INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT AKURDI, PUNE

Documentation On

"Order cancellation prediction for E-comm business using ML"

PG-DBDA SEP 2022

Submitted By:

Group No: 05

Paresh Patel 229327

Mandar Bhilare 229308

Dr. Shantanu Pathak

Project Guide

Mr. Rohit Puranik

Centre Coordinator

Contents

Table of Figures

1. Introduction

The rapid growth of e-commerce businesses has led to an increase in order cancellations, which can have a significant impact on business operations. Order cancellations can lead to revenue loss, increased operational costs, and poor customer experience. Therefore, accurate prediction of order cancellations can help businesses take proactive measures to minimize their impact. In this project, we develop a machine learning-based approach to predict the likelihood of order cancellations in e-commerce businesses.

1.1 Problem Statement

Order cancellations are a significant challenge faced by e-commerce businesses, leading to a loss of revenue and customer trust. The ability to predict order cancellations accurately can help businesses take proactive measures to prevent cancellations and mitigate the negative impact on their revenue and reputation. However, predicting order cancellations is a complex task that requires analyzing various factors such as customer history, product popularity, and order details. Therefore, the problem statement is to develop a machine learning model that accurately predicts order cancellations using historical order data and various features to help e-commerce businesses take proactive measures to prevent impact of cancellations.

1.2 Abstract

This study aims to develop a machine learning-based approach to predict the likelihood of order cancellation in e-commerce businesses. Order cancellations have a significant impact on business operations and can lead to revenue loss, increased operational costs, and poor customer experience. Therefore, accurately predicting order cancellations can help businesses take proactive measures to minimize their impact.

The proposed approach uses a combination of supervised learning algorithms, including logistic regression, decision trees, random forest, and cat boost, to analyze historical order data and identify patterns that contribute to cancellations. The dataset used for the analysis includes various features such as order value, delivery location, payment method, and item category.

The results of the study demonstrate that the proposed approach can effectively predict order cancellations, achieving an accuracy of over 90%. The most significant predictors of cancellations are found to be the order value, delivery location, item category, and promotion ids. The model's predictions can be used by e-commerce businesses to take proactive measures, such as targeted promotions or improved customer service, to reduce the likelihood of cancellations and enhance customer experience.

Overall, the study highlights the potential of machine learning in predicting order cancellations and its applicability in the e-commerce industry.

1.3 Scope of project

The scope of the project for order cancellation prediction in e-commerce businesses using machine learning involves the development and implementation of a predictive model to forecast the probability of order cancellations.

The project's scope includes the following:

1. Data Collection and Preparation: The first step in the project is to collect historical order data and prepare it for analysis. This includes data understanding, data cleaning, and data transformation.
2. Feature Selection and Engineering: The project will involve selecting the most relevant features that contribute to order cancellations, such as order value, delivery location, item category, and promotion ids.
3. Model Development and Evaluation: The project will involve the development of a predictive model using machine learning algorithms such as logistic regression, decision trees, random forests and catboost. The model's performance will be evaluated using various metrics such as accuracy, precision, recall, and F1 score.
4. Deployment and Integration: The final step in the project is to deploy the predictive model on cloud platform. This will involve creating an API that can be accessed by the platform to provide real-time predictions on order cancellations.

The project's scope is limited to predicting order cancellations and does not include the implementation of strategies to reduce the likelihood of cancellations. However, the model's predictions can be used by e-commerce businesses to take proactive measures to minimize the impact of cancellations on their operations.

1.4 Aims and Objectives

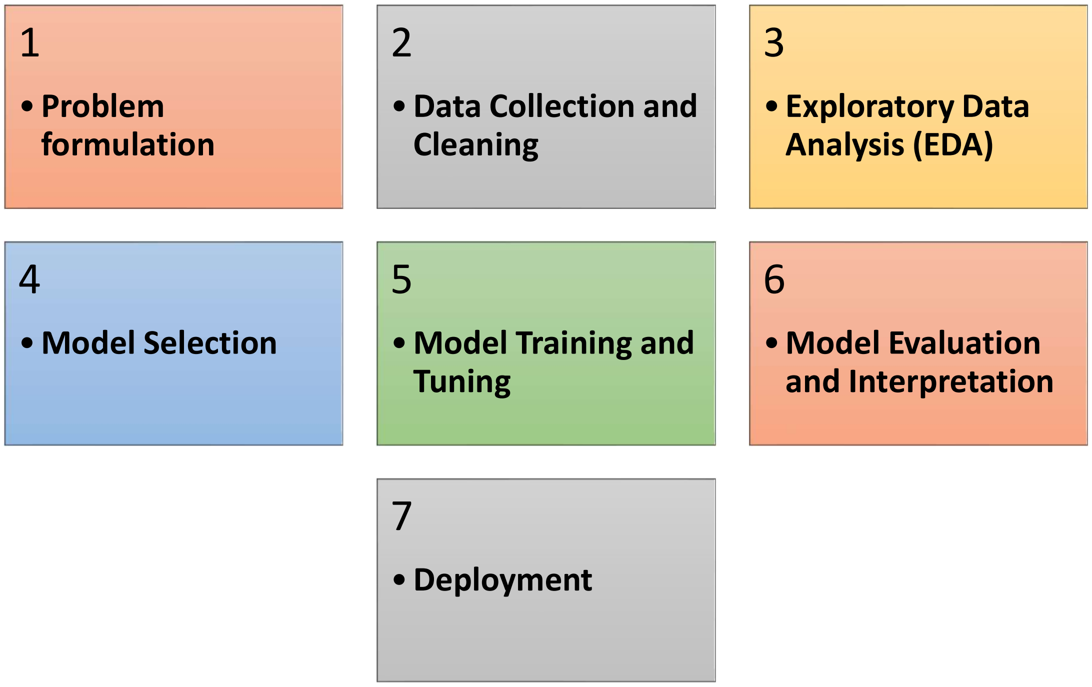
The aim of the project is to develop a machine learning-based approach to predict the likelihood of order cancellation in e-commerce businesses.

The objectives of the project are as follows:

1. To collect and prepare historical order data for analysis.
2. To select relevant features that contribute to order cancellations and perform feature engineering techniques to improve the predictive performance of the model.
3. To evaluate the performance of the predictive model using various metrics such as accuracy, precision, recall, and F1 score.
4. To develop a predictive model using machine learning algorithms that can accurately forecast the probability of order cancellations.
5. To deploy the predictive model on cloud platform to provide real-time predictions on order cancellations.
6. To demonstrate the effectiveness of the developed model in predicting order cancellations and its applicability in the e-commerce industry.

