1. Separate training/testing first
2. Feature selection on training set
3. Feature selection: AVG ranking top 70, use:

from mlxtend.feature\_selection import SequentialFeatureSelector, use top 30

sbs = SequentialFeatureSelector(LogisticRegression(), k\_features=25, forward=True, floating=False, scoring = 'roc\_auc', cv=2)

Sfs.k\_feature\_names\_

score\_df = pd.DataFrame.from\_dict(sfs.get\_metric\_dict()).T

Score\_df

from sklearn.metrics import confusion\_matrix

Note:

1. Run all models on training, testing, and OOT separately
2. Calculate FDR at 0.03
3. Save combinations of parameters and respective FDR at 3% in Google Sheet

Parameter:

**Logistic Regression: Cheng**

logit\_model1 = sm.Logit(y\_train, x\_train)

result1 = logit\_model1.fit(maxiter=5000)

**Neural Net: Minglu**

from sklearn.neural\_network import MLPRegressor

NN = MLPRegressor(

hidden\_layer\_sizes=(5,), range(5,11,1), range(10,50,10)

activation='relu',

learning\_rate='adaptive',

max\_iter=10000,

alpha=.01 range(0.01,0.1,0.01)

epoch=20 range(20,50,10)

)

**Random Forest: Chengjun**

From sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(

n\_estimators = nf\_estimators, range(50,200,25)

min\_samples\_leaf, range(50,100,25) range(200,500,100)

criterion = ‘gini’

)

**Boosted Tree: Selene**

from sklearn.ensemble import GradientBoostingClassifier

params = {'n\_estimators':n, range(50,200,25)

'max\_depth':10,

'min\_samples\_leaf':50, range(50,100,25) range(200,500,100)

'learning\_rate':0.01, or 0.1

}

GB = GradientBoostingClassifier(\*\*params)

from xgboost import XGBClassifier

xgc = XGBClassifier(base\_score=0.5, booster='gbtree', colsample\_bylevel=1,

colsample\_bytree=1, gamma=0, learning\_rate=0.9, max\_delta\_step=0,

max\_depth = 7, min\_child\_weight=1, missing=None,

n\_estimators=100, range()

n\_jobs=1, nthread=None, objective='binary:logistic', random\_state=0,

reg\_alpha=0, reg\_lambda=1, scale\_pos\_weight=1, seed=None,

silent=True, subsample=1)

from lightgbm import LGBMClassifier

lgbm\_c = LGBMClassifier(boosting\_type='gbdt', class\_weight=None, colsample\_bytree=1.0,

learning\_rate=0.5, max\_depth=7, min\_child\_samples=20,

min\_child\_weight=0.001, min\_split\_gain=0.0, n\_estimators=100,

n\_jobs=-1, num\_leaves=500, objective='binary', random\_state=None,

reg\_alpha=0.0, reg\_lambda=0.0, silent=True, subsample=1.0,

subsample\_for\_bin=200000, subsample\_freq=0)

y\_pred\_tn = logreg.predict\_proba(X\_train)

y\_pred\_tn[:,1]

----code for calculating FDR (log regression)

#3% FDR for training

# calcuate number of records in a 3% bin

a = int(round(len(train)\*0.03))

# count how many fraud records are there in the training data

total\_fraud\_train = len(train[train['fraud\_label']==1])

train\_1 = train.copy()

train\_1['pred\_score'] = y\_pred\_tn[:,1]

# sort training records according to pred\_score, so we can get top 3%

train\_1 = train\_1.sort\_values(by='pred\_score', ascending = False)

# count in the top a records (3%) in the training data ranked by pred\_score,

# the proportion of fraud caught from all the fraud records in the training set

print("3% FDR for training is ", len(train\_1[:a][train\_1['fraud\_label']==1])/total\_fraud\_train)

------variables selected

**Chao’s run of 20 variables with sfs, training 0.50, testing 0.48, oot 0.48**

['addr\_lag30\_count', 'addr\_#days\_since', 'address\_#days\_since', 'address\_lag14\_count', 'address\_lag7\_count', 'addr\_lag3\_count', 'addr\_lag1\_count', 'addr-homephone\_lag30\_count', 'ssn-dob\_lag30\_count', 'ssn-name-dob\_lag30\_count', 'addr-homephone\_#days\_since', 'ssn-name\_lag30\_count', 'ssn-firstname\_#days\_since', 'ssn-name\_#days\_since', 'ssn-name-dob\_lag14\_count', 'name-dob\_lag7\_count', 'name\_lag14\_count', 'name\_#days\_since', 'ssn\_lag7\_count', 'ssn-lastname\_lag7\_count']

Cheng’s rfecv of 20 variables

['addr-homephone\_#days\_since', 'addr-homephone\_lag14\_count', 'addr-homephone\_lag30\_count', 'addr-homephone\_lag7\_count', 'addr\_#days\_since', 'addr\_lag14\_count', 'addr\_lag1\_count', 'addr\_lag1\_lag14\_avg', 'addr\_lag30\_count', 'addr\_lag3\_count', 'addr\_lag7\_count', 'address\_#days\_since', 'address\_lag14\_count', 'address\_lag1\_count', 'address\_lag1\_lag14\_avg', 'address\_lag30\_count', 'address\_lag3\_count', 'address\_lag7\_count', 'homephone\_lag7\_count', 'name-dob\_#days\_since']