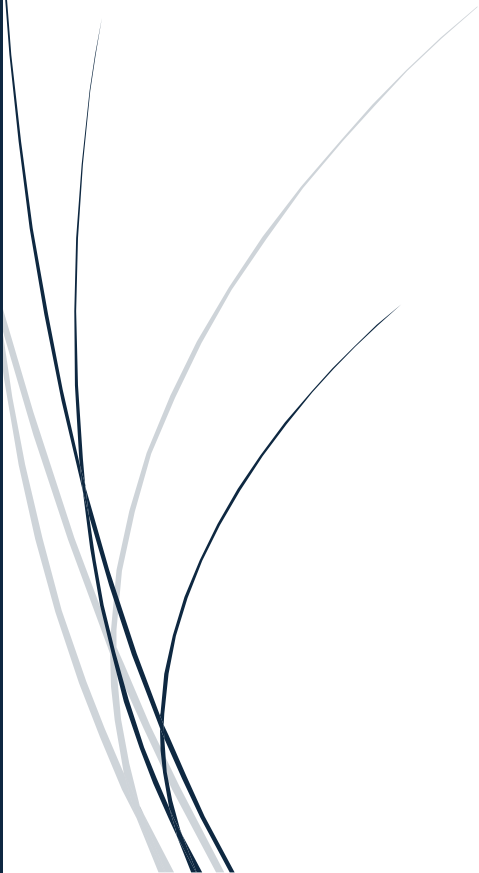


5/11/2025

Capital Bikeshare Report

Machine Learning II_DNSC_6315_10



Shah, Savera
MSBA, GWSB

Table of Contents

Business Understanding.....	2
Exploratory Analysis.....	2
Predictive Modeling	3
Linear and Regularized Regression Models	3
K-Nearest Neighbors (KNN)	3
Tree-Based Models.....	3
Neural Networks	3
Performance Evaluation	4
Prediction performance:	4
1. Linear and Regularized Regression Models (Linear, LASSO, Ridge, Elastic Net)	4
2. Tree-Based Models (Regression Tree, Random Forest, Gradient Boosting)	4
3. Non-Parametric Model (K-Nearest Neighbors)	4
4. Neural Networks.....	4
Best In-Sample (Train MSE) Performance.....	4
Best Out-of-Sample (Test MSE) & Average Performance	4
Decision performance.....	5
Training the Final Model	6
Conclusion and Recommendation:	6
Annexures	7
Figure 1	7
Figure 2	7
Figure 3	8
Figure 4	8
Figure 5	9
Figure 6	9
Figure 7	10
Figure 8	10
Figure 9	11
Figure 10.....	11
Figure 11.....	12
Figure 12.....	12
Figure 13.....	13
Figure 14.....	13
Figure 15.....	13

Business Understanding

Capital Bikeshare is a bike-sharing system in the Washington D.C. area, allowing users to rent and return bikes at various stations. To ensure smooth operations, each station must balance the number of bikes available for pickup and docks available for drop-off. Imbalances—like too few bikes in the morning or too few docks in the evening—can lead to poor user experience and inefficiencies.

This project builds predictive models to forecast daily demand for two target variables: **Pick-ups (PU_ct)** and **Drop-offs (DO_ct)**. These forecasts help allocate a fixed total station capacity of **17 slots** between bikes (x) and docks (y), such that $x + y = 17$.

We evaluate each model using a **cost-based decision metric**: unmet pickups incur a penalty of **2 units (α)**, while unmet drop-offs incur **3 units (β)**, reflecting their relative impact. By minimizing this out-of-sample cost, we identify models that not only predict well but also support better operational decisions. To simplify modeling, we applied **Principal Component Analysis (PCA)** to reduce dimensionality and retained **four principal components** as inputs for predicting PU_ct and DO_ct.

Exploratory Analysis

An initial exploratory analysis reveals important relationships between weather conditions (expressed as principal components) and bike demand, measured by pick-ups (PU_ct) and drop-offs (DO_ct). The correlation heatmap shows that both TVs highly correlated (0.90), suggesting similar patterns in bike usage behavior. Among the weather PCs, temperature ($temp_PC1$) and visibility (vis_PC1) display moderate positive correlations with both target variables, whereas wind speed ($wind_PC1$) is negatively associated—particularly with PU_ct (-0.64), indicating fewer pick-ups on windier days. Scatter plots reinforce these patterns, highlighting clearer trends in temperature and wind speed compared to precipitation. The time series line plot demonstrates increasing demand over time for both pick-ups and drop-offs. Furthermore, the boxplot of weather PCs indicates that visibility and wind speed have wide variability, which could introduce noise into model predictions.

- 1. Correlation Heatmap (Figure-1):** Shows strong correlation between Pick-ups and Drop-offs (0.90), while weather components like *Temperature* and *Visibility* are moderately correlated with bike usage, and *Windspeed* is negatively correlated—especially with pick-ups.
- 2. Time Series Plot (Figure-2):** Displays upward trends and seasonal fluctuations in bike pick-ups and drop-offs over time, with both targets moving in tandem.
- 3. Scatter Plots - PU_ct vs Weather PCs (Figure-3):** Indicate that higher temperatures and better visibility generally lead to more pick-ups, while wind has a negative effect; precipitation shows limited influence.
- 4. Scatter Plots - DO_ct vs Weather PCs (Figure-4):** Similar patterns to pick-ups, with drop-offs increasing in better weather conditions, and declining with stronger winds.
- 5. Boxplot of Weather PCs (Figure-5):** Highlights the spread and outliers in each weather variable after PCA; visibility and wind speed show high variance, which may impact prediction stability.
- 6. Descriptive Statistics Table (Figure6):** Summarizes central tendency and dispersion in the dataset; both target variables (Pick-ups and Drop-offs) have nearly identical means (~26), with moderate spread.

