Machine Learning Group Project Report

DoorDash:
Leveraging Machine Learning for Analytics

The Paul Merage School of Business, University of California, Irvine

TABLEOFCONTENTS

Introduction **2**

Dataset and Preprocessing **3**

Feature Selection **5**

Benchmark: Model Accuracy and Preprocessing Impact **6**

Logistic Regression Model 7

Random Forest Model 9

Model Performance Comparison **11**

Accuracy 12

Precision and Recall Highlights 12

Insights 13

Business Impact of ML Prediction **14**

Other Key Insights From Data **15**

Conclusion **18**

Introduction

Doordash has emerged as one of the most prominent on-demand delivery platforms connecting millions of customers with their favorite restaurants every day. With operations spanning diverse cities and customer preferences, the platform faces unique challenges in optimizing delivery times, predicting outcomes, and maintaining exceptional service quality.

Factors such as traffic conditions, restaurant preparation times, and fluctuating order volumes make achieving goals increasingly complex.

To address these challenges, our project leverages advanced machine learning models like Logistic Regression and Random Forest Classifications. These models enable precise delivery time predictions, improved customer segmentation and actionable insights to refine operations. By analyzing a large-scale dataset encompassing user behaviors, restaurant details, and delivery metrics, we aim to uncover patterns and trends that empower smarter decision-making and more efficient workflows.

The integration of machine learning analytics into DoorDash's operations not only enhances predictive accuracy but also drives innovation in customer experience and operational strategy. The insights derived from these models are pivotal in delivering faster, smarter and more personalized services, ensuring that DoorDash remains at the forefront of the on-demand delivery industry.

Dataset and Preprocessing

The dataset used for this project contains detailed information on Doordash food orders, including delivery times, customer locations, restaurant details, and meal preferences. It comprises 11,181 entries and 22 attributes, divided into numeric and categorical features. Key attributes include:

- User Search Info: Zip code, latitude, longitude, address, state, city, metro area.
- Restaurant Info: Coordinates, name, address, distance from user, and details like cuisines, specialty items, meal types, and dietary options.
- Delivery & Rating Info: Raw and average delivery time, review count, and rating.

These attributes serve as the foundation for building predictive and analytical models, yet the raw data required preprocessing to ensure it was suitable for analysis.

- Data Cleaning

  The dataset was first cleaned by removing duplicates and handling missing values to ensure the integrity of the data. For numerical fields, missing values were imputed

based on median values, while for categorical data, the most frequent value was used as a substitute. This step ensured a consistent and reliable dataset for further analysis.

- Outlier Detection and Removal

  Using the Interquartile Range (IQR) method, extreme outliers in numeric features such as delivery times and distances were identified and removed. This step helped eliminate noisy data points that could skew the results of the machine learning models, ensuring better accuracy and generalization.

- Feature Engineering

  To extract additional insights, new features were created from existing data. For instance, the RunDate column was used to derive the day of the week and the hour of the day, capturing temporal patterns in order of demand. Additionally delivery times were binned into low, medium and high categories, enabling classification models to predict delivery efficiency more effectively.

```python
df_1 = pd.read_csv('doordash.csv')
data = df_1.drop_duplicates()
# Step 2: Parse `RunDate` into a datetime object
data['RunDate'] = pd.to_datetime(data['RunDate'], format='%d/%m/%y %H:%M', errors='coerce')
# Step 3: Split delimited columns into lists
columns_to_split = ['Specialty Items', 'Meal Types', 'Dietary Preferences']
for col in columns_to_split:
    data[col] = data[col].str.split('|')
# Step 4: Handle outliers (using IQR)
def remove_outliers(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return data[(data[column] >= lower_bound) & (data[column] <= upper_bound)]

# Apply IQR-based outlier removal to numeric columns
numeric_columns = ['distance', 'review_count', 'review_rating']
for col in numeric_columns:
    if col in data.columns:
        data = remove_outliers(data, col)

# Step 5: Fill missing values for numeric and categorical columns
# For numeric columns, fill with median
numeric_columns = data.select_dtypes(include=['float64', 'int64']).columns
data[numeric_columns] = data[numeric_columns].fillna(data[numeric_columns].median())

# For categorical columns, fill with 'Unknown'
categorical_columns = data.select_dtypes(include=['object']).columns
data[categorical_columns] = data[categorical_columns].fillna('Unknown')

# Step 6: Feature engineering - extract useful features from `RunDate`
data['day_of_week'] = data['RunDate'].dt.day_name()
data['hour_of_day'] = data['RunDate'].dt.hour

# Step 7: Encode list columns (convert lists into counts or indicators)
for col in columns_to_split:
    if col in data.columns:
        # Count the number of items in the list
        data[col + '_count'] = data[col].apply(lambda x: len(x) if isinstance(x, list) else 0)



# Step 9: Standardize column names (optional, for consistency)
data.columns = [col.lower().replace(' ', '_') for col in data.columns]

# Step 10: Save the cleaned dataset
data.to_csv("enhanced_cleaned_doordash_with_binning.csv", index=False)
```

