Introduction:

- **In the given dataset invoice details by product description and category is presented.**

Problem statement:

- **To Identify product category basis items.**

Solution approach:

- **Our goal is to identify or ascertain the product category.**

Importing Libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import numpy as np
```

Loading training data

```
data= pd.read_csv("/content/train_set.csv")
```

```
data.head()
```

	Inv_Id	Vendor_Code	GL_Code	Inv_Amt	Item_Description
0	15001	VENDOR-1676	GL-6100410	83.24	Artworking/Typesetting Production Jun 2009 Cha...
1	15002	VENDOR-1883	GL-2182000	51.18	Auto Leasing Corporate Services Corning Inc /N...
2	15004	VENDOR-1999	GL-6050100	79.02	Store Management Lease/Rent Deltona Corp Real ...
		VENDOR-	GL-		Store Construction General Requirements

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5288 entries, 0 to 5287
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Inv_Id           5288 non-null   int64
 1   Vendor_Code      5288 non-null   object
```

```
 2   GL_Code           5288 non-null   object
 3   Inv_Amt           5288 non-null   float64
 4   Item_Description  5288 non-null   object
 5   Product_Category  5288 non-null   object
dtypes: float64(1), int64(1), object(4)
memory usage: 248.0+ KB
```

```
data.shape
```

```
(5288, 6)
```

```
data.columns
```

```
Index(['Inv_Id', 'Vendor_Code', 'GL_Code', 'Inv_Amt', 'Item_Description',
       'Product_Category'],
      dtype='object')
```

```
data.dtypes
```

```
Inv_Id                int64
Vendor_Code          object
GL_Code              object
Inv_Amt             float64
Item_Description     object
Product_Category     object
dtype: object
```

EDA(Exploratory Data Analysis)

Checking the Null Values

```
data.isnull().sum()
```

```
Inv_Id              0
Vendor_Code         0
GL_Code             0
Inv_Amt             0
Item_Description    0
Product_Category    0
dtype: int64
```

Observation: There are no null values in the training data.

Summarizing the Data

```
data.describe()
```

	Inv_Id	Inv_Amt
count	5288.000000	5288.000000
mean	19016.049924	49.890034
std	2310.739549	28.835716
min	15001.000000	0.010000
25%	17013.750000	25.062500
50%	19023.000000	49.560000
75%	21004.250000	74.945000

Checking The Unique Values of features

```
data['Vendor_Code'].unique()
```

```
array(['VENDOR-1676', 'VENDOR-1883', 'VENDOR-1999', ..., 'VENDOR-1309',
       'VENDOR-1969', 'VENDOR-1691'], dtype=object)
```

```
data['GL_Code'].unique()
```

```
array(['GL-6100410', 'GL-2182000', 'GL-6050100', 'GL-6101400',
       'GL-6050310', 'GL-6060100', 'GL-6100500', 'GL-6121905',
       'GL-6020600'], dtype=object)
```

```
data['Product_Category'].unique()
```

```
array(['CLASS-1963', 'CLASS-1250', 'CLASS-1274', 'CLASS-1522',
       'CLASS-1376', 'CLASS-1758', 'CLASS-2141', 'CLASS-1429',
       'CLASS-1652', 'CLASS-1249', 'CLASS-1721', 'CLASS-1870',
       'CLASS-1828', 'CLASS-2112', 'CLASS-1567', 'CLASS-1309',
       'CLASS-1477', 'CLASS-1805', 'CLASS-1919', 'CLASS-1322',
       'CLASS-1838', 'CLASS-1850', 'CLASS-2003', 'CLASS-1248',
       'CLASS-1964', 'CLASS-2241', 'CLASS-1867', 'CLASS-1983',
       'CLASS-1294', 'CLASS-1688', 'CLASS-2038', 'CLASS-1770',
       'CLASS-2152', 'CLASS-2146', 'CLASS-1957', 'CLASS-2015'],
      dtype=object)
```

Checking The Count of no. of values features

```
data['Product_Category'].value_counts()
```

```
CLASS-1758    1421
CLASS-1274     939
CLASS-1522     803
CLASS-1250     440
CLASS-1376     347
CLASS-1963     215
CLASS-1249     167
CLASS-1828     107
CLASS-2141     103
CLASS-1721     103
```

```
CLASS-1567      80
CLASS-1919      61
CLASS-2112      52
CLASS-1850      51
CLASS-1477      48
CLASS-2241      36
CLASS-1870      35
CLASS-1309      31
CLASS-2003      31
CLASS-1429      30
CLASS-1322      28
CLASS-1964      27
CLASS-1294      24
CLASS-1770      19
CLASS-1983      16
CLASS-1867      15
CLASS-1652      14
CLASS-2038      13
CLASS-1805      10
CLASS-2152       9
CLASS-1248       4
CLASS-1688       4
CLASS-2146       2
CLASS-1838       1
CLASS-1957       1
CLASS-2015       1
Name: Product_Category, dtype: int64
```

```
data['GL_Code'].value_counts()
```

```
GL-6050310    1536
GL-2182000    1248
GL-6050100     916
GL-6101400     759
GL-6100410     365
GL-6100500     204
GL-6060100     149
GL-6121905      80
GL-6020600      31
Name: GL_Code, dtype: int64
```

```
data['Vendor_Code'].value_counts()
```

```
VENDOR-1883    322
VENDOR-1513    238
VENDOR-1944    170
VENDOR-1551    170
VENDOR-2513    158
               ...
VENDOR-1815      1
VENDOR-1395      1
VENDOR-2321      1
VENDOR-2322      1
VENDOR-1691      1
Name: Vendor_Code, Length: 1206, dtype: int64
```

Analysis

- Objective of the below analysis is to ascertain the invoice details as in quantity and valuation basis **product category**. This analysis helps in tracking the invoice amount and the number of invoices.

```
prod= data.groupby(['Product_Category']).agg({'Inv_Id':'count','Inv_Amt':'sum'})
prod.sort_values(by='Inv_Amt',ascending=False)
```

Product_Category	Inv_Id	Inv_Amt
CLASS-1758	1421	71136.15
CLASS-1274	939	47153.57
CLASS-1522	803	38624.53
CLASS-1250	440	22902.55
CLASS-1376	347	17814.97
CLASS-1963	215	10231.40
CLASS-1249	167	8688.27
CLASS-1828	107	5638.99
CLASS-2141	103	5021.15
CLASS-1721	103	4599.74
CLASS-2112	52	2841.73

- This analysis provides details for item description which falls under each product category and the invoice amount pertaining to respective items.

```
prod_desc = data.groupby(['Product_Category','Item_Description']).agg({'Inv_Amt':'sum'})
prod_desc
```

Product_Category	Item_Description	Inv_Amt
CLASS-1248	**2001-Apr Avalon Corp Audit & Risk Consulting Finance Consulting Corporate Services Consulting**	43.76
	Audit & Risk Consulting Corporate Services Avalon Corp 2019Feb Finance Consulting Consulting	36.35
	Combined Insurance Co Of America Dec2011 Corporate Services Consulting Finance Consulting Audit & Risk Consulting	78.84
	Consulting Finance Consulting Audit & Risk Consulting Carter Day Industries Inc May 2007 Corporate Services	14.78
CLASS-1249	**2000-Feb Daly John J Auto Fleet Repair and Maintenance Auto Leasing and Maintenance Corporate Services Other Corporate Services**	7.76
...	**...**	...
CLASS-2241	**Workmen's Insurance Corporate Services Asarco Inc Oct-2013 Workmen's Insurance Commercial Insurance**	92.75
	Workmen's Insurance Corporate Services Bates Charles Howard	41.27

In the below analysis Vendor code is classified basis Product category and the total invoice amount for each is available.