Predictive Modeling

To assess the relationship between explanatory variables and the two target outcomes—**PU_ct** and **DO_ct**—a diverse set of predictive modeling techniques were implemented. These included both linear and non-linear models: **Linear Regression**, **Ridge Regression**, **LASSO**, **Elastic Net**, **K-Nearest Neighbors (KNN)**, **Regression Tree**, **Random Forest**, **Gradient Boosting**, and **Neural Networks**. All models were fine-tuned using cross-validation to identify optimal hyperparameters before evaluating their performance on hold-out test data. **(Figure – 7)** Please refer to **Python Code** file for further modeling details.

Linear and Regularized Regression Models

Linear Regression served as the benchmark model and performed reasonably well on both Pickup and Drop-off, with balanced train-test MSE and R^2 values, indicating low variance and good generalization. Among the regularized variants:

- **Ridge Regression** slightly outperformed LASSO and Elastic Net, especially on *PU_ct*, offering the best compromise between bias and variance.
- **LASSO** and **Elastic Net** yielded slightly higher test errors and lower R^2 scores, however since they were hyper tuned therefore seemed to have good results. Overall, these models exhibited stable performance across both target variables.

K-Nearest Neighbors (KNN)

KNN showed modest predictive accuracy, with slightly lower test R^2 than the linear models, especially for *DO_ct*. However, its generalization was relatively stable compared to tree-based and neural models. Due to its non-parametric nature, KNN can model complex relationships, but in this case, it may have suffered from high dimensionality and insufficient distance-based separation in the input features.

Tree-Based Models

- The **Regression Tree** exhibited **severe overfitting**, with a large discrepancy between train and test errors and negative R^2 values on test data, particularly for *DO_ct*. This suggests that a single decision tree lacked the complexity control necessary for generalization.
- **Random Forest** showed better training performance, especially for *PU_ct*, and some improvement in test R^2 , but it still suffered from mild overfitting.
- **Gradient Boosting** achieved the **lowest training error and highest in-sample R^2** , demonstrating strong learning capacity. However, its test performance sharply deteriorated, with negative R^2 on both targets, indicating **high overfitting** and poor real-world prediction utility without further regularization or tuning.

Neural Networks

Despite their theoretical capacity to approximate complex functions, **Neural Networks underperformed on test data**. While the model fit the training data well, it failed to generalize, yielding negative or near-zero R^2 values on test sets. This points to overfitting and sensitivity to hyperparameters or insufficient training data for robust performance.

Performance Evaluation

Prediction performance:

This section evaluates and compares the prediction performance of *nine machine learning* models applied to two target variables. Performance metrics include Mean Squared Error (MSE) and R^2 for both train and test sets, along with a computed out-of-sample cost. (Figure – 7)

1. Linear and Regularized Regression Models (Linear, LASSO, Ridge, Elastic Net)

These models show relatively consistent performance with moderate to high generalization capability, as seen in the small difference between training and test MSE values. Among these:

- **Linear Regression** has the lowest test MSE for PU_ct (55.58) among regularized models, though R^2 values are modest (PU_ct test $R^2 = 0.3367$; DO_ct test $R^2 = 0.2013$).
- **LASSO** and **Elastic Net** slightly underperform compared to Ridge, with higher test MSEs and lower R^2 values, suggesting that neither L1 nor L1+L2 regularization significantly improves predictive power in this case.
- All regularized models yield **test MSEs for DO_ct around ~70** with **low R^2 (< 0.21)**, indicating **weak explanatory power** for the second dependent variable across linear model families.

2. Tree-Based Models (Regression Tree, Random Forest, Gradient Boosting)

Tree-based models exhibit **divergent behavior**:

- The **Regression Tree** suffers from severe overfitting with PU_ct train MSE = 42.71 vs test MSE = 140.65 and a negative test R^2 for DO_ct (-0.9244), rendering it unreliable for generalization.
- **Random Forest** performs considerably better, with low train MSE (**PU_ct = 21.19; DO_ct = 40.45**) and moderate generalization (PU_ct test $R^2 = 0.1314$; DO_ct = 0.1232). It improves over linear models for training data but still struggles on unseen data.
- **Gradient Boosting** achieves the lowest train MSE (**PU_ct = 11.87; DO_ct = 34.76**) and the highest train R^2 (0.87 and 0.64), indicating strong in-sample learning. However, it fails to generalize, with negative test R^2 for both outcomes, suggesting significant overfitting.

3. Non-Parametric Model (K-Nearest Neighbors)

- KNN shows moderate training and test performance, with test $R^2 = 0.2920$ (PU_ct) and 0.1861 (DO_ct). The gap between train and test metrics is less pronounced than in tree models, indicating more stable generalization but limited predictive strength.

4. Neural Networks

- Neural Networks also show significant overfitting, with a large gap between train MSE (PU_ct = 42.30) and test MSE (PU_ct = 80.00). The **test R^2 is very low (0.0452 for DO_ct)** and even negative in some cases, suggesting poor test performance despite high model capacity.

