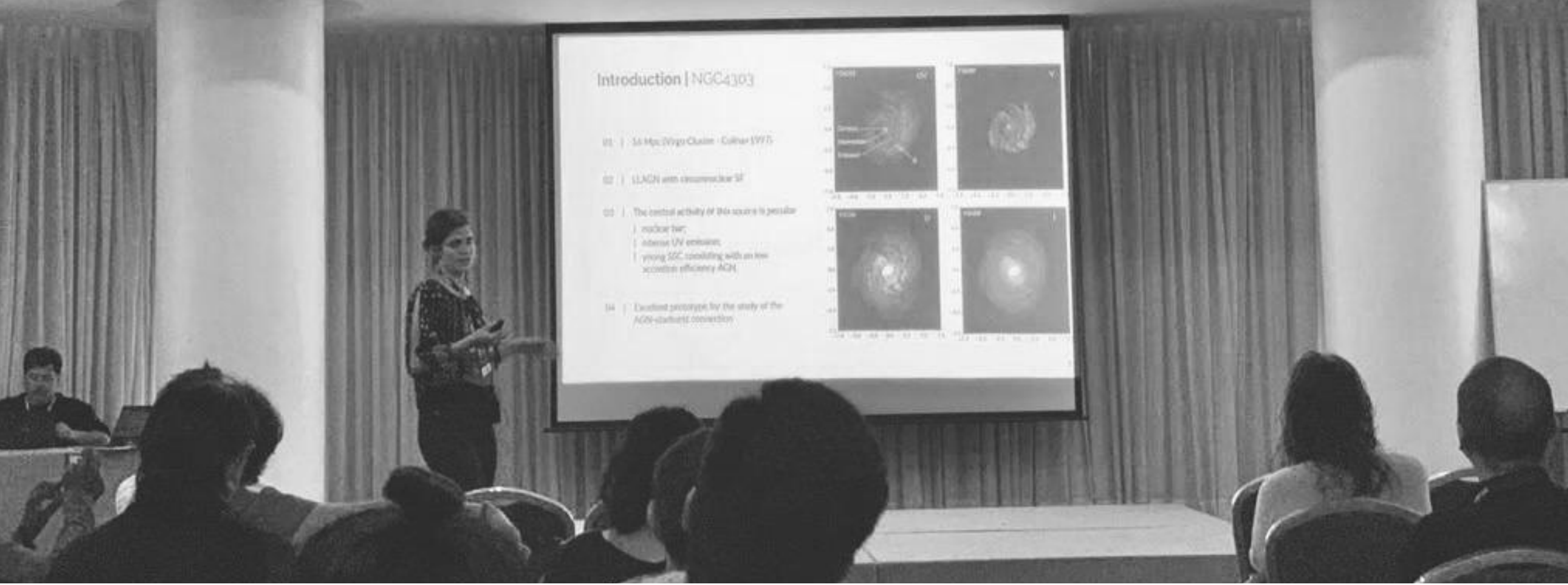


Natacha Zanon Dametto

PhD in Astrophysics | Data Scientist | Problem-Solving | Fast-Learner | Polyglot | Communication Skills

📍 Greater Bergamo Region

Introduction.

Study Case.

Dataset

- **quotations** requested by Esprinet customers.

Objectives

- to analyse the dataset and extract meaningful **insights**;
- Communicate and **share** results;
- Develop a **model** for predicting the acceptance or rejection of a quotation.

Introduction.

Structured Plan

1. Data Acquisition.
2. Data Understanding.
3. Data Processing.
4. Modeling
5. Model Evaluation

Data Acquisition.

```
In [45]: import numpy as np
         import pandas as pd
         from ydata_profiling import ProfileReport
         from cleaning_data import load_clean_data
         import seaborn as sns
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.linear_model import LogisticRegression
         from sklearn.dummy import DummyClassifier
         from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve, roc_a
         from sklearn.pipeline import make_pipeline
         from sklearn.preprocessing import LabelEncoder
         from plots import generate_histogram_plot, plot_channelid_distribution
         from model import evaluate_classifier

         %matplotlib inline
```

```
In [2]: df = pd.read_csv("case_study_anonymized.csv", sep="|", encoding="latin")
```

Data Acquisition.

```
In [45]: import numpy as np
         import pandas as pd
         from ydata_profiling import ProfileReport
         from cleaning_data import load_clean_data
         import seaborn as sns
         from sklearn.model_selection import train_test_split
         from sklearn.preprocessing import StandardScaler
         from sklearn.linear_model import LogisticRegression
         from sklearn.dummy import DummyClassifier
         from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_auc_score, roc_curve, roc_a
         from sklearn.pipeline import make_pipeline
         from sklearn.preprocessing import LabelEncoder
         from plots import generate_histogram_plot, plot_channelid_distribution
         from model import evaluate_classifier

         %matplotlib inline
```

```
In [2]: df = pd.read_csv("case_study_anonymized.csv", sep="|", encoding="latin")
```

Data Understanding.

