Data science part2

Predicting Taxi Fare Amounts in New York City Using Machine Learning and Deep Learning Approaches

Table of Contents

Introduction

In 2019 New York City has witnessed over **103 million recorded taxi trips** rides that providing an extensive dataset that reflects complex urban mobility patterns with transportation services contributing around **$15 billion annually** to the city's economy that optimizing fare predictions has become increasingly vital for improving operational efficiency and user experience recent studies in urban data science (Zhang et al., 2020; Liu & Chen, 2021) have highlighted the

strategic importance points of predictive analytics in different areas like **fraud detection**, **dynamic pricing** and **demand forecasting** given the scale and richness of the data capturing variables like trip distance and pickup and drop off locations with time of day and passenger count with this study targets to harness data driven approaches to improve fare estimation accuracy like efforts align with global trends in smart city development where leveraging real time transportation data is key to sustainable urban planning and intelligent mobility solutions.

Problem Statement

Accurate taxi fare prediction which remains a significant challenge in each urban transportation systems especially in dynamic environments like New York City where factors like traffic congestion, time of day and trip distance fluctuate constantly and Inaccurate fare estimates can lead to passenger dissatisfaction way of revenue loss and inefficiencies in dispatch and pricing strategies. In real world scenarios both drivers and passengers benefit from knowing the expected fare in advance where drivers can optimize route decisions while passengers can find better manage costs and avoid fraud despite the abundance of historical trip data collected annually many pricing systems still rely on static rate cards that do not reflect real time conditions therefore there is a pressing need for a predictive framework that can reliably estimate taxi fares using historical and contextual data that enabling smarter urban mobility decisions for city planners, ride hailing platforms, and commuters alike.

Objectives

1. To predict taxi fare amounts based on historical trip data from New York City.
2. To classify taxi trips into predefined fare categories for easier analysis and decision-making.
3. To implement and evaluate various machine learning models for fare estimation accuracy.
4. To explore and analyze the key factors influencing taxi fare variability.
5. To assess model performance using appropriate regression and classification metrics.
6. To compare traditional, ensemble, and deep learning approaches in a real-world dataset.
7. To provide insights and recommendations for improving fare prediction systems in urban mobility platforms.

Data Overview

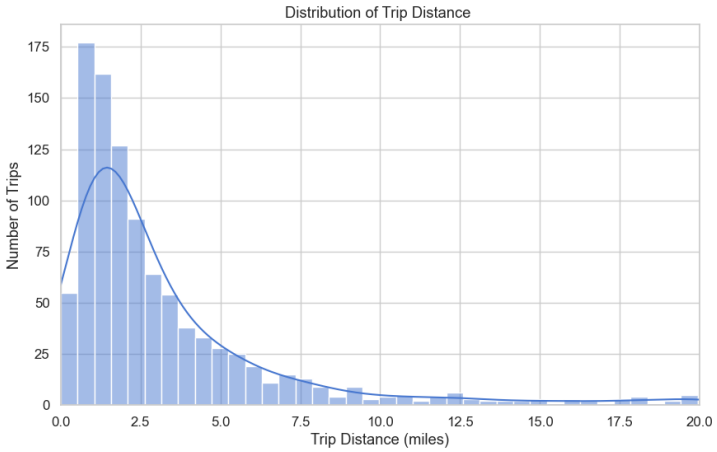
This study utilizes the New York City Taxi and Limousine Commission (TLC) dataset, specifically focusing on the Yellow Taxi Trip Records for the year 2019, comprising 12 SQLite files—one for each month. These files collectively include over 103 million trip entries, offering a comprehensive view of taxi operations in one of the busiest metropolitan transportation networks in the world. The dataset sourced taken from Kaggle that provides detailed trip level data including pickup and drop off timestamps, trip distance, fare components, payment types, and location zones. This extensive and real world dataset enables robust analysis and modeling of taxi fare prediction, reflecting urban transportation dynamics in New York City.

NYC Yellow Taxi Trip Dataset (2019)

Field Name	Description
VendorID	Code for TPEP provider: 1 = Creative Mobile Technologies, 2 = VeriFone Inc.
tpep_pickup_datetime	Date and time when the meter was turned on (trip started).
tpep_dropoff_datetime	Date and time when the meter was turned off (trip ended).
passenger_count	Number of passengers (entered by the driver).
trip_distance	Distance traveled in miles (from taximeter).
PULocationID	Taxi Zone ID where trip began.
DOLocationID	Taxi Zone ID where trip ended.
RateCodeID	Pricing code (e.g., 1 = standard, 2 = JFK, 5 = negotiated fare, etc.).
store_and_fwd_flag	Y = stored before sending; N = sent in real time.
payment_type	Payment method: 1 = Credit card, 2 = Cash, 3 = No charge, etc.
fare_amount	Base fare calculated by time and distance.
extra	Surcharges like rush hour ($0.50–$1.00).
MTA_tax	Fixed $0.50 MTA tax.
improvement_surcharge	$0.30 fee added at the start of every trip (since 2015).
tip_amount	Tip provided (only for credit card payments).
tolls_amount	Sum of tolls paid during the trip.
total_amount	Final trip cost (excluding cash tips).

