In [1]:

*# import libraries*  
**import** **os**   
**import** **re**   
**import** **pandas** **as** **pd**

In [2]:

**import** **matplotlib.pyplot** **as** **plt**  
**import** **pandas** **as** **pd**  
**import** **numpy** **as** **np**  
**import** **seaborn** **as** **sns**  
**import** **plotly**  
**import** **plotly.offline** **as** **pyoff**  
  
**from** **plotly.offline** **import** init\_notebook\_mode, iplot, plot  
**import** **plotly.graph\_objs** **as** **go**  
  
%**matplotlib** inline

In [3]:

*# use nltk libraries for text preprocessing*  
**import** **nltk**  
**from** **textblob** **import** TextBlob  
**from** **wordcloud** **import** WordCloud

In [4]:

init\_notebook\_mode(connected=**True**)

In [5]:

os.getcwd()

Out[5]:

'C:\\Users\\Sowmya\\Downloads\\PHD\\End\_To\_End\_Submission'

**Importing train Grievance Data**[**¶**](#gjdgxs)

In [6]:

GD\_train = pd.read\_csv('GrievancesData\_Train.csv')

In [7]:

GD\_train.shape

Out[7]:

(53680, 10)

**Checking for null values**[**¶**](#30j0zll)

In [8]:

GD\_train.isnull().sum()

Out[8]:

GrievanceID 0  
BankID 0  
State 121  
DateOfGrievance 0  
Grievance\_Category 0  
GrievanceDescription 0  
LineOfBusiness 0  
ResolutionComments 0  
Disputed 0  
DateOfResolution 0  
dtype: int64

**Dropping null values**[**¶**](#1fob9te)

In [9]:

GD\_train.dropna(inplace=**True**)

In [10]:

GD\_train.isnull().sum()

Out[10]:

GrievanceID 0  
BankID 0  
State 0  
DateOfGrievance 0  
Grievance\_Category 0  
GrievanceDescription 0  
LineOfBusiness 0  
ResolutionComments 0  
Disputed 0  
DateOfResolution 0  
dtype: int64

In [11]:

GD\_MM\_DF = GD\_train[["GrievanceID","BankID","State","DateOfGrievance","Grievance\_Category","LineOfBusiness","ResolutionComments","Disputed","DateOfResolution"]]

In [12]:

GD\_MM\_DF.shape

Out[12]:

(53559, 9)

**Extrating number of days from Date of grievance and date of resolution**[**¶**](#3znysh7)

In [13]:

GD\_MM\_DF['DateOfGrievance'] = GD\_MM\_DF.DateOfGrievance.astype('datetime64[ns]')

C:\Users\Sowmya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:1: 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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

In [14]:

GD\_MM\_DF['DateOfResolution'] = GD\_MM\_DF.DateOfResolution.astype('datetime64[ns]')

C:\Users\Sowmya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:1: 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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

In [15]:

GD\_MM\_DF['Days'] = GD\_MM\_DF.apply(**lambda** row: row.DateOfResolution - row.DateOfGrievance, axis=1)

In [16]:

GD\_MM\_DF.head()

Out[16]:

|  | **GrievanceID** | **BankID** | **State** | **DateOfGrievance** | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **DateOfResolution** | **Days** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512411 | Bank5279 | State31 | 2016-01-19 | Settlement process and costs | Mortgage | Closed with explanation | No | 2016-01-30 | 11 days |
| **1** | GID512412 | Bank5287 | State26 | 2016-01-19 | Application, originator, mortgage broker | Mortgage | Closed with explanation | Yes | 2016-02-10 | 22 days |
| **2** | GID512413 | Bank5286 | State14 | 2016-01-19 | Billing disputes | Credit card | Closed with monetary relief | Yes | 2016-02-10 | 22 days |
| **3** | GID512415 | Bank5279 | State53 | 2016-01-19 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 2016-01-30 | 11 days |
| **4** | GID512417 | Bank5286 | State37 | 2016-01-19 | Loan modification,collection,foreclosure | Mortgage | Closed with explanation | No | 2016-01-30 | 11 days |

In [17]:

GD\_MM\_DF = GD\_MM\_DF.drop(['DateOfGrievance','DateOfResolution'],axis=1)

**Getting days as integer**[**¶**](#2et92p0)

In [18]:

GD\_MM\_DF['Days'] = GD\_MM\_DF['Days'].dt.days

**Updating Data types**[**¶**](#tyjcwt)

In [19]:

GD\_MM\_DF['BankID'] = GD\_MM\_DF['BankID'].astype('category')  
GD\_MM\_DF['State'] = GD\_MM\_DF['State'].astype('category')  
GD\_MM\_DF['Grievance\_Category'] = GD\_MM\_DF['Grievance\_Category'].astype('category')  
GD\_MM\_DF['LineOfBusiness'] = GD\_MM\_DF['LineOfBusiness'].astype('category')  
GD\_MM\_DF['ResolutionComments'] = GD\_MM\_DF['ResolutionComments'].astype('category')  
GD\_MM\_DF['Disputed'] = GD\_MM\_DF['Disputed'].astype('category')

In [20]:

GD\_MM\_DF.dtypes

Out[20]:

GrievanceID object  
BankID category  
State category  
Grievance\_Category category  
LineOfBusiness category  
ResolutionComments category  
Disputed category  
Days int64  
dtype: object

**Importing train.csv**[**¶**](#3dy6vkm)

In [21]:

main\_train = pd.read\_csv('Train.csv')

In [22]:

main\_train.head()

Out[22]:

|  | **BankID** | **BankGrade** |
| --- | --- | --- |
| **0** | Bank5298 | satisfactory |
| **1** | Bank5421 | deficient |
| **2** | Bank5326 | satisfactory |
| **3** | Bank5432 | outstanding |
| **4** | Bank5439 | deficient |

**Tranforming train.csv, to a dictonary**[**¶**](#1t3h5sf)

In [23]:

main\_dict = main\_train.set\_index('BankID').to\_dict()['BankGrade']

In [24]:

GD\_MM\_DF.columns

Out[24]:

Index(['GrievanceID', 'BankID', 'State', 'Grievance\_Category',  
 'LineOfBusiness', 'ResolutionComments', 'Disputed', 'Days'],  
 dtype='object')

In [25]:

GD\_MM\_DF\_temp = GD\_MM\_DF[["GrievanceID","BankID","State","Grievance\_Category","LineOfBusiness","ResolutionComments","Disputed","Days"]]

In [26]:

columns\_up = (['GrievanceID', 'BankID', 'State','Grievance\_Category', 'LineOfBusiness','ResolutionComments', 'Disputed','Days','Target'])

In [27]:

GD\_MM\_DF\_Target = GD\_MM\_DF\_temp.reindex(columns = columns\_up)

In [28]:

GD\_MM\_DF\_Target.head()

Out[28]:

|  | **GrievanceID** | **BankID** | **State** | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **Days** | **Target** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512411 | Bank5279 | State31 | Settlement process and costs | Mortgage | Closed with explanation | No | 11 | NaN |
| **1** | GID512412 | Bank5287 | State26 | Application, originator, mortgage broker | Mortgage | Closed with explanation | Yes | 22 | NaN |
| **2** | GID512413 | Bank5286 | State14 | Billing disputes | Credit card | Closed with monetary relief | Yes | 22 | NaN |
| **3** | GID512415 | Bank5279 | State53 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 | NaN |
| **4** | GID512417 | Bank5286 | State37 | Loan modification,collection,foreclosure | Mortgage | Closed with explanation | No | 11 | NaN |

**Code to copy target into the main data frame**[**¶**](#4d34og8)

In [29]:

**for** i **in** range(0, len(GD\_MM\_DF\_Target)):  
 GD\_MM\_DF\_Target.Target.iloc[[i]] = main\_dict.get(GD\_MM\_DF\_Target.iloc[i]['BankID'])

C:\Users\Sowmya\Anaconda3\lib\site-packages\pandas\core\indexing.py:189: SettingWithCopyWarning:  
  
  
A value is trying to be set on a copy of a slice from a DataFrame  
  
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

**Continuing after copying the Target into the main data frame**[**¶**](#2s8eyo1)

In [30]:

GD\_MM\_DF\_Target = pd.read\_csv('GD\_MM\_Target.csv')

In [31]:

GD\_MM\_DF\_Target.head()

Out[31]:

|  | **Unnamed: 0** | **GrievanceID** | **BankID** | **State** | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **Days** | **Target** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0 | GID512411 | Bank5279 | State31 | Settlement process and costs | Mortgage | Closed with explanation | No | 11 | outstanding |
| **1** | 1 | GID512412 | Bank5287 | State26 | Application, originator, mortgage broker | Mortgage | Closed with explanation | Yes | 22 | deficient |
| **2** | 2 | GID512413 | Bank5286 | State14 | Billing disputes | Credit card | Closed with monetary relief | Yes | 22 | satisfactory |
| **3** | 3 | GID512415 | Bank5279 | State53 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 | outstanding |
| **4** | 4 | GID512417 | Bank5286 | State37 | Loan modification,collection,foreclosure | Mortgage | Closed with explanation | No | 11 | satisfactory |

In [32]:

GD\_MM\_DF\_Target.columns

Out[32]:

Index(['Unnamed: 0', 'GrievanceID', 'BankID', 'State', 'Grievance\_Category',  
 'LineOfBusiness', 'ResolutionComments', 'Disputed', 'Days', 'Target'],  
 dtype='object')

In [33]:

GD\_MM\_DF\_Target\_1 = GD\_MM\_DF\_Target.drop(['Unnamed: 0'],axis=1)

**Dropping "State" as this is not a deciding foctor contributing towards classification.¶**

## **Because Grading of the bank is irrespective of state in which it is operating in or from.**[**¶**](#17dp8vu)

In [34]:

GD\_MM\_DF\_Tar\_1hot = GD\_MM\_DF\_Target\_1.drop(['State'],axis=1)

**Also dropping "Bank ID" from model building. It will reintroduced after the "Target" has been prdicted.¶**

In [35]:

GD\_MM\_DF\_Tar\_1hot = GD\_MM\_DF\_Tar\_1hot.drop(['BankID'],axis=1)

**One hot encoder import**[**¶**](#3rdcrjn)

In [36]:

**from** **sklearn.preprocessing** **import** LabelEncoder

**Initializing label encoders**[**¶**](#26in1rg)

In [37]:

le\_GrievCat = LabelEncoder()  
le\_LOB = LabelEncoder()  
le\_ResComm = LabelEncoder()  
le\_Disputed = LabelEncoder()

**Introducing Label encoded values into the dataframe**[**¶**](#lnxbz9)

In [38]:

le\_GrievCat\_trainfit = le\_GrievCat.fit(GD\_MM\_DF\_Tar\_1hot.Grievance\_Category)  
GD\_MM\_DF\_Tar\_1hot['GrievCat\_encoded'] = le\_GrievCat\_trainfit.transform(GD\_MM\_DF\_Tar\_1hot.Grievance\_Category)

In [39]:

le\_LOB\_trainfit = le\_LOB.fit(GD\_MM\_DF\_Tar\_1hot.LineOfBusiness)  
GD\_MM\_DF\_Tar\_1hot['LOB\_encoded'] = le\_LOB\_trainfit.transform(GD\_MM\_DF\_Tar\_1hot.LineOfBusiness)

In [40]:

le\_ResComm\_trainfit = le\_ResComm.fit(GD\_MM\_DF\_Tar\_1hot.ResolutionComments)  
GD\_MM\_DF\_Tar\_1hot['ResComm\_encoded'] = le\_ResComm\_trainfit.transform(GD\_MM\_DF\_Tar\_1hot.ResolutionComments)

In [41]:

le\_Disputed\_trainfit = le\_Disputed.fit(GD\_MM\_DF\_Tar\_1hot.Disputed)  
GD\_MM\_DF\_Tar\_1hot['Dis\_encoded'] = le\_Disputed\_trainfit.transform(GD\_MM\_DF\_Tar\_1hot.Disputed)

**Importing onehot encoder**[**¶**](#35nkun2)

In [42]:

**from** **sklearn.preprocessing** **import** OneHotEncoder

In [43]:

GrievCat\_ohe = OneHotEncoder()  
LOB\_ohe = OneHotEncoder()  
ResComm\_ohe = OneHotEncoder()  
Dis\_ohe = OneHotEncoder()

**Fit and transform method expects a 2D array, reshape to transform from 1D to a 2D array**[**¶**](#1ksv4uv)

In [44]:

XGrievCat\_test\_1 = GrievCat\_ohe.fit(GD\_MM\_DF\_Tar\_1hot.GrievCat\_encoded.values.reshape(-1,1))  
XGrievCat = XGrievCat\_test\_1.transform(GD\_MM\_DF\_Tar\_1hot.GrievCat\_encoded.values.reshape(-1,1)).toarray()

C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\preprocessing\\_encoders.py:363: FutureWarning:  
  
The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.  
If you want the future behaviour and silence this warning, you can specify "categories='auto'".  
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

In [45]:

XLOB\_test\_1 = LOB\_ohe.fit(GD\_MM\_DF\_Tar\_1hot.LOB\_encoded.values.reshape(-1,1))  
XLOB = XLOB\_test\_1.transform(GD\_MM\_DF\_Tar\_1hot.LOB\_encoded.values.reshape(-1,1)).toarray()

C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\preprocessing\\_encoders.py:363: FutureWarning:  
  
The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.  
If you want the future behaviour and silence this warning, you can specify "categories='auto'".  
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

In [46]:

XResComm\_1 = ResComm\_ohe.fit(GD\_MM\_DF\_Tar\_1hot.ResComm\_encoded.values.reshape(-1,1))  
XResComm = XResComm\_1.transform(GD\_MM\_DF\_Tar\_1hot.ResComm\_encoded.values.reshape(-1,1)).toarray()

C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\preprocessing\\_encoders.py:363: FutureWarning:  
  
The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.  
If you want the future behaviour and silence this warning, you can specify "categories='auto'".  
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

In [47]:

XDis\_1 = Dis\_ohe.fit(GD\_MM\_DF\_Tar\_1hot.Dis\_encoded.values.reshape(-1,1))  
XDis = XDis\_1.transform(GD\_MM\_DF\_Tar\_1hot.Dis\_encoded.values.reshape(-1,1)).toarray()

C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\preprocessing\\_encoders.py:363: FutureWarning:  
  
The handling of integer data will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future they will be determined based on the unique values.  
If you want the future behaviour and silence this warning, you can specify "categories='auto'".  
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneHotEncoder directly.

**Adding back 1hot encoded values to the dataframe**[**¶**](#44sinio)

In [48]:

dfOneHot = pd.DataFrame(XGrievCat, columns = ["XGrievCat\_"+str(int(i)) **for** i **in** range(XGrievCat.shape[1])])  
GD\_MM\_DF\_Tar\_1hot = pd.concat([GD\_MM\_DF\_Tar\_1hot, dfOneHot], axis=1)

In [49]:

dfOneHot = pd.DataFrame(XLOB, columns = ["XLOB"+str(int(i)) **for** i **in** range(XLOB.shape[1])])  
GD\_MM\_DF\_Tar\_1hot = pd.concat([GD\_MM\_DF\_Tar\_1hot, dfOneHot], axis=1)

In [50]:

dfOneHot = pd.DataFrame(XResComm, columns = ["XResComm"+str(int(i)) **for** i **in** range(XResComm.shape[1])])  
GD\_MM\_DF\_Tar\_1hot = pd.concat([GD\_MM\_DF\_Tar\_1hot, dfOneHot], axis=1)

In [51]:

dfOneHot = pd.DataFrame(XDis, columns = ["XDis"+str(int(i)) **for** i **in** range(XDis.shape[1])])  
GD\_MM\_DF\_Tar\_1hot = pd.concat([GD\_MM\_DF\_Tar\_1hot, dfOneHot], axis=1)

In [52]:

GD\_MM\_DF\_Tar\_1hot.columns

Out[52]:

Index(['GrievanceID', 'Grievance\_Category', 'LineOfBusiness',  
 'ResolutionComments', 'Disputed', 'Days', 'Target', 'GrievCat\_encoded',  
 'LOB\_encoded', 'ResComm\_encoded', 'Dis\_encoded', 'XGrievCat\_0',  
 'XGrievCat\_1', 'XGrievCat\_2', 'XGrievCat\_3', 'XGrievCat\_4',  
 'XGrievCat\_5', 'XGrievCat\_6', 'XGrievCat\_7', 'XGrievCat\_8',  
 'XGrievCat\_9', 'XGrievCat\_10', 'XGrievCat\_11', 'XGrievCat\_12',  
 'XGrievCat\_13', 'XGrievCat\_14', 'XGrievCat\_15', 'XGrievCat\_16',  
 'XGrievCat\_17', 'XGrievCat\_18', 'XGrievCat\_19', 'XGrievCat\_20',  
 'XGrievCat\_21', 'XGrievCat\_22', 'XGrievCat\_23', 'XGrievCat\_24',  
 'XGrievCat\_25', 'XGrievCat\_26', 'XGrievCat\_27', 'XGrievCat\_28',  
 'XGrievCat\_29', 'XGrievCat\_30', 'XGrievCat\_31', 'XGrievCat\_32',  
 'XGrievCat\_33', 'XGrievCat\_34', 'XGrievCat\_35', 'XGrievCat\_36',  
 'XGrievCat\_37', 'XGrievCat\_38', 'XGrievCat\_39', 'XGrievCat\_40',  
 'XGrievCat\_41', 'XGrievCat\_42', 'XGrievCat\_43', 'XGrievCat\_44',  
 'XGrievCat\_45', 'XGrievCat\_46', 'XGrievCat\_47', 'XGrievCat\_48',  
 'XGrievCat\_49', 'XGrievCat\_50', 'XGrievCat\_51', 'XGrievCat\_52',  
 'XGrievCat\_53', 'XGrievCat\_54', 'XGrievCat\_55', 'XGrievCat\_56',  
 'XGrievCat\_57', 'XGrievCat\_58', 'XGrievCat\_59', 'XGrievCat\_60',  
 'XGrievCat\_61', 'XGrievCat\_62', 'XGrievCat\_63', 'XGrievCat\_64',  
 'XGrievCat\_65', 'XGrievCat\_66', 'XLOB0', 'XLOB1', 'XLOB2', 'XLOB3',  
 'XResComm0', 'XResComm1', 'XResComm2', 'XResComm3', 'XDis0', 'XDis1'],  
 dtype='object')

