What's in this assignment:

(1) Solving a semi-structured machine learning problem

(2) Using Python's Natural Language Tool Kit (NLTK)

Load Packages

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
#Step 1
# NLTK--------------------------------
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem.snowball import SnowballStemmer
```

```
[nltk_data] Downloading package punkt to C:\Users\Jinhang
[nltk_data]     Jiang\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [3]:
```
#Step 2 & 3
# Transformation
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
```

In [4]:
```
#Step 6
from sklearn.metrics import confusion_matrix,classification_report,roc_auc_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
```

Change Path

In [5]:
```
print(os.getcwd())
os.chdir('D:/OneDrive/ASU/2020 Fall/CIS 508/Assignment4')
print(os.getcwd())
```

```
C:\Users\Jinhang Jiang
D:\OneDrive\ASU\2020 Fall\CIS 508\Assignment4
```

Data Overview

In [6]:
```
customer = pd.read_csv("Customers.csv")
comment = pd.read_csv("Comments.csv")
```

In [7]:
```
customer
```

Out[7]:

	ID	Sex	Status	Children	Est_Income	Car_Owner	Usage	Age	RatePlan	LongDist
0	1	F	S	1	38000.00	N	229.64	24.393333	3	2
1	6	M	M	2	29616.00	N	75.29	49.426667	2	2
2	8	M	M	0	19732.80	N	47.25	50.673333	3	2
3	11	M	S	2	96.33	N	59.01	56.473333	1	2
4	14	F	M	2	52004.80	N	28.14	25.140000	1	
...	...	...	...	...	...	...	...	...	...	
2065	3821	F	S	0	78851.30	N	29.04	48.373333	4	
2066	3822	F	S	1	17540.70	Y	36.20	62.786667	1	2
2067	3823	F	M	0	83891.90	Y	74.40	61.020000	4	2
2068	3824	F	M	2	28220.80	N	38.95	38.766667	4	2
2069	3825	F	S	0	28589.10	N	100.28	15.600000	3	1

2070 rows × 17 columns

In [8]:
```
comment
```

Out[8]:

	ID	Comments
0	1309	Does not like the way the phone works. It is t...
1	3556	Wanted to know the nearest store location. Wan...
2	2230	Wants to know how to do text messaging. Referr...
3	2312	Asked how to disable call waiting. referred hi...
4	3327	Needs help learning how to use the phone. I su...
...	...	...
2065	3034	Needed help figuring out his bill. I explained...
2066	271	He lost his phone and called to cancel service...
2067	783	Lost the directions to phone and wants another...
2068	1295	Wants to change address.
2069	1807	He lost his phone and called to cancel service...

2070 rows × 2 columns

In [9]:
```
customer.TARGET.value_counts()
```

Out[9]:
```
Current      1266
Cancelled     804
Name: TARGET, dtype: int64
```

In [10]:
```
# Encoding the Target
le = LabelEncoder()

data = customer.drop(["TARGET"],axis=1)
label = le.fit(customer["TARGET"]).transform(customer["TARGET"])
```

Tokenize and stem the comments

In [11]:
```
# First, tokenize the comments
comment["TokenizedComments"]=comment["Comments"].apply(word_tokenize)
comment.head()
```

Out[11]:

	ID	Comments	TokenizedComments
0	1309	Does not like the way the phone works. It is t...	[Does, not, like, the, way, the, phone, works,...
1	3556	Wanted to know the nearest store location. Wan...	[Wanted, to, know, the, nearest, store, locati...
2	2230	Wants to know how to do text messaging. Referr...	[Wants, to, know, how, to, do, text, messaging...
3	2312	Asked how to disable call waiting. referred hi...	[Asked, how, to, disable, call, waiting, ., re...
4	3327	Needs help learning how to use the phone. I su...	[Needs, help, learning, how, to, use, the, pho...

In [12]:
```
#build stmmers
porter = PorterStemmer()
snow = SnowballStemmer("english")
lancaster = LancasterStemmer()
```

In [13]:
```
## Porter, Snowball, Lancaster
#Now do stemming - create a new dataframe to store stemmed version
newTextData=pd.DataFrame()
newTextData=comment.drop(["TokenizedComments","Comments"],axis=1)


## Apply different stemmers and join the strings together
# Porter
newTextData['Porter'] = comment['TokenizedComments'].apply(lambda x: [porter.stem(y) for
newTextData['Porter'] = newTextData['Porter'].apply(lambda x: " ".join(x))

# Snowball
newTextData['Snow'] = comment['TokenizedComments'].apply(lambda x: [snow.stem(y) for y in
newTextData['Snow'] = newTextData['Snow'].apply(lambda x: " ".join(x))


# Lancaster
newTextData['Lancaster'] = comment['TokenizedComments'].apply(lambda x: [lancaster.stem(y
newTextData['Lancaster'] = newTextData['Lancaster'].apply(lambda x: " ".join(x))


#newTextData.to_csv('Stemmers.csv',index=False)
newTextData.head()
```

Out[13]:

	ID	Porter	Snow	Lancaster
0	1309	doe not like the way the phone work . It is to...	doe not like the way the phone work . it is to...	doe not lik the way the phon work . it is to d...
1	3556	want to know the nearest store locat . want to...	want to know the nearest store locat . want to...	want to know the nearest stor loc . want to bu...
2	2230	want to know how to do text messag . refer him...	want to know how to do text messag . refer him...	want to know how to do text mess . refer him t...
3	2312	ask how to disabl call wait . refer him to web...	ask how to disabl call wait . refer him to web...	ask how to dis cal wait . refer him to web sit .
4	3327	need help learn how to use the phone . I sugge...	need help learn how to use the phone . i sugge...	nee help learn how to us the phon . i suggest ...

1. It looks like Porter tends to save the uppercase or lowercase of the original content.
2. Lancaster is very agressive. For example, for ID 3034, while the other two kept the word "need", lancaster converted it to "nee".
3. Porter also had difficult time recognizing simple words, like "his". In most cases, porter somehow converted "his" to "hi"
4. Therefore, I picked snowball to go forward.

After stemming, construct the term-document matrix and eliminate stop words

In [14]:
```
count_vect = CountVectorizer(stop_words='english',lowercase=False)
TD_counts = count_vect.fit_transform(newTextData["Snow"])
DF_TD_Counts=pd.DataFrame(TD_counts.toarray())
DF_TD_Counts.columns = count_vect.get_feature_names()
#DF_TD_Counts.to_csv('TD_counts.csv',index=False)
```

In [15]:
```
print(DF_TD_Counts.shape)
DF_TD_Counts
```

(2070, 354)

Out[15]:

	3399	3g	abysm	access	accessori	adapt	add	addit	additon	address	...	wish	wll	w
0	0	0	0	0	0	0	0	0	0	0	...	0	0	
1	0	0	0	0	1	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	1	...	0	0	
2069	0	0	0	0	0	0	0	0	0	0	...	0	0	

2070 rows × 354 columns

Construct the TF-IDF matrix from the term-document matrix

In [16]:
```
#Compute TF-IDF Matrix
tfidf_transformer = TfidfTransformer()
tfidf = tfidf_transformer.fit_transform(TD_counts)
DF_TF_IDF=pd.DataFrame(tfidf.toarray())
DF_TF_IDF.columns=count_vect.get_feature_names()
DF_TF_IDF["ID"]=comment["ID"]

#DF_TF_IDF.to_csv('TFIDF_counts.csv',index=False)
```

In [17]:
```
DF_TF_IDF.head()
```

Out[17]:

	3399	3g	abysm	access	accessori	adapt	add	addit	additon	address	...	wll	wold	w
0	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.209
1	0.0	0.0	0.0	0.0	0.27568	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000
2	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000
3	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000
4	0.0	0.0	0.0	0.0	0.00000	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.000

5 rows × 355 columns

Combine the TF-IDF matrix with Customer data. Then do one-hot encoding on the categorical variables

In [18]:
```
EncodeData = pd.merge(data,DF_TF_IDF, how = "left", on="ID")
```

In [19]:
```
X_cat = EncodeData.select_dtypes(exclude=['int','float64'])
X_cat=X_cat.drop(["ID"],axis=1)
X_cat
```

Out[19]:

	Sex	Status	Children	Car_Owner	RatePlan	Dropped	Paymethod	LocalBilltype	LongDistan
0	F	S	1	N	3	0	CC	Budget	Intr
1	M	M	2	N	2	0	CH	FreeLocal	
2	M	M	0	N	3	0	CC	FreeLocal	
3	M	S	2	N	1	1	CC	Budget	
4	F	M	2	N	1	0	CH	Budget	Intr
...	...	...	...	...	...	...	...	...	
2065	F	S	0	N	4	0	CC	FreeLocal	
2066	F	S	1	Y	1	0	Auto	Budget	
2067	F	M	0	Y	4	0	CH	Budget	
2068	F	M	2	N	4	0	CC	FreeLocal	
2069	F	S	0	N	3	0	CC	FreeLocal	

2070 rows × 9 columns

In [20]:
```
# One Hot Encoding
EncodeData = pd.get_dummies(EncodeData,columns=X_cat.columns)
```

In [21]:
```
#EncodeData.to_csv("Combined.csv",index=False)
EncodeData.head()
```

Out[21]:

	ID	Est_Income	Usage	Age	LongDistance	International	Local	3399	3g	abysm	...	D
0	1	38000.00	229.64	24.393333	23.56	0.0	206.08	0.0	0.0	0.0	...	
1	6	29616.00	75.29	49.426667	29.78	0.0	45.50	0.0	0.0	0.0	...	
2	8	19732.80	47.25	50.673333	24.81	0.0	22.44	0.0	0.0	0.0	...	
3	11	96.33	59.01	56.473333	26.13	0.0	32.88	0.0	0.0	0.0	...	
4	14	52004.80	28.14	25.140000	5.03	0.0	23.11	0.0	0.0	0.0	...	

5 rows × 387 columns

Split data

In [22]:
```
# split data at 80% 20% for orginal data and combined data
data_train, data_test, label_train, label_test = train_test_split (pd.get_dummies(data.dro
                                                    test_size = 0.2,
                                                    random_state = 42)

X_train, X_test, y_train, y_test = train_test_split(EncodeData.drop(["ID"],axis=1), label,
                                                    test_size = 0.2,
                                                    random_state = 42)
```

Base Score

In [23]:
```
cat=CatBoostClassifier()
cat.fit(data_train,label_train)
```

```
Learning rate set to 0.012778
0:      learn: 0.6832829        total: 63.3ms   remaining: 1m 3s
1:      learn: 0.6780644        total: 66.6ms   remaining: 33.2s
2:      learn: 0.6685113        total: 71.2ms   remaining: 23.7s
3:      learn: 0.6612322        total: 73.7ms   remaining: 18.3s
4:      learn: 0.6538588        total: 76ms     remaining: 15.1s
5:      learn: 0.6453109        total: 78.1ms   remaining: 12.9s
6:      learn: 0.6376290        total: 80.2ms   remaining: 11.4s
7:      learn: 0.6313905        total: 83ms     remaining: 10.3s
8:      learn: 0.6241016        total: 85.4ms   remaining: 9.4s
9:      learn: 0.6177194        total: 88.2ms   remaining: 8.73s
10:     learn: 0.6117498        total: 90.5ms   remaining: 8.14s
11:     learn: 0.6072482        total: 93ms     remaining: 7.66s
12:     learn: 0.6005468        total: 98.5ms   remaining: 7.48s
13:     learn: 0.5963567        total: 101ms    remaining: 7.1s
14:     learn: 0.5915387        total: 103ms    remaining: 6.76s
15:     learn: 0.5860339        total: 105ms    remaining: 6.46s
16:     learn: 0.5793949        total: 107ms    remaining: 6.18s
17:     learn: 0.5729104        total: 109ms    remaining: 5.96s
18:     learn: 0.5670215        total: 111ms    remaining: 5.75s
```

In [24]:
```
cat_predictions = cat.predict_proba(data_test)
print("ROC score (training): {0:.6f}".format(roc_auc_score(label_test,cat_predictions[:,1]
print("Confusion Matrix:")
print(confusion_matrix(label_test, cat_predictions[:,1].round()))
print("Classification Report")
print(classification_report(label_test, cat_predictions[:,1].round()))
```

```
ROC score (training): 0.903393
Confusion Matrix:
[[129  28]
 [ 21 236]]
Classification Report
              precision    recall  f1-score   support

           0       0.86      0.82      0.84       157
           1       0.89      0.92      0.91       257

    accuracy                           0.88       414
   macro avg       0.88      0.87      0.87       414
weighted avg       0.88      0.88      0.88       414
```

In [25]:
```
xgb = XGBClassifier()
xgb.fit(data_train,label_train)
```

Out[25]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [26]:
```
xgb_predictions = xgb.predict_proba(data_test)
print("ROC score (training): {0:.6f}".format(roc_auc_score(label_test,xgb_predictions[:,1]
print("Confusion Matrix:")
print(confusion_matrix(label_test, xgb_predictions[:,1].round()))
print("Classification Report")
print(classification_report(label_test, xgb_predictions[:,1].round()))
```

