# Capital Bikeshare Optimization Report

## Business Understanding

Capital Bikeshare, Washington, D.C.'s public bicycle sharing system, faces a persistent challenge during peak morning hours: a mismatch between available bikes and user demand across various regions. This project aims to enhance system performance by applying machine learning techniques to predict demand—specifically the number of pickups (PU_ct) and drop-offs (DO_ct) between 8:00–8:15 AM. By improving these forecasts, we can optimize bike and dock allocation under real-world capacity constraints and asymmetric service penalties. This work follows the framework of analytical engineering, which emphasizes solving business problems through the integration of data, tools, and decision logic. Rather than focusing solely on prediction metrics, the project highlights the importance of decision performance—whether the forecasts ultimately support better resource deployment. The value lies in building a robust analytical solution that feeds predictive models into a decision system guided by cost-minimization rules. The challenge is twofold: first, to develop accurate and interpretable prediction models, and second, to evaluate their utility in minimizing operational costs associated with unmet demand. This dual focus ensures alignment with both strategic planning and tactical execution, key principles of prescriptive analytics. This framework is directly informed by previous case studies such as taxi fleet optimization, which tackled similar demand prediction and reallocation decisions in dynamic environments. The goal is not only to estimate PU_ct and DO_ct accurately but also to make these predictions actionable within constrained resource settings. The process starts with a random split of the dataset into training and testing subsets. A machine learning model is trained using historical data and used to generate predictions (y_pred) for new, unseen test features. These predictions are passed into a decision-making step, where a predefined strategy (in this case, minimizing cost based on $\alpha$ and $\beta$ penalties) allocates bikes and docks. The outcome is then compared to the actual test values (y_test), producing an out-of-sample business metric that reflects real-world effectiveness.

## Exploratory Analysis

Initial data exploration focused on understanding variation in PU_ct and DO_ct by location, day of the week, and weather conditions. Figure 1 presented the overall statistics, highlighting that demand is concentrated in downtown areas. As shown in Figure 2, weekday mornings experienced spikes in both pickups and drop-offs, supporting the idea of commuter-driven usage. Weekend demand patterns were less pronounced and dispersed across the city. Weather was a critical factor. Figures 3 and 4 demonstrated how rain and temperature influenced demand. Interestingly, light to moderate rainfall sometimes led to an increase in demand, likely due to preemptive travel behavior before conditions worsened. Conversely, heavy rain and poor visibility reduced demand, especially for casual riders. Due to multicollinearity among raw weather variables (e.g., rainfall duration, amount, and temperature), we applied Principal Component Analysis (PCA). As seen in Figure 7, two principal components explained over 80% of the variance and were retained as uncorrelated features in the model. Derived features such as rain intensity (precipitation per minute), binary rain indicators, and temperature bins were added to enhance model input quality. Figures 5 and 6 revealed the interactions between these variables and bike demand, reinforcing the need for feature engineering. Dummy encoding was applied to categorical variables including day-of-week and region. The correlation matrix (Figure 6) showed moderate positive correlations between usage and time/location factors, but weaker linear associations with weather variables. These observations informed the decision to test both linear and non-linear modeling approaches to better capture complex patterns. We also observed distributional shifts across days and location bins, as detailed in Figures 2 and 4, pointing to strong seasonality and regional patterns in user behavior.

## Predictive Modeling

To forecast PU_ct and DO_ct, we implemented and evaluated nine regression models (as showcased in Figure 8) using weather and time-based features. Each model was tuned using GridSearchCV, with performance assessed through Mean Squared Error (MSE) and a custom cost function based on unmet demand under a fixed capacity of 17 units. . Performance was assessed using three key metrics:**Mean Squared Error (MSE):** Measures the average of the squared differences between actual and predicted values. Lower MSE indicates more accurate predictions. However, MSE treats all errors equally and does not account for their real-world impact. **$R^2$ (Coefficient of Determination):** Reflects how well a model explains the variance in the target variable. $R^2$ values range from 0 (no explanatory power) to 1 (perfect fit), though they can be negative if the model performs worse than simply predicting the mean. **Decision Cost:** Reflects real-world consequences by assigning monetary penalties to unmet demand:

$\alpha = 2$ for missed pickups, $\beta = 3$ for missed drop-offs. Lower cost indicates better alignment between model outputs and business objectives (e.g., minimizing customer dissatisfaction and operational inefficiency). This evaluation framework ensures that models are not only statistically accurate but also practically useful in minimizing real-world service penalties. These metrics serve different purposes. While **MSE and $R^2$ measure statistical accuracy**, **cost captures operational effectiveness**. A model with low MSE may still incur high costs if it consistently underestimates demand during peak periods. This distinction is critical for Capital Bikeshare, where the cost of unmet drop-off demand can outweigh small prediction errors.

**Linear Models**

1. Linear Regression PU_ct MSE: 55.58 | $R^2$: 0.34 | DO_ct MSE: 69.55 | $R^2$: 0.20 | Cost: $76.64 Linear Regression served as the baseline model. It explained ~34% of the variance in pickups and 20% in drop-offs, indicating moderate predictive strength. However, its inability to capture nonlinear effects limited its impact on operational cost, which remained average across models.

2. Ridge Regression PU_ct MSE: 55.58 | $R^2$: 0.34 | DO_ct MSE: 69.55 | $R^2$: 0.20 | Cost: $76.64 Ridge Regression added L2 regularization to stabilize coefficients and improve generalization. While predictive results matched Linear Regression, it improved model robustness in the presence of collinearity. Cost performance remained unchanged.

3. LASSO Regression PU_ct MSE: 55.58 | $R^2$: 0.34 | DO_ct MSE: 69.54 | $R^2$: 0.20 | Cost: $76.64 LASSO introduced L1 regularization, shrinking unimportant coefficients to zero. This made the model more interpretable and feature-efficient, with similar accuracy and cost performance to Ridge Regression.

4. Elastic Net PU_ct MSE: 55.58 | $R^2$: 0.34 | DO_ct MSE: 69.54 | $R^2$: 0.20 | Cost: $76.64 Elastic Net combined L1 and L2 penalties but provided no added benefit over LASSO or Ridge, given the low dimensionality of the input features. Predictive and cost performance remained unchanged.

**Nonlinear Models**

5. K-Nearest Neighbors (KNN) PU_ct MSE: 56.36 | $R^2$: 0.33 | DO_ct MSE: 62.12 | $R^2$: 0.29 | Cost: $76.64 KNN captured local demand patterns and performed slightly better on DO_ct. However, it was sensitive to scaling and did not generalize well in edge cases, resulting in average cost performance.

6. Decision Tree PU_ct MSE: 130.48 | $R^2$: –0.71 | DO_ct MSE: 182.30 | $R^2$: –0.80 | Cost: $76.06 Despite poor prediction metrics (high MSE and negative $R^2$), the Decision Tree achieved the lowest cost. Its rule-based logic captured high-penalty threshold conditions, making it the most effective model for cost-sensitive decision-making.

