```
!pip install xgboost

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import r2_score

plt.style.use('ggplot')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (1.7.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.22.4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.10.1)
```

Reading the Data from the Given excel file

```
data_train=pd.read_csv('/content/Consumer_Complaints_train.csv')
data_train.head()
```

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response
0	2015-10-14	Credit reporting	NaN	Incorrect information on credit report	Information is not mine	NaN	NaN
1	2015-04-26	Bank account or service	Other bank product/service	Deposits and withdrawals	NaN	RE : XXXX XXXX XXXX- PRIVILEGED AND CONFIDENTIA...	NaN
2	2013-12-20	Credit card	NaN	Other	NaN	NaN	NaN
3	2016-03-03	Debt collection	Other (i.e. phone, health club, etc.)	Disclosure verification of debt	Not given enough info to verify debt	NaN	Company has responded to the consumer and the ...
4	2015-01-30	Debt collection	Medical	Disclosure verification of debt	Not given enough info to verify debt	NaN	NaN

```
data_test=pd.read_csv('/content/Consumer_Complaints_test.csv')
data_test.head()
```

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags
0	2015-01-17	Credit card	NaN	Customer service / Customer relations	NaN	NaN	NaN	Citibank	TX	75241	NaN
1	2016-06-22	Consumer Loan	Title loan	Payment to acct not credited	NaN	NaN	Company believes it acted appropriately as aut...	Larsen MacColl Partners II, L.P.	TX	76548	Servicemembe
2	2015-09-04	Credit card	NaN	Credit line increase/decrease	NaN	I WANT TO REQUEST A CREDIT LINE INCREASE OF XX...	NaN	Capital One	NC	271XX	NaN
3	2016-05-17	Consumer Loan	Installment loan	Problems when you are unable to pay	NaN	I have asked One Main Financial not to call my...	NaN	OneMain Financial Holdings, LLC	MO	634XX	NaN
						I have	Company				

Checking the data type for both data (test file and train file) bold text

```
1 data_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 358810 entries, 0 to 358809
Data columns (total 18 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   Date received                 358810 non-null  object
 1   Product                       358810 non-null  object
 2   Sub-product                   255024 non-null  object
 3   Issue                         358810 non-null  object
 4   Sub-issue                     139436 non-null  object
 5   Consumer complaint narrative  56180 non-null   object
 6   Company public response       67931 non-null   object
 7   Company                       358810 non-null  object
 8   State                         355907 non-null  object
 9   ZIP code                      355899 non-null  object
 10  Tags                          50226 non-null   object
 11  Consumer consent provided?    101580 non-null  object
 12  Submitted via                 358810 non-null  object
 13  Date sent to company          358810 non-null  object
 14  Company response to consumer  358810 non-null  object
 15  Timely response?              358810 non-null  object
 16  Consumer disputed?            358810 non-null  object
 17  Complaint ID                  358810 non-null  int64
dtypes: int64(1), object(17)
memory usage: 49.3+ MB
```

```
1 data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119606 entries, 0 to 119605
Data columns (total 17 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   Date received                 119606 non-null  object
 1   Product                       119606 non-null  object
 2   Sub-product                   84923 non-null   object
 3   Issue                         119606 non-null  object
 4   Sub-issue                     46356 non-null   object
 5   Consumer complaint narrative  18914 non-null   object
 6   Company public response       22460 non-null   object
 7   Company                       119606 non-null  object
 8   State                         118670 non-null  object
 9   ZIP code                      118669 non-null  object
 10  Tags                          16977 non-null   object
 11  Consumer consent provided?    33907 non-null   object
 12  Submitted via                 119606 non-null  object
 13  Date sent to company          119606 non-null  object
 14  Company response to consumer  119606 non-null  object
 15  Timely response?              119606 non-null  object
 16  Complaint ID                  119606 non-null  int64
dtypes: int64(1), object(16)
memory usage: 15.5+ MB
```

```
1 data_train.describe()
```

	Complaint ID
count	3.588100e+05
mean	1.043850e+06
std	5.945511e+05
min	5.000000e+00
25%	5.339312e+05
50%	1.064641e+06
75%	1.561380e+06
max	2.126221e+06

```
1 data_test.describe()
```

	Complaint ID
count	1.196060e+05
mean	1.043759e+06
std	5.937896e+05
min	7.000000e+00
25%	5.331275e+05
50%	1.065378e+06
75%	1.557638e+06
max	2.128060e+06

```
1 data_train.isnull().sum()
```

```
Date received                       0
Product                             0
Sub-product                    103786
Issue                               0
Sub-issue                      219374
Consumer complaint narrative   302630
Company public response        290879
Company                             0
State                            2903
ZIP code                         2911
Tags                           308584
Consumer consent provided?     257230
Submitted via                       0
Date sent to company                0
Company response to consumer        0
Timely response?                    0
Consumer disputed?                  0
Complaint ID                        0
dtype: int64
```

Doing missing value analysis and dropcolumns where more than 25% of data are missing

```
1 #Missing Values analysis on data_train
2 round(100*data_train.isna().sum()/len(data_train),2)
```

```
Date received                   0.00
Product                         0.00
Sub-product                    28.93
Issue                           0.00
Sub-issue                      61.14
Consumer complaint narrative   84.34
Company public response        81.07
Company                         0.00
State                           0.81
ZIP code                        0.81
Tags                           86.00
Consumer consent provided?     71.69
Submitted via                   0.00
Date sent to company            0.00
Company response to consumer    0.00
Timely response?                0.00
Consumer disputed?              0.00
Complaint ID                    0.00
dtype: float64
```

```
1 ##Dropping columns of data_train with more than 40% null values
```

```
1 cols=data_train.columns
2
3 for i in cols:
4     if((100*(data_train[i].isnull().sum()/len(data_train))) >= 40):
5         data_train.drop(i, 1, inplace = True)
```

```
1 # Again checking null values percentage
2 round(100*data_train.isna().sum()/len(data_train),2)
```

```
Date received                   0.00
Product                         0.00
Sub-product                    28.93
Issue                           0.00
Company                         0.00
State                           0.81
ZIP code                        0.81
Submitted via                   0.00
Date sent to company            0.00
Company response to consumer    0.00
Timely response?                0.00
Consumer disputed?              0.00
Complaint ID                    0.00
dtype: float64
```

