AML 2203 - Advanced Python AI and ML Tools

DataCo Supply Chain Sales Prediction

Presented by:
Neil Patrick Discaya (C0856896)
Rubylyn Padillo (C0846590)
Vengie Dinampo (C0849806)

Presented to: **Prof. Mezbah Uddin**

December 10, 2022

Agenda

A **Introduction**

B **Data Preprocessing/Cleaning**

C **Visualizations**

D **Machine Learning Models**

E **Python Codes/Output**

Sales Prediction Presentation Overview

INTRODUCTION
This part include Problem Statement & Dataset and the Variables and their types

PREPROCESSING
All data cleaning and data exploratory parts are included in this section.

VISUALIZATIONS
All data visualizations will be discussed here.

MODELLING
Machine Learning preparations and implementation takes place here. With the results and conclusion.

CODES/OUTPUT
Python Codes with Sample Outputs.

A. Project Overview

1) Problem Statement and Dataset

- Our team had considered the dataset ***DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS***.
- The dataset mostly comprises details on DataCo's operating supply chain. The collection includes some crucial data, including details about **buyers (sellers), orders, shipments, and products.**
- It also includes information from commercial activities including **sales, order processing, and order package delivery.**
- The purpose of our team in using this dataset is make predictions on the company’s **sales for their products that include clothing, sports equipment, an d electronic supplies.**

A. Project Overview

The DataCo Supply Chain dataset contains **180519** rows and **53** columns shown in the screenshots to see dataset shape and information.

2) DATASET VARIABLES & THEIR TYPES

```
df.columns
```

```
Index(['Type', 'Days for shipping (real)', 'Days for shipment (scheduled)', 'Benefit per order', 'Sales per customer', 'Deliver
y Status', 'Late_delivery_risk', 'Category Id', 'Category Name', 'Customer City', 'Customer Country', 'Customer Email', 'Custom
er Fname', 'Customer Id', 'Customer Lname', 'Customer Password', 'Customer Segment', 'Customer State', 'Customer Street', 'Cust
omer Zipcode', 'Department Id', 'Department Name', 'Latitude', 'Longitude', 'Market', 'Order City', 'Order Country', 'Order Cus
tomer Id', 'order date (DateOrders)', 'Order Id', 'Order Item Cardprod Id', 'Order Item Discount', 'Order Item Discount Rate',
'Order Item Id', 'Order Item Product Price', 'Order Item Profit Ratio', 'Order Item Quantity', 'Sales', 'Order Item Total', 'Or
der Profit Per Order', 'Order Region', 'Order State', 'Order Status', 'Order Zipcode', 'Product Card Id', 'Product Category I
d', 'Product Description', 'Product Image', 'Product Name', 'Product Price', 'Product Status',
       'shipping date (DateOrders)', 'Shipping Mode'],
      dtype='object')
```

The target variable for the sales prediction for the DataCo Supply Chain data set is the company sales and the independent variables can be seen from the columns shown in the screenshot below.

```
df.select_dtypes(include = "object").columns
```

```
Index(['Type', 'Delivery Status', 'Category Name', 'Customer City', 'Customer Country', 'Customer Email', 'Customer Fname', 'Cu
stomer Lname', 'Customer Password', 'Customer Segment', 'Customer State', 'Customer Street', 'Department Name', 'Market', 'Orde
r City', 'Order Country', 'order date (DateOrders)', 'Order Region', 'Order State', 'Order Status', 'Product Image', 'Product N
ame', 'shipping date (DateOrders)', 'Shipping Mode'], dtype='object')
```

```
df.select_dtypes(include=["int64","float64"]).columns
```

```
Index(['Days for shipping (real)', 'Days for shipment (scheduled)', 'Benefit per order', 'Sales per customer', 'Late_delivery_r
isk', 'Category Id', 'Customer Id', 'Customer Zipcode', 'Department Id', 'Latitude', 'Longitude', 'Order Customer Id', 'Order I
d', 'Order Item Cardprod Id', 'Order Item Discount', 'Order Item Discount Rate', 'Order Item Id', 'Order Item Product Price',
'Order Item Profit Ratio', 'Order Item Quantity', 'Sales', 'Order Item Total', 'Order Profit Per Order', 'Order Zipcode', 'Prod
uct Card Id', 'Product Category Id', 'Product Description', 'Product Price', 'Product Status'], dtype='object')
```

B. Data Preprocessing

Contents

This section of the Project includes preparation of the dataset. It pertains to data preprocessing, cleaning and feature engineering.

