Wow Project Report: Team 100

Member: Yuanzheng Yin, Guodong Zhu

Introduction

In the realm of the bustling New York City taxi industry, where millions of trips are completed daily, maximizing earnings for taxicab drivers is not just about navigating through traffic but also understanding passenger behaviors and preferences. This project endeavors to tackle this challenge head-on by developing a predictive model aimed at boosting taxi drivers' tip earnings.

The primary objective of this project is to construct a robust predictive model leveraging historical data from the New York City Taxi and Limousine Commission (TLC). By analyzing trip attributes and customer behavior, we aim to predict which passengers are more likely to be generous tippers. This predictive model holds the potential to revolutionize the way taxi drivers prioritize routes and passengers, ultimately maximizing their earnings.

At the heart of this endeavor lies a dataset sourced from over 200,000 licensees, representing a vast network facilitating approximately one million trips daily. However, it's essential to acknowledge that while this dataset serves educational purposes, it may not entirely reflect the nuanced behaviors of taxicab riders in New York City. Nonetheless, it provides a rich foundation upon which we can build and refine our predictive model.

The dataset includes various features such as trip distances, pickup and drop-off locations, payment types, fare amounts, and most importantly, tip amounts. These features serve as valuable insights into understanding the dynamics between trip attributes and tipping behaviors.

Throughout this report, we will delve into the process of data cleaning, exploratory data analysis, and the application of machine learning techniques to construct a predictive model. By harnessing the power of data-driven insights, we aim to empower taxicab drivers with real-time guidance on route prioritization and service adaptation strategies, ultimately enhancing their earning potential.

In essence, this project not only seeks to optimize individual taxi drivers' earnings but also aims to contribute to the broader ecosystem of the taxi industry by providing actionable insights for taxicab companies to optimize driver incentives and reward programs. Through collaborative efforts and innovative solutions, we strive to unlock new avenues for revenue generation and service enhancement in the dynamic landscape of urban transportation.

Features

The dataset comprises several key variables providing insights into taxi trips within New York City. These include the VendorID, denoting the TPEP provider (1 for Creative Mobile Technologies, LLC and 2 for VeriFone Inc.), and timestamps for trip

initiation and conclusion, captured by TPEP_pickup_datetime and TPEP_dropoff_datetime, respectively. Passenger_count quantifies the number of passengers in the vehicle, while Trip_distance measures the elapsed distance in miles. PULocationID and DOLocationID categorize the pickup and dropoff TLC Taxi Zones. RateCodeID delineates the final fare code applicable at trip termination, encompassing standard rates, airport fares (JFK, Newark), negotiated fares, and group rides. Store_and_fwd_flag indicates if trip records were stored in the vehicle before transmission. Payment_type signifies the method of payment (credit card, cash, etc.). Fare_amount represents the meter-calculated fare, whereas Extra encompasses miscellaneous surcharges. MTA_tax and Improvement_surcharge detail additional levies. Tip_amount, automatically logged for credit card transactions, and Tolls_amount, summing toll payments, offer insights into passenger generosity and route selection. Finally, Total_amount aggregates charges excluding cash tips. These variables collectively form a comprehensive dataset, illuminating diverse facets of taxi trips crucial for predictive modeling and optimization strategies.

Data Overview

The dataset consists of 22,699 entries and 18 columns. Each row represents a taxi trip in New York City, and each column provides specific information about the trip. The columns include attributes such as the trip's pickup and dropoff timestamps, passenger count, trip distance, fare details, payment type, and location identifiers.
The dataset contains no missing values or duplicates, ensuring data integrity and completeness. Descriptive statistics reveal valuable insights into the distribution and characteristics of numeric variables. For instance, the mean trip distance is approximately 2.91 miles, with a standard deviation of 3.65 miles, suggesting variability in trip lengths. Similarly, the mean fare amount is $13.03, with a standard deviation of $13.24, indicating variations in fare charges.

```
Unnamed: 0              0
VendorID                0
tpep_pickup_datetime    0
tpep_dropoff_datetime   0
passenger_count         0
trip_distance           0
RatecodeID              0
store_and_fwd_flag      0
PULocationID            0
DOLocationID            0
payment_type            0
fare_amount             0
extra                   0
mta_tax                 0
tip_amount              0
tolls_amount            0
improvement_surcharge   0
total_amount            0
dtype: int64
```

```
df.shape
```

```
(22699, 18)
```

```
df.nunique()
```

```
Unnamed: 0              22699
VendorID                    2
tpep_pickup_datetime    22687
tpep_dropoff_datetime   22688
passenger_count             7
trip_distance            1545
RatecodeID                  6
store_and_fwd_flag          2
PULocationID              152
DOLocationID              216
payment_type                4
fare_amount               185
extra                       6
mta_tax                     3
tip_amount                742
tolls_amount               38
improvement_surcharge       3
total_amount             1369
dtype: int64
```