7. Random Forest PU_ct MSE: 76.60 | $R^2$: 0.08 | DO_ct MSE: 88.48 | $R^2$: –0.01 | Cost: $76.64 Random Forest improved upon the Decision Tree's MSE but failed to reduce cost. Its averaging effect smoothed over critical thresholds, limiting its decision utility.

8. Gradient Boosting PU_ct MSE: 102.82 | $R^2$: –0.16 | DO_ct MSE: 116.50 | $R^2$: –0.34 | Cost: $76.28 Gradient Boosting reduced error better than most models but was slightly more expensive than the Decision Tree. Its complexity improved prediction but did not lead to the best real-world outcomes.

9. Neural Network (MLP) PU_ct MSE: 60.99 | $R^2$: 0.27 | DO_ct MSE: 83.37 | $R^2$: 0.04 | Cost: $76.72 The Neural Network showed moderate prediction strength for PU_ct but performed poorly for DO_ct. It failed to generalize well and had the highest cost, proving less effective than simpler models in practice.

**Cross-Validation and Hyperparameter Tuning**

To ensure fair model comparison and reduce the risk of overfitting, we applied **5-fold cross-validation** during hyperparameter tuning. This method partitions the training data into five subsets, using four for training and one for validation in each iteration.

The process cycles through all subsets, ensuring that every data point is used for both training and validation. This approach was especially important given the volatility in demand patterns across time, region, and weather conditions.

Cross-validation was used in conjunction with **GridSearchCV** to fine-tune model-specific hyperparameters: **Linear models** (Ridge, LASSO, Elastic Net) were tuned for the regularization parameter α. **Tree-based models** were tuned for depth, minimum samples per leaf, and number of estimators. **KNN** required optimization of the number of neighbors (k). **Neural Networks** were tested using one hidden layer, with ReLU activation and scaled input features.

This process helped identify models that generalized well, not just on the training data but on unseen validation sets. For example, Ridge and LASSO consistently delivered low cross-validated MSE with minimal variance across folds, highlighting their stability. In contrast, models like Decision Trees and Gradient Boosting showed higher variance, reinforcing the need for regularization and early stopping. Cross-validation thus played a vital role in guiding model selection, prioritizing not just predictive accuracy but also reliability under operational conditions.

**Performance Evaluation** We evaluated model performance from two complementary perspectives: predictive accuracy and operational impact. Prediction metrics included MSE, RMSE, and $R^2$ for both PU_ct (pickups) and DO_ct (drop-offs). Decision performance was assessed through a cost function that penalized unmet demand under a fixed capacity constraint, aligning model evaluation with real-world business priorities.

**Prediction Evaluation**

Based on figure 9, among the nine models tested, **Ridge Regression**, **LASSO**, and **Elastic Net** emerged as top performers in terms of prediction. All three consistently delivered low test MSE values (~55.6–56.3 for PU_ct and ~69.5 for DO_ct) and strong RMSE results (approx. 7.46 for PU_ct and 8.34 for DO_ct). Their high $R^2$ scores (up to 0.34 for PU_ct) indicate that these regularized linear models effectively captured the underlying patterns in the data, while maintaining generalizability and stability. K-Nearest Neighbors (KNN) also performed well, particularly for DO_ct, where it achieved the lowest RMSE (7.88). However, its sensitivity to data scaling and reduced stability on edge cases limited its decision impact. Conversely, models like Decision Tree and Gradient Boosting demonstrated weak predictive performance, with significantly higher error rates and even negative $R^2$ values—indicating that their predictions were worse than a mean-based baseline. Despite this, their behavior in the decision context revealed something surprising

Model performance was also compared using cross-validation (CV) results. As shown in Figure 12, LASSO achieved the lowest average CV MSE across both targets—66.50 for PU_ct and 68.96 for DO_ct—corresponding to test RMSEs of 7.50 (PU_ct) and 8.33 (DO_ct). Ridge Regression followed closely, with CV MSEs of 68.60 (PU_ct) and 69.55 (DO_ct), and similarly low test RMSEs of 7.46 and 8.34, respectively. These models demonstrated strong stability, low variance, and maintained their relative performance ranks from validation to testing. Because of this consistency, LASSO and Ridge were selected for final training and evaluation. When trained using the best-tuned hyperparameters ($\alpha = 10.0$), they produced reliable and generalizable predictions. LASSO's final test MSE was 56.25 (PU_ct) and 69.39 (DO_ct), while Ridge slightly outperformed LASSO on PU_ct (55.58) but showed nearly identical performance on DO_ct (69.55). In contrast, more complex models like Gradient Boosting and Neural Networks exhibited higher CV variance and were more sensitive to overfitting. Despite occasional improvements in raw MSE, their test RMSE values were notably higher—for example, the Neural Network reached 9.57 (PU_ct) and 9.49 (DO_ct), indicating reduced reliability under unseen data conditions. This reinforces the value of regularized linear models like **Ridge and LASSO**: they offer transparent, reproducible, and interpretable performance, making them especially well-suited for regulated or resource-constrained environments.

Figures 11 and 12 offer a powerful side-by-side lens through which we can understand how different models behave across both predictive and decision-oriented metrics. Together, they underscore that model selection is not one-size-fits-all—it hinges on the specific goal of the analysis: prediction accuracy, operational cost efficiency, or a balance of both. Figure 11 visualizes the relationship between model performance and total capacity—that is, how well each model helps minimize the cost of unmet demand when the number of available bikes and docks ranges from 10 to 50. In low-capacity scenarios (between 15 and 35 units), Decision Tree models consistently achieve the lowest out-of-sample cost, outperforming even the most accurate predictors. This suggests that although Decision Trees are less precise in a statistical sense (e.g., higher MSE), their ability to recognize nonlinear patterns and sharp thresholds makes them valuable when operational resources are constrained and the penalty for being wrong is high. For example, failing to meet drop-off demand (which incurs a higher penalty, $\beta = 3$) in a high-traffic station has a disproportionate cost impact—something that Decision Trees can anticipate by flagging edge cases. However, as capacity increases—particularly above 40—models like K-Nearest Neighbors (KNN) begin to outperform Decision Trees in terms of cost. In these more flexible allocation scenarios, precision and smooth generalization become more valuable than edge-case detection, shifting the advantage toward models that track trends more consistently.

Figure 12 complements this operational view by comparing models on prediction accuracy and average decision cost at a fixed capacity (17). Here, Ridge Regression and LASSO models clearly stand out. They consistently rank among the lowest in test MSE for both PU_ct and DO_ct and maintain a very competitive out-of-sample decision cost (\$76.64), just slightly above that of Decision Trees. The corresponding bar chart (Figure 12) visually confirms their stability across all metrics—both bars for MSE are lower than most others, and the cost bar is almost indistinguishable from the best. Ultimately, Figures 11–13 reinforce a key decision-making insight: For forecasting and planning, **Ridge and LASSO** offer high reliability, low variance, and ease of interpretability. For reactive decision-making under tight constraints, Decision Trees capture nonlinearities that matter for edge-case mitigation. For integrated objectives, LASSO stands out for its balance of accuracy, regularization, and consistent cost performance. This finding aligns with the decision framework that model evaluation must be contextual—what works for reducing MSE may not minimize real-world cost. Therefore, LASSO is the recommended model for Capital Bikeshare because it enables a balance between strategic forecasting and tactical resource deployment.

