Interim Report

Degree: Data Science BSc (Hons)
Module: CIS3140 Research and Development Project

By: Christopher Diaz Montoya
ID: 24707686

Module Lecturer: Ella Pereira
Module Supervisor: Marcello Trovati

Contents

Introduction

In 2023 there are many companies that use data, one use of data in e-commerce is for inventory management, this would track sales and inventory. Being able to use tools to forecast inventory is something companies are happy to invest money in as it would improve profits (Abeysekara and Rupasinghe, 2019). Inventory forecasting does this by reducing overstocking as there is a cost to store goods, paired with reducing the amount of stock that would pass its sell by date, it also reduces understocking inventory which leads to unhappy customers and loss of sales, another benefit being that it reduces administrative work (Puneet et al, 2021).

This report will show the progress of the creation of an inventory forecasting and analytical tool. This would accept a spreadsheet of data in a specific format that can be updated and then produce a forecast of inventory along with having a basic analytical dashboard for viewing historical data. This would be done with underlying machine learning algorithms which are designed to use historic data to predict future trends.

This report is split into multiple sections, and they will be the project background and the literature review; the projects aims and objectives; the methods, changes to the project; progress report against the plan; the updated plan; and the conclusion with the appendices at the end of the document.

Project Background

One aim is to make this tool accessible as accessible as possible for struggling e-commerce small and medium enterprises (SMEs) as data tools are not cheap (Benhamida et al, 2020). One important project aims and cause for the project is to create a cheap, trustworthy, accessible Artificial Intelligence and analytical tool for small e-commerce business owners as data tools are not cheap (Benhamida et al, 2020). The goal is to create a place where the past years sales and inventory can be tracked and analysed along with being able to forecast inventory to allow for better business decisions.

Currently analytical tools exist such as Stock & Buy for e-commerce companies, but they cost over a £100 a month which is not viable for small businesses. Businesses track their sales and track inventory as close as possible to not make any decisions which could cause an effect to the business.

When looking at machine learning techniques which could be used for inventory forecasting, a regression-based algorithm was found to be of the best fit when there is historic data for a product, this algorithm is used in industry (Ranjitha and Spandana, 2021). When there is little to no historical data it can be difficult to predict the future, here a clustering-based algorithm was found to be best, this is as it allows to be grouped to similar products (Benhamida et al., 2020). No algorithm would be able to be 100% accurate. This meant that a buffer should be added. When thinking of worst-case scenarios an algorithm would likely cause the user to overstock or understock when wrong. As these algorithms are not vastly off, an overstock buffer could be included to ensure understocking does not 5 happen and has been proven to work well with regression algorithms (Puneet, 2021). The algorithm is meant to be a predictive guide and not a concrete view into the future, although that is the aim.

When looking into how to visualise the data a free to create interactive dashboard, without having to learn HTML or Java script, was found, Dash. This allows to integrate python libraries and their analytical tools onto the web, the loading time was quick as well (Clement et al, 2020).

Aim and Objectives

The aim of the project is to create an accessible analytical inventory forecasting and analytics tools. To do this some key objectives were established to measure success in the project.

Objectives	How	Approach
Trustworthy	The forecasting algorithm must have a high degree of confidence.	The machine learning algorithm created will be evaluated using ensemble learning and then evaluating the algorithm for accuracy in a number of ways such as through accuracy, f1-score or Mean Squared Error (MSE). Along with a forecasting buffer and using k-folds for improved ML algorithm accuracy
Cheap and Accessible	Trying to not include a charge, making it easy to access online.	By making this into a free to use website where the data can be viewed and shown, a tool that is being considered is Dash or Flask.
Visually Appropriate	Making it easy to understand the visualised data and navigate through the page.	This is done by not only ensuring that the graphs are colourful in way that makes them easy to interpret, but ensuring they show what is needed, along with having the graphs appropriately labelled.
Check if weather effects inventory forecasting	Adding historical weather data to the date of sales to see if there is a correlation.	An API will be used to connect historic data of the weather

		and will be added into the data frame, this allows for the weather to be a analysed as a variable in a heat map as weather is overlooked in the found papers (Abeysekara and Rupasinghe, 2019).

Changes to Original Project Proposal

In terms of changes to the original project proposal there are none.

Progress Report Against the Plan / Selected methods

Finding an available data set that was free to use and had the necessary columns was difficult to find, eBay and amazon had some available datasets at a cost. As such different websites such as google, GitHub and Kaggle were investigated to find a free dataset suitable for inventory forecasting and sales data. A data set was found that had columns which were deemed a good fit, after consideration on if the columns had the date, price, stock information, location of sale and total transaction value. As the data set took longer to find than anticipated, it set the project back a week.

The weather API found which had a lot of resources on its website to understand how to use it was found called OpenWeather. The base URL was added in, and the API key and they were merged together to gain access to the data. It has the temperature, wind speed, description of the forecast and more details. This API has been connected to the google collab notebook and experimented with to see the available data. The temperature was given in Kelvin from the API, so this had to be turned into Celsius through a function. The API data was in the format of a JSON, and this needed to be converted to a data frame. This was done successfully by checking the data type of how the data was stored in the API keys. The weather data that pulled across was only of the current day. The next step is to call historic and not current data from the OpenWeather API in the weather data frame and convert from Kelvin to Celsius, as this was done as trial code to ensure it works. Then connect the historic data from Open weather to the main data frame on each row, following this a heatmap can be done on the whole data set to see the correlation between columns. This can be seen in Appendix F and G.

The data set was stored in google drive and was connected through google collab using the drive.mount() function, this allows for easy access and as the notebook and data set are both stored on the cloud it allows for easy accessibility and eliminates the risk of losing the work completed. At first the dataset would not upload so chardet.detect() was used to help detect what the csv dataset is encoded in as it

would not open. This was learned from second year which in layman terms is when certain spreadsheets have different byte standards, this means that some are read differently than others. This can be seen in Appendix D.

Now the data is uploaded with the help of drive.mount(), the file path, and the pandas library, the data set can be explored. To begin with df.head(), df.isnull().sum(), df.info(), and df.describe() were all used to further understand the dataset. The first helps view the dataset in tabular format with as many rows as desired. The second allows to view missing values, there were some in the dataset and as of now the product description has a lot of missing values, this column will be checked against stock code to see if these are the same column, the product code and product description. The Customer ID has missing values and this can not be replaced with the mean, mode or median, this will be looked at further to check if it has a correlation with the country or invoice number to fill missing values or consider reasons as to why this data is missing. This can be seen in Appendix D and E.

In Appendix I, a couple of things were done, the date columns were turned into the date time data type and split into a date and time column. The data set had some order codes which begin with a C, these had to be dropped as they were cancelled orders. Finally, a total order price column was non existent so the quantity was multiplied by the unit price, this will be looked at as it would be important to do it based of the invoice number to match other rows as well as the quantity.

The data analysis is currently under way using matplotlib and seaborn and so far, the country sales, the total monthly and the total weekly sales have been looked at. This was done using the .groupby().sum() function to help group countries and add up the prices of all the orders. This is shown in Appendix H, J and K. A bar chart was used for the countries total sales to see where the most money is flowing from and a line chart for the total monthly and weekly sales to help spot trends throughout the year.

Next Steps

In terms of next steps, the data analysis needs to be completed, to do this the Open Weather API needs to be fully integrated into the main data frame. Once the data analysis is complete Appendix L shows the next steps. The data would then need to be encoded to all be a numeric value. Then a seaborn heatmap can be created for a correlation analysis. After this a modified k-folds will be used on the time series data, instead of splitting it up into equal chunks I will start with 50% of the data set using the hold out method for training and test data. Increasing the data set 10% at a time until 100% of the data set is used. This will be used for machine learning models, ARIMA, XG boost and a clustering-based algorithm. The algorithms will be optimised and displayed on a dashboard.

Updated Project Plan/Work Schedule

In terms of the new updated project pan, it can be seen in Appendix B. Appendix A shows the previous project plan. Most activities have been pushed back by one week; this was because a dataset needed to be found and a learning curve on using the weather API. Given this the project is still on schedule and the majority of April is

still going to be used for tidying up the dashboard and optimising the machine learning algorithms further.

