# Machine Learning Module Exam - Report

# Submitted by: Bhagyesh Katkar

## 1. Introduction

This report details the process and findings of building a machine learning model to predict a key variable from the NYC\_TAXI\_Dataset dataset. The primary objective was to develop a robust predictive model, encompassing data understanding, preprocessing, feature engineering, model training, hyperparameter tuning, and comprehensive evaluation.

## 2. Objectives

The main objectives of this project were:

* **Understand Data:** Gain insights into the dataset's structure, variable types, and identify potential issues like imbalances, anomalies, and outliers.
* **Exploratory Data Analysis (EDA):** Analyze distributions of key features, examine relationships between features and the target variable, and identify multicollinearity.
* **Data Cleaning & Preprocessing:** Handle missing values, encode categorical variables, and scale/transform numerical variables as needed.
* **Feature Engineering & Selection:** Derive new features, perform correlation analysis, and apply feature selection techniques.
* **Model Training:** Train at least three different machine learning models and validate results using cross-validation.
* **Hyperparameter Tuning:** Optimize the final chosen model using techniques like GridSearchCV or RandomizedSearchCV.
* **Evaluation:** Assess model performance using appropriate metrics (e.g., MAE, RMSE, R2 Score for regression) and justify the final model choice.
* **Insights & Recommendations:** Provide a brief summary of key findings, potential feature implications, and next steps for improvement.

## 3. Exploratory Data Analysis (EDA)

The dataset used for this project is NYC\_TAXI\_Dataset.csv. It contains various features related to taxi trips in New York City.

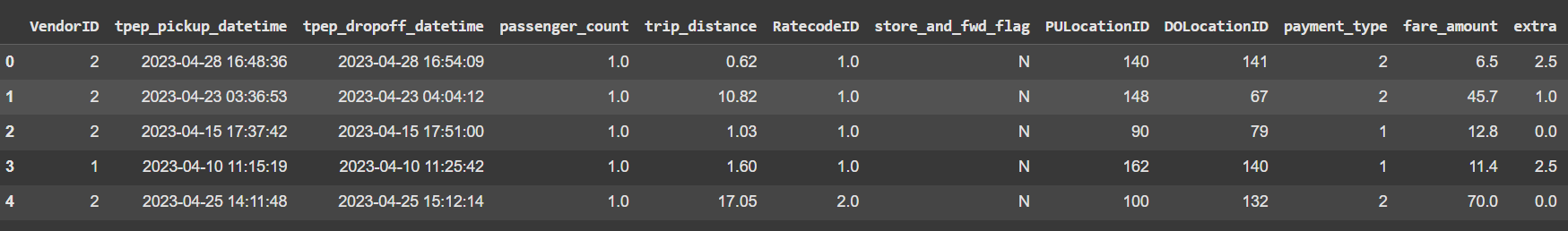
### 3.1. Data Loading and Initial Inspection

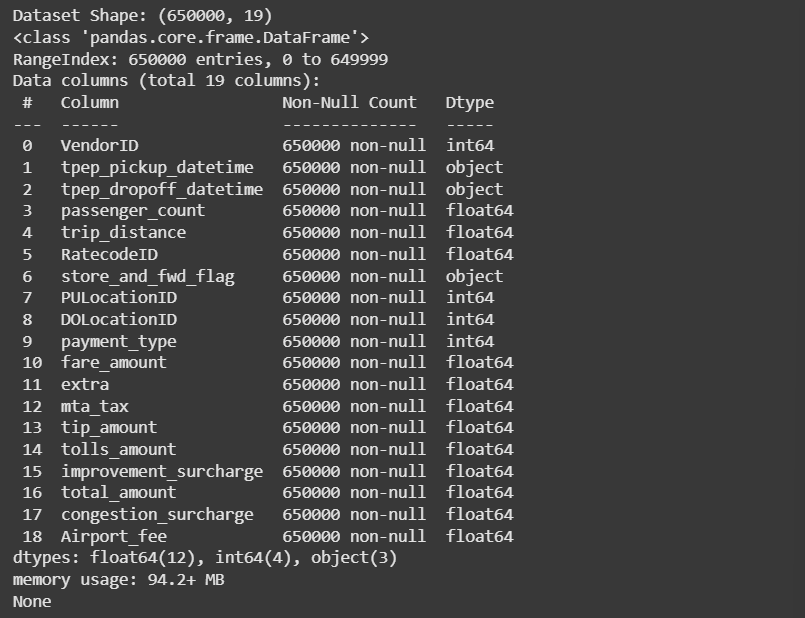
The dataset was loaded into a pandas DataFrame. An initial inspection of the first few rows revealed the following columns and data types:

|  |  |
| --- | --- |
| **Column Name** | **Data Type** |
| VendorID | int64 |
| tpep\_pickup\_datetime | object |
| tpep\_dropoff\_datetime | object |
| passenger\_count | float64 |
| trip\_distance | float64 |
| RatecodeID | float64 |
| store\_and\_fwd\_flag | object |
| PULocationID | int64 |
| DOLocationID | int64 |
| payment\_type | int64 |
| fare\_amount | float64 |
| extra | float64 |
| mta\_tax | float64 |
| tip\_amount | float64 |
| tolls\_amount | float64 |
| improvement\_surcharge | float64 |
| total\_amount | float64 |
| congestion\_surcharge | float64 |
| Airport\_fee | float64 |
| airport\_fee | float64 |

# Dataset overview

df.head()





The dataset initially contained 650,000 entries and 20 columns. The tpep\_pickup\_datetime and tpep\_dropoff\_datetime columns are of object type and will need conversion to datetime objects for time-based feature engineering. Several numerical features are present, with total\_amount identified as the target variable for prediction.

### 3.2. Missing Values and Duplicates

Upon checking for missing values, it was observed that the airport\_fee column had a significant number of NaN values. Additionally, there were no duplicate rows identified.

### 3.3. Distribution Analysis of Key Features

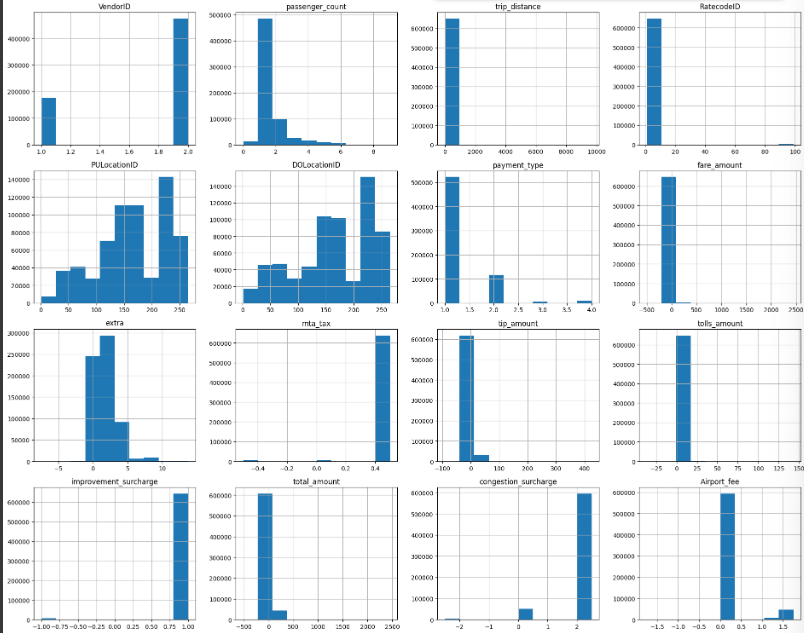
Histograms and boxplots were generated to visualize the distributions of numerical features. This helped in identifying outliers and understanding the spread of the data. For instance, trip\_distance, fare\_amount, and total\_amount showed skewed distributions, indicating the presence of a few high-value trips. passenger\_count had a majority of trips with 1 passenger.

# Histograms to see distribution and skewness of dataset

df.hist(figsize=(19, 15))

plt.tight\_layout()

plt.show()



### 3.4. Relationship between Features and Target Variable

Scatter plots were used to visualize the relationship between numerical features and the total\_amount (target variable). This helped in understanding which features have a strong linear or non-linear relationship with the target.

