Predicting order cancellations and couriers efficiency

Andreas Coclet
a.andreas@innopolis.university

1 Motivation

From a machine learning perspective, the rapid rise of food delivery companies is driven by the convenience of ordering from any Places. To optimize efficiency and cut costs, they use machine learning algorithms to predict order cancellations and plan resources, such as couriers efficiency. Predicting order cancellations is like having a crystal ball for food delivery companies. By harnessing machine learning algorithms to foresee cancellations, businesses can minimize food wastage, reduce operational expenses, and enhance the overall customer experience. This predictive power allows companies to prepare only the orders that are likely to be fulfilled, cutting down on unnecessary costs and resource allocation. Predicting the number of orders a courier can complete is fundamental for food delivery companies seeking to deliver orders promptly, maintain operational efficiency, reduce costs, and ultimately thrive in a competitive market.

2 Data

Regression Dataset (Features and Predictor)

This dataset comprises approximately 11,000 samples and is designed for a regression task. The target variable is 'total_deliveries', representing the total number of orders delivered by a courier. The features include:

1. `courier_id` (int): The unique ID of the courier.
2. `courier_transport` (text): The type of transport used by the courier.
3. `max_unique_pickups` (int): maximum number of unique locations from where the courier picked up orders.
4. `work_start` (Timestamp): The time when the courier started working.
5. `work_finish` (Timestamp): The time when the courier finished working.
6. `late_pickups` (float): The percentage of orders picked up late from the vendor.
7. `late_deliveries` (float): The percentage of orders delivered late to clients. During inference, this feature represents the maximum acceptable late deliveries.

Classification Dataset (Features and Predictor)

This dataset consists of around 290,000 samples and is intended for a classification task. The target variable is 'order_status,' indicating the final status of an order (F - finished, C - cancelled). The features include:

1. `order_create_time` (Timestamp): The time when the order was placed.
2. `total_order_items` (int): The total number of items in an order.
3. `tot_unique_items` (int): The total number of unique items in an order.
4. `cost(USD)` (float): The amount paid by the client for the order.
5. `vendor_id` (int): The ID of the vendor.
6. `payment_type` (text): The type of payment chosen by the client.
7. `vendor_client_distance` (float): The distance between the client's location and the vendor in meters.
8. `estimated_delivery_time` (float): The estimated time in minutes for delivering the order from the vendor to the client.
9. `predicted_order_preparation_time` (float): The estimated order preparation time by the vendor.

These datasets are essential for optimizing courier efficiency (regression) and predicting order outcomes (classification) in a food delivery business.

3 Exploratory data analysis

Regression Dataset "Couriers_data"

During our data exploration phase, several valuable insights were uncovered, shaping our approach to feature selection and data preprocessing:

1. **Outliers Were Identified:** Visual inspection of the dataset revealed the presence of outliers, which was evident from graphical representations. To address this issue, we applied the Robust Scaler to mitigate the potential impact of outliers on our model.
2. **High Correlation Was Noted:** Notably, there was a high correlation between the features 'total_deliveries' and 'max_unique_pickups.' This correlation suggested that these two features could provide redundant information. Consequently, we considered this factor during feature selection to prevent multicollinearity in our model.
3. **Categorical Encoding Was Employed:** We opted for one-hot encoding for the 'courier_transport' feature, representing the type of transportation used. This choice was made because the speed of delivery could depend on various factors, such as distance, traffic conditions, and delivery volume. One-hot encoding allowed our model to treat each transportation type as

an independent category, effectively capturing these nuanced distinctions.
4. **Timestamp Features Were Engineered:** From the 'work_start' and 'work_finish' timestamp features, we derived meaningful features. These included the time of day, day of the week, and the duration of working hours. These engineered features offered valuable insights into the time-based patterns of courier efficiency.
5. **Dimensionality Reduction Through PCA:** In our effort to visualize the data in a 2D space, Principal Component Analysis (PCA) was applied. This technique reduced the dimensionality of the data while retaining the most significant variance. Our goal was to retain at least "90" of the data's variance, which led us to conclude that approximately 5 principal components were sufficient to describe the data. This resulted in a reduction of dimensionality by approximately 50%.
6. **Impact on Feature Selection:** The insights gained from our data exploration greatly influenced our feature selection process. We prioritized feature engineering, one-hot encoding, and dimensionality reduction to improve model performance and interpretability. Furthermore, the identification of outliers and the acknowledgment of high correlation guided our decisions during data preprocessing, ensuring that our model was both robust and capable of capturing essential data patterns.

Classification Dataset "order_cancellation_data"

During our exploratory data analysis, several noteworthy observations were made:

1. **High Correlations Identified:** There were high correlations between features 'total_order_items,' 'tot_unique_items,' and 'cost(USD).' Similarly, 'estimated_delivery_time' and 'predicted_order_preparation_time' exhibited significant correlations.
2. **Imbalanced Target Variable:** The target variable 'order_status' displayed a high degree of imbalance, which needs to be addressed during modeling.
3. **Missing Values Present:** Approximately 12% of missing values were observed in the columns 'cost(USD),' 'vendor_client_distance,' 'estimated_delivery_time,' and 'predicted_order_preparation_time.'
4. **Skewed Data:**features, such as 'vendor_client_distance' and 'estimated_delivery_time,' exhibited skewed distributions.

Data Preprocessing

In this section, we outline our data preprocessing steps for the classification dataset:

1. **Feature Engineering:** We plan to extract meaningful features from the timestamp features to capture time-based patterns effectively.
2. **Removing Rows with Zero Values:** Rows where the number of 'total_order_items' is zero or where the 'cost(USD)' is zero were removed from the dataset.
3. **Categorical Encoding:** Categorical values were encoded to prepare them for model training.
4. **Train-Test Split:** The dataset was be split into training and test sets to facilitate model evaluation.
5. **Missing Value Imputation:** We filled missing values as follows:
   - For 'payment_type,' we used the mode to fill missing values.
   - For 'vendor_client_distance' and 'estimated_delivery_time,' which have heavily skewed distributions, we used the median for imputation.
   - For 'predicted_order_preparation_time,' we used the mean for imputation.
6. **Dimensionality Reduction:** To visualize the data in 2D, Principal Component Analysis (PCA) was applied. We aim to retain approximately 90% of the data's variance, which suggests that around 6 principal components will be used to reduce the dimensionality by about 50%.

These data preprocessing steps are crucial for preparing the classification dataset for modeling, addressing missing values, and ensuring that the model can effectively capture essential patterns in the data.

4 Task

In this section, we define the learning tasks for both regression and classification in terms of machine learning.

4.1 Regression Tasks

4.1.1 Linear Regression. Linear regression is a supervised learning task that aims to estimate a linear relationship between the input features (X) and the target variable (y). It assumes that the relationship can be expressed as:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n + \epsilon$$

Where:

y is the target variable.

β_0 is the intercept.

$\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients of the features $X_1, X_2, \ldots, X_n$.

ϵ represents the error term.

4.1.2 2nd Order Polynomial Regression with Lasso Regularization.

Polynomial regression extends linear regression by allowing for higher-order polynomial relationships between the input features and the target variable. The equation for 2nd order polynomial regression with Lasso regularization can be written as:

$$y = \beta_0 + \beta_1 X + \beta_2 X^2 + \ldots + \beta_n X^n + \epsilon - \lambda \sum_{i=1}^{n} |\beta_i|$$

Where:

y is the target variable.
X is the input feature.
$\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ are the coefficients of the polynomial terms.
ϵ represents the error term.
λ is the regularization parameter for Lasso regularization.

4.1.3 Support Vector Regression (SVR).

Support Vector Regression is a regression technique that aims to find the best-fitting hyperplane while minimizing the margin violations. The equation for SVR is typically written as:

$$y = \sum_{i=1}^{n} \alpha_i K(x_i, x) + b$$

Where:

y is the predicted output.
α_i are the Lagrange multipliers.
x_i are the support vectors.
x is the input feature vector.
K is the kernel function.
b is the bias term.

4.2 Classification Tasks

4.2.1 Logistic Regression with L1 Penalty.

Logistic regression is a binary classification algorithm that models the probability of belonging to one of two classes. With L1 penalty for regularization, the equation becomes:

$$\text{logit}(P(y = 1)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n - \lambda \sum_{i=1}^{n} |\beta_i|$$

Where:

$P(y = 1)$ is the probability of class 1.
β_0 is the intercept.
$\beta_1, \beta_2, \ldots, \beta_n$ are the coefficients of the features $X_1, X_2, \ldots, X_n$.
λ is the regularization parameter for L1 penalty.

4.2.2 Support Vector Machine (SVM) Classifier.

SVM is a powerful classification algorithm that aims to find the hyperplane that best separates different classes in the feature space. The decision boundary is represented as:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b$$

Where:

$f(x)$ is the decision function.
α_i are the Lagrange multipliers.
y_i are the class labels.
x_i are the support vectors.
x is the input feature vector.
K is the kernel function.
b is the bias term.

4.2.3 Neural Network.

Neural networks are a versatile class of models for classification tasks. The architecture and equations of a neural network depend on its specific configuration, including the number of layers, neurons, and activation functions. A general representation of a neural network's output is:

$$y = f(W^{(L)} \cdot f(W^{(L-1)} \cdot \ldots \cdot f(W^{(1)} \cdot X + b^{(1)}) + b^{(L-1)}) + b^{(L)})$$

Where:

y is the predicted output.
$W^{(i)}$ are the weight matrices for layer i.
$b^{(i)}$ are the bias vectors for layer i.
f represents the activation function.