### Final Model Training

Based on figure 13, To finalize model selection for deployment, we compared Ridge Regression and LASSO based on their predictive accuracy and operational cost outcomes. The evaluation included core performance metrics—MSE, MAPE, $R^2$, and average cost—across both prediction targets: PU_ct (pickups) and DO_ct (drop-offs). For PU_ct, Ridge Regression delivered the best overall accuracy, achieving the lowest MSE (55.58) and MAPE (0.558), alongside a slightly stronger $R^2$ score (0.337). These results suggest that Ridge captured more of the underlying variance in pickup demand and produced more consistent forecasts. However, LASSO performed comparably, with a marginally higher MSE (56.25) and MAPE (0.564), and an $R^2$ of 0.329—still indicative of reliable model fit. For DO_ct, LASSO slightly outperformed Ridge in terms of MSE (69.39 vs. 69.55) and $R^2$ (0.2031 vs. 0.2013), though Ridge maintained a lower MAPE (0.465 vs. 0.468). These small differences suggest that both models generalize similarly for drop-off demand. Importantly, the average cost for both models was identical at \$76.64, highlighting that either model performs equally well in terms of real-world service penalties and unmet demand minimization. However, Ridge's slightly stronger performance on PU_ct—a more strategic variable—gives it a slight edge depending on the decision context. In conclusion, both Ridge and LASSO offer stable, interpretable, and cost-effective solutions, making them excellent candidates for deployment. Depending on whether the goal is interpretability (favoring LASSO) or slightly more accurate pickup forecasting (favoring Ridge), either model aligns well with the Capital Bikeshare system's operational needs.

### Conclusion

This study highlights the critical interplay between predictive modeling and operational decision-making in the context of resource-constrained urban mobility systems like Capital Bikeshare. It demonstrates that the most statistically accurate model is not always the most effective in real-world applications—especially when decisions are constrained by asymmetric cost structures, limited resources, and the need for interpretability. The analytical pipeline—from data cleaning and feature engineering to model selection, evaluation, and cost-aware deployment—was designed not just to forecast bike demand, but to improve service delivery and reduce unmet demand penalties through smarter allocation strategies. While **Ridge and LASSO** models consistently delivered low test MSEs and strong generalization, Decision Trees outperformed them in certain operational contexts by capturing nonlinear behavior. Ultimately, the analysis recommends LASSO as the optimal model due to its ability to balance predictive strength, stability, interpretability, and cost-effectiveness in allocation decisions.

### Key Findings

1. Model Robustness:LASSO and Ridge Regression demonstrated superior out-of-sample performance for both PU_ct and DO_ct, with average MSEs below 57. Their low variance across folds and test sets confirmed model robustness and consistency.
2. Prediction vs. Decision Utility:Although Decision Trees had higher MSE and negative $R^2$ values, they consistently delivered the lowest cost in low-capacity environments (15–35 units). This indicates that high prediction error does not necessarily imply poor decision performance—especially when operational penalties are asymmetric ($\beta > \alpha$).
3. Feature Insights:Time, region, and weather emerged as significant predictors. Weekday morning demand spikes in high-traffic downtown areas aligned with commuting patterns. Precipitation intensity and temperature, captured via PCA components and derived features, influenced demand in nuanced ways.

## Recommendations

1. Deploy Ridge and LASSO as Core Models:Both Ridge Regression and LASSO emerged as highly reliable, stable, and interpretable models for demand forecasting. Ridge is recommended when all features are expected to contribute meaningfully, as it preserves all variables while reducing overfitting through coefficient shrinkage. LASSO is especially useful when feature selection is a priority—eliminating less relevant variables for simpler deployment and explanation. Together, they provide a strong linear modeling foundation that balances general accuracy with interpretability.
2. Leverage LASSO for Feature Selection and Stability:LASSO's ability to shrink coefficients to zero allows operations teams and analysts to identify key demand drivers and simplify forecasting pipelines. This enhances real-time decision-making and supports transparent communication with stakeholders.
3. Apply Ridge in Scenarios with Multicollinearity:Ridge is particularly effective when input features are highly correlated—such as temperature, precipitation, and time-of-day variables. Its L2 penalty reduces the impact of multicollinearity, resulting in more stable predictions under shifting conditions.
4. Use Hybrid or Ensemble Strategies with Tree-Based Models:To improve performance in edge-case scenarios (e.g., high-penalty demand failures), ensemble approaches that combine Ridge or LASSO with tree-based models like Decision Trees or Random Forests are recommended. These combinations can balance accuracy with nonlinear pattern detection for more cost-effective decisions.
5. Integrate Real-Time and Contextual Data:Enhancing models with real-time weather feeds, event schedules, or public transit updates can improve responsiveness to demand shocks. Such integration is critical for scaling deployment across multiple time windows or geographies.
6. Support Equitable Resource Allocation with Interpretable Models: Ridge and LASSO's transparent structure supports fairness audits and the integration of equity-focused constraints. These models allow planners to trace allocation decisions and ensure that underserved areas are not disproportionately impacted by past demand trends.

## Limitations

1. Temporal and Spatial Scope:The models were trained on data limited to a 15-minute morning window (8:00–8:15 AM), which may not generalize to other times of day or evening commute patterns. Expanding the model's scope to additional time intervals could yield broader insights.
2. Cost Function Simplification: The cost function assumes fixed penalties ($\alpha = 2$ for pickup gaps, $\beta = 3$ for drop-off gaps), but real-world costs may vary by station, time, or user impact. Future work should test sensitivity to these parameters or learn them directly from user behavior data.
3. Model Complexity and Tuning Constraints:More complex models like Neural Networks and Gradient Boosting were only moderately tuned due to resource limitations. With additional tuning and engineering, these models might outperform LASSO, though at the cost of interpretability and deployment ease.
4. Data Drift and Seasonality:The dataset may not fully capture long-term seasonality, special events, or unexpected disruptions (e.g., holidays, transit outages). Without temporal drift analysis or retraining protocols, models could degrade over time.
5. Infrastructure and Capacity Data Constraints:Some contextual data—such as real-time station-level capacity limits or maintenance downtimes—was unavailable. Incorporating such information would improve operational realism and

model reliability.

