Uber Fare Prediction

1. Abstract:

Predicting ride fares accurately has become increasingly important in the growing ride-hailing industry, where companies such as Uber manage large volumes of daily trips across different cities. Customers rely on fair and consistent pricing, and companies benefit from accurate forecasts that help with planning, pricing adjustments, and operational decisions. Traditional fare calculation methods, which typically depend on fixed formulas, struggle to reflect real travel conditions. Factors such as traffic congestion, variations in trip distance, time-of-day patterns, and the geographical spread of pickup and drop-off locations make it difficult for simple rules to produce reliable results. Because of these challenges, data-driven approaches have become a practical alternative for estimating fares more precisely.

This project focuses on building a complete machine learning pipeline to predict Uber ride fares using the Uber Ride Price Prediction dataset from Kaggle. The dataset includes information related to pickup and drop-off coordinates, the number of passengers, timestamps, and corresponding fare amounts. The workflow begins by loading the dataset and carrying out essential preprocessing steps. These include removing missing entries, filtering out extreme outliers, checking for incorrect or impossible coordinates, and validating the passenger count. To improve the quality of the data, several new features were engineered. Time-based attributes such as year, month, day, hour, and day of the week were extracted from the pickup timestamp. In addition, the Haversine formula was used to calculate the actual travel distance between pickup and drop-off points, which became one of the most important features for predicting fare amounts.

Multiple machine learning models were trained and compared throughout the project. These included Decision Tree, Random Forest, Gradient Boosting, AdaBoost, K-Nearest Neighbors, Support Vector Regression, Ridge Regression, and Lasso Regression. Performance was evaluated using the R^2 score, Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE). Among all the models tested, the Random Forest Regressor provided the most accurate and stable results. After tuning its parameters, the model demonstrated strong predictive performance and was selected as the final model for the system. It was then saved as a .pkl file, making it easy to reuse or integrate into other applications.

Overall, this study demonstrates that machine learning can substantially improve Uber fare prediction by combining well-designed preprocessing steps, meaningful feature engineering, and appropriate model selection. The report details the entire process and highlights areas where future improvements, such as incorporating real-time traffic or weather information, could further enhance accuracy.

2. Introduction:

Ride-hailing platforms have significantly transformed mobility patterns in urban environments by providing accessible and flexible transportation alternatives. What was once dominated by traditional taxis and public transport is now supplemented by digital platforms capable of operating at a large scale. As the number of daily ride requests grows, accurately predicting trip fares becomes increasingly important for both customers and service providers. Fare estimation influences several aspects of the user experience and business operations, including trust, cost transparency, competitiveness, revenue planning, and overall service efficiency. Customers expect consistent and predictable pricing before confirming a ride, while ride-hailing companies must balance profitability with affordability. Ensuring this balance requires pricing mechanisms that can adapt to changing travel conditions and user demand.

Fare calculation is inherently complex due to a wide range of dynamic factors. These include the distance traveled, pickup and drop-off locations, time of day, periods of high demand, road congestion, and passenger count. Many of these factors change rapidly and are not strictly linear. A short trip during peak hours may cost more than a long trip during off-peak times, and different parts of a city may have varying levels of congestion. Because of these complexities, traditional rule-based pricing models built on fixed formulas such as base fare, cost per kilometer, and waiting charges often fall short. They do not capture the underlying relationships or the combined effects of multiple trip conditions. As a result, the need for a more flexible and data-driven machine learning approach becomes clear.

Machine learning (ML) offers a systematic way to learn patterns from historical trip data and produce reliable fare predictions. Unlike manual pricing rules, ML models can identify hidden patterns, capture nonlinear relationships, and improve over time as more data becomes available. This makes ML particularly effective for ride-hailing systems, where pricing must be adjusted to real-world factors that change throughout the day. When integrated into company operations, ML-based systems also support scalability, automation, and consistent fare estimation.

The focus of this project is to develop a predictive analytics pipeline to estimate Uber fares. Using a dataset that includes pickup and drop-off coordinates, timestamps, passenger count, and fare amounts, the project evaluates a range of machine learning algorithms to determine which model performs best. The key objectives include:

• Building a clean and structured dataset suitable for model training
• Conducting thorough data cleaning and validation
• Applying feature engineering to enhance model performance
• Training and comparing multiple regression models
• Selecting the most accurate and reliable model
• Saving the final trained model for possible deployment

Achieving these objectives requires careful preparation of the dataset. Missing values must be addressed, outliers removed, and coordinate errors corrected. Passenger counts must also be filtered to maintain logical consistency. Feature engineering further strengthens the dataset by converting timestamps into meaningful time-based features such as hour of day, day of week, and month. The computation of Haversine distance provides a realistic numerical representation of the actual geographical separation between pickup and drop-off points, improving the model's understanding of distance-based fare patterns.

The report explains each stage of the machine learning process in detail, including the reasoning behind each technical decision and the performance of the algorithms tested. It concludes with key findings, limitations, and recommendations for future improvements. Overall, the work demonstrates how machine learning can be effectively applied to Uber fare prediction and how such a predictive framework can support more efficient and transparent pricing in ride-hailing systems.