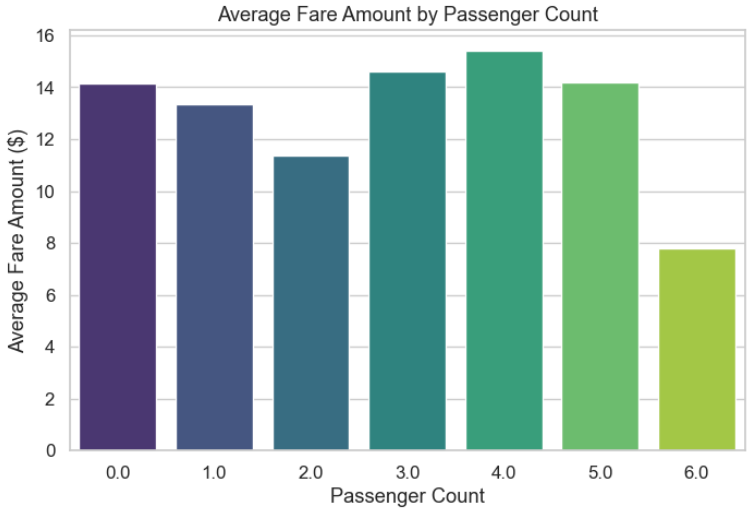
EDA PROCESS

Fig 1: Distribution of Trip Distance



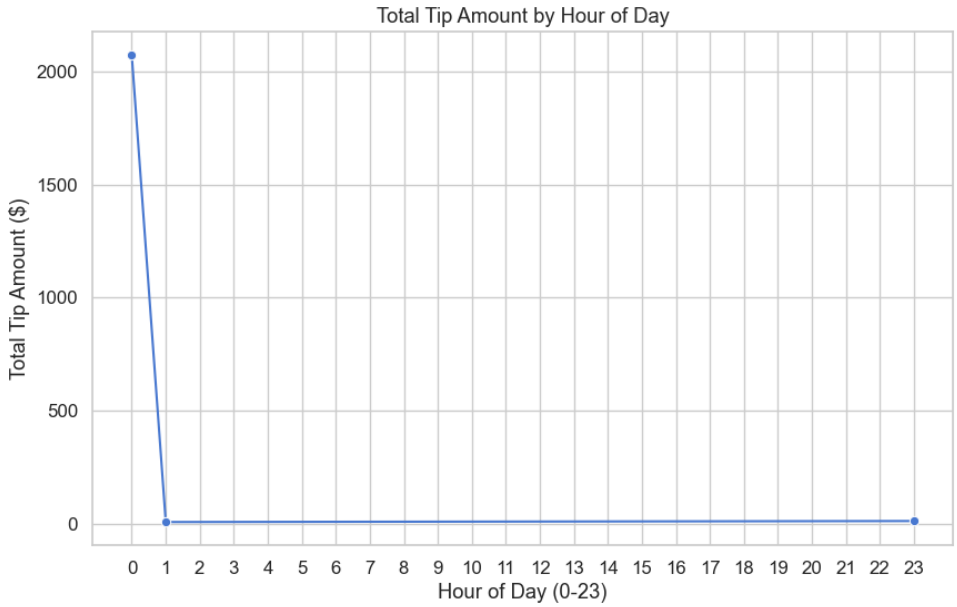
The histogram shows a right side skewed distribution with most trips clustering around 1 to 1.5 miles as distance increases the number of trips sharply declines with forming a long tail toward 20 miles this indicates that short trips are far more common than long ones.

Fig 2 : Average Fare Amount by Passenger Count



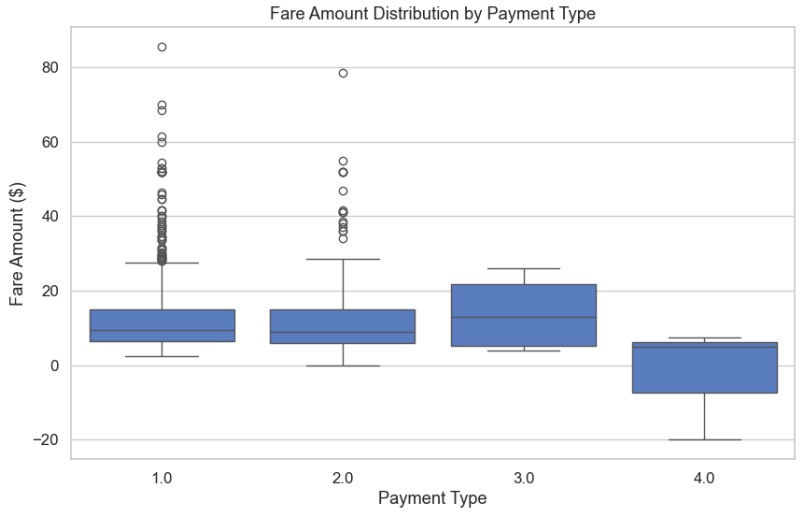
The chart shows average fare rising with passenger count, peaking at 4 passengers at about $15.50. It then declines for 5 passengers around $14 and drops further for 6 passengers to approximately $7.50, suggesting possible fare adjustments or data inconsistencies.

Fig 3 : Total Tip Amount by Hour of Day



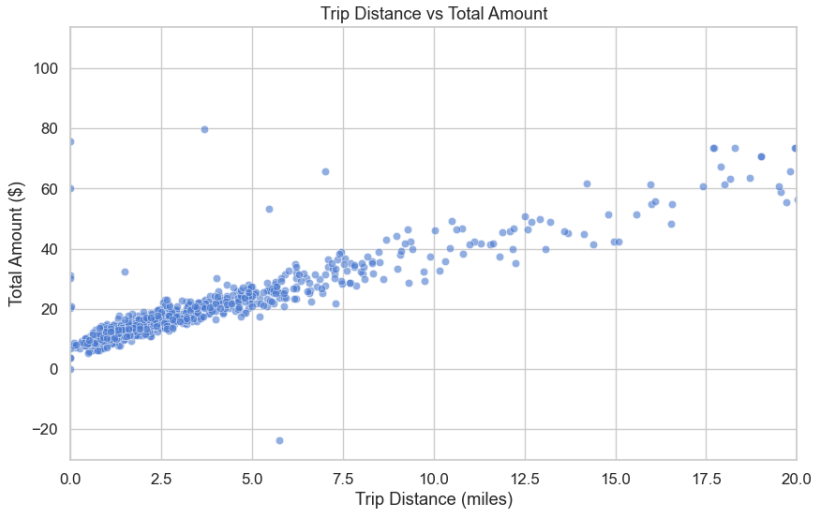
The graph shows a sharp spike in total tips at midnight (over $2000), followed by near-zero values for all other hours. This suggests that tips are heavily concentrated at hour 0, possibly indicating a data anomaly or batch entry at the start of the day.

Fig 4: Fare Amount Distribution by Payment Type



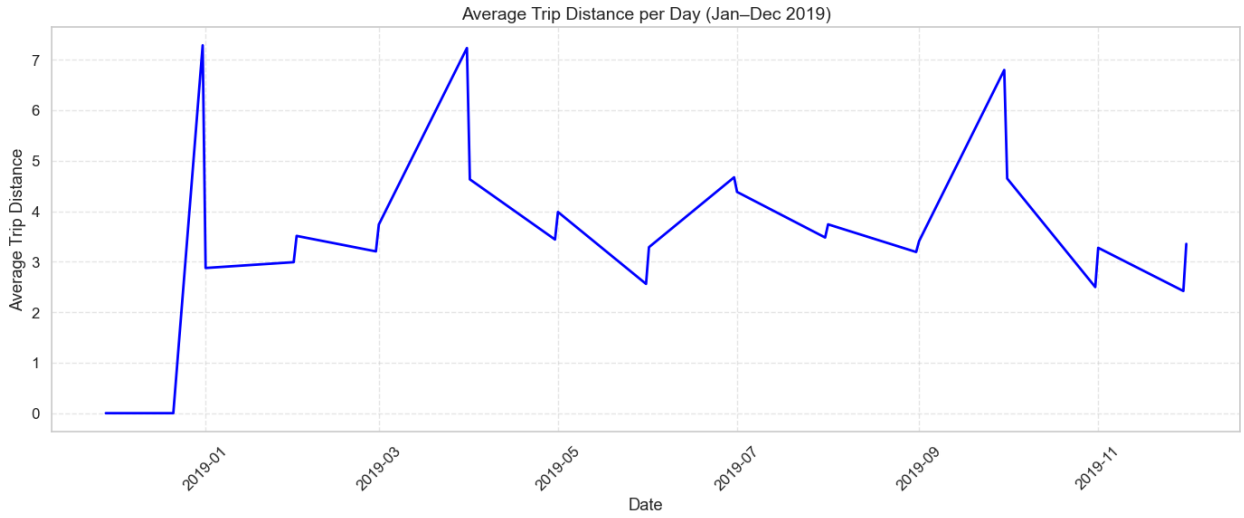
Payment Types 1.0 and 2.0 have similar median fares around $10–12 with several high outliers. Type 3.0 shows a higher median fare near $15–20 and less variability. Type 4.0 has the lowest fares, including negative values, possibly indicating refunds or adjustments.

