TOML-MIRI Homework 2

Black carbon proxy sensor in air quality sensor monitoring networks

Ezgi Sena Karabacak
Spring 2024-2025

Introduction

Air pollution remains a pressing public health concern, and Black Carbon (BC) has emerged as a critical pollutant due to its strong associations with respiratory illnesses, climate forcing, and urban emissions. Accurate prediction of BC concentration can enable more effective monitoring and mitigation strategies. In this study, we aim to develop predictive models for BC concentration using a rich dataset containing various meteorological and atmospheric pollutant features collected throughout 2019.

The dataset includes variables such as PM (particulate matter), NO_x, O_3, CO, SO_2, temperature, and humidity. After preprocessing and exploratory analysis, we applied a variety of regression models including Support Vector Regression (SVR), Random Forests, Gradient Boosting Machines (GBMs), and Feedforward Neural Networks (FFNNs). To improve model performance, we explored different feature selection strategies such as Forward Sequential Selection (FSS) and Lasso, and also applied log-transformations to skewed variables.

We evaluated models under both shuffled and non-shuffled (time-aware) train-test splits, and tuned their hyperparameters using cross-validation. Performance was assessed using the coefficient of determination (R^2) and root mean squared error (RMSE), and model behavior was further analyzed through learning curves and temporal prediction plots.

Data Exploration and Visualization

We begin by loading the dataset provided in the assignment: `BC-Data-Set.csv`. The dataset contains measurements related to air quality, including Black Carbon (BC), various PM concentrations, gas pollutants, and meteorological conditions.

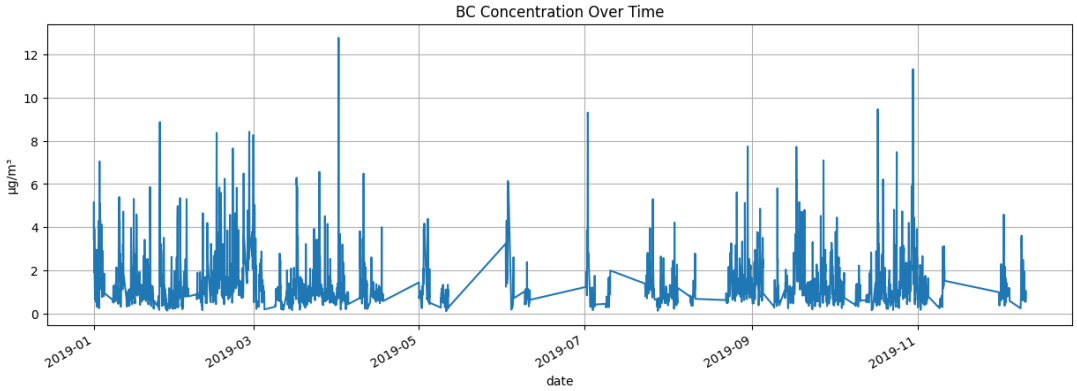
Initial Dataset Overview

We first inspect the structure and quality of the data:

- The dataset contains **4223 records** and **14 columns**.
- The `date` column is successfully parsed as a datetime object.
- All other columns are cast to numeric types.
- There are **no missing values**, making the dataset ready for analysis.

Temporal Trend of Black Carbon

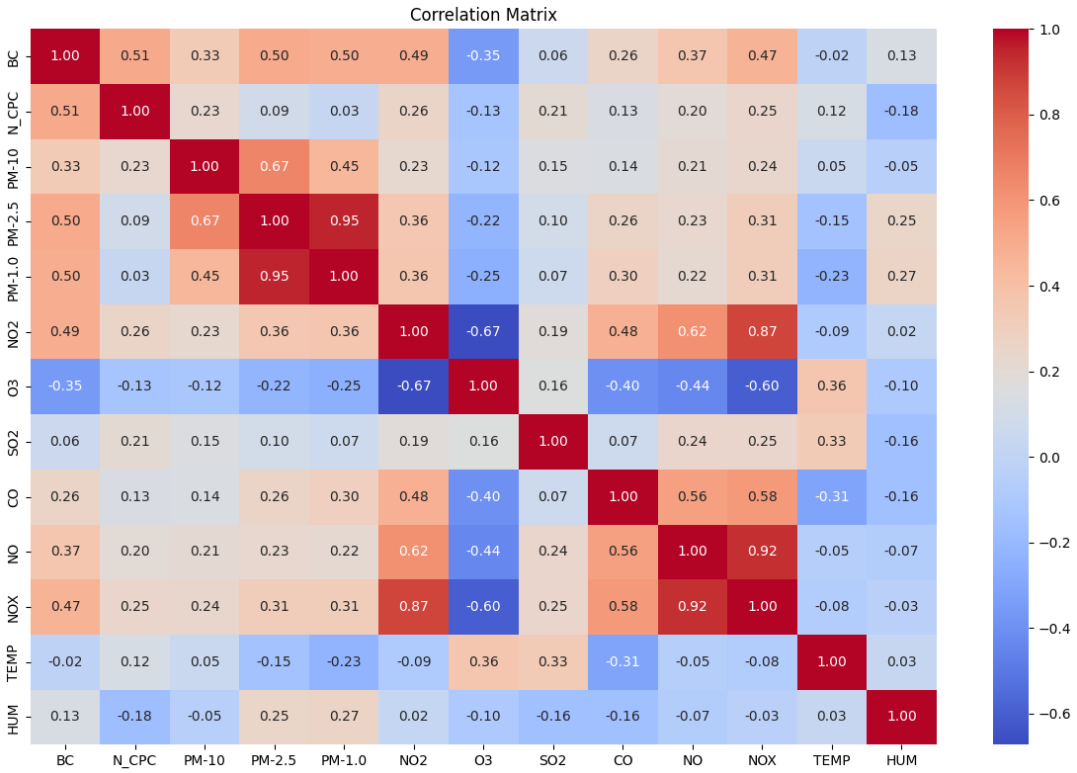
To understand temporal patterns in the target variable (BC), we plot its concentration over time:



We observe that BC concentration exhibits high variability and many short-duration peaks across the year. These likely correspond to localized emissions such as traffic or other combustion sources.

Correlation Analysis

We compute the Pearson correlation matrix between all numeric variables. This helps identify redundant features and strong predictors:



Notably, BC has relatively high correlation with:

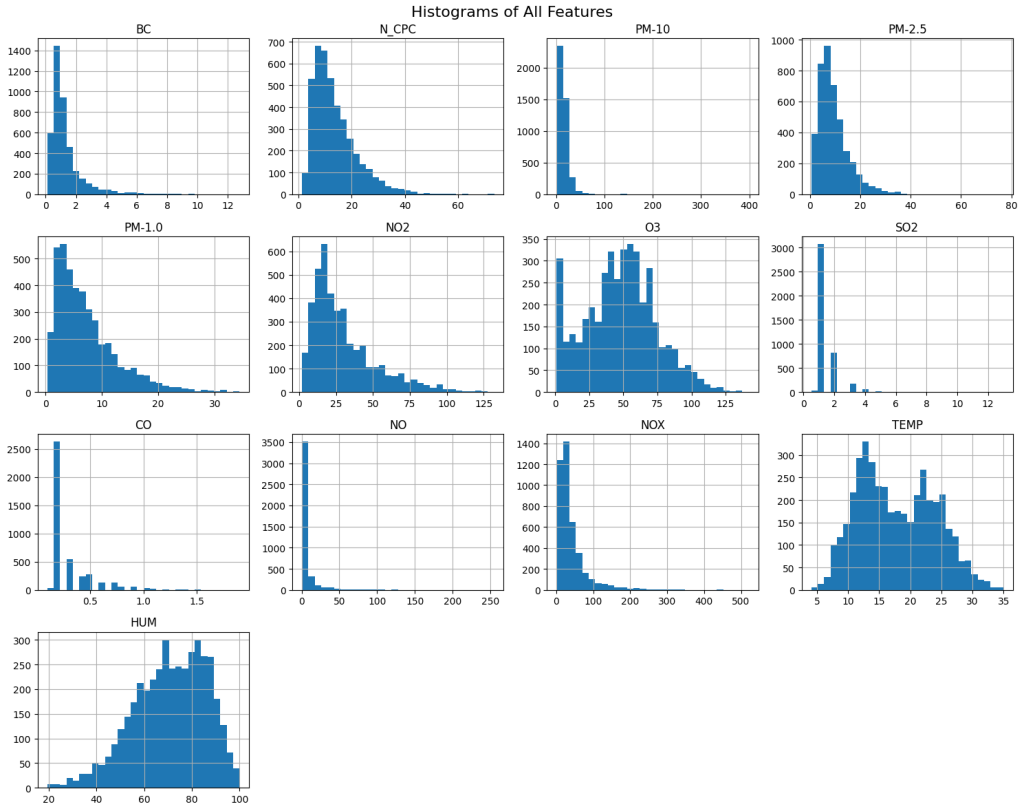
- N_CPC: 0.51

- PM-2.5: 0.50
- PM-1.0: 0.50
- NO_2: 0.49
- NOX: 0.47

These variables are likely to be the most useful for regression.