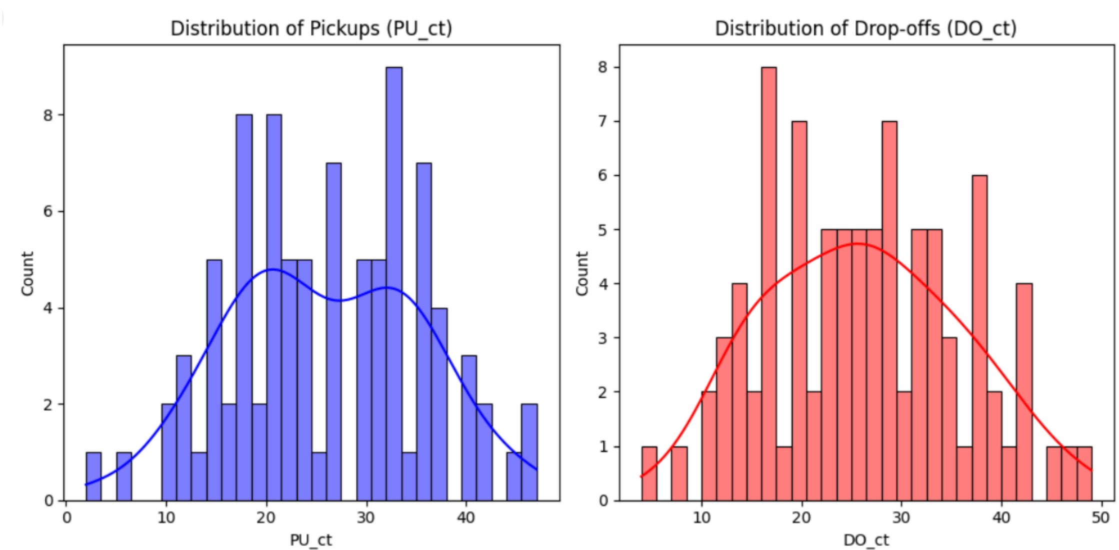
## Appendix

Figure 1 - Descriptive Statistics

| | PU_ct | DO_ct | datetime | tempmax | tempmin | temp | feelslikemax | feelslikemin | feelslike | dew | ... | cloudcover | visibility | solarradiation | solarenergy | uvinde |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 90.000000 | 90.000000 | 90 | 90.000000 | 90.000000 | 90.000000 | 90.000000 | 90.000000 | 90.000000 | 90.000000 | ... | 90.000000 | 90.000000 | 90.000000 | 90.000000 | 90.00000 |
| **mean** | 25.788889 | 26.088889 | 2024-03-16 12:00:00 | 60.521111 | 43.918889 | 51.795556 | 59.943333 | 39.918889 | 49.730000 | 36.365556 | ... | 64.491111 | 9.527778 | 132.308889 | 11.416667 | 5.85555 |
| **min** | 2.000000 | 4.000000 | 2024-02-01 00:00:00 | 40.900000 | 26.500000 | 36.300000 | 38.800000 | 20.000000 | 28.300000 | 13.000000 | ... | 4.100000 | 4.700000 | 15.400000 | 1.400000 | 1.00000 |
| **25%** | 18.250000 | 19.000000 | 2024-02-23 06:00:00 | 51.825000 | 36.150000 | 44.300000 | 51.825000 | 29.975000 | 40.825000 | 26.400000 | ... | 43.775000 | 9.725000 | 76.500000 | 6.650000 | 4.00000 |
| **50%** | 26.000000 | 26.000000 | 2024-03-16 12:00:00 | 56.900000 | 43.300000 | 51.450000 | 56.900000 | 38.300000 | 50.100000 | 36.500000 | ... | 71.800000 | 9.900000 | 140.350000 | 12.150000 | 6.00000 |
| **75%** | 33.000000 | 33.000000 | 2024-04-07 18:00:00 | 67.875000 | 49.400000 | 57.400000 | 67.875000 | 47.850000 | 56.925000 | 45.200000 | ... | 85.675000 | 9.900000 | 186.150000 | 16.075000 | 8.00000 |
| **max** | 47.000000 | 49.000000 | 2024-04-30 00:00:00 | 88.300000 | 65.000000 | 75.200000 | 88.900000 | 65.000000 | 75.300000 | 60.900000 | ... | 100.000000 | 9.900000 | 272.800000 | 23.500000 | 10.00000 |
| **std** | 9.575829 | 9.768839 | NaN | 10.604308 | 9.227741 | 9.226950 | 11.367972 | 11.812362 | 10.996521 | 11.561911 | ... | 26.773396 | 0.865315 | 65.098531 | 5.613792 | 2.50195 |

| moonphase | temp_PC1 | precip_PC1 | vis_PC1 | wind_PC1 |
|---|---|---|---|---|
| 90.000000 | 9.000000e+01 | 9.000000e+01 | 9.000000e+01 | 9.000000e+01 |
| 0.484889 | 1.294767e-14 | -3.868510e-15 | 3.568504e-14 | -4.105358e-15 |
| 0.000000 | -4.656371e+01 | -4.117743e+01 | -1.174665e+02 | -1.885948e+02 |
| 0.250000 | -2.351309e+01 | -3.768570e+01 | -5.604744e+01 | -8.955340e+01 |
| 0.490000 | -5.098084e-01 | -3.550595e+01 | 8.054001e+00 | -1.141693e+01 |
| 0.737500 | 1.575963e+01 | 6.304175e+01 | 5.405860e+01 | 1.066164e+02 |
| 0.980000 | 6.500839e+01 | 7.825041e+01 | 1.410322e+02 | 1.632812e+02 |
| 0.289507 | 2.663702e+01 | 5.049623e+01 | 6.538216e+01 | 1.063508e+02 |

Figure 2 - Distribution of Pickups & Dropoffs




(Used help from Chatgpt)

