Advanced Concepts in Data Analytics

Lab: Data Mining

This page was intentionally left blank.

Table of Contents

Data Mining

Introduction

In this lab, you'll perform data mining on an online retail dataset available from UCI's Machine Learning repository (https://archive.ics.uci.edu/ml/datasets/Online+Retail). This transactional dataset contains all transactions made between December 1, 2010 and December 9, 2011 for a UK-based online retailer (Chen et al., 2012). To understand the characteristics of the data and discover interesting patterns, you'll use pandas, seaborn, and mlxtend libraries for data manipulation, visualization and association rules mining.

You'll also learn good software engineering practices to manage data analytics projects, using a virtual environment for dependency management and Git for version control. We only cover. gitignore in this lab, but knowing how to work with Git is an essential skill for IT professionals. You're encouraged to save your work in private GitHub repos (which are free of charge).

Note: If you're not familiar with Git, see the Git Cheat Sheet website (https://github.github.com/training-kit/downloads/github-git-cheat-sheet/).

Equipment and Materials

- BYOD laptop
- Python
- Visual Studio Code
- CSV input file: rectangles.csv from Lab 1
- Excel input file: Online Retail.xlsx
- Text file: requirements.txt
- Git file: .gitignore

Lab Activity 1: Working with CSV files using pandas

In this section, you'll take the **rectangles.csv** file you used in Lab 1, Activity 3: Working with CSV files, and rewrite it using pandas, a powerful data analysis tool.

To refresh your memory, it reads a CSV file, generates simple stats (min/max/etc), prints the information in the console, and saves the information in a CSV file.

To rewrite your CSV file

1. Create a **lab2** folder and populate it with the provided **requirements.txt** and. gitignore files (Figure 1).

   Note: The. gitignore file specifies what not to commit in a Git repository.

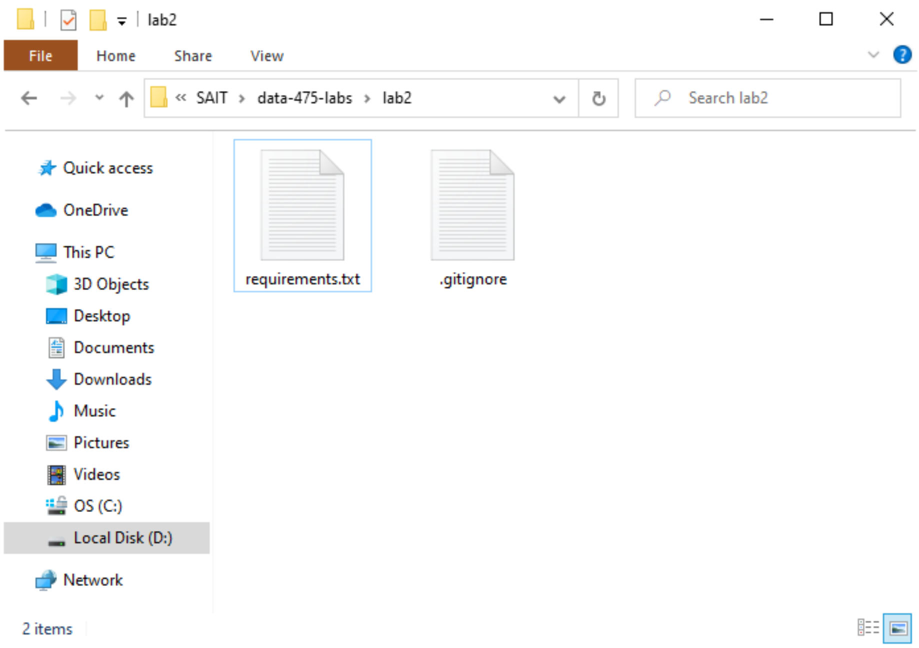


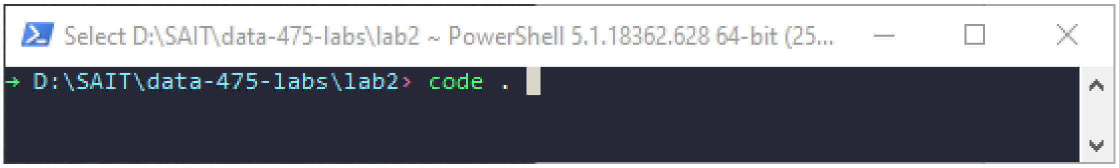
Figure 1: Lab 2 Folder Creation

Used with permission from Microsoft.