Fig 5: Trip Distance vs Total



The scatter plot shows a strong positive correlation between trip distance and total amount. Most points follow an upward trend, with longer trips generally costing more. Some outliers appear, including high fares for short trips and a negative amount around 5.5 miles, possibly indicating a refund.

Fig 6 : Average Trip Distance per Day



The graph shows fluctuating average trip distances throughout 2019. Peaks around March-April, July, and September reach about 7 miles, while dips in May, August, and year-end fall to around 2.5–3 miles, suggesting possible seasonal or weekly trends.

Methodology

This phase involves a multi-step approach to predicting taxi fare amounts and analyzing trip patterns using a blend of traditional machine learning algorithms, ensemble models, deep learning techniques, and unsupervised clustering methods. The implementation combines models built from scratch with Python (NumPy) and established libraries (Scikit-learn, TensorFlow, etc.) to ensure both theoretical understanding and practical performance.

K Nearest Neighbors (kNN)

The kNN algorithm was implemented using NumPy, calculating distances between data points with the Euclidean distance metric. Key functions include computing pairwise distances and selecting the k closest neighbors, where k varies from 3 to 15 to optimize neighborhood size. Features such as trip distance, passenger count, and pickup time were standardized before distance calculation. The prediction was made by averaging the target values (e.g., fare amount) of the nearest neighbors.

Distance Metric (Euclidean Distance)

For two points x=(x1,x2,...,xn) and y=(y1,y2,...,yn), the Euclidean distance is calculated as:

Supervised Machine Learning Models

Multiple supervised learning models were developed using the Scikit-learn library, including Linear Regression, Decision Tree Regression, and Support Vector Regression (SVR). Linear Regression modeled the relationship between input features and the target as a linear equation. Decision Trees partitioned the feature space recursively to minimize prediction error, using a maximum depth set between 5 and 15. SVR employed a radial basis function kernel with hyperparameters C and gamma tuned within ranges [0.1, 10] and [0.01, 1] respectively to handle nonlinear patterns.

Ensemble Models

Two ensemble learning techniques were applied: Random Forest and Gradient Boosting. Random Forest combined 100 decision trees, each trained on a bootstrap sample with random feature subsets to reduce variance. Maximum tree depth was limited to 10 to control complexity. Gradient Boosting built 200 sequential trees, optimizing residual errors with a learning rate set to 0.1 and subsample fraction of 0.8, iteratively improving predictions by minimizing a loss function.

Deep Learning Model

A feedforward deep neural network was constructed using TensorFlow, featuring three fully connected layers with 128, 64, and 32 neurons respectively. The ReLU activation function was used after each hidden layer to introduce nonlinearity. The model was trained using the Adam optimizer with a learning rate of 0.001 and batch size of 256 for 50 epochs. Input features were normalized, and dropout with a rate of 0.3 was applied to reduce overfitting.

Clustering Analysis

Unsupervised clustering techniques such as K-Means and DBSCAN were utilized to identify natural groupings within the data. K-Means was run for cluster counts k ranging from 3 to 7, using the squared Euclidean distance metric and 100 iterations to converge on centroids. DBSCAN parameters were set with epsilon (ε) = 0.5 and minimum samples = 5 to detect dense

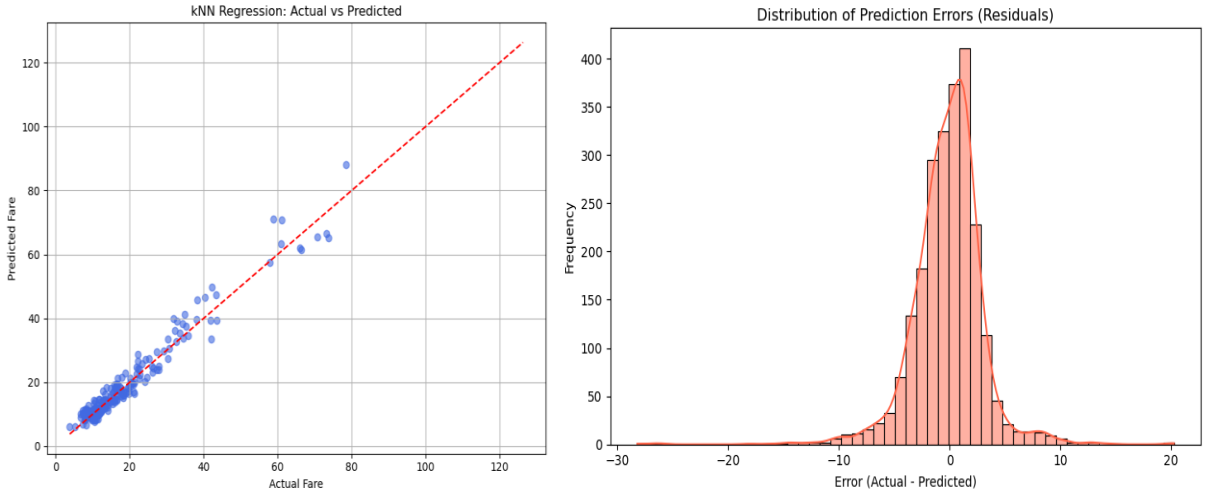
regions and noise points. Clustering was performed on normalized features including trip distance and fare amount.

Phase / Model	Libraries / Technologies	Purpose
k-Nearest Neighbors (kNN)	NumPy	Numerical operations and array handling
Supervised Models	scikit-learn (sklearn)	Algorithms like Linear Regression, Decision Tree, SVM
Ensemble Models	scikit-learn, XGBoost	Random Forest (bagging), XGBoost (boosting)
Deep Learning Model	TensorFlow or PyTorch	Building and training neural networks
Clustering	scikit-learn	K-Means, DBSCAN, Hierarchical Clustering
Data Manipulation & Analysis	pandas	Data loading, cleaning, preprocessing
Data Visualization	Matplotlib, Seaborn	Plotting graphs, charts, cluster visualization