Step 2: Data Understanding

2.1 General Data Visualization (option 1)

In [3]:
```python
from profile_report import report
```

In [4]:
```python
report()
```

Summarize dataset: 100% 132/132 [00:13<00:00, 6.08it/s, Completed]

Generate report structure: 100% 1/1 [00:05<00:00, 5.72s/it]

Render HTML: 100% 1/1 [00:01<00:00, 1.76s/it]

Export report to file: 100% 1/1 [00:00<00:00, 66.34it/s]

Out[4]: 'dataset_report.html'

Data Understanding.

Step 2: Data Understanding

- Check the first few rows of data using `df.head()`.
- Use `df.info()` to get an overview of data types and missing values.
- Check basic statistics with `df.describe()` to understand numeric variables
- Explore the dataset to understand its structure and contents.
- Get the unique values of a variable.

```
unique_categories = df['CHANNELID'].unique()
```

- Use a simple, fast and efficient tool to overview the dataset

Before start analysing our data, it is important to understand what it is the dataset we are working with.

- **Dataset statistics**
- **Variable types**

Data Processing.

Step 3: Data Preprocessing

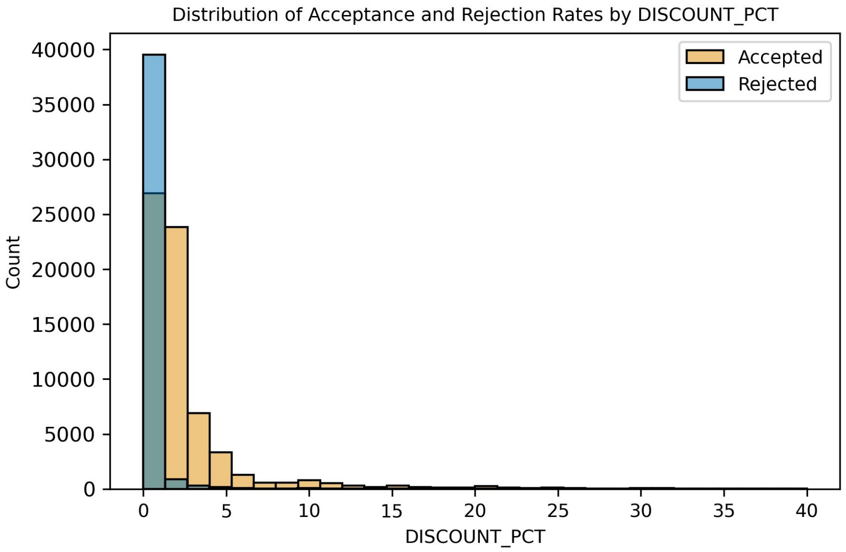
In [13]:
```
dataset = load_clean_data()
```