Best In-Sample (Train MSE) Performance

- **Gradient Boosting** achieved the lowest train MSE (11.88) in both PU_ct and DO_ct, indicating the best in-sample fit. (Figure – 8)

Best Out-of-Sample (Test MSE) & Average Performance

- **PU_ct: Linear Regression** had the lowest test MSE (55.58), suggesting the most reliable generalization on unseen data. However, it is not hyper tuned, therefore we will look at the next best which is **Ridge Regression and Lasso (Test MSE 56.3 & 59.4)** which yielded better least average and least out-sample MSE's. (Figure – 9 & 10)

Decision performance

Figure 11 presents a tabular and visual comparison of out-of-sample costs for various predictive models across increasing station capacities (10 to 50). Each model's decision performance is evaluated based on how effectively it minimizes the cost of unmet bike pickups and drop-offs. At every capacity level, the table shows the exact average cost, while the plot below offers a clearer view of model trends over the capacity range.

From the visual trend, we observe that simpler models like Linear Regression, Lasso, and Ridge Regression follow nearly identical paths and improve steadily with increased capacity, but consistently underperform relative to more advanced models. KNN, Gradient Boosting, and Random Forest show the steepest decline in cost, especially beyond capacity 25, indicating strong decision performance when more flexibility in resource allocation is available. Notably, Elastic Net slightly outperforms other linear models due to its hybrid regularization. Regression Tree and Neural Network show moderate gains but do not surpass the ensemble methods.

Figure 12 illustrates how the average out-of-sample cost evolves for each predictive model as total station capacity (bikes + docks) increases from 10 to 50. Initially, all models start with an equal cost of 76.64 at capacity 17, highlighting their baseline performance without re-optimization. As capacity increases, nearly all models show reduced out-of-sample cost, with more expressive and flexible models—like KNN, Gradient Boosting, and Random Forest—achieving sharper improvements. Simpler models, such as Linear Regression and Ridge, lag slightly in performance as they lack the non-linear learning capacity needed to fully capture demand dynamics at higher capacities.

By capacity 50, **KNN**, Gradient Boosting, and Random Forest consistently outperform others, reaching the lowest cost levels (~18). Elastic Net also performs competitively in the mid-capacity range. The Regression Tree shows decent performance but flattens out at higher capacities. Notably, Linear Regression consistently underperforms across all capacity levels, confirming the limitations of linear models in complex decision contexts. This reinforces the importance of re-tuning decision rules when capacity changes and supports the use of more sophisticated models for optimal bike-dock allocation.

Key Insights:

- **At Capacity 17:** All models perform equally with **cost \approx 76.64**; decision rules not yet re-optimized.
- **Top performers at higher capacities (45-50):**
 - **KNN (lowest cost: 17.89 at cap 50),**
 - Random Forest,
 - Gradient Boosting.
- **Elastic Net:** Performs well at moderate capacities (25–40) due to regularized flexibility.
- **Regression Tree:** Improves modestly but levels off beyond capacity 40.
- **Linear Models (LR, Lasso, Ridge):** Lag; limited in capturing non-linear demand patterns.
- **Neural Network:** Slightly better than linear models, but not competitive with ensemble methods at higher capacities.
- **Decision insight:** Non-linear and ensemble models provide significantly better cost optimization as capacity expands, especially beyond 25. Therefore, more flexible models with re-optimized decisions are critical to reducing unmet demand costs at higher capacities.

Training the Final Model

To further train the models and finally show predictive and decision-making performance, I trained three models—**Ridge Regression, LASSO, and K-Nearest Neighbors (KNN)**—on the same dataset, using two distinct evaluation criteria. Ridge and LASSO models were tuned and assessed using traditional predictive metrics like Mean Squared Error (MSE), Mean Absolute Percentage Error (MAPE), and R^2 . Based on **prediction performance at capacity 17**, Ridge Regression slightly outperformed LASSO with the lowest MSE for both PU_ct and DO_ct, suggesting better generalization on unseen data. (Figure 14)

	Model	MSE (PU_ct)	MAPE (PU_ct)	R^2 (PU_ct)	MSE (DO_ct)	MAPE (DO_ct)	R^2 (DO_ct)	Avg Cost
0	Ridge Regression	55.580437	0.557821	0.336695	69.546437	0.465370	0.201317	76.638889
1	LASSO	55.625362	0.558412	0.336159	69.518944	0.465575	0.201633	76.638889

However, when focusing on **decision performance using a real-world-inspired cost function at capacity 50**, a model with the **lowest average out-of-sample cost** (from the KNN implementation) was considered optimal. This dual approach reflects how predictive accuracy doesn't always align with cost-effective decision outcomes, highlighting the importance of aligning model selection with the final business objective. Having said that, there is always a trade-off to be made between Prediction Performance and Decision Performance and based on business goals and risk appetite these models could be selected. (Figure 15)

	Model	MSE (PU_ct)	MAPE (PU_ct)	R^2 (PU_ct)	MSE (DO_ct)	MAPE (DO_ct)	R^2 (DO_ct)	Avg Cost
0	KNN	59.727732	0.594146	0.287201	70.842057	0.47441	0.186438	17.305556

Key Takeaways:

- **Ridge Regression** showed the **lowest MSE** and best prediction accuracy at capacity 17.
- **LASSO** was nearly identical in predictive performance but slightly underperformed Ridge.
- **KNN** was evaluated based on **decision performance**, showing lower realized costs in operational settings (capacity 50).
- **Model choice varies depending on the objective**: use Ridge for better prediction, KNN for cost-optimized decision-making.

