This feature selection notebook does a filter followed by a wrapper for a binary dependent variable (binary classification). It's capable of doing the filter on more than one file. The variable files are called vars1.csv, vars2.csv ... Or you can make the input file name(s) anything you want.

The filter runs separately on each vars file and keeps the top num_filter variables from each file. If there are more than one vars files we'll again select the top num_filter variables across all the vars.csv files.

If balance = 0 the entire files are used. If balance != 0 then balance is the RATIO OF BADS TO GOODS retained for the rest of the feature selection. We keep all the rare class (bads) and downsample the goods. I think in general it's better to keep balance = 0.

I've got an annoying warning message from the wrapper and I can't figure out how to get rid of it. If anybody figures this out please send a message to stevecoggeshall@gmail.com

In [1]:

```
import pandas as pd
import numpy as np
import scipy.stats as sps
import matplotlib.pyplot as plt
import datetime as dt
import gc
from sklearn.ensemble import RandomForestClassifier
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from lightgbm import LGBMClassifier
%matplotlib inline
start_time = dt.datetime.now()
```

In [2]:

```
# set some parameters
num_files = 1
# I recommend set num_filter to be about 10 to 20% of the original # variabl
num_filter = 200
# I recommend set num_wrapper to be about 50, then look for a saturation of
# Then you can run it again with num_wrapper just a bit above this saturatic
num_wrapper = 20
balance = 0
detect_rate = .03
index_name = 'Recnum'
y_name = 'Fraud'
good_label = 0
bad_label = 1
```

Run a filter on all the files

In [3]:

```
%%time
filter_score_df_list = []
for i in range(num_files):
#     file_name = "vars"+str(i+1)+'.csv'
```

```
    file_name = 'candidate_variables.csv'
    df = pd.read_csv(file_name)
    print("********** working on",file_name,"size is",df.shape)
    df = df.set_index(index_name)
    df = df[df.index <= 85264] # remove the last two months as the out-of-ti
    df = df[df.index >= 3466] # remove the first 2 weeks of records since th
    df['RANDOM'] = np.random.ranf(len(df)) # add a random number variable to
    goods = df[df[y_name] == good_label]
    bads = df[df[y_name] == bad_label]
    del df # don't need this file anymore
    num_goods = len(goods)
    num_bads = len(bads)
    num_vars = len(bads.columns)-2
    if(balance != 0):
        if(i == 0):
            num_goods_desired = int(min(num_goods,num_bads*balance))
            goods = goods.sample(n=num_goods_desired,random_state=1)
            goods_keep = list(goods.index)
            goods_keep.sort()

        if(i > 0):
            goods = goods.loc[goods_keep]

    df_sampled = pd.concat([goods,bads])
    df_sampled.sort_index(inplace=True)
    filter_score = pd.DataFrame(np.zeros((num_vars+1,2)))
    filter_score.columns = ['variable','filter score']
    j = 0
    for column in df_sampled:
        filter_score.loc[j,'variable'] = column
        filter_score.loc[j,'filter score'] = sps.ks_2samp(goods[column],bads
        j = j+1
        if j%100 == 0:
            print(j)

    filter_score.sort_values(by=['filter score'], ascending=False, inplace=T
    vars_keep = list(filter_score['variable'][1:num_filter+1])
    print(file_name,filter_score.head(20))
    if(i == 0): # if first time through need to initialize some stuff
        Y = pd.DataFrame(df_sampled[y_name], index=df_sampled.index)
        df_top = df_sampled.filter(vars_keep, axis=1)

    if(i > 0): # if more than one variable file we use this loop
        data_new_top = df_sampled.filter(vars_keep, axis=1)
        df_top = pd.concat([df_top,data_new_top], axis=1)

    filter_score_df_list.append(filter_score)

    del goods # delete these before starting the next file, if any
    del bads
    gc.collect()
filter_score = pd.concat(filter_score_df_list)
```

```
********** working on candidate_variables.csv size is (97496, 2635)
```

```
<timed exec>:10: PerformanceWarning: DataFrame is highly fragmented.  This
is usually the result of calling `frame.insert` many times, which has poor
performance.  Consider joining all columns at once using pd.concat(axis=1)
instead. To get a de-fragmented frame, use `newframe = frame.copy()`
```

```
100
200
300
400
500
600
700
800
900
1000
1100
1200
1300
1400
1500
1600
1700
1800
1900
2000
2100
2200
2300
2400
2500
2600
candidate_variables.csv                    variable  filter score
1                      Fraud      1.000000
28           Cardnum_total_3      0.621932
19           Cardnum_total_1      0.619658
10           Cardnum_total_0      0.591318
15           Cardnum_count_1      0.567429
37           Cardnum_total_7      0.564245
24           Cardnum_count_3      0.563356
586         Card_dow_total_7      0.542303
17             Cardnum_max_1      0.532080
33           Cardnum_count_7      0.526897
8              Cardnum_max_0      0.525418
6            Cardnum_count_0      0.516123
595        Card_dow_total_14      0.511203
1577   Cardnum_vdratio_1by30      0.504966
1578   Cardnum_vdratio_1by60      0.502927
26             Cardnum_max_3      0.501170
46          Cardnum_total_14      0.494375
1645  Card_dow_vdratio_0by30      0.489227
1646  Card_dow_vdratio_0by60      0.486480
584           Card_dow_max_7      0.486177
CPU times: user 52.6 s, sys: 3.09 s, total: 55.7 s
Wall time: 56.3 s
```