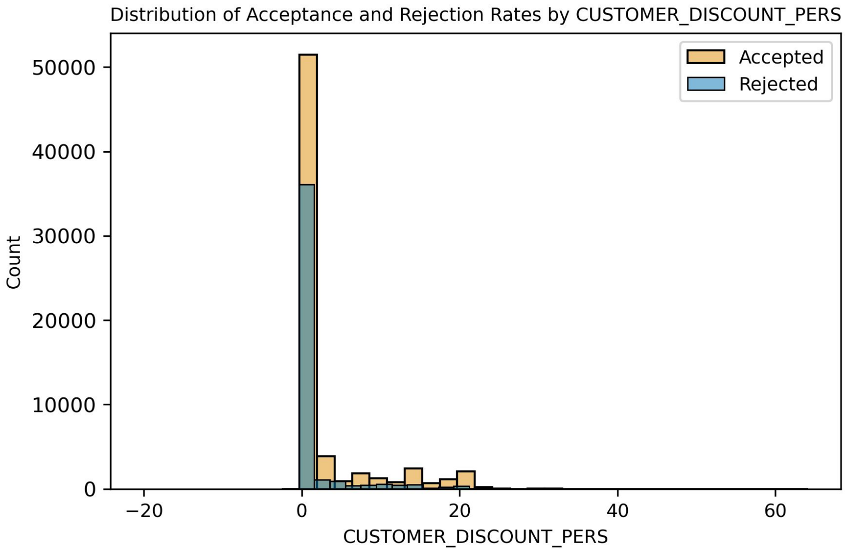
3.1. Handling missing data

In [14]:
```
missing_values = dataset.isnull().sum()

# Calculate the percentage of missing values
percentage_missing = (missing_values / len(df)) * 100

# Create a summary of missing values
missing_data_summary = pd.DataFrame({
    'Missing Values': missing_values,
    'Percentage Missing': percentage_missing
})

# Print the summary
print(missing_data_summary)
```

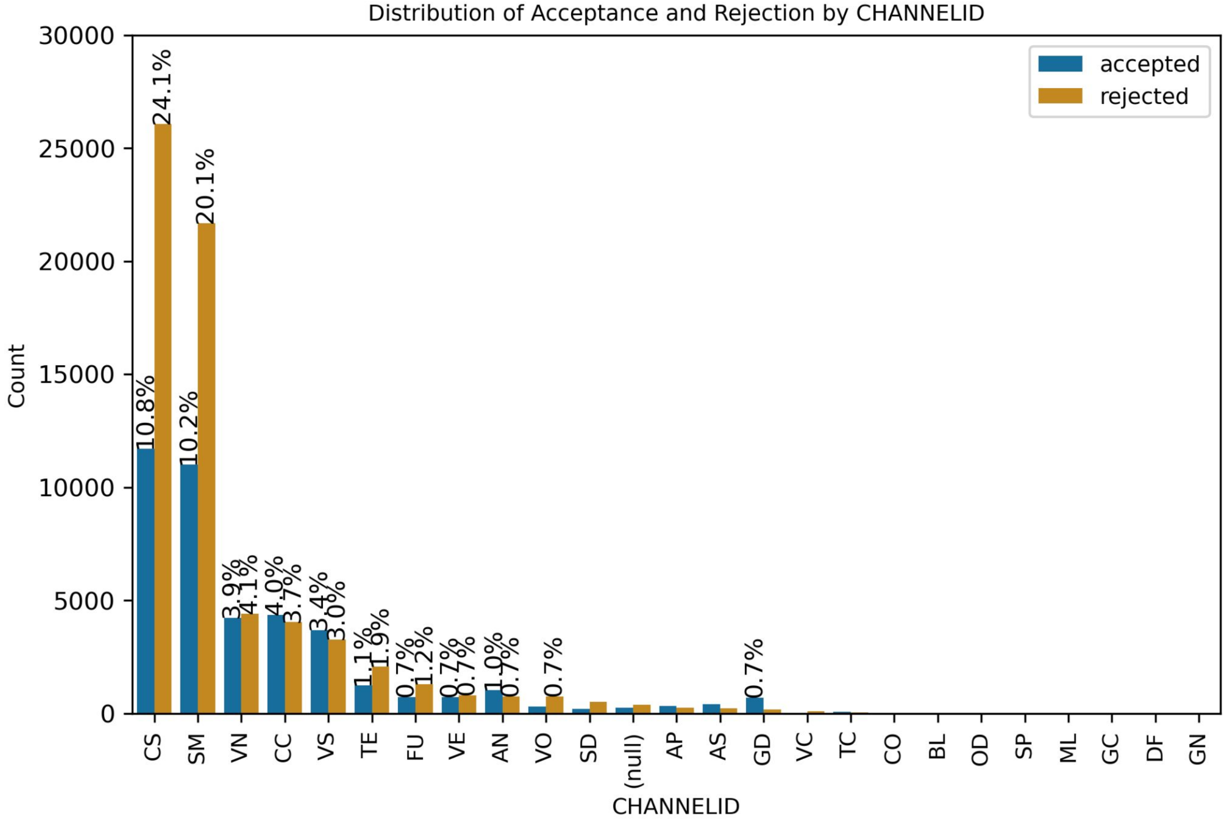
```
                                 Missing Values  Percentage Missing
Unnamed                                       0            0.000000
DEALKEY                                       0            0.000000
DEALDETKEY                                    0            0.000000
CUSTOMERID                                    0            0.000000
CHANNELID                                     0            0.000000
CHANNELDSC                                  650            0.601774
HAS_REAL_ORDER                                0            0.000000
DEALVALUE                                     0            0.000000
DEALQTY                                       0            0.000000
ORDER_PRICE                                   0            0.000000
ORIGINAL_PRICE                                0            0.000000
```

Insights.


Distribution of Acceptance and Rejection Rates by DISCOUNT_PCT
Accepted
Rejected
Count
DISCOUNT_PCT
0
5000
10000
15000
20000
25000
30000
35000
40000
0
5
10
15
20
25
30
35
40

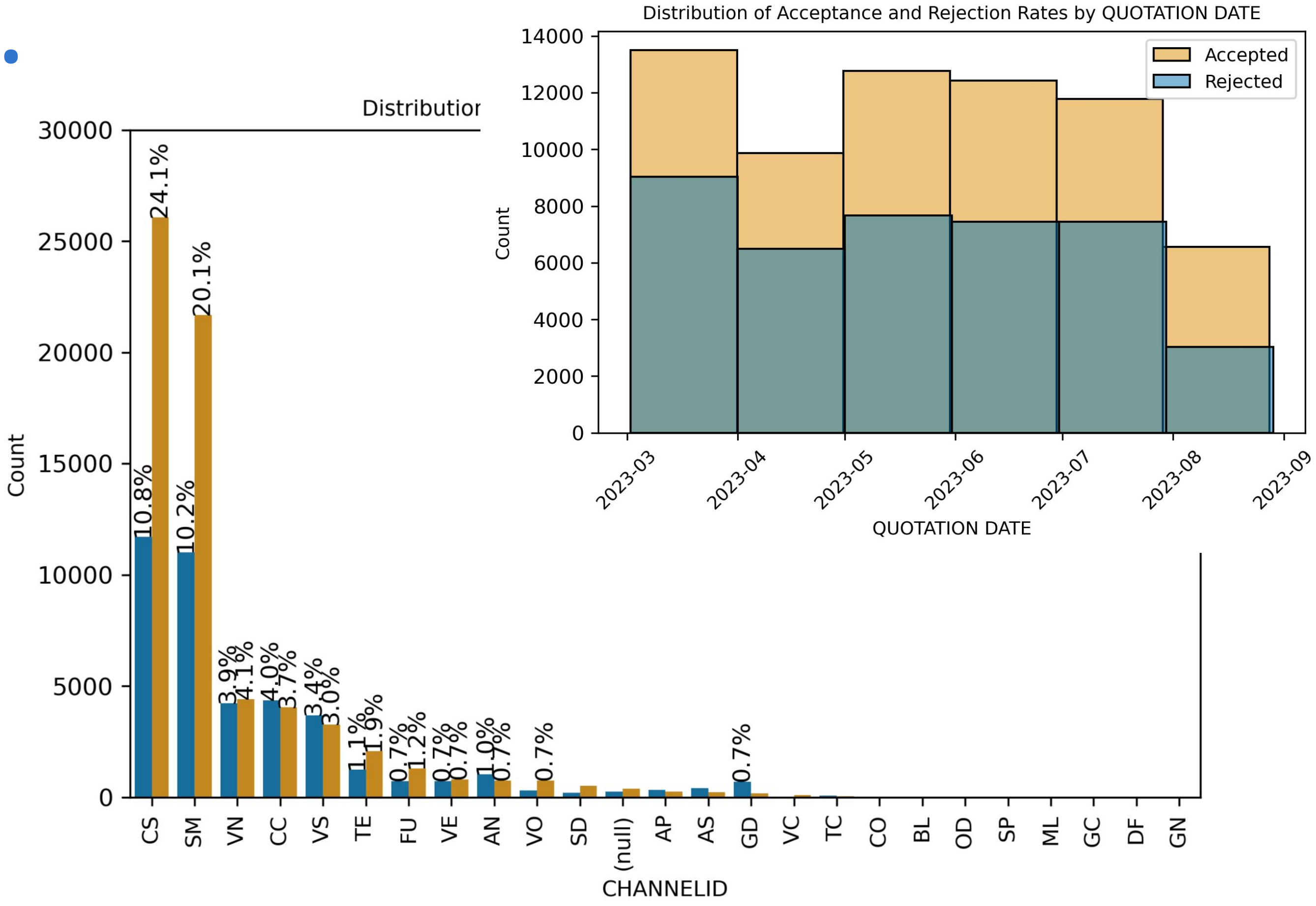

Distribution of Acceptance and Rejection Rates by CUSTOMER_DISCOUNT_PERS
Accepted
Rejected
Count
CUSTOMER_DISCOUNT_PERS
0
10000
20000
30000
40000
50000
−20
0
20
40
60

Insights.