2. Open Visual Studio Code from a command line by typing the following in your lab2 folder:

   ```
   code.
   ```

   VS Code automatically sets the current folder as the working area, as shown below.



```
Select D:\SAIT\data-475-labs\lab2 ~ PowerShell 5.1.18362.628 64-bit (25...
→ D:\SAIT\data-475-labs\lab2› code .
```

Figure 2: Open VS Code from PowerShell

Used with permission from Microsoft.

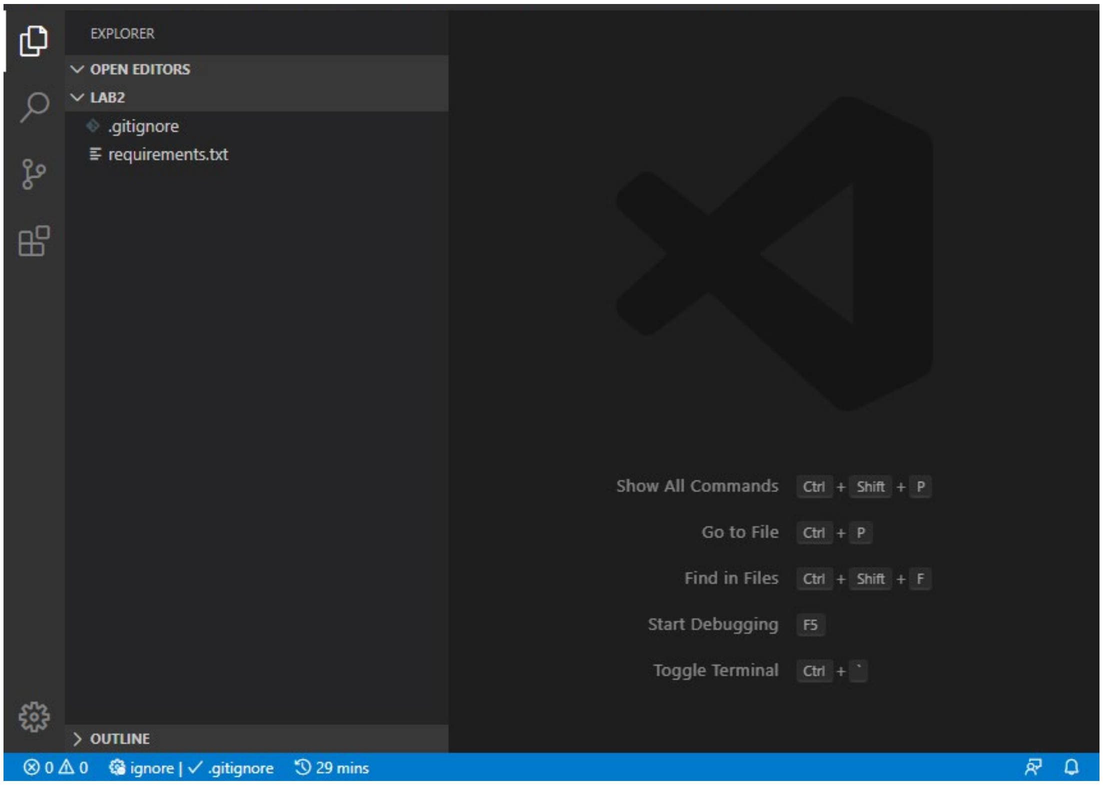


Figure 3: VS Code Opened from PowerShell

Used with permission from Microsoft.

3. Create a virtual environment for Lab 2.

   Tip: Isolating your project dependencies is a good software engineering practice for Python projects, because you're much less likely to affect your system-wide (global) Python environment and cause discrepancies across projects. For example, different projects may require the same Python library but require a specific version because of other dependencies within the project. In this case, installing a global Python environment won't work.