Feature Distributions

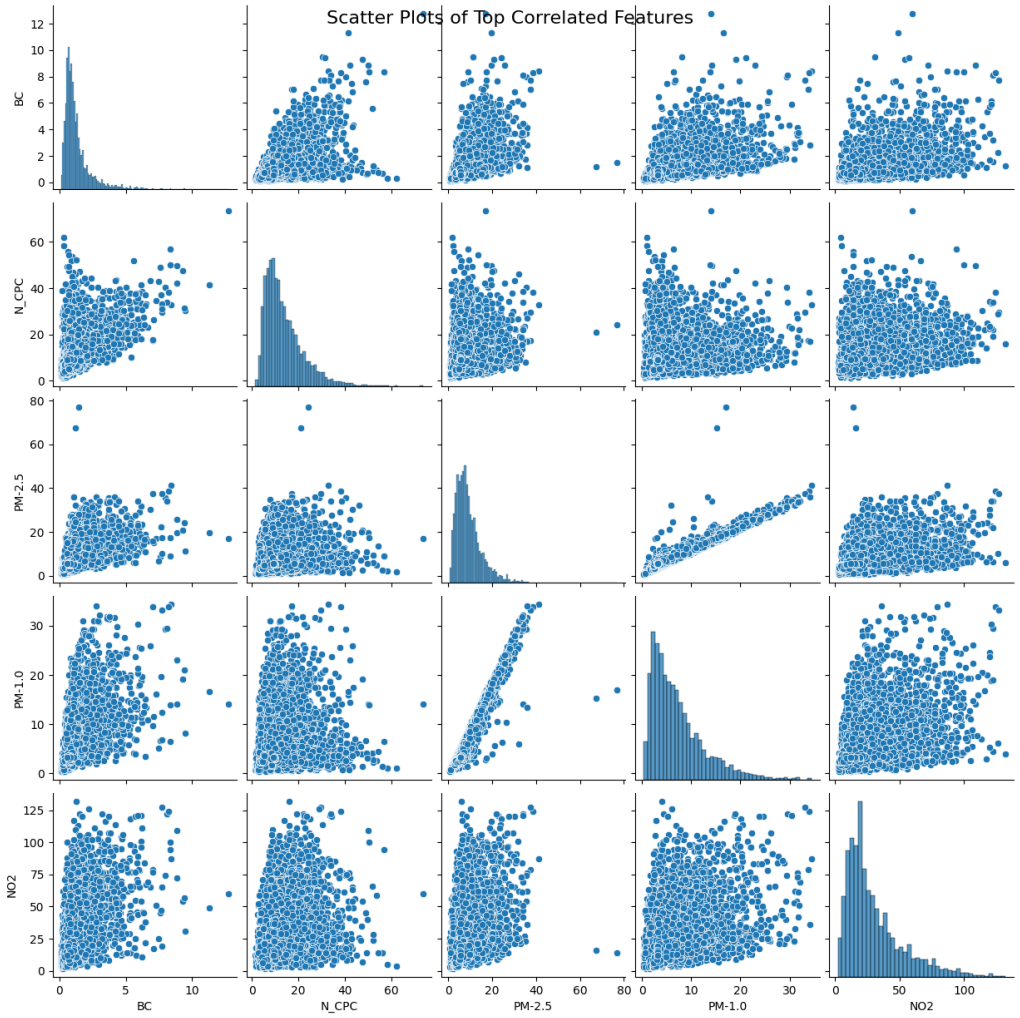
We visualize histograms of all numeric variables to check for skewness and outliers:



Most pollutant variables (including BC, NOX, CO) are right-skewed, which is expected due to occasional spikes in emissions. To mitigate this skewness, we applied a log-transformation using the function $\log(1 + x)$ to selected features. This helped to compress extreme values, reduce heteroscedasticity, and linearize relationships between variables. As a result, models such as Support Vector Regression (SVR) showed improved performance, achieving higher R^2 scores and lower RMSE values compared to models trained only with raw inputs.

Scatter Plots of Top Correlated Features

To visually assess linear/non-linear relationships, we created pairwise scatter plots between BC and its top correlated features:



A general positive trend is observable, though the relationships are not strictly linear. This motivates the use of non-linear regression models such as SVR, Random Forest, or Feedforward Neural Networks in the next steps of the study.

Selective Log Transformation and SVR Modeling

To mitigate the effect of skewed distributions and outlier sensitivity, we selectively applied a log-transformation $\log(1 + x)$ to a subset of pollutant variables based on prior inspection of skewness and dynamic range. Specifically, we transformed the following features: `N_CPC`, `PM-10`, `PM-2.5`, `PM-1.0`, `NO`, `NOX`, `CO`, `O3`, and `SO2`. These variables exhibited right-skewed distributions and/or spanned multiple orders of magnitude. To avoid multicollinearity, we removed the original versions after transformation.

We then trained a SVR model using a radial basis function (RBF) kernel with default hyperparameters ($C = 1.0$, $\epsilon = 0.1$). After scaling the data with standard normalization, the SVR model trained on all available (transformed) features achieved an R^2 score of **0.743** and an RMSE of **0.527**.

To identify a more compact and interpretable feature set, we applied Forward Subset Selection (FSS) using SVR as the estimator. We evaluated models by varying the number of selected features from 5 to 10. Table 1 summarizes the results. The model using **9 features** yielded the best performance ($R^2 = \mathbf{0.740}$, RMSE = **0.530**), nearly matching

the full-feature model. The selected features included transformed particle metrics and gases that are known contributors to BC concentration, such as `log_PM-2.5`, `log_NO`, and `log_O3`.

