Mini Project 2– ConsumerComplaintResolutionAnalysis

Submitted to: Edureka

Data Science and Machine Learning Internship Program

Submitted by: Umang Parti

Scenario: Product review is the most basic function/factor in resolvingcustomerissues and increasingthe sales growth of any product.Wecan understandtheir mindset toward our service withoutasking each customer.When consumers are unhappywith some aspect of a business, they reach out to customer service and might raise a complaint. Companiestry their best to resolve the complaints that they receive. However, it might not always be possible to appease every customer.So Here, we will analyze data, and with the help ofdifferent algorithms, we are finding the best classification ofcustomercategoryso thatwe can predict our test data

Objective: Use Python libraries such as Pandas for data operations, Seaborn and Matplotlib for data visualization and EDA tasks, Sklearn for model building and performance visualization, andbased on the best model,make a prediction for the test file and save the output.The mainobjective is to predict whetherour customer is disputedor not with the help ofgiven data.

Customers faced some issues and tried to report their problems to customer care. Dispute: This is our target variable based on train data; we have two groups, one with a dispute with the bank and another don't have any issue with the bank. Date received: The day complaint was received. Product: different products offered by the bank (credit cards, debit cards, different types of transaction methods, accounts, locker services, and money-related) Sub-product: loan, insurance, other mortgage options Issue: Complaint of customers Company public response: Company's response to consumer complaint Company: Company name State: State where the customer lives (different state of USA) ZIP code: Where the customer lives Submitted via: Register complaints via different platforms (online web, phone, referral, fax, post mail) Date sent to company: The day complaint was registered Timely response?: Yes/no Consumer disputed?: yes/no (target variable) Complaint ID: unique to each consumer

In [1]:
```
# importing libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Read the Data from the Given excel file. Importing the dataset
consumer_train_df = pd.read_csv(
    r"C:\Users\umang\Desktop\edureka\mini project-2\Consumer_Complaints_train.csv")
consumer_train_df.head()
```

Out[2]:

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	Stat
0	2015-10-14	Credit reporting	NaN	Incorrect information on credit report	Information is not mine	NaN	NaN	Equifax	G.
1	2015-04-26	Bank account or service	Other bank product/service	Deposits and withdrawals	NaN	RE : XXXX XXXX XXXX- PRIVILEGED AND CONFIDENTIA...	NaN	Wells Fargo & Company	G.
2	2013-12-20	Credit card	NaN	Other	NaN	NaN	NaN	Citibank	S
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 ...	FAIR COLLECTIONS & OUTSOURCING, INC.	O
4	2015-01-30	Debt collection	Medical	Disclosure verification of debt	Not given enough info to verify debt	NaN	NaN	HCFS Health Care Financial Services, Inc.	C.

In [3]:

```
consumer_test_df = pd.read_csv(r"C:\Users\umang\Desktop\edureka\mini project-2\Consumer_Compl
consumer_test_df.head()
```

Out[3]:

	Date received	Product	Sub-product	Issue	Sub-issue	Consumer complaint narrative	Company public response	Company	State	
0	2015-01-17	Credit card	NaN	Customer service / Customer relations	NaN	NaN	NaN	Citibank	TX	7
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	7
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	2
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	6
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	2

In [4]:
```
# Check the data type for both data (test file and train file)
consumer_train_df.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
```

In [5]:
```
consumer_test_df.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
```

In [6]:
```
# converting date column from object to datetime
consumer_train_df['Date received'] = pd.to_datetime(consumer_train_df['Date received'])
consumer_train_df['Date sent to company'] = pd.to_datetime(consumer_train_df['Date sent to co
consumer_test_df['Date received'] = pd.to_datetime(consumer_test_df['Date received'])
consumer_test_df['Date sent to company'] = pd.to_datetime(consumer_test_df['Date sent to comp
```

In [7]:
```
# Do missing value analysis and drop columns where more than 25% of data are missing
consumer_train_df.isnull().sum()
```

Out[7]:
```
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
```

In [8]:
```
consumer_test_df.isnull().sum()
```

Out[8]:
```
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
```

In [9]:
```
# dropping columns with more than 25% nulll values (25% of 358810 = 89702)
columns_to_drop_train = ['Sub-product','Sub-issue','Consumer complaint narrative',
                         'Company public response','Tags','Consumer consent provided?']
consumer_train_df.drop(columns_to_drop_train, axis=1,inplace=True)
```

In [10]:
```
# dropping columns with more than 25% nulll values (25% of 119606 = 29901)
columns_to_drop_test = ['Sub-product','Sub-issue','Consumer complaint narrative',
                         'Company public response','Tags','Consumer consent provided?']
consumer_test_df.drop(columns_to_drop_test, axis=1,inplace=True)
```

In [11]:
```
# filling null values with mode
mode_value_state = consumer_train_df['State'].mode()[0]
mode_value_zip = consumer_train_df['ZIP code'].mode()[0]
consumer_train_df['State'].fillna(mode_value_state, inplace=True)
consumer_train_df['ZIP code'].fillna(mode_value_zip, inplace=True)
```

In [12]:
```
mode_value_state_test = consumer_test_df['State'].mode()[0]
mode_value_zip_test = consumer_test_df['ZIP code'].mode()[0]
consumer_test_df['State'].fillna(mode_value_state_test, inplace=True)
consumer_test_df['ZIP code'].fillna(mode_value_zip_test, inplace=True)
```

In [13]:
```
# we have successfully handled all null values
print(consumer_train_df.isnull().sum())
print(consumer_test_df.isnull().sum())
```

```
Date received                   0
Product                         0
Issue                           0
Company                         0
State                           0
ZIP code                        0
Submitted via                   0
Date sent to company            0
Company response to consumer    0
Timely response?                0
Consumer disputed?              0
Complaint ID                    0
dtype: int64
Date received                   0
Product                         0
Issue                           0
Company                         0
State                           0
ZIP code                        0
Submitted via                   0
Date sent to company            0
Company response to consumer    0
Timely response?                0
Complaint ID                    0
dtype: int64
```

In [14]:
```
# Extracting Day, Month,and Year from Date Received Column and
# create new fields for a month, year,and day
consumer_train_df['year']= consumer_train_df['Date received'].dt.year
consumer_train_df['month']=consumer_train_df['Date received'].dt.month
consumer_train_df['day']=consumer_train_df['Date received'].dt.day
```

In [15]:
```
consumer_test_df['year']=consumer_test_df['Date received'].dt.year
consumer_test_df['month']=consumer_test_df['Date received'].dt.month
consumer_test_df['day']=consumer_test_df['Date received'].dt.day
```

In [16]:
```
# Calculate the Number of Days the Complaint was with the Company as a new field “Days held”
consumer_train_df['Days Held'] = (consumer_train_df['Date sent to company'] -
                                  consumer_train_df['Date received']).dt.days
consumer_test_df['Days Held'] = (consumer_test_df['Date sent to company'] -
                                 consumer_test_df['Date received']).dt.days
```

In [17]:
```
# with the help of the days we calculated above,create a newfield 'Week_Received'where we
# calculate the week based on the day of receiving.
consumer_train_df['Week_Received'] = (consumer_train_df['Date received'].dt.week +
                                      consumer_train_df['Days Held'] // 7)
consumer_test_df['Week_Received'] = (consumer_test_df['Date received'].dt.week +
                                     consumer_test_df['Days Held'] // 7)
```

```
C:\Users\umang\AppData\Local\Temp\ipykernel_6848\2606253115.py:3: FutureWarning: Series.dt.we
ekofyear and Series.dt.week have been deprecated. Please use Series.dt.isocalendar().week ins
tead.
  consumer_train_df['Week_Received'] = (consumer_train_df['Date received'].dt.week +
C:\Users\umang\AppData\Local\Temp\ipykernel_6848\2606253115.py:5: FutureWarning: Series.dt.we
ekofyear and Series.dt.week have been deprecated. Please use Series.dt.isocalendar().week ins
tead.
  consumer_test_df['Week_Received'] = (consumer_test_df['Date received'].dt.week +
```

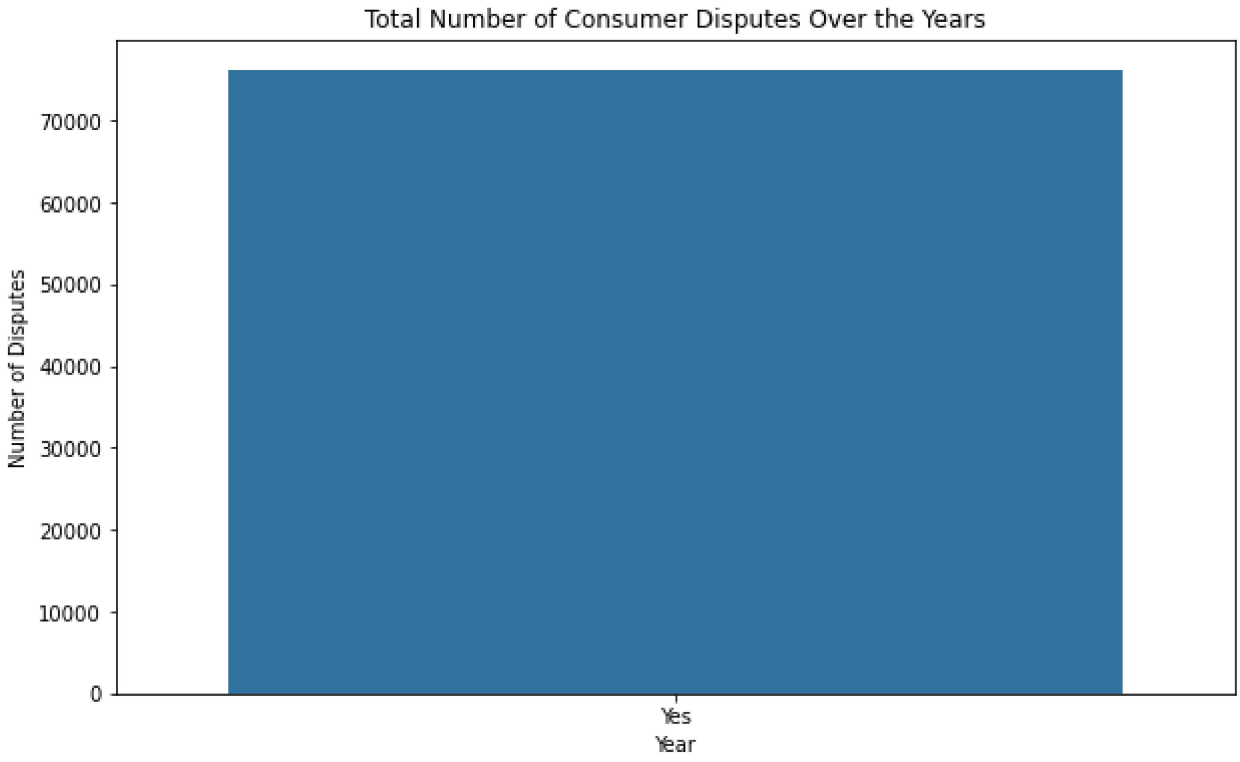
In [18]:
```
# Drop "Date Received","Date Sent to Company","ZIP Code", "Complaint ID"fields
columns = ['Date received','Date sent to company','ZIP code','Complaint ID']
consumer_train_df.drop(columns, axis=1,inplace=True)
consumer_test_df.drop(columns, axis=1,inplace=True)
```

In [19]:
```
# store data of disputed people into the “disputed_cons” variable for future tasks
disputed_cons = consumer_train_df[consumer_train_df['Consumer disputed?'] == 'Yes']
disputed_cons.head()
```

Out[19]:

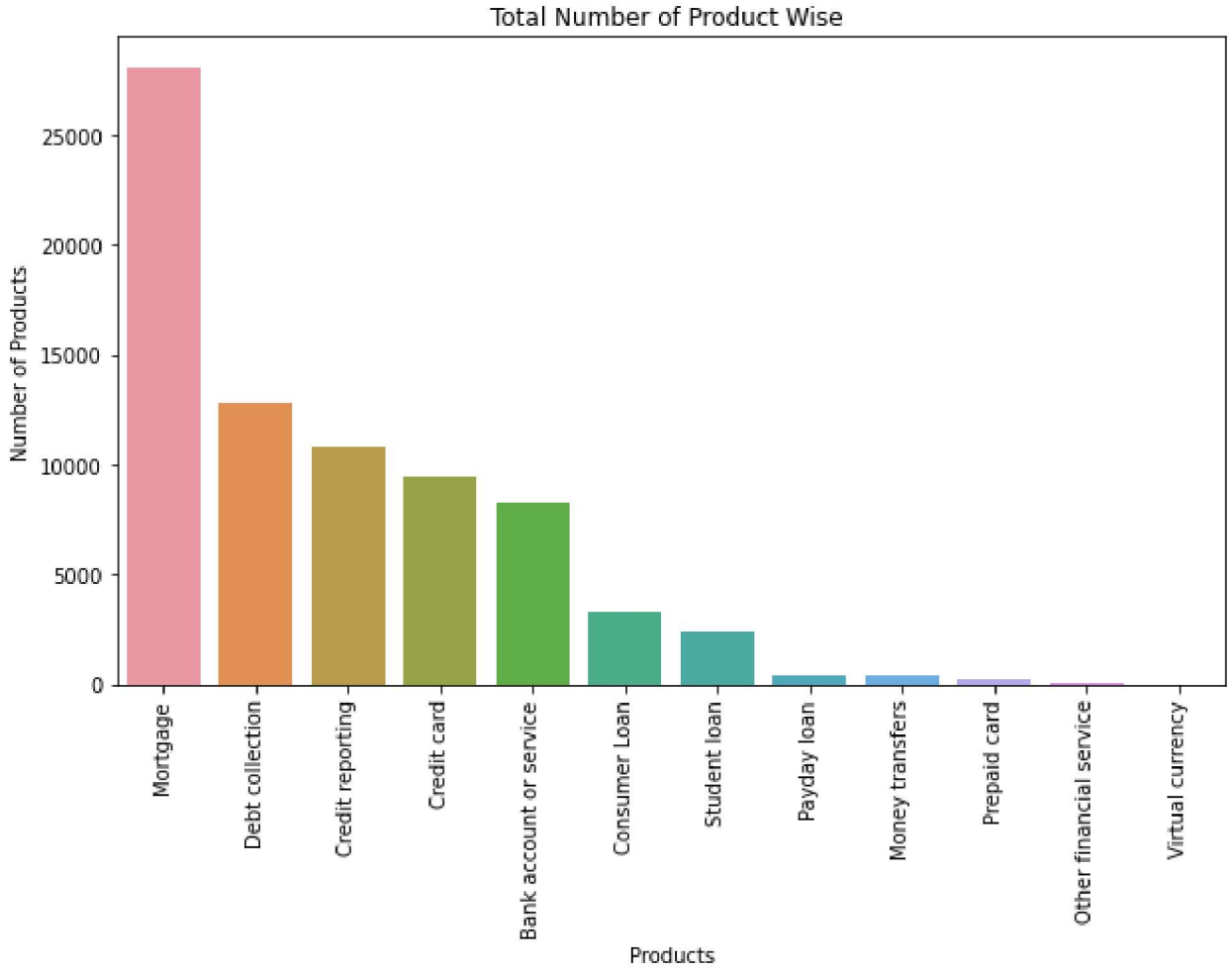
	Product	Issue	Company	State	Submitted via	Company response to consumer	Timely response?	Consumer disputed?	year	month	day	I I
1	Bank account or service	Deposits and withdrawals	Wells Fargo & Company	GA	Web	Closed with explanation	Yes	Yes	2015	4	26	
4	Debt collection	Disclosure verification of debt	HCFS Health Care Financial Services, Inc.	CA	Web	Closed with explanation	Yes	Yes	2015	1	30	
5	Credit card	APR or interest rate	TD Bank US Holding Company	FL	Web	Closed with explanation	Yes	Yes	2014	1	10	
7	Credit card	Payoff process	Capital One	IL	Web	Closed with explanation	Yes	Yes	2015	12	7	
8	Bank account or service	Deposits and withdrawals	Citizens Financial Group, Inc.	PA	Web	Closed with relief	Yes	Yes	2012	4	5	