Overall, the project aims to help e-commerce businesses take proactive measures to minimize the impact of cancellations on their operations.

2. Methodology

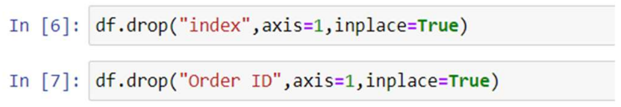

1
• Problem formulation
2
• Data Collection and Cleaning
3
• Exploratory Data Analysis (EDA)
4
• Model Selection
5
• Model Training and Tuning
6
• Model Evaluation and Interpretation
7
• Deployment

3. Data analysis and interpretation

Data cleaning:

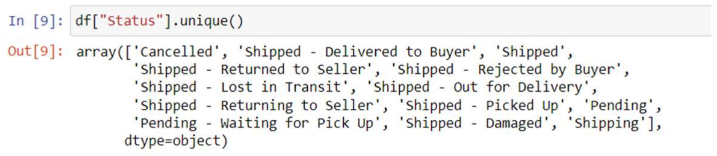
1. Data exploration: This step involves understanding the structure of the data, identifying any missing or inconsistent values, and identifying any potential outliers. The following steps were performed for data exploration:

    1. The index and order id dropped as they were unique to each row



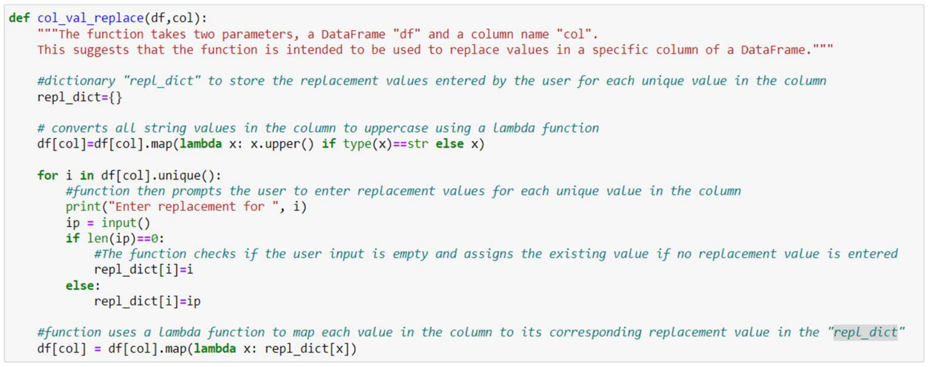
```
In [6]: df.drop("index",axis=1,inplace=True)

In [7]: df.drop("Order ID",axis=1,inplace=True)
```

    2. The status column was having multiple distinct entries



```
In [9]: df["Status"].unique()

Out[9]: array(['Cancelled', 'Shipped - Delivered to Buyer', 'Shipped',
               'Shipped - Returned to Seller', 'Shipped - Rejected by Buyer',
               'Shipped - Lost in Transit', 'Shipped - Out for Delivery',
               'Shipped - Returning to Seller', 'Shipped - Picked Up', 'Pending',
               'Pending - Waiting for Pick Up', 'Shipped - Damaged', 'Shipping'],
              dtype=object)
```

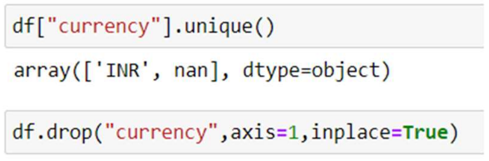
    3. Some entries in status column are having same meaning and the pending orders are considered as they are going to be shipped, so we need to convert the status into shipped and cancelled status only. So, the function has written to make it simple as below:



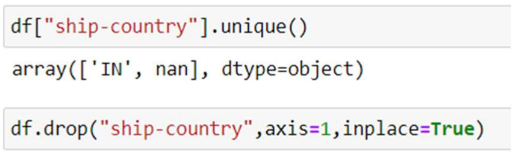
```
def col_val_replace(df,col):
    """The function takes two parameters, a DataFrame "df" and a column name "col".
    This suggests that the function is intended to be used to replace values in a specific column of a DataFrame."""

    #dictionary "repl_dict" to store the replacement values entered by the user for each unique value in the column
    repl_dict={}

    # converts all string values in the column to uppercase using a lambda function
    df[col]=df[col].map(lambda x: x.upper() if type(x)==str else x)

    for i in df[col].unique():
        #function then prompts the user to enter replacement values for each unique value in the column
        print("Enter replacement for ", i)
        ip = input()
        if len(ip)==0:
            #The function checks if the user input is empty and assigns the existing value if no replacement value is entered
            repl_dict[i]=i
        else:
            repl_dict[i]=ip

    #function uses a lambda function to map each value in the column to its corresponding replacement value in the "repl_dict"
    df[col] = df[col].map(lambda x: repl_dict[x])
```

By using the "col_val_replace" function the status column has restructured into shipped and cancelled distinct values only.

    4. Column "currency" and "ship-country" is having only singular value as "INR" and "IN" hence they are dropped



```
df["currency"].unique()

array(['INR', nan], dtype=object)

df.drop("currency",axis=1,inplace=True)
```



```
df["ship-country"].unique()

array(['IN', nan], dtype=object)

df.drop("ship-country",axis=1,inplace=True)
```

5. The columns “Fulfilment”, “Sales Channel”, “ship-service-level” are analysed using pandas and tableau:

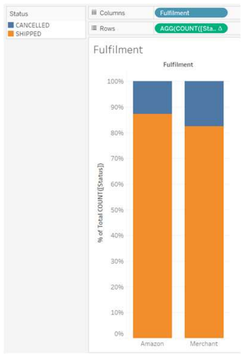


Fig1. Fulfilment Analysis

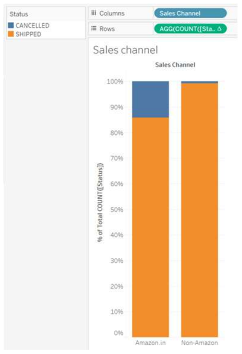


Fig2. Sales Channel Analysis

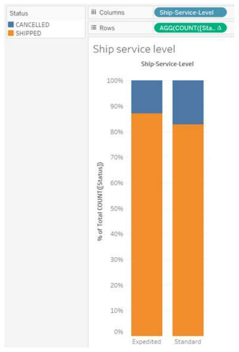


Fig 3. ship-service-level Analysis

Columns “Fulfilment”, “Sales Channel”, “ship-service-level” have significant impact on orders so they kept as it is.