Table 1: SVR Performance with FSS on Selectively Log-Transformed Features

# Features	Selected Features	R^2	RMSE
5	TEMP, log_N_CPC, log_PM-2.5, log_NO, log_O3	0.719	0.551
6	TEMP, log_N_CPC, log_PM-2.5, log_PM-1.0, log_NO, log_O3	0.721	0.549
7	TEMP, log_N_CPC, log_PM-10, log_PM-2.5, log_PM-1.0, log_NO, log_O3	0.733	0.537
8	TEMP, HUM, log_N_CPC, log_PM-10, log_PM-2.5, log_PM-1.0, log_NO, log_O3	0.738	0.532
9	TEMP, HUM, log_N_CPC, log_PM-10, log_PM-2.5, log_PM-1.0, log_NO, log_NOX, log_O3	**0.740**	**0.530**
10	TEMP, HUM, log_N_CPC, log_PM-10, log_PM-2.5, log_PM-1.0, log_NO, log_NOX, log_O3, log_SO2	0.739	0.531

While the SVR model trained on all available features slightly outperformed those using subsets selected by FSS, the difference in performance was minimal ($R^2 = 0.743$ vs. $R^2 = 0.740$). This is expected, as SVR with an RBF kernel can handle moderate-dimensional feature spaces effectively, and the full set retained subtle feature interactions that may have been excluded during greedy selection. However, FSS still offered a more compact and interpretable model with near-identical performance.

Feature Count vs. SVR Performance

To better understand how the number of selected features impacts model performance, we visualized the evolution of R^2 and RMSE as we varied the number of features selected by FSS from 5 to 10. As shown in Figure 1, increasing the number of features generally led to improved performance, with diminishing returns beyond 9 features.

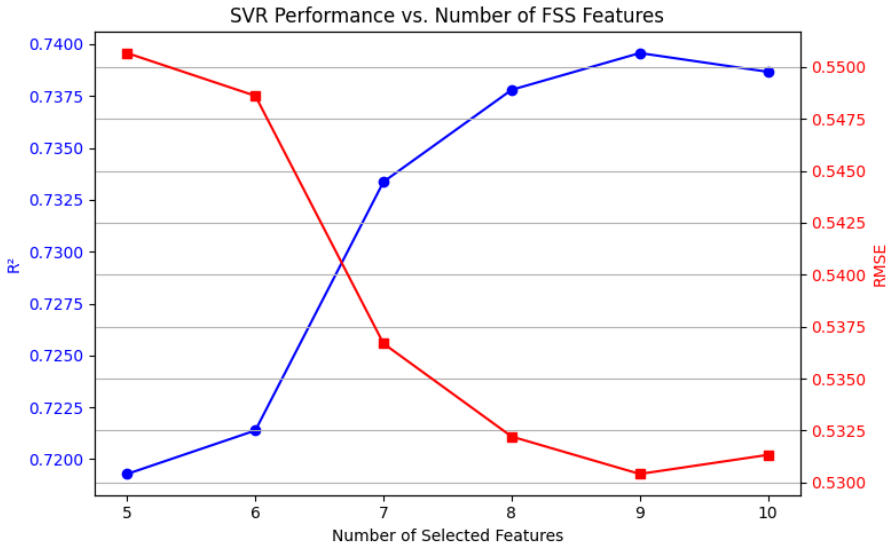


Figure 1: SVR Performance (R^2 and RMSE) vs. Number of FSS-Selected Features

Hyperparameter Tuning of SVR Model

To further improve our SVR model performance, we conducted a grid search over several hyperparameters using the best subset of features selected from FSS.

Grid Search Setup

We searched across the following SVR hyperparameters:

- **C**: $0.1, 1, 10, 100$
- **epsilon**: $0.01, 0.1, 0.5, 1$
- **kernel**: `rbf`, `linear`
- **gamma**: `scale`, `auto`

The search was conducted using 3-fold cross-validation, with the negative mean squared error (MSE) as the scoring metric. We used the best FSS feature set consisting of 9 variables:

```
['TEMP', 'HUM', 'log_N_CPC', 'log_PM-10', 'log_PM-2.5', 'log_PM-1.0',
'log_NO', 'log_NOX', 'log_O3']
```

Best Parameters and Performance

The best hyperparameters found were:

- **C** = 10
- **epsilon** = 0.1
- **kernel** = rbf
- **gamma** = auto

Using these parameters, the final evaluation metrics on the test set were:

- **R^2**: 0.789
- **RMSE**: 0.477

This tuning step provided a noticeable improvement over the untuned model (R^2 = 0.740, RMSE = 0.530), confirming that appropriate hyperparameter tuning enhances generalization.

To better understand the interaction between C and *epsilon*, we visualized the grid search results using a heatmap (Figure 2). The heatmap shows the cross-validated negative MSE across different combinations of C and *epsilon* (for kernel = rbf, gamma = scale).