Extracting Day, Month,and Year from Date Received Column and create new fields for a month, year,and day

```
1 ### Extracting Day, Month,and Year from Date Received Column and create new fields for a month, year,and day of a data_train dataset
```

```
1 date_values=pd.DatetimeIndex(data_train['Date received'])
2 data_train['Day']=date_values.day
3 data_train['Month']=date_values.month
4 data_train['Year']=date_values.year
5 data_train.head()
```

	Date received	Product	Sub-product	Issue	Company	State	ZIP code	Submitted via	Date sent to company	Company response to consumer	Timely response?	Co dis
0	2015-10-14	Credit reporting	NaN	Incorrect information on credit report	Equifax	GA	30134	Web	2015-10-14	Closed with explanation	Yes	
1	2015-04-26	Bank account or service	Other bank product/service	Deposits and withdrawals	Wells Fargo & Company	GA	319XX	Web	2015-04-26	Closed with explanation	Yes	
2	2013-12-20	Credit card	NaN	Other	Citibank	SC	29203	Phone	2014-01-03	Closed with non-monetary relief	Yes	
3	2016-03-03	Debt collection	Other (i.e. phone, health club, etc.)	Disclosure verification of debt	FAIR COLLECTIONS & OUTSOURCING, INC.	OH	43082	Referral	2016-03-04	Closed with explanation	Yes	
4	2015-01-30	Debt collection	Medical	Disclosure verification of debt	HCFS Health Care Financial Services, Inc.	CA	90036	Web	2015-01-30	Closed with explanation	Yes	

```
1 data_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 358810 entries, 0 to 358809
Data columns (total 16 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   Date received                 358810 non-null  object
 1   Product                       358810 non-null  object
 2   Sub-product                   255024 non-null  object
 3   Issue                         358810 non-null  object
```

```
 4   Company                       358810 non-null  object
 5   State                         355907 non-null  object
 6   ZIP code                      355899 non-null  object
 7   Submitted via                 358810 non-null  object
 8   Date sent to company          358810 non-null  object
 9   Company response to consumer  358810 non-null  object
 10  Timely response?              358810 non-null  object
 11  Consumer disputed?            358810 non-null  object
 12  Complaint ID                  358810 non-null  int64
 13  Day                           358810 non-null  int64
 14  Month                         358810 non-null  int64
 15  Year                          358810 non-null  int64
dtypes: int64(4), object(12)
memory usage: 43.8+ MB
```

```
1 data_test.isnull().sum()
```

```
Date received                       0
Product                             0
Sub-product                     34683
Issue                               0
Sub-issue                       73250
Consumer complaint narrative    100692
Company public response          97146
Company                             0
State                             936
ZIP code                          937
Tags                            102629
Consumer consent provided?       85699
Submitted via                       0
Date sent to company                0
Company response to consumer        0
Timely response?                    0
Complaint ID                        0
dtype: int64
```

```
1 #Missing Values analysis on data_test
2 round(100*data_test.isna().sum()/len(data_test),2)
```

```
Date received                    0.00
Product                          0.00
Sub-product                     29.00
Issue                            0.00
Sub-issue                       61.24
Consumer complaint narrative    84.19
Company public response         81.22
Company                          0.00
State                            0.78
ZIP code                         0.78
Tags                            85.81
Consumer consent provided?      71.65
Submitted via                    0.00
Date sent to company             0.00
Company response to consumer     0.00
Timely response?                 0.00
Complaint ID                     0.00
dtype: float64
```

```
1 ##Dropping columns of data_test with more than 40% null values
```

```
1 #Dropping columns of data_test with more than 40% null values
2
3 cols=data_test.columns
4 for i in cols:
5     if((100*(data_test[i].isnull().sum()/len(data_test))) >= 40):
6         data_test.drop(i, 1, inplace = True)
```

```
1 # Again checking null values percentage of data_test
2 round(100*data_test.isna().sum()/len(data_test),2)
```

```
Date received                    0.00
Product                          0.00
Sub-product                     29.00
Issue                            0.00
Sub-issue                       61.24
Consumer complaint narrative    84.19
Company public response         81.22
Company                          0.00
State                            0.78
ZIP code                         0.78
Tags                            85.81
Consumer consent provided?      71.65
Submitted via                    0.00
Date sent to company             0.00
```

```
Company response to consumer     0.00
Timely response?                 0.00
Complaint ID                     0.00
dtype: float64
```

```
#Extracting Day, Month,and Year from Date Received Column and create new fields for a month, year,and day of a data_test dataset
```

```
date_values=pd.DatetimeIndex(data_test['Date received'])
data_test['Day']=date_values.day
data_test['Month']=date_values.month
data_test['Year']=date_values.year
data_test.head()
```

