**A PROJECT REPORT ON**

**ORDERS IN ONTARIO**

****

**By**

**Tejas Dhatrika**

**Amulya Gutha**

**Venkata Sai Nandella**

**Bajan Basha Shaik**

**Sesha Sai Meka**

**Shashank Reddy**

**Vasu Bejugam**

**Manzar shajiuddin ahmed**

**Pavan Kalyan Meda**

**Yamini Guduru**

**TABLE OF CONTENTS**

**CONTENTS**  **Page No**

ABSTRACT ……………..…………………………………………………………………. 3

**Chapter-1 INTRODUCTION**

* 1. Introduction………………………………………………………………………………4
  2. Methodology………………………………………………………………………..........5

**Chapter-2**

2.1 Algorithms Used…………………………………………………………………………6

2.2 Metrics ……………………………………………………………………………..........8

**Chapter-3 CODE AND OUTPUT**………………………………………………………….9

**Chapter-4 CONCLUSION**………………………………………………………………...46

**Chapter-5 REFERENCES**………………………………………………………………...47

**ABSTRACT**

The Orders in Ontario Project intends to use advanced analytics to extract insights from a large dataset of orders. This report summarises the project's essential phases, which include data cleaning, feature engineering, and building the model. The team methodically addressed data quality issues, engineered innovative features, and built two classification models—Logistic Regression and Decision Tree Classification. Performance measures were used to evaluate the models, establish the groundwork for fine-tuning, investigate ensemble approaches, and develop a deployment strategy. This abstract provides a concise assessment of the project's development, highlighting the team's commitment to data-driven decision-making and collaborative efforts in uncovering important patterns within the dataset.

**CHAPTER-1**

* 1. **INTRODUCTION**

The Orders in Ontario Project is a crucial endeavour in the landscape of modern business and e-commerce, focused at harnessing the power of machine learning to extract useful insights from a vast dataset of orders. The project concentrates around conducting in-depth analyses of client transactions in order to identify trends that can drive strategic decisions and improve overall operational efficiency.

**Background:**

The study delves into a varied dataset containing a spectrum of order-related factors, emphasising the province of Ontario, Canada. These characteristics include anything from consumer demographics and payment methods to the finer points of the order processing pipeline. The importance of this endeavour stems from its potential to identify hidden links, forecast order outcomes, and contribute to better decision-making.

**Objectives:**

The key goals of the Ontario Machine Learning Project are as follows:

* **Understanding and Cleaning Data:**

Learn about the structure and unique features of the dataset.Implement rigorous data cleaning practices to ensure the quality and integrity of your data.

* **Feature Engineering:**

Introduce innovative features that capture the core of customer behaviour and transactional dynamics to innovate. Improve the dataset to boost the models' prediction capability.

* **Model Building and Evaluation:**

Putting two classification models into action: logistic regression and decision tree classification. To determine predictive accuracy, evaluate model performance using industry-standard measures.

* **Analytics and Visualization:**

The project moved on to analytics and visualization following the modelling phase. To discover detailed insights, the team used advanced approaches to analyze model outputs, recognize trends, and undertake exploratory data analysis. Developing interactive dashboards with visualizations has improved the project's ability to communicate complicated findings in an understandable and actionable manner.

* **Deployment on GitHub:**

Deploying the machine learning models on GitHub was an important stage in the project's evolution. This strategic approach promotes collaboration and transparency while also ensuring accessibility. The models, codebase, and documentation are all stored in a centralized repository, which serves as a hub for ongoing development and version control.

* **Future Steps and Optimization:**

For optimal performance, fine-tune models depending on evaluation findings. Investigate ensemble approaches and advanced ways to improve forecasting skills. Create a plan for incorporating machine learning insights into operational operations.

**Significance:**

The Orders in Ontario Project is situated at the intersection of data science and business intelligence, with the potential to transform how orders are processed, interpreted, and optimized. The project's goal is to provide stakeholders with actionable insights that help drive strategic choices, improve customer experiences, and contribute to the overall success of Ontario's e-commerce operations by interpreting patterns within the dataset.

In the following sections of this report, we will go over the project's thorough progression, from data cleaning and feature engineering to the implementation and evaluation of machine learning models. The project team's collaborative efforts and commitment to data-driven decision-making highlight the significance of this program in the changing field of machine learning applications.

**1.2 METHODOLOGY**

The methodology used in the Ontario Machine Learning Project is systematic and iterative. It began with a careful data cleaning process, during which the team rectified missing numbers and guaranteed the dataset's general integrity. Following that, feature engineering took place, which involved the development of new variables to capture hidden aspects of customer behavior and transactional dynamics. Following that, two categorization models—Logistic Regression and Decision Tree—were developed, each exploiting its capabilities in predictive analysis. The models were trained and assessed using industry-standard metrics, with a focus on interpreting the outputs of the models to derive relevant insights. Analytics and visualization were critical in post-modeling, allowing for pattern discovery and the development of interactive dashboards for stakeholders.

The models were hosted on GitHub during the deployment phase to ensure accessibility and version control. A user-centric approach characterizes the process, with ongoing feedback loops directing model adaption and future enhancements. This iterative process, which emphasizes cooperation and transparency, sets the project for long-term success and adaptation to changing business needs.

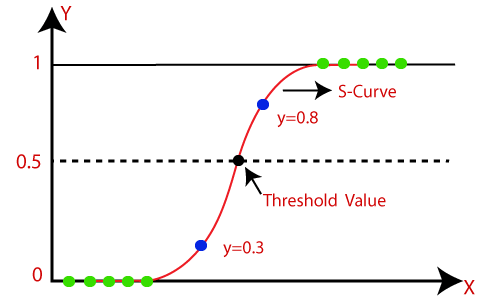
**CHAPTER-2**

**2.1 ALGORITHMS USED**

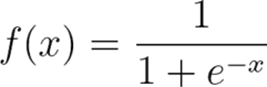
**Logistic regression**

Logistic regression is a supervised machine learning technique that predicts the likelihood of an outcome, occurrence, or observation to perform binary classification tasks. The model produces a binary or categorical result with only two possible outcomes: yes/no, 0/1, or true/false. Logical regression examines the relationship between multiple independent variables and categorizes data. It is often used in predictive modelling, in which the model calculates the mathematical likelihood of whether or not an occurrence corresponds to a given category.

For example, 0 denotes a negative class and 1 denotes a positive class. In binary classification issues, where the outcome variable exposes one of two categories (0 or 1), logistic regression is typically utilized.



**Assumptions and Logistic Regression Equation**

****

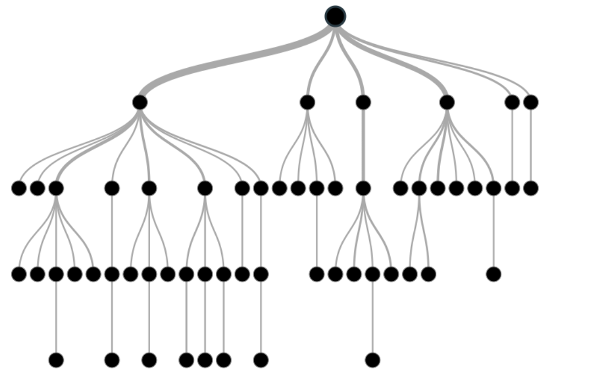
To map predictions and their probabilities, logistic regression employs a logistic function known as a sigmoid function. The sigmoid function is an S-shaped curve that converts any real value to a 0 to 1 range.

Furthermore, if the sigmoid function output (estimated probability) exceeds an established limit on the graph, the model predicts that the instance belongs to that class. The model predicts that the instance does not belong to the class if the calculated probability is less than or equal to the predefined threshold.

**Decision Tree**

A decision tree is a non-parametric supervised learning algorithm that can be used to perform classification and regression problems. It is a hierarchy tree with a root node, branches, internal nodes, and leaf nodes. Decision trees are used for regression and classification applications, resulting in simple models.

A decision tree is a hierarchical decision support model that displays options and their probable outcomes, including chance occurrences, resources expenses, and utility. This non-parametric, supervised learning algorithmic approach employs conditional control statements and is suitable for both classification and regression applications. A root node, branches, internal nodes, and leaf nodes form a hierarchical, tree-like structure.



**2.2 METRICS**

In the Ontario Machine Learning Project, the evaluation of the classifying models included the use of numerous industry-standard measures to assess their effectiveness. Metrics that are regularly used include:

* **Accuracy:**

Accuracy is the fraction of correctly classified instances among every instance. It provides a broad assessment of the model's accuracy.

* **Precision:**

Precision focuses on positive accuracy of prediction, calculating the proportion of real positive predictions among all instances projected as positive. It is especially important when the cost of false positives is significant.

* **Recall Sensitivity (or True Positive Rate):**

The fraction of true positive predictions out of all actual positive cases is calculated by recall. When the cost of false negatives is significant, it is critical to emphasize the model's ability to capture all positive events.

* **F1 Score:**

The harmonic mean of precision and recall is the F1 score. It gives a balanced measure that takes into account both false positives and false negatives, making it useful when class distribution is uneven.

* **Confusion matrix:**

The confusion matrix provides a precise breakdown of true positive, true negative, false positive, and false negative predictions, providing a comprehensive picture of the model's performance.

* **Curve of Receiver Operating Characteristic (ROC):**

At various categorization thresholds, the ROC curve depicts the trade-off between true positive rate and false positive rate. The area under the ROC curve (AUC-ROC) is a popular statistic for assessing model discrimination abilities.

These metrics provide a complete assessment of the models' predictive skills, allowing the project team to make informed decisions about model effectiveness and possible areas for development. The metrics chosen are frequently adapted to the project's specific aims and requirements, as well as the nature of the information.

**CHAPTER -3**

**CODE AND OUTPUT**

#Importing data

**import** pandas **as** pd

**import** numpy **as** np

**from** matplotlib **import** pyplot **as** plt

**import** seaborn **as** sns

**import** warnings

*# Ignore all warnings*

warnings**.**filterwarnings("ignore")

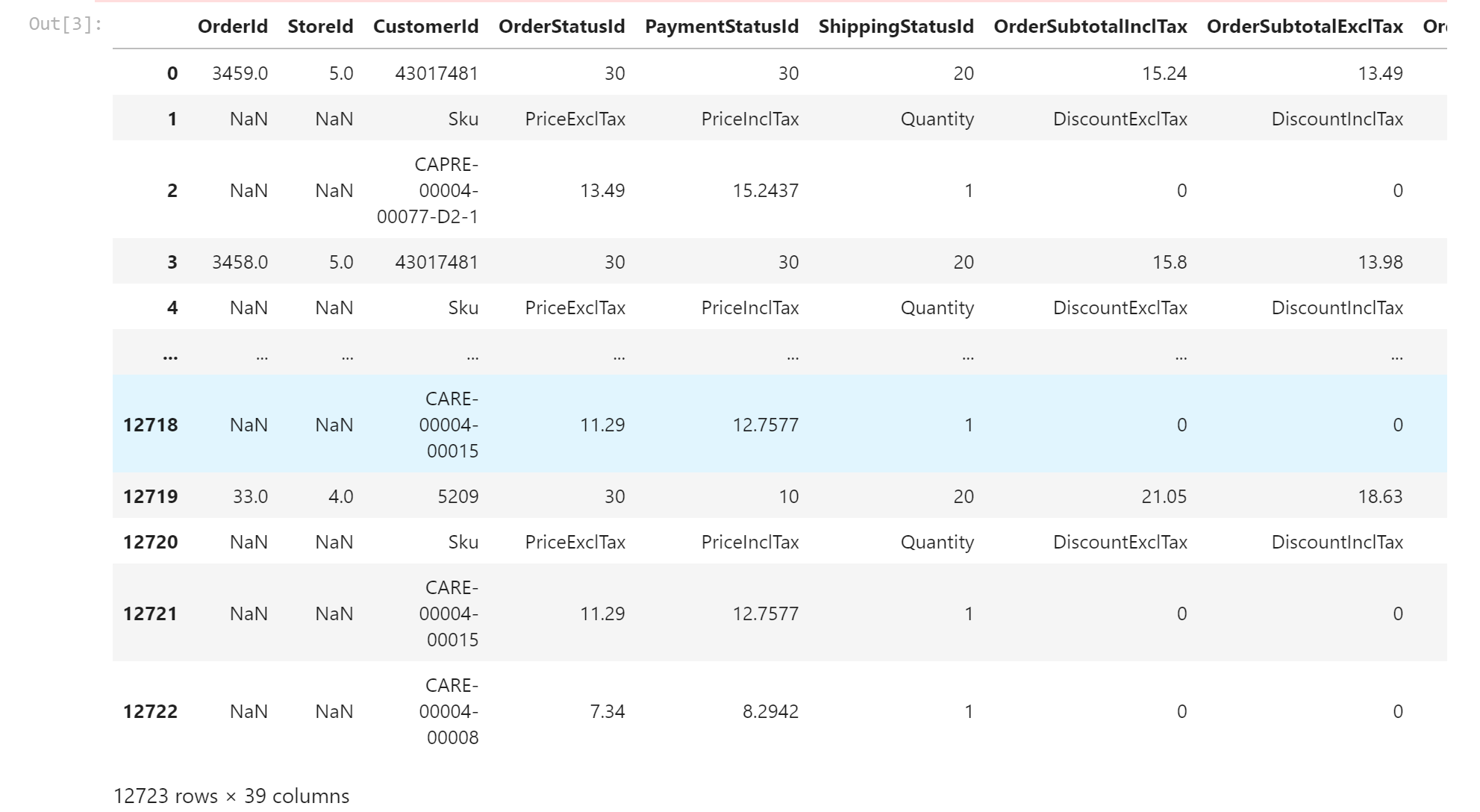
*# Your code here*

*# Reset the warnings to default behavior (optional)*

warnings**.**filterwarnings("default")