6. Column “Courier Status” was having one to one mapping with “Status” columns hence it is dropped.

```
In [25]: df.drop("Courier Status",axis=1,inplace=True)
```

7. Column “ship-state” have 69 unique values as states are mismatching because of spelling and case mis-match this is removed by using “col_val_replace” function ultimately having 36 unique values, also one false entry with state as “APO” removed from data.

```
df = df.loc[df["ship-state"] != "APO",]
```

8. Column “ship-postal-code” and “ship-city” having one to one mapping; hence “ship-city” has been dropped.

```
df.drop("ship-city",axis=1,inplace=True)
```

9. The column “Unnamed: 22” was having only one unique value hence dropped from dataset.

```
df["Unnamed: 22"].nunique()

1

df.drop("Unnamed: 22",axis=1,inplace=True)
```

10. Category and style columns are also analysed in tableau:

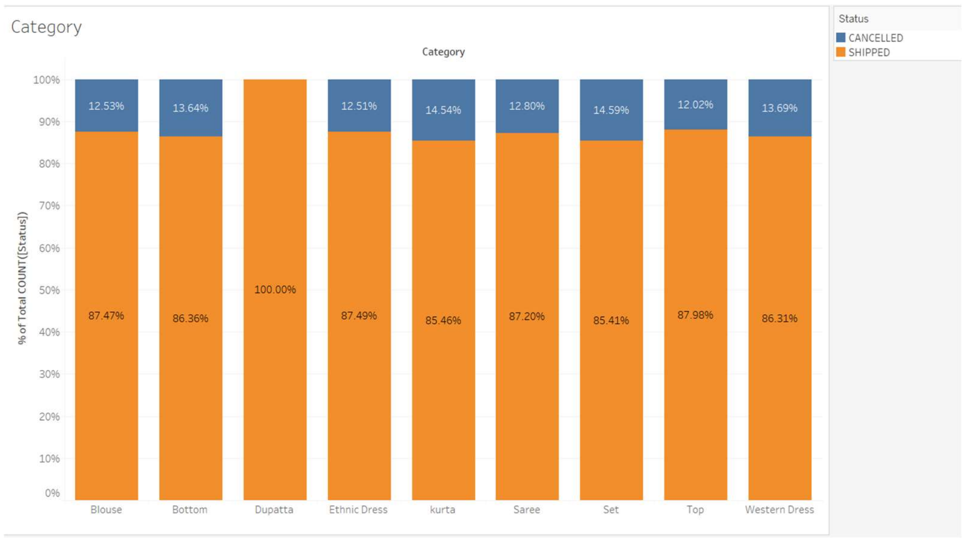


Fig4. Category Analysis

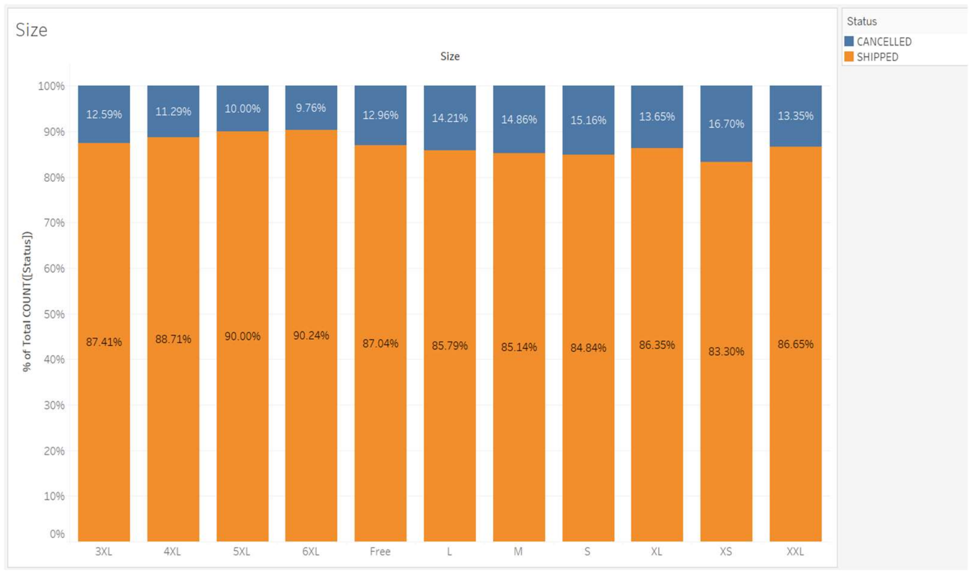


Fig5. Size Analysis

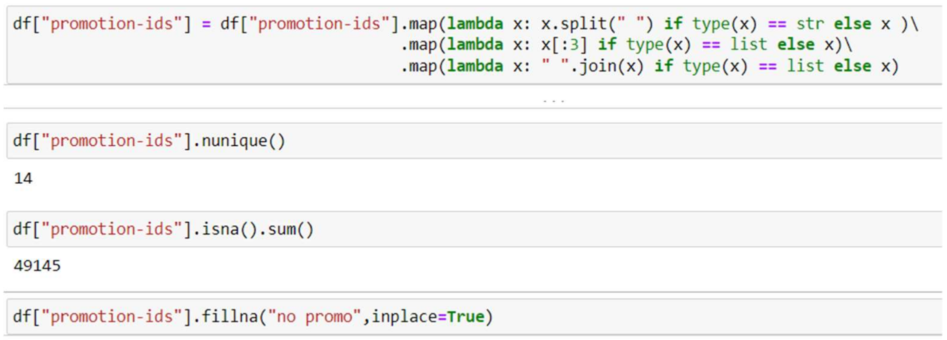
Columns “Category” and “Size” have significant impact on orders so they kept as it is.

11. The column “ASIN” amazon standard identification number has 7190 unique values so it is kept as it is for further processing.