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	ZIP code	Tags
0	2015-01-17	Credit card	NaN	Customer service / Customer relations	NaN	NaN	NaN	Citibank	TX	75241	NaN
1	2016-06-22	Consumer Loan	Title loan	Payment to acct not credited	NaN	NaN	Company believes it acted appropriately as aut...	Larsen MacColl Partners II, L.P.	TX	76548	Servicemembe
2	2015-09-04	Credit card	NaN	Credit line increase/decrease	NaN	I WANT TO REQUEST A CREDIT LINE INCREASE OF XX...	NaN	Capital One	NC	271XX	NaN
3	2016-05-17	Consumer Loan	Installment loan	Problems when you are unable to pay	NaN	I have asked One Main Financial not to call my...	NaN	OneMain Financial Holdings, LLC	MO	634XX	NaN
4	2016-07-07	Debt collection	Other (i.e. phone, health club, etc.)	Improper contact or sharing of info	Contacted employer after asked not to	I have received several calls from a XXXX XXXX...	Company has responded to the consumer and the ...	GMA Investments, LLC	SC	296XX	NaN

```
data_train['Date sent to company'] = pd.to_datetime(data_train['Date sent to company'])
data_train['Date received'] = pd.to_datetime(data_train['Date received'])
data_train['Days_held'] = (data_train['Date sent to company'] - data_train['Date received']).dt.days

data_test['Date sent to company'] = pd.to_datetime(data_test['Date sent to company'])
data_test['Date received'] = pd.to_datetime(data_test['Date received'])
data_test['Days_held'] = (data_test['Date sent to company'] - data_test['Date received']).dt.days
```

##Calculate the Number of Days the Complaint was with the Company and create a new field as “Days held

```
data_train['Days_held'] = (data_train['Date sent to company'] - data_train['Date received']).dt.days
data_test['Days_held'] = (data_test['Date sent to company'] - data_test['Date received']).dt.days
```

```
#Dropping Unnecessary columns of train_dataset
```

```
data_train.drop(['Date received'], axis = 1, inplace = True)
data_train.drop(['Date sent to company'], axis = 1, inplace = True)
data_train.drop(['ZIP code'], axis = 1, inplace = True)
data_train.drop(['Complaint ID'], axis = 1, inplace = True)
```

```
data_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 358810 entries, 0 to 358809
Data columns (total 13 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
```

```
 0   Product                       358810 non-null  object
 1   Sub-product                   255024 non-null  object
 2   Issue                         358810 non-null  object
 3   Company                       358810 non-null  object
 4   State                         355907 non-null  object
 5   Submitted via                 358810 non-null  object
 6   Company response to consumer  358810 non-null  object
 7   Timely response?              358810 non-null  object
 8   Consumer disputed?            358810 non-null  object
 9   Day                           358810 non-null  int64
 10  Month                         358810 non-null  int64
 11  Year                          358810 non-null  int64
 12  Days_held                     358810 non-null  int64
dtypes: int64(4), object(9)
memory usage: 35.6+ MB
```

Drop "Date Received","Date Sent to Company","ZIP Code", "Complaint ID"fields•Imputing Nullvalue in “State”by Mode

```
#Dropping Unnecessary Columns of data_test set
```

```
data_test.drop(['Date received'], axis = 1, inplace = True)
data_test.drop(['Date sent to company'], axis = 1, inplace = True)
data_test.drop(['ZIP code'], axis = 1, inplace = True)
data_test.drop(['Complaint ID'], axis = 1, inplace = True)
```

```
data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119606 entries, 0 to 119605
Data columns (total 17 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   Product                       119606 non-null  object
 1   Sub-product                   84923 non-null   object
 2   Issue                         119606 non-null  object
 3   Sub-issue                     46356 non-null   object
 4   Consumer complaint narrative  18914 non-null   object
 5   Company public response       22460 non-null   object
 6   Company                       119606 non-null  object
 7   State                         118670 non-null  object
 8   Tags                          16977 non-null   object
 9   Consumer consent provided?    33907 non-null   object
 10  Submitted via                 119606 non-null  object
 11  Company response to consumer  119606 non-null  object
 12  Timely response?              119606 non-null  object
 13  Day                           119606 non-null  int64
 14  Month                         119606 non-null  int64
 15  Year                          119606 non-null  int64
 16  Days_held                     119606 non-null  int64
dtypes: int64(4), object(13)
memory usage: 15.5+ MB
```

Imputing Nullvalue in “State”by Mode

```
data_train['State'].fillna(data_train['State'].mode()[0], inplace=True)
data_test['State'].fillna(data_test['State'].mode()[0], inplace=True)
```

```
#Create a newfield 'Week_Received' where we calculate the week based on the day of receiving
```

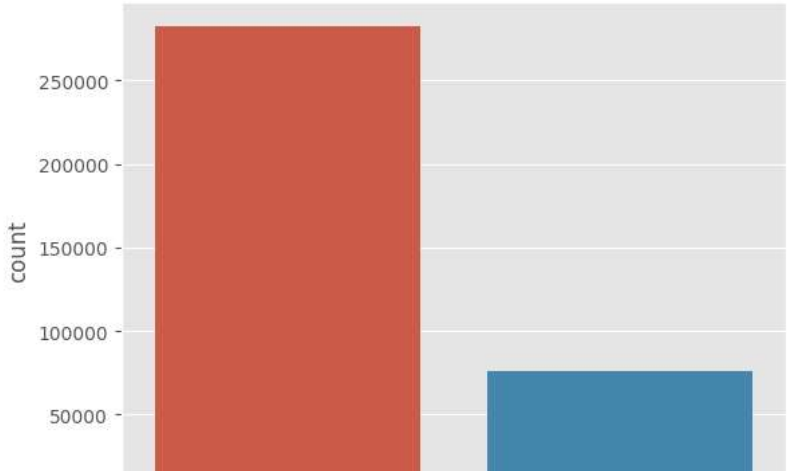
```
data_train['Week_Received'] = np.ceil(data_train['Days_held'] / 7).astype(int)

data_test['Week_Received'] = np.ceil(data_test['Days_held'] / 7).astype(int)
```