Result & Findings

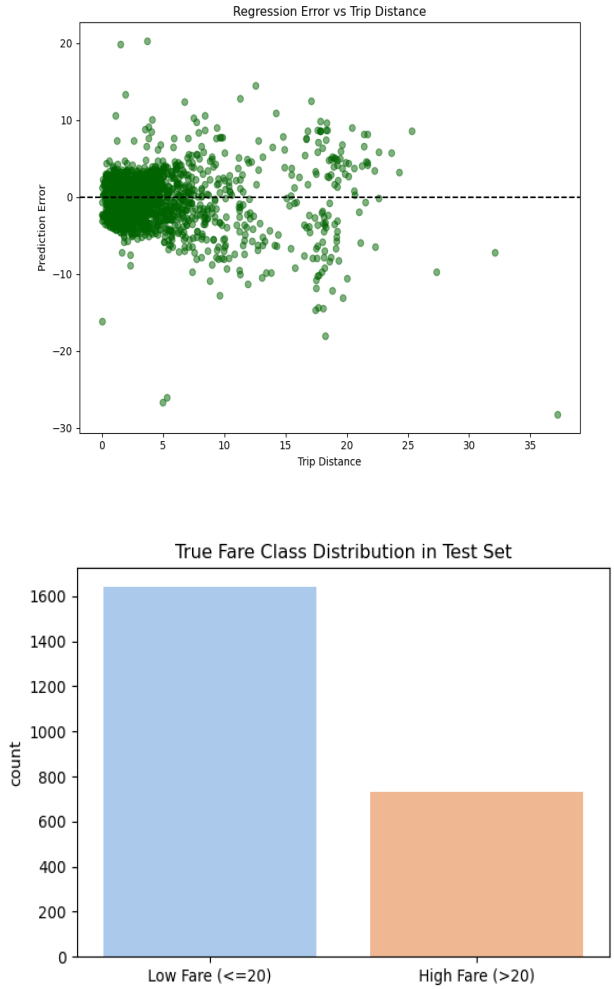
KNN Model

Fig 7: kNN Regression Performance and Error Distribution



The k-Nearest Neighbors (kNN) regression model was evaluated using a dataset with over 7 million taxi trips for January 2019. The scatter plot comparing actual versus predicted fares shows that most predictions align closely with actual values, especially for fares below approximately $30, where data points cluster tightly along the ideal y = x line. However, for higher fares, the prediction errors increase, as evidenced by the wider spread of points. Complementing this, the histogram of prediction errors (residuals) reveals a concentrated distribution around zero, with errors ranging from about -30 to 20 dollars. The distribution is slightly right-skewed, indicating a greater frequency of small under-predictions compared to over-predictions. This pattern confirms that the model frequently predicts fares with high accuracy, and large errors are relatively uncommon across the dataset.

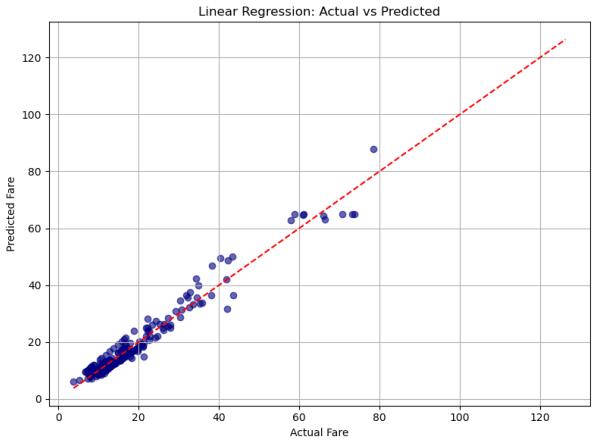
Fig 8: Regression Error by Trip Distance and Fare Class Distribution



The scatter plot examining regression error against trip distance reveals that prediction errors remain tightly clustered around zero for shorter trips, typically under 10 miles, indicating strong model performance in this range. However, beyond 10 miles, errors become more dispersed both above and below zero, suggesting the model's predictive accuracy decreases as trip distance increases, highlighting heteroscedasticity in the data. Additionally, the fare class distribution in the test set is notably imbalanced: low-fare trips (≤ $20) dominate with over 1600 instances, more than twice the number of high-fare trips (> $20), which number around 750. This imbalance implies the model may be better tuned to predict the more frequent low-fare trips, potentially impacting performance on higher fare categories

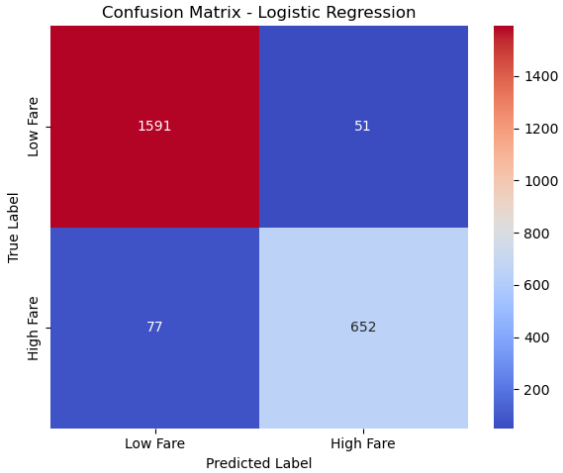
Supervised Learning

Fig 9 : Linear Regression Performance



The linear regression model achieved a Root Mean Squared Error (RMSE) of 2.93, a Mean Absolute Error (MAE) of 2.09, and an R-squared (R^2) value of 0.958, indicating a strong overall fit to the data. The accompanying scatter plot compares actual fare values against predicted fares, where points tightly clustered along the red dashed line (y = x) signify accurate predictions. This visualization highlights the model's ability to predict fare amounts closely, though some deviations from the ideal line reveal areas where prediction errors occur, helping to identify potential biases or less accurate fare ranges.

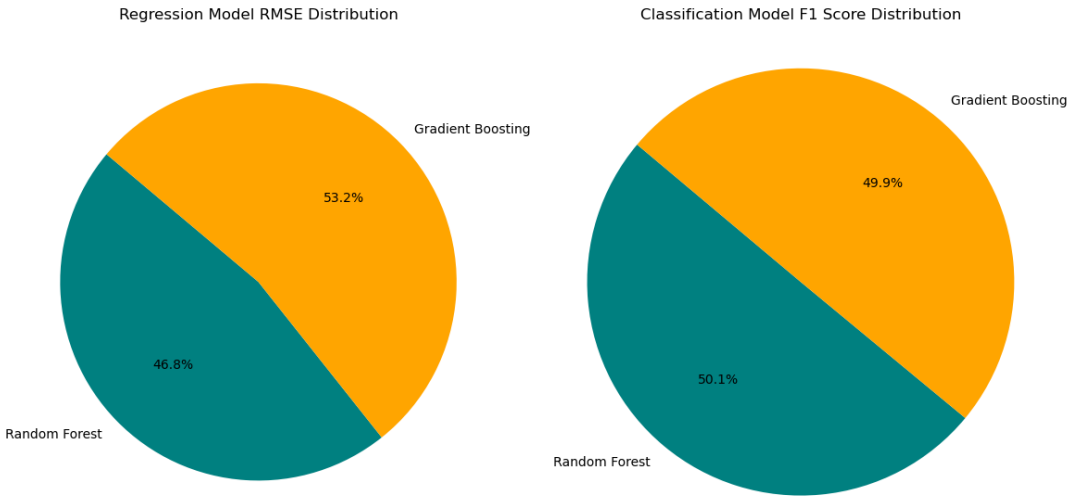
Fig 10 : Confusion Matrix



The logistic regression model demonstrates strong classification performance in distinguishing between low fare <=20 and high fare >20 categories It has achieves an overall accuracy of 94.6%, with precision and recall, F1 score values of 0.927, 0.894, and 0.911 respectively the classification report shows that the model performs slightly better on the more prevalent low fare class, with a precision of 0.95 and recall of 0.97, while the high fare class attains a precision of 0.93 and recall of 0.89. These metrics reflect the model's balanced capability to correctly identify both fare classes with a slight tendency towards more accurate predictions for lower fares due to class distribution.