Conclusion and Recommendation:

This study demonstrates that no single model excels universally across both predictive and decision-making objectives. While Ridge Regression provided the most reliable forecasts at the base capacity of 17, K-Nearest Neighbors (KNN) significantly outperformed in minimizing real-world operational costs at higher capacities. These findings emphasize the importance of aligning model selection with specific business goals: **Ridge Regression is recommended when accurate demand forecasting is critical**, while **KNN is preferred when the primary goal is cost-effective allocation of limited resources**. For future implementations, organizations should consider retraining and re-optimizing models as station capacities or user behaviors evolve, and explore hybrid approaches that integrate predictive strength with decision flexibility.

Annexures

Figure 1

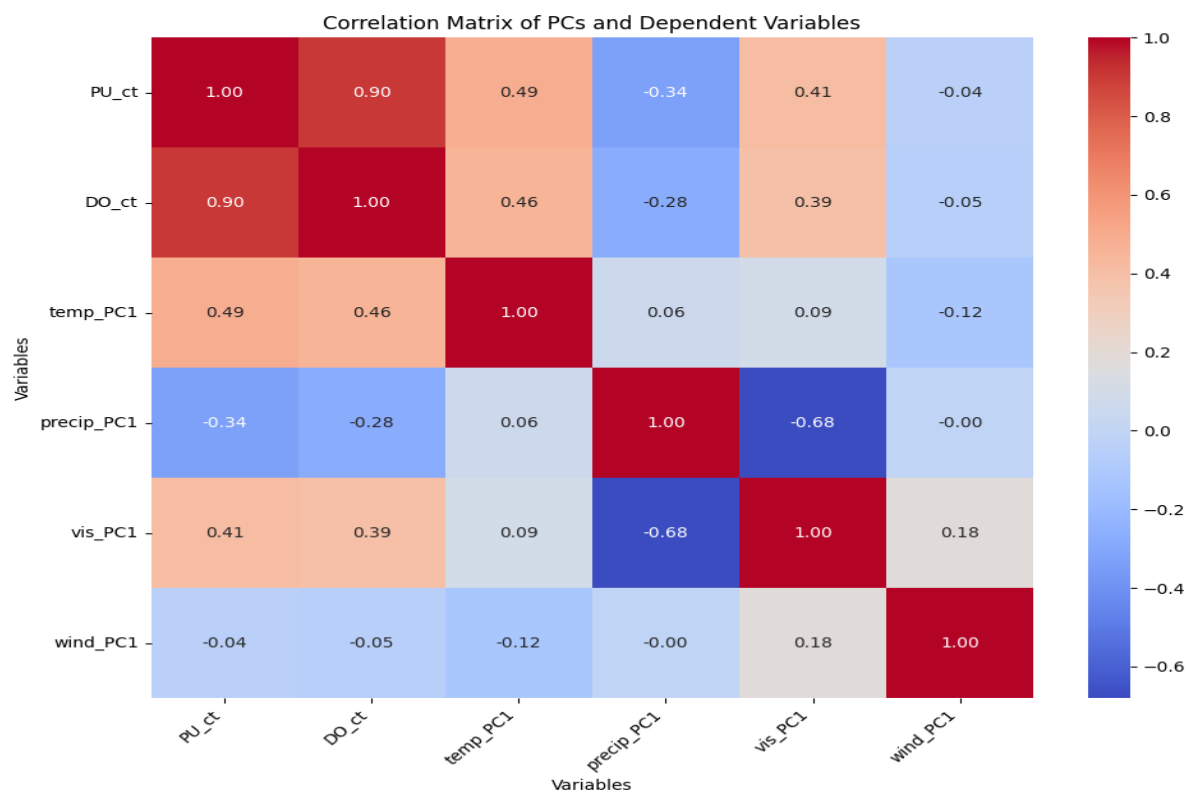


Figure 2

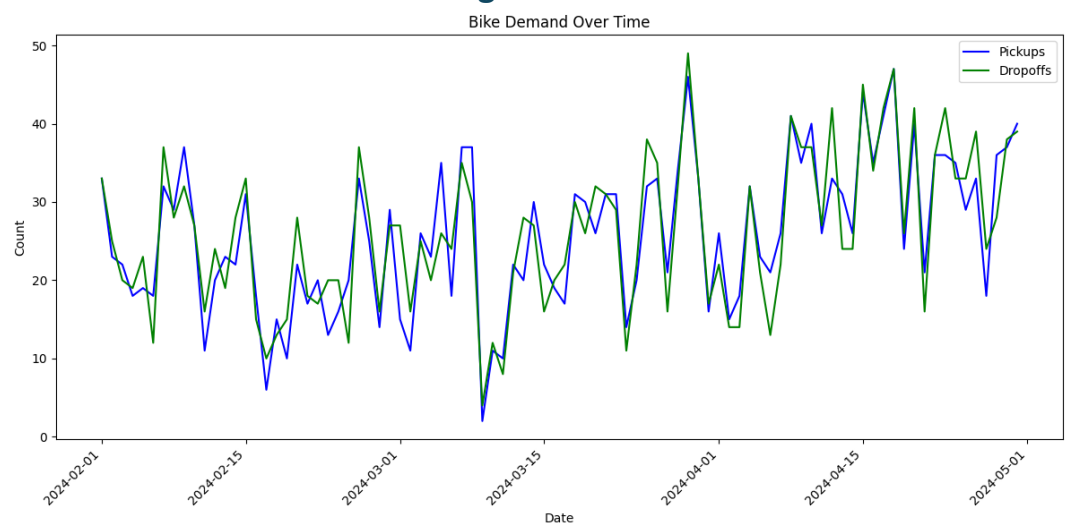


Figure 3

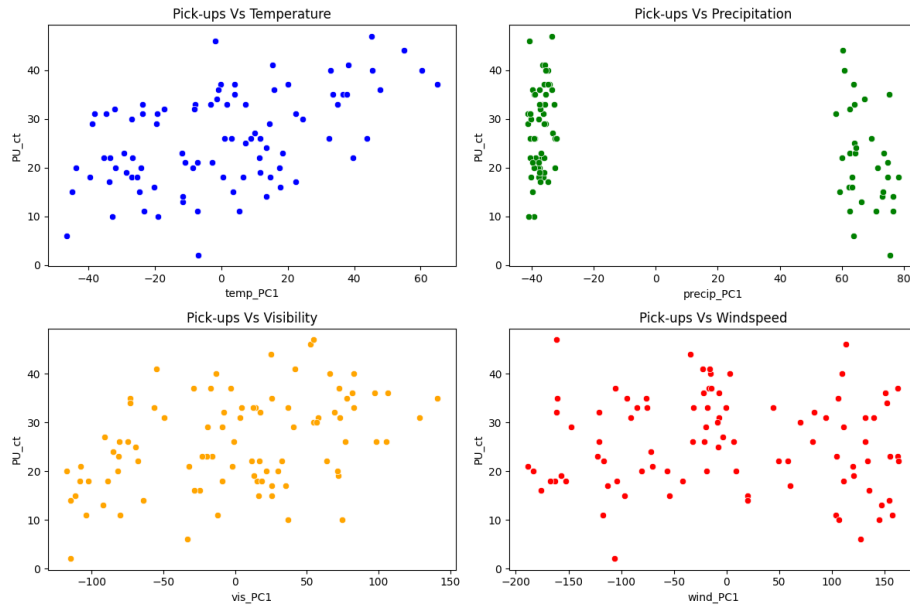


Figure 4

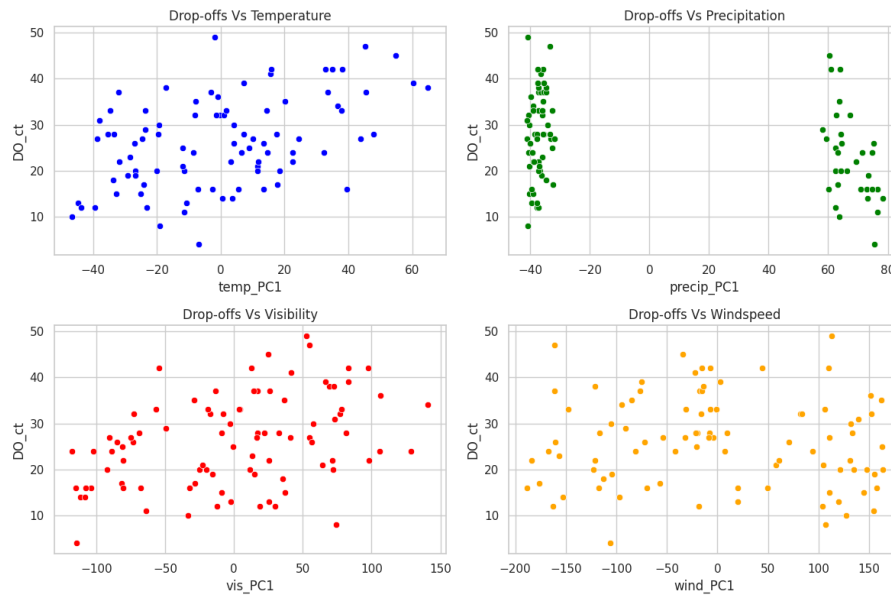


Figure 5
Reduced Weather Features

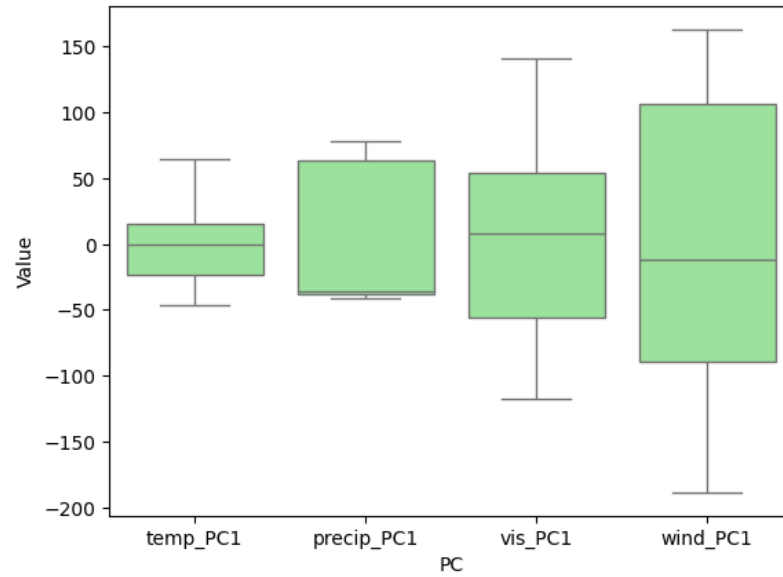


Figure 6

	temp_PC1	precip_PC1	vis_PC1	wind_PC1	PU_ct \
count	9.000000e+01	9.000000e+01	9.000000e+01	9.000000e+01	90.000000
mean	1.294767e-14	-3.868510e-15	3.568504e-14	-4.105358e-15	25.788889
std	2.663702e+01	5.049623e+01	6.538216e+01	1.063508e+02	9.575829