**Dropping the original values and keeping the one hot encoded values alone**[**¶**](#2jxsxqh)

In [53]:

GD\_MM\_DF\_1hot\_tar = GD\_MM\_DF\_Tar\_1hot.drop(['Grievance\_Category','LineOfBusiness','ResolutionComments','Disputed'],axis=1)

In [54]:

GD\_MM\_DF\_1hot\_tar.head()

Out[54]:

|  | **GrievanceID** | **Days** | **Target** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **Dis\_encoded** | **XGrievCat\_0** | **XGrievCat\_1** | **XGrievCat\_2** | **...** | **XLOB0** | **XLOB1** | **XLOB2** | **XLOB3** | **XResComm0** | **XResComm1** | **XResComm2** | **XResComm3** | **XDis0** | **XDis1** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512411 | 11 | outstanding | 57 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **1** | GID512412 | 22 | deficient | 4 | 2 | 1 | 1 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 |
| **2** | GID512413 | 22 | satisfactory | 12 | 1 | 2 | 1 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 |
| **3** | GID512415 | 11 | outstanding | 43 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **4** | GID512417 | 11 | satisfactory | 42 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |

5 rows × 84 columns

**Updating "Target" to catagorical and then numetical¶**

In [55]:

GD\_MM\_DF\_1hot\_tar.Target = pd.Categorical(GD\_MM\_DF\_1hot\_tar.Target)

In [56]:

GD\_MM\_DF\_1hot\_tar.dtypes

Out[56]:

GrievanceID object  
Days int64  
Target category  
GrievCat\_encoded int32  
LOB\_encoded int32  
ResComm\_encoded int32  
Dis\_encoded int32  
XGrievCat\_0 float64  
XGrievCat\_1 float64  
XGrievCat\_2 float64  
XGrievCat\_3 float64  
XGrievCat\_4 float64  
XGrievCat\_5 float64  
XGrievCat\_6 float64  
XGrievCat\_7 float64  
XGrievCat\_8 float64  
XGrievCat\_9 float64  
XGrievCat\_10 float64  
XGrievCat\_11 float64  
XGrievCat\_12 float64  
XGrievCat\_13 float64  
XGrievCat\_14 float64  
XGrievCat\_15 float64  
XGrievCat\_16 float64  
XGrievCat\_17 float64  
XGrievCat\_18 float64  
XGrievCat\_19 float64  
XGrievCat\_20 float64  
XGrievCat\_21 float64  
XGrievCat\_22 float64  
 ...   
XGrievCat\_47 float64  
XGrievCat\_48 float64  
XGrievCat\_49 float64  
XGrievCat\_50 float64  
XGrievCat\_51 float64  
XGrievCat\_52 float64  
XGrievCat\_53 float64  
XGrievCat\_54 float64  
XGrievCat\_55 float64  
XGrievCat\_56 float64  
XGrievCat\_57 float64  
XGrievCat\_58 float64  
XGrievCat\_59 float64  
XGrievCat\_60 float64  
XGrievCat\_61 float64  
XGrievCat\_62 float64  
XGrievCat\_63 float64  
XGrievCat\_64 float64  
XGrievCat\_65 float64  
XGrievCat\_66 float64  
XLOB0 float64  
XLOB1 float64  
XLOB2 float64  
XLOB3 float64  
XResComm0 float64  
XResComm1 float64  
XResComm2 float64  
XResComm3 float64  
XDis0 float64  
XDis1 float64  
Length: 84, dtype: object

**Target numeric to "Code", into a new column in the dataframe.¶**

In [57]:

GD\_MM\_DF\_1hot\_tar['code'] = GD\_MM\_DF\_1hot\_tar.Target.cat.codes

In [58]:

GD\_MM\_DF\_1hot\_tar.head()

Out[58]:

|  | **GrievanceID** | **Days** | **Target** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **Dis\_encoded** | **XGrievCat\_0** | **XGrievCat\_1** | **XGrievCat\_2** | **...** | **XLOB1** | **XLOB2** | **XLOB3** | **XResComm0** | **XResComm1** | **XResComm2** | **XResComm3** | **XDis0** | **XDis1** | **code** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512411 | 11 | outstanding | 57 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **1** | GID512412 | 22 | deficient | 4 | 2 | 1 | 1 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0 |
| **2** | GID512413 | 22 | satisfactory | 12 | 1 | 2 | 1 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 2 |
| **3** | GID512415 | 11 | outstanding | 43 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **4** | GID512417 | 11 | satisfactory | 42 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 2 |

5 rows × 85 columns

**Changed valued for target : 1- Outstanding, 0 - deficient, 2 - satisfactory**[**¶**](#z337ya)

[**¶**](#3j2qqm3)

Target values codes - Outstanding = 1, deficient = 0, satisfactory = 2

**Dropping Target column**[**¶**](#1y810tw)

In [59]:

GD\_MM\_DF\_1hot\_tarDrop = GD\_MM\_DF\_1hot\_tar.drop(['Target'],axis=1)

**Remider - Keeping the outlier intact in Days. Build two models with and witout**[**¶**](#4i7ojhp)

# Renaming column "Code" to "Target"

In [60]:

GD\_MM\_DF\_days\_Tar = GD\_MM\_DF\_1hot\_tarDrop

In [61]:

GD\_MM\_DF\_days\_Tar.head()

Out[61]:

|  | **GrievanceID** | **Days** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **Dis\_encoded** | **XGrievCat\_0** | **XGrievCat\_1** | **XGrievCat\_2** | **XGrievCat\_3** | **...** | **XLOB1** | **XLOB2** | **XLOB3** | **XResComm0** | **XResComm1** | **XResComm2** | **XResComm3** | **XDis0** | **XDis1** | **code** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512411 | 11 | 57 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **1** | GID512412 | 22 | 4 | 2 | 1 | 1 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0 |
| **2** | GID512413 | 22 | 12 | 1 | 2 | 1 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 2 |
| **3** | GID512415 | 11 | 43 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **4** | GID512417 | 11 | 42 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 2 |

5 rows × 84 columns

In [62]:

GD\_MM\_DF\_days\_Tar.Days.describe()

Out[62]:

count 53559.000000  
mean 17.318154  
std 11.240063  
min 1.000000  
25% 11.000000  
50% 11.000000  
75% 22.000000  
max 629.000000  
Name: Days, dtype: float64

**Finding the outlier in "Days" column¶**

**Defining a fuction to meet needs**[**¶**](#2xcytpi)

In [63]:

**import** **numpy** **as** **np**  
**def** outliers\_z\_score(ys,threshold = 3):  
 *""" This function takes a numeric column and returns the outliers.*  
 *Default threshold is 3, but takes any integer or float value"""*  
 mean\_y = np.mean(ys)  
 stdev\_y = np.std(ys)  
 z\_scores = [(y - mean\_y) / stdev\_y **for** y **in** ys]  
 **return** np.where(np.abs(z\_scores) > threshold)

In [64]:

num\_attr = ["Days"]

**Finding the row ID's of the outliers**[**¶**](#1ci93xb)

In [65]:

GD\_MM\_DF\_days\_Tar.loc[:,num\_attr].apply(outliers\_z\_score)

Out[65]:

Days ([62, 388, 408, 613, 617, 1402, 1451, 1828, 26...  
dtype: object

In [66]:

Days\_Outliers = GD\_MM\_DF\_days\_Tar.loc[:,num\_attr].apply(outliers\_z\_score)

**Row ID's of the outliers**[**¶**](#3whwml4)

In [67]:

Days\_Outliers

Out[67]:

Days ([62, 388, 408, 613, 617, 1402, 1451, 1828, 26...  
dtype: object

In [68]:

Days\_Outliers.values

Out[68]:

array([(array([ 62, 388, 408, 613, 617, 1402, 1451, 1828, 2665,  
 2719, 2988, 3335, 3380, 3762, 3911, 4156, 4336, 4477,  
 4557, 4582, 4740, 4775, 5384, 5483, 5520, 5839, 5856,  
 5924, 6135, 6200, 6331, 6349, 6448, 6910, 6982, 6984,  
 7075, 7081, 7112, 7138, 7152, 7154, 7158, 7219, 7400,  
 7440, 7568, 7970, 8277, 8289, 8368, 8428, 8469, 8498,  
 8504, 8715, 8908, 8924, 9765, 10390, 10417, 10420, 10465,  
 10513, 10525, 10604, 11154, 11453, 11512, 11551, 11567, 11846,  
 11877, 12010, 12059, 12356, 12807, 13641, 14574, 14629, 14645,  
 14667, 14811, 14866, 15167, 15313, 15344, 15364, 15382, 15587,  
 15633, 16177, 16641, 16800, 16816, 16851, 17435, 17471, 17652,  
 17676, 17973, 18002, 18300, 18481, 18550, 18657, 18706, 18864,  
 18887, 18894, 18936, 19147, 19154, 19525, 19546, 19778, 19819,  
 19825, 19829, 19834, 20111, 20177, 20303, 20383, 20384, 20459,  
 20471, 20740, 20852, 20931, 21109, 21156, 21209, 21236, 21248,  
 21277, 21287, 21316, 21323, 21328, 21343, 21617, 21691, 21702,  
 21777, 21778, 21913, 22088, 22242, 22437, 22449, 22505, 22512,  
 22835, 23034, 23107, 23501, 23503, 23583, 23586, 23594, 23633,  
 23634, 23646, 23773, 23914, 23983, 24034, 24123, 24259, 24430,  
 24524, 24645, 24650, 24671, 24672, 24762, 24784, 24974, 24991,  
 25017, 25035, 25047, 25209, 25397, 25438, 25782, 25994, 26109,  
 26145, 26153, 26194, 26202, 26208, 26229, 26230, 26265, 26268,  
 26304, 26307, 26378, 26421, 26454, 26497, 26553, 26556, 26851,  
 26867, 26931, 26960, 27016, 27070, 27107, 27123, 27131, 27144,  
 27201, 27207, 27247, 27381, 27666, 27682, 27693, 27820, 27863,  
 28051, 28088, 28113, 28134, 28156, 28162, 28203, 28277, 28317,  
 28365, 28410, 28440, 28492, 28501, 28559, 28569, 28632, 28676,  
 28698, 28775, 28945, 29171, 29173, 29244, 29302, 29334, 29707,  
 29733, 29798, 29806, 29857, 29868, 29946, 30257, 30300, 30354,  
 30429, 30492, 30537, 30618, 30721, 30789, 30842, 30894, 31018,  
 31071, 31295, 31306, 31391, 31448, 31460, 31505, 31542, 31740,  
 31897, 31922, 31941, 31998, 32051, 32098, 32115, 32282, 32308,  
 32497, 32600, 32783, 32943, 32948, 32964, 32993, 32998, 33023,  
 33045, 33070, 33117, 33162, 33185, 33920, 33972, 34097, 34100,  
 34150, 34154, 34178, 34180, 34208, 34221, 34236, 34386, 34623,  
 34795, 34819, 34889, 35401, 35590, 35597, 35610, 35661, 35664,  
 35695, 35696, 35708, 35714, 36001, 36071, 36119, 36127, 36135,  
 36141, 36166, 36197, 36211, 36232, 36295, 36311, 36323, 36353,  
 36402, 36421, 36451, 36487, 36687, 36720, 36789, 36927, 36955,  
 37091, 37092, 37229, 37337, 37523, 37595, 37659, 37819, 37882,  
 37886, 37950, 37977, 38059, 38072, 38090, 38510, 38681, 38697,  
 38857, 38906, 39029, 39070, 39289, 39737, 39817, 39837, 40009,  
 40110, 40165, 40167, 40187, 40190, 40315, 40331, 40345, 40429,  
 40469, 40717, 40748, 40779, 40835, 40893, 40972, 40979, 41025,  
 41184, 41197, 41271, 41272, 41274, 41275, 41277, 41350, 41394,  
 41423, 41484, 41530, 41562, 41632, 41718, 41780, 41785, 41971,  
 42068, 42069, 42072, 42083, 42335, 42434, 42454, 42506, 42561,  
 42570, 42609, 42625, 42660, 42687, 42773, 42845, 42913, 42923,  
 42930, 42959, 42974, 42977, 42983, 43048, 43060, 43073, 43129,  
 43182, 43233, 43325, 43342, 43357, 43402, 43437, 43666, 43703,  
 43776, 43803, 43812, 43829, 43897, 43948, 44045, 44078, 44172,  
 44174, 44360, 44574, 44798, 44855, 45180, 45311, 45392, 45751,  
 45823, 45844, 46372, 46627, 47023, 47202, 47362, 48116, 48119,  
 48438, 48684, 48776, 48861, 48964, 48993, 49379, 49675, 49757,  
 49760, 49806, 49957, 49959, 50429, 52319, 52685, 52794, 52849,  
 52906, 52956, 52998, 53205], dtype=int64),)], dtype=object)

**Dropping values in Days using row numbers**[**¶**](#2bn6wsx)

In [69]:

GD\_MM\_DF\_days\_Tar.drop(GD\_MM\_DF\_days\_Tar.index[[62,388,408,613,617,1402,1451,1828,2665, 2719,2988,3335,3380,3762,3911,4156,4336,4477, 4557,4582,4740,4775,5384,5483,5520,5839,5856, 5924,6135,6200,6331,6349,6448,6910,6982,6984, 7075,7081,7112,7138,7152,7154,7158,7219,7400, 7440,7568,7970,8277,8289,8368,8428,8469,8498, 8504,8715,8908,8924,9765, 10390, 10417, 10420, 10465, 10513, 10525, 10604, 11154, 11453, 11512, 11551, 11567, 11846, 11877, 12010, 12059, 12356, 12807, 13641, 14574, 14629, 14645, 14667, 14811, 14866, 15167, 15313, 15344, 15364, 15382, 15587, 15633, 16177, 16641, 16800, 16816, 16851, 17435, 17471, 17652, 17676, 17973, 18002, 18300, 18481, 18550, 18657, 18706, 18864, 18887, 18894, 18936, 19147, 19154, 19525, 19546, 19778, 19819, 19825, 19829, 19834, 20111, 20177, 20303, 20383, 20384, 20459, 20471, 20740, 20852, 20931, 21109, 21156, 21209, 21236, 21248, 21277, 21287, 21316, 21323, 21328, 21343, 21617, 21691, 21702, 21777, 21778, 21913, 22088, 22242, 22437, 22449, 22505, 22512, 22835, 23034, 23107, 23501, 23503, 23583, 23586, 23594, 23633, 23634, 23646, 23773, 23914, 23983, 24034, 24123, 24259, 24430, 24524, 24645, 24650, 24671, 24672, 24762, 24784, 24974, 24991, 25017, 25035, 25047, 25209, 25397, 25438, 25782, 25994, 26109, 26145, 26153, 26194, 26202, 26208, 26229, 26230, 26265, 26268, 26304, 26307, 26378, 26421, 26454, 26497, 26553, 26556, 26851, 26867, 26931, 26960, 27016, 27070, 27107, 27123, 27131, 27144, 27201, 27207, 27247, 27381, 27666, 27682, 27693, 27820, 27863, 28051, 28088, 28113, 28134, 28156, 28162, 28203, 28277, 28317, 28365, 28410, 28440, 28492, 28501, 28559, 28569, 28632, 28676, 28698, 28775, 28945, 29171, 29173, 29244, 29302, 29334, 29707, 29733, 29798, 29806, 29857, 29868, 29946, 30257, 30300, 30354, 30429, 30492, 30537, 30618, 30721, 30789, 30842, 30894, 31018, 31071, 31295, 31306, 31391, 31448, 31460, 31505, 31542, 31740, 31897, 31922, 31941, 31998, 32051, 32098, 32115, 32282, 32308, 32497, 32600, 32783, 32943, 32948, 32964, 32993, 32998, 33023, 33045, 33070, 33117, 33162, 33185, 33920, 33972, 34097, 34100, 34150, 34154, 34178, 34180, 34208, 34221, 34236, 34386, 34623, 34795, 34819, 34889, 35401, 35590, 35597, 35610, 35661, 35664, 35695, 35696, 35708, 35714, 36001, 36071, 36119, 36127, 36135, 36141, 36166, 36197, 36211, 36232, 36295, 36311, 36323, 36353, 36402, 36421, 36451, 36487, 36687, 36720, 36789, 36927, 36955, 37091, 37092, 37229, 37337, 37523, 37595, 37659, 37819, 37882, 37886, 37950, 37977, 38059, 38072, 38090, 38510, 38681, 38697, 38857, 38906, 39029, 39070, 39289, 39737, 39817, 39837, 40009, 40110, 40165, 40167, 40187, 40190, 40315, 40331, 40345, 40429, 40469, 40717, 40748, 40779, 40835, 40893, 40972, 40979, 41025, 41184, 41197, 41271, 41272, 41274, 41275, 41277, 41350, 41394, 41423, 41484, 41530, 41562, 41632, 41718, 41780, 41785, 41971, 42068, 42069, 42072, 42083, 42335, 42434, 42454, 42506, 42561, 42570, 42609, 42625, 42660, 42687, 42773, 42845, 42913, 42923, 42930, 42959, 42974, 42977, 42983, 43048, 43060, 43073, 43129, 43182, 43233, 43325, 43342, 43357, 43402, 43437, 43666, 43703, 43776, 43803, 43812, 43829, 43897, 43948, 44045, 44078, 44172, 44174, 44360, 44574, 44798, 44855, 45180, 45311, 45392, 45751, 45823, 45844, 46372, 46627, 47023, 47202, 47362, 48116, 48119, 48438, 48684, 48776, 48861, 48964, 48993, 49379, 49675, 49757, 49760, 49806, 49957, 49959, 50429, 52319, 52685, 52794, 52849, 52906, 52956, 52998, 53205]], inplace=**True**)

In [70]:

GD\_MM\_DF\_days\_Tar.shape

Out[70]:

(53060, 84)

**dropping the outliers from witih respect to days colums, using row indexs**[**¶**](#qsh70q)

[**¶**](#3j2qqm3)

Observation - After removing outliers, row numbers have reduced from 53559 to 53060. As the number of rows dropped are 499 out of 53559, which is 0.9% compared to the total number of rows. I am going ahead with updated dataframe, with 53060 rows.

