1.Import Libraies

```
#Import Libraries
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics
import matplotlib.pyplot as plt
from sklearn.svm import LinearSVC
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text  import TfidfTransformer
import warnings
warnings.filterwarnings('ignore')
```

2 import Datasets

```
#import datasets
data=pd.read_csv("/content/drive/MyDrive/complaints.csv")
```

3.Data Undestanding

```
#Pick up nessasary columns
data=data[['Product', 'Consumer complaint narrative']]
```

```
#check whether any null value avilible
data.isnull().sum()
```

```
Product                            0
Consumer complaint narrative    1653441
dtype: int64
```

```
#Drop null values
data=data.dropna()
```

```
#Number of rows and columns
data.shape
```

```
(888973, 2)
```

```
#Data Type
data.dtypes
```

```
Product                         object
Consumer complaint narrative    object
dtype: object
```

```
#Rename column name to easy interpritation
data.columns=['product','complaint']
```

```
#Encode the object as an enumerated type or categorical variable
data['category_id'] = data['product'].factorize()[0]
```

```
#drop_duplicates() method helps in removing duplicates from the data frame
#Returns a sorted Data Frame with Same dimensions as of the function caller Data Frame.
complain_df=data[['product','category_id']].drop_duplicates().sort_values('category_id').reset_index(drop=True)
category_id_df=dict(complain_df.values)
category_id_df
```

```
{'Bank account or service': 11,
 'Checking or savings account': 0,
 'Consumer Loan': 9,
 'Credit card': 14,
 'Credit card or prepaid card': 2,
 'Credit reporting': 12,
 'Credit reporting, credit repair services, or other personal consumer reports': 1,
 'Debt collection': 4,
```

```
 'Money transfer, virtual currency, or money service': 8,
 'Money transfers': 16,
 'Mortgage': 3,
 'Other financial service': 15,
 'Payday loan': 10,
 'Payday loan, title loan, or personal loan': 7,
 'Prepaid card': 13,
 'Student loan': 6,
 'Vehicle loan or lease': 5,
 'Virtual currency': 17}
```

```
#Top 5 rows of datasets.
data=data.reset_index(drop=True)
data.head()
```