DUPLICATE VALUES

NULL VALUES

IRRELEVANT FEATURES

LABEL ENCODER

FEATURE ENGINEERING

B. Data Preprocessing

Summary of the Dataset details shown below.

1) Duplicate Values

```
print('Dataset Details:')
print(f"Dataset has {df.shape[0]} rows and {df.shape[1]} columns")
print(f"Duplicates: {df.duplicated().sum()}")
print(f"Total Missing Values: {df.isna().sum().sum()}")
print(f"Number of rows with missing values: {df.isna().any(axis=1).sum()}")
```

```
Dataset Details:
Dataset has 180519 rows and 53 columns
Duplicates: 0
Total Missing Values: 336209
Number of rows with missing values: 180519
```

There are more Null values compare to 0 duplicate value. Out of 180519 rows, there are also 180519 missing values, which denote that there is one column that does not have any value at all. Overall, there are total of 336209 values are missing.

2) Null Values

Customer Id	0
Customer Lname	8
Customer Password	0
Customer Segment	0
Customer State	0
Customer Street	0
Customer Zipcode	3
Department Id	0
Department Name	0
Latitude	0

Order State	0
Order Status	0
Order Zipcode	155679
Product Card Id	0
Product Category Id	0
Product Description	180519
Product Image	0
Product Name	0
Product Price	0

B. Data Preprocessing

3) Irrelevant Features

No need to fill or individually replace the null values because these columns with null values won't affect the Sales prediction.

```
features = ['Type', 'Benefit per order', 'Sales per customer',  'Delivery Status', 'Late_delivery_risk', 'Category Name', 'Custom
```

```
# Showing the features to be included in the predication
data = df[features]
data.head()
```

Instead, only the relevant features will be considered.

	Type	Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	Category Name	Customer City	Customer Country	Customer Id	Customer Segment	Customer State	Customer Zipcode	Department Name	
0	DEBIT	91.250000	314.640015	Advance shipping	0	Sporting Goods	Caguas	Puerto Rico	20755	Consumer	PR	725.0	Fitness	1

4) Sales Prediction Features

The target for the Prediction is the company's Sales and the features were also added below.

```
# Dataframe for the Sales Prediction
data_sales =df[['Type', 'Benefit per order', 'Sales per customer',  'Delivery Status', 'Late_delivery_risk', 'Category Name', 'Cu
```

```
# Removing irrelevant features
features=data_sales.drop(columns=['Sales', 'Order Item Quantity', 'Order Item Product Price'])
target=data_sales['Sales']
```

B. Data Preprocessing

```
#Label Encoder will transform/change Categorical data to Numerical
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
def Labelencoder_feature(x):
    le=LabelEncoder()
    x=le.fit_transform(x)
    return x
features=features.apply(Labelencoder_feature)
```

