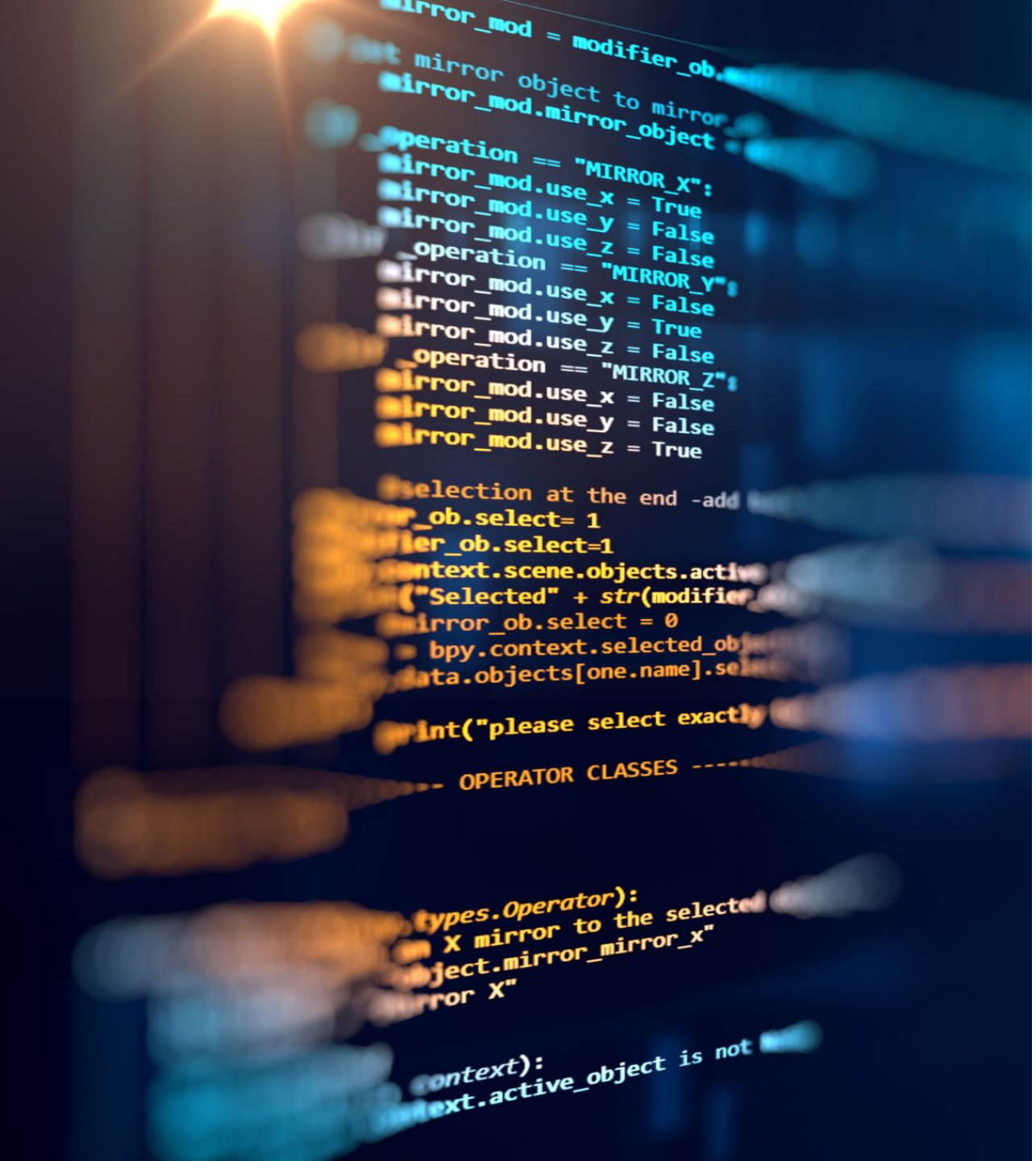


Text Mining using NLTK

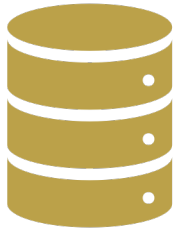
SUBMITTED BY:

SEERAT CHHABRA

Data Summary

Customer Comments

- Unstructured data
- 2070 customers comments



Customer data

- Structured data
- 2070 customers
- 17 attributes – Sex,Status,Children,Est_Income, Car_Owner, Usage,Age, RatePlan, LongDistance, International,Local,Dropped,Paymethod, LocalBilltype, LongDistanceBilltype
- Target column - Churn

Customers Comments Processing and Model

Steps for processing Text Data

- Stemming
  - Snowball
  - Porter
  - Lancaster
- Lemmatize

- Filter - SelectKBest
- Wrapper-
- Step Forward Feature Selection

Tokenize Words

```
[ ] #tokenise the sentences into words
    textData['CommentsTokenized'] = textData['Comments'].apply(word_tokenize)
    print(textData['CommentsTokenized'])
```

Breaking down full sentences into words

```
0       [Does, not, like, the, way, the, phone, works, ., It, is, to, difficult, compared, to, his, last, phone, .]
1       [Wanted, to, know, the, nearest, store, location, ., Wants, to, buy, aditional, accessories, .]
2       [Wants, to, know, how, to, do, text, messaging, ., Referred, him, to, website, .]
3       [Asked, how, to, disable, call, waiting, ., referred, him, to, web, site, .]
4       [Needs, help, learning, how, to, use, the, phone, ., I, suggested, he, go, back, to, the, store, and, have, the, rep, teach, him, .]
                                                                ...
2065    [Needed, help, figuring, out, his, bill, ., I, explained, our, minute, charges, .]
2066    [He, lost, his, phone, and, called, to, cancel, service, ., I, told, him, we, would, suspend, until, we, hear, back, from, him, ., He, will, contact, us, soon, .]
2067    [Lost, the, directions, to, phone, and, wants, another, manual, ., I, referred, him, to, web, site, .]
2068    [Wants, to, change, address, .]
2069    [He, lost, his, phone, and, called, to, cancel, service, ., I, told, him, we, would, suspend, until, we, hear, back, from, him, ., He, will, contact, us, soon, .]
Name: CommentsTokenized, Length: 2070, dtype: object
```