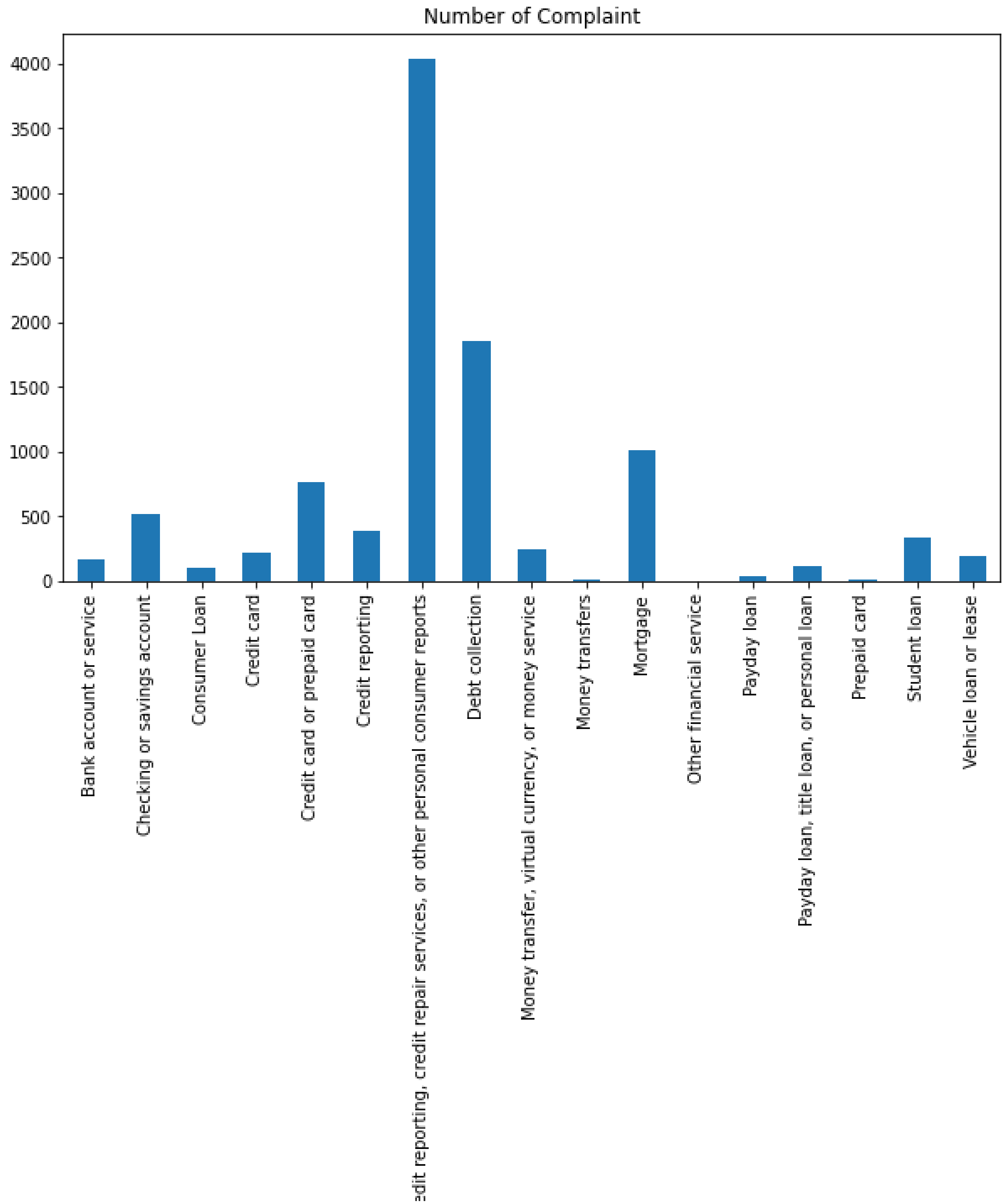
	product	complaint	category_id
0	Checking or savings account	This Bank FNBO is terrible! Stay away from the...	0
1	Credit reporting, credit repair services, or o...	First Progress Card was notified throughout th...	1
2	Credit reporting, credit repair services, or o...	As by Law, under 15 U.S Code 1601- Congression...	1
3	Credit reporting, credit repair services, or o...	This is a copy of my most recent email to XXXX...	1
4	Credit card or prepaid card	This is in reference to case number XXXX. This...	2

```
# Input: Consumer_complaint_narrative
# Output: product
```

4.Imbalance Data

```
plt.figure(figsize=(10,6))
data.groupby("product").complaint.count().plot.bar()
plt.title("Number of Complaint")
# We see that the number of complaints per product is imbalanced.
# Consumers' complaints are more biased towards Debt collection,Credit reporting and Mortgage.
```

```
Text(0.5, 1.0, 'Number of Complaint')
```




```
# calculate a tf-idf vector for each of consumer complaint narratives:
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
tfid=TfidfVectorizer(stop_words='english',  #stop_words is set to "english" to remove all common pronouns ("a", "the", ...)
                     norm='l2',             #ensure all our feature vectors have a euclidian norm of 1
                     sublinear_tf=True,     #set to True to use a logarithmic form for frequency.
                     encoding='latin-1',    #maps possible byte values to first Unicode code points,ensures decoding errors will never occur regardless of the configured en
                     min_df=5,        # minimum numbers of documents a word must be present in to be kept.
                    ngram_range=(1,2)) #set to (1, 2) to indicate that we want to consider both unigrams and bigrams


#We Randomly pick up 10000 data for easy interpretation
data=data.sample(10000,random_state=12).reset_index(drop=True,)
data.head()
```