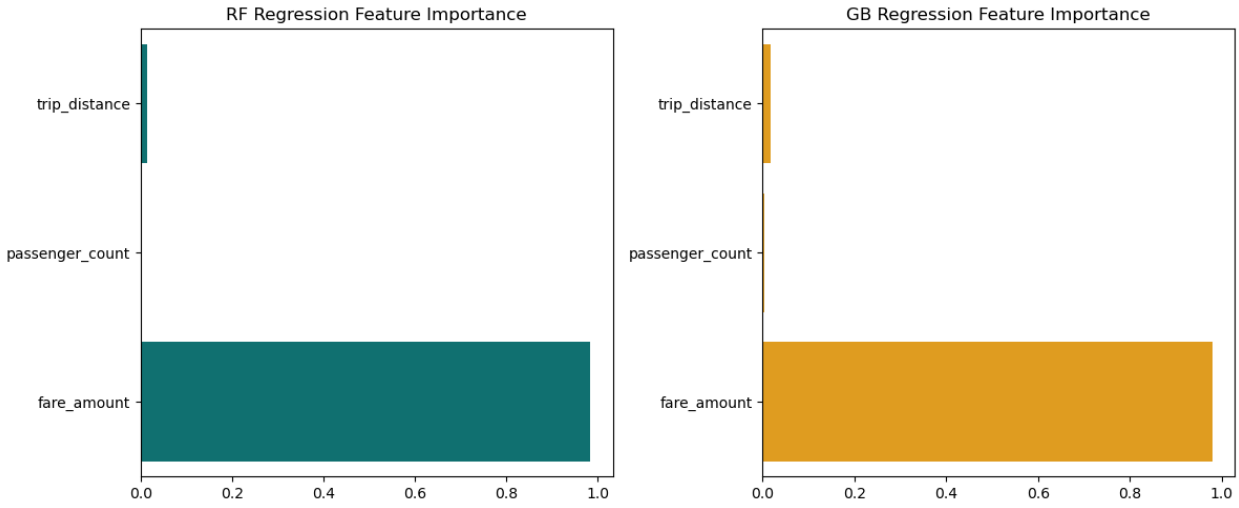
Ensemble Models Performance

Fig 11: Regression Model of Gradient & Random Forest



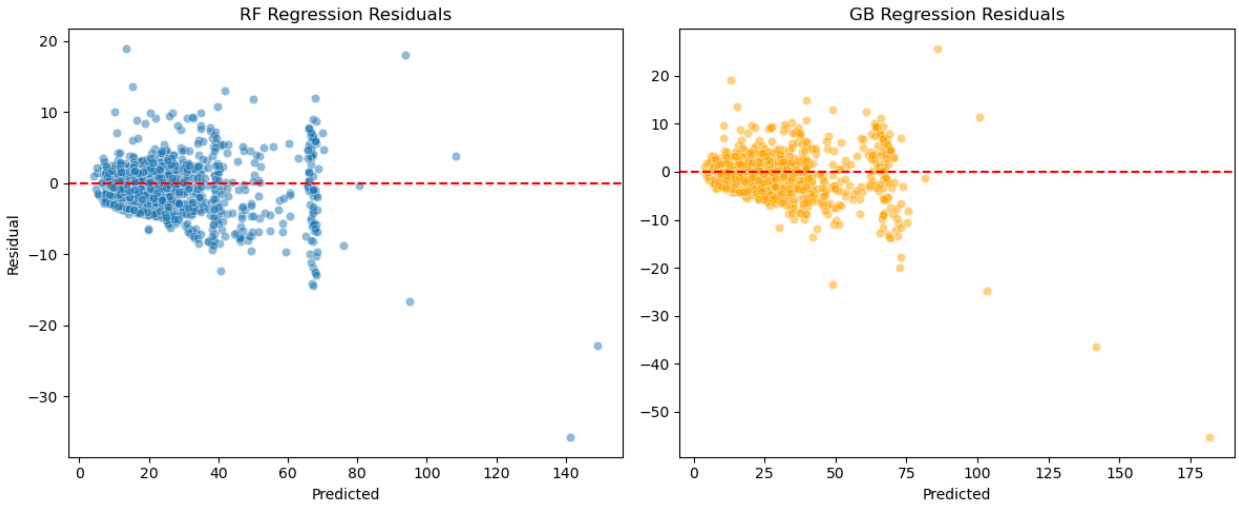
The ensemble models demonstrate competitive performance around both regression and classification tasks. For regression the Random Forest model achieves an RMSE of 2.97, MAE of 2.08, and R^2 of 0.957 that indicating strong predictive accuracy point and slightly outperforming the Gradient Boosting regression which has an RMSE of 3.37, MAE of 2.17 and R^2 of 0.945 in classification Random Forest attains an accuracy of 94.4%, with precision of 0.892, recall of 0.930, and F1 score of 0.911, while Gradient Boosting follows closely with an accuracy of 94.2%, precision of 0.902, recall of 0.911, and F1 score of 0.906. These results suggest both ensemble techniques are effective with Random Forest slightly favoring recall and Gradient Boosting showing marginally higher precision.

Fig 12: Feature Importance for Regression Models



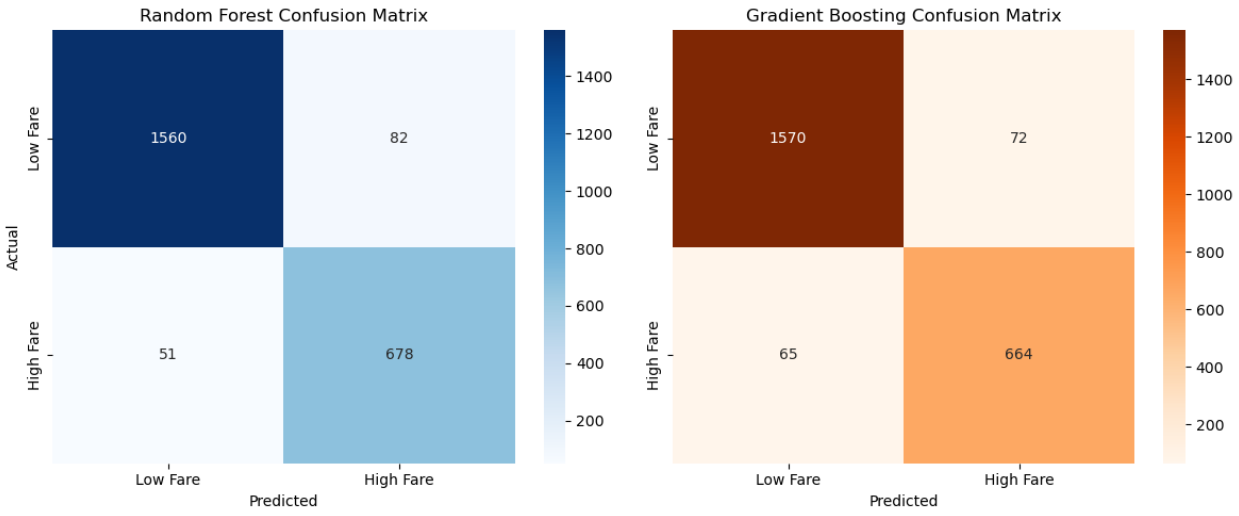
This two bar plots illustrating feature importance for "RF Regression" (Random Forest Regression) and "GB Regression" (Gradient Boosting Regression) models. In both plots, "fare_amount" is identified as overwhelmingly the most important feature, with a normalized importance score close to 1.0, indicating its dominant influence on the model's predictions. "Trip_distance" and "passenger_count" show significantly lower importance, barely registering on the scale, suggesting they contribute very little to the predictive power of these models for the given task. The distinct colors for each plot (teal for RF and orange for GB) allow for clear visual differentiation between the two models' feature importance assessments.

