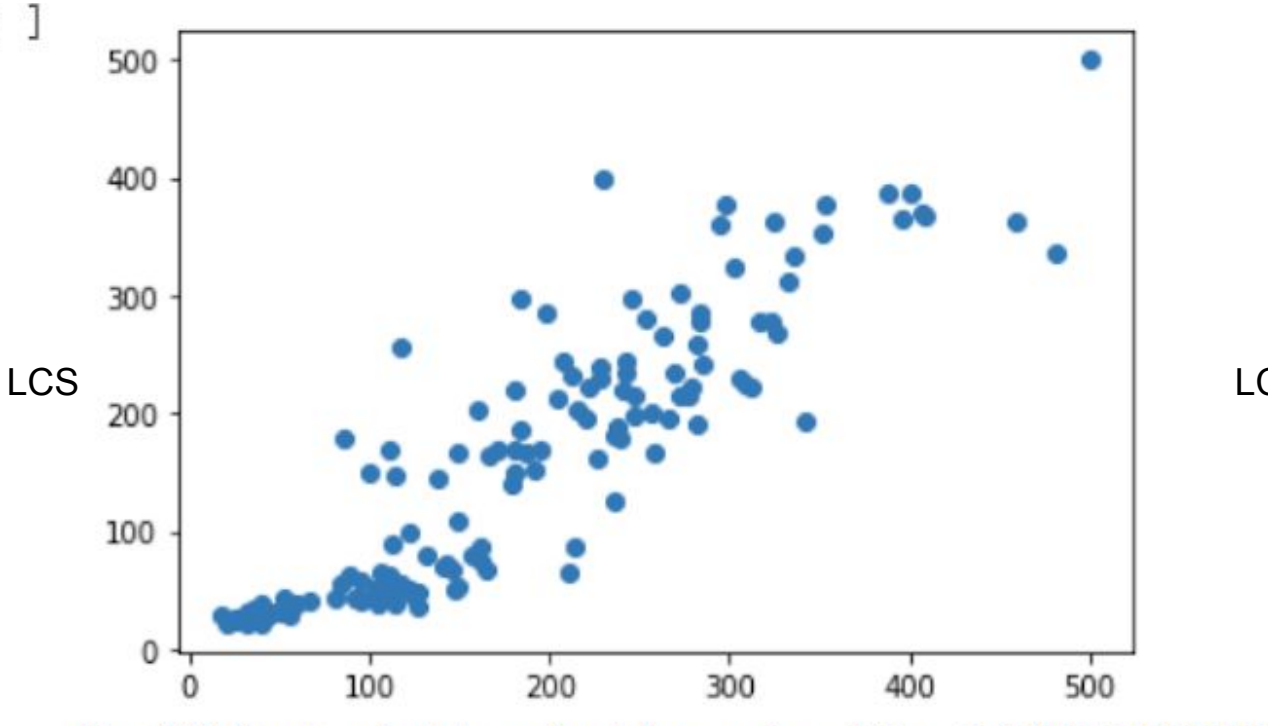
Calibrated using Neural Networks

Hyperparameters -

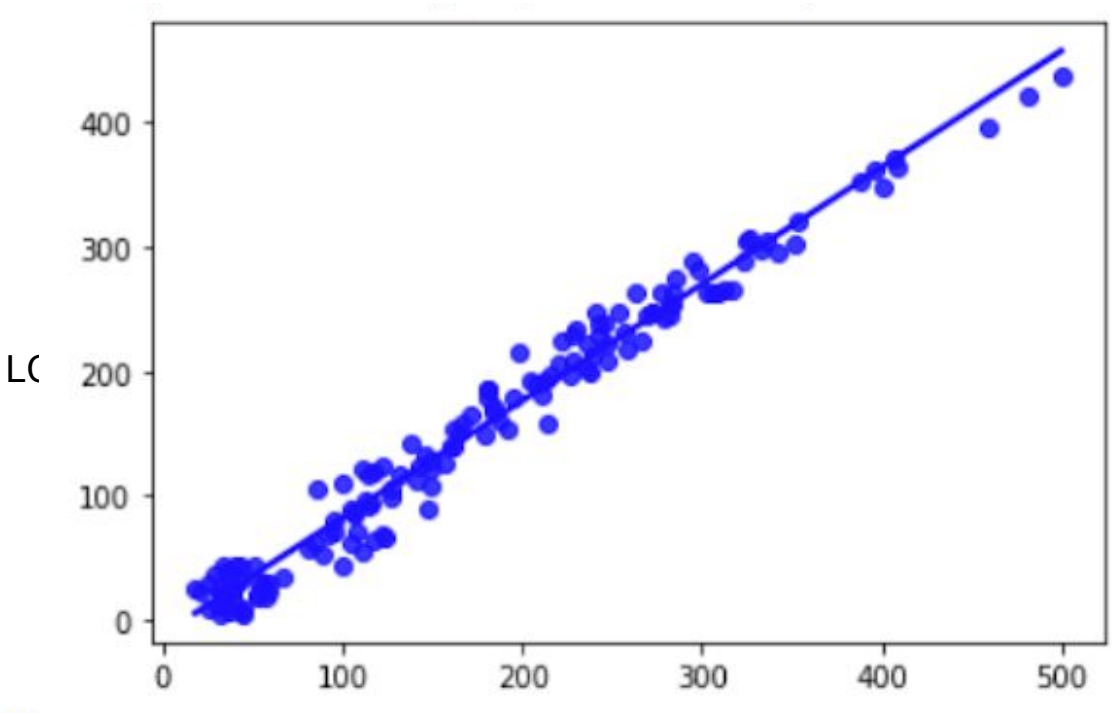
```
learning_rate=0.0001, beta_1=0.99, beta_2=0.999,epochs=50, batch_size=32,validation_
split=0.15, number_of_nodes_per_layer=200 , number_of_layers=9, activation= 'relu'
```

Support Vector Regression

[ ]



Uncalibrated data



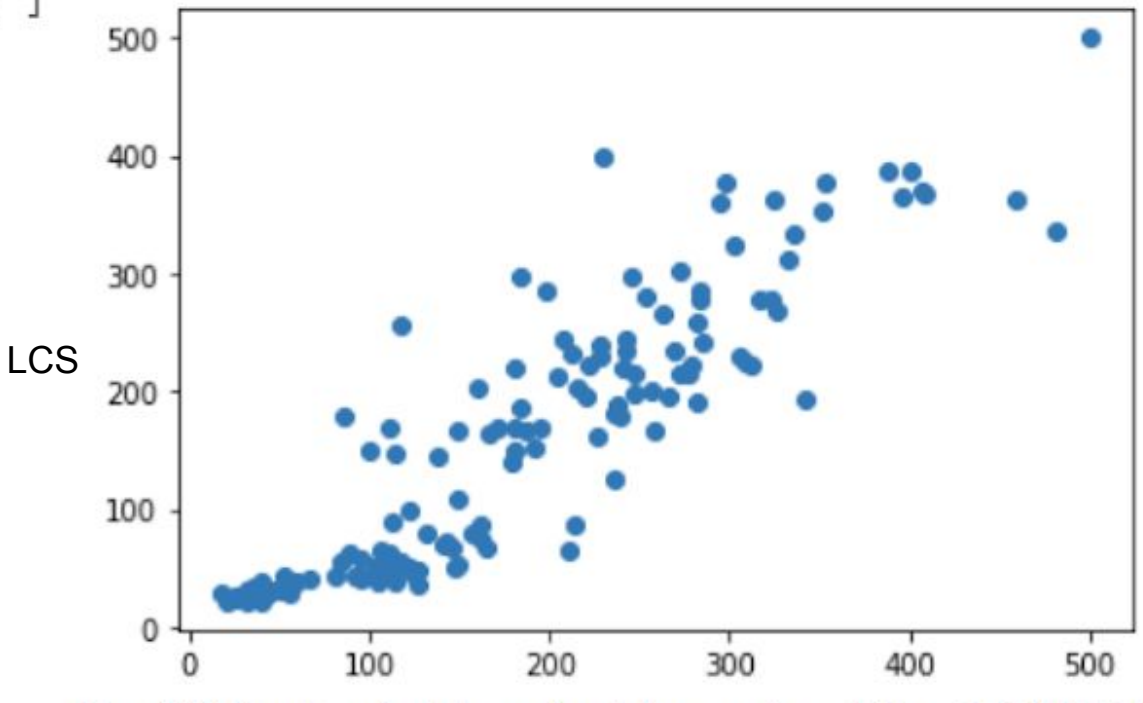
Calibrated Support Vector Regression

Hyperparameters -

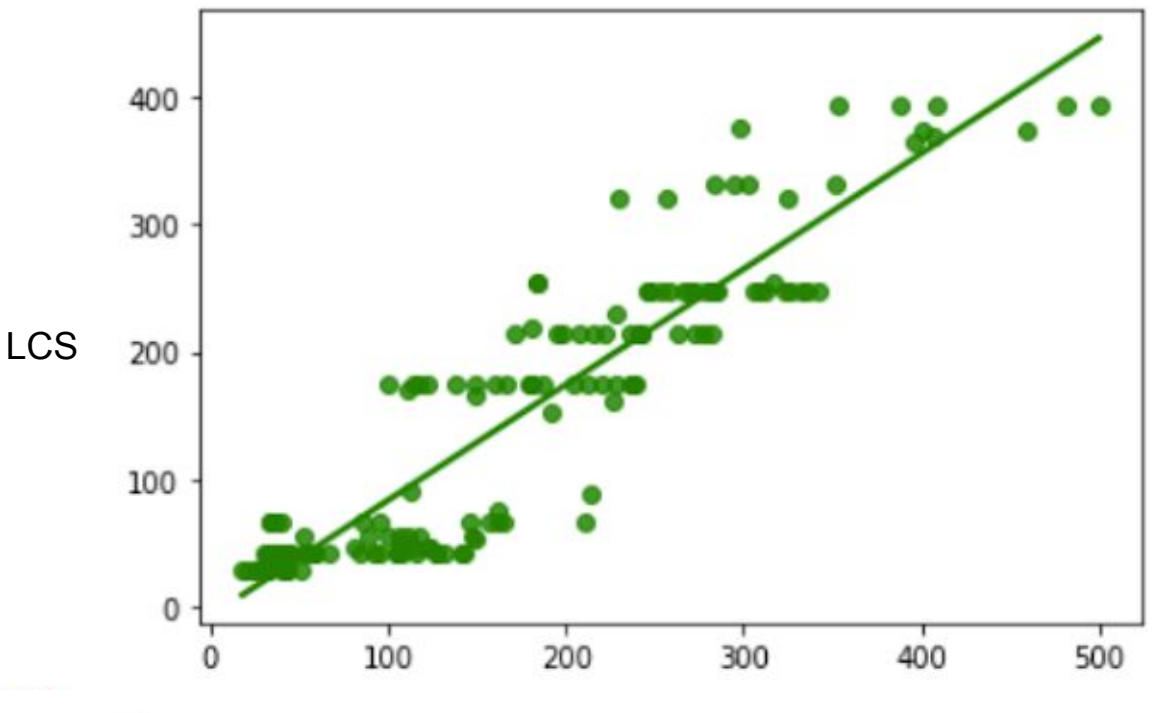
```
{'C': [0.1, 1, 10, 100, 1000],'gamma': [1, 0.1, 0.01, 0.001, 0.0001],'kernel': ['lin
ear']                }
```

Regression Trees

[ ]