In [20]:
```
# Plot bar graph of thetotal no of disputes of consumers with the help ofseaborn
plt.figure(figsize=(10, 6))
sns.countplot(x='Consumer disputed?', data=disputed_cons)
plt.xlabel('Year')
plt.ylabel('Number of Disputes')
plt.title('Total Number of Consumer Disputes Over the Years')
plt.show()
```

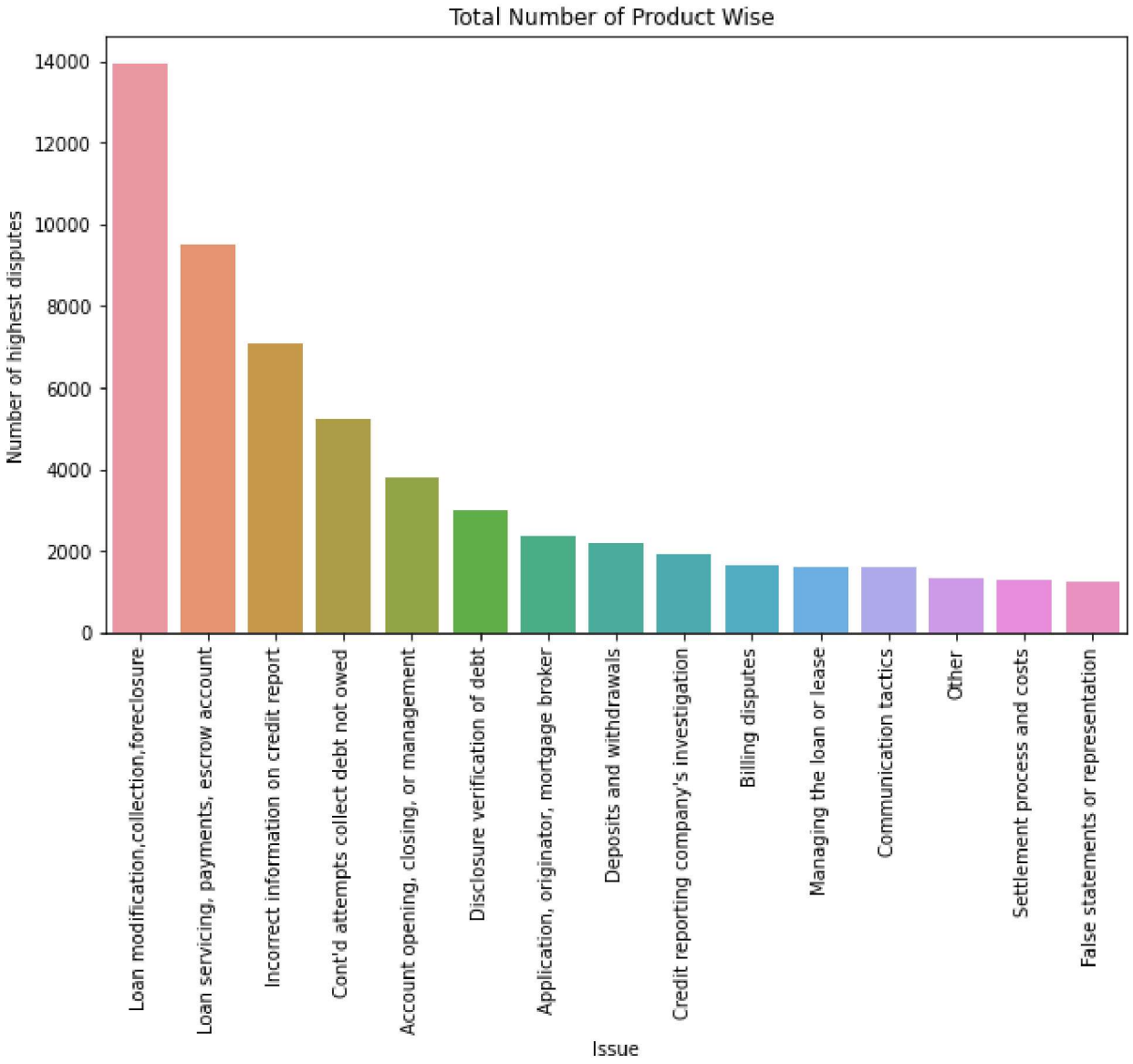


In [21]:
```
# Plot bar graph of the total no of disputes products-wise with the help ofseaborn
plt.figure(figsize=(10, 6))
sns.countplot(x='Product', data=disputed_cons, order=disputed_cons['Product'].value_counts().
plt.xlabel('Products')
plt.ylabel('Number of Products')
```

```
plt.title('Total Number of Product Wise ')
plt.xticks(rotation=90, ha='center')
plt.show()
```

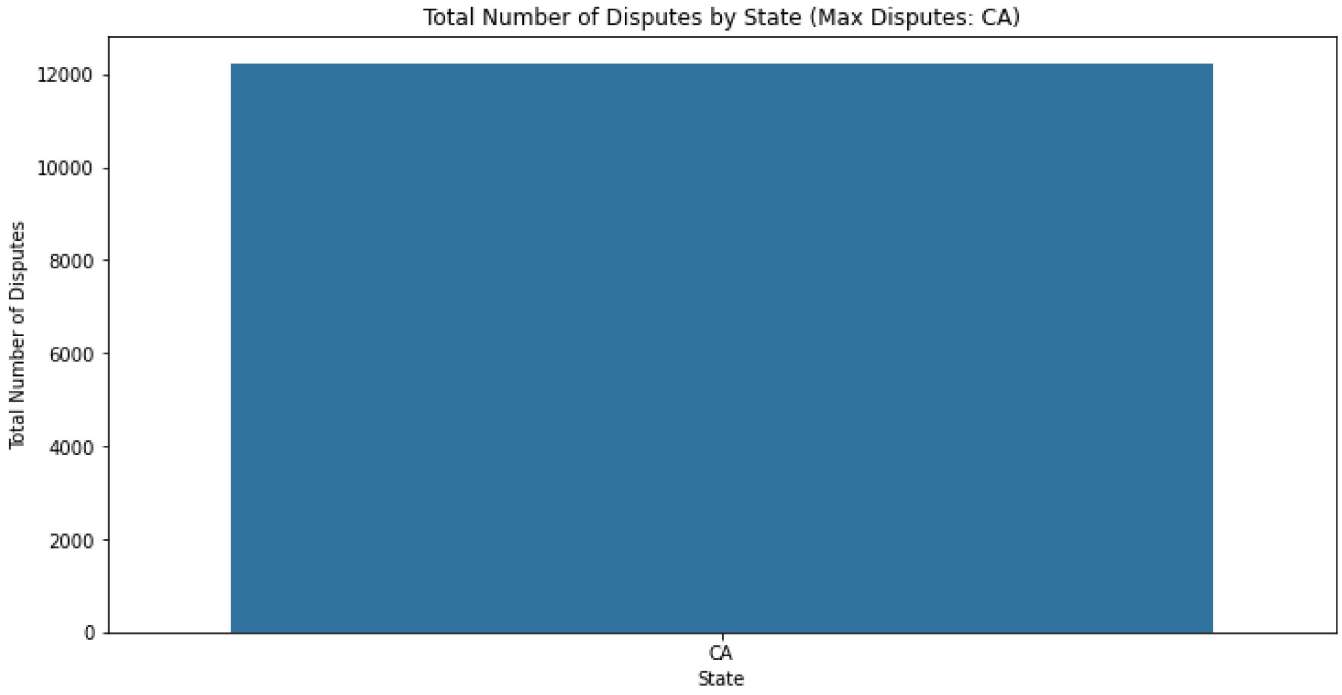


In [22]:
```
# Plot bar graph of the total no of disputes with Top Issues by Highest Disputes
top_issue = disputed_cons['Issue'].value_counts().nlargest(15)
top_issue_df = pd.DataFrame({'Issue': top_issue.index, 'Count': top_issue.values})

plt.figure(figsize=(10, 6))
sns.barplot(x='Issue', y='Count', data=top_issue_df)
plt.xlabel('Issue')
plt.ylabel('Number of highest disputes')
plt.title('Total Number of Product Wise ')
plt.xticks(rotation=90, ha='center')
plt.show()
```

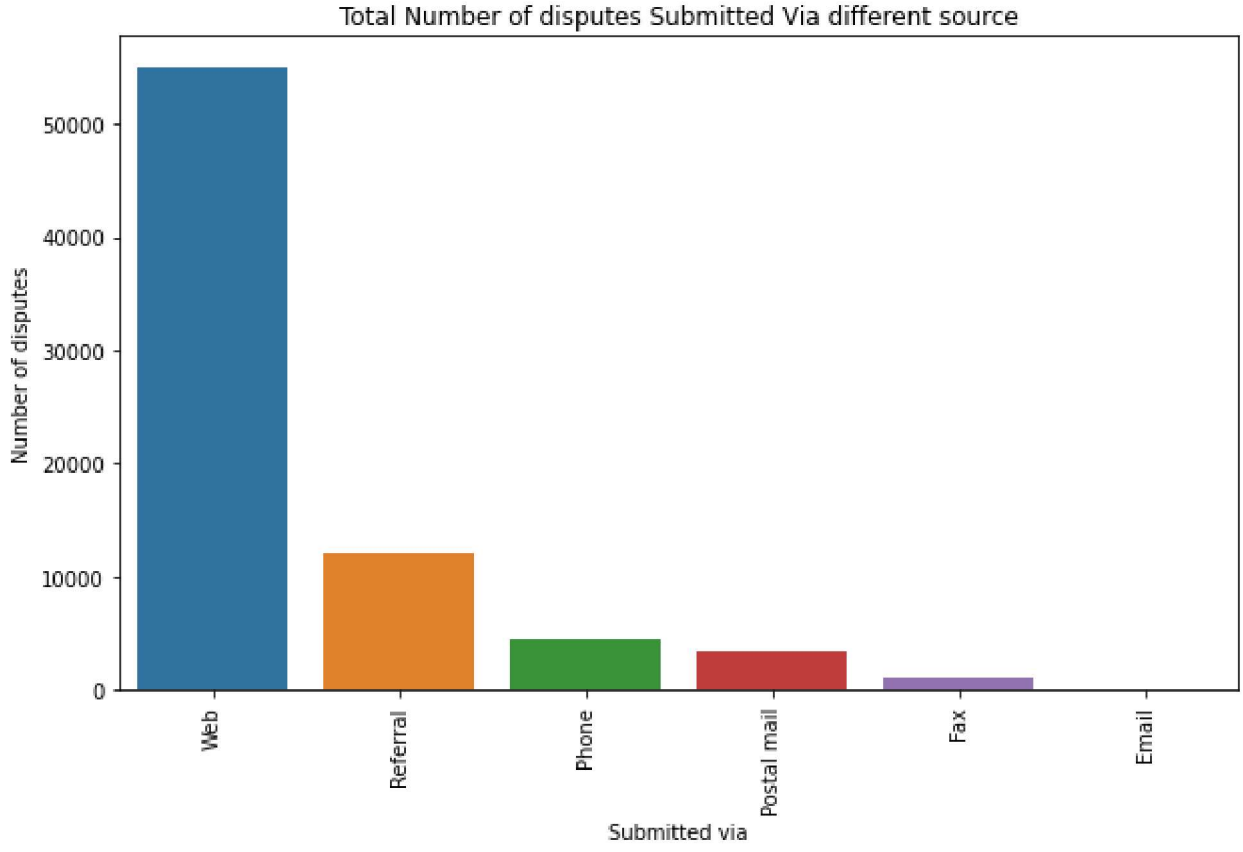


In [23]:

```
# Plot bar graph of the total no of disputes by State with Maximum Disputes
state_with_max_disputes = disputed_cons['State'].value_counts().idxmax()
total_disputes_by_state = disputed_cons.groupby('State').size().reset_index(name='Total Dispu
max_disputes_state_df = total_disputes_by_state[total_disputes_by_state['State'] ==
                                                state_with_max_disputes]

plt.figure(figsize=(12, 6))
sns.barplot(x='State', y='Total Disputes', data=max_disputes_state_df)
plt.xlabel('State')
plt.ylabel('Total Number of Disputes')
plt.title(f'Total Number of Disputes by State (Max Disputes: {state_with_max_disputes})')
plt.show()
```

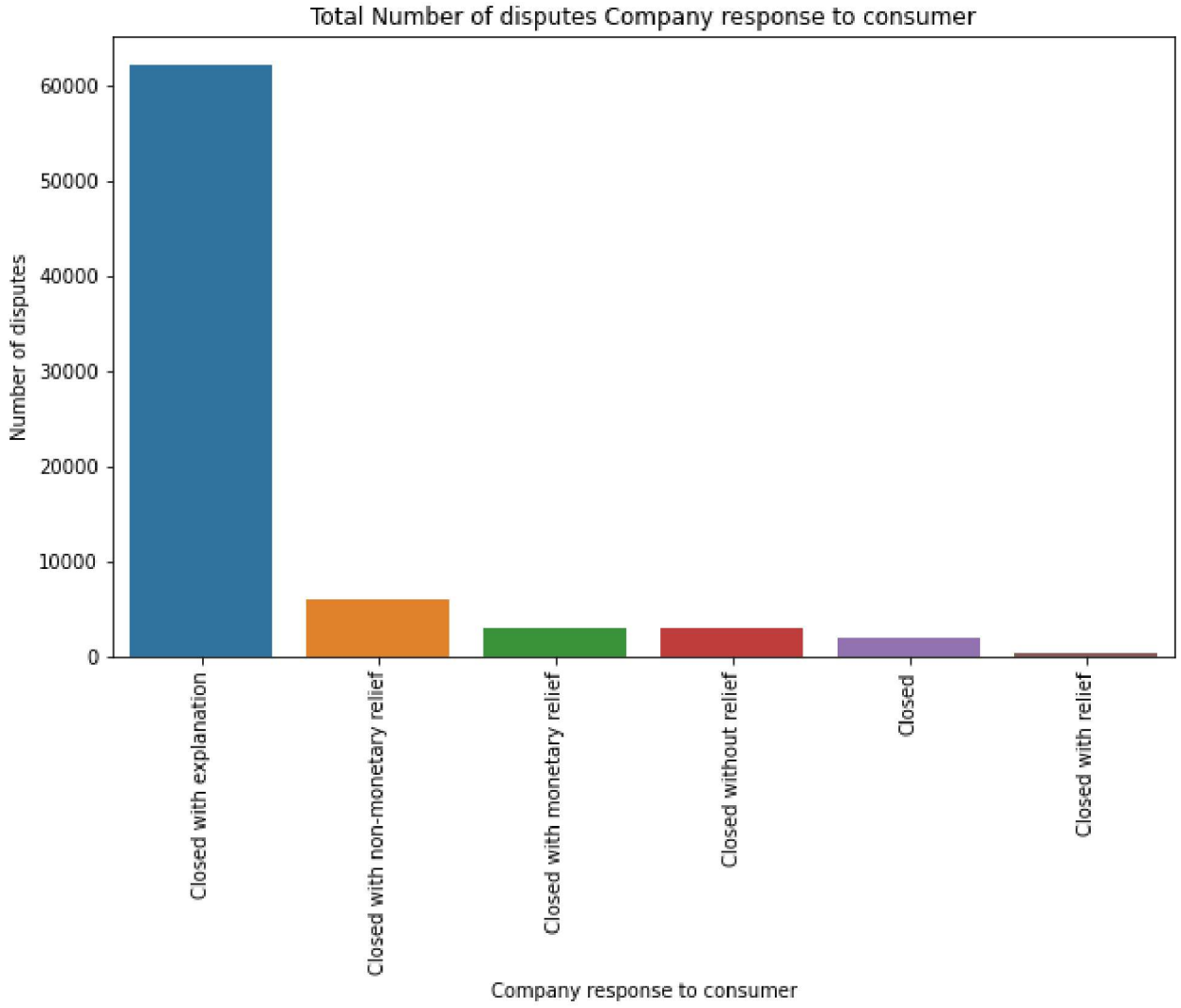


In [24]:
```
# Plotbar graph of the total no of disputes Submitted Via different source
top_submitted = disputed_cons['Submitted via'].value_counts().nlargest(15)
top_submitted_df = pd.DataFrame({'Submitted via': top_submitted.index, 'Count': top_submitted

plt.figure(figsize=(10, 6))
sns.barplot(x='Submitted via', y='Count', data=top_submitted_df)
plt.xlabel('Submitted via')
plt.ylabel('Number of disputes')
plt.title('Total Number of disputes Submitted Via different source')
plt.xticks(rotation=90, ha='center')
plt.show()
```

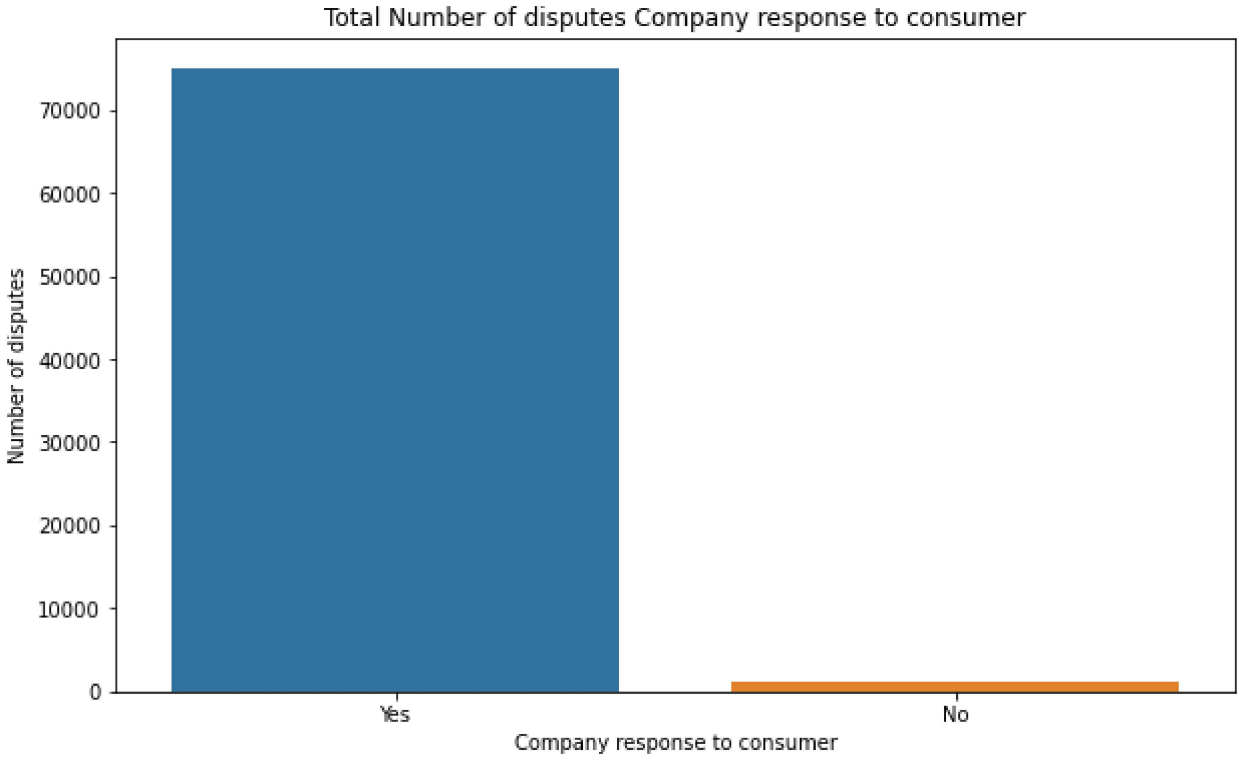


In [25]:
```
# Plot bar graph of the total no of disputes where the Company's Response to the Complaints
top_response = disputed_cons['Company response to consumer'].value_counts().nlargest(15)
top_response_df = pd.DataFrame({'Company response to consumer': top_response.index,
                                'Count': top_response.values})

plt.figure(figsize=(10, 6))
```

