forecastsforproductdemand

January 4, 2024

[3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.preprocessing import LabelEncoder
```

1 Preprocessing and data cleaning:

[4]:
```
df = pd.read_csv("C:\ComputerScience\ForecastsForProductDemand\Historical␣
↪Product Demand.csv")
df.head()
```

[4]:

	Product_Code	Warehouse	Product_Category	Date	Order_Demand
0	Product_0993	Whse_J	Category_028	2012/7/27	100
1	Product_0979	Whse_J	Category_028	2012/1/19	500
2	Product_0979	Whse_J	Category_028	2012/2/3	500
3	Product_0979	Whse_J	Category_028	2012/2/9	500
4	Product_0979	Whse_J	Category_028	2012/3/2	500

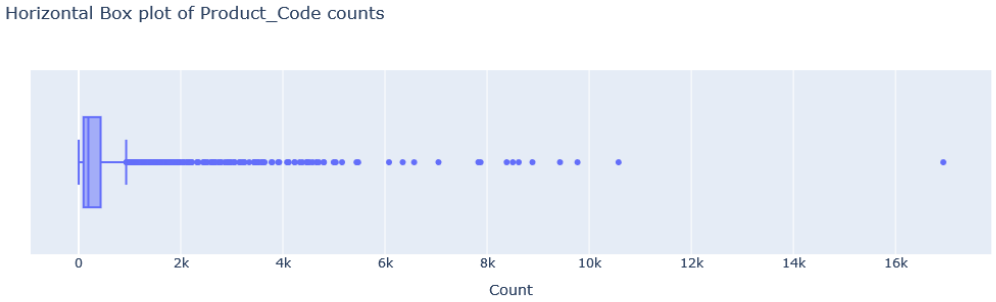
- Let's figure out how the product codes are distributed:

[5]:
```
product_code_counts = df["Product_Code"].value_counts()
print(product_code_counts.mean())
print(product_code_counts.min())
print(product_code_counts.max())
```

```
485.4513888888889
1
16936
```

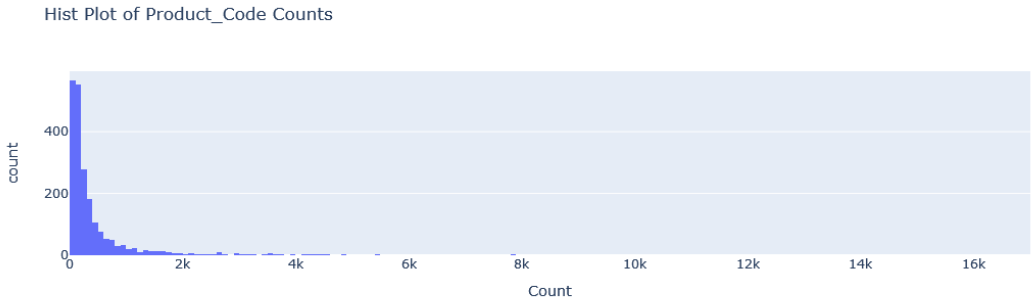
[6]:
```
import plotly.express as px

product_code_counts_df = product_code_counts.reset_index()
product_code_counts_df.columns = ['Product_Code', 'Count']
plt.figure(figsize=(12, 8))
fig = px.box(product_code_counts_df, x='Count',
```

```
            title='Horizontal Box plot of Product_Code counts',
            labels={'Count': 'Count', 'count': 'Frequency'})

fig.update_traces(hovertemplate='Count: %{x}<br>Frequency: %{y}')

fig.show()
```



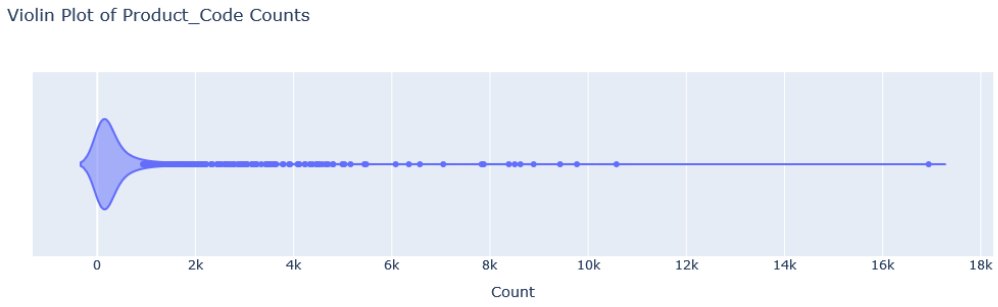

```
<Figure size 1200x800 with 0 Axes>
```

[7]:
```
plt.figure(figsize=(12, 8))