- This analysis not only helps to identify multiple vendors for each product category but also helps in deriving information about the invoice amount for each vendor category by products (we can further refer to the item description for each product category to do in-depth analysis).

```
vendor_data = data.groupby(['Product_Category','Vendor_Code']).agg({'Inv_Amt':'sum'})
vendor_data.sort_values(by='Inv_Amt', ascending=False)
```

		Inv_Amt
Product_Category	**Vendor_Code**	
CLASS-1250	**VENDOR-1883**	16292.98
CLASS-1758	**VENDOR-1513**	11497.44
	VENDOR-2513	7563.67
CLASS-1250	**VENDOR-1551**	6512.19
CLASS-1249	**VENDOR-1944**	6380.70
...	**...**	...
CLASS-1477	**VENDOR-1469**	0.99
CLASS-1274	**VENDOR-1589**	0.55
	VENDOR-1996	0.41
CLASS-2003	**VENDOR-1936**	0.40
CLASS-1828	**VENDOR-1958**	0.39

1284 rows × 1 columns

The below analysis provides information by GL_Code and the invoice amount falling under each.

- This can be used to track the type of transaction amount as per GL_Code description. The business can derive insights on its payables or recievables as the case may be.
- The visualizaion of the information is presented in a bar graph form to have more clarity and insight.

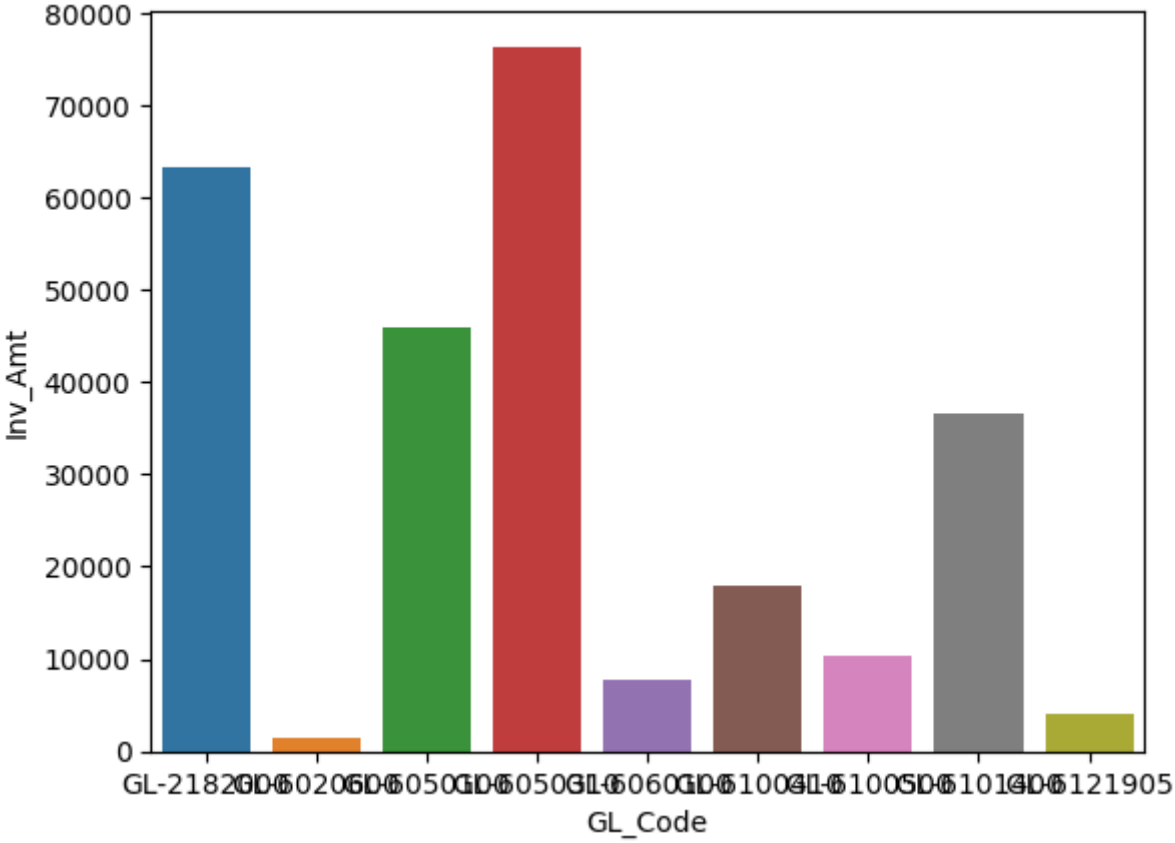
```
trans_analytics = data.groupby(['GL_Code']).agg({'Inv_Amt':'sum'})
trans_analytics
```

GL_Code	Inv_Amt
GL-2182000	63401.57
GL-6020600	1441.95
GL-6050100	45909.73
GL-6050310	76446.32
GL-6060100	7708.36
GL-6100410	17848.06
GL-6100500	10393.48
GL-6101400	36696.22

Visualization

```
sns.barplot(x=trans_analytics.index, y='Inv_Amt', data=trans_analytics)
```

```
<Axes: xlabel='GL_Code', ylabel='Inv_Amt'>
```