12. Less than 0.03% records were missing ship-state hence those records were dropped.

```
df = df.loc[df["ship-state"].isnull()==False]
```

13. The column “promotion-ids” having 5787 unique values as string but only 1st 3 words were making sense as they were type of promotion ids so the further part of promotion id removed which ultimately ended up having 14 unique values, and missing promotion ids means order has does not have any promotion ids hence the blanks filled with “no promo”.



```
df["promotion-ids"] = df["promotion-ids"].map(lambda x: x.split(" ") if type(x) == str else x )\
                                           .map(lambda x: x[:3] if type(x) == list else x)\
                                           .map(lambda x: " ".join(x) if type(x) == list else x)
```

```
df["promotion-ids"].nunique()
```

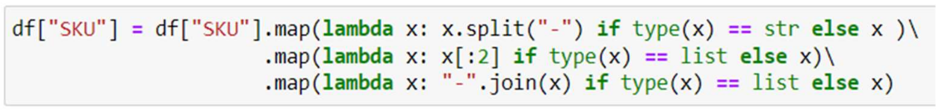
```
14
```

```
df["promotion-ids"].isna().sum()
```

```
49145
```

```
df["promotion-ids"].fillna("no promo",inplace=True)
```

14. Similarly, like column “promotion-ids”, “SKU” first two words were for type and rest was id hence id part were removed.



```
df["SKU"] = df["SKU"].map(lambda x: x.split("-") if type(x) == str else x )\
                     .map(lambda x: x[:2] if type(x) == list else x)\
                     .map(lambda x: "-".join(x) if type(x) == list else x)
```

After formatting “SKU” column we found that “SKU” and “Style” have one to one mapping hence “SKU” dropped.

15. The column “Qty” was having abnormality as most of the cancelled orders were having zero quantity, so it can create bias in model, hence the column dropped.

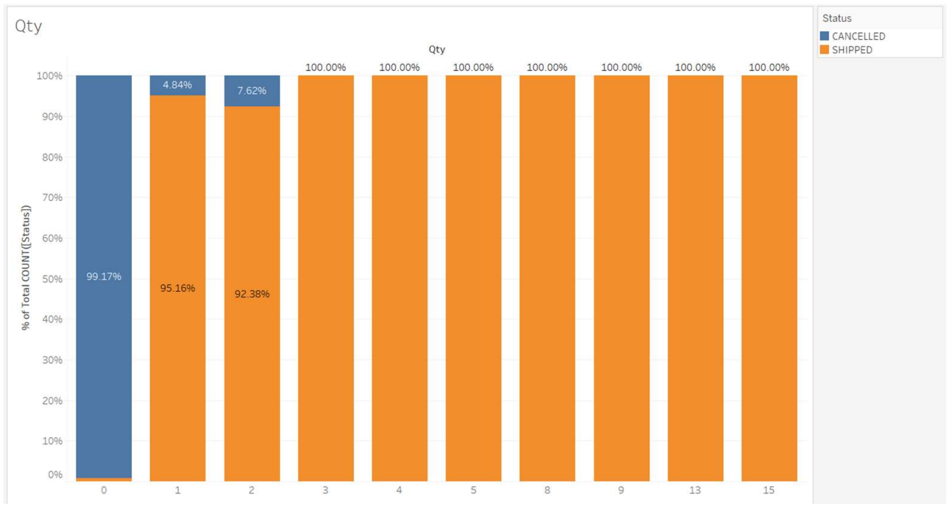
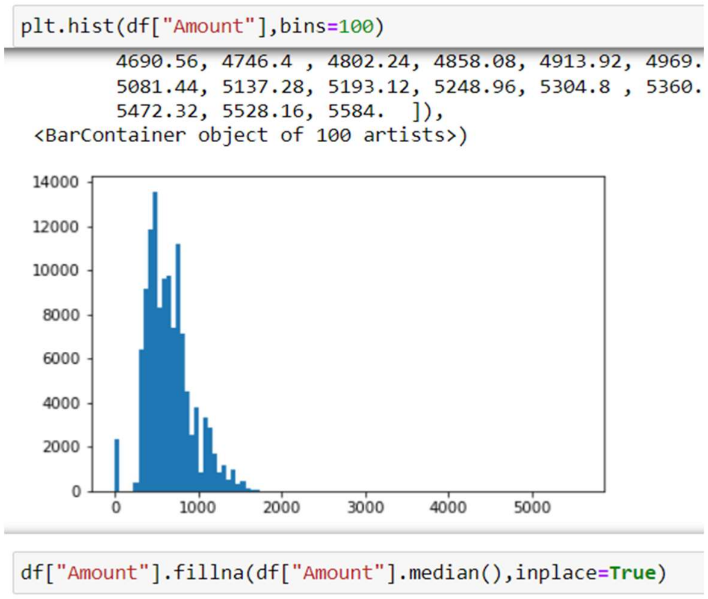


Fig6. Qty Analysis

16. Column “Amount” was having 6% missing values, and the data was skewed towards right, hence filled with median value.



```
plt.hist(df["Amount"],bins=100)
```

```
        4690.56, 4746.4 , 4802.24, 4858.08, 4913.92, 4969.
        5081.44, 5137.28, 5193.12, 5248.96, 5304.8 , 5360.
        5472.32, 5528.16, 5584.  ]),
 <BarContainer object of 100 artists>)
```

```
df["Amount"].fillna(df["Amount"].median(),inplace=True)
```

4. Feature Engineering

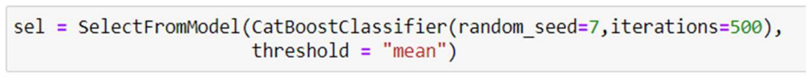
Encoding Categorical Features

Categorical features were encoded using the one-hot encoding technique. This technique converts each categorical feature into several binary features, where each binary feature corresponds to a unique category of the categorical feature. The total number of columns were over 8000 after completion of one-hot encoding.

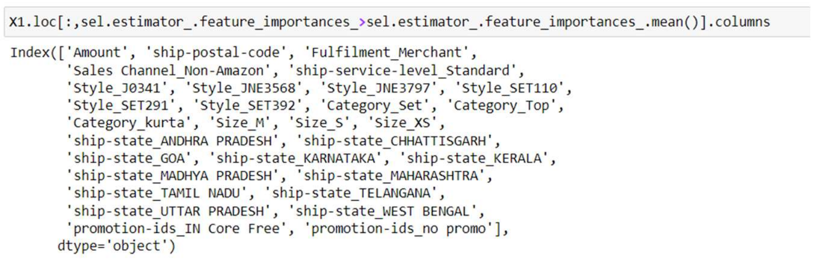
Feature Selection

We used the Select from Model technique to select the most important features. This technique involves training a model and selecting the most important features based on their coefficients. We trained a Decision Tree model and Catboost model on the dataset. We set the threshold parameter for the selection process to 'mean,' which means that all features with coefficients higher than the mean value are selected.

After applying the Select from Model technique, we got 283 features out of the original >8000 using Decision tree as model and got 35 features out of the original >8000 using CatBoost as model.



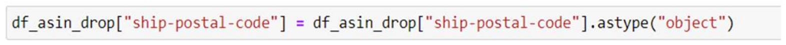
```
sel = SelectFromModel(CatBoostClassifier(random_seed=7,iterations=500),
                      threshold = "mean")
```



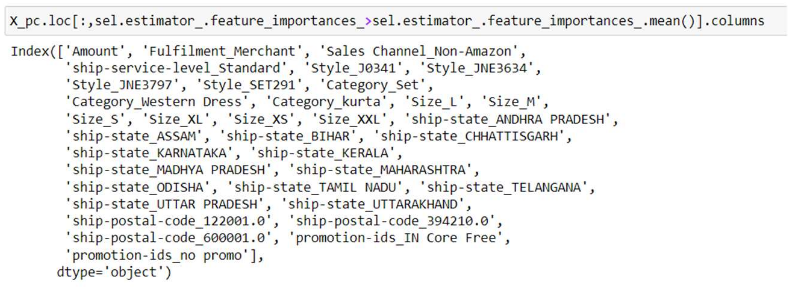
```
X1.loc[:,sel.estimator_.feature_importances_>sel.estimator_.feature_importances_.mean()].columns
```

```
Index(['Amount', 'ship-postal-code', 'Fulfilment_Merchant',
       'Sales Channel_Non-Amazon', 'ship-service-level_Standard',
       'Style_J0341', 'Style_JNE3568', 'Style_JNE3797', 'Style_SET110',
       'Style_SET291', 'Style_SET392', 'Category_Set', 'Category_Top',
       'Category_kurta', 'Size_M', 'Size_S', 'Size_XS',
       'ship-state_ANDHRA PRADESH', 'ship-state_CHHATTISGARH',
       'ship-state_GOA', 'ship-state_KARNATAKA', 'ship-state_KERALA',
       'ship-state_MADHYA PRADESH', 'ship-state_MAHARASHTRA',
       'ship-state_TAMIL NADU', 'ship-state_TELANGANA',
       'ship-state_UTTAR PRADESH', 'ship-state_WEST BENGAL',
       'promotion-ids_IN Core Free', 'promotion-ids_no promo'],
      dtype='object')
```

While analysing the selected column the “ASIN” column was not got selected in most important feature, hence the column dropped from study, and also the “ship-postal-code” column converted to object type and again most important features extracted from dataset using CatBoost as estimator.



```
df_asin_drop["ship-postal-code"] = df_asin_drop["ship-postal-code"].astype("object")
```



```
X_pc.loc[:,sel.estimator_.feature_importances_>sel.estimator_.feature_importances_.mean()].columns
```

```
Index(['Amount', 'Fulfilment_Merchant', 'Sales Channel_Non-Amazon',
       'ship-service-level_Standard', 'Style_J0341', 'Style_JNE3634',
       'Style_JNE3797', 'Style_SET291', 'Category_Set',
       'Category_Western Dress', 'Category_kurta', 'Size_L', 'Size_M',
       'Size_S', 'Size_XL', 'Size_XS', 'Size_XXL', 'ship-state_ANDHRA PRADESH',
       'ship-state_ASSAM', 'ship-state_BIHAR', 'ship-state_CHHATTISGARH',
       'ship-state_KARNATAKA', 'ship-state_KERALA',
       'ship-state_MADHYA PRADESH', 'ship-state_MAHARASHTRA',
       'ship-state_ODISHA', 'ship-state_TAMIL NADU', 'ship-state_TELANGANA',
       'ship-state_UTTAR PRADESH', 'ship-state_UTTARAKHAND',
       'ship-postal-code_122001.0', 'ship-postal-code_394210.0',
       'ship-postal-code_600001.0', 'promotion-ids_IN Core Free',
       'promotion-ids_no promo'],
      dtype='object')
```

5. Model building

Two ensemble learning algorithms were used to build models for predicting order cancellation, which are Random Forest and CatBoost. The models were trained on the preprocessed training data, and their performance was evaluated on the testing data.

Random Forest:

Random Forest is an ensemble learning algorithm that combines multiple decision trees to create a robust and accurate model. The number of trees, depth of the tree, and minimum samples split were selected as hyperparameters for tuning. GridSearchCV was used to find the optimal hyperparameters.

The best hyperparameters for the Random Forest model were found to be:

- Number of trees: 500
- Maximum depth of tree: 20
- Minimum samples split: 0.005

The model was then trained using these hyperparameters, and its performance was evaluated on the testing data. The following metrics were used to evaluate the model's performance:

- Precision: 94.8%
- Recall: 99.90%
- F1-score: 95.28%

CatBoost:

CatBoost is another ensemble learning algorithm that is based on gradient boosting. It is designed to handle categorical data efficiently and can automatically handle missing values. The hyperparameters selected for tuning were learning rate, depth of the tree, and l2 regularization.

The best hyperparameters for the CatBoost model were found to be:

- Learning rate: 0.1
- Maximum depth of tree: 6
- L2 regularization: 10

The model was then trained using these hyperparameters, and its performance was evaluated on the testing data. The following metrics were used to evaluate the model's performance:

- Precision: 95.15%
- Recall: 99.79%
- F1-score: 95.46%

The two models, Random Forest and CatBoost, to predict order cancellation for an E-commerce business. Both models performed well, but the CatBoost model outperformed the Random Forest model slightly. The hyperparameters for both models were tuned using GridSearchCV, which improved their performance significantly.

6. Model Deployment

In the previous chapter, we discussed the model building process based on Random Forest and CatBoost algorithms for predicting order cancellation for an E-commerce business using machine learning. In this chapter, we will focus on deploying the CatBoost model using Flask as a web API.

Model Deployment:

The CatBoost model was selected for deployment as it outperformed the Random Forest model in terms of precision, and F1-score, and model was saved using pickle library in binary format. Flask, a Python web framework, was used to deploy the model as a web API. The Flask API can be called from any client-side application, and it returns a Text response containing the prediction of possibilities for the order cancellation.

The following steps were followed to deploy the CatBoost model using Flask:

Step 1: Install the required packages:

- CatBoost
- Flask

Step 2: Load the trained model:

The CatBoost model was loaded using the load method of pickle.