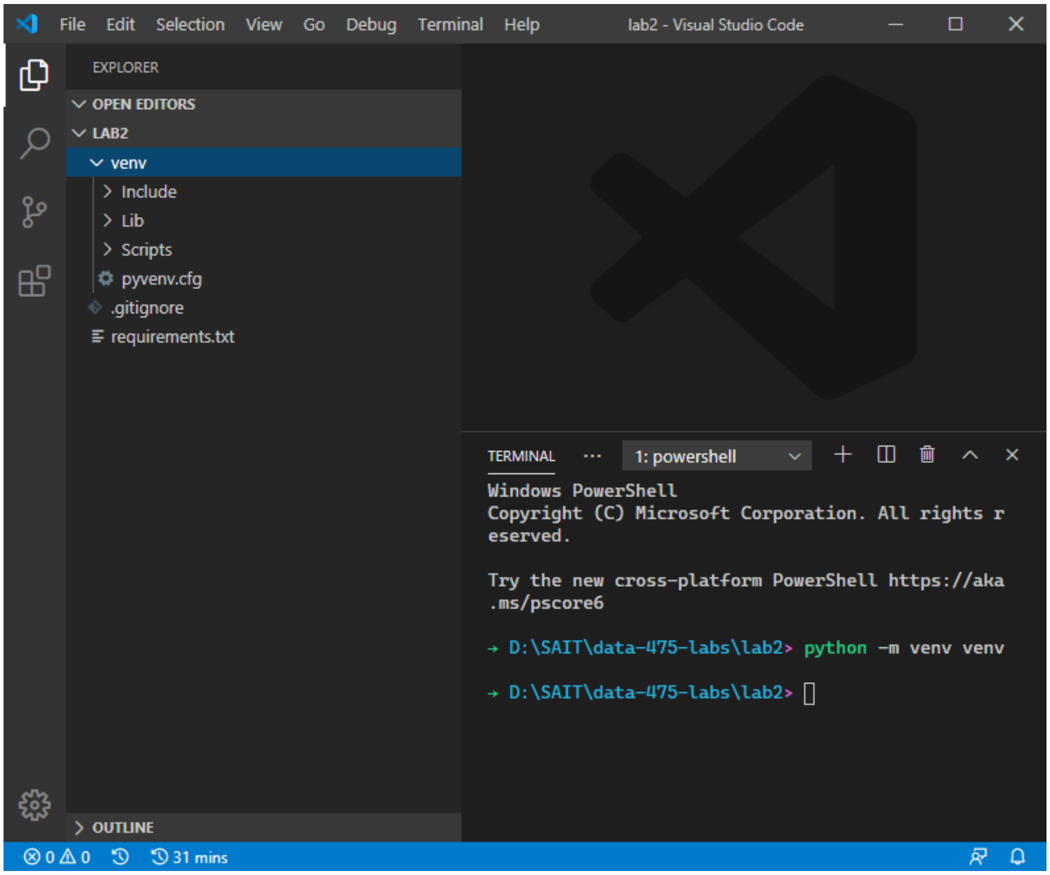
   Using project-wide virtual environments can solve this issue, since each project can install a specific version of the same library in its own virtual environment without interfering with other projects.

   From now on, all labs will require their own virtual environments.

4. Open the terminal window and type the following command to create a virtual environment named **venv**:

   ```
   python -m venv venv
   ```

   Once created, a folder named **venv** appears, as shown below.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights r
eserved.

Try the new cross-platform PowerShell https://aka
.ms/pscore6

→ D:\SAIT\data-475-labs\lab2> python -m venv venv

→ D:\SAIT\data-475-labs\lab2>
```

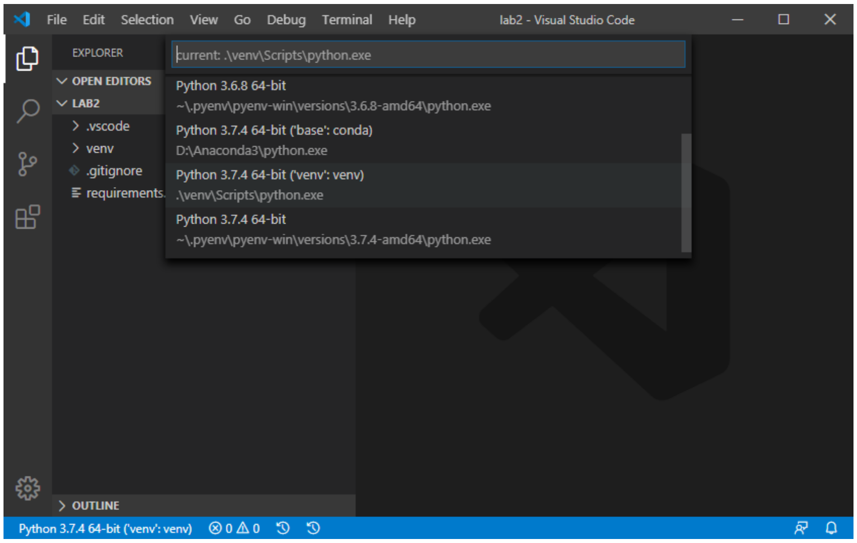
Figure 4: Virtual Environment Created

Used with permission from Microsoft.

5. Configure Visual Studio Code to use the newly created virtual environment as the default Python interpreter.

    a. Click the **Settings** icon in the lower-left corner of the screen and select **Command Palette**.

    b. Click **Python: select interpreter** and select the Python executable in the newly created virtual environment, as shown below.

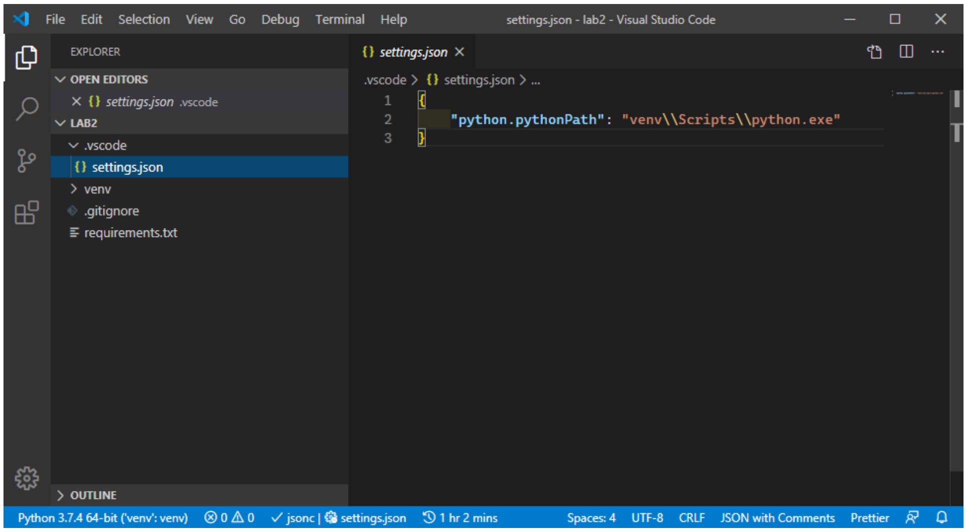