In [71]:

GD\_MM\_DF\_Norm = GD\_MM\_DF\_days\_Tar

In [72]:

GD\_MM\_DF\_Norm.Days.describe()

Out[72]:

count 53060.000000  
mean 16.806276  
std 8.756380  
min 1.000000  
25% 11.000000  
50% 11.000000  
75% 22.000000  
max 51.000000  
Name: Days, dtype: float64

**Standardizing the "Days" column¶**

In [73]:

**from** **sklearn.preprocessing** **import** StandardScaler

In [74]:

scaler = StandardScaler(with\_mean=**True**, with\_std=**True**).fit(GD\_MM\_DF\_Norm[num\_attr])

C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\preprocessing\data.py:617: DataConversionWarning:  
  
Data with input dtype int64 were all converted to float64 by StandardScaler.

In [75]:

scaler

Out[75]:

StandardScaler(copy=True, with\_mean=True, with\_std=True)

In [76]:

scaler.mean\_

Out[76]:

array([16.80627591])

**Transforming Days column to standardized values**[**¶**](#3as4poj)

**Moving the standardized, days value into the main dataframe**[**¶**](#1pxezwc)

In [77]:

GD\_MM\_DF\_Norm['Days'] = scaler.transform(GD\_MM\_DF\_Norm[num\_attr])

C:\Users\Sowmya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:1: DataConversionWarning:  
  
Data with input dtype int64 were all converted to float64 by StandardScaler.

**Data frame after scaling the Days column**[**¶**](#49x2ik5)

In [78]:

GD\_MM\_DF\_Norm.head()

Out[78]:

|  | **GrievanceID** | **Days** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **Dis\_encoded** | **XGrievCat\_0** | **XGrievCat\_1** | **XGrievCat\_2** | **XGrievCat\_3** | **...** | **XLOB1** | **XLOB2** | **XLOB3** | **XResComm0** | **XResComm1** | **XResComm2** | **XResComm3** | **XDis0** | **XDis1** | **code** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512411 | -0.663097 | 57 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **1** | GID512412 | 0.593142 | 4 | 2 | 1 | 1 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0 |
| **2** | GID512413 | 0.593142 | 12 | 1 | 2 | 1 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 2 |
| **3** | GID512415 | -0.663097 | 43 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **4** | GID512417 | -0.663097 | 42 | 2 | 1 | 0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 2 |

5 rows × 84 columns

**checking "code"( which is the target) value for class imbalace¶**

In [79]:

GD\_MM\_DF\_Norm.code.value\_counts()

Out[79]:

2 34462  
0 14125  
1 4473  
Name: code, dtype: int64

**There is significant difference in the values at all three levels. A case of class imbalance.**[**¶**](#2p2csry)

* Using smote to tackle class imbalance

In [80]:

GD\_MM\_DF\_Norm.columns

Out[80]:

Index(['GrievanceID', 'Days', 'GrievCat\_encoded', 'LOB\_encoded',  
 'ResComm\_encoded', 'Dis\_encoded', 'XGrievCat\_0', 'XGrievCat\_1',  
 'XGrievCat\_2', 'XGrievCat\_3', 'XGrievCat\_4', 'XGrievCat\_5',  
 'XGrievCat\_6', 'XGrievCat\_7', 'XGrievCat\_8', 'XGrievCat\_9',  
 'XGrievCat\_10', 'XGrievCat\_11', 'XGrievCat\_12', 'XGrievCat\_13',  
 'XGrievCat\_14', 'XGrievCat\_15', 'XGrievCat\_16', 'XGrievCat\_17',  
 'XGrievCat\_18', 'XGrievCat\_19', 'XGrievCat\_20', 'XGrievCat\_21',  
 'XGrievCat\_22', 'XGrievCat\_23', 'XGrievCat\_24', 'XGrievCat\_25',  
 'XGrievCat\_26', 'XGrievCat\_27', 'XGrievCat\_28', 'XGrievCat\_29',  
 'XGrievCat\_30', 'XGrievCat\_31', 'XGrievCat\_32', 'XGrievCat\_33',  
 'XGrievCat\_34', 'XGrievCat\_35', 'XGrievCat\_36', 'XGrievCat\_37',  
 'XGrievCat\_38', 'XGrievCat\_39', 'XGrievCat\_40', 'XGrievCat\_41',  
 'XGrievCat\_42', 'XGrievCat\_43', 'XGrievCat\_44', 'XGrievCat\_45',  
 'XGrievCat\_46', 'XGrievCat\_47', 'XGrievCat\_48', 'XGrievCat\_49',  
 'XGrievCat\_50', 'XGrievCat\_51', 'XGrievCat\_52', 'XGrievCat\_53',  
 'XGrievCat\_54', 'XGrievCat\_55', 'XGrievCat\_56', 'XGrievCat\_57',  
 'XGrievCat\_58', 'XGrievCat\_59', 'XGrievCat\_60', 'XGrievCat\_61',  
 'XGrievCat\_62', 'XGrievCat\_63', 'XGrievCat\_64', 'XGrievCat\_65',  
 'XGrievCat\_66', 'XLOB0', 'XLOB1', 'XLOB2', 'XLOB3', 'XResComm0',  
 'XResComm1', 'XResComm2', 'XResComm3', 'XDis0', 'XDis1', 'code'],  
 dtype='object')

In [81]:

cols=['Days', 'GrievCat\_encoded', 'LOB\_encoded',  
 'ResComm\_encoded', 'Dis\_encoded', 'XGrievCat\_0', 'XGrievCat\_1',  
 'XGrievCat\_2', 'XGrievCat\_3', 'XGrievCat\_4', 'XGrievCat\_5',  
 'XGrievCat\_6', 'XGrievCat\_7', 'XGrievCat\_8', 'XGrievCat\_9',  
 'XGrievCat\_10', 'XGrievCat\_11', 'XGrievCat\_12', 'XGrievCat\_13',  
 'XGrievCat\_14', 'XGrievCat\_15', 'XGrievCat\_16', 'XGrievCat\_17',  
 'XGrievCat\_18', 'XGrievCat\_19', 'XGrievCat\_20', 'XGrievCat\_21',  
 'XGrievCat\_22', 'XGrievCat\_23', 'XGrievCat\_24', 'XGrievCat\_25',  
 'XGrievCat\_26', 'XGrievCat\_27', 'XGrievCat\_28', 'XGrievCat\_29',  
 'XGrievCat\_30', 'XGrievCat\_31', 'XGrievCat\_32', 'XGrievCat\_33',  
 'XGrievCat\_34', 'XGrievCat\_35', 'XGrievCat\_36', 'XGrievCat\_37',  
 'XGrievCat\_38', 'XGrievCat\_39', 'XGrievCat\_40', 'XGrievCat\_41',  
 'XGrievCat\_42', 'XGrievCat\_43', 'XGrievCat\_44', 'XGrievCat\_45',  
 'XGrievCat\_46', 'XGrievCat\_47', 'XGrievCat\_48', 'XGrievCat\_49',  
 'XGrievCat\_50', 'XGrievCat\_51', 'XGrievCat\_52', 'XGrievCat\_53',  
 'XGrievCat\_54', 'XGrievCat\_55', 'XGrievCat\_56', 'XGrievCat\_57',  
 'XGrievCat\_58', 'XGrievCat\_59', 'XGrievCat\_60', 'XGrievCat\_61',  
 'XGrievCat\_62', 'XGrievCat\_63', 'XGrievCat\_64', 'XGrievCat\_65',  
 'XGrievCat\_66', 'XLOB0', 'XLOB1', 'XLOB2', 'XLOB3', 'XResComm0',  
 'XResComm1', 'XResComm2', 'XResComm3', 'XDis0', 'XDis1']

**Declaring X, and Y columns**[**¶**](#147n2zr)

In [82]:

X=GD\_MM\_DF\_Norm[cols]  
Y=GD\_MM\_DF\_Norm['code']

**Using smote to counter class imbalance**[**¶**](#3o7alnk)

**Importing smote**[**¶**](#23ckvvd)

In [83]:

**from** **imblearn.over\_sampling** **import** SMOTE

In [84]:

**from** **sklearn.model\_selection** **import** train\_test\_split

**Splitting the train , into train and validation**[**¶**](#ihv636)

In [85]:

X\_train, X\_test, Y\_train, Y\_test=train\_test\_split(X,Y,test\_size=0.3,random\_state=123)

**Checking the values before implementing smote**[**¶**](#32hioqz)

In [86]:

print("Number transactions X\_train dataset: ", X\_train.shape)  
print("Number transactions y\_train dataset: ", Y\_train.shape)  
print("Number transactions X\_test dataset: ", X\_test.shape)  
print("Number transactions y\_test dataset: ", Y\_test.shape)

Number transactions X\_train dataset: (37142, 82)  
Number transactions y\_train dataset: (37142,)  
Number transactions X\_test dataset: (15918, 82)  
Number transactions y\_test dataset: (15918,)

In [87]:

print("Before OverSampling, counts of label '0': **{}**".format(sum(Y\_train==0)))  
print("Before OverSampling, counts of label '1': **{}** **\n**".format(sum(Y\_train==1)))  
print("Before OverSampling, counts of label '2': **{}** **\n**".format(sum(Y\_train==2)))

Before OverSampling, counts of label '0': 9867  
Before OverSampling, counts of label '1': 3139   
  
Before OverSampling, counts of label '2': 24136

**Implimenting smote**[**¶**](#1hmsyys)

[**¶**](#3j2qqm3)

Oversmapling the minority class alone, which is laebl '1'

In [88]:

sm = SMOTE(sampling\_strategy='minority',random\_state=42,k\_neighbors=10)  
  
  
*## Splitting based on SMOTE*   
  
X\_train\_res, Y\_train\_res = sm.fit\_sample(X\_train, Y\_train)

In [89]:

print('After OverSampling, the shape of train\_X: **{}**'.format(X\_train\_res.shape))  
print('After OverSampling, the shape of train\_Y: **{}** **\n**'.format(Y\_train\_res.shape))  
  
print("After OverSampling, counts of label '0': **{}**".format(sum(Y\_train\_res==0)))  
print("After OverSampling, counts of label '1': **{}**".format(sum(Y\_train\_res==1)))  
print("After OverSampling, counts of label '2': **{}**".format(sum(Y\_train\_res==2)))

After OverSampling, the shape of train\_X: (58139, 82)  
After OverSampling, the shape of train\_Y: (58139,)   
  
After OverSampling, counts of label '0': 9867  
After OverSampling, counts of label '1': 24136  
After OverSampling, counts of label '2': 24136

**Trying logistic, with smote values.**[**¶**](#41mghml)

In [90]:

**from** **sklearn.linear\_model** **import** LogisticRegression  
  
predictiveModel = LogisticRegression()

In [91]:

predictiveModel.fit(X\_train\_res, Y\_train\_res)

C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\linear\_model\logistic.py:432: FutureWarning:  
  
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.  
  
C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\linear\_model\logistic.py:459: FutureWarning:  
  
Default multi\_class will be changed to 'auto' in 0.22. Specify the multi\_class option to silence this warning.

Out[91]:

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,  
 intercept\_scaling=1, max\_iter=100, multi\_class='warn',  
 n\_jobs=None, penalty='l2', random\_state=None, solver='warn',  
 tol=0.0001, verbose=0, warm\_start=False)

**Y predictions, based on Logistic regression model**[**¶**](#2grqrue)

In [92]:

Y\_preds = predictiveModel.predict(X\_test)

**Importing confusion matrix**[**¶**](#vx1227)

In [93]:

**from** **sklearn.metrics** **import** confusion\_matrix

**Getting confusion matrix for validation v/s train**[**¶**](#3fwokq0)

In [94]:

confusion\_matrix(Y\_test, Y\_preds)

Out[94]:

array([[2050, 509, 1699],  
 [ 8, 1003, 323],  
 [1795, 3647, 4884]], dtype=int64)

In [95]:

**from** **sklearn.metrics** **import** accuracy\_score

**Getting accuracy**[**¶**](#1v1yuxt)

In [96]:

accuracy\_score(Y\_test, Y\_preds)

Out[96]:

0.4986179168237216

**Trying with PCA for logistical regression, with smote**[**¶**](#4f1mdlm)

In [97]:

**from** **sklearn.decomposition** **import** PCA  
  
pca = PCA(n\_components=10)

In [98]:

pca.fit(X\_train\_res)

Out[98]:

PCA(copy=True, iterated\_power='auto', n\_components=10, random\_state=None,  
 svd\_solver='auto', tol=0.0, whiten=False)

In [99]:

X\_new\_train = pca.transform(X\_train\_res)  
  
X\_new\_test = pca.transform(X\_test)

**Getting logistic regression prediction on PCA valeus**[**¶**](#2u6wntf)

In [100]:

predictiveModel = LogisticRegression()

In [101]:

predictiveModel.fit(X\_new\_train, Y\_train\_res)

C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\linear\_model\logistic.py:432: FutureWarning:  
  
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.  
  
C:\Users\Sowmya\Anaconda3\lib\site-packages\sklearn\linear\_model\logistic.py:459: FutureWarning:  
  
Default multi\_class will be changed to 'auto' in 0.22. Specify the multi\_class option to silence this warning.

Out[101]:

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,  
 intercept\_scaling=1, max\_iter=100, multi\_class='warn',  
 n\_jobs=None, penalty='l2', random\_state=None, solver='warn',  
 tol=0.0001, verbose=0, warm\_start=False)

In [102]:

Y\_preds = predictiveModel.predict(X\_new\_test)

In [103]:

confusion\_matrix(Y\_test, Y\_preds)

Out[103]:

array([[2093, 590, 1575],  
 [ 7, 1057, 270],  
 [1867, 4227, 4232]], dtype=int64)

**Accuracy with PCA, on logistic regression**[**¶**](#19c6y18)

In [104]:

accuracy\_score(Y\_test, Y\_preds)

Out[104]:

0.4637517276039703

**Building SM model**[**¶**](#3tbugp1)

In [105]:

**from** **sklearn** **import** svm  
**from** **sklearn** **import** metrics  
**from** **sklearn** **import** tree

**Building Decision Tree model with SMOTE values**[**¶**](#28h4qwu)

In [106]:

*# Create tree object*   
model = tree.DecisionTreeClassifier(criterion='gini')

In [107]:

model.fit(X\_train\_res, Y\_train\_res)

Out[107]:

DecisionTreeClassifier(class\_weight=None, criterion='gini', max\_depth=None,  
 max\_features=None, max\_leaf\_nodes=None,  
 min\_impurity\_decrease=0.0, min\_impurity\_split=None,  
 min\_samples\_leaf=1, min\_samples\_split=2,  
 min\_weight\_fraction\_leaf=0.0, presort=False, random\_state=None,  
 splitter='best')

In [108]:

predicted= model.predict(X\_test)

**Accuracy with Decision tree**[**¶**](#nmf14n)

In [109]:

accuracy\_score(Y\_test,predicted)

Out[109]:

0.5149516270888302

In [110]:

confusion\_matrix(Y\_test, predicted)

Out[110]:

array([[1685, 421, 2152],  
 [ 18, 955, 361],  
 [1432, 3337, 5557]], dtype=int64)

**Building Gradient boost model**[**¶**](#37m2jsg)

In [111]:

**from** **sklearn.ensemble** **import** GradientBoostingClassifier

In [112]:

GB\_clf = GradientBoostingClassifier(n\_estimators=100, learning\_rate=1.0, max\_depth=1)

**Gradient boost with Smote Vales**[**¶**](#1mrcu09)

In [113]:

GB\_clf.fit(X\_train\_res, Y\_train\_res)

Out[113]:

GradientBoostingClassifier(criterion='friedman\_mse', init=None,  
 learning\_rate=1.0, loss='deviance', max\_depth=1,  
 max\_features=None, max\_leaf\_nodes=None,  
 min\_impurity\_decrease=0.0, min\_impurity\_split=None,  
 min\_samples\_leaf=1, min\_samples\_split=2,  
 min\_weight\_fraction\_leaf=0.0, n\_estimators=100,  
 n\_iter\_no\_change=None, presort='auto', random\_state=None,  
 subsample=1.0, tol=0.0001, validation\_fraction=0.1,  
 verbose=0, warm\_start=False)

In [114]:

GB\_predicted= GB\_clf.predict(X\_test)

**Accuracy with gradient boost model**[**¶**](#46r0co2)

In [115]:

accuracy\_score(Y\_test,GB\_predicted)

Out[115]:

0.5052142228923232

In [116]:

confusion\_matrix(Y\_test, predicted)

Out[116]:

array([[1685, 421, 2152],  
 [ 18, 955, 361],  
 [1432, 3337, 5557]], dtype=int64)

**Multi-layer perceptron model building**[**¶**](#2lwamvv)

In [117]:

*## for perceptron models*  
**from** **sklearn.model\_selection** **import** train\_test\_split  
**from** **sklearn.decomposition** **import** PCA  
**from** **sklearn.metrics** **import** confusion\_matrix, roc\_curve, auc  
  
**from** **keras.models** **import** Sequential  
**from** **keras.layers** **import** Dense  
**from** **keras.utils** **import** to\_categorical

Using TensorFlow backend.