5 Results

5.1 Regression Models Performance

Table 1. Logistic Regression with L1 Penalty

Metric	Train	Test
Mean Squared Error	12.550	13.617
R2 Score	0.662	0.628

we can see the that this model perfoms enough for a simple model. we can observe that the increase of the complexity

Table 2. 2nd Order Polynomial Lasso Regression (Alpha = 0.0005)

Metric	Train	Test
Mean Squared Error	10.092	10.501
R2 Score	0.728	0.713

leads to gain it performance on both training and test set.

Table 3. 2nd Order Polynomial Lasso Regression (Alpha = 0.0045)

Metric	Train	Test
Mean Squared Error	10.105	10.495
R2 Score	0.728	0.713

Table 4. Support Vector Regression (Kernel = Linear)

Metric	Train	Validation
Mean Squared Error	14.840	14.647
R2 Score	0.601	0.599

Table 5. Support Vector Regression (Kernel = Poly)

Metric	Train	Validation
Mean Squared Error	18.125	18.168
R2 Score	0.512	0.503

The SVR with polynomial kernel is clearly underfitting. the Radial Basis Function kernel performs better. This so far

Table 6. Support Vector Regression (Kernel = RBF)

Metric	Train	Validation
Mean Squared Error	10.486	10.796
R2 Score	0.718	0.705

our best model for this task.

5.2 Classification Models Performance

6 Data Imbalance

In the classification task, it was observed that Lasso Logistic Regression struggled to predict even a single example from class 1, resulting in an F1 score of 0 . This issue is due to the imbalanced class distribution. Various approaches were employed to tackle this challenge:

1. **Over Sampling the Minority Class:** This method involved randomly selecting samples from the minority class and replicating them until a more balanced class distribution was achieved.
2. **Under Sampling the Majority Class:** Under-sampling entailed randomly removing samples from the majority class, either with or without replacement, until a more balanced class distribution was obtained.
3. **Class Weights:** The concept behind class weighting was to assign different weights to each class during the training phase. This approach ensured that the contribution of each class was balanced and was integrated into various classifiers available in the sklearn library.

Table 7. Logistic Regression with L1 Penalty

Performance Metric	Train	Test
Accuracy	0.602312	0.529254
Precision	0.587168	0.164657
Recall	0.689180	0.673425
F1 Score	0.634097	0.264614

Table 8. Logistic Regression with L1 Penalty (Under Sampling)

Performance Metric	Train	Test
Accuracy	0.599977	0.531218
Precision	0.585996	0.164431
Recall	0.681265	0.668219
F1 Score	0.630049	0.263918

Table 9. Logistic Regression with L1 Penalty (Class Weight)

Performance Metric	Train	Test
Accuracy	0.537552	0.530081
Precision	0.172940	0.164517
Recall	0.686293	0.670959
F1 Score	0.276264	0.264243

Table 10. Support Vector Machine (Kernel = Poly)

Performance Metric	Train	Validation
Accuracy	0.649213	0.402466
Precision	0.589344	0.179494
Recall	0.984255	0.987815
F1 Score	0.737246	0.303787

In this work, we explored the effectiveness of the first two methods (over sampling and under sampling) as well as the class weighting approach. These techniques were implemented using the "imbalanced-learn" library, which complements skearn and provides specialized tools for addressing class imbalance in machine learning tasks. In term of metrics improvement , taking the case of under-sampling the model went from 0 to 0.68 in term of recall however the model presented a poor f1 score of 0.63 due to the low precision of the model. When we trained our SVC classifier we achieve f1 score 0.92 and the training set and 0.7 on the validation set.

7 Conclusion

So far, we have built various models and analyzed their performance. We encountered and successfully addressed the issue of imbalanced classes in the classification task. In our analysis, we found that the Support Vector Regression (SVR) with the RBF kernel outperformed other models for the regression task. Surprisingly, in the classification task, despite

Table 11. Support Vector Machine (Kernel = RBF)

Performance Metric	Train	Validation
Accuracy	0.925112	0.895287
Precision	0.895568	0.559712
Recall	0.962457	0.968088
F1 Score	0.927808	0.709321

Table 12. Neural Network

Performance Metric	Train	Validation
Accuracy	0.500761	0.990111
Precision	0.500793	0.968254
Recall	0.480187	0.952603
F1 Score	0.490274	0.960365

our initial expectations that the neural network would excel, the results clearly favor the SVM with the RBF kernel as the top-performing model.

8 References

1. Murphy, K. P. (2012). Machine Learning: A Probabilistic Perspective.
2. Hastie, T., Tibshirani, R., Friedman, J. (2009). Elements of Statistical Learning.
3. Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning.
4. Müller, A. C., Guido, S. (2016). Introduction to Machine Learning with Python.