Snowball Stemming

```
#Snowball stemming
# Use English stemmer.
stemmer = SnowballStemmer("english")
newTextData=pd.DataFrame()
newTextData=textData.drop(columns=["CommentsTokenized","Comments"])
newTextData['CommentsTokenizedStemmedSnowball'] = textData['CommentsTokenized'].apply(lambda x: [stemmer.stem(y) for y in x])
print('-------------Snowball Stemming------------------')
newTextData['CommentsTokenizedStemmedSnowball']
#Join stemmed strings
newTextData['CommentsTokenizedStemmedSnowball'] = newTextData['CommentsTokenizedStemmedSnowball'].apply(lambda x: " ".join(x))
newTextData['CommentsTokenizedStemmedSnowball']
```

```
-------------Snowball Stemming------------------
0       doe not like the way the phone work . it is to difficult compar to his last phone .
1       want to know the nearest store locat . want to buy adit accessori .
2       want to know how to do text messag . refer him to websit .
3       ask how to disabl call wait . refer him to web site .
4       need help learn how to use the phone . i suggest he go back to the store and have the rep teach him .
                                                ...
2065    need help figur out his bill . i explain our minut charg .
2066    he lost his phone and call to cancel servic . i told him we would suspend until we hear back from him . he will contact us soon .
2067    lost the direct to phone and want anoth manual . i refer him to web site .
2068    want to chang address .
2069    he lost his phone and call to cancel servic . i told him we would suspend until we hear back from him . he will contact us soon .
Name: CommentsTokenizedStemmedSnowball, Length: 2070, dtype: object
```

Converting all words into their root words

Term Document Vector - Snowball

```
[ ] #Do Bag-Of-Words model - Term - Document Matrix - Snowball
    #Learn the vocabulary dictionary and return term-document matrix.
    #count_vect = CountVectorizer(stop_words=None)
    count_vect = CountVectorizer(stop_words='english',lowercase=False)
    count_vect
    TD_counts = count_vect.fit_transform(newTextData.CommentsTokenizedStemmedSnowball)
    print("Vocabulary ",(count_vect.vocabulary_))
    print("Number of words in vocabulary",len(count_vect.vocabulary_))
    print("Shape of text vector",TD_counts.shape)
    DF_TD_Counts=pd.DataFrame(TD_counts.toarray(), columns= count_vect.get_feature_names())
    print(DF_TD_Counts)
```

```
Vocabulary  {'doe': 98, 'like': 170, 'way': 337, 'phone': 220, 'work': 347, 'difficult': 94, 'compar': 60, 'want': 335, 'know': 164, 'nea
Number of words in vocabulary 354
Shape of text vector (2070, 354)
      3399  3g  abysm  access  accessori  ...  worst  wrong  xvyx  year  york
0        0   0      0       0          0  ...      0      0     0     0     0
1        0   0      0       0          1  ...      0      0     0     0     0
2        0   0      0       0          0  ...      0      0     0     0     0
3        0   0      0       0          0  ...      0      0     0     0     0
4        0   0      0       0          0  ...      0      0     0     0     0
...    ...  ..    ...     ...        ...  ...    ...    ...   ...   ...   ...
2065     0   0      0       0          0  ...      0      0     0     0     0
2066     0   0      0       0          0  ...      0      0     0     0     0
2067     0   0      0       0          0  ...      0      0     0     0     0
2068     0   0      0       0          0  ...      0      0     0     0     0
2069     0   0      0       0          0  ...      0      0     0     0     0

[2070 rows x 354 columns]
```

Removed stop words

Snowball Created 354 words

Creating a matrix for all the broken words and converting into 0,1 form for computation

TF-IDF Matrix

```
[ ]  #Compute TF-IDF Matrix
     tfidf_transformer = TfidfTransformer()
     X_train_tfidf = tfidf_transformer.fit_transform(DF_TD_Counts)
     print(X_train_tfidf.shape)
     DF_TF_IDF=pd.DataFrame(X_train_tfidf.toarray(),columns= DF_TD_Counts.columns)
     print(DF_TF_IDF)
```

```
(2070, 354)
      3399   3g  abysm  access  accessori  ...  worst  wrong  xvyx  year  york
0      0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0
1      0.0  0.0    0.0     0.0    0.27568  ...    0.0    0.0   0.0   0.0   0.0
2      0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0
3      0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0
4      0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0
...    ...  ...    ...     ...        ...  ...    ...    ...   ...   ...   ...
2065   0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0
2066   0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0
2067   0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0
2068   0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0
2069   0.0  0.0    0.0     0.0    0.00000  ...    0.0    0.0   0.0   0.0   0.0

[2070 rows x 354 columns]
```

Creating a TF-IDF matrix which is high for words which appear less in comments and less for words which are very commonly used

