C:\Python27\python.exe -u C:\Program Files (x86)\JetBrains\PyCharm 3.0\helpers\pydev\pydevconsole.py 49180 49181

PyDev console: using IPython 0.13.2

import sys; print('Python %s on %s' % (sys.version, sys.platform))

Python 2.7.6 (default, Nov 10 2013, 19:24:24) [MSC v.1500 64 bit (AMD64)] on win32

sys.path.extend(['C:\\Users\\bgregory\\MyDocs\\Dropbox\\DataScience\\Kaggle\\Hackathon-311'])

>>> ['description\_length', 'neighborhood', 'source', 'weekend\_fg']

Out[3]: ['description\_length', 'neighborhood', 'source', 'weekend\_fg']

>>> .3908

Out[4]: 0.3908

>>>

import munge

import train

import data\_io

import features

import ensembles

from scipy import sparse

from sklearn.externals import joblib

import sys

import csv

import json

import argparse

import numpy as np

import pandas as pd

from sklearn.datasets import dump\_svmlight\_file

from sklearn import (metrics, cross\_validation, linear\_model, ensemble, tree, preprocessing, svm, neighbors, gaussian\_process, naive\_bayes, neural\_network, pipeline, lda)

class Model(object):

#Class attributes:

#model\_name = descriptive name of model

#target = target variable for this model

#segment = segment of data for this model

#classifier\_name = descriptive name of classifier to use

#classifer = the sklearn classifier to use

#features = a dictionary of features to use in model. The key is feature name, the value is an binary array with

# actual values of feature for train set and actual values of feature for test set.

#postprocess\_scalar = the scalar to apply to all predictions generated from model

def \_\_init\_\_(self, model\_name, target, segment, classifier\_name, features, postprocess\_scalar):

self.model\_name = model\_name

self.target = target

self.segment = segment

self.classifier\_name = classifier\_name

self.set\_classifier(classifier\_name)

self.set\_features(features)

self.postprocess\_scalar = round(np.float32(postprocess\_scalar), 4)

def set\_classifier(self, classifier):

if classifier[:3] == 'GBM':

self.classifier = ensemble.GradientBoostingRegressor(max\_depth=1, random\_state=888, loss='ls')

self.classifier.n\_estimators = np.float32(classifier[4:classifier[4:].find("\_")+4])

self.classifier.learning\_rate = round(np.float32(classifier[8:]), 4)

if classifier[:3] == 'SGD':

self.classifier = linear\_model.SGDRegressor(shuffle=True)

self.classifier.n\_iter = np.float32(classifier[4:classifier[4:].find("\_")+4])

self.classifier.alpha = round(np.float32(classifier[8:]), 4)

self.classifier.random\_state = 8888

self.classifier.alpha = .0001

if classifier[:3] == 'ADA':

self.classifier = ensemble.AdaBoostRegressor()

self.classifier.n\_iter = np.float32(classifier[4:classifier[4:].find("\_")+4])

self.classifier.alpha = round(np.float32(classifier[8:]), 4)

self.classifier.random\_state = 8888

if classifier[:7] == 'LASSOCV':

self.classifier = linear\_model.LassoCV()

self.classifier.cv = np.int(classifier[8:])

def set\_features(self, features):

#Accepts a list of features

self.features = dict((feature,['','']) for feature in features)

def predict(self,dfTrn,dfTest):

#Segment the train and test data to match current model

dfTrn, dfTest = munge.segment\_data(dfTrn, dfTest, self.segment)

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn, dfTest, self.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn, dfTest, self.features)

#Make predictions

mtxTrn, mtxTest, mtxTrnTarget, mtxTestTarget = train.combine\_features(self, dfTrn, dfTest)

train.predict(mtxTrn,mtxTrnTarget.ravel(),mtxTest,dfTest,self)

#Store predictions in dataframe as class attribute

self.dfPredictions = dfTest.ix[:,['id',self.target]]

#Export predictions (optional)

if settings["export\_predictions\_each\_model"] == 'true':

data\_io.save\_predictions(dfTest,self,'test')

#def main():

#-------------Load Settings/Models----------------------#

#Get environment settings

settings = json.loads(open("SETTINGS.json").read())

#Get model settings

model\_settings = json.loads(open("MODELS.json").read())

#TODO: If settings are use cached model, then skip initializing and load models instead

#Initialize list of model classes

models = []

for model in model\_settings:

new\_model = Model(model\_name=model,target=model\_settings[model]['target'],segment=model\_settings[model]['segment'],

classifier\_name=model\_settings[model]['classifier\_name'],features=model\_settings[model]['features'],

postprocess\_scalar=model\_settings[model]['postprocess\_scalar'])

models.append(new\_model)

#Manually add a new model:

#models.append(Model(model\_name="Weak Descr Len",target="num\_views",segment="Richmond",

# classifier\_name="RF",features=["description\_length"],postprocess\_scalar=1))

#-------Data Loading/Cleaning/Munging------------#

#Load the data

dfTrn = data\_io.load\_flatfile\_to\_df(settings['filename\_train'])

dfTest = data\_io.load\_flatfile\_to\_df(settings['filename\_test'])

dfCV = data\_io.load\_flatfile\_to\_df('Data/CV.csv')

#Clean/Munge the data

dfTrn = munge.clean(dfTrn)

dfTest = munge.clean(dfTest)

#-------Feature creation-------------------------#

#Add all currently used hand crafted features to dataframes

features.add(dfTrn)

features.add(dfTest)

#---------Data slicing/parsing--------------------------#

#Temporal split of data for CV

if settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'april':

dfTrnCV, dfTestCV = munge.temporal\_split(dfTrn, (2013, 04, 1))

elif settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'march':

#take an addtional week from February b/c of lack of remote\_api source issues in March

dfTrnCV, dfTestCV = munge.temporal\_split(dfTrn, (2013, 02, 21))

elif settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'list\_split':

#load stored list of data points and use those for CV

dfCVlist = pd.DataFrame({'id': data\_io.load\_cached\_object("Cache/cv\_issue\_ids.pkl"), 'dummy': 0})

dfTrnCV, dfTestCV = munge.list\_split(dfTrn, dfCVlist)

>>> features.sub\_feature(dfTrnCV,'zipcode','neighborhood',["Richmond","Oakland","Manchester","Chicago","New Haven"])

features.sub\_feature(dfTestCV,'zipcode','neighborhood',["Richmond","Oakland","Manchester","Chicago","New Haven"])

features.knn\_thresholding(dfTrnCV,'neighborhood',5)

features.knn\_thresholding(dfTestCV,'neighborhood',5)

Out[51]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 140975 entries, 0 to 223127

Data columns (total 33 columns):

id 140975 non-null values

latitude 140975 non-null values

longitude 140975 non-null values

summary 140975 non-null values

description 140975 non-null values

num\_votes 140975 non-null values

num\_comments 140975 non-null values

num\_views 140975 non-null values

source 140975 non-null values

created\_time 140975 non-null values

tag\_type 140975 non-null values

zipcode 140975 non-null values

street 140975 non-null values

city 140975 non-null values

neighborhood 140975 non-null values

est\_pop 140975 non-null values

avg\_income 140975 non-null values

tot\_income 140975 non-null values

created\_date 140975 non-null values

month 140975 non-null values

created\_time\_hrs 140975 non-null values

created\_time\_range 140975 non-null values

age 140975 non-null values

long\_rnd2 140975 non-null values

lat\_rnd2 140975 non-null values

long\_lat\_rnd2 140975 non-null values

dayofweek 140975 non-null values

description\_length 140975 non-null values

description\_length\_log 140975 non-null values

description\_fg 140975 non-null values

tagtype\_fg 140975 non-null values

weekend\_fg 140975 non-null values

nbr\_longlat 140975 non-null values

dtypes: float64(8), int64(8), object(17)

Out[52]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 26769 entries, 2 to 223124

