Market Basket Analysis with Python & AWS

1. Launch an EC2 Instance Running Windows

Sign in to AWS Management Console:

- Go to AWS Management Console.

Launch an EC2 Instance:

- Navigate to the EC2 dashboard.
- Click on "Launch Instance."

Choose an Amazon Machine Image (AMI):

- Select a Windows Server AMI, such as “Microsoft Windows Server 2019 Base.”

Choose an Instance Type:

- Select an instance type suitable for your needs, such as `t2.micro` (eligible for the free tier).

Configure Instance Details:

- Adjust settings as needed, but the default settings are often fine for initial setup.

Add Storage:

- You can keep the default storage or adjust based on your requirements.

Add Tags:

- Optionally, add tags to help identify your instance.

Configure Security Group:

- Add rules to allow RDP (Remote Desktop Protocol) access on port 3389.

Review and Launch:

- Review your configuration and click “Launch.”
- Select or create a new key pair for SSH access and download the key file.

Connect to Your Instance:

- Use Remote Desktop to connect to your Windows instance. You'll need the instance's public DNS and the Administrator password, which you can obtain using the key pair

EC2 WINDOWS INSTANCE

2.Install Python and an IDE

Download Python:

- Go to the Python Downloads page.
- Download and install the latest version of Python for Windows.

Install Python:

- Run the installer.
- Ensure the option to "Add Python to PATH" is checked.
- Select "Install Now" to install Python.

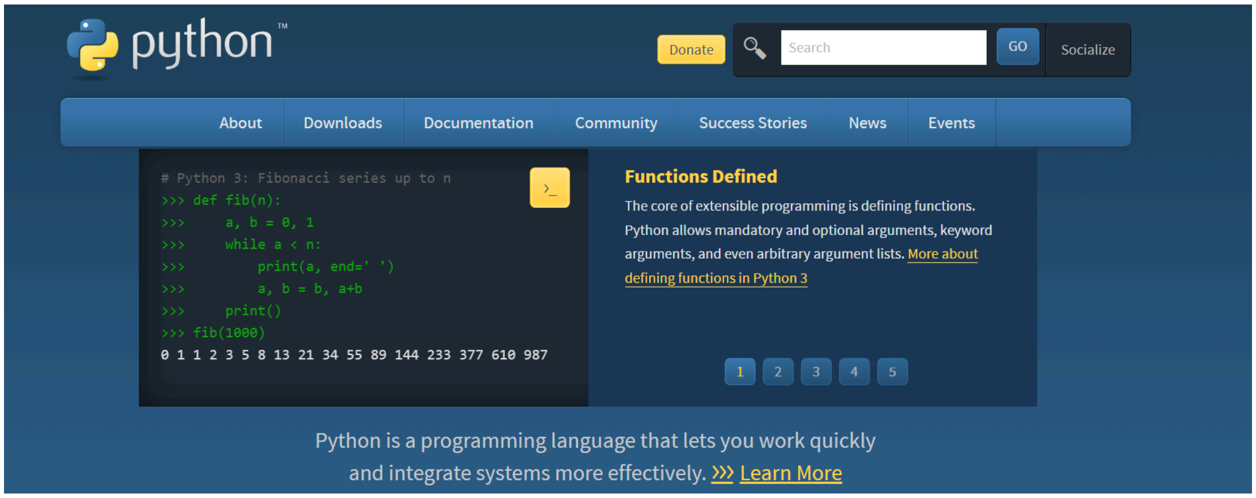
Verify Python Installation:

Open Command Prompt and type:

```
python --version
pip --version
```

Install an IDE:

- You can use an IDE like PyCharm or Visual Studio Code.
- **For Visual Studio Code:**
  - Download from Visual Studio Code website.
  - Install and open Visual Studio Code.
- **For PyCharm:**
  - Download from PyCharm website.
  - Install and open PyCharm



```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

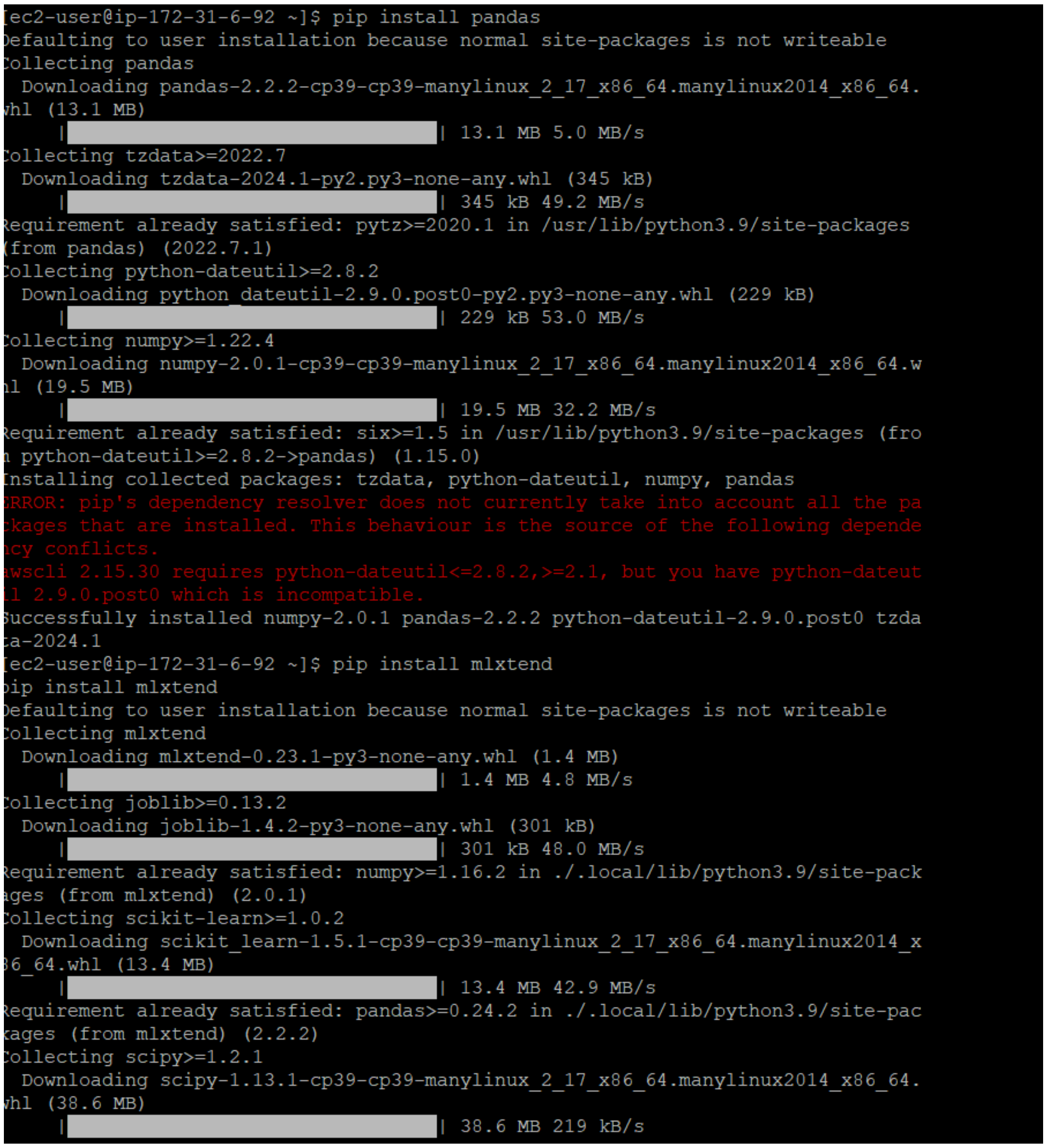
The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. More about defining functions in Python 3

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

3. Install Required Libraries

Open Command Prompt or the integrated terminal in your IDE and run the following commands to install the required libraries:

