ibimbing

Study Case Business Intelligence

Customer Churn Analysis & Lifetime Value Optimization

Presented by : Anniza Mega

Information available in audio.

Agenda

About Me

- Hello everyone, introducing my name Anniza Mega, a student in Dibimbing.id Business Intelligence batch 9. I am happy to share my project.
- In today's competitive market, understanding customer behavior and optimizing customer relationships are crucial for business success. This Business Intelligence (BI) project focuses on two essential aspects: Customer Churn Analysis and Customer Lifetime Value (CLV) Optimization. The project aims to leverage data analytics to identify at-risk customers and develop strategies to retain them while maximizing their long-term value to the company.

Back to Agenda

Introduction

Customer churn is a term used to describe the loss of customers from a business or service over a period of time. Churn can be caused by a variety of factors, including customer dissatisfaction, high prices, or better offers from competitors. C

Customer Lifetime Value (CLV) is an estimate of the total monetary value that results from the relationship between a business and a customer over a period of time. By understanding CLV, businesses can allocate resources more effectively, improve customer retention, and increase long-term profits.

Customer Churn Analysis

Goal # 1

Understanding Project

Goal # 2

Handling and Processing Data

Goal # 3

Model Evaluation and Conclusion

Back to Agenda

Project Objective Churn Analysis

The objective of the customer churn analysis is to develop accurate predictive models to identify customers at risk of churning. By understanding the factors that lead to churn, the project aims to:

1. **Reduce Churn Rates:** Implement strategies to retain customers identified as likely to churn, thereby reducing overall churn rates.
2. **Improve Customer Retention:** Enhance customer retention efforts by targeting at-risk customers with tailored interventions.
3. **Optimize Marketing Efforts:** Allocate marketing resources more efficiently to focus on retaining valuable customers.
4. **Increase Profitability:** By reducing churn, increase the overall profitability and lifetime value of the customer base.

Business Problem Churn Analysis

The business problem addressed by the customer churn analysis is the loss of revenue and profitability due to high churn rates. Specific challenges include:

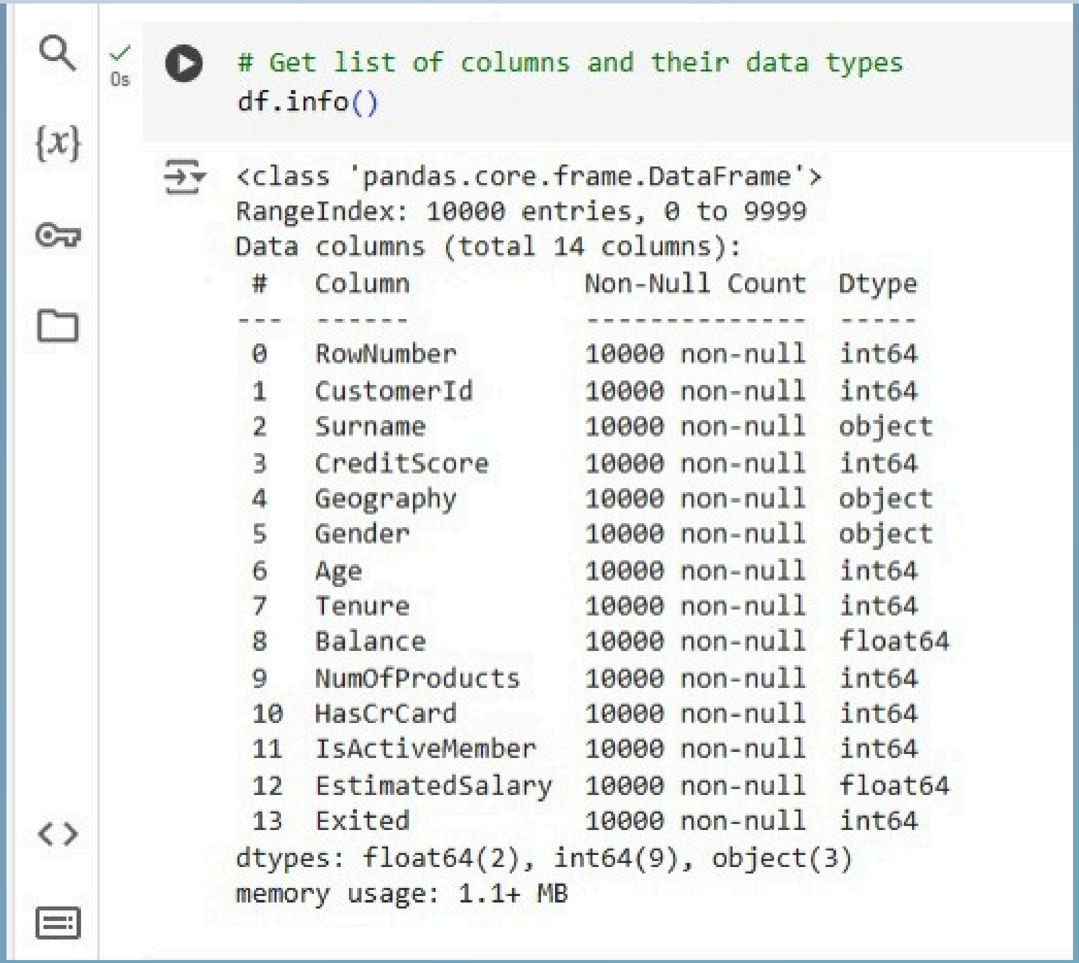
1. **Identifying At-Risk Customers:** Difficulty in accurately identifying which customers are likely to churn, leading to ineffective retention efforts.
2. **Resource Allocation:** Inefficient use of marketing and customer service resources due to a lack of targeted strategies for at-risk customers.
3. **Customer Satisfaction:** Decreased customer satisfaction and loyalty as a result of not addressing the underlying reasons for churn.
4. **Revenue Loss:** Significant revenue loss from customers who leave, impacting the company's financial performance.

Data Overview

This dataset contains information on a bank's customers, which is used to analyse the factors that influence a customer's decision to leave the bank (churn). Content :

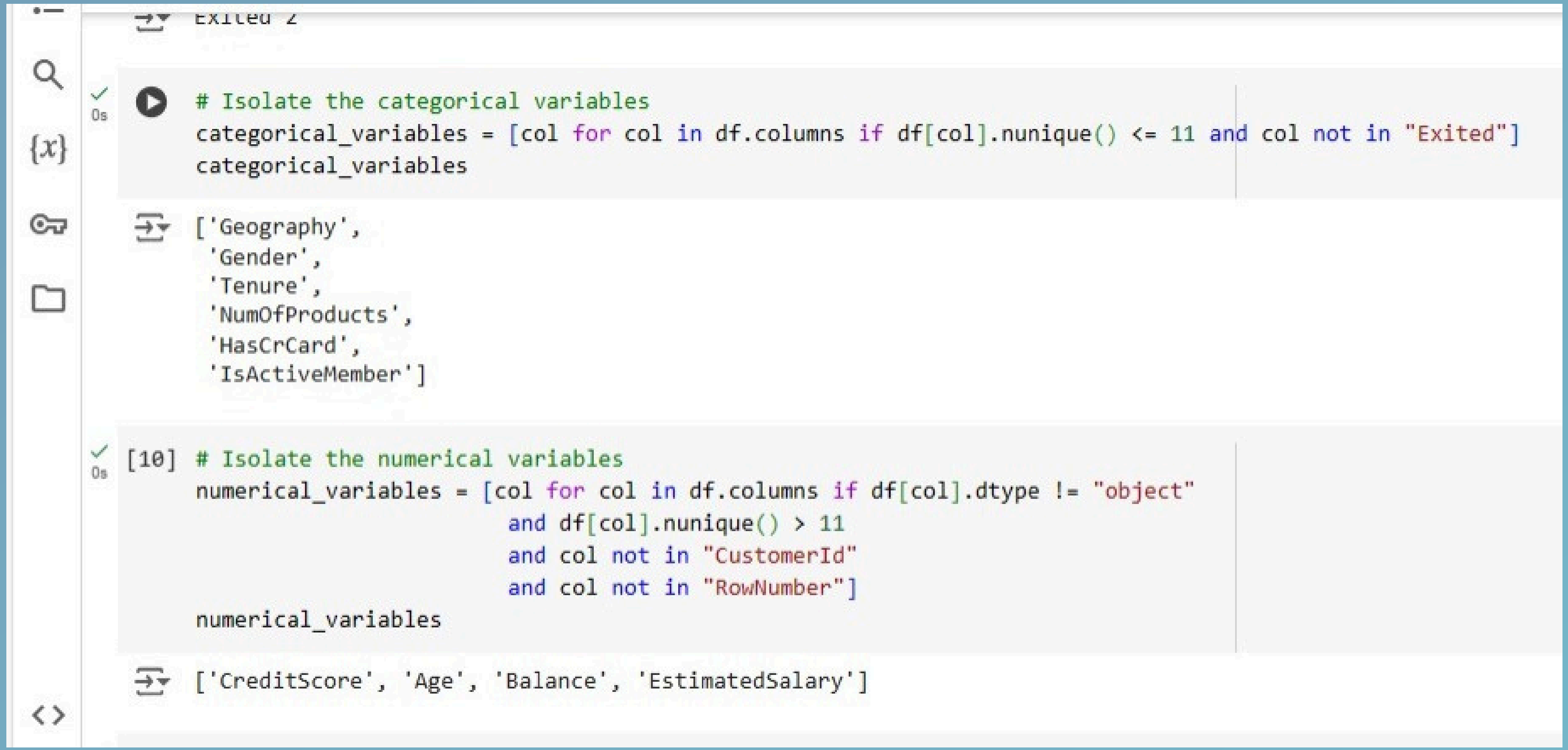
- RowNumber—corresponds to the record (row) number and has no effect on the output.
- CustomerId—contains random values and has no effect on customer leaving the bank.
- Surname—the surname of a customer has no impact on their decision to leave the bank.
- CreditScore—can have an effect on customer churn, since a customer with a higher credit score is less likely to leave the bank.
- Geography—a customer’s location can affect their decision to leave the bank.
- Gender—it’s interesting to explore whether gender plays a role in a customer leaving the bank.
- Age—this is certainly relevant, since older customers are less likely to leave their bank than younger ones.
- Tenure—refers to the number of years that the customer has been a client of the bank. Normally, older clients are more loyal and less
- likely to leave a bank.
- Balance—also a very good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave the
- bank compared to those with lower balances.
- NumOfProducts—refers to the number of products that a customer has purchased through the bank.
- HasCrCard—denotes whether or not a customer has a credit card. This column is also relevant, since people with a credit card are less
- likely to leave the bank.
- IsActiveMember—active customers are less likely to leave the bank.
- EstimatedSalary—as with balance, people with lower salaries are more likely to leave the bank compared to those with higher salaries.
- Exited—whether or not the customer left the bank.

EDA



```
# Get list of columns and their data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   RowNumber        10000 non-null  int64
 1   CustomerId       10000 non-null  int64
 2   Surname          10000 non-null  object
 3   CreditScore      10000 non-null  int64
 4   Geography        10000 non-null  object
 5   Gender           10000 non-null  object
 6   Age              10000 non-null  int64
 7   Tenure           10000 non-null  int64
 8   Balance          10000 non-null  float64
 9   NumOfProducts    10000 non-null  int64
 10  HasCrCard        10000 non-null  int64
 11  IsActiveMember   10000 non-null  int64
 12  EstimatedSalary  10000 non-null  float64
 13  Exited           10000 non-null  int64
dtypes: float64(2), int64(9), object(3)
memory usage: 1.1+ MB
```



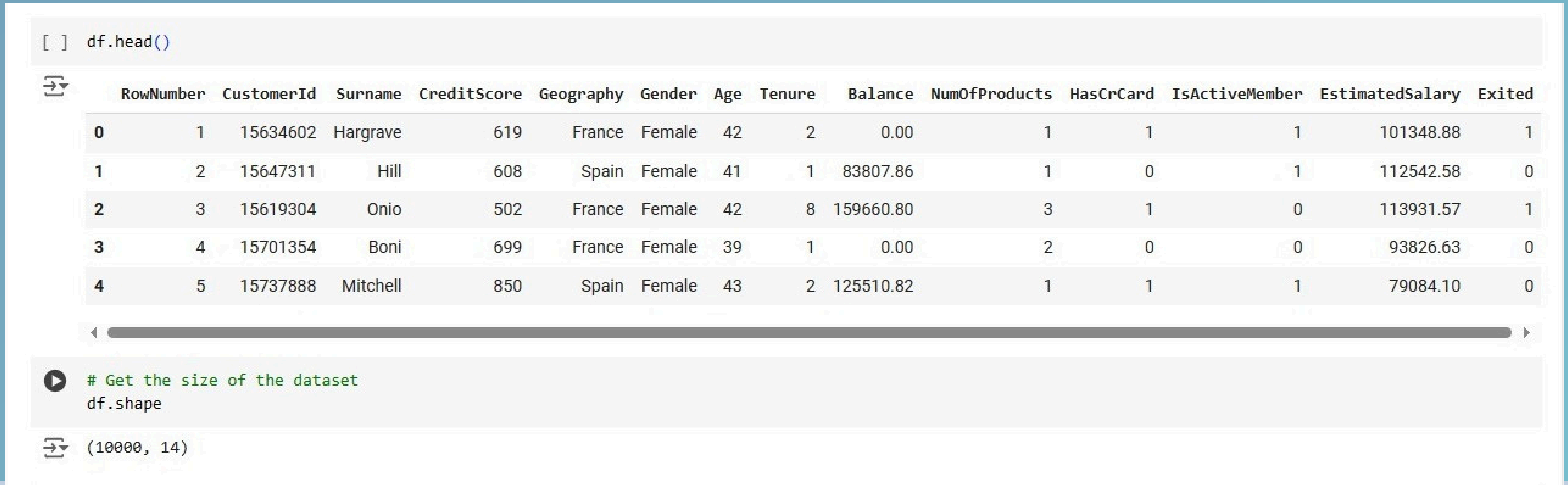
```
Exited 2
```

```
# Isolate the categorical variables
categorical_variables = [col for col in df.columns if df[col].nunique() <= 11 and col not in "Exited"]
categorical_variables
```

```
['Geography',
 'Gender',
 'Tenure',
 'NumOfProducts',
 'HasCrCard',
 'IsActiveMember']
```

```
[10] # Isolate the numerical variables
numerical_variables = [col for col in df.columns if df[col].dtype != "object"
                         and df[col].nunique() > 11
                         and col not in "CustomerId"
                         and col not in "RowNumber"]
numerical_variables
```

```
['CreditScore', 'Age', 'Balance', 'EstimatedSalary']
```



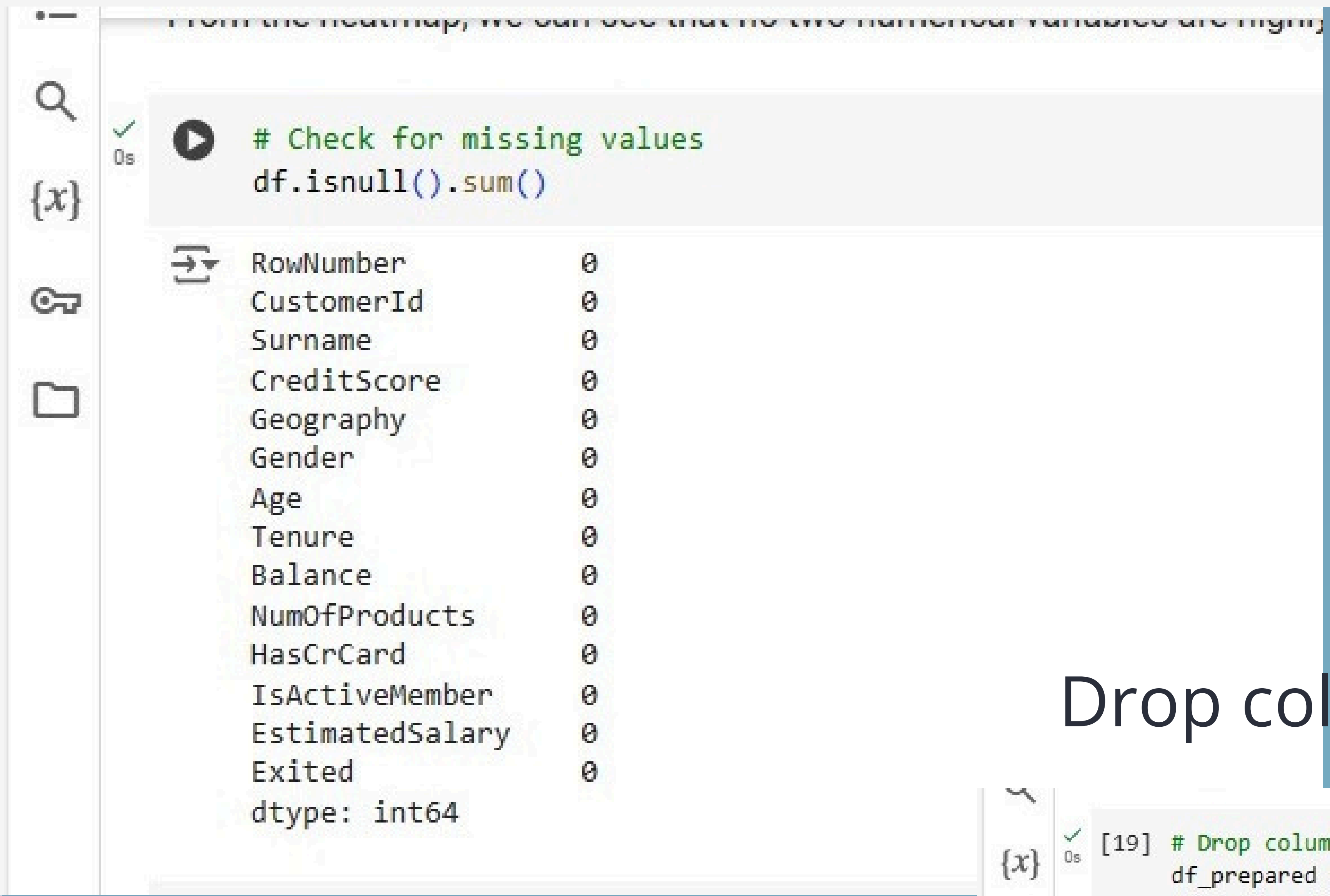
```
[ ] df.head()
```

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	1	15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	2	15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	3	15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	4	15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

```
# Get the size of the dataset
df.shape
```

```
(10000, 14)
```