Categorical variables such as VendorID, RatecodeID, store_and_fwd_flag, and payment_type provide additional context about the taxi trips, such as the service provider, rate code, whether the trip record was stored in the vehicle memory, and the payment method used.

Furthermore, the data types have been appropriately assigned to each column, with numeric variables represented as either integers or floats and categorical variables identified as objects.
During the exploratory data analysis (EDA), potential issues regarding outliers and high correlation were observed within the dataset.

	Data Type	Missing Values%	Unique Values%	Minimum Value	Maximum Value	DQ Issue
Unnamed: 0	int64	0.000000	100	65654.000000	113456738.000000	Possible ID column: drop before modeling step.
VendorID	int64	0.000000	0	1.000000	2.000000	No issue
tpep_pickup_datetime	object	0.000000	100			Possible ID column: drop before modeling step.
tpep_dropoff_datetime	object	0.000000	100			Possible ID column: drop before modeling step.
passenger_count	int64	0.000000	0	1.000000	6.000000	Column has 204 outliers greater than upper bound (3.50) or lower than lower bound(-0.50). Cap them or remove them.
trip_distance	float64	0.000000	NA	0.000000	33.960000	Column has 224 outliers greater than upper bound (6.25) or lower than lower bound(-2.15). Cap them or remove them.
RatecodeID	int64	0.000000	0	1.000000	5.000000	Column has 53 outliers greater than upper bound (1.00) or lower than lower bound(1.00). Cap them or remove them.
store_and_fwd_flag	object	0.000000	0			No issue
PULocationID	int64	0.000000	4	4.000000	264.000000	No issue
DOLocationID	int64	0.000000	6	1.000000	265.000000	No issue
payment_type	int64	0.000000	0	1.000000	4.000000	Column has 5 outliers greater than upper bound (3.50) or lower than lower bound(-0.50). Cap them or remove them.
fare_amount	float64	0.000000	NA	-120.000000	150.000000	Column has 180 outliers greater than upper bound (26.50) or lower than lower bound(-5.50). Cap them or remove them., Column has a high correlation with ['trip_distance']. Consider dropping one of them.
extra	float64	0.000000	NA	0.000000	4.500000	Column has 11 outliers greater than upper bound (1.25) or lower than lower bound(-0.75). Cap them or remove them.
mta_tax	float64	0.000000	0	0.000000	0.500000	No issue
tip_amount	float64	0.000000	NA	0.000000	21.300000	Column has 105 outliers greater than upper bound (6.03) or lower than lower bound(-3.62). Cap them or remove them.
tolls_amount	float64	0.000000	NA	0.000000	16.500000	Column has 91 outliers greater than upper bound (0.00) or lower than lower bound(0.00). Cap them or remove them.
improvement_surcharge	float64	0.000000	0	-0.300000	0.300000	No issue
total_amount	float64	0.000000	NA	-120.300000	150.300000	Column has 187 outliers greater than upper bound (31.36) or lower than lower bound(-4.80). Cap them or remove them., Column has a high correlation with ['trip_distance', 'fare_amount']. Consider dropping one of them.