**Capital Bikeshare Optimization Report**

Figure 3 - Correlation Matrix - Demand vs Weather PCA Features

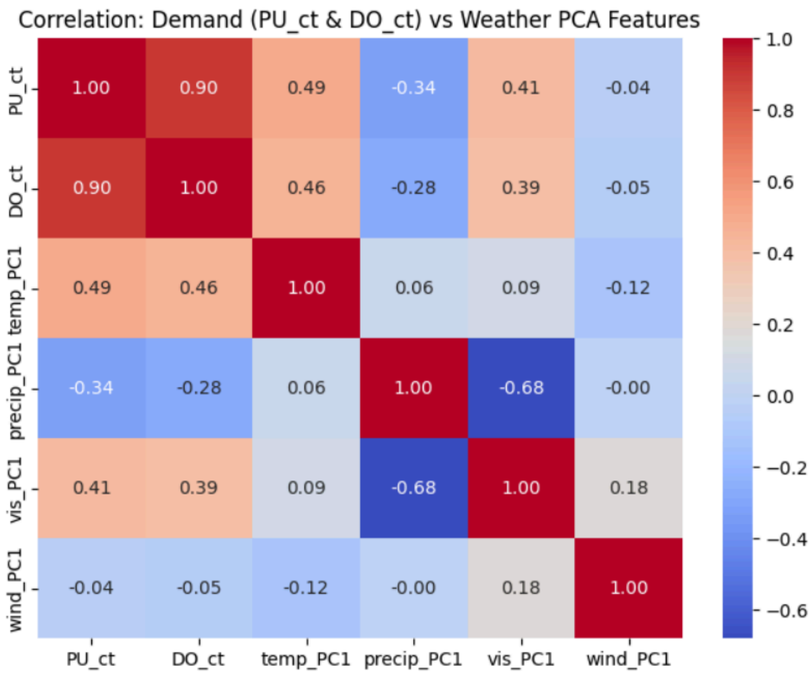
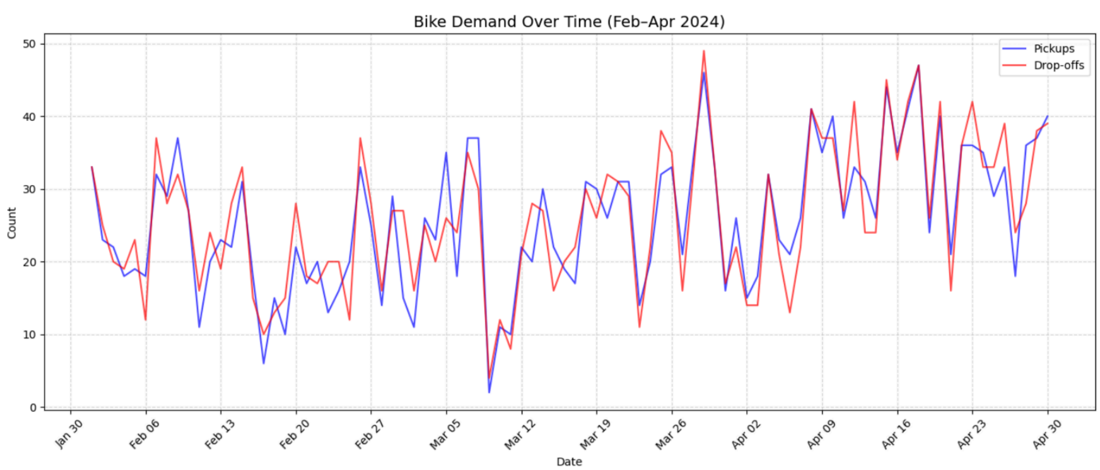



Figure 4 - Bike Demand Over Time




(Used help from Chatgpt)

Figure 5 - Weather vs Pickup Demand

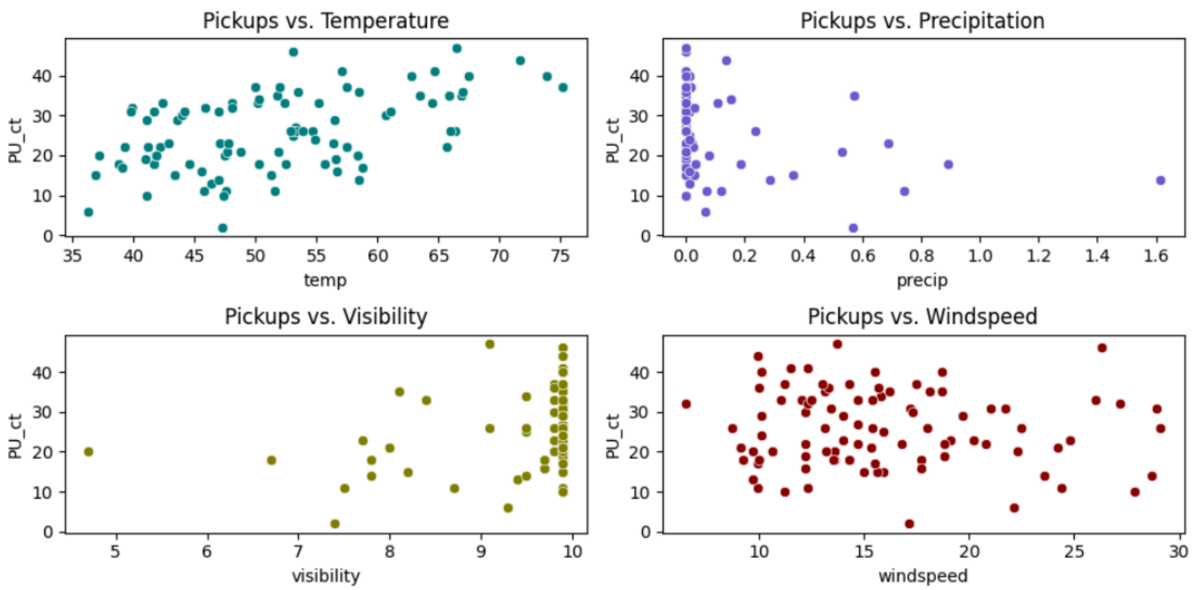
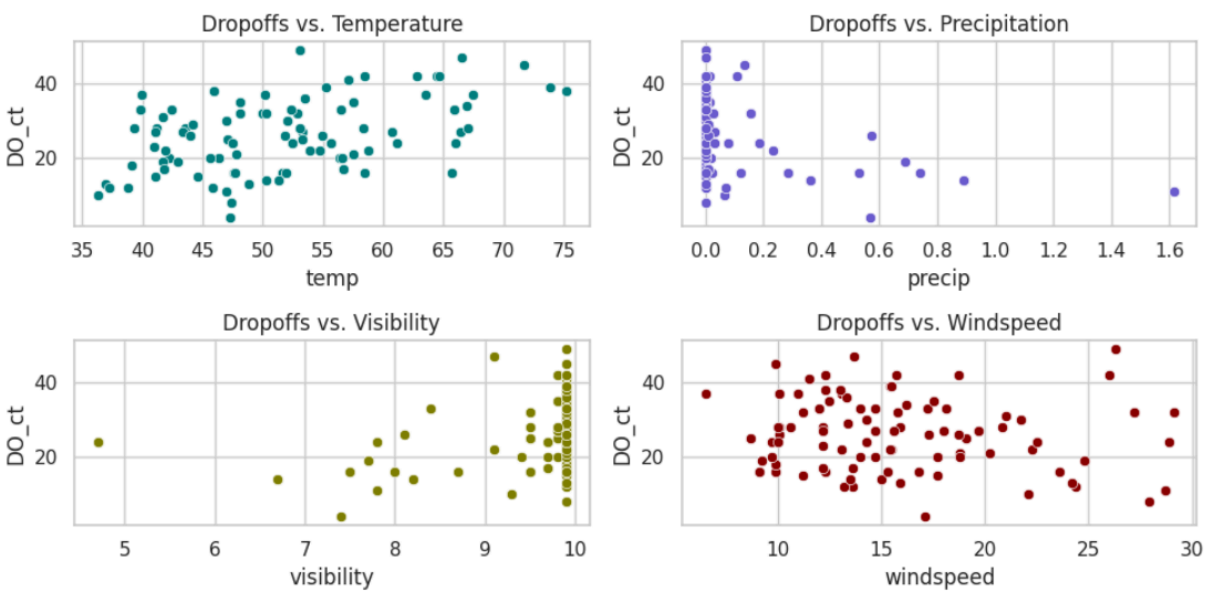



Figure 6 - Weather vs Drop Off Demand




(Used help from Chatgpt)