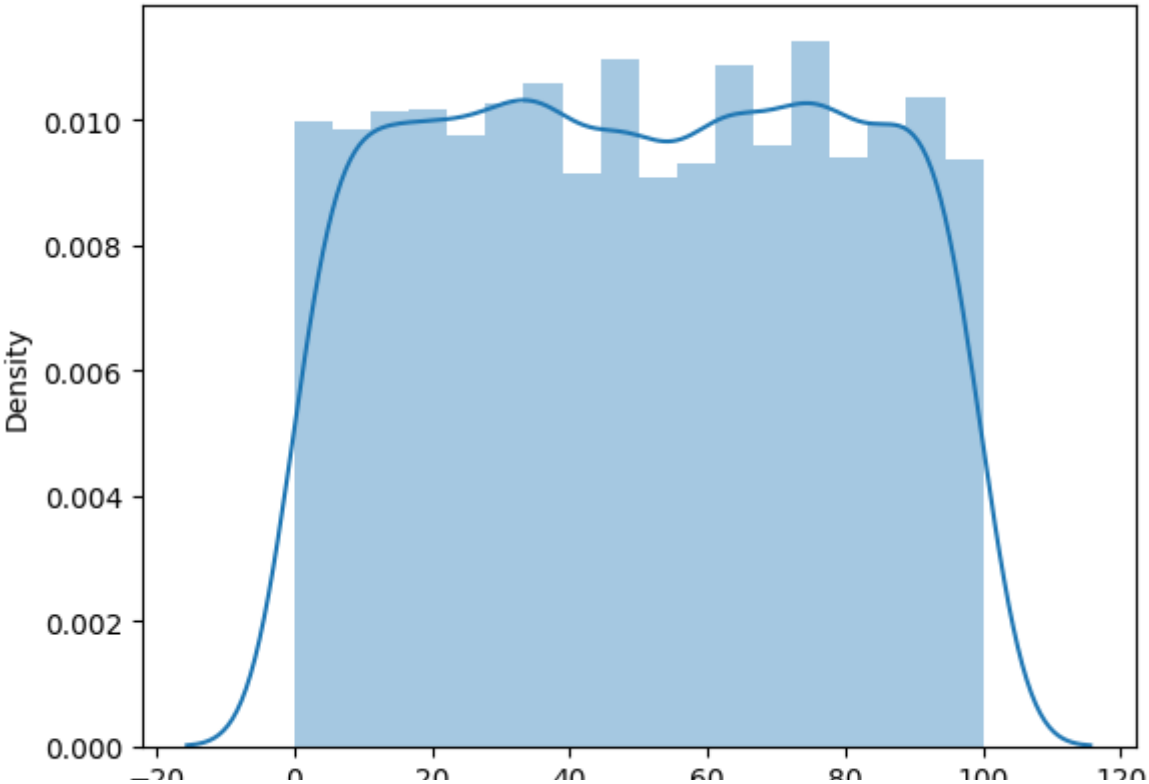


Visualizing the Data

```
sns.distplot(data['Inv_Amt']);
```

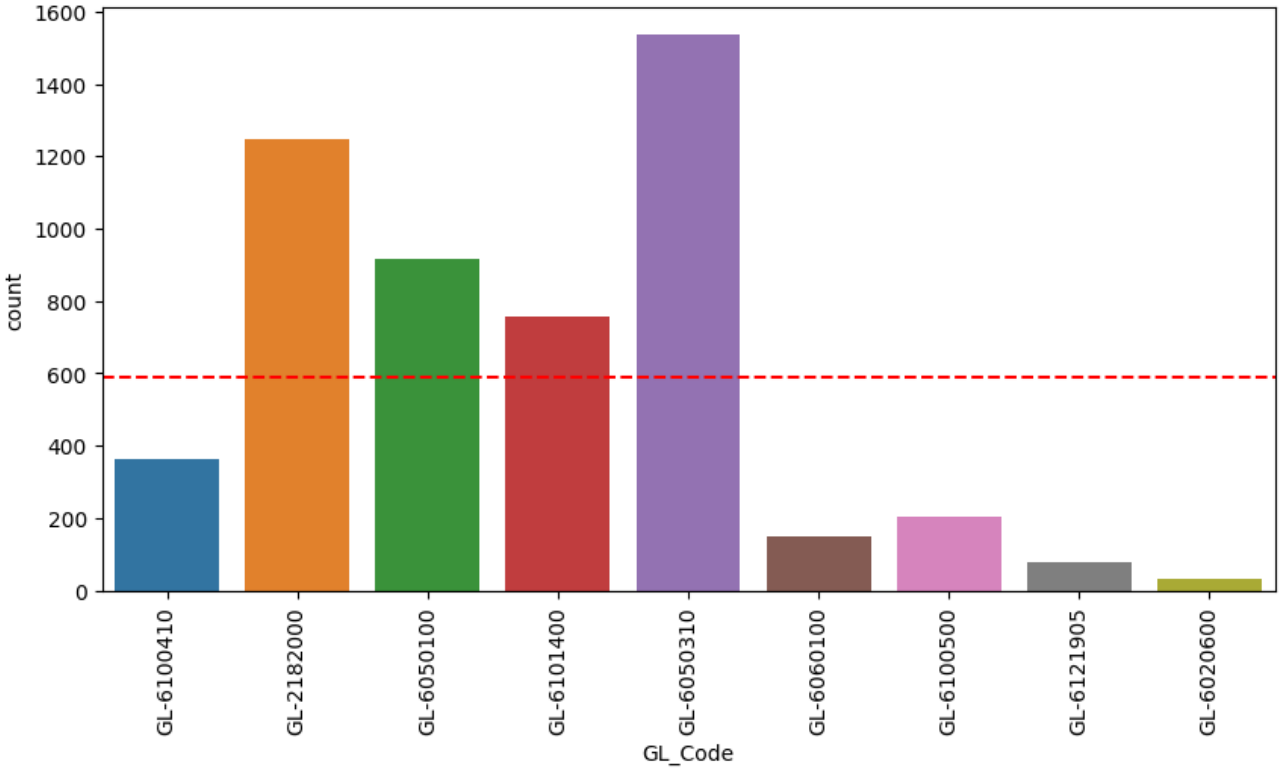
```
<ipython-input-21-32b18f181f22>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
sns.distplot(data['Inv_Amt']);
```



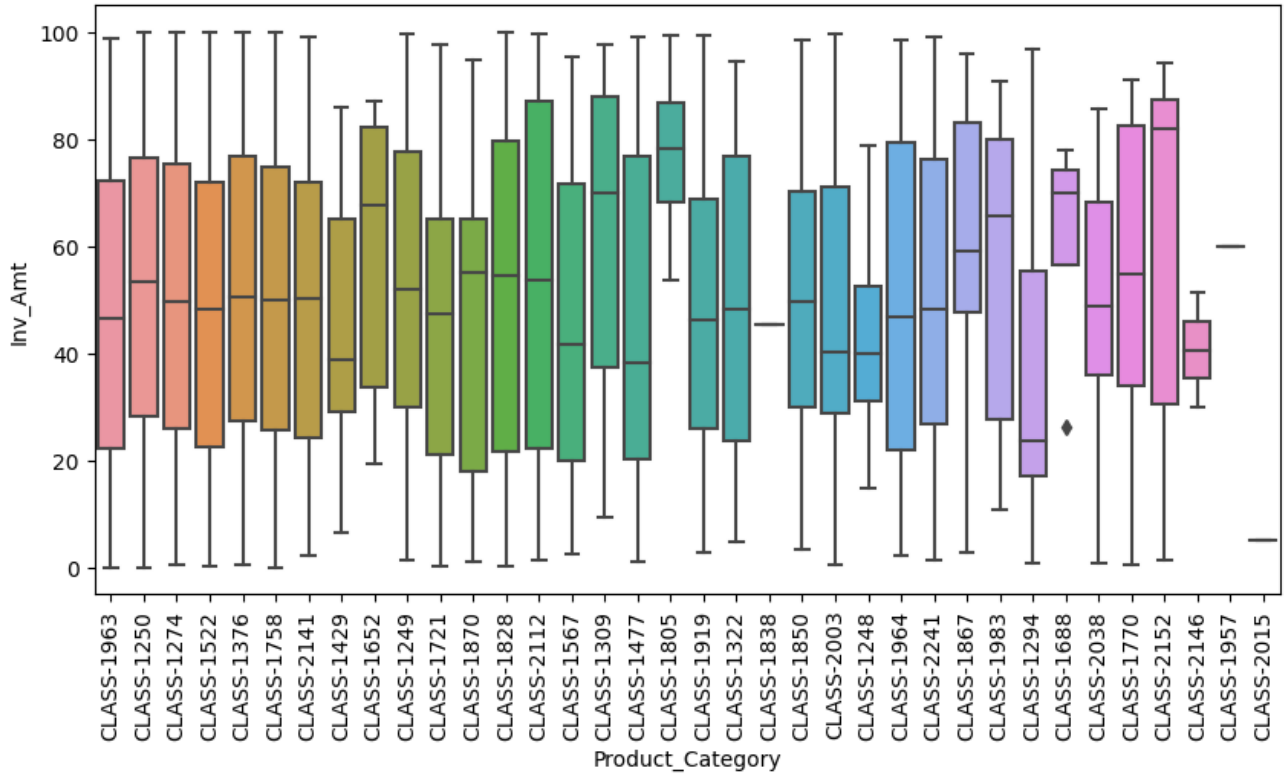

```
plt.figure(figsize=(15,5))
sns.countplot(x='Product_Category',data=data)
plt.axhline(y=data['Product_Category'].value_counts().mean(),c='r',linestyle='--')
plt.xticks(rotation=90);
```


```
plt.figure(figsize=(10,5))
sns.countplot(x='GL_Code',data=data)
plt.axhline(y=data['GL_Code'].value_counts().mean(),c='r',linestyle='--')
plt.xticks(rotation=90);
```