References

ABEYSEKARA, K, T. and RUPASINGHE, S., 2019. Analysis of Influential Factors for Inventory Forecasting Systems. 2019 IEEE 5th International Conference for Convergence in Technology (I2CT). 29-31 March 2019. Bombay, India. Piscataway, N.J.: Institute of Electrical and Electronics Engineers. pp. 1-4. Available from: https://ieeexplore.ieee.org/document/9033725 [Accessed 8 November 2022].

BENHAMIDA, F, Z. KADDOURI, O. OUHROUCHE, T. BENAICHOUCHE, M. CASADO-MANSILLA, D. and LOPEZ-DE-LPINA, D., 2020. Stock&Buy: A New Demand Forecasting Tool for Inventory Control. 2020 5th International Conference on Smart and Sustainable Technologies (SpliTech), 23-26 September 2020, Split, Croatia. Piscataway, N.J.: Institute of Electrical and Electronics Engineers. Pp. 1-6. Available from: https://ieeexplore.ieee.org/document/9243824 [Accessed 07 November 2022].

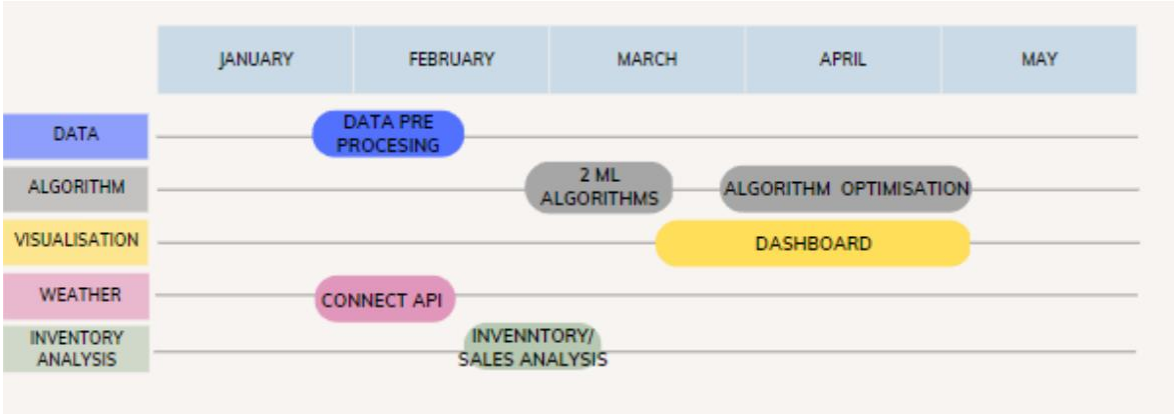
CLEMEMT, F., KAUR, A., SEDGHI, M., KRISHNASWAMY, D. and PUNITHAKUMAR, K. 2020. Interactive Data Driven Visualization for Covid-19 with Trends, Analysis and Forecasting. 2020 24th International Conference Information Visualization (IV). 07– 11 September 2020. Melbourne, Australia. Piscataway, N.J.: Institute of Electrical and Electronics Engineers. pp. 593-598. Available from: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9373291 [Accessed 11 of November 2022].

OPENWEATHER., N/A. *History API* [online]. Available from: https://openweathermap.org/history [Accessed 25 February 2023].