In [4]:
```
filter_score.sort_values(by=['filter score'], ascending=False, inplace=True)
filter_score.reset_index(drop=True,inplace=True)
```

In [5]:

```
filter_score.head(30)
```

Out[5]:

	variable	filter score
0	Fraud	1.000000
1	Cardnum_total_3	0.621932
2	Cardnum_total_1	0.619658
3	Cardnum_total_0	0.591318
4	Cardnum_count_1	0.567429
5	Cardnum_total_7	0.564245
6	Cardnum_count_3	0.563356
7	Card_dow_total_7	0.542303
8	Cardnum_max_1	0.532080
9	Cardnum_count_7	0.526897
10	Cardnum_max_0	0.525418
11	Cardnum_count_0	0.516123
12	Card_dow_total_14	0.511203
13	Cardnum_vdratio_1by30	0.504966
14	Cardnum_vdratio_1by60	0.502927
15	Cardnum_max_3	0.501170
16	Cardnum_total_14	0.494375
17	Card_dow_vdratio_0by30	0.489227
18	Card_dow_vdratio_0by60	0.486480
19	Card_dow_max_7	0.486177
20	Cardnum_vdratio_1by14	0.485431
21	Cardnum_variability_max_0	0.484245
22	Card_dow_count_7	0.482384
23	Cardnum_actual/toal_0	0.479550
24	Card_dow_vdratio_0by14	0.479086
25	Cardnum_variability_max_1	0.477836
26	Cardnum_unique_count_for_card_state_1	0.476067
27	Cardnum_unique_count_for_card_zip_1	0.474960
28	Card_dow_total_30	0.474759
29	Cardnum_unique_count_for_Merchnum_1	0.472017

In [6]:

```
filter_score.tail(10)
```

Out[6]:

	variable	filter score
2625	card_merch_unique_count_for_card_state_7	0.000088
2626	Merchnum_desc_Zip_unique_count_for_Merchnum_de...	0.000075
2627	card_merch_unique_count_for_card_state_3	0.000063
2628	Merchdesc_Zip_unique_count_for_Merchdesc_State_7	0.000038
2629	Card_Merchdesc_Zip_unique_count_for_Merchdesc_...	0.000038
2630	Merchdesc_Zip_unique_count_for_Merchdesc_State_14	0.000038
2631	merch_zip_unique_count_for_merch_state_14	0.000025
2632	merch_zip_unique_count_for_merch_state_7	0.000013
2633	Merchnum_desc_Zip_unique_count_for_Merchnum_de...	0.000013
2634	card_merch_unique_count_for_Cardnum_1	0.000000

In [7]:

```
filter_score.shape
```

Out[7]:

```
(2635, 2)
```

In [8]:

```
filter_score.head(80).to_csv('filter_top.csv')
vars_keep = list(filter_score['variable'][num_files:num_filter+3])
print(i,' vars_keep:',vars_keep)
```

```
0  vars_keep: ['Cardnum_total_3', 'Cardnum_total_1', 'Cardnum_total_0', 'Ca
rdnum_count_1', 'Cardnum_total_7', 'Cardnum_count_3', 'Card_dow_total_7',
'Cardnum_max_1', 'Cardnum_count_7', 'Cardnum_max_0', 'Cardnum_count_0', 'Ca
rd_dow_total_14', 'Cardnum_vdratio_1by30', 'Cardnum_vdratio_1by60', 'Cardnu
m_max_3', 'Cardnum_total_14', 'Card_dow_vdratio_0by30', 'Card_dow_vdratio_0
by60', 'Card_dow_max_7', 'Cardnum_vdratio_1by14', 'Cardnum_variability_max_
0', 'Card_dow_count_7', 'Cardnum_actual/toal_0', 'Card_dow_vdratio_0by14',
'Cardnum_variability_max_1', 'Cardnum_unique_count_for_card_state_1', 'Card
num_unique_count_for_card_zip_1', 'Card_dow_total_30', 'Cardnum_unique_coun
t_for_Merchnum_1', 'Card_dow_max_14', 'Card_dow_vdratio_0by7', 'Cardnum_vdr
atio_1by7', 'Cardnum_unique_count_for_card_state_3', 'Cardnum_unique_count_
for_card_zip_3', 'Cardnum_total_amount_1_by_60', 'Cardnum_unique_count_for_
Merchnum_3', 'Cardnum_actual/toal_1', 'Card_dow_unique_count_for_merch_stat
e_1', 'Card_dow_unique_count_for_Card_Merchdesc_1', 'Card_dow_unique_count_
for_state_des_1', 'Card_dow_unique_count_for_merch_zip_1', 'Cardnum_unique_
count_for_card_state_7', 'Cardnum_actual/max_0', 'Cardnum_count_14', 'Card_
dow_count_14', 'Cardnum_unique_count_for_card_zip_7', 'Cardnum_unique_count
_for_Merchnum_7', 'Card_dow_total_60', 'Cardnum_total_amount_1_by_30', 'Car
d_dow_day_since', 'Cardnum_day_since', 'Cardnum_count_1_by_30', 'Cardnum_co
unt_1_by_30_sq', 'Cardnum_vdratio_0by60', 'Card_dow_max_30', 'Cardnum_uniqu
e_count_for_card_state_14', 'Cardnum_count_1_by_60_sq', 'Cardnum_count_1_by
_60', 'Card_dow_unique_count_for_merch_zip_7', 'Cardnum_actual/max_1', 'Car
d_dow_unique_count_for_merch_state_7', 'Cardnum_unique_count_for_card_zip_1
4', 'Cardnum_unique_count_for_Merchnum_14', 'Cardnum_count_1_by_14_sq', 'Ca
rdnum_count_1_by_14', 'Cardnum_vdratio_0by30', 'Cardnum_total_30', 'Cardnum
_max_7', 'Cardnum_actual/toal_3', 'Cardnum_variability_max_3', 'Card_dow_va
riability_max_7', 'Card_dow_unique_count_for_merch_state_14', 'Card_dow_uni
que_count_for_merch_zip_14', 'Card_dow_unique_count_for_Card_Merchdesc_7',
'Card_dow_unique_count_for_state_des_7', 'Card_dow_count_30', 'Card_dow_act
ual/toal_7', 'Cardnum_unique_count_for_card_state_30', 'Card_dow_max_60',
'Cardnum_vdratio_0by14', 'Cardnum_total_amount_0_by_60', 'Card_dow_unique_c
ount_for_state_des_14', 'Card_dow_unique_count_for_Card_Merchdesc_14', 'Car
dnum_unique_count_for_card_zip_30', 'card_state_total_3', 'card_state_total
_1', 'Card_dow_unique_count_for_merch_zip_30', 'Card_dow_unique_count_for_m
erch_state_30', 'Cardnum_avg_0', 'Cardnum_unique_count_for_Merchnum_30', 'C
ardnum_total_amount_1_by_14', 'Card_dow_count_0_by_60_sq', 'Card_dow_count_
0_by_60', 'Card_dow_variability_max_14', 'Cardnum_avg_1', 'Card_dow_actual/
max_7', 'Cardnum_vdratio_0by7', 'Cardnum_count_30', 'Cardnum_max_60', 'Card
num_unique_count_for_card_state_60', 'card_state_max_3', 'Cardnum_total_6
0', 'card_state_total_7', 'Card_dow_total_amount_0_by_60', 'card_state_tota
l_0', 'card_state_max_1', 'Card_dow_unique_count_for_state_des_30', 'Card_d
ow_unique_count_for_Card_Merchdesc_30', 'Cardnum_avg_3', 'Cardnum_actual/ma
x_3', 'Cardnum_total_amount_0_by_30', 'Card_Merchdesc_State_total_14', 'Car
d_Merchdesc_total_14', 'card_zip_total_14', 'Cardnum_unique_count_for_card_
zip_60', 'card_state_total_14', 'Card_Merchdesc_Zip_total_14', 'Card_dow_ac
tual/toal_14', 'card_state_vdratio_1by60', 'Card_Merchnum_desc_total_14',
'card_state_max_7', 'card_merch_total_14', 'Card_Merchnum_State_total_14',
'Card_Merchnum_Zip_total_14', 'Card_dow_count_0_by_30_sq', 'Card_dow_count_
0_by_30', 'Card_dow_avg_7', 'card_zip_total_7', 'Cardnum_variability_avg_
0', 'Card_Merchdesc_State_total_7', 'Card_Merchdesc_total_7', 'card_zip_tot
al_3', 'Card_Merchdesc_Zip_total_7', 'Card_dow_avg_14', 'card_zip_total_1',
'Card_dow_count_60', 'Card_Merchnum_desc_total_7', 'card_merch_total_3', 'C
ard_Merchnum_State_total_3', 'Card_dow_unique_count_for_merch_state_60', 'c
ard_merch_total_7', 'Card_Merchnum_State_total_7', 'card_state_total_amount
_1_by_60', 'Card_dow_unique_count_for_merch_zip_60', 'Card_Merchnum_Zip_tot
al_7', 'Card_Merchdesc_State_total_3', 'Card_Merchdesc_total_3', 'card_zip_
total_amount_1_by_60', 'Card_Merchnum_Zip_total_3', 'Card_Merchdesc_Zip_tot
al_3', 'card_zip_total_30', 'Cardnum_max_14', 'card_state_max_0', 'Cardnum_
count_0_by_60_sq', 'Cardnum_count_0_by_60', 'Card_Merchdesc_State_total_3
0', 'Card_Merchdesc_total_30', 'Card_Merchnum_desc_total_30', 'card_merch_t
otal_1', 'Card_Merchnum_State_total_30', 'card_merch_total_30', 'Card_Merch
desc_Zip_total_30', 'state_des_total_3', 'Card_Merchnum_Zip_total_1', 'card
_zip_count_1_by_60', 'card_zip_count_1_by_60_sq', 'Card_Merchnum_Zip_total_
30', 'card_state_vdratio_1by30', 'card_zip_max_3', 'Card_Merchnum_desc_tota
l_3', 'Merchdesc_Zip_total_3', 'state_des_total_1', 'Cardnum_count_1_by_7',
'Cardnum_count_1_by_7_sq', 'merch_state_total_3', 'Merchnum_total_3', 'card
_zip_max_7', 'Merchdesc_Zip_total_1', 'merch_zip_total_3', 'card_state_coun
t_1_by_60_sq', 'card_state_count_1_by_60', 'card_merch_max_3', 'Card_Merchn
um_Zip_max_3', 'Card_Merchnum_State_max_3', 'card_zip_max_1', 'Merchnum_des
c_State_total_3', 'Merchnum_desc_total_3', 'card_state_total_30', 'Cardnum_
variability_max_7', 'Card_dow_avg_30', 'Card_dow_variability_max_30', 'Card
num_unique_count_for_Merchnum_60', 'Card_Merchnum_desc_total_60', 'card_mer
ch_max_7', 'Card_Merchnum_State_max_7', 'Card_Merchnum_Zip_max_7', 'card_st
ate_max_14', 'Merchnum_desc_Zip_total_3', 'Card_Merchnum_State_total_60',
'card_merch_total_60', 'Merchnum_desc_State_total_1', 'Merchnum_desc_total_
1']
```

In [9]:

```
vars_keep_df = pd.DataFrame({'col':vars_keep})
vars_keep_df.to_csv('vars_keep_filter.csv',index=False)
df_keep = df_top.filter(vars_keep, axis=1)
df_keep.head()
```

Out[9]:

	Cardnum_total_3	Cardnum_total_1	Cardnum_total_0	Cardnum_count_1	Cardnum_
Recnum					
3466	1964.73	1551.02	333.47	3	
3467	834.91	660.87	4.37	2	
3468	8459.94	2904.08	271.93	7	
3469	838.53	664.49	7.99	3	
3470	16.28	16.28	16.28	1	

5 rows × 200 columns

In [10]:

```
df_keep.shape
```

Out[10]:

```
(81476, 200)
```

In [11]:

```
Y.head()
```

Out[11]:

	Fraud
Recnum	
3466	0
3467	0
3468	0
3469	0
3470	0

In [12]:
```
Y = Y.values.ravel()
Y_save = Y.copy()
```

In [13]:
```
# Y = np.array(Y)
X = df_keep
print(Y)
```

```
[0 0 0 ... 0 0 0]
```

In [14]:
```
print('time to here:', dt.datetime.now() - start_time)
```

```
time to here: 0:00:56.408409
```

In [15]:
```
print(X.shape,Y.shape)
```

```
(81476, 200) (81476,)
```

In [16]:
```
print(type(X),type(Y))
```

```
<class 'pandas.core.frame.DataFrame'> <class 'numpy.ndarray'>
```

In [17]:
```
# I'd like to define a scoring for the wrapper that's KS, but I haven't gott
# def KSscore(classifier, x,y)
```

In [18]:
```
def fdr(classifier, x, y, cutoff=detect_rate):
# Calculates FDR score for the given classifier on dataset x and y with cutc
# get the probability list from the given classifier
    return fdr_prob(y, classifier.predict_proba(x), cutoff)
def fdr_prob(y, y_prob, cutoff=detect_rate):
    if len(y_prob.shape) != 1:    # sometimes the proba list can contain mar
        y_prob = y_prob[:, -1:]   # only the last one (fraud_label==1) is us
    num_fraud = len(y[y == 1])    # count the total nunber of frauds
# sort the proba list from high to low while retain the true (not predicted)
    sorted_prob = np.asarray(sorted(zip(y_prob, y), key=lambda x: x[0], reve
    cutoff_bin = sorted_prob[0:int(len(y) * cutoff), 1:]  # 3% cutoff
# return the FDR score (#fraud_in_cutoff / #total_fraud)
    return len(cutoff_bin[cutoff_bin == 1]) / num_fraud
```

Run a wrapper on the remaining top variables

In [19]:
```
# This is a parallel running parameter. You can try it set to -1, but someti
# be divided into that many threads and the next cell quits. A safe value is
# the runs slower. You might experiment to see how big you can set this for
njobs = 1
```

In [20]:

```
%%time
import warnings
warnings.filterwarnings("ignore")
# If you're doing forward selection it's enough to stop at num_wrapper varia
# If you're doing backward selection you need to go through all the variable

nfeatures = len(X.columns)
# clf = RandomForestClassifier(n_estimators=5) # simple, fast nonlinear mode
clf = LGBMClassifier(n_estimators=10,num_leaves=3) # simple, fast nonlinear
sfs = SFS(clf,k_features=num_wrapper,forward=True,verbose=0,scoring=fdr,cv=2
# sfs = SFS(clf,k_features=1,forward=False,verbose=0,scoring=fdr,cv=4,n_jobs
sfs.fit(X,Y)
```

```
CPU times: user 16min 15s, sys: 1min 22s, total: 17min 38s
Wall time: 11min 55s
```

Out[20]:

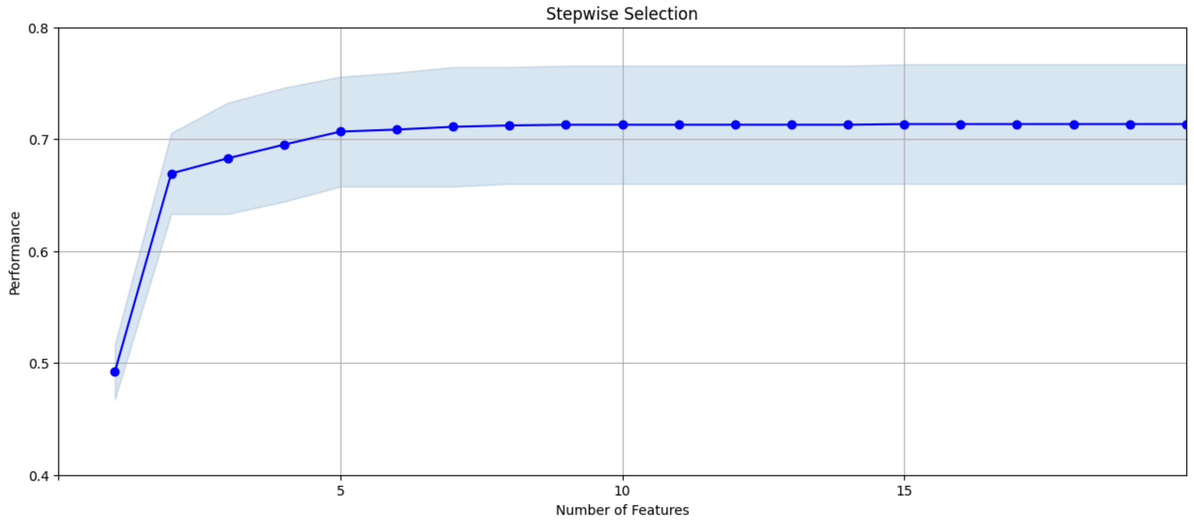
▸ **SequentialFeatureSelector**

▸ **estimator: LGBMClassifier**

▸ LGBMClassifier

In [21]:

```
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
fig1 = plot_sfs(sfs.get_metric_dict(),kind='std_dev', figsize=(15, 6))
# plt.xticks(np.arange(0, len(X.columns), step=5))
plt.xticks(np.arange(0, num_wrapper, step=5))
plt.yticks(np.arange(0,1,step=.1))
plt.ylim([.4,.8])
plt.xlim(0,num_wrapper)
plt.title('Stepwise Selection')
plt.grid()
plt.savefig('performance_nvars.png')
plt.show()
```



In [22]:

```
vars_FS = pd.DataFrame.from_dict(sfs.get_metric_dict()).T
```

In [23]:

```
ordered_vars_FS = vars_FS.copy()
for i in range(len(ordered_vars_FS)):
    ordered_vars_FS.loc[i+1,'add variables in this order'] = int(i+1)
    if i+1 == 1:
        ordered_vars_FS.loc[i+1,'variable name'] = (list(ordered_vars_FS.loc
    else:
        ordered_vars_FS.loc[i+1,'variable name'] = (list(set(ordered_vars_FS
# You might also need this following line. It converts a list to a string
#         ordered_vars_FS.loc[i+1,'variable name'] = ordered_vars_FS.loc[i+1
```

In [24]:

```
ordered_vars_FS
```

Out[24]:

	feature_idx	cv_scores	avg_score	feature_names
1	(25,)	[0.5165644171779141, 0.46748466257668714]	0.492025	(Cardnum_unique_count_for_card_state_1,)
2	(25, 129)	[0.7055214723926381, 0.6331288343558282]	0.669325	(Cardnum_unique_count_for_card_state_1, Card_M...
3	(25, 51, 129)	[0.7325153374233129, 0.6331288343558282]	0.682822	(Cardnum_unique_count_for_card_state_1, Cardnu...
4	(25, 51, 129, 151)	[0.7460122699386503, 0.6441717791411042]	0.695092	(Cardnum_unique_count_for_card_state_1, Cardnu...
5	(17, 25, 51, 129, 151)	[0.7558282208588957, 0.6576687116564417]	0.706748	(Card_dow_vdratio_0by60, Cardnum_unique_count_...
6	(17, 23, 25, 51, 129, 151)	[0.7595092024539877, 0.6576687116564417]	0.708589	(Card_dow_vdratio_0by60, Card_dow_vdratio_0by1...
7	(17, 23, 25, 51, 129, 151, 185)	[0.7644171779141105, 0.6576687116564417]	0.711043	(Card_dow_vdratio_0by60, Card_dow_vdratio_0by1...
8	(17, 23, 25, 51, 129, 130, 151, 185)	[0.7644171779141105, 0.660122699386503]	0.71227	(Card_dow_vdratio_0by60, Card_dow_vdratio_0by1...
9	(17, 23, 25, 51, 58, 129, 130, 151, 185)	[0.7656441717791411, 0.660122699386503]	0.712883	(Card_dow_vdratio_0by60, Card_dow_vdratio_0by1...
10	(17, 22, 23, 25, 51, 58, 129, 130, 151, 185)	[0.7656441717791411, 0.660122699386503]	0.712883	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
11	(17, 22, 23, 25, 30, 51, 58, 129, 130, 151, 185)	[0.7656441717791411, 0.660122699386503]	0.712883	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
12	(17, 22, 23, 25, 30, 31, 51, 58, 129, 130, 151...	[0.7656441717791411, 0.660122699386503]	0.712883	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
13	(17, 22, 23, 25, 30, 31, 32, 51, 58, 129, 130,...	[0.7656441717791411, 0.660122699386503]	0.712883	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
14	(17, 22, 23, 25, 30, 31, 32, 33, 51, 58, 129, ...	[0.7656441717791411, 0.660122699386503]	0.712883	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
15	(17, 22, 23, 25, 30, 31,	[0.7668711656441718, 0.660122699386503]	0.713497	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...