min	-4.656371e+01	-4.117743e+01	-1.174665e+02	-1.885948e+02	2.000000
25%	-2.351309e+01	-3.768570e+01	-5.604744e+01	-8.955340e+01	18.250000
50%	-5.098084e-01	-3.550595e+01	8.054001e+00	-1.141693e+01	26.000000
75%	1.575963e+01	6.304175e+01	5.405860e+01	1.066164e+02	33.000000
max	6.500839e+01	7.825041e+01	1.410322e+02	1.632812e+02	47.000000

	D0_ct
count	90.000000
mean	26.088889
std	9.768839
min	4.000000
25%	19.000000
50%	26.000000
75%	33.000000
max	49.000000

Figure 7

	Model	Train MSE (PU_ct)	Test MSE (PU_ct)	Train R ² (PU_ct)	Test R ² (PU_ct)	Train MSE (DO_ct)	Test MSE (DO_ct)	Train R ² (DO_ct)	Test R ² (DO_ct)	Out-of-sample Cost
0	Linear Regression	56.2071	55.5805	0.4003	0.3367	58.0363	69.5466	0.4060	0.2013	76.6389
1	LASSO	58.1256	59.4939	0.3798	0.2900	60.4860	70.4646	0.3809	0.1908	76.6389
2	Ridge Regression	56.8039	56.3761	0.3939	0.3272	58.6145	69.5842	0.4001	0.2009	76.6389
3	Elastic Net	58.9533	60.0218	0.3710	0.2837	60.5701	70.5515	0.3801	0.1898	76.6389
4	Regression Tree	42.7148	140.6536	0.5443	-0.6786	45.0615	167.5664	0.5388	-0.9244	76.6389
5	KNN	61.2695	59.3253	0.3463	0.2920	62.3721	70.8712	0.3616	0.1861	76.6389
6	Random Forest	21.1962	72.7813	0.7738	0.1314	40.4538	76.3468	0.5860	0.1232	76.6389
7	Gradient Boosting	11.8773	92.2931	0.8733	-0.1014	34.7620	100.1938	0.6442	-0.1506	76.6389
8	Neural Networks	65.1216	79.2685	0.3052	0.0540	70.8219	106.9631	0.2751	-0.2284	76.6389