Checking the outliers

```
plt.figure(figsize=(10,5))
sns.boxplot( x="Product_Category", y='Inv_Amt', data=data)
plt.xticks(rotation=90);
```



Observation: There are no outliers in the above.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5288 entries, 0 to 5287
Data columns (total 6 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Inv_Id            5288 non-null   int64
 1   Vendor_Code       5288 non-null   object
 2   GL_Code           5288 non-null   object
 3   Inv_Amt           5288 non-null   float64
 4   Item_Description  5288 non-null   object
 5   Product_Category  5288 non-null   object
dtypes: float64(1), int64(1), object(4)
memory usage: 248.0+ KB
```

Data Preprocessing

Text Preprocessing

After analysing each feature the most suitable feature identified is " **Item_description**" which can be used to classify the **Product_category**.

- Remove unwanted characters, numbers and symbols
- Convert to lowercase
- Remove stopwords
- Lemmatization

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
nltk.download('stopwords')
nltk.download('wordnet')
# Remove unwanted characters, numbers and symbols
data['processed_text'] = data['Item_Description'].map(lambda x: re.sub('[^a-zA-Z]', ' ', x
# Convert to lowercase
data['processed_text'] = data['processed_text'].map(lambda x: x.lower())
# Remove stopwords
stop_words = set(stopwords.words('english'))
data['processed_text'] = data['processed_text'].apply(lambda x: ' '.join([word for word in
# Lemmatization
lemmatizer = WordNetLemmatizer()
data['processed_text'] = data['processed_text'].apply(lambda x: ' '.join([lemmatizer.lemma
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
data.columns
```

```
Index(['Inv_Id', 'Vendor_Code', 'GL_Code', 'Inv_Amt', 'Item_Description',
       'Product_Category', 'processed_text'],
      dtype='object')
```

```
data["Item_Description"]=data['processed_text']
data["Item_Description"].head()
```

```
0    artworking typesetting production jun champion...
1    auto leasing corporate service corning inc ny ...
2    store management lease rent deltona corp real ...
3    store construction general requirement colonia...
4    jul aydin corp contingent labor temp labor con...
Name: Item_Description, dtype: object
```

```
!pip install wordcloud
from wordcloud import WordCloud
import matplotlib.pyplot as plt
# Join all preprocessed text data into a single string
text = ' '.join(data['Item_Description'].tolist())
# Generate word cloud
wordcloud = WordCloud(width=800, height=800, background_color='white', stopwords=set(stopw
# Plot word cloud
```

```
plt.figure(figsize=(8, 8), facecolor=None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
Requirement already satisfied: wordcloud in /usr/local/lib/python3.9/dist-packages (1
Requirement already satisfied: pillow in /usr/local/lib/python3.9/dist-packages (from
Requirement already satisfied: matplotlib in /usr/local/lib/python3.9/dist-packages (
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.9/dist-
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.9/dist-pack
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-packages
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.9/dist-pac
```

Feature Engineering

```
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.9/dist-pac
```

```
gl_codes = data['GL_Code'].tolist()
gl_codes_without_prefix = [code[3:] for code in gl_codes]
print(gl_codes_without_prefix)
```

```
['6100410', '2182000', '6050100', '6101400', '2182000', '6101400', '6050310', '610146
```

```
df = pd.DataFrame({'GL_Code':gl_codes_without_prefix })
data["GL_Code"]=df
```

```
data["GL_Code"].head()
```

```
0    6100410
1    2182000
2    6050100
3    6101400
4    2182000
Name: GL_Code, dtype: object
```

```
ven_codes = data['Vendor_Code'].tolist()
ven_codes_without_prefix = [code[7:] for code in ven_codes]
print(ven_codes_without_prefix)
df = pd.DataFrame({'Vendor_Code':ven_codes_without_prefix })
data["Vendor_Code"]=df
data["Vendor_Code"].head()
```

```
['1676', '1883', '1999', '1771', '1331', '2076', '1802', '1191', '2120', '1704', '251
0    1676
1    1883
2    1999
3    1771
4    1331
Name: Vendor_Code, dtype: object
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5288 entries, 0 to 5287
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype
```

```
---  ------            --------------  -----
 0   Inv_Id            5288 non-null   int64
 1   Vendor_Code       5288 non-null   object
 2   GL_Code           5288 non-null   object
 3   Inv_Amt           5288 non-null   float64
 4   Item_Description  5288 non-null   object
 5   Product_Category  5288 non-null   object
 6   processed_text    5288 non-null   object
dtypes: float64(1), int64(1), object(5)
memory usage: 289.3+ KB
```

```
data = data.drop('processed_text', axis=1)
data = data.drop('Inv_Id', axis=1)
```

Feature Transformation

Encoding

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
data["Vendor_Code"] = le.fit_transform(data["Vendor_Code"])
le1 = LabelEncoder()
data["GL_Code"] = le1.fit_transform(data["GL_Code"])
```

```
data.head()
```

	Vendor_Code	GL_Code	Inv_Amt	Item_Description	Product_Category
0	527	5	83.24	artworking typesetting production jun champion...	CLASS-1963
1	686	0	51.18	auto leasing corporate service corning inc ny ...	CLASS-1250
2	777	2	79.02	store management lease rent deltona corp real ...	CLASS-1274
3	[illegible]	7	[illegible]	store construction general requirement	[illegible]

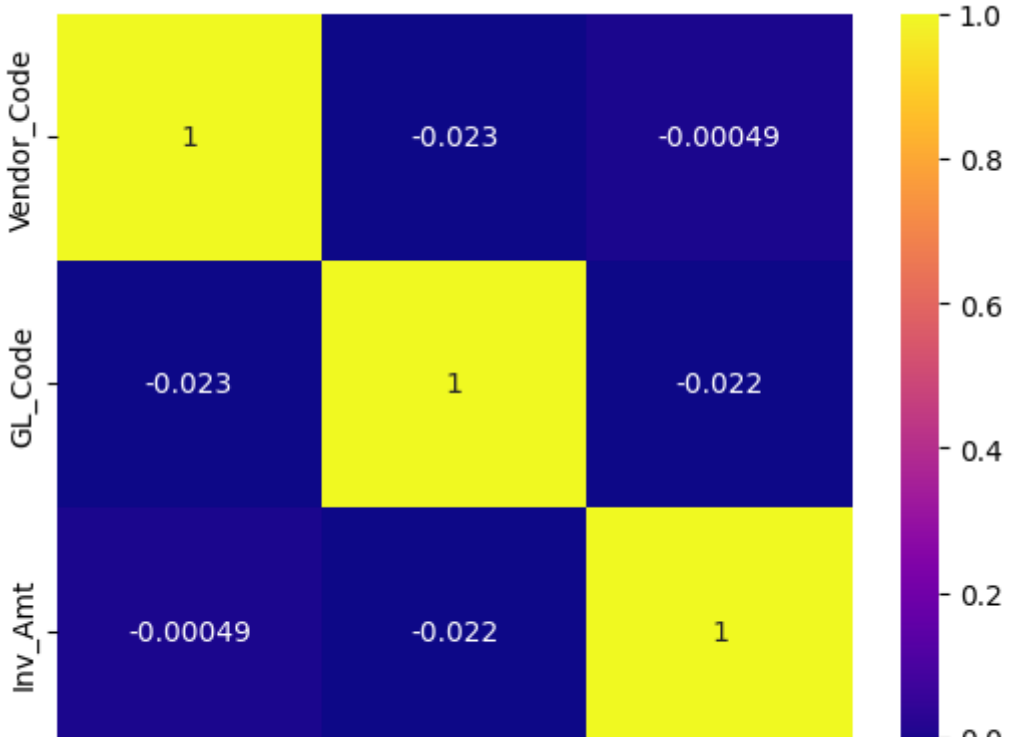
Correlation Analysis

```
corr =data.corr()
```

```
<ipython-input-38-87b0fef96621>:1: FutureWarning: The default value of numeric_only i
  corr =data.corr()
```

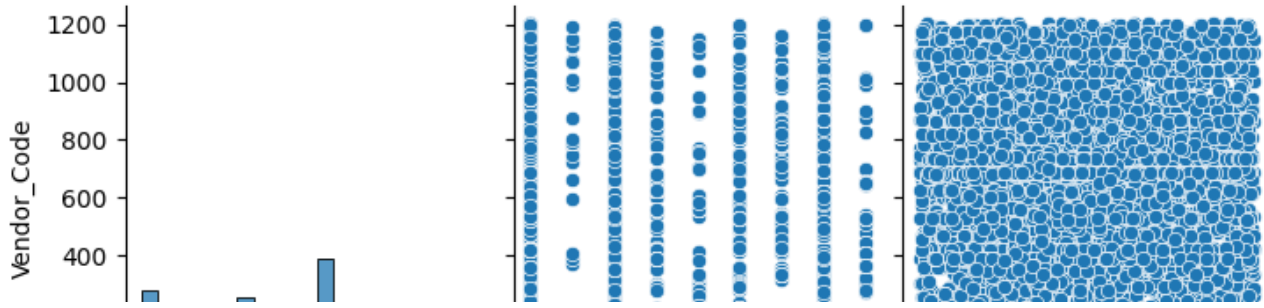
```
sns.heatmap(corr, annot=True, cmap='plasma')
```

```
<Axes: >
```




```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7ff76328cdf0>
```



Feature Selection

```
features = ['Item_Description']
X = data[features]
y = data['Product_Category']
```

Splitting the given data into training and testing data

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
X_train_cv = cv.fit_transform(X_train['Item_Description'])
X_test_cv = cv.transform(X_test['Item_Description'])
```

ML Model

Model Evaluation

Naive Bayes Classifier

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
nb = MultinomialNB()
nb.fit(X_train_cv, y_train)
y_pred = nb.predict(X_test_cv)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy: {:.2%}'.format(accuracy))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 98.68%
              precision    recall  f1-score   support

   CLASS-1248       1.00      1.00      1.00         1
   CLASS-1249       1.00      0.97      0.99        38
   CLASS-1250       0.99      1.00      0.99        81
   CLASS-1274       1.00      1.00      1.00       198
   CLASS-1294       1.00      1.00      1.00         4
```

```
     CLASS-1309       1.00      0.17      0.29         6
     CLASS-1322       1.00      1.00      1.00        12
     CLASS-1376       1.00      1.00      1.00        73
     CLASS-1429       1.00      1.00      1.00         6
     CLASS-1477       1.00      1.00      1.00        10
     CLASS-1522       1.00      1.00      1.00       142
     CLASS-1567       1.00      1.00      1.00        16
     CLASS-1652       1.00      1.00      1.00         3
     CLASS-1721       0.63      1.00      0.78        19
     CLASS-1758       1.00      1.00      1.00       303
     CLASS-1770       1.00      1.00      1.00         4
     CLASS-1805       0.00      0.00      0.00         3
     CLASS-1828       0.89      1.00      0.94        16
     CLASS-1850       1.00      1.00      1.00         9
     CLASS-1867       1.00      1.00      1.00         5
     CLASS-1870       1.00      1.00      1.00         6
     CLASS-1919       1.00      1.00      1.00        10
     CLASS-1963       1.00      1.00      1.00        40
     CLASS-1964       1.00      1.00      1.00         4
     CLASS-1983       1.00      1.00      1.00         2
     CLASS-2003       1.00      1.00      1.00         8
     CLASS-2038       0.00      0.00      0.00         2
     CLASS-2112       1.00      1.00      1.00        14
     CLASS-2141       1.00      1.00      1.00        14
     CLASS-2152       0.00      0.00      0.00         3
     CLASS-2241       1.00      1.00      1.00         6

       accuracy                           0.99      1058
      macro avg       0.89      0.88      0.87      1058
   weighted avg       0.98      0.99      0.98      1058

/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.9/dist-packages/sklearn/metrics/_classification.py:1344: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```

```
cm = confusion_matrix(y_test, y_pred)
cm
```

```
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,  16,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   9,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   5,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   6,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,  10,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,  40,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   4,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   2,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   8,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   2,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,  14,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,  14,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   3,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   6]])
```

Model Tuning

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
lr_model = LogisticRegression(max_iter=1000)
lr_model.fit(X_train_cv, y_train)
y_pred = lr_model.predict(X_test_cv)
accuracy = accuracy_score(y_test, y_pred)
print('Accuracy: {:.2%}'.format(accuracy))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 100.00%
              precision    recall  f1-score   support

  CLASS-1248       1.00      1.00      1.00         1
  CLASS-1249       1.00      1.00      1.00        38
  CLASS-1250       1.00      1.00      1.00        81
  CLASS-1274       1.00      1.00      1.00       198
  CLASS-1294       1.00      1.00      1.00         4
  CLASS-1309       1.00      1.00      1.00         6
  CLASS-1322       1.00      1.00      1.00        12
  CLASS-1376       1.00      1.00      1.00        73
  CLASS-1429       1.00      1.00      1.00         6
  CLASS-1477       1.00      1.00      1.00        10
  CLASS-1522       1.00      1.00      1.00       142
  CLASS-1567       1.00      1.00      1.00        16
  CLASS-1652       1.00      1.00      1.00         3
  CLASS-1721       1.00      1.00      1.00        19
  CLASS-1758       1.00      1.00      1.00       303
  CLASS-1770       1.00      1.00      1.00         4
  CLASS-1805       1.00      1.00      1.00         3
  CLASS-1828       1.00      1.00      1.00        16
  CLASS-1850       1.00      1.00      1.00         9
  CLASS-1867       1.00      1.00      1.00         5
  CLASS-1870       1.00      1.00      1.00         6
  CLASS-1919       1.00      1.00      1.00        10
  CLASS-1963       1.00      1.00      1.00        40
  CLASS-1964       1.00      1.00      1.00         4
  CLASS-1983       1.00      1.00      1.00         2
  CLASS-2003       1.00      1.00      1.00         8
  CLASS-2038       1.00      1.00      1.00         2
  CLASS-2112       1.00      1.00      1.00        14
  CLASS-2141       1.00      1.00      1.00        14
  CLASS-2152       1.00      1.00      1.00         3
  CLASS-2241       1.00      1.00      1.00         6

     accuracy                           1.00      1058
    macro avg       1.00      1.00      1.00      1058
weighted avg       1.00      1.00      1.00      1058
```

```
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[  1,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,  38,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,  81,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0, 198,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   4,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
```

```
[  0,   0,   0,   0,   0,   6,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,  12,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,  73,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   6,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,  10,   0,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 142,   0,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  16,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  19,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0, 303,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   4,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   3,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,  16,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,   9,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0],
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train_cv, y_train)
y_pred = clf.predict(X_test_cv)
accuracy = clf.score(X_test_cv, y_test)
print('Accuracy: {:.2%}'.format(accuracy))
print(classification_report(y_test, y_pred))
```

