Calibrate Away

Binchilling
CS771 - Introduction to Machine Learning
Indian Institute of Technology, Kanpur
2022-23 Semester II

1 Notation

We define the notations and terminology which will be consistently used throughout this report.

$Time \rightarrow$ Time stamp of recording
$Temp \rightarrow$ this tells us the temperature at that time stamp in degrees Celcius (between 0 and 100)
$Humidity \rightarrow$ this tells us the relative humidity at that time stamp as a percentage (between 0
$no2p1 \rightarrow$ Voltage values from Sensor 1 for NO2
$no2p2 \rightarrow$ Voltage values from Sensor 2 for O3
$o3p1 \rightarrow$ Voltage values from Sensor 1 for NO2
$o3p2 \rightarrow$ Voltage values from Sensor 2 for O3
$OZONE \rightarrow$ True levels of O3
$NO2 \rightarrow$ True levels of NO2
$MAE \rightarrow$ Mean Absolute Error

2 Comparison of Linear Models using only the 4 voltage values

Initially, we only predict the O_3 and NO_2 values using the four voltage values using linear models. Using different loss functions and different regularizers, we train the different models with the training data and check the MAE (Mean Absolute Error). Tuning the hyper-parameters and loss functions, we tabulate the optimal loss functions and the parameters for that model along with the Mean Absolute Error.

Model	Most Optimal Parameters	Most Optimal Loss Function	MAE
Ridge Regression	$\alpha = 1$	Least Square Loss	MAE (0ZONE) = 5.5735 MAE (NO2) = 6.6911
Lasso Regression	$\alpha = 1$	Absolute Loss	MAE (0ZONE) = 5.5730 MAE (NO2) = 6.6858
ElasticNet Regression	$\alpha = 1$ L1_ratio = 0.5	Epsilon-Insensitive Loss	MAE (0ZONE) = 5.6212 MAE (NO2) = 6.7681
Linear SVR	$\epsilon = 0.2$ tol = 1e-5 max_iter = 1000	Epsilon-Insensitive Loss	MAE (0ZONE) = 5.5944 MAE (NO2) = 6.9661

Preprint. Under review.

However, we see that the models are not upto mark and do not provide us with any satisfactory result. This is because we are basing our result only on the four voltage values and thus, data is insufficient to predict with lower MAE. Using the other features in the dataset might help to improve the predictions and lower the MAE score to acheive higher accuracy.

3 Non Linear Model

We now try to fit non-linear models to the dataset to improve our prediction values and reduce the MAE. Section 3.1 covers the preliminary analysis and exploratory non-linear testing performed on the data. Section 3.2 covers the feature extraction, the most optimal model found and goes over its hyperparameter tuning and MAE results

3.1 Preliminary Analysis

We first use the following set of features:-

- no2op1
- no2op2
- o3op1
- o3op2
- temp
- humidity

The table summarizes the best fit models with respective MAE_{NO2} and MAE_{O3}

Model	Best Hyperparameters	MAE(NO2)	MAE(O2)
Decision Tree	'max_depth' = 10	4.109	5.111
K-Neighbours Regressor	'n_neighbors' = 5	3.339	4.040
Support Vector Regressor	'C': 10, 'epsilon': 0.1	5.919	5.097
Neural Network	'hidden_layer_sizes': (100,), 'alpha': 0.001	5.149	5.232

Based on the results from the table, we decided further to optimize a KNeighbours Regressor and a Random Forest Regressor.

3.2 Most Optimal Model

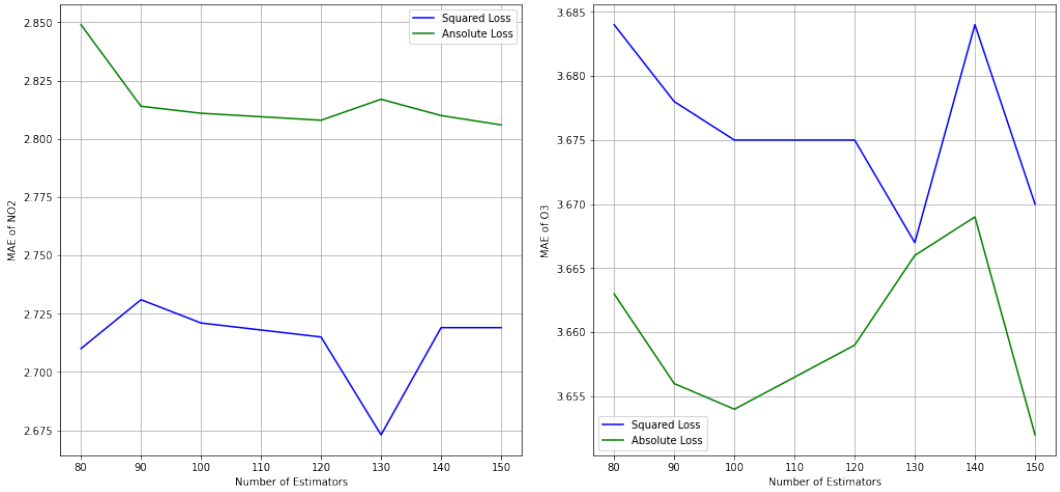
We used the time feature to generate a vector with the following heuristic to include the variation in the day-night cycle. We first extracted the Hour value from the time stamp column in the 24-hour format. Two new columns to categorize a day and a night were created based on simple thresholding. If $7 \leq \text{Hour} \geq 19$, we mark it as a Day and assign [1, 0] for the Day-Night columns.
Else we assign [0, 1] and mark the time stamp as a Night. We add these new features to the dataset and train the new non-linear models.