Data columns (total 33 columns):

id 26769 non-null values

latitude 26769 non-null values

longitude 26769 non-null values

summary 26769 non-null values

description 26769 non-null values

num\_votes 26769 non-null values

num\_comments 26769 non-null values

num\_views 26769 non-null values

source 26769 non-null values

created\_time 26769 non-null values

tag\_type 26769 non-null values

zipcode 26769 non-null values

street 26769 non-null values

city 26769 non-null values

neighborhood 26769 non-null values

est\_pop 26769 non-null values

avg\_income 26769 non-null values

tot\_income 26769 non-null values

created\_date 26769 non-null values

month 26769 non-null values

created\_time\_hrs 26769 non-null values

created\_time\_range 26769 non-null values

age 26769 non-null values

long\_rnd2 26769 non-null values

lat\_rnd2 26769 non-null values

long\_lat\_rnd2 26769 non-null values

dayofweek 26769 non-null values

description\_length 26769 non-null values

description\_length\_log 26769 non-null values

description\_fg 26769 non-null values

tagtype\_fg 26769 non-null values

weekend\_fg 26769 non-null values

nbr\_longlat 26769 non-null values

dtypes: float64(8), int64(8), object(17)

>>> dfCVvotes = dfCV[dfCV['act\_status'] !='Closed']

dfCVvotes = dfCVvotes[dfCVvotes['act\_status'] !='Archived'].dropna()

>>> models

Out[55]:

[<\_\_main\_\_.Model at 0x12316128>,

<\_\_main\_\_.Model at 0x12316160>,

<\_\_main\_\_.Model at 0x12316208>,

<\_\_main\_\_.Model at 0x123163c8>,

<\_\_main\_\_.Model at 0x12316438>,

<\_\_main\_\_.Model at 0x123164a8>,

<\_\_main\_\_.Model at 0x12316518>,

<\_\_main\_\_.Model at 0x12316588>,

<\_\_main\_\_.Model at 0x123165f8>,

<\_\_main\_\_.Model at 0x123166d8>,

<\_\_main\_\_.Model at 0x12316748>,

<\_\_main\_\_.Model at 0x123167b8>,

<\_\_main\_\_.Model at 0x12316828>]

>>> models[13]

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-56-2f0e3da5ddcb>", line 1, in <module>

models[13]

IndexError: list index out of range

>>> models[12]

Out[57]: <\_\_main\_\_.Model at 0x12316828>

>>> models.append(Model(model\_name="Rich Votes",target="num\_votes",segment="Richmond",

classifier\_name="RF",features=['description\_length', 'neighborhood', 'source', 'weekend\_fg'],postprocess\_scalar=1))

>>> models[13]

Out[58]: <\_\_main\_\_.Model at 0x2a562be0>

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_votes

FEATURES: ['description\_length', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

C:\Python27\lib\site-packages\sklearn\utils\validation.py:278: UserWarning: StandardScaler assumes floating point values as input, got int64

"got %s" % (estimator, X.dtype))

Temporal CV started at: 11-26-13 16:38

==============================================================================================================

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-59-db0720d2d325>", line 20, in <module>

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\train.py", line 23, in cross\_validate

cv\_preds = cross\_validate\_temporal(mtxTrn,mtxTest,mtxTrnTarget.ravel(),mtxTestTarget.ravel(),model)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\train.py", line 123, in cross\_validate\_temporal

model.classifier.fit(train\_cv, y\_target)

AttributeError: 'Model' object has no attribute 'classifier'

>>> models[13].classifier = linear\_model.SGDRegressor(alpha=0.001, n\_iter=800,shuffle=True); clf\_name='SGD\_001\_800'

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_votes

FEATURES: ['description\_length', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 16:38

==============================================================================================================

Error Measure: 1.77838242893

Prediction mean: 2.481658

Prediction std dev: 0.309772

Prediction max/min: 6.357808/1.761062

==============================================================================================================

Temporal CV completed at: 11-26-13 16:38. Total runtime: 0:00:02.660000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1638--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 16:38

==============================================================================================================

Prediction mean: 2.469114

Prediction std dev: 0.293570

Prediction max/min: 8.323612/1.557507

==============================================================================================================

Predictions completed at: 11-26-13 16:39. Total runtime: 0:00:02.759000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1639--Rich Votes\_RF\_test.csv

>>> for target in ["num\_views","num\_votes","num\_comments"]:

dfCV[target] = np.log(dfCV[target] + 1)

>>> dfCVvotes = dfCV[dfCV['act\_status'] !='Closed']

dfCVvotes = dfCVvotes[dfCVvotes['act\_status'] !='Archived'].dropna()

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_votes

FEATURES: ['description\_length', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 16:39

==============================================================================================================

Error Measure: 0.378999581505

Prediction mean: 2.475378

Prediction std dev: 0.316157

Prediction max/min: 6.650589/1.735736

==============================================================================================================

Temporal CV completed at: 11-26-13 16:39. Total runtime: 0:00:02.619000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1639--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features(['description\_length', 'neighborhood', 'source', 'weekend\_fg','summary\_descr\_tfidf\_word'])

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_votes

FEATURES: ['description\_length', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 16:40

==============================================================================================================

Error Measure: 0.364798007874

Prediction mean: 2.508745

Prediction std dev: 0.414064

Prediction max/min: 5.765498/1.772178

==============================================================================================================

Temporal CV completed at: 11-26-13 16:40. Total runtime: 0:00:02.918000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1640--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> del models[13].dfPredictions

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_votes

FEATURES: ['description\_length', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 16:41

==============================================================================================================

Error Measure: 0.363555438412

Prediction mean: 2.500016

Prediction std dev: 0.413255

Prediction max/min: 5.735911/1.765570

==============================================================================================================

Temporal CV completed at: 11-26-13 16:41. Total runtime: 0:00:03.028000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1641--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 16:41

==============================================================================================================

Prediction mean: 2.453076

Prediction std dev: 0.353068

Prediction max/min: 7.320851/1.699489

==============================================================================================================

Predictions completed at: 11-26-13 16:41. Total runtime: 0:00:03.126000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1641--Rich Votes\_RF\_test.csv

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_votes"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/rich\_votes\_test.csv")

#munge

for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Richmond"].reset\_index())

dfCVvotes = dfCV[dfCV['act\_status'] !='Closed']

dfCVvotes = dfCVvotes[dfCVvotes['act\_status'] !='Archived'].dropna()

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner")

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

Model loaded into index 0

Model loaded into index 1

Model loaded into index 2

True value for target variables successfully loaded into self.df\_true\_segment

>>> weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_votes":.1,"num\_comments":.3}]

File "<ipython-input-87-7bc908339ddd>", line 1

weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_votes":.1,"num\_comments":.3}]

^

IndentationError: unexpected indent

If you want to paste code into IPython, try the %paste and %cpaste magic functions.

>>> weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_votes":.1,"num\_comments":.3}]

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(3):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(3):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-89-cb4ad3ddff06>", line 2, in <module>

clf.fit\_intercept=False

NameError: name 'clf' is not defined

>>> clf=linear\_model.LinearRegression();clf\_name = 'LinReg'

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(3):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(3):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[ 0.19881553 -0.09046749 0.01116892]]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_votes: 0.386761

Total RMSLE score: 0.386761

>>> weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":0},{"num\_views":.3,"num\_votes":.55,"num\_comments":0},{"num\_views":.4,"num\_votes":.1,"num\_comments":1}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_votes: 1.293009

Total RMSLE score: 1.293009

>>> dfCVvotes

Out[101]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 5682 entries, 3 to 25144

Data columns (total 6 columns):

id 5682 non-null values

segment 5682 non-null values

num\_views 5682 non-null values

num\_votes 5682 non-null values

num\_comments 5682 non-null values

act\_status 5682 non-null values

dtypes: float64(3), int64(1), object(2)

>>> dfSegTest

C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

Out[102]:

<class 'pandas.core.frame.DataFrame'>

Int64Index: 15409 entries, 0 to 15408

Data columns (total 6 columns):

id 15409 non-null values

segment 15409 non-null values

num\_views 15409 non-null values

num\_votes 15409 non-null values

num\_comments 15409 non-null values

act\_status 15408 non-null values

dtypes: float64(3), int64(1), object(2)

>>> dfSegTest = dfCVvotes.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner")

>>> dfSegTest

Out[104]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1878 entries, 0 to 1877

Data columns (total 6 columns):

id 1878 non-null values

segment 1878 non-null values

num\_views 1878 non-null values

num\_votes 1878 non-null values

num\_comments 1878 non-null values

act\_status 1878 non-null values

dtypes: float64(3), int64(1), object(2)

>>>

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

#Set weights

#Remote\_API: weights = [{"num\_views":.16,"num\_votes":.3,"num\_comments":.9},{"num\_views":.84,"num\_votes":.7,"num\_comments":.1}]