Distribution of Acceptance and Rejection by CHANNELID
accepted
rejected
Count
CHANNELID
30000
25000
20000
15000
10000
5000
0
CS
SM
VN
CC
VS
TE
FU
VE
AN
VO
SD
(null)
AP
AS
GD
VC
TC
CO
BL
OD
SP
ML
GC
DF
GN
10.8%
24.1%
10.2%
20.1%
3.9%
4.1%
4.0%
3.7%
3.4%
3.0%
1.1%
1.9%
0.7%
1.2%
0.7%
0.7%
1.0%
0.7%
0.7%
0.7%

Insights.



Distributior
30000
25000
20000
15000
10000
5000
0
Count
24.1%
20.1%
10.8%
10.2%
3.9%
4.1%
4.0%
3.7%
3.4%
3.0%
1.1%
1.9%
0.7%
1.2%
0.7%
0.7%
1.0%
0.7%
0.7%
0.7%
CS
SM
VN
CC
VS
TE
FU
VE
AN
VO
SD
(null)
AP
AS
GD
VC
TC
CO
BL
OD
SP
ML
GC
DF
GN
CHANNELID
Distribution of Acceptance and Rejection Rates by QUOTATION DATE
Accepted
Rejected
14000
12000
10000
8000
6000
4000
2000
0
Count
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
2023-09
QUOTATION DATE

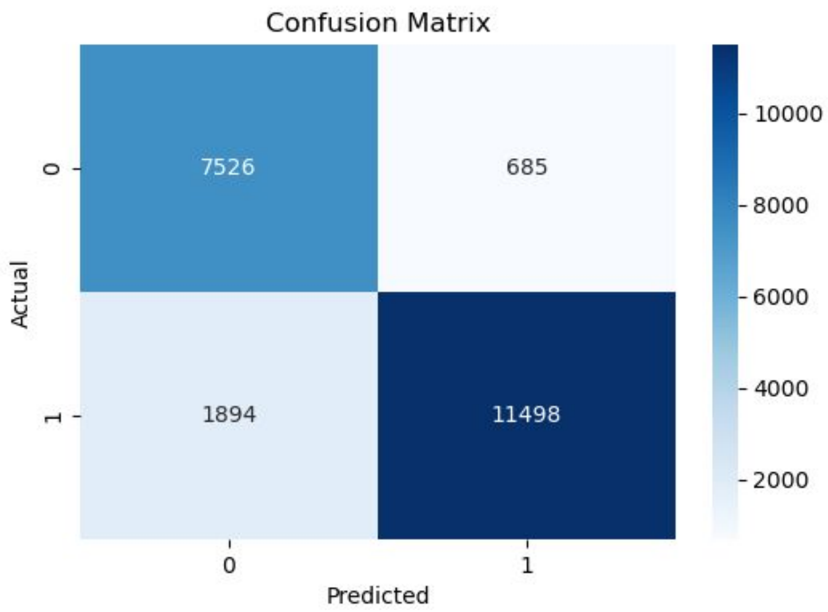
Model.

```python
def evaluate_classifier(dataset, classifier, save=True):
    """
    Evaluate a logistic regression model on the given dataset.

    Parameters:
    - dataset: The dataset containing features and the target variable.
    - classifier: The classifier to evaluate (LogisticRegression or DummyClassifier).
    - save: Whether to save the plots as image files.

    Returns:
    - A dictionary with evaluation metrics including accuracy, classification report, confusion matrix, AUC score,
      and plots for confusion matrix, ROC curve, precision-recall curve, and feature importance (for Logistic Regre
    """
    # Split data into features (X) and target (y)
    exclude_columns = ['HAS_REAL_ORDER', 'Unnamed','DEALKEY', 'DEALDETKEY', 'CHANNELDSC', 'CATEGORYDSC', 'ARTDSC',
    X = dataset.drop(columns=exclude_columns)
    y = dataset['HAS_REAL_ORDER']

    # Split data into training and testing sets (e.g., 80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Initialize the label encoder
    label_encoder = LabelEncoder()

    # Fit the label encoder on your target variable and transform it
    y_train_encoded = label_encoder.fit_transform(y_train)
    y_test_encoded = label_encoder.transform(y_test)

    if classifier == 'LogisticRegression':
        # Create a pipeline with data scaling and logistic regression
        pipe = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000))
    elif classifier == 'DummyClassifier':
        # Create a DummyClassifier
        pipe = DummyClassifier(strategy='most_frequent')  # You can change the strategy here
    else:
        raise ValueError("Invalid classifier. Supported classifiers are 'LogisticRegression' and 'DummyClassifier'.

    # Fit the model on the training data
    pipe.fit(X_train, y_train_encoded)

    # Make predictions on the testing data
    y_pred = pipe.predict(X_test)

    # Calculate evaluation metrics
    accuracy = accuracy_score(y_test_encoded, y_pred)
    report = classification_report(y_test_encoded, y_pred)
    confusion = confusion_matrix(y_test_encoded, y_pred)
    auc_score = roc_auc_score(y_test_encoded, y_pred)
```