Data Preprocessing



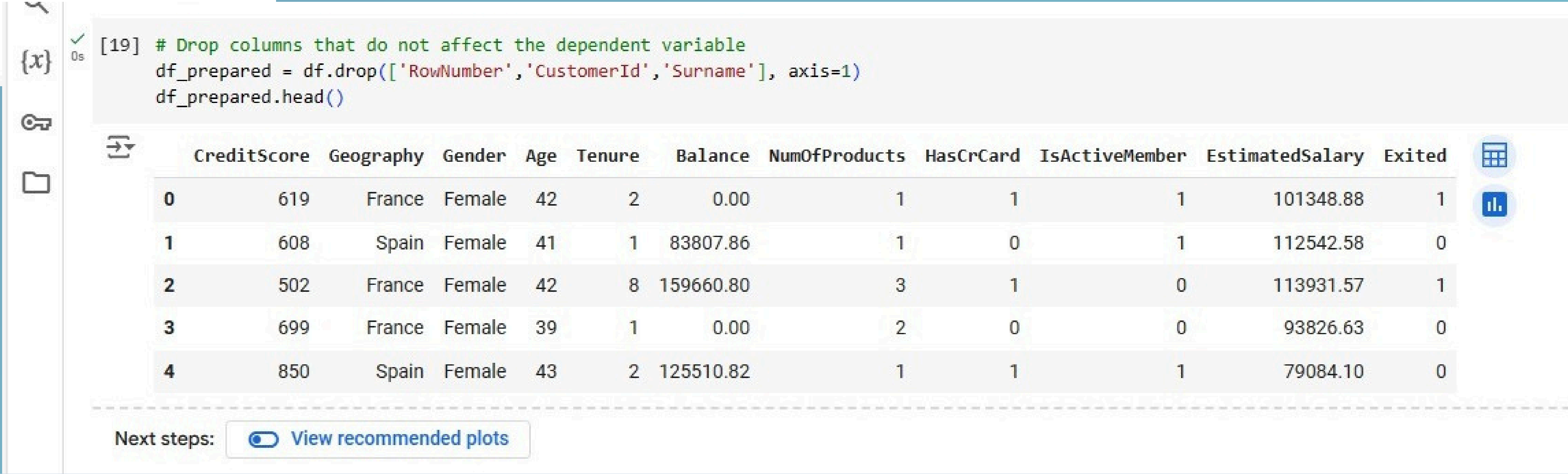
From the heatmap, we can see that no two numerical variables are highly

```python
# Check for missing values
df.isnull().sum()
```

```
RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

Check missing value

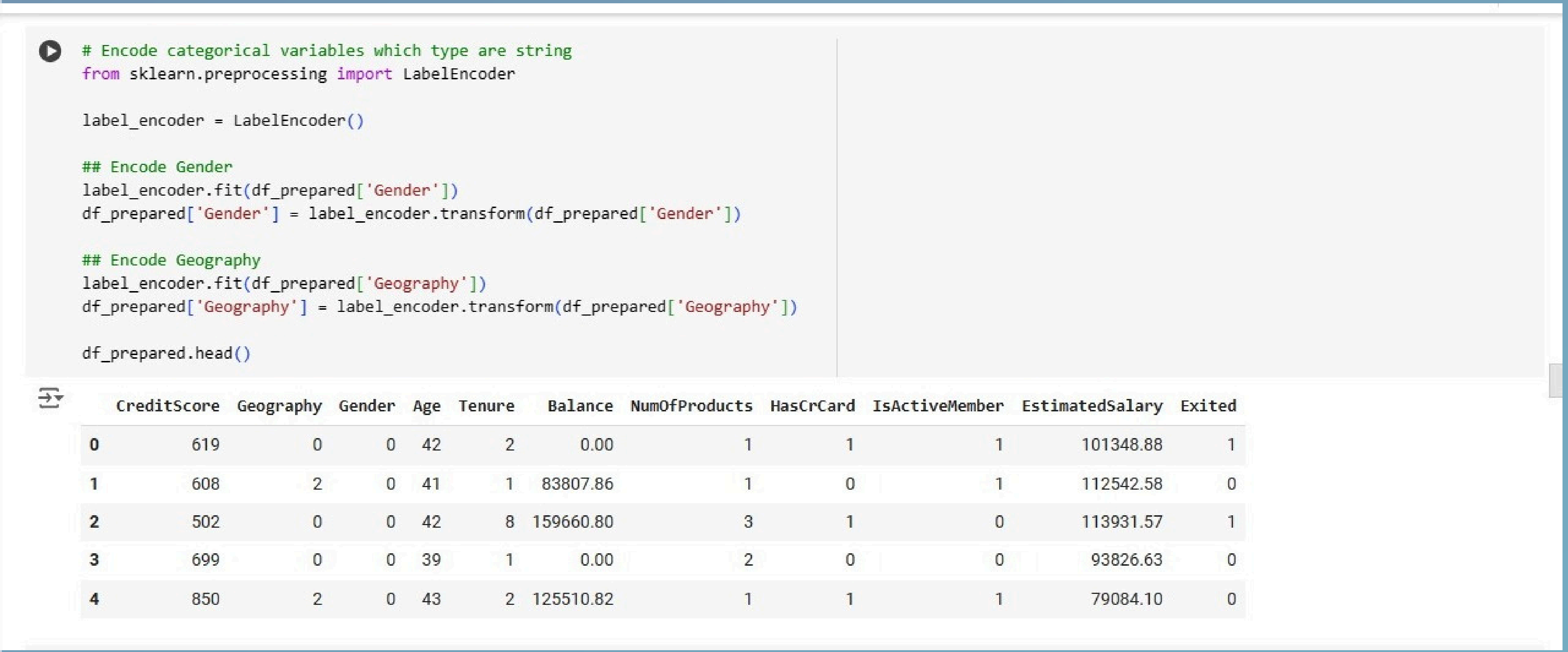
Drop columns that do not affect the dependent variable



```python
[19] # Drop columns that do not affect the dependent variable
     df_prepared = df.drop(['RowNumber','CustomerId','Surname'], axis=1)
     df_prepared.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	France	Female	42	2	0.00	1	1	1	101348.88	1
1	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
2	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
3	699	France	Female	39	1	0.00	2	0	0	93826.63	0
4	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

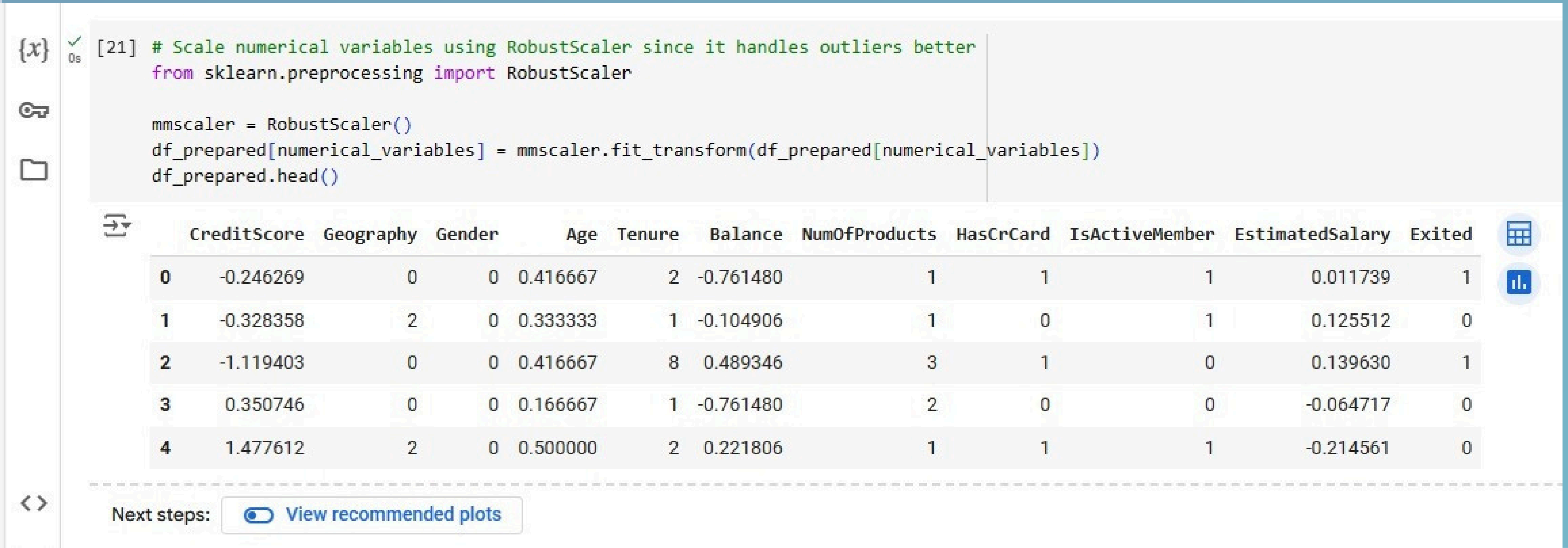
Feature Engineering



```python
# Encode categorical variables which type are string
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

## Encode Gender
label_encoder.fit(df_prepared['Gender'])
df_prepared['Gender'] = label_encoder.transform(df_prepared['Gender'])

## Encode Geography
label_encoder.fit(df_prepared['Geography'])
df_prepared['Geography'] = label_encoder.transform(df_prepared['Geography'])

df_prepared.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	619	0	0	42	2	0.00	1	1	1	101348.88	1
1	608	2	0	41	1	83807.86	1	0	1	112542.58	0
2	502	0	0	42	8	159660.80	3	1	0	113931.57	1
3	699	0	0	39	1	0.00	2	0	0	93826.63	0
4	850	2	0	43	2	125510.82	1	1	1	79084.10	0

This code converts the ‘Gender’ and ‘Geography’ columns that originally contained categorical string data into numeric values. This is important because many machine learning algorithms require numerical input to function properly



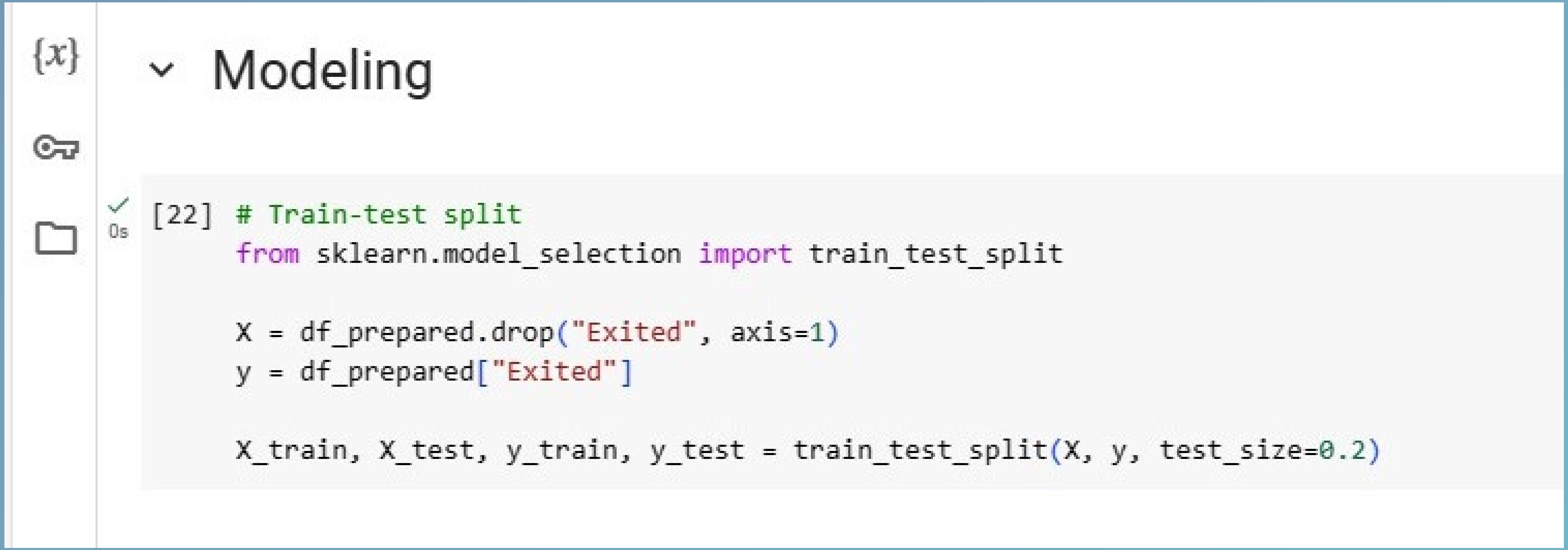
```python
[21] # Scale numerical variables using RobustScaler since it handles outliers better
     from sklearn.preprocessing import RobustScaler

     mmscaler = RobustScaler()
     df_prepared[numerical_variables] = mmscaler.fit_transform(df_prepared[numerical_variables])
     df_prepared.head()
```

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	-0.246269	0	0	0.416667	2	-0.761480	1	1	1	0.011739	1
1	-0.328358	2	0	0.333333	1	-0.104906	1	0	1	0.125512	0
2	-1.119403	0	0	0.416667	8	0.489346	3	1	0	0.139630	1
3	0.350746	0	0	0.166667	1	-0.761480	2	0	0	-0.064717	0
4	1.477612	2	0	0.500000	2	0.221806	1	1	1	-0.214561	0

Next steps: View recommended plots

It converts the numeric columns in a ‘dataframe into a scale that is more stable against outliers, using the median and IQR to normalize the data. ‘RobustScaler’ scales by subtracting the median and then dividing by the IQR, which makes it more resistant to extreme values compared to scales that only use the mean and standard deviation.

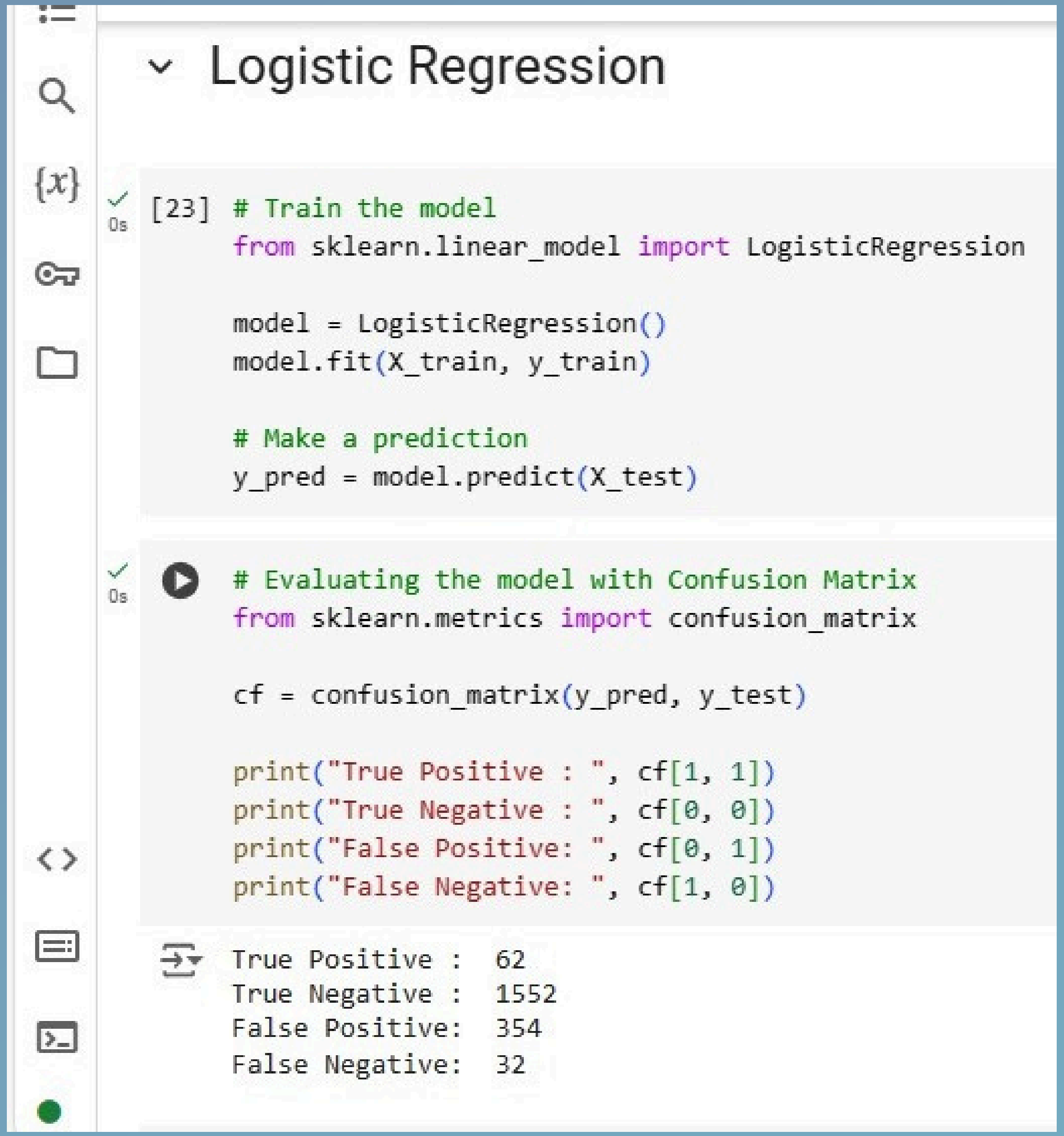
Modeling



```
[22] # Train-test split
     from sklearn.model_selection import train_test_split

     X = df_prepared.drop("Exited", axis=1)
     y = df_prepared["Exited"]

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

This code prepares the data for the machine learning process by separating features and targets, then splitting the data into two subsets, one for training and one for testing. This division is important to ensure that the trained model can be evaluated with data that has not been seen before, thus providing a more accurate picture of the model's performance on new data.