#Richmond: weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_comments":.3}]

#Oakland weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.7},{"num\_views":.8,"num\_votes":.9,"num\_comments":.3}]

True value for target variables successfully loaded into self.df\_true\_segment

>>>

>>>

>>> weights

Out[117]:

[{'num\_comments': 0, 'num\_views': 0.7, 'num\_votes': 0.45},

{'num\_comments': 0, 'num\_views': 0.3, 'num\_votes': 0.55},

{'num\_comments': 1, 'num\_views': 0.4, 'num\_votes': 0.1}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-121-b79dd3fe69eb>", line 1, in <module>

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ensembles.py", line 131, in score\_rmsle

target\_score = ml\_metrics.rmsle(df\_true[target], df[target])

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ml\_metrics.py", line 155, in rmsle

return np.sqrt(msle(actual, predicted))

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ml\_metrics.py", line 111, in msle

return np.mean(sle(actual, predicted))

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ml\_metrics.py", line 200, in sle

np.log(np.array(predicted)+1), 2))

ValueError: operands could not be broadcast together with shapes (1878) (15409)

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_votes"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/rich\_votes\_test.csv")

#munge

for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCVvotes.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Richmond"].reset\_index())

dfSegTest = dfCVvotes.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner")

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

Model loaded into index 0

Model loaded into index 1

Model loaded into index 2

True value for target variables successfully loaded into self.df\_true\_segment

>>>

>>>

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_votes: 0.431121

Total RMSLE score: 0.431121

>>> weights

Out[141]:

[{'num\_comments': 0, 'num\_views': 0.7, 'num\_votes': 0.45},

{'num\_comments': 0, 'num\_views': 0.3, 'num\_votes': 0.55},

{'num\_comments': 1, 'num\_views': 0.4, 'num\_votes': 0.1}]

>>> weights = [{"num\_views":.7,"num\_votes":.0,"num\_comments":.7},{"num\_views":.3,"num\_votes":0,"num\_comments":.3},{"num\_views":1,"num\_comments":.3}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-143-6f9bcfa1fb2c>", line 1, in <module>

ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ensembles.py", line 173, in create\_ensemble\_segment

self.df\_ensemble\_segment[target] += self.sub\_models\_segment[submodel][target] \* weights[submodel][target]

KeyError: 'num\_votes'

RMSLE score for num\_votes: 1.138075

Total RMSLE score: 1.138075

>>> weights = [{"num\_views":.7,"num\_votes":.0,"num\_comments":.7},{"num\_views":.3,"num\_votes":0,"num\_comments":.3},{"num\_views":1,"num\_comments":.3,"num\_votes":1}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_votes: 0.363168

Total RMSLE score: 0.363168

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(3):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(3):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[-0.03754983 1.70501294 -0.81353959]]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_votes: 0.303983

Total RMSLE score: 0.303983

>>> weights = [{"num\_views":.7,"num\_votes":.0,"num\_comments":.7},{"num\_views":.3,"num\_votes":1,"num\_comments":.3},{"num\_views":1,"num\_comments":.3,"num\_votes":0}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_votes: 0.353077

Total RMSLE score: 0.353077

>>> weights = [{"num\_views":.7,"num\_votes":.0,"num\_comments":.7},{"num\_views":.3,"num\_votes":1.3,"num\_comments":.3},{"num\_views":1,"num\_comments":.3,"num\_votes":0}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_votes: 0.630516

Total RMSLE score: 0.630516

>>> model[13]

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-167-af81fba39501>", line 1, in <module>

model[13]

TypeError: 'Model' object does not support indexing

>>> models[13]

Out[168]: <\_\_main\_\_.Model at 0x2a562be0>

>>> models[13].target='num\_views'

>>> models[13].segment="New Haven"

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 16:55

==============================================================================================================

Error Measure: 0.809637818495

Prediction mean: 35.977873

Prediction std dev: 27.023417

Prediction max/min: 282.992820/3.981973

==============================================================================================================

Temporal CV completed at: 11-26-13 16:55. Total runtime: 0:00:01.577000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1655--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features("neighborhood")

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['b', 'd', 'e', 'g', 'h', 'i', 'n', 'o', 'r']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-173-db0720d2d325>", line 20, in <module>

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\train.py", line 20, in cross\_validate

mtxTrn, mtxTest, mtxTrnTarget, mtxTestTarget = combine\_features(model, dfTrn\_Segment, dfTest\_Segment)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\train.py", line 37, in combine\_features

mtxTrn = hstack([mtxTrn, model.features[feature][0]])

File "C:\Python27\lib\site-packages\scipy\sparse\construct.py", line 423, in hstack

return bmat([blocks], format=format, dtype=dtype)

File "C:\Python27\lib\site-packages\scipy\sparse\construct.py", line 539, in bmat

dtype = upcast(\*tuple([A.dtype for A in blocks[block\_mask]]))

File "C:\Python27\lib\site-packages\scipy\sparse\sputils.py", line 60, in upcast

raise TypeError('no supported conversion for types: %r' % (args,))

TypeError: no supported conversion for types: (dtype('S1'), dtype('S1'))

>>> models[13].set\_features(["neighborhood"])

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['neighborhood']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 16:57

==============================================================================================================

Error Measure: 1.20185031267

Prediction mean: 14.650232

Prediction std dev: 15.693470

Prediction max/min: 70.426408/1.334834

==============================================================================================================

Temporal CV completed at: 11-26-13 16:57. Total runtime: 0:00:01.309000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1657--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features(["neighborhood","description\_length\_log"])

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCVvotes,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'neighborhood']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 16:57

==============================================================================================================

Error Measure: 1.23739792698

Prediction mean: 14.859598

Prediction std dev: 16.362712

Prediction max/min: 112.483708/1.304131

==============================================================================================================

Temporal CV completed at: 11-26-13 16:57. Total runtime: 0:00:01.325000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1657--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> for model in models[12:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: richmond\_cmts SEGMENT: Richmond TARGET: num\_comments

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: GBM\_700\_.1 POST-PROCESS SCALAR: 0.85

==============================================================================================================

Temporal CV started at: 11-26-13 16:59

==============================================================================================================

^CTraceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-178-21a8f910fb1e>", line 20, in <module>

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\train.py", line 23, in cross\_validate

cv\_preds = cross\_validate\_temporal(mtxTrn,mtxTest,mtxTrnTarget.ravel(),mtxTestTarget.ravel(),model)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\train.py", line 126, in cross\_validate\_temporal

model.classifier.fit(train\_cv.todense(), y\_target)

File "C:\Python27\lib\site-packages\sklearn\ensemble\gradient\_boosting.py", line 1126, in fit

return super(GradientBoostingRegressor, self).fit(X, y)

File "C:\Python27\lib\site-packages\sklearn\ensemble\gradient\_boosting.py", line 624, in fit

self.train\_score\_[i] = self.loss\_(y, y\_pred)

File "C:\Python27\lib\site-packages\sklearn\ensemble\gradient\_boosting.py", line 200, in \_\_call\_\_

return np.mean((y - pred.ravel()) \*\* 2.0)

File "C:\Python27\lib\site-packages\numpy\core\fromnumeric.py", line 2488, in mean

out=out, keepdims=keepdims)

File "C:\Python27\lib\site-packages\numpy\core\\_methods.py", line 51, in \_mean

out=out, keepdims=keepdims)

KeyboardInterrupt

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'neighborhood']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 16:59

==============================================================================================================

Error Measure: 1.20140563015

Prediction mean: 14.796793

Prediction std dev: 16.277017

Prediction max/min: 108.067742/1.161867

==============================================================================================================

Temporal CV completed at: 11-26-13 16:59. Total runtime: 0:00:01.336000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1659--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> dfTest\_Segment

Out[180]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 2734 entries, 0 to 2733

Data columns (total 35 columns):

id 2734 non-null values

latitude 2734 non-null values

longitude 2734 non-null values

summary 2734 non-null values

description 2734 non-null values

source 2734 non-null values

created\_time 2734 non-null values

tag\_type 2734 non-null values

zipcode 2734 non-null values

street 2734 non-null values

city 2734 non-null values

neighborhood 2734 non-null values

est\_pop 2734 non-null values

avg\_income 2734 non-null values

tot\_income 2734 non-null values

created\_date 2734 non-null values

month 2734 non-null values

created\_time\_hrs 2734 non-null values

created\_time\_range 2734 non-null values

age 2734 non-null values

long\_rnd2 2734 non-null values

lat\_rnd2 2734 non-null values

long\_lat\_rnd2 2734 non-null values

dayofweek 2734 non-null values

description\_length 2734 non-null values

description\_length\_log 2734 non-null values

description\_fg 2734 non-null values

tagtype\_fg 2734 non-null values

weekend\_fg 2734 non-null values

nbr\_longlat 2734 non-null values

segment 2734 non-null values

num\_views 2734 non-null values

num\_votes 2734 non-null values

num\_comments 2734 non-null values

act\_status 2734 non-null values

dtypes: float64(8), int64(8), object(19)

>>> dfTrn\_Segment

Out[181]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 6239 entries, 47 to 223057

Data columns (total 33 columns):

id 6239 non-null values

latitude 6239 non-null values

longitude 6239 non-null values

summary 6239 non-null values

description 6239 non-null values

num\_votes 6239 non-null values

num\_comments 6239 non-null values

num\_views 6239 non-null values

source 6239 non-null values

created\_time 6239 non-null values

tag\_type 6239 non-null values

zipcode 6239 non-null values

street 6239 non-null values

city 6239 non-null values

neighborhood 6239 non-null values

est\_pop 6239 non-null values

avg\_income 6239 non-null values

tot\_income 6239 non-null values

created\_date 6239 non-null values

month 6239 non-null values

created\_time\_hrs 6239 non-null values

created\_time\_range 6239 non-null values

age 6239 non-null values

long\_rnd2 6239 non-null values

lat\_rnd2 6239 non-null values

long\_lat\_rnd2 6239 non-null values

dayofweek 6239 non-null values

description\_length 6239 non-null values

description\_length\_log 6239 non-null values

description\_fg 6239 non-null values

tagtype\_fg 6239 non-null values

weekend\_fg 6239 non-null values

nbr\_longlat 6239 non-null values

dtypes: float64(8), int64(8), object(17)

>>> models[13].set\_features(["hours\_range", "description\_length\_log", "long\_lat", "neighborhood", "source", "summary\_descr\_tfidf\_word", "weekend\_fg"])

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'hours\_range', 'long\_lat', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1.0

==============================================================================================================

Temporal CV started at: 11-26-13 17:00

==============================================================================================================

Error Measure: 0.837247854767

Prediction mean: 37.044281

Prediction std dev: 30.609042

Prediction max/min: 249.570101/2.104306

==============================================================================================================

Temporal CV completed at: 11-26-13 17:00. Total runtime: 0:00:01.621000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1700--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.8

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'hours\_range', 'long\_lat', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.8

==============================================================================================================

Temporal CV started at: 11-26-13 17:00

==============================================================================================================

Error Measure: 0.801099613455

Prediction mean: 29.260287

Prediction std dev: 24.210072

Prediction max/min: 197.457186/1.644366

==============================================================================================================

Temporal CV completed at: 11-26-13 17:00. Total runtime: 0:00:01.655000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1700--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.7

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'hours\_range', 'long\_lat', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.7

==============================================================================================================

Temporal CV started at: 11-26-13 17:01

==============================================================================================================

Error Measure: 0.807324235299

Prediction mean: 25.711114

Prediction std dev: 21.280517

Prediction max/min: 173.523242/1.440484

==============================================================================================================

Temporal CV completed at: 11-26-13 17:01. Total runtime: 0:00:01.616000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1701--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.6

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'hours\_range', 'long\_lat', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.6

==============================================================================================================

Temporal CV started at: 11-26-13 17:01

==============================================================================================================

Error Measure: 0.834528547813

Prediction mean: 22.229002

Prediction std dev: 18.346389

Prediction max/min: 149.559622/1.267022

==============================================================================================================

Temporal CV completed at: 11-26-13 17:01. Total runtime: 0:00:01.651000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1701--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.75

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'hours\_range', 'long\_lat', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.75

==============================================================================================================

Temporal CV started at: 11-26-13 17:01

==============================================================================================================

Error Measure: 0.802113714615

Prediction mean: 27.427496

Prediction std dev: 22.720657

Prediction max/min: 185.253103/1.530379

==============================================================================================================

Temporal CV completed at: 11-26-13 17:01. Total runtime: 0:00:01.650000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1701--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features(["neighborhood"])

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['neighborhood']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.75

==============================================================================================================

Temporal CV started at: 11-26-13 17:01

==============================================================================================================

Error Measure: 1.32233130722

Prediction mean: 10.975777

Prediction std dev: 11.722874

Prediction max/min: 52.820241/1.001465

==============================================================================================================

Temporal CV completed at: 11-26-13 17:02. Total runtime: 0:00:01.315000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1702--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features(["hours\_range", "description\_length\_log", "neighborhood", "source", "summary\_descr\_tfidf\_word", "weekend\_fg"])

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.75

==============================================================================================================

Temporal CV started at: 11-26-13 17:02

==============================================================================================================

Error Measure: 0.806763924701

Prediction mean: 27.139708

Prediction std dev: 21.510138

Prediction max/min: 182.635746/1.634138

==============================================================================================================

Temporal CV completed at: 11-26-13 17:02. Total runtime: 0:00:01.652000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1702--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features(["hours\_range", "description\_length\_log", "neighborhood", "summary\_descr\_tfidf\_word", "weekend\_fg"])

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.75

==============================================================================================================

Temporal CV started at: 11-26-13 17:02

==============================================================================================================

Error Measure: 0.834997883419

Prediction mean: 27.460686

Prediction std dev: 21.193653

Prediction max/min: 172.836683/1.549102

==============================================================================================================

Temporal CV completed at: 11-26-13 17:02. Total runtime: 0:00:01.633000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1702--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features(["hours\_range", "description\_length", "neighborhood", "source", "summary\_descr\_tfidf\_word", "weekend\_fg"])

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.75

==============================================================================================================

Temporal CV started at: 11-26-13 17:02

==============================================================================================================

Error Measure: 0.797565933749

Prediction mean: 26.992143

Prediction std dev: 22.425276

Prediction max/min: 357.823844/1.809710

==============================================================================================================

Temporal CV completed at: 11-26-13 17:02. Total runtime: 0:00:01.639000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1702--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].classifier = linear\_model.Ridge();clf\_name = 'RidgeReg'

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.75

==============================================================================================================

Temporal CV started at: 11-26-13 17:03

==============================================================================================================

Error Measure: 0.817309619293

Prediction mean: 30.393235

Prediction std dev: 31.706655

Prediction max/min: 515.878434/1.840081

==============================================================================================================

Temporal CV completed at: 11-26-13 17:03. Total runtime: 0:00:00.114000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1703--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].classifier = linear\_model.SGDRegressor(alpha=0.001, n\_iter=800,shuffle=True); clf\_name='SGD\_001\_800'

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.75

==============================================================================================================

Temporal CV started at: 11-26-13 17:03

==============================================================================================================

Error Measure: 0.796780382251

Prediction mean: 27.160375

Prediction std dev: 22.320525

Prediction max/min: 341.737189/1.876994

==============================================================================================================

Temporal CV completed at: 11-26-13 17:03. Total runtime: 0:00:01.650000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1703--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> del models[13].dfPredictions

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.75

==============================================================================================================

Temporal CV started at: 11-26-13 17:03

==============================================================================================================

Error Measure: 0.797298265487

Prediction mean: 27.182519

Prediction std dev: 22.639158

Prediction max/min: 366.652969/1.818641

==============================================================================================================

Temporal CV completed at: 11-26-13 17:04. Total runtime: 0:00:01.649000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1704--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 17:04

==============================================================================================================

Prediction mean: 23.168101

Prediction std dev: 21.879792

Prediction max/min: 245.627968/1.911854

==============================================================================================================

Predictions completed at: 11-26-13 17:04. Total runtime: 0:00:01.618000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1704--Rich Votes\_RF\_test.csv

>>>

#munge

for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "New Haven"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner")

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

#Set weights

True value for target variables successfully loaded into self.df\_true\_segment

>>>

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(3):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(3):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[-0.08304813 4.24217135 -2.84270269]]

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/rich\_votes\_test.csv")

Model loaded into index 0

Model loaded into index 1

Model loaded into index 2

>>> for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "New Haven"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner")

>>>

>>> dfSegTest

C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

Out[225]:

<class 'pandas.core.frame.DataFrame'>

Int64Index: 2734 entries, 0 to 2733

Data columns (total 6 columns):

id 2734 non-null values

segment 2734 non-null values

num\_views 2734 non-null values

num\_votes 2734 non-null values

num\_comments 2734 non-null values

act\_status 2734 non-null values

dtypes: float64(3), int64(1), object(2)

C:\Python27\lib\site-packages\pandas\core\frame.py:571: FutureWarning: The \_verbose\_info property will be removed in version 0.13. please use "max\_info\_rows"

'0.13. please use "max\_info\_rows"', FutureWarning)

>>> dfSegTest.num\_views

Out[226]:

0 6.551509

1 0.000000

2 6.548935

3 3.034953

4 1.029619

5 1.987874

6 5.990714

7 0.000000

8 4.130355

9 1.029619

10 4.699571

11 1.029619

12 0.641854

13 5.988460

14 0.641854

...

