Machine Learning Project Technical Report

Authors: Skylar Harrison, Cynthia Kaye, Curtis Moore, Rebecca Rickard

Course: CSCI/DASC 6020

Date: 12/2/2025

Table of Contents

1. Executive Summary

This project is a Machine Learning solution to a toy problem. The purpose is to replicate the outputs of a legacy system for travel reimbursement. The data was historical reimbursement cases, which the model was trained on. It was found the model could replicate the outputs at a high rate, with an R^2 score of 0.85. This shows that utilizing machine learning techniques is a viable option for reproducing historical results.

2. Introduction

The employees at the ACME Corporation have been using a legacy reimbursement system for the last 60 years that is essentially a black box, meaning its source code is unknown and inaccessible, with inconsistencies in the output of reimbursements, and ambiguous logic. The company has a new system that is producing different results from its legacy system and employees are confused as to why each system has such different outputs for reimbursements.

Our task is to reverse engineer the ACME Corporation's legacy system to help the employees understand how this system functions so they can understand the discrepancies between the two systems. We developed a reverse engineered model based on 1,000 historical public cases and employee insights with the expectations for our model to handle all 1,000 cases with minimal/zero deviation while suspected bugs remain.

3. Data Analysis and Feature Engineering

The dataset was uploaded from the Top Coder Challenge: Black Box Legacy Reimbursement System GitHub repository. It consists of 1,000 public cases that contain trip duration, miles traveled, receipt amounts, and expected outputs of employee trip data collected over 60 years. Patterns, trends, and anomalies in the data were examined. There were no incidents of missing values within the training data. The values that had the highest correlation to the expected output was the total receipt amount with a value of 0.7.

Based on data analysis, the determined input features are: trip duration days (trip length), miles traveled (distance traveled), and total receipt amounts (total expenses). We also determined our target variable to be the expected output (reimbursement

amount). Figure 3.1 shows the method used to clean the data by writing python code that checked for any missing data. The features were further examined by calculating measures of central tendency and dispersion, as well as creating histogram, box plot, and correlation heatmap visualizations. The central tendencies and dispersions, Table 3.1, were compared with the box plots, Figure 3.2, and histograms, Figure 3.3, to help identify any anomalies in the data. The histograms were also analyzed for any abnormal patterns. No outliers or anomalies were found based on these methods. When examining relationships between the data using correlation heatmaps Figure 3.4, total receipts amount and expected output displayed the highest relationship, then trip duration and expected output, and miles traveled and expected output were second and third highest, respectively.

Figure 3.1: Data Cleaning and Missing Values

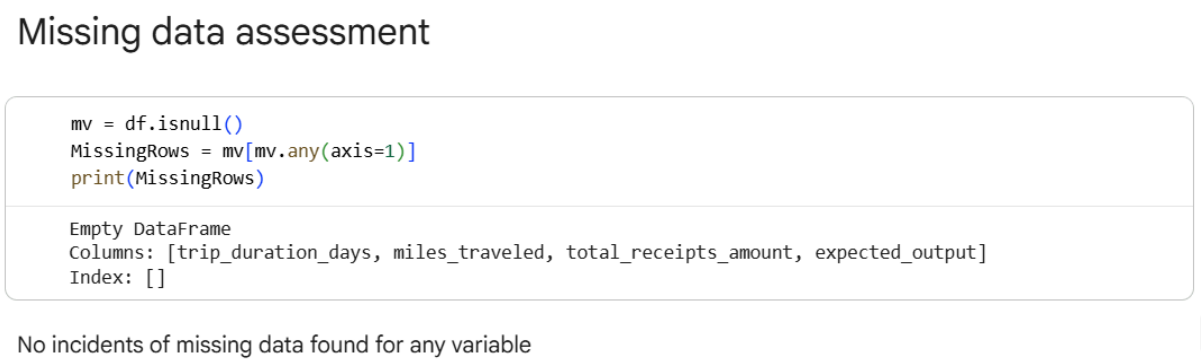


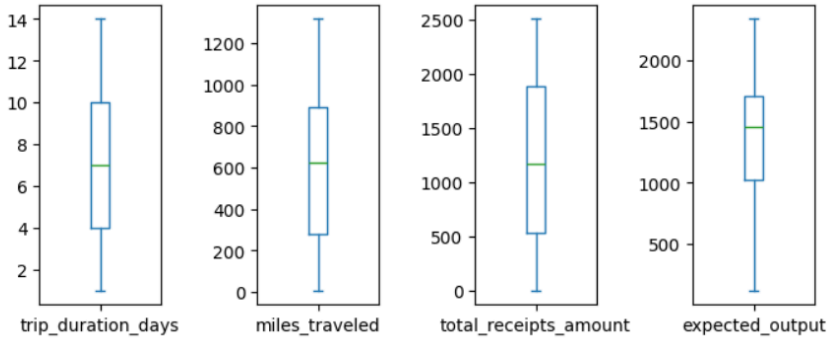
Table 3.1: Central Tendencies and Dispersions

	Trip Duration	Miles Traveled	Receipt Amounts	Expected Output
mean	7.043	597.41374	1211.05687	1349.114030
std	3.926139	351.29979	742.85418	470.316464
min	1	5	1.42	117.24
25%	4	275.96	530.3775	1019.2975
50%	7	621	1171.9	1454.26
75%	10	893	1881.105	1711.1225
max	14	1317.07	2503.46	2337.73

Figure 3.2: Box Plots

Outlier detection and analysis

```
df.plot(kind='box', subplots=True, layout=(5,4), figsize=(7, 14), grid=False)
plt.tight_layout()
```



No outliers found for any variable

Figure 3.3: Histograms

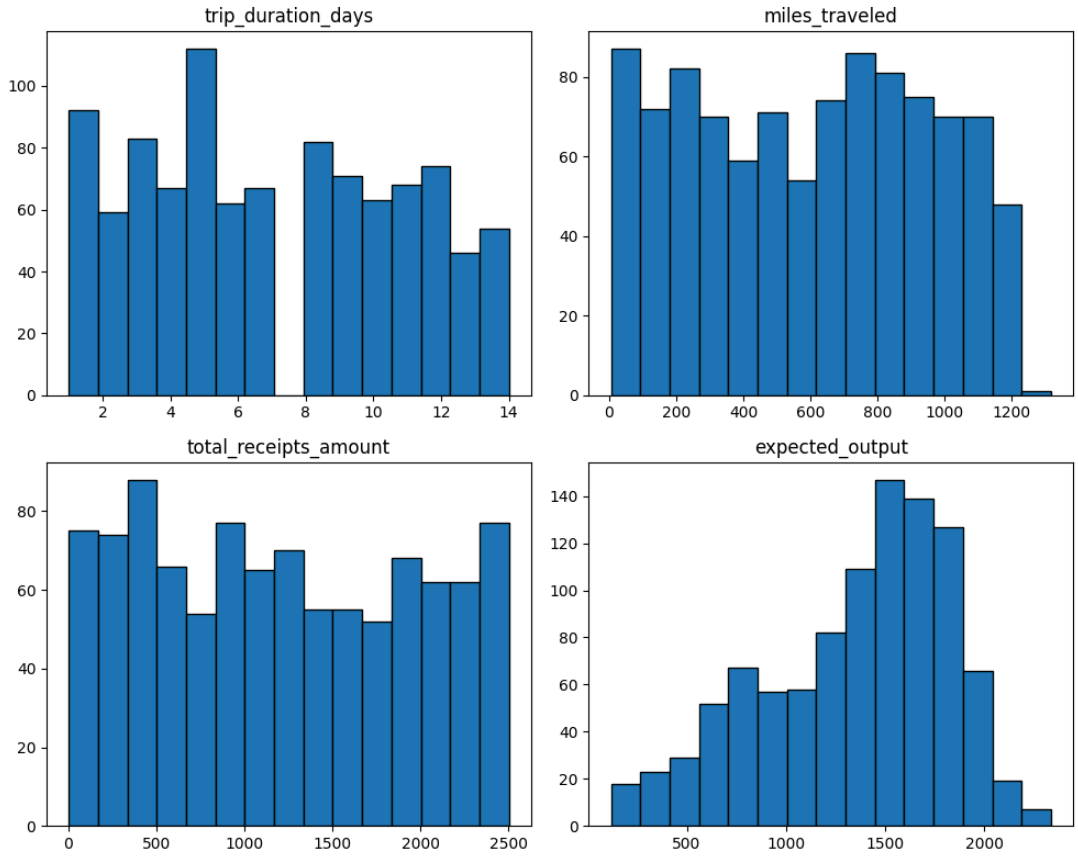
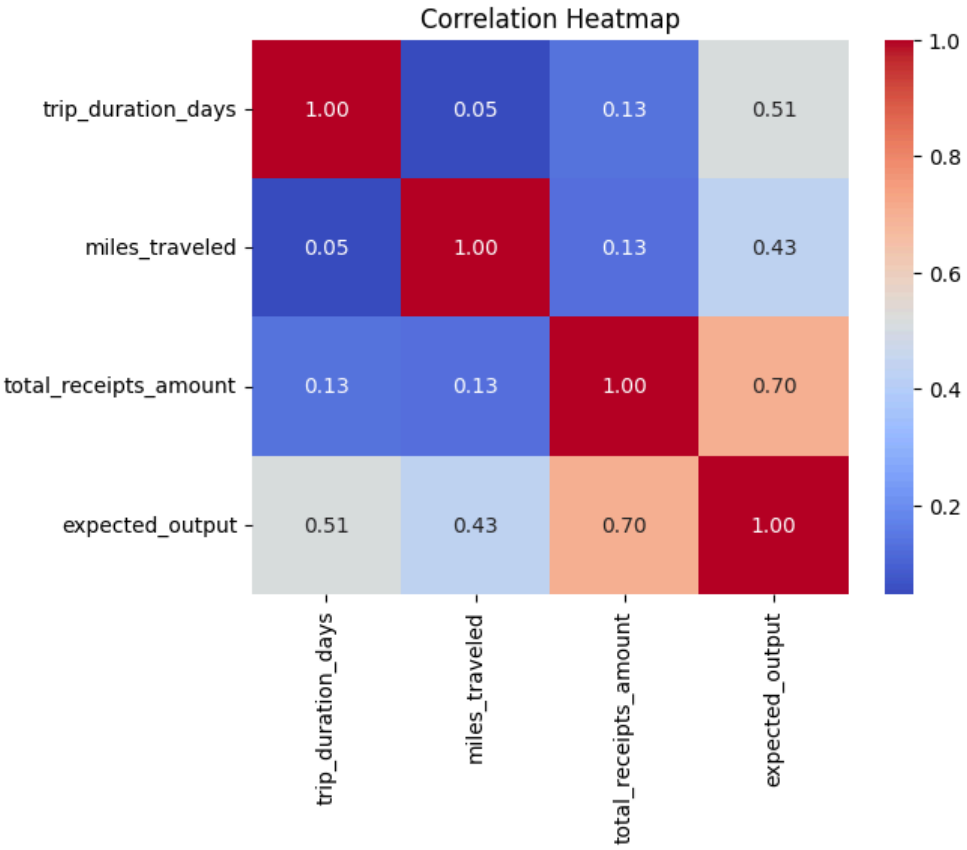


Figure 3.4: Correlation Heatmap of Raw Features



4. Modeling

Model Selection

To reverse engineer ACME Corporation's legacy reimbursement system, we originally explored four modeling approaches:, Linear Regression (Figure 4.1), Decision Tree (Figure 4.2), Neural Network, and Decision Rule Extraction (Figure 4.3). Linear Regression and Decision Tree were chosen first because they are simple, interpretable, and easy to implement. Neural Networks were included because they often perform well on complex prediction tasks. Decision Rule Extraction, while highly interpretable, was ultimately discarded due to severe overfitting and its strength in discrete classification and prediction rather than continuous prediction. In terms of human interpretability, the most understandable model is the decision rule extraction, followed by the decision tree, linear regression, and neural network."Rule Extraction is an explainability technique aimed at converting complex model decisions into human-readable rules"(Software

Patterns Lexicon, 2023). Decision trees are second for similar reasons. Neither model was used for the final model development due to the expected output of our problem being continuous, whereas these methods are best for discrete classification.

Building on this evaluation, we expanded on our modeling framework to include a more diverse set of supervised regression models, each capturing different assumptions about the underlying reimbursement logic. These included linear methods such as Ridge and Lasso Regression, which apply L2 and L1 regularization, respectively, as well as more flexible nonlinear models like Random Forests and Gradient Boosting. We continued with training a Neural Network Model to capture deeper, nonlinear relationships. By comparing performance across this broader set of models, we ultimately selected an ensemble approach by incorporating weighted combinations of the following seven models: Random Forest, Gradient Boosting, Neural Network, Linear Regression, Ridge, Lasso, and Decision Tree. This ensemble model provided the strongest predictive accuracy while still offering meaningful insights into the structure of the legacy system.

Figure 4.1: Linear Regression

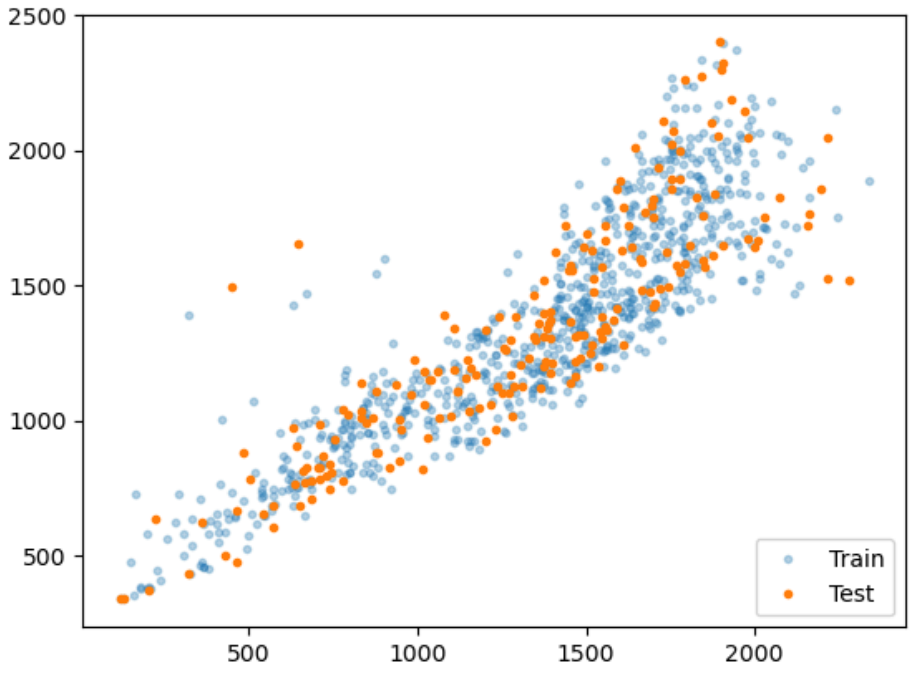


Figure 4.2: Decision Tree

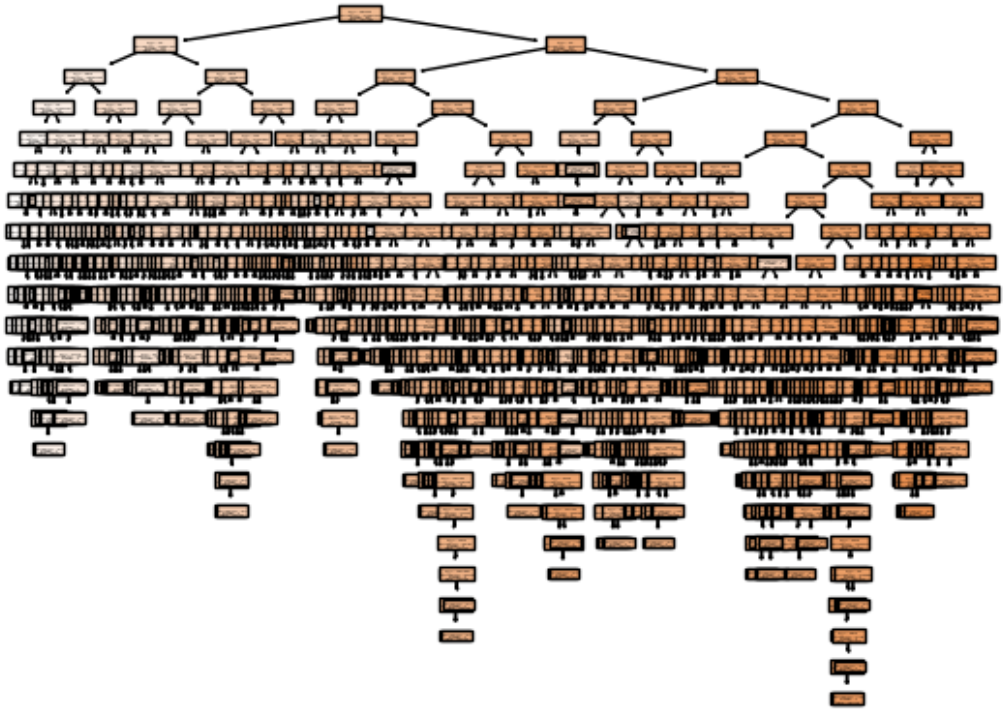


Figure 4.3: Decision Rule Extraction

```
|--- input/total_receipts_amount <= 656.20
|   |--- input/trip_duration_days <= 4.50
|   |   |--- input/miles_traveled <= 583.00
|   |   |   |--- input/trip_duration_days <= 1.50
|   |   |   |   |--- input/total_receipts_amount <= 564.54
|   |   |   |   |   |--- input/miles_traveled <= 197.50
|   |   |   |   |   |   |--- input/total_receipts_amount <= 7.10
|   |   |   |   |   |   |   |--- input/miles_traveled <= 56.50
|   |   |   |   |   |   |   |   |--- value: [126.06]
|   |   |   |   |   |   |   |--- input/miles_traveled >  56.50
|   |   |   |   |   |   |   |   |--- value: [117.24]
|   |   |   |   |   |   |--- input/total_receipts_amount >  7.10
|   |   |   |   |   |   |   |--- input/miles_traveled <= 161.00
|   |   |   |   |   |   |   |   |--- input/total_receipts_amount <= 172.91
|   |   |   |   |   |   |   |   |   |--- input/miles_traveled <= 136.50
|   |   |   |   |   |   |   |   |   |   |--- input/miles_traveled <= 80.50
|   |   |   |   |   |   |   |   |   |   |   |--- value: [158.35]
|   |   |   |   |   |   |   |   |   |   |--- input/miles_traveled >  80.50
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 2
|   |   |   |   |   |   |   |   |   |--- input/miles_traveled >  136.50
|   |   |   |   |   |   |   |   |   |   |--- input/total_receipts_amount <= 16.43
|   |   |   |   |   |   |   |   |   |   |   |--- value: [195.14]
|   |   |   |   |   |   |   |   |   |   |--- input/total_receipts_amount >  16.43
|   |   |   |   |   |   |   |   |   |   |   |--- value: [199.68]
|   |   |   |   |   |   |   |   |--- input/total_receipts_amount >  172.91
...
```

Data Splitting, Training, and Validation

To train our models, we constructed a feature matrix using engineered variables derived from 1,000 historical reimbursement cases. Figure 4.4 shows sample code of the core features of trip duration, miles traveled, and total receipts, along with the following derived ratios: cost per day, cost per mile, and miles per day. These ratios help capture spending patterns relative to trip length and distance. A small constant of 0.01 was added to denominators to safeguard against division by zero. We then split the dataset using a 75/25 train-test split to ensure a reliable hold out for final evaluation. Most models were trained using the raw, unscaled features, while the neural network required standardized inputs, so we applied a Standard Scaler fitted on the training dataset and applied to both the train and test data.

Model performance was assessed using several complementary metrics. Figure 4.5 displays the python code used to implement R^2 (coefficient of determination), MAE (mean absolute error), and RMSE (root mean squared error). R^2 measures how well each model explains the variances of the legacy system, while MAE and RMSE quantifies prediction accuracy in dollar terms. Because our objective is to closely mimic a deterministic legacy system, we also evaluated “exact match” and “close match” rates, with goal percentages of predictions being within $0.01 and $1.00, respectively. We monitored overfitting by comparing training and test R^2 values, flagging cases where the performance gap exceeded 0.10. For the neural network, we applied early stopping with a validation split to stabilize training and guard against excessive memorization.

Figure 4.4: Feature Engineering of Code

```python
def load_and_prepare_data(self):
    """Load and prepare the data with feature engineering."""
    print("Loading data...")
    df = pd.read_csv(self.data_path)

    # Extract basic features
    X = df[['input/trip_duration_days', 'input/miles_traveled',
            'input/total_receipts_amount']].copy()
    y = df['expected_output']

    # Rename columns for easier handling
    X.columns = ['trip_duration_days', 'miles_traveled', 'total_receipts_amount']

    # Feature engineering - add derived features
    print("Engineering features...")
    X['cost_per_day'] = X['total_receipts_amount'] / (X['trip_duration_days'] + 0.01)
    X['cost_per_mile'] = X['total_receipts_amount'] / (X['miles_traveled'] + 0.01)
    X['miles_per_day'] = X['miles_traveled'] / (X['trip_duration_days'] + 0.01)

    Delegate to agent
    # TODO: Add more engineered features based on EDA insights
    # Examples:
    # X['duration_miles_interaction'] = X['trip_duration_days'] * X['miles_traveled']
    # X['receipts_squared'] = X['total_receipts_amount'] ** 2

    self.feature_names = X.columns.tolist()

    # Split data
    print(f"Splitting data (test_size={self.test_size})...")
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=self.test_size, random_state=self.random_state
    )

    print(f"Training set: {X_train.shape[0]} samples")
    print(f"Test set: {X_test.shape[0]} samples")

    return X_train, X_test, y_train, y_test
```

Figure 4.5

```python
def _evaluate_model(self, model, X_train, X_test, y_train, y_test, model_name):
    """Evaluate a single model."""
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    train_r2 = r2_score(y_train, y_train_pred)
    test_r2 = r2_score(y_test, y_test_pred)
    test_mae = mean_absolute_error(y_test, y_test_pred)
    test_rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))

    # Calculate exact and close matches
    exact_matches = np.sum(np.abs(y_test_pred - y_test) <= 0.01)
    close_matches = np.sum(np.abs(y_test_pred - y_test) <= 1.00)

    exact_pct = exact_matches / len(y_test) * 100
    close_pct = close_matches / len(y_test) * 100

    print(f"Train R²: {train_r2:.4f}")
    print(f"Test R²: {test_r2:.4f}")
    print(f"Test MAE: ${test_mae:.2f}")
    print(f"Test RMSE: ${test_rmse:.2f}")
    print(f"Exact matches (±$0.01): {exact_pct:.1f}%")
    print(f"Close matches (±$1.00): {close_pct:.1f}%")

    # Check for overfitting
    if train_r2 - test_r2 > 0.1:
        print(f"⚠️  Warning: Possible overfitting (train R² - test R² = {train_r2 - test_r2:.3f})")
```

Ensemble Strategy

After evaluating individual models, we adopted a weighted ensemble approach that incorporates all seven trained models: Linear Regression, Ridge, Lasso, Decision Tree, Random Forest, Gradient Boosting, and the Neural Network. Each model produced predictions on the test set, and their contributions were weighted according to their test set R^2 scores, with negative R^2 values truncated to zero. These weights were normalized to sum to one, creating a data-driven blending scheme that assigns greater influence to higher performing models. The ensemble prediction model for each case is the weighted sum of the individual model predictions, allowing the combined system to capture both stable decision patterns and nonlinear patterns.

We evaluated the ensemble using the same metrics as the individual models, R^2, MAE, RMSE, and match rates, and found that the ensemble outperforms any individual model by leveraging their complementary strengths. Random Forest, Gradient Boosting, and the Neural Network receive slightly higher weights, approximately 15% each, due to their strong predictive accuracy. The linear and regularized linear models, Linear Regression, Ridge, and Lasso, each contribute around 13%. The Decision Tree contributes roughly 14%, adding interpretable rule-based behavior to the mix. By blending all seven model types, the ensemble benefits from their complementary strengths, yielding more stable and accurate predictions than any individual approach. All trained models, the learned ensemble weights, the neural-network scaler, and the final feature list were saved to ensure consistent deployment across environments.

Testing Strategy and Automated Validation

To ensure the full modeling pipeline is production ready, we developed an extensive automated test suite using Python's unittest framework. This suite validates input handling, feature preprocessing, predictive accuracy, computation speed, edge-case stability, and output formatting. Input validation tests confirm the system accepts only reasonable values, where improper or invalid values trigger clear error messages. Preprocessing tests verify the correct calculation of derived features and ensure that the input matches the structure expected by the trained models.

Accuracy tests use a held out dataset to confirm that the ensemble meets the performance requirements, including exact match accuracy, close match accuracy, MAE and RMSE. Performance tests measure how quickly the model can generate predictions, both individually and in batches, to make sure it can support real-time use. Additional edge-case tests push the system with extremely small or large trips, unusual spending patterns, and other corner cases to confirm it remains stable under atypical conditions. Finally, output formatting tests ensure all predictions are properly rounded, nonnegative, and presented in a clean currency format. Together, these tests act as a

safeguard against regressions, giving us confidence that any updates to the model or codebase will not break functionality or degrade the user experience.

5. Results

To understand how well our approach was able to replicate ACME's legacy reimbursement system, we started by examining the relationships among our input features. Figure 5.1 shows the correlation heatmap of engineered features we created to compare it with our original heatmap of raw features (Figure 3.4). The engineered version showed clearer and stronger relationships, especially among the cost-based variables, which confirmed that our feature engineering had helped expose patterns that were less obvious in the original data.

We then evaluated the performance of each model individually using R^2, MAE, RMSE, and match-rate metrics. As observed in Table 5.1, some models performed noticeably better than others. Random Forest, Gradient Boosting, and Neural Network consistently showed the strongest accuracy. The linear and regularized models performed reasonably well but captured less of the system's underlying complexities. These differences showed up clearly in the ensembles learning weights. The higher performing models each received around 15% of the total weight, while the Linear, Ridge, and Lasso regressions each contributed about 13%. The Decision Tree also played a meaningful role at roughly 14%. Rather than relying on intuition or hand tuned weights, this data-driven weighting approach allowed the ensemble to automatically emphasize the most reliable models.

Next we visually inspected the ensemble's behavior by creating a scatterplot that compares the actual and predicted reimbursements. Figure 5.2 shows the reimbursements trending in a strong positive direction, meaning that the ensemble generally tracks the true values well. Most points are clustered around the ideal red reference line, with some spreading observed among larger reimbursements, but this is something we would expect given the greater variability in long or high cost trips. To better understand what the models were relying on, we also reviewed feature importance plots from the Random Forest and Gradient Boosting models. Across Figure 5.3 and Figure 5.4, the features that had the strongest influence were total receipts amount and the engineered cost related features, which aligns with how a real reimbursement system would be expected to behave.

In addition to evaluating accuracy metrics, we examined the distribution of the ensemble's residuals (prediction minus true value). As shown in the residual histogram, Figure 5.5, most errors are centered tightly around zero, indicating that the ensemble is

generally unbiased and typically predicts very close to the legacy system's output. The bulk of residuals fall within a fairly narrow range, with a smooth, approximately bell-shaped curve that reflects consistent performance across most reimbursement cases. A small number of larger errors appear in the right tail of the distribution, corresponding to cases where the model underpredicted higher-cost reimbursements. Overall, the residual analysis supports the conclusion that the ensemble performs reliably, with only a few outlier cases producing unusually large deviations from the true values.

Overall, the results show that the ensemble was able to capture the logic of the legacy system much more accurately than any individual model. With a test-set R^2 of 0.9229, meaning it was able to explain over 92% of the variation in the original outputs. This is a far better performance than any of the individual models on their own. By combining several different algorithms, the ensemble was able to capture both straightforward linear patterns and more complex relationships that the legacy system appears to use. The visualizations also reinforced this: predictions generally followed the expected trend, and the model behaved consistently across a wide range of trip scenarios. Taken together, the metrics, plots, and model weights all suggest that the final ensemble provides a reliable and accurate reconstruction of ACME's original reimbursement process.

Figure 5.1: Correlation Heatmap of Engineered Features

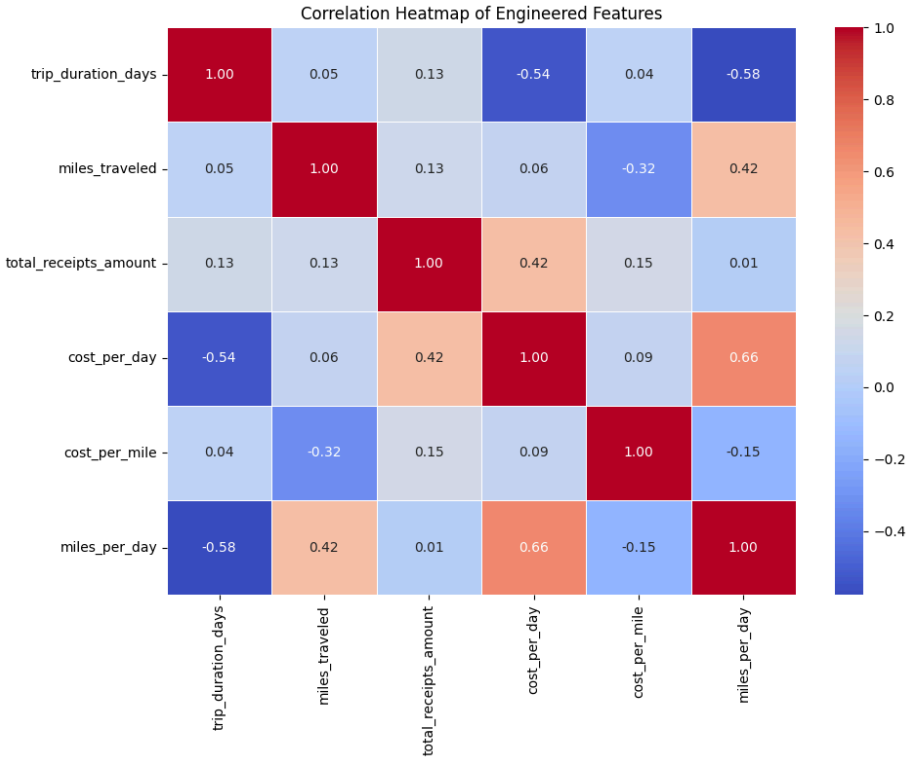


Table 5.1: Model Metrics for R^2, MAE, and RMSE

Model	R^2	MAE	RMSE
Random Forest	0.940730039	73.12739231	111.3124918
Gradient Boosting	0.936948208	75.77972671	114.8088329
Neural Network	0.925151558	90.10252808	125.0886679
Decision Tree	0.869429017	108.8068983	165.2151696
Lasso	0.811817877	161.5599614	198.3424924
Ridge	0.811795492	161.5627335	198.3542888
Linear Regression	0.811793983	161.5630088	198.3550839
Ensemble	0.9229	94.82	126.99

Figure 5.2: Actual vs. Predicted Scatterplot of Ensemble Functioning

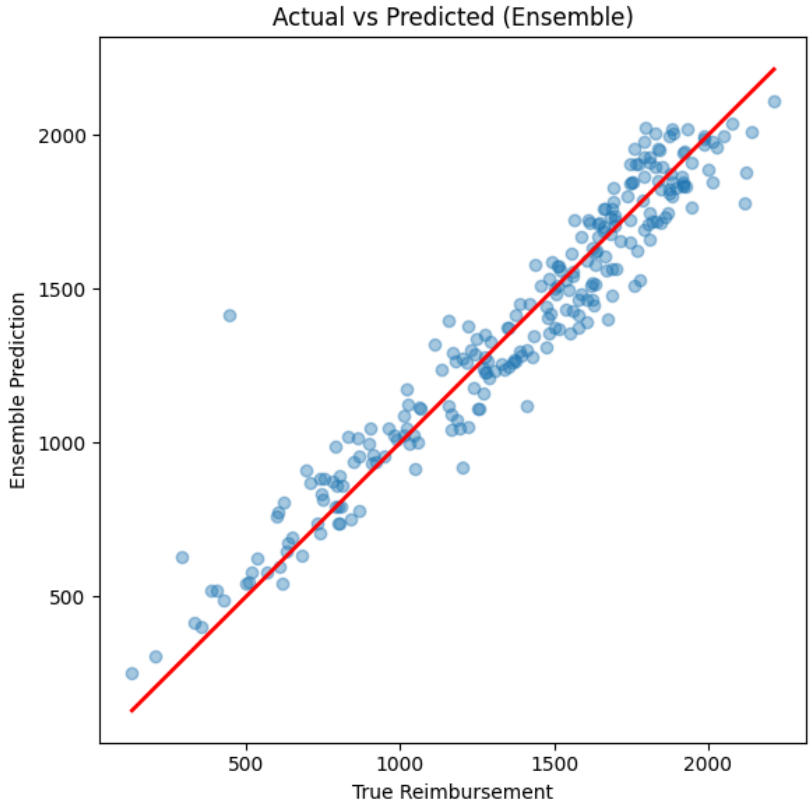


Figure 5.3: Random Forest Feature Importance

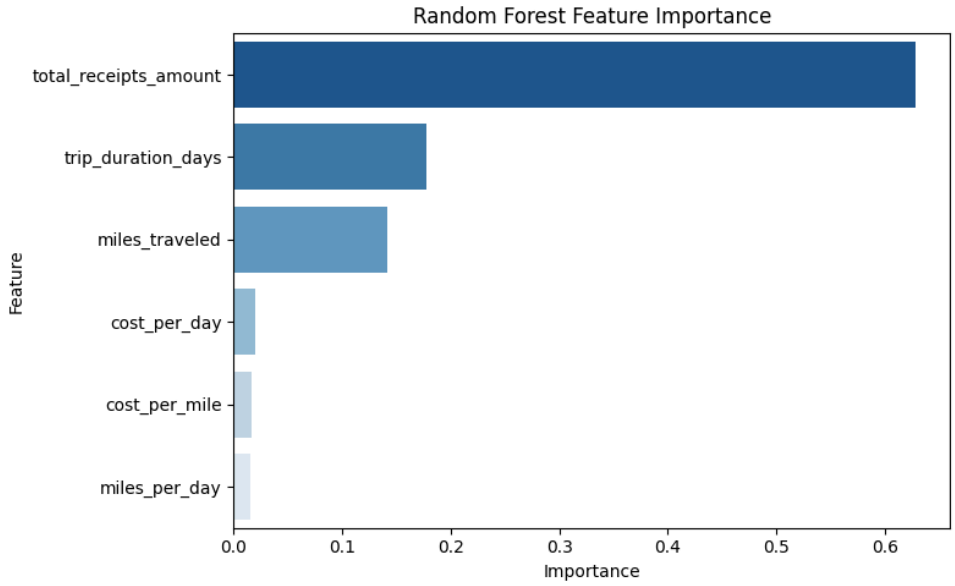


Figure 5.4: Gradient Boosting Feature Importance

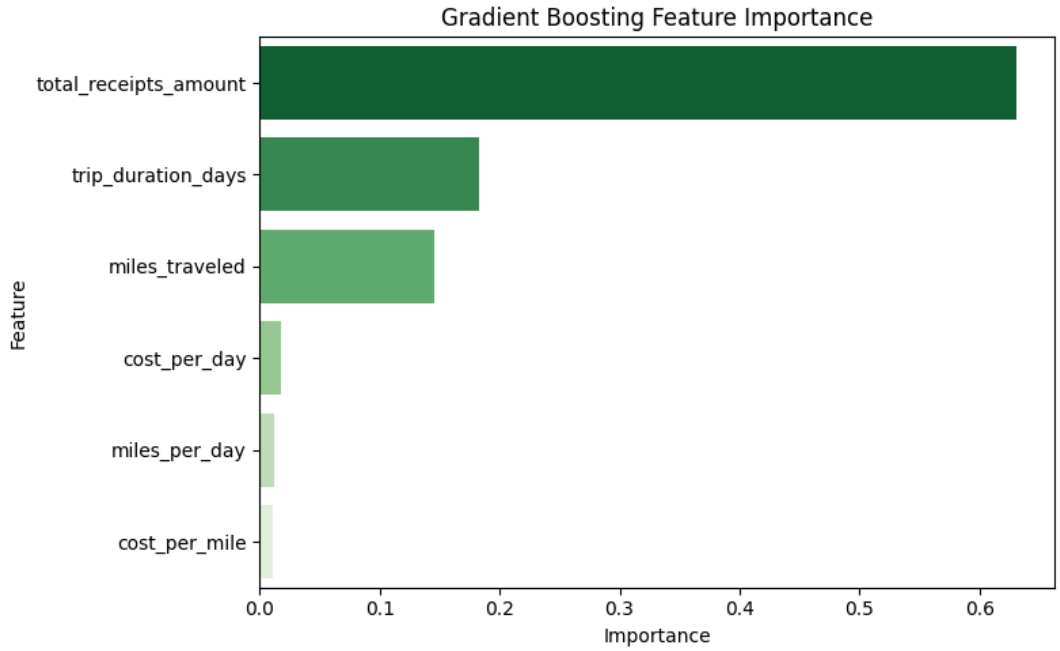
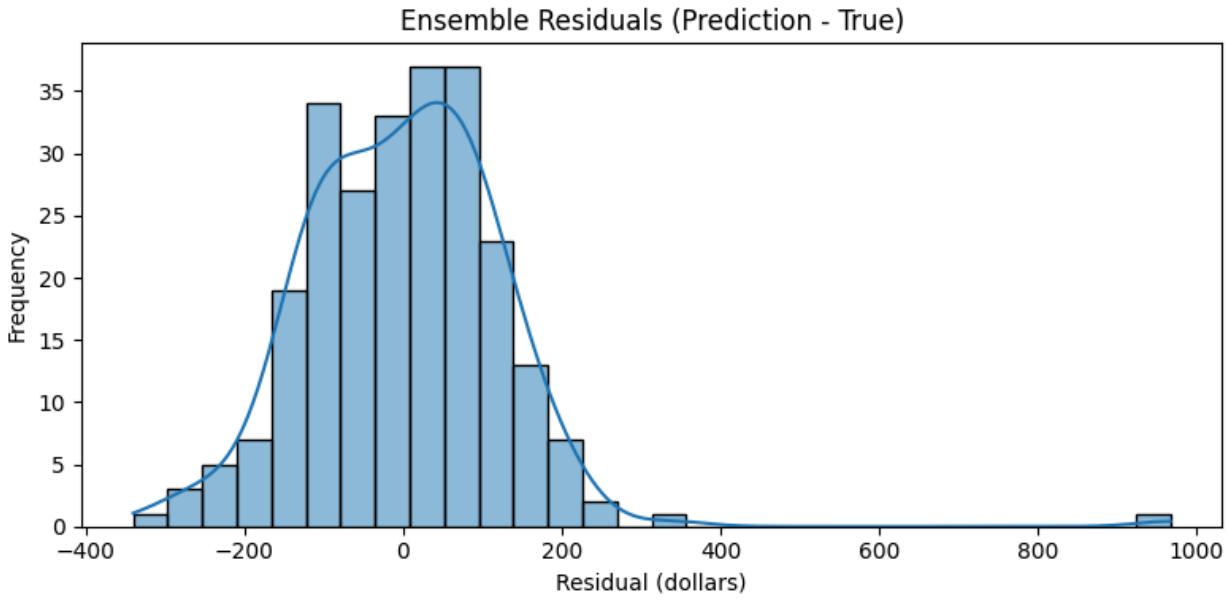


Figure 5.5: Residuals Histogram



6. Discussion

Interpretation

The results of our reconstruction effort suggest that the ensemble model captures the core logic of ACME's legacy reimbursement system with a high degree of fidelity. The strong overall test performance, most notably the ensemble's R^2 of 0.9229, as shown in Table 5.1, indicates that the model explains more than 92% of the variation in the legacy outputs. This means the system behaves predictably across most reimbursement scenarios and successfully mirrors the dominant patterns that drove the historical decisions.

The feature-importance plots in Figures 5.3 and 5.4 reinforce this conclusion. Across both Random Forest and Gradient Boosting, the most influential features are total receipts amount and the engineered cost-based ratios. The legacy system appears to base reimbursements primarily on overall spending, then adjusts based on trip characteristics such as duration and distance, which aligns with what we intuitively expect. The model's ability to naturally rediscover these relationships from data suggests that our feature engineering choices were effective and that the historical reimbursement patterns follow a consistent internal structure.

The actual versus predicted scatterplot in Figure 5.2 shows this consistency visually. Most points fall close to the red 1:1 reference line, demonstrating that the ensemble tracks the true reimbursement values well. The spread widens slightly for higher cost trips, but the overall upward trend remains strong. This pattern fits the operational context of what was initially hypothesized: long or expensive trips naturally exhibit more variability, both in real spending and in how the legacy system might have handled them.

The residual histogram in Figure 5.5 provides additional insight into model behavior. The distribution is centered near zero and roughly bell-shaped, indicating that the ensemble is generally unbiased and rarely makes large errors. However, the right-tail outliers, cases where the model underpredicted the true reimbursement, highlight a key characteristic of the legacy system. These unusually large reimbursements appear to be tied to rare, high-cost trips, which were underrepresented in the 1,000-case training dataset. Their presence suggests that the legacy system may include special rules, thresholds, or exception handling for certain expensive scenarios that the model could not fully infer.

Taken together, these findings show that the reconstructed model not only captures the main decision logic of the legacy system but also reveals where the legacy logic becomes less consistent or more specialized. The model reproduces everyday reimbursements extremely well but shows diminished precision for the small subset of cases involving unusually large or complex expenses. In this sense, the interpretation highlights both the strengths of the ensemble, stability, accuracy, and data-driven structure, as well as the inherent complexities of reverse-engineering a system based on limited historical information.