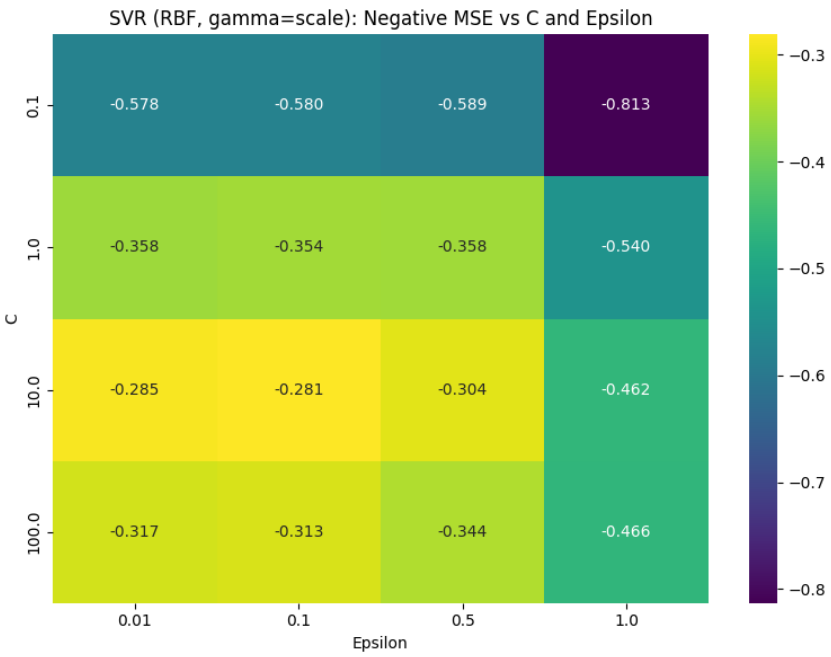


Figure 2: Grid Search Results: Negative MSE for SVR (RBF kernel, gamma=scale)

We can see from the heatmap that, lowest MSE is achieved when C is 10, and *epsilon* is 0.1. We can also see the trend where decreasing C and increasing *epsilon* result with worse performance.

Additionally, we plotted the predicted vs. true BC values for the tuned SVR model in Figure 3.

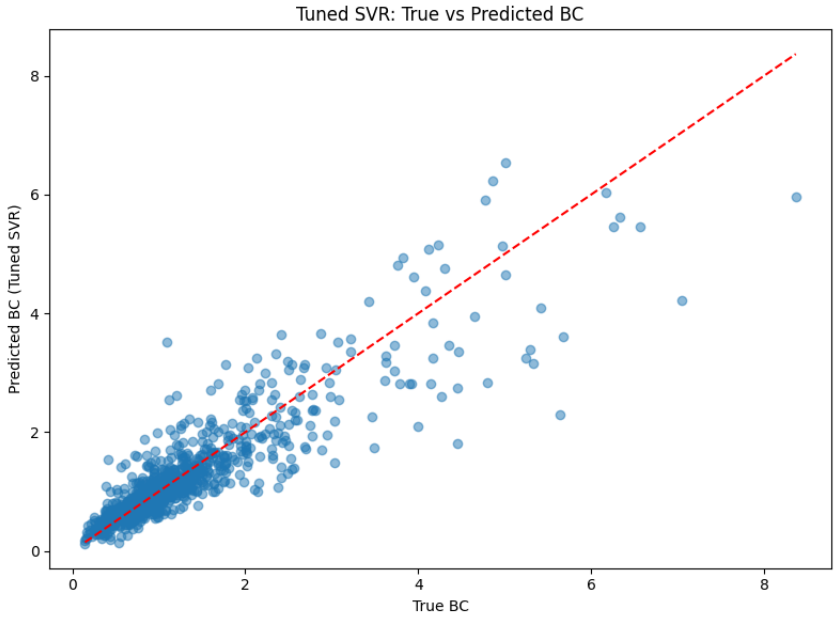


Figure 3: Tuned SVR: True vs. Predicted BC Concentration

The predictions align closely with the ground truth, especially for lower concentration values, indicating strong model reliability.

SVR with Lasso-Based Feature Selection

To further explore feature selection approaches, we applied Lasso regression as a filter method prior to using SVR. Lasso not only performs regression but also inherently selects features by shrinking some coefficients to exactly zero.

We standardized the full feature set and used 5-fold cross-validation with `LassoCV`. The non-zero coefficients were interpreted as selected features. These were:

```
['NO2', 'TEMP', 'HUM', 'log_N_CPC', 'log_PM-10', 'log_PM-2.5', 'log_PM-1.0',
'log_NO', 'log_NOX', 'log_CO', 'log_O3', 'log_SO2']
```

Using this subset, we trained a standard SVR (`kernel='rbf'`, `C=1.0`, `epsilon=0.1`) and obtained the following performance:

- **R^2:** 0.743
- **RMSE:** 0.527

This performance was comparable to using the full feature set or FSS, indicating Lasso was effective at isolating a meaningful subset. Yet, tuned FSS performed much better (**0.789**) than LASSO (**0.743**), so we continued our analyses with FSS selected features.