Outliers, which are data points significantly deviating from the majority of the dataset, were identified in several numeric columns such as `passenger_count`, `trip_distance`, `RatecodeID`, `payment_type`, `fare_amount`, `extra`, `tip_amount`, `tolls_amount`, and `total_amount`. For instance, there were 204 outliers in the `passenger_count` column, indicating values greater than the upper bound (3.50) or lower than the lower bound (-0.50). Similarly, the `trip_distance` column had 224 outliers beyond the upper bound (6.25) or lower than the lower bound (-2.15). These outliers could potentially skew statistical analyses or machine learning models if left unaddressed.
Furthermore, certain columns exhibited a high degree of correlation with each other, indicating redundancy or multicollinearity within the dataset. For example, the `fare_amount` column showed a strong correlation with the `trip_distance` column, suggesting that the fare charged for a taxi trip is closely related to the distance traveled. Similarly, the `total_amount` column exhibited a high correlation with both `trip_distance` and `fare_amount`, indicating that the total amount charged to passengers is influenced by both trip distance and fare.
Addressing these issues is crucial to ensure the robustness and accuracy of subsequent analyses and machine learning models. Strategies for dealing with outliers may include capping or removing extreme values, while addressing high correlation may involve dropping one of the correlated variables to reduce redundancy in the dataset. Additionally, further investigation and domain knowledge may be necessary to determine the most appropriate course of action for handling these issues effectively.

Data Pre-Processing

The objective of building a machine learning model is to predict customers who are likely to be particularly generous in tipping, defined as those who will tip 20% or more. This modified objective aims to assist taxi drivers in increasing their earnings from tips while avoiding the ethical implications and potential problems associated with

predicting customers who won't tip at all.
To achieve this objective, the model requires features such as behavioral history for each customer, including their tipping patterns on previous taxi rides, as well as information on times, dates, and locations of pickups and drop-offs, estimated fares, and payment methods.
The target variable for the model is a binary variable (1 or 0) indicating whether the customer is expected to tip ≥ 20%. This transforms the task into a supervised learning classification problem.
In order to create the target variable indicating tip percent, we engineered a new column named "tip_percent" in the dataframe. This was achieved by calculating the tip percentage for each trip using the formula:

$$tippercentage = \frac{tip\ amount}{total - tip\ amount}$$

A new column named 'generousornot' was created in the dataframe 'df1' to serve as the target variable. This column was initially populated with the values from the 'tip_percentage' column. Then, a binary classification was performed to label customers as generous or not generous based on whether their tip percentage was greater than or equal to 0.2 (indicating a tip of 20% or more). The resulting boolean values were then converted to integers (0 or 1) to represent the target classes, where 1 indicates a generous tipper and 0 indicates otherwise.

```
# Define 'rushinmorning()' conversion function [06:00 - 10:00)
def rushinmorning(hour):
        if 6 <= hour['rushinmorning'] < 10:
                val = 1
        else:
                val = 0
        return val
# Apply 'rushinmorning' function to the 'rushinmorning' series
df1['rushinmorning'] = df1.apply(rushinmorning, axis=1)
df1['rushinmorning'].head()
```

```
0    1
1    0
2    1
3    0
5    0
Name: rushinmorning, dtype: int64
```

The dataframe's 'tpep_pickup_datetime' and 'tpep_dropoff_datetime' columns were transformed into datetime format using the provided format string. Following this, several new columns were introduced to capture different time periods throughout the day.
To begin, a 'day' column was created to denote the day of the week when the pickup took place, derived from the 'tpep_pickup_datetime' column.
Subsequently, four additional columns were generated to categorize pickups into distinct time intervals:
- 'rushinmorning': Identifies whether the pickup occurred during the morning rush hours (06:00–10:00).
- 'morning': Indicates if the pickup transpired during regular morning hours (10:00–16:00).

- 'rushinevening': Highlights pickups made during the evening rush hours (16:00–20:00).
- 'night': Specifies pickups made during the nighttime hours (20:00–06:00).

These columns were populated with binary values, where 1 signifies true and 0 indicates false, based on the corresponding time periods using defined conversion functions. These functions evaluated the hour component of the pickup time and assigned the binary value accordingly.

```
# Drop columns
drop_cols = ['Unnamed: 0', 'tpep_pickup_datetime', 'tpep_dropoff_datetime',
             'payment_type', 'trip_distance', 'store_and_fwd_flag', 'payment_type',
             'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount',
             'improvement_surcharge', 'total_amount', 'tip_percentage']

df1 = df1.drop(drop_cols, axis=1)
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15265 entries, 0 to 22698
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   VendorID         15265 non-null  int64
 1   passenger_count  15265 non-null  int64
 2   RatecodeID       15265 non-null  int64
 3   PULocationID     15265 non-null  int64
 4   DOLocationID     15265 non-null  int64
 5   generousornot    15265 non-null  int64
 6   day              15265 non-null  object
 7   rushinmorning    15265 non-null  int64
 8   morning          15265 non-null  int64
 9   rushinevening    15265 non-null  int64
 10  night            15265 non-null  int64
 11  month            15265 non-null  object
dtypes: int64(10), object(2)
memory usage: 1.5+ MB
```

The next step is dropping redundant and irrelevant columns as well as those that would not be available when the model is deployed. This includes information like payment type, trip distance, tip amount, tip percentage, total amount, toll amount, etc. The target variable (generous) must remain in the data because it will get isolated as the y data for modeling.

```
# 1. Define list of cols to convert to string
cols_to_str = ['RatecodeID', 'PULocationID', 'DOLocationID', 'VendorID']

# 2. Convert each column to string
for col in cols_to_str:
        df1[col] = df1[col].astype('str')
# Convert categoricals to binary
df2 = pd.get_dummies(df1, drop_first=True)
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15265 entries, 0 to 22698
Columns: 344 entries, passenger_count to month_sep
dtypes: bool(338), int64(6)
memory usage: 5.7 MB
```

To prepare the categorical variables for encoding, we convert several numeric columns that encode categorical information into type 'str'. This includes columns like 'RatecodeID' and the pickup and dropoff locations. By converting these columns to string type, I'll ensure that they are recognized as categorical variables by the 'get_dummies()' function during encoding. This step is crucial for accurate representation of categorical data in the model.

```
generousornot
1    0.526368
0    0.473632
Name: proportion, dtype: float64
```

The dataset indicates that slightly over half of the customers were classified as