Logistic Regression

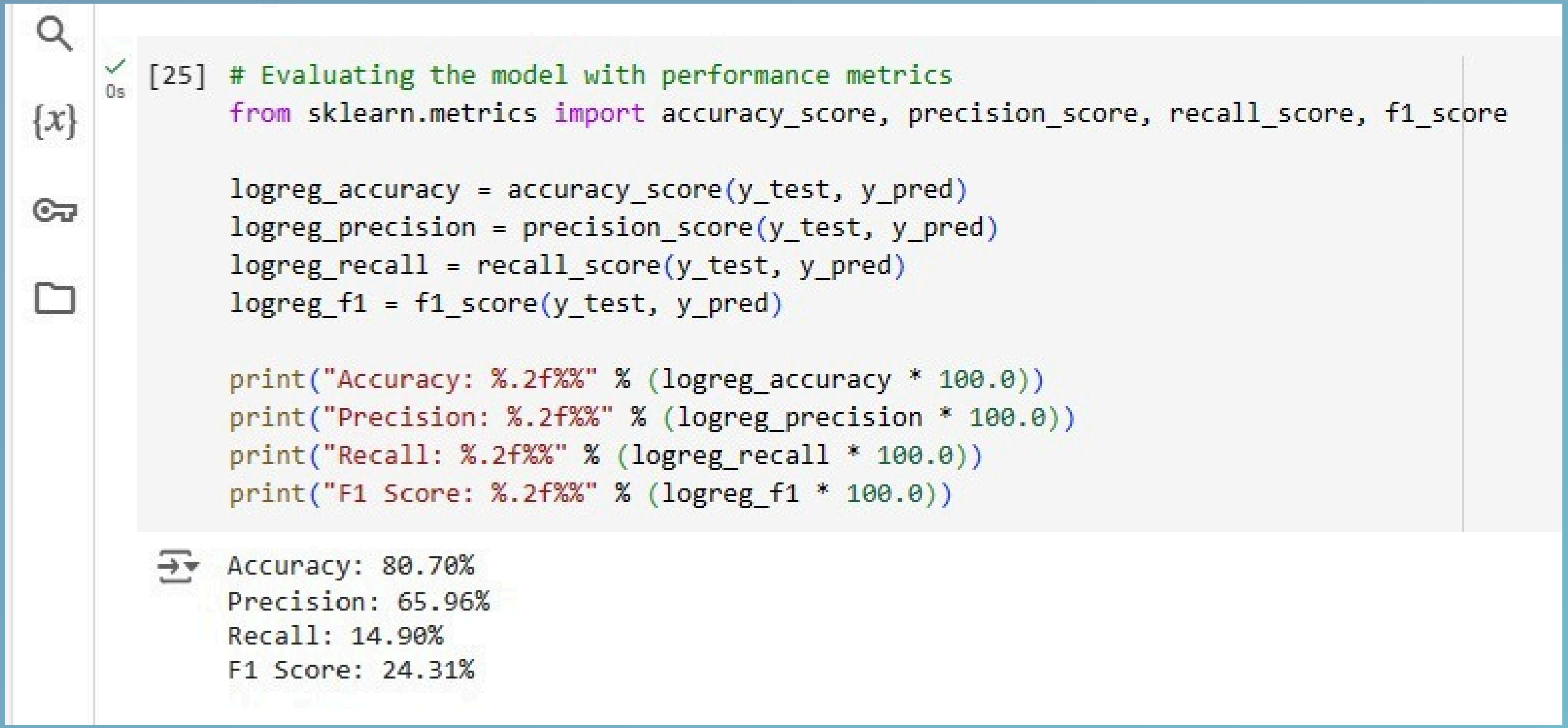


Logistic Regression

```
[23] # Train the model
     from sklearn.linear_model import LogisticRegression

     model = LogisticRegression()
     model.fit(X_train, y_train)

     # Make a prediction
     y_pred = model.predict(X_test)
```

```
# Evaluating the model with Confusion Matrix
from sklearn.metrics import confusion_matrix

cf = confusion_matrix(y_pred, y_test)

print("True Positive : ", cf[1, 1])
print("True Negative : ", cf[0, 0])
print("False Positive: ", cf[0, 1])
print("False Negative: ", cf[1, 0])
```

```
True Positive :  62
True Negative :  1552
False Positive:  354
False Negative:  32
```

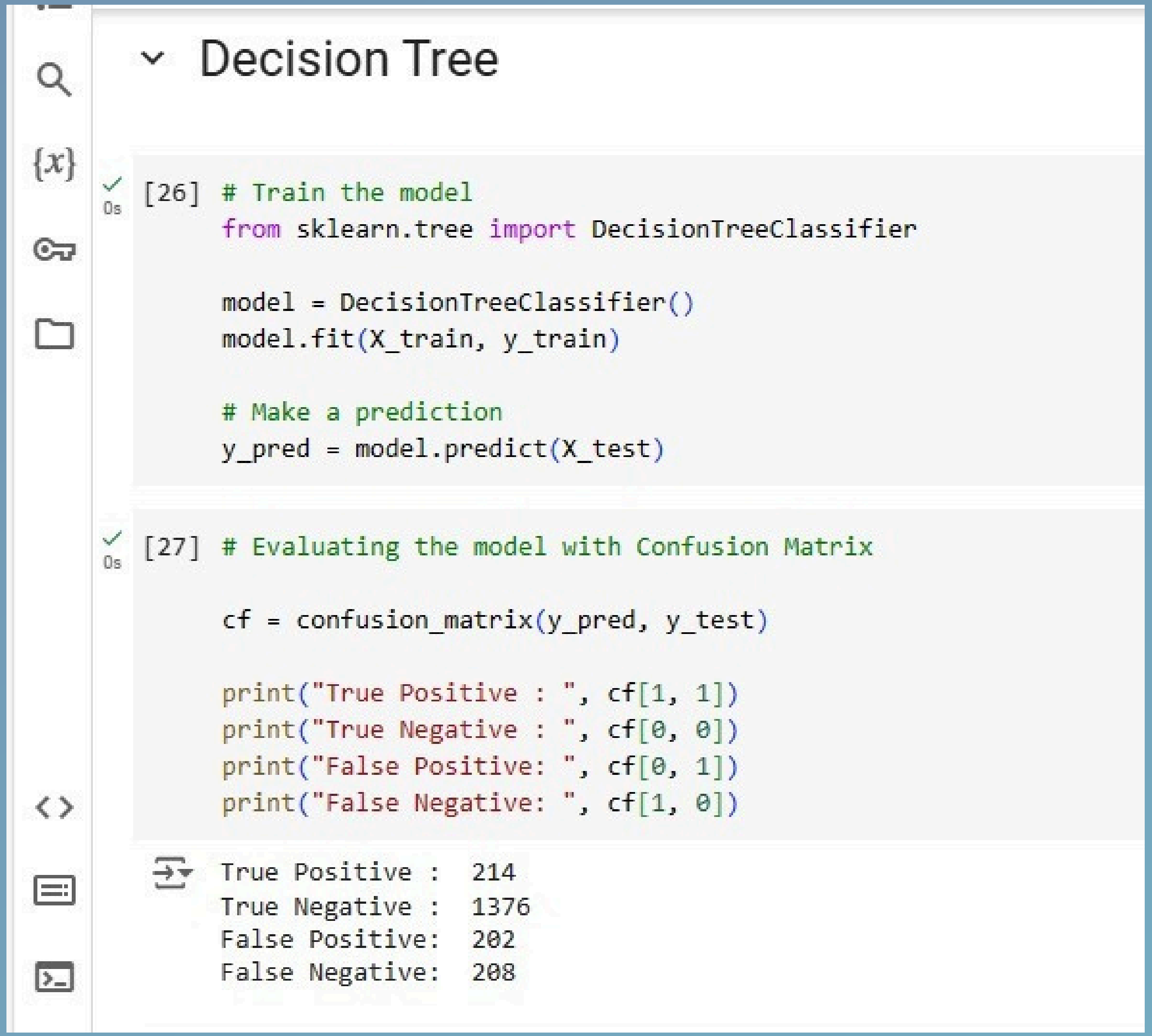


```
[25] # Evaluating the model with performance metrics
     from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

     logreg_accuracy = accuracy_score(y_test, y_pred)
     logreg_precision = precision_score(y_test, y_pred)
     logreg_recall = recall_score(y_test, y_pred)
     logreg_f1 = f1_score(y_test, y_pred)

     print("Accuracy: %.2f%%" % (logreg_accuracy * 100.0))
     print("Precision: %.2f%%" % (logreg_precision * 100.0))
     print("Recall: %.2f%%" % (logreg_recall * 100.0))
     print("F1 Score: %.2f%%" % (logreg_f1 * 100.0))
```

```
Accuracy: 80.70%
Precision: 65.96%
Recall: 14.90%
F1 Score: 24.31%
```

Logistic regression is a supervised machine learning algorithm designed for classification tasks, aiming to predict the probability that an instance belongs to a particular class. It is a statistical algorithm that examines the relationship between two data variables. This article delves into the basics of logistic regression, including its types and implementations.

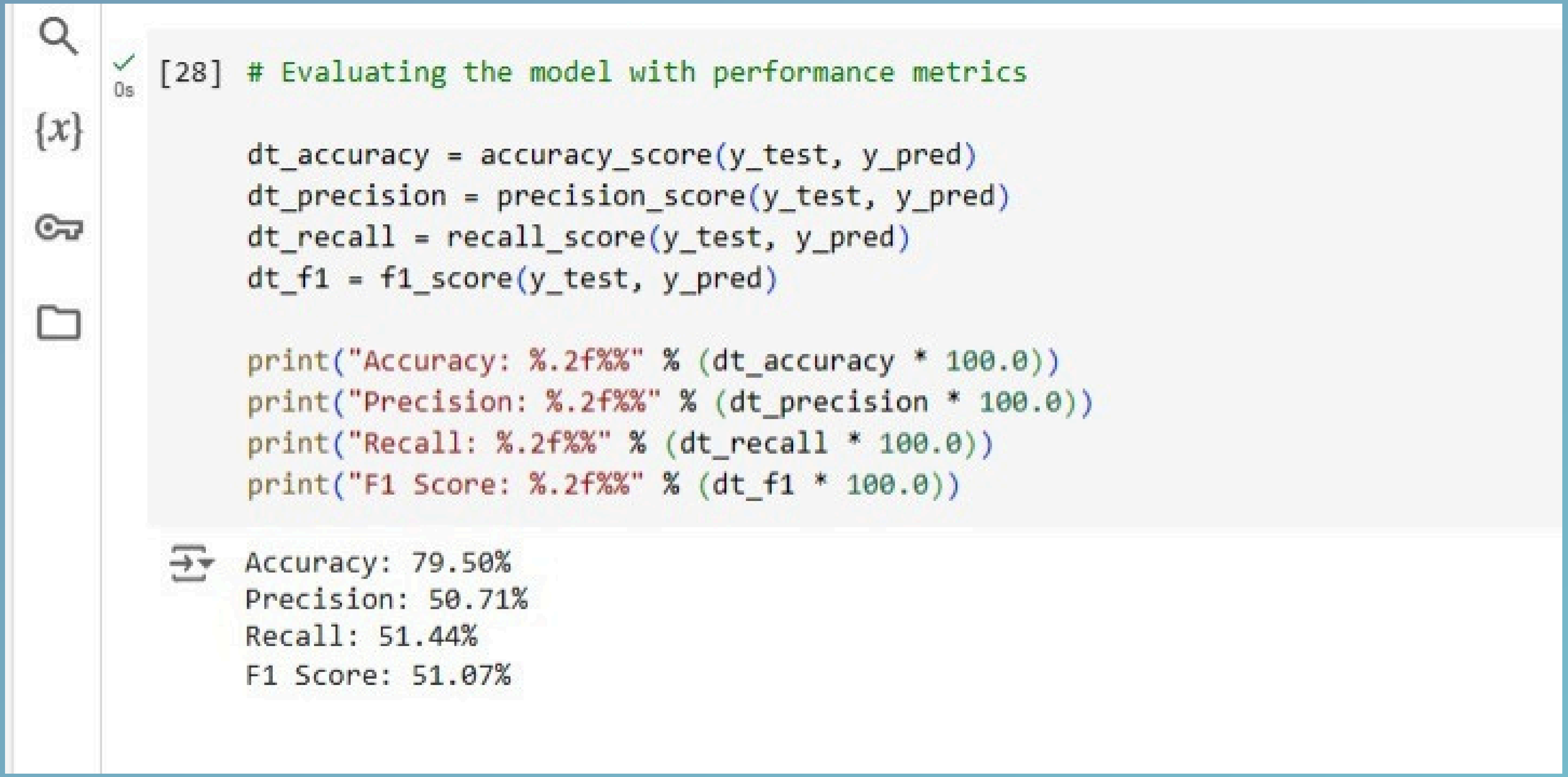
Decision Tree


Decision Tree

```
[26] # Train the model
     from sklearn.tree import DecisionTreeClassifier

     model = DecisionTreeClassifier()
     model.fit(X_train, y_train)

     # Make a prediction
     y_pred = model.predict(X_test)
```

```
[27] # Evaluating the model with Confusion Matrix

     cf = confusion_matrix(y_pred, y_test)

     print("True Positive : ", cf[1, 1])
     print("True Negative : ", cf[0, 0])
     print("False Positive: ", cf[0, 1])
     print("False Negative: ", cf[1, 0])
```

```
True Positive :  214
True Negative :  1376
False Positive:  202
False Negative:  208
```

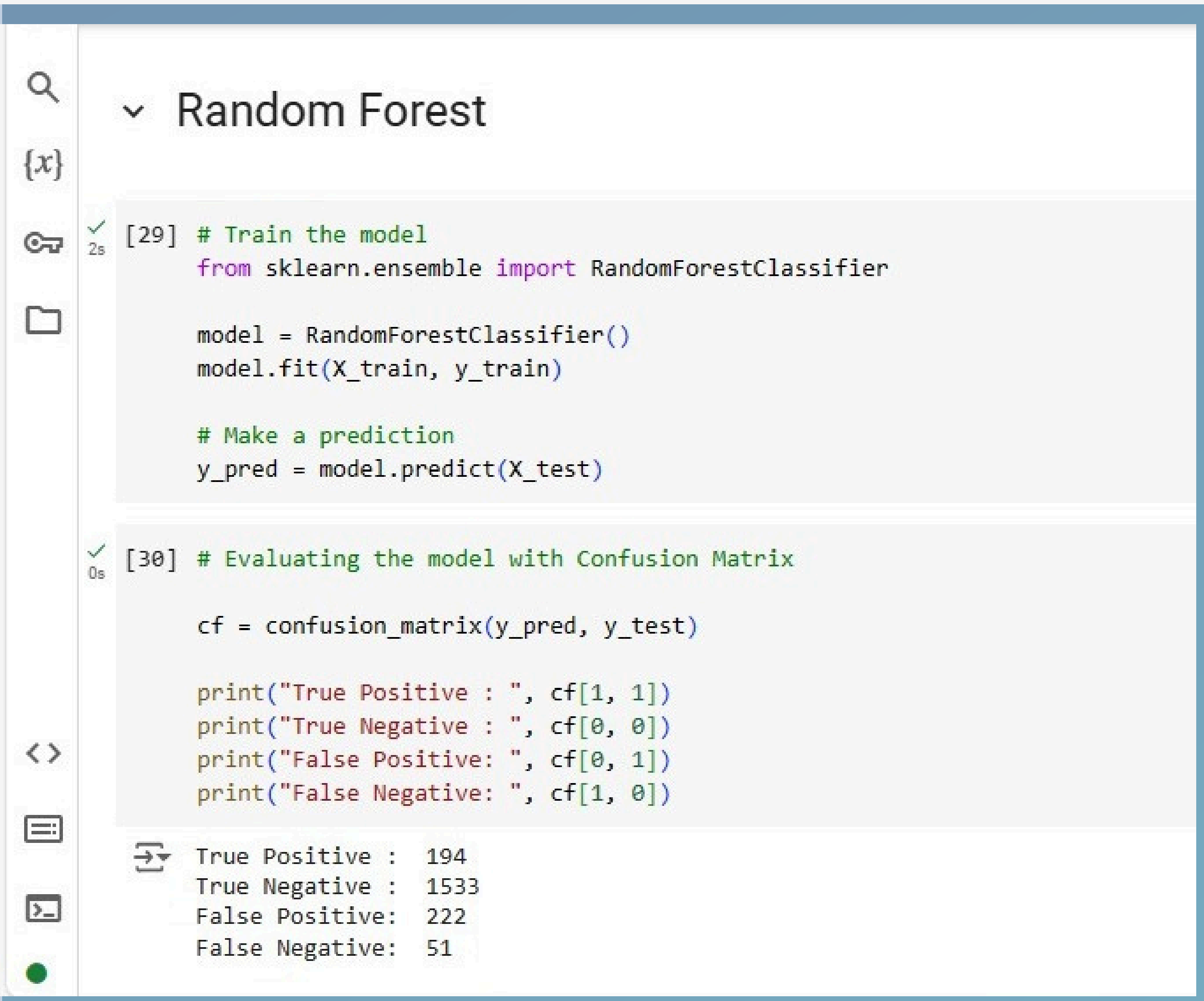


```
[28] # Evaluating the model with performance metrics

     dt_accuracy = accuracy_score(y_test, y_pred)
     dt_precision = precision_score(y_test, y_pred)
     dt_recall = recall_score(y_test, y_pred)
     dt_f1 = f1_score(y_test, y_pred)

     print("Accuracy: %.2f%%" % (dt_accuracy * 100.0))
     print("Precision: %.2f%%" % (dt_precision * 100.0))
     print("Recall: %.2f%%" % (dt_recall * 100.0))
     print("F1 Score: %.2f%%" % (dt_f1 * 100.0))
```

```
Accuracy: 79.50%
Precision: 50.71%
Recall: 51.44%
F1 Score: 51.07%
```

A decision tree is a flowchart-like structure where each internal node represents a feature, the branches represent decision rules, and the leaf nodes indicate the outcomes. This versatile supervised machine learning algorithm is used for both classification and regression tasks. It is a highly powerful algorithm and is also utilized in Random Forests, where it trains on different subsets of the training data, contributing to Random Forests being one of the most robust algorithms in machine learning.