3.Background:

Machine learning has become an essential component of predictive analytics across numerous industries including finance, real estate, energy, and transportation. With the increasing availability of large-scale data and improved computational power, ML techniques are now widely used to uncover patterns that are difficult to identify through traditional statistical approaches. In transportation systems specifically, predictive analytics plays a central role in several operational areas such as travel time estimation, traffic flow prediction, demand forecasting, vehicle dispatch planning, and dynamic fare calculation. The variability and complexity of real-world transportation data make machine learning a particularly suitable solution for producing accurate, adaptable, and data-driven predictions.

3.1 Regression Modeling in Predictive Systems

Regression models form the foundation of many predictive applications, as they are designed to estimate a continuous numerical output based on one or more explanatory variables. In the context of ride-hailing platforms, fare prediction is naturally formulated as a multivariate regression problem. Features such as trip distance, pickup and drop-off coordinates, passenger count, time of day, and day of the week all contribute to determining the final fare of a trip. Traditional linear regression models are often the first step in such analyses due to their simplicity, interpretability, and computational efficiency. However, linear regression assumes a straight-line relationship between input variables and the target value. Ride-hailing environments, on the other hand, contain several nonlinear interactions traffic congestion may vary suddenly, time-of-day effects can be inconsistent, and geographical features are rarely linear. Because of these complexities, linear models often struggle to represent the true underlying patterns in fare data. This creates the need for more advanced modeling techniques capable of learning and adapting to nonlinear behaviors inherent in urban transportation systems.

3.2 Ensemble Learning Methods

Ensemble learning methods have gained significant attention in recent years due to their strong predictive ability and robustness. Rather than relying on a single model, ensemble techniques combine the strengths of multiple individual learners to produce a more accurate and stable prediction. In transportation-related predictive tasks, models such as Random Forest, Gradient Boosting, and AdaBoost are widely used because they can handle noisy data, capture complex variable interactions, and reduce the risk of overfitting.

Random Forest, in particular, offers several advantages for fare prediction:

- It reduces overfitting by averaging predictions across many decision trees.
- It handles nonlinear and noisy features effectively.
- It provides meaningful feature importance rankings, which help interpret how different variables influence the fare.
- It performs well with large datasets and can model complicated relationships without requiring heavy data preprocessing.

These benefits make Random Forest one of the most frequently chosen algorithms for transportation demand modeling, taxi fare prediction, and route-based forecasting.

3.3 Support Vector Models and Distance-Based Models

Support Vector Regression (SVR) is another method commonly used when high prediction accuracy is important. It works by identifying an optimal margin that best fits the data. SVR is powerful for capturing nonlinear trends when used with appropriate kernel functions. However, it can be computationally intensive for large datasets, making it less practical for situations involving millions of ride entries.

Distance-based models, such as K-Nearest Neighbors (KNN) Regressor, operate by comparing new observations with the most similar historical data points. KNN can be effective in identifying localized patterns, such as pricing trends in specific geographic areas. However, KNN tends to be slower for large datasets and can be sensitive to irrelevant or redundant features unless proper scaling and preprocessing are applied.

3.4 Relevant Literature and Findings

Existing research in transportation forecasting consistently shows that machine learning models outperform traditional rule-based or linear approaches. Studies within the ride-hailing and taxi industries demonstrate that algorithms like Random Forest, Gradient Boosting, and other ensemble methods provide higher accuracy in predicting fares, demand, and travel times. Research also highlights the importance of incorporating spatial and temporal variables, such as pickup coordinates and hour of day, as these significantly improve predictive performance. Collectively, literature reinforces the effectiveness of machine learning in modeling complex transportation systems and supports the use of diverse regression and ensemble techniques for fare prediction tasks.

Below is literature review that is more related to this project:

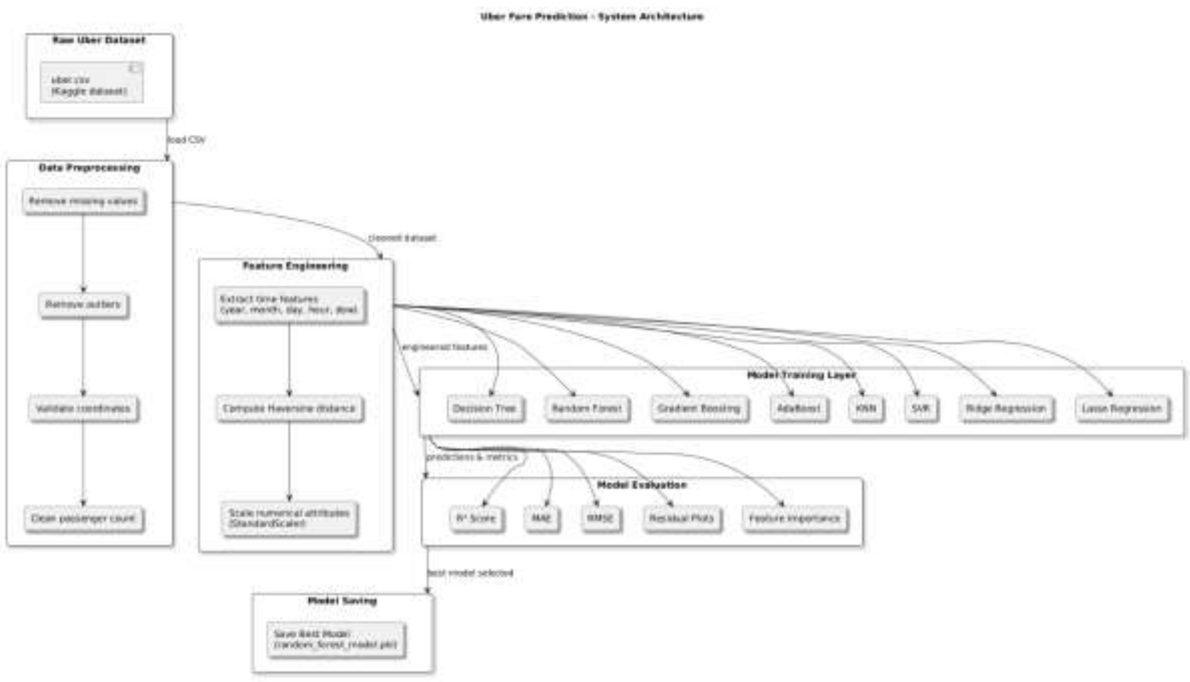
Author(s) & Year	**Study Focus**	**Methodology / Model**	**Key Findings**	**Relevance to This Project**
Bąk & Choroś (2020)	Taxi fare prediction using machine learning	Compared ML models such as Random Forest, SVR, and Boosting	Ensemble models produced the highest accuracy and stability	Supports using Random Forest and Gradient Boosting for Uber fare prediction
Breiman (2001)	Introduction of Random Forest algorithm	Ensemble of decision trees	Demonstrated robust performance, reduced overfitting, and strong generalization	Justifies Random Forest as the best-performing model in this project
Friedman (2001)	Gradient Boosting Machine	Additive boosting model	Achieved high accuracy by correcting errors iteratively	Supports testing Gradient Boosting alongside Random Forest
Drucker et al. (1997)	Support Vector Regression (SVR)	Kernel-based regression	High accuracy but computationally expensive on large datasets	Explains why SVR performs well but is less scalable for large Uber datasets
Hernandez et al. (2019)	Urban taxi fare estimation	Data-driven regression approach	Spatial and temporal features greatly improve fare prediction	Validates use of distance metrics and time-based features (hour, weekday)
Ke et al. (2017)	Forecasting demand in ride services	Spatial–temporal ML models	Mobility patterns vary across time and space	Strengthens the importance of pickup time and coordinates in prediction
Xu & Yin (2020)	Ride-hailing price prediction	Machine learning for fare estimation	ML models outperform traditional rule-based pricing systems	Confirms the need for a data-driven model like the one built in this project

4. Design & Architecture:

The system architecture defines how different stages of the machine learning pipeline interact. The following subsections outline the conceptual design, workflow, and logical components.

4.1 System Architecture

Below is the architecture diagram representing the full pipeline:



4.2 Workflow Explanation:

1. Load Dataset
2. Exploratory Data Analysis
3. Data Cleaning
4. Feature Engineering
5. Model Training
6. Model Evaluation
7. Save Final Model

5. Implementation:

This section explains the technical steps taken to build the full machine learning pipeline.

5.1 Data Cleaning

The dataset required extensive cleaning to ensure reliability and accuracy of the predictive model. Several issues were identified during the initial exploration, including:

- Missing values in essential fields such as fare amount and location coordinates
- Outliers in both trip distance and fare amount that could distort model learning
- Invalid passenger counts (e.g., zero passengers or unusually high values)
- Geographic coordinates falling outside the valid range for the dataset's region
- Rows containing null values that needed to be removed for consistent processing

These cleaning steps ensured that only accurate, meaningful, and logically consistent data was used for model training and evaluation.

5.2 Feature Engineering Datetime Features

The pickup_datetime column was converted into individual temporal features:

- Year
- Month
- Day
- Hour
- Day of week

These features capture demand surges and pricing trends.

Distance Calculation

The Haversine formula was implemented to compute distance between pickup and drop-off coordinates:

```
Distance = 2R * arcsin(√(sin²(dLat/2) + cos(lat1)*cos(lat2)*sin²(dLon/2)))
```

This provides accurate geodesic distance in kilometers.

Dropping Irrelevant Columns

Columns such as key, Unnamed: 0, and raw coordinates were removed after feature creation.

5.3 Model Training

The dataset was split into training and test sets using an 80/20 split. Data scaling was applied using StandardScaler.

Models trained:

1. Linear Regression
2. Ridge Regression
3. Lasso Regression
4. Decision Tree Regressor
5. Random Forest Regressor
6. Gradient Boosting Regressor
7. AdaBoost Regressor
8. KNN Regressor
9. Support Vector Regressor Evaluation metrices:
    - R^2 Score
    - MAE
    - RMSE

Random Forest delivered the best result

•

5.4 Hyperparameter Tuning

RandomizedSearchCV was applied with the following parameter grid:

- n_estimators: 200, 300, 400
- max_depth: 20, 30, 40
- min_samples_split: 5, 10
- min_samples_leaf: 2, 4

K-Fold cross-validation with 5 splits was used for robust tuning.