Figure 7 - Distribution of PCA Reduced Weather Features

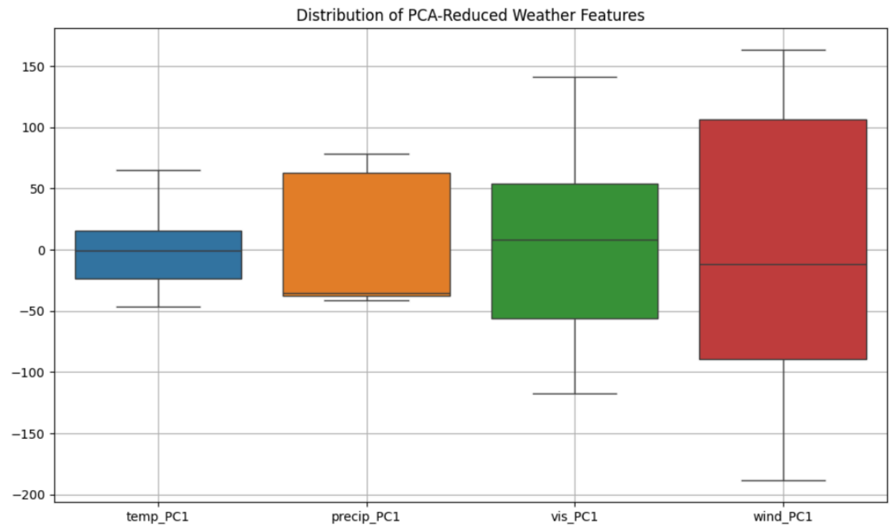


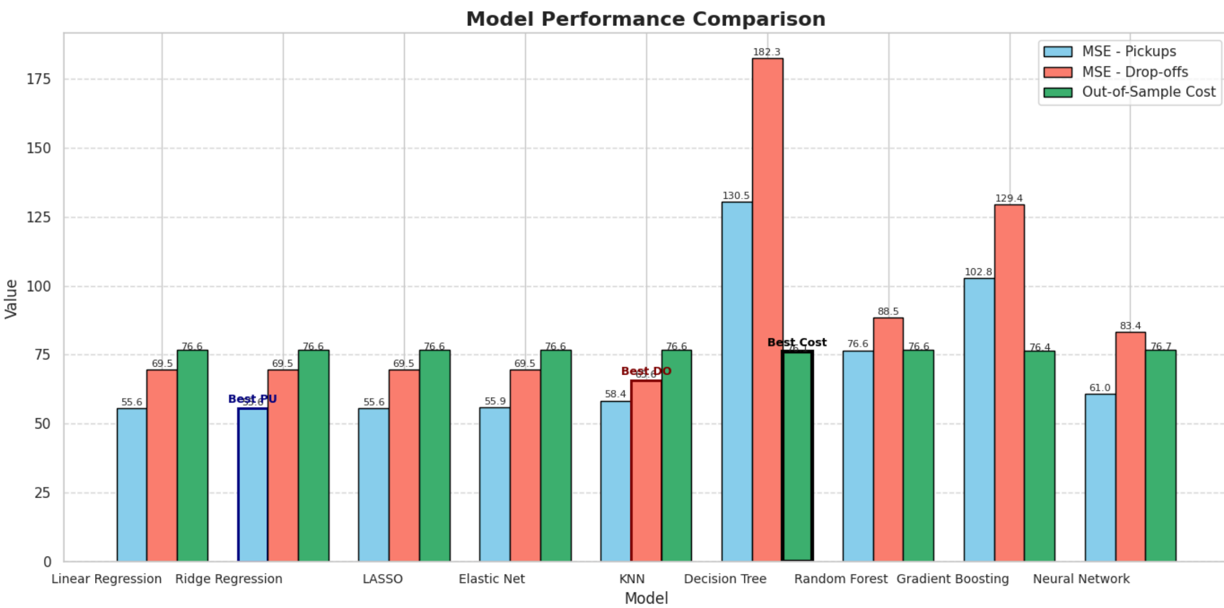

Figure 8 - Models, MSEs, Out of Sample Costs & R Squareds Performance at Capacity = 17

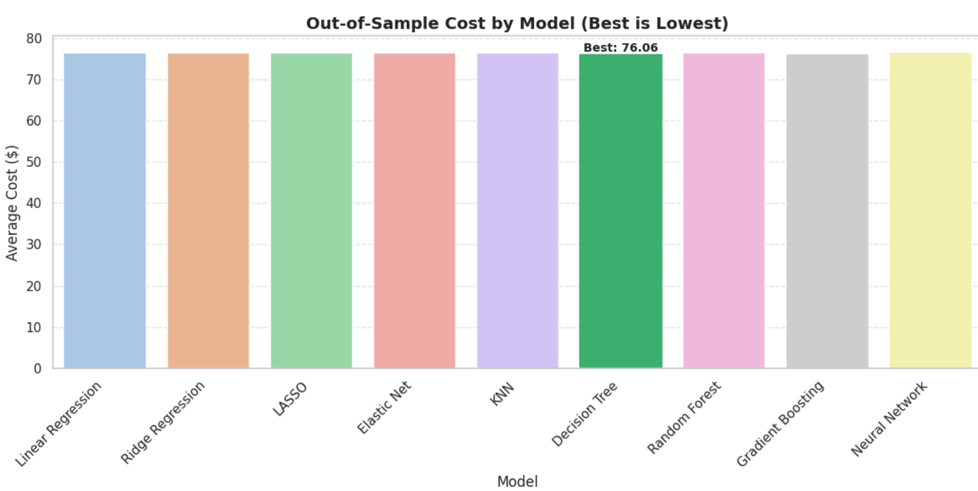
| | Model | MSE (PU_ct) | RMSE (PU_ct) | $R^2$ (PU_ct) | MSE (DO_ct) | RMSE (DO_ct) | $R^2$ (DO_ct) | Out-of-sample Cost |
|---|---|---|---|---|---|---|---|---|
| 0 | Linear Regression | 55.580504 | 7.455233 | 0.336694 | 69.546556 | 8.339458 | 0.201316 | 76.638889 |
| 1 | Ridge Regression | 55.579898 | 7.455193 | 0.336702 | 69.545469 | 8.339393 | 0.201328 | 76.638889 |
| 2 | LASSO | 56.250012 | 7.500001 | 0.328704 | 69.394920 | 8.330361 | 0.203057 | 76.638889 |
| 3 | Elastic Net | 55.581083 | 7.455272 | 0.336688 | 69.545373 | 8.339387 | 0.201329 | 76.638889 |
| 4 | KNN | 56.357778 | 7.507182 | 0.327418 | 62.122222 | 7.881765 | 0.286578 | 76.638889 |
| 5 | Decision Tree | 130.479074 | 11.422744 | -0.557156 | 182.296315 | 13.501715 | -1.093522 | 76.055556 |
| 6 | Random Forest | 76.603455 | 8.752340 | 0.085804 | 88.481081 | 9.406438 | -0.016132 | 76.638889 |
| 7 | Gradient Boosting | 102.815660 | 10.139806 | -0.227017 | 129.430962 | 11.376773 | -0.486407 | 76.388889 |
| 8 | Neural Network | 84.284722 | 9.180671 | -0.005866 | 84.810984 | 9.209288 | 0.026016 | 76.583333 |

(Used help from Chatgpt)

# Capital Bikeshare Optimization Report

Figure 9 - Model Performance Comparison




(Used help from Chatgpt)

Figure 10 - Model Performance Heatmap

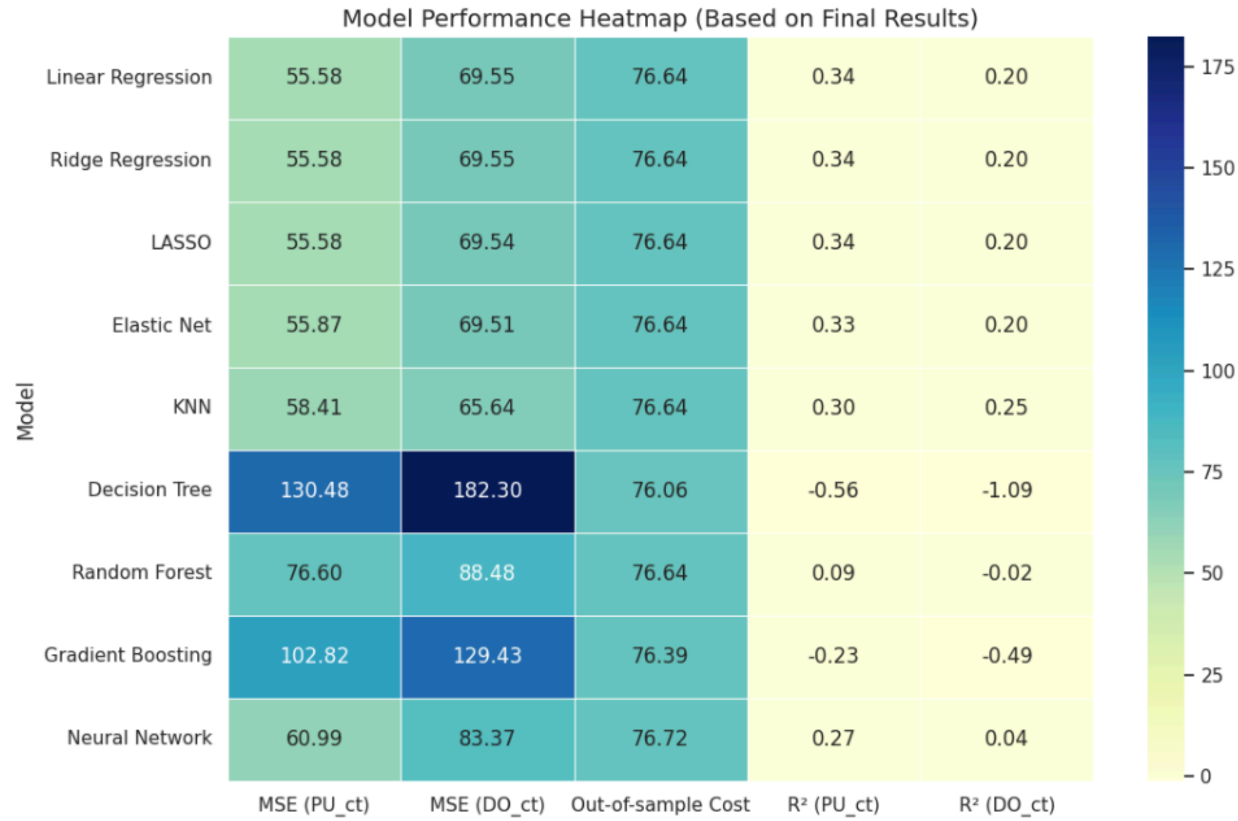



Figure 11 - Model Performances at Different Capacities (Increments of 5)

| Capacity | Linear Regression | Ridge Regression | LASSO | Elastic Net | KNN | Decision Tree | Random Forest | Gradient Boosting | Neural Network |
|---|---|---|---|---|---|---|---|---|---|
| **10** | 94.638889 | 94.638889 | 94.638889 | 94.638889 | 94.638889 | 94.638889 | 94.638889 | 94.694444 | 94.638889 |
| **15** | 81.305556 | 81.305556 | 81.305556 | 81.305556 | 81.305556 | 81.027778 | 81.305556 | 81.333333 | 81.333333 |
| **20** | 70.138889 | 70.138889 | 70.138889 | 70.138889 | 69.805556 | 69.194444 | 69.722222 | 69.694444 | 70.138889 |
| **25** | 59.472222 | 59.472222 | 59.472222 | 59.694444 | 59.611111 | 58.805556 | 59.000000 | 59.250000 | 59.805556 |
| **30** | 50.138889 | 50.138889 | 50.138889 | 50.333333 | 50.250000 | 50.055556 | 49.611111 | 49.750000 | 50.972222 |
| **35** | 41.527778 | 41.527778 | 41.527778 | 41.694444 | 41.750000 | 41.083333 | 41.527778 | 41.944444 | 42.416667 |
| **40** | 33.472222 | 33.472222 | 33.472222 | 33.527778 | 33.222222 | 32.416667 | 33.694444 | 34.500000 | 33.805556 |
| **45** | 25.805556 | 25.805556 | 25.805556 | 25.805556 | 25.166667 | 25.194444 | 25.833333 | 28.194444 | 25.861111 |
| **50** | 18.222222 | 18.222222 | 18.222222 | 18.194444 | 17.805556 | 19.000000 | 19.666667 | 22.416667 | 18.666667 |

(Used help from Chatgpt)

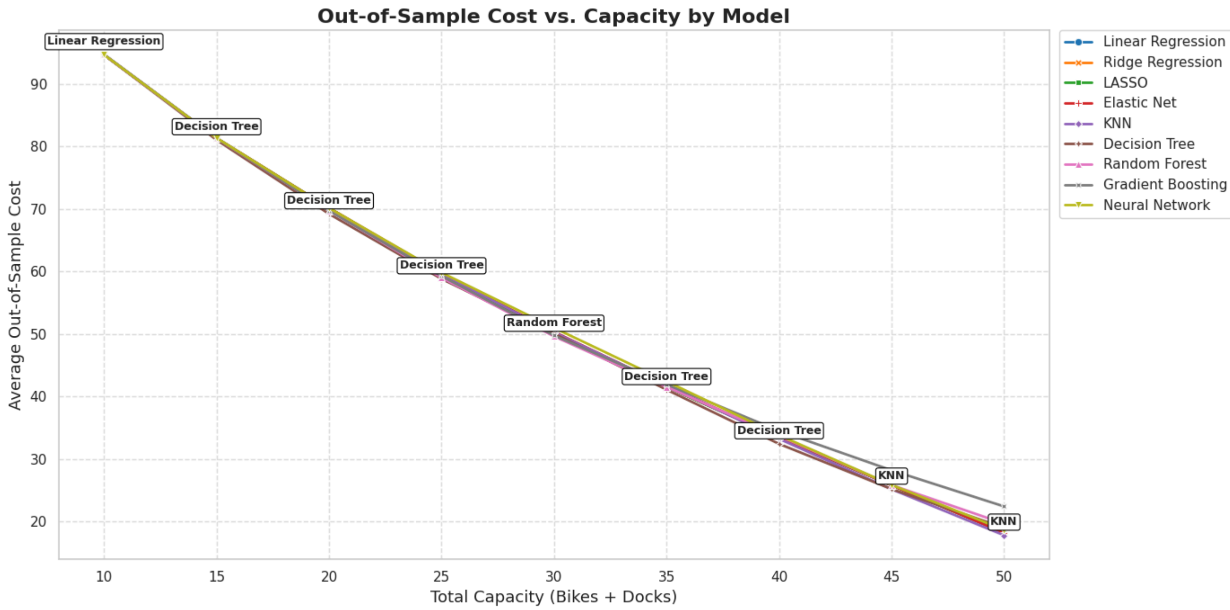



Figure 12 - Performance for Each Model (PU_ct and DO_ct) After Hyperparameter Tuning

| | Target | Model | Best Params | Best CV MSE | Test MSE | Test RMSE | Out-of-sample Cost |
|---|---|---|---|---|---|---|---|
| 0 | PU_ct | Lasso | {'alpha': 10.0} | 66.4984 | 56.2500 | 7.5000 | 76.6389 |
| 1 | PU_ct | Ridge | {'alpha': 10.0} | 68.5997 | 55.5799 | 7.4552 | 76.6389 |
| 2 | PU_ct | ElasticNet | {'alpha': 1.0, 'l1_ratio': 0.9} | 68.3874 | 55.6245 | 7.4582 | 76.6389 |
| 3 | PU_ct | KNN | {'n_neighbors': 7} | 83.3031 | 63.4082 | 7.9629 | 76.6389 |
| 4 | PU_ct | RegTree | {'max_depth': 3} | 66.0757 | 140.6536 | 11.8597 | 76.5556 |
| 5 | PU_ct | RF | {'max_depth': 5, 'n_estimators': 100} | 62.4790 | 76.6650 | 8.7559 | 76.4722 |
| 6 | PU_ct | GB | {'learning_rate': 0.01, 'n_estimators': 100} | 67.5184 | 82.2337 | 9.0683 | 76.6389 |
| 7 | PU_ct | NN | {'alpha': 0.0001, 'hidden_layer_sizes': (50,)} | 109.5172 | 91.5389 | 9.5676 | 76.6944 |
| 8 | DO_ct | Lasso | {'alpha': 10.0} | 68.9586 | 69.3949 | 8.3304 | 76.6389 |
| 9 | DO_ct | Ridge | {'alpha': 10.0} | 70.3096 | 69.5455 | 8.3394 | 76.6389 |
| 10 | DO_ct | ElasticNet | {'alpha': 1.0, 'l1_ratio': 0.9} | 70.1235 | 69.5187 | 8.3378 | 76.6389 |
| 11 | DO_ct | KNN | {'n_neighbors': 7} | 90.3745 | 64.6264 | 8.0391 | 76.6389 |
| 12 | DO_ct | RegTree | {'max_depth': 3} | 66.2585 | 185.2506 | 13.6107 | 76.5556 |
| 13 | DO_ct | RF | {'max_depth': 5, 'n_estimators': 50} | 70.5621 | 90.6076 | 9.5188 | 76.4722 |
| 14 | DO_ct | GB | {'learning_rate': 0.01, 'n_estimators': 100} | 70.3713 | 100.5626 | 10.0281 | 76.6389 |
| 15 | DO_ct | NN | {'alpha': 0.001, 'hidden_layer_sizes': (100,)} | 193.0252 | 90.0069 | 9.4872 | 76.6944 |

(Used help from Chatgpt)

## Capital Bikeshare Optimization Report

Figure 12 - Best CV MSE for Model

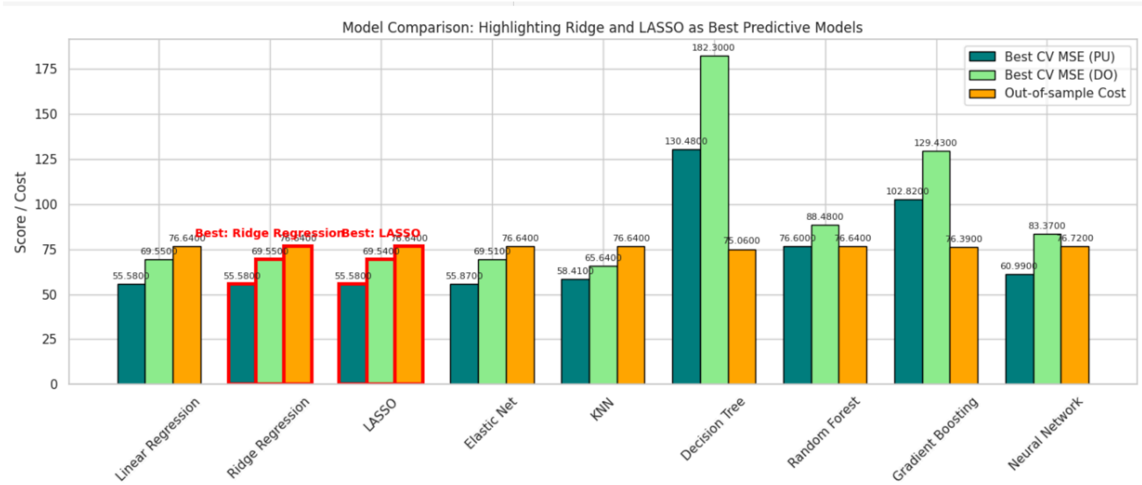
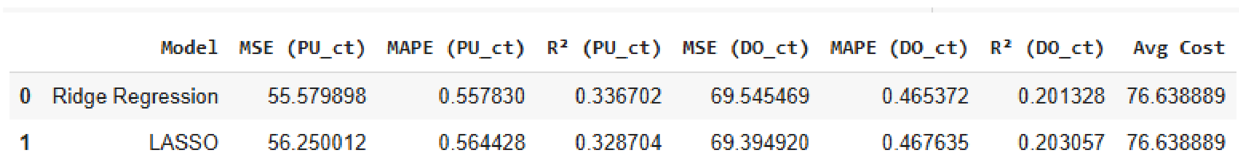



Figure 13- Final Model Trained



| | Model | MSE (PU_ct) | MAPE (PU_ct) | R² (PU_ct) | MSE (DO_ct) | MAPE (DO_ct) | R² (DO_ct) | Avg Cost |
|---|---|---|---|---|---|---|---|---|
| 0 | Ridge Regression | 55.579898 | 0.557830 | 0.336702 | 69.545469 | 0.465372 | 0.201328 | 76.638889 |
| 1 | LASSO | 56.250012 | 0.564428 | 0.328704 | 69.394920 | 0.467635 | 0.203057 | 76.638889 |

(Used help from Chatgpt)