```
current: .\venv\Scripts\python.exe
Python 3.6.8 64-bit
~\.pyenv\pyenv-win\versions\3.6.8-amd64\python.exe
Python 3.7.4 64-bit ('base': conda)
D:\Anaconda3\python.exe
Python 3.7.4 64-bit ('venv': venv)
.\venv\Scripts\python.exe
Python 3.7.4 64-bit
~\.pyenv\pyenv-win\versions\3.7.4-amd64\python.exe
```

Figure 5: Select Python Interpreter

Used with permission from Microsoft.

Visual Studio Code creates a config folder named. vscode in the root of your project folder, as shown below. You can manually configure options by modifying the **settings.json** file.



```
{
    "python.pythonPath": "venv\\Scripts\\python.exe"
}
```

Figure 6: Created. vscode Folder

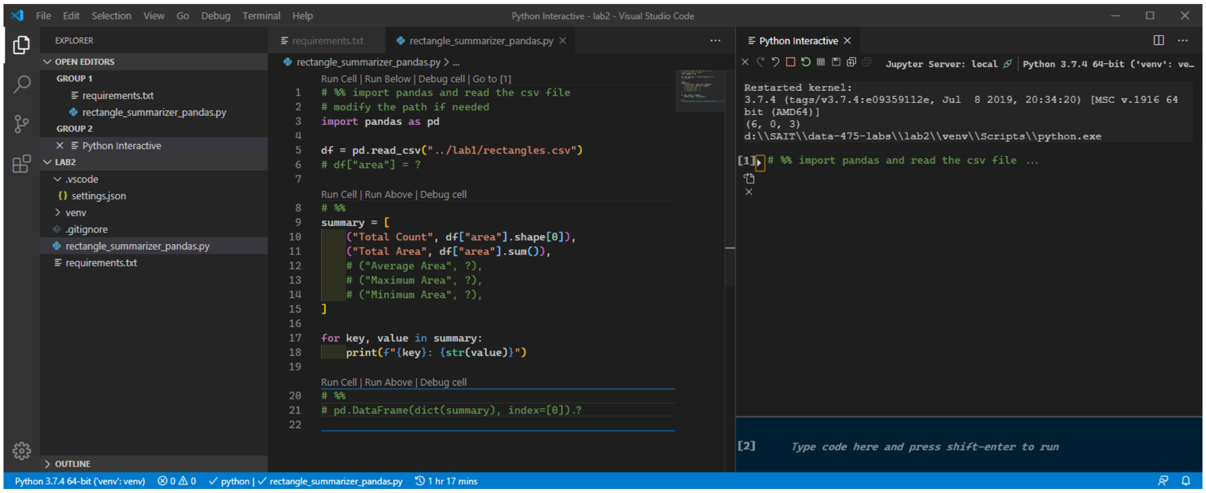
Used with permission from Microsoft.

6. Install dependencies for lab 2.
    a. Open the terminal.
    b. Ensure that the virtual environment is activated.
    c. Type `pip install -r requirements.txt`.
7. Using the **rectangle_summarizer_pandas.py** skeleton file, complete the code in lines 6, 12, 13, 14 and 21.
    a. To run the codes by block, click **Run Cell** on each code block.

       Visual Studio Code opens an interactive Python window, sends the code block to a jupyter server and evaluates the code block, as shown in Figure 7.
    b. Try your code in the input window located in the lower-right section of the screen.

The following panda's API documents may be useful:

- Descriptive statistics (https://pandas.pydata.org/pandas-docs/stable/getting_started/basics.html#descriptive-statistics)
- Getting data in/out (https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#getting-data-in-out)


File
Edit
Selection
View
Go
Debug
Terminal
Help
Python Interactive - lab2 - Visual Studio Code
EXPLORER
OPEN EDITORS
GROUP 1
requirements.txt
rectangle_summarizer_pandas.py
GROUP 2
Python Interactive
LAB2
.vscode
settings.json
venv
.gitignore
rectangle_summarizer_pandas.py
requirements.txt
OUTLINE
requirements.txt
rectangle_summarizer_pandas.py
rectangle_summarizer_pandas.py > ...
Run Cell | Run Below | Debug cell | Go to [1]
Run Cell | Run Above | Debug cell
Python Interactive
Jupyter Server: local
Python 3.7.4 64-bit ('venv': ve...
Type code here and press shift-enter to run
Python 3.7.4 64-bit ('venv': venv)
⊗ 0 ⚠ 0
✓ python | ✓ rectangle_summarizer_pandas.py
1 hr 17 mins