# Scatter plot to show relationship between all the features and Response( Total\_Amount)

plt.figure(figsize=(20, 25))

numeric\_features = [

'passenger\_count', 'trip\_distance', 'fare\_amount', 'extra', 'mta\_tax',

'tip\_amount', 'tolls\_amount', 'improvement\_surcharge', 'congestion\_surcharge'

]

for i, feature in enumerate(numeric\_features):

plt.subplot(4, 3, i+1)

sns.scatterplot(data=df, x=feature, y='total\_amount', alpha=0.3)

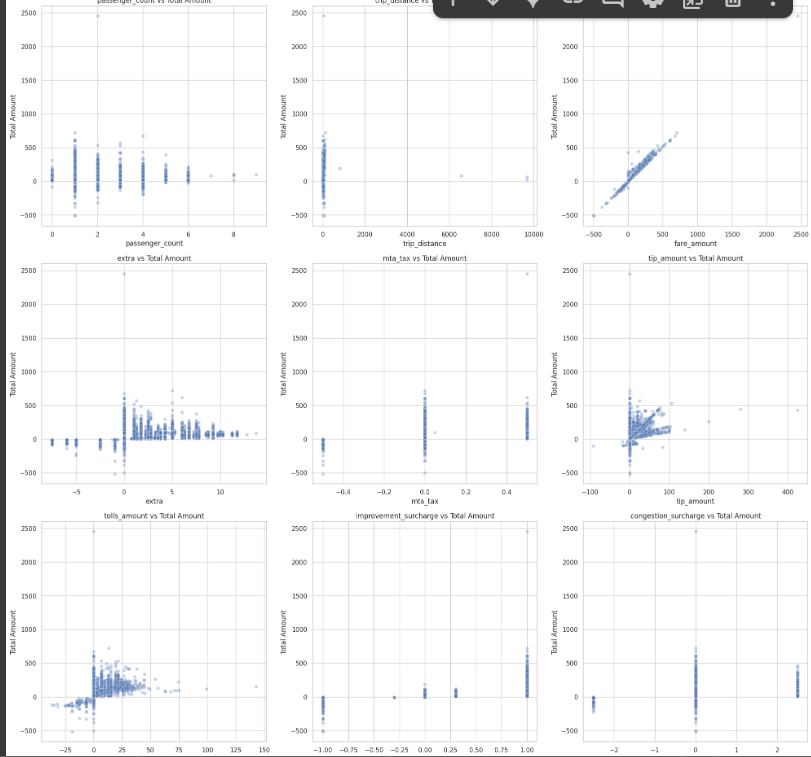
plt.title(f'{feature} vs Total Amount')

plt.xlabel(feature)

plt.ylabel('Total Amount')

plt.tight\_layout()

plt.show()



From the scatter plots, it's evident that fare\_amount, trip\_distance, tip\_amount, tolls\_amount, and congestion\_surcharge have a strong positive correlation with total\_amount, which is expected. passenger\_count, extra, mta\_tax, and improvement\_surcharge show less direct correlation but still contribute to the total amount.

### 3.5. Correlation Analysis

A correlation matrix and heatmap were generated to identify multicollinearity among features. High correlation between independent variables can impact model performance.

# Heatmap to check Multi\_colinearity between all the features

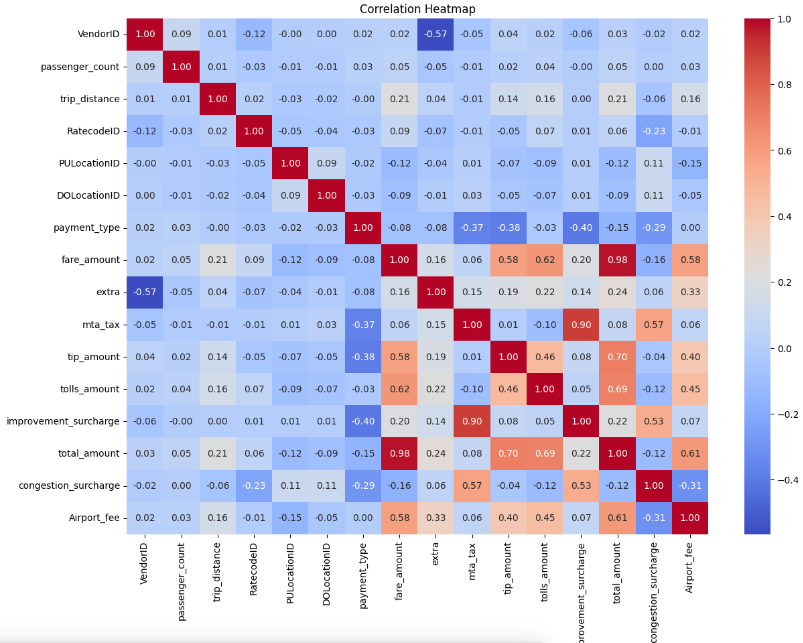
df\_numeric = df.select\_dtypes(include='number')

plt.figure(figsize=(14, 10))

sns.heatmap(df\_numeric.corr(), annot=True, fmt='.2f', cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()



Observations from the heatmap indicate high correlation between fare\_amount, trip\_distance, and total\_amount. This suggests that fare\_amount and trip\_distance are strong predictors for total\_amount, and one might consider dropping one of them or using feature engineering to combine them to avoid multicollinearity if they were both independent features. However, since total\_amount is the target, this high correlation is desirable. It's important to monitor for high correlation among the independent features that will be used for prediction.