Random Forest Regression

After SVR, we evaluated a tree-based ensemble method, Random Forest Regression, using the 9 features selected by FSS. As Random Forest is not sensitive to feature scaling, we applied standardization primarily for consistency.

The initial untuned model achieved the following results:

- $\mathbf{R^2}$: 0.677
- **RMSE**: 0.591

To improve performance, we performed hyperparameter tuning using a grid search over the following space:

- `n_estimators`: $200, 300, 400, 500, 600$
- `max_depth`: $None, 10, 20$
- `min_samples_split`: $2, 5$
- `min_samples_leaf`: $1, 2$

Grid search used 3-fold cross-validation and negative MSE as the scoring metric. The best parameters found were:

- `n_estimators` = 300
- `max_depth` = None
- `min_samples_split` = 2
- `min_samples_leaf` = 2

With these settings, the final Random Forest model achieved:

- $\mathbf{R^2}$: 0.690
- **RMSE**: 0.579

Figure 4 illustrates how the number of estimators affects cross-validated MSE, with 300 trees offering the best balance between complexity and performance.

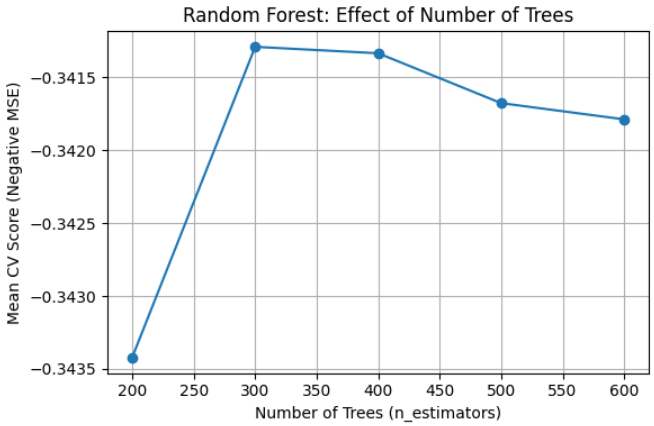


Figure 4: Effect of Number of Trees on Random Forest CV Score

While not outperforming SVR, Random Forest provided a good baseline performance with relatively low tuning effort.

Gradient Boosting Regression

We further evaluated Gradient Boosting Regression, another ensemble learning method known for its strong performance in structured data tasks. The same 9 FSS-selected features were used. The initial untuned Gradient Boosting model achieved:

- $\mathbf{R^2}$: 0.694
- **RMSE**: 0.575

To improve performance, we conducted a grid search with 3-fold cross-validation, tuning the following hyperparameters:

- `n_estimators`: $100, 200$
- `learning_rate`: $0.01, 0.1, 0.2$
- `max_depth`: $2, 3, 4$
- `min_samples_split`: $2, 5, 7$
- `min_samples_leaf`: $1, 2$

The best combination found by GridSearchCV was:

- `n_estimators` = 200
- `learning_rate` = 0.1
- `max_depth` = 4
- `min_samples_split` = 7
- `min_samples_leaf` = 1

With these hyperparameters, the tuned Gradient Boosting model reached:

- $\mathbf{R^2}$: 0.730
- **RMSE**: 0.540

Figure 5 shows a subset of the grid search results as a heatmap over `learning_rate` and `n_estimators`. This visualization highlights that higher learning rates and more trees generally yield better validation performance.

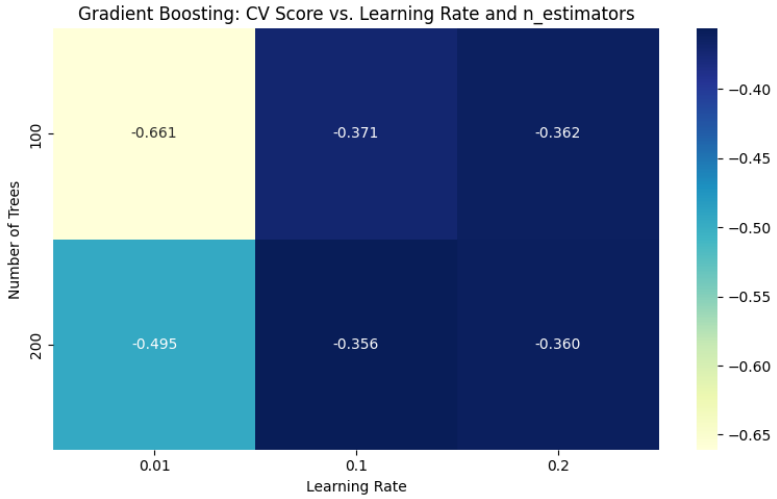


Figure 5: Gradient Boosting: CV Score vs. Learning Rate and Number of Trees

Gradient Boosting showed competitive performance compared to SVR and outperformed Random Forest in this study, making it a strong candidate for structured environmental prediction tasks.