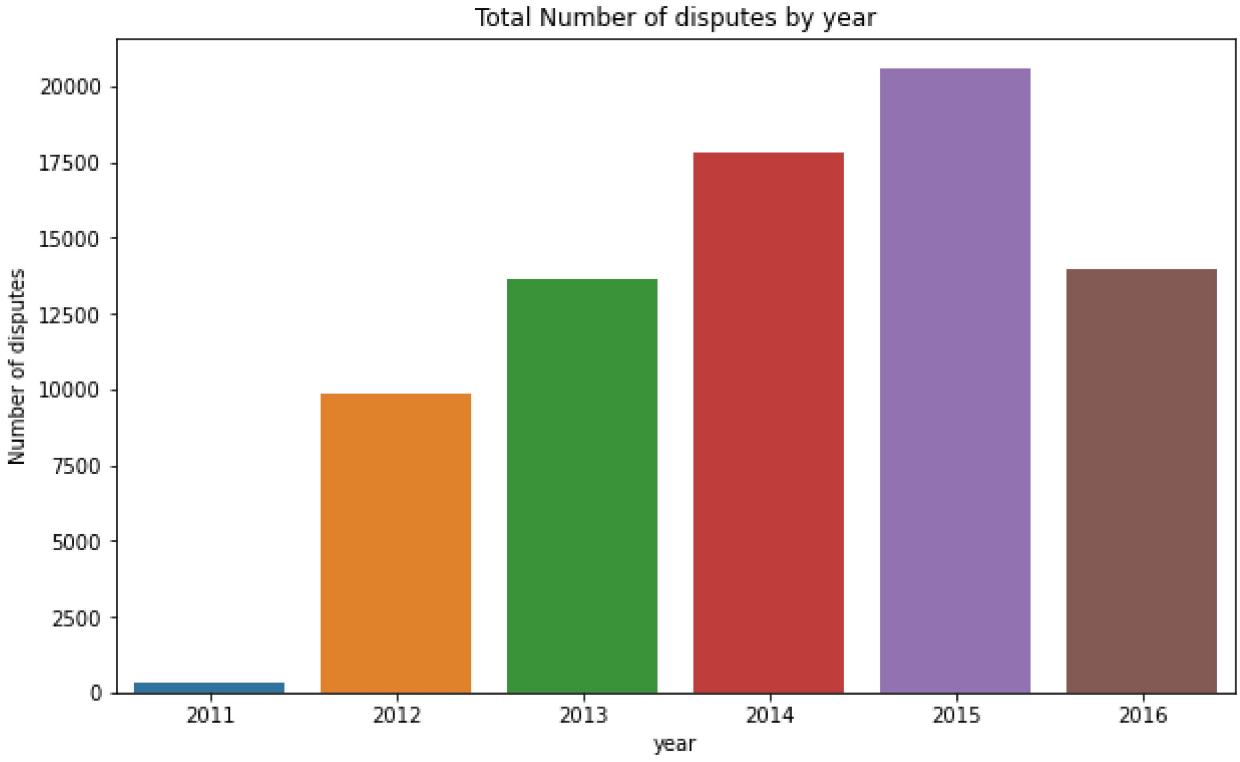
```
sns.barplot(x='Company response to consumer', y='Count', data=top_response_df)
plt.xlabel('Company response to consumer')
plt.ylabel('Number of disputes')
plt.title('Total Number of disputes Company response to consumer')
plt.xticks(rotation=90, ha='center')
plt.show()
```



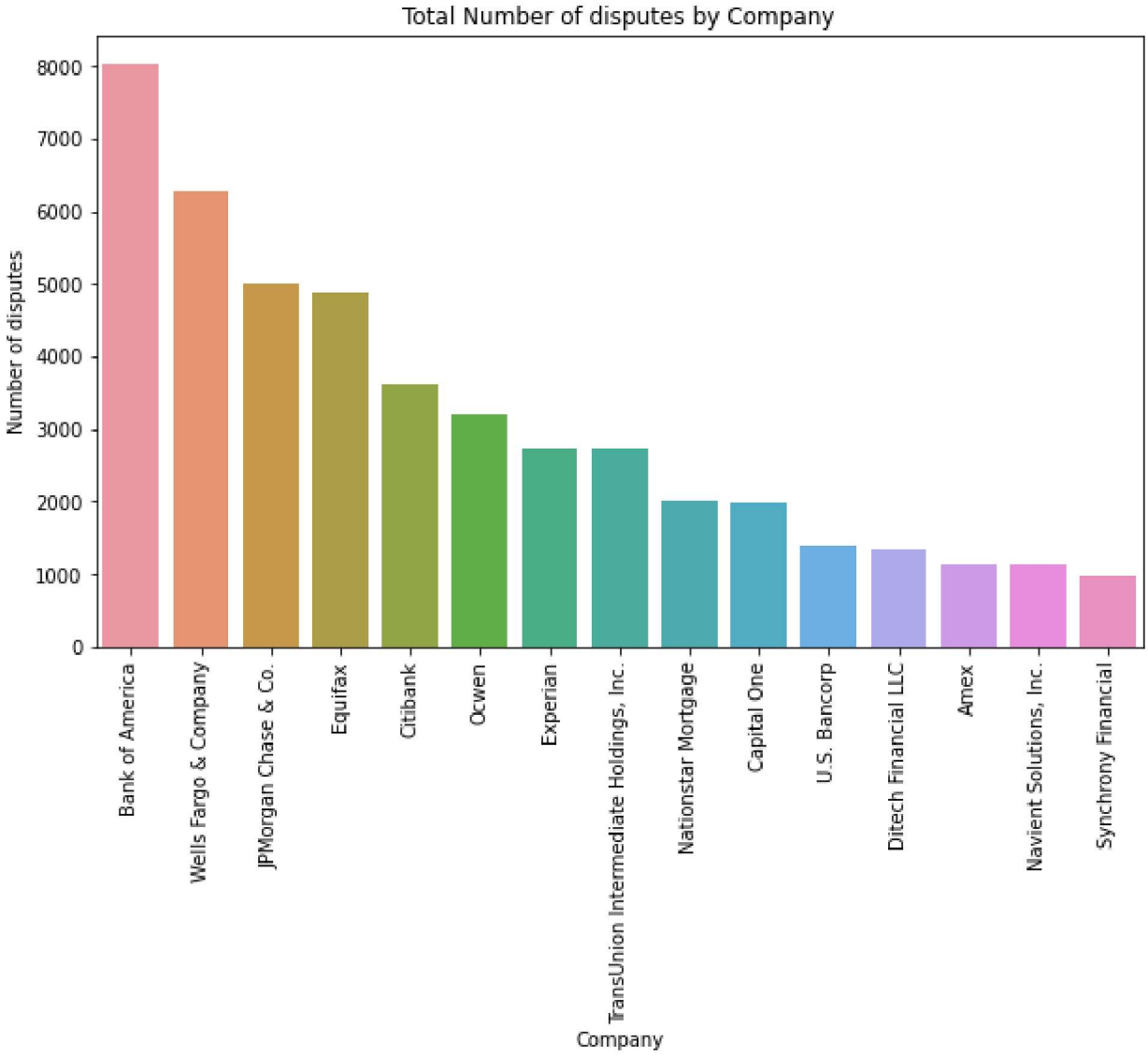
In [26]:
```
# Plot bar graph ofthe total no of disputes.Whether there are Disputes Instead of Timely Resp
plt.figure(figsize=(10, 6))
sns.countplot(x='Timely response?', data=disputed_cons)
plt.xlabel('Company response to consumer')
plt.ylabel('Number of disputes')
plt.title('Total Number of disputes Company response to consumer')
plt.show()
```