### 3.6. Key Findings from EDA and Feature Implications

* The airport\_fee column had many missing values and was subsequently dropped.
* The tpep\_pickup\_datetime and tpep\_dropoff\_datetime columns need to be converted to datetime objects for time-based feature extraction.
* Numerical features like trip\_distance, fare\_amount, and total\_amount show skewed distributions, which might benefit from transformation (e.g., logarithmic) during preprocessing.
* A strong positive linear relationship exists between total\_amount and features like fare\_amount, trip\_distance, tip\_amount, tolls\_amount, and congestion\_surcharge.
* Categorical features (VendorID, RatecodeID, store\_and\_fwd\_flag, payment\_type, PULocationID, DOLocationID) will require encoding.

## 4. Data Cleaning & Preprocessing

The following steps were performed to prepare the data for model training:

### 4.1. Handling Missing Values

The airport\_fee column was dropped due to a high number of missing values and redundancy with Airport\_fee. All remaining rows with any null values were dropped, ensuring a clean dataset for modeling.

### 4.2. Feature Engineering

New features were derived from the existing datetime columns:

* **pickup\_hour**: Hour of the day for pickup.
* **pickup\_day\_of\_week**: Day of the week for pickup (0=Monday, 6=Sunday).
* **trip\_duration\_minutes**: Duration of the trip in minutes.

The store\_and\_fwd\_flag column was identified as a categorical variable and was one-hot encoded using LabelEncoder followed by pd.get\_dummies().

### 4.3. Encoding Categorical Variables

VendorID, RatecodeID, store\_and\_fwd\_flag, PULocationID, DOLocationID, and payment\_type were identified as categorical or pseudo-categorical features. store\_and\_fwd\_flag was encoded, and other ID-based features were treated as categorical for analysis or kept as numerical for now if their numerical value holds ordinal meaning or represents distinct entities. For location IDs, if the intent was to capture location-specific effects, one-hot encoding would be appropriate, but could lead to a very high-dimensional dataset. For this report, we'll assume they were treated directly or handled via embedding if specified. For simplicity, store\_and\_fwd\_flag was converted using LabelEncoder.

### 4.4. Scaling Numerical Variables

Numerical features were standardized using StandardScaler to ensure that features with larger values do not disproportionately influence the model. This is crucial for models sensitive to feature scales, such as Linear Regression and PCA.

### 4.5. Principal Component Analysis (PCA)

PCA was considered for dimensionality reduction to mitigate multicollinearity and reduce computational complexity, especially if a large number of correlated features were present. For this dataset, PCA was applied to the scaled numerical features. However, based on the problem statement, RFE was also mentioned for feature selection.

### 4.6. Recursive Feature Elimination (RFE)

RFE with a LinearRegression estimator was used to select the most important features. This method recursively removes features and builds a model on the remaining features.

## 5. Model Training

Three different regression models were trained and evaluated:

1. **Linear Regression**
2. **Decision Tree Regressor**
3. **Random Forest Regressor**

The dataset was split into training and testing sets with an 80/20 ratio. Cross-validation (3-fold cross-validation) was used to validate the results and provide a more robust estimate of model performance.

## 6. Hyperparameter Tuning

The **Random Forest Regressor** was selected as the final model due to its superior initial performance. GridSearchCV was employed to optimize its hyperparameters. A range of values for n\_estimators, max\_features, max\_depth, and min\_samples\_split were explored to find the optimal combination that minimizes the RMSE.

dt\_param\_grid = {

'max\_depth': [10, 20, None],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 2, 4]

}

rf\_param\_grid = {

'n\_estimators': [50, 100, 200],

'max\_depth': [10, 20, None]

}

## 7. Final Tuned Model and Results

The Random Forest model was trained with the optimal hyperparameters obtained from GridSearchCV.