```
Accuracy: 100.00%
              precision    recall  f1-score   support
```

```
     CLASS-1248       1.00      1.00      1.00         1
     CLASS-1249       1.00      1.00      1.00        38
     CLASS-1250       1.00      1.00      1.00        81
     CLASS-1274       1.00      1.00      1.00       198
     CLASS-1294       1.00      1.00      1.00         4
     CLASS-1309       1.00      1.00      1.00         6
     CLASS-1322       1.00      1.00      1.00        12
     CLASS-1376       1.00      1.00      1.00        73
     CLASS-1429       1.00      1.00      1.00         6
     CLASS-1477       1.00      1.00      1.00        10
     CLASS-1522       1.00      1.00      1.00       142
     CLASS-1567       1.00      1.00      1.00        16
     CLASS-1652       1.00      1.00      1.00         3
     CLASS-1721       1.00      1.00      1.00        19
     CLASS-1758       1.00      1.00      1.00       303
     CLASS-1770       1.00      1.00      1.00         4
     CLASS-1805       1.00      1.00      1.00         3
     CLASS-1828       1.00      1.00      1.00        16
     CLASS-1850       1.00      1.00      1.00         9
     CLASS-1867       1.00      1.00      1.00         5
     CLASS-1870       1.00      1.00      1.00         6
     CLASS-1919       1.00      1.00      1.00        10
     CLASS-1963       1.00      1.00      1.00        40
     CLASS-1964       1.00      1.00      1.00         4
     CLASS-1983       1.00      1.00      1.00         2
     CLASS-2003       1.00      1.00      1.00         8
     CLASS-2038       1.00      1.00      1.00         2
     CLASS-2112       1.00      1.00      1.00        14
     CLASS-2141       1.00      1.00      1.00        14
     CLASS-2152       1.00      1.00      1.00         3
     CLASS-2241       1.00      1.00      1.00         6

       accuracy                           1.00      1058
      macro avg       1.00      1.00      1.00      1058
   weighted avg       1.00      1.00      1.00      1058
```

```
cm = confusion_matrix(y_test, y_pred)
cm
```

```
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   9,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   5,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   6,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,  10,   0,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,  40,   0,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   4,   0,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   2,   0,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   8,
          0,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          2,   0,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,  14,   0,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,  14,   0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   3,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   6]])
```

Loading Test Data

```
test_data = pd.read_csv("/content/test_set.csv")
```

EDA(Exploratory Data Analysis)

```
test_data.shape
```

```
(278, 5)
```

```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 278 entries, 0 to 277
Data columns (total 5 columns):
```

```
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Inv_Id            278 non-null    int64
 1   Vendor_Code       278 non-null    object
 2   GL_Code           278 non-null    object
 3   Inv_Amt           278 non-null    float64
 4   Item_Description  278 non-null    object
dtypes: float64(1), int64(1), object(3)
memory usage: 11.0+ KB
```

```
test_data.head()
```

	Inv_Id	Vendor_Code	GL_Code	Inv_Amt	Item_Description
0	15041	VENDOR-1181	GL-6050100	88.80	Base Rent Store Management Aig Equity Sales Co...
1	15094	VENDOR-1554	GL-6100410	80.08	Prototype and Comp Production/Packaging Design...
2	15112	VENDOR-1513	GL-6050310	24.23	Ground Transportation Travel and Entertainment...
[illegible]	[illegible]	VENDOR-	GL-	[illegible]	Store Construction General Requirements

```
test_data.dtypes
```

```
Inv_Id                int64
Vendor_Code          object
GL_Code              object
Inv_Amt             float64
Item_Description     object
dtype: object
```

```
test_data.columns
```

```
Index(['Inv_Id', 'Vendor_Code', 'GL_Code', 'Inv_Amt', 'Item_Description'],
dtype='object')
```

Checking the null values

```
test_data.isnull().sum()
```

```
Inv_Id              0
Vendor_Code         0
GL_Code             0
Inv_Amt             0
Item_Description    0
dtype: int64
```

Observation: There are no null values in the testing data.

Summarizing tha data

```
test_data.describe()
```

	Inv_Id	Inv_Amt
count	278.000000	278.000000
mean	18910.679856	51.694317
std	2228.205945	30.157571
min	15041.000000	0.160000
25%	16934.500000	24.000000
50%	18830.500000	52.470000
75%	20767.750000	79.380000
max	23012.000000	99.890000

Checking the unique values of the features

```
test_data['Vendor_Code'].unique()
```

```
array(['VENDOR-1181', 'VENDOR-1554', 'VENDOR-1513', 'VENDOR-1044',
       'VENDOR-1114', 'VENDOR-1406', 'VENDOR-1883', 'VENDOR-1640',
       'VENDOR-1509', 'VENDOR-2229', 'VENDOR-1046', 'VENDOR-1069',
       'VENDOR-1338', 'VENDOR-2287', 'VENDOR-1065', 'VENDOR-2008',
       'VENDOR-2513', 'VENDOR-1199', 'VENDOR-1066', 'VENDOR-1676',
       'VENDOR-1471', 'VENDOR-1551', 'VENDOR-2034', 'VENDOR-2408',
       'VENDOR-1326', 'VENDOR-1802', 'VENDOR-1425', 'VENDOR-1312',
       'VENDOR-1690', 'VENDOR-2480', 'VENDOR-1955', 'VENDOR-1451',
       'VENDOR-1793', 'VENDOR-1019', 'VENDOR-2294', 'VENDOR-1935',
       'VENDOR-1094', 'VENDOR-2116', 'VENDOR-1651', 'VENDOR-2344',
       'VENDOR-2334', 'VENDOR-1235', 'VENDOR-1963', 'VENDOR-1119',
       'VENDOR-2012', 'VENDOR-2278', 'VENDOR-2130', 'VENDOR-2460',
       'VENDOR-2117', 'VENDOR-2416', 'VENDOR-1771', 'VENDOR-1064',
       'VENDOR-2333', 'VENDOR-1944', 'VENDOR-1191', 'VENDOR-2485',
       'VENDOR-1117', 'VENDOR-1608', 'VENDOR-2279', 'VENDOR-1151',
       'VENDOR-2447', 'VENDOR-1717', 'VENDOR-1200', 'VENDOR-1080',
       'VENDOR-1873', 'VENDOR-2465', 'VENDOR-2187', 'VENDOR-1241',
       'VENDOR-1062', 'VENDOR-1150', 'VENDOR-1036', 'VENDOR-2051',
       'VENDOR-2365', 'VENDOR-2360', 'VENDOR-2552', 'VENDOR-1982',
       'VENDOR-2220', 'VENDOR-2247', 'VENDOR-2427', 'VENDOR-1866',
       'VENDOR-1215', 'VENDOR-1769', 'VENDOR-1339', 'VENDOR-2418',
       'VENDOR-1122', 'VENDOR-1174', 'VENDOR-2155', 'VENDOR-1575',
       'VENDOR-1256', 'VENDOR-2047', 'VENDOR-1488', 'VENDOR-1867',
       'VENDOR-1013', 'VENDOR-2160', 'VENDOR-1459', 'VENDOR-2479',
       'VENDOR-2550', 'VENDOR-2514', 'VENDOR-2110', 'VENDOR-1357',
       'VENDOR-1931', 'VENDOR-1430', 'VENDOR-1330', 'VENDOR-2032',
       'VENDOR-2146', 'VENDOR-1254', 'VENDOR-1813', 'VENDOR-1424',
       'VENDOR-1300', 'VENDOR-2549', 'VENDOR-1693', 'VENDOR-1934',
       'VENDOR-1043', 'VENDOR-1669', 'VENDOR-1081', 'VENDOR-1448',
       'VENDOR-1629', 'VENDOR-2111', 'VENDOR-1115', 'VENDOR-2174',
       'VENDOR-2478', 'VENDOR-1859', 'VENDOR-1317', 'VENDOR-1164',
       'VENDOR-2458', 'VENDOR-1409', 'VENDOR-1948', 'VENDOR-1623',
       'VENDOR-1555', 'VENDOR-1592', 'VENDOR-2343', 'VENDOR-1404',
```