In [27]:
```
# Plot bar graph of the total no of disputes over Year Wise Complaints
plt.figure(figsize=(10, 6))
sns.countplot(x='year', data=disputed_cons)
plt.xlabel('year')
plt.ylabel('Number of disputes')
plt.title('Total Number of disputes by year')
plt.show()
```



In [28]:
```
# Plot bar graph of Top Companies with Highest Complaints
top_company = disputed_cons['Company'].value_counts().nlargest(15)
top_company_df = pd.DataFrame({'Company': top_company.index, 'Count': top_company.values})

plt.figure(figsize=(10, 6))
sns.barplot(x='Company', y='Count', data=top_company_df)
plt.xlabel('Company')
plt.ylabel('Number of disputes')
plt.title('Total Number of disputes by Company')
plt.xticks(rotation=90, ha='center')
plt.show()
```



In [29]:
```
# Converte all negative days held to zero(time taken by the authority that can't be negative)
disputed_cons['Days Held'] = disputed_cons['Days Held'].apply(lambda x: max(0, x))
disputed_cons['Days Held'].unique()
```

```
C:\Users\umang\AppData\Local\Temp\ipykernel_6848\1912939366.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  disputed_cons['Days Held'] = disputed_cons['Days Held'].apply(lambda x: max(0, x))
```

Out[29]:
```
array([  0,   5,   3,   2,   4,   1,   6,  18,   7,  20,   9,  10,  37,
        19,   8, 120,  11,  22,  17,  13,  27,  30,  53,  21, 104,  67,
        16,  55,  84,  48,  29,  44,  71,  70,  12,  15, 113,  14,  28,
        85,  50,  32,  23,  24,  33,  45,  51,  39,  34, 196,  38, 210,
        35, 205,  94, 200, 127,  40, 154,  25,  43,  36,  63,  54,  41,
       156,  74,  42,  26,  64, 102,  76,  31, 380,  49,  66, 202, 117,
       177, 197,  56,  95,  86, 105, 175,  78,  73,  52, 143, 191, 153,
        90,  98,  83, 182,  97,  62, 224,  57, 161,  58,  69,  91, 203,
        47,  68, 111, 254,  72, 176, 135, 308,  87, 186,  46, 204,  93,
       244, 140, 124, 189,  61, 141, 134, 218, 233, 170, 263,  81, 178,
       201, 230, 109, 126,  75, 180, 265, 279, 152, 160, 165, 147, 172,
       211, 100,  60, 215,  89, 136, 258,  82, 193, 119, 133, 121, 345,
       188,  80, 217,  59, 214, 216, 343,  96, 171, 132, 183, 213, 253,
        65, 192, 273, 116, 305, 145, 199, 115, 181, 246, 462,  99, 122,
       325, 212, 232, 128, 103, 167, 231, 463, 169, 157, 198, 151, 150,
       163, 378, 123, 129, 158, 190, 173, 131, 207, 226, 252, 267, 289,
       220, 138, 209, 108, 187, 227,  77, 179, 168, 206,  79, 101, 208,
       159, 299, 155, 376, 324, 162, 236, 315, 118, 125, 229, 148, 334,
       251, 436, 194, 240, 139, 184, 107, 290, 631, 174, 144, 219, 545],
      dtype=int64)
```

In [30]:
```
drop_columns=['Company','State','year','Days Held','Issue']
```

In [31]:
```
# Drop UnnecessaryColumns for the Model Buildinglike:'Company', 'State', 'Year_Received', 'Da
consumer_train_df.drop(drop_columns, axis=1, inplace=True)
```

In [32]:
```
consumer_test_df.drop(drop_columns, axis=1, inplace=True)
```

In [33]:
```
# Change Consumer Disputed Column to 0 and 1(yes to 1, and no to 0)
consumer_train_df['Consumer disputed?'] = consumer_train_df['Consumer disputed?'].map(
    {'Yes': 1, 'No': 0})
```

In [34]:
```
# Create Dummy Variables for categoricalfeaturesand concat with the original data frame
# like: 'Product,''Submitted via,''Company response to consumer,''Timely response?'
categorical_columns = ['Product', 'Submitted via', 'Company response to consumer', 'Timely re
dummy_df_train = pd.get_dummies(consumer_train_df[categorical_columns])
consumer_train_df = pd.concat([consumer_train_df, dummy_df_train], axis=1)
consumer_train_df = consumer_train_df.drop(categorical_columns, axis=1)
```

In [35]:
```
dummy_df_test = pd.get_dummies(consumer_test_df[categorical_columns])
consumer_test_df = pd.concat([consumer_test_df, dummy_df_test], axis=1)
consumer_test_df = consumer_test_df.drop(categorical_columns, axis=1)
```

In [36]:
```
consumer_train_df.head()
```

Out[36]:

	Consumer disputed?	month	day	Week_Received	Product_Bank account or service	Product_Consumer Loan	Product_Credit card	Product_Credit reporting
0	0	10	14	42	0	0	0	1
1	1	4	26	17	1	0	0	0
2	0	12	20	53	0	0	1	0
3	0	3	3	9	0	0	0	0
4	1	1	30	5	0	0	0	0

5 rows × 30 columns

In [37]:
```
consumer_test_df.head()
```

Out[37]:

	month	day	Week_Received	Product_Bank account or service	Product_Consumer Loan	Product_Credit card	Product_Credit reporting	Product_Del collectic
0	1	17	3	0	0	1	0	
1	6	22	25	0	1	0	0	
2	9	4	36	0	0	1	0	
3	5	17	20	0	1	0	0	
4	7	7	27	0	0	0	0	

5 rows × 29 columns

In [38]:
```
# Scaling the Data Sets (note:discard dependent variable before doing standardization)and
# Makefeature Selection with the help ofPCAup to 80% of the information.
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

In [39]:
```
# we take just the train dataset as it contains label ie consumer disrupted column
# Splitting the Data Sets Into X and Y by the dependent and independent variables
# (data selected by PCA)
x = consumer_train_df.drop('Consumer disputed?', axis=1)
y = consumer_train_df['Consumer disputed?']
```

In [40]:
```
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)
```

In [41]:
```
pca = PCA(n_components=0.8)
x_pca = pca.fit_transform(x_scaled)
```

In [42]:
```
print(f"Number of selected features: {x_pca.shape[1]}")
print(f"Percentage of retained information: {round(sum(pca.explained_variance_ratio_) * 100,
```

```
Number of selected features: 18
Percentage of retained information: 81.28%
```

In [43]:
```
selected_features = pd.DataFrame(x_pca, columns=[f'PC{i}' for i in range(1, x_pca.shape[1] +
final_data = pd.concat([selected_features, y], axis=1)
final_data.head()
```

Out[43]:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
0	1.145045	-1.383511	1.376860	-0.118538	0.824420	-0.148605	0.966063	-0.612357	0.115575	0.179778
1	-0.225579	-0.024005	-0.383479	-0.521998	-1.150206	-1.209781	-0.104197	-1.905336	-0.135968	-0.238240
2	-0.115953	-1.122111	0.613870	3.843353	0.220286	-0.698536	-0.428130	3.167303	1.060380	1.020582
3	-1.024260	1.227007	-0.561448	-0.842868	0.159377	-0.966556	-1.396899	-0.074157	-1.742122	0.620138
4	0.813587	-0.002474	-1.324258	-2.020335	-0.693276	-0.841795	-1.304882	0.151414	-1.139069	-0.601536

In [44]:
```
# train test split
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 21)
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[44]:
```
((251167, 29), (107643, 29), (251167,), (107643,))
```

