**Agent Production Model**

**Models:**

All Agent Production Model, two versions, one version is RF ML in H20, MAE around 440 for PIFSUM prediction, which is translated at 30% MAPE; the other version is simple 3 Year Average, MAE around 140 for PIFSUM prediction, which is translated at 9.5%; PREMSUM has similar performances in the two models.

  NEW Agent Production Model, RF ML in H2O, MAE around 410 for PIFSUM prediction, which is translated at 28% MAPE.

Rationality: established agents settled and performance remains

**Predictors:**

All Agent Model: Each agent’s top 10 pol source zip codes aggregated demographics and business attributes (previous year PIFSUM,PREMSUM included)

New Agent Model: Each agent’s (located ZIP’s) top 10 pol source zip codes aggregated demographics and business attributes + Tenure(months) + Assignment Type + Assigned A&F PREM + Assigned A&F PIF

**Raw Data:**

Agents and Pol data with pol effective and expiration dates included:  agents\_and\_policies\_auto.csv + Demographics data+ auto\_agts\_pif\_prem\_sum.csv + Agents\_Sep\_2016.csv+ 2010-2016New\_Agent\_Assignments.xlsx

**Feature Engineering:**

**See the Scripts descriptions**

**Scripts:**

**radius\_dataPrep.py**

INPUTS: datapull/agents\_and\_policies\_auto.csv which has agents and their pol details

    Agents\_Sep\_2016.csv which has all the agents loc, and other metadata of SF as of 2016/09/

OUTPUTS:radiusAnalysis/agtspols\_processed.csv which has distance between agent office to all of the pols location calculated in UTM coordinate

Maybe rerun this part using agents\_and\_policies\_auto\_new.csv

**radius\_production.py**

INPUTS: radiusAnalysis/agtspols\_processed.csv

OUTPUTS: radiusAnalysis/radiusZips which has the top 10 pol source zips for each agent

this script takes long to run, use nohup python radius\_production.py & to run it

in the edge node back end

**radius\_features.py**

INPUTS: radiusAnalysis/radiusZips

Demographics/zpmerge\_2010.csv

Demographics/zpmerge\_2011.csv

Demographics/zpmerge\_2012.csv

Demographics/zpmerge\_2013.csv

Demographics/zpmerge\_2014.csv

Demographics/zpmerge\_2015.csv

OUTPUTS:

top10ZipFeatures/top10ZipFeatures\_2010.csv

top10ZipFeatures/top10ZipFeatures\_2011.csv

top10ZipFeatures/top10ZipFeatures\_2012.csv

top10ZipFeatures/top10ZipFeatures\_2013.csv

top10ZipFeatures/top10ZipFeatures\_2014.csv

top10ZipFeatures/top10ZipFeatures\_2015.csv

this script now needs manually define input file and save output, needs

some small automation

**MISSING SCRIPT:**

INPUTS: datapull/auto\_agts\_pif\_prem\_sum.csv

OUTPUS: datapull/targets2010.csv

    datapull/targets2011.csv

datapull/targets2012.csv

datapull/targets2013.csv

datapull/targets2014.csv

datapull/targets2015.csv

the input data was not effective expiration date included, and thus probably count

pif sum and prem sum including cancelled pols

**tplusxy\_job\_run.py**:

INPUTS:

datapull/targets2010.csv

    datapull/targets2011.csv

datapull/targets2012.csv

datapull/targets2013.csv

datapull/targets2014.csv

datapull/targets2015.csv

top10ZipFeatures/top10ZipFeatures\_2010.csv

top10ZipFeatures/top10ZipFeatures\_2011.csv

top10ZipFeatures/top10ZipFeatures\_2012.csv

top10ZipFeatures/top10ZipFeatures\_2013.csv

top10ZipFeatures/top10ZipFeatures\_2014.csv

top10ZipFeatures/top10ZipFeatures\_2015.csv

INTERMEDIA OUTPUTS:

datapull/targets201X\_features201Y.csv where X> Y

with top 10 zip attributed separated like  curr\_pop\_zip0,curr\_pop\_zip1

datapull/targets201X\_aggfeatures201Y.csv where X> Y

with top 10 zip attributes aggregated into 1 like curr\_pop

OUTPUTS:

datapull/tplus1XY.csv: 1 year forward XY df for training aggregated

datapull/tplus2XY.csv: 2 year forward XY df for training aggregated

datapull/tplus3XY.csv: 3 year forward XY df for training aggregated

datapull/tplus4XY.csv: 4 year forward XY df for training aggregated

datapull/tplus1XY\_nonagged.csv: 1 year forward XY df for training

datapull/tplus2XY\_nonagged.csv: 2 year forward XY df for training

datapull/tplus3XY\_nonagged.csv: 3 year forward XY df for training

datapull/tplus4XY\_nonagged.csv: 4 year forward XY df for training

this script takes long to run, suggest to run in background, manually change the line a bit we output non aggregated version