```
       'VENDOR-1465', 'VENDOR-2120', 'VENDOR-2410', 'VENDOR-1462',
       'VENDOR-1301', 'VENDOR-1700', 'VENDOR-1682', 'VENDOR-1680',
       'VENDOR-1811', 'VENDOR-1469', 'VENDOR-2457', 'VENDOR-1841',
       'VENDOR-1153', 'VENDOR-1102', 'VENDOR-1940', 'VENDOR-2101',
       'VENDOR-2388', 'VENDOR-2318', 'VENDOR-1104', 'VENDOR-1993',
       'VENDOR-1792', 'VENDOR-2242', 'VENDOR-2387', 'VENDOR-1011',
       'VENDOR-2018', 'VENDOR-2301', 'VENDOR-1061', 'VENDOR-2466',
       'VENDOR-1926', 'VENDOR-1086', 'VENDOR-1132', 'VENDOR-2140'],
      dtype=object)
```

```
test_data['GL_Code'].unique()
```

```
array(['GL-6050100', 'GL-6100410', 'GL-6050310', 'GL-6101400',
       'GL-6020600', 'GL-2182000', 'GL-6060100', 'GL-6100500',
       'GL-6121905'], dtype=object)
```

Checking the count of no. of values of features

```
test_data['Vendor_Code'].value_counts()
```

```
VENDOR-1513    15
VENDOR-2513    12
VENDOR-1944     9
VENDOR-1802     8
VENDOR-1883     8
               ..
VENDOR-1241     1
VENDOR-2187     1
VENDOR-2465     1
VENDOR-1200     1
VENDOR-2140     1
Name: Vendor_Code, Length: 164, dtype: int64
```

```
test_data['GL_Code'].value_counts()
```

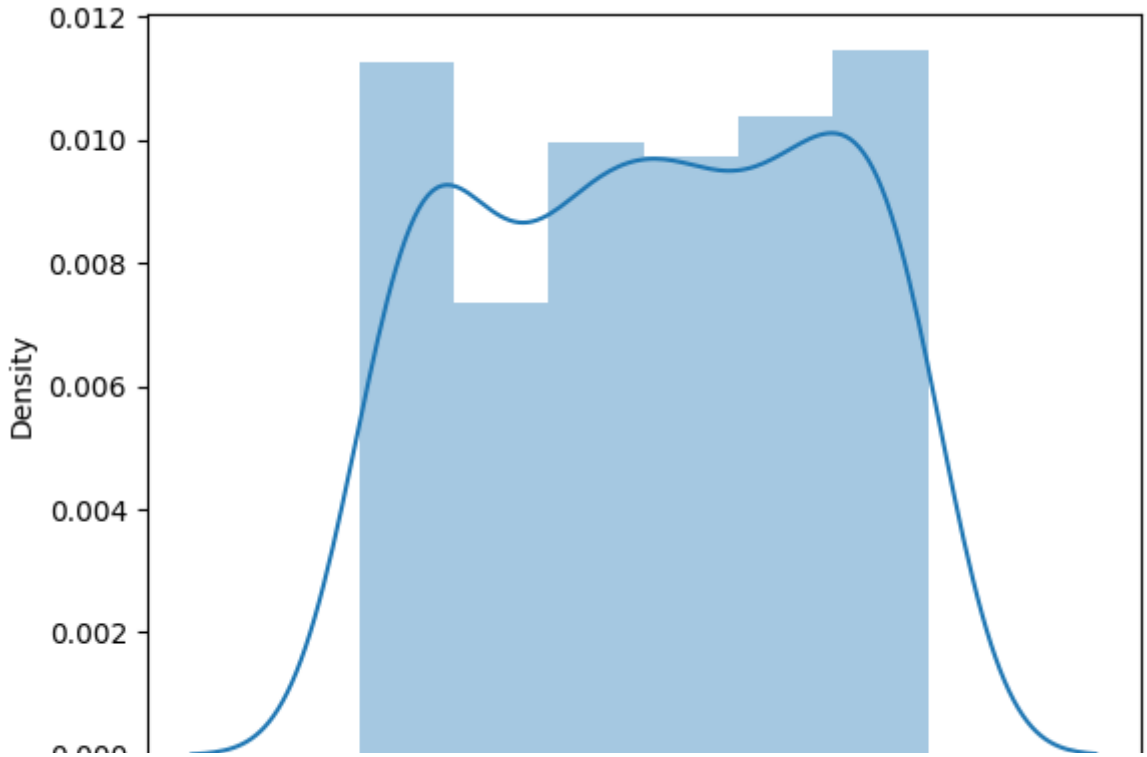
```
GL-6050310    82
GL-2182000    49
GL-6050100    46
GL-6101400    43
GL-6100410    23
GL-6100500    16
GL-6060100     9
GL-6121905     9
GL-6020600     1
Name: GL_Code, dtype: int64
```

Visualization

```
sns.distplot(test_data['Inv_Amt']);
```

```
<ipython-input-62-4586698f5cbd>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751
```

```
sns.distplot(test_data['Inv_Amt']);
```



Data Preprocessing

Text Preprocessing

- Remove unwanted characters, numbers and symbols
- Convert to lowercase
- Remove stopwords
- Lemmatization

```
test_data['processed_text'] = test_data['Item_Description'].map(lambda x: re.sub('[^a-zA-Z
test_data['processed_text'] = test_data['processed_text'].map(lambda x: x.lower())
test_data['processed_text'] = test_data['processed_text'].apply(lambda x: ' '.join([word f
test_data['processed_text'] = test_data['processed_text'].apply(lambda x: ' '.join([lemmat
```

```
test_data['Item_Description']=test_data['processed_text']
```

Feature Engineering

```
gl_codes = test_data['GL_Code'].tolist()
gl_codes_without_prefix = [code[3:] for code in gl_codes]
print(gl_codes_without_prefix)
```

```
['6050100', '6100410', '6050310', '6101400', '6050310', '6020600', '2182000', '605016
```

```
df = pd.DataFrame({'GL_Code':gl_codes_without_prefix })
test_data["GL_Code"]=df
test_data["GL_Code"].head()
```

```
0    6050100
1    6100410
2    6050310
3    6101400
4    6050310
Name: GL_Code, dtype: object
```

```
ven_codes = test_data['Vendor_Code'].tolist()
ven_codes_without_prefix = [code[7:] for code in ven_codes]
print(ven_codes_without_prefix)
df = pd.DataFrame({'Vendor_Code':ven_codes_without_prefix })
test_data["Vendor_Code"]=df
test_data["Vendor_Code"].head()
```

```
['1181', '1554', '1513', '1044', '1114', '1406', '1883', '1640', '1509', '2229', '104
0    1181
1    1554
2    1513
3    1044
4    1114
Name: Vendor_Code, dtype: object
```

```
test_data = test_data.drop('processed_text', axis=1)
```

Feature Transformation

Encoding