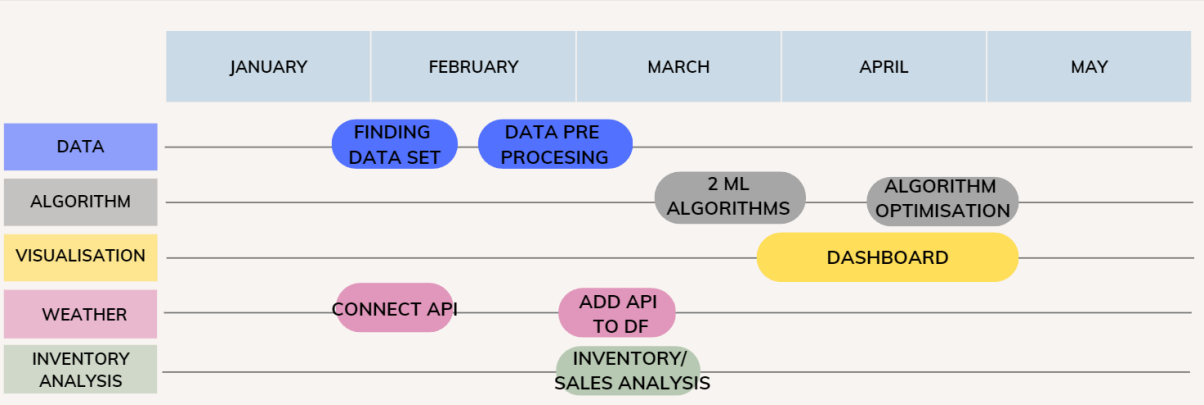
PUNEET,. SHARMA, S. DEEPIKA, D. and SINGH, G., 2021. Intelligent Warehouse Stocking Using Machine Leaning. 2021 International Conference of Mobile Networks and Wireless Communications (ICMNWC), 03-04 December 2021, Tumkur, Karnataka, India. Piscataway, N.J.: Institute of Electrical and Electronics Engineers. pp. 1-6. Available from: https://ieeexplore.ieee.org/document/9688530 [Accessed 06 November 2022].

RANJITHA, P., and SPANDANA, M., 2021. Predictive Analysis for Big Mart Sales Using Machine Learning Algorithms. 5 th International Conference on Intelligent Computing and Control Systems (ICICCS). 06-08 May 2021. Madurai, India. Piscataway, N.J.: Institute of Electrical and Electronics Engineers. pp. 1416-1421. Available from: https://ieeexplore.ieee.org/abstract/document/9432109/metrics#metrics [Accessed 11 November 2022].

Appendix A



Appendix B



Appendix C

Libraries imported

```
import pandas as pd
import requests
import seaborn as sns
import matplotlib.pyplot as plt

from google.colab import drive # Gets my data stored in the cloud
drive.mount('/content/drive')
# For weather API

# Libraries imported, if imported correctly the below will print
print("Libraries imported.")
```

```
Mounted at /content/drive
Libraries imported.
```

Appendix D

Upload data set

```
[2] # Data set upload
    filepath = '/content/drive/MyDrive/Dissertation/data.csv'
```

```
[3] # I needed the below code
    """
    import chardet # Chardet to detect the encoding

    # Opens the file in a binary mode (EXPLAIN MORE)
    with open(filepath, 'rb') as f:
        result = chardet.detect(f.read())

    print(result['encoding']) # Prints how the CSV is encoded
    """
```

```
'\nimport chardet # Chardet to detect the encoding\n\n# Opens the file in a binary m
LAIN MORE)\nwith open(filepath, 'rb') as f: \n    result = chardet.detect(f.read())\
(result['encoding']) # Prints how the CSV is encoded\n'
```

```
[4] # First look at the dataset
    df = pd.read_csv(filepath, encoding='ISO-8859-1') # Encoding as would not open
    df.head(1)
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0

Appendix E

```
# Initial look at the data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
```

```
df.isnull().sum()
```

```
InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID    135080
Country            0
Date               0
InvoiceTime        0
Cancelled          0
TotalPrice         0
dtype: int64
```

```
[10] # Checking the df
df.describe()
```

	Quantity	UnitPrice	CustomerID	TotalPrice
count	541909.000000	541909.000000	406829.000000	541909.000000
mean	9.552250	4.611114	15287.690570	17.987795
std	218.081158	96.759853	1713.600303	378.810824
min	-80995.000000	-11062.060000	12346.000000	-168469.600000
25%	1.000000	1.250000	13953.000000	3.400000
50%	3.000000	2.080000	15152.000000	9.750000
75%	10.000000	4.130000	16791.000000	17.400000
max	80995.000000	38970.000000	18287.000000	168469.600000

Appendix F