Feature Selection – Filter Method

Feature Selection – Filter method

```
#Feature selection
selector = SelectKBest(score_func=chi2, k=30)
selector_fit = selector.fit(DF_TF_IDF,y_train)
#get scores along with features
names = DF_TF_IDF.columns.values[selector.get_support()]
scores = selector.scores_[selector.get_support()]
names_scores = list(zip(names, scores))
ns_df = pd.DataFrame(data = names_scores, columns= ['Feat_names','F_Scores'])
ns_df_sorted = ns_df.sort_values(['F_Scores','Feat_names'], ascending = [False, True])
print(ns_df_sorted)


new_DF_TF_IDF = selector.fit_transform(DF_TF_IDF,y_train)
print("Shape of TF-IDF after feature selection", new_DF_TF_IDF.shape)

DF_TF_IDF_SelectedFeatures= pd.DataFrame(new_DF_TF_IDF, columns = list(DF_TF_IDF.columns[selector.get_support(indices=True)]) )
print(DF_TF_IDF_SelectedFeatures)
```

```
   Feat_names  F_Scores
19  receiv     4.025158
26  transeff   3.563271
1   alway      3.464947
23  screen     3.174275
16  ot         2.574465
27  turn       2.574465
5   continu    2.468678
9   figur      2.306164
7   explain    2.162567
3   charg      2.145596
8   famili     1.837066
28  unlimit    1.828956
13  market     1.818156
17  peopl      1.818156
22  rid        1.818156
25  sold       1.818156
10  hochi      1.556822
15  momma      1.556822
14  minut      1.547129
20  relat      1.489672
0   adress     1.446067
6   current    1.435394
21  result     1.435394
2   chang      1.404096
18  plan       1.382037
12  manag      1.381503
24  signal     1.379083
29  weak       1.379083
4   charger    1.297766
11  internet   1.294745
Shape of TF-IDF after feature selection (2070, 30)
      adress  alway     chang     charg  ...  transeff  turn  unlimit  weak
```

Using SelectKbest filter which ranks features on chi square score. Used top 30 features
Converting from 354 features to 30!

Split data into Train & Test

```
[ ] #split data into training and test
    #split the entire training data in training and test data
    X_Train, X_Test, Y_Train, Y_Test = train_test_split( DF_TF_IDF_SelectedFeatures, y_train, test_size=0.20, random_state=42)
    print("Initial shape for entire data:",DF_TF_IDF_SelectedFeatures.shape)
    print("Shape of new training data:", X_Train.shape)
    print("Shape of new test split data:", X_Test.shape)
```

```
Initial shape for entire data: (2070, 30)
Shape of new training data: (1656, 30)
Shape of new test split data: (414, 30)
```

Split data as 80-20% into Train and Test data

Classification Model on only comments

Precision important metric here- Company wants higher % of correct prediction for people churning so that they can invest on right people to save them rather than investing on people who won't churn!

```
#Construct a Random Forest Classifier on text data
rfc=RandomForestClassifier()
rfc.fit(X_Train,Y_Train)

#Prediction on training data
pred_rf=pd.DataFrame(rfc.predict(X_Train),columns=["Prediction"])
print("----------------------------Random Forest: Training Data----------------------------------
print("Accuracy Score on training data using Random Forest:",accuracy_score(Y_Train,pred_rf["Pred
print("Confusion Matrix on training data using Random Forest\n", confusion_matrix(Y_Train,pred_rf

# prediction on test data
pred_rf=pd.DataFrame(rfc.predict(X_Test),columns=["Prediction"])
print("----------------------------Random Forest: Test Data--------------------------------------
print("Accuracy Score on test data using Random Forest:",accuracy_score(Y_Test,pred_rf["Predictio
print("Balanced Accuracy Score on test data using Random Forest:",balanced_accuracy_score(Y_Test,
print("Confusion Matrix on test data using Random Forest\n", confusion_matrix(Y_Test,pred_rf["Pre
print("Classification report on test data using Random Forest\n", classification_report(Y_Test,pr
```

```
----------------------------Random Forest: Training Data----------------------------------

Accuracy Score on training data using Random Forest: 0.6280193236714976
Confusion Matrix on training data using Random Forest
 [[ 74 573]
 [ 43 966]]
----------------------------Random Forest: Test Data--------------------------------------

Accuracy Score on test data using Random Forest: 0.6231884057971014
Balanced Accuracy Score on test data using Random Forest: 0.5217725346353069
Confusion Matrix on test data using Random Forest
 [[ 16 141]
 [ 15 242]]
Classification report on test data using Random Forest
              precision    recall  f1-score   support

   Cancelled       0.52      0.10      0.17       157
     Current       0.63      0.94      0.76       257

    accuracy                           0.62       414
   macro avg       0.57      0.52      0.46       414
weighted avg       0.59      0.62      0.53       414
```

Customer Data Model

Processing of customer data

1

```
CustInfoData = CustInfoData.drop(columns=["TARGET"])
#Do one Hot encoding for categorical features
X_cat = ["Sex","Status","Car_Owner","Paymethod","LocalBilltype","LongDistanceBilltype"]
#X_cat = CustInfoData.select_dtypes(include=['object'])
customer_one_hot = pd.get_dummies(CustInfoData,columns=X_cat)

customer_one_hot = pd.DataFrame(customer_one_hot)
print(customer_one_hot.head())
```

```
   ID  ...  LongDistanceBilltype_Standard
0   1  ...                              0
1   6  ...                              1
2   8  ...                              1
3  11  ...                              1
4  14  ...                              0

[5 rows x 24 columns]
```

One hot encoding of categorical variables

2

```
#split data into training and test
#split the entire training data in training and test data
X_Train, X_Test, Y_Train, Y_Test = train_test_split(customer_one_hot, y_train, test_size=0.20, random_state=42)
print("Initial shape for entire data:",DF_TF_IDF_SelectedFeatures.shape)
print("Shape of new training data:", X_Train.shape)
print("Shape of new test split data:", X_Test.shape)
```

```
Initial shape for entire data: (2070, 30)
Shape of new training data: (1656, 24)
Shape of new test split data: (414, 24)
```

Split data as 80-20% into Train and Test data

Classification Model only customer data

Precision important metric here- Company wants higher % of correct prediction for people churning so that they can invest on right people to save them rather than investing on people who won't churn!

```
[ ] rfc=RandomForestClassifier()
    rfc.fit(X_Train,Y_Train)

    #Prediction on training data
    pred_rf=pd.DataFrame(rfc.predict(X_Train),columns=["Prediction"])
    print("----------------------------Random Forest: Training Data----------------------------------------------\n")
    print("Accuracy Score on training data using Random Forest:",accuracy_score(Y_Train,pred_rf["Prediction"]))
    print("Confusion Matrix on training data using Random Forest\n", confusion_matrix(Y_Train,pred_rf["Prediction"]))

    # prediction on test data
    pred_rf=pd.DataFrame(rfc.predict(X_Test),columns=["Prediction"])
    print("----------------------------Random Forest: Test Data----------------------------------------------\n")
    print("Accuracy Score on test data using Random Forest:",accuracy_score(Y_Test,pred_rf["Prediction"]))
    print("Balanced Accuracy Score on test data using Random Forest:",balanced_accuracy_score(Y_Test,pred_rf["Prediction"]))
    print("Confusion Matrix on test data using Random Forest\n", confusion_matrix(Y_Test,pred_rf["Prediction"]))
    print("Classification report on test data using Random Forest\n", classification_report(Y_Test,pred_rf["Prediction"]))
```

```
----------------------------Random Forest: Training Data----------------------------------------------

Accuracy Score on training data using Random Forest: 0.9897342995169082
Confusion Matrix on training data using Random Forest
 [[643   4]
 [ 13 996]]
----------------------------Random Forest: Test Data----------------------------------------------

Accuracy Score on test data using Random Forest: 0.8599033816425121
Balanced Accuracy Score on test data using Random Forest: 0.854940642890778
Confusion Matrix on test data using Random Forest
 [[131  26]
 [ 32 225]]
Classification report on test data using Random Forest
              precision    recall  f1-score   support

   Cancelled       0.80      0.83      0.82       157
     Current       0.90      0.88      0.89       257

    accuracy                           0.86       414
   macro avg       0.85      0.85      0.85       414
weighted avg       0.86      0.86      0.86       414
```

Comments + Customer Data Model

Processing of customer data

1

```
#Merge files comments and customer data
combined=pd.concat([customer_one_hot, DF_TF_IDF_SelectedFeatures], axis=1)
print(combined.shape)
print(combined)
```

```
(2070, 54)
          ID  Children  Est_Income    Usage  ...  transeff  turn  unlimit  weak
0          1         1    38000.00   229.64  ...       0.0   0.0      0.0   0.0
1          6         2    29616.00    75.29  ...       0.0   0.0      0.0   0.0
2          8         0    19732.80    47.25  ...       0.0   0.0      0.0   0.0
3         11         2       96.33    59.01  ...       0.0   0.0      0.0   0.0
4         14         2    52004.80    28.14  ...       0.0   0.0      0.0   0.0
...       ..       ..         ...      ...  ...       ...   ...      ...   ...
2065    3821         0    78851.30    29.04  ...       0.0   0.0      0.0   0.0
2066    3822         1    17540.70    36.20  ...       0.0   0.0      0.0   0.0
2067    3823         0    83891.90    74.40  ...       0.0   0.0      0.0   0.0
2068    3824         2    28220.80    38.95  ...       0.0   0.0      0.0   0.0
2069    3825         0    28589.10   100.28  ...       0.0   0.0      0.0   0.0

[2070 rows x 54 columns]
```

Combine comments (TD-IDF) matrix and customer data

2

```
#split data into training and test
#split the entire training data in training and test data
X_Train, X_Test, Y_Train, Y_Test = train_test_split( combined, y_train, test_size=0.20, random_state=42)
print("Initial shape for entire data:",DF_TF_IDF_SelectedFeatures.shape)
print("Shape of new training data:", X_Train.shape)
print("Shape of new test split data:", X_Test.shape)
```

```
Initial shape for entire data: (2070, 30)
Shape of new training data: (1656, 54)
Shape of new test split data: (414, 54)
```

Split 80-20 into Train and Test data

Classification Model comments + customer data

Precision important metric here- Company wants higher % of correct prediction for people churning so that they can invest on right people to save them rather than investing on people who won't churn!

```
[ ] #Construct a Random Forest Classifier on combined data
    rfc=RandomForestClassifier()
    rfc.fit(X_Train,Y_Train)

    #Prediction on training data
    pred_rf=pd.DataFrame(rfc.predict(X_Train),columns=["Prediction"])
    print("----------------------------Random Forest: Training Data----------------------------------------------\n")
    print("Accuracy Score on training data using Random Forest:",accuracy_score(Y_Train,pred_rf["Prediction"]))
    print("Confusion Matrix on training data using Random Forest\n", confusion_matrix(Y_Train,pred_rf["Prediction"]))

    # prediction on test data
    pred_rf=pd.DataFrame(rfc.predict(X_Test),columns=["Prediction"])
    print("----------------------------Random Forest: Test Data----------------------------------------------\n")
    print("Accuracy Score on test data using Random Forest:",accuracy_score(Y_Test,pred_rf["Prediction"]))
    print("Balanced Accuracy Score on test data using Random Forest:",balanced_accuracy_score(Y_Test,pred_rf["Prediction"]))
    print("Confusion Matrix on test data using Random Forest\n", confusion_matrix(Y_Test,pred_rf["Prediction"]))
    print("Classification report on test data using Random Forest\n", classification_report(Y_Test,pred_rf["Prediction"]))
```

```
----------------------------Random Forest: Training Data----------------------------------------------

Accuracy Score on training data using Random Forest: 0.9963768115942029
Confusion Matrix on training data using Random Forest
 [[ 645    2]
 [   4 1005]]
----------------------------Random Forest: Test Data----------------------------------------------

Accuracy Score on test data using Random Forest: 0.8671497584541062
Balanced Accuracy Score on test data using Random Forest: 0.8502988425983296
Confusion Matrix on test data using Random Forest
 [[129  28]
 [ 27 230]]
Classification report on test data using Random Forest
              precision    recall  f1-score   support

   Cancelled       0.83      0.82      0.82       157
     Current       0.89      0.89      0.89       257

    accuracy                           0.87       414
   macro avg       0.86      0.86      0.86       414
weighted avg       0.87      0.87      0.87       414
```

Feature Selection – Wrapper Method

Feature Selection – Wrapper method

Precision important metric here- Company wants higher % of correct prediction for people churning so that they can invest on right people to save them rather than investing on people who won't churn!

```
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
#Construct a Random Forest Classifier on text data
rfc=RandomForestClassifier()

sfs1 = SFS(rfc,
           k_features=20,
           forward=True,
           floating=False,
           verbose=2,
           scoring='accuracy',
           cv=0)

sfs1 = sfs1.fit(DF_TF_IDF,y_train, custom_feature_names= DF_TF_IDF.columns)
print("Top 20 feature", sfs1.k_feature_names_)
```

Using Step Forward Selection method - selecting top 20 features

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
/usr/local/lib/python3.6/dist-packages/sklearn/ensemble/forest.py:245: FutureWarning: The default value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 354 out of 354 | elapsed:    7.7s finished

[2019-12-11 01:19:47] Features: 1/20 -- score: 0.6236714975845411[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 353 out of 353 | elapsed:    8.9s finished

[2019-12-11 01:19:56] Features: 2/20 -- score: 0.6275362318840579[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 352 out of 352 | elapsed:    9.0s finished

[2019-12-11 01:20:05] Features: 3/20 -- score: 0.6318840579710145[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 351 out of 351 | elapsed:    9.5s finished

[2019-12-11 01:20:15] Features: 4/20 -- score: 0.633816425120773[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 350 out of 350 | elapsed:    9.5s finished

[2019-12-11 01:20:24] Features: 5/20 -- score: 0.6347826086956522[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done 349 out of 349 | elapsed:    9.6s finished
```

Feature Selection – Wrapper method -2

1

```
#top 20 features after Forward
print("Top 20 feature", sfs1.k_feature_names_)

DF_TF_IDF_SelectedFeatures = DF_TF_IDF.loc[:, DF_TF_IDF.columns.isin(list(sfs1.k_feature_names_))]
print("Shape of TF-IDF after feature selection", DF_TF_IDF_SelectedFeatures.shape)
print(DF_TF_IDF_SelectedFeatures)
```

Get Dataframe with selected new features

```
Top 20 feature ('3399', '3g', 'abysm', 'access', 'accessori', 'adapt', 'add', 'addit', 'additon', 'address', 'advertis', 'afraid', 'alway', 'angel', 'ani', 'complain', 'expec
Shape of TF-IDF after feature selection (2070, 20)
      3399   3g  abysm  access  ...  expect  hochi     phone      want
0      0.0  0.0    0.0     0.0  ...     0.0    0.0  0.311374  0.000000
1      0.0  0.0    0.0     0.0  ...     0.0    0.0  0.000000  0.312065
2      0.0  0.0    0.0     0.0  ...     0.0    0.0  0.000000  0.195324
3      0.0  0.0    0.0     0.0  ...     0.0    0.0  0.000000  0.000000
4      0.0  0.0    0.0     0.0  ...     0.0    0.0  0.243227  0.000000
...    ...  ...    ...     ...  ...     ...    ...       ...       ...
2065   0.0  0.0    0.0     0.0  ...     0.0    0.0  0.000000  0.000000
2066   0.0  0.0    0.0     0.0  ...     0.0    0.0  0.180489  0.000000
2067   0.0  0.0    0.0     0.0  ...     0.0    0.0  0.178295  0.144882
2068   0.0  0.0    0.0     0.0  ...     0.0    0.0  0.000000  0.324251
2069   0.0  0.0    0.0     0.0  ...     0.0    0.0  0.180489  0.000000

[2070 rows x 20 columns]
```

2

```
#split data into training and test
#split the entire training data in training and test data
X_Train, X_Test, Y_Train, Y_Test = train_test_split( DF_TF_IDF_SelectedFeatures, y_train, test_size=0.20, random_state=42)
print("Initial shape for entire data:",DF_TF_IDF_SelectedFeatures.shape)
print("Shape of new training data:", X_Train.shape)
print("Shape of new test split data:", X_Test.shape)
```

Split 80-20 into Train and Test data

```
Initial shape for entire data: (2070, 20)
Shape of new training data: (1656, 20)
Shape of new test split data: (414, 20)
```

Classification Model comments

Precision important metric here- Company wants higher % of correct prediction for people churning so that they can invest on right people to save them rather than investing on people who won't churn!

```
[ ] #Construct a Random Forest Classifier on text data
    rfc=RandomForestClassifier()
    rfc.fit(X_Train,Y_Train)

    #Prediction on training data
    pred_rf=pd.DataFrame(rfc.predict(X_Train),columns=["Prediction"])
    print("----------------------------Random Forest: Training Data----------------------------------------------\n")
    print("Accuracy Score on training data using Random Forest:",accuracy_score(Y_Train,pred_rf["Prediction"]))
    print("Confusion Matrix on training data using Random Forest\n", confusion_matrix(Y_Train,pred_rf["Prediction"]))

    # prediction on test data
    pred_rf=pd.DataFrame(rfc.predict(X_Test),columns=["Prediction"])
    print("----------------------------Random Forest: Test Data----------------------------------------------\n")
    print("Accuracy Score on test data using Random Forest:",accuracy_score(Y_Test,pred_rf["Prediction"]))
    print("Balanced Accuracy Score on test data using Random Forest:",balanced_accuracy_score(Y_Test,pred_rf["Prediction"]))
    print("Confusion Matrix on test data using Random Forest\n", confusion_matrix(Y_Test,pred_rf["Prediction"]))
    print("Classification report on test data using Random Forest\n", classification_report(Y_Test,pred_rf["Prediction"]))
```

```
----------------------------Random Forest: Training Data----------------------------------------------

Accuracy Score on training data using Random Forest: 0.6364734299516909
Confusion Matrix on training data using Random Forest
 [[ 93 554]
 [ 48 961]]
----------------------------Random Forest: Test Data----------------------------------------------

Accuracy Score on test data using Random Forest: 0.6135265700483091
Balanced Accuracy Score on test data using Random Forest: 0.5152296215519592
Confusion Matrix on test data using Random Forest
 [[ 17 140]
 [ 20 237]]
Classification report on test data using Random Forest
              precision    recall  f1-score   support

   Cancelled       0.46      0.11      0.18       157
     Current       0.63      0.92      0.75       257

    accuracy                           0.61       414
   macro avg       0.54      0.52      0.46       414
weighted avg       0.56      0.61      0.53       414
```

Classification Model comments + customer data

Precision important metric here- Company wants higher % of correct prediction for people churning so that they can invest on right people to save them rather than investing on people who won't churn!

```
[ ] #Construct a Random Forest Classifier on combined data
    rfc=RandomForestClassifier()
    rfc.fit(X_Train,Y_Train)

    #Prediction on training data
    pred_rf=pd.DataFrame(rfc.predict(X_Train),columns=["Prediction"])
    print("----------------------------Random Forest: Training Data----------------------------------------------\n")
    print("Accuracy Score on training data using Random Forest:",accuracy_score(Y_Train,pred_rf["Prediction"]))
    print("Confusion Matrix on training data using Random Forest\n", confusion_matrix(Y_Train,pred_rf["Prediction"]))

    # prediction on test data
    pred_rf=pd.DataFrame(rfc.predict(X_Test),columns=["Prediction"])
    print("----------------------------Random Forest: Test Data----------------------------------------------\n")
    print("Accuracy Score on test data using Random Forest:",accuracy_score(Y_Test,pred_rf["Prediction"]))
    print("Balanced Accuracy Score on test data using Random Forest:",balanced_accuracy_score(Y_Test,pred_rf["Prediction"]))
    print("Confusion Matrix on test data using Random Forest\n", confusion_matrix(Y_Test,pred_rf["Prediction"]))
    print("Classification report on test data using Random Forest\n", classification_report(Y_Test,pred_rf["Prediction"]))
```

```
----------------------------Random Forest: Training Data----------------------------------------------

Accuracy Score on training data using Random Forest: 0.9897342995169082
Confusion Matrix on training data using Random Forest
 [[641   6]
 [ 11 998]]
----------------------------Random Forest: Test Data----------------------------------------------

Accuracy Score on test data using Random Forest: 0.8695652173913043
Balanced Accuracy Score on test data using Random Forest: 0.8627227440580931
Confusion Matrix on test data using Random Forest
 [[131  26]
 [ 28 229]]
Classification report on test data using Random Forest
              precision    recall  f1-score   support

   Cancelled       0.82      0.83      0.83       157
     Current       0.90      0.89      0.89       257

    accuracy                           0.87       414
   macro avg       0.86      0.86      0.86       414
weighted avg       0.87      0.87      0.87       414
```

Porter Stemming

```
[ ]  #Porter stemming
     # Use English stemmer.
     porter = PorterStemmer()
     newTextData=pd.DataFrame()
     newTextData=textData.drop(columns=["CommentsTokenized","Comments"])
     newTextData['CommentsTokenizedStemmedPorter'] = textData['CommentsTokenized'].apply(lambda x: [porter.stem(y) for y in x])
     print('-------------Porter Stemming------------------')
     newTextData['CommentsTokenizedStemmedPorter']
     #Join stemmed strings
     newTextData['CommentsTokenizedStemmedPorter'] = newTextData['CommentsTokenizedStemmedPorter'].apply(lambda x: " ".join(x))
     newTextData['CommentsTokenizedStemmedPorter']
```

```
-------------Porter Stemming------------------
0       doe not like the way the phone work . It is to difficult compar to hi last phone .
1       want to know the nearest store locat . want to buy adit accessori .
2       want to know how to do text messag . refer him to websit .
3       ask how to disabl call wait . refer him to web site .
4       need help learn how to use the phone . I suggest he go back to the store and have the rep teach him .
                                                  ...
2065    need help figur out hi bill . I explain our minut charg .
2066    He lost hi phone and call to cancel servic . I told him we would suspend until we hear back from him . He will contact us soon .
2067    lost the direct to phone and want anoth manual . I refer him to web site .
2068    want to chang address .
2069    He lost hi phone and call to cancel servic . I told him we would suspend until we hear back from him . He will contact us soon .
Name: CommentsTokenizedStemmedPorter, Length: 2070, dtype: object
```

Converting all words into their root words

Term Document Vector - Porter