Random Forest

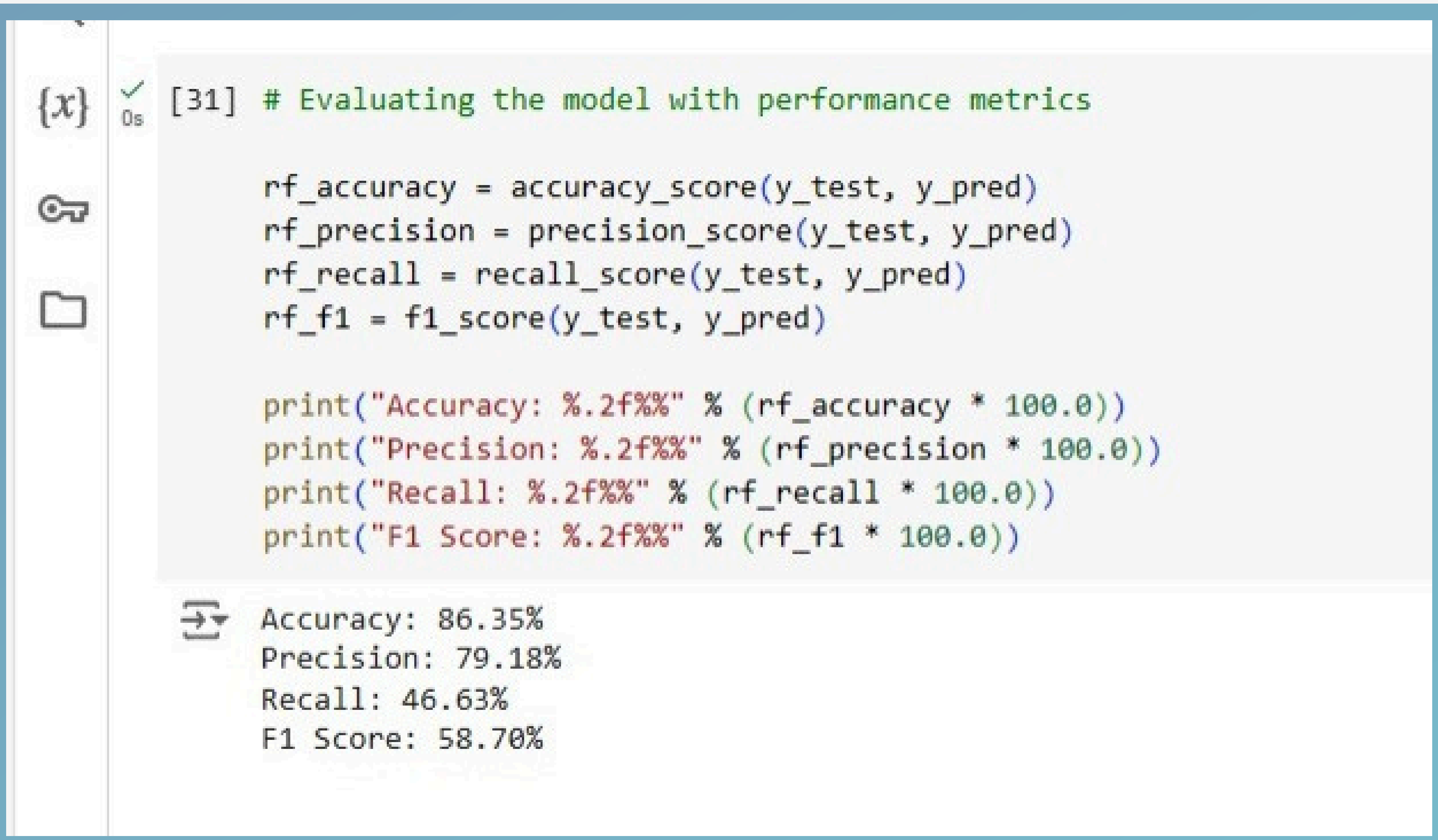


Random Forest

```
[29] # Train the model
     from sklearn.ensemble import RandomForestClassifier

     model = RandomForestClassifier()
     model.fit(X_train, y_train)

     # Make a prediction
     y_pred = model.predict(X_test)
```

```
[30] # Evaluating the model with Confusion Matrix

     cf = confusion_matrix(y_pred, y_test)

     print("True Positive : ", cf[1, 1])
     print("True Negative : ", cf[0, 0])
     print("False Positive: ", cf[0, 1])
     print("False Negative: ", cf[1, 0])
```

```
True Positive :  194
True Negative :  1533
False Positive:  222
False Negative:  51
```

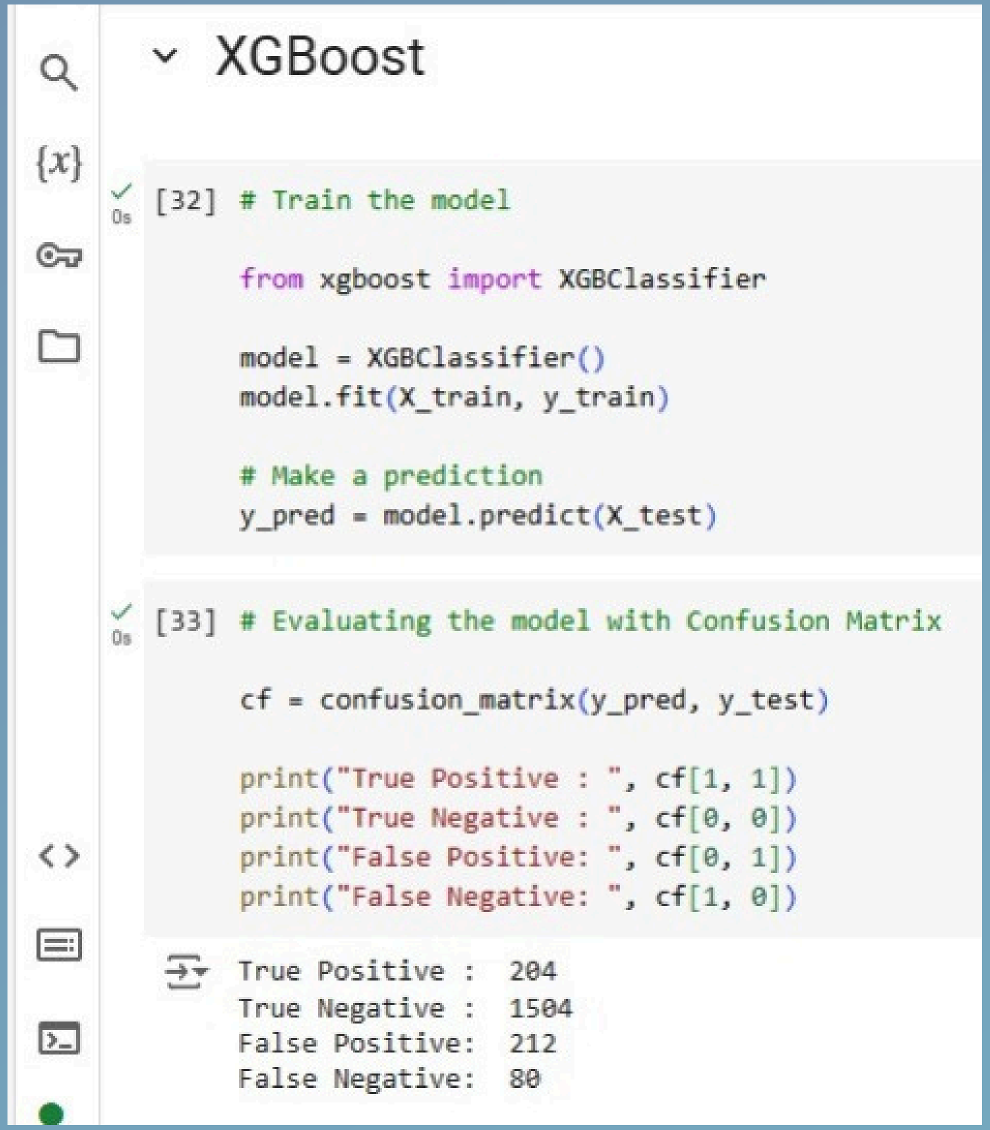


```
[31] # Evaluating the model with performance metrics

     rf_accuracy = accuracy_score(y_test, y_pred)
     rf_precision = precision_score(y_test, y_pred)
     rf_recall = recall_score(y_test, y_pred)
     rf_f1 = f1_score(y_test, y_pred)

     print("Accuracy: %.2f%%" % (rf_accuracy * 100.0))
     print("Precision: %.2f%%" % (rf_precision * 100.0))
     print("Recall: %.2f%%" % (rf_recall * 100.0))
     print("F1 Score: %.2f%%" % (rf_f1 * 100.0))
```

```
Accuracy: 86.35%
Precision: 79.18%
Recall: 46.63%
F1 Score: 58.70%
```

The Random Forest algorithm is a powerful machine learning technique that constructs multiple decision trees using random subsets of the dataset and features. This randomness helps reduce overfitting and improves prediction accuracy. For predictions, the algorithm combines results from all the trees, using voting for classification or averaging for regression. This ensemble approach ensures stable and accurate results, making Random Forests highly popular for both classification and regression tasks due to their ability to manage complex data and provide reliable predictions.

XGBoost

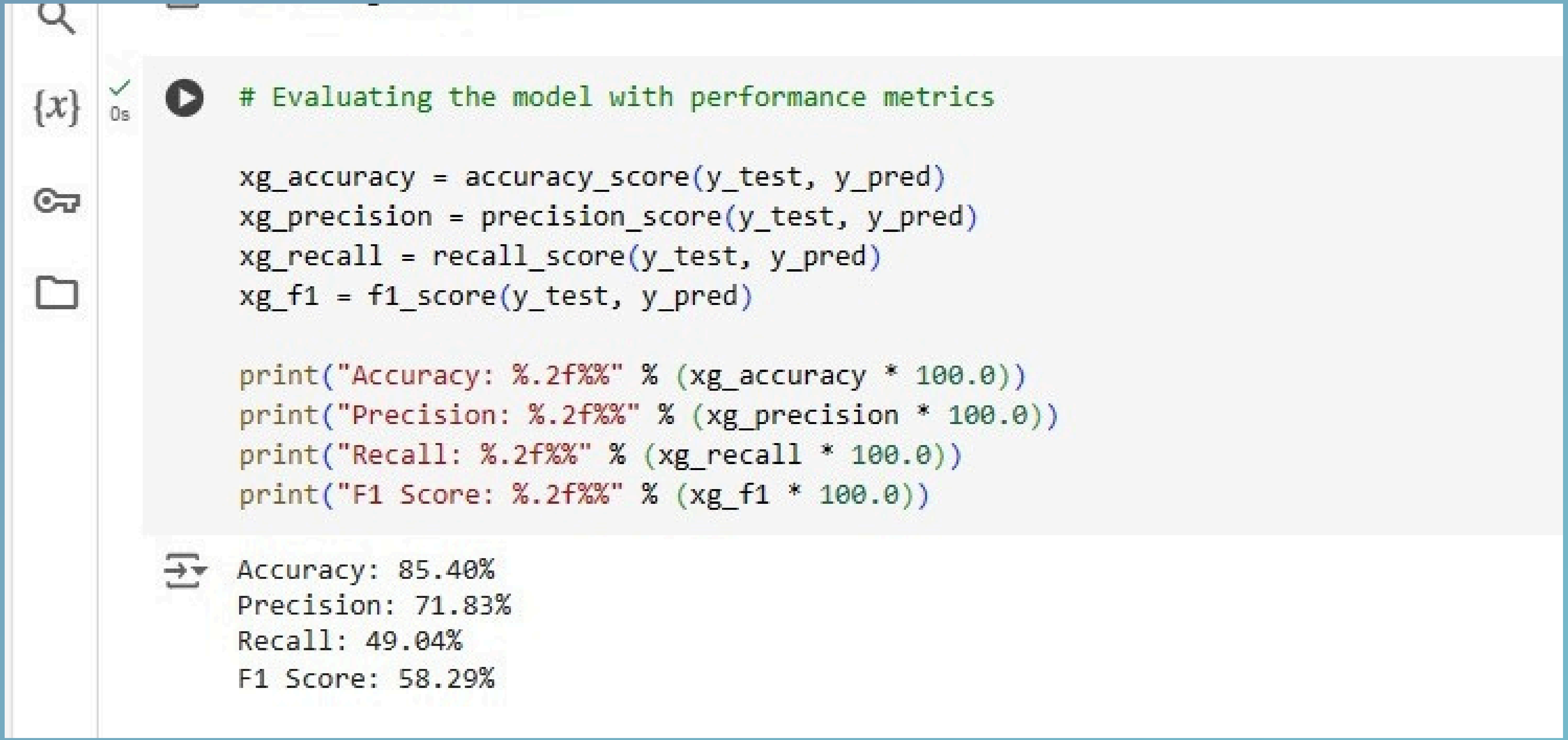


XGBoost

```
[32] # Train the model

     from xgboost import XGBClassifier

     model = XGBClassifier()
     model.fit(X_train, y_train)

     # Make a prediction
     y_pred = model.predict(X_test)
```

```
[33] # Evaluating the model with Confusion Matrix

     cf = confusion_matrix(y_pred, y_test)

     print("True Positive : ", cf[1, 1])
     print("True Negative : ", cf[0, 0])
     print("False Positive: ", cf[0, 1])
     print("False Negative: ", cf[1, 0])
```

```
True Positive :  204
True Negative :  1504
False Positive:  212
False Negative:  80
```

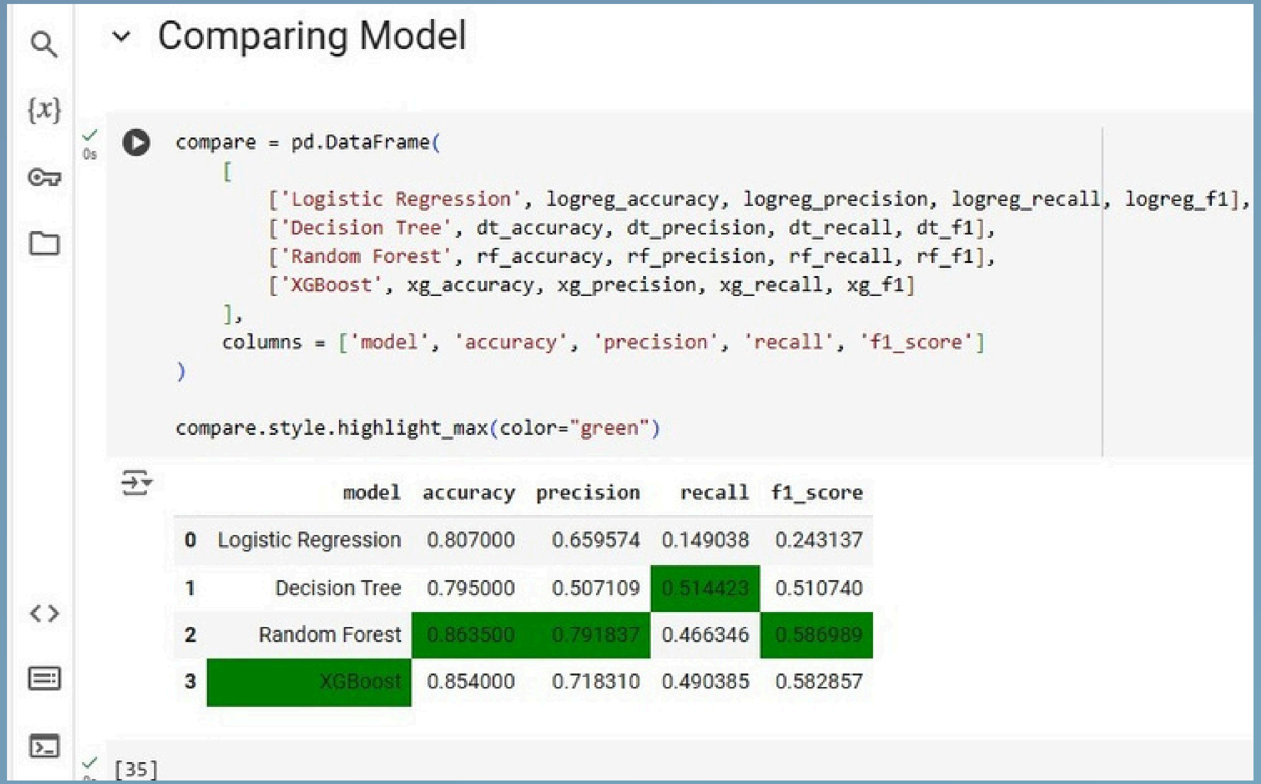


```
# Evaluating the model with performance metrics

xg_accuracy = accuracy_score(y_test, y_pred)
xg_precision = precision_score(y_test, y_pred)
xg_recall = recall_score(y_test, y_pred)
xg_f1 = f1_score(y_test, y_pred)

print("Accuracy: %.2f%%" % (xg_accuracy * 100.0))
print("Precision: %.2f%%" % (xg_precision * 100.0))
print("Recall: %.2f%%" % (xg_recall * 100.0))
print("F1 Score: %.2f%%" % (xg_f1 * 100.0))
```

```
Accuracy: 85.40%
Precision: 71.83%
Recall: 49.04%
F1 Score: 58.29%
```

XGBoost, or Extreme Gradient Boosting, is a leading machine learning algorithm known for its outstanding predictive performance. It is considered the gold standard in ensemble learning, particularly for gradient-boosting algorithms. XGBoost constructs a sequence of weak learners, typically decision trees, one after another to form a reliable and accurate predictive model. By using a boosting technique, it builds a powerful ensemble model where each weak learner corrects the errors of its predecessors, resulting in highly accurate predictions.