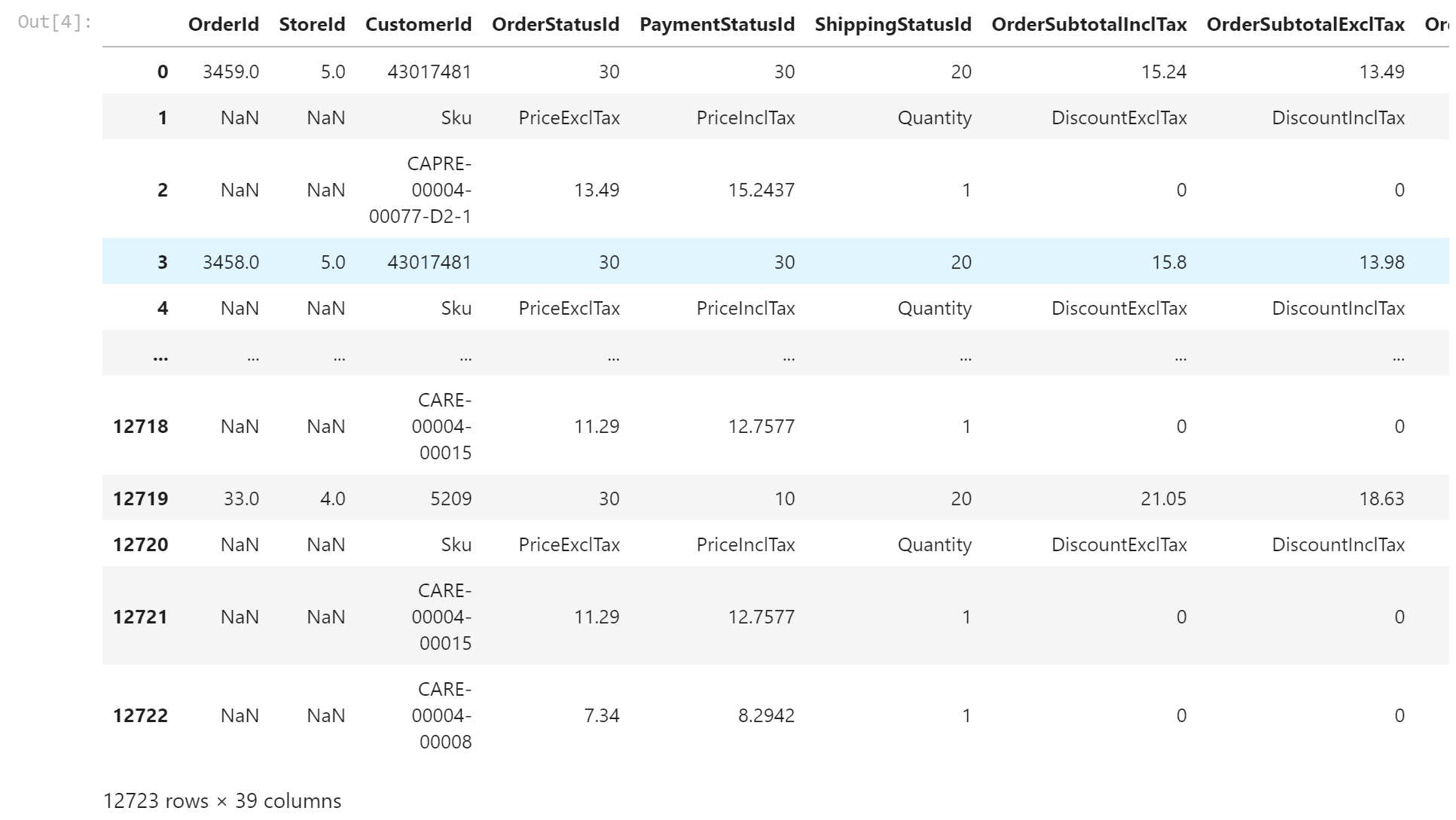
df **=** pd**.**read\_excel("orders\_A1.xlsx")

df

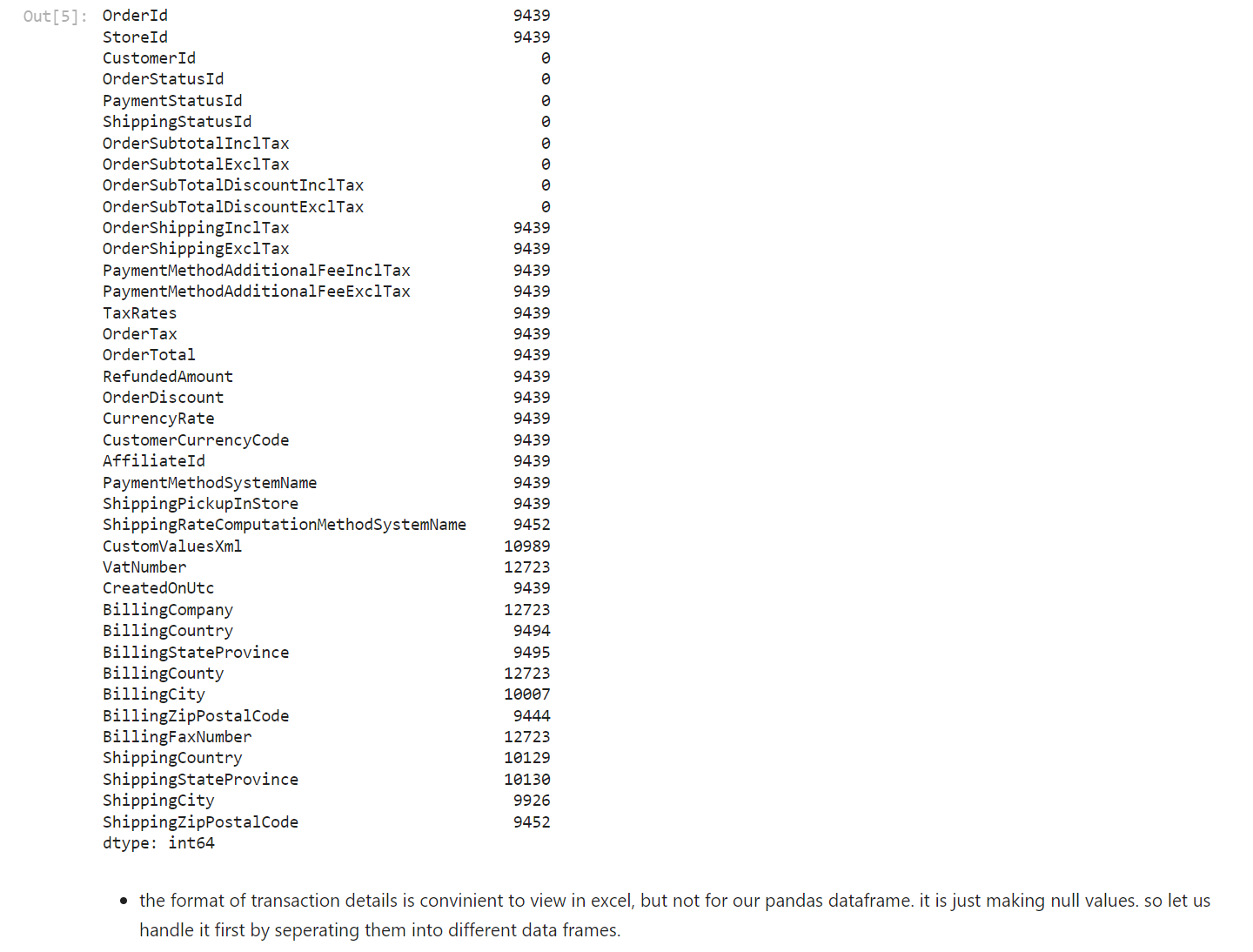


df\_copy **=** df**.**copy()

df\_copy



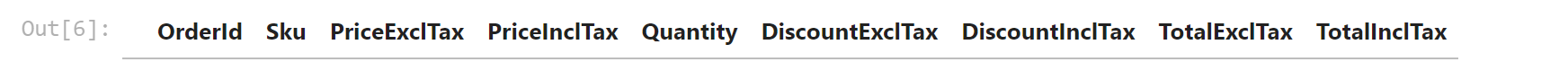
df**.**isna()**.**sum()



column\_names **=** ["OrderId","Sku", "PriceExclTax", "PriceInclTax","Quantity","DiscountExclTax","DiscountInclTax","TotalExclTax","TotalInclTax"]

product\_details\_df **=** pd**.**DataFrame(columns**=**column\_names)

product\_details\_df



**for** index, row **in** df**.**iterrows():

**if** pd**.**isna(row["OrderId"]):

**if** row["CustomerId"]**==**"Sku":

df **=** df**.**drop(index)

**continue**

product\_detail **=** {

"OrderId": current\_OrderId,

"Sku": row["CustomerId"],

"PriceExclTax": row["OrderStatusId"],

"PriceInclTax": row["PaymentStatusId"],

"Quantity": row["ShippingStatusId"],

"DiscountExclTax": row["OrderSubtotalInclTax"],

"DiscountInclTax": row["OrderSubtotalExclTax"],

"TotalExclTax": row["OrderSubTotalDiscountInclTax"],

"TotalInclTax": row["OrderSubTotalDiscountExclTax"],

}

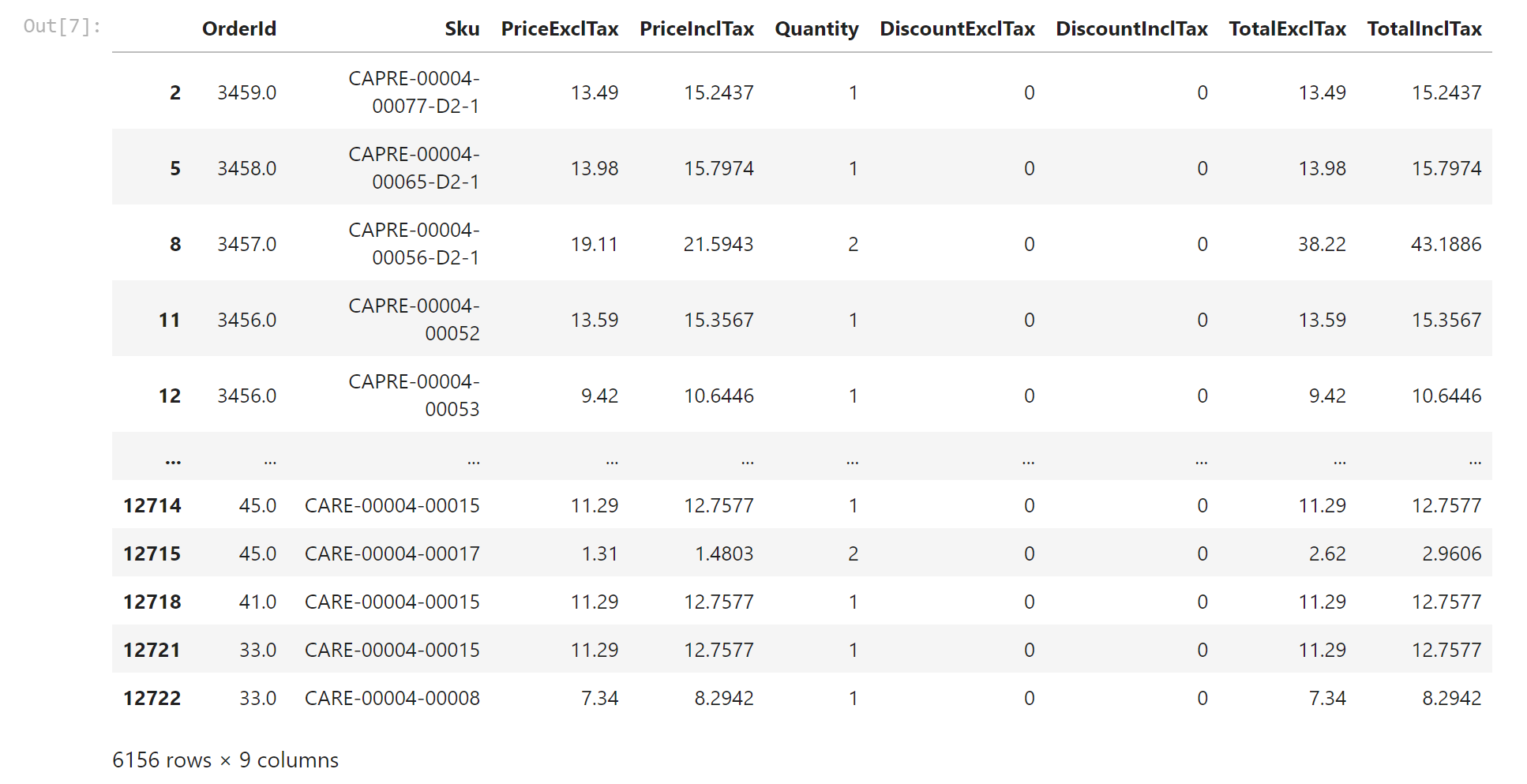
product\_details\_df **=** pd**.**concat([product\_details\_df,pd**.**DataFrame(product\_detail,index**=**[index,])])

df **=** df**.**drop(index)

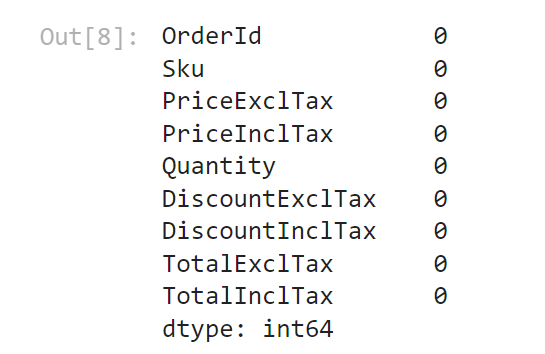
**else** :

current\_OrderId **=** row["OrderId"]