The Random Forest Regressor produced the best MAE results when we performed held-out validation with a 0.2 test fraction from the initial train data. We tuned the number of estimators and the loss function to obtain the most optimal model for each target variable.

- NO2RF Regressor: 130 estimators : Squared Loss : Time Thresholded data
- O3RF Regressor: 150 estimators : Absolute Loss : Unthresholded data

3.3 Plots

These plots show the trend of MAE value on the validation set as we vary the no. of estimators and the loss function. we considered Squared loss and Absolute Loss error function to train the Regressor. Other losses like Friedman MSE and Poisson gave poor MAE results on the validations set, and we have neglected them.



4 The Code

4.1 Training function

The

`my_train`

helps perform data morphing and training the RF Regressors.

```
def my_train(data,split):
    """
    Takes input data in pandas data frame format

    Outputs trained Random Forest Regressors for [NO2,O3] independently in that order
    """


    data['Hour'] = data['Time'].str[11:13].astype(int)
    data['DayNO2'] = data['Hour'].apply(lambda x: 1 if x >= 7 and x <= 19 else 0)
    data['NightNO2'] = data['DayNO2'].apply(lambda x: 0 if x == 1 else 1)
    colsNO2 = ['temp', 'humidity', 'o3op1', 'o3op2', 'no2op1', 'no2op2', 'DayNO2', 'NightNO2']
    colsO3 = ['temp', 'humidity', 'o3op1', 'o3op2', 'no2op1', 'no2op2']
    XNO2 = data[colsNO2]
    Y = data[['OZONE','NO2']]

    NO2RR = RandomForestRegressor(n_estimators=130, criterion='squared_error')
    O3RR = RandomForestRegressor(n_estimators=150, criterion='absolute_error')

    if split == True:
        X_trainNO2, X_testNO2, y_train, y_test = train_test_split(XNO2, y, test_size=0.2, random_state=42)
        NO2RR.fit(X_trainNO2,y_train.iloc[:,1])
        O3RR.fit(X_trainNO2,y_train.iloc[:,0])
        return [NO2RR,O3RR,X_test,y_test]
    else:
        NO2RR.fit(XNO2,y.iloc[:,1])
        O3RR.fit(XNO2,y.iloc[:,0])

    return [NO2RR,O3RR]
```

4.2 Prediction

The

`my_predict`

function takes testing data and calls pre-trained models to make NO2 and O3 level predictions

```
import numpy as np
import pandas as pd
import pickle as pkl

# Define your prediction method here
# df is a dfframe containing timestamps, weather df and potentials
def my_predict( df ):

        # Load your model file
        with open('NO2_RF.pkl', 'rb') as f:
                model1 = pkl.load(f)

        with open('O3_RF.pkl', 'rb') as f:
                model2 = pkl.load(f)

        # Make two sets of predictions, one for O3 and another for NO2

        df['Hour'] = df['Time'].str[11:13].astype(int)
        df['DayNO2'] = df['Hour'].apply(lambda x: 1 if x >= 7 and x <= 19 else 0)
        df['NightNO2'] = df['DayNO2'].apply(lambda x: 0 if x == 1 else 1)
        colsNO2 = ['temp', 'humidity', 'o3op1', 'o3op2', 'no2op1', 'no2op2',
        ↪  'DayNO2', 'NightNO2']
        colsO3 = ['temp', 'humidity', 'o3op1', 'o3op2', 'no2op1', 'no2op2']

        XNO2 = df[colsNO2]

        pred_o3 = model1.predict(XNO2)
        pred_no2 = model2.predict(XNO2)
        # Return both sets of predictions
        return ( pred_o3, pred_no2 )
```