	product	complaint	category_id
0	Student loan	I was charged {$42.00} per month for the past ...	6
1	Mortgage	My mortgage servicer, Nationstar Mortgage, is ...	3
2	Debt collection	I have been receiving numerous phone calls fro...	4
3	Credit reporting, credit repair services, or o...	I faxed over a copy of my drivers license, soc...	1
4	Credit card	I SAW PROMOTION FOR CREDIT CARD ON XXXX FOR WE...	14

```
#Define Fetures and labels
fetures=tfid.fit_transform(data.complaint).toarray()
labels=data.category_id


#devide train and test dataset
x_train,y_test,y_train,y_test=train_test_split(data['complaint'],data['product'],random_state=0)


# tokenize a collection of text documents and build a vocabulary of known words, but also to encode new documents using that vocabulary.
cv=CountVectorizer()
x_train_count=cv.fit_transform(x_train)


# Transform a count matrix to a normalized tf or tf-idf representation.
tfid_transformer=TfidfTransformer()
x_train_tfid=tfid_transformer.fit_transform(x_train_count)
```

```
# Naive Bayes Classifier .the one most suitable for word counts is the multinomial variant.
clf=MultinomialNB().fit(x_train_tfid,y_train)
```

5.prediction

```
data[data['complaint']== "I have been monitoring my credit report with Transunion only for about 3 months. I had an identity theft a few years ago and it was never resolvec
```

	product	complaint	category_id
9998	Credit reporting, credit repair services, or o...	I have been monitoring my credit report with T...	1

```
print(clf.predict(cv.transform(['After receiving this paper bill through the mail, I called my dental office to verify. My dental office said I was all paid up and had a X>
```

```
['Credit reporting, credit repair services, or other personal consumer reports']
```

```
print(clf.predict(cv.transform(["In XXXX I got an American Express card, In XXXX I was not receiving my bills, I contacted American Express, They told me that my account wa
```

```
['Credit reporting, credit repair services, or other personal consumer reports']
```

```
#Prediction
print(clf.predict(cv.transform(['On XX/XX/2021 my unemployment weekly amount was put on my Way2go card. Immediately after I transferred the entire amount to my XXXX XXXX X>
```

```
['Credit reporting, credit repair services, or other personal consumer reports']
```

```
data[data['complaint']=='My mortgage servicer, Nationstar Mortgage, is attempting to perform a foreclosure sale on my property, by using falsified documents. There are many
```

	product	complaint	category_id
1	Mortgage	My mortgage servicer, Nationstar Mortgage, is ...	3

```
print(clf.predict(cv.transform(['I recently applied for a home loan and was denied do to fraudulent activities. I have never applied for AMEX or have I used this credit car
```

```
['Credit reporting, credit repair services, or other personal consumer reports']
```

6.Model Building

```
#Split data in training and testing
x_train, x_test, y_train, y_test, indices_train, indices_test = train_test_split(fetures, labels, data.index, test_size=0.33, random_state=0)
```

```
# The objective of a Linear SVC (Support Vector Classifier) is to fit to the data you provide,
# returning a "best fit" hyperplane that divides, or categorizes, your data.
#Build the model
model=LinearSVC()
model.fit(x_train,y_train)
```

```
LinearSVC()
```

