1. Data cleaning
2. Variable creation, Z scale, get rid of first two weeks of data (remove Fedex only for Benford’s law)
3. Separate modeling data & oot (Nov-Dec)
4. Use modeling data to do feature selection
5. Use modeling data to try building models (log reg, random forest, boosted tree, neural net, decision tree), cross validation, each version run ~10 times and take average (FDR 3%), select favorite model type; Take the favorite model type and tune hyperparameters and select favorite version --- freeze model and its parameter (4)
6. Split modeling data into training and testing (8:2) data, run the favorite version of the favorite model with training and testing separately and build training and testing FDR tables (Chengjun)
7. Rebuild the final model with modeling data, without cross validation, and run the model on OOT, and build OOT FDR table (Chengjun)
8. Build Fraud Savings Plot to suggest cutoff (Ocean)
9. Plot two fraud examples to demonstrate how our model catches fraud behavior over time (Ocean)

Task:

•**Cover Page**: Title, project team number, names, date.

•**Table of Contents**. Make sure all pages are numbered.

•**Executive Summary** (few paragraphs). High level summary of project and results.

•**Description of Data** (few pages). Include most important (not all) distributions/histograms, put the full DQR in appendix.

•**Data Cleaning** (few paragraphs or pages). How did you handle exclusions, outliers, missing fields, frivolous field values.

•**Candidate Variables** Before calculating variables, Z-scale. (few pages). Describe in complete detail the formulas and logic for creating all the candidate model variables. Include a list of all variables; if it’s more than about 5 pages it could be in an appendix. (entity, different types of variables)

•**Feature Selection Process** (few pages). Describe what you did to select features. List of final variables.

•**Model Algorithms** (a few pages). Short explanation of all algorithms tried. Table of high level results for each algorithm (FDR for training, testing, oot).

•**Results** (few pages). For your final algorithm and parameter selection, show the three tables for training, testing, oot populations.

* Additional plots

•**Conclusions** (few paragraphs). Summarize everything you did and what else you might do with more time.

•**Appendix.** DQR

Chao + proofread

Minglu

Cheng + format

Selene

Cheng: log reg

Selene: boosted tree

MInglu: neural net

Chengjun: random forest

Ocean + table

EOD

4/7 Variable created

4/8 Feature selected (keep to 30 with rank ordering)

4/10 Favorite model and version selected

4/11 meeting (6p-8p)

4/12 Finish running on training, testing, OOT with the correct procedure;

4/12 evening - 4/13 morning run the best model

4/13 afternoon - 4/14 evening final graphs and tables

4/14 Finalize project process and ppt first draft ready; night meeting to go over ppt (7p-9p)

4/15 night rehearse (10p-11:30p)

4/16 Presentation

4/26 Report due

Train different models, test:

1. Different numbers of features
2. Different parameter combinations (same as last time)
3. Calculate FDR 3%, get the cross validated average FDR 3%

For reference of cross validation:

## Train with cross validation

for key in dTrain.keys():

# Try different estimators and learning rates

# With each train-test split, run 20 versions of model

for estimator in [200,250,300,350]:

for rate in [0.01, 0.05, 0.1, 0.15, 0.02]:

# Split data into 10 folds, therefore run each version of model 10 times

kfolds = KFold(n\_splits=10, shuffle=True, random\_state=1)

split=0

rsq\_train\_total=0

rsq\_test\_total=0

for train\_index, test\_index in kfolds.split(dTrain[key]):

Y\_train = dTrain[key].iloc[train\_index,-1]

Y\_test = dTrain[key].iloc[test\_index,-1]

X\_train = dTrain[key].iloc[train\_index,0:-1]

X\_test = dTrain[key].iloc[test\_index,0:-1]

params = {'n\_estimators':estimator, 'max\_depth':10, 'min\_samples\_split':10, 'learning\_rate':rate,

'loss':'ls'}

GB = GradientBoostingRegressor(\*\*params)

GB.fit(X\_train,Y\_train)

predict\_GB\_train = GB.predict(X\_train)

predict\_GB\_test = GB.predict(X\_test)

rsq\_train = r2\_score(Y\_train,predict\_GB\_train)

rsq\_test = r2\_score(Y\_test,predict\_GB\_test)

print(f'Split {split}:\nTraining R-squared: {rsq\_train}\nTesting R-squared: {rsq\_test}')

print()

rsq\_train\_total += rsq\_train

rsq\_test\_total += rsq\_test

split+=1

rsq\_train\_avg = rsq\_train\_total/10

rsq\_test\_avg = rsq\_test\_total/10

print(f'For Y {key}, estimator {estimator}, rate {rate}:\nAverage training R-squared: {rsq\_train\_avg}\nAverage testing R-squared: {rsq\_test\_avg}\n')

print()

print()