2719 3.788725

2720 2.895912

2721 6.034763

2722 3.453157

2723 3.481240

2724 2.944439

2725 3.768153

2726 3.610918

2727 3.363842

2728 2.895912

2729 4.395683

2730 3.117950

2731 3.788725

2732 4.023564

2733 3.703768

Name: num\_views, Length: 2734, dtype: float64

>>>

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

True value for target variables successfully loaded into self.df\_true\_segment

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-234-1bc681b6e36d>", line 1, in <module>

ensemble\_test.transform\_targets\_log()

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ensembles.py", line 98, in transform\_targets\_log

self.sub\_models[i][target] = np.log(self.sub\_models[i][target] + 1)

File "C:\Python27\lib\site-packages\pandas\core\frame.py", line 2003, in \_\_getitem\_\_

return self.\_get\_item\_cache(key)

File "C:\Python27\lib\site-packages\pandas\core\generic.py", line 667, in \_get\_item\_cache

values = self.\_data.get(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1655, in get

\_, block = self.\_find\_block(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1935, in \_find\_block

self.\_check\_have(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1942, in \_check\_have

raise KeyError('no item named %s' % com.pprint\_thing(item))

KeyError: u'no item named num\_views'

>>>

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

#ensemble\_test.load\_models\_csv(filepath="Submits/rich\_votes\_test.csv")

#munge

for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "New Haven"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner")

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

Model loaded into index 0

Model loaded into index 1

True value for target variables successfully loaded into self.df\_true\_segment

>>>

>>> weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.6},{"num\_views":.8,"num\_votes":.9,"num\_comments":.4}]

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-251-3e0774b7ef09>", line 5, in <module>

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

IndexError: list index out of range

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[ 0.90132517 0.09843419]]

>>> ensemble\_test.create\_ensemble\_segment([0,1],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 0.789681

Total RMSLE score: 0.789681

>>> ensemble\_test.targets=['num\_comments']

>>> for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

True value for target variables successfully loaded into self.df\_true\_segment

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

File "<ipython-input-265-36017f158252>", line 1

for target\_label in ensemble\_test.targets:

^

IndentationError: unexpected indent

If you want to paste code into IPython, try the %paste and %cpaste magic functions.

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[ 0.96979079 0.54252882]]

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

#ensemble\_test.load\_models\_csv(filepath="Submits/rich\_votes\_test.csv")

#munge

for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Chicago\_Other"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner")

Model loaded into index 0

Model loaded into index 1

>>>

>>>

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

True value for target variables successfully loaded into self.df\_true\_segment

>>>

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-282-428dd84d6ab1>", line 8, in <module>

clf.fit(train\_2,target\_2)

File "C:\Python27\lib\site-packages\sklearn\linear\_model\base.py", line 379, in fit

linalg.lstsq(X, y)

File "C:\Python27\lib\site-packages\scipy\linalg\basic.py", line 507, in lstsq

a1,b1 = map(np.asarray\_chkfinite, (a,b))

File "C:\Python27\lib\site-packages\numpy\lib\function\_base.py", line 590, in asarray\_chkfinite

"array must not contain infs or NaNs")

ValueError: array must not contain infs or NaNs

>>> dfSegTest

Out[283]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1155 entries, 0 to 1154

Data columns (total 6 columns):

id 1155 non-null values

segment 1155 non-null values

num\_views 1153 non-null values

num\_votes 1155 non-null values

num\_comments 1155 non-null values

act\_status 1153 non-null values

dtypes: float64(3), int64(1), object(2)

>>> train\_2

Out[284]:

array([[ 1.70942955, 1.84873594],

[ 1.42470271, 1.77804593],

[ 1.30012624, 1.39757358],

...,

[ 1.721794 , 1.99046135],

[ 1.89926854, 2.10115543],

[ 1.721794 , 1.77893794]])

>>> target\_2

Out[285]:

array([[ 3.21887582],

[ 2.00148 ],

[ 1.22377543],

...,

[ 2.00148 ],

[ 2.43361336],

[ 2.56494936]])

>>> target\_2.shape

Out[286]: (1155L, 1L)

>>> train\_2.shape

Out[287]: (1155L, 2L)

>>> weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.6},{"num\_views":.8,"num\_votes":.9,"num\_comments":.4}]

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-289-428dd84d6ab1>", line 8, in <module>

clf.fit(train\_2,target\_2)

File "C:\Python27\lib\site-packages\sklearn\linear\_model\base.py", line 379, in fit

linalg.lstsq(X, y)

File "C:\Python27\lib\site-packages\scipy\linalg\basic.py", line 507, in lstsq

a1,b1 = map(np.asarray\_chkfinite, (a,b))

File "C:\Python27\lib\site-packages\numpy\lib\function\_base.py", line 590, in asarray\_chkfinite

"array must not contain infs or NaNs")

ValueError: array must not contain infs or NaNs

>>> dfSegTest

Out[290]:

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1155 entries, 0 to 1154

Data columns (total 6 columns):

id 1155 non-null values

segment 1155 non-null values

num\_views 1153 non-null values

num\_votes 1155 non-null values

num\_comments 1155 non-null values

act\_status 1153 non-null values

dtypes: float64(3), int64(1), object(2)

C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

>>> dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

>>> dfSegTest

Out[292]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1153 entries, 0 to 1154

Data columns (total 6 columns):

id 1153 non-null values

segment 1153 non-null values

num\_views 1153 non-null values

num\_votes 1153 non-null values

num\_comments 1153 non-null values

act\_status 1153 non-null values

dtypes: float64(3), int64(1), object(2)

>>> ensemble\_test.sub\_models\_segment[0]

Out[293]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1155 entries, 0 to 1154

Data columns (total 6 columns):

index 1155 non-null values

id 1155 non-null values

num\_views 1155 non-null values

num\_votes 1155 non-null values

num\_comments 1155 non-null values

Segment 1155 non-null values

dtypes: float64(3), int64(2), object(1)

>>> ensemble\_test.sub\_models\_segment[1]

C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

Out[294]:

<class 'pandas.core.frame.DataFrame'>

Int64Index: 1155 entries, 0 to 1154

Data columns (total 6 columns):

index 1155 non-null values

id 1155 non-null values

num\_views 1155 non-null values

num\_votes 1155 non-null values

num\_comments 1155 non-null values

Segment 1155 non-null values

dtypes: float64(3), int64(2), object(1)

>>> ensemble\_test.sub\_models\_segment[1].num\_views

Out[295]:

0 1.848736

1 1.778046

2 1.397574

3 1.399395

4 1.171600

5 1.720912

6 1.415998

7 1.401042

8 1.380065

9 1.928196

10 1.591158

11 1.321028

12 1.913601

13 1.769043

14 1.617398

...

1140 2.001750

1141 2.222591

1142 1.531457

1143 1.695070

1144 2.298124

1145 2.006648

1146 1.391932

1147 1.709538

1148 1.387692

1149 1.480460

1150 1.629517

1151 1.638583

1152 1.990461

1153 2.101155

1154 1.778938

Name: num\_views, Length: 1155, dtype: float64

>>> ensemble\_test.sub\_models\_segment[0].num\_views

Out[296]:

0 1.709430

1 1.424703

2 1.300126

3 1.763604

4 1.300126

5 1.721794

6 1.300126

7 1.500051

8 1.385198

9 1.736546

10 1.424703

11 1.500051

12 1.424703

13 1.721794

14 1.721794

...