```
le = LabelEncoder()
test_data["Vendor_Code"] = le.fit_transform(test_data["Vendor_Code"])
le2 = LabelEncoder()
test_data["GL_Code"]=le2.fit_transform(test_data["GL_Code"])
test_data.head()
```

	Inv_Id	Vendor_Code	GL_Code	Inv_Amt	Item_Description
0	15041	30	2	88.80	base rent store management aig equity sale cor...
1	15094	65	5	80.08	prototype comp production packaging design feb...

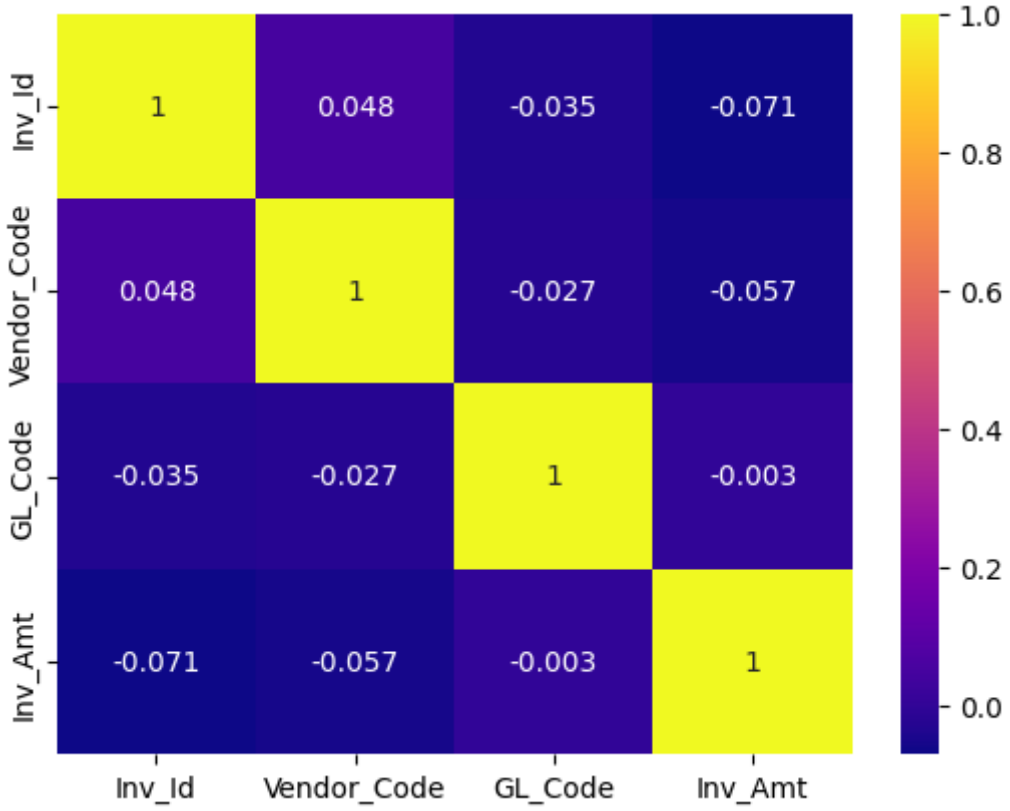
Correlation Analysis

```
corr=test_data.corr()
```

```
<ipython-input-70-c7151ce8fbe1>:1: FutureWarning: The default value of numeric_only i
  corr=test_data.corr()
```

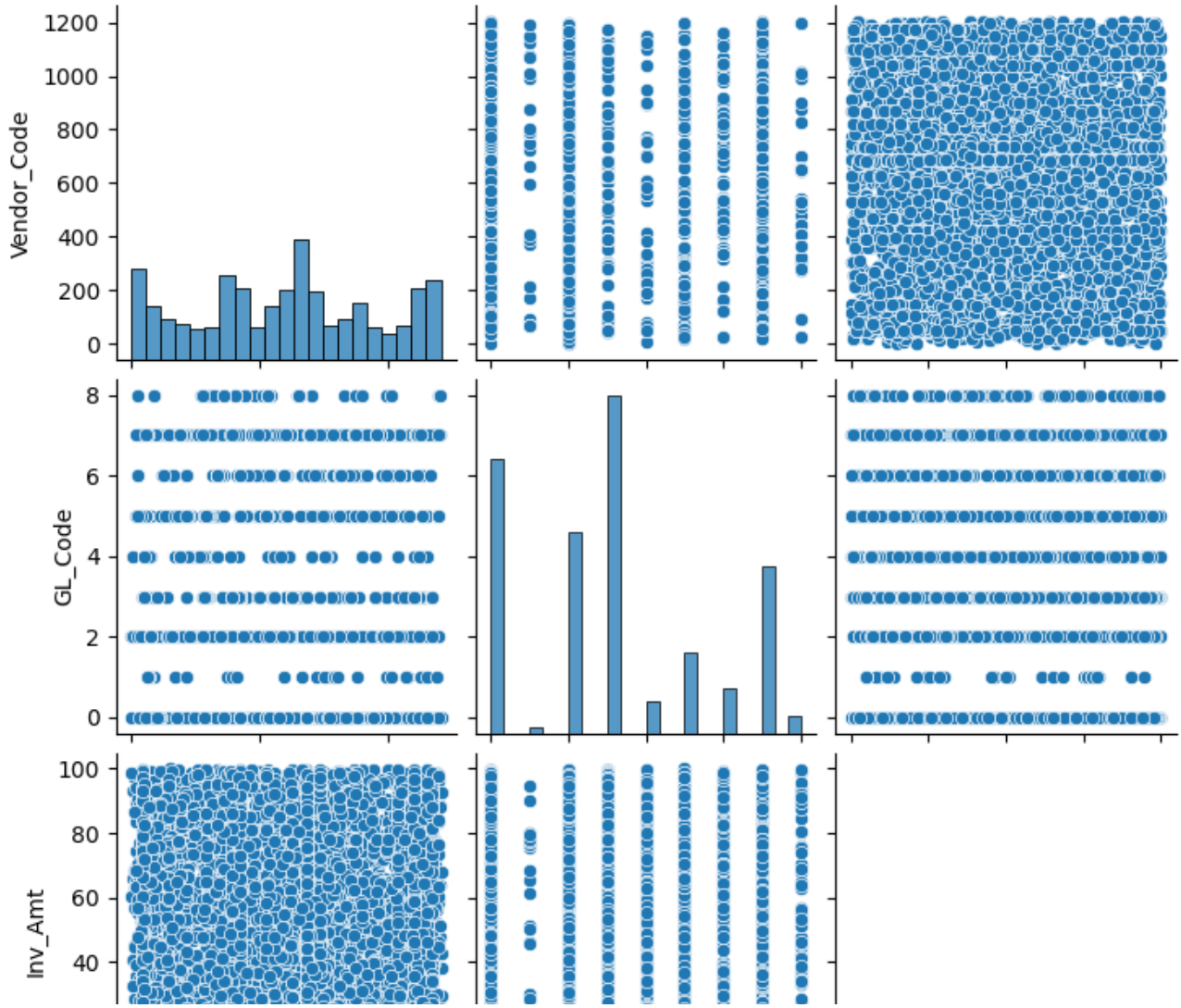
```
sns.heatmap(corr, annot=True, cmap='plasma')
```

```
<Axes: >
```




```
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7ff7633dc700>
```



Model Prediction

```
X_test_cv = cv.transform(test_data['Item_Description'])
y_pred = clf.predict(X_test_cv)
```

Code To Generate submission.csv

```
submission = pd.DataFrame({'Inv_Id': test_data['Inv_Id'], 'Product_Category': y_pred})
submission.to_csv('submission.csv', index=False)
```