Figure 8

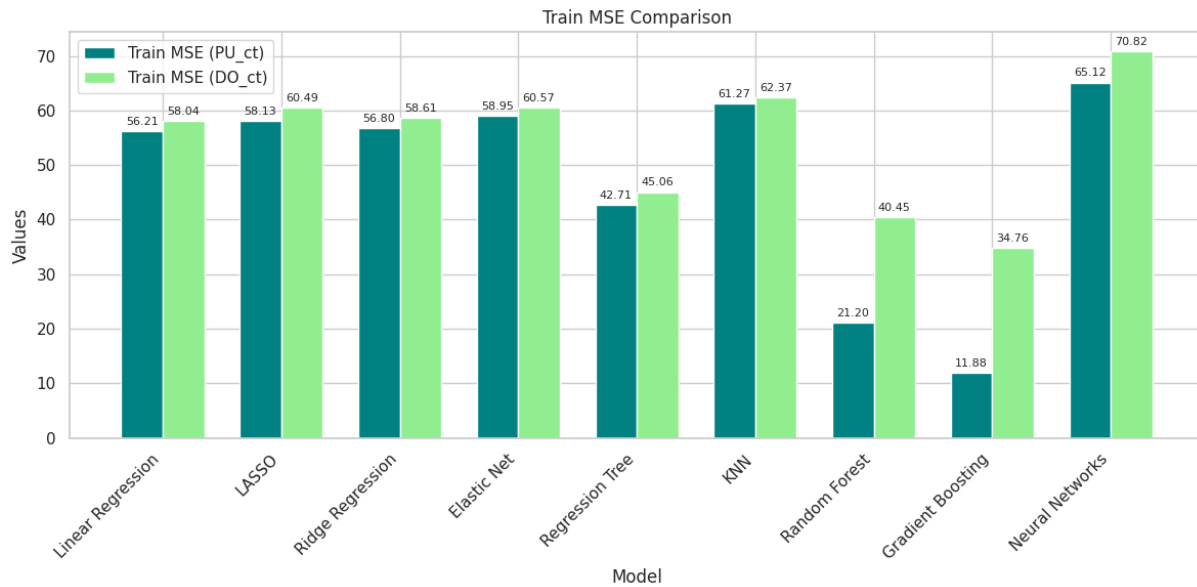


Figure 9

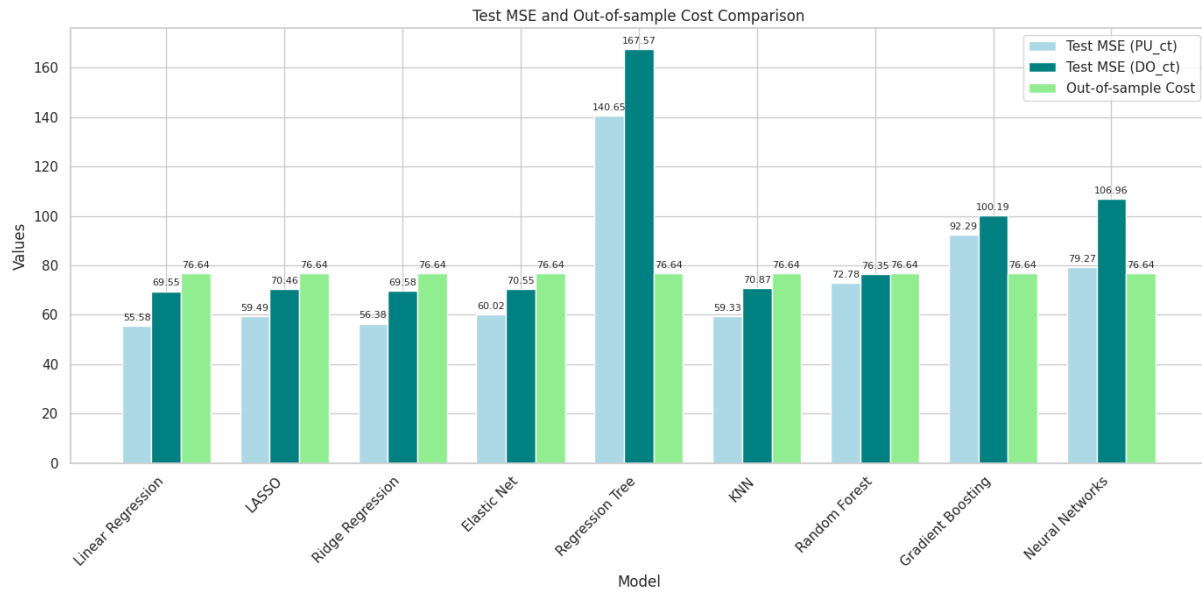


Figure 10

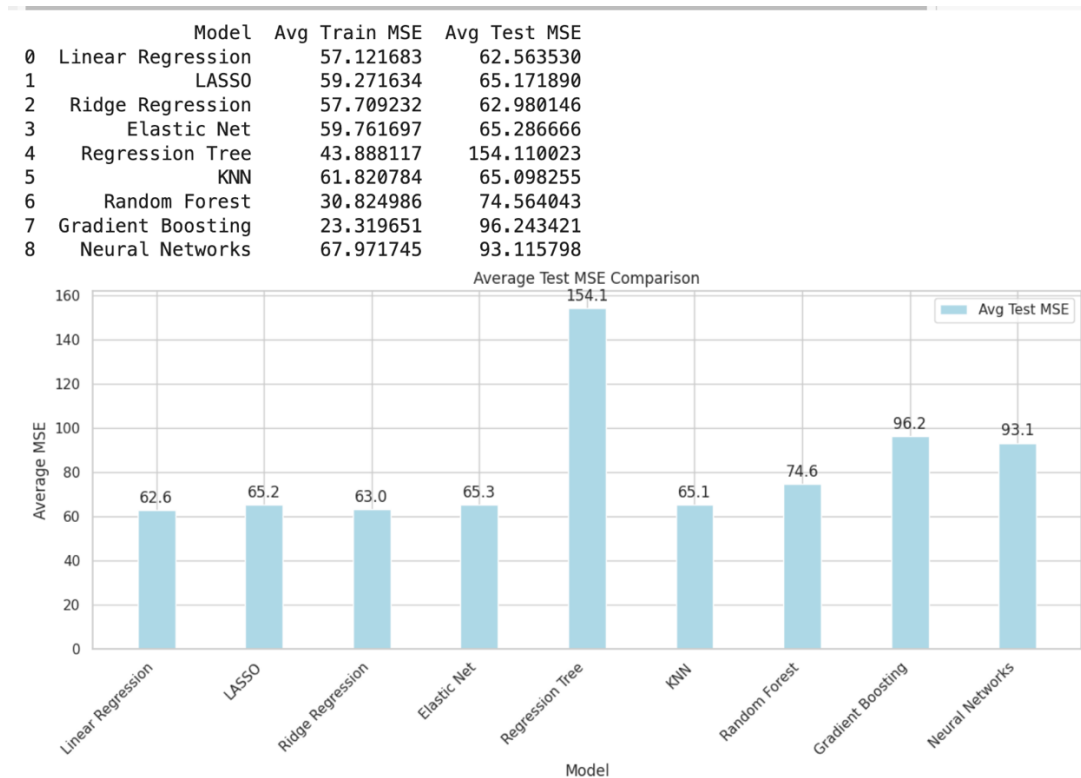


Figure 11

	Linear Regression	LASSO	Ridge Regression	Elastic Net	Regression Tree	KNN	Random Forest	Gradient Boosting	Neural Network
Capacity									
10	94.638889	94.638889	94.638889	94.638889	94.638889	94.638889	94.638889	94.638889	94.638889
15	81.305556	81.305556	81.305556	81.305556	81.305556	81.305556	81.305556	81.305556	81.305556
20	70.138889	70.138889	70.138889	70.138889	70.111111	70.138889	69.972222	70.138889	70.138889
25	59.472222	59.750000	59.666667	59.750000	60.500000	59.972222	59.638889	60.388889	59.972222
30	50.138889	50.388889	50.305556	50.416667	51.555556	50.388889	50.333333	51.722222	51.333333
35	41.527778	41.388889	41.555556	41.361111	43.611111	41.472222	41.722222	43.138889	43.666667
40	33.472222	32.944444	33.222222	32.916667	36.138889	32.916667	33.305556	34.194444	35.972222
45	25.805556	25.333333	25.611111	25.305556	29.611111	24.833333	25.416667	26.055556	28.194444
50	18.222222	18.055556	18.000000	18.055556	24.000000	17.888889	18.777778	19.583333	20.805556