```
[ ]  #Do Bag-Of-Words model - Term - Document Matrix - Snowball
     #Learn the vocabulary dictionary and return term-document matrix.
     #count_vect = CountVectorizer(stop_words=None)
     count_vect = CountVectorizer(stop_words='english',lowercase=False)
     count_vect
     TD_counts = count_vect.fit_transform(newTextData.CommentsTokenizedStemmedPorter)
     print("Vocabulary ",(count_vect.vocabulary_))
     print("Number of words in vocabulary",len(count_vect.vocabulary_))
     print("Shape of text vector",TD_counts.shape)
     DF_TD_Counts=pd.DataFrame(TD_counts.toarray(), columns= count_vect.get_feature_names())
     print(DF_TD_Counts)
```

```
Vocabulary  {'doe': 106, 'like': 181, 'way': 349, 'phone': 231, 'work': 359, 'It': 8, 'difficult': 102, 'compar': 68, '
Number of words in vocabulary 366
Shape of text vector (2070, 366)
      3399  3g  As  CC  He  If  In  ...  work  wors  worst  wrong  xvyx  year  york
0        0   0   0   0   0   0   0  ...     1     0      0      0     0     0     0
1        0   0   0   0   0   0   0  ...     0     0      0      0     0     0     0
2        0   0   0   0   0   0   0  ...     0     0      0      0     0     0     0
3        0   0   0   0   0   0   0  ...     0     0      0      0     0     0     0
4        0   0   0   0   0   0   0  ...     0     0      0      0     0     0     0
...     ..  ..  ..  ..  ..  ..  ..  ...    ..    ..     ..     ..    ..    ..    ..
2065     0   0   0   0   0   0   0  ...     0     0      0      0     0     0     0
2066     0   0   0   0   2   0   0  ...     0     0      0      0     0     0     0
2067     0   0   0   0   0   0   0  ...     0     0      0      0     0     0     0
2068     0   0   0   0   0   0   0  ...     0     0      0      0     0     0     0
2069     0   0   0   0   2   0   0  ...     0     0      0      0     0     0     0

[2070 rows x 366 columns]
```

Porter Created 366 words

Creating a matrix for all the broken words and converting into 0,1 form for computation

Lancaster Stemming

```
[7]  pd.options.display.max_rows
     pd.set_option('display.max_colwidth', -2)
     #Lancaster stemming
     # Use English stemmer.
     lancaster=LancasterStemmer()
     newTextData=pd.DataFrame()
     newTextData=textData.drop(columns=["CommentsTokenized","Comments"])
     newTextData['CommentsTokenizedStemmedLancaster'] = textData['CommentsTokenized'].apply(lambda x: [lancaster.stem(y) for y in x])
     print('-------------Lancaster Stemming------------------')
     newTextData['CommentsTokenizedStemmedLancaster']
     #Join stemmed strings
     newTextData['CommentsTokenizedStemmedLancaster'] = newTextData['CommentsTokenizedStemmedLancaster'].apply(lambda x: " ".join(x))
     newTextData['CommentsTokenizedStemmedLancaster']
```

```
-------------Lancaster Stemming------------------
0       doe not lik the way the phon work . it is to difficult comp to his last phon .
1       want to know the nearest stor loc . want to buy adit access .
2       want to know how to do text mess . refer him to websit .
3       ask how to dis cal wait . refer him to web sit .
4       nee help learn how to us the phon . i suggest he go back to the stor and hav the rep teach him .
                                                ...
2065    nee help fig out his bil . i explain our minut charg .
2066    he lost his phon and cal to cancel serv . i told him we would suspend until we hear back from him . he wil contact us soon .
2067    lost the direct to phon and want anoth man . i refer him to web sit .
2068    want to chang address .
2069    he lost his phon and cal to cancel serv . i told him we would suspend until we hear back from him . he wil contact us soon .
Name: CommentsTokenizedStemmedLancaster, Length: 2070, dtype: object
```

Converting all words into their root words

Term Document Vector - Lancaster

```
[8] #Do Bag-Of-Words model - Term - Document Matrix - Lancaster
    #Learn the vocabulary dictionary and return term-document matrix.
    #count_vect = CountVectorizer(stop_words=None)
    count_vect = CountVectorizer(stop_words='english',lowercase=False)
    count_vect
    TD_counts = count_vect.fit_transform(newTextData.CommentsTokenizedStemmedLancaster)
    print("Vocabulary ",(count_vect.vocabulary_))
    print("Number of words in vocabulary",len(count_vect.vocabulary_))
    print("Shape of text vector",TD_counts.shape)
    DF_TD_Counts=pd.DataFrame(TD_counts.toarray(), columns= count_vect.get_feature_names())
    print(DF_TD_Counts)
```

```
Vocabulary  {'doe': 96, 'lik': 173, 'way': 341, 'phon': 224, 'work': 356, 'difficult': 92, 'comp': 58, 'want': 339, 'know': 167, 'ne
Number of words in vocabulary 364
Shape of text vector (2070, 364)
      3399  3g  abysm  access  ad  adapt  ...  worst  wrong  xvyx  year  yo  york
0        0   0      0       0   0      0  ...      0      0     0     0   0     0
1        0   0      0       1   0      0  ...      0      0     0     0   0     0
2        0   0      0       0   0      0  ...      0      0     0     0   0     0
3        0   0      0       0   0      0  ...      0      0     0     0   0     0
4        0   0      0       0   0      0  ...      0      0     0     0   0     0
...    ...  ..    ...     ...  ..    ...  ...    ...    ...   ...   ...  ..   ...
2065     0   0      0       0   0      0  ...      0      0     0     0   0     0
2066     0   0      0       0   0      0  ...      0      0     0     0   0     0
2067     0   0      0       0   0      0  ...      0      0     0     0   0     0
2068     0   0      0       0   0      0  ...      0      0     0     0   0     0
2069     0   0      0       0   0      0  ...      0      0     0     0   0     0