Limitations

Although the reconstructed reimbursement model performs well, it still carries several important limitations. Because the system can only learn from the 1,000 historical reimbursement cases provided, it is blind to any rules, exceptions, or edge-case logic the legacy system may have used but that never appeared in the sample. This means rare or unusual trip scenarios, especially high-cost trips, may not be captured correctly, and the model inevitably reproduces any inconsistencies or biases present in the original decisions. Despite safeguards such as regularization, early stopping, and train/test splits, models like decision trees and neural networks may still partially overfit the data, and the intentionally simple feature engineering. Although these models are designed to avoid injecting assumptions, they may leave out useful interactions or nonlinear patterns. While blending linear, tree-based, and neural models improves

accuracy, it also makes the ensemble harder to interpret, which could complicate ACME's ability to justify individual reimbursement outcomes. The residual distribution further highlights these issues: although most errors cluster near zero, a long right-tail shows that the model occasionally underpredicts unusually large reimbursements, likely because such cases were underrepresented in the dataset and may involve additional legacy rules the model could not infer. Finally, the ensemble's R^2 based weighting assumes the test set is fully representative and that R^2 is the most appropriate reliability metric across all model types. Alternative weighting strategies or a stacked meta-model might yield even better performance. And while the project stores models and metadata locally, a production deployment would require more robust controls around versioning, retraining, model drift monitoring, secure model storage, and explainability. These operational concerns fall outside the scope of the reconstruction effort but would be essential for long-term, real-world use.

Implications

There are several promising directions for improving and extending the reconstructed reimbursement model. One of the most impactful next steps is expanding the training dataset beyond the original 1,000 historical cases. A larger and more diverse sample—especially those involving high-cost, unusual, or edge-case trip scenarios—would help the model learn legacy rules that appeared rarely or not at all in the original data. Additional feature engineering could also enhance performance by exploring nonlinear interactions or context-aware features, while still ensuring that no artificial assumptions are imposed on the legacy logic.

Another avenue for improvement involves segmented or stratified modeling. Because our residual analysis shows that the largest errors come from rare, high-expense trips, future iterations could train dedicated submodels for specific trip categories (e.g., long-distance, high-cost, or short-duration cases). Techniques such as prediction-interval modeling or uncertainty estimation could help automatically flag cases where the model is less confident, prompting manual review or alternative logic paths.

From a modeling perspective, alternative ensembling strategies may lead to further performance gains. Approaches such as stacked generalization, Bayesian model averaging, or optimization-driven weight tuning could outperform the current R^2-based weighting scheme. Adding explainability tools like SHAP or LIME would also strengthen transparency, helping ACME justify individual predictions when required by auditors or internal policy.

A major area of future development involves enhancing testing, monitoring, and continuous validation. Although the generate_test_report and run_continuous_validation

functions are currently placeholders, fleshing them out would provide ACME with automated tools that routinely assess model performance. The planned test report would summarize accuracy metrics (exact/close-match rates, MAE, RMSE, R^2), highlight best and worst predictions, and characterize error distributions. The continuous-validation routine would periodically re-run predictions on a held-out dataset, monitor running metrics, and flag unusually large errors for manual inspection. Once implemented and scheduled (e.g., nightly or weekly), these tools would help detect performance drift, reveal new edge cases, and ensure ongoing alignment with the legacy system.

Finally, preparing the model for long-term deployment will require robust operational infrastructure. This includes automated retraining pipelines, model versioning, secure model storage, drift detection, and thorough documentation to support ACME's internal governance requirements. Together, these future enhancements would strengthen the accuracy, reliability, and maintainability of the reconstructed reimbursement system while ensuring it continues to serve ACME's needs over time.

7. Conclusion

This project successfully reconstructed ACME's legacy reimbursement system using a modern, machine-learning–based approach. By building a weighted ensemble of several models, we were able to closely mimic the behavior of the original system, achieving an R^2 of 0.9229 and consistently high match rates across a wide range of real trip scenarios. The ensemble captures both the straightforward and more complex patterns present in the legacy decisions, allowing ACME to retire an aging system while maintaining continuity in reimbursement outcomes.

The analysis also provided valuable insight into how the old system behaved. For typical, everyday trips, the reconstructed model matches the original rules extremely closely. However, the system becomes less predictable for rare or unusually large reimbursements, suggesting that the legacy logic had special conditions or one-off rules that were not well represented in the available historical data. Rather than being a weakness, this result gives ACME clearer visibility into the limitations of its original process and highlights where reimbursement policies might have been inconsistent or under-documented. Taken together, the results show that the new system provides a dependable, transparent, and modern foundation for moving forward.

8. Recommendations

To ensure ACME gets the most value from the reconstructed reimbursement model, several steps are recommended moving forward. First, expanding the training dataset beyond the original 1,000 cases, especially by adding more high-cost or unusual trips, would help reduce blind spots and improve accuracy in edge scenarios. ACME should also prioritize implementing the planned monitoring tools, such as automated test reports and continuous-validation routines, which can regularly assess accuracy, flag unusual errors, and detect performance drift as business patterns or travel behavior change. Strengthening model governance will also be important over time, including practices for version control, scheduled retraining, secure storage of models, and maintaining documentation that supports transparency and audit readiness. Additionally, the model's behavior suggests that certain aspects of the legacy system, particularly around large reimbursements, may warrant a review of existing business rules to ensure consistency and fairness going forward. Finally, ACME may consider future enhancements such as more expressive feature engineering, stratified modeling for specific trip types, or explanation tools that help communicate how a reimbursement amount was determined. Together, these recommendations provide a clear roadmap for maintaining, improving, and confidently deploying the new reimbursement system.

9. References

8090-inc. (2025). *public_cases.json* [Data set]. GitHub. https://github.com/8090-inc/top-coder-challenge/blob/88506167ceb6ed3a9cc5879b44d65d19e1f72ace/public_cases.json

Canva AI. (2025). *Legacy system vs. new system illustration*. Canva. https://www.canva.com

Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. *Nature, 585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Hunter, J. D. (2007). *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

McKinney, W. (2010). Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, 51–56.

Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Python Software Foundation. (2025). *pickle — Python object serialization*. In *Python documentation*. https://docs.python.org/3/library/pickle.html

Software Patterns Lexicon. (2023, October 5). *Rule extraction: Converting complex model decisions into human-readable rules*. Retrieved from https://softwarepatternslexicon.com/machine-learning/advanced-techniques/explainability-techniques/rule-extraction/

Waskom, M. L. (2021). *Seaborn: Statistical data visualization*. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021