Uncalibrated data



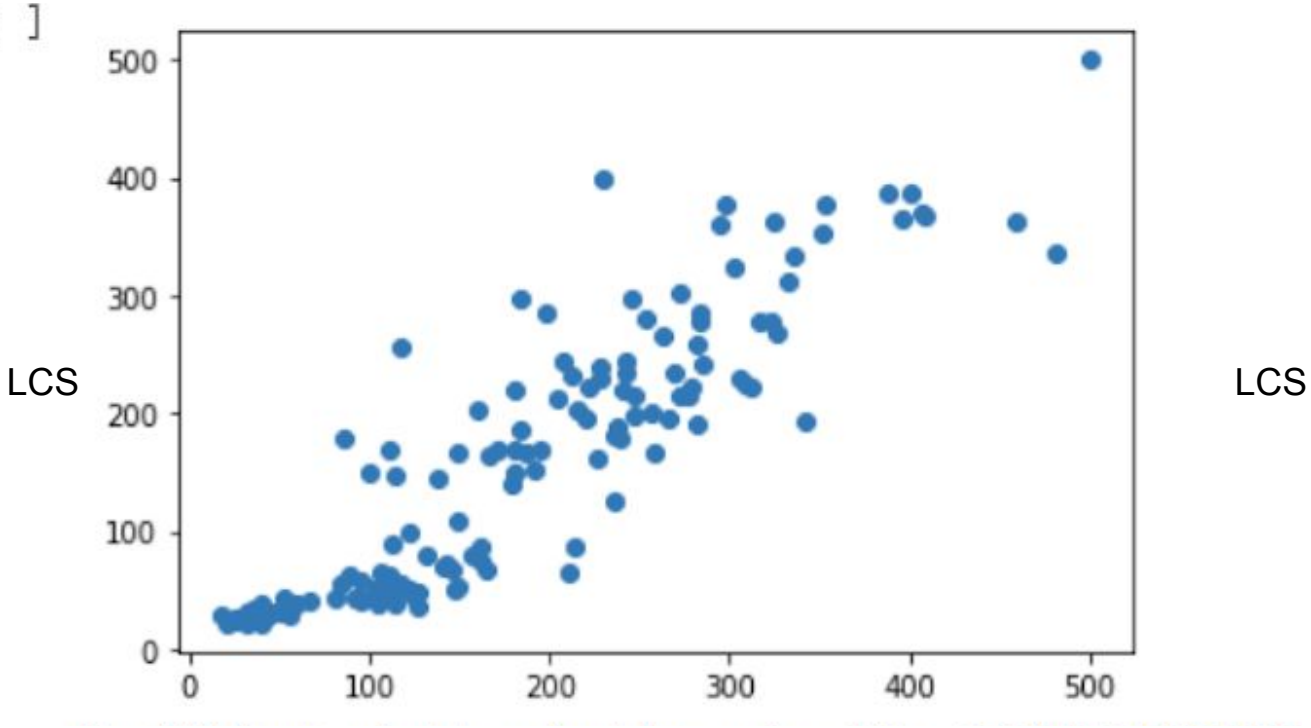
Calibrated using Regression Trees

Hyperparameters -

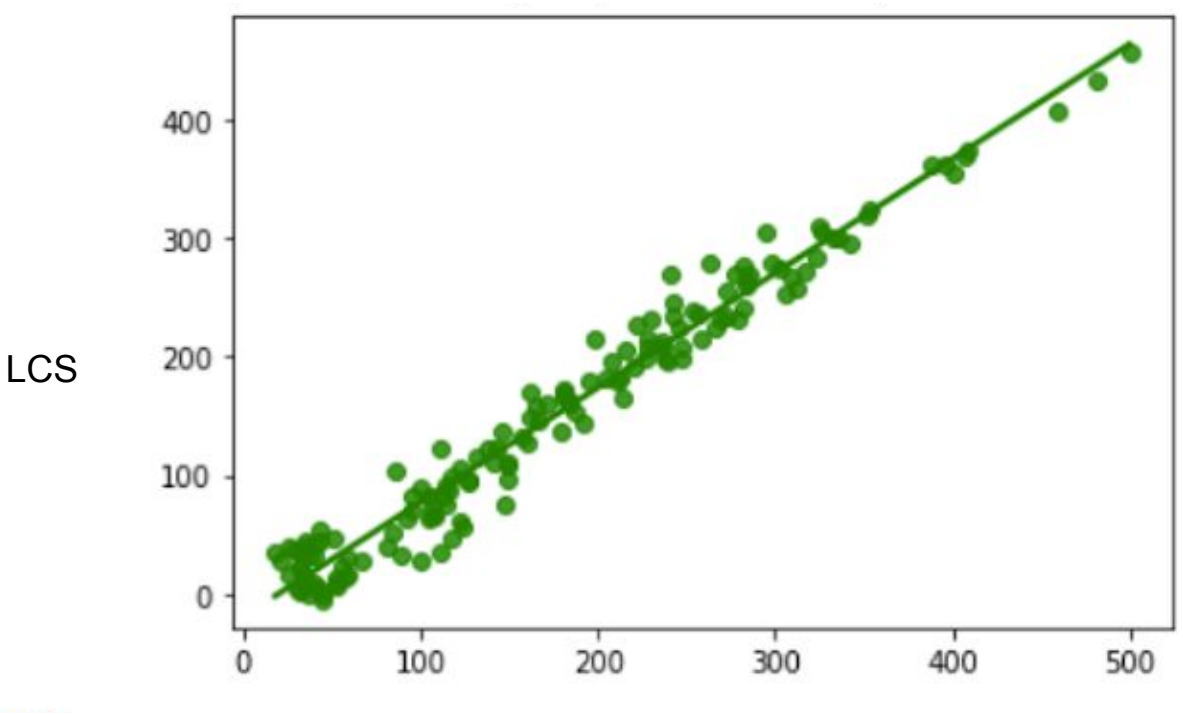
```
{'decisiontreeregressor__max_depth' : range(1,20),'decisiontreeregressor__max_features': range(1, 5),'decisiontreeregressor__min_samples_leaf' : range(1, 5)}
```

Kernel-Ridge Regression – Linear Kernel

[ ]