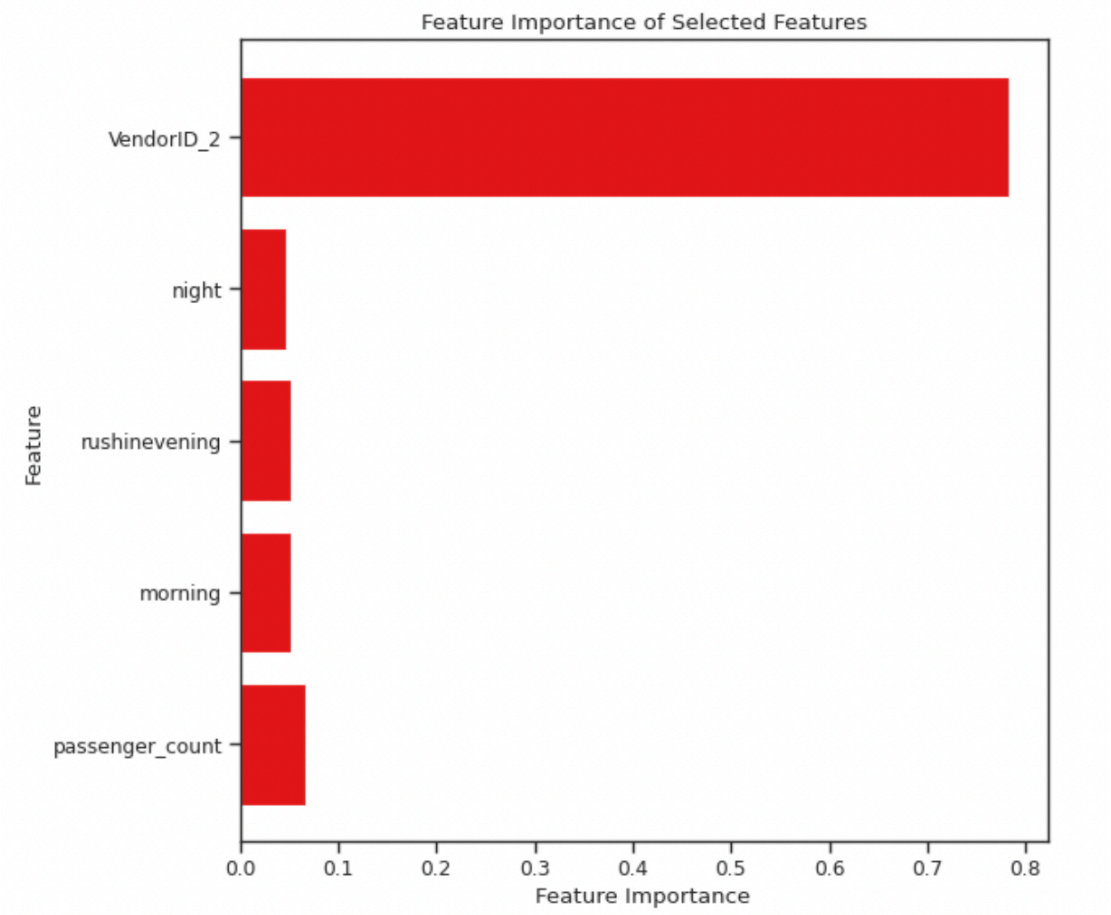
"generous" (tipping ≥ 20%), with approximately 52.64% falling into this category. This suggests a nearly balanced dataset in terms of the target variable.
To determine an appropriate evaluation metric for the model, it's crucial to consider the implications of both false positives and false negatives:
- False positives: These occur when the model predicts a tip of 20% or more, but the customer does not actually give one. This scenario is unfavorable for cab drivers as they might expect a good tip but end up disappointing, impacting driver satisfaction.
- False negatives: These happen when the model predicts a tip of less than 20%, but the customer ends up giving a more generous tip. This scenario is detrimental to customers as the model might cause cab drivers to prioritize other passengers who were predicted to tip more, potentially leading to customer dissatisfaction.
Given the relatively even stakes and the desire to balance the interests of both drivers and customers, a metric that weighs both precision and recall equally is sought after. The F1 score, which considers both true positives and false positives, thereby balancing precision and recall, is a suitable choice for this purpose.

```
# check skewness of quants
df2[['morning', 'rushinevening', 'night','generousornot','rushinmorning']].skew()
```

```
morning          0.975067
rushinevening    1.271456
night            0.748016
generousornot   -0.105627
rushinmorning    1.830267
dtype: float64
```

The skewness of the quantitative variables in the dataset indicates varying degrees of departure from a normal distribution:
- The 'morning' variable has a skewness of approximately 0.975, indicating moderate positive skewness.
- 'rushinevening' shows a skewness of around 1.271, indicating considerable positive skewness.
- 'night' has a skewness of approximately 0.748, indicating mild positive skewness.
- 'rushinmorning' exhibits substantial positive skewness, with a value of about 1.830.
- 'generousornot' demonstrates a slight negative skewness of approximately -0.106.
To correct for skewness, the Box-Cox transformation is applied to each variable. This transformation helps normalize the distribution of the variables. Subsequently, several machine learning models are considered for regression analysis, including linear regression, k-nearest neighbors regression, support vector regression, decision tree regression, random forest regression, and XGBoost regression. These models will be evaluated based on various regression metrics such as mean squared error, R-squared, mean absolute error, and mean absolute percentage error using cross-validation techniques like RepeatedKFold.
In the feature selection process, Recursive Feature Elimination (RFE) is employed to identify the most important features for predicting the target variable. This method recursively removes features from the dataset and evaluates their impact on the performance of a classifier. The goal is to select the optimal subset of features that