```
1   # %% import pandas and read the csv file
2   # modify the path if needed
3   import pandas as pd
4
5   df = pd.read_csv("../lab1/rectangles.csv")
6   # df["area"] = ?
7

8   # %%
9   summary = [
10      ("Total Count", df["area"].shape[0]),
11      ("Total Area", df["area"].sum()),
12      # ("Average Area", ?),
13      # ("Maximum Area", ?),
14      # ("Minimum Area", ?),
15  ]
16
17  for key, value in summary:
18      print(f"{key}: {str(value)}")
19

20  # %%
21  # pd.DataFrame(dict(summary), index=[0]).?
22
```

```
Restarted kernel:
3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64
bit (AMD64)]
(6, 0, 3)
d:\\SAIT\\data-475-labs\\lab2\\venv\\Scripts\\python.exe
```

```
[1] # %% import pandas and read the csv file ...

[2]
```

Figure 7: Interactive Python Session

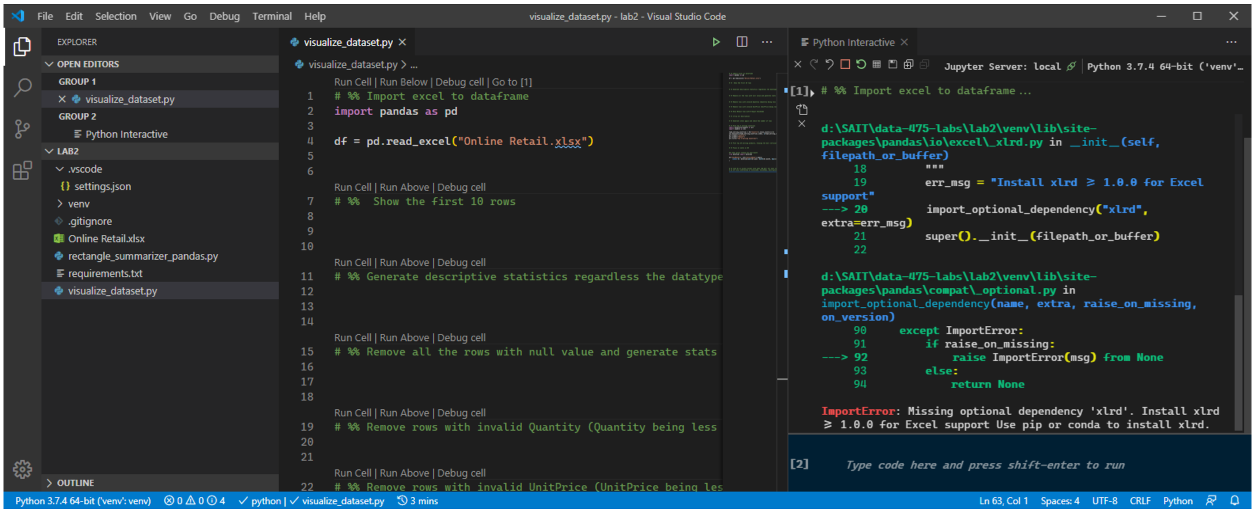
Used with permission from Microsoft.

Lab Activity 2: Visualizing Online Retail Data

In this section, you will use the panda's library again to understand and analyze the online retail dataset and fill in all the empty code blocks in the visualize_dataset.py file to complete an exploratory data analysis.

1. Open the **visualize_dataset.py** file.
2. Read the contents of the dataset Excel file.
    a. Look for a function in pandas named `read_excel`.
    b. Execute the first code block.

       As you might expect, an error appears, as shown in Figure 8.
    c. Read the error message and resolve the issue.
3. To install a dependency from the command line, type `pip install xxx`, where **xxx** is the name of the dependency. Since **xxx** is a must-have dependency for this project, add an item in the requirements.txt file to lock **xxx** to a specific version.