Storing data of disputed people into the “disputed_cons” variable for future tasks

```
disputed_cons = data_train[data_train['Consumer disputed?'] == 'Yes']
```

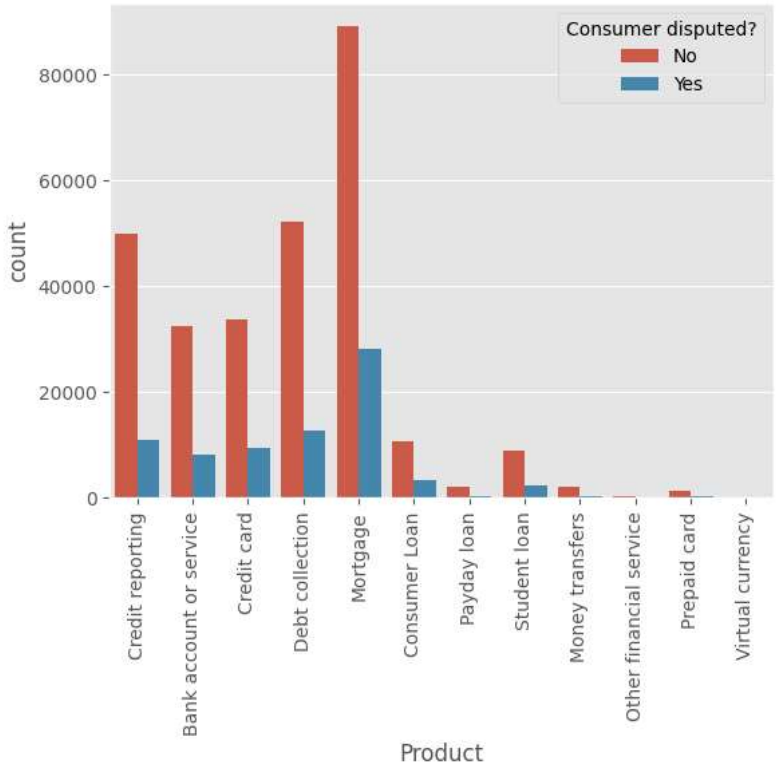
Plot bar graph of the total no of disputes of consumers with the help of seaborn

```
sns.countplot(x='Consumer disputed?', data=data_train)
plt.show()
```



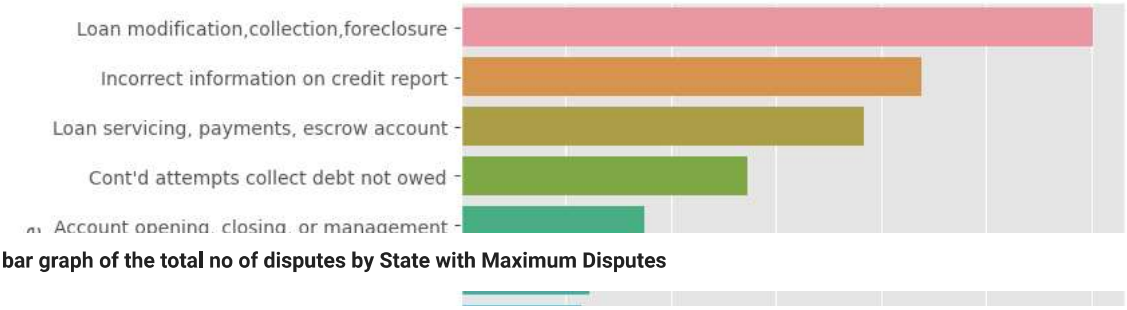
Plot bar graph of the total no of disputes products-wise with the help of seaborn

```
1 sns.countplot(x='Product', hue='Consumer disputed?', data=data_train)
2 plt.xticks(rotation=90)
3 plt.show()
```



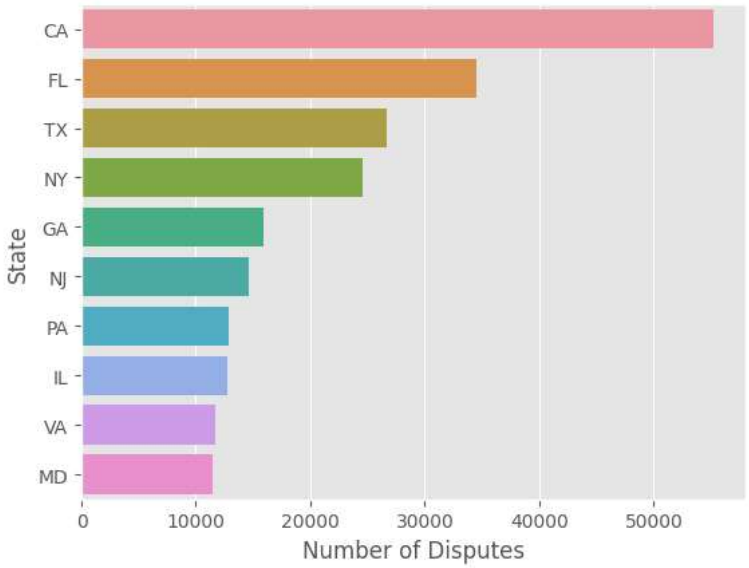
Plot bar graph of the total no of disputes with Top Issues by Highest Disputes, with the help of seaborn

```
1 top_issues = data_train['Issue'].value_counts().nlargest(10)
2 sns.barplot(x=top_issues.values, y=top_issues.index)
3 plt.xlabel('Number of Disputes')
4 plt.ylabel('Issue')
5 plt.show()
```