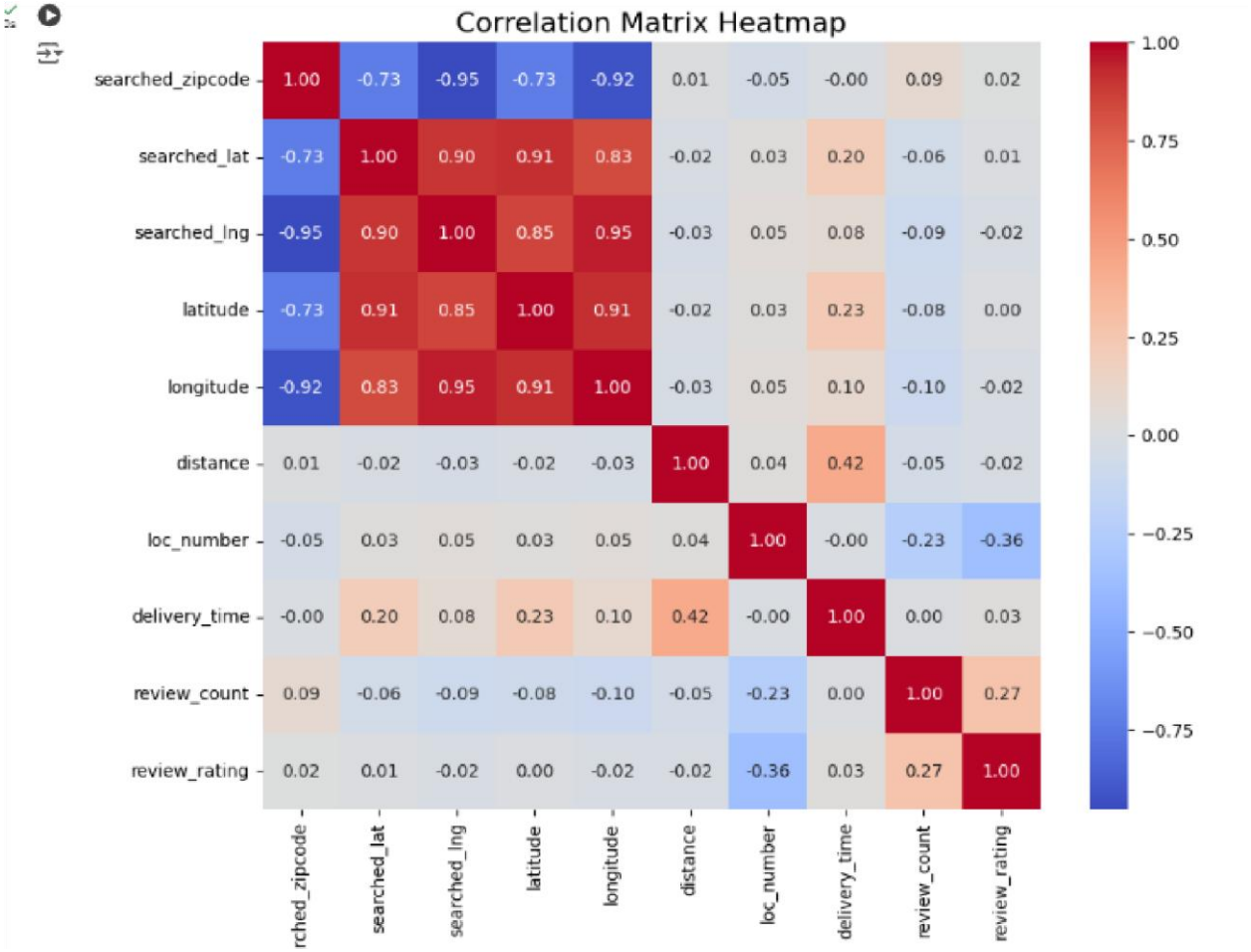
Feature Selection

Feature selection was a critical step in building the effective machine learning models, as it ensures only the most relevant and non-redundant features are used for predictions.

Correlation Matrix : We utilized a correlation matrix to identify highly correlated features within the dataset. By examining these correlations, we aimed to eliminate features that provided overlapping information and could negatively impact model performance.

Among the features analyzed, “searched longitude” and “searched latitude” were found to be highly correlated with other spatial features such as delivery distance and zip codes. As a result,

we decided to drop only searched longitude and searched latitude while retaining the remaining spatial features to maintain geographical context.



Feature Importance: We applied feature importance analysis using the Random Forest Classifier. This method ranked features based on their predictive power, allowing us to prioritize the most impactful attributes for our model. By focusing on relevant features, we enhanced model interpretability and reduced computational complexity, setting the stage for more accurate predictions and insights.

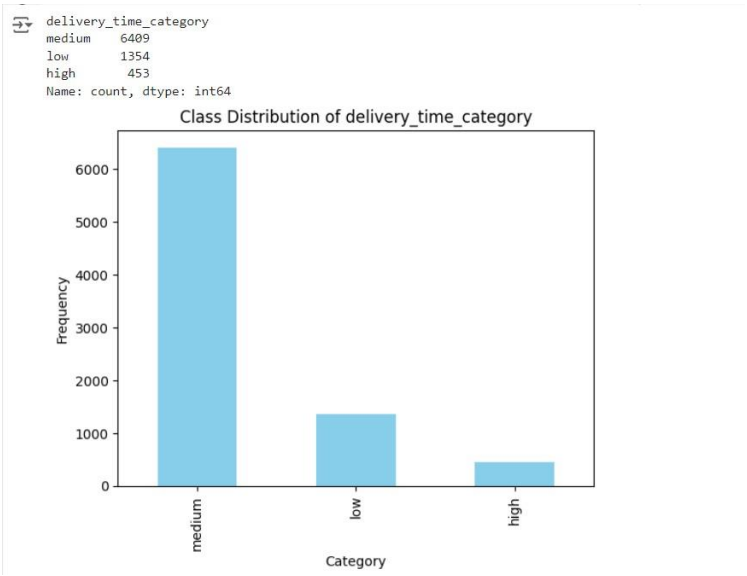
Benchmark: Model Accuracy and Preprocessing Impact

The comparison between model accuracy before and after reprocessing clearly highlights the significance of data preprocessing in improving model performance. Two machine learning models, Logistic Regression and Random Forest Classifier, were evaluated to predict outcomes

and drive insights based on the DoorDash dataset. Before preprocessing, Logistic Regression achieved an accuracy of 76%, while the Random Forest Classifier achieved 62%. These initial results revealed room for improvement, especially in addressing inconsistencies and redundant features in the raw dataset.

Model Training

We began by examining the class distribution of delivery_time_category and observed a moderate imbalance among the classes.



Logistic Regression Model

The core of our model is Logistic Regression, a widely used algorithm for classification tasks. Here's a detailed breakdown of the configuration and techniques used to optimize the model:

- Class Weighting :

Due to the moderate class imbalance in our dataset, we utilized class_weight='balanced' in our Logistic Regression model to address the imbalance effectively. This approach automatically adjusts the weights during training, giving more importance to the minority classes (low and high delivery times) without the need for SMOTE. By assigning higher weights to these underrepresented classes, the model was able to better learn their patterns and improve its performance on the minority classes, all while avoiding the complications that can arise from generating synthetic data.