Feedforward Neural Network (FFNN)

As a non-linear baseline, we evaluated a Feedforward Neural Network using the features selected by FSS. The initial FFNN model (hidden layers = (64, 32), activation = ReLU, optimizer = Adam) yielded the best performance overall:

- $\mathbf{R^2}$: 0.796
- **RMSE**: 0.469

To further investigate, we conducted a hyperparameter search using GridSearchCV across the following space:

- `hidden_layer_sizes`: (64,), (100,), (64, 32), (128, 64)
- `activation`: ReLU, tanh
- `solver`: Adam, L-BFGS
- `alpha` (L2 penalty): 0.0001, 0.001, 0.01
- `learning_rate_init`: 0.0001, 0.001, 0.01, 0.1

The best FFNN configuration discovered was:

- hidden layers = (128, 64)
- activation = tanh
- solver = Adam

- learning rate = 0.0001
- alpha = 0.0001

This tuned FFNN model achieved:

- $\mathbf{R^2}$: 0.783
- **RMSE**: 0.484

This is interesting because with grid search, we could not find better parameters then the initial model.

In Figure 6, we can see the effect of different hidden layer configurations on the model's performance. We observe that shallow architectures such as (64,) or (64, 32) provide competitive cross-validation scores, while deeper networks (e.g., (128, 64)) do not necessarily lead to better generalization. This suggests that increased complexity does not guarantee improved performance for this dataset.

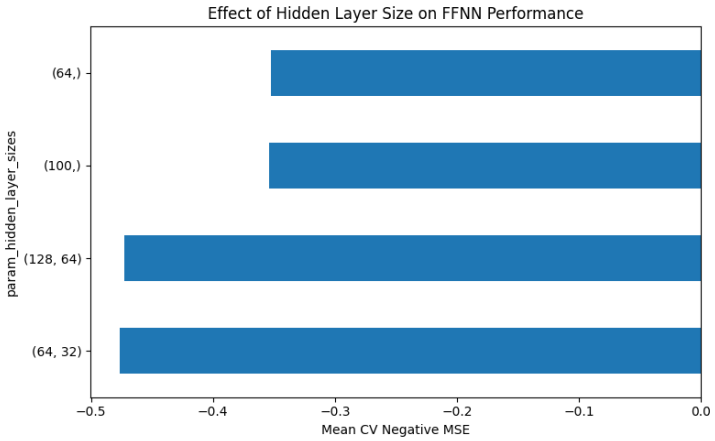


Figure 6: Effect of Hidden Layer Size on FFNN Performance (CV MSE)

Figure 7 illustrates how learning rate influences the model's training stability and performance. Extremely high learning rates (e.g., 0.1) lead to poor convergence and high error, while moderate values (e.g., 0.001) provide more stable and accurate results.

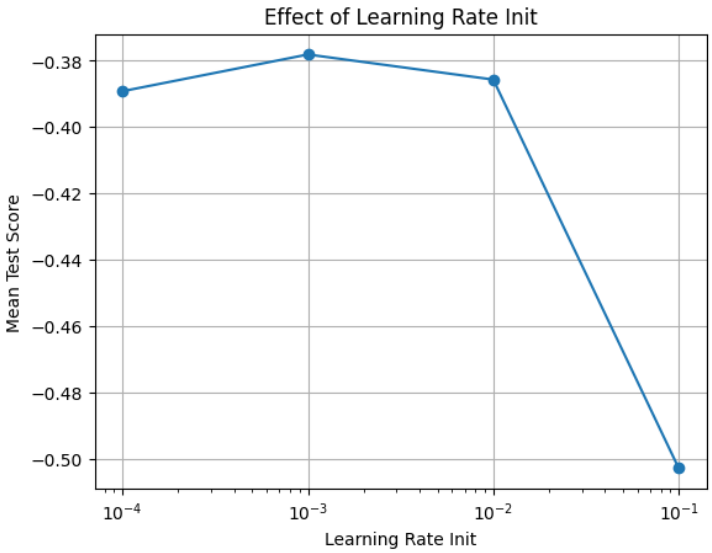


Figure 7: Effect of Learning Rate on FFNN CV Performance

Finally, Figure 8 presents the learning curve of the FFNN, showing training and validation R^2 scores as the training set size increases. The relatively small gap between the curves indicates a good balance between bias and variance, with no strong overfitting observed.