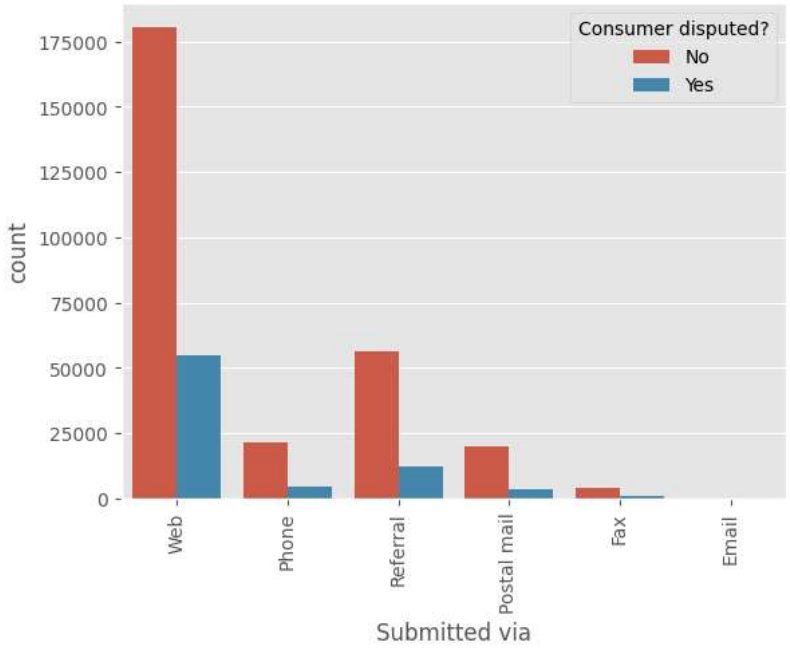
Plot bar graph of the total no of disputes by State with Maximum Disputes

```
state_disputes = data_train['State'].value_counts().nlargest(10)
sns.barplot(x=state_disputes.values, y=state_disputes.index)
plt.xlabel('Number of Disputes')
plt.ylabel('State')
plt.show()
```



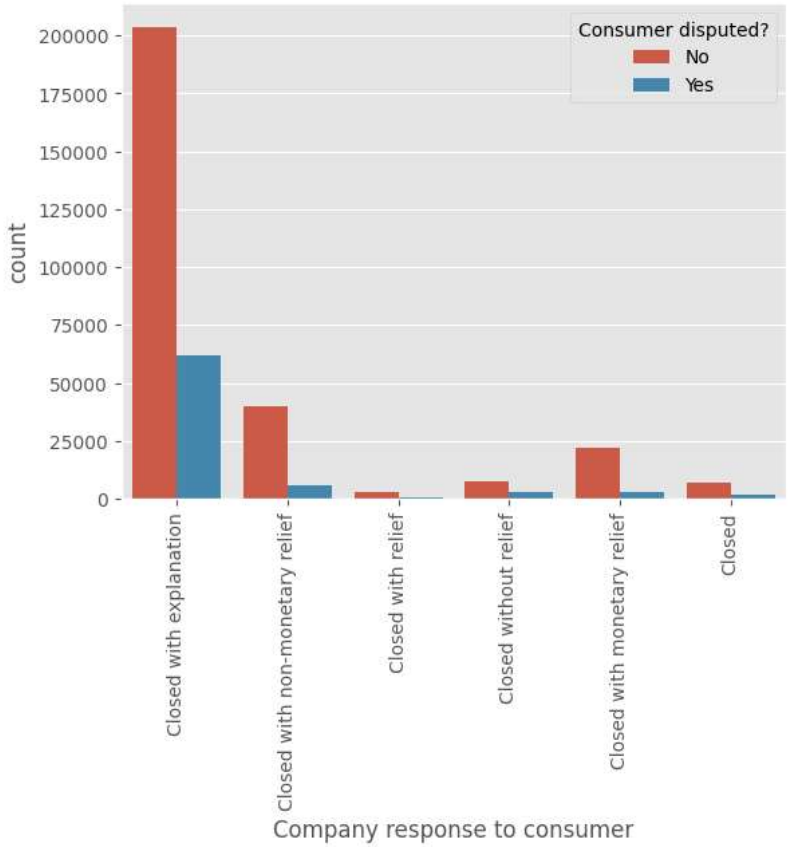
#Plot bar graph of the total no of disputes Submitted Via different source

```
sns.countplot(x='Submitted via', hue='Consumer disputed?', data=data_train)
plt.xticks(rotation=90)
plt.show()
```



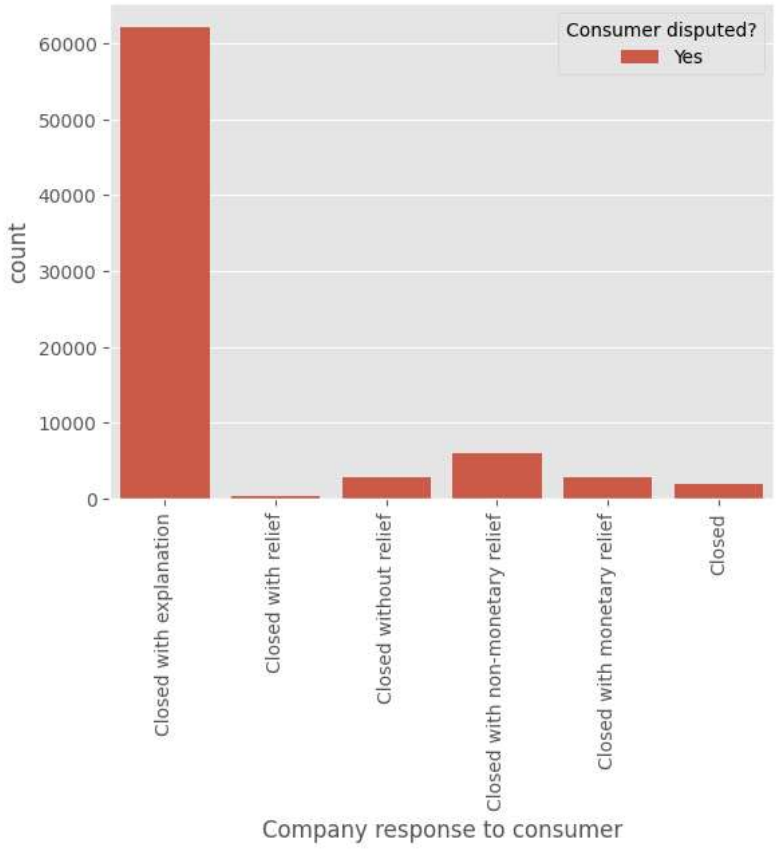
Plot bar graph of the total no of disputes where the Company's Response to the Complaints

```
sns.countplot(x='Company response to consumer', hue='Consumer disputed?', data=data_train)
plt.xticks(rotation=90)
plt.show()
```