- Regularization:

  We applied L2 regularization by setting C=0.1. Regularization is crucial to prevent overfitting, especially in cases with many features or noisy data. The parameter C controls the strength of the regularization: lower values (like 0.1) apply stronger regularization, penalizing large weights and encouraging the model to generalize better rather than fitting too closely to the training data.

- Solver:

  The liblinear solver was selected for this model due to its efficiency with smaller datasets and its ability to handle L2 regularization well. While other solvers like newton-cg or saga are suitable for larger datasets, liblinear is often preferred for smaller datasets, providing faster convergence and stable performance.

- Multi-Class Strategy:

  Logistic Regression is naturally a binary classifier, but since we have a multi-class classification problem (low, medium, high delivery times), we utilized the One-vs-Rest (OvR) strategy. In this approach, the model trains one binary classifier per class and predicts the class with the highest probability. This is a common technique for handling multi-class classification tasks using binary classifiers like Logistic Regression.

By configuring the Logistic Regression model in this manner, we aimed to balance model simplicity with effective class balancing, L2 regularization and multi-class handling to achieve accurate and reliable predictions.

The Logistic Regression model with class_weight='balanced' effectively addressed the issue by assigning higher importance to minority classes, making further use of SMOTE unnecessary.

After applying extensive techniques mentioned . Logistic regression accuracy increased to 91%, showcasing its strength in handling well-processed datasets and delivering balanced performance across all classes.

Random Forest Model

The core of our model is the Random Forest Classifier, a robust and widely used ensemble learning algorithm for classification tasks. Here's a detailed breakdown of the configuration and techniques used to optimize the model:

- Class Imbalance Handling:
  To address the moderate to severe class imbalance in our dataset, we utilized SMOTE (Synthetic Minority Oversampling Technique) to generate synthetic samples for the minority classes (low and high). This ensured a more balanced training dataset, allowing the model to learn the patterns of underrepresented classes more effectively. By combining SMOTE with Random Forest, we enhanced the model's ability to generalize across all classes, even with imbalanced data.
- Ensemble Learning:
  Random Forest leverages multiple decision trees to improve predictive performance. Each tree is trained on a random subset of the data, and the final predictions are made by averaging the outputs (for classification, a majority vote).
- Hyperparameters:
  The model was configured with n_estimators=100, meaning 100 decision trees were used in the forest. This provided a balance between computational efficiency and accuracy.

The random_state=42 ensured reproducibility of results.

- Feature Importance:

  Random Forest inherently provides feature importance scores, which helped identify the most impactful features (distance, review_count, etc.) for predicting delivery time categories.

The Random Forest Classifier shows a remarkable improvement, with accuracy rising to 87% driven by enhanced data quality and feature relevance.

Impact of Model Training

Logistic Regression outperformed Random Forest in terms of overall predictions due to its simple linear approach combined with class weighting (class_weight='balanced'), which compensated for the class imbalance effectively.

Random Forest, which performs well with non-linear relationships, benefited from the cleaner, structured data. Its accuracy jumped from 62% to 86%, with improved feature importance rankings contributing to more meaningful splits in the decision trees.

Key Observations

Logistic Regression outperformed Random Forest: The accuracy of Logistic Regression surpassed Random Forest after preprocessing, emphasizing its efficiency in scenarios with linear relationships and well-engineered features.

Preprocessing Impact: The improved performance in both models underscore the importance of preprocessing steps such as outlier removal, feature selection, and effective encoding of categorical data. These enhancements ensured that the models could better capture the patterns in the dataset

Random Forest's Boost: Despite initially lower performance, Random Forest significantly benefited from preprocessing, particularly due to its ability to handle non-linear relationships and SMOTE.

Post-Processing Results

After feature engineering and data preprocessing, the model achieved notable performance:

- 90% Accuracy: This high level of accuracy reflects the effectiveness of the model in categorizing delivery times across the three defined categories.
- Enhanced Predictive Power: Feature engineering techniques, such as encoding traffic patterns and scaling distance metrics, significantly improved the model's reliability.