Figure 12

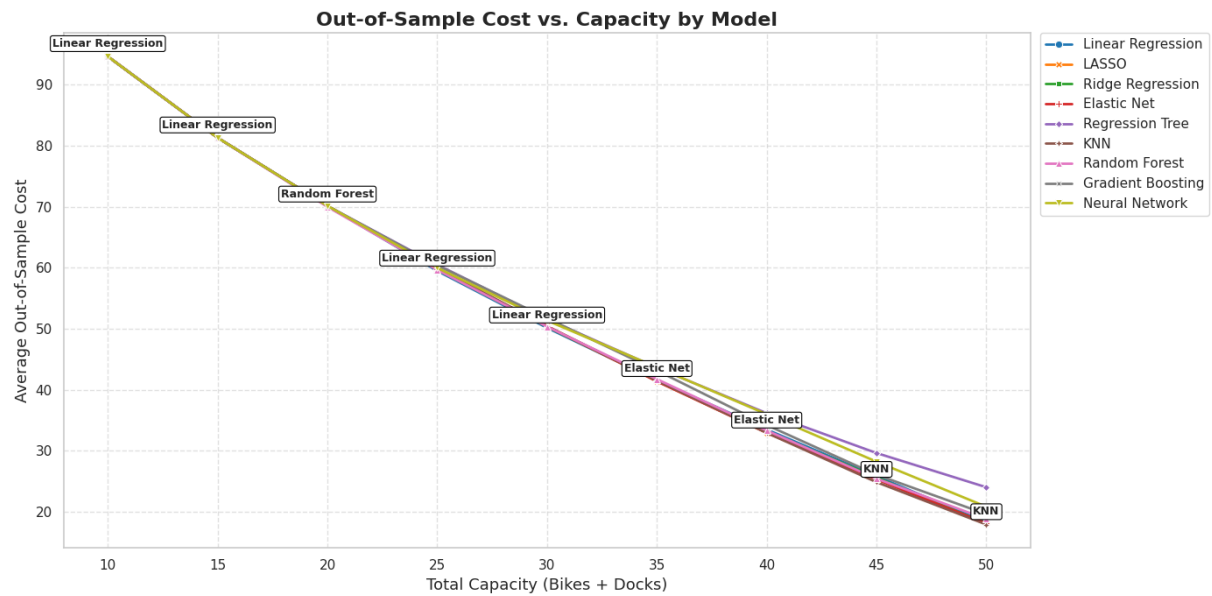


Figure 13

	Model	Best Params	Best CV MSE
0	Linear Regression (PU_ct)	{}	-inf
1	Linear Regression (DO_ct)	{}	-inf
2	Lasso (PU_ct)	0.911163	66.932512
3	Lasso (DO_ct)	1.097499	67.136574
4	Ridge (PU_ct)	10.0	70.284368
5	Ridge (DO_ct)	10.0	70.284368
6	Elastic Net (PU_ct)	{'alpha': 1.0, 'l1_ratio': 0.9}	67.336230
7	Elastic Net (DO_ct)	{'alpha': 1.0, 'l1_ratio': 0.9}	67.336230
8	KNN (PU_ct)	{'metric': 'euclidean', 'n_neighbors': 11, 'we...	71.149421
9	KNN (DO_ct)	{'metric': 'euclidean', 'n_neighbors': 11, 'we...	71.149421
10	Decision Tree (PU_ct)	{'max_depth': 3, 'min_samples_leaf': 3, 'min_s...	71.693178
11	Decision Tree (DO_ct)	{'max_depth': 3, 'min_samples_leaf': 4, 'min_s...	71.693178
12	Random Forest (PU_ct)	{'max_depth': 5, 'min_samples_leaf': 2, 'min_s...	68.838103
13	Random Forest (DO_ct)	{'max_depth': 3, 'min_samples_leaf': 4, 'min_s...	68.838103
14	Gradient Boosting (PU_ct)	{'learning_rate': 0.1, 'max_depth': 3, 'min_sa...	72.219455
15	Gradient Boosting (DO_ct)	{'learning_rate': 0.01, 'max_depth': 5, 'min_s...	72.219455
16	Neural Networks (PU_ct)	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	67.587453
17	Neural Networks (DO_ct)	{'activation': 'tanh', 'alpha': 0.01, 'hidden_...	67.587453

Figure 14

	Model	MSE (PU_ct)	MAPE (PU_ct)	R ² (PU_ct)	MSE (DO_ct)	MAPE (DO_ct)	R ² (DO_ct)	Avg Cost
0	Ridge Regression	55.580437	0.557821	0.336695	69.546437	0.465370	0.201317	76.638889
1	LASSO	55.625362	0.558412	0.336159	69.518944	0.465575	0.201633	76.638889

Figure 15

	Model	MSE (PU_ct)	MAPE (PU_ct)	R ² (PU_ct)	MSE (DO_ct)	MAPE (DO_ct)	R ² (DO_ct)	Avg Cost
0	KNN	59.727732	0.594146	0.287201	70.842057	0.47441	0.186438	17.305556

**Note: I have taken help from GenAI/Chat GPT in interpreting the results.*