fig = px.histogram(product_code_counts_df, x='Count',
                   title='Hist Plot of Product_Code Counts',
                   labels={'Count': 'Count', 'count': 'Frequency'})

fig.update_traces(hovertemplate='Count: %{x}<br>Frequency: %{y}')
fig.show()
```




```
<Figure size 1200x800 with 0 Axes>
```

[8]:
```
violin_plot = px.violin(product_code_counts_df, x='Count',
                        title='Violin Plot of Product_Code Counts',
                        labels={'Count': 'Count', 'count': 'Frequency'})
violin_plot.show()
```



[9]:
```
product_code_counts_df[product_code_counts_df["Count"] < 99].count()
```

[9]:
```
Product_Code    562
Count           562
dtype: int64
```

[10]:
```
df.isnull().sum()
```

[10]:
```
Product_Code            0
Warehouse               0
Product_Category        0
Date                11239
Order_Demand            0
dtype: int64
```

[11]:
```
(df["Date"].isnull().sum() / df["Date"].size) * 100
```

[11]:
```
1.0718355863910547
```

Conclusion: It's just one percent of data, so we can replace them with sth and then if we see not coorelation we can delete the corresponding rows.

[12]:
```
df["Date"].fillna("0000/0/0" ,inplace=True)
```

[13]:
```
df.isnull().sum()
```

[13]:
```
Product_Code        0
Warehouse           0
Product_Category    0
Date                0
```

```
Order_Demand        0
dtype: int64
```

[14]:
```
df["Product_Code"].value_counts().mean()
```

[14]:
```
485.4513888888889
```

[19]:
```
X = df.drop(["Order_Demand"], axis = 1)
Y = pd.DataFrame(df["Order_Demand"])
```

[20]:
```
X.head()
```

[20]:

	Product_Code	Warehouse	Product_Category	Date
0	Product_0993	Whse_J	Category_028	2012/7/27
1	Product_0979	Whse_J	Category_028	2012/1/19
2	Product_0979	Whse_J	Category_028	2012/2/3
3	Product_0979	Whse_J	Category_028	2012/2/9
4	Product_0979	Whse_J	Category_028	2012/3/2

[21]:
```
Y.head()
```

[21]:

	Order_Demand
0	100
1	500
2	500
3	500
4	500

[22]:
```
df["Warehouse"].describe()
```

[22]:
```
count     1048575
unique          4
top         Whse_J
freq       764447
Name: Warehouse, dtype: object
```

[23]:
```
df["Product_Code"].describe()
```

[23]:
```
count          1048575
unique            2160
top       Product_1359
freq             16936
Name: Product_Code, dtype: object
```

[24]:
```
df["Product_Category"].describe()
```

[24]:
```
count          1048575
unique              33
```

```
top       Category_019
freq            481099
Name: Product_Category, dtype: object
```

[25]:
```
label_encoder = LabelEncoder()
X ["Product_Code"] = label_encoder.fit_transform(X["Product_Code"])

X = pd.get_dummies(X, columns = ["Warehouse" ,"Product_Category"], prefix =␣
 ↪[None, "Product"])
X.head()
```

[25]:

	Product_Code	Date	Whse_A	Whse_C	Whse_J	Whse_S	\
0	982	2012/7/27	False	False	True	False	
1	968	2012/1/19	False	False	True	False	
2	968	2012/2/3	False	False	True	False	
3	968	2012/2/9	False	False	True	False	
4	968	2012/3/2	False	False	True	False	

	Product_Category_001	Product_Category_002	Product_Category_003	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Product_Category_004	…	Product_Category_024	Product_Category_025	\
0	False	…	False	False	
1	False	…	False	False	
2	False	…	False	False	
3	False	…	False	False	
4	False	…	False	False	

	Product_Category_026	Product_Category_027	Product_Category_028	\
0	False	False	True	
1	False	False	True	
2	False	False	True	
3	False	False	True	
4	False	False	True	

	Product_Category_029	Product_Category_030	Product_Category_031	\
0	False	False	False	
1	False	False	False	
2	False	False	False	
3	False	False	False	
4	False	False	False	

	Product_Category_032	Product_Category_033

```
0                False                False
1                False                False
2                False                False
3                False                False
4                False                False

[5 rows x 39 columns]
```

[26]:
```
x_train, x_test, y_train, y_test = train_test_split(X,Y,
                                                    test_size = 0.2␣
 ↪,random_state =42)
```

[27]:
```
print(x_train.size)
print(x_test.size)
```

```
32715540
8178885
```

[ ]:

[ ]:

[ ]: