Assignment 4 Text Mining

To Do List:

(1) Use at least two of those stemmers and compare the differences in some of the stemmed words. Use the word tokenizer to tokenize words before stemming.

(2) After stemming, construct the term-document matrix. Eliminate stop words when constructing the term document matrix.

(3) Then construct the TF-IDF matrix from the term-document matrix.

(4) Now combine the TF-IDF matrix with Customer data. Then do one-hot encoding on the categorical variables.

(5) There are two types of feature selection methods - the filter type and the wrapper type. Use both types to determine the best set of features.

(6) Split the combined dataset into a training (80%) and a test set (20%). Using the best set of features from each method (filter and wrapper), build new classification models and evaluate them on the test data.

Loading Packages

In [220]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import warnings
warnings.filterwarnings("ignore")
```

In [9]:

```
#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]   Unzipping tokenizers\punkt.zip.
```

In [39]:

```
#Step 2 & 3
# Transformation
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
```

In [221]:

```
#Step 4 & 5
# Feature Selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs

from sklearn.metrics import confusion_matrix,classification_report, roc_auc_score
from sklearn.preprocessing import LabelEncoder
```

In [121]:

```
#Step 6
from sklearn.model_selection import train_test_split
#from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from catboost import CatBoostClassifier
```

Change Path

In [2]:

```
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
```

In [4]:

```
customer = pd.read_csv(”Customers.csv”)
comment = pd.read_csv(”Comments.csv”)
```

In [5]:
```
customer
```

Out[5]:

	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 [6]:
```
comment
```

Out[6]:

	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 [7]:
```
print(customer.shape, comment.shape)
```

```
(2070, 17) (2070, 2)
```

In [8]:

```
customer.TARGET.value_counts()
```

Out[8]:

```
Current      1266
Cancelled     804
Name: TARGET, dtype: int64
```

In [157]:

```
le = LabelEncoder()

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

In [158]:

```
label
```

Out[158]:

```
array([0, 1, 1, ..., 0, 0, 0])
```

Step 1

(1) The NLTK library has a number of stemmers, such as the Porter, Lancaster and Snowball Stemmers. Use at least two of those stemmers and compare the differences in some of the stemmed words. Use the word tokenizer to tokenize words before stemming. Select one stemmer for the rest of the analysis.

In [16]:

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

Out[16]:

	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 [35]:

```
#build stmmers
porter = PorterStemmer()
snow = SnowballStemmer("english")
lancaster = LancasterStemmer()
```

In [36]:

```
## 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 i
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)
```

In [37]:

```
newTextData
```

Out[37]:

	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 ...
...	...	...	...	...
2065	3034	need help figur out hi bill . I explain our mi...	need help figur out his bill . i explain our m...	nee help fig out his bil . i explain our minut...
2066	271	He lost hi phone and call to cancel servic . I...	he lost his phone and call to cancel servic	he lost his phon and cal to cancel serv . i to...
2067	783	lost the direct to phone and want anoth manual...	lost the direct to phone and want anoth manual...	lost the direct to phon and want anoth man . i...
2068	1295	want to chang address .	want to chang address .	want to chang address .
2069	1807	He lost hi phone and call to cancel servic . I...	he lost his phone and call to cancel servic	he lost his phon and cal to cancel serv . i to...

2070 rows × 4 columns

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.

Step 2

(2) After stemming, construct the term-document matrix. Eliminate stop words when constructing the term document matrix.

In [61]:

```
#Do Bag-Of-Words model - Term - Document Matrix
#Learn the vocabulary dictionary and return term-document matrix.

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()
#print(DF_TD_Counts)
DF_TD_Counts.to_csv('TD_counts.csv',index=False)
```

```
In [62]: print(count_vect.get_feature_names())
```

```
['3399', '3g', 'abysm', 'access', 'accessori', 'adapt', 'add', 'addit', 'additon', 'add
ress', 'adit', 'adress', 'advertis', 'afraid', 'alway', 'angel', 'angri', 'ani', 'anot
h', 'anyth', 'anytim', 'area', 'asap', 'ask', 'bad', 'basic', 'bateri', 'batteri', 'bec
aus', 'believ', 'better', 'bigger', 'book', 'bought', 'brain', 'bring', 'built', 'bus
i', 'button', 'buy', 'cancel', 'cancer', 'car', 'care', 'carrier', 'caus', 'cc', 'cel
l', 'certain', 'chang', 'charg', 'charger', 'check', 'chip', 'citi', 'claim', 'clearit
i', 'cold', 'comapr', 'compani', 'compar', 'competit', 'complain', 'complaint', 'concep
t', 'connect', 'consisit', 'consist', 'constan', 'contact', 'continu', 'contract', 'cor
rect', 'cost', 'coupl', 'cover', 'coverag', 'creat', 'credit', 'cstmer', 'cstmr', 'curr
ent', 'cust', 'custom', 'customr', 'date', 'day', 'dead', 'decent', 'defect', 'deo', 'd
id', 'die', 'differ', 'difficult', 'digiti', 'direct', 'disabl', 'doe', 'don', 'dont',
'drop', 'dure', 'easier', 'effect', 'encount', 'end', 'enemi', 'equip', 'everytim', 'ev
erywher', 'evrey', 'exact', 'expect', 'expir', 'explain', 'facepl', 'fals', 'famili',
'featur', 'fed', 'figur', 'fine', 'fix', 'forev', 'forward', 'friend', 'function', 'fur
thermor', 'futur', 'gave', 'goat', 'good', 'great', 'gsm', 'handset', 'happi', 'hard',
'hate', 'hear', 'heard', 'help', 'higher', 'highway', 'hochi', 'hole', 'home', 'hope',
'horribl', 'hous', 'implement', 'improv', 'inadequ', 'includ', 'info', 'inform', 'ing',
'internet', 'intersect', 'issu', 'june', 'just', 'kid', 'kno', 'know', 'lame', 'later',
'lctn', 'learn', 'leroy', 'like', 'line', 'list', 'local', 'locat', 'locatn', 'long',
'los', 'lost', 'lot', 'love', 'major', 'make', 'manag', 'mani', 'manual', 'market', 'me
an', 'messag', 'metropolitian', 'minut', 'misl', 'mistak', 'model', 'momma', 'mr', 'nap
eleon', 'near', 'nearest', 'need', 'network', 'new', 'news', 'notic', 'number', 'nume
r', 'offer', 'old', 'om', 'open', 'option', 'ori', 'ot', 'outbound', 'pass', 'pay', 'pd
a', 'peopl', 'perform', 'person', 'phone', 'piec', 'plan', 'pleas', 'point', 'polici',
'poor', 'possibl', 'probabl', 'problem', 'proper', 'provid', 'provis', 'purpos', 'rat
e', 'rater', 'realiz', 'realli', 'reason', 'receiv', 'recept', 'recption', 'reenter',
'refer', 'relat', 'rep', 'replac', 'respect', 'result', 'rid', 'right', 'ring', 'roam',
'roll', 'rubbish', 'rude', 'said', 'sale', 'say', 'screen', 'self', 'send', 'servic',
'shitti', 'shut', 'sign', 'signal', 'signific', 'simm', 'simpli', 'sinc', 'site', 'slo
w', 'sold', 'someon', 'sometim', 'soon', 'speak', 'speed', 'start', 'static', 'stole',
'store', 'stuff', 'stupid', 'substant', 'subtract', 'suck', 'suggest', 'supervisor', 's
upport', 'sure', 'surpris', 'suspect', 'suspend', 'switch', 'teach', 'technic', 'tell',
'terribl', 'test', 'text', 'think', 'thought', 'ticket', 'till', 'time', 'tire', 'toda
y', 'toilet', 'told', 'tone', 'tower', 'transeff', 'transf', 'transfer', 'travel', 'tr
i', 'trust', 'turn', 'uncomfort', 'understand', 'unhappi', 'unlimit', 'unreli', 'unwi
l', 'upset', 'usag', 'use', 'useless', 'valu', 'veri', 'vm', 'wa', 'wait', 'want', 'was
t', 'way', 'weak', 'web', 'websit', 'week', 'whi', 'wife', 'wish', 'wll', 'wold', 'wor
k', 'wors', 'worst', 'wrong', 'xvyx', 'year', 'york']
```

In [63]:
```
print(DF_TD_Counts.shape)
print(DF_TD_Counts)
```

```
(2070, 354)
      3399  3g  abysm  access  accessori  adapt  add  addit  additon  address  \
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        1
2069     0   0      0       0          0      0    0      0        0        0

      ...  wish  wll  wold  work  wors  worst  wrong  xvyx  year  york
0     ...     0    0     0     1     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 354 columns]
```

Step 3

(3) Then construct the TF-IDF matrix from the term-document matrix.

In [60]:
```
#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 [56]:
```
DF_TF_IDF
```

Out[56]:
```
      3399   3g  abysm  access  accessori  adapt  add  addit  additon   address  ...  wll  wold
   0   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0
   1   0.0  0.0    0.0     0.0    0.27568    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0
   2   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0
   3   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0
   4   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0
 ...   ...  ...    ...     ...        ...    ...  ...    ...      ...       ...  ...  ...   ...
2065   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0
2066   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0
2067   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0
2068   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.772949  ...  0.0   0.0  0
2069   0.0  0.0    0.0     0.0    0.00000    0.0  0.0    0.0      0.0  0.000000  ...  0.0   0.0  0

2070 rows × 355 columns
```

Step 4

(4) Now combine the TF-IDF matrix with Customer data. Then do one-hot encoding on the categorical variables.

In [85]:
```
EncodeData = pd.merge(data,DF_TF_IDF, how = "left", on="ID")
```

In [86]:
```
X_cat = EncodeData.select_dtypes(exclude=['int','float64'])
#X_cat=["Sex", "Status", "Car_Owner", "Paymethod", "LocalBilltype", "LongDistanceBilltype"]
X_cat=X_cat.drop(["ID"],axis=1)
X_cat
```

Out[86]:

	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 [87]:
```
# One Hot Encoding
EncodeData = pd.get_dummies(EncodeData,columns=X_cat.columns)
```

In [88]:
```
EncodeData.to_csv("Combined.csv",index=False)
EncodeData
```

Out[88]:

	ID	Est_Income	Usage	Age	LongDistance	International	Local	3399	3g	abysm
0	1	38000.00	229.64	24.393333	23.56	0.00	206.08	0.0	0.0	0.0
1	6	29616.00	75.29	49.426667	29.78	0.00	45.50	0.0	0.0	0.0
2	8	19732.80	47.25	50.673333	24.81	0.00	22.44	0.0	0.0	0.0
3	11	96.33	59.01	56.473333	26.13	0.00	32.88	0.0	0.0	0.0
4	14	52004.80	28.14	25.140000	5.03	0.00	23.11	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
2065	3821	78851.30	29.04	48.373333	0.37	0.00	28.66	0.0	0.0	0.0
2066	3822	17540.70	36.20	62.786667	22.17	0.57	13.45	0.0	0.0	0.0
2067	3823	83891.90	74.40	61.020000	28.92	0.00	45.47	0.0	0.0	0.0
2068	3824	28220.80	38.95	38.766667	26.49	0.00	12.46	0.0	0.0	0.0
2069	3825	28589.10	100.28	15.600000	13.19	0.00	87.09	0.0	0.0	0.0

2070 rows × 387 columns

Step 5 & 6

(5) There are two types of feature selection methods - the filter type and the wrapper type. Use both types to determine the best set of features. Use at least two different classification algorithms for feature selection (in both filter and wrapper type).

Filter type - Use Python's SelectKBest module to find the K best features. Use a variety of K values to determine the best set of features for the combined data. Wrapper type - Use the Step Forward Feature Selection method in Python (see reference above) to find the best set of features on the combined data. Use cross-validation.

(6) Split the combined dataset into a training (80%) and a test set (20%). Using the best set of features from each method (filter and wrapper), build new classification models and evaluate them on the test data.

filter

In [292]:

```
# 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)
```

In [293]:

```
#classification without text featues
rfc = RandomForestClassifier()

rfc.fit(data_train,label_train)
temp_pred=rfc.predict_proba(data_test)
print("ROC score (training): {0:.6f}".format(roc_auc_score(label_test,temp_pred[:,1])))
```

```
ROC score (training): 0.901732
```

In [205]:

```
# Filter type

# score table
df_list = []

#build random forest classifier
rfc = RandomForestClassifier()

# build a for loop to find the best k

for i in range(10,386,4):
    #build select model and fit
    select_classifier = SelectKBest(score_func=chi2, k=i)
    selectbest = select_classifier.fit_transform(X_train,y_train)


    #extract column names and relabel
    feature_names = list(X_train.columns.values)
    mask = select_classifier.get_support() #list of booleans
    new_features = [] # The list of your K best features

    for bool, feature in zip(mask, feature_names):
        if bool:
            new_features.append(feature)

    temp = pd.DataFrame(selectbest, columns=new_features)

    #calculate
    rfc.fit(X_train,y_train)
    rfc_pred = rfc.predict_proba(X_test)
    df_list.append(roc_auc_score(y_test,rfc_pred[:,1]))


score = pd.DataFrame({"k":range(10,386,4),
                     "score":df_list})
```

In [206]:
```
score.sort_values(by=["score"],ascending=False)
```

Out[206]:

	k	score
80	330	0.930423
13	62	0.929412
52	218	0.929165
18	82	0.928881
41	174	0.928573
...	...	...
63	262	0.918128
16	74	0.917980
67	278	0.917228
32	138	0.915982
73	302	0.914687

94 rows × 2 columns

**According to the roc score table, we can see that when k = 330 the model returns highest roc score.**

In [250]:

```
select_classifier = SelectKBest(score_func=chi2, k=330)
selectbest = select_classifier.fit_transform(EncodeData.drop(["ID"],axis=1),label)

#extract column names and relabel
feature_names = list(EncodeData.drop(["ID"],axis=1).columns.values)
mask = select_classifier.get_support() #list of booleans
new_features = [] # The list of your K best features

for bool, feature in zip(mask, feature_names):
    if bool:
        new_features.append(feature)

Feature331 = pd.DataFrame(selectbest, columns=new_features)
Feature331["ID"]=EncodeData["ID"]
Feature331.to_csv("Feature331.csv",index=False)
Feature331
```

Out[250]:

	Est_Income	Usage	Age	LongDistance	International	Local	3399	3g	access	acces
0	38000.00	229.64	24.393333	23.56	0.00	206.08	0.0	0.0	0.0	
1	29616.00	75.29	49.426667	29.78	0.00	45.50	0.0	0.0	0.0	
2	19732.80	47.25	50.673333	24.81	0.00	22.44	0.0	0.0	0.0	
3	96.33	59.01	56.473333	26.13	0.00	32.88	0.0	0.0	0.0	
4	52004.80	28.14	25.140000	5.03	0.00	23.11	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
2065	78851.30	29.04	48.373333	0.37	0.00	28.66	0.0	0.0	0.0	
2066	17540.70	36.20	62.786667	22.17	0.57	13.45	0.0	0.0	0.0	
2067	83891.90	74.40	61.020000	28.92	0.00	45.47	0.0	0.0	0.0	
2068	28220.80	38.95	38.766667	26.49	0.00	12.46	0.0	0.0	0.0	
2069	28589.10	100.28	15.600000	13.19	0.00	87.09	0.0	0.0	0.0	

2070 rows × 331 columns

In [294]:

```
#split the feature 331
X, x, Y, y = train_test_split(pd.get_dummies(Feature331.drop(["ID"],axis=1)), label,
                                                    test_size = 0.2,
                                                   random_state = 42)
```

In [295]:

```
# catboost
cat=CatBoostClassifier()
cat.fit(X,Y)
```

```
100:	learn: 0.3525557	total: 426ms	remaining: 3.79s
101:	learn: 0.3517244	total: 430ms	remaining: 3.79s
102:	learn: 0.3501677	total: 434ms	remaining: 3.78s
103:	learn: 0.3489879	total: 438ms	remaining: 3.77s
104:	learn: 0.3484566	total: 441ms	remaining: 3.76s
105:	learn: 0.3478148	total: 445ms	remaining: 3.75s
106:	learn: 0.3469609	total: 449ms	remaining: 3.74s

107:	learn: 0.3452833	total: 452ms	remaining: 3.74s
108:	learn: 0.3447843	total: 457ms	remaining: 3.74s
109:	learn: 0.3437185	total: 462ms	remaining: 3.74s
110:	learn: 0.3422590	total: 466ms	remaining: 3.73s
111:	learn: 0.3405673	total: 470ms	remaining: 3.72s
112:	learn: 0.3388857	total: 474ms	remaining: 3.72s
113:	learn: 0.3378368	total: 477ms	remaining: 3.71s
114:	learn: 0.3369632	total: 481ms	remaining: 3.7s
115:	learn: 0.3362697	total: 485ms	remaining: 3.69s
116:	learn: 0.3356392	total: 490ms	remaining: 3.7s
117:	learn: 0.3348649	total: 495ms	remaining: 3.7s
118:	learn: 0.3337122	total: 498ms	remaining: 3.69s
119:	learn: 0.3329681	total: 502ms	remaining: 3.68s
```

In [296]:

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

```
ROC score (training): 0.915884
Confusion Matrix:
[[134  23]
 [ 20 237]]
Classification Report
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       157
           1       0.91      0.92      0.92       257

    accuracy                           0.90       414
   macro avg       0.89      0.89      0.89       414
weighted avg       0.90      0.90      0.90       414
```

In [297]:

```
np.mean(cat.get_feature_importance())
```

Out[297]:

```
0.30303030303030293
```

In [298]:

```
dtc=DecisionTreeClassifier()
dtc.fit(X,Y)
dtc_predictions = dtc.predict_proba(x)
print("ROC score (training): {0:.6f}".format(roc_auc_score(y,dtc_predictions[:,1])))
print("Confusion Matrix:")
print(confusion_matrix(y, dtc_predictions[:,1].round()))
print("Classification Report")
print(classification_report(y, dtc_predictions[:,1].round()))
```

```
ROC score (training): 0.851446
Confusion Matrix:
[[135  22]
 [ 39 218]]
Classification Report
              precision    recall  f1-score   support

           0       0.78      0.86      0.82       157
           1       0.91      0.85      0.88       257

    accuracy                           0.85       414
   macro avg       0.84      0.85      0.85       414
weighted avg       0.86      0.85      0.85       414
```

In [299]:

```
rfc=RandomForestClassifier()
rfc.fit(X,Y)
rfc_predictions = rfc.predict_proba(x)
print("ROC score (training): {0:.6f}".format(roc_auc_score(y,rfc_predictions[:,1])))
print("Confusion Matrix:")
print(confusion_matrix(y, rfc_predictions[:,1].round()))
print("Classification Report")
print(classification_report(y, rfc_predictions[:,1].round()))
```

```
ROC score (training): 0.911658
Confusion Matrix:
[[134  23]
 [ 24 233]]
Classification Report
              precision    recall  f1-score   support

           0       0.85      0.85      0.85       157
           1       0.91      0.91      0.91       257

    accuracy                           0.89       414
   macro avg       0.88      0.88      0.88       414
weighted avg       0.89      0.89      0.89       414
```

wrapper - SequentialFeatureSelection

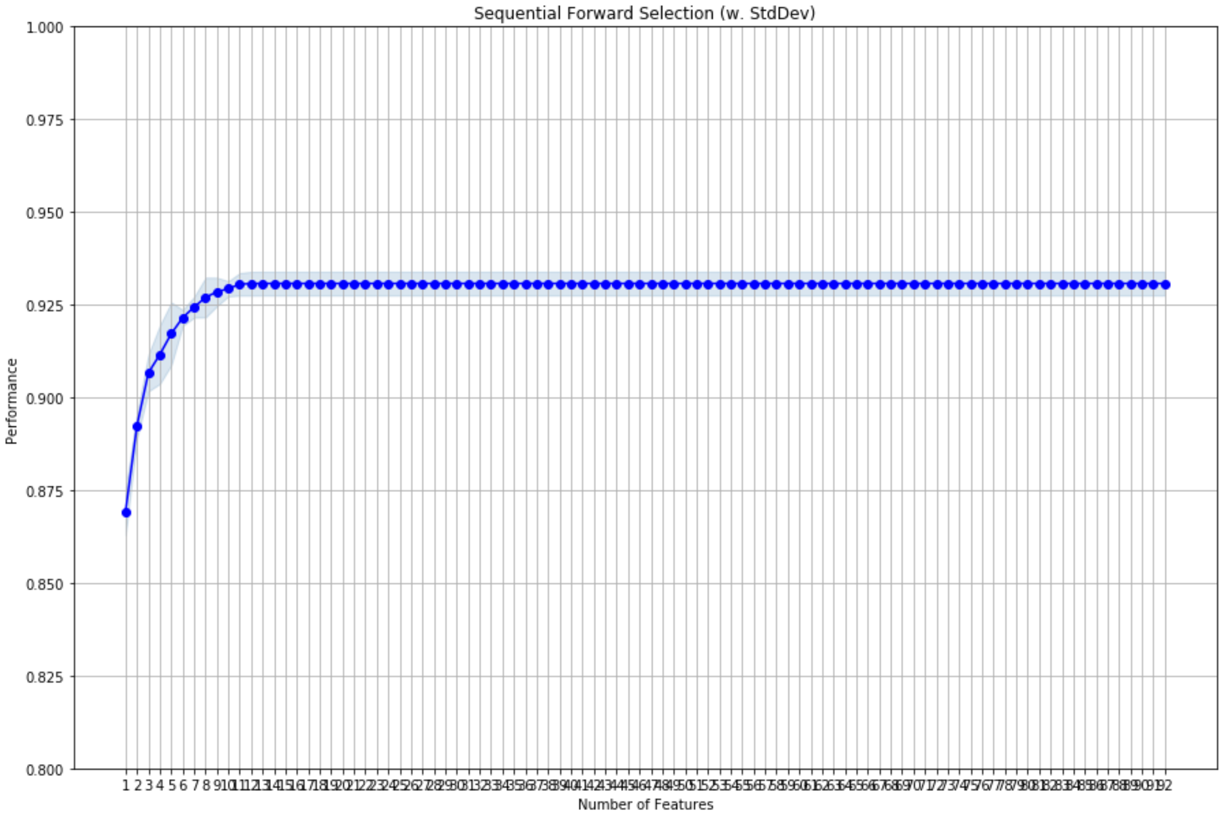
In [225]:

```
# Wrapper type

#build random forest classifier
xgb = XGBClassifier()

# build a for loop to find the best 331

sfs1_classifier = sfs(xgb,
                      k_features= 330,
                      forward=True,
                      floating=False,
                      verbose=2,
                      scoring='roc_auc',
                      n_jobs=-1,
                      cv=3)

# Perform SFFS
sfs1 = sfs1_classifier.fit(X_train, y_train)

fig1 = plot_sfs(sfs1.get_metric_dict(),
                kind='std_dev',
                figsize=(6, 4))

plt.ylim([0.8, 1])
plt.title('Sequential Forward Selection (w. StdDev)')
plt.grid()
plt.show()
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    4.5s
[Parallel(n_jobs=-1)]: Done 234 tasks      | elapsed:    6.9s
[Parallel(n_jobs=-1)]: Done 386 out of 386 | elapsed:    8.6s finished

[2020-10-30 17:54:59] Features: 1/330 -- score: 0.8692475789570538[Parallel(n_jobs=-
1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    0.7s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    3.7s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:    8.7s
[Parallel(n_jobs=-1)]: Done 385 out of 385 | elapsed:    9.6s finished

[2020-10-30 17:55:09] Features: 2/330 -- score: 0.8923860787155246[Parallel(n_jobs=-
1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    0.9s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    4.5s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   10.6s
[Parallel(n_jobs=-1)]: Done 384 out of 384 | elapsed:   11.5s finished

[2020-10-30 17:55:21] Features: 3/330 -- score: 0.8966063898742848[Parallel(n_jobs=-
```

In [227]:

```
fig1 = plot_sfs(sfs1.get_metric_dict(),
                kind='std_dev',
                figsize=(15, 10))

plt.ylim([0.8, 1])
plt.title('Sequential Forward Selection (w. StdDev)')
plt.grid()
plt.show()
```



```
In  [239]: featurenames92
```

```
Out[239]: ('Est_Income’,
           'Age’,
           'LongDistance’,
           '3g’,
           'abysm’,
           'access’,
           'adapt’,
           'adit’,
           'adress’,
           'advertis’,
           'afraid’,
           'alway’,
           'angel’,
           'angri’,
           'anyth’,
           'anytim’,
           'bateri’,
           'believ’,
           'bigger’,
           'book’,
           'bought’,
           'brain’,
           'busi’,
           'button’,
           'buy’,
           'cancer’,
           'carrier’,
           'caus’,
           'cc’,
           'cell’,
           'certain’,
           'charger’,
           'check’,
           'chip’,
           'citi’,
           'claim’,
           'cold’,
           'comapr’,
           'compani’,
           'compar’,
           'competit’,
           'complain’,
           'complaint’,
           'concept’,
           'connect’,
           'consisit’,
           'consist’,
           'constan’,
           'continu’,
           'contract’,
           'correct’,
           'cost’,
           'cover’,
           'coverag’,
           'creat’,
           'credit’,
           'cstmer’,
           'cstmr’,
           'current’,
           'cust’,
           'customr’,
           'date’,
           'day’,
           'decent’,
           'defect’,
           'deo’,
           'die’,
           'differ’,
           'difficult’,
           'digiti’,
           'direct’,
           'don’,
           'dont’,
           'drop’,
           'dure’,
           'easier’,
           'effect’,
           'encount’,
           'enemi’,
           'equip’,
           'everytim’,
           'everywher’,
           'evrey’,
           'exact’,
           'expir’,
           'pay’,
           'phone’,
           'Sex_F’,
           'Status_M’,
           'Children_2’,
           'RatePlan_2’,
           'RatePlan_4’)
```

In [230]:

```
xgb = XGBClassifier()

# build a for loop to find the best 12

sfs1_classifier = sfs(xgb,
                      k_features= 12,
                      forward=True,
                      floating=False,
                      verbose=2,
                      scoring='roc_auc',
                      n_jobs=-1,
                      cv=3)

# Perform SFFS
sfs1 = sfs1_classifier.fit(X_train, y_train)
```

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    6.0s
[Parallel(n_jobs=-1)]: Done 205 tasks      | elapsed:    8.4s
[Parallel(n_jobs=-1)]: Done 371 out of 386 | elapsed:   10.5s remaining:    0.3s
[Parallel(n_jobs=-1)]: Done 386 out of 386 | elapsed:   10.6s finished

[2020-10-30 19:07:12] Features: 1/12 -- score: 0.8692475789570538[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    0.8s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    4.1s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:    9.7s
[Parallel(n_jobs=-1)]: Done 385 out of 385 | elapsed:   10.7s finished

[2020-10-30 19:07:22] Features: 2/12 -- score: 0.8923860787155246[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    1.0s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    5.5s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   12.4s
[Parallel(n_jobs=-1)]: Done 384 out of 384 | elapsed:   13.6s finished

[2020-10-30 19:07:36] Features: 3/12 -- score: 0.9066063993743243[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    1.2s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    5.7s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   14.0s
[Parallel(n_jobs=-1)]: Done 383 out of 383 | elapsed:   15.4s finished

[2020-10-30 19:07:52] Features: 4/12 -- score: 0.9115312952867297[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    1.4s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    7.0s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   18.1s
[Parallel(n_jobs=-1)]: Done 382 out of 382 | elapsed:   19.9s finished

[2020-10-30 19:08:12] Features: 5/12 -- score: 0.9171531755342396[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    1.6s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    8.0s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   18.5s
```

```
[Parallel(n_jobs=-1)]: Done 381 out of 381 | elapsed:   20.2s finished

[2020-10-30 19:08:32] Features: 6/12 -- score: 0.9214615968532193[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    2.0s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:   10.2s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   23.1s
[Parallel(n_jobs=-1)]: Done 380 out of 380 | elapsed:   25.1s finished

[2020-10-30 19:08:57] Features: 7/12 -- score: 0.9243139419869805[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    1.8s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    9.3s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   20.6s
[Parallel(n_jobs=-1)]: Done 379 out of 379 | elapsed:   22.3s finished

[2020-10-30 19:09:20] Features: 8/12 -- score: 0.9269224576173718[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    9.6s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   22.4s
[Parallel(n_jobs=-1)]: Done 378 out of 378 | elapsed:   24.3s finished

[2020-10-30 19:09:44] Features: 9/12 -- score: 0.9284498424309341[Parallel(n_jobs=-1)]:
Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    9.5s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   22.8s
[Parallel(n_jobs=-1)]: Done 377 out of 377 | elapsed:   24.6s finished

[2020-10-30 19:10:09] Features: 10/12 -- score: 0.9292894440227268[Parallel(n_jobs=-
1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    1.7s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:    8.6s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   22.4s
[Parallel(n_jobs=-1)]: Done 376 out of 376 | elapsed:   24.3s finished

[2020-10-30 19:10:33] Features: 11/12 -- score: 0.9304993904262416[Parallel(n_jobs=-
1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:    2.3s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:   10.4s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:   23.4s
[Parallel(n_jobs=-1)]: Done 375 out of 375 | elapsed:   24.9s finished

[2020-10-30 19:10:58] Features: 12/12 -- score: 0.9306512087962643
```

In [253]:

```
featurenames12=['Est_Income',
 'Age',
 'LongDistance',
 'bigger',
 'buy',
 'pay',
 'phone',
 'Sex_F',
 'Status_M',
 'Children_2',
 'RatePlan_2',
 'RatePlan_4']
```

In [255]:

```
Feature13 = pd.DataFrame(EncodeData[featurenames12])
Feature13["ID"]=EncodeData["ID"]
Feature13.to_csv("Feature13.csv",index=False)
Feature13
```

Out[255]:

	Est_Income	Age	LongDistance	bigger	buy	pay	phone	Sex_F	Status_M	Childre
0	38000.00	24.393333	23.56	0.0	0.0	0.0	0.000000	1	0	
1	29616.00	49.426667	29.78	0.0	0.0	0.0	0.243227	0	1	
2	19732.80	50.673333	24.81	0.0	0.0	0.0	0.243227	0	1	
3	96.33	56.473333	26.13	0.0	0.0	0.0	0.243227	0	0	
4	52004.80	25.140000	5.03	0.0	0.0	0.0	0.243227	1	1	
...	...	...	...	...	...	...	...	...	...	
2065	78851.30	48.373333	0.37	0.0	0.0	0.0	0.264422	1	0	
2066	17540.70	62.786667	22.17	0.0	0.0	0.0	0.264422	1	0	
2067	83891.90	61.020000	28.92	0.0	0.0	0.0	0.264422	1	1	
2068	28220.80	38.766667	26.49	0.0	0.0	0.0	0.264422	1	1	
2069	28589.10	15.600000	13.19	0.0	0.0	0.0	0.264422	1	0	

2070 rows × 13 columns

In [300]:

```
#split the feature 13
X, x, Y, y = train_test_split(pd.get_dummies(Feature13.drop(["ID"],axis=1)), label,
                                                    test_size = 0.2,
                                                   random_state = 42)
```

In [301]:

```
cat=CatBoostClassifier()
cat.fit(X,Y)
```

```
108:	learn: 0.3450322	total: 260ms	remaining: 2.12s
109:	learn: 0.3444191	total: 262ms	remaining: 2.12s
110:	learn: 0.3434343	total: 264ms	remaining: 2.11s
111:	learn: 0.3421930	total: 266ms	remaining: 2.11s
112:	learn: 0.3406827	total: 268ms	remaining: 2.1s
113:	learn: 0.3401853	total: 270ms	remaining: 2.1s
114:	learn: 0.3388709	total: 272ms	remaining: 2.1s
115:	learn: 0.3378507	total: 275ms	remaining: 2.1s
116:	learn: 0.3370100	total: 278ms	remaining: 2.1s
117:	learn: 0.3365231	total: 280ms	remaining: 2.09s
118:	learn: 0.3356905	total: 282ms	remaining: 2.09s
119:	learn: 0.3345480	total: 284ms	remaining: 2.08s
120:	learn: 0.3334834	total: 286ms	remaining: 2.08s
121:	learn: 0.3326608	total: 289ms	remaining: 2.08s
122:	learn: 0.3320038	total: 291ms	remaining: 2.07s
123:	learn: 0.3312155	total: 293ms	remaining: 2.07s
124:	learn: 0.3304272	total: 295ms	remaining: 2.06s
125:	learn: 0.3297294	total: 297ms	remaining: 2.06s
126:	learn: 0.3289325	total: 299ms	remaining: 2.06s
127:	learn: 0.3282520	total: 302ms	remaining: 2.06s
```

In [302]:

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

```
ROC score (training): 0.909973
Confusion Matrix:
[[132  25]
 [ 20 237]]
Classification Report
              precision    recall  f1-score   support

           0       0.87      0.84      0.85       157
           1       0.90      0.92      0.91       257

    accuracy                           0.89       414
   macro avg       0.89      0.88      0.88       414
weighted avg       0.89      0.89      0.89       414
```

In [303]:

```
dtc=DecisionTreeClassifier()
dtc.fit(X,Y)
dtc_predictions = dtc.predict_proba(x)
print("ROC score (training): {0:.6f}".format(roc_auc_score(y,dtc_predictions[:,1])))
print("Confusion Matrix:")
print(confusion_matrix(y, dtc_predictions[:,1].round()))
print("Classification Report")
print(classification_report(y, dtc_predictions[:,1].round()))
```

```
ROC score (training): 0.851087
Confusion Matrix:
[[132  25]
 [ 43 214]]
Classification Report
              precision    recall  f1-score   support

           0       0.75      0.84      0.80       157
           1       0.90      0.83      0.86       257

    accuracy                           0.84       414
   macro avg       0.82      0.84      0.83       414
weighted avg       0.84      0.84      0.84       414
```

In [304]:

```
rfc=RandomForestClassifier()
rfc.fit(X,Y)
rfc_predictions = rfc.predict_proba(x)
print("ROC score (training): {0:.6f}".format(roc_auc_score(y,rfc_predictions[:,1])))
print("Confusion Matrix:")
print(confusion_matrix(y, rfc_predictions[:,1].round()))
print("Classification Report")
print(classification_report(y, rfc_predictions[:,1].round()))
```

```
ROC score (training): 0.904880
Confusion Matrix:
[[128  29]
 [ 24 233]]
Classification Report
              precision    recall  f1-score   support

           0       0.84      0.82      0.83       157
           1       0.89      0.91      0.90       257

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