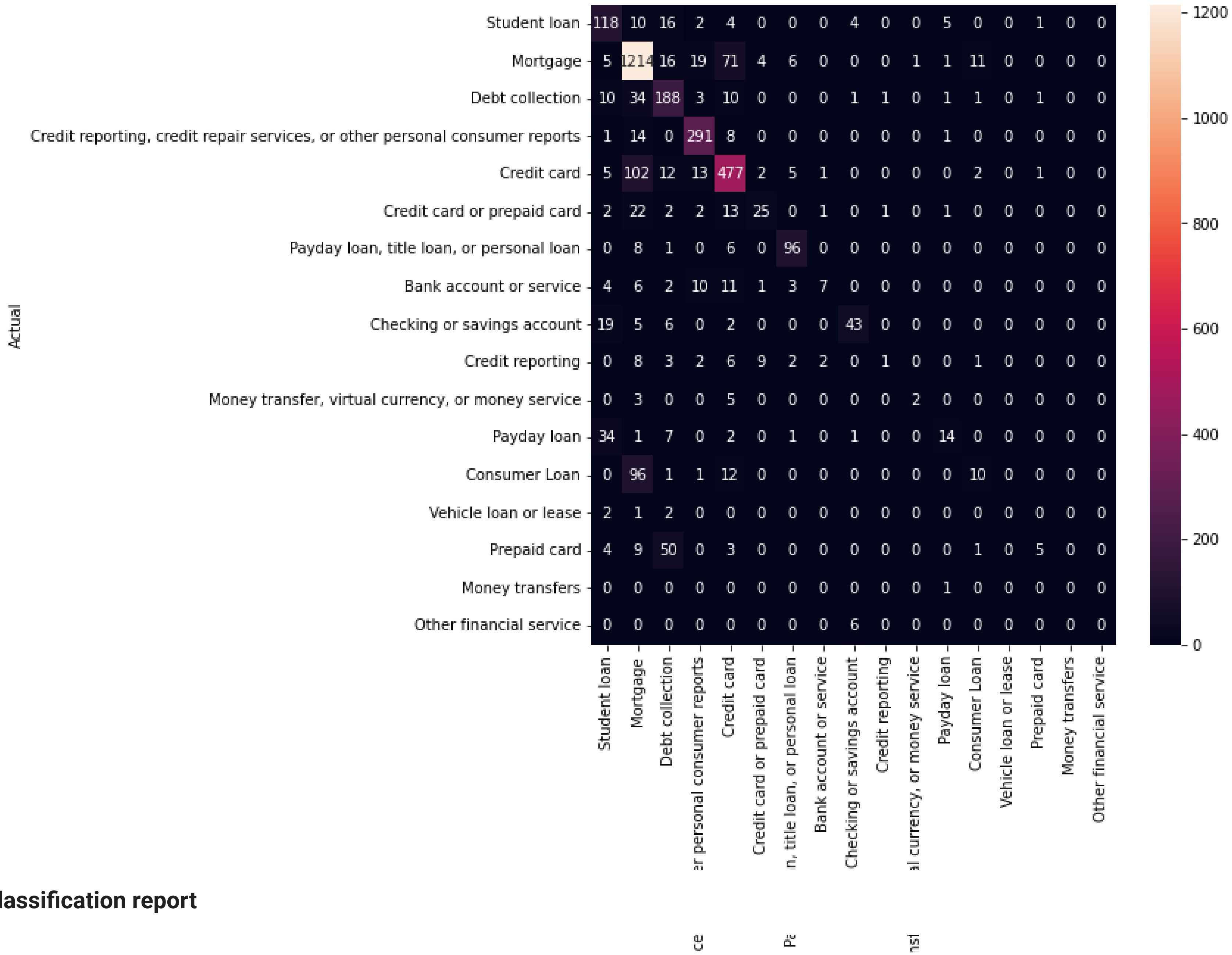
7.Model Testing

```
#Predict test data with help of model
y_pred=model.predict(x_test)
```

```
#Summary of the number of correct and incorrect predictions made by a classifier.
con_matrix=confusion_matrix(y_test,y_pred)
```

7.1 Heat Map

```
#Plot Heat map
#Heatmap contains values representing various shades of the same colour for each value to be plotted
fig,ax=plt.subplots(figsize=(8,8))
sns.heatmap(con_matrix,annot=True,fmt='d',xticklabels=data['product'].unique(),yticklabels=data['product'].unique())
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



7.2 classification report

```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.74      0.58      0.65       204
           1       0.90      0.79      0.84      1533
           2       0.75      0.61      0.68       306
           3       0.92      0.85      0.88       343
```

```
           4       0.77      0.76      0.76       630
           5       0.36      0.61      0.45        41
           6       0.86      0.85      0.86       113
           7       0.16      0.64      0.25        11
           8       0.57      0.78      0.66        55
           9       0.03      0.33      0.05         3
          10       0.20      0.67      0.31         3
          11       0.23      0.58      0.33        24
          12       0.08      0.38      0.14        26
          13       0.00      0.00      0.00         0
          14       0.07      0.62      0.12         8
          15       0.00      0.00      0.00         0
          16       0.00      0.00      0.00         0

    accuracy                           0.75      3300
   macro avg       0.39      0.53      0.41      3300
weighted avg       0.82      0.75      0.78      3300
```

```
#From we get 75% accuracy
```

✓ 0s completed at 1:39 PM