5) Label Encoder

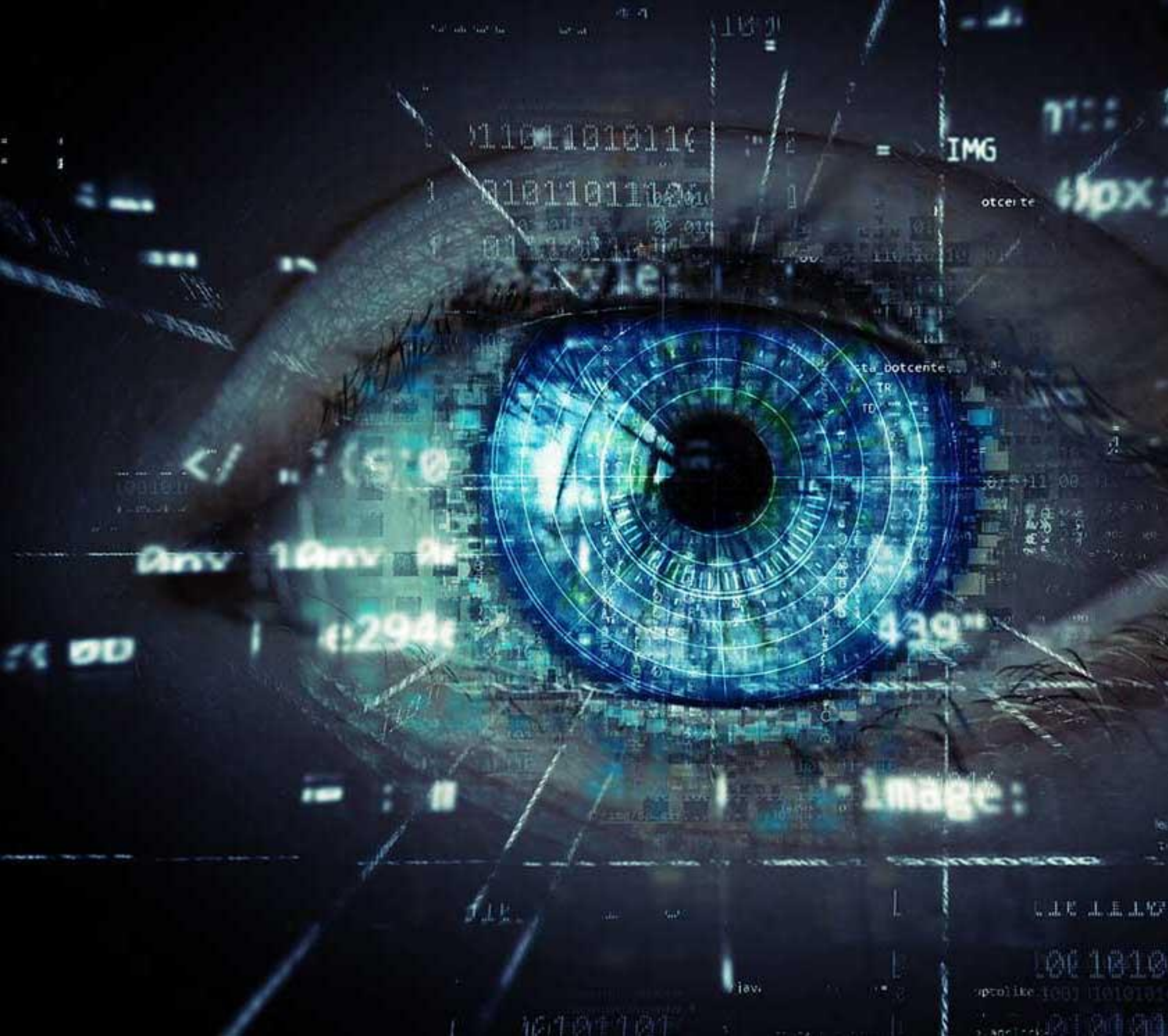
This will change all the Categorical data in the features to numerical data.

6) Feature Engineering using f_regression

```
#Using f_regression the relevance of the features can be checked
from sklearn.feature_selection import f_regression
F_values, p_values  = f_regression(features, target)
```

```
import itertools
f_reg = [(i, v, z) for i, v, z in itertools.zip_longest(features.columns, F_values,  ['%.3f' % p for p in p_values])]
f_reg=pd.DataFrame(f_reg, columns=['Variable','F_Value', 'P_Value'])
```

C. EDA / VISUALIZATIONS



1) Order Country Visualization
2) Product Visualization
3) Product and Delivery Status Visualization
4) Product and Delivery Status Visualization
5) Product Category Name Visualization
6) Type of Payment Visualization
7) Type of Payment for Orders Visualization
8) Annual Orders Visualization
9) Quarterly Orders Visualization
10) Monthly Orders Visualization
11) Heatmap Correlation