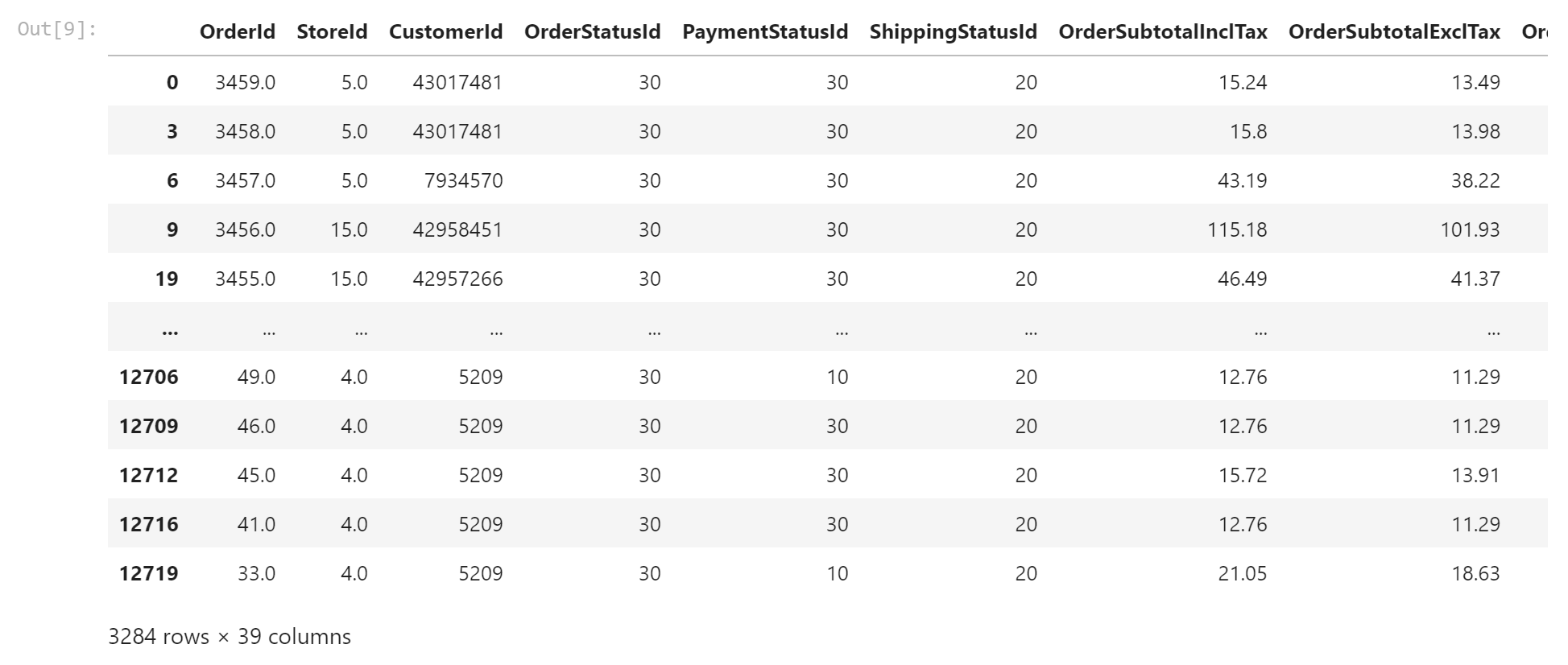
product\_details\_df



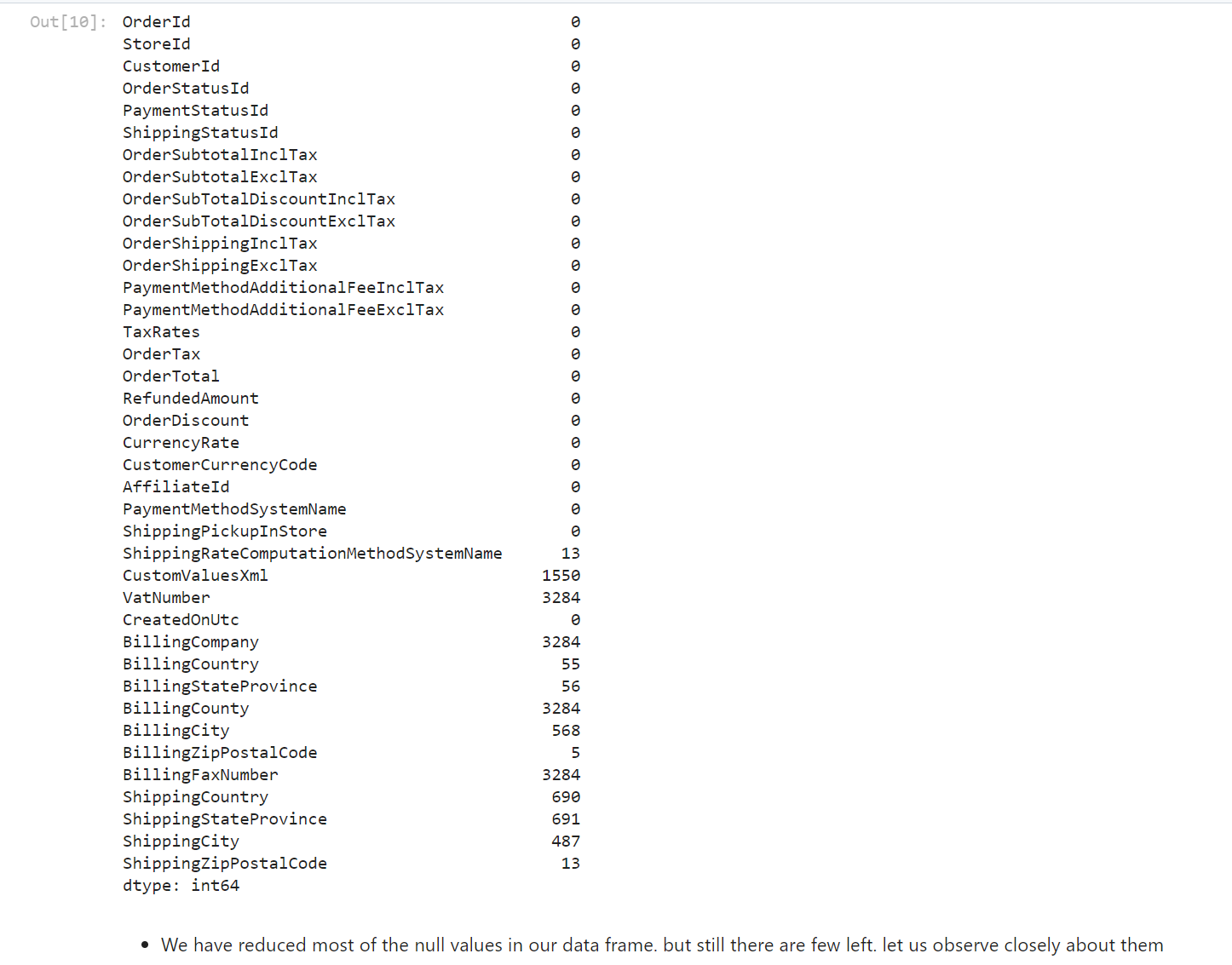
product\_details\_df**.**isna()**.**sum()



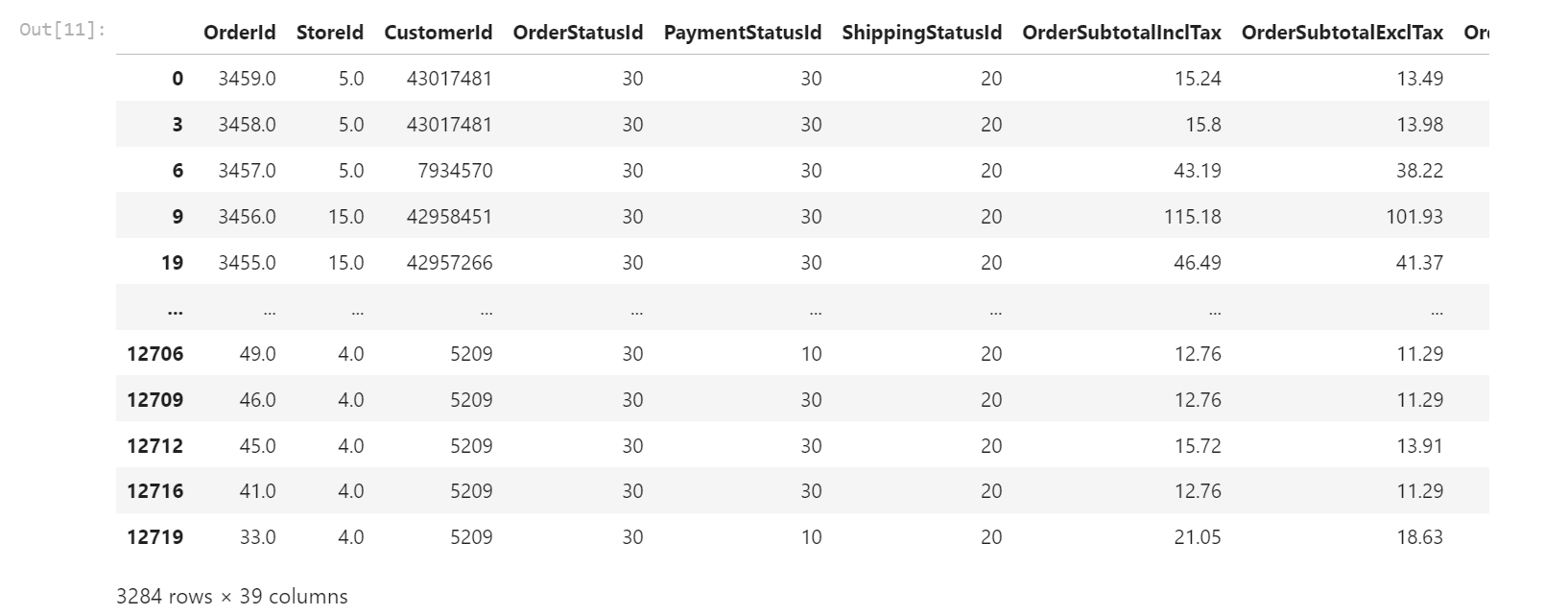
df



df**.**isna()**.**sum()

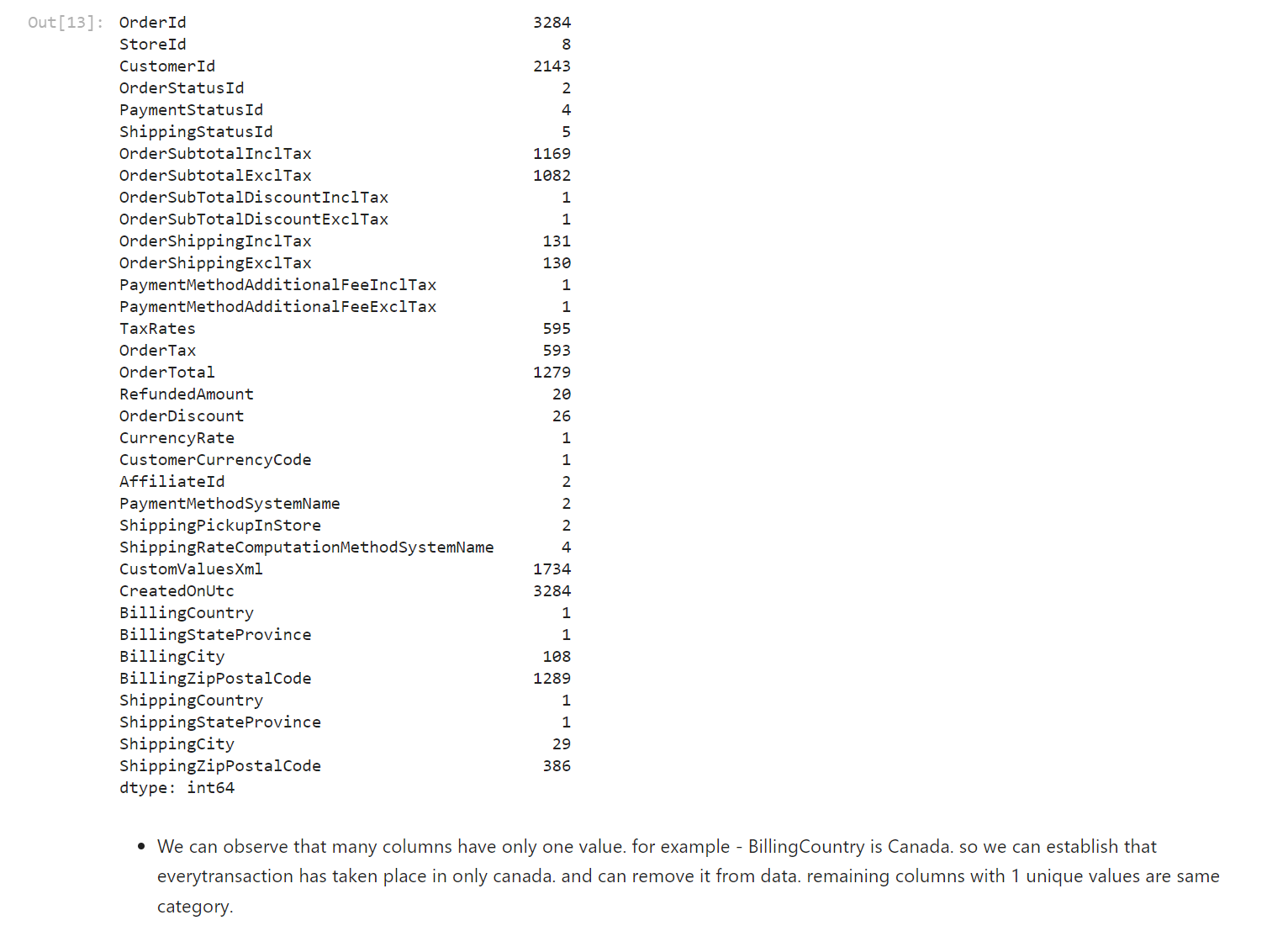


df[df**.**isnull()**.**any(axis**=**1)]