Model.

```
def evaluate_classifier(dataset, classifier, save=True):
    """
    Evaluate a logistic regression model on the given dataset.

    Parameters:
    - dataset: The dataset containing features and the target variable.
    - classifier: The classifier to evaluate (LogisticRegression or DummyClassifier).
    - save: Whether to save the plots as image files.

    Returns:
    - A dictionary with evaluation metrics including accuracy, classification report, confusion matrix, AUC score,
      and plots for confusion matrix, ROC curve, precision-recall curve, and feature importance (for Logistic Regre
    """
    # Split data into features (X) and target (y)
    exclude_columns = ['HAS_REAL_ORDER', 'Unnamed','DEALKEY', 'DEALDETKEY', 'CHANNELDSC', 'CATEGORYDSC', 'ARTDSC',
    X = dataset.drop(columns=exclude_columns)
    y = dataset['HAS_REAL_ORDER']

    # Split data into training and testing sets (e.g., 80% train, 20% test)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    # Initialize the label encoder
    label_encoder = LabelEncoder()

    # Fit the label encoder on your target variable and transform it
    y_train_encoded = label_encoder.fit_transform(y_train)
    y_test_encoded = label_encoder.transform(y_test)

    if classifier == 'LogisticRegression':
        # Create a pipeline with data scaling and logistic regression
        pipe = make_pipeline(StandardScaler(), LogisticRegression(max_iter=1000))
    elif classifier == 'DummyClassifier':
        # Create a DummyClassifier
        pipe = DummyClassifier(strategy='most_frequent')  # You can change the strategy here
    else:
        raise ValueError("Invalid classifier. Supported classifiers are 'LogisticRegression' and 'DummyClassifier'.

    # Fit the model on the training data
    pipe.fit(X_train, y_train_encoded)

    # Make predictions on the testing data
    y_pred = pipe.predict(X_test)

    # Calculate evaluation metrics
    accuracy = accuracy_score(y_test_encoded, y_pred)
    report = classification_report(y_test_encoded, y_pred)
    confusion = confusion_matrix(y_test_encoded, y_pred)
    auc_score = roc_auc_score(y_test_encoded, y_pred)
```

Model Evaluation.



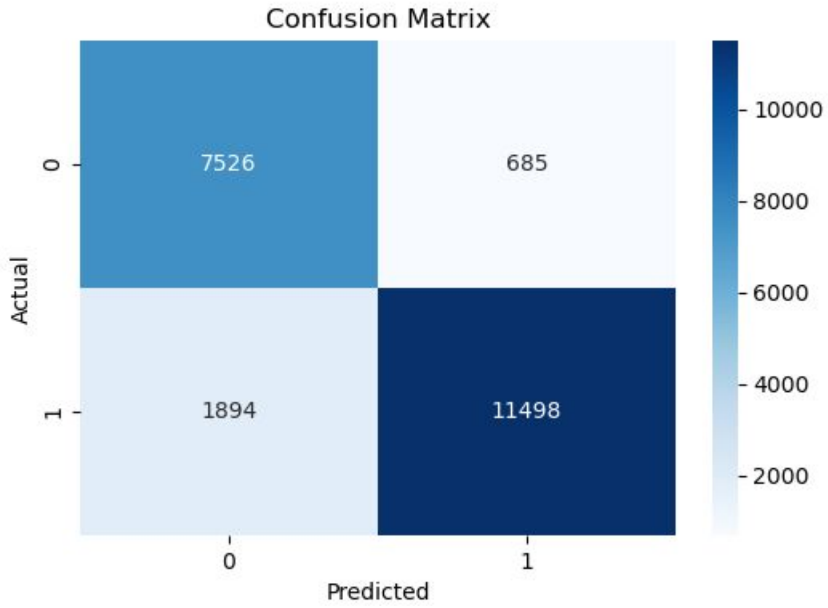
Logistic Regression

Model Evaluation.