```
1   # %% Import excel to dataframe
2   import pandas as pd
3
4   df = pd.read_excel("Online Retail.xlsx")
5
6

7   # %%  Show the first 10 rows
8
9
10

11  # %% Generate descriptive statistics regardless the datatype
12
13
14

15  # %% Remove all the rows with null value and generate stats
16
17
18

19  # %% Remove rows with invalid Quantity (Quantity being less
20
21

22  # %% Remove rows with invalid UnitPrice (UnitPrice being les
```

```
[1] # %% Import excel to dataframe ...

d:\SAIT\data-475-labs\lab2\venv\lib\site-packages\pandas\io\excel\_xlrd.py in __init__(self, filepath_or_buffer)
     18         """
     19         err_msg = "Install xlrd ≥ 1.0.0 for Excel support"
---> 20         import_optional_dependency("xlrd", extra=err_msg)
     21         super().__init__(filepath_or_buffer)
     22

d:\SAIT\data-475-labs\lab2\venv\lib\site-packages\pandas\compat\_optional.py in import_optional_dependency(name, extra, raise_on_missing, on_version)
     90     except ImportError:
     91         if raise_on_missing:
---> 92             raise ImportError(msg) from None
     93         else:
     94             return None

ImportError: Missing optional dependency 'xlrd'. Install xlrd ≥ 1.0.0 for Excel support Use pip or conda to install xlrd.