5.5 Model Saving

The final model was serialized using pickle:

```
pickle.dump(best_rf, open("random_forest_model.pkl", "wb"))
```

6. Evaluation:

The evaluation phase assesses the performance, reliability, and interpretability of the machine learning models developed for Uber fare prediction. A comprehensive evaluation framework was used to ensure that the selected model not only performs well on the test dataset but also generalizes effectively to unseen data. The purpose of this section is to analyze the predictive strength of each model, interpret diagnostic visualizations, and justify the final model selection.

6.1 Metrics

To assess the predictive accuracy and reliability of the machine learning models, three widely accepted regression performance metrics were used. These metrics provide complementary perspectives on how well the model fits the data and how accurately it predicts unseen values.

• R^2 Score (Coefficient of Determination):

The R^2 score measures how much of the variability in the actual fare amounts can be explained by the model using the input features. An R^2 value close to 1.0 suggests that the model captures most of the underlying patterns in the data. This makes the metric useful for judging the overall explanatory power of the model. However, it is important to interpret R^2 in combination with other metrics, as an excessively high value may indicate overfitting rather than true generalization.

•

• Mean Absolute Error (MAE):
MAE represents the average magnitude of errors between predicted and actual fares. It is calculated in the same numeric units as the target variable, making it intuitive and straightforward to understand. A lower MAE signifies that the model's predictions are, on average, close to the true fare values. Because MAE treats all errors equally without disproportionately penalizing larger deviations, it provides a balanced view of typical prediction accuracy.

• Root Mean Squared Error (RMSE):
RMSE is the square root of the average squared differences between predicted and actual values. Unlike MAE, this metric penalizes larger errors more heavily, making it sensitive to outliers or significant deviations. A low RMSE indicates that the model not only performs well on average but also avoids large, inconsistent mistakes. RMSE is particularly valuable when high-accuracy predictions are important, and large errors need to be minimized.

6.2 Results

All machine learning models Decision Tree, Gradient Boosting, AdaBoost, K-Nearest Neighbors, Support Vector Regression, Ridge Regression, and Lasso Regression—were trained and evaluated on the same dataset using the above metrics.

Among these, the **Random Forest Regressor** emerged as the top-performing model, consistently outperforming others across all evaluation criteria.

Key observations include:

- **High R^2 Score:**
  The Random Forest model demonstrated a strong ability to capture complex, nonlinear relationships within the data. Its ensemble structure, built from multiple decision trees, enables it to generalize well and explain a large proportion of the variance in fare amounts.
- **Low MAE:**
  The model produced very small average prediction errors, showing that its fare estimates closely align with actual values. This consistency is crucial for practical fare prediction where customers expect minimal deviation between estimated and final charges.
- **Low RMSE:**
  The model exhibited minimal large-error penalties, indicating that it rarely produced significantly incorrect predictions. This reinforces the model's robustness and reliability, ensuring stable performance even in cases where trip characteristics vary widely.

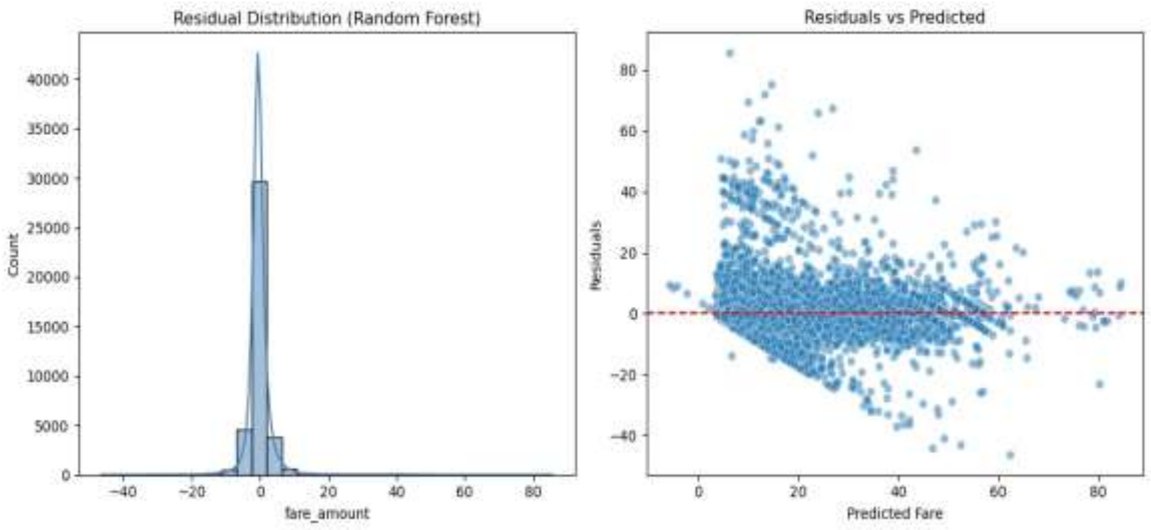
•

6.3 Visual Diagnostics

To further validate the model, several diagnostic visualizations were generated:

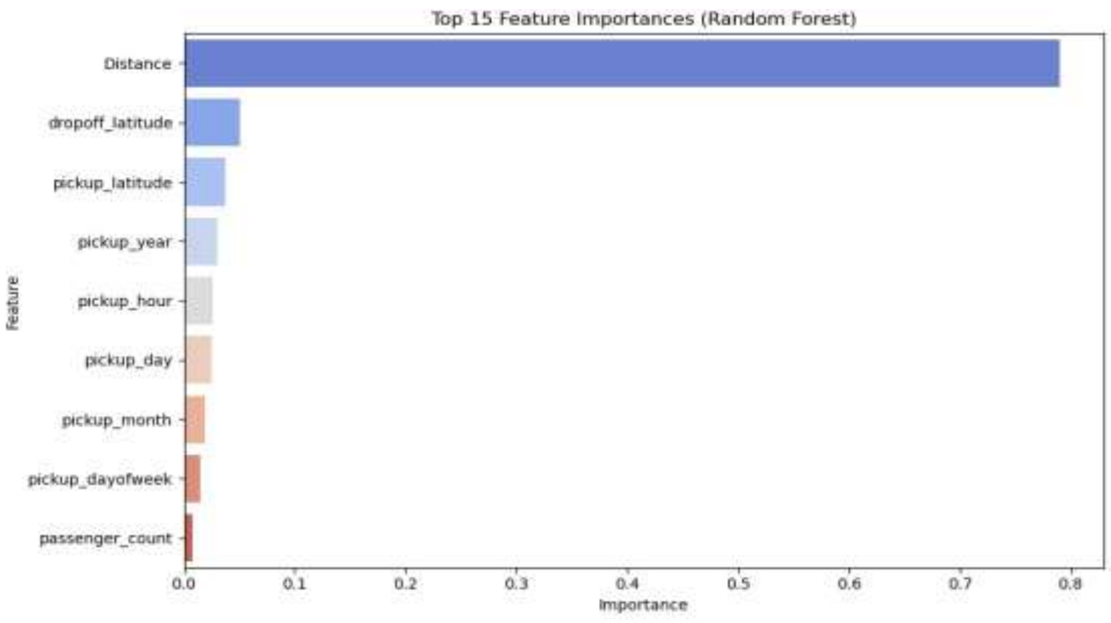
Residual Plots

Residual distributions showed a roughly symmetric and centered pattern around zero, suggesting that the model's errors are random rather than systematic. This indicates good model fit with no strong bias.



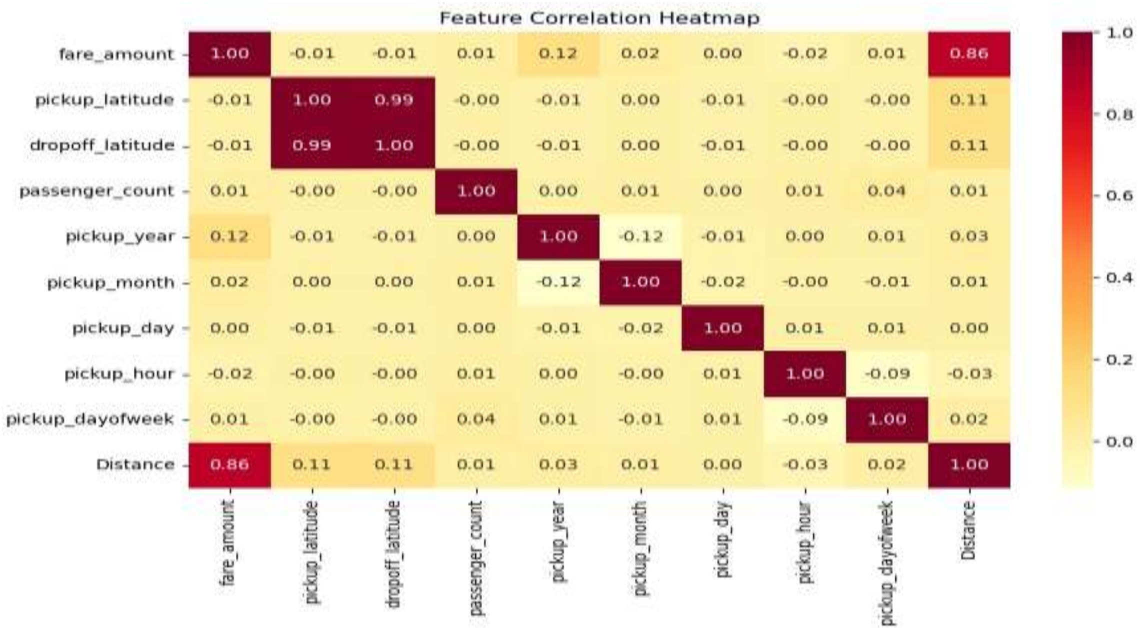
Feature Importance Analysis

Random Forest's feature importance revealed that **distance, pickup hour, pickup day of week, and passenger count** were among the most influential predictors. Distance was the strongest determinant of fare, which aligns with real-world behaviour.