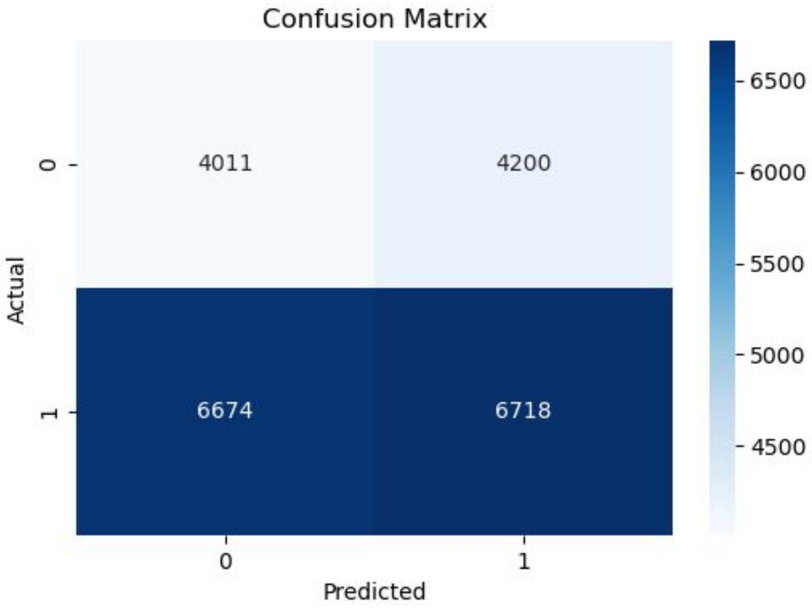
```
Metrics for LogisticRegression:
Accuracy: 0.88
Classification Report:               precision    recall  f1-score   support

           0       0.80      0.92      0.85      8211
           1       0.94      0.86      0.90     13392

    accuracy                           0.88     21603
   macro avg       0.87      0.89      0.88     21603
weighted avg       0.89      0.88      0.88     21603

Confusion Matrix: [[ 7526   685]
 [ 1894 11498]]
AUC Score: 0.89
```

```
Metrics for DummyClassifier:
Accuracy: 0.50
Classification Report:               precision    recall  f1-score   support

           0       0.38      0.49      0.42      8211
           1       0.62      0.50      0.55     13392

    accuracy                           0.50     21603
   macro avg       0.50      0.50      0.49     21603
weighted avg       0.52      0.50      0.50     21603

Confusion Matrix: [[4011 4200]
 [6674 6718]]
AUC Score: 0.50
```

Model Evaluation.