1140 2.005947

1141 2.005947

1142 1.709430

1143 1.736546

1144 2.005947

1145 1.721794

1146 1.385198

1147 1.709430

1148 1.500051

1149 1.500051

1150 1.692864

1151 1.736546

1152 1.721794

1153 1.899269

1154 1.721794

Name: num\_views, Length: 1155, dtype: float64

>>> dfSegTest.num\_views

Out[297]:

0 3.218876

1 2.001480

2 1.223775

3 1.435085

4 1.757858

5 1.887070

6 1.435085

7 2.624669

8 1.435085

9 1.887070

10 2.001480

11 3.280911

12 1.887070

13 1.223775

14 1.609438

...

1140 1.609438

1141 1.435085

1142 2.001480

1143 1.435085

1144 1.757858

1145 1.887070

1146 1.223775

1147 2.001480

1148 1.757858

1149 1.609438

1150 0.955511

1151 2.282382

1152 2.001480

1153 2.433613

1154 2.564949

Name: num\_views, Length: 1153, dtype: float64

>>> for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Chicago\_Other"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

>>>

>>>

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

True value for target variables successfully loaded into self.df\_true\_segment

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-308-428dd84d6ab1>", line 8, in <module>

clf.fit(train\_2,target\_2)

File "C:\Python27\lib\site-packages\sklearn\linear\_model\base.py", line 379, in fit

linalg.lstsq(X, y)

File "C:\Python27\lib\site-packages\scipy\linalg\basic.py", line 518, in lstsq

raise ValueError('incompatible dimensions')

ValueError: incompatible dimensions

>>> train\_2

Out[309]:

array([[ 0.99673812, 1.04687537],

[ 0.88570892, 1.02174778],

[ 0.83296401, 0.87445722],

...,

[ 1.00129122, 1.09542767],

[ 1.06445848, 1.13177476],

[ 1.00129122, 1.02206882]])

>>> train\_2.shape

Out[310]: (1155L, 2L)

>>> target\_2.shape

Out[311]: (1153L, 1L)

>>> dfCV =dfCV.dropna()

>>> dfCV

C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

Out[313]:

<class 'pandas.core.frame.DataFrame'>

Int64Index: 25142 entries, 0 to 25144

Data columns (total 6 columns):

id 25142 non-null values

segment 25142 non-null values

num\_views 25142 non-null values

num\_votes 25142 non-null values

num\_comments 25142 non-null values

act\_status 25142 non-null values

dtypes: float64(3), int64(1), object(2)

>>> for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Chicago\_Other"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

>>>

>>>

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

True value for target variables successfully loaded into self.df\_true\_segment

>>>

for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-324-428dd84d6ab1>", line 8, in <module>

clf.fit(train\_2,target\_2)

File "C:\Python27\lib\site-packages\sklearn\linear\_model\base.py", line 379, in fit

linalg.lstsq(X, y)

File "C:\Python27\lib\site-packages\scipy\linalg\basic.py", line 518, in lstsq

raise ValueError('incompatible dimensions')

ValueError: incompatible dimensions

>>> ensemble\_test.create\_ensemble\_segment([0,1],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-328-b79dd3fe69eb>", line 1, in <module>

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ensembles.py", line 131, in score\_rmsle

target\_score = ml\_metrics.rmsle(df\_true[target], df[target])

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ml\_metrics.py", line 155, in rmsle

return np.sqrt(msle(actual, predicted))

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ml\_metrics.py", line 111, in msle

return np.mean(sle(actual, predicted))

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\ml\_metrics.py", line 200, in sle

np.log(np.array(predicted)+1), 2))

ValueError: operands could not be broadcast together with shapes (1153) (1155)

>>> for i in range(2):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

>>> ensemble\_test.create\_ensemble\_segment([0,1],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 30.699778

Total RMSLE score: 30.699778

>>>

for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[ -6.15199675 15.65759819]]

>>> models[13]

Out[335]: <\_\_main\_\_.Model at 0x2a562be0>

>>> models[13].target

Out[336]: 'num\_views'

>>> models[13].target="num\_votes"

>>> models[13].set\_features(["hours\_range", "description\_length", "neighborhood", "source", "summary\_descr\_tfidf\_word", "weekend\_fg"])

>>> models[13].segment

Out[339]: 'New Haven'

>>> models[13].postprocess\_scalar

Out[340]: 0.75

>>> models[13].postprocess\_scalar = 1

>>> del models[13].dfPredictions

>>> for model in models[12:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: richmond\_cmts SEGMENT: Richmond TARGET: num\_comments

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: GBM\_700\_.1 POST-PROCESS SCALAR: 0.85

==============================================================================================================

Temporal CV started at: 11-26-13 17:24

==============================================================================================================

Error Measure: 0.299330998222

Prediction mean: 0.117728

Prediction std dev: 0.085338

Prediction max/min: 1.311823/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 17:24. Total runtime: 0:00:14.890000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1724--richmond\_cmts\_GBM\_700\_.1\_CV\_list.csv

Model predictions started at: 11-26-13 17:24

==============================================================================================================

Prediction mean: 0.117118

Prediction std dev: 0.086004

Prediction max/min: 2.128954/0.000000

==============================================================================================================

Predictions completed at: 11-26-13 17:24. Total runtime: 0:00:14.742000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1724--richmond\_cmts\_GBM\_700\_.1\_test.csv

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_votes

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1

==============================================================================================================

Temporal CV started at: 11-26-13 17:24

==============================================================================================================

Error Measure: 1.122712344

Prediction mean: 3.546499

Prediction std dev: 1.057301

Prediction max/min: 10.597946/1.514063

==============================================================================================================

Temporal CV completed at: 11-26-13 17:24. Total runtime: 0:00:01.622000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1724--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 17:24

==============================================================================================================

Prediction mean: 3.307199

Prediction std dev: 1.133698

Prediction max/min: 9.511855/1.665353

==============================================================================================================

Predictions completed at: 11-26-13 17:24. Total runtime: 0:00:01.863000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1724--Rich Votes\_RF\_test.csv

>>> del models[13].dfPredictions

>>> models[13].target="num\_comments"

>>> models[13].postprocess\_scalar = .\_117

File "<ipython-input-346-58cf0ef6695d>", line 1

models[13].postprocess\_scalar = .\_117

^

SyntaxError: invalid syntax

>>> models[13].postprocess\_scalar = .7

>>> for model in models[12:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: richmond\_cmts SEGMENT: Richmond TARGET: num\_comments

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: GBM\_700\_.1 POST-PROCESS SCALAR: 0.85

==============================================================================================================

Temporal CV started at: 11-26-13 17:26

==============================================================================================================

Error Measure: 0.299330998222

Prediction mean: 0.117728

Prediction std dev: 0.085338

Prediction max/min: 1.311823/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 17:26. Total runtime: 0:00:15.336000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1726--richmond\_cmts\_GBM\_700\_.1\_CV\_list.csv

Predictions found. Using cached predictions.

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_comments

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.7

==============================================================================================================

Temporal CV started at: 11-26-13 17:26

==============================================================================================================

Error Measure: 0.683135616577

Prediction mean: 0.563861

Prediction std dev: 0.442659

Prediction max/min: 5.801923/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 17:26. Total runtime: 0:00:01.668000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1726--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 17:27

==============================================================================================================

Prediction mean: 0.495357

Prediction std dev: 0.456509

Prediction max/min: 4.575910/0.000000

==============================================================================================================

Predictions completed at: 11-26-13 17:27. Total runtime: 0:00:01.696000

==============================================================================================================

Submission file saved as Submits/11-26-13\_1727--Rich Votes\_RF\_test.csv

>>> dfTrn = data\_io.load\_flatfile\_to\_df(settings['filename\_train'])

dfTest = data\_io.load\_flatfile\_to\_df(settings['filename\_test'])

dfCV = data\_io.load\_flatfile\_to\_df('Data/CV.csv')

#Clean/Munge the data

dfTrn = munge.clean(dfTrn)

dfTest = munge.clean(dfTest)

#-------Feature creation-------------------------#

#Add all currently used hand crafted features to dataframes

features.add(dfTrn)

features.add(dfTest)