In [118]:

mlp\_model = Sequential()  
  
mlp\_model.add(Dense(12, input\_dim=82, activation='relu', kernel\_initializer='normal'))  
mlp\_model.add(Dense(3, activation='sigmoid', kernel\_initializer='normal'))

In [119]:

mlp\_model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

In [120]:

**from** **keras.utils** **import** to\_categorical

In [121]:

Y\_binary = to\_categorical(Y\_train)

In [122]:

mlp\_model.fit(X\_train, Y\_binary, epochs=30, batch\_size=10)

Epoch 1/30  
37142/37142 [==============================] - 3s 86us/step - loss: 0.7452 - acc: 0.6539  
Epoch 2/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7114 - acc: 0.6666  
Epoch 3/30  
37142/37142 [==============================] - 3s 72us/step - loss: 0.7101 - acc: 0.6669  
Epoch 4/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7091 - acc: 0.6687  
Epoch 5/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7088 - acc: 0.6688  
Epoch 6/30  
37142/37142 [==============================] - 2s 65us/step - loss: 0.7083 - acc: 0.6678  
Epoch 7/30  
37142/37142 [==============================] - 3s 68us/step - loss: 0.7082 - acc: 0.6684  
Epoch 8/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7075 - acc: 0.6692  
Epoch 9/30  
37142/37142 [==============================] - 3s 71us/step - loss: 0.7078 - acc: 0.6687  
Epoch 10/30  
37142/37142 [==============================] - 3s 71us/step - loss: 0.7067 - acc: 0.6691  
Epoch 11/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7068 - acc: 0.6688  
Epoch 12/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7068 - acc: 0.6691  
Epoch 13/30  
37142/37142 [==============================] - 2s 64us/step - loss: 0.7063 - acc: 0.6690  
Epoch 14/30  
37142/37142 [==============================] - 2s 65us/step - loss: 0.7063 - acc: 0.6693  
Epoch 15/30  
37142/37142 [==============================] - 3s 71us/step - loss: 0.7061 - acc: 0.6677  
Epoch 16/30  
37142/37142 [==============================] - 3s 68us/step - loss: 0.7057 - acc: 0.6681  
Epoch 17/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7060 - acc: 0.6691  
Epoch 18/30  
37142/37142 [==============================] - 3s 69us/step - loss: 0.7057 - acc: 0.6693  
Epoch 19/30  
37142/37142 [==============================] - 3s 69us/step - loss: 0.7056 - acc: 0.6684  
Epoch 20/30  
37142/37142 [==============================] - 2s 65us/step - loss: 0.7054 - acc: 0.6686  
Epoch 21/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7054 - acc: 0.6692  
Epoch 22/30  
37142/37142 [==============================] - 3s 72us/step - loss: 0.7052 - acc: 0.6695  
Epoch 23/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7053 - acc: 0.6699  
Epoch 24/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7048 - acc: 0.6692  
Epoch 25/30  
37142/37142 [==============================] - 2s 65us/step - loss: 0.7053 - acc: 0.6703  
Epoch 26/30  
37142/37142 [==============================] - 2s 65us/step - loss: 0.7051 - acc: 0.6696  
Epoch 27/30  
37142/37142 [==============================] - 3s 70us/step - loss: 0.7047 - acc: 0.6695  
Epoch 28/30  
37142/37142 [==============================] - 3s 74us/step - loss: 0.7046 - acc: 0.6703  
Epoch 29/30  
37142/37142 [==============================] - 3s 73us/step - loss: 0.7047 - acc: 0.6695  
Epoch 30/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7048 - acc: 0.6698

Out[122]:

<keras.callbacks.History at 0x25ca6320>

In [123]:

test\_pred\_mlp=mlp\_model.predict\_classes(X\_test)  
train\_pred\_mlp=mlp\_model.predict\_classes(X\_train)

In [124]:

confusion\_matrix\_test\_mlp = confusion\_matrix(Y\_test, test\_pred\_mlp)  
confusion\_matrix\_train\_mlp=confusion\_matrix(Y\_train, train\_pred\_mlp)  
  
print(confusion\_matrix\_train\_mlp)  
print(confusion\_matrix\_test\_mlp)

[[ 3032 0 6835]  
 [ 14 0 3125]  
 [ 2265 0 21871]]  
[[1277 0 2981]  
 [ 5 0 1329]  
 [1055 0 9271]]

In [125]:

train\_pred\_prob\_mlp=mlp\_model.predict(X\_train)

**Accuracy achived with MLP is 67.16%¶**

**Building MLP model with dropout**[**¶**](#111kx3o)

In [126]:

**from** **keras.layers** **import** Dropout

In [127]:

mlp\_model\_drop = Sequential()

In [128]:

mlp\_model\_drop.add(Dropout(0.3, input\_shape=(82,)))  
  
mlp\_model\_drop.add(Dense(24, input\_dim=82, activation='relu', kernel\_initializer='normal'))  
  
mlp\_model\_drop.add(Dense(3, activation='sigmoid', kernel\_initializer='normal'))

In [129]:

mlp\_model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

In [130]:

Y\_binary = to\_categorical(Y\_train)

In [131]:

mlp\_model.fit(X\_train, Y\_binary, epochs=30, batch\_size=10)

Epoch 1/30  
37142/37142 [==============================] - 3s 74us/step - loss: 0.7043 - acc: 0.6708  
Epoch 2/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7044 - acc: 0.6698  
Epoch 3/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7043 - acc: 0.6699  
Epoch 4/30  
37142/37142 [==============================] - 3s 72us/step - loss: 0.7041 - acc: 0.6696  
Epoch 5/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7046 - acc: 0.6688  
Epoch 6/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7043 - acc: 0.6712  
Epoch 7/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7041 - acc: 0.6692  
Epoch 8/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7043 - acc: 0.6696  
Epoch 9/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7039 - acc: 0.6700  
Epoch 10/30  
37142/37142 [==============================] - 3s 71us/step - loss: 0.7043 - acc: 0.6700  
Epoch 11/30  
37142/37142 [==============================] - 3s 68us/step - loss: 0.7040 - acc: 0.6698  
Epoch 12/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7041 - acc: 0.6686  
Epoch 13/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7038 - acc: 0.6701  
Epoch 14/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7038 - acc: 0.6704  
Epoch 15/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7038 - acc: 0.6704  
Epoch 16/30  
37142/37142 [==============================] - 3s 71us/step - loss: 0.7036 - acc: 0.6688  
Epoch 17/30  
37142/37142 [==============================] - 3s 69us/step - loss: 0.7035 - acc: 0.6692  
Epoch 18/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7039 - acc: 0.6701  
Epoch 19/30  
37142/37142 [==============================] - 2s 67us/step - loss: 0.7035 - acc: 0.6694  
Epoch 20/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7036 - acc: 0.6705  
Epoch 21/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7036 - acc: 0.6704  
Epoch 22/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7035 - acc: 0.6722  
Epoch 23/30  
37142/37142 [==============================] - 3s 71us/step - loss: 0.7034 - acc: 0.6712  
Epoch 24/30  
37142/37142 [==============================] - 3s 77us/step - loss: 0.7035 - acc: 0.6705  
Epoch 25/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7031 - acc: 0.6702  
Epoch 26/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7034 - acc: 0.6699  
Epoch 27/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7032 - acc: 0.6708  
Epoch 28/30  
37142/37142 [==============================] - 2s 66us/step - loss: 0.7031 - acc: 0.6707  
Epoch 29/30  
37142/37142 [==============================] - 3s 73us/step - loss: 0.7029 - acc: 0.6709  
Epoch 30/30  
37142/37142 [==============================] - 3s 69us/step - loss: 0.7032 - acc: 0.6691

Out[131]:

<keras.callbacks.History at 0x262db470>

In [132]:

test\_pred\_mlp=mlp\_model\_drop.predict\_classes(X\_test)  
train\_pred\_mlp=mlp\_model\_drop.predict\_classes(X\_train)

In [133]:

confusion\_matrix\_test\_mlp = confusion\_matrix(Y\_test, test\_pred\_mlp)  
confusion\_matrix\_train\_mlp=confusion\_matrix(Y\_train, train\_pred\_mlp)  
  
print(confusion\_matrix\_train\_mlp)  
print(confusion\_matrix\_test\_mlp)

[[ 751 8252 864]  
 [ 87 3022 30]  
 [ 1496 21345 1295]]  
[[ 322 3539 397]  
 [ 38 1283 13]  
 [ 611 9182 533]]

**Accuracy with MLP with dropout = 67.06%. And with better results in confusion matrix.¶**

**Importing test data set and preprocessing**[**¶**](#3l18frh)

In [134]:

pwd()

Out[134]:

'C:\\Users\\Sowmya\\Downloads\\PHD\\End\_To\_End\_Submission'

In [135]:

Griev\_data\_test = pd.read\_csv('GrievancesData\_Test.csv')

In [136]:

Griev\_data\_test.head()

Out[136]:

|  | **GrievanceID** | **BankID** | **State** | **DateOfGrievance** | **Grievance\_Category** | **GrievanceDescription** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **DateOfResolution** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | State43 | 2016-01-19 | Loan servicing, payments, escrow account | I currently have a mortgage with Flag star ba... | Mortgage | Closed with explanation | No | 2016-01-30 |
| **1** | GID512416 | Bank5278 | State16 | 2016-01-19 | Problems when you are unable to pay | Bank5278 Auto Finance repossessed and subsequ... | Consumer Loan | Closed with explanation | No | 2016-02-03 |
| **2** | GID515121 | Bank5372 | State56 | 2016-03-19 | Loan servicing, payments, escrow account | We fell behind in our payments back in 2014 f... | Mortgage | Closed with explanation | No | 2016-03-30 |
| **3** | GID515123 | Bank5372 | State47 | 2016-03-19 | Loan servicing, payments, escrow account | My home is on XXXX parcels of land. The first... | Mortgage | Closed with explanation | No | 2016-03-30 |
| **4** | GID515124 | Bank5310 | State10 | 2016-03-19 | Settlement process and costs | We mortgaged our home with XXXX on XXXX/XXXX/... | Mortgage | Closed with explanation | No | 2016-03-30 |

In [137]:

Griev\_data\_test.describe(include='all')

Out[137]:

|  | **GrievanceID** | **BankID** | **State** | **DateOfGrievance** | **Grievance\_Category** | **GrievanceDescription** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **DateOfResolution** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **count** | 27954 | 27954 | 27879 | 27954 | 27954 | 27954 | 27954 | 27954 | 27954 | 27954 |
| **unique** | 27954 | 22 | 61 | 1168 | 59 | 24137 | 4 | 4 | 2 | 1198 |
| **top** | GID384860 | Bank5373 | State9 | 2014-01-19 | Loan servicing, payments, escrow account | I was shocked when I reviewed my credit repor... | Credit card | Closed with explanation | No | 2014-02-19 |
| **freq** | 1 | 8221 | 3839 | 727 | 4403 | 6 | 9561 | 23505 | 22524 | 291 |

**Checking for null values**[**¶**](#30j0zll)

In [138]:

Griev\_data\_test.isnull().sum()

Out[138]:

GrievanceID 0  
BankID 0  
State 75  
DateOfGrievance 0  
Grievance\_Category 0  
GrievanceDescription 0  
LineOfBusiness 0  
ResolutionComments 0  
Disputed 0  
DateOfResolution 0  
dtype: int64

**dropping all the null values**[**¶**](#206ipza)

In [139]:

Griev\_data\_test.dropna(inplace=**True**)

**Dropping the columns for Grev description**[**¶**](#4k668n3)

In [140]:

Griev\_test = Griev\_data\_test[['GrievanceID', 'BankID', 'State', 'DateOfGrievance',  
 'Grievance\_Category', 'LineOfBusiness',  
 'ResolutionComments', 'Disputed', 'DateOfResolution']]

In [141]:

pd.set\_option('mode.chained\_assignment','warn')

**Getting Days values from Date columns**[**¶**](#2zbgiuw)

In [142]:

Griev\_test['DateOfGrievance'] = Griev\_test.DateOfGrievance.astype('datetime64[ns]')  
Griev\_test['DateOfResolution'] = Griev\_test.DateOfResolution.astype('datetime64[ns]')

C:\Users\Sowmya\Anaconda3\lib\site-packages\ipykernel\_launcher.py:1: 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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy  
  
C:\Users\Sowmya\Anaconda3\lib\site-packages\ipykernel\_launcher.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: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

**Getting the differenc of "Days" between Date of Grievance and Date of resolution¶**

In [143]:

Griev\_test['Days'] = Griev\_test.apply(**lambda** row: row.DateOfResolution - row.DateOfGrievance, axis=1)

In [144]:

Griev\_test.head()

Out[144]:

|  | **GrievanceID** | **BankID** | **State** | **DateOfGrievance** | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **DateOfResolution** | **Days** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | State43 | 2016-01-19 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 2016-01-30 | 11 days |
| **1** | GID512416 | Bank5278 | State16 | 2016-01-19 | Problems when you are unable to pay | Consumer Loan | Closed with explanation | No | 2016-02-03 | 15 days |
| **2** | GID515121 | Bank5372 | State56 | 2016-03-19 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 2016-03-30 | 11 days |
| **3** | GID515123 | Bank5372 | State47 | 2016-03-19 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 2016-03-30 | 11 days |
| **4** | GID515124 | Bank5310 | State10 | 2016-03-19 | Settlement process and costs | Mortgage | Closed with explanation | No | 2016-03-30 | 11 days |

**Dropping Date columns**[**¶**](#1egqt2p)

In [145]:

Griev\_test\_Days = Griev\_test.drop(['DateOfGrievance','DateOfResolution'],axis=1)

In [146]:

Griev\_test\_Days.head()

Out[146]:

|  | **GrievanceID** | **BankID** | **State** | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **Days** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | State43 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 days |
| **1** | GID512416 | Bank5278 | State16 | Problems when you are unable to pay | Consumer Loan | Closed with explanation | No | 15 days |
| **2** | GID515121 | Bank5372 | State56 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 days |
| **3** | GID515123 | Bank5372 | State47 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 days |
| **4** | GID515124 | Bank5310 | State10 | Settlement process and costs | Mortgage | Closed with explanation | No | 11 days |

**Updating "Days" columns to get format to integer¶**

In [147]:

Griev\_test\_Days['Days'] = Griev\_test\_Days['Days'].dt.days

In [148]:

Griev\_test\_Days.head()

Out[148]:

|  | **GrievanceID** | **BankID** | **State** | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **Days** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | State43 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 |
| **1** | GID512416 | Bank5278 | State16 | Problems when you are unable to pay | Consumer Loan | Closed with explanation | No | 15 |
| **2** | GID515121 | Bank5372 | State56 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 |
| **3** | GID515123 | Bank5372 | State47 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 |
| **4** | GID515124 | Bank5310 | State10 | Settlement process and costs | Mortgage | Closed with explanation | No | 11 |

In [149]:

Griev\_test\_Days.dtypes

Out[149]:

GrievanceID object  
BankID object  
State object  
Grievance\_Category object  
LineOfBusiness object  
ResolutionComments object  
Disputed object  
Days int64  
dtype: object

**checking for catagorical columns**[**¶**](#3ygebqi)

In [150]:

Griev\_test\_Days\_temp = Griev\_test\_Days[['Grievance\_Category',  
 'LineOfBusiness', 'ResolutionComments', 'Disputed']]

In [151]:

Griev\_test\_Days\_temp.describe()

Out[151]:

|  | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** |
| --- | --- | --- | --- | --- |
| **count** | 27879 | 27879 | 27879 | 27879 |
| **unique** | 59 | 4 | 4 | 2 |
| **top** | Loan servicing, payments, escrow account | Credit card | Closed with explanation | No |
| **freq** | 4399 | 9537 | 23443 | 22465 |

**Updating featers as catagory in main data frame**[**¶**](#2dlolyb)

In [152]:

Griev\_test\_Days['Grievance\_Category'] = Griev\_test\_Days['Grievance\_Category'].astype('category')  
Griev\_test\_Days['LineOfBusiness'] = Griev\_test\_Days['LineOfBusiness'].astype('category')  
Griev\_test\_Days['ResolutionComments'] = Griev\_test\_Days['ResolutionComments'].astype('category')  
Griev\_test\_Days['Disputed'] = Griev\_test\_Days['Disputed'].astype('category')

In [153]:

Griev\_test\_Days.dtypes

Out[153]:

GrievanceID object  
BankID object  
State object  
Grievance\_Category category  
LineOfBusiness category  
ResolutionComments category  
Disputed category  
Days int64  
dtype: object

**Dropping State**[**¶**](#sqyw64)

In [154]:

Griev\_test\_final = Griev\_test\_Days.drop(['State'],axis=1)

In [155]:

Griev\_test\_final.columns

Out[155]:

Index(['GrievanceID', 'BankID', 'Grievance\_Category', 'LineOfBusiness',  
 'ResolutionComments', 'Disputed', 'Days'],  
 dtype='object')

In [156]:

Griev\_test\_final.head()

Out[156]:

|  | **GrievanceID** | **BankID** | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **Days** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 |
| **1** | GID512416 | Bank5278 | Problems when you are unable to pay | Consumer Loan | Closed with explanation | No | 15 |
| **2** | GID515121 | Bank5372 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 |
| **3** | GID515123 | Bank5372 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11 |
| **4** | GID515124 | Bank5310 | Settlement process and costs | Mortgage | Closed with explanation | No | 11 |

In [157]:

Griev\_test\_final.shape

Out[157]:

(27879, 7)

**One hot encoding of test data frame**[**¶**](#3cqmetx)

In [158]:

Griev\_test\_final['GrievCat\_encoded'] = le\_GrievCat\_trainfit.transform(Griev\_test\_final.Grievance\_Category)

In [159]:

Griev\_test\_final['LOB\_encoded'] = le\_LOB\_trainfit.transform(Griev\_test\_final.LineOfBusiness)

In [160]:

Griev\_test\_final['ResComm\_encoded'] = le\_ResComm\_trainfit.transform(Griev\_test\_final.ResolutionComments)

In [161]:

Griev\_test\_final['Dis\_encoded'] = le\_Disputed\_trainfit.transform(Griev\_test\_final.Disputed)

In [162]:

Griev\_test\_final.shape

Out[162]:

(27879, 11)

In [163]:

Griev\_test\_final.describe()

Out[163]:

|  | **Days** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **Dis\_encoded** |
| --- | --- | --- | --- | --- | --- |
| **count** | 27879.000000 | 27879.000000 | 27879.000000 | 27879.000000 | 27879.000000 |
| **mean** | 18.069479 | 33.835898 | 1.887478 | 1.207253 | 0.194196 |
| **std** | 10.216651 | 15.581502 | 0.871301 | 0.548086 | 0.395588 |
| **min** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 11.000000 | 25.000000 | 1.000000 | 1.000000 | 0.000000 |
| **50%** | 11.000000 | 32.000000 | 2.000000 | 1.000000 | 0.000000 |
| **75%** | 22.000000 | 43.000000 | 3.000000 | 1.000000 | 0.000000 |
| **max** | 170.000000 | 66.000000 | 3.000000 | 3.000000 | 1.000000 |

**Importing one hot encoder**[**¶**](#1rvwp1q)

**fit\_transform method expects a 2D array, reshape to transform from 1D to a 2D array**[**¶**](#4bvk7pj)

In [164]:

XGrievCat\_test = XGrievCat\_test\_1.transform(Griev\_test\_final.GrievCat\_encoded.values.reshape(-1,1)).toarray()  
XLOB\_test = XLOB\_test\_1.transform(Griev\_test\_final.LOB\_encoded.values.reshape(-1,1)).toarray()  
XResComm\_test = XResComm\_1.transform(Griev\_test\_final.ResComm\_encoded.values.reshape(-1,1)).toarray()  
XDis\_test = XDis\_1.transform(Griev\_test\_final.Dis\_encoded.values.reshape(-1,1)).toarray()