	feature_idx	cv_scores	avg_score	feature_names
	32, 33, 51, 58, 129, ...			
16	(17, 22, 23, 25, 30, 31, 32, 33, 35, 51, 58, 1...	[0.7668711656441718, 0.660122699386503]	0.713497	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
17	(17, 22, 23, 25, 30, 31, 32, 33, 35, 36, 51, 5...	[0.7668711656441718, 0.660122699386503]	0.713497	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
18	(17, 22, 23, 25, 30, 31, 32, 33, 35, 36, 41, 5...	[0.7668711656441718, 0.660122699386503]	0.713497	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
19	(17, 22, 23, 25, 30, 31, 32, 33, 35, 36, 41, 4...	[0.7668711656441718, 0.660122699386503]	0.713497	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...
20	(17, 22, 23, 25, 30, 31, 32, 33, 35, 36, 37, 4...	[0.7668711656441718, 0.660122699386503]	0.713497	(Card_dow_vdratio_0by60, Cardnum_actual/toal_0...

In [25]:

```
ordered_vars_FS.to_csv('Wrapper_selection_info.csv', index=False)
```

In [26]:

```
vars_keep = ordered_vars_FS['variable name']
vars_keep_list = ordered_vars_FS['variable name'].tolist()
vars_keep.to_csv('final_vars_list.csv',index=False)
vars_keep
```

Out[26]:

```
1         Cardnum_unique_count_for_card_state_1
2                  Card_Merchdesc_State_total_7
3                         Cardnum_count_1_by_30
4                                Cardnum_max_14
5                         Card_dow_vdratio_0by60
6                         Card_dow_vdratio_0by14
7                   Merchnum_desc_State_total_3
8                        Card_Merchdesc_total_7
9         Card_dow_unique_count_for_merch_zip_7
10                        Cardnum_actual/toal_0
11                         Card_dow_vdratio_0by7
12                          Cardnum_vdratio_1by7
13        Cardnum_unique_count_for_card_state_3
14          Cardnum_unique_count_for_card_zip_3
15                    Merchnum_desc_Zip_total_3
16          Cardnum_unique_count_for_Merchnum_3
17                        Cardnum_actual/toal_1
18        Cardnum_unique_count_for_card_state_7
19                         Cardnum_actual/max_0
20      Card_dow_unique_count_for_merch_state_1
Name: variable name, dtype: object
```

In [27]:

```python
filter_score.set_index('variable',drop=True,inplace=True)
filter_score = filter_score.iloc[1:,:]
filter_score
```

Out[27]:

	filter score
variable	
Cardnum_total_3	0.621932
Cardnum_total_1	0.619658
Cardnum_total_0	0.591318
Cardnum_count_1	0.567429
Cardnum_total_7	0.564245
...	...
Merchdesc_Zip_unique_count_for_Merchdesc_State_14	0.000038
merch_zip_unique_count_for_merch_state_14	0.000025
merch_zip_unique_count_for_merch_state_7	0.000013
Merchnum_desc_Zip_unique_count_for_Merchnum_desc_State_14	0.000013
card_merch_unique_count_for_Cardnum_1	0.000000

2634 rows × 1 columns

In [28]:

```python
vars_keep_sorted = pd.DataFrame(vars_keep_list)
vars_keep_sorted.columns=['variable']
vars_keep_sorted.set_index('variable',drop=True,inplace=True)
vars_keep_sorted.head()
```

Out[28]:

variable
Cardnum_unique_count_for_card_state_1
Card_Merchdesc_State_total_7
Cardnum_count_1_by_30
Cardnum_max_14
Card_dow_vdratio_0by60

In [29]:

```
vars_keep_sorted = pd.concat([vars_keep_sorted,filter_score],axis=1,join='ir
```

In [30]:

```
vars_keep_sorted.reset_index(inplace=True)
vars_keep_sorted.reset_index(inplace=True)
vars_keep_sorted['index'] = vars_keep_sorted['index'] + 1
vars_keep_sorted.rename(columns={'index':'wrapper order'},inplace=True)
vars_keep_sorted.to_csv('vars_keep_sorted.csv',index=False)
vars_keep_sorted
```

Out[30]:

	wrapper order	variable	filter score
0	1	Cardnum_unique_count_for_card_state_1	0.476067
1	2	Card_Merchdesc_State_total_7	0.324668
2	3	Cardnum_count_1_by_30	0.428229
3	4	Cardnum_max_14	0.318826
4	5	Card_dow_vdratio_0by60	0.486480
5	6	Card_dow_vdratio_0by14	0.479086
6	7	Merchnum_desc_State_total_3	0.308586
7	8	Card_Merchdesc_total_7	0.324631
8	9	Card_dow_unique_count_for_merch_zip_7	0.418943
9	10	Cardnum_actual/toal_0	0.479550
10	11	Card_dow_vdratio_0by7	0.467961
11	12	Cardnum_vdratio_1by7	0.466766
12	13	Cardnum_unique_count_for_card_state_3	0.466410
13	14	Cardnum_unique_count_for_card_zip_3	0.464323
14	15	Merchnum_desc_Zip_total_3	0.305656
15	16	Cardnum_unique_count_for_Merchnum_3	0.460748
16	17	Cardnum_actual/toal_1	0.459715
17	18	Cardnum_unique_count_for_card_state_7	0.445967
18	19	Cardnum_actual/max_0	0.445726
19	20	Card_dow_unique_count_for_merch_state_1	0.447357

In [31]:
```
vars_keep_list.append(index_name)
vars_keep_list.append(y_name)
vars_keep_list
```

Out[31]:
```
['Cardnum_unique_count_for_card_state_1',
 'Card_Merchdesc_State_total_7',
 'Cardnum_count_1_by_30',
 'Cardnum_max_14',
 'Card_dow_vdratio_0by60',
 'Card_dow_vdratio_0by14',
 'Merchnum_desc_State_total_3',
 'Card_Merchdesc_total_7',
 'Card_dow_unique_count_for_merch_zip_7',
 'Cardnum_actual/toal_0',
 'Card_dow_vdratio_0by7',
 'Cardnum_vdratio_1by7',
 'Cardnum_unique_count_for_card_state_3',
 'Cardnum_unique_count_for_card_zip_3',
 'Merchnum_desc_Zip_total_3',
 'Cardnum_unique_count_for_Merchnum_3',
 'Cardnum_actual/toal_1',
 'Cardnum_unique_count_for_card_state_7',
 'Cardnum_actual/max_0',
 'Card_dow_unique_count_for_merch_state_1',
 'Recnum',
 'Fraud']
```

In [32]:
```
filter_score
```

Out[32]:

	filter score
variable	
Cardnum_total_3	0.621932
Cardnum_total_1	0.619658
Cardnum_total_0	0.591318
Cardnum_count_1	0.567429
Cardnum_total_7	0.564245
...	...
Merchdesc_Zip_unique_count_for_Merchdesc_State_14	0.000038
merch_zip_unique_count_for_merch_state_14	0.000025
merch_zip_unique_count_for_merch_state_7	0.000013
Merchnum_desc_Zip_unique_count_for_Merchnum_desc_State_14	0.000013
card_merch_unique_count_for_Cardnum_1	0.000000

2634 rows × 1 columns

In [33]:
```
%%time
df = pd.read_csv(file_name)
```

```
df.shape
```

```
CPU times: user 15.8 s, sys: 2.08 s, total: 17.9 s
Wall time: 18.1 s
```

Out[33]:
```
(97496, 2635)
```

In [34]:
```
df_keep = df.filter(vars_keep_list, axis=1)
# df_keep = df[df.index.isin(vars_keep_list)]
print(df_keep.shape)
```

```
(97496, 22)
```

In [35]:
```
df_keep.to_csv('vars_final.csv',index=False)
```

In [36]:
```
print("duration: ", dt.datetime.now() - start_time)
```

```
duration:  0:13:10.886970
```

In [37]:
```
%pwd
```

Out[37]:
```
'/Users/stevecoggeshall/Documents/Teaching/Data sets/done/transactions'
```

In [ ]:
```

```