#---------Data slicing/parsing--------------------------#

#Temporal split of data for CV

if settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'april':

dfTrnCV, dfTestCV = munge.temporal\_split(dfTrn, (2013, 04, 1))

elif settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'march':

#take an addtional week from February b/c of lack of remote\_api source issues in March

dfTrnCV, dfTestCV = munge.temporal\_split(dfTrn, (2013, 02, 21))

elif settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'list\_split':

#load stored list of data points and use those for CV

dfCVlist = pd.DataFrame({'id': data\_io.load\_cached\_object("Cache/cv\_issue\_ids.pkl"), 'dummy': 0})

dfTrnCV, dfTestCV = munge.list\_split(dfTrn, dfCVlist)

features.sub\_feature(dfTrnCV,'zipcode','neighborhood',["Richmond","Oakland","Manchester","Chicago","New Haven"])

features.sub\_feature(dfTestCV,'zipcode','neighborhood',["Richmond","Oakland","Manchester","Chicago","New Haven"])

features.knn\_thresholding(dfTrnCV,'neighborhood',5)

features.knn\_thresholding(dfTestCV,'neighborhood',5)

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-354-e292f15c9686>", line 1, in <module>

dfTest = munge.clean(dfTest)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\munge.py", line 16, in clean

df['created\_time'] = [datetime.datetime.strptime(x, "%m/%d/%Y %H:%M") for x in df.created\_time]

TypeError: must be string, not float

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-358-f3ef9f35e2b0>", line 1, in <module>

features.add(dfTest)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\features.py", line 19, in add

hours(df)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\features.py", line 33, in hours

df['created\_time\_hrs'] = [str(x.hour) for x in df['created\_time']]

AttributeError: 'str' object has no attribute 'hour'

Out[364]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 140975 entries, 0 to 223127

Data columns (total 33 columns):

id 140975 non-null values

latitude 140975 non-null values

longitude 140975 non-null values

summary 140975 non-null values

description 140975 non-null values

num\_votes 140975 non-null values

num\_comments 140975 non-null values

num\_views 140975 non-null values

source 140975 non-null values

created\_time 140975 non-null values

tag\_type 140975 non-null values

zipcode 140975 non-null values

street 140975 non-null values

city 140975 non-null values

neighborhood 140975 non-null values

est\_pop 140975 non-null values

avg\_income 140975 non-null values

tot\_income 140975 non-null values

created\_date 140975 non-null values

month 140975 non-null values

created\_time\_hrs 140975 non-null values

created\_time\_range 140975 non-null values

age 140975 non-null values

long\_rnd2 140975 non-null values

lat\_rnd2 140975 non-null values

long\_lat\_rnd2 140975 non-null values

dayofweek 140975 non-null values

description\_length 140975 non-null values

description\_length\_log 140975 non-null values

description\_fg 140975 non-null values

tagtype\_fg 140975 non-null values

weekend\_fg 140975 non-null values

nbr\_longlat 140975 non-null values

dtypes: float64(8), int64(8), object(17)

>>>

Out[365]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 26769 entries, 2 to 223124

Data columns (total 33 columns):

id 26769 non-null values

latitude 26769 non-null values

longitude 26769 non-null values

summary 26769 non-null values

description 26769 non-null values

num\_votes 26769 non-null values

num\_comments 26769 non-null values

num\_views 26769 non-null values

source 26769 non-null values

created\_time 26769 non-null values

tag\_type 26769 non-null values

zipcode 26769 non-null values

street 26769 non-null values

city 26769 non-null values

neighborhood 26769 non-null values

est\_pop 26769 non-null values

avg\_income 26769 non-null values

tot\_income 26769 non-null values

created\_date 26769 non-null values

month 26769 non-null values

created\_time\_hrs 26769 non-null values

created\_time\_range 26769 non-null values

age 26769 non-null values

long\_rnd2 26769 non-null values

lat\_rnd2 26769 non-null values

long\_lat\_rnd2 26769 non-null values

dayofweek 26769 non-null values

description\_length 26769 non-null values

description\_length\_log 26769 non-null values

description\_fg 26769 non-null values

tagtype\_fg 26769 non-null values

weekend\_fg 26769 non-null values

nbr\_longlat 26769 non-null values

dtypes: float64(8), int64(8), object(17)

>>>

>>>

>>>

>>>

>>>

>>>

>>> dfTrn = data\_io.load\_flatfile\_to\_df(settings['filename\_train'])

dfTest = data\_io.load\_flatfile\_to\_df(settings['filename\_test'])

dfCV = data\_io.load\_flatfile\_to\_df('Data/CV.csv')

>>> dfTrn = munge.clean(dfTrn)

dfTest = munge.clean(dfTest)

>>>

>>>

>>>

>>> features.add(dfTrn)

features.add(dfTest)

>>>

#---------Data slicing/parsing--------------------------#

#Temporal split of data for CV

if settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'april':

dfTrnCV, dfTestCV = munge.temporal\_split(dfTrn, (2013, 04, 1))

elif settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'march':

#take an addtional week from February b/c of lack of remote\_api source issues in March

dfTrnCV, dfTestCV = munge.temporal\_split(dfTrn, (2013, 02, 21))

elif settings['generate\_cv\_score'] == 'true' and settings['cv\_method'] == 'list\_split':

#load stored list of data points and use those for CV

dfCVlist = pd.DataFrame({'id': data\_io.load\_cached\_object("Cache/cv\_issue\_ids.pkl"), 'dummy': 0})

dfTrnCV, dfTestCV = munge.list\_split(dfTrn, dfCVlist)

>>>

>>>

>>> for target in ["num\_views","num\_votes","num\_comments"]:

dfCV[target] = np.log(dfCV[target] + 1)

>>> models[15]

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-377-8d7bd5d1a956>", line 1, in <module>

models[15]

IndexError: list index out of range

>>> models[13]

Out[378]: <\_\_main\_\_.Model at 0x2a562be0>

>>> models[13].target

Out[379]: 'num\_comments'

>>> models[13].segment

Out[380]: 'New Haven'

>>> models[13].classifier

Out[381]:

SGDRegressor(alpha=0.001, epsilon=0.1, eta0=0.01, fit\_intercept=True,

l1\_ratio=0.15, learning\_rate='invscaling', loss='squared\_loss',

n\_iter=800, penalty='l2', power\_t=0.25, random\_state=None, rho=None,

shuffle=True, verbose=0, warm\_start=False)

>>> models[13].set\_features(["hours\_range", "neighborhood", "source","description\_length", "summary\_descr\_tfidf\_word", "weekend\_fg"])

>>> for model in models[12:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: richmond\_cmts SEGMENT: Richmond TARGET: num\_comments

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: GBM\_700\_.1 POST-PROCESS SCALAR: 0.85

==============================================================================================================

Temporal CV started at: 11-26-13 22:23

==============================================================================================================

Error Measure: 0.299323100908

Prediction mean: 0.117729

Prediction std dev: 0.085335

Prediction max/min: 1.311823/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 22:23. Total runtime: 0:00:14.937000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2223--richmond\_cmts\_GBM\_700\_.1\_CV\_list.csv

Predictions found. Using cached predictions.

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_comments

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.7

==============================================================================================================

Temporal CV started at: 11-26-13 22:23

==============================================================================================================

Error Measure: 0.684376767424

Prediction mean: 0.564872

Prediction std dev: 0.445164

Prediction max/min: 5.928692/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 22:23. Total runtime: 0:00:01.699000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2223--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features(["hours\_range", "neighborhood", "source","description\_length", "summary\_vec", "weekend\_fg"])

>>> for model in models[12:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: richmond\_cmts SEGMENT: Richmond TARGET: num\_comments

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: GBM\_700\_.1 POST-PROCESS SCALAR: 0.85

==============================================================================================================

Temporal CV started at: 11-26-13 22:25

==============================================================================================================

Error Measure: 0.299323100908

Prediction mean: 0.117729

Prediction std dev: 0.085335

Prediction max/min: 1.311823/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 22:25. Total runtime: 0:00:15.036000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2225--richmond\_cmts\_GBM\_700\_.1\_CV\_list.csv

Predictions found. Using cached predictions.