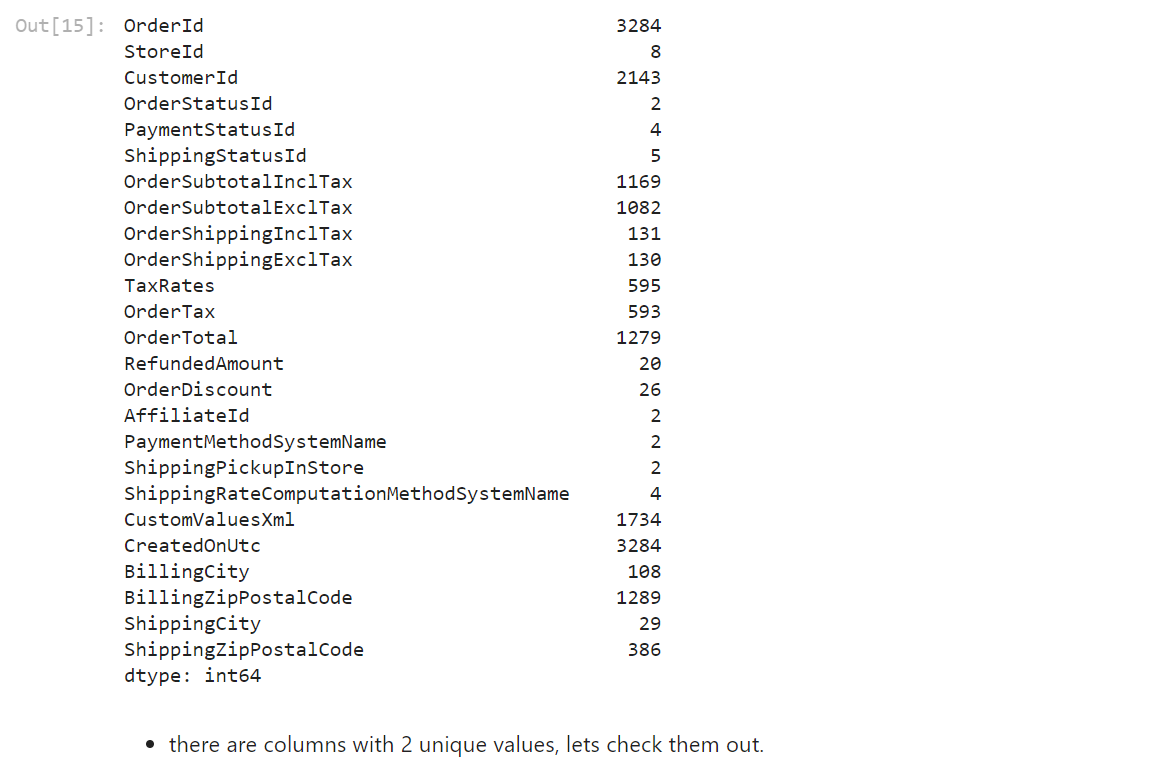
df**.**drop(['VatNumber','BillingCompany','BillingCounty','BillingFaxNumber'],axis**=**1,inplace**=True**)

df**.**nunique()

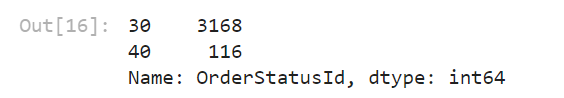


df**.**drop(['OrderSubTotalDiscountInclTax','OrderSubTotalDiscountExclTax','PaymentMethodAdditionalFeeInclTax','PaymentMethodAdditionalFeeExclTax',"CurrencyRate","CustomerCurrencyCode","BillingCountry","BillingStateProvince","ShippingCountry","ShippingStateProvince"],axis**=**1,inplace**=True**)

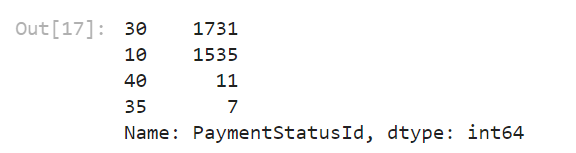
df**.**nunique()



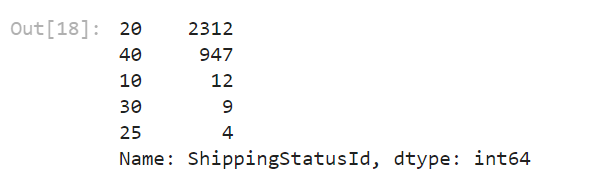
df["OrderStatusId"]**.**value\_counts()



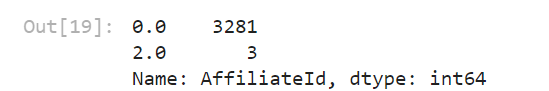
df["PaymentStatusId"]**.**value\_counts()



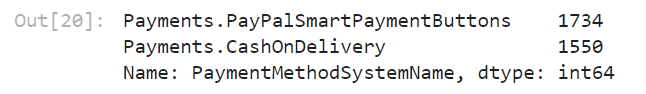
df["ShippingStatusId"]**.**value\_counts()



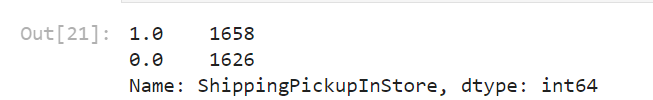
df["AffiliateId"]**.**value\_counts()



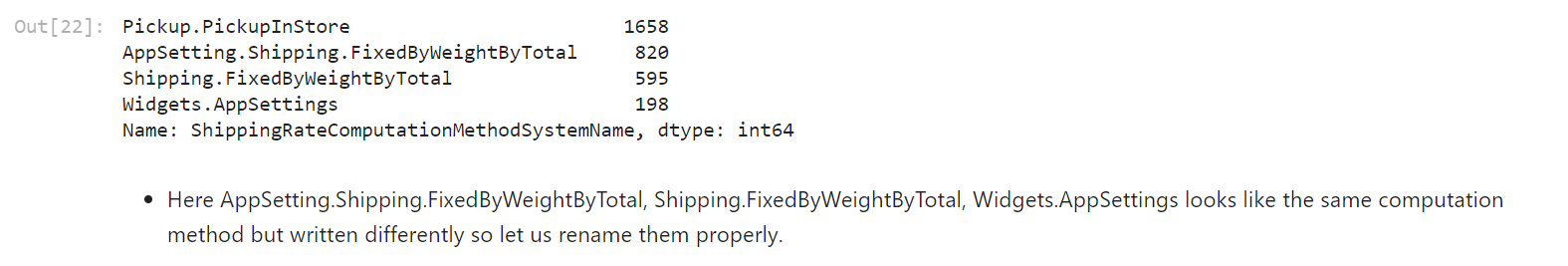
df["PaymentMethodSystemName"]**.**value\_counts()



df["ShippingPickupInStore"]**.**value\_counts()



df["ShippingRateComputationMethodSystemName"]**.**value\_counts()

****

df['ShippingRateComputationMethodSystemName'] = df['ShippingRateComputationMethodSystemName'].replace({

'Pickup.PickupInStore' : 'Pickup\_In\_Store',

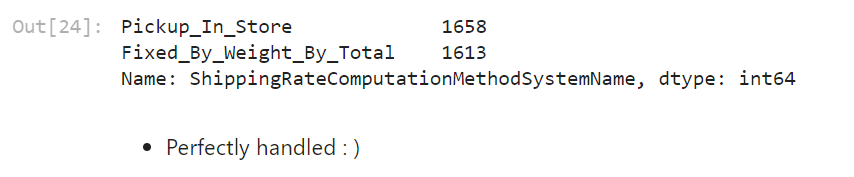
'AppSetting.Shipping.FixedByWeightByTotal' : 'Fixed\_By\_Weight\_By\_Total',

'Shipping.FixedByWeightByTotal' : 'Fixed\_By\_Weight\_By\_Total',

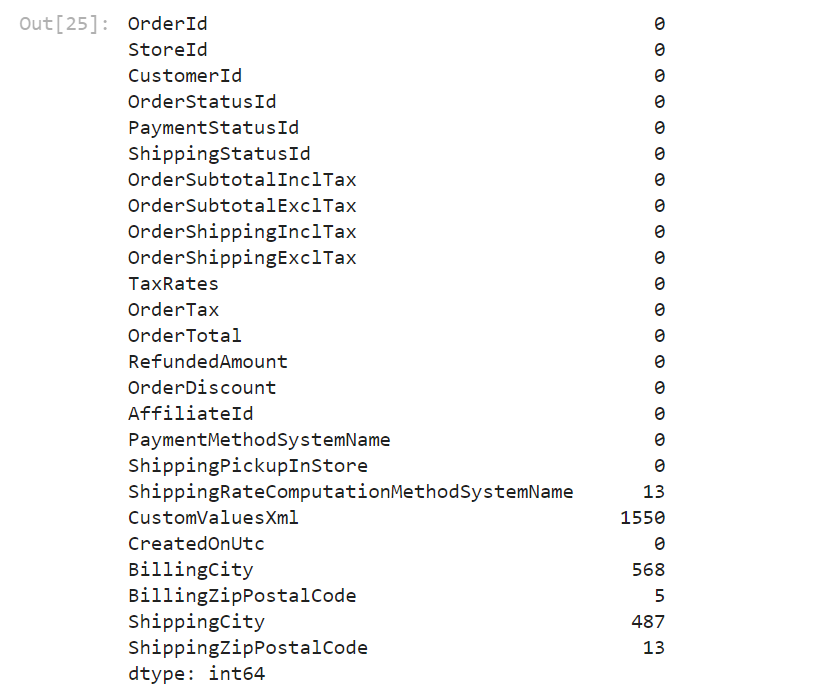
'Widgets.AppSettings' : 'Fixed\_By\_Weight\_By\_Total'

})

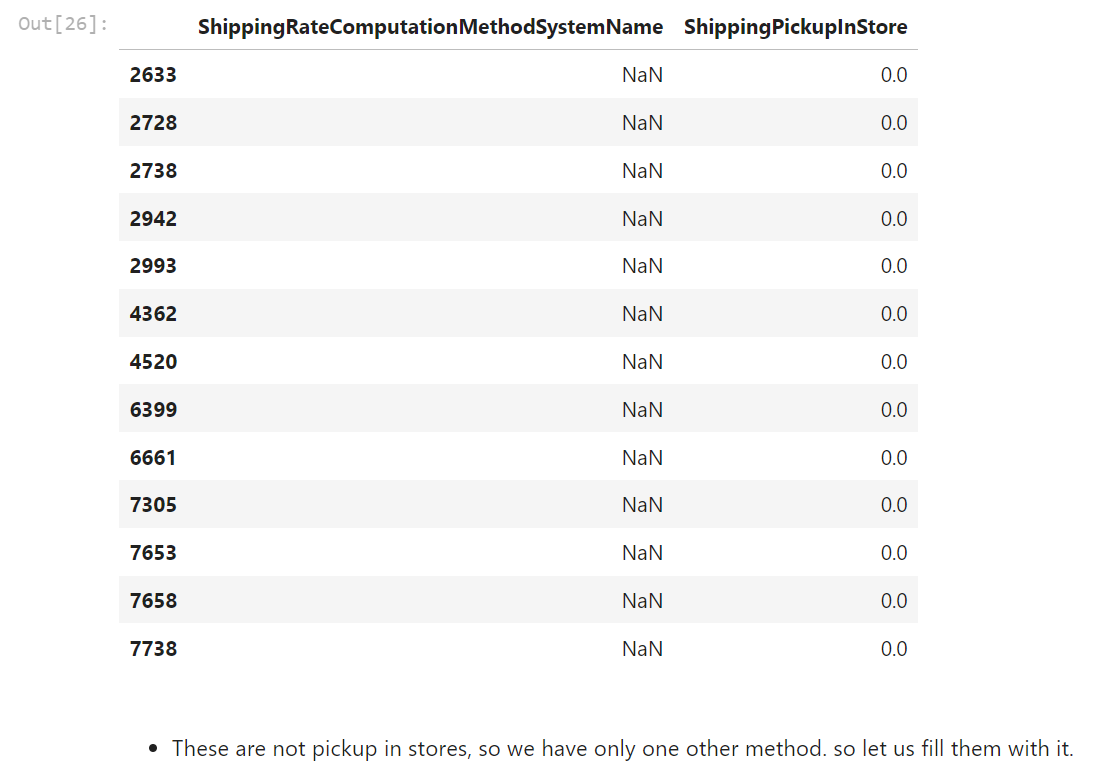
df["ShippingRateComputationMethodSystemName"].value\_counts()

****

df.isnull().sum()

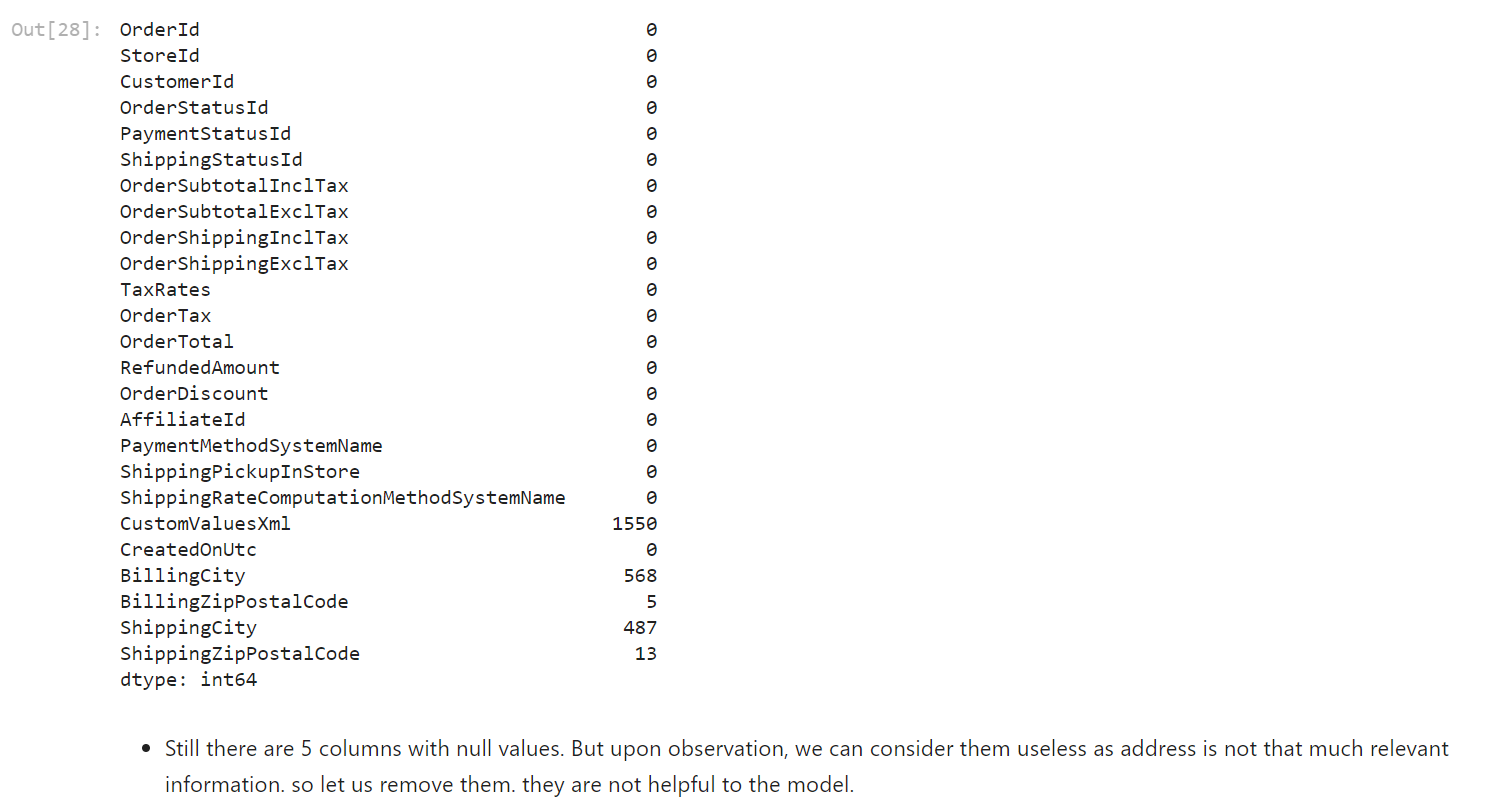
****

df[df["ShippingRateComputationMethodSystemName"].isnull()][["ShippingRateComputationMethodSystemName", "ShippingPickupInStore"]]