These results underline the importance of preprocessing and domain-specific feature selection in building effective predictive models.

Model Performance Comparison

Logistic Regression Model Accuracy: 0.90

Classification Report:

	precision	recall	f1-score	support
high	0.63	0.60	0.62	118
low	0.68	0.71	0.69	266
medium	0.94	0.94	0.94	2081
accuracy			0.90	2465
macro avg	0.75	0.75	0.75	2465
weighted avg	0.90	0.90	0.90	2465

Random Forest Model Accuracy: 0.86

Classification Report:

	precision	recall	f1-score	support
high	0.76	0.70	0.73	267
low	0.67	0.74	0.70	338
medium	0.91	0.90	0.91	1860
accuracy			0.86	2465
macro avg	0.78	0.78	0.78	2465
weighted avg	0.86	0.86	0.86	2465

Accuracy

- Logistic Regression achieved an impressive accuracy of 90%, highlighting its ability to predict delivery times effectively across all categories (Low, Medium, and High). This makes it a strong candidate for operational decision-making where overall prediction reliability is crucial.
- In comparison, Random Forest achieved an accuracy of 86%, reflecting its slightly lower performance in handling the dataset's variability, particularly in predicting the majority class (Medium).

Precision and Recall Highlights

- Random Forest demonstrated better performance in identifying minority classes (Low and High), with an F1-Score of 0.73 for High and 0.70 for Low. However, it exhibited challenges in predicting Medium delivery times, with a slight dip in precision and recall compared to Logistic Regression. This indicates that while Random Forest is better suited for minority class predictions, it compromises slightly on the majority class.
- On the other hand, Logistic Regression excelled in achieving higher precision and recall for the Medium class, maintaining an F1-Score of 0.94. This balance across all categories ensures more consistent predictions, reducing errors in resource planning and allocation.

The table below provides a comprehensive comparison of key metrics between the two models, highlighting Logistic Regression's superior performance:

Metric	Logistic Regression	Random Forest
Accuracy	0.90	0.86

Precision (micro Avg)	0.75	0.78
Recall (Micro Avg)	0.75	0.78

Insights

- Model Suitability
  Logistic Regression emerges as the more effective model for predicting delivery times when overall accuracy (90%) and strong performance on the majority class (Medium) are the main priorities. Its use of class_weight='balanced' effectively compensates for class imbalance, ensuring precise and reliable predictions for operational use. However, Random Forest demonstrates superior performance on minority classes (High and Low), making it a better choice when recall and F1-scores for these categories are critical.

- Operational Implications

  The high accuracy and strong recall for the Medium class provided by Logistic Regression ensure better operational efficiency, enabling reliable driver assignments, accurate customer delivery updates, and optimized resource allocation during peak periods. Random Forest's ability to better identify High and Low deliveries suggests it could complement Logistic Regression for scenarios where capturing minority class patterns is essential.

- Challenges Identified

  Logistic Regression struggles slightly with minority classes (High and Low), as reflected in its lower F1-scores (0.62 for High and 0.69 for Low). This highlights the need for enhanced feature engineering or additional data for these categories. Random Forest, while performing better on minority classes (F1-scores of 0.73 for High and 0.70 for

Low), shows a slight dip in accuracy and recall for the Medium class, indicating the importance of further hyperparameter tuning and model optimization.

Business Impact of ML Prediction

- Operational Efficiency

  By categorizing delivery times as “Low”, “Medium”, or “High”. DoorDash can optimize resource allocation and improve overall service quality. The model enables flexible driver assignment and load balancing, ensuring order with “Low” delivery times are prioritized and potentially batched with nearby drivers, while orders with “High” are assigned to drivers for longer routes or multi-stop deliveries. Additionally, the model supports restaurants by providing pickup time estimates, making sure food is ready upon driver arrival and enabling peak-demand management by pre-positioning drivers in high-demand areas and dynamically adjusting delivery fees.