**Adding back to the datafrom**[**¶**](#2r0uhxc)

In [165]:

dfOneHot\_test = pd.DataFrame(XGrievCat\_test, columns = ["XGrievCat\_test\_"+str(int(i)) **for** i **in** range(XGrievCat\_test.shape[1])])  
Griev\_test\_final = pd.concat([Griev\_test\_final, dfOneHot\_test], axis=1)

In [166]:

dfOneHot\_test = pd.DataFrame(XLOB\_test, columns = ["XLOB\_test\_"+str(int(i)) **for** i **in** range(XLOB\_test.shape[1])])  
Griev\_test\_final = pd.concat([Griev\_test\_final, dfOneHot\_test], axis=1)

In [167]:

dfOneHot\_test = pd.DataFrame(XResComm\_test, columns = ["XResComm\_test"+str(int(i)) **for** i **in** range(XResComm\_test.shape[1])])  
Griev\_test\_final = pd.concat([Griev\_test\_final, dfOneHot\_test], axis=1)

In [168]:

dfOneHot\_test = pd.DataFrame(XDis\_test, columns = ["XDis\_test"+str(int(i)) **for** i **in** range(XDis\_test.shape[1])])  
Griev\_test\_final = pd.concat([Griev\_test\_final, dfOneHot\_test], axis=1)

In [169]:

Griev\_test\_final.head()

Out[169]:

|  | **GrievanceID** | **BankID** | **Grievance\_Category** | **LineOfBusiness** | **ResolutionComments** | **Disputed** | **Days** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **...** | **XLOB\_test\_0** | **XLOB\_test\_1** | **XLOB\_test\_2** | **XLOB\_test\_3** | **XResComm\_test0** | **XResComm\_test1** | **XResComm\_test2** | **XResComm\_test3** | **XDis\_test0** | **XDis\_test1** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11.0 | 43.0 | 2.0 | 1.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **1** | GID512416 | Bank5278 | Problems when you are unable to pay | Consumer Loan | Closed with explanation | No | 15.0 | 53.0 | 0.0 | 1.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **2** | GID515121 | Bank5372 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11.0 | 43.0 | 2.0 | 1.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **3** | GID515123 | Bank5372 | Loan servicing, payments, escrow account | Mortgage | Closed with explanation | No | 11.0 | 43.0 | 2.0 | 1.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **4** | GID515124 | Bank5310 | Settlement process and costs | Mortgage | Closed with explanation | No | 11.0 | 57.0 | 2.0 | 1.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |

5 rows × 88 columns

In [170]:

Griev\_test\_final.isnull().sum()

Out[170]:

GrievanceID 75  
BankID 75  
Grievance\_Category 75  
LineOfBusiness 75  
ResolutionComments 75  
Disputed 75  
Days 75  
GrievCat\_encoded 75  
LOB\_encoded 75  
ResComm\_encoded 75  
Dis\_encoded 75  
XGrievCat\_test\_0 75  
XGrievCat\_test\_1 75  
XGrievCat\_test\_2 75  
XGrievCat\_test\_3 75  
XGrievCat\_test\_4 75  
XGrievCat\_test\_5 75  
XGrievCat\_test\_6 75  
XGrievCat\_test\_7 75  
XGrievCat\_test\_8 75  
XGrievCat\_test\_9 75  
XGrievCat\_test\_10 75  
XGrievCat\_test\_11 75  
XGrievCat\_test\_12 75  
XGrievCat\_test\_13 75  
XGrievCat\_test\_14 75  
XGrievCat\_test\_15 75  
XGrievCat\_test\_16 75  
XGrievCat\_test\_17 75  
XGrievCat\_test\_18 75  
 ..  
XGrievCat\_test\_47 75  
XGrievCat\_test\_48 75  
XGrievCat\_test\_49 75  
XGrievCat\_test\_50 75  
XGrievCat\_test\_51 75  
XGrievCat\_test\_52 75  
XGrievCat\_test\_53 75  
XGrievCat\_test\_54 75  
XGrievCat\_test\_55 75  
XGrievCat\_test\_56 75  
XGrievCat\_test\_57 75  
XGrievCat\_test\_58 75  
XGrievCat\_test\_59 75  
XGrievCat\_test\_60 75  
XGrievCat\_test\_61 75  
XGrievCat\_test\_62 75  
XGrievCat\_test\_63 75  
XGrievCat\_test\_64 75  
XGrievCat\_test\_65 75  
XGrievCat\_test\_66 75  
XLOB\_test\_0 75  
XLOB\_test\_1 75  
XLOB\_test\_2 75  
XLOB\_test\_3 75  
XResComm\_test0 75  
XResComm\_test1 75  
XResComm\_test2 75  
XResComm\_test3 75  
XDis\_test0 75  
XDis\_test1 75  
Length: 88, dtype: int64

In [171]:

Griev\_test\_final.shape

Out[171]:

(27954, 88)

**Dropping allthe NA values**[**¶**](#1664s55)

In [172]:

Griev\_test\_final.dropna(inplace=**True**)

In [173]:

Griev\_test\_final.shape

Out[173]:

(27804, 88)

**For model predictions**[**¶**](#3q5sasy)

In [174]:

Griev\_test\_Pred = Griev\_test\_final.drop(['Grievance\_Category','LineOfBusiness','ResolutionComments','Disputed'],axis=1)

**Standardizing days column in test set**[**¶**](#25b2l0r)

**Using the scalar from train dataset**[**¶**](#kgcv8k)

In [175]:

Griev\_test\_Pred['Days'] = scaler.transform(Griev\_test\_Pred[num\_attr])

In [176]:

Griev\_test\_Pred.head()

Out[176]:

|  | **GrievanceID** | **BankID** | **Days** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **Dis\_encoded** | **XGrievCat\_test\_0** | **XGrievCat\_test\_1** | **XGrievCat\_test\_2** | **...** | **XLOB\_test\_0** | **XLOB\_test\_1** | **XLOB\_test\_2** | **XLOB\_test\_3** | **XResComm\_test0** | **XResComm\_test1** | **XResComm\_test2** | **XResComm\_test3** | **XDis\_test0** | **XDis\_test1** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **1** | GID512416 | Bank5278 | -0.206283 | 53.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **2** | GID515121 | Bank5372 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **3** | GID515123 | Bank5372 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **4** | GID515124 | Bank5310 | -0.663097 | 57.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |

5 rows × 84 columns

In [177]:

X\_final\_test = Griev\_test\_Pred[['Days', 'GrievCat\_encoded', 'LOB\_encoded',  
 'ResComm\_encoded', 'Dis\_encoded', 'XGrievCat\_test\_0',  
 'XGrievCat\_test\_1', 'XGrievCat\_test\_2', 'XGrievCat\_test\_3',  
 'XGrievCat\_test\_4', 'XGrievCat\_test\_5', 'XGrievCat\_test\_6',  
 'XGrievCat\_test\_7', 'XGrievCat\_test\_8', 'XGrievCat\_test\_9',  
 'XGrievCat\_test\_10', 'XGrievCat\_test\_11', 'XGrievCat\_test\_12',  
 'XGrievCat\_test\_13', 'XGrievCat\_test\_14', 'XGrievCat\_test\_15',  
 'XGrievCat\_test\_16', 'XGrievCat\_test\_17', 'XGrievCat\_test\_18',  
 'XGrievCat\_test\_19', 'XGrievCat\_test\_20', 'XGrievCat\_test\_21',  
 'XGrievCat\_test\_22', 'XGrievCat\_test\_23', 'XGrievCat\_test\_24',  
 'XGrievCat\_test\_25', 'XGrievCat\_test\_26', 'XGrievCat\_test\_27',  
 'XGrievCat\_test\_28', 'XGrievCat\_test\_29', 'XGrievCat\_test\_30',  
 'XGrievCat\_test\_31', 'XGrievCat\_test\_32', 'XGrievCat\_test\_33',  
 'XGrievCat\_test\_34', 'XGrievCat\_test\_35', 'XGrievCat\_test\_36',  
 'XGrievCat\_test\_37', 'XGrievCat\_test\_38', 'XGrievCat\_test\_39',  
 'XGrievCat\_test\_40', 'XGrievCat\_test\_41', 'XGrievCat\_test\_42',  
 'XGrievCat\_test\_43', 'XGrievCat\_test\_44', 'XGrievCat\_test\_45',  
 'XGrievCat\_test\_46', 'XGrievCat\_test\_47', 'XGrievCat\_test\_48',  
 'XGrievCat\_test\_49', 'XGrievCat\_test\_50', 'XGrievCat\_test\_51',  
 'XGrievCat\_test\_52', 'XGrievCat\_test\_53', 'XGrievCat\_test\_54',  
 'XGrievCat\_test\_55', 'XGrievCat\_test\_56', 'XGrievCat\_test\_57',  
 'XGrievCat\_test\_58', 'XGrievCat\_test\_59', 'XGrievCat\_test\_60',  
 'XGrievCat\_test\_61', 'XGrievCat\_test\_62', 'XGrievCat\_test\_63',  
 'XGrievCat\_test\_64', 'XGrievCat\_test\_65', 'XGrievCat\_test\_66',  
 'XLOB\_test\_0', 'XLOB\_test\_1', 'XLOB\_test\_2', 'XLOB\_test\_3',  
 'XResComm\_test0', 'XResComm\_test1', 'XResComm\_test2', 'XResComm\_test3',  
 'XDis\_test0', 'XDis\_test1']]

**XG\_Boost model Predictions on test data**[**¶**](#34g0dwd)

In [178]:

GB\_predicted= GB\_clf.predict(X\_final\_test)

In [179]:

GB\_predicted.shape

Out[179]:

(27804,)

**checking the test data**[**¶**](#1jlao46)

In [180]:

Griev\_test\_Pred.head()

Out[180]:

|  | **GrievanceID** | **BankID** | **Days** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **Dis\_encoded** | **XGrievCat\_test\_0** | **XGrievCat\_test\_1** | **XGrievCat\_test\_2** | **...** | **XLOB\_test\_0** | **XLOB\_test\_1** | **XLOB\_test\_2** | **XLOB\_test\_3** | **XResComm\_test0** | **XResComm\_test1** | **XResComm\_test2** | **XResComm\_test3** | **XDis\_test0** | **XDis\_test1** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **1** | GID512416 | Bank5278 | -0.206283 | 53.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **2** | GID515121 | Bank5372 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **3** | GID515123 | Bank5372 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |
| **4** | GID515124 | Bank5310 | -0.663097 | 57.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 |

5 rows × 84 columns

**adding code (Target) to the data set**[**¶**](#43ky6rz)

In [181]:

Griev\_test\_Pred['code'] = GB\_predicted

**Rechecking**[**¶**](#2iq8gzs)

In [182]:

Griev\_test\_Pred.head()

Out[182]:

|  | **GrievanceID** | **BankID** | **Days** | **GrievCat\_encoded** | **LOB\_encoded** | **ResComm\_encoded** | **Dis\_encoded** | **XGrievCat\_test\_0** | **XGrievCat\_test\_1** | **XGrievCat\_test\_2** | **...** | **XLOB\_test\_1** | **XLOB\_test\_2** | **XLOB\_test\_3** | **XResComm\_test0** | **XResComm\_test1** | **XResComm\_test2** | **XResComm\_test3** | **XDis\_test0** | **XDis\_test1** | **code** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | GID512414 | Bank5334 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **1** | GID512416 | Bank5278 | -0.206283 | 53.0 | 0.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **2** | GID515121 | Bank5372 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **3** | GID515123 | Bank5372 | -0.663097 | 43.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 1 |
| **4** | GID515124 | Bank5310 | -0.663097 | 57.0 | 2.0 | 1.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 0.0 | 1.0 | 0.0 | 2 |

5 rows × 85 columns

**Grouping by Bank ID and code**[**¶**](#xvir7l)

**And getting counts of each value under code column**[**¶**](#3hv69ve)

In [183]:

Griev\_test\_code\_1 = Griev\_test\_Pred.groupby(['BankID','code']).size().reset\_index(name='counts')

In [184]:

Griev\_test\_code\_1

Out[184]:

|  | **BankID** | **code** | **counts** |
| --- | --- | --- | --- |
| **0** | Bank5246 | 0 | 482 |
| **1** | Bank5246 | 1 | 518 |
| **2** | Bank5246 | 2 | 706 |
| **3** | Bank5259 | 0 | 252 |
| **4** | Bank5259 | 1 | 301 |
| **5** | Bank5259 | 2 | 450 |
| **6** | Bank5271 | 0 | 31 |
| **7** | Bank5271 | 1 | 64 |
| **8** | Bank5271 | 2 | 25 |
| **9** | Bank5272 | 0 | 17 |
| **10** | Bank5272 | 1 | 37 |
| **11** | Bank5272 | 2 | 17 |
| **12** | Bank5278 | 0 | 822 |
| **13** | Bank5278 | 1 | 966 |
| **14** | Bank5278 | 2 | 1269 |
| **15** | Bank5284 | 0 | 38 |
| **16** | Bank5284 | 1 | 40 |
| **17** | Bank5284 | 2 | 61 |
| **18** | Bank5299 | 0 | 70 |
| **19** | Bank5299 | 1 | 56 |
| **20** | Bank5299 | 2 | 71 |
| **21** | Bank5310 | 0 | 580 |
| **22** | Bank5310 | 1 | 1209 |
| **23** | Bank5310 | 2 | 1069 |
| **24** | Bank5312 | 0 | 28 |
| **25** | Bank5312 | 1 | 94 |
| **26** | Bank5312 | 2 | 53 |
| **27** | Bank5316 | 0 | 24 |
| **28** | Bank5316 | 1 | 22 |
| **29** | Bank5316 | 2 | 40 |
| **...** | ... | ... | ... |
| **36** | Bank5334 | 0 | 49 |
| **37** | Bank5334 | 1 | 91 |
| **38** | Bank5334 | 2 | 92 |
| **39** | Bank5372 | 0 | 630 |
| **40** | Bank5372 | 1 | 1571 |
| **41** | Bank5372 | 2 | 1338 |
| **42** | Bank5373 | 0 | 1160 |
| **43** | Bank5373 | 1 | 3105 |
| **44** | Bank5373 | 2 | 3876 |
| **45** | Bank5374 | 0 | 24 |
| **46** | Bank5374 | 1 | 14 |
| **47** | Bank5374 | 2 | 38 |
| **48** | Bank5391 | 0 | 28 |
| **49** | Bank5391 | 1 | 30 |
| **50** | Bank5391 | 2 | 41 |
| **51** | Bank5393 | 0 | 94 |
| **52** | Bank5393 | 1 | 174 |
| **53** | Bank5393 | 2 | 147 |
| **54** | Bank5403 | 0 | 127 |
| **55** | Bank5403 | 1 | 161 |
| **56** | Bank5403 | 2 | 218 |
| **57** | Bank5416 | 0 | 10 |
| **58** | Bank5416 | 1 | 41 |
| **59** | Bank5416 | 2 | 53 |
| **60** | Bank5424 | 0 | 230 |
| **61** | Bank5424 | 1 | 444 |
| **62** | Bank5424 | 2 | 297 |
| **63** | Bank5437 | 0 | 1188 |
| **64** | Bank5437 | 1 | 1055 |
| **65** | Bank5437 | 2 | 1873 |

66 rows × 3 columns

**GEtting the max count "code" value for each bank¶**

In [185]:

df = Griev\_test\_code\_1[Griev\_test\_code\_1.groupby('BankID')['counts'].transform('max') == Griev\_test\_code\_1['counts']]  
print (df)

BankID code counts  
2 Bank5246 2 706  
5 Bank5259 2 450  
7 Bank5271 1 64  
10 Bank5272 1 37  
14 Bank5278 2 1269  
17 Bank5284 2 61  
20 Bank5299 2 71  
22 Bank5310 1 1209  
25 Bank5312 1 94  
29 Bank5316 2 40  
32 Bank5318 2 44  
34 Bank5322 1 55  
38 Bank5334 2 92  
40 Bank5372 1 1571  
44 Bank5373 2 3876  
47 Bank5374 2 38  
50 Bank5391 2 41  
52 Bank5393 1 174  
56 Bank5403 2 218  
59 Bank5416 2 53  
61 Bank5424 1 444  
65 Bank5437 2 1873

**Extracccting "BankID" and "Code" into a different dataframe¶**

In [186]:

Sub\_1= df[['BankID','code']]

In [187]:

Sub\_1

Out[187]:

|  | **BankID** | **code** |
| --- | --- | --- |
| **2** | Bank5246 | 2 |
| **5** | Bank5259 | 2 |
| **7** | Bank5271 | 1 |
| **10** | Bank5272 | 1 |
| **14** | Bank5278 | 2 |
| **17** | Bank5284 | 2 |
| **20** | Bank5299 | 2 |
| **22** | Bank5310 | 1 |
| **25** | Bank5312 | 1 |
| **29** | Bank5316 | 2 |
| **32** | Bank5318 | 2 |
| **34** | Bank5322 | 1 |
| **38** | Bank5334 | 2 |
| **40** | Bank5372 | 1 |
| **44** | Bank5373 | 2 |
| **47** | Bank5374 | 2 |
| **50** | Bank5391 | 2 |
| **52** | Bank5393 | 1 |
| **56** | Bank5403 | 2 |
| **59** | Bank5416 | 2 |
| **61** | Bank5424 | 1 |
| **65** | Bank5437 | 2 |

**Moving predictions to a csv**[**¶**](#1x0gk37)

In [188]:

Sub\_1.to\_csv('Sub\_1.csv')

**End of main model**[**¶**](#4h042r0)

**Full report for the comlete analytical model**[**¶**](#2w5ecyt)

## **Approach -**[**¶**](#1baon6m)

Banks were given accumulated rating, based on how they handle grievances keeping time taken and how well they maintain communication with customers. The factor of, if they keep a check on repeated grievances, under same business line was also an important factor to decide the bank grade.

Train.csv, has independent bank grades for individual banks. Grievenaces\_Train.csv, has details related to individual Grievances, with all respective details such as bank ID, Grievances description, line of business and more.

## **Towards Analytical model building -**[**¶**](#3vac5uf)

* Excluded Grievance Description column. And kept aside to utilize for Text mining.
* Mapped "BankGrade" values from train.csv, to all individual grievance ID's
* Extracted number of days spend on a grievance, with the help of Data of grievance and resolution of grievance
* With preliminary Exploratory data analysis, pre-processed the train dataset.
* Keeping "BankGrade" as target for Grievance ID's , checked for data imbalance.
* Number of Grievance, with BankGrade as outstanding was 3139, where as for deficit was 9867 and satisfactory was 24136. With case of class imbalance, for "outstanding" grade have lower number of values, before starting with model building, used SMOTE to over come the issue of imbalance.
* While SMOTE , the minority class, which is "outstanding" in this case was over sampled.
* Went ahead with building a basic logistic regression model, achieved an accuracy of 49.86%, on validation data.
* With near 50% separation using logistic regression, went ahead with finding PCA and build a logistic model. But accuracy reduced to 46.37%. So, omitted it.
* Decision trees and Gradient boost were next model built. Accuracy improved, with 51.48% and 50.52% for respective models, with validation data.

## **Multiple Perceptron models -**[**¶**](#2afmg28)

* Built a sequential, multi-layer , dense ANN model and achieved accuracy of 66.78%. But confusion matrix, had an entire column with 0's. The model was not predicted a particular class of values.
  + To overcome the 0's, another ANN model was built, but this time drop out rate of 0.3 used. The accuracy achieved was 66.91%. With an improved confusion matrix.

## After comparing accuracy of all the models, I choose Gradient boost to make the final prediction on the test data. And the final accuracy was 27.27%.

**Summary and conclusion -**[**¶**](#pkwqa1)

With 27.27% as the final accuracy on test data, it appears there is need to look into finer details of the available data and build better models.

## **What more could have done?**[**¶**](#39kk8xu)

* Mapping the bank code, to the grievance ID, was done taking Bank ID, as the key value. Instead, a better statistical approach in mapping, Bank grades to grv ID, could have helped build a better model and get way better accuracy.
* While pre-processing test data, 75 NA, values emerged during transformation to 1hot encoded values for categorical columns. These rows with "NA" values, were dropped and prediction on these Grv ID's were lost.
* While taking mode of bank code on individual banks, in more than one case there was difference of values less that 10, among the bank grades. With the lost 75 predictions, model accuracy could have improved significantly.

**Continuing with Text mining,for sentiment analysis**[**¶**](#1opuj5n)

In [189]:

**from** **sklearn.feature\_extraction.text** **import** TfidfVectorizer,CountVectorizer

**TRAIN Grev\_Desc DATA - Getting lematized values of Grivenace desccription under train data**[**¶**](#48pi1tg)

In [190]:

clean\_train\_text = pd.read\_csv('lemm\_clean\_Grev\_All.csv')

In [191]:

clean\_train\_text.head()

Out[191]:

|  | **Unnamed: 0** | **0** |
| --- | --- | --- |
| **0** | 0 | research apt ca name customer loan finance hom... |
| **1** | 1 | mortgage never miss payment finance citizens m... |
| **2** | 2 | give credit limit sear credit card purchase in... |
| **3** | 3 | mortgage sell buy never miss payment late chan... |
| **4** | 4 | start deal city mortgage back modification pro... |

**TRAIN Grev\_Desc DATA - Picking up the text column alone**[**¶**](#2nusc19)

In [192]:

clean\_train\_text\_GrevDes = clean\_train\_text[['0']]

In [193]:

clean\_train\_text\_GrevDes.head()

Out[193]:

|  | **0** |
| --- | --- |
| **0** | research apt ca name customer loan finance hom... |
| **1** | mortgage never miss payment finance citizens m... |
| **2** | give credit limit sear credit card purchase in... |
| **3** | mortgage sell buy never miss payment late chan... |
| **4** | start deal city mortgage back modification pro... |

**TRAIN Grev\_Desc DATA - Column renaming to "Grev\_Desc"¶**

In [194]:

clean\_train\_text\_GrevDes.rename(columns={'0': 'Grev\_Desc'},inplace=**True**)

C:\Users\Sowmya\Anaconda3\lib\site-packages\pandas\core\frame.py:3781: SettingWithCopyWarning:  
  
  
A value is trying to be set on a copy of a slice from a DataFrame  
  
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

In [195]:

clean\_train\_text\_GrevDes.head()

Out[195]:

|  | **Grev\_Desc** |
| --- | --- |
| **0** | research apt ca name customer loan finance hom... |
| **1** | mortgage never miss payment finance citizens m... |
| **2** | give credit limit sear credit card purchase in... |
| **3** | mortgage sell buy never miss payment late chan... |
| **4** | start deal city mortgage back modification pro... |

**TRAIN Grev\_Desc DATA - Creating a back up data frame**[**¶**](#1302m92)

In [196]:

clean\_train\_text\_sample = clean\_train\_text\_GrevDes

In [197]:

clean\_train\_text\_sample.describe()

Out[197]:

|  | **Grev\_Desc** |
| --- | --- |
| **count** | 53680 |
| **unique** | 41183 |
| **top** | shock review credit report find late payment d... |
| **freq** | 70 |

**TRAIN Grev\_Desc DATA - Adding back the column for "Disputed", to the dataframe of lamatized text¶**

In [198]:

clean\_train\_text\_sample['Disputed'] = GD\_train.Disputed

In [199]:

clean\_train\_text\_sample.head()

Out[199]:

|  | **Grev\_Desc** | **Disputed** |
| --- | --- | --- |
| **0** | research apt ca name customer loan finance hom... | No |
| **1** | mortgage never miss payment finance citizens m... | Yes |
| **2** | give credit limit sear credit card purchase in... | Yes |
| **3** | mortgage sell buy never miss payment late chan... | No |
| **4** | start deal city mortgage back modification pro... | No |

**TRAIN Grev\_Desc DATA - Saperating Disputed Greviances**[**¶**](#3mzq4wv)

In [200]:

Disputed\_Grev = clean\_train\_text\_sample.loc[clean\_train\_text\_sample['Disputed'] == 'Yes']

In [201]:

Disputed\_Grev.head()

Out[201]:

|  | **Grev\_Desc** | **Disputed** |
| --- | --- | --- |
| **1** | mortgage never miss payment finance citizens m... | Yes |
| **2** | give credit limit sear credit card purchase in... | Yes |
| **6** | big fail care last child go college decide dow... | Yes |
| **8** | owen loan serve service loan refuse talk anyth... | Yes |
| **10** | make payment loan apply finance charge princip... | Yes |

**TRAIN Grev\_Desc DATA - Getting Undisputed values**[**¶**](#2250f4o)

In [202]:

UnDisputed\_Grev = clean\_train\_text\_sample.loc[clean\_train\_text\_sample['Disputed'] == 'No']

In [203]:

UnDisputed\_Grev.head()

Out[203]:

|  | **Grev\_Desc** | **Disputed** |
| --- | --- | --- |
| **0** | research apt ca name customer loan finance hom... | No |
| **3** | mortgage sell buy never miss payment late chan... | No |
| **4** | start deal city mortgage back modification pro... | No |
| **5** | trouble payments handle husband lose job unemp... | No |
| **7** | loan send payment form money order lose paymen... | No |

**TRAIN Grev\_Desc DATA - Tfidf vectorizer for Disputed text in train data**[**¶**](#haapch)

In [204]:

tfidf\_vectorizer = TfidfVectorizer(ngram\_range=(1,3),min\_df=0.1,max\_df=0.5,max\_features=500) *# stop\_words='english'*  
tfidf\_Dis= tfidf\_vectorizer.fit(Disputed\_Grev.Grev\_Desc)   
tfidf\_all = tfidf\_Dis.transform(Disputed\_Grev.Grev\_Desc)

**TRAIN Grev\_Desc DATA - Getting features of disputed train**[**¶**](#319y80a)

In [205]:

pd.DataFrame(tfidf\_all.toarray(), columns= tfidf\_vectorizer.get\_feature\_names())

Out[205]:

|  | **able** | **account** | **additional** | **address** | **allow** | **also** | **amount** | **another** | **apply** | **ask** | **...** | **use** | **want** | **way** | **well** | **without** | **work** | **would** | **would not** | **write** | **years** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.000000 | 0.000000 | 0.140038 | 0.000000 | 0.142351 | 0.048797 | 0.052147 | 0.061499 | 0.000000 | 0.096317 | ... | 0.000000 | 0.000000 | 0.069641 | 0.000000 | 0.000000 | 0.059304 | 0.272462 | 0.066597 | 0.000000 | 0.000000 |
| **1** | 0.069414 | 0.195025 | 0.000000 | 0.000000 | 0.073889 | 0.101314 | 0.216538 | 0.127687 | 0.000000 | 0.049994 | ... | 0.113214 | 0.057584 | 0.072296 | 0.069790 | 0.000000 | 0.000000 | 0.202034 | 0.000000 | 0.000000 | 0.056502 |
| **2** | 0.119080 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.086902 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.138636 | 0.000000 | 0.000000 | 0.000000 |
| **3** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.060044 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **4** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.224345 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.213484 |
| **5** | 0.000000 | 0.194987 | 0.000000 | 0.045055 | 0.000000 | 0.063309 | 0.000000 | 0.039894 | 0.074206 | 0.093721 | ... | 0.000000 | 0.000000 | 0.180704 | 0.000000 | 0.000000 | 0.000000 | 0.025249 | 0.043202 | 0.000000 | 0.141227 |
| **6** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.040404 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.060316 | 0.000000 | 0.000000 | 0.084341 |
| **7** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.100225 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.149619 | 0.000000 | 0.120129 | 0.000000 |
| **8** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.108402 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.206308 |
| **9** | 0.000000 | 0.239690 | 0.000000 | 0.049231 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.068271 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.042035 | 0.275893 | 0.094411 | 0.000000 | 0.000000 |
| **10** | 0.000000 | 0.074961 | 0.000000 | 0.069285 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.171168 | 0.000000 | ... | 0.000000 | 0.055334 | 0.000000 | 0.134126 | 0.000000 | 0.118317 | 0.077656 | 0.000000 | 0.000000 | 0.000000 |
| **11** | 0.000000 | 0.300070 | 0.000000 | 0.000000 | 0.000000 | 0.048714 | 0.000000 | 0.122789 | 0.171295 | 0.192306 | ... | 0.054435 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.177607 | 0.194284 | 0.000000 | 0.062396 | 0.054334 |
| **12** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.080227 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.091198 | 0.000000 | 0.000000 | 0.000000 | 0.097501 | 0.063994 | 0.000000 | 0.000000 | 0.089484 |
| **13** | 0.000000 | 0.380262 | 0.000000 | 0.000000 | 0.000000 | 0.049386 | 0.000000 | 0.124483 | 0.057886 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.078786 | 0.067401 | 0.000000 | 0.000000 |
| **14** | 0.096993 | 0.000000 | 0.000000 | 0.000000 | 0.103245 | 0.000000 | 0.000000 | 0.089209 | 0.000000 | 0.000000 | ... | 0.000000 | 0.080462 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.056461 | 0.096604 | 0.000000 | 0.157901 |
| **15** | 0.000000 | 0.425115 | 0.000000 | 0.000000 | 0.000000 | 0.157745 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.188740 | 0.107645 | 0.000000 | 0.000000 |
| **16** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.064533 | 0.000000 | 0.103628 | 0.090239 |
| **17** | 0.000000 | 0.046455 | 0.000000 | 0.300556 | 0.000000 | 0.030166 | 0.000000 | 0.114056 | 0.000000 | 0.148857 | ... | 0.033709 | 0.034291 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.024062 | 0.000000 | 0.000000 | 0.000000 |
| **18** | 0.000000 | 0.000000 | 0.140038 | 0.000000 | 0.142351 | 0.048797 | 0.052147 | 0.061499 | 0.000000 | 0.096317 | ... | 0.000000 | 0.000000 | 0.069641 | 0.000000 | 0.000000 | 0.059304 | 0.272462 | 0.066597 | 0.000000 | 0.000000 |
| **19** | 0.000000 | 0.074961 | 0.000000 | 0.069285 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.171168 | 0.000000 | ... | 0.000000 | 0.055334 | 0.000000 | 0.134126 | 0.000000 | 0.118317 | 0.077656 | 0.000000 | 0.000000 | 0.000000 |
| **20** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.108402 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.206308 |
| **21** | 0.119080 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.086902 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.138636 | 0.000000 | 0.000000 | 0.000000 |
| **22** | 0.000000 | 0.046455 | 0.000000 | 0.300556 | 0.000000 | 0.030166 | 0.000000 | 0.114056 | 0.000000 | 0.148857 | ... | 0.033709 | 0.034291 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.024062 | 0.000000 | 0.000000 | 0.000000 |
| **23** | 0.000000 | 0.194987 | 0.000000 | 0.045055 | 0.000000 | 0.063309 | 0.000000 | 0.039894 | 0.074206 | 0.093721 | ... | 0.000000 | 0.000000 | 0.180704 | 0.000000 | 0.000000 | 0.000000 | 0.025249 | 0.043202 | 0.000000 | 0.141227 |
| **24** | 0.000000 | 0.425115 | 0.000000 | 0.000000 | 0.000000 | 0.157745 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.188740 | 0.107645 | 0.000000 | 0.000000 |
| **25** | 0.069414 | 0.195025 | 0.000000 | 0.000000 | 0.073889 | 0.101314 | 0.216538 | 0.127687 | 0.000000 | 0.049994 | ... | 0.113214 | 0.057584 | 0.072296 | 0.069790 | 0.000000 | 0.000000 | 0.202034 | 0.000000 | 0.000000 | 0.056502 |
| **26** | 0.000000 | 0.074961 | 0.000000 | 0.069285 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.171168 | 0.000000 | ... | 0.000000 | 0.055334 | 0.000000 | 0.134126 | 0.000000 | 0.118317 | 0.077656 | 0.000000 | 0.000000 | 0.000000 |
| **27** | 0.000000 | 0.000000 | 0.000000 | 0.644217 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **28** | 0.000000 | 0.142900 | 0.088767 | 0.088052 | 0.000000 | 0.061863 | 0.132219 | 0.000000 | 0.000000 | 0.122107 | ... | 0.207386 | 0.000000 | 0.088288 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.079238 | 0.276002 |
| **29** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **11734** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.137338 | 0.000000 | 0.173089 | 0.000000 | 0.000000 | ... | 0.230204 | 0.000000 | 0.098002 | 0.000000 | 0.000000 | 0.250364 | 0.054774 | 0.000000 | 0.000000 | 0.000000 |
| **11735** | 0.000000 | 0.269585 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.124718 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11736** | 0.000000 | 0.096221 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.133543 | 0.000000 | 0.366185 | 0.061665 | ... | 0.000000 | 0.000000 | 0.089173 | 0.086082 | 0.000000 | 0.000000 | 0.049839 | 0.000000 | 0.000000 | 0.000000 |
| **11737** | 0.000000 | 0.209720 | 0.000000 | 0.000000 | 0.198641 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11738** | 0.000000 | 0.000000 | 0.000000 | 0.299979 | 0.000000 | 0.210756 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.268694 | 0.000000 | 0.000000 | 0.000000 | 0.269953 | 0.000000 |
| **11739** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11740** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11741** | 0.000000 | 0.272404 | 0.000000 | 0.000000 | 0.000000 | 0.147407 | 0.126022 | 0.000000 | 0.138224 | 0.116384 | ... | 0.032944 | 0.000000 | 0.000000 | 0.040617 | 0.000000 | 0.000000 | 0.094065 | 0.000000 | 0.000000 | 0.000000 |
| **11742** | 0.000000 | 0.000000 | 0.000000 | 0.495738 | 0.000000 | 0.208974 | 0.000000 | 0.000000 | 0.081648 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.177615 | 0.084657 | 0.000000 | 0.000000 | 0.089223 | 0.000000 |
| **11743** | 0.085444 | 0.000000 | 0.000000 | 0.088753 | 0.000000 | 0.062355 | 0.000000 | 0.078587 | 0.000000 | 0.123079 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.049738 | 0.000000 | 0.000000 | 0.000000 |
| **11744** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.279127 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.362265 | 0.000000 |
| **11745** | 0.000000 | 0.519343 | 0.000000 | 0.106669 | 0.000000 | 0.074943 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.095545 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11746** | 0.000000 | 0.177108 | 0.000000 | 0.000000 | 0.000000 | 0.230014 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11747** | 0.000000 | 0.073392 | 0.000000 | 0.000000 | 0.139029 | 0.095316 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.244176 | 0.000000 |
| **11748** | 0.000000 | 0.000000 | 0.000000 | 0.073203 | 0.075016 | 0.000000 | 0.000000 | 0.000000 | 0.120565 | 0.000000 | ... | 0.000000 | 0.116925 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.123071 | 0.140383 | 0.000000 | 0.057364 |
| **11749** | 0.000000 | 0.092881 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.076894 | 0.073300 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11750** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.220394 | 0.000000 | 0.000000 | ... | 0.000000 | 0.198785 | 0.000000 | 0.000000 | 0.000000 | 0.212526 | 0.139488 | 0.000000 | 0.000000 | 0.000000 |
| **11751** | 0.000000 | 0.519343 | 0.000000 | 0.106669 | 0.000000 | 0.074943 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.095545 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11752** | 0.000000 | 0.177108 | 0.000000 | 0.000000 | 0.000000 | 0.230014 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11753** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11754** | 0.000000 | 0.000000 | 0.185595 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.164899 | 0.000000 | 0.103171 | 0.000000 | 0.000000 | 0.000000 |
| **11755** | 0.000000 | 0.083103 | 0.000000 | 0.153620 | 0.000000 | 0.000000 | 0.346014 | 0.000000 | 0.000000 | 0.106517 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.276486 | 0.000000 |
| **11756** | 0.000000 | 0.083103 | 0.000000 | 0.153620 | 0.000000 | 0.000000 | 0.346014 | 0.000000 | 0.000000 | 0.106517 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.276486 | 0.000000 |
| **11757** | 0.000000 | 0.000000 | 0.094151 | 0.000000 | 0.000000 | 0.000000 | 0.210357 | 0.000000 | 0.076908 | 0.000000 | ... | 0.073321 | 0.074587 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.219555 |
| **11758** | 0.000000 | 0.000000 | 0.000000 | 0.500378 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.224096 | 0.213623 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11759** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.142899 | 0.084264 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.190840 | 0.000000 | 0.085240 | 0.000000 | 0.106662 | 0.000000 | 0.000000 | 0.000000 |
| **11760** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.119549 | 0.227924 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11761** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.142899 | 0.084264 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.190840 | 0.000000 | 0.085240 | 0.000000 | 0.106662 | 0.000000 | 0.000000 | 0.000000 |
| **11762** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **11763** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.099153 | 0.000000 | 0.000000 | 0.000000 |

11764 rows × 162 columns

**TRAIN Grev\_Desc DATA - Getting the list of vocabulary**[**¶**](#1gf8i83)

In [206]:

print(tfidf\_Dis.vocabulary\_)

{'mortgage': 89, 'never': 92, 'payment': 105, 'finance': 56, 'loan': 81, 'home': 67, 'current': 36, 'additional': 2, 'late': 77, 'start': 141, 'process': 111, 'monthly': 87, 'close': 24, 'would': 158, 'need': 91, 'provide': 113, 'every': 51, 'document': 46, 'ask': 9, 'sign': 138, 'allow': 4, 'pay': 104, 'come': 25, 'way': 154, 'file': 55, 'complaint': 27, 'should': 136, 'even': 50, 'tell': 147, 'end': 49, 'email': 48, 'go': 64, 'phone': 107, 'receive': 117, 'say': 129, 'deny': 43, 'also': 5, 'reason': 116, 'state': 142, 'amount': 6, 'work': 157, 'now': 97, 'money': 86, 'get': 62, 'another': 7, 'financial': 57, 'submit': 145, 'back': 11, 'hold': 66, 'would not': 159, 'give': 63, 'card': 18, 'purchase': 114, 'include': 69, 'still': 144, 'owe': 102, 'due': 47, 'correct': 31, 'letter': 79, 'explain': 52, 'send': 132, 'however': 68, 'fee': 54, 'want': 153, 'payments': 106, 'call': 17, 'speak': 140, 'name': 90, 'months': 88, 'interest': 72, 'keep': 74, 'account': 1, 'make': 83, 'report': 122, 'first': 59, 'time': 148, 'issue': 73, 'since': 139, 'balance': 12, 'years': 161, 'open': 101, 'use': 152, 'able': 0, 'could': 32, 'offer': 99, 'well': 155, 'change': 20, 'find': 58, 'see': 130, 'credit card': 34, 'fail': 53, 'last': 76, 'place': 108, 'inform': 70, 'sell': 131, 'company': 26, 'new': 93, 'follow': 60, 'continue': 29, 'know': 75, 'right': 128, 'bank': 13, 'like': 80, 'us': 151, 'one': 100, 'many': 84, 'serve': 133, 'service': 134, 'refuse': 119, 'statement': 143, 'return': 127, 'property': 112, 'take': 146, 'charge': 21, 'believe': 14, 'case': 19, 'try': 150, 'full': 61, 'apply': 8, 'resolve': 125, 'several': 135, 'notice': 96, 'show': 137, 'past': 103, 'number': 98, 'do': 45, 'days': 41, 'customer': 37, 'later': 78, 'put': 115, 'day': 40, 'address': 3, 'nothing': 95, 'record': 118, 'customer service': 38, 'could not': 33, 'modification': 85, 'point': 110, 'please': 109, 'help': 65, 'write': 160, 'request': 123, 'remove': 121, 'business': 16, 'response': 126, 'contract': 30, 'information': 71, 'check': 22, 'attempt': 10, 'claim': 23, 'regard': 120, 'bill': 15, 'mail': 82, 'date': 39, 'credit report': 35, 'dispute': 44, 'not receive': 94, 'transfer': 149, 'contact': 28, 'without': 156, 'debt': 42, 'require': 124}

**TRAIN Grev\_Desc DATA - Soring the vocabulary with decresing order of frequency**[**¶**](#40ew0vw)

In [207]:

sorted(tfidf\_Dis.vocabulary\_.items(), key=**lambda** x: x[1])

Out[207]:

[('able', 0),  
 ('account', 1),  
 ('additional', 2),  
 ('address', 3),  
 ('allow', 4),  
 ('also', 5),  
 ('amount', 6),  
 ('another', 7),  
 ('apply', 8),  
 ('ask', 9),  
 ('attempt', 10),  
 ('back', 11),  
 ('balance', 12),  
 ('bank', 13),  
 ('believe', 14),  
 ('bill', 15),  
 ('business', 16),  
 ('call', 17),  
 ('card', 18),  
 ('case', 19),  
 ('change', 20),  
 ('charge', 21),  
 ('check', 22),  
 ('claim', 23),  
 ('close', 24),  
 ('come', 25),  
 ('company', 26),  
 ('complaint', 27),  
 ('contact', 28),  
 ('continue', 29),  
 ('contract', 30),  
 ('correct', 31),  
 ('could', 32),  
 ('could not', 33),  
 ('credit card', 34),  
 ('credit report', 35),  
 ('current', 36),  
 ('customer', 37),  
 ('customer service', 38),  
 ('date', 39),  
 ('day', 40),  
 ('days', 41),  
 ('debt', 42),  
 ('deny', 43),  
 ('dispute', 44),  
 ('do', 45),  
 ('document', 46),  
 ('due', 47),  
 ('email', 48),  
 ('end', 49),  
 ('even', 50),  
 ('every', 51),  
 ('explain', 52),  
 ('fail', 53),  
 ('fee', 54),  
 ('file', 55),  
 ('finance', 56),  
 ('financial', 57),  
 ('find', 58),  
 ('first', 59),  
 ('follow', 60),  
 ('full', 61),  
 ('get', 62),  
 ('give', 63),  
 ('go', 64),  
 ('help', 65),  
 ('hold', 66),  
 ('home', 67),  
 ('however', 68),  
 ('include', 69),  
 ('inform', 70),  
 ('information', 71),  
 ('interest', 72),  
 ('issue', 73),  
 ('keep', 74),  
 ('know', 75),  
 ('last', 76),  
 ('late', 77),  
 ('later', 78),  
 ('letter', 79),  
 ('like', 80),  
 ('loan', 81),  
 ('mail', 82),  
 ('make', 83),  
 ('many', 84),  
 ('modification', 85),  
 ('money', 86),  
 ('monthly', 87),  
 ('months', 88),  
 ('mortgage', 89),  
 ('name', 90),  
 ('need', 91),  
 ('never', 92),  
 ('new', 93),  
 ('not receive', 94),  
 ('nothing', 95),  
 ('notice', 96),  
 ('now', 97),  
 ('number', 98),  
 ('offer', 99),  
 ('one', 100),  
 ('open', 101),  
 ('owe', 102),  
 ('past', 103),  
 ('pay', 104),  
 ('payment', 105),  
 ('payments', 106),  
 ('phone', 107),  
 ('place', 108),  
 ('please', 109),  
 ('point', 110),  
 ('process', 111),  
 ('property', 112),  
 ('provide', 113),  
 ('purchase', 114),  
 ('put', 115),  
 ('reason', 116),  
 ('receive', 117),  
 ('record', 118),  
 ('refuse', 119),  
 ('regard', 120),  
 ('remove', 121),  
 ('report', 122),  
 ('request', 123),  
 ('require', 124),  
 ('resolve', 125),  
 ('response', 126),  
 ('return', 127),  
 ('right', 128),  
 ('say', 129),  
 ('see', 130),  
 ('sell', 131),  
 ('send', 132),  
 ('serve', 133),  
 ('service', 134),  
 ('several', 135),  
 ('should', 136),  
 ('show', 137),  
 ('sign', 138),  
 ('since', 139),  
 ('speak', 140),  
 ('start', 141),  
 ('state', 142),  
 ('statement', 143),  
 ('still', 144),  
 ('submit', 145),  
 ('take', 146),  
 ('tell', 147),  
 ('time', 148),  
 ('transfer', 149),  
 ('try', 150),  
 ('us', 151),  
 ('use', 152),  
 ('want', 153),  
 ('way', 154),  
 ('well', 155),  
 ('without', 156),  
 ('work', 157),  
 ('would', 158),  
 ('would not', 159),  
 ('write', 160),  
 ('years', 161)]

**TRAIN Grev\_Desc DATA - Word cloud for Disputed train data - Top 60 words**[**¶**](#2fk6b3p)

In [208]:

wc = WordCloud(max\_font\_size=500, background\_color="black",width=3000,height=3000, max\_words=60,relative\_scaling=0.1,normalize\_plurals=**False**).generate\_from\_frequencies(tfidf\_Dis.vocabulary\_)  
plt.figure( figsize=(13,10), facecolor='k')  
plt.imshow(wc,aspect=.5, interpolation='bilinear')  
plt.axis("off")  
plt.show()

**TRAIN Grev\_Desc DATA - Applying TF IDF vectorizer on undisputed train greviance values**[**¶**](#upglbi)

In [209]:

tfidf\_vectorizer\_unDis = TfidfVectorizer(ngram\_range=(1,3),min\_df=0.1,max\_df=0.5,max\_features=1000) *# stop\_words='english'*  
tfidf\_unDis= tfidf\_vectorizer\_unDis.fit(UnDisputed\_Grev.Grev\_Desc)   
tfidf\_train\_unDis = tfidf\_unDis.transform(UnDisputed\_Grev.Grev\_Desc)

**TRAIN Grev\_Desc DATA - Getting vocaboulary of undisputed train grievance text**[**¶**](#3ep43zb)

In [210]:

print(tfidf\_unDis.vocabulary\_)

{'name': 81, 'customer': 31, 'loan': 72, 'finance': 47, 'home': 59, 'new': 84, 'statement': 124, 'close': 19, 'show': 118, 'pay': 94, 'receive': 104, 'check': 17, 'amount': 3, 'date': 33, 'however': 60, 'payment': 95, 'would': 141, 'like': 71, 'know': 66, 'happen': 56, 'balance': 9, 'account': 1, 'call': 13, 'speak': 121, 'give': 54, 'mortgage': 80, 'sell': 113, 'never': 83, 'late': 68, 'change': 15, 'several': 116, 'time': 130, 'state': 123, 'send': 114, 'back': 8, 'use': 135, 'ask': 6, 'should': 117, 'do': 36, 'money': 77, 'explain': 43, 'go': 55, 'letter': 70, 'put': 103, 'also': 2, 'make': 74, 'now': 88, 'say': 111, 'get': 53, 'could': 26, 'find': 49, 'well': 138, 'start': 122, 'last': 67, 'still': 125, 'take': 127, 'us': 134, 'issue': 64, 'try': 133, 'payments': 96, 'could not': 27, 'would not': 142, 'modification': 76, 'process': 100, 'months': 79, 'continue': 24, 'document': 37, 'help': 57, 'tell': 128, 'need': 82, 'end': 40, 'contact': 23, 'today': 131, 'work': 140, 'monthly': 78, 'way': 137, 'keep': 65, 'past': 93, 'due': 38, 'bank': 10, 'information': 62, 'since': 120, 'day': 34, 'credit': 28, 'service': 115, 'point': 99, 'report': 107, 'file': 46, 'every': 42, 'years': 144, 'days': 35, 'apply': 5, 'even': 41, 'company': 21, 'follow': 51, 'remove': 106, 'credit report': 30, 'submit': 126, 'bill': 12, 'refuse': 105, 'see': 112, 'inform': 61, 'hold': 58, 'come': 20, 'email': 39, 'nothing': 86, 'first': 50, 'want': 136, 'resolve': 109, 'contract': 25, 'later': 69, 'card': 14, 'another': 4, 'mail': 73, 'able': 0, 'credit card': 29, 'transfer': 132, 'phone': 97, 'one': 91, 'fee': 45, 'owe': 92, 'request': 108, 'without': 139, 'please': 98, 'not receive': 85, 'provide': 101, 'charge': 16, 'purchase': 102, 'interest': 63, 'write': 143, 'full': 52, 'claim': 18, 'offer': 90, 'number': 89, 'customer service': 32, 'may': 75, 'complaint': 22, 'attempt': 7, 'return': 110, 'financial': 48, 'think': 129, 'sign': 119, 'notice': 87, 'fail': 44, 'believe': 11}

**TRAIN Grev\_Desc DATA - Sorting the list by frequencey**[**¶**](#1tuee74)

In [211]:

sorted(tfidf\_unDis.vocabulary\_.items(), key=**lambda** x: x[1])

Out[211]:

[('able', 0),  
 ('account', 1),  
 ('also', 2),  
 ('amount', 3),  
 ('another', 4),  
 ('apply', 5),  
 ('ask', 6),  
 ('attempt', 7),  
 ('back', 8),  
 ('balance', 9),  
 ('bank', 10),  
 ('believe', 11),  
 ('bill', 12),  
 ('call', 13),  
 ('card', 14),  
 ('change', 15),  
 ('charge', 16),  
 ('check', 17),  
 ('claim', 18),  
 ('close', 19),  
 ('come', 20),  
 ('company', 21),  
 ('complaint', 22),  
 ('contact', 23),  
 ('continue', 24),  
 ('contract', 25),  
 ('could', 26),  
 ('could not', 27),  
 ('credit', 28),  
 ('credit card', 29),  
 ('credit report', 30),  
 ('customer', 31),  
 ('customer service', 32),  
 ('date', 33),  
 ('day', 34),  
 ('days', 35),  
 ('do', 36),  
 ('document', 37),  
 ('due', 38),  
 ('email', 39),  
 ('end', 40),  
 ('even', 41),  
 ('every', 42),  
 ('explain', 43),  
 ('fail', 44),  
 ('fee', 45),  
 ('file', 46),  
 ('finance', 47),  
 ('financial', 48),  
 ('find', 49),  
 ('first', 50),  
 ('follow', 51),  
 ('full', 52),  
 ('get', 53),  
 ('give', 54),  
 ('go', 55),  
 ('happen', 56),  
 ('help', 57),  
 ('hold', 58),  
 ('home', 59),  
 ('however', 60),  
 ('inform', 61),  
 ('information', 62),  
 ('interest', 63),  
 ('issue', 64),  
 ('keep', 65),  
 ('know', 66),  
 ('last', 67),  
 ('late', 68),  
 ('later', 69),  
 ('letter', 70),  
 ('like', 71),  
 ('loan', 72),  
 ('mail', 73),  
 ('make', 74),  
 ('may', 75),  
 ('modification', 76),  
 ('money', 77),  
 ('monthly', 78),  
 ('months', 79),  
 ('mortgage', 80),  
 ('name', 81),  
 ('need', 82),  
 ('never', 83),  
 ('new', 84),  
 ('not receive', 85),  
 ('nothing', 86),  
 ('notice', 87),  
 ('now', 88),  
 ('number', 89),  
 ('offer', 90),  
 ('one', 91),  
 ('owe', 92),  
 ('past', 93),  
 ('pay', 94),  
 ('payment', 95),  
 ('payments', 96),  
 ('phone', 97),  
 ('please', 98),  
 ('point', 99),  
 ('process', 100),  
 ('provide', 101),  
 ('purchase', 102),  
 ('put', 103),  
 ('receive', 104),  
 ('refuse', 105),  
 ('remove', 106),  
 ('report', 107),  
 ('request', 108),  
 ('resolve', 109),  
 ('return', 110),  
 ('say', 111),  
 ('see', 112),  
 ('sell', 113),  
 ('send', 114),  
 ('service', 115),  
 ('several', 116),  
 ('should', 117),  
 ('show', 118),  
 ('sign', 119),  
 ('since', 120),  
 ('speak', 121),  
 ('start', 122),  
 ('state', 123),  
 ('statement', 124),  
 ('still', 125),  
 ('submit', 126),  
 ('take', 127),  
 ('tell', 128),  
 ('think', 129),  
 ('time', 130),  
 ('today', 131),  
 ('transfer', 132),  
 ('try', 133),  
 ('us', 134),  
 ('use', 135),  
 ('want', 136),  
 ('way', 137),  
 ('well', 138),  
 ('without', 139),  
 ('work', 140),  
 ('would', 141),  
 ('would not', 142),  
 ('write', 143),  
 ('years', 144)]

**TRAIN Grev\_Desc DATA - Creating word cloud for undisputed Grev desc text - Top 60 words**[**¶**](#4du1wux)

In [212]:

wc = WordCloud(max\_font\_size=500, background\_color="black",width=3000,height=3000, max\_words=60,relative\_scaling=0.1,normalize\_plurals=**False**).generate\_from\_frequencies(tfidf\_unDis.vocabulary\_)  
plt.figure( figsize=(13,10), facecolor='k')  
plt.imshow(wc,aspect=.5, interpolation='bilinear')  
plt.axis("off")  
plt.show()

**TEST Grev\_Desc DATA - Importing lemmatized Griveance Descprtion**[**¶**](#2szc72q)

In [213]:

clean\_test\_text = pd.read\_csv('lemm\_clean\_Test\_Grev\_All.csv')

In [214]:

clean\_test\_text.head()

Out[214]:

|  | **Unnamed: 0** | **0** |
| --- | --- | --- |
| **0** | 0 | currently mortgage flag star payment late reac... |
| **1** | 1 | auto finance possess subsequently sell husband... |
| **2** | 2 | fell behind payments back follow really unreas... |
| **3** | 3 | home parcel land first live home fail real est... |
| **4** | 4 | mortgage home assign loan transfer loan screw ... |

In [215]:

clean\_test\_text\_GrevDes = clean\_test\_text[['0']]

In [216]:

clean\_test\_text\_GrevDes.head()

Out[216]:

|  | **0** |
| --- | --- |
| **0** | currently mortgage flag star payment late reac... |
| **1** | auto finance possess subsequently sell husband... |
| **2** | fell behind payments back follow really unreas... |
| **3** | home parcel land first live home fail real est... |
| **4** | mortgage home assign loan transfer loan screw ... |

In [217]:

clean\_test\_text\_GrevDes.rename(columns={'0': 'Grev\_Desc'},inplace=**True**)

In [218]:

clean\_test\_text\_GrevDes.head()

Out[218]:

|  | **Grev\_Desc** |
| --- | --- |
| **0** | currently mortgage flag star payment late reac... |
| **1** | auto finance possess subsequently sell husband... |
| **2** | fell behind payments back follow really unreas... |
| **3** | home parcel land first live home fail real est... |
| **4** | mortgage home assign loan transfer loan screw ... |

**TEST Grev\_Desc DATA - Getting "Disputed" column from main test data frame¶**

In [219]:

test\_for\_target = pd.read\_csv('GrievancesData\_Test.csv')

In [220]:

test\_for\_target.shape

Out[220]:

(27954, 10)

In [221]:

test\_for\_target.columns

Out[221]:

Index(['GrievanceID', 'BankID', 'State', 'DateOfGrievance',  
 'Grievance\_Category', 'GrievanceDescription', 'LineOfBusiness',  
 'ResolutionComments', 'Disputed', 'DateOfResolution'],  
 dtype='object')