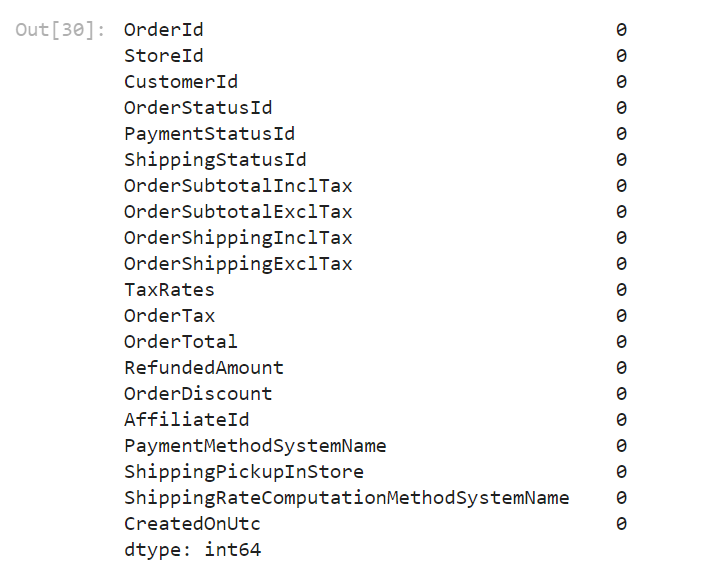
df["ShippingRateComputationMethodSystemName"]**.**fillna("Fixed\_By\_Weight\_By\_Total", inplace**=True**)

df**.**isna()**.**sum()

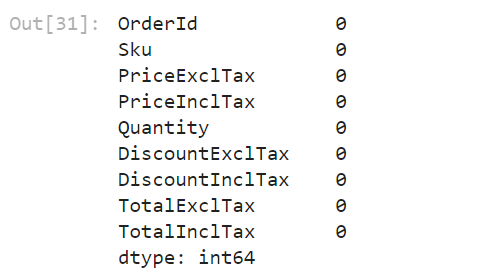


df**.**drop(['CustomValuesXml','BillingCity','BillingZipPostalCode','ShippingCity',"ShippingZipPostalCode"],axis**=**1,inplace**=True**)

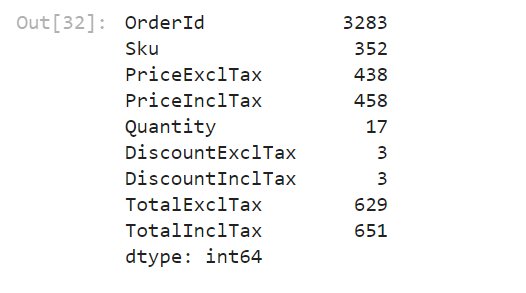
df**.**isna()**.**sum()



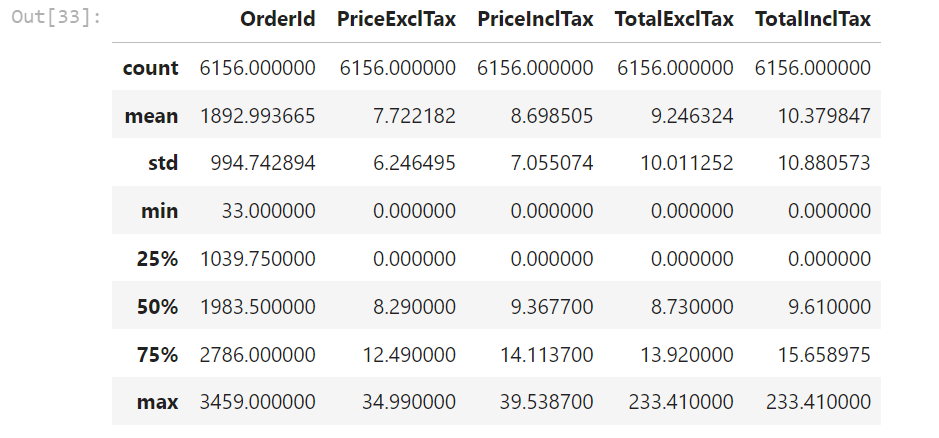
product\_details\_df**.**isna()**.**sum()



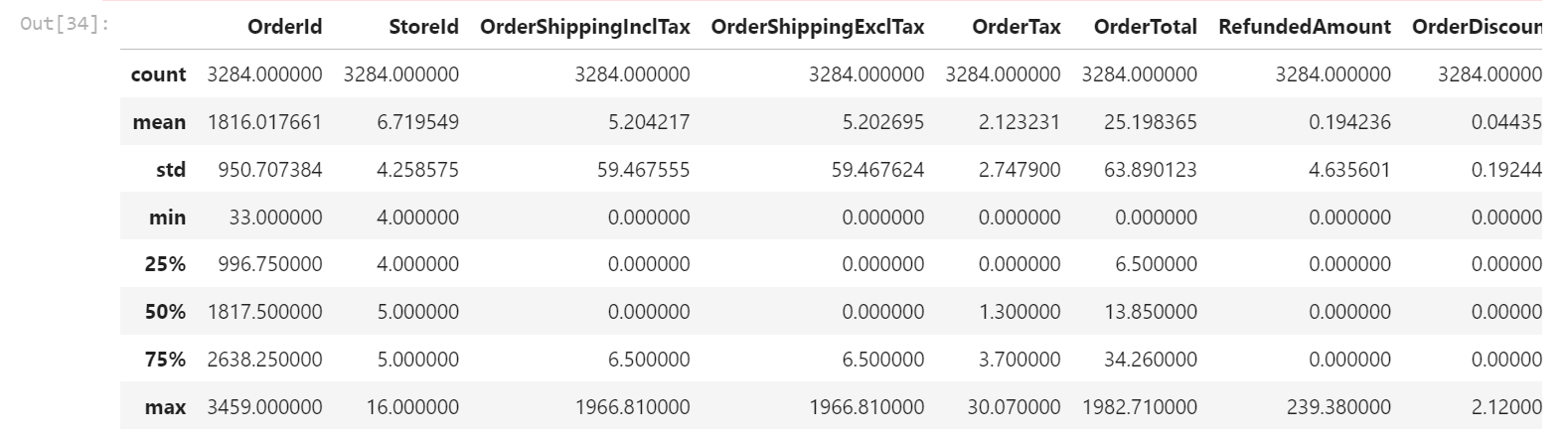
product\_details\_df**.**nunique()



product\_details\_df**.**describe()



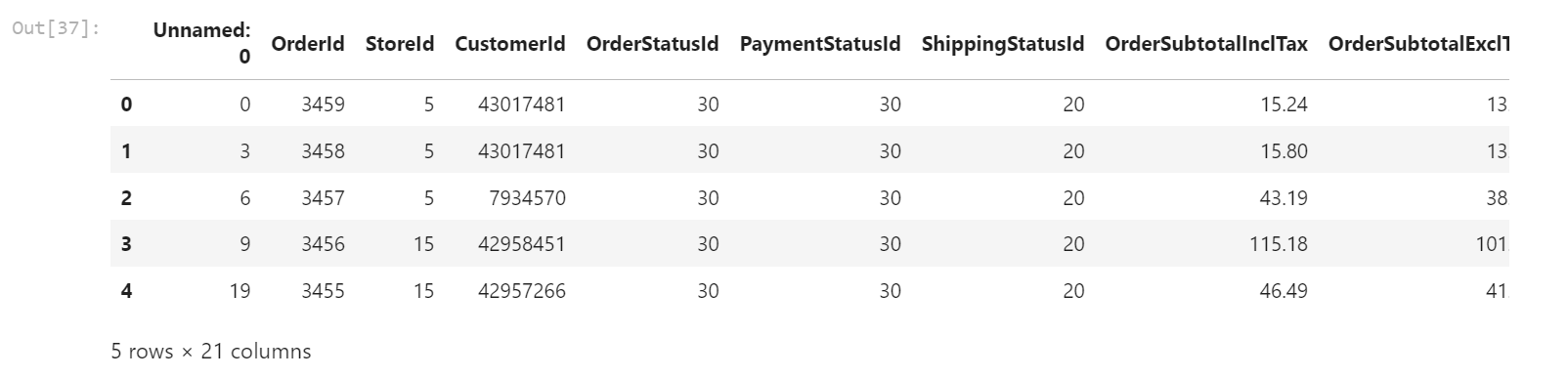
df**.**describe()



df**.**to\_excel("order11.xlsx")

data **=** pd**.**read\_excel("order11.xlsx")

data**.**head()



**import** seaborn **as** sn

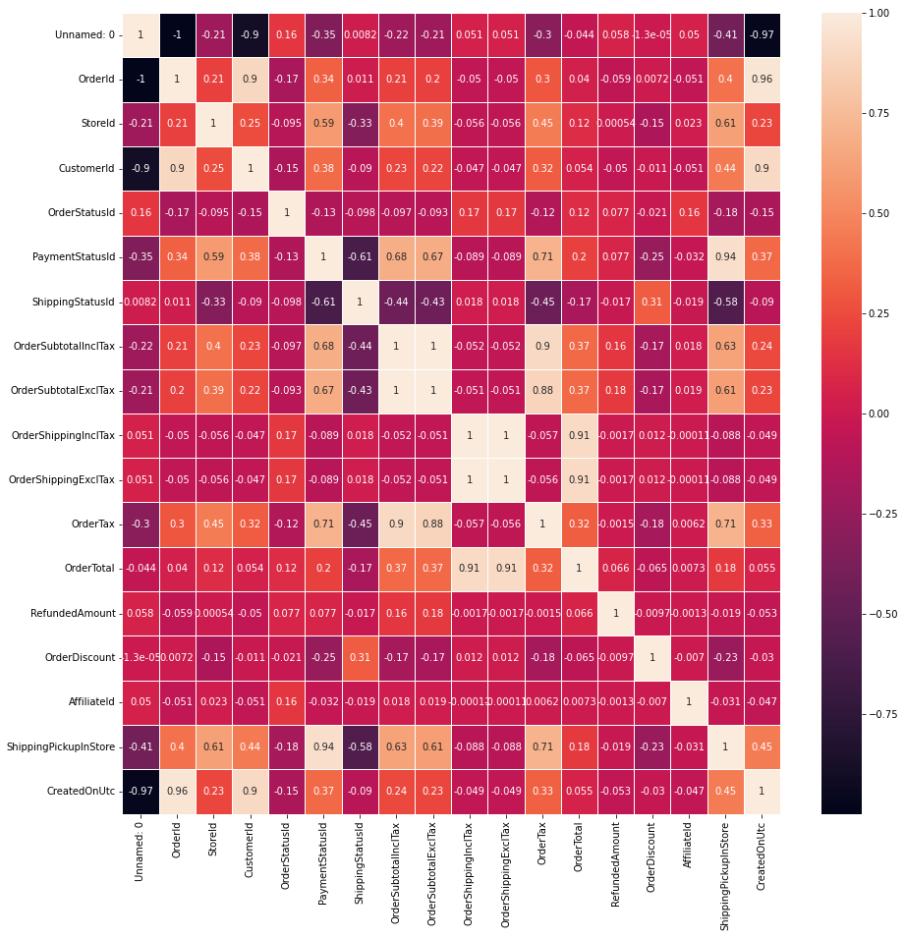
**import** matplotlib.pyplot **as** plt

corr\_matrix **=** data**.**corr()

fig, ax **=** plt**.**subplots(figsize**=**(15,15))

sn**.**heatmap(corr\_matrix, annot**=True**, linewidths**=**0.5, ax**=**ax)

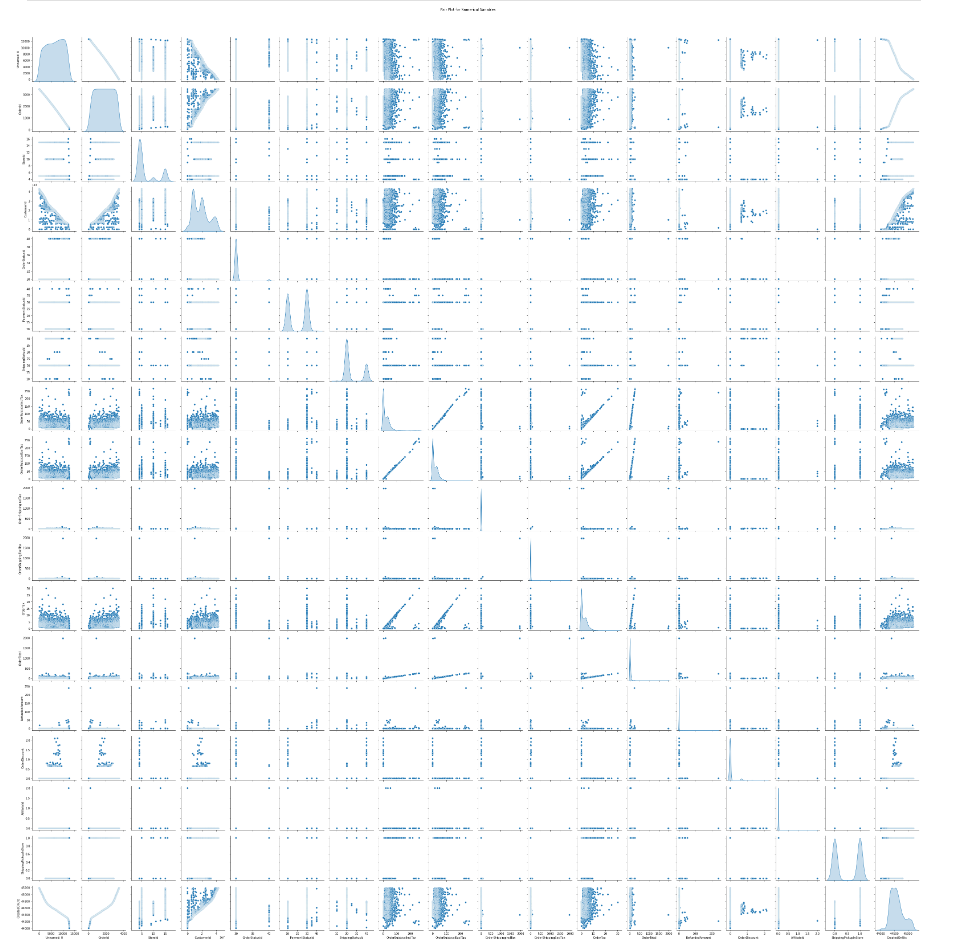
plt**.**show()



sn**.**pairplot(data, diag\_kind**=**'kde')

plt**.**suptitle('Pair Plot for Numerical Variables', y**=**1.02)