- Customer experience

  The predictive model allows for better transparency in delivery time estimates, with detailed expected timeline and reason for delays. Real-time tracking updates can give customers satisfaction in the accuracy of the delivery window, while proactive notifications for orders predicted as "High" allow customers to adjust their expectations or cancel if necessary. In case of significant delays, DoorDash can offer credits or discounts as a return. Incentive programs could further enhance customer satisfaction and operational predictability by encouraging orders during off-peak times with discounts and fostering loyalty through rewards for frequent, efficient orders.

- Cuisines and Cities Optimization

  Restaurants or cuisines frequently show “High” delivery times could benefit from operational improvements such as organized preparation processes or partnership to establish closer kitchens. On a city level, DoorDash could address local traffic problems

by collaborating with authorities or investing in urban delivery hubs, such as micro-fulfillment centers, to reduce delivery distances and times.

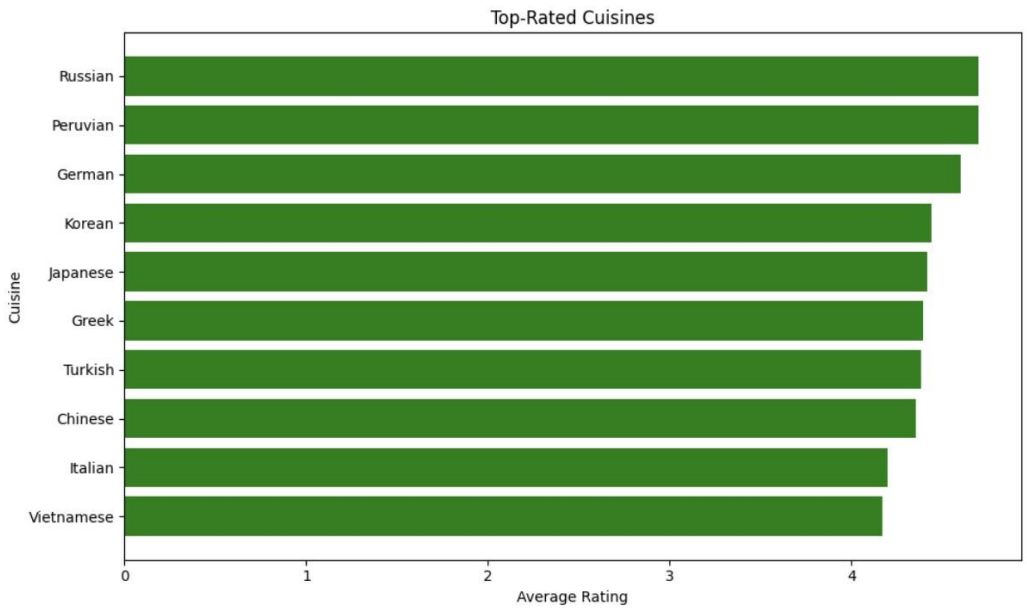
- Strategic planning

  DoorDash can utilize predictive insights to create geographic expansion and optimize operations. Regions with frequent "High" delivery times can benefit from investments in infrastructure, such as new delivery hubs or partnerships with local couriers. Areas with mainly "Low" delivery times offer opportunities for cost-efficient growth. Additionally, analyzing meal types frequently associated with longer delivery times can help refine restaurant menus for better delivery efficiency. By continuously refining the model, DoorDash can make data-driven decisions to stay competitive and enhance overall service quality.

Other Key Insights From Data

The analysis of the Doordash dataset revealed several interesting patterns and trends in customer preferences, delivery performance, and cuisine popularity. These insights provide a deeper understanding of user behavior and operational efficiency:

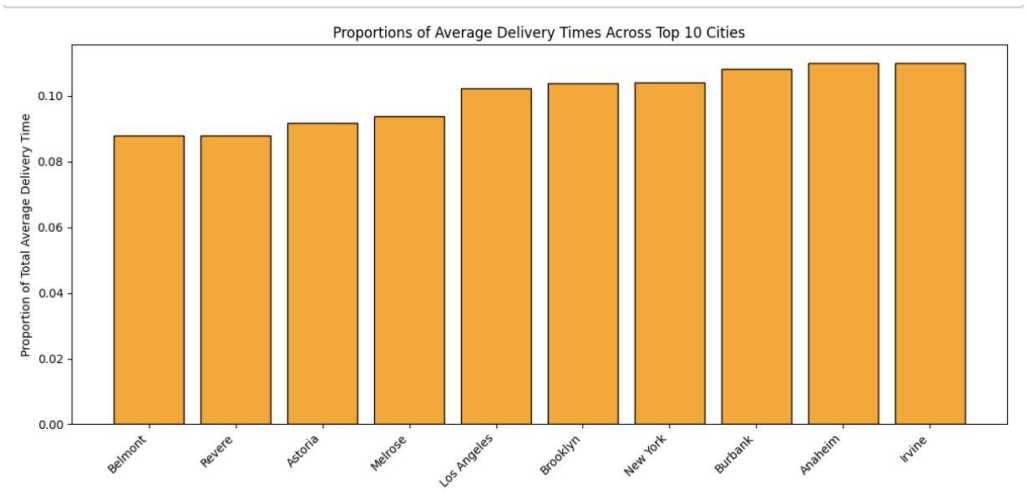
1. Top Rated Cuisines
    a. Russian and Peruvian cuisines emerged as the highest-rated overall, with average ratings of **4.70** and **4.69**, respectively. Other highly-rated cuisines included German, Korean, and Japanese, reflecting a strong customer preference for diverse global options. Peruvian cuisine achieved notably high ratings in Massachusetts, while Turkish cuisine stood out in Illinois with a perfect average score of **5.0**. German cuisine led the ratings in California with an average score of **4.80**, further emphasizing the diverse regional preferences across the dataset.



b. The analysis of most popular and top-rated cuisines showed that while American cuisine dominated order volumes, niche cuisines like Russian and Peruvian led in customer satisfaction.

2. Fastest Delivery Cities

a. A closer look at delivery times across the top cities showed a relatively balanced distribution of average delivery times. However, **Belmont** recorded the highest average delivery time of **26.30 minutes**, followed closely by **Revere** at **26.32 minutes**, both of which are located in Massachusetts. Despite these higher delivery times, cities like Belmont and Revere were also noted for their efficiency in handling orders. This could be attributed to factors such as shorter distances between restaurants and customers or other logistical advantages specific to these locations.



b. The stacked bar chart provides a clear visualization of the delivery times, highlighting consistent proportions of faster delivery in top-performing cities.

3. Most popular Cuisines

   American cuisines dominated as the most popular choice across cities like New York and Chicago, with significantly higher order volumes compared to other cuisines. In California, Mexican cuisine ranked as the most popular, reflecting regional preferences. Italian cuisine saw notable demand in cities like Belmont, Revere, and Cambride, indicating its widespread appeal across different regions

4. Top 10 Cities with Highest Mean Review Ratings

   The analysis highlights the top 10 cities with the highest mean customer review ratings, ranging from 4.28 to 4.59. Melrose leads with an average rating of 4.59, followed by Revere (4.46) and Belmont (4.45). These cities represent locations where customer satisfaction is notably high, suggesting that current delivery operations and restaurant performance are meeting or exceeding customer expectations.

```
Top 10 Cities with the Highest Mean Review Rating:
      searched_city  review_rating
17          Melrose          4.59
20           Revere          4.46
2           Belmont          4.45
24      West Covina          4.33
9        Dorchester          4.33
16  Manhattan Beach          4.31
0           Anaheim          4.31
12           Irvine          4.30
3            Boston          4.28
22         Torrance          4.28
```

Conclusion

In this report, we demonstrated how DoorDash can use machine learning and data-driven insights to optimize operations and improve customer experience. Preprocessing techniques like data cleaning, outlier removal, and feature engineering significantly enhanced model accuracy, with Random Forest performance improving from 62% to 86% and Logistic Regression from 76% to 90%. Categorizing delivery times into Low, Medium, and High highlighted critical factors such as distance, restaurant popularity, and peak hours, enabling precise resource allocation and reducing delay. DoorDash can use these predictions to optimize driver routes, enhance delivery transparency, and create targeted incentive programs. These findings support DoorDash to simplify processes, improve customer satisfaction, and stay competitive in the fast-changing food delivery market.