In [45]:
```
# Build given models and measure their test and validation accuracy:
# Logistic Regression
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
logreg=LogisticRegression(solver="liblinear")
```

In [46]:
```
logreg.fit(x_train,y_train)
y_pred_test_logreg = logreg.predict(x_test)
y_pred_train_logreg = logreg.predict(x_train)
```

In [47]:
```
print("Accuracy Score of Logistic Regression Model on Test",
      accuracy_score(y_test, y_pred_test_logreg))
print("Accuracy Score of Logistic Regression Model on Train",
      accuracy_score(y_train, y_pred_train_logreg))
```

```
Accuracy Score of Logistic Regression Model on Test 0.7880865453396877
Accuracy Score of Logistic Regression Model on Train 0.7875477272093866
```

In [48]:
```
# DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
dt_model = DecisionTreeClassifier()
```

In [49]:
```
dt_model.fit(x_train,y_train)
y_pred_test_dt = dt_model.predict(x_test)
y_pred_train_dt = dt_model.predict(x_train)
```

In [50]:
```
print("Accuracy Score of Decision Tree Model on Test",
      accuracy_score(y_test, y_pred_test_dt))
print("Accuracy Score of Decision Tree Model on Train",
      accuracy_score(y_train, y_pred_train_dt))
```

```
Accuracy Score of Decision Tree Model on Test 0.7673699172263873
Accuracy Score of Decision Tree Model on Train 0.8162218762815179
```

In [51]:
```
# RandomForestClassifier
from sklearn.ensemble import RandomForestClassifier
rfc_model = RandomForestClassifier(n_estimators=50)
```

In [52]:
```
rfc_model.fit(x_train,y_train)
y_pred_test_rfc = rfc_model.predict(x_test)
y_pred_train_rfc = rfc_model.predict(x_train)
```

In [53]:
```
print("Accuracy Score of Random Forest Classifier Model on Test",
      accuracy_score(y_test, y_pred_test_rfc))
print("Accuracy Score of Random Forest Classifier Model on Train",
      accuracy_score(y_train, y_pred_train_rfc))
```

```
Accuracy Score of Random Forest Classifier Model on Test 0.7687355424876676
Accuracy Score of Random Forest Classifier Model on Train 0.8159989170551863
```

In [54]:
```
# AdaBoostClassifier
from sklearn.ensemble import AdaBoostClassifier
ada_model = AdaBoostClassifier(n_estimators=80)
```

In [55]:
```
ada_model.fit(x_train,y_train)
y_pred_test_abc = ada_model.predict(x_test)
y_pred_train_abc = ada_model.predict(x_train)
```

In [56]:
```
print("Accuracy Score of Ada Boost Classifier Model on Test",
      accuracy_score(y_test, y_pred_test_abc))
print("Accuracy Score of Ada Boost Classifier Model on Train",
      accuracy_score(y_train, y_pred_train_abc))
```

```
Accuracy Score of Ada Boost Classifier Model on Test 0.7880865453396877
Accuracy Score of Ada Boost Classifier Model on Train 0.7875517086241425
```

In [57]:
```
# GradientBoostingClassifier
from sklearn.ensemble import GradientBoostingClassifier
gb_model = GradientBoostingClassifier(n_estimators=20,learning_rate=0.02)
```

In [58]:
```
gb_model.fit(x_train,y_train)
y_pred_test_gbc = gb_model.predict(x_test)
y_pred_train_gbc = gb_model.predict(x_train)
```

In [59]:
```
print("Accuracy Score of Gradient Boosting Classifier Model on Test",
      accuracy_score(y_test, y_pred_test_gbc))
print("Accuracy Score of Gradient Boosting Classifier Model on Train",
      accuracy_score(y_train, y_pred_train_gbc))
```

```
Accuracy Score of Gradient Boosting Classifier Model on Test 0.7880865453396877
Accuracy Score of Gradient Boosting Classifier Model on Train 0.7875477272093866
```

In [60]:
```
# XGBClassifier
from xgboost.sklearn import XGBClassifier
```

```
xgb_model = XGBClassifier(n_estimators=20, objective="multi:softmax", num_class=2)
```

In [61]:
```
xgb_model.fit(x_train,y_train)
y_pred_test_xgb = xgb_model.predict(x_test)
y_pred_train_xgb = xgb_model.predict(x_train)
```

In [62]:
```
print("Accuracy Score of XG Boost Classifier Model on Test",
      accuracy_score(y_test, y_pred_test_xgb))
print("Accuracy Score of XG Boost Classifier Model on Train",
      accuracy_score(y_train, y_pred_train_xgb))
```

```
Accuracy Score of XG Boost Classifier Model on Test 0.7881794450173258
Accuracy Score of XG Boost Classifier Model on Train 0.7877467979471825
```

In [63]:
```
# Whoever gives the most accurate result uses it and predicts the outcome for the test file a
# fills its dispute column so the business team can take some action accordingly.
# our highest accuracy is 78% for both test and train dataset for Logreg, Ada boost, Gradient
# XG boost classifier
x_test['Consumer disputed?'] = y_pred_test_logreg
print(x_test.head())
```

```
        month  day  Week_Received  Product_Bank account or service  \
349630     11    6             45                                0
123755      4    2             14                                0
85288      12   20             51                                0
282735     11   28             48                                0
92036       6    6             23                                0

        Product_Consumer Loan  Product_Credit card  Product_Credit reporting  \
349630                      0                    0                         1
123755                      0                    1                         0
85288                       0                    0                         0
282735                      0                    0                         0
92036                       0                    0                         0

        Product_Debt collection  Product_Money transfers  Product_Mortgage  \
349630                        0                        0                 0
123755                        0                        0                 0
85288                         0                        0                 1
282735                        0                        0                 0
92036                         0                        0                 1

        ...  Submitted via_Web  Company response to consumer_Closed  \
349630  ...                  1                                   0
123755  ...                  0                                   0
85288   ...                  1                                   0
282735  ...                  1                                   0
92036   ...                  0                                   0

        Company response to consumer_Closed with explanation  \
349630                                                  1
123755                                                  1
85288                                                   1
282735                                                  1
92036                                                   1

        Company response to consumer_Closed with monetary relief  \
349630                                                  0
123755                                                  0
85288                                                   0
282735                                                  0
92036                                                   0

        Company response to consumer_Closed with non-monetary relief  \
349630                                                  0
123755                                                  0
85288                                                   0
282735                                                  0
92036                                                   0

        Company response to consumer_Closed with relief  \
349630                                                0
123755                                                0
85288                                                 0
282735                                                0
92036                                                 0

        Company response to consumer_Closed without relief  \
349630                                                  0
123755                                                  0
85288                                                   0
282735                                                  0
92036                                                   0

        Timely response?_No  Timely response?_Yes  Consumer disputed?
349630                    0                     1                   0
123755                    0                     1                   0
85288                     0                     1                   0
282735                    0                     1                   0
92036                     0                     1                   0
```

```
[5 rows x 30 columns]
```

In [ ]:
```
#.....................................................................................
```