Model Evaluation

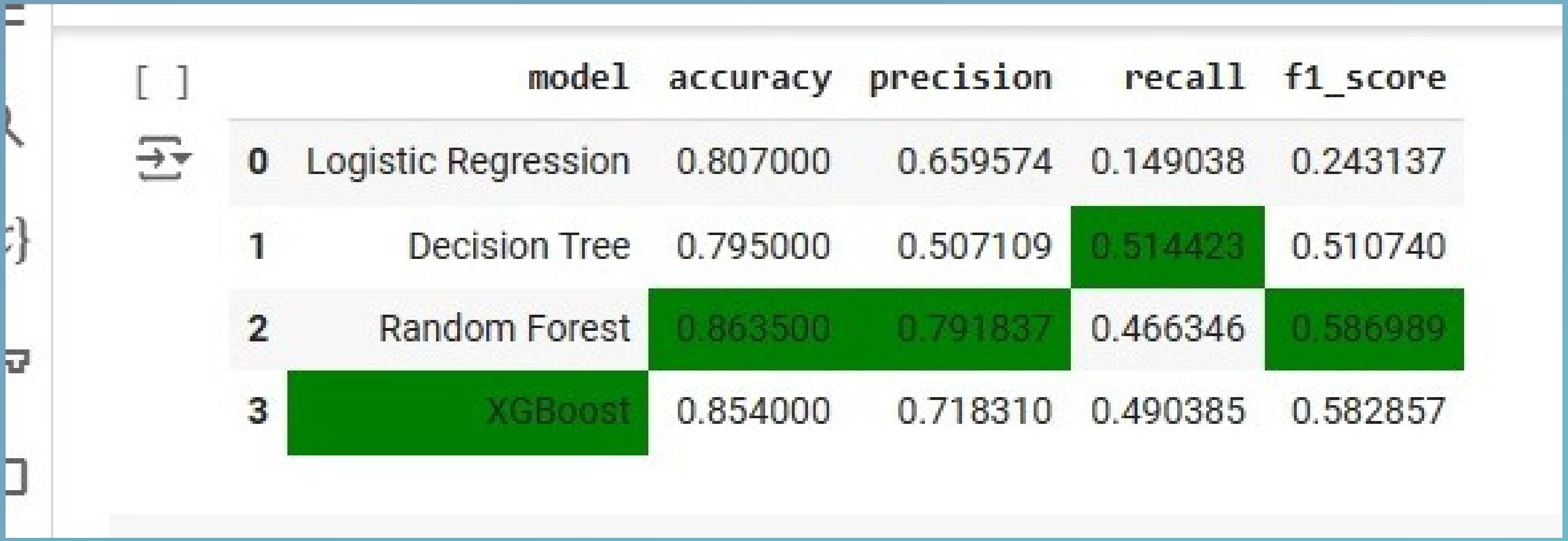


Comparing Model

```
compare = pd.DataFrame(
    [
        ['Logistic Regression', logreg_accuracy, logreg_precision, logreg_recall, logreg_f1],
        ['Decision Tree', dt_accuracy, dt_precision, dt_recall, dt_f1],
        ['Random Forest', rf_accuracy, rf_precision, rf_recall, rf_f1],
        ['XGBoost', xg_accuracy, xg_precision, xg_recall, xg_f1]
    ],
    columns = ['model', 'accuracy', 'precision', 'recall', 'f1_score']
)

compare.style.highlight_max(color="green")
```

	model	accuracy	precision	recall	f1_score
0	Logistic Regression	0.807000	0.659574	0.149038	0.243137
1	Decision Tree	0.795000	0.507109	0.514423	0.510740
2	Random Forest	0.863500	0.791837	0.466346	0.586989
3	XGBoost	0.854000	0.718310	0.490385	0.582857

[35]



	model	accuracy	precision	recall	f1_score
0	Logistic Regression	0.807000	0.659574	0.149038	0.243137
1	Decision Tree	0.795000	0.507109	0.514423	0.510740
2	Random Forest	0.863500	0.791837	0.466346	0.586989
3	XGBoost	0.854000	0.718310	0.490385	0.582857

Conclusion

- **Best Model:** Random Forest is the best-performing model overall, with the highest accuracy, precision, and F1 score, making it the most reliable for predicting customer churn.
- **High Recall:** XGBoost stands out for its high recall, making it a strong contender for identifying churned customers effectively.
- **Trade-Offs:** While Logistic Regression and Decision Tree have their strengths, they are outperformed by Random Forest and XGBoost in terms of the key metrics, especially in balancing precision and recall.

Based on this evaluation, Random Forest and XGBoost are recommended for deployment, with Random Forest slightly edging out as the best option for its overall superior performance.

Customer Lifetime Value

Back to Agenda

Goal # 1

Understanding Project

Goal # 2

Handling and Processing Data

Goal # 3

Model Evaluation NAd Conclusion

Project Objective CLV

The primary objective of this project is to accurately predict Customer Lifetime Value (CLV) using advanced modeling techniques, such as the BG/NBD and Gamma-Gamma models. By understanding and forecasting CLV, the project aims to:

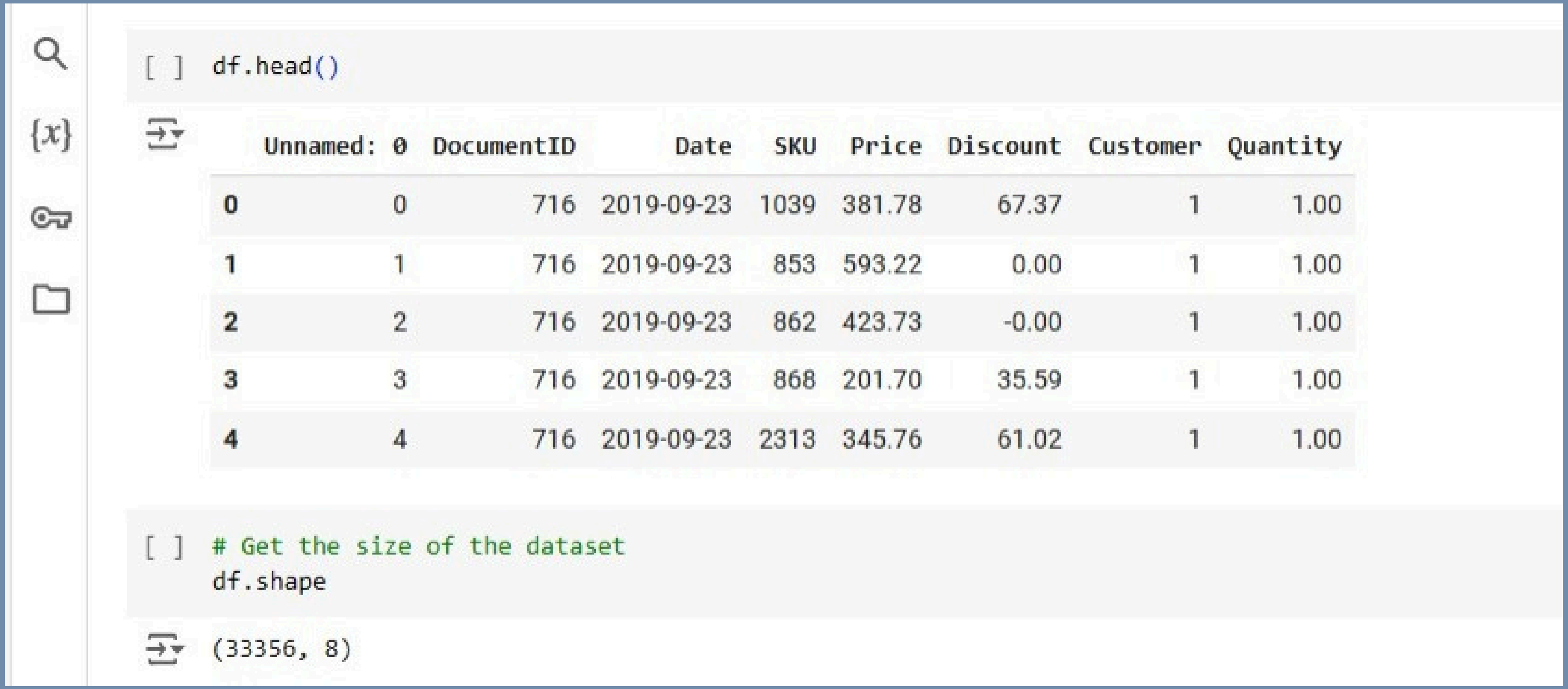
1. **Identify High-Value Customers:** Determine which customers contribute the most to the company's revenue over their lifetime.
2. **Optimize Resource Allocation:** Allocate marketing and retention resources more effectively by focusing on high-CLV customers.
3. **Enhance Customer Retention:** Develop strategies to retain high-value customers and increase their lifetime value.
4. **Improve Profitability:** Increase the overall profitability of the company by maximizing the revenue derived from each customer relationship.

Business Problem CLV

The business problem addressed by this project is the need for better customer value management and targeted marketing efforts. Specifically:

1. **Revenue Concentration:** Understanding that a small percentage of customers may contribute a large portion of the revenue, the business needs to identify and focus on these high-value customers.
2. **Resource Allocation:** Inefficient allocation of marketing and customer service resources can lead to suboptimal returns. There is a need to direct efforts towards customers who will generate the most revenue over time.
3. **Customer Retention:** High churn rates among valuable customers can significantly impact the company's profitability. Identifying factors that influence CLV can help in creating effective retention strategies.
4. **Market Segmentation:** The business requires a clear segmentation of its customer base to tailor marketing strategies effectively, aiming for personalized and impactful customer interactions.

Data Overview



```
[ ]  df.head()
```

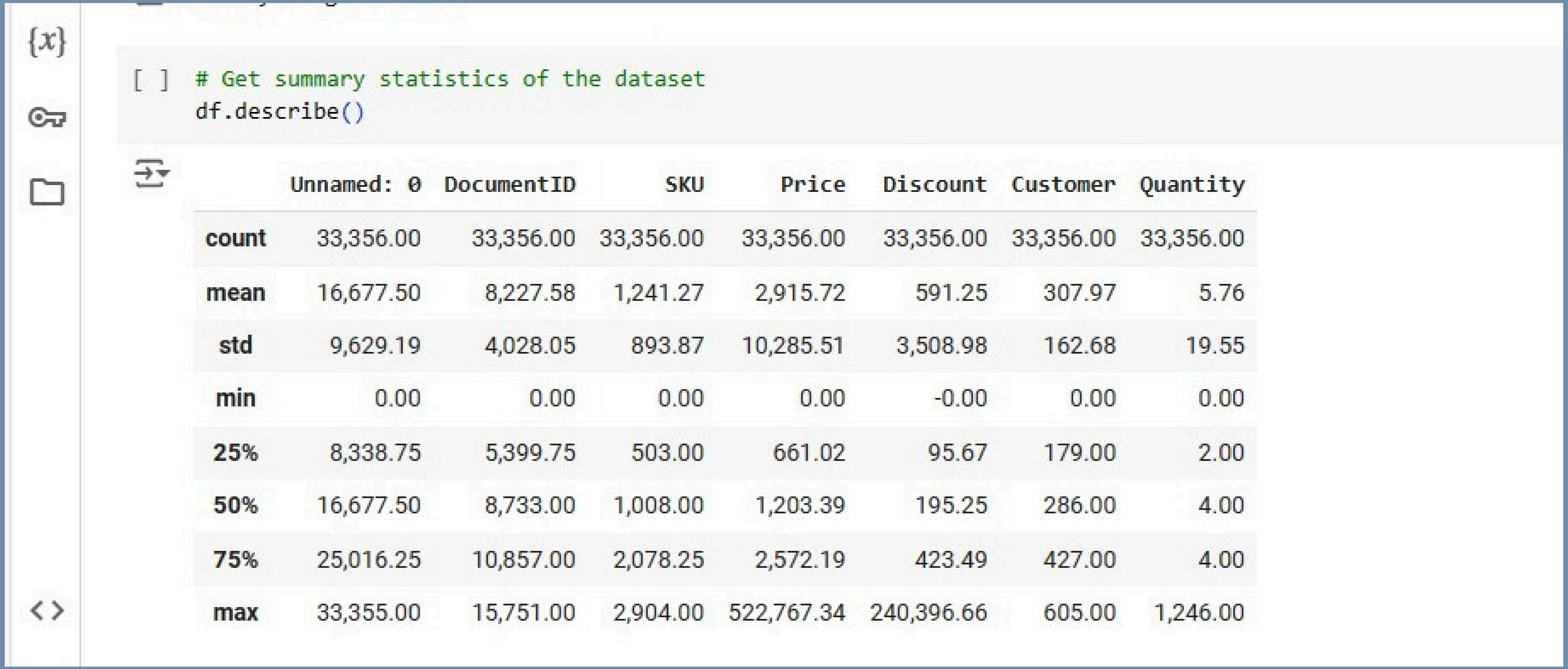
```
   Unnamed: 0  DocumentID        Date   SKU   Price  Discount  Customer  Quantity
0           0         716  2019-09-23  1039  381.78     67.37         1      1.00
1           1         716  2019-09-23   853  593.22      0.00         1      1.00
2           2         716  2019-09-23   862  423.73     -0.00         1      1.00
3           3         716  2019-09-23   868  201.70     35.59         1      1.00
4           4         716  2019-09-23  2313  345.76     61.02         1      1.00
```

```
[ ]  # Get the size of the dataset
     df.shape
```

```
(33356, 8)
```

- **InvoiceID** : ID of the transaction. A transaction might hold multiple records for the same customer at the same date with multiple products (SKU). DocumentID might be useful for combining the transactions and detecting the items sold together.
- **Date** : Date of transaction / sell. In the date time format.
- **ProductID** : Item / Product code. The unique code for each item sold.
- **TotalSales** : Sales price for the transaction. If you want to get *unit_price* , divide *TotalSales* column to *Quantity* column
- **Discount** : Discount amount for the transaction.
- **CustomerID** : Unique customer id for each customer. For the data set, customer can be a reseller or a branch of the company.
- **Quantity** : Number of items sold in the transaction.

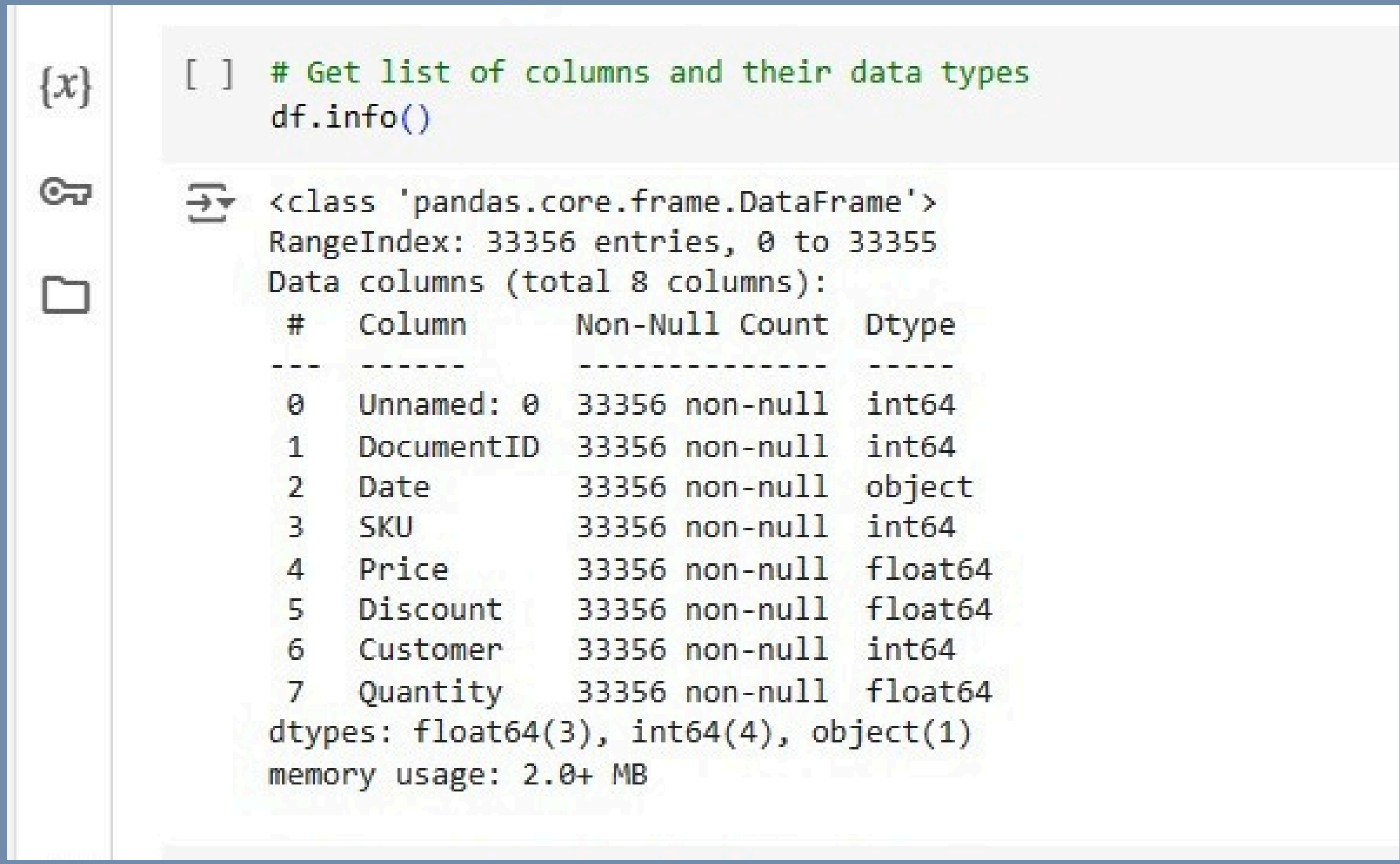
EDA



```
# Get summary statistics of the dataset
df.describe()
```

	Unnamed: 0	DocumentID	SKU	Price	Discount	Customer	Quantity
count	33,356.00	33,356.00	33,356.00	33,356.00	33,356.00	33,356.00	33,356.00
mean	16,677.50	8,227.58	1,241.27	2,915.72	591.25	307.97	5.76
std	9,629.19	4,028.05	893.87	10,285.51	3,508.98	162.68	19.55
min	0.00	0.00	0.00	0.00	-0.00	0.00	0.00
25%	8,338.75	5,399.75	503.00	661.02	95.67	179.00	2.00
50%	16,677.50	8,733.00	1,008.00	1,203.39	195.25	286.00	4.00
75%	25,016.25	10,857.00	2,078.25	2,572.19	423.49	427.00	4.00
max	33,355.00	15,751.00	2,904.00	522,767.34	240,396.66	605.00	1,246.00

There are two points to understand in the Exploratory Data Analysis (EDA) process: Understanding the Data: First, examine the data structure, data types, and descriptive statistics for each column. The EDA process this time there are several processes that are passed, namely: Get the size of the data set Get a list of columns and their data types Get the summary statistics of the data set Calculate the unique values in each column



```
# Get list of columns and their data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 33356 entries, 0 to 33355
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  33356 non-null  int64
 1   DocumentID  33356 non-null  int64
 2   Date        33356 non-null  object
 3   SKU         33356 non-null  int64
 4   Price       33356 non-null  float64
 5   Discount    33356 non-null  float64
 6   Customer    33356 non-null  int64
 7   Quantity    33356 non-null  float64
dtypes: float64(3), int64(4), object(1)
memory usage: 2.0+ MB
```