```
[13] # The weather API to link the histprical weather data
     Base_URL = "http://api.openweathermap.org/data/2.5/weather?"
     API_Key = "1e81f8cce84f885d279386c4a8f53d23"
     CITY = "London"

     url = Base_URL + "appid=" + API_Key + "&q=" + CITY
     API_json_data = requests.get(url).json()

     print(API_json_data)
```

```
{'coord': {'lon': -0.1257, 'lat': 51.5085}, 'weather': [{'id': 804, 'main': 'Clouds',
```

```
# Function the turns weather to calsius
def kelvin2celsius(kelvin):
  celsius = kelvin - 273.15
  return celsius
```

```
[22] # Temp is in kelvin so turned it into celsius
     temp_kelvin = API_json_data ["main"]["temp"]
     temp_celsius = kelvin2celsius(temp_kelvin)
     feels_like_kelvin = API_json_data ["main"]["feels_like"]
     feels_like_celsius = kelvin2celsius(feels_like_kelvin)
     # Description is the weather description which could be important
     description = API_json_data["weather"][0]["description"]

     print(f"Current temperature in {CITY}: {temp_celsius:.2f} C")
     print(f"Current temperature in {CITY} feels like: {feels_like_celsius:.2f} C")
     print(f"General Weather in {CITY}: {description}")
```

```
Current temperature in London: 2.77 C
Current temperature in London feels like: 0.19 C
General Weather in London: overcast clouds
```

Appendix G

```
# Shows available keys
API_json_data.keys()
```

```
dict_keys(['coord', 'weather', 'base', 'main', 'visibility', 'wind', 'clouds', 'dt', '
'timezone', 'id', 'name', 'cod'])
```

```
[24] # The two I will use look at type to convert to df
print("Data type for main is: ", type(API_json_data["main"]))
print("Data type for weather is: ", type(API_json_data["weather"]))
```

```
Data type for main is:  <class 'dict'>
Data type for weather is:  <class 'list'>
```

```
[25] # Convert to df
MainDf = pd.DataFrame(API_json_data["weather"], index = [0])
MainDf
```

	id	main	description	icon
0	804	Clouds	overcast clouds	04n

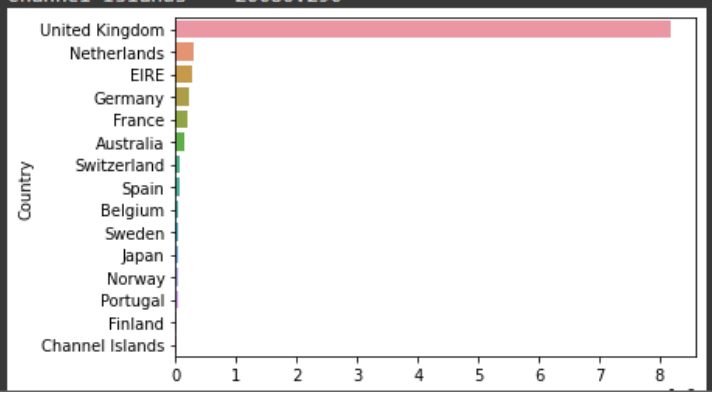
```
# Converted to df but temp in kelvin need to fix
WeatherDf = pd.DataFrame(API_json_data["main"], index = [0])
WeatherDf
```

	temp	feels_like	temp_min	temp_max	pressure	humidity
0	275.92	273.34	274.29	277.01	1009	84

Appendix H

```
# Creating a new df to see sales across the countries
df_countries = df[["Country", "TotalPrice"]]

# Below is grouping all the countries and sum to see biggest buyers
df_countries = df_countries.groupby("Country").sum()

# Checking sales in top 50 countries
df_countries = df_countries.sort_values("TotalPrice", ascending=False)
print(df_countries.head(15))