maximizes the classifier's performance.
Initially, the data is preprocessed by scaling the features using standardization (z-score normalization) to ensure that all features have a mean of 0 and a standard deviation of 1. This step is crucial for some machine learning algorithms to perform effectively.
Subsequently, the Synthetic Minority Over-sampling Technique (SMOTE) is applied to address the issue of class imbalance in the dataset. SMOTE generates synthetic samples for the minority class (in this case, customers who are "generous" tippers) to balance the distribution of the target variable.
Once the data preprocessing is complete, RFE is performed using a Random Forest classifier as the estimator. RFE selects the top features deemed most important for classification based on their contribution to the model's performance. In this case, the number of features to select is set to 2.
After feature selection, the XGBoost classifier is trained on the selected features using the SMOTE-transformed training data. The trained classifier is then evaluated on the original (untransformed) test data to assess its performance. The accuracy of the classifier is calculated as the ratio of correctly predicted instances to the total number of instances in the test set.



It is interesting that VendorID is the most predictive feature. This seems to indicate that one of the two vendors tends to attract more generous customers. It may be worth performing statistical tests on the different vendors to examine this further. 'VendorID' might be considered an important feature because it could be associated with different

taxi vendors, and different vendors may have varying service qualities, pricing strategies, or other factors that could influence whether passengers are inclined to give higher tips. However, 'VendorID' being merely an identifier, does not directly explain the importance of the feature. In practical applications, more background knowledge may be needed to explain why 'VendorID' was selected as an important feature, or further analysis may be required to determine if it indeed holds significance.

Model Selection

We built a multi-classifier workflow to standardize the data, apply SMOTE (synthetic minority oversampling technique) to deal with unbalanced data, train individual classifiers, and report performance metrics, and find the best classifier parameters. The classifiers we trained include LogisticRegression, SVC, DecisionTreeClassifier,KNeighborsClassifier, MLPClassifier, RandomForestClassifier, XGBClassifier, LGBMClassifier.

Classifier	AUC
XGBoost	0.702
Logistic Regression	0.695
LightGBM	0.694
Decisiom Tree	0.694
Support Vector Machine	0.689
Multilayer Perceptron	0.611
Random Forest	0.584
kNN	0.533

Finally, by comparing the Average AUC of each model, we selected the XG Boost model with the best performance.

Hyperparameters and Prediction

Start with data preprocessing, work with unbalanced data through normalization and SMOTE, and then train and evaluate the model.

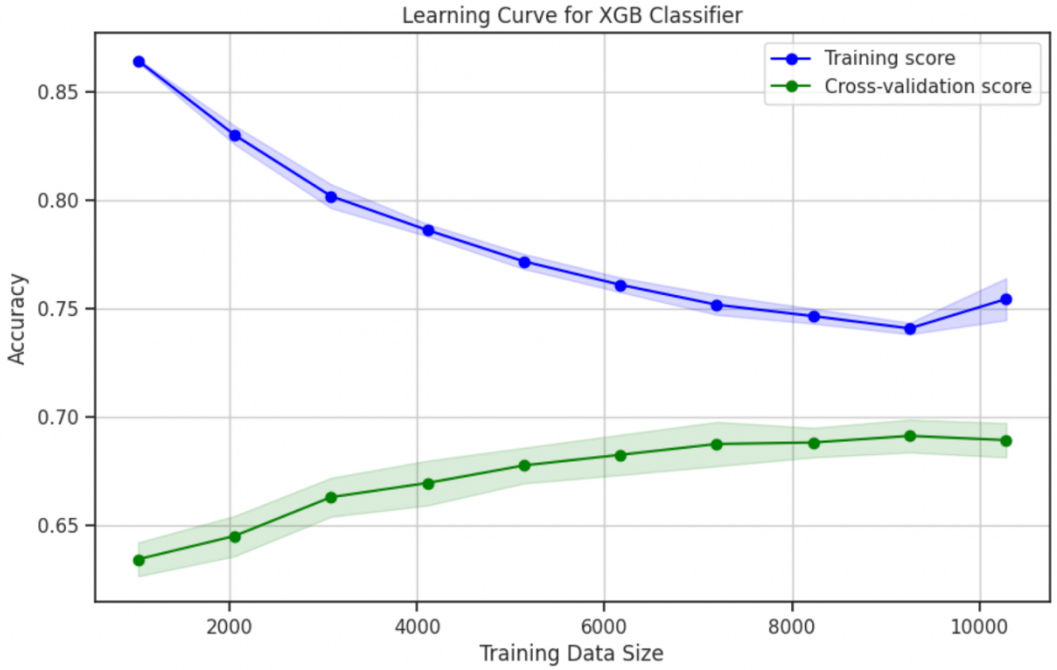
```
Precision: 0.9634343434343435
Recall: 0.96
F1 Score: 0.9598319029897976
Accuracy: 0.96
```