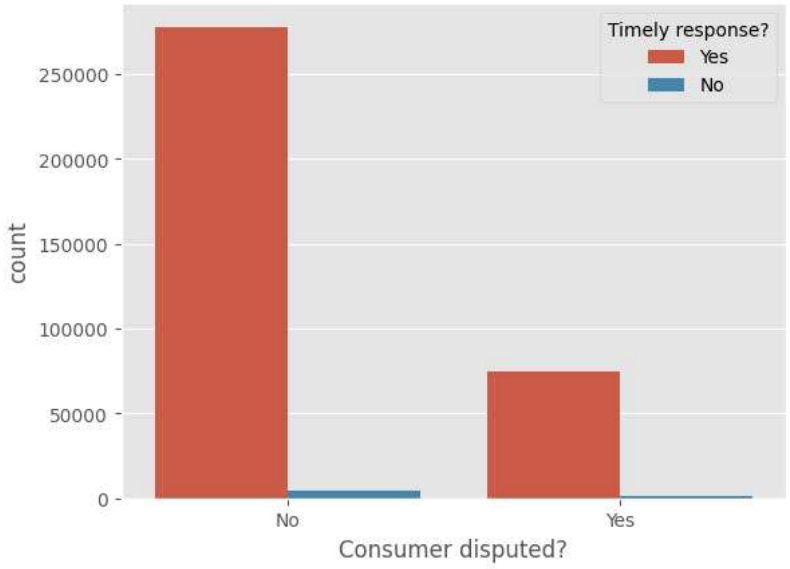
Plot bar graph of the total no of disputes where the Company's Response Leads to Disputes

```
sns.countplot(x='Company response to consumer', hue='Consumer disputed?', data=disputed_cons)
plt.xticks(rotation=90)
plt.show()
```



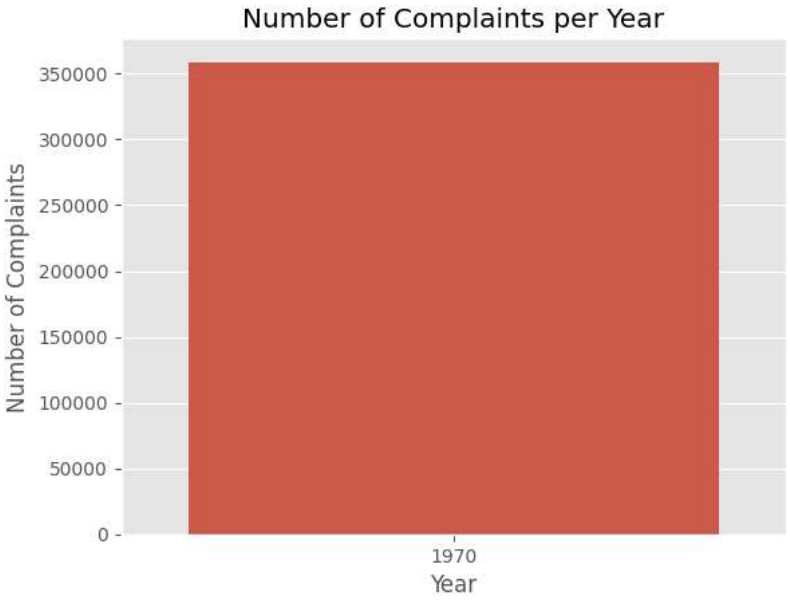
Plot bar graph of the total no of disputes.Whether there are Disputes Instead of Timely Response

```
sns.countplot(x='Consumer disputed?', hue='Timely response?', data=data_train)
plt.show()
```



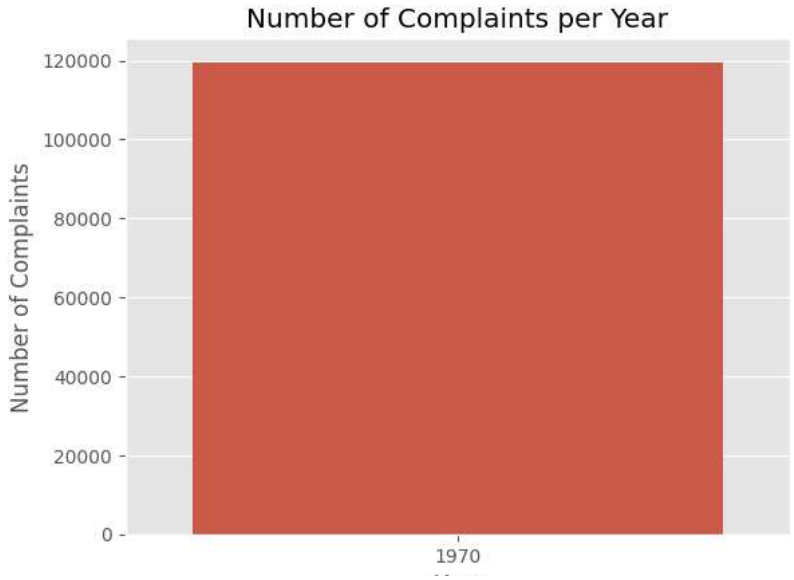
Plot bar graph of the total no of disputes over Year Wise Complaints of train dataset

```
data_train['Year_Received'] = pd.to_datetime(data_train['Year']).dt.year
year_complaints = data_train['Year_Received'].value_counts().sort_index()

sns.barplot(x=year_complaints.index, y=year_complaints.values)
plt.xlabel('Year')
plt.ylabel('Number of Complaints')
plt.title('Number of Complaints per Year')
plt.show()
```