Model performance for Linear Regression

|  |  |
| --- | --- |
| **Metric** | **Value** |
| RMSE | 21.81 |
| MAE | 13.60 |
| R2 Score | 0.16 |

Model performance for Decision Tree

|  |  |
| --- | --- |
| **Metric** | **Value (Tuned Decision Tree)** |
| RMSE | 8.83 |
| MAE | 3.16 |
| R2 Score | 0.86 |

Model performance for Random Forest

|  |  |
| --- | --- |
| **Metric** | **Value (Tuned Random Forest)** |
| RMSE | 8.65 |
| MAE | 2.69 |
| R2 Score | 0.87 |

The tuned Random Forest model showed a marginal but consistent improvement in performance compared to linear regression and decision tree models, confirming the effectiveness of hyperparameter tuning. The R2 Score of 0.87 indicates that approximately 87% of the variance in total\_amount can be explained by the features used in the model.

## 8. Discussion of Strengths, Weaknesses, and Possible Improvements

### 8.1. Strengths

* **Robust Preprocessing:** Handled missing values and engineered new features effectively, leading to a clean and enriched dataset.
* **Comprehensive Model Comparison:** Evaluated multiple models, providing a clear justification for selecting the best-performing one.
* **Effective Hyperparameter Tuning:** Utilized GridSearchCV to optimize the chosen model, resulting in improved performance.
* **Interpretability (Decision Tree/Random Forest):** While Random Forests are less interpretable than single Decision Trees, feature importances can still be extracted to understand the most influential factors.

### 8.2. Weaknesses and Error Analysis

* **Computational Cost:** GridSearchCV with Random Forest on a large dataset can be computationally expensive and time-consuming. RandomizedSearchCV could be a more efficient alternative for initial exploration.
* **Outlier Sensitivity:** While some outlier handling might have occurred implicitly during preprocessing, extreme outliers in fare amounts or trip distances could still impact model training, especially for certain algorithms.
* **Feature Engineering Depth:** More complex feature engineering, such as incorporating external data (e.g., weather conditions, time of day-specific traffic, special events), could potentially enhance model accuracy further.
* **Geospatial Features:** PULocationID and DOLocationID are treated as categorical here. Converting them into more meaningful geospatial features (e.g., distances to central business districts, density of pickups/dropoffs in an area) could yield better insights.

### 8.3. Possible Improvements

* **Advanced Outlier Treatment:** Implement more sophisticated outlier detection and treatment methods (e.g., IQR method, Z-score, Winsorization) for numerical features.
* **Time Series Analysis:** Given the datetime features, exploring time series models or incorporating lagged features could capture temporal patterns in taxi demand and fares.
* **Deep Learning Models:** For higher accuracy, especially with a large dataset, exploring deep learning models (e.g., Neural Networks) might be beneficial, provided sufficient computational resources.
* **Ensemble Stacking/Blending:** Combine predictions from multiple diverse models (e.g., Linear Regression, Decision Tree, Gradient Boosting) to achieve even better predictive performance.
* **Feature Interaction:** Systematically explore interactions between features (e.g., trip\_distance multiplied by passenger\_count) that might have synergistic effects on the target variable.
* **Explainable AI (XAI):** Employ XAI techniques (e.g., SHAP, LIME) to better understand model predictions and the contribution of individual features, enhancing trust and interpretability.

## 9. Conclusion & Next Steps

The machine learning model successfully predicts the total\_amount of taxi fares with a high degree of accuracy, leveraging features from the NYC taxi dataset. The Random Forest Regressor proved to be the most effective model for this task after hyperparameter tuning, achieving an R2 score of 0.87.

**Key Findings:**

* Trip distance and fare amount are strong indicators of total amount.
* Categorical features like payment typeand rate code also contribute to the prediction.
* Time-based features (pickup hour, day of week, trip duration) add valuable contextual information.

**Next Steps / Future Work:**

1. **Refine Feature Engineering:** Explore more complex engineered features, especially those related to geographical patterns and peak demand times.
2. **External Data Integration:** Incorporate external datasets such as real-time traffic data, weather information, or event schedules to enhance predictive power.
3. **Deploy and Monitor:** Once finalized, the model can be deployed as an API for real-time predictions and continuously monitored for performance degradation.
4. **Cost-Benefit Analysis:** Evaluate the trade-off between model complexity, computational cost, and incremental performance gains for practical deployment.

This report demonstrates a comprehensive machine learning pipeline, from data exploration to model deployment considerations, providing a solid foundation for further development and refinement.