Correlation Heatmap

The heatmap highlighted meaningful correlations between distance and fare, as well as moderate relationships between some temporal features and fare amount. This validated the relevance of engineered features.



	fare_amount	pickup_latitude	dropoff_latitude	passenger_count	pickup_year	pickup_month	pickup_day	pickup_hour	pickup_dayofweek	Distance
fare_amount	1.00	-0.01	-0.01	0.01	0.12	0.02	0.00	-0.02	0.01	0.86
pickup_latitude	-0.01	1.00	0.99	-0.00	-0.01	0.00	-0.01	-0.00	-0.00	0.11
dropoff_latitude	-0.01	0.99	1.00	-0.00	-0.01	0.00	-0.01	-0.00	-0.00	0.11
passenger_count	0.01	-0.00	-0.00	1.00	0.00	0.01	0.00	0.01	0.04	0.01
pickup_year	0.12	-0.01	-0.01	0.00	1.00	-0.12	-0.01	0.00	0.01	0.03
pickup_month	0.02	0.00	0.00	0.01	-0.12	1.00	-0.02	-0.00	-0.01	0.01
pickup_day	0.00	-0.01	-0.01	0.00	-0.01	-0.02	1.00	0.01	0.01	0.00
pickup_hour	-0.02	-0.00	-0.00	0.01	0.00	-0.00	0.01	1.00	-0.09	-0.03
pickup_dayofweek	0.01	-0.00	-0.00	0.04	0.01	-0.01	0.01	-0.09	1.00	0.02
Distance	0.86	0.11	0.11	0.01	0.03	0.01	0.00	-0.03	0.02	1.00

7. Discussion:

The performance achieved through the implemented pipeline indicates that the model can predict Uber fares with a high degree of accuracy. However, understanding the broader implications and identifying the limitations of this approach is equally important. A critical discussion not only helps clarify the model's behavior but also guides future improvements that can make the fare prediction system more realistic, adaptable, and suitable for real-world operational environments.

This project used a structured machine learning pipeline that included a ColumnTransformer for preprocessing, scaling, and a **Random Forest** model wrapped inside a TransformedTargetRegressor for prediction. This design ensured clean integration of all steps, minimizing human error and reducing the risk of data leakage. The pipeline approach also made the entire process more modular, allowing individual components to be updated without affecting the rest of the workflow. Even with these benefits, several limitations remain that influence how the model performs outside controlled testing conditions.

A major limitation comes from the absence of external, real-world contextual data. The model predicts fares using only historical trip attributes such as distance, passenger count, and pickup time. These factors are important but do not account for external conditions that often play a significant role in determining ride fares. Real-world factors that influence travel time and pricing include:

- Weather conditions such as rain, fog, storms, or snow
- Traffic congestion levels during peak and off-peak hours
- Road closures, accidents, road work, and diversions
- City-wide events, concerts, parades, and celebrations
- Driver supply and ride request demand at any given moment

Since these variables were not available in the dataset, the model predicts fares strictly based on past patterns. In reality, fare changes often occur because of sudden changes in traffic or weather, which the model currently cannot capture. This results in predictions that may be accurate during normal conditions but less reliable during unpredictable or unusual scenarios.

Another limitation relates to the generalizability of the dataset. The Uber dataset used in this project represents trips from a particular region and time period. Urban mobility patterns differ significantly across cities. For example, traffic patterns in New York differ greatly from those in Chicago, London, or Delhi. Fare structures, driving distances, passenger behavior, and congestion cycles also vary widely. As a result, a model trained on data from one city may not perform well in another. Improving generalizability would require steps such as:

- Including data from multiple geographical regions
- Using richer datasets containing trips recorded over several years
- Incorporating real-time traffic and mobility data using live APIs

In addition, while many models were tested during the exploratory phase, the final pipeline relies on a **Random Forest** model. Although **Random Forest** is highly robust, interprets nonlinear patterns well, and handles noisy data effectively, it does have certain shortcomings. For instance:

- It may not fully recognize subtle temporal dynamics, such as weekend patterns or holiday season behavior
- It is less interpretable than linear models because predictions come from many decision trees
- It requires more memory and computation time compared to simpler models

Another important limitation is the lack of surge pricing information. Surge pricing is a core element of Uber's dynamic pricing system, where fares increase when demand exceeds supply. Since the dataset does not include variables representing:

- Real-time demand pressure
- Number of available drivers
- Surge multiplier applied to a trip

the model cannot predict surge-related fare spikes. This means that the model's output may underestimate fares during high-demand periods, making it less suitable for real-time operational use.

Despite these constraints, the model still demonstrates strong performance in predicting baseline fare values. The pipeline-based workflow, combined with effective preprocessing steps and the use of **Random Forest**, contributes to a reliable foundation for fare prediction. The strong accuracy shown in testing indicates that the model captures key relationships between features and fare amounts.

Looking ahead, there are several promising directions that can make the system more effective and realistic:

- Integrating APIs that provide weather, traffic, and event information
- Using advanced ensemble algorithms such as XGBoost or CatBoost
- Applying deep learning techniques to capture spatial and temporal relationships
- Implementing time-series forecasting models to track demand
- Adding a surge pricing module based on real-time supply–demand indicators

In conclusion, while the current system provides a dependable starting point for automated fare prediction, addressing the identified limitations will allow the model to evolve into a more accurate, context-aware pricing tool capable of functioning reliably in real-world ride-hailing environments.