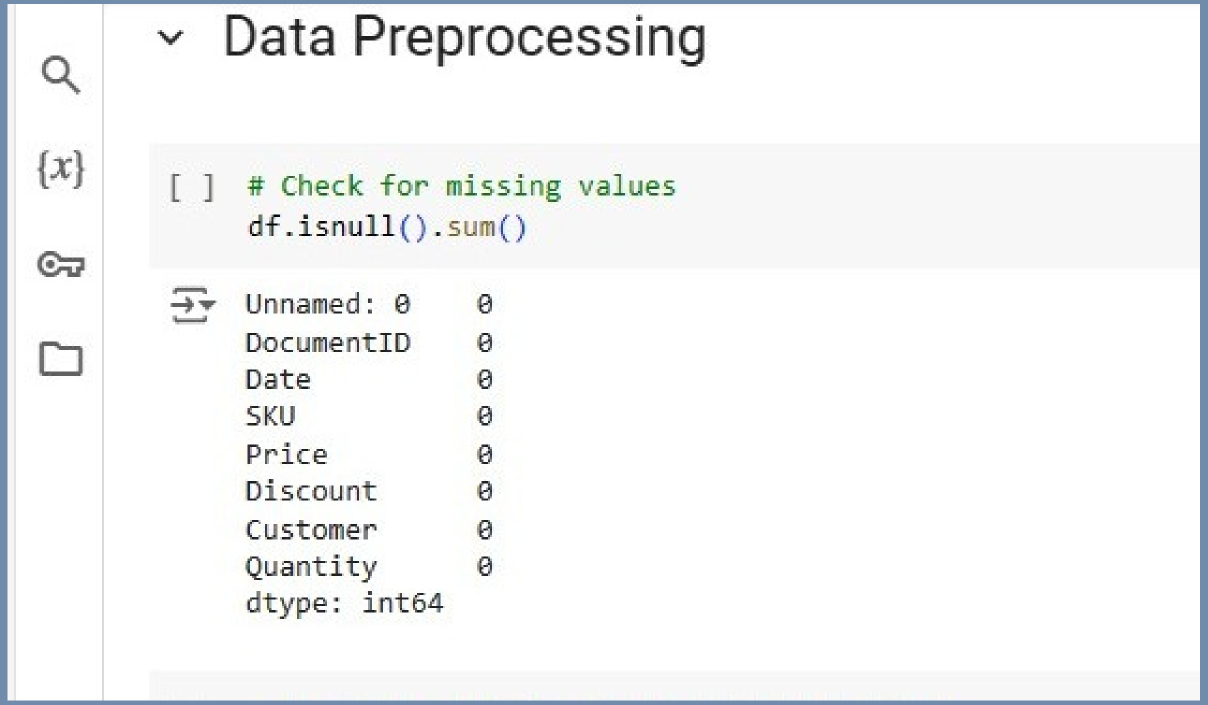
Data Preprocessing

In the preprocessing phase for Customer Lifetime Value (CLV) analysis, several steps are undertaken. These include:

1. **Feature Engineering:** Additional features are created to enhance the analysis of churn, such as the duration of a customer's tenure or their purchase frequency.
2. **Encoding**: Categorical variables are transformed into a format understandable by the model, typically through techniques like one-hot encoding or label encoding.
3. **Data Segregation**: The data is divided into training and testing sets to facilitate model training and evaluation.

Specifically, during this preprocessing stage, the following tasks are completed:
- Checking for missing values in the dataset.
- Removing all categorical columns except for the "Customer" column.
- Renaming the "Customer" column for clarity.
- Converting the "Date" column from a string format to datetime for accurate temporal analysis.

Data Preprocessing



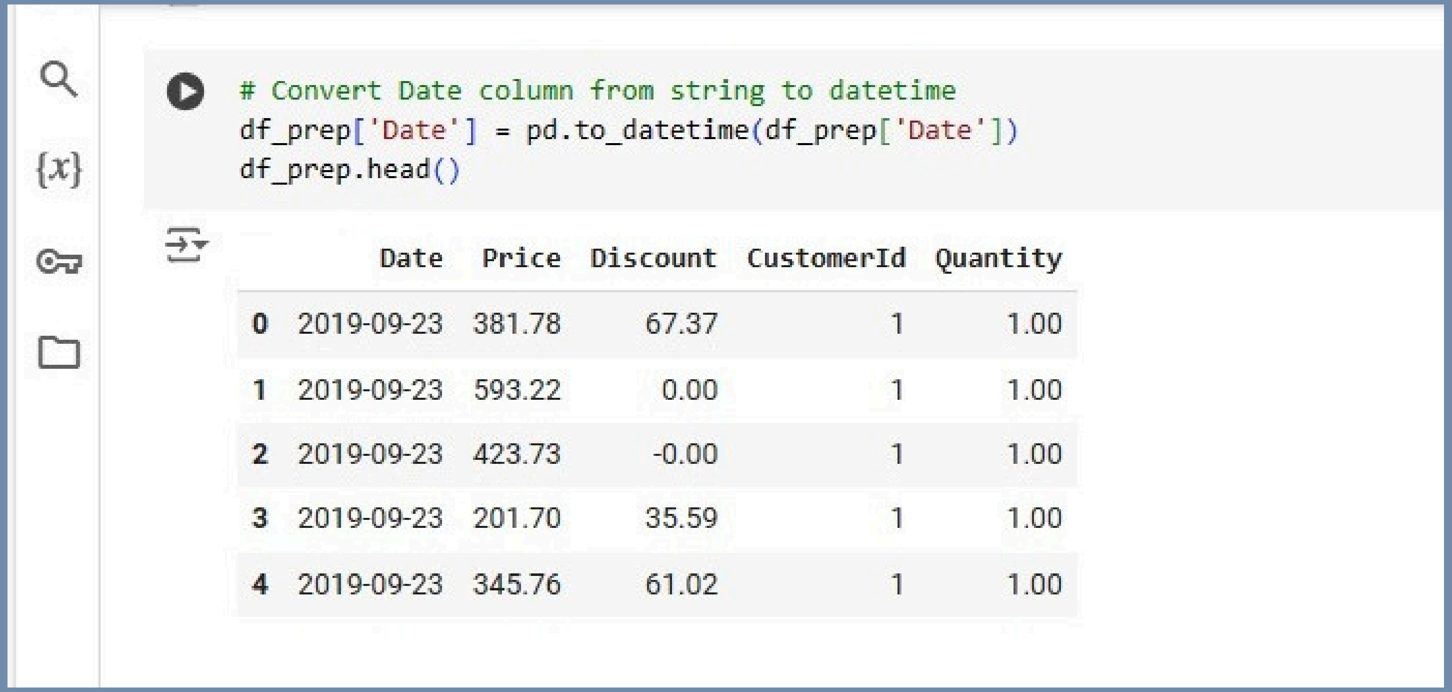
Data Preprocessing

```
[ ] # Check for missing values
    df.isnull().sum()
```

```
Unnamed: 0    0
DocumentID    0
Date          0
SKU           0
Price         0
Discount      0
Customer      0
Quantity      0
dtype: int64
```

Missing Value Check

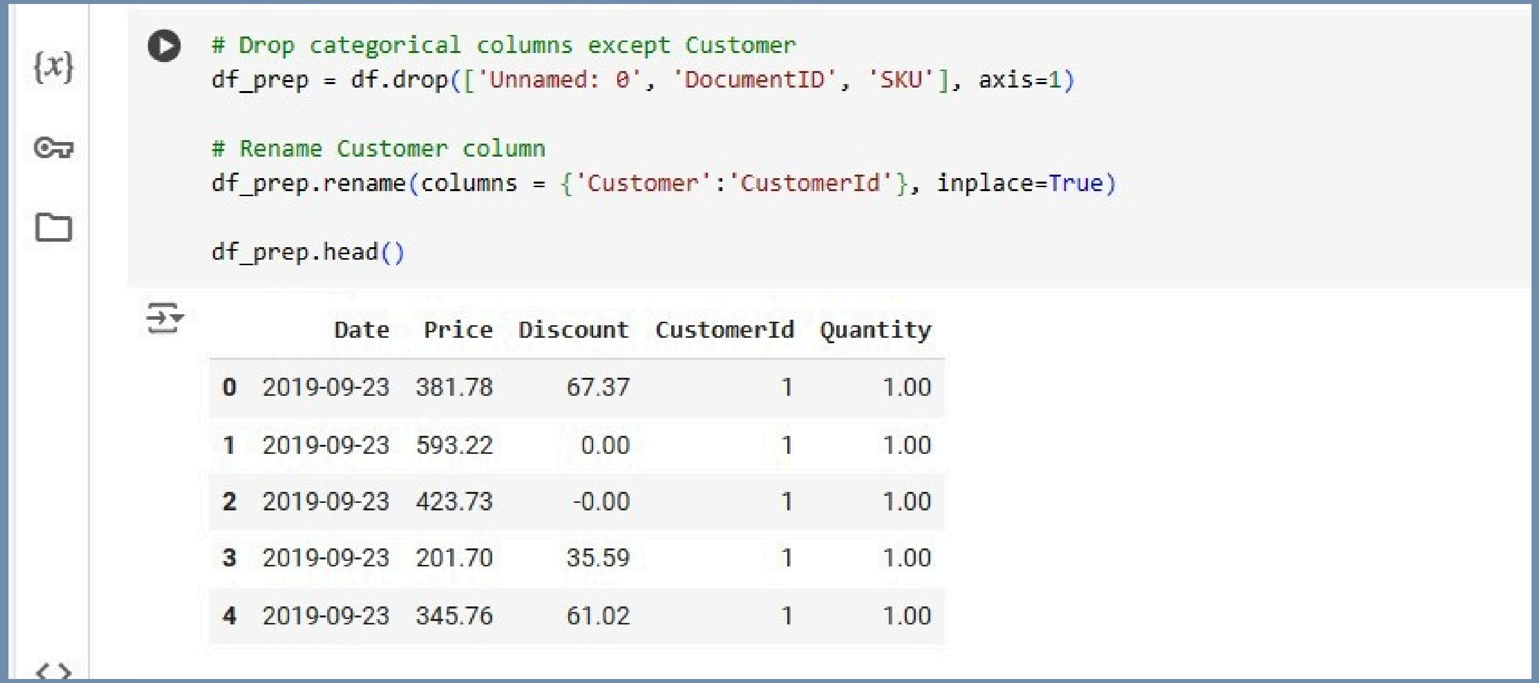
Convert Date to Date time



```
# Convert Date column from string to datetime
df_prep['Date'] = pd.to_datetime(df_prep['Date'])
df_prep.head()
```

	Date	Price	Discount	CustomerId	Quantity
0	2019-09-23	381.78	67.37	1	1.00
1	2019-09-23	593.22	0.00	1	1.00
2	2019-09-23	423.73	-0.00	1	1.00
3	2019-09-23	201.70	35.59	1	1.00
4	2019-09-23	345.76	61.02	1	1.00

Drop & Rename Column

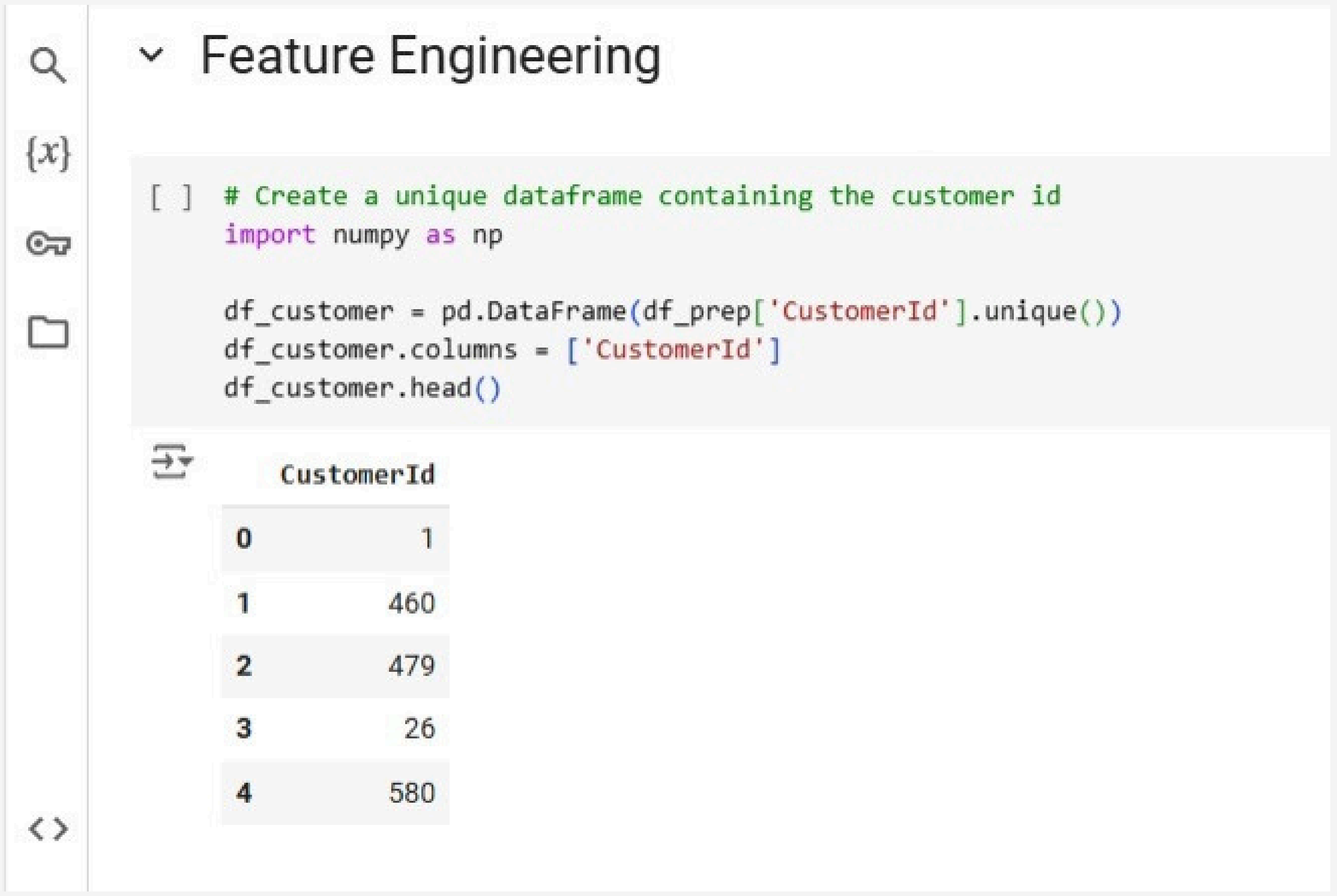


```
# Drop categorical columns except Customer
df_prep = df.drop(['Unnamed: 0', 'DocumentID', 'SKU'], axis=1)

# Rename Customer column
df_prep.rename(columns = {'Customer':'CustomerId'}, inplace=True)

df_prep.head()
```

	Date	Price	Discount	CustomerId	Quantity
0	2019-09-23	381.78	67.37	1	1.00
1	2019-09-23	593.22	0.00	1	1.00
2	2019-09-23	423.73	-0.00	1	1.00
3	2019-09-23	201.70	35.59	1	1.00
4	2019-09-23	345.76	61.02	1	1.00

Feature Engineering

This process is the initial step in CLV analysis, as CLV is intrinsically linked to the behavior and purchase history of individual customers. By creating a unique DataFrame for each customer ID, we can monitor and analyze the purchase activity of each customer individually. This approach allows for the development of more accurate and effective CLV models and strategies.



Feature Engineering

```
# Create a unique dataframe containing the customer id
import numpy as np

df_customer = pd.DataFrame(df_prep['CustomerId'].unique())
df_customer.columns = ['CustomerId']
df_customer.head()
```

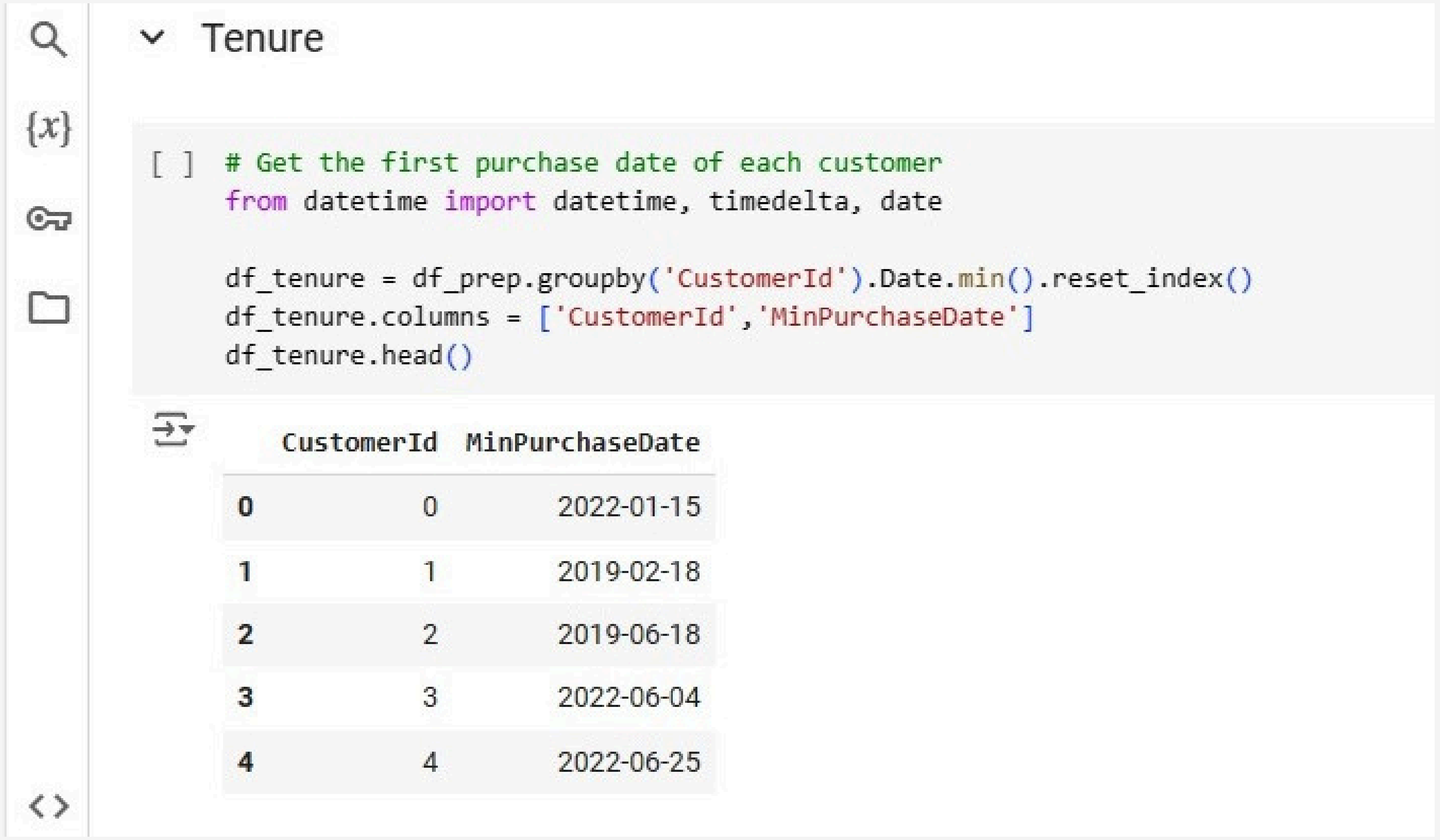
	CustomerId
0	1
1	460
2	479
3	26
4	580

Calculate RFM-T

Calculating the RFM-T Kembali ke Agenda Calculating the RFM-T involves analyzing customer behavior based on three key metrics: Recency (R), Frequency (F), Monetary Value (M), and Time (T). Recency refers to how recently a customer made a purchase, Frequency measures how often they make purchases, and Monetary Value reflects the total amount they spent. The addition of Time (T) factor accounts for the duration or length of the customer relationship. By combining these metrics, RFM-T helps segment customers into different groups based on their buying patterns and allows businesses to tailor marketing strategies, predict future behavior, and identify high-value customers for targeted campaigns

Calculate RFM-T

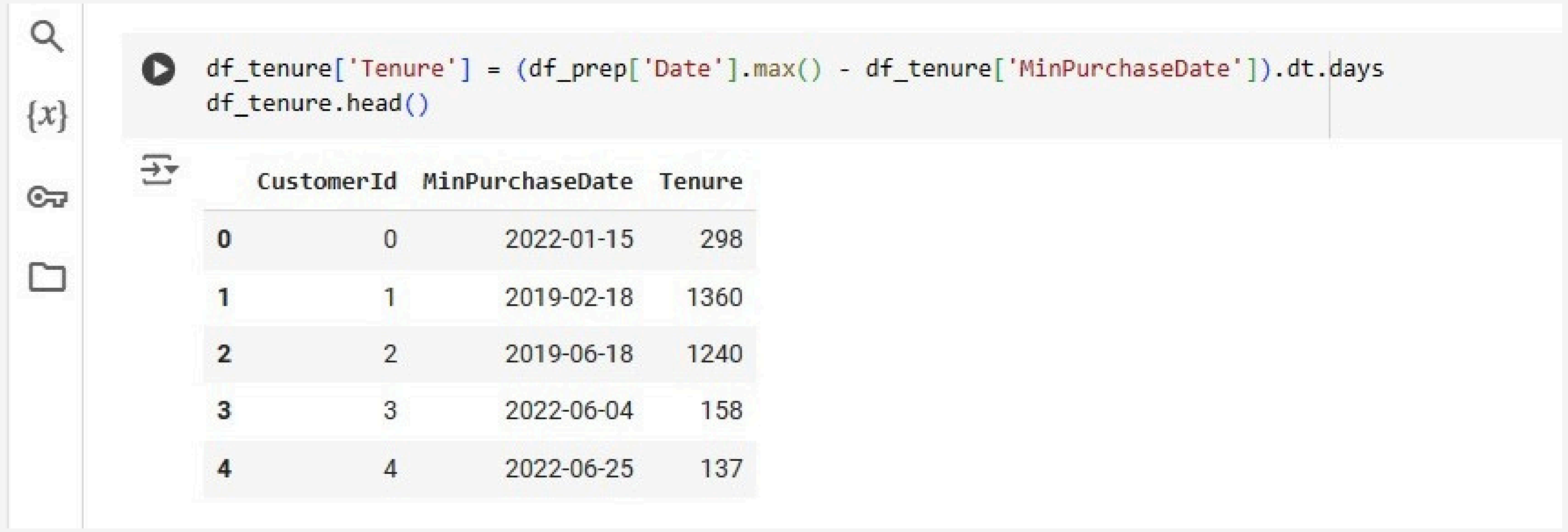
Tenure is the process of calculating the length of time or duration that has passed since the customer first made a transaction or became a customer of the company. So that the process on the side is a calculation process which is then created a new column called tenure.



Tenure

```
[ ] # Get the first purchase date of each customer
    from datetime import datetime, timedelta, date

    df_tenure = df_prep.groupby('CustomerId').Date.min().reset_index()
    df_tenure.columns = ['CustomerId','MinPurchaseDate']
    df_tenure.head()
```

	CustomerId	MinPurchaseDate
0	0	2022-01-15
1	1	2019-02-18
2	2	2019-06-18
3	3	2022-06-04
4	4	2022-06-25

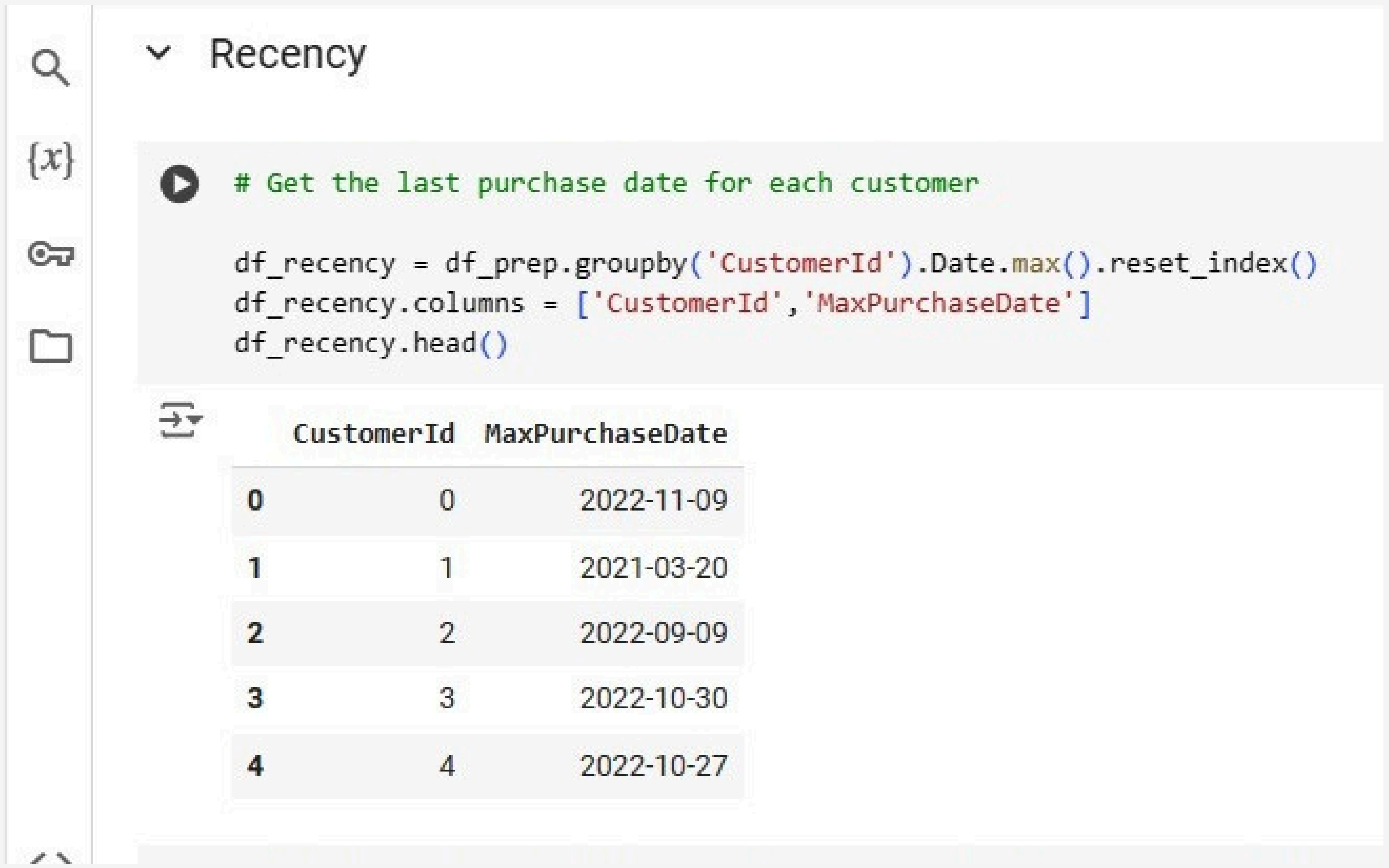


```
df_tenure['Tenure'] = (df_prep['Date'].max() - df_tenure['MinPurchaseDate']).dt.days
df_tenure.head()
```

	CustomerId	MinPurchaseDate	Tenure
0	0	2022-01-15	298
1	1	2019-02-18	1360
2	2	2019-06-18	1240
3	3	2022-06-04	158
4	4	2022-06-25	137

Calculate RFM-T

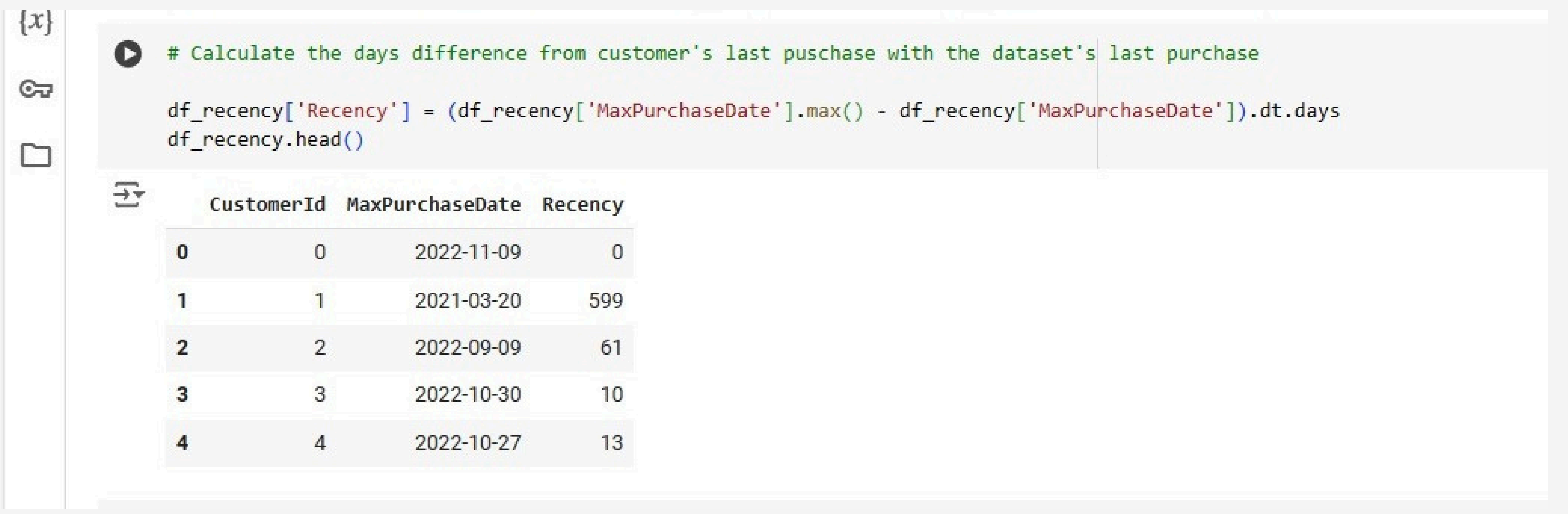
Recency (R) measures how recent or long the time has passed since the customer last had a transaction or interaction with the company. The more recent the transaction, the lower the recency value, indicating a more engaged and active customer. in this process there are 3 stages that are carried out, namely: Get the last purchase date for each customer Calculate the days difference from customer ' s last puschase with the dataset' s last purchase. And the last is visualization



Recency

```
# Get the last purchase date for each customer

df_recency = df_prep.groupby('CustomerId').Date.max().reset_index()
df_recency.columns = ['CustomerId','MaxPurchaseDate']
df_recency.head()
```

```
   CustomerId  MaxPurchaseDate
0           0       2022-11-09
1           1       2021-03-20
2           2       2022-09-09
3           3       2022-10-30
4           4       2022-10-27
```



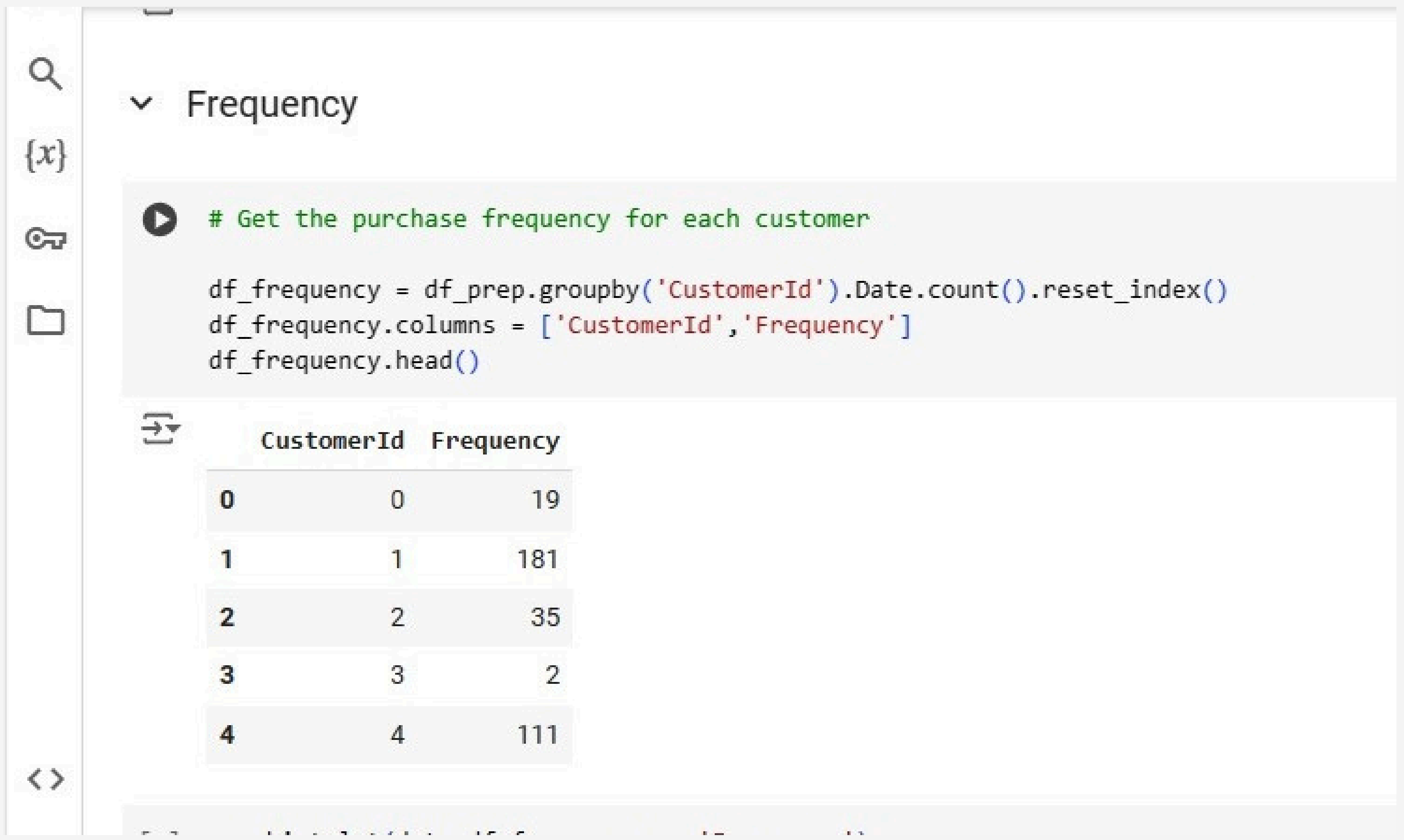
```
# Calculate the days difference from customer's last puschase with the dataset's last purchase

df_recency['Recency'] = (df_recency['MaxPurchaseDate'].max() - df_recency['MaxPurchaseDate']).dt.days
df_recency.head()
```

```
   CustomerId  MaxPurchaseDate  Recency
0           0       2022-11-09        0
1           1       2021-03-20      599
2           2       2022-09-09       61
3           3       2022-10-30       10
4           4       2022-10-27       13
```

Calculate RFM-T

Frequency (F) denotes the total number of transactions or interactions a customer has with a company over a specific period. A higher frequency indicates more frequent customer interactions and higher engagement. This process involves two stages:

1. Calculate the purchase frequency for each customer.
2. Visualize the data.



Frequency

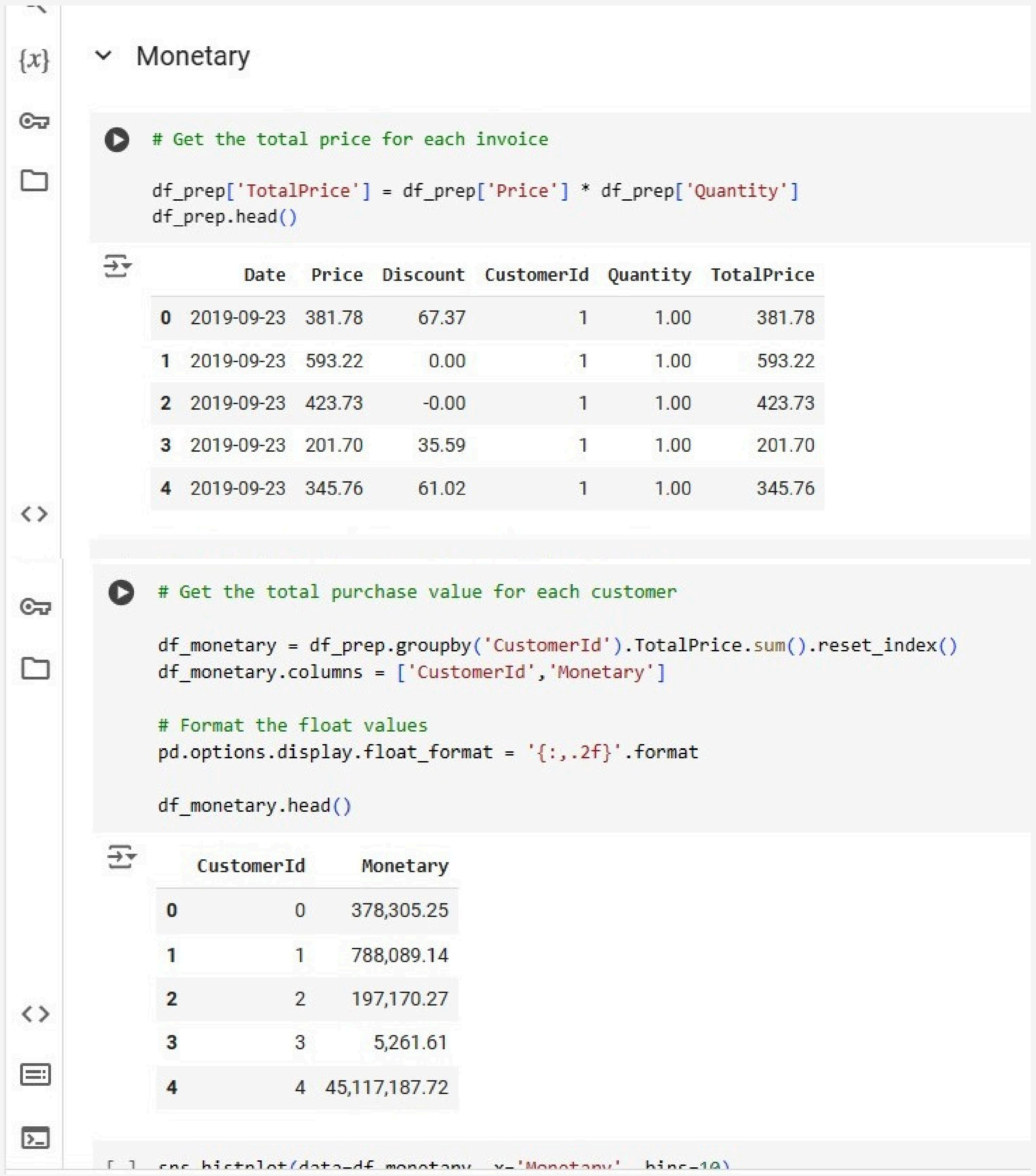
```
# Get the purchase frequency for each customer

df_frequency = df_prep.groupby('CustomerId').Date.count().reset_index()
df_frequency.columns = ['CustomerId','Frequency']
df_frequency.head()
```

```
   CustomerId  Frequency
0           0         19
1           1        181
2           2         35
3           3          2
4           4        111
```

Calculate RFM-T

Monetary (M) represents the total value of purchases made by customers over a specific period. A higher monetary value signifies a greater contribution to the company's revenue. This process involves four stages:

1. Calculate the total price for each invoice.
2. Determine the total purchase value for each customer.
3. Format the float values.
4. Visualize the data.



Monetary

```
# Get the total price for each invoice

df_prep['TotalPrice'] = df_prep['Price'] * df_prep['Quantity']
df_prep.head()
```

```
         Date   Price  Discount  CustomerId  Quantity  TotalPrice
0  2019-09-23  381.78     67.37           1      1.00      381.78
1  2019-09-23  593.22      0.00           1      1.00      593.22
2  2019-09-23  423.73     -0.00           1      1.00      423.73
3  2019-09-23  201.70     35.59           1      1.00      201.70
4  2019-09-23  345.76     61.02           1      1.00      345.76
```

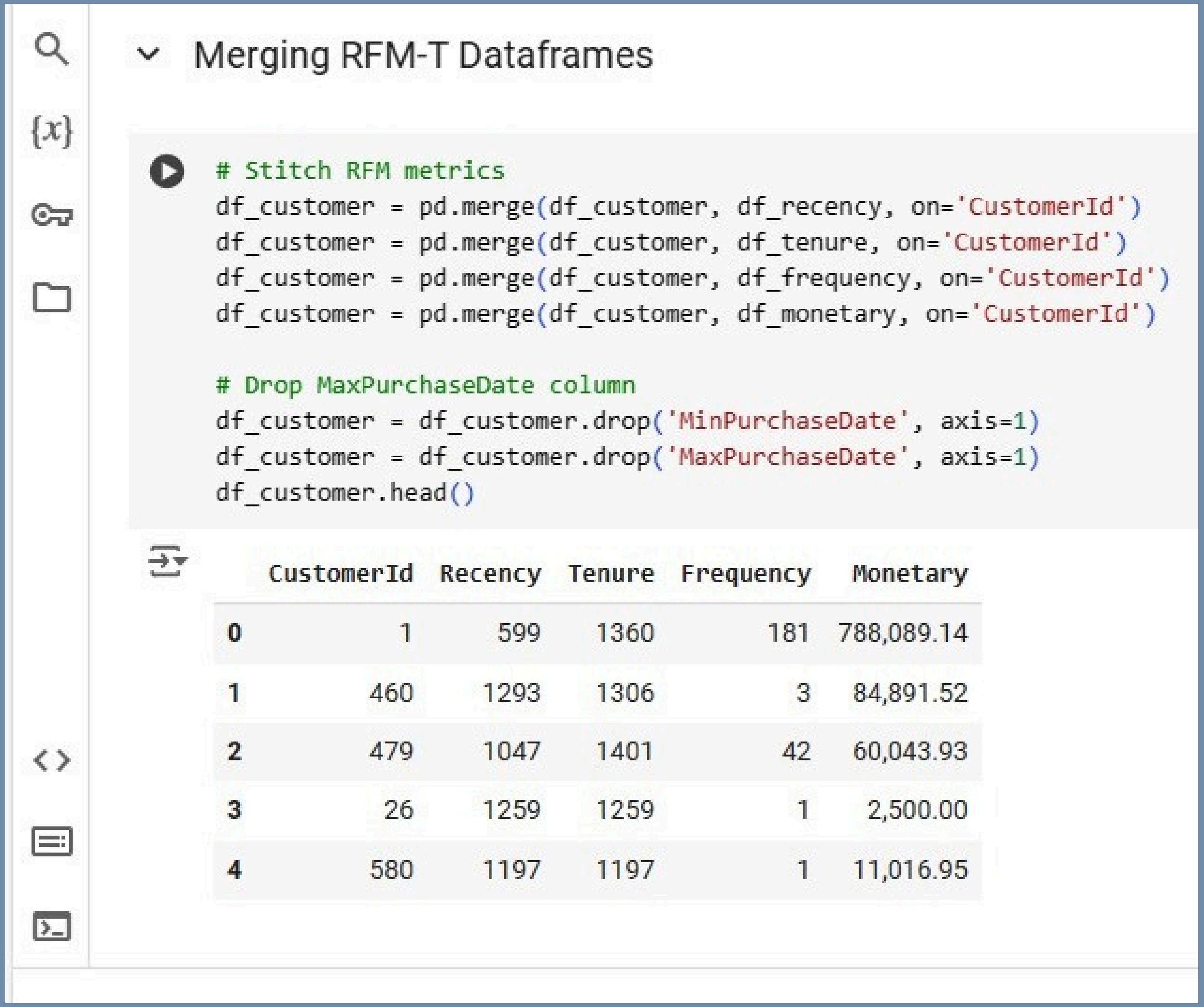
```
# Get the total purchase value for each customer

df_monetary = df_prep.groupby('CustomerId').TotalPrice.sum().reset_index()
df_monetary.columns = ['CustomerId','Monetary']

# Format the float values
pd.options.display.float_format = '{:,.2f}'.format

df_monetary.head()
```

```
   CustomerId       Monetary
0           0     378,305.25
1           1     788,089.14
2           2     197,170.27
3           3       5,261.61
4           4  45,117,187.72
```

```
[ ] sns.histplot(data=df_monetary, x='Monetary', bins=10)
```

Merging RFM-T Data frame

Merging RFM-T Dataframes involves merging dataframes containing Recency, Frequency, Monetary, and Tenure values for each customer. This is done to generate a comprehensive dataset that includes key information about the purchasing behavior and relationship duration with each customer. This process is important in CLV analysis as it enables identification and mapping of customers based on their RFM-T values. By having an aggregated dataset, companies can use more sophisticated analytical techniques to understand customer buying patterns, classify customers into different segments, and develop more effective marketing strategies to increase customer retention and profitability.


Merging RFM-T Dataframes

```
# Stitch RFM metrics
df_customer = pd.merge(df_customer, df_recency, on='CustomerId')
df_customer = pd.merge(df_customer, df_tenure, on='CustomerId')
df_customer = pd.merge(df_customer, df_frequency, on='CustomerId')
df_customer = pd.merge(df_customer, df_monetary, on='CustomerId')

# Drop MaxPurchaseDate column
df_customer = df_customer.drop('MinPurchaseDate', axis=1)
df_customer = df_customer.drop('MaxPurchaseDate', axis=1)
df_customer.head()
```

```
   CustomerId  Recency  Tenure  Frequency    Monetary
0           1      599    1360        181  788,089.14
1         460     1293    1306          3   84,891.52
2         479     1047    1401         42   60,043.93
3          26     1259    1259          1    2,500.00
4         580     1197    1197          1   11,016.95
```

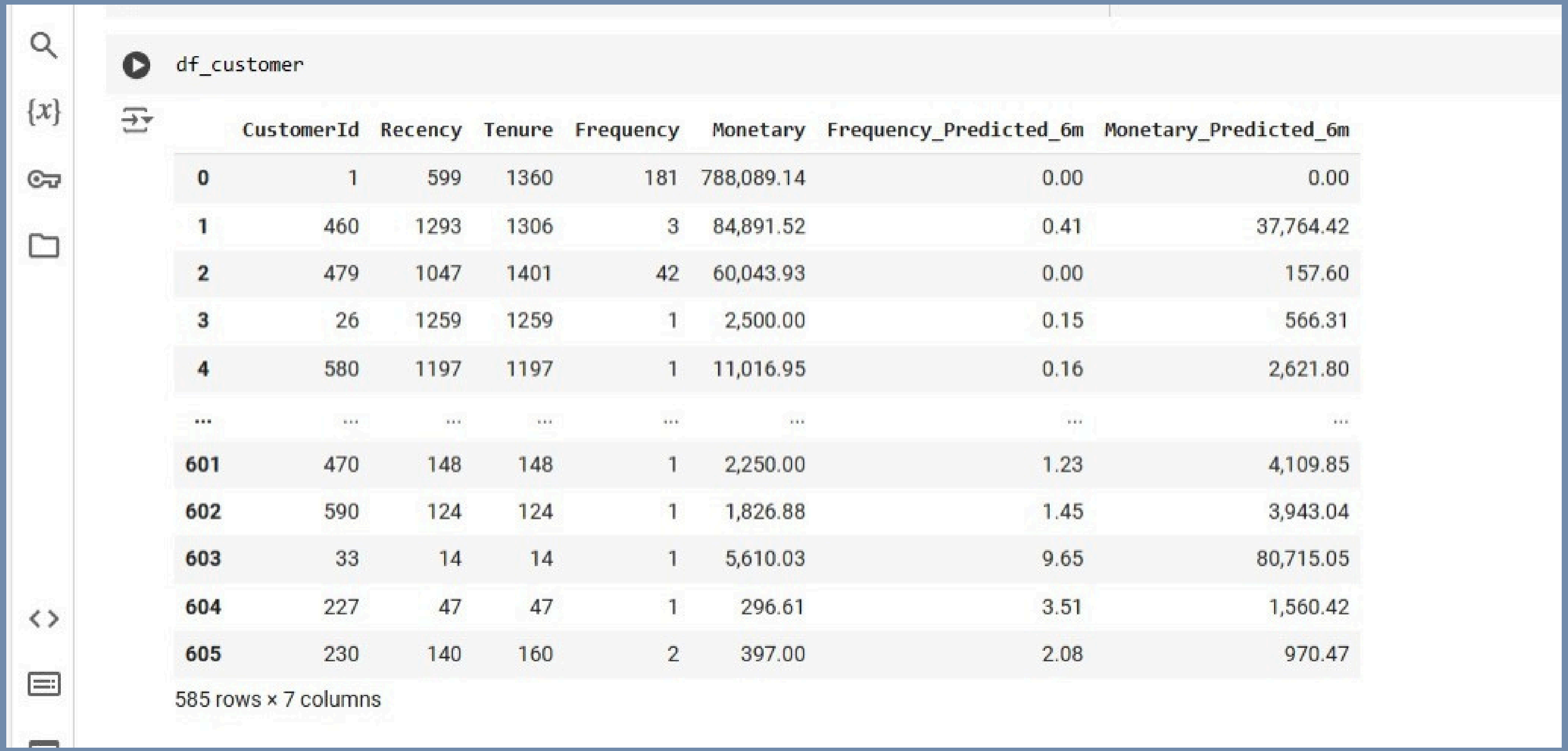
Modeling to Predict CLV

The first step in the modeling process is to install the Lifetime package, which provides essential tools and functions for analyzing and modeling customer data, particularly for Customer Lifetime Value (CLV) analysis. The Lifetime package simplifies the calculation of CLV metrics, building churn prediction models, and conducting customer retention analysis. It offers various techniques and models that can be customized to meet business needs, enabling deeper insights into customer behavior and enhancing marketing and retention strategies.

Steps for Modeling:

1. BG/NBD Model:
   - Suitable for customers with non-zero variables.
   - Train the BG/NBD model.
   - Predict the purchase frequency for each customer over the next 6 months.
   - Visualize the results using plotting.
   - Assess the model fit.
2. Gamma-Gamma Model:
   - Applicable if there is a low correlation between frequency (F) and monetary value (M).
   - Train the Gamma-Gamma model.
   - Predict the monetary value for each customer over the next 6 months.

CLV Modeling Result



```
df_customer
```

	CustomerId	Recency	Tenure	Frequency	Monetary	Frequency_Predicted_6m	Monetary_Predicted_6m
0	1	599	1360	181	788,089.14	0.00	0.00
1	460	1293	1306	3	84,891.52	0.41	37,764.42
2	479	1047	1401	42	60,043.93	0.00	157.60
3	26	1259	1259	1	2,500.00	0.15	566.31
4	580	1197	1197	1	11,016.95	0.16	2,621.80
...	...	...	...	...	...	...	...
601	470	148	148	1	2,250.00	1.23	4,109.85
602	590	124	124	1	1,826.88	1.45	3,943.04
603	33	14	14	1	5,610.03	9.65	80,715.05
604	227	47	47	1	296.61	3.51	1,560.42
605	230	140	160	2	397.00	2.08	970.47

585 rows × 7 columns

Analysis of customer churn showed that Random Forest emerged as the most effective model, with the highest accuracy, precision, recall, and F1 value of 0.846394. This signifies Random Forest' s superior ability to accurately predict customer churn. Turning to the Customer Lifetime Value (CLV) analysis, some important insights emerged. The average CLV for all customers is $ 84,238.88, underscoring the revenue contribution of each customer over the duration of their relationship. Notably, the distribution of CLV shows significant variability among customers, which emphasizes the importance of prioritizing customers who have higher CLV values to increase overall revenue. Factors that influence CLV, such as purchase frequency, value, and relationship length, are critical. Customers with higher frequency, value, and longer relationships tend to generate higher CLV. Recommendations for companies are to focus efforts on customers with high CLV, increase purchase frequency and value, and expand customer relationships. Implementing these strategies promises to increase overall CLV and improve profitability

Summary And Insight

- **Average CLV:** The average CLV for all customers is $84,238.88, indicating the average revenue contribution of each customer throughout their relationship with the company.
- **CLV Distribution:** There is a significant variation in CLV values among customers. Some customers have much higher CLV values than others, suggesting that focusing on these high-CLV customers can substantially boost the company's overall revenue.
- **Factors Influencing CLV:** Key factors affecting CLV include purchase frequency, purchase value, and relationship duration. Customers who purchase more frequently, buy higher-value products, and maintain longer relationships with the company tend to have higher CLV.
- **Customer Segmentation:** Segmenting customers based on their CLV can help in tailoring marketing strategies. High-CLV customers can be targeted with premium offers and personalized services, while strategies to increase engagement and retention can be devised for lower-CLV customers.
- **Personalized Marketing:** Implementing personalized marketing campaigns based on individual customer behavior and preferences can enhance customer satisfaction and loyalty, ultimately increasing CLV.

Summary And Insight

- **Customer Retention Strategies:** Developing targeted retention strategies for high-value customers, such as loyalty programs, exclusive discounts, and proactive customer service, can help in maintaining long-term relationships and maximizing CLV.
- **Upselling and Cross-Selling:** Identifying opportunities for upselling and cross-selling to high-CLV customers can further enhance their value. Offering complementary products or premium versions of existing products can increase the average transaction value.
- **Lifecycle Marketing:** Implementing lifecycle marketing strategies that address different stages of the customer journey—from acquisition to retention—can help in optimizing CLV. This includes welcome offers for new customers, regular engagement for active customers, and win-back campaigns for inactive customers.
- **Predictive Analytics:** Utilizing predictive analytics to anticipate customer behavior and needs can help in preemptively addressing potential churn risks and identifying opportunities for increasing customer value.
- **Continuous Monitoring and Improvement:** Regularly monitoring CLV metrics and adjusting strategies based on real-time data and insights can ensure that the company remains responsive to changing customer behaviors and market conditions, thus continually enhancing profitability.

THANK YOU

We look forward to working with you

OFFICE

annizarmegas@gmail.com

@annizamega

Linkedin