```
pip install pandas numpy scikit-learn mlxtend matplotlib seaborn
```



```
[ec2-user@ip-172-31-6-92 ~]$ pip install pandas
Defaulting to user installation because normal site-packages is not writeable
Collecting pandas
  Downloading pandas-2.2.2-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.
whl (13.1 MB)
     |████████████████████████████████| 13.1 MB 5.0 MB/s
Collecting tzdata>=2022.7
  Downloading tzdata-2024.1-py2.py3-none-any.whl (345 kB)
     |████████████████████████████████| 345 kB 49.2 MB/s
Requirement already satisfied: pytz>=2020.1 in /usr/lib/python3.9/site-packages
(from pandas) (2022.7.1)
Collecting python-dateutil>=2.8.2
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
     |████████████████████████████████| 229 kB 53.0 MB/s
Collecting numpy>=1.22.4
  Downloading numpy-2.0.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.w
hl (19.5 MB)
     |████████████████████████████████| 19.5 MB 32.2 MB/s
Requirement already satisfied: six>=1.5 in /usr/lib/python3.9/site-packages (fro
m python-dateutil>=2.8.2->pandas) (1.15.0)
Installing collected packages: tzdata, python-dateutil, numpy, pandas
ERROR: pip's dependency resolver does not currently take into account all the pa
ckages that are installed. This behaviour is the source of the following depende
ncy conflicts.
awscli 2.15.30 requires python-dateutil<=2.8.2,>=2.1, but you have python-dateut
il 2.9.0.post0 which is incompatible.
Successfully installed numpy-2.0.1 pandas-2.2.2 python-dateutil-2.9.0.post0 tzda
ta-2024.1
[ec2-user@ip-172-31-6-92 ~]$ pip install mlxtend
pip install mlxtend
Defaulting to user installation because normal site-packages is not writeable
Collecting mlxtend
  Downloading mlxtend-0.23.1-py3-none-any.whl (1.4 MB)
     |████████████████████████████████| 1.4 MB 4.8 MB/s
Collecting joblib>=0.13.2
  Downloading joblib-1.4.2-py3-none-any.whl (301 kB)
     |████████████████████████████████| 301 kB 48.0 MB/s
Requirement already satisfied: numpy>=1.16.2 in ./.local/lib/python3.9/site-pack
ages (from mlxtend) (2.0.1)
Collecting scikit-learn>=1.0.2
  Downloading scikit_learn-1.5.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x
86_64.whl (13.4 MB)
     |████████████████████████████████| 13.4 MB 42.9 MB/s
Requirement already satisfied: pandas>=0.24.2 in ./.local/lib/python3.9/site-pac
kages (from mlxtend) (2.2.2)
Collecting scipy>=1.2.1
  Downloading scipy-1.13.1-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.
whl (38.6 MB)
     |████████████████████████████████| 38.6 MB 219 kB/s
```

4. Create and Run Your Python Code

Create a Python Script:

- Open your IDE and create a new Python file, e.g., analysis.py.

Add the Following Code:

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder

# Create the dataset
data = {'TransactionID': [1, 1, 1, 2, 2, 3, 3, 4, 4, 4],
        'Item': ['Ferrari', 'Lamborghini', 'Porsche', 'Ferrari', 'Rolls-Royce', 'Ferrari', 'Lamborghini',
'Porsche', 'Rolls-Royce', 'Bugatti']}
df = pd.DataFrame(data)

# One-hot encode the dataset
encoder = OneHotEncoder(sparse=False)
df_encoded = pd.DataFrame(encoder.fit_transform(df[['Item']]),
columns=encoder.get_feature_names_out())

# Add TransactionID back to the encoded DataFrame
df_encoded['TransactionID'] = df['TransactionID']

# Pivot the DataFrame to the format required by apriori
df_pivot = df_encoded.groupby('TransactionID').sum()

# Apply the Apriori algorithm
frequent_itemsets = apriori(df_pivot, min_support=0.1, use_colnames=True)

# Generate the association rules
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1)

# Print the results
print("Frequent Itemsets:")
print(frequent_itemsets)
print("\nAssociation Rules:")
print(rules)

# Plot results
```

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=rules, x='support', y='confidence', hue='lift')
plt.title('Association Rules')
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.show()
```

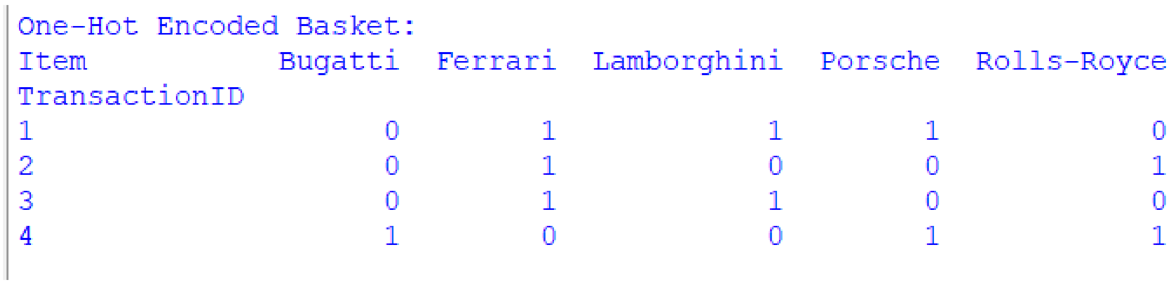
5. Dataset Used