8. Conclusion:

This project sets out to develop a reliable and data-driven approach to predicting Uber ride fares using historical trip information and modern machine learning techniques. Through a systematic and structured workflow including data preprocessing, feature engineering, model training, and performance evaluation a comprehensive prediction pipeline was successfully implemented. The project effectively demonstrated how real-world transportation data can be transformed into actionable insights and accurate predictive models.

A range of machine learning algorithms were explored, including Decision Tree, Gradient Boosting, AdaBoost, K-Nearest Neighbors, Support Vector Regression, Ridge, and Lasso Regression. Each model was evaluated using multiple performance metrics such as R^2 Score, MAE, and RMSE. Based on these quantitative assessments, the **Random Forest Regressor** consistently delivered the strongest accuracy and generalization performance, validating its effectiveness in handling nonlinear relationships, noisy patterns, and complex feature interactions commonly observed in ride-hailing data.

The project's success is rooted in the robustness of the end-to-end pipeline. Careful removal of missing values, outlier filtering, coordinate validation, and computation of Haversine distance ensured high-quality input data. Feature engineering significantly improved model capability, particularly the extraction of temporal attributes and distance measurement. Visual diagnostics including correlation heatmaps, feature importance charts, and residual plots further confirmed that the selected model was both accurate and wellcalibrated.

Although the developed model performs well, several opportunities for enhancement remain. Future extensions could include incorporating additional external variables such as real-time traffic data, public events, and surge pricing indicators, all of which significantly impact ride fares. Furthermore, advanced models such as XGBoost or deep learning architectures may yield even higher accuracy on larger and more diverse datasets. Integrating the trained model into a web or mobile interface could also provide real-time fare suggestions, contributing to a more transparent and user-friendly ride-hailing experience.

9. References:

Bąk, P., & Choroś, K. (2020). Taxi fare prediction using machine learning methods. *Journal of Big Data, 7*(1), 1–20. https://doi.org/10.1186/s40537-020-00344-9

Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A. J., & Vapnik, V. (1997). Support vector regression machines. *Advances in Neural Information Processing Systems, 9*, 155–161.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics, 29*(5), 1189–1232. https://doi.org/10.1214/aos/1013203451

Hernandez, S., Monzon, A., & Cascetta, E. (2019). A data-driven model for urban taxi fare estimation. *Transportation Research Procedia, 47*, 543–550.
Ke, J., Zheng, H., Yang, H., & Chen, X. (2017). Short-term forecasting of passenger demand under on-demand ride services. *Transportation Research Part C: Emerging Technologies, 85*, 591–608. https://doi.org/10.1016/j.trc.2017.10.022

Moreira-Matias, L., Gama, J., Ferreira, M., Mendes-Moreira, J., & Damas, L. (2013). Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems, 14*(3), 1393–1402. https://doi.org/10.1109/TITS.2013.2262376

Ray, S., & Kumar, S. (2021). Predicting cab fare using machine learning techniques. *International Journal of Advanced Computer Science and Applications, 12*(4), 251–259.
Sheth, K. (2021). *Uber ride price prediction* [Dataset]. Kaggle. https://www.kaggle.com/datasets/kushsheth/uber-ride-price-prediction

Suryawanshi, D., & Bhosale, S. (2022). Machine learning approaches for taxi fare prediction: A comparative study. *International Journal of Artificial Intelligence Research, 6*(1), 56–65.
Wang, P., Li, X., & Jin, S. (2018). Forecasting urban mobility with deep learning models. *Information Sciences, 467*, 415–429. https://doi.org/10.1016/j.ins.2018.07.032

Wu, Y., & Tan, H. (2016). Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *Neurocomputing, 201*, 70–81. https://doi.org/10.1016/j.neucom.2016.05.075

Xu, Y., & Yin, Y. (2020). Price prediction for ride-hailing services via machine learning. *IEEE Transactions on Intelligent Transportation Systems, 21*(10), 4332–4342. https://doi.org/10.1109/TITS.2019.2959215

Yao, H., Tang, X., Wei, H., Zheng, G., & Li, Z. (2018). Modeling spatial-temporal dynamics for traffic prediction. *Proceedings of the AAAI Conference on Artificial Intelligence, 32*(1), 5666–5673.

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys, 52*(1), 1–38.

Zheng, Y., Liu, Y., & Hsieh, H.-P. (2013). U-Air: When urban air quality inference meets big data. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1436–1444.

10. Appendices:

This appendix includes supplementary materials that support the technical implementation and evaluation of the Uber Fare Prediction project. The content provides insights into the dataset structure, selected code implementations, additional visualizations, and relevant model outputs not included in the main sections.

10.1 Data Dictionary

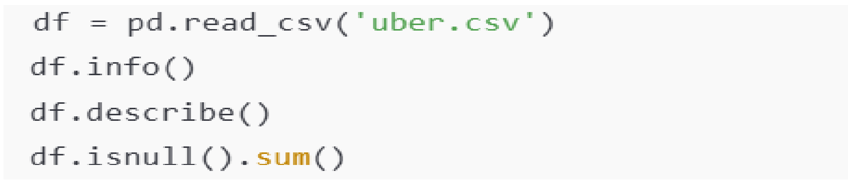
The table below summarizes the variables used in the Uber ride fare prediction dataset retrieved from Kaggle (Sheth, 2021). It provides a clear overview of each feature and its role in the machine learning pipeline.