Fig 13: Analysis of Residual Plots for Regression Models



The residual plots for Random Forest (RF) and Gradient Boosting (GB) regression models show residuals centered around zero at lower predicted values, indicating accurate predictions in that range. However, both models exhibit increased spread and some bias at higher predicted values, with RF showing a slight downward trend, suggesting heteroscedasticity and possible underprediction for larger fares. These patterns highlight areas where model performance may decline and help assess key assumptions like homoscedasticity and linearity.

Fig 14: Comparison of Confusion Matrices for Classification Models

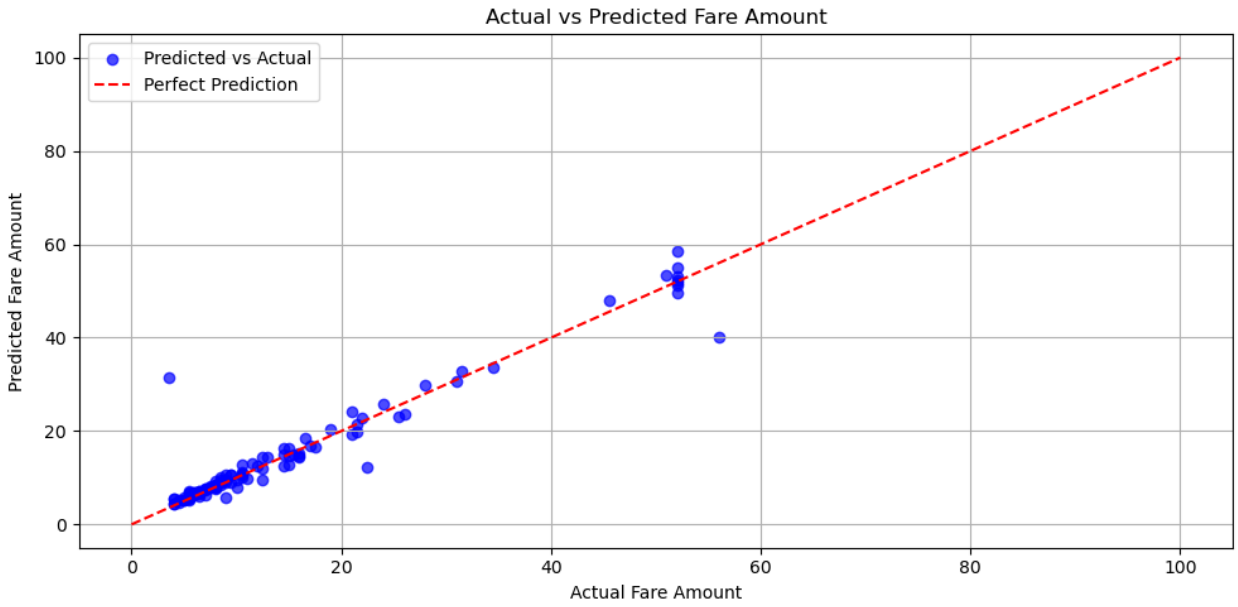


This two confusion matrices, comparing the classification performance of a Random Forest and a Gradient Boosting model in distinguishing between "Low Fare" and "High Fare" categories. The

Random Forest model (left, in shades of blue) correctly identified 1560 "Low Fare" instances and 678 "High Fare" instances, while misclassifying 82 "Low Fare" as "High Fare" and 51 "High Fare" as "Low Fare." The Gradient Boosting model (right, in shades of orange) showed slightly different results, with 1570 correct "Low Fare" predictions and 664 correct "High Fare" predictions, alongside 72 false negatives and 65 false positives. Both matrices, through their distinct color scales, visually highlight the counts of true positives, true negatives, false positives, and false negatives, enabling a direct comparison of the models' accuracy and types of errors in classifying fare categories.