```
data = {'TransactionID': [1, 1, 1, 2, 2, 3, 3, 4, 4, 4],
       'Item': ['Ferrari', 'Lamborghini', 'Porsche', 'Ferrari', 'Rolls-Royce', 'Ferrari', 'Lamborghini',
'Porsche', 'Rolls-Royce', 'Bugatti']}
```

6.Run the Script:

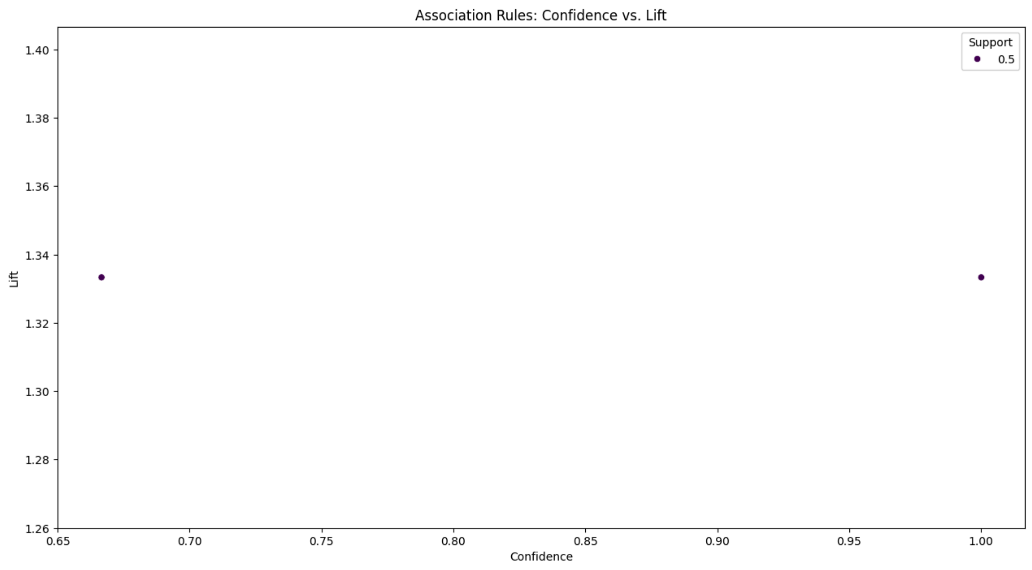
- In your IDE, use the run button or open Command Prompt and navigate to the directory where `analysis.py` is saved

One-hot encode the dataset

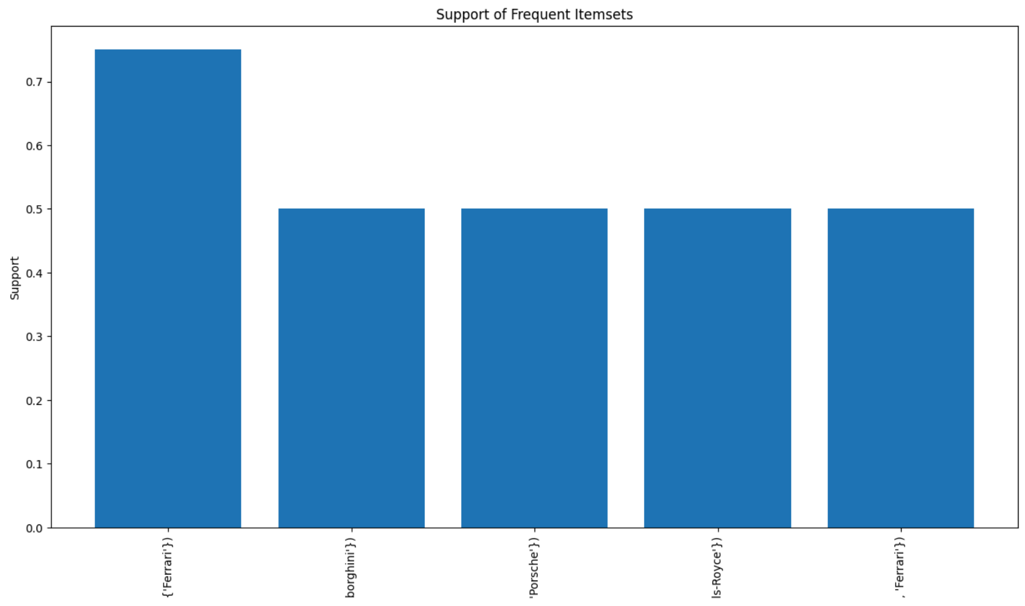


```
One-Hot Encoded Basket:
Item           Bugatti  Ferrari  Lamborghini  Porsche  Rolls-Royce
TransactionID
1                    0        1            1        1            0
2                    0        1            0        0            1
3                    0        1            1        0            0
4                    1        0            0        1            1
```

Confidence vs Lift



Visualization


Support of Frequent Itemsets
Support
{'Ferrari'})
borghini'})
'Porsche'})
ls-Royce'})
, 'Ferrari'})