# Seaborn to create a bar plot
sns.barplot(x = "TotalPrice", y = "Country", data = df_countries.reset_index().head(15))

plt.show()
```

```
                TotalPrice
Country
United Kingdom  8187806.364
Netherlands      284661.540
EIRE             263276.820
Germany          221698.210
France           197403.900
Australia        137077.270
Switzerland       56385.350
Spain             54774.580
Belgium           40910.960
Sweden            36595.910
Japan             35340.620
Norway            35163.460
Portugal          29367.020
Finland           22326.740
Channel Islands   20086.290
```



Appendix I

```
[6]  # Turns columns to datetime
     df["InvoiceDate"] = pd.to_datetime(df["InvoiceDate"])

     df["Date"] = df["InvoiceDate"].dt.date
     df["InvoiceTime"] = df["InvoiceDate"].dt.time
```

```
     # Find the str not letting us change the data tyoe
     print(df[df["InvoiceNo"] == "C536383"])

     # Data set says these are cancelled orders if start with C so dropped
     df["Cancelled"] = ["C" in str(FirstLetter) for FirstLetter in df["InvoiceNo"]]
```

```
     InvoiceNo StockCode                      Description  Quantity  \
154   C536383    35004C  SET OF 3 COLOURED  FLYING DUCKS        -1

            InvoiceDate  UnitPrice  CustomerID         Country        Date  \
154 2010-12-01 09:49:00       4.65     15311.0  United Kingdom  2010-12-01

    InvoiceTime
154    09:49:00
```

```
[8]  # Have a total order column
     df["TotalPrice"] = df["Quantity"] * df["UnitPrice"]

     df.head(2)
```

Appendix J

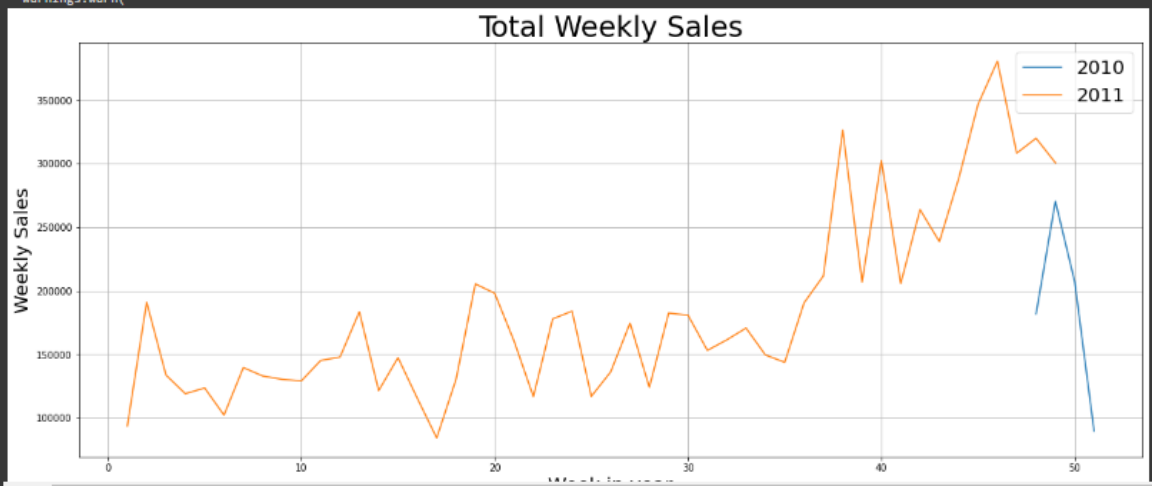
Checking Sales across the year

```
[22] # Create own month and year column with pandas to see if month has any correlation with the data set
     df["Year"] = pd.to_datetime(df["Date"]).dt.year
     df["Month"] = pd.to_datetime(df["Date"]).dt.month
     df["Week"] = pd.to_datetime(df["Date"]).dt.isocalendar().week
     # df.head(1)
```

```
[29] # Data to go in the graph, done by year and then specfically week compared to weekly sales column
     # Gives the mean of each week in the year
     country_sales_year1 = df[df["Country"] == 2010]["TotalPrice"].groupby(df["Week"]).sum()
     country_sales_year2 = df[df["Year"] == 2011]["TotalPrice"].groupby(df["Week"]).sum()
```