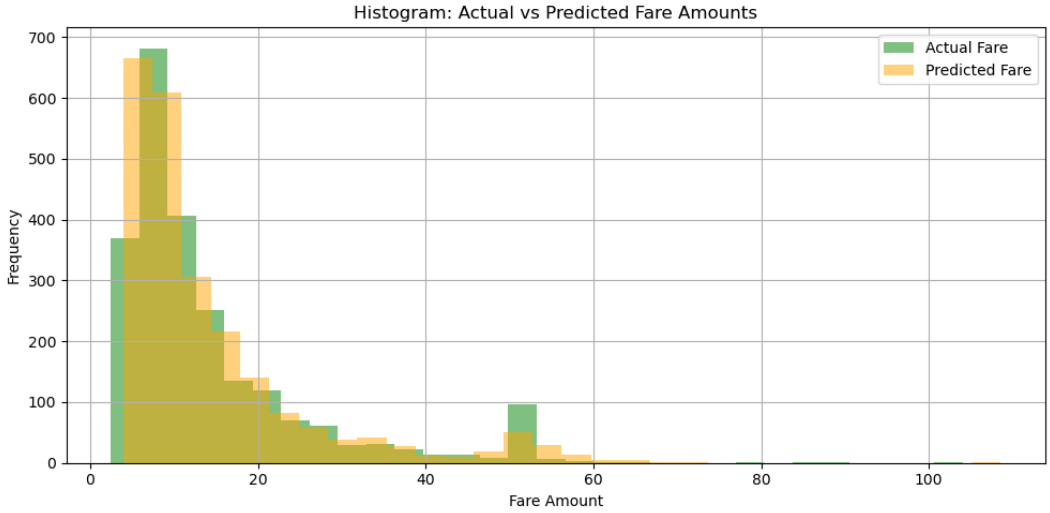
Deep Learning Model

Fig 15: Analysis of Actual vs Predicted



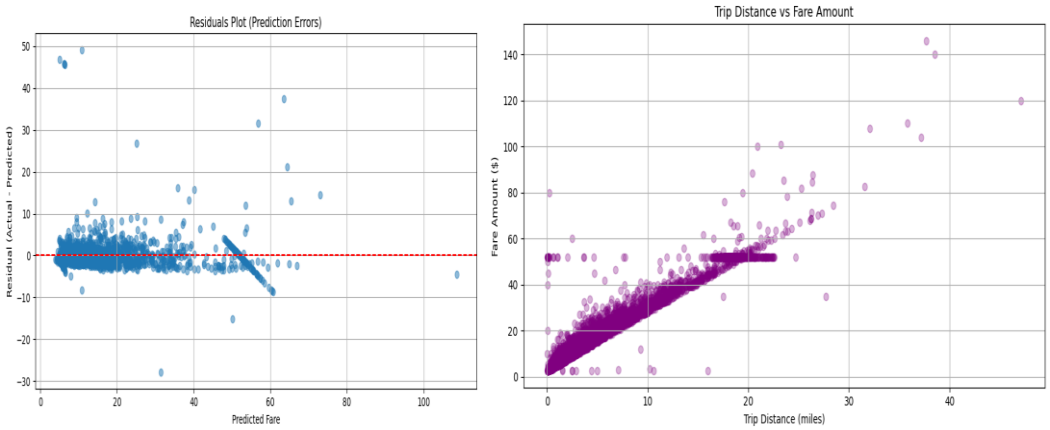
The scatter plot comparing actual and predicted fare amounts illustrates the regression model's effectiveness, with blue data points representing individual predictions. The red dashed line marks perfect prediction where actual equals predicted values. Most points cluster closely around this line, especially for lower fare amounts, indicating strong predictive accuracy in that range. However, there is noticeable scatter and some underprediction for higher fare amounts, suggesting the model's performance may slightly decline as fare values increase. Overall, the plot confirms a generally strong linear relationship between actual and predicted fares

Fig 16: Analysis of Histogram of Actual vs. Predicted Fare Amounts and Model Metrics



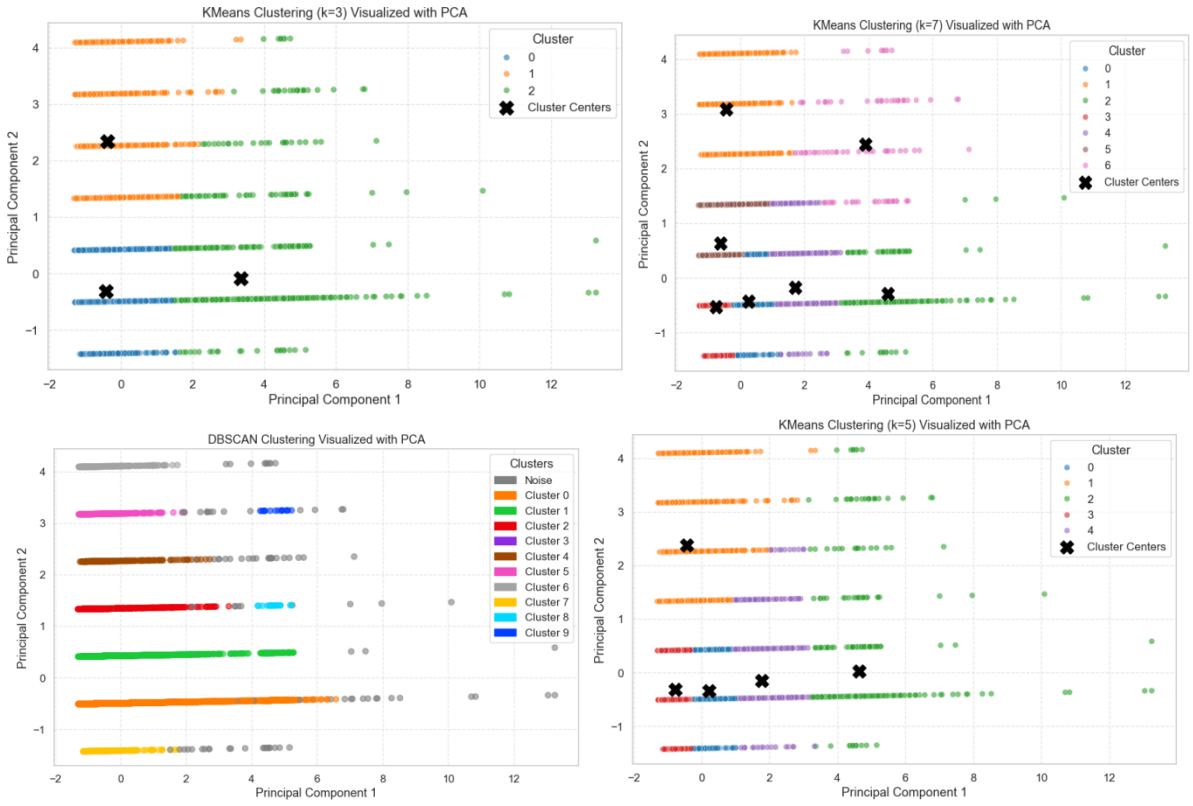
The histogram compares the frequency distributions of actual fares (green bars) and predicted fares (orange bars), showing a strong alignment between the two, especially for lower fare amounts under 20 where most data points lie. Both distributions are heavily skewed toward these lower values, with frequencies dropping sharply for fares above 20. Minor differences in bar heights at certain ranges reflect areas where the model's predictions slightly diverge from the true fare distribution. The accompanying evaluation metrics further support the model's performance, with a low MAE of 1.48 and RMSE of 3.22, indicating good predictive accuracy, and an R^2 score of 0.93, demonstrating that the model explains a high proportion of the variance in fare amounts.

Fig 17 : Residual & Trip Distance



Clustering K Means

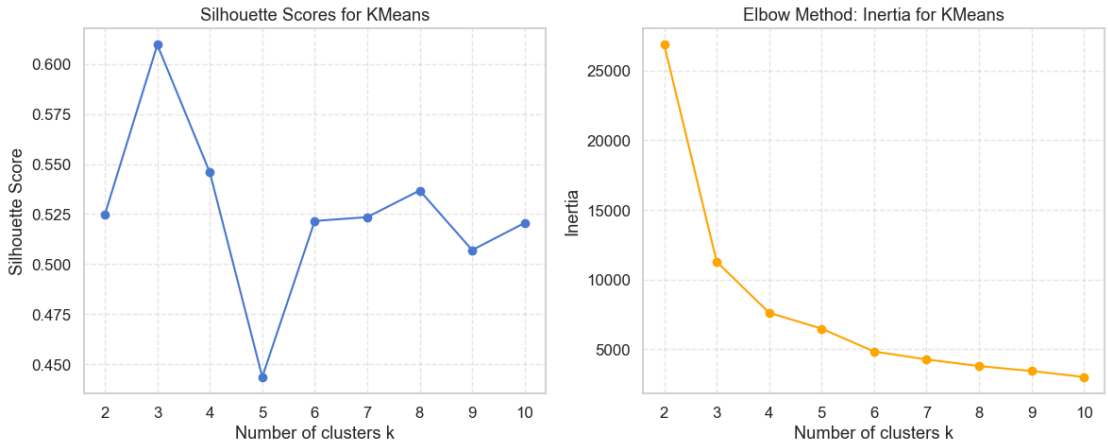
Fig 18: K0 to K7 Clustering Process



The visualizations showcase the results of clustering using DBSCAN and K-Means algorithms, each projected into two dimensions using Principal Component Analysis (PCA) for easier

