8f6tvqtb6

April 18, 2025

[ ]:
```
#----Introduction to Machine Learning & AI - DAT-5329 - LMBAN1----#
#---A3: Individual Assignment---#
#---Sayefaldeen Suleiman---#
```

[89]:
```
#Importing pandas as Pd to ensure analysis use
import pandas as pd
# ------------------ File Display ------------------ #
df = pd.read_excel("Online Retail.xlsx")  # Specify the UK Online sales data
df.head()  # review the first few rows
df.info()  # Checking data types and missing values
df.describe()  # Summary statistics
```

```
<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  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

[89]:

	Quantity	InvoiceDate	UnitPrice	\
count	541909.000000	541909	541909.000000	
mean	9.552250	2011-07-04 13:34:57.156386048	4.611114	
min	-80995.000000	2010-12-01 08:26:00	-11062.060000	
25%	1.000000	2011-03-28 11:34:00	1.250000	
50%	3.000000	2011-07-19 17:17:00	2.080000	
75%	10.000000	2011-10-19 11:27:00	4.130000	
max	80995.000000	2011-12-09 12:50:00	38970.000000	
std	218.081158	NaN	96.759853	

```
          CustomerID
count  406829.000000
mean    15287.690570
min     12346.000000
25%     13953.000000
50%     15152.000000
75%     16791.000000
max     18287.000000
std      1713.600303
```

[91]:
```
# ------------------ Data Cleaning ------------------#

df = df[~df['InvoiceNo'].astype(str).str.startswith('C')]  # Removing cancelled␣
 ↪orders Which can obstrect my analysis
df = df.dropna(subset=['CustomerID'])  # Drop missing Customer IDs
df = df[df['Quantity'] > 0]
df = df[df['UnitPrice'] > 0]
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])  # Convert date
```

[93]:
```
# ------------------ Feature Engineering ------------------#

# feature engineering: Revenue. To ensure greate analysis for my report
# calculating to reflect the total sale value per line item
df['Revenue'] = df['Quantity'] * df['UnitPrice']

# Displaying both cleaned and feature-engineered rows
df.head()
```

[93]:

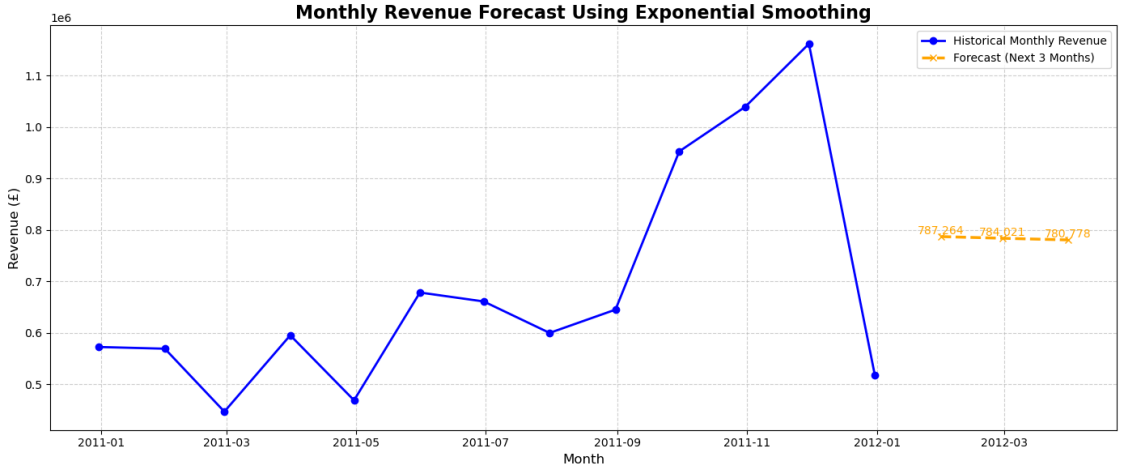
	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Revenue
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34

[121]:
```
from statsmodels.tsa.holtwinters import ExponentialSmoothing
import matplotlib.pyplot as plt
import numpy as np
# ------------------ Exponential Smoothing Model ------------------ #
monthly_revenue_ts = monthly_revenue.set_index('InvoiceDate')
```

```
monthly_revenue_ts = monthly_revenue_ts.asfreq('M')  # Ensureing the proper␣
 ↪monthly datetime index

# Fitting Exponential Smoothing model with correct datetime index
model = ExponentialSmoothing(
    monthly_revenue_ts['Revenue'],
    trend='add',
    seasonal=None,
    initialization_method="legacy-heuristic"
)
fit = model.fit()
forecast = fit.forecast(3)

# ------------------ Plot Forecast ------------------ #
plt.figure(figsize=(14, 6))

# Plotting historical monthly revenue
plt.plot(monthly_revenue_ts.index, monthly_revenue_ts['Revenue'],␣
 ↪label='Historical Monthly Revenue', linewidth=2, marker='o', color='blue')

# Plotting forecasted revenue
plt.plot(forecast.index, forecast, label='Forecast (Next 3 Months)',␣
 ↪linestyle='--', linewidth=2.5, marker='x', color='orange')

# Annotating forecast values
for date, value in forecast.items():
    plt.text(date, value, f'{value:,.0f}', ha='center', va='bottom',␣
 ↪fontsize=10, color='orange')

# chart formatting
plt.title("Monthly Revenue Forecast Using Exponential Smoothing", fontsize=16,␣
 ↪weight='bold')
plt.xlabel("Month", fontsize=12)
plt.ylabel("Revenue (£)", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.tight_layout()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_19400\2425412558.py:6:
FutureWarning:

'M' is deprecated and will be removed in a future version, please use 'ME'
instead.

C:\Users\Admin\Desktop\Other\Anaconda\Lib\site-
packages\statsmodels\tsa\holtwinters\model.py:918: ConvergenceWarning:
```

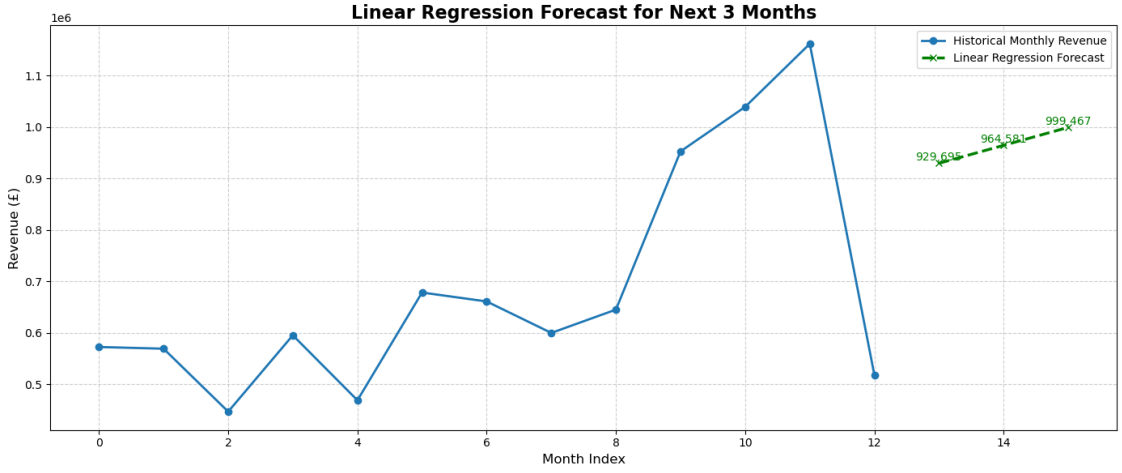
```
Optimization failed to converge. Check mle_retvals.
```



[105]:
```
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt

# I'm reseting the index to convert 'InvoiceDate' from index to a column
monthly_revenue = monthly_revenue.reset_index()

# Creating a numerical index to represent each month. This is needed because␣
 ↪linear regression doesn't handle datetime objects directly
monthly_revenue['MonthIndex'] = np.arange(len(monthly_revenue))

# Defineing features (X) and target (y)
X = monthly_revenue[['MonthIndex']]          # Feature: time as a numeric index
y = monthly_revenue['Revenue']               # Target: revenue for each month

# Initialize and train the Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X, y)

# Forecasting revenue for the next 3 months
# I used the next indices after the last observed point
lr_pred = lr_model.predict([[len(X)], [len(X)+1], [len(X)+2]])

# ------------------ Plot ------------------ #

# Ploting revenue
plt.figure(figsize=(14, 6))
```

```
plt.plot(monthly_revenue['MonthIndex'], y, label='Historical Monthly Revenue',␣
 ↪linewidth=2, marker='o')

# Ploting predicted values
future_indices = [len(X), len(X)+1, len(X)+2]
plt.plot(future_indices, lr_pred, label='Linear Regression Forecast',␣
 ↪linestyle='--', linewidth=2.5, marker='x', color='green')

# Annotating predicted values
for i, value in zip(future_indices, lr_pred):
    plt.text(i, value, f'{value:,.0f}', ha='center', va='bottom', fontsize=10,␣
 ↪color='green')

# Customizing the plot
plt.title("Linear Regression Forecast for Next 3 Months", fontsize=16,␣
 ↪weight='bold')
plt.xlabel("Month Index", fontsize=12)
plt.ylabel("Revenue (£)", fontsize=12)
plt.grid(True, linestyle='--', alpha=0.6)
plt.legend()
plt.tight_layout()
plt.show()
```

```
C:\Users\Admin\Desktop\Other\Anaconda\Lib\site-packages\sklearn\base.py:493:
UserWarning:

X does not have valid feature names, but LinearRegression was fitted with
feature names
```

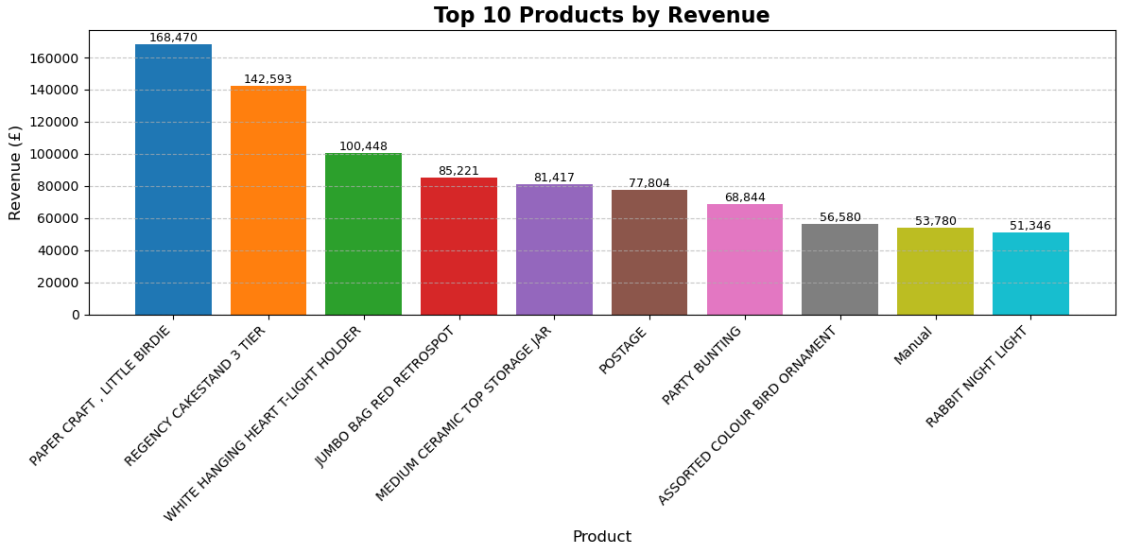


[107]:
```
import pandas as pd
import plotly.express as px # I'm Importing this to get a geo chart

# Grouping revenue by country
revenue_by_country = df.groupby('Country')['Revenue'].sum().reset_index()

# Filtering low revenue countries
revenue_by_country = revenue_by_country[revenue_by_country['Revenue'] > 1000]

# Defining custom color scale
custom_colors = ['red', 'yellow', 'green']  # red, yellow, green

# Creating and plotting geo chart
fig = px.choropleth(
    revenue_by_country,
    locations='Country',
    locationmode='country names',
    color='Revenue',
    hover_name='Country',
    color_continuous_scale=custom_colors,
    title='Revenue by Country'
)

# Formating the layout
fig.update_layout(
    geo=dict(showframe=False, showcoastlines=True),
    coloraxis_colorbar=dict(title='Revenue (£)'),
    title_x=0.5
)

fig.show()
```

[109]:
```
# Grouping by product description and calculating total revenue
top_products = df.groupby('Description')['Revenue'].sum().reset_index()

# Sorting to get the top 10 products by revenue
```

```
top_products = top_products.sort_values(by='Revenue', ascending=False).head(10)

# Setting a unique color for each bar using colormap
colors = plt.cm.get_cmap('tab10', 10)  # 'tab10' helps provide 10 visually␣
 ↪distinct colors

# Plotting the bar chart
plt.figure(figsize=(12, 6))
bars = plt.bar(
    top_products['Description'],
    top_products['Revenue'],
    color=[colors(i) for i in range(10)]
)

# Customizing labels and appearances
plt.xticks(rotation=45, ha='right', fontsize=10)
plt.title("Top 10 Products by Revenue", fontsize=16, weight='bold')
plt.xlabel("Product", fontsize=12)
plt.ylabel("Revenue (£)", fontsize=12)
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Adding revenue labels on each bar for more readability and better evaluation
for bar in bars:
    height = bar.get_height()
    plt.text(
        bar.get_x() + bar.get_width() / 2.,
        height,
        f'{height:,.0f}',
        ha='center',
        va='bottom',
        fontsize=9
    )

plt.tight_layout()
plt.show()
```

```
C:\Users\Admin\AppData\Local\Temp\ipykernel_19400\2674381886.py:8:
MatplotlibDeprecationWarning:

The get_cmap function was deprecated in Matplotlib 3.7 and will be removed in
3.11. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap()``
or ``pyplot.get_cmap()`` instead.
```



[ ]:
```
### AI Assistance Disclosure ###

Parts of this report were guided and debugged with the help of AI tools like␣
 ↪ChatGPT and Julius to structure and clean the dataset professionally ␣
 ↪removing cancelled orders and generating Revenue column
generate examples of well commented Python codes for visualizations such as bar␣
 ↪charts and refine colors suggest appropriate models Exponential Smoothing␣
 ↪for forecasting
refactoring and formating code for readability and clarity

All AI generated code was reviewed, understood, modified, and integrated by me.
I ensured that the logic aligns with course concepts and is consistent with␣
 ↪best practices in machine learning and data analytics.

#References

#OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. https://chat.
 ↪openai.com/
```