In [222]:

clean\_test\_text\_GrevDes['Disputed'] = test\_for\_target.Disputed

In [223]:

clean\_test\_text\_GrevDes.head()

Out[223]:

|  | **Grev\_Desc** | **Disputed** |
| --- | --- | --- |
| **0** | currently mortgage flag star payment late reac... | No |
| **1** | auto finance possess subsequently sell husband... | No |
| **2** | fell behind payments back follow really unreas... | No |
| **3** | home parcel land first live home fail real est... | No |
| **4** | mortgage home assign loan transfer loan screw ... | No |

**TEST Grev\_Desc DATA - Saperating Disputed and undisputed**[**¶**](#184mhaj)

In [224]:

Disputed\_test\_Grev = clean\_test\_text\_GrevDes.loc[clean\_test\_text\_GrevDes['Disputed'] == 'Yes']

In [225]:

Disputed\_test\_Grev.head()

Out[225]:

|  | **Grev\_Desc** | **Disputed** |
| --- | --- | --- |
| **7** | ca not decrease monthly payments try deal pati... | Yes |
| **8** | charge fee speak overnight change mortgage num... | Yes |
| **12** | forward explanation aware husband already file... | Yes |
| **26** | ca not decrease monthly payments try deal pati... | Yes |
| **34** | hello veteran unemployed time injuries militar... | Yes |

In [226]:

UnDisputed\_test\_Grev = clean\_test\_text\_GrevDes.loc[clean\_test\_text\_GrevDes['Disputed'] == 'No']

In [227]:

UnDisputed\_test\_Grev.head()

Out[227]:

|  | **Grev\_Desc** | **Disputed** |
| --- | --- | --- |
| **0** | currently mortgage flag star payment late reac... | No |
| **1** | auto finance possess subsequently sell husband... | No |
| **2** | fell behind payments back follow really unreas... | No |
| **3** | home parcel land first live home fail real est... | No |
| **4** | mortgage home assign loan transfer loan screw ... | No |

**TEST Grev\_Desc DATA - TF IDF on Disputed text on test data**[**¶**](#3s49zyc)

In [228]:

tfidf\_vectorizer\_test\_des = TfidfVectorizer(ngram\_range=(1,3),min\_df=0.1,max\_df=0.5,max\_features=500) *# stop\_words='english'*  
tfidf\_test\_Dis= tfidf\_vectorizer\_test\_des.fit(Disputed\_test\_Grev.Grev\_Desc)   
tfidf\_test\_Disputed = tfidf\_test\_Dis.transform(Disputed\_test\_Grev.Grev\_Desc)

**TEST Grev\_Desc DATA - Getting the vocabulary**[**¶**](#279ka65)

In [229]:

print(tfidf\_test\_Dis.vocabulary\_)

{'ca': 15, 'monthly': 86, 'payments': 104, 'try': 148, 'patient': 101, 'regard': 117, 'loan': 82, 'take': 143, 'name': 89, 'phone': 105, 'give': 63, 'money': 85, 'amount': 5, 'want': 151, 'document': 46, 'call': 17, 'start': 136, 'letter': 80, 'well': 153, 'financial': 57, 'put': 112, 'time': 145, 'say': 125, 'nothing': 93, 'file': 55, 'agree': 3, 'ask': 8, 'even': 50, 'send': 128, 'without': 154, 'home': 67, 'need': 90, 'make': 84, 'go': 64, 'know': 75, 'one': 98, 'payment': 103, 'number': 96, 'check': 22, 'account': 1, 'receive': 114, 'mail': 83, 'state': 137, 'find': 58, 'write': 158, 'date': 40, 'still': 139, 'also': 4, 'end': 49, 'contact': 28, 'refuse': 116, 'every': 51, 'close': 24, 'use': 150, 'pay': 102, 'months': 87, 'never': 91, 'do': 45, 'sign': 133, 'way': 152, 'please': 107, 'help': 65, 'ca not': 16, 'charge': 21, 'fee': 54, 'speak': 135, 'change': 20, 'mortgage': 88, 'service': 129, 'another': 6, 'keep': 74, 'bill': 14, 'record': 115, 'should': 131, 'complaint': 27, 'believe': 13, 'us': 149, 'interest': 72, 'balance': 11, 'apply': 7, 'sell': 127, 'get': 62, 'show': 132, 'tell': 144, 'could': 32, 'however': 68, 'continue': 29, 'late': 77, 'additional': 2, 'statement': 138, 'days': 42, 'right': 124, 'leave': 79, 'first': 59, 'now': 95, 'since': 134, 'notice': 94, 'come': 25, 'work': 155, 'fail': 53, 'last': 76, 'issue': 73, 'would': 156, 'company': 26, 'back': 10, 'resolve': 122, 'follow': 60, 'provide': 110, 'explain': 52, 'place': 106, 'customer': 38, 'customer service': 39, 'card': 18, 'due': 47, 'new': 92, 'past': 100, 'request': 120, 'current': 37, 'submit': 142, 'transfer': 147, 'see': 126, 'today': 146, 'several': 130, 'bank': 12, 'hold': 66, 'information': 71, 'full': 61, 'return': 123, 'attempt': 9, 'email': 48, 'claim': 23, 'could not': 33, 'credit': 34, 'report': 119, 'contract': 30, 'dispute': 44, 'include': 69, 'remove': 118, 'credit report': 36, 'would not': 157, 'day': 41, 'owe': 99, 'years': 159, 'later': 78, 'inform': 70, 'student': 140, 'rate': 113, 'student loan': 141, 'purchase': 111, 'like': 81, 'require': 121, 'correct': 31, 'finance': 56, 'able': 0, 'process': 109, 'debt': 43, 'case': 19, 'credit card': 35, 'point': 108, 'offer': 97}

**TEST Grev\_Desc DATA - Sorting the word with freqency**[**¶**](#meukdy)

In [230]:

sorted(tfidf\_test\_Dis.vocabulary\_.items(), key=**lambda** x: x[1])

Out[230]:

[('able', 0),  
 ('account', 1),  
 ('additional', 2),  
 ('agree', 3),  
 ('also', 4),  
 ('amount', 5),  
 ('another', 6),  
 ('apply', 7),  
 ('ask', 8),  
 ('attempt', 9),  
 ('back', 10),  
 ('balance', 11),  
 ('bank', 12),  
 ('believe', 13),  
 ('bill', 14),  
 ('ca', 15),  
 ('ca not', 16),  
 ('call', 17),  
 ('card', 18),  
 ('case', 19),  
 ('change', 20),  
 ('charge', 21),  
 ('check', 22),  
 ('claim', 23),  
 ('close', 24),  
 ('come', 25),  
 ('company', 26),  
 ('complaint', 27),  
 ('contact', 28),  
 ('continue', 29),  
 ('contract', 30),  
 ('correct', 31),  
 ('could', 32),  
 ('could not', 33),  
 ('credit', 34),  
 ('credit card', 35),  
 ('credit report', 36),  
 ('current', 37),  
 ('customer', 38),  
 ('customer service', 39),  
 ('date', 40),  
 ('day', 41),  
 ('days', 42),  
 ('debt', 43),  
 ('dispute', 44),  
 ('do', 45),  
 ('document', 46),  
 ('due', 47),  
 ('email', 48),  
 ('end', 49),  
 ('even', 50),  
 ('every', 51),  
 ('explain', 52),  
 ('fail', 53),  
 ('fee', 54),  
 ('file', 55),  
 ('finance', 56),  
 ('financial', 57),  
 ('find', 58),  
 ('first', 59),  
 ('follow', 60),  
 ('full', 61),  
 ('get', 62),  
 ('give', 63),  
 ('go', 64),  
 ('help', 65),  
 ('hold', 66),  
 ('home', 67),  
 ('however', 68),  
 ('include', 69),  
 ('inform', 70),  
 ('information', 71),  
 ('interest', 72),  
 ('issue', 73),  
 ('keep', 74),  
 ('know', 75),  
 ('last', 76),  
 ('late', 77),  
 ('later', 78),  
 ('leave', 79),  
 ('letter', 80),  
 ('like', 81),  
 ('loan', 82),  
 ('mail', 83),  
 ('make', 84),  
 ('money', 85),  
 ('monthly', 86),  
 ('months', 87),  
 ('mortgage', 88),  
 ('name', 89),  
 ('need', 90),  
 ('never', 91),  
 ('new', 92),  
 ('nothing', 93),  
 ('notice', 94),  
 ('now', 95),  
 ('number', 96),  
 ('offer', 97),  
 ('one', 98),  
 ('owe', 99),  
 ('past', 100),  
 ('patient', 101),  
 ('pay', 102),  
 ('payment', 103),  
 ('payments', 104),  
 ('phone', 105),  
 ('place', 106),  
 ('please', 107),  
 ('point', 108),  
 ('process', 109),  
 ('provide', 110),  
 ('purchase', 111),  
 ('put', 112),  
 ('rate', 113),  
 ('receive', 114),  
 ('record', 115),  
 ('refuse', 116),  
 ('regard', 117),  
 ('remove', 118),  
 ('report', 119),  
 ('request', 120),  
 ('require', 121),  
 ('resolve', 122),  
 ('return', 123),  
 ('right', 124),  
 ('say', 125),  
 ('see', 126),  
 ('sell', 127),  
 ('send', 128),  
 ('service', 129),  
 ('several', 130),  
 ('should', 131),  
 ('show', 132),  
 ('sign', 133),  
 ('since', 134),  
 ('speak', 135),  
 ('start', 136),  
 ('state', 137),  
 ('statement', 138),  
 ('still', 139),  
 ('student', 140),  
 ('student loan', 141),  
 ('submit', 142),  
 ('take', 143),  
 ('tell', 144),  
 ('time', 145),  
 ('today', 146),  
 ('transfer', 147),  
 ('try', 148),  
 ('us', 149),  
 ('use', 150),  
 ('want', 151),  
 ('way', 152),  
 ('well', 153),  
 ('without', 154),  
 ('work', 155),  
 ('would', 156),  
 ('would not', 157),  
 ('write', 158),  
 ('years', 159)]

**TEST Grev\_Desc DATA - Creating a word cloud Disputed grev, under test data - Top 60 words**[**¶**](#36ei31r)

In [231]:

wc = WordCloud(max\_font\_size=500, background\_color="black",width=3000,height=3000, max\_words=60,relative\_scaling=0.1,normalize\_plurals=**False**).generate\_from\_frequencies(tfidf\_test\_Dis.vocabulary\_)  
plt.figure( figsize=(13,10), facecolor='k')  
plt.imshow(wc,aspect=.5, interpolation='bilinear')  
plt.axis("off")  
plt.show()

**TEST Grev\_Desc DATA - Applying tf idf vectorizer on undisputed Grev Desc**[**¶**](#1ljsd9k)

In [232]:

tfidf\_vectorizer\_test\_undes = TfidfVectorizer(ngram\_range=(1,3),min\_df=0.1,max\_df=0.5,max\_features=500) *# stop\_words='english'*  
tfidf\_test\_unDis= tfidf\_vectorizer\_test\_undes.fit(UnDisputed\_test\_Grev.Grev\_Desc)   
tfidf\_test\_unDisputed = tfidf\_test\_unDis.transform(UnDisputed\_test\_Grev.Grev\_Desc)

**TEST Grev\_Desc DATA - Checking the vocabulary**[**¶**](#45jfvxd)

In [233]:

voc\_test = tfidf\_test\_unDis.vocabulary\_

In [234]:

print(voc\_test)

{'mortgage': 72, 'payment': 87, 'late': 62, 'loan': 66, 'now': 79, 'first': 46, 'time': 118, 'due': 37, 'issue': 58, 'payments': 88, 'receive': 96, 'notice': 78, 'go': 50, 'service': 104, 'interest': 57, 'company': 20, 'nothing': 77, 'account': 1, 'tell': 117, 'call': 13, 'back': 7, 'credit': 27, 'report': 98, 'show': 107, 'amount': 3, 'owe': 84, 'give': 49, 'take': 116, 'credit report': 29, 'years': 129, 'never': 75, 'months': 71, 'also': 2, 'income': 54, 'home': 52, 'date': 32, 'letter': 64, 'contract': 24, 'contact': 22, 'inform': 55, 'need': 74, 'phone': 89, 'us': 120, 'could': 25, 'make': 67, 'since': 108, 'get': 48, 'help': 51, 'even': 40, 'send': 103, 'another': 4, 'put': 94, 'say': 101, 'bill': 10, 'ask': 6, 'would': 127, 'days': 34, 'one': 82, 'know': 60, 'number': 80, 'name': 73, 'fee': 43, 'request': 100, 'information': 56, 'file': 44, 'do': 35, 'should': 106, 'refuse': 97, 'card': 14, 'provide': 92, 'rate': 95, 'purchase': 93, 'balance': 8, 'complaint': 21, 'several': 105, 'still': 113, 'apply': 5, 'credit card': 28, 'like': 65, 'full': 47, 'charge': 16, 'offer': 81, 'use': 121, 'customer': 30, 'student': 114, 'continue': 23, 'find': 45, 'however': 53, 'close': 18, 'speak': 109, 'representative': 99, 'keep': 59, 'customer service': 31, 'student loan': 115, 'statement': 112, 'document': 36, 'try': 119, 'change': 15, 'new': 76, 'day': 33, 'way': 123, 'without': 125, 'work': 126, 'state': 111, 'later': 63, 'process': 91, 'past': 85, 'every': 41, 'would not': 128, 'start': 110, 'want': 122, 'please': 90, 'could not': 26, 'money': 69, 'monthly': 70, 'bank': 9, 'come': 19, 'end': 39, 'well': 124, 'able': 0, 'last': 61, 'email': 38, 'see': 102, 'explain': 42, 'check': 17, 'ca': 11, 'ca not': 12, 'option': 83, 'make payments': 68, 'patient': 86}

**TEST Grev\_Desc DATA - Sorting the vocabulary with frequency**[**¶**](#2koq656)

In [235]:

sorted(voc\_test.items(), key=**lambda** x: x[1])

Out[235]:

[('able', 0),  
 ('account', 1),  
 ('also', 2),  
 ('amount', 3),  
 ('another', 4),  
 ('apply', 5),  
 ('ask', 6),  
 ('back', 7),  
 ('balance', 8),  
 ('bank', 9),  
 ('bill', 10),  
 ('ca', 11),  
 ('ca not', 12),  
 ('call', 13),  
 ('card', 14),  
 ('change', 15),  
 ('charge', 16),  
 ('check', 17),  
 ('close', 18),  
 ('come', 19),  
 ('company', 20),  
 ('complaint', 21),  
 ('contact', 22),  
 ('continue', 23),  
 ('contract', 24),  
 ('could', 25),  
 ('could not', 26),  
 ('credit', 27),  
 ('credit card', 28),  
 ('credit report', 29),  
 ('customer', 30),  
 ('customer service', 31),  
 ('date', 32),  
 ('day', 33),  
 ('days', 34),  
 ('do', 35),  
 ('document', 36),  
 ('due', 37),  
 ('email', 38),  
 ('end', 39),  
 ('even', 40),  
 ('every', 41),  
 ('explain', 42),  
 ('fee', 43),  
 ('file', 44),  
 ('find', 45),  
 ('first', 46),  
 ('full', 47),  
 ('get', 48),  
 ('give', 49),  
 ('go', 50),  
 ('help', 51),  
 ('home', 52),  
 ('however', 53),  
 ('income', 54),  
 ('inform', 55),  
 ('information', 56),  
 ('interest', 57),  
 ('issue', 58),  
 ('keep', 59),  
 ('know', 60),  
 ('last', 61),  
 ('late', 62),  
 ('later', 63),  
 ('letter', 64),  
 ('like', 65),  
 ('loan', 66),  
 ('make', 67),  
 ('make payments', 68),  
 ('money', 69),  
 ('monthly', 70),  
 ('months', 71),  
 ('mortgage', 72),  
 ('name', 73),  
 ('need', 74),  
 ('never', 75),  
 ('new', 76),  
 ('nothing', 77),  
 ('notice', 78),  
 ('now', 79),  
 ('number', 80),  
 ('offer', 81),  
 ('one', 82),  
 ('option', 83),  
 ('owe', 84),  
 ('past', 85),  
 ('patient', 86),  
 ('payment', 87),  
 ('payments', 88),  
 ('phone', 89),  
 ('please', 90),  
 ('process', 91),  
 ('provide', 92),  
 ('purchase', 93),  
 ('put', 94),  
 ('rate', 95),  
 ('receive', 96),  
 ('refuse', 97),  
 ('report', 98),  
 ('representative', 99),  
 ('request', 100),  
 ('say', 101),  
 ('see', 102),  
 ('send', 103),  
 ('service', 104),  
 ('several', 105),  
 ('should', 106),  
 ('show', 107),  
 ('since', 108),  
 ('speak', 109),  
 ('start', 110),  
 ('state', 111),  
 ('statement', 112),  
 ('still', 113),  
 ('student', 114),  
 ('student loan', 115),  
 ('take', 116),  
 ('tell', 117),  
 ('time', 118),  
 ('try', 119),  
 ('us', 120),  
 ('use', 121),  
 ('want', 122),  
 ('way', 123),  
 ('well', 124),  
 ('without', 125),  
 ('work', 126),  
 ('would', 127),  
 ('would not', 128),  
 ('years', 129)]

**TEST Grev\_Desc DATA - Word cloud - Top 60 words**[**¶**](#zu0gcz)

In [236]:

wc = WordCloud(max\_font\_size=500, background\_color="black",width=3000,height=3000, max\_words=60,relative\_scaling=0.1,normalize\_plurals=**False**).generate\_from\_frequencies(tfidf\_test\_unDis.vocabulary\_)  
plt.figure( figsize=(13,10), facecolor='k')  
plt.imshow(wc,aspect=.5, interpolation='bilinear')  
plt.axis("off")  
plt.show()

**Text mining approach -**[**¶**](#3jtnz0s)

I have extracted grievance description column and the Disputed column to form key word and phrases for Disputed and non-disputed grievances.

* After pre-processing the text, under grievance description column,i have used tiidf vectorizer to find the key phrases.
* Taking train data, and separating disputed and undisputed grievances, for Griveance description columns alone, which was pre-processed.
* With tfidf transformation, extracted the list of words and phrases and ranked him according to the frequency of occurrences.
* Formed a word cloud,highlighting the top 60 words under the category.

**What more could I have done?**[**¶**](#1yyy98l)

* Utilize the corpse of word from the Federal Trade Commission Act.