[2]     Type code here and press shift-enter to run
```

Figure 8: Error Loading Excel File

Used with permission from Microsoft.

4. Complete the code blocks in line 7, 11, 15, 19, 23, 27, 31 and 35 in the **visualize_dataset.py** file.

   Note: The following pandas API documents may be useful:

   - pandas.DataFrame.head (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.head.html)
   - pandas.DataFrame.describe (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.describe.html)

- pandas.DataFrame.dropna (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html)
- Boolean indexing (https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html#boolean-indexing)
- pandas.DataFrame.astype (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.astype.html)
- pandas.Index.str (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Index.str.html)
- pandas.DataFrame.groupby (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html)

5. Use Python's seaborn library to create a bar chart of the top five selling countries.

   Tip: Look for the **Plot top 5 selling countries** code block.

   If you've correctly filled in the missing code blocks, your output should resemble the image below.

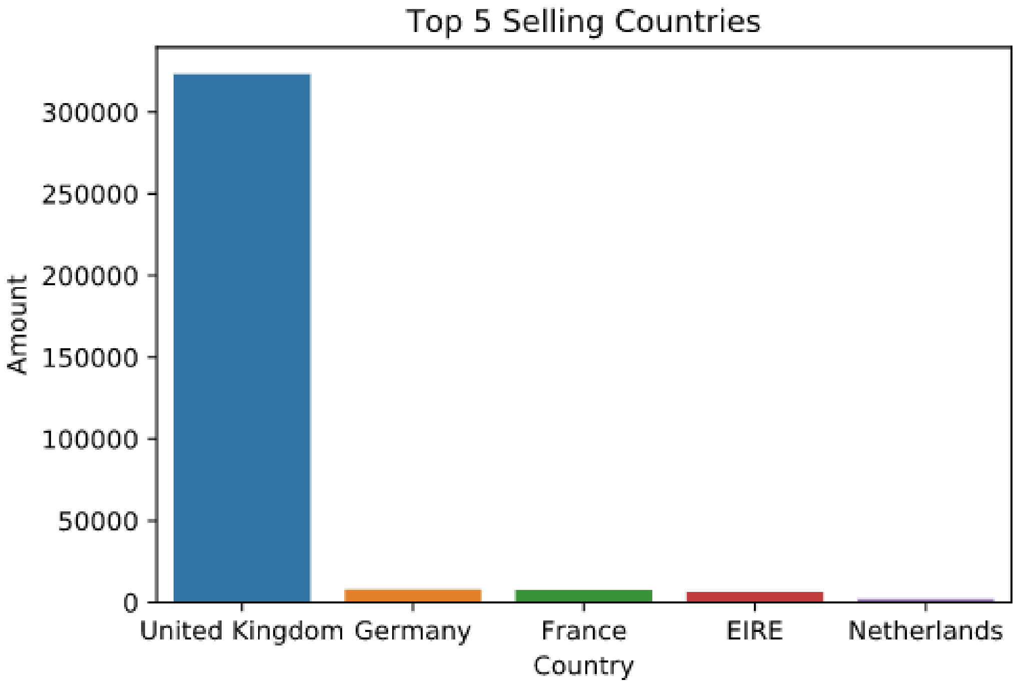


Figure 9: Top Five Selling Countries

6. Now plot the top 20 selling products (in appearance, not by quantity or sale amount), using the production descriptions for the y-axis to improve the appearance of the graph.

   Your output should resemble the image below.

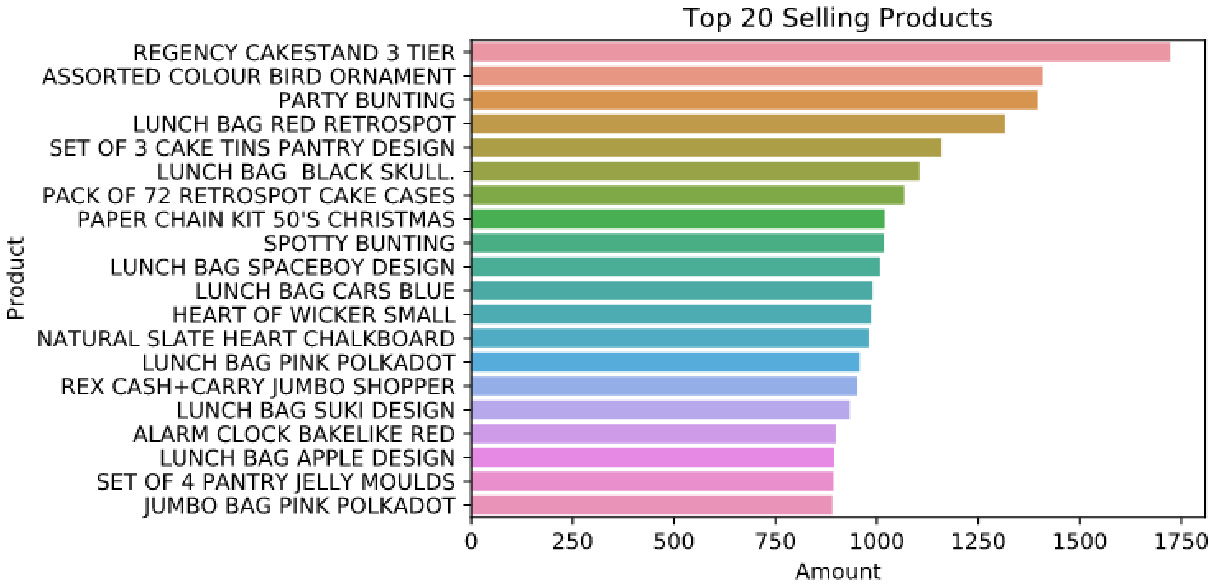


Figure 10: Top 20 Selling Products

© 2020, Southern Alberta Institute of Technology

7. Visualize the gross revenue by month and year.
    a. Look for the second last code block, which contains sample code to create a new column named “YearMonth” to capture only the year and month information.
    b. Group the dataset by the “YearMonth” column and sum up the sales.

Tips:

- Use seaborn’s `lineplot` function.
- pandas’ `groupby` object has an attribute named `index`, which can be used for your x-axis.

Your output should resemble the image below.

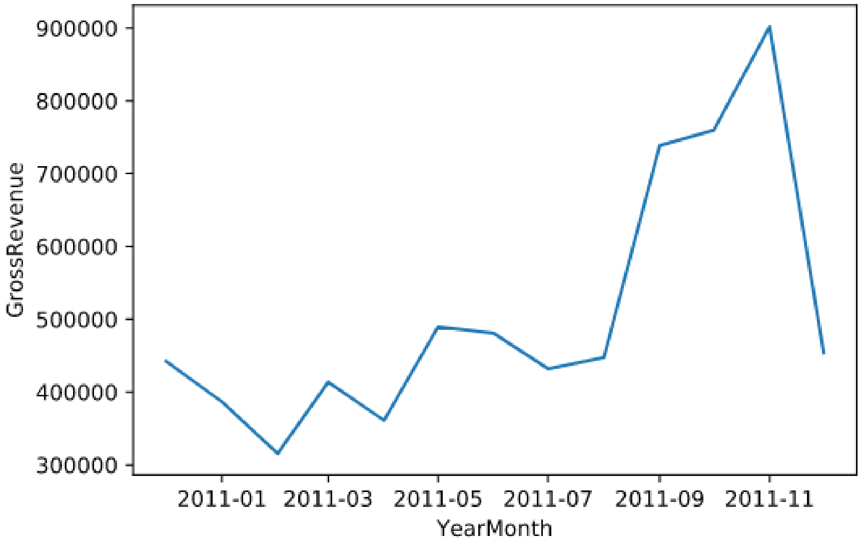


Figure 11: Gross Revenue by YearMonth

© 2020, Southern Alberta Institute of Technology