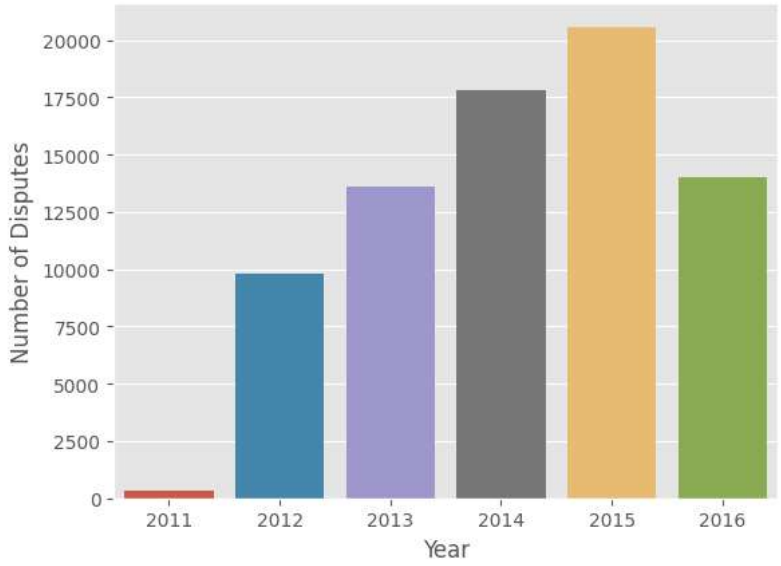

```
##Plot bar graph of the total no of disputes over Year Wise Complaints of test dataset
```

```
data_test['Year_Received'] = pd.to_datetime(data_test['Year']).dt.year
year_complaints = data_test['Year_Received'].value_counts().sort_index()

sns.barplot(x=year_complaints.index, y=year_complaints.values)
plt.xlabel('Year')
plt.ylabel('Number of Complaints')
plt.title('Number of Complaints per Year')
plt.show()
```



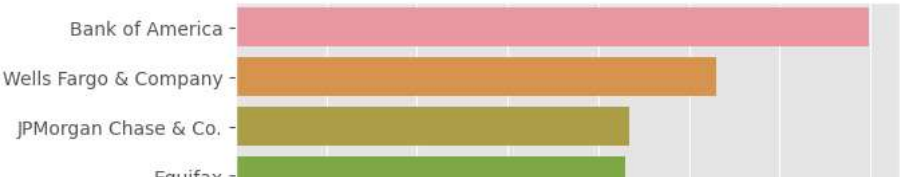
Plot bar graph of the total no of disputes over Year Wise Disputes

```
year_disputes = data_train[data_train['Consumer disputed?'] == 'Yes']['Year'].value_counts().sort_index()
sns.barplot(x=year_disputes.index, y=year_disputes.values)
plt.xlabel('Year')
plt.ylabel('Number of Disputes')
plt.show()
```



Plot bar graph of Top Companies with Highest Complaints

```
top_companies = data_train['Company'].value_counts().nlargest(10)
sns.barplot(x=top_companies.values, y=top_companies.index)
plt.xlabel('Number of Complaints')
plt.ylabel('Company')
plt.show()
```



#Convert all negative days held to zero

```
1 data_train['Days_held'] = data_train['Days_held'].apply(lambda x: 0 if x < 0 else x)
```

```
1 data_train.columns
```

```
Index(['Product', 'Sub-product', 'Issue', 'Company', 'State', 'Submitted via',
       'Company response to consumer', 'Timely response?',
       'Consumer disputed?', 'Day', 'Month', 'Year', 'Days_held',
       'Week_Received', 'Year_Received'],
      dtype='object')
```



```
1 data_test.columns
```

```
Index(['Product', 'Sub-product', 'Issue', 'Sub-issue',
       'Consumer complaint narrative', 'Company public response', 'Company',
       'State', 'Tags', 'Consumer consent provided?', 'Submitted via',
       'Company response to consumer', 'Timely response?', 'Day', 'Month',
       'Year', 'Days_held', 'Week_Received', 'Year_Received'],
      dtype='object')
```

Drop UnnecessaryColumns for the Model Buildinglike:'Company', 'State', 'Year_Received', 'Days_held

```
1 columns_to_drop = ['Company', 'State', 'Year_Received', 'Days_held','Sub-product','Issue']
2 data_train.drop(columns=columns_to_drop, inplace=True)
3 data_test.drop(columns=columns_to_drop, inplace=True)
```

Change Consumer Disputed Column to 0 and 1(yes to 1, and no to 0)

```
1 data_train['Consumer disputed?'] = data_train['Consumer disputed?'].map({'Yes': 1, 'No': 0})
2
```

