CS771A Assignment 3

Adit Jain
Junior Undergraduate
Dept. of Electrical Engineering
Roll no.: 200038
`aditj20@iitk.ac.in`

Akshit Verma
Junior Undergraduate
Dept. of Electrical Engineering
Roll no.: 200091
`akshitv20@iitk.ac.in`

Ahmad Nabeel
Junior Undergraduate
Dept. of Material Science Engineering
Roll no.: 200063
`ahmad20@iitk.ac.in`

Abstract

This document is the submission of our group, "No Brainers" for Assignment 3. We have answered Part 1 and 2 with all the relevant level of detail required, providing mathematical proofs wherever required.

1 Solution to Question 1

For attempting this question efficiently, we list down the models we tried after giving a brief explanation about each of them:

- **Bayesian Ridge Regression**: It is a probabilistic linear regression method used to model the relationship between a dependent variable and one or more independent variables. The Bayesian approach differs from the traditional frequentist approach by incorporating prior knowledge about the distribution of the model parameters into the regression model. The Bayesian Ridge Regression model assumes that the distribution of the model parameters follows a Gaussian distribution with zero mean and a precision parameter that can be estimated from the data. The algorithm uses Bayesian inference to estimate the posterior distribution of the model parameters, which can be used to make predictions on new data. The Bayesian Ridge Regression method is known to be robust to multicollinearity and can handle datasets with a large number of features.
- **Elastic Net**: It is a linear regression method that combines the L_1 (Lasso) and L_2 (Ridge) regularization techniques to overcome their individual limitations. The method adds a penalty term to the loss function of the linear regression model, which is a linear combination of the L_1 and L_2 penalties. The L_1 penalty shrinks the coefficients of the model towards zero, which helps in feature selection and eliminates the less important features. The L_2 penalty, on the other hand, controls the magnitude of the coefficients, preventing them from becoming too large. The Elastic Net Regression method can handle datasets with a large number of features and is effective in dealing with multicollinearity. The method allows for the selection of relevant features while reducing the risk of overfitting.
- **Lasso Regression**: Also known as L_1 regularization, it is a linear regression method that adds a penalty term to the loss function of the model. The penalty term is the sum of the absolute values of the coefficients of the model, multiplied by a constant lambda. This penalty term shrinks the coefficients of the less important features towards zero, effectively eliminating them from the model. This feature selection property of Lasso Regression makes it useful when dealing with datasets with a large number of features, where some of them may be irrelevant or redundant. Lasso Regression can also be used for variable selection and is effective in dealing with multicollinearity. The method is known to produce sparse models, which are easier to interpret than models with a large number of features.
- **Linear Regression**: It is a popular supervised learning algorithm used for predicting a continuous output variable based on one or more input variables. The algorithm models the relationship between the input and output variables as a linear function, with the aim of minimizing the difference between the predicted values and the actual values of the output variable. The algorithm estimates the coefficients of the linear equation using a method such as ordinary least squares, which involves finding the line of best fit that minimizes the sum of the squared errors. Linear Regression is a simple and interpretable model that can be applied to a wide range of datasets. However, it assumes a linear relationship between the input and output variables and may not perform well when the relationship is nonlinear or when there are interactions between the input variables.
- **Orthogonal Matching Pursuit**: It is a linear regression method that selects a subset of the features of the dataset to build the model. The algorithm starts with an empty set of features and iteratively adds features to the model based on their correlation with the output variable. At each iteration, OMP selects the feature that has the highest correlation with the residual of the previous iteration and updates the coefficients of the model accordingly. OMP is a computationally efficient method that can handle datasets with a large number of features. The method is known to produce sparse models with a small number of non-zero coefficients, which makes them easier to interpret than models with a large number of features.
- **Ridge Regression**: Also known as L_2 regularization, it is a linear regression method that adds a penalty term to the loss function of the model. The penalty term is the sum of the squares of the coefficients of the model, multiplied by a constant lambda. This penalty term shrinks the coefficients of the model towards zero, effectively reducing the magnitude of the coefficients. This reduces the variance of the model, making it less sensitive to the noise in the dataset and reducing the risk of overfitting. Ridge Regression is useful when

dealing with datasets with a large number of features, where some of the features may be correlated or irrelevant. The method is known to perform well when the number of features is larger than the number of observations in the dataset. Ridge Regression is a simple and interpretable method that can be used for a wide range of applications in regression analysis.

Below, we have listed down the linear models that were available in sklearn's `sklearn.linear_model` module that we tried and the mean absolute error (MAE) for both O_3 and NO_2 we got in each case. It is to be noted that in order to get accurate results, we took a 3:1 split of the provided data for training and testing respectively:

Model	**MAE** O_3	**MAE** NO_2
Bayesian Ridge Regression	5.3449	6.4272
Elastic Net	5.4518	6.4733
Lasso Regression	5.2858	6.5171
Linear Regression	5.2694	6.5686
Orthogonal Matching Pursuit	10.4403	7.3005
Passive Aggressive Regression	16.1427	13.8048
Ridge Regression	5.3500	6.5115

The mean absolute error obtained on the provided data when the entire data was used for training is as follows:

Model	**MAE** O_3	**MAE** NO_2
Bayesian Ridge Regression	5.3588	6.4891
Elastic Net	5.3844	6.4917
Lasso Regression	5.3843	6.4884
Linear Regression	5.3588	6.4895
Orthogonal Matching Pursuit	10.3047	7.2985
Passive Aggressive Regression	12.0530	15.1639
Ridge Regression	5.3588	6.4895

As we can clearly see that normal Linear Regression works pretty good for the provided data set if only linear models are considered, as claimed by the manufacturer. The metric used for evaluation is mean absolute error (the lower the better) on the training data set itself.

2 Solution to Question 2

While attempting this question, as stated in the assignment, we ignored the timestamp. That is, timestamp was not considered to be a feature while making a prediction. This essentially left us with 6 parameters:

- Humidity
- Temperature
- First output voltage from NO_2 sensor
- Second output voltage from NO_2 sensor
- First output voltage from O_3 sensor
- Second output voltage from O_3 sensor

Next, in order to choose a particular model, we first needed to understand the data and the best way to do so is by plotting it. We used `matplotlib` and `seaborn` library in order to visualize it as shown in Fig. 1. As can be clearly seen, a linear fit would not produce good results as the data doesn't appear linear at all as is evident from Fig. 1. So, we need to look for more sophisticated non-linear models.

After long discussion and evaluations, we settled on **K Nearest Neighbour** algorithm. KNN is a non-parametric regression method used to predict the continuous output variable of a new data point based on the values of its K nearest neighbors in the training set. The value of K is a hyperparameter that determines the number of neighbors to consider for making the prediction. In KNN Regression, the predicted output value of a new data point is the average of the output values of its K nearest neighbors. Some of the advantages of choosing KNN are as follows:

1. Non-parametric: KNN is a non-parametric method, which means it does not make any assumptions about the distribution of the data. This makes it more flexible and adaptable to a wide range of datasets.
2. No training phase: KNN does not require a training phase, which means that the model can be used immediately after the data is collected. This makes it useful for real-time applications.
3. Suitable for small datasets: KNN is particularly suitable for small datasets where the number of features is relatively small compared to the number of observations.
4. Good performance for non-linear data: KNN can capture non-linear relationships between the input and output variables, making it suitable for datasets with complex and non-linear relationships.

The distance between a new data point and the training set is typically calculated using the Euclidean distance metric. However, other distance metrics such as Manhattan distance can also be used. In this case, we used **Manhattan distance** since we wanted to minimize mean absolute loss (MAE). The formula for calculating the Manhattan distance between a new data point and a training data point i is:

$$d(x, x_i) = \sum_{j=1}^{m} |x_j - x_{ij}| \tag{1}$$

where x is the new data point, x_i is the training data point, and m is the number of features in the dataset, which in our case equals to 4.

Once the distances between the new data point and all the training data points have been calculated, the K nearest neighbors are selected based on the distance metric and the value of K. The weights parameter determines the weighting of the neighbors in the prediction. In this case, we used the **'distance'** weight parameter, which weights each neighbor by the inverse of its distance to the new data point. Choosing weights='distance' over weights='uniform' is often preferred in KNN Regression because it gives more weight to closer neighbors, which tends to produce more accurate predictions.

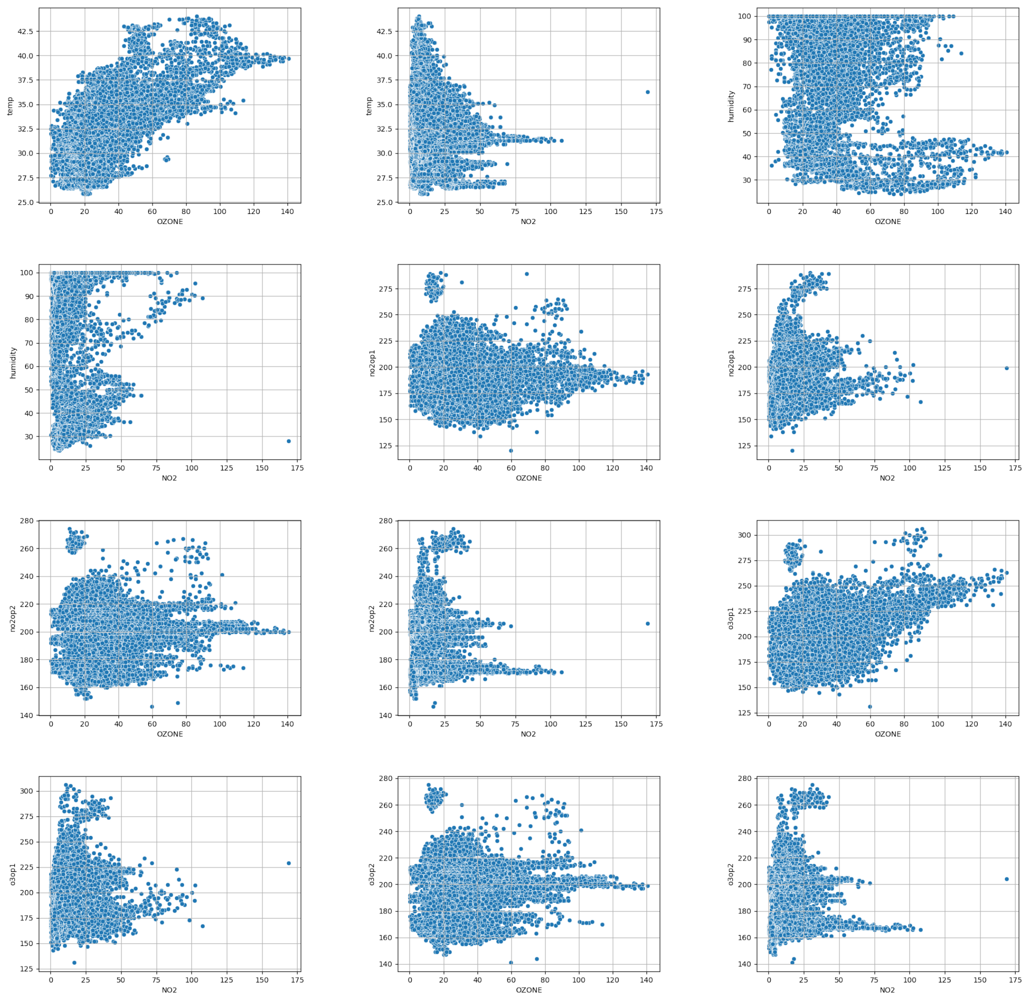

Figure 1: Variation of NO_2 and $OZONE$ with parameters

In uniform weighting, all neighbors within the specified number of K nearest neighbors have the same weight, regardless of their distance from the new data point. This approach can lead to inaccurate predictions when there are outliers or noisy data points in the dataset. Distance weighting assigns a weight to each neighbor based on its distance from the new data point. Closer neighbors are assigned higher weights, while farther neighbors are assigned lower weights. This approach is more robust to outliers and noisy data points, as it reduces the influence of distant neighbors on the prediction. More specifically, the weight assigned to each neighbor i can be defined as:

$$w_i = \frac{1}{d(d, x_i)^p} \tag{2}$$

where $d(x, x_i)$ represents the distance between the new data point x and the i-th neighbor x_i as explained in (1), and p is a hyperparameter that determines the degree of weighting, typically set to 1 for Manhattan distance. The weighted average prediction for the new data point x can then be calculated as:

$$y_{pred} = \frac{\sum_{i=1}^{K} w_i y_i}{\sum_{i=1}^{K} w_i} \tag{3}$$

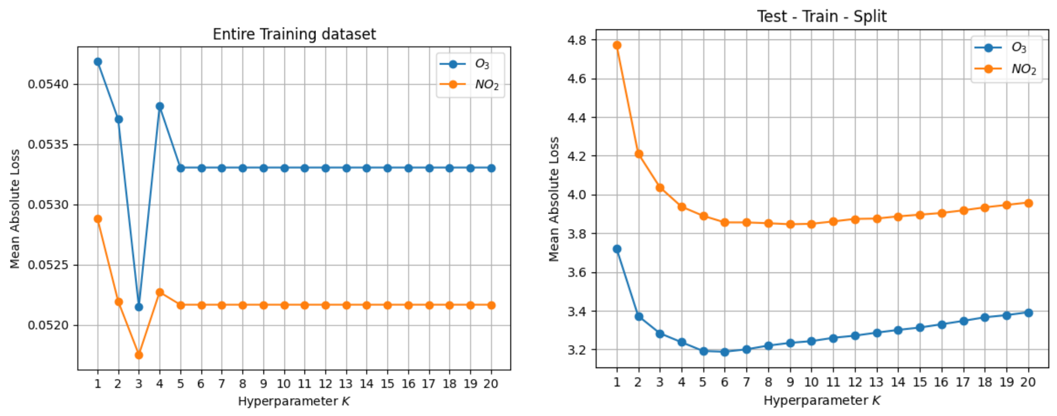

Figure 2: Hyperparameter tuning: K

where y_i represents the output value of the i-th neighbor and K is the number of nearest neighbors to be considered. So, the prediction for the new data point is calculated as a weighted average of the output values of its K nearest neighbors, where the weights assigned to each neighbor are determined by their distance from the new data point. Closer neighbors have higher weights, and farther neighbors have lower weights, which tends to produce more accurate predictions.

Now, all we need to do is to choose an ideal hyperparameter K that produces the most optimal results. The plot for the same is as shown in Fig. 2. As we can clearly see, we get the best results on choosing $K = 6$.

Finally, the code used for creating the model is as follows:

```
model = KNeighborsRegressor(n_neighbors=6, p=1, algorithm='auto',
    weights='distance', n_jobs=-1)
```

The stats for the k-Nearest Neighbor Regressor model we implemented are as given below:

- Model size: $1.6MB$
- Testing time: $0.1618s$ for 5,000 rows
- MAE O_3: 0.0298346
- MAE NO_2: 0.02141280

3 Solution to Question 3

The code is available in **submit.py** in the submitted zip file. The code in `submit.py` is as listed below:

```python
import numpy as np
import pickle as pkl


# Define your prediction method here
# df is a dataframe containing timestamps, weather data and potentials
def my_predict(df):
    # Separate the data-sets
    x_data = np.array(df.iloc[:, 1:7].values)

    # Load your model file
    model_file = open("./model.pkl", "rb")
    model = pkl.load(model_file)

    # Make two sets of predictions, one for O3 and another for NO2
    y_pred = model.predict(x_data)

    # Return both sets of predictions
    return (np.array(y_pred[:, 0]), np.array(y_pred[:, 1]))
```