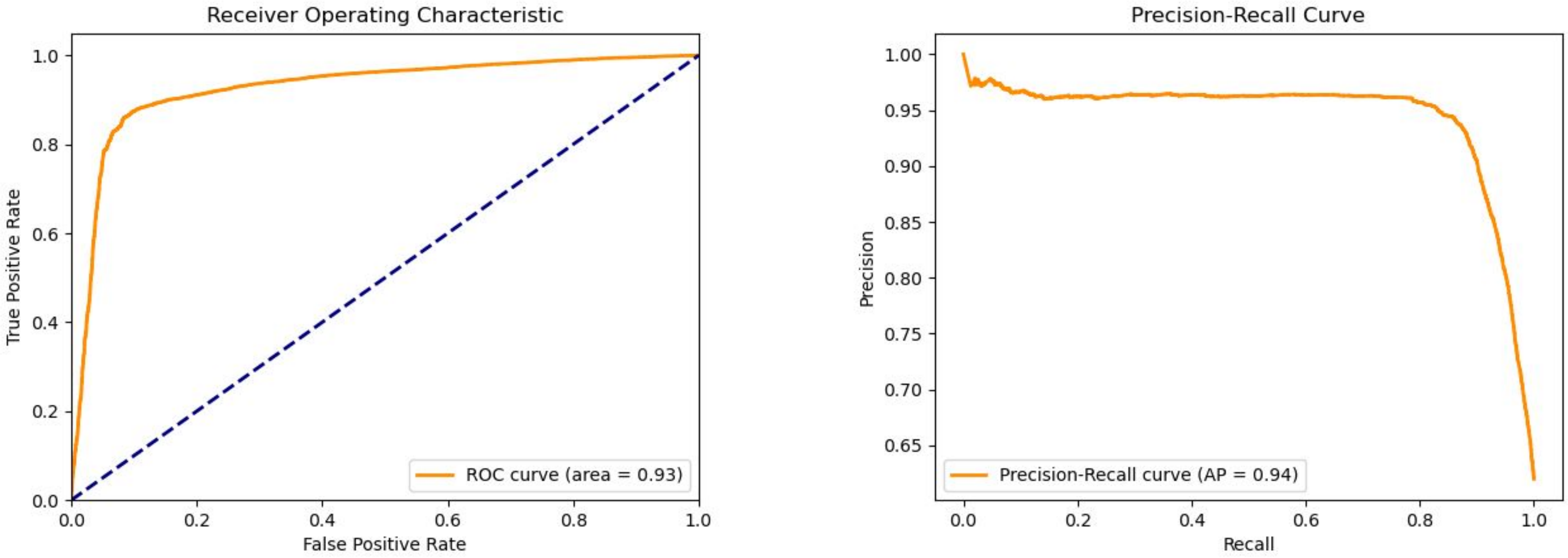


Logistic Regression

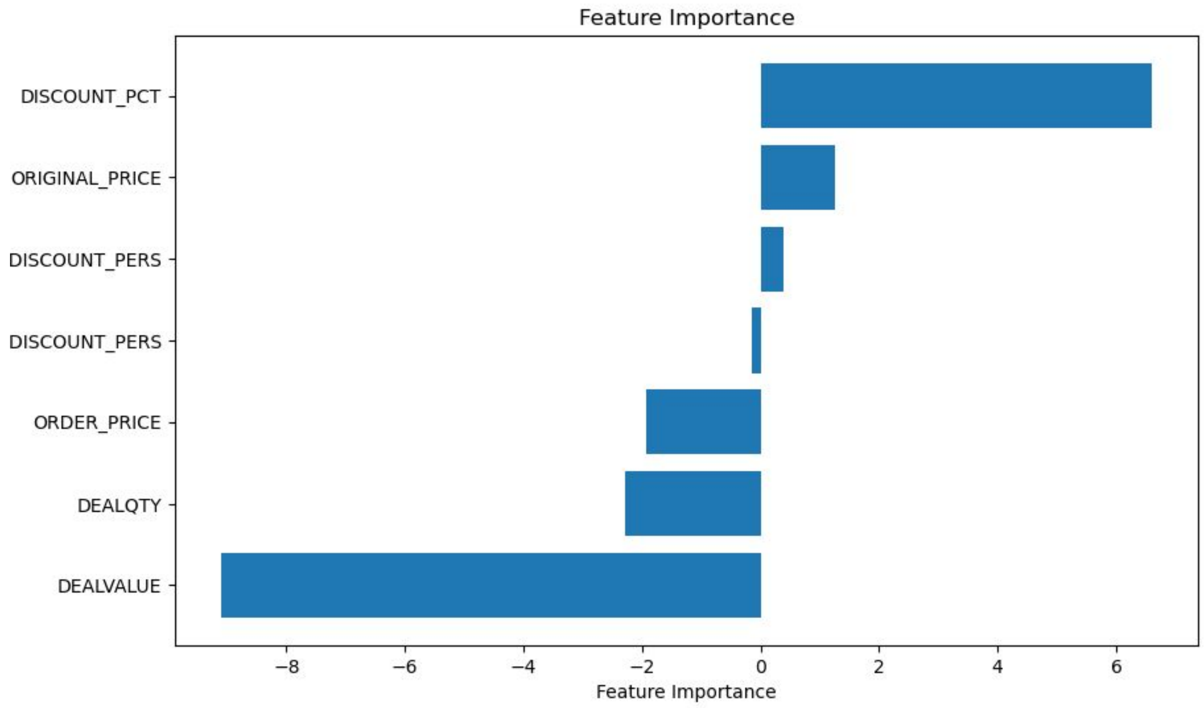


Dummy CLassifier

Model Evaluation.



Model Evaluation.



Thank you for your attention.