[2070 rows x 364 columns]
```

Lancaster Created 364 words

Creating a matrix for all the broken words and converting into 0,1 form for computation

Lemmatize

```
[ ] lem = WordNetLemmatizer()
    newTextData=pd.DataFrame()
    newTextData=textData.drop(columns=["CommentsTokenized","Comments"])
    newTextData['CommentsTokenizedLematize'] = textData['CommentsTokenized'].apply(lambda x: [lem.lemmatize(y) for y in x])
    newTextData['CommentsTokenizedLematize']
    #Join stemmed strings
    newTextData['CommentsTokenizedLematize'] = newTextData['CommentsTokenizedLematize'].apply(lambda x: " ".join(x))
    newTextData['CommentsTokenizedLematize']
```

```
0       Does not like the way the phone work . It is to difficult compared to his last phone .
1       Wanted to know the nearest store location . Wants to buy aditional accessory .
2       Wants to know how to do text messaging . Referred him to website .
3       Asked how to disable call waiting . referred him to web site .
4       Needs help learning how to use the phone . I suggested he go back to the store and have the rep te
                                                    ...
2065    Needed help figuring out his bill . I explained our minute charge .
2066    He lost his phone and called to cancel service . I told him we would suspend until we hear back from him . He will contact u soon .
2067    Lost the direction to phone and want another manual . I referred him to web site .
2068    Wants to change address .
2069    He lost his phone and called to cancel service . I told him we would suspend until we hear back from him . He will contact u soon .
Name: CommentsTokenizedLematize, Length: 2070, dtype: object
```

Converting all words into their root words
These words make more sense!

Term Document Vector - Lemmatize

```
#Do Bag-Of-Words model - Term - Document Matrix - Lematize
#Learn the vocabulary dictionary and return term-document matrix.
#count_vect = CountVectorizer(stop_words=None)
count_vect = CountVectorizer(stop_words='english',lowercase=False)
count_vect
TD_counts = count_vect.fit_transform(newTextData.CommentsTokenizedLematize)
print("Vocabulary ",(count_vect.vocabulary_))
print("Number of words in vocabulary",len(count_vect.vocabulary_))
print("Shape of text vector",TD_counts.shape)
DF_TD_Counts=pd.DataFrame(TD_counts.toarray(), columns= count_vect.get_feature_names())
print(DF_TD_Counts)
```

```
Vocabulary  {'Does': 17, 'like': 247, 'way': 420, 'phone': 295, 'work': 430, 'It': 34, 'difficult': 165, 'compared': 131, 'Wanted
Number of words in vocabulary 437
Shape of text vector (2070, 437)
      3399  3g  ASAP  Also  Angeles  ...  working  worse  worst  wrong  year
0        0   0     0     0        0  ...        0      0      0      0     0
1        0   0     0     0        0  ...        0      0      0      0     0
2        0   0     0     0        0  ...        0      0      0      0     0
3        0   0     0     0        0  ...        0      0      0      0     0
4        0   0     0     0        0  ...        0      0      0      0     0
...    ...  ..    ..    ..      ...  ...      ...    ...    ...    ...   ...
2065     0   0     0     0        0  ...        0      0      0      0     0
2066     0   0     0     0        0  ...        0      0      0      0     0
2067     0   0     0     0        0  ...        0      0      0      0     0
2068     0   0     0     0        0  ...        0      0      0      0     0
2069     0   0     0     0        0  ...        0      0      0      0     0

[2070 rows x 437 columns]
```

Lemmatize created 437 words

Creating a matrix for all the broken words and converting into 0,1 form for computation

of words comparison

Stemming/Lemmatize	# of stemmed words
Snowball Stemming	354
Porter Stemming	366
Lancaster Stemming	364
Lemmatize	437

Score using only customer data

Accuracy on Test	Precision of Cancelled on Test
85.99%	80%

Stemming/Lemmatize	Feature selection type	Comments/Customer + Comments	Accuracy on Test	Precision of Cancelled on Test
Snowball Stemming	Filter	Comments	62.31%	52%
Snowball Stemming	Filter	Customer + Comments	86.71%	83%
Snowball Stemming	Wrapper	Comments	61.35%	46%
Snowball Stemming	Wrapper	Customer + Comments	86.95%	82%
Porter Stemming	Filter	Comments	62.07%	50%
Porter Stemming	Filter	Customer + Comments	86.71%	82%
Porter Stemming	Wrapper	Comments	60.6%	38%
Porter Stemming	Wrapper	Customer + Comments	85.5%	81%
Lancaster Stemming	Filter	Comments	62.8%	57%
Lancaster Stemming	Filter	Customer + Comments	85.99%	82%
Lancaster Stemming	Wrapper	Comments	62.31%	52%
Lancaster Stemming	Wrapper	Customer + Comments	85.26%	79%
Lemmatize	Filter	Comments	62.56%	56%
Lemmatize	Filter	Customer + Comments	87.19%	82%
Lemmatize	Wrapper	Comments	61.59%	47%
Lemmatize	Wrapper	Customer + Comments	85.26%	80%

Learnings from assignment

01

Text mining using NLTK library

02

Word tokenization, stemming, lemmatization, term document vector, TF-IDF matrix

03

Feature selection methods – Select K best and step forward selection method