Uncalibrated data

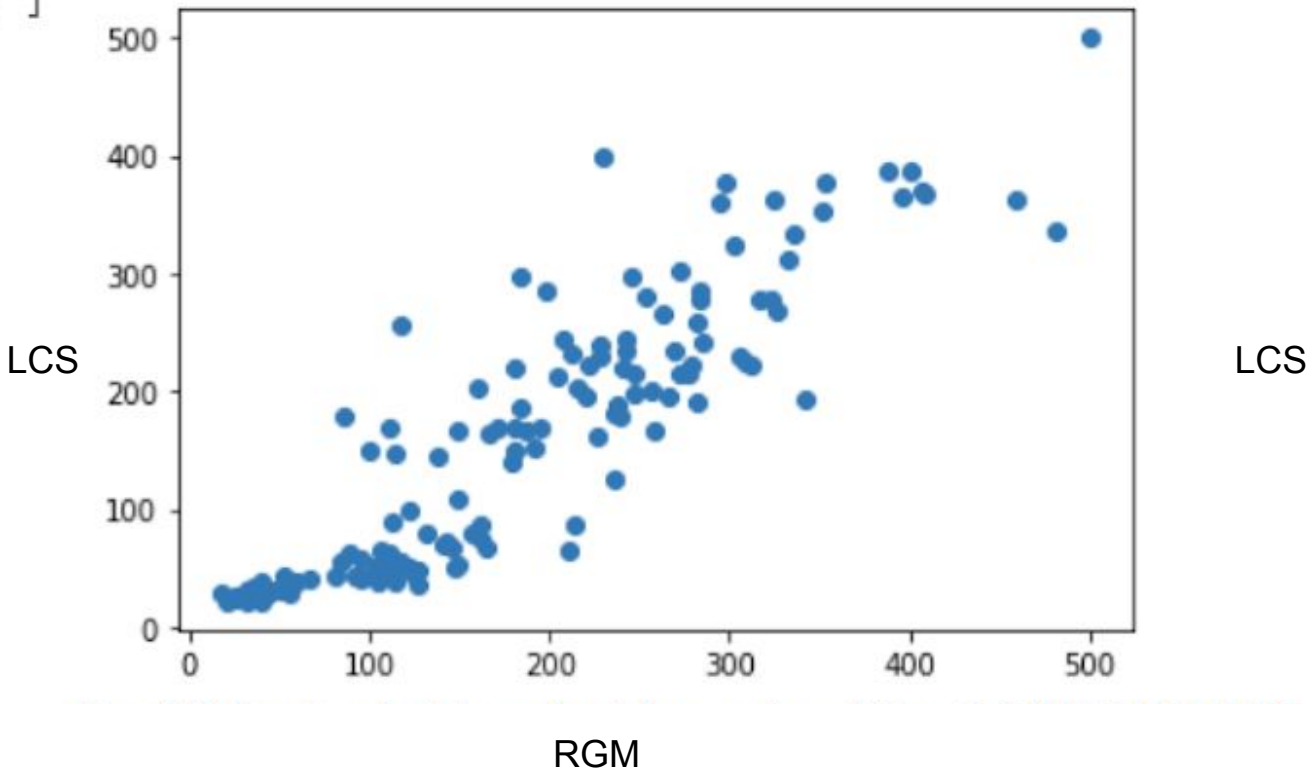


Calibrated using Kernel-Ridge Regression – Linear Kernel

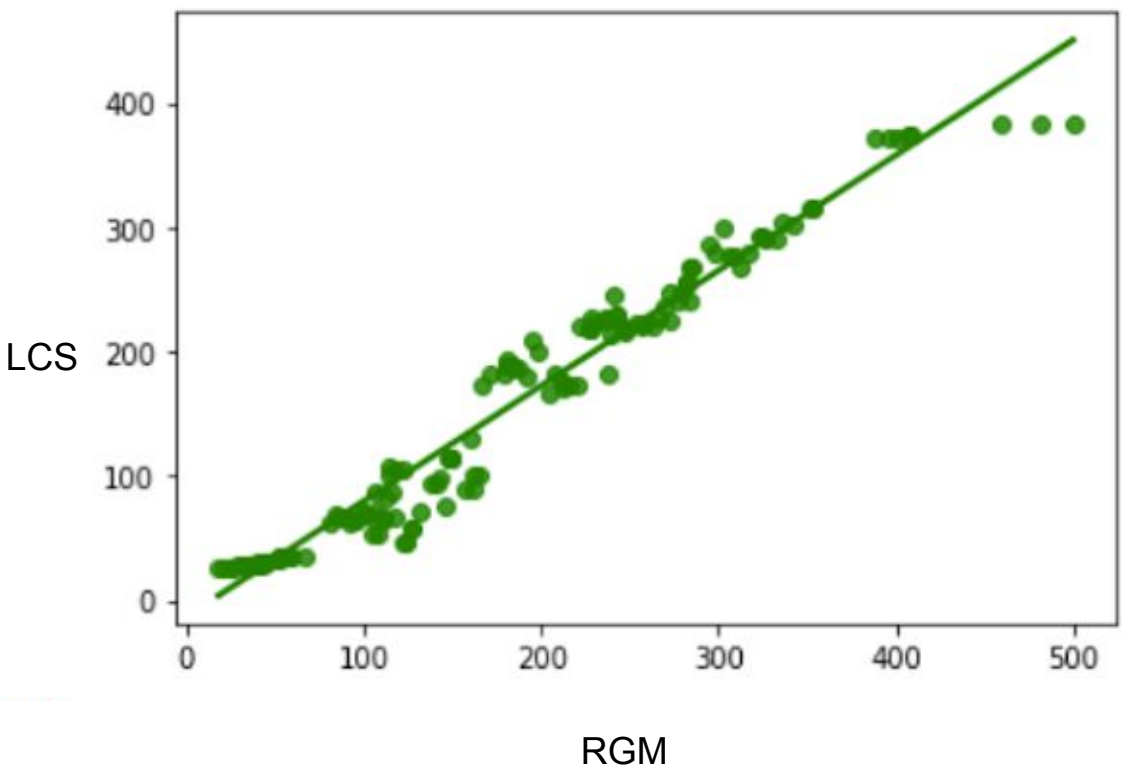
Hyperparameters - `alpha=1.52,kernel='linear'`

K-Nearest Neighbors Regression

[ ]



Uncalibrated data

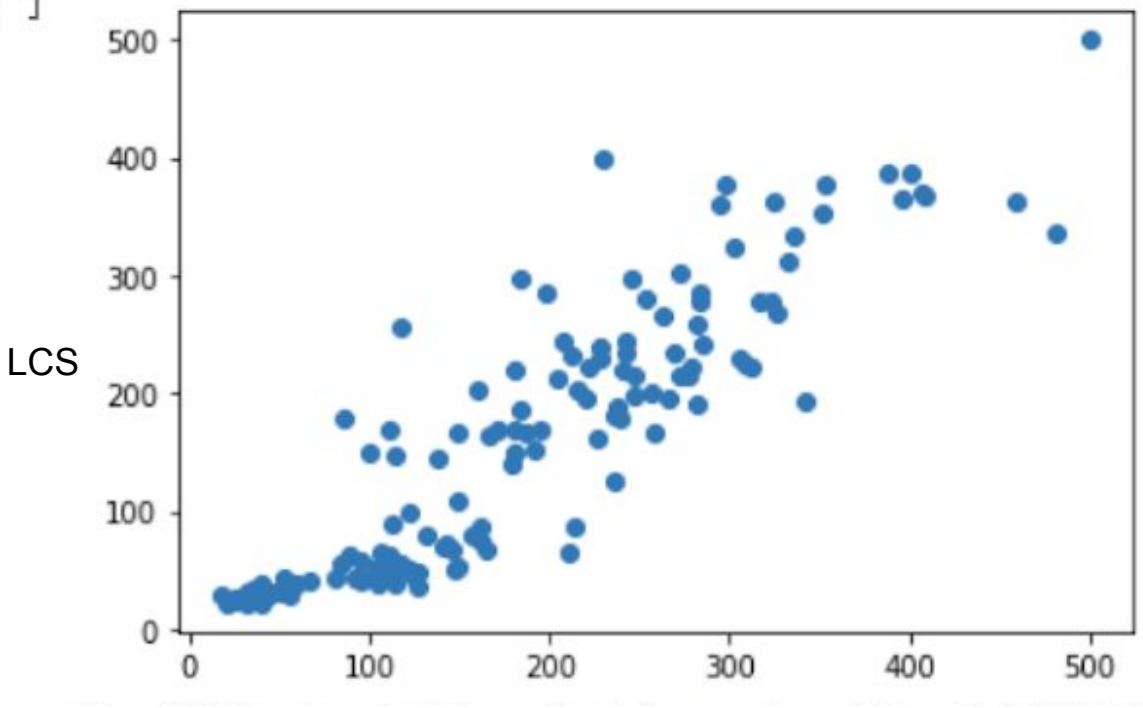


Calibrated using K-Nearest Neighbors Regression

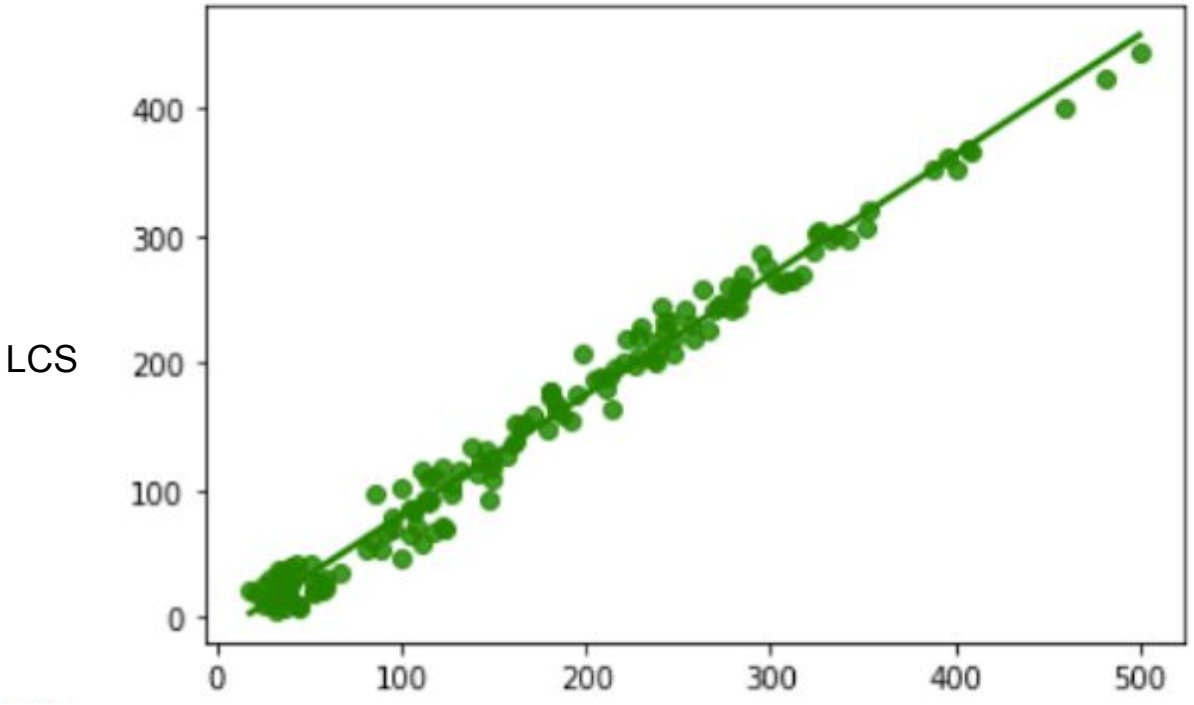
Hyperparameters -
```
leaf_size = list(range(1,10))
n_neighbors = list(range(1,10))
p=[1,2]
```