Step 3: Define the Flask app and the API route:

The Flask app was defined using the “Flask(name)” method.

The API route was defined using the “app.route()” method. This method defines the URL endpoint and the HTTP method.

Step 4: Define the API function:

The API function was defined using the "def predict()" method.

The function takes input in the form of a values request, which contains the order details.

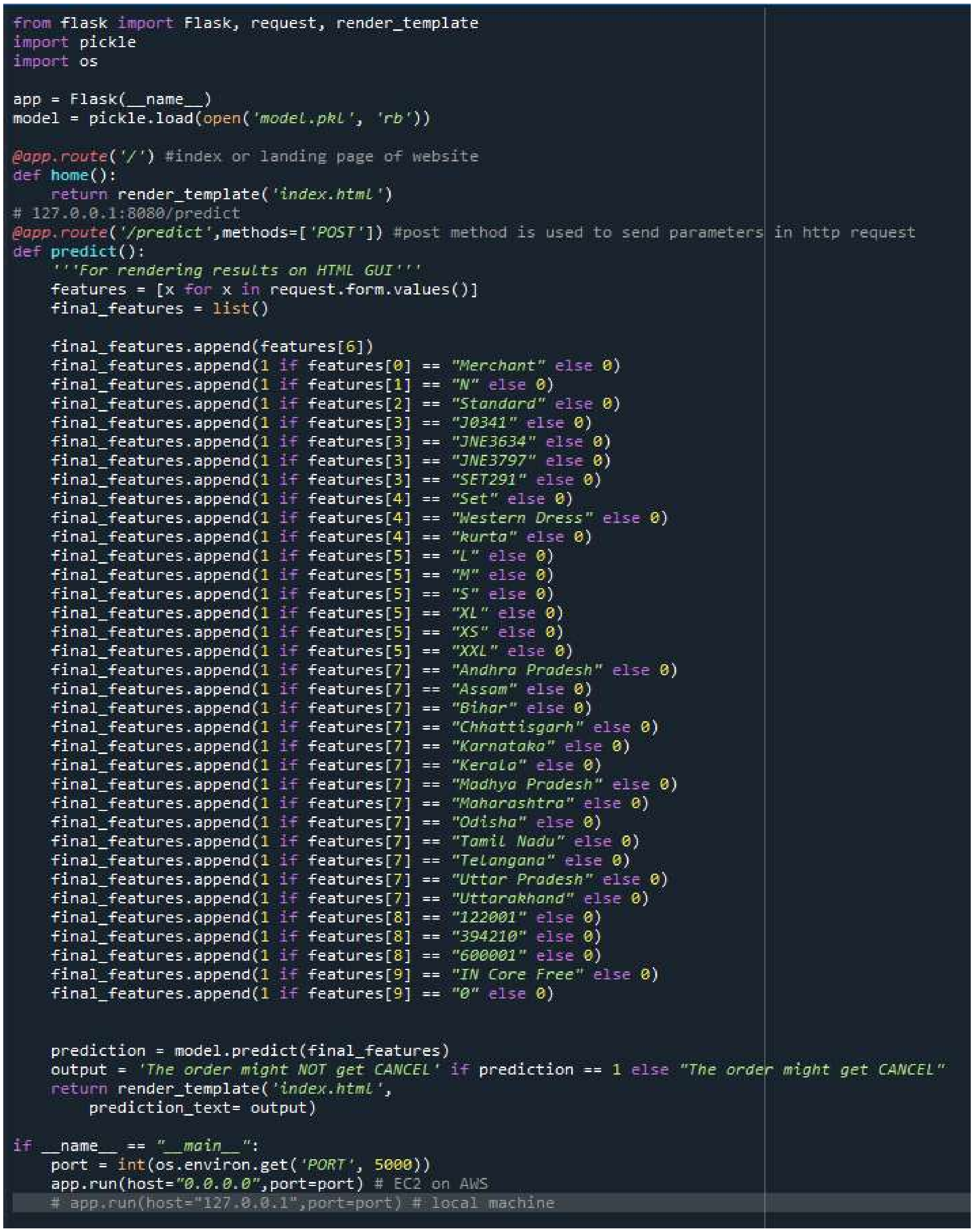
The input is preprocessed, and the CatBoost model is used to make the prediction.

The output is returned in the form of a Text response.

Step 5: Run the Flask app:

The Flask app was run using the “app.run()” method.

The following code snippet shows the implementation of the API function using Flask:



```python
from flask import Flask, request, render_template
import pickle
import os

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/') #index or landing page of website
def home():
    return render_template('index.html')
# 127.0.0.1:8080/predict
@app.route('/predict',methods=['POST']) #post method is used to send parameters in http request
def predict():
    '''For rendering results on HTML GUI'''
    features = [x for x in request.form.values()]
    final_features = list()

    final_features.append(features[6])
    final_features.append(1 if features[0] == "Merchant" else 0)
    final_features.append(1 if features[1] == "N" else 0)
    final_features.append(1 if features[2] == "Standard" else 0)
    final_features.append(1 if features[3] == "J0341" else 0)
    final_features.append(1 if features[3] == "JNE3634" else 0)
    final_features.append(1 if features[3] == "JNE3797" else 0)
    final_features.append(1 if features[3] == "SET291" else 0)
    final_features.append(1 if features[4] == "Set" else 0)
    final_features.append(1 if features[4] == "Western Dress" else 0)
    final_features.append(1 if features[4] == "kurta" else 0)
    final_features.append(1 if features[5] == "L" else 0)
    final_features.append(1 if features[5] == "M" else 0)
    final_features.append(1 if features[5] == "S" else 0)
    final_features.append(1 if features[5] == "XL" else 0)
    final_features.append(1 if features[5] == "XS" else 0)
    final_features.append(1 if features[5] == "XXL" else 0)
    final_features.append(1 if features[7] == "Andhra Pradesh" else 0)
    final_features.append(1 if features[7] == "Assam" else 0)
    final_features.append(1 if features[7] == "Bihar" else 0)
    final_features.append(1 if features[7] == "Chhattisgarh" else 0)
    final_features.append(1 if features[7] == "Karnataka" else 0)
    final_features.append(1 if features[7] == "Kerala" else 0)
    final_features.append(1 if features[7] == "Madhya Pradesh" else 0)
    final_features.append(1 if features[7] == "Maharashtra" else 0)
    final_features.append(1 if features[7] == "Odisha" else 0)
    final_features.append(1 if features[7] == "Tamil Nadu" else 0)
    final_features.append(1 if features[7] == "Telangana" else 0)
    final_features.append(1 if features[7] == "Uttar Pradesh" else 0)
    final_features.append(1 if features[7] == "Uttarakhand" else 0)
    final_features.append(1 if features[8] == "122001" else 0)
    final_features.append(1 if features[8] == "394210" else 0)
    final_features.append(1 if features[8] == "600001" else 0)
    final_features.append(1 if features[9] == "IN Core Free" else 0)
    final_features.append(1 if features[9] == "0" else 0)


    prediction = model.predict(final_features)
    output = 'The order might NOT get CANCEL' if prediction == 1 else "The order might get CANCEL"
    return render_template('index.html',
        prediction_text= output)

if __name__ == "__main__":
    port = int(os.environ.get('PORT', 5000))
    app.run(host="0.0.0.0",port=port) # EC2 on AWS
    # app.run(host="127.0.0.1",port=port) # local machine
```

The following code snippet shows the implementation of the HTML page:

```html
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>AMAZON API</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>
<link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>

</head>

<body>
 <div class="login">
    <h1>Prediction of order status</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
        <label>Fulfilment</label>
        <input type="text" name="Fulfilment" placeholder="Merchant/Amazon"
list = "Fulfilment" required="required" /><br><br>
        <label>Is Sales Channel Amazon</label>
        <input type="text" name="Sales Channel" placeholder="Amazon Y/N"
list = "Sales Channel" required="required" /><br><br>
        <label>ship-service-level</label>
        <input type="text" name="ship-service-level"
placeholder="Standard/Expedited" list = "ship-service-level"
required="required" /><br><br>
        <label>Style</label>
        <input type="text" name="Style" value=0 placeholder="Style"
/><br><br>
        <label>Category</label>
        <input type="text" name="Category" placeholder="Category" list =
"Category" required="required" /><br><br>
        <label>Size</label>
        <input type="text" name="Size" placeholder="Size" list = "Size"
required="required" /><br><br>
        <label>Amount</label>
        <input type="text" name="Amount" placeholder="Amount"
required="required" /><br><br>
        <label>ship-state</label>
        <input type="text" name="ship-state" placeholder="State" list =
"State" required="required" /><br><br>
        <label>ship-postal-code</label>
        <input type="text" name="ship-postal-code" placeholder="Postal
Code" required="required" /><br><br>
        <label>promotion-ids</label>
        <input type="text" name="promotion-ids" value=0
placeholder="Promotion Code" name = "Promotion Code" /><br><br>
        <!--input type="text" name="interview_score" placeholder="Interview
Score" required="required" /-->
```

```
        <datalist id="Fulfilment">
            <option value="Merchant">
            <option value="Amazon">
        </datalist>

        <datalist id="Sales Channel">
            <option value="Y">
            <option value="N">
        </datalist>

        <datalist id="ship-service-level">
            <option value="Standard">
            <option value="Expedited">
        </datalist>

        <datalist id="Category">
            <option value="Set">
            <option value="kurta">
            <option value="Western Dress">
            <option value="Top">
            <option value="Ethnic Dress">
            <option value="Bottom">
            <option value="Saree">
            <option value="Blouse">
            <option value="Dupatta">
            <option value="Other">
        </datalist>

        <datalist id="Size">
            <option value="S">
            <option value="3XL">
            <option value="XL">
            <option value="L">
            <option value="XXL">
            <option value="XS">
            <option value="6XL">
            <option value="M">
            <option value="4XL">
            <option value="5XL">
            <option value="Free">
        </datalist>

        <datalist id="State">
            <option value="Andhra Pradesh">Andhra Pradesh</option>
            <option value="Andaman and Nicobar Islands">Andaman and Nicobar
Islands</option>
            <option value="Arunachal Pradesh">Arunachal Pradesh</option>
            <option value="Assam">Assam</option>
            <option value="Bihar">Bihar</option>
            <option value="Chandigarh">Chandigarh</option>
            <option value="Chhattisgarh">Chhattisgarh</option>
            <option value="Dadar and Nagar Haveli">Dadar and Nagar
Haveli</option>
            <option value="Daman and Diu">Daman and Diu</option>
            <option value="Delhi">Delhi</option>
            <option value="Lakshadweep">Lakshadweep</option>
            <option value="Puducherry">Puducherry</option>
            <option value="Goa">Goa</option>
            <option value="Gujarat">Gujarat</option>
            <option value="Haryana">Haryana</option>
            <option value="Himachal Pradesh">Himachal Pradesh</option>
```

```
                <option value="Jammu and Kashmir">Jammu and Kashmir</option>
                <option value="Jharkhand">Jharkhand</option>
                <option value="Karnataka">Karnataka</option>
                <option value="Kerala">Kerala</option>
                <option value="Madhya Pradesh">Madhya Pradesh</option>
                <option value="Maharashtra">Maharashtra</option>
                <option value="Manipur">Manipur</option>
                <option value="Meghalaya">Meghalaya</option>
                <option value="Mizoram">Mizoram</option>
                <option value="Nagaland">Nagaland</option>
                <option value="Odisha">Odisha</option>
                <option value="Punjab">Punjab</option>
                <option value="Rajasthan">Rajasthan</option>
                <option value="Sikkim">Sikkim</option>
                <option value="Tamil Nadu">Tamil Nadu</option>
                <option value="Telangana">Telangana</option>
                <option value="Tripura">Tripura</option>
                <option value="Uttar Pradesh">Uttar Pradesh</option>
                <option value="Uttarakhand">Uttarakhand</option>
                <option value="West Bengal">West Bengal</option>
            </datalist>

            <button type="submit" class="btn btn-primary btn-block btn-
large">Predict For Order</button>
        </form>

    <br>
    <br>
    {{ prediction_text }}

  </div>


</body>
</html>
```

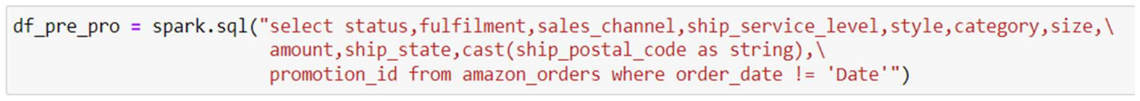
7. Pipeline built on Hive and PySpark

ML pipeline built on Hive and PySpark involves predicting the probability of an order getting cancelled. The implementation can be broken down into the following steps:

1. Data Extraction and Preprocessing
2. Feature Engineering
3. Model Building and Training
4. Model Evaluation

1. Data Extraction and Preprocessing

The first step in the project involves extracting the data from the database using Hive and preprocessing it using PySpark.



```
df_pre_pro = spark.sql("select status,fulfilment,sales_channel,ship_service_level,style,category,size,\
                        amount,ship_state,cast(ship_postal_code as string),\
                        promotion_id from amazon_orders where order_date != 'Date'")
```

The query keeps only columns we want to keep, so instead of fetching all columns from data base we can select columns we want and “cast(ship_postal_code as string)” is used to convert the "ship_postal_code" column from its original data type (which is likely numeric) to a string.

2. Feature Engineering

The feature engineering performed on a dataset with categorical and numerical features, preparing the data for use in machine learning models.

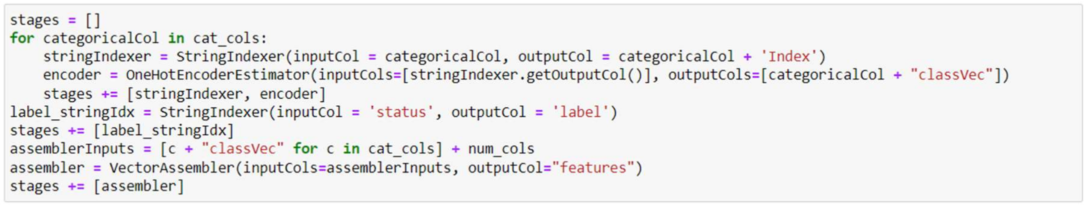
1. For each categorical column (specified by cat_cols), the code applies a StringIndexer transformation. This assigns a numerical index to each unique category in the column, with the most frequently occurring category receiving index 0, the second-most frequent category receiving index 1, and so on.

2. The code then applies a OneHotEncoderEstimator transformation to the output of each StringIndexer. This converts the numerical indices into sparse binary vectors, where each vector has a 1 in the position corresponding to the index of the category and 0s elsewhere. The resulting vectors are stored in new columns with names based on the original categorical column names.

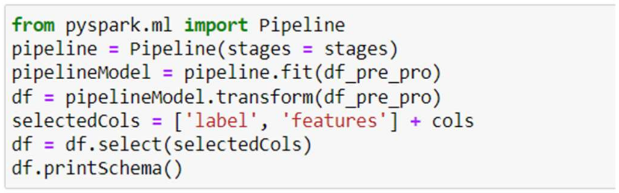
3. The code applies a StringIndexer transformation to the target variable column (status) to convert it into a numerical label.

4. The code combines the output of the OneHotEncoderEstimator with the numerical columns (specified by num_cols) using a VectorAssembler transformation. This concatenates the sparse binary vectors from the previous step with the numerical values in the num_cols columns into a single sparse/dense vector, which will be used as the feature vector for the machine learning models.

All of the above transformations are stored in the stages list, which can then be passed to a Pipeline object for processing the entire dataset as shown below.



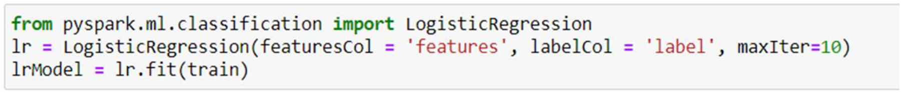
```
stages = []
for categoricalCol in cat_cols:
    stringIndexer = StringIndexer(inputCol = categoricalCol, outputCol = categoricalCol + 'Index')
    encoder = OneHotEncoderEstimator(inputCols=[stringIndexer.getOutputCol()], outputCols=[categoricalCol + "classVec"])
    stages += [stringIndexer, encoder]
label_stringIdx = StringIndexer(inputCol = 'status', outputCol = 'label')
stages += [label_stringIdx]
assemblerInputs = [c + "classVec" for c in cat_cols] + num_cols
assembler = VectorAssembler(inputCols=assemblerInputs, outputCol="features")
stages += [assembler]
```



```
from pyspark.ml import Pipeline
pipeline = Pipeline(stages = stages)
pipelineModel = pipeline.fit(df_pre_pro)
df = pipelineModel.transform(df_pre_pro)
selectedCols = ['label', 'features'] + cols
df = df.select(selectedCols)
df.printSchema()
```

3. Model Building and training

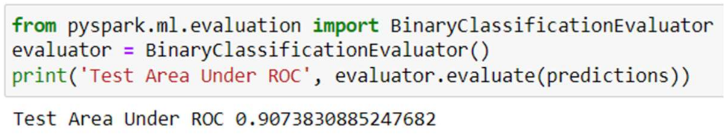
Logistic regression is a popular algorithm for binary classification problems. After applying the feature engineering techniques to the data, the model based on logistic regression trained on the transformed data to predict the label column 'status'.



```
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(featuresCol = 'features', labelCol = 'label', maxIter=10)
lrModel = lr.fit(train)
```

4. Model Evaluation

By using BinaryClassificationEvaluator class from the pyspark.ml.evaluation module, creates an instance of it, and uses it to evaluate the performance of a model by computing the area under the receiver operating characteristic (ROC) curve for binary classification problems.



```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
evaluator = BinaryClassificationEvaluator()
print('Test Area Under ROC', evaluator.evaluate(predictions))
```

```
Test Area Under ROC 0.9073830885247682
```

Overall, the performance of a binary classification model using the area under ROC metric is 90.73% and accuracy is 88.90%.

8. Conclusion

In conclusion, the Order Cancellation Prediction project for E-commerce business using Machine Learning is an effective solution to address the problem of order cancellations, which is a common issue in the e-commerce industry. By analyzing customer and order data, the model can accurately predict the likelihood of order cancellations, providing businesses with valuable insights to take proactive measures to minimize the impact of cancellations on their operations.

The project involved the use of various machine learning techniques, including data preprocessing, feature engineering, model selection, and hyperparameter tuning, to build an accurate and reliable prediction model. The model was trained and evaluated using real-world data and achieved high accuracy and precision in predicting order cancellations.

Overall, this project provides significant benefits to e-commerce businesses by enabling them to reduce the impact of order cancellations, and increase revenue. Moreover, the techniques and methodologies used in this project can be adapted and extended to solve similar problems in other industries as well.