```
# Data to go in the graph, done by year and then specfically week compared to weekly sales column
# Gives the mean of each week in the year
weekly_sales_year1 = df[df["Year"] == 2010]["TotalPrice"].groupby(df["Week"]).sum()
weekly_sales_year2 = df[df["Year"] == 2011]["TotalPrice"].groupby(df["Week"]).sum()

# Creates graph to show weekly sales in the weeks to see if there is a correlation
plt.figure(figsize=(20,8))
sns.lineplot(weekly_sales_year1.index, weekly_sales_year1.values)
sns.lineplot(weekly_sales_year2.index, weekly_sales_year2.values)

# Graph adjustments
plt.grid() # To make the graph easier to read
plt.legend(["2010", "2011"], fontsize = 20)
plt.title("Total Weekly Sales", fontsize = 30)
plt.ylabel("Weekly Sales", fontsize = 20)
plt.xlabel("Week in year", fontsize = 20)

# Displays graph
plt.show()
```

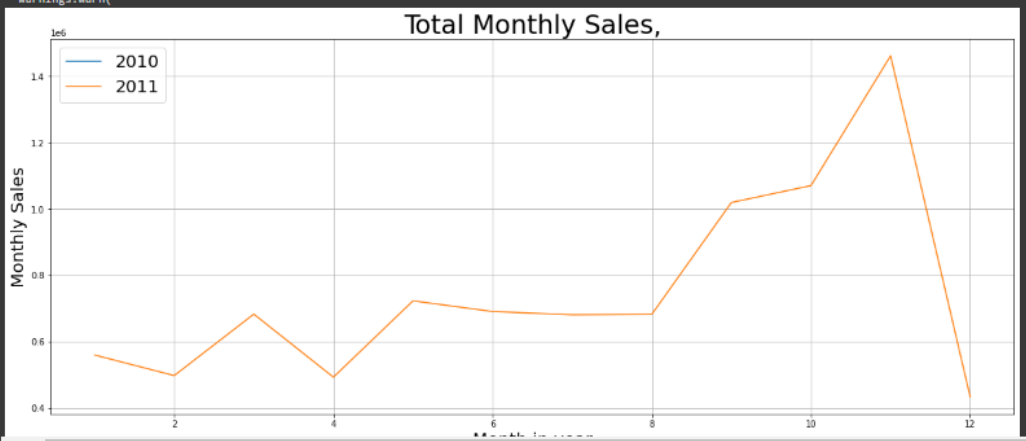
```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the
  warnings.warn(
```



Appendix K

```
# Data to go in the graph, done by year and then specfiallly week compared to weekly sales column
# Gives the mean of each week in the year
monthly_sales_year1 = df[df["Year"] == 2010]["TotalPrice"].groupby(df["Month"]).sum()
monthly_sales_year2 = df[df["Year"] == 2011]["TotalPrice"].groupby(df["Month"]).sum()

# Creates graph to show weekly sales in the weeks to see if there is a correlation
plt.figure(figsize=(20,8))
sns.lineplot(monthly_sales_year1.index, monthly_sales_year1.values)
sns.lineplot(monthly_sales_year2.index, monthly_sales_year2.values)

# Graph adjustments
plt.grid() # To make the graph easier to read
plt.legend(["2010", "2011"], fontsize = 20)
plt.title("Total Monthly Sales,", fontsize = 30)
plt.ylabel("Monthly Sales", fontsize = 20)
plt.xlabel("Month in year", fontsize = 20)

# Displays graph
plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid
  warnings.warn(
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following variables as keyword args: x, y. From version 0.12, the only valid
  warnings.warn(
```



Appendix L

Encoding to numeric/binary value

```
# One hot encoding

# ChoosenColumns = ["InvoiceNo", "StockCode", "Description", "Country"]

# OneHotEncoding = pd.get_dummies(df[ChoosenColumns], prefix = ChoosenColumns
```

Correlation Matrix

Test Train Split

Machine Learning Algorithms

Results

Connect to flask or Dash

```
[ ]  Flask connects code to html
```