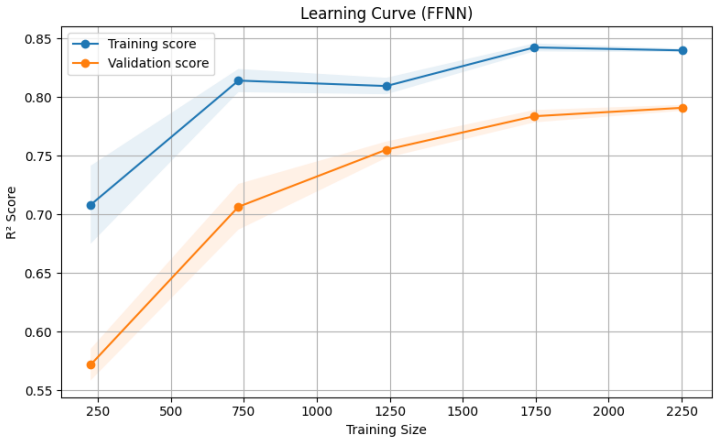


Figure 8: FFNN Learning Curve: Training vs. Validation R^2

In summary, FFNN outperformed other models in our evaluation, achieving the highest generalization performance when using hand-selected architecture without hyperparameter tuning.

Non-Shuffled Evaluation and Temporal Analysis

To further assess generalization in a time-aware setup, we re-trained the best FFNN model using a non-shuffled train-test split. This simulates a more realistic temporal prediction task.

The non-shuffled FFNN achieved:

- $\mathbf{R^2}$: 0.739

- **RMSE**: 0.571

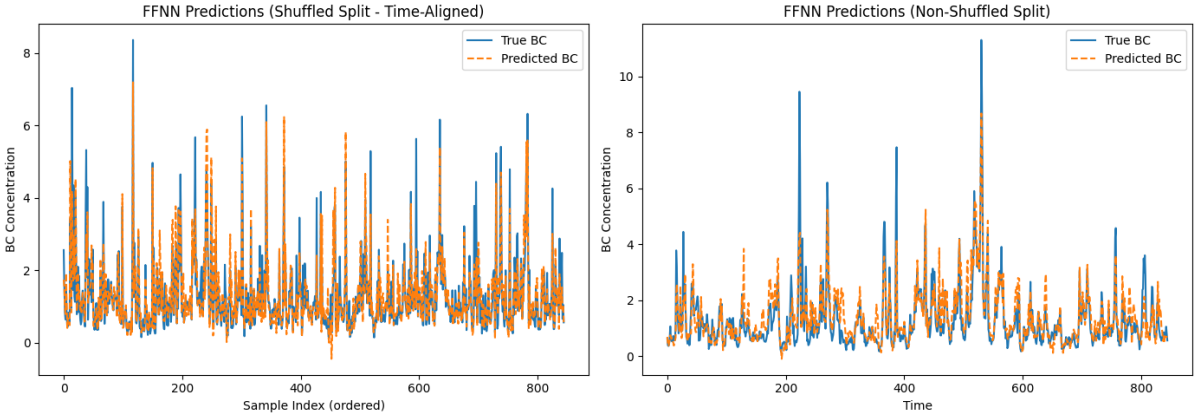


Figure 9: FFNN Prediction Performance: Shuffled vs. Non-Shuffled Split

Figure 9 compares FFNN predictions on the test set under shuffled (left) and non-shuffled (right) train-test splits. The left plot shows predictions for randomly split data re-ordered temporally. On the right, the model is evaluated on future data points. We observe a noticeable decrease in performance under the non-shuffled setup. This indicates that models trained on i.i.d. data (as in shuffled splits) generalize better in this case. For this specific task, shuffling the data during training improved prediction performance.

Overall Model Comparison

Finally, we compare all models based on their test R^2 and RMSE:

Model	**R^2**	**RMSE**
Full Features	0.739	0.531
SVR (FSS)	0.740	0.530
SVR (Lasso)	0.743	0.527
SVR (FSS & Tuned)	0.789	0.477
Random Forest (FSS)	0.677	0.591
Random Forest (FSS & Tuned)	0.690	0.579
Gradient Boosting (FSS)	0.694	0.575
Gradient Boosting (FSS & Tuned)	0.730	0.540
FFNN (FSS)	**0.796**	**0.469**
FFNN (FSS & Tuned)	0.783	0.484
FFNN (Non-Shuffled)	0.739	0.571

Table 2: Final Comparison of All Regression Models

Conclusion

In this work, we presented a comparative analysis of several regression models for predicting BC concentration from atmospheric and meteorological variables. We found that

appropriate preprocessing—particularly log-transforming skewed features—substantially improved model accuracy.

Among all models, the Feedforward Neural Network (FFNN) with FSS-selected features achieved the highest performance, reaching an R^2 of 0.796 and RMSE of 0.469. Although hyperparameter tuning led to more regularized models, it did not consistently improve prediction accuracy. This highlights the significance of good initial architecture design over exhaustive grid search. Support Vector Regression (SVR), when tuned, also performed strongly, with an R^2 of 0.789.

Evaluation on non-shuffled (temporally ordered) splits revealed a clear drop in performance, highlighting the challenges of generalization in real-world forecasting scenarios. This emphasizes the need for models to be evaluated under both i.i.d. and time-aware setups.

Overall, the combination of careful feature engineering, model selection, and realistic validation strategies allowed us to build interpretable and effective models for BC prediction.