The model's overall performance is satisfactory. It demonstrates a high precision of 0.96, indicating that most of the predicted generous tips are correct. The recall rate of

0.96 suggests that the model is highly effective in identifying actual instances of generous tips. Furthermore, the F1 score of 0.9598 and accuracy of 0.96 reflect a well-balanced performance, maintaining a high level of correctness across all predictions.

Learning Curve

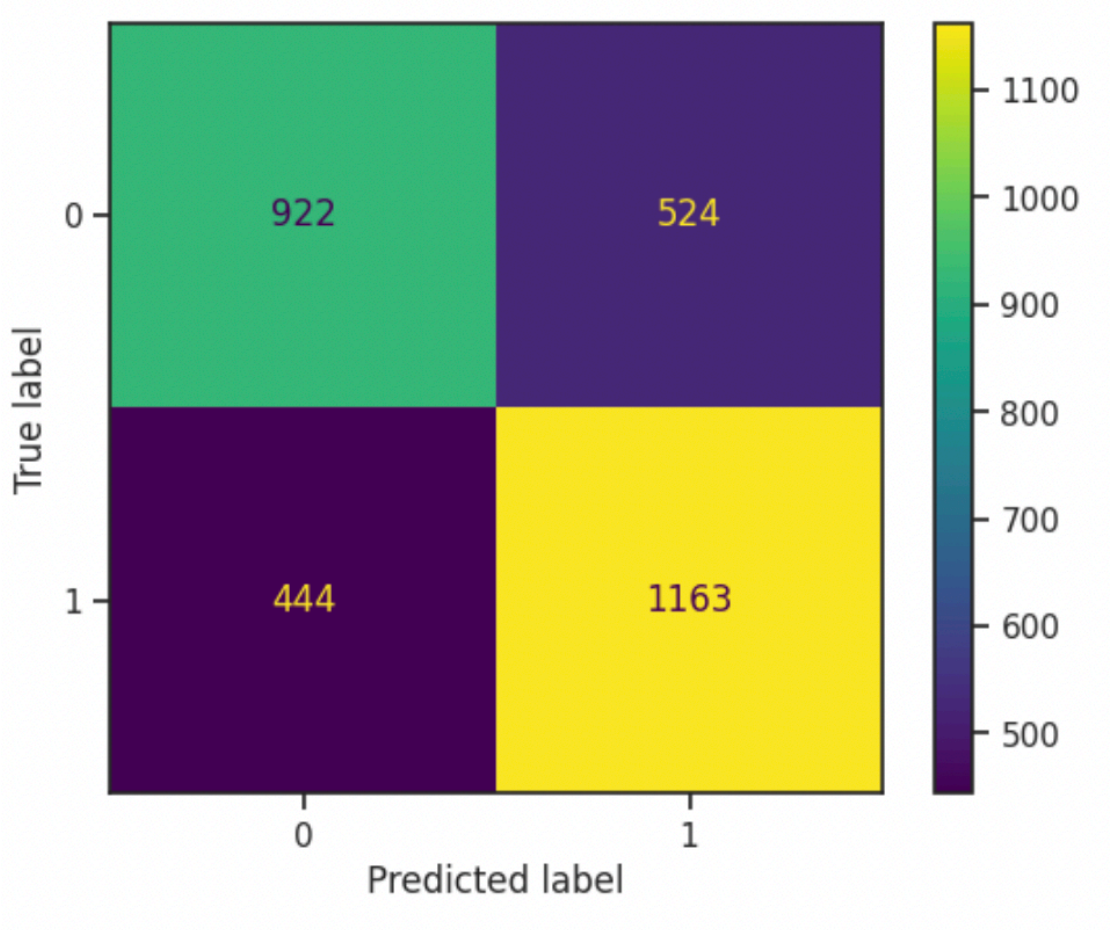
We draw a Learning Curve to evaluate the model's performance at different training set sizes to help identify whether the model is overfitting or underfitting.