B. Data Preprocessing

6) Feature Engineering using f_regression

```
f_reg=pd.DataFrame(f_reg, columns=['Variable','F_Value', 'P_Value'])
f_reg = f_reg.sort_values(by=['P_Value'])
f_reg.P_Value= f_reg.P_Value.astype(float)
f_reg=f_reg[f_reg.P_Value<0.05]
f_reg
```

	Variable	F_Value	P_Value
20	Order Id	1165.171704	0.000
22	Order Item Discount	57166.125441	0.000
21	Order Item Cardprod Id	12782.968321	0.000
39	shipping date (DateOrders)	142.652140	0.000
19	order date (DateOrders)	128.461963	0.000
18	Order Customer Id	673.464036	0.000
27	Order Profit Per Order	13782.670150	0.000
15	Market	240.910781	0.000
28	Order Region	140.517795	0.000
29	Order State	27.929090	0.000
26	Order Item Total	481682.347274	0.000
12	Department Name	524.094617	0.000
32	Product Card Id	12782.968321	0.000
8	Customer Id	673.464036	0.000

8	Customer Id	673.464036	0.000
33	Product Category Id	10095.780861	0.000
35	Product Image	37751.723070	0.000
5	Category Name	26066.331991	0.000
36	Product Name	37751.723070	0.000
37	Product Price	116680.120560	0.000
2	Sales per customer	481682.347274	0.000
1	Benefit per order	13782.670150	0.000
31	Order Zipcode	60.375900	0.000
24	Order Item Id	1133.743612	0.000
16	Order City	8.763986	0.003
9	Customer Segment	4.266892	0.039

D. Modeling and Evaluations

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(final_features, target, test_size = 0.3, random_state = 42)
```

1) Random Forest Regressor

```
rf = RandomForestRegressor(n_estimators = 1000, random_state = 42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
```

```
rf_pred = pd.DataFrame({'actual' : y_test,
                        'predicted' : rf.predict(X_test)})
rf_pred.head()
```

	actual	predicted
80120	199.990005	199.990005
19670	250.000000	250.000000
114887	249.899994	249.899994
120110	299.980011	299.980011
56658	119.970001	119.970001

a) Random Forest Regressor Performance Results

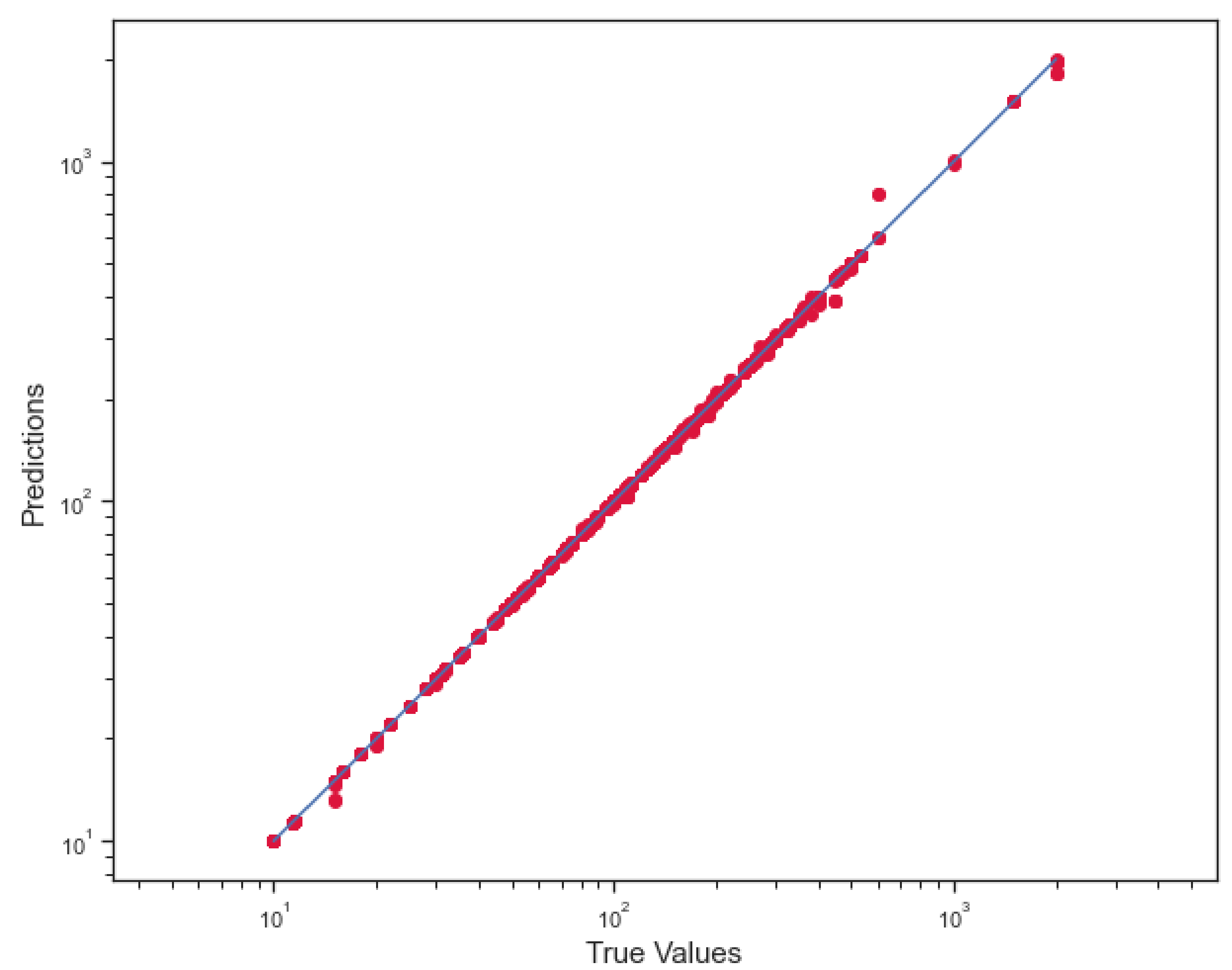
```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred_rf, multioutput='uniform_average')
```

```
0.9997962219902725
```

D. Modeling and Evaluations

b) Random Forest Regressor Results Visualization

```
#Visualization of the Results
plt.figure(figsize=(10,8))
plt.scatter(y_test, y_pred_rf, c='crimson')
plt.yscale('log')
plt.xscale('log')
p1 = max(max(y_pred_rf), max(y_test))
p2 = min(min(y_pred_rf), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('True Values', fontsize=15)
plt.ylabel('Predictions', fontsize=15)
plt.axis('equal')
plt.show()
```



D. Modeling and Evaluations

2) XGBRegressor

```python
xgb_reg = XGBRegressor(learning_rate=1.0, max_depth=6, min_child_weight=40, seed=0)

xgb_reg.fit(X_train, y_train)
y_pred_xgb = xgb_reg.predict(X_test)
```

a) XGBRegressor Performance Results

```python
r2_score(y_test, y_pred_xgb, multioutput='uniform_average')
```

0.9990990149904566

```python
xgb_pred = pd.DataFrame({'actual' : y_test,
                         'predicted' : xgb_reg.predict(X_test)})
xgb_pred.head()
```

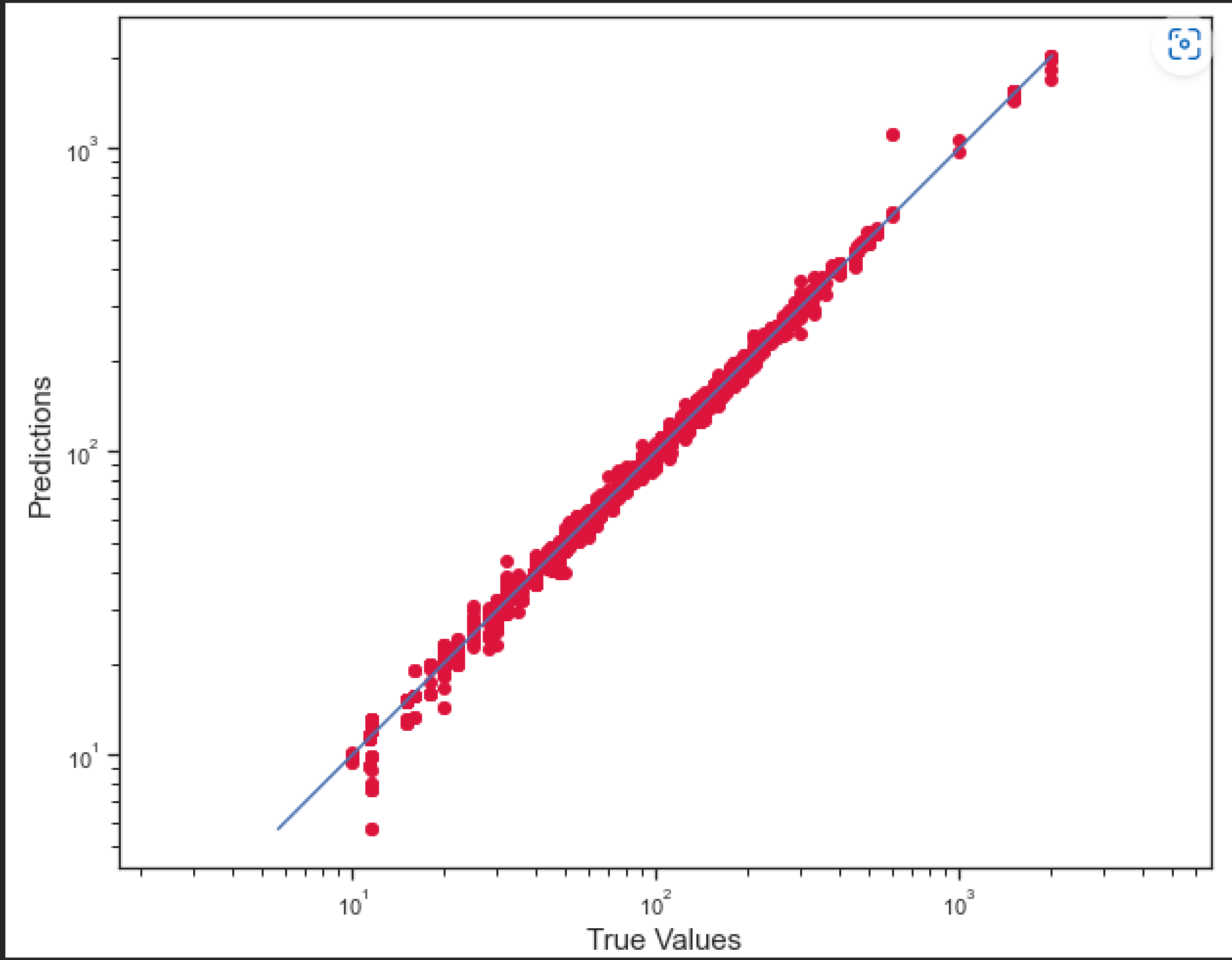
	actual	predicted
80120	199.990005	200.058777
19670	250.000000	249.714874
114887	249.899994	249.781052
120110	299.980011	299.991943
56658	119.970001	119.921814

D. Modeling and Evaluations

b) XGBRegressor Results Visualization

```
#Visualization of the Results
plt.figure(figsize=(10,8))
plt.scatter(y_test, y_pred_xgb, c='crimson')
plt.yscale('log')
plt.xscale('log')

p1 = max(max(y_pred_xgb), max(y_test))
p2 = min(min(y_pred_xgb), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('True Values', fontsize=15)
plt.ylabel('Predictions', fontsize=15)
plt.axis('equal')
plt.show()
```



E. PYTHON CODES/OUTPUT

(will be shown in Jupyter)

THANK YOU

HAVE A WONDERFUL HOLIDAY!