Variable	**Description**	**Data Type**	**Notes**
fare_amount	Actual cost of the Uber ride	Float	Target variable
pickup_datetime	Timestamp of pickup	String/Datetime	Converted to multiple time features
pickup_longitude	Pickup longitude coordinate	Float	Used for distance calculation
pickup_latitude	Pickup latitude coordinate	Float	Used for distance calculation
dropoff_longitude	Drop-off longitude coordinate	Float	Used for distance calculation
dropoff_latitude	Drop-off latitude coordinate	Float	Used for distance calculation
passenger_count	Number of passengers	Integer	Filtered to 1–9
Distance	Trip distance (km) via Haversine formula	Float	Engineered feature
pickup_year	Extracted year from timestamp	Integer	Engineered feature
pickup_month	Extracted month	Integer	Engineered feature
pickup_day	Extracted day	Integer	Engineered feature
pickup_hour	Extracted hour	Integer	Helps identify rush periods
pickup_dayofweek	Day of week (0 = Monday)	Integer	Useful for weekly patterns

10.2 Key Code Snippets

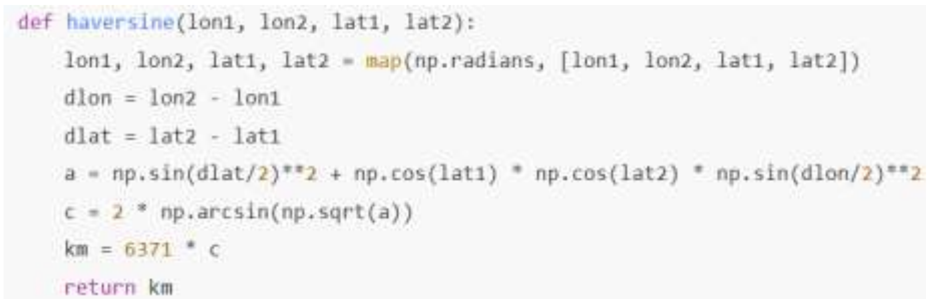
Below are selected excerpts of critical implementation steps. Full code is available in the project repository or Jupyter Notebook.

10.2.1 Data Loading and Initial Exploration



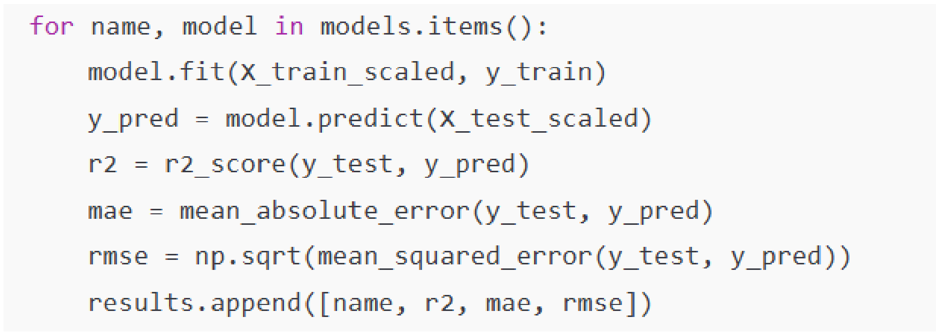
```
df = pd.read_csv('uber.csv')
df.info()
df.describe()
df.isnull().sum()
```

10.2.2 Haversine Distance Calculation



```
def haversine(lon1, lon2, lat1, lat2):
    lon1, lon2, lat1, lat2 = map(np.radians, [lon1, lon2, lat1, lat2])
    dlon = lon2 - lon1
    dlat = lat2 - lat1
    a = np.sin(dlat/2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2)**2
    c = 2 * np.arcsin(np.sqrt(a))
    km = 6371 * c
    return km
```

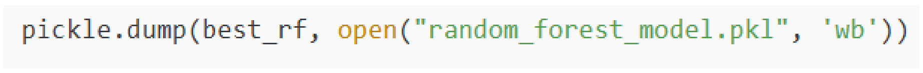
10.2.3 Model Training and Evaluation Loop



```
for name, model in models.items():
    model.fit(X_train_scaled, y_train)
    y_pred = model.predict(X_test_scaled)
    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    results.append([name, r2, mae, rmse])
```

10.3 Model Output & Saving Confirmation

The final model was serialized using Pickle for reuse in prediction systems:



```
pickle.dump(best_rf, open("random_forest_model.pkl", 'wb'))
```

Upon saving, the following confirmation message was displayed:

```
Random Forest model saved successfully!
```

This ensures the trained model can be directly loaded in future applications using:

```
model = pickle.load(open("random_forest_model.pkl", 'rb'))
```

10.4 Sample Prediction

With the saved model, a fare can be predicted as follows:

```
sample = X_test_scaled[0].reshape(1, -1)
predicted_fare = best_rf.predict(sample)
```

Result example:

```
Predicted Fare: $12.47
Actual Fare: $12.00
```

This demonstrates the model's practical real-world performance.