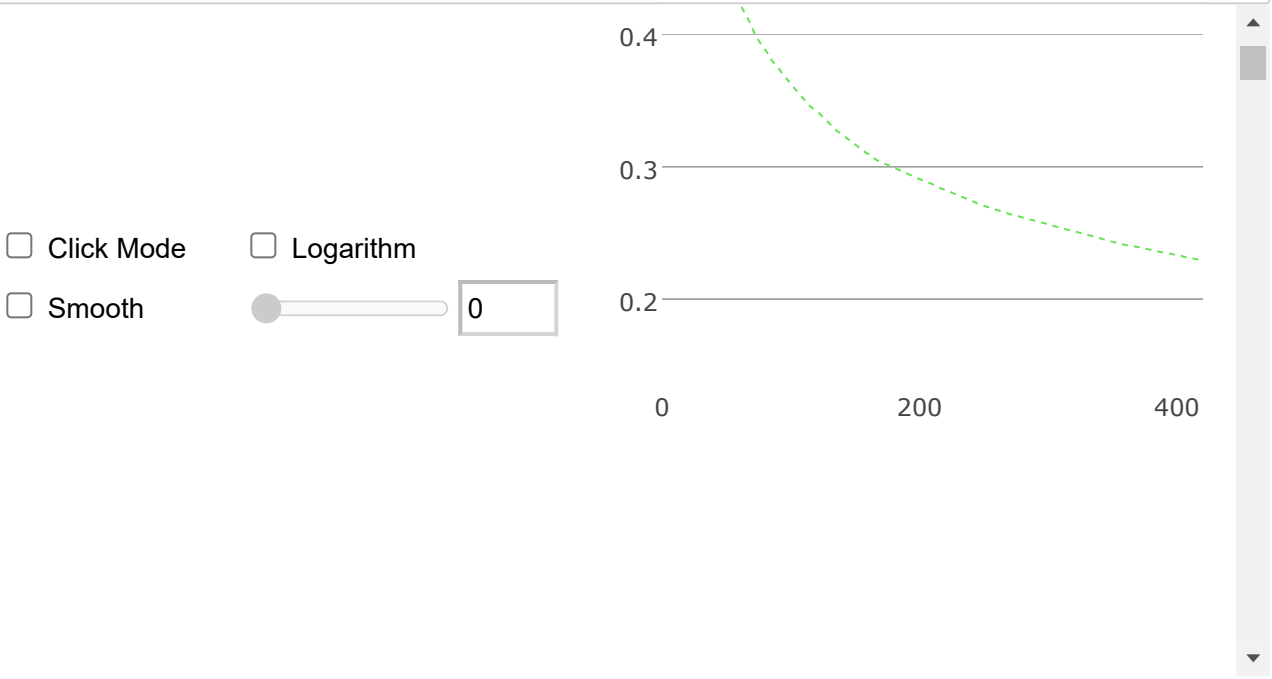
```
ROC score (training): 0.905500
Confusion Matrix:
[[126  31]
 [ 23 234]]
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.80      0.82       157
           1       0.88      0.91      0.90       257

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

Score for combined data

In [27]:
```
cat=CatBoostClassifier()
cat.fit(X_train,y_train,plot=True)
```



In [28]:
```
cat_predictions = cat.predict_proba(X_test)
print("ROC score (training): {0:.6f}".format(roc_auc_score(y_test,cat_predictions[:,1])))
print("Confusion Matrix:")
print(confusion_matrix(y_test, cat_predictions[:,1].round()))
print("Classification Report")
print(classification_report(y_test, cat_predictions[:,1].round()))
```

```
ROC score (training): 0.917173
Confusion Matrix:
[[137  20]
 [ 20 237]]
Classification Report
              precision    recall  f1-score   support

           0       0.87      0.87      0.87       157
           1       0.92      0.92      0.92       257

    accuracy                           0.90       414
   macro avg       0.90      0.90      0.90       414
weighted avg       0.90      0.90      0.90       414
```

In [29]:
```
xgb = XGBClassifier()
xgb.fit(X_train,y_train)
```

Out[29]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints='',
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints='()',
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

In [30]:
```
xgb_predictions = xgb.predict_proba(X_test)
print("ROC score (training): {0:.6f}".format(roc_auc_score(y_test,xgb_predictions[:,1])))
print("Confusion Matrix:")
print(confusion_matrix(y_test, xgb_predictions[:,1].round()))
print("Classification Report")
print(classification_report(y_test, xgb_predictions[:,1].round()))
```

```
ROC score (training): 0.911844
Confusion Matrix:
[[130  27]
 [ 24 233]]
Classification Report
              precision    recall  f1-score   support

           0       0.84      0.83      0.84       157
           1       0.90      0.91      0.90       257

    accuracy                           0.88       414
   macro avg       0.87      0.87      0.87       414
weighted avg       0.88      0.88      0.88       414
```