Create Dummy Variables for categorical features and concat with the original data framelike: 'Product,"Submitted via,"Company response to consumer,"Timely response?

```
1 categorical_features = ['Product', 'Submitted via', 'Company response to consumer', 'Timely response?']
2 data_train = pd.get_dummies(data_train, columns=categorical_features,drop_first=True)
```

```
1 data_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 358810 entries, 0 to 358809
Data columns (total 9 columns):
 #   Column                        Non-Null Count   Dtype
---  ------                        --------------   -----
 0   Product                       358810 non-null  object
 1   Submitted via                 358810 non-null  object
 2   Company response to consumer  358810 non-null  object
 3   Timely response?              358810 non-null  object
 4   Consumer disputed?            358810 non-null  object
 5   Day                           358810 non-null  int64
 6   Month                         358810 non-null  int64
 7   Year                          358810 non-null  int64
 8   Week_Received                 358810 non-null  int64
dtypes: int64(4), object(5)
memory usage: 24.6+ MB
```

```
1 data_test = pd.get_dummies(data_test, columns=categorical_features,drop_first=True)
```

```
1 data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 119606 entries, 0 to 119605
Data columns (total 26 columns):
 #   Column                                                    Non-Null Count   Dtype
---  ------                                                    --------------   -----
 0   Day                                                       119606 non-null  int64
```

```
 1   Month                                                    119606 non-null  int64
 2   Year                                                     119606 non-null  int64
 3   Week_Received                                            119606 non-null  int64
 4   Product_Consumer Loan                                    119606 non-null  uint8
 5   Product_Credit card                                      119606 non-null  uint8
 6   Product_Credit reporting                                 119606 non-null  uint8
 7   Product_Debt collection                                  119606 non-null  uint8
 8   Product_Money transfers                                  119606 non-null  uint8
 9   Product_Mortgage                                         119606 non-null  uint8
 10  Product_Other financial service                          119606 non-null  uint8
 11  Product_Payday loan                                      119606 non-null  uint8
 12  Product_Prepaid card                                     119606 non-null  uint8
 13  Product_Student loan                                     119606 non-null  uint8
 14  Product_Virtual currency                                 119606 non-null  uint8
 15  Submitted via_Fax                                        119606 non-null  uint8
 16  Submitted via_Phone                                      119606 non-null  uint8
 17  Submitted via_Postal mail                                119606 non-null  uint8
 18  Submitted via_Referral                                   119606 non-null  uint8
 19  Submitted via_Web                                        119606 non-null  uint8
 20  Company response to consumer_Closed with explanation     119606 non-null  uint8
 21  Company response to consumer_Closed with monetary relief 119606 non-null  uint8
 22  Company response to consumer_Closed with non-monetary relief 119606 non-null  uint8
 23  Company response to consumer_Closed with relief          119606 non-null  uint8
 24  Company response to consumer_Closed without relief       119606 non-null  uint8
 25  Timely response?_Yes                                     119606 non-null  uint8
dtypes: int64(4), uint8(22)
memory usage: 6.2 MB
```

Scaling the data set

```
X_train = data_train.drop('Consumer disputed?', axis=1)
X_test = data_test
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

Perform feature selection using Principal Component Analysis (PCA) to retain 80% of the information

```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.8)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)
```

Splitting the Data Sets Into X and Y by the dependent and independent variables (data selected by PCA)

```
y_train = data_train['Consumer disputed?']
```

Building the given models and measure their test and validation accuracy

```
from sklearn.tree import DecisionTreeClassifier
models = [
    LogisticRegression(),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    AdaBoostClassifier(),
    GradientBoostingClassifier(),
    KNeighborsClassifier(),
    XGBClassifier()
]

for model in models:
    model.fit(X_train_pca, y_train)
    train_accuracy = model.score(X_train_pca, y_train)
    print(f"Training accuracy for {model.__class__.__name__}: {train_accuracy}")

```

```
Training accuracy for LogisticRegression: 0.787709372648477
Training accuracy for DecisionTreeClassifier: 0.8337309439536245
Training accuracy for RandomForestClassifier: 0.8337253699729662
Training accuracy for AdaBoostClassifier: 0.787709372648477
Training accuracy for GradientBoostingClassifier: 0.7899473258827792
Training accuracy for KNeighborsClassifier: 0.790019787631337
Training accuracy for XGBClassifier: 0.7936595970011984
```

Searching for the most accurate result, uses it and predicts the outcomefor the test file and fills its dispute column so the business team can take some action accordingly

```
from sklearn.metrics import accuracy_score
best_model = None
best_accuracy = 0.0

for model in models:
    model.fit(X_train_pca, y_train)
    train_accuracy = accuracy_score(y_train, model.predict(X_train_pca))
    print(f"Training accuracy for {model.__class__.__name__}: {train_accuracy}")

    if train_accuracy > best_accuracy:
        best_model = model
        best_accuracy = train_accuracy

print(f"Best model: {best_model.__class__.__name__}")
```

```
Training accuracy for LogisticRegression: 0.787709372648477
Training accuracy for DecisionTreeClassifier: 0.8337309439536245
Training accuracy for RandomForestClassifier: 0.8337170090019788
Training accuracy for AdaBoostClassifier: 0.787709372648477
Training accuracy for GradientBoostingClassifier: 0.7899473258827792
Training accuracy for KNeighborsClassifier: 0.790019787631337
Training accuracy for XGBClassifier: 0.7936595970011984
Best model: DecisionTreeClassifier
```

Predicting the Outcome for the Test File: Use the model with the highest accuracy to predict the outcome for the test file and fill its 'Dispute' column

```
data_test['Consumer disputed?'] = best_model.predict(X_test_pca)


```

#Save the Predictions: Save the test file with the predicted dispute column.

```
data_test.to_csv('Consumer_Complaint_test_predicted.csv', index=False)

```