8. Using the `to_pickle` function to save in pickle format, save all the United Kingdom transactions and name your file **UK.pkl** so that you can practice association rules mining.

Lab Activity 3: Mining Association Rules from Online Retail Data

In this section, you will apply the Apriori algorithm to find frequent items and mine association rules from the online retail dataset. You'll fill in the code blocks in the association_rule_mining.py file to complete this activity.

1. Open the **association_rule_mining.py** file.
2. Read the pickle file **UK.pkl** into a dataframe. Sample code is provided in the first code block.
3. Convert the dataframe into transactional format as needed using the Apriori algorithm.
    a. Read Frequent Itemsets via Apriori Algorithm (http://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/#frequent-itemsets-via-apriori-algorithm).
    b. Since all items in the same invoice should belong to a transaction, group the dataframe by invoice number.
    c. The aggregate function returns “Description” in a list.
4. Determine an appropriate **min_support** value for the Apriori algorithm. (Hint: It won't be as high as the number we saw for the toy project in class.) What is the reason for this?
5. Examine the frequent itemsets and determine how many itemsets contain over 1/2/3 items.
6. Extract the frequent itemsets with the most items.
7. Extract the top 10 association rules with highest lift value.
8. Use a seaborn scatterplot to visualize the relationship between support and confidence. Sample code is provided in the second last code block.

    Your output should resemble the image below.

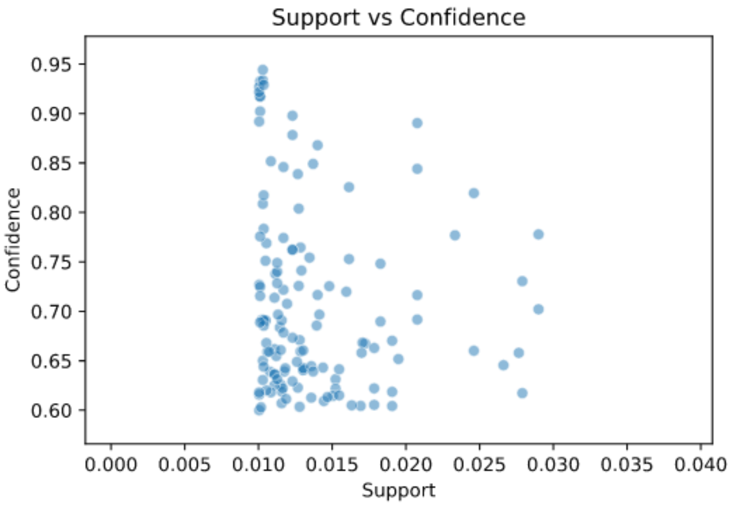


Figure 15: Support vs Confidence

© 2020, Southern Alberta Institute of Technology

9. Perform a similar exercise to visualize the relationship between support and lift.

   Your output should resemble the image below.

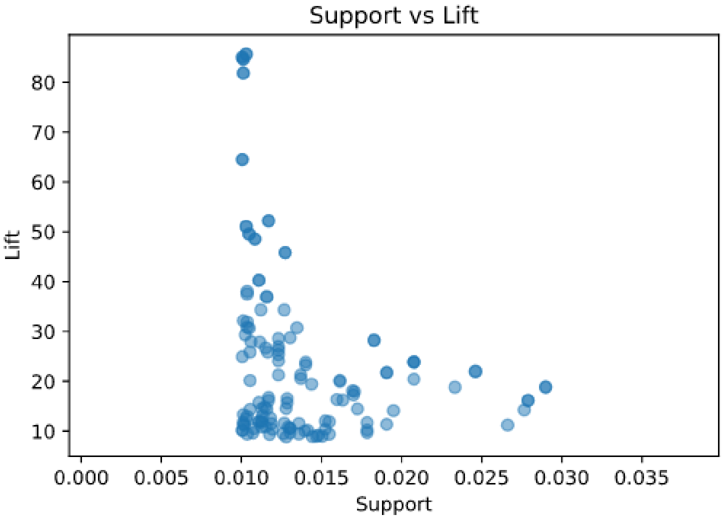


Figure 16: Support vs. Lift

© 2020, Southern Alberta Institute of Technology

References

Chen, D., Sain, S.L., & Guo, K. (2012). Data mining for the online retail industry: A case study of RFM model-based customer segmentation using data mining. *Journal of Database Marketing and Customer Strategy Management*, *19*(3), 197–208.