With the increase of the amount of training data, the training accuracy and cross-validation accuracy of XGBoost classifier converge gradually, indicating that the model tends to be stable. The performance of the model on the training set and the verification set is more and more similar, and the overfitting phenomenon is reduced.

Confusion Matrix

Look at the confusion matrix to further understand the classification performance of the model, including the amount of TP (True Positive), TN (True Negative), FP (False Positive), FN (False Negative).



The accuracy rate of the model is about 68.94% when predicting the positive class, the recall rate is about 72.37% in the actual positive class sample, the F1 score is about 70.61%, and the overall accuracy is 68.28%. These indicators show that the model can identify most positive samples, but there is a certain proportion of misclassification.

Conclusion

The model used to predict whether a driver will get a good tip shows some interesting characteristics and performance indicators. One notable aspect is that the model is almost twice as likely to produce false positive errors (type I errors) as false negative errors (type II errors). This means that the model tends to predict tipping more often than it actually happens. While this may make drivers pleasantly surprised by an unexpected fat tip, it can also lead to disappointment when the expected fat tip doesn't materialize. From the app's perspective, it is undesirable to have more false positives than false negatives, as the disappointment of not receiving the generous tip expected may outweigh the positive surprise of receiving an unexpected tip. Despite the tendency of false positives, the overall performance of the model is satisfactory. It shows a high precision of 0.96, which suggests that most of the predicted generous tips are correct. The recall rate of 0.96 indicates that the model is very effective at identifying actual instances of generous tipping. In addition, F1 scored 0.9598 with an accuracy of 0.96, reflecting good balance performance and maintaining a high level of correctness across all predictions.

In addition to performance metrics, feature importance maps provide insight into which features are most influential in model predictions. VendorID_2: This feature has the highest importance and contributes significantly to the prediction of the model, with an importance score close to 0.9. This suggests that it plays a crucial role in deciding whether or not to give a generous tip. night, morning, rushinevening, passenger_count these properties also contribute to the model's prediction, although less so than VendorID_2. Through feature importance analysis, we can find out the relationship between customer behavior and tip generosity.

- **Vendor influence:** VendorID_2 as a key feature indicates that this vendor may provide better service or have a better payment system, making passengers more willing to tip. The brand reputation of the supplier, the professionalism of the driver and the comfort of the vehicle can all be factors. Suggestion: Taxi companies can improve tip income by improving service quality, optimizing payment experience and providing more passenger services.
- **Time factor:** The difference in tipping behavior at night and rushinevening may be because passengers at these times are more stressed and more willing to tip for better service. Passengers who ride at night may be willing to pay more for safety and convenience. Tipping behavior may be relatively low in the morning hours, when passengers are in a rush to get to work and may focus more on time than tips. Tip: Drivers can improve their tip income by providing better service during these critical hours, such as increased security at night and more efficient service during peak hours.
- **Number of passengers:** Passengers may be more inclined to tip when riding in a group, because the total cost can be divided among passengers, and the amount of tips paid by individuals is relatively small. When traveling in large groups, you may have a higher willingness to tip. Suggestion: Drivers can increase their tip income by providing more personalized services such as

route suggestions or extra assistance to passengers when riding in multiple groups.

In conclusion, while the model tends to produce more false positives that can lead to occasional driver disappointment, its overall performance is still reliable and accurate in most cases. High accuracy, recall rates, F1 scores, and accuracy indicate that the model is effective in predicting generous tipping. To improve the model's performance and improve the driver's experience, the model's thresholds can be adjusted or other techniques implemented to reduce the incidence of false positives. By solving this problem, models can become more reliable and useful in real-world applications.