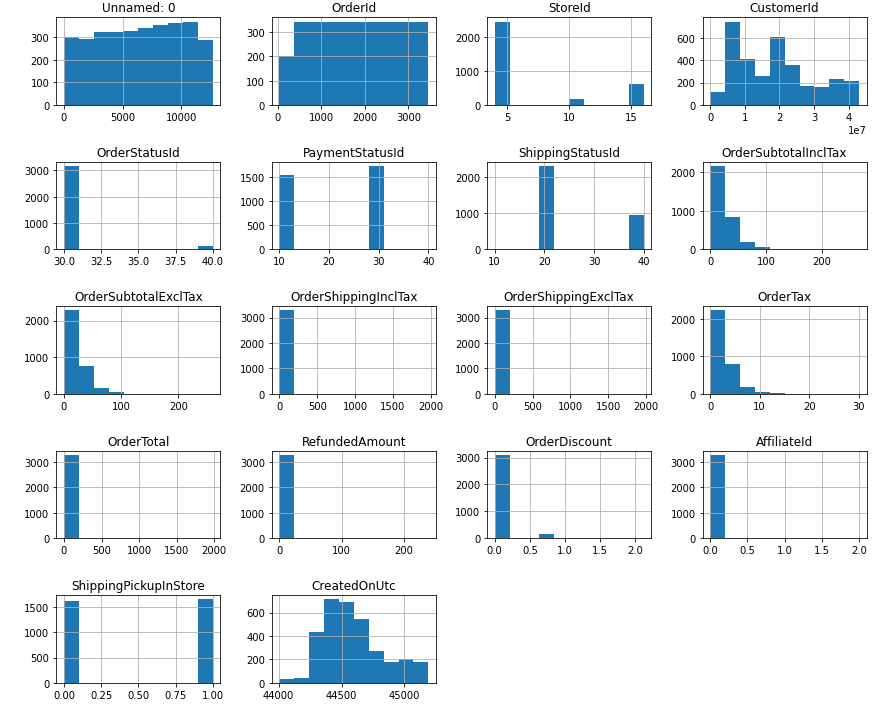
plt**.**show()



data**.**hist(figsize**=**(12, 10))

plt**.**tight\_layout()

plt**.**show()



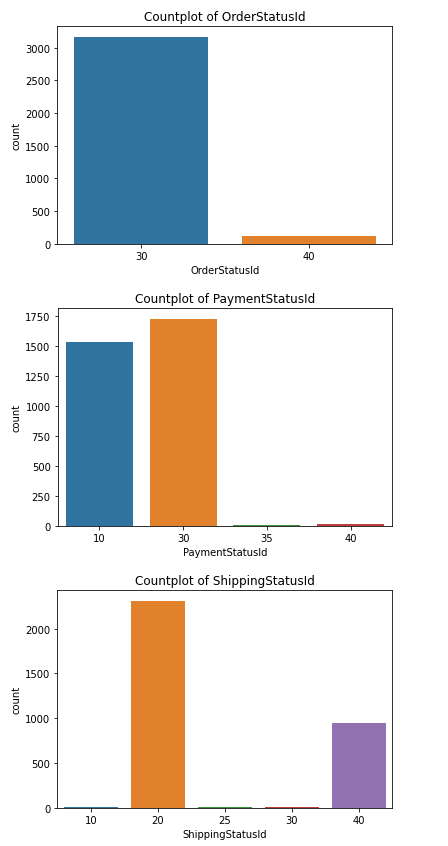
categorical\_columns **=** ['OrderStatusId', 'PaymentStatusId', 'ShippingStatusId']

**for** column **in** categorical\_columns:

sn**.**countplot(x**=**column, data**=**data)

plt**.**title(f'Countplot of {column}')

plt**.**show()

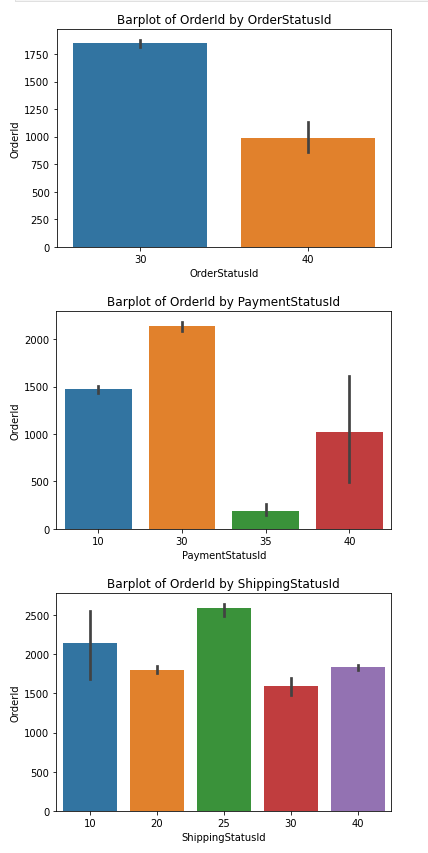


for column in categorical\_columns:

sn.barplot(x=column, y='OrderId', data=data)

plt.title(f'Barplot of OrderId by {column}')

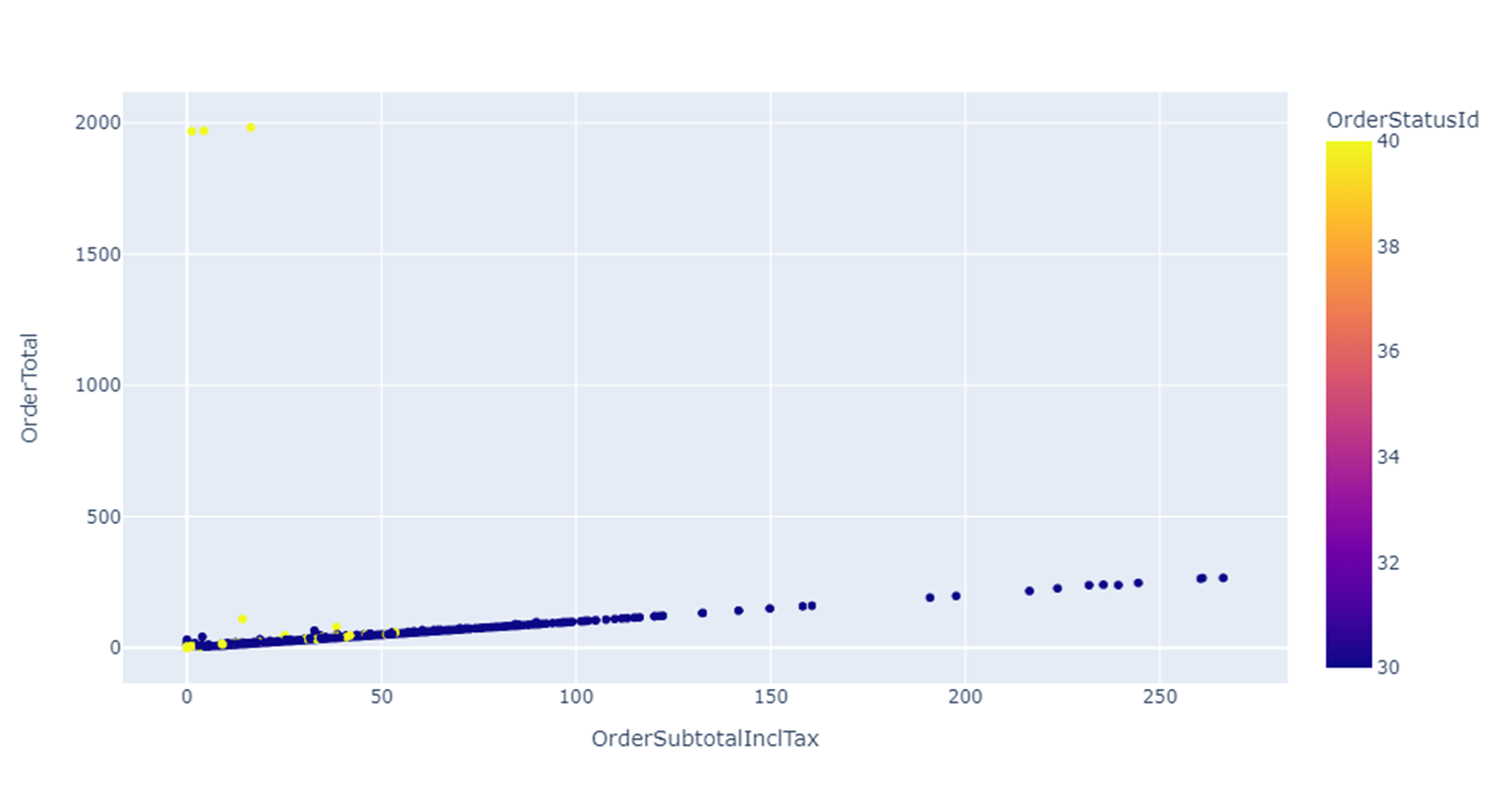
plt.show()



**import** plotly.express **as** px

fig **=** px**.**scatter(data, x**=**'OrderSubtotalInclTax', y**=**'OrderTotal', color**=**'OrderStatusId', hover\_data**=**['CustomerId'])

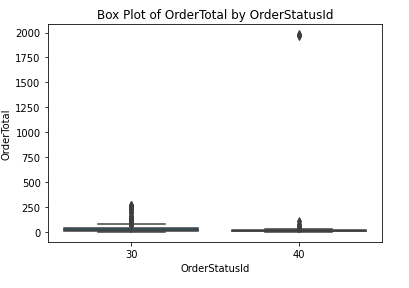
fig**.**show()



sn**.**boxplot(x**=**'OrderStatusId', y**=**'OrderTotal', data**=**data)

plt**.**title('Box Plot of OrderTotal by OrderStatusId')

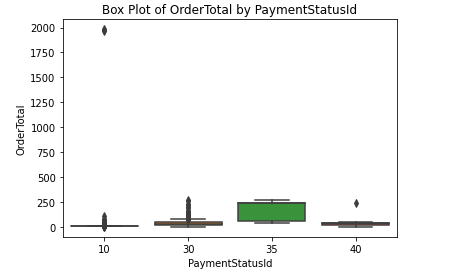
plt**.**show()



sn**.**boxplot(x**=**'PaymentStatusId', y**=**'OrderTotal', data**=**data)

plt**.**title('Box Plot of OrderTotal by PaymentStatusId')

plt**.**show()



sn**.**boxplot(x**=**'ShippingStatusId', y**=**'OrderTotal', data**=**data)

plt**.**title('Box Plot of OrderTotal by ShippingStatusId')

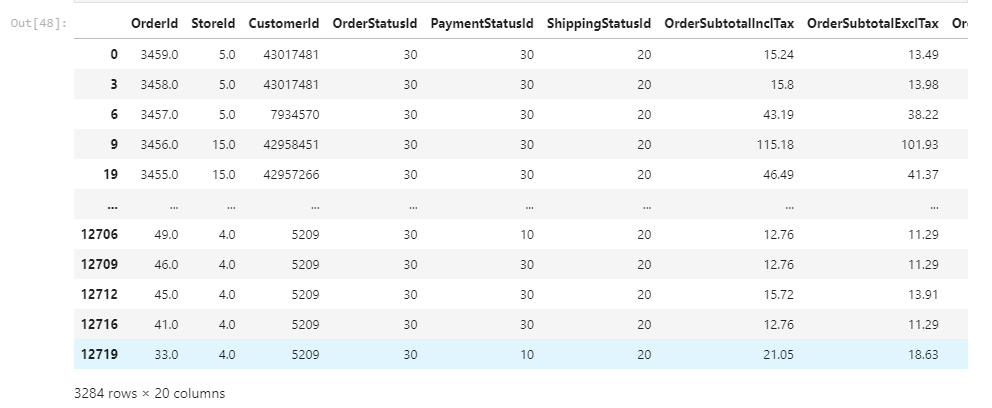
plt**.**show()



new\_df **=** df**.**copy()

new\_df.merge(product\_details\_df,on='OrderId',how='right')

new\_df



*# Sample data*

data1 **=** new\_df

data2 **=** product\_details\_df

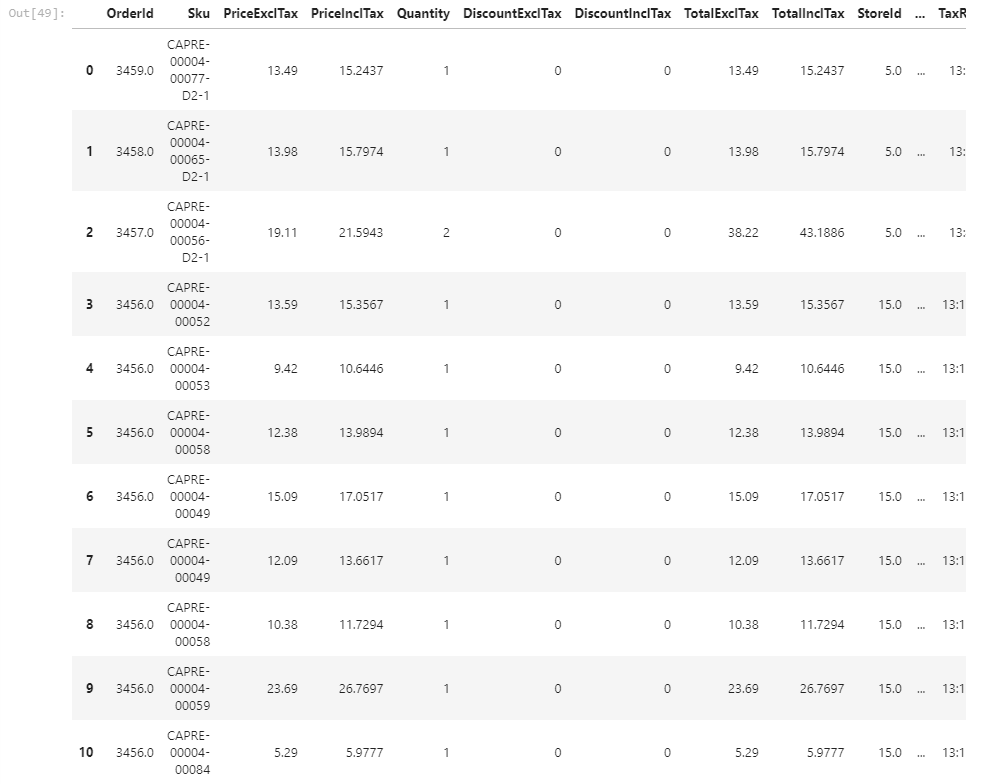
df1 **=** pd**.**DataFrame(data1)

df2 **=** pd**.**DataFrame(data2)

*# Merge DataFrames on the 'ID' column*

result\_df **=** pd**.**merge(df2, df1, on**=**'OrderId', how**=**'left')

result\_df**.**head(20)

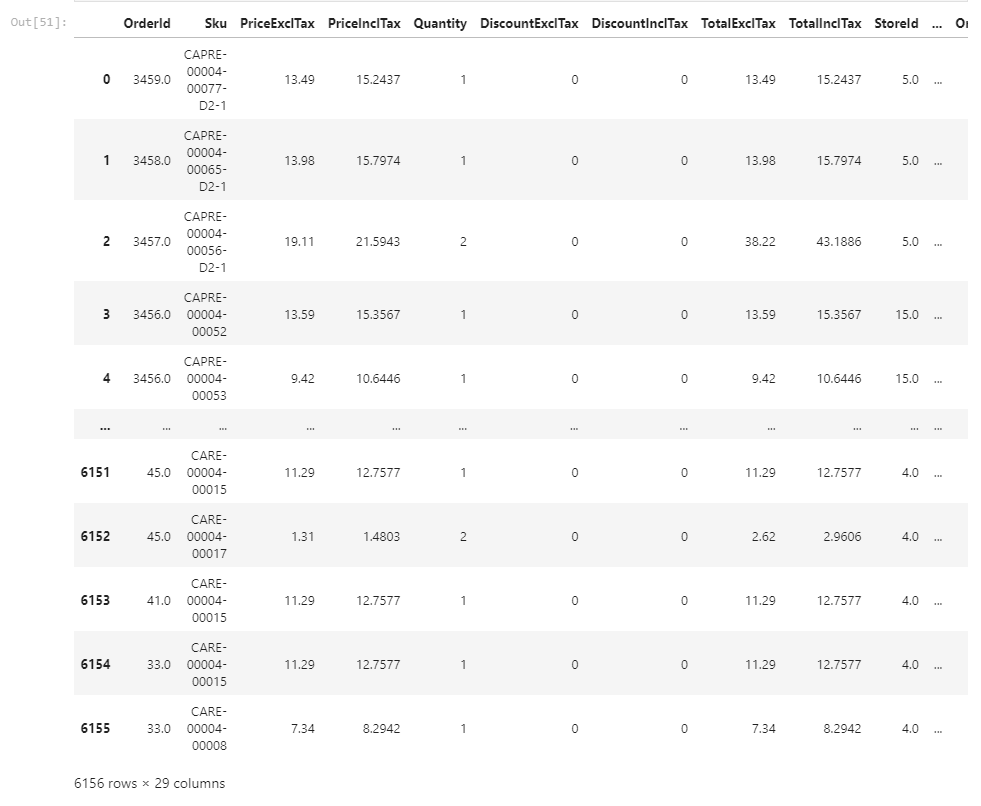


FEATURE ENGINEERING

#Creating a Feature that calculates the price per quantity, indicating the average price per unit.

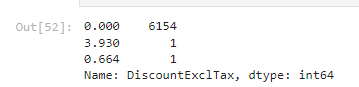
result\_df['PricePerQuantity'] = result\_df['PriceExclTax'] / result\_df['Quantity']

result\_df#.head(5)



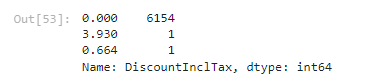
result\_df["DiscountExclTax"]**.**unique()

result\_df["DiscountExclTax"]**.**value\_counts()



result\_df["DiscountInclTax"]**.**unique()

result\_df["DiscountInclTax"]**.**value\_counts()



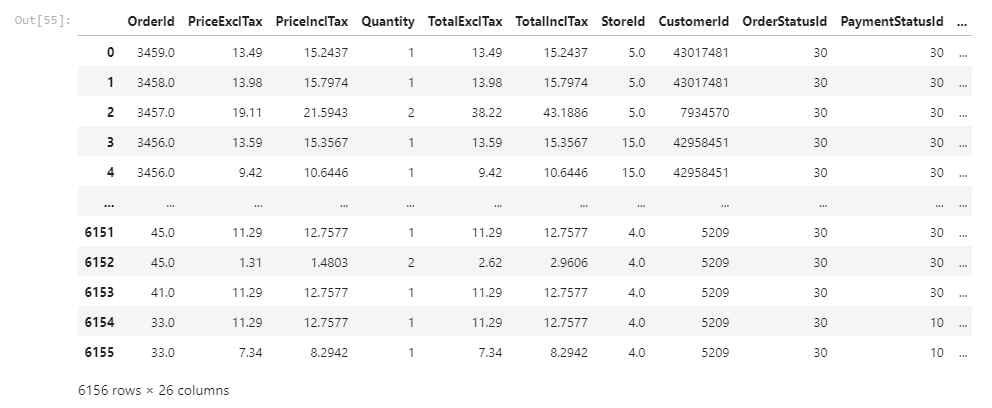
*#As the most are 0.00 we are dropping the two features.*

*#copying the result\_df as df1*

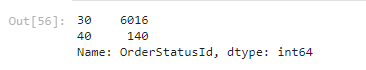
df1**=**result\_df**.**copy()

result\_df**.**drop(["DiscountInclTax","DiscountExclTax","Sku"],axis**=**1,inplace**=True**)

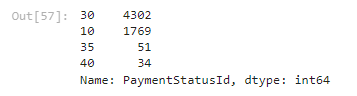
result\_df



result\_df['OrderStatusId']**.**value\_counts()



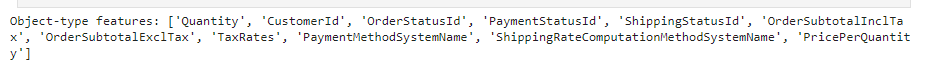
result\_df['PaymentStatusId']**.**value\_counts()



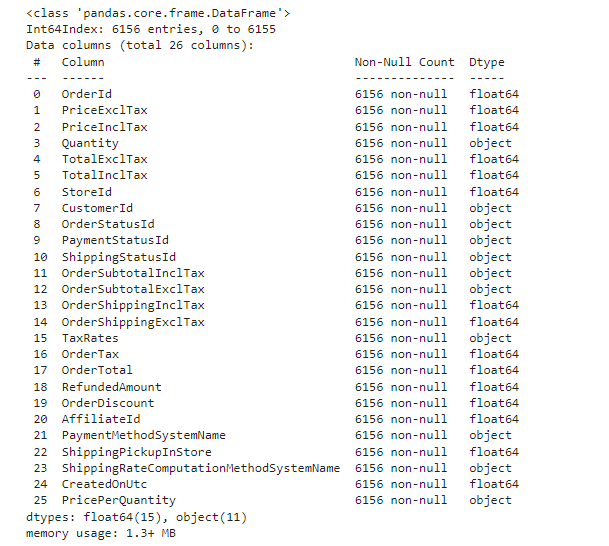
object\_features **=** result\_df**.**select\_dtypes(include**=**'object')**.**columns**.**tolist()

*# Print the result*

print("Object-type features:", object\_features)



result\_df**.**info()



**for** j **in** object\_features:

result\_df[j] **=** result\_df[j]**.**astype(str)

*# Perform string operation*

result\_df[j] **=** result\_df[j]**.**str**.**replace('.', '')

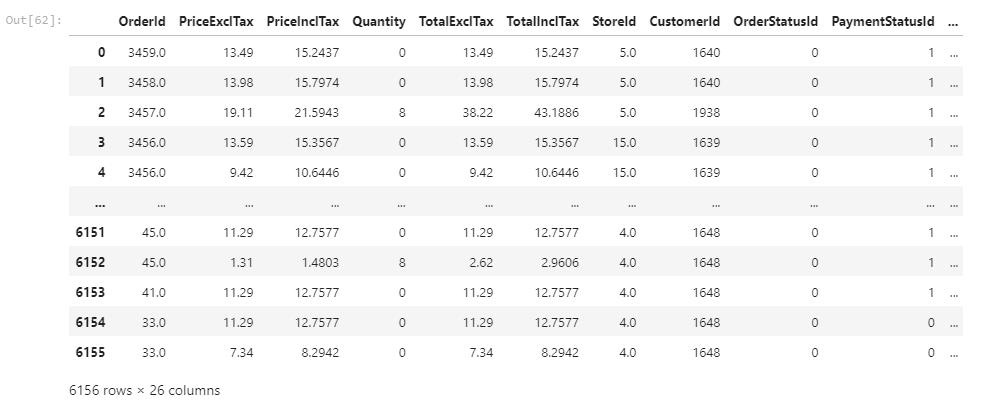
**from** sklearn.preprocessing **import** LabelEncoder

**for** i **in** object\_features:

le **=** LabelEncoder()

result\_df[i] **=** le**.**fit\_transform(result\_df[i])

**result\_df**

****

*#splitting the datset into training and testing*

from sklearn.model\_selection import train\_test\_split

X = result\_df.drop('PaymentMethodSystemName', axis=1)

y = result\_df['PaymentMethodSystemName']

*# Split the data into training and testing sets*

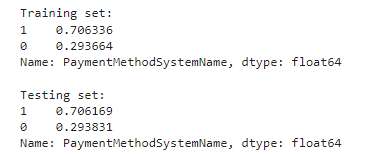
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42, stratify=y)

print("Training set:")

print(y\_train**.**value\_counts(normalize**=True**))

print("\nTesting set:")

print(y\_test**.**value\_counts(normalize**=True**))



**LOGISTIC REGRESSION**

**from** sklearn.linear\_model **import** LogisticRegression

**from** sklearn.metrics **import** confusion\_matrix, classification\_report, accuracy\_score, roc\_curve, roc\_auc\_score, precision\_recall\_curve

# Apply logistic regression

model = LogisticRegression(random\_state=42)

model.fit(X\_train, y\_train)

*# Make predictions on the test set*

y\_pred **=** model**.**predict(X\_test)

y\_prob **=** model**.**predict\_proba(X\_test)[:, 1]

*# Confusion matrix*

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

print(conf\_matrix)

from sklearn.metrics import confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns *# Optional, for a nicer visualization*

*# Assuming "model" is your trained classifier, and X\_test, y\_test are your test data*

y\_pred = model.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

*# Visualization*

plt.figure(figsize=(8, 6))

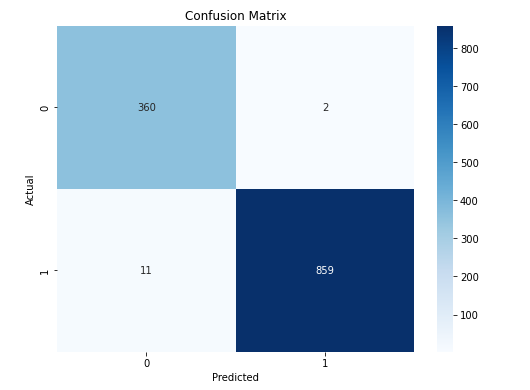
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=model.classes\_, yticklabels=model.classes\_)

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

****

*# Classification report*

class\_report = classification\_report(y\_test, y\_pred)

# Accuracy

accuracy = accuracy\_score(y\_test, y\_pred)

# ROC curve and AUC

fpr, tpr, \_ = roc\_curve(y\_test, y\_prob)

roc\_auc = auc(fpr, tpr)

# Precision-Recall curve and AUC

precision, recall, \_ = precision\_recall\_curve(y\_test, y\_prob)

pr\_auc = auc(recall, precision)

# Visualize ROC curve

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2f})'.format(roc\_auc))

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

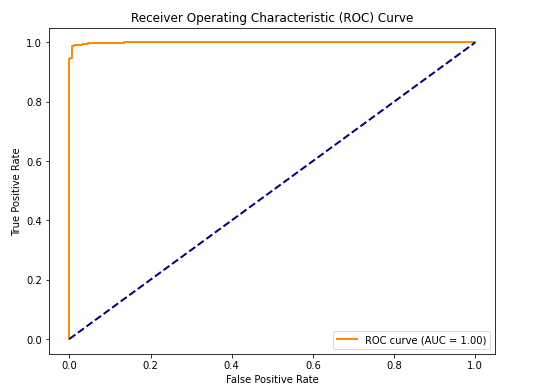
plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc='lower right')

plt.show()



*# Visualize Precision-Recall curve*

plt**.**figure(figsize**=**(8, 6))

plt**.**plot(recall, precision, color**=**'darkorange', lw**=**2, label**=**'PR curve (AUC = {:.2f})'**.**format(pr\_auc))