**agtProdRes\_analysis.py:**

INPUTS:

datapull/tplus1XY.csv

datapull/tplus2XY.csv

datapull/tplus3XY.csv

datapull/tplus4XY.csv

2010-2016New\_Agent\_Assignments.xlsx

OUTPUTS:

datapull/tplus1XY\_newAgents.csv

datapull/tplus2XY\_newAgents.csv

datapull/tplus3XY\_newAgents.csv

datapull/tplus4XY\_newAgents.csv

these are training xy data frame for new agents model

**3yearSmoothAPM.py:**

INPUTS:

datapull/targets2011.csv

datapull/targets2012.csv

datapull/targets2013.csv

datapull/targets2014.csv

datapull/targets2015.csv

  OUTPUTS:     mean\_absolute\_error(df\_final['3yrAvg'], df\_final.pifsum2015)

this script runs a simple 3-yr smooth and use that to predict 2015 agent performance

**agtProductionModel.ipynb:**

INPUTS: datapull/tplus1XY.csv

datapull/tplus2XY.csv

datapull/tplus3XY.csv

datapull/tplus4XY.csv

datapull/tplus1XY\_newAgents.csv

datapull/tplus2XY\_newAgents.csv

datapull/tplus3XY\_newAgents.csv

datapull/tplus4XY\_newAgents.csv

OUTPUTS: all agents model outputs:

rf\_prem1,perf\_prem1,testRes\_prem1

rf\_prem2,perf\_prem2,testRes\_prem2

rf\_prem3,perf\_prem3,testRes\_prem3

rf\_prem4,perf\_prem4,testRes\_prem4

rf\_pif1,perf\_pif1,testRes\_pif1

rf\_pif2,perf\_pif2,testRes\_pif2

rf\_pif3,perf\_pif3,testRes\_pif3

rf\_pif4,perf\_pif4,testRes\_pif4

new agents model outputs:

rf\_pif1\_newAgents,perf\_pif1\_newAgents,testRes\_pif1\_newAgents

rf\_pif2\_newAgents,perf\_pif2\_newAgents,testRes\_pif2\_newAgents

rf\_pif3\_newAgents,perf\_pif3\_newAgents,testRes\_pif3\_newAgents

rf\_pif4\_newAgents,perf\_pif4\_newAgents,testRes\_pif4\_newAgents

rf\_prem1\_newAgents,perf\_prem1\_newAgents,testRes\_prem1\_newAgents

rf\_prem2\_newAgents,perf\_prem2\_newAgents,testRes\_prem2\_newAgents

rf\_prem3\_newAgents,perf\_prem3\_newAgents,testRes\_prem3\_newAgents

rf\_prem4\_newAgents,perf\_prem4\_newAgents,testRes\_prem4\_newAgents

**radiusFeaturesZIP.py**

INPUTS: top10ZipFeatures/top10ZipFeatures\_2015->2010 which has the top 10 pol source zips’ demographics features for each agent

OUTPUTS: top10ZipFeatures/byZIPtop10ZipFeatures\_2010🡪2015.csv produce top 10 pol source zips’ features for each zip instead

top10ZipFeatures/byZIPtop10ZipFeaturesAgg\_2010.csv' produce top 10 pol source zips’ aggregated version features for each zip instead

top10ZipFeatures/top10zipPerzip2010.csv: each zip codes top 10 neibor zip

**prepData4\_getMostSimZip.py**

INPUTS:

# get each zip's 2 encoded

zipsim\_df = pd.read\_csv('zipSimilarity/zipsimilarity\_tsne.csv')

# Read zip merged file with all demographics features

zipdf = pd.read\_csv('zipmerged2010-2015.csv')

# Read zipinfo csv which has the state, county info per zip

zipinfo = pd.read\_csv('us\_postal\_codes.csv')

OUTPUTS: ' zipSimilarity/allzips\_sim\_info.csv' which has all 40k zips 2 digit encoded representation, and completed info including size(city,rural,urban)

**getMostSimZip.py**

INPUTS: this function will use zipSimilarity/allzips\_sim\_info.csv and top10ZipFeatures/byZIPtop10ZipFeaturesAgg\_2010->5.csv

OUTPUTS: function getMostSimZip(target\_ZIP,targetdframe,candidframe) to return the most similar zip from an existing state farm zip to any non state farm existence zip in an approximate 75 miles radius range and with zip size matched: city zip match to city zip, etc

**ZIP Production Model**

**Raw Data:**

Demographics data+ auto\_zips\_pif\_prem\_sum.csv(from pif\_prem\_sum\_autozips query)

**Feature Engineering:**

**radius\_production.py**

INPUTS: radiusAnalysis/agtspols\_processed.csv

: radiusAnalysis/radiusZips which has the top 10 pol source zips for each agent

OUTPUTS: produce top 10 pol source zips for each zip instead