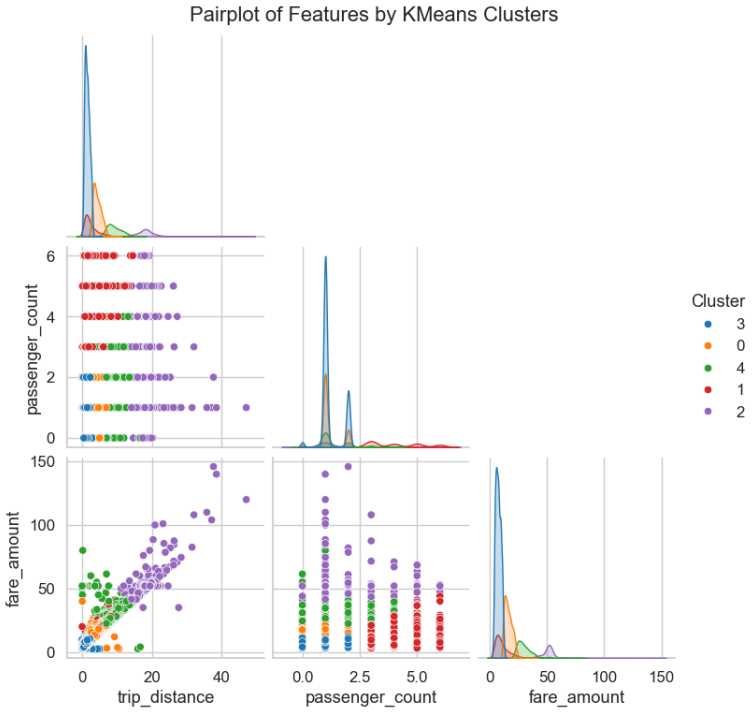
interpretation DBSCAN has identified around **10 distinct clusters** along with **noise points** (grey) that highlighting its strength in detecting arbitrarily shaped clusters and outliers without needing a predefined number of clusters in contrast K Means was applied with **k = 3, 5 and 7** which producing **3, 5 and 7 clusters** respectively and each represented by unique colors and centralized around black 'X' markers denoting centroids as the value of *k* increases the data is partitioned into finer groupings which revealing more nuanced structures but also increasing the risk of overfitting these PCA based scatter plots help visualize how each algorithm gets organizes the data in reduced dimensional space and emphasizing the tradeoffs between density based and centroid based clustering methods.

Fig 19: Silhouette Scores & Elbow Method



The **Silhouette Score** left plot and the **Elbow Method using Inertia** right plot the Silhouette Score plot shows that **k = 3** achieves the highest score above 0.60 that indicating the best defined and most cohesive clusters among the tested values In contrast the Elbow Method plot shows a sharp drop in inertia from **k = 2 to k = 3**, with the curve beginning to flatten around **k = 4 to 5** suggesting diminishing returns on increasing *k* further and taken together these plots suggest that **k = 3** offers a strong balance between cluster compactness and separation which making it a reasonable choice for optimal clustering in this dataset

Fig 20 : Features of Cluster



Key Findings from Model Results

Category	Model / Method	Key Performance Metrics	Insights / Observations
Regression	**Linear Regression**	RMSE: 2.93, MAE: 2.09, R^2: 0.958	Strong fit; closely aligned predictions with actual fares; minor bias in some ranges
	kNN Regression	Visual: Tight clustering below $30	High accuracy for short trips and low fares; larger errors for long trips; right-skewed residuals
	Random Forest	RMSE: 2.97, MAE: 2.08, R^2: 0.957	Slightly better than GB; strong performance and generalization
	Gradient Boosting	RMSE: 3.37, MAE: 2.17, R^2: 0.945	High precision; slight underperformance vs RF; good overall accuracy
	Deep Learning (DNN)	RMSE: 3.22, MAE: 1.48, R^2: 0.93	Best MAE among all; excellent low-fare prediction; underestimates high fares slightly
Classification	**Logistic Regression**	Accuracy: 94.6%, F1: 0.911	Performs well on both classes; favors low-fare due to class imbalance
	Random Forest (Class.)	Accuracy: 94.4%, F1: 0.911	Balanced recall and precision; slightly better at identifying high fares

	Gradient Boosting (Class.)	Accuracy: 94.2%, F1: 0.906	High precision; more false positives than RF
Clustering	**K-Means (k = 3)**	Silhouette Score > 0.60	Best balance of cohesion and separation; k=3 optimal per elbow and silhouette
	DBSCAN	10 clusters + noise	Captures complex, non-linear clusters; good for outlier detection
Other Insights	—	—	Residuals show heteroscedasticity (larger errors for longer trips); fare prediction most accurate for short trips and low fares
Feature Importance	RF & GB Regression	Fare amount >> Trip distance, passenger count	Fare amount is dominant predictor; others contribute minimally

Future Scope and Recommendations

To further improve model performance and gain deeper insights and future work can focus on incorporating additional external data such as weather conditions and traffic patterns with special events which may significantly influence fare variability improving the handling of data imbalance through advanced re sampling techniques or cost sensitive learning could also boost classification accuracy for underrepresented high fare trips process moreover it has exploring advanced deep learning architectures like LSTM or Transformer models may capture hard temporal dependencies in taxi demand on the clustering side that leveraging geospatial features with algorithms like HDBSCAN can enhance the detection of meaningful location based patterns Finally deploying the models in a real time fare prediction system with interactive dashboards could offer valuable tools for both drivers and passengers increasing operational efficiency and customer satisfaction.

Overall Conclusion

This project successfully explored and evaluated a variety of machine learning and deep learning models to predict taxi fare amounts and classify fare categories using the NYC Yellow Taxi 2019 dataset among regression models linear regression and ensemble methods like Random Forest and Gradient Boosting demonstrated strong predictive accuracy with the deep learning model achieving the lowest MAE and classification models that particularly logistic regression and Random Forest has achieved high accuracy and F1 scores and effectively distinguishing

between low and high fare classes despite class imbalance Clustering analysis using K Means and DBSCAN revealed meaningful patterns and outliers by aiding in understanding fare groupings and passenger behaviors. Overall these models has performed best on short trips and lower fare ranges with fare amount identified as the most critical feature, and residual analysis highlighting heteroscedasticity in the data these findings provide valuable insights into fare prediction and passenger trends with practical implications for pricing strategies and service optimization.

1. Zhang, Y., Wang, J., & Li, X. (2020). Predictive analytics for urban taxi demand: A review. *Journal of Urban Computing*, *8*(2), 101–118. https://doi.org/10.1016/j.juc.2020.03.005
2. Liu, H., & Chen, W. (2021). Dynamic pricing and fare prediction in ride-hailing services using machine learning. *Transportation Research Part C: Emerging Technologies*, *129*, 103247. https://doi.org/10.1016/j.trc.2021.103247
3. New York City Taxi & Limousine Commission. (2019). *Yellow taxi trip data*. Retrieved from https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
4. Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324
5. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018
6. Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
7. Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). https://doi.org/10.1145/2939672.2939785
8. Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining* (pp. 226–231). AAAI Press.
9. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *International Conference on Learning Representations* (ICLR). https://arxiv.org/abs/1412.6980
10. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine*

Learning Research, *12*, 2825–2830. Retrieved from http://jmlr.org/papers/v12/pedregosa11a.html