Gaussian Process Regression

[ ]



Uncalibrated data



Calibrated using Gaussian Process Regression

Hyperparameters -

```
kernel=DotProduct+WhiteKernel,random_state= 200,n_restarts_optimizer= 10,alpha=0.025
```

Results

Model	R2 Score	MAE	Bias-Corrected MAE	RMSE	Bias-Corrected RMSE
Multiple Linear	0.87835272073687	31.439077413532154	5.069825233983e-17	1639.243542146327	0.02189087949712553
Elastic net	0.82029833769955	35.702603897544	4.542563409648 9e-12	2421.548523101618	1.0227895096866129
Random Forest	0.9833503950963756	9.728650023110072	-68.68278527573638	224.35978414707242	1.1594472973548777
Neural Networks	0.8651274361231321	28.944263405188195	1.427876592080 35e-08	1817.459302724297	0.022598247652384 59
Support Vector	0.8417385563565789	31.20870164212426	200.95735156192654	1788.8450986230682	1.7069897756749888
Regression Trees	0.9939987222381 67	3.807009551195655	0.23984522983123	80.86950957967669	1.0745660505547563
Kernel Ridge	0.8365606921755824	32.5573688196638	4.101216669157 22e-17	1964.6199620671018	0.0242156247257755
KNN	0.8283995337062441	28.73734955220596	-1.026896264332e-17	1919.1974902793347	0.02396552721517065
Gaussian Process	0.834467358741199	31.56940244032075	3.045705988033 2e-17	1868.7144363279638	0.0234019714235519

Conclusions

- Regression Trees outperform all other models in all except one metric – Bias Corrected RMSE.
- Gaussian Process Regression outperforms in Bias Corrected RMSE.
- Multiple linear regression gives good results, despite being the simplest method being used.
- Elastic Net Regression provides the best linear fit among all models.
- Tuning the hyperparameters better could yield better results.
- Incorporating more regression models – Long Short-Term Memory Neural Networks, Polynomial Regression, etc.
- More performance metrics could be evaluated for a thorough analysis of all models.
- Future predictions could be made using the already available time series data.

Google Colaboratory Notebook links

- https://colab.research.google.com/drive/1Wc2aQuKnh5T270-bI9uvQKY2RS0YYw0J#scrollTo=tgvmwBF_CE4M
- https://colab.research.google.com/drive/1Y3UVWEeNrUkN1jLxcPA9ziwjG1E9QnA5#scrollTo=mMS3if3i08yd
- https://colab.research.google.com/drive/1g59kH_gilCRimcd8-Wc-F1D8dmA3QBp8#scrollTo=R2kauHlgnqNN

THANK YOU!