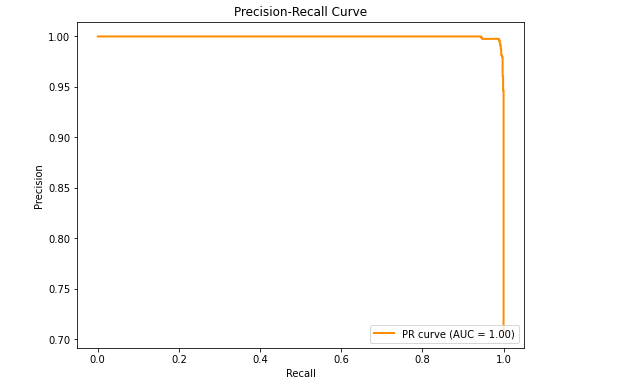
plt**.**xlabel('Recall')

plt**.**ylabel('Precision')

plt**.**title('Precision-Recall Curve')

plt**.**legend(loc**=**'lower right')

plt**.**show()



*# Display evaluation metrics*

print(f"Accuracy: {accuracy:.4f}")

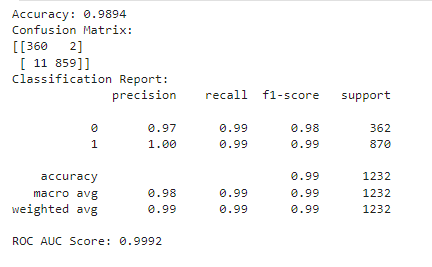
print("Confusion Matrix:")

print(conf\_matrix)

print("Classification Report:")

print(class\_report)

print(f"ROC AUC Score: {roc\_auc:.4f}")



**DECISION TREE CLASSIFICATION**

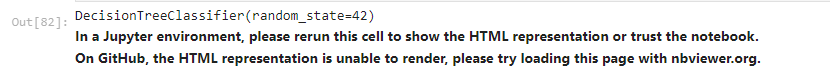
**from** sklearn.tree **import** DecisionTreeClassifier

**from** sklearn.metrics **import** accuracy\_score, confusion\_matrix, classification\_report

# Initialize and fit the Decision Tree Classifier

dt\_classifier = DecisionTreeClassifier(random\_state=42)

dt\_classifier.fit(X\_train, y\_train)



*# Make predictions on the test set*

y\_pred **=** dt\_classifier**.**predict(X\_test)

*# Evaluate the model*

accuracy **=** accuracy\_score(y\_test, y\_pred)

conf\_matrix **=** confusion\_matrix(y\_test, y\_pred)

classification\_rep **=** classification\_report(y\_test, y\_pred)

# Display evaluation metrics

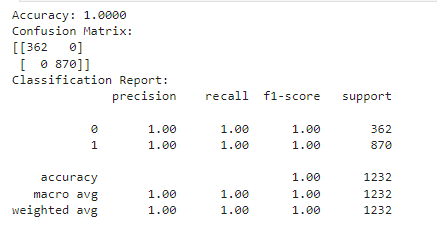
print(f"Accuracy: {accuracy:.4f}")

print("Confusion Matrix:")

print(conf\_matrix)

print("Classification Report:")

print(classification\_rep)



**from** sklearn.metrics **import** confusion\_matrix

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns *# Optional, for a nicer visualization*

*# Assuming "model" is your trained classifier, and X\_test, y\_test are your test data*

y\_pred **=** dt\_classifier**.**predict(X\_test)

cm **=** confusion\_matrix(y\_test, y\_pred)

*# Visualization*

plt**.**figure(figsize**=**(8, 6))

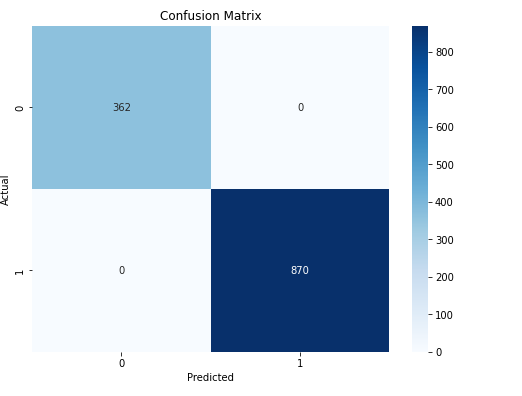
sns**.**heatmap(cm, annot**=True**, fmt**=**"d", cmap**=**"Blues", xticklabels**=**model**.**classes\_, yticklabels**=**model**.**classes\_)

plt**.**title('Confusion Matrix')

plt**.**xlabel('Predicted')

plt**.**ylabel('Actual')

plt**.**show()

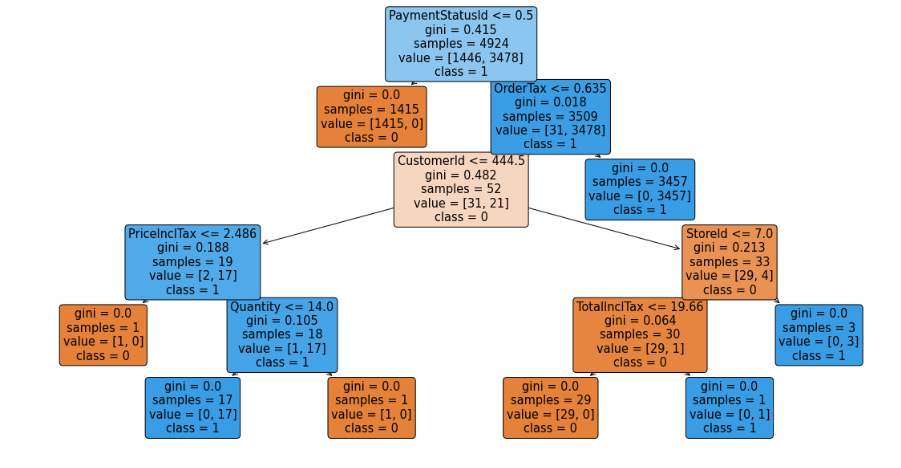


**from** sklearn.tree **import** plot\_tree

plt**.**figure(figsize**=**(20, 10))

plot\_tree(dt\_classifier, feature\_names**=**X**.**columns, class\_names**=**[str(i) **for** i **in** dt\_classifier**.**classes\_], filled**=True**, rounded**=True**)

plt**.**show()



**METHOD-2**

x1 **=** result\_df**.**drop('ShippingRateComputationMethodSystemName', axis**=**1)

Y1 **=** result\_df['ShippingRateComputationMethodSystemName']

*# Split the data into training and testing sets*

x1\_train, x1\_test, Y1\_train, Y1\_test **=** train\_test\_split(x1, Y1, test\_size**=**0.2, random\_state**=**42)

# Apply logistic regression

model = LogisticRegression(random\_state=42)

model.fit(x1\_train, Y1\_train)

*# Make predictions on the test set*

Y1\_pred **=** model**.**predict(x1\_test)

*# Evaluate the model*

accuracy **=** accuracy\_score(Y1\_test, Y1\_pred)

conf\_matrix **=** confusion\_matrix(Y1\_test, Y1\_pred)

classification\_rep **=** classification\_report(Y1\_test, Y1\_pred)

roc\_auc **=** roc\_auc\_score(Y1\_test, model**.**predict\_proba(x1\_test)[:, 1])

# Display evaluation metrics

print(f"Accuracy: {accuracy:.4f}")

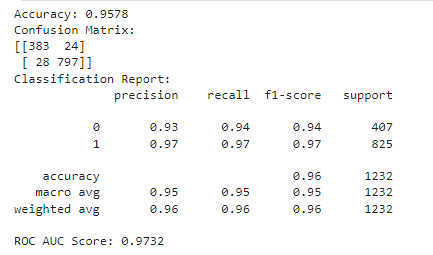
print("Confusion Matrix:")

print(conf\_matrix)

print("Classification Report:")

print(classification\_rep)

print(f"ROC AUC Score: {roc\_auc:.4f}")



*# Plot the ROC curve*

fpr, tpr, thresholds **=** roc\_curve(Y1\_test, model**.**predict\_proba(x1\_test)[:, 1])

plt**.**figure(figsize**=**(8, 6))

plt**.**plot(fpr, tpr, label**=**'ROC Curve')

plt**.**plot([0, 1], [0, 1], linestyle**=**'--', color**=**'gray', label**=**'Random')

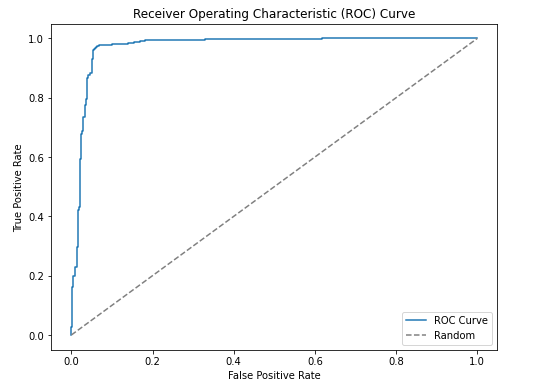
plt**.**xlabel('False Positive Rate')

plt**.**ylabel('True Positive Rate')

plt**.**title('Receiver Operating Characteristic (ROC) Curve')

plt**.**legend()

plt**.**show()



*#Decision Tree Classifier*

dt\_classifier1 **=** DecisionTreeClassifier(random\_state**=**42)

dt\_classifier1**.**fit(x1\_train, Y1\_train)

# Make predictions on the test set

Y\_pred = dt\_classifier1.predict(x1\_test)

# Evaluate the model

accuracy = accuracy\_score(Y1\_test, Y1\_pred)

conf\_matrix = confusion\_matrix(Y1\_test, Y1\_pred)

classification\_rep = classification\_report(Y1\_test, Y1\_pred)

# Display evaluation metrics

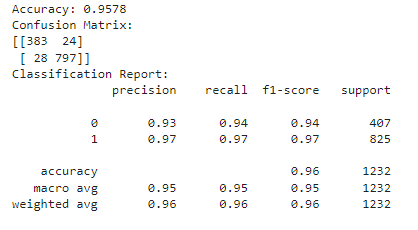
print(f"Accuracy: {accuracy:.4f}")

print("Confusion Matrix:")

print(conf\_matrix)

print("Classification Report:")

print(classification\_rep)



**from** sklearn.metrics **import** confusion\_matrix

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns *# Optional, for a nicer visualization*

*# Assuming "model" is your trained classifier, and X\_test, y\_test are your test data*

y\_pred **=** dt\_classifier1**.**predict(x1\_test)

cm **=** confusion\_matrix(Y1\_test, Y1\_pred)

*# Visualization*

plt**.**figure(figsize**=**(8, 6))

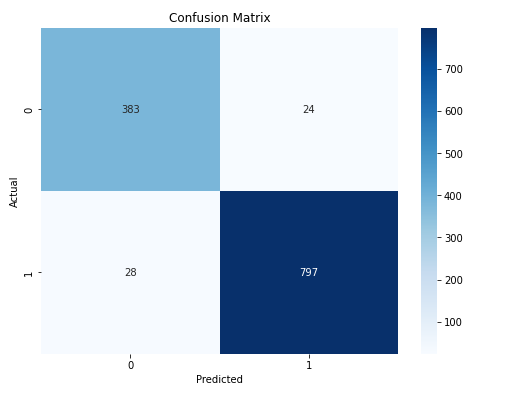
sns**.**heatmap(cm, annot**=True**, fmt**=**"d", cmap**=**"Blues", xticklabels**=**model**.**classes\_, yticklabels**=**model**.**classes\_)

plt**.**title('Confusion Matrix')

plt**.**xlabel('Predicted')

plt**.**ylabel('Actual')

plt**.**show()

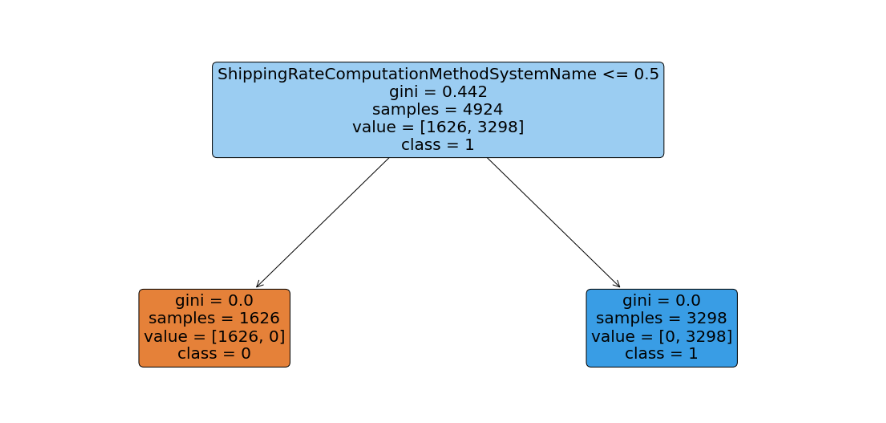


from sklearn.tree import plot\_tree

plt.figure(figsize=(20, 10))

plot\_tree(dt\_classifier1, feature\_names=X.columns, class\_names=[str(i) for i in dt\_classifier.classes\_], filled=True, rounded=True)

plt.show()



**CHAPTER - 4**

**CONCLUSION**

Finally, the Orders in Ontario Project has completed a transformative journey that began with data pretreatment and ended with model creation, analytics, visualization, and deployment. The thorough technique used, which included data cleaning, novel feature engineering, and the application of Logistic Regression and Decision Tree models, paved the way for informed decision-making in the field of e-commerce order processing. The post-modeling phases, which included analytics and visualization, not only revealed intricate patterns in the information but also provided user-friendly insights via interactive dashboards. The deployment of models on GitHub demonstrates a commitment to openness and cooperation. As the project moves forward, the methodology's iterative structure, combined with a user-centric approach, prepares it for continual adaptation and improvement.

**CHAPTER - 5**

**REFERENCES**

* <https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/>
* <https://www.sciencedirect.com/topics/computer-science/logistic-regression#:~:text=Logistic%20regression%20is%20a%20process,%2Fno%2C%20and%20so%20on>.
* <https://towardsdatascience.com/exploratory-data-analysis-8fc1cb20fd15>
* <https://towardsdatascience.com/what-is-feature-engineering-importance-tools-and-techniques-for-machine-learning-2080b0269f10>