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_comments

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_vec', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.7

==============================================================================================================

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-385-21a8f910fb1e>", line 20, in <module>

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\train.py", line 20, in cross\_validate

mtxTrn, mtxTest, mtxTrnTarget, mtxTestTarget = combine\_features(model, dfTrn\_Segment, dfTest\_Segment)

File "C:\Users\bgregory\MyDocs\Dropbox\DataScience\Kaggle\Hackathon-311\train.py", line 37, in combine\_features

mtxTrn = hstack([mtxTrn, model.features[feature][0]])

File "C:\Python27\lib\site-packages\scipy\sparse\construct.py", line 423, in hstack

return bmat([blocks], format=format, dtype=dtype)

File "C:\Python27\lib\site-packages\scipy\sparse\construct.py", line 523, in bmat

raise ValueError('blocks[%d,:] has incompatible row dimensions' % i)

ValueError: blocks[0,:] has incompatible row dimensions

>>> models[13].set\_features(["hours\_range", "neighborhood", "source","description\_length", "summary\_cat", "weekend\_fg"])

>>> for model in models[12:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: richmond\_cmts SEGMENT: Richmond TARGET: num\_comments

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: GBM\_700\_.1 POST-PROCESS SCALAR: 0.85

==============================================================================================================

Temporal CV started at: 11-26-13 22:26

==============================================================================================================

Error Measure: 0.299323100908

Prediction mean: 0.117729

Prediction std dev: 0.085335

Prediction max/min: 1.311823/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 22:26. Total runtime: 0:00:14.842000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2226--richmond\_cmts\_GBM\_700\_.1\_CV\_list.csv

Predictions found. Using cached predictions.

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_comments

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_cat', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.7

==============================================================================================================

Temporal CV started at: 11-26-13 22:26

==============================================================================================================

Error Measure: 0.697267336353

Prediction mean: 0.538761

Prediction std dev: 0.406547

Prediction max/min: 3.501812/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 22:26. Total runtime: 0:00:01.521000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2226--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].set\_features(["hours\_range", "neighborhood", "source","description\_length", "summary\_descr\_tfidf\_word", "weekend\_fg"])

>>> del models[13].dfPredictions

>>> for model in models[12:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: richmond\_cmts SEGMENT: Richmond TARGET: num\_comments

FEATURES: ['description\_length\_log', 'hours\_range', 'neighborhood', 'source', 'weekend\_fg']

LEARNING ALGORITHM: GBM\_700\_.1 POST-PROCESS SCALAR: 0.85

==============================================================================================================

Temporal CV started at: 11-26-13 22:29

==============================================================================================================

Error Measure: 0.299323100908

Prediction mean: 0.117729

Prediction std dev: 0.085335

Prediction max/min: 1.311823/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 22:29. Total runtime: 0:00:14.981000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2229--richmond\_cmts\_GBM\_700\_.1\_CV\_list.csv

Predictions found. Using cached predictions.

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_comments

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.7

==============================================================================================================

Temporal CV started at: 11-26-13 22:29

==============================================================================================================

Error Measure: 0.68240956467

Prediction mean: 0.576070

Prediction std dev: 0.447053

Prediction max/min: 5.933734/0.000000

==============================================================================================================

Temporal CV completed at: 11-26-13 22:29. Total runtime: 0:00:01.722000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2229--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 22:29

==============================================================================================================

Prediction mean: 0.484884

Prediction std dev: 0.451075

Prediction max/min: 4.511387/0.000000

==============================================================================================================

Predictions completed at: 11-26-13 22:29. Total runtime: 0:00:01.702000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2229--Rich Votes\_RF\_test.csv

>>> del models[13].dfPredictions

>>> models[13].target = 'num\_votes'

>>> models[13].postprocess\_scalar

Out[393]: 0.7

>>> models[13].postprocess\_scalar=1

>>>

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_votes

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 1

==============================================================================================================

Temporal CV started at: 11-26-13 22:34

==============================================================================================================

Error Measure: 1.12333631776

Prediction mean: 3.548929

Prediction std dev: 1.060534

Prediction max/min: 10.693467/1.510467

==============================================================================================================

Temporal CV completed at: 11-26-13 22:34. Total runtime: 0:00:01.623000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2234--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 22:34

==============================================================================================================

Prediction mean: 3.298676

Prediction std dev: 1.124955

Prediction max/min: 9.344513/1.667866

==============================================================================================================

Predictions completed at: 11-26-13 22:34. Total runtime: 0:00:01.644000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2234--Rich Votes\_RF\_test.csv

>>> models[13].postprocess\_scalar=.74

>>> models[13].target = 'num\_views'

>>> del models[13].dfPredictions

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: New Haven TARGET: num\_views

FEATURES: ['description\_length', 'hours\_range', 'neighborhood', 'source', 'summary\_descr\_tfidf\_word', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.74

==============================================================================================================

Temporal CV started at: 11-26-13 22:35

==============================================================================================================

Error Measure: 0.805019339239

Prediction mean: 26.556359

Prediction std dev: 21.793927

Prediction max/min: 334.193390/1.851198

==============================================================================================================

Temporal CV completed at: 11-26-13 22:35. Total runtime: 0:00:01.619000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2235--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 22:35

==============================================================================================================

Prediction mean: 22.879069

Prediction std dev: 21.568814

Prediction max/min: 245.475575/1.899685

==============================================================================================================

Predictions completed at: 11-26-13 22:35. Total runtime: 0:00:01.669000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2235--Rich Votes\_RF\_test.csv

>>> del models[13].dfPredictions

>>> models[13].target = 'num\_views'

>>> models[13].segment = "New Haven"

>>> models[13].postprocess\_scalar=.64

>>> models[13].segment = "Richmond"

>>> models[13].set\_features(["neighborhood","weekend\_fg","summary\_count","hours\_range"])

>>> features.sub\_feature(dfTrnCV,'zipcode','neighborhood',["Richmond","Oakland","Manchester","Chicago","New Haven"])

features.sub\_feature(dfTestCV,'zipcode','neighborhood',["Richmond","Oakland","Manchester","Chicago","New Haven"])

features.knn\_thresholding(dfTrnCV,'neighborhood',8)

features.knn\_thresholding(dfTestCV,'neighborhood',8)

>>>

Out[408]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 140975 entries, 0 to 223127

Data columns (total 33 columns):

id 140975 non-null values

latitude 140975 non-null values

longitude 140975 non-null values

summary 140975 non-null values

description 140975 non-null values

num\_votes 140975 non-null values

num\_comments 140975 non-null values

num\_views 140975 non-null values

source 140975 non-null values

created\_time 140975 non-null values

tag\_type 140975 non-null values

zipcode 140975 non-null values

street 140975 non-null values

city 140975 non-null values

neighborhood 140975 non-null values

est\_pop 140975 non-null values

avg\_income 140975 non-null values

tot\_income 140975 non-null values

created\_date 140975 non-null values

month 140975 non-null values

created\_time\_hrs 140975 non-null values

created\_time\_range 140975 non-null values

age 140975 non-null values

long\_rnd2 140975 non-null values

lat\_rnd2 140975 non-null values

long\_lat\_rnd2 140975 non-null values

dayofweek 140975 non-null values

description\_length 140975 non-null values

description\_length\_log 140975 non-null values

description\_fg 140975 non-null values

tagtype\_fg 140975 non-null values

weekend\_fg 140975 non-null values

nbr\_longlat 140975 non-null values

dtypes: float64(8), int64(8), object(17)

Out[409]: C:\Python27\lib\site-packages\pandas\core\config.py:570: DeprecationWarning: height has been deprecated.

warnings.warn(d.msg, DeprecationWarning)

<class 'pandas.core.frame.DataFrame'>

Int64Index: 26769 entries, 2 to 223124

Data columns (total 33 columns):

id 26769 non-null values

latitude 26769 non-null values

longitude 26769 non-null values

summary 26769 non-null values

description 26769 non-null values

num\_votes 26769 non-null values

num\_comments 26769 non-null values

num\_views 26769 non-null values

source 26769 non-null values

created\_time 26769 non-null values

tag\_type 26769 non-null values

zipcode 26769 non-null values

street 26769 non-null values

city 26769 non-null values

neighborhood 26769 non-null values

est\_pop 26769 non-null values

avg\_income 26769 non-null values

tot\_income 26769 non-null values

created\_date 26769 non-null values

month 26769 non-null values

created\_time\_hrs 26769 non-null values

created\_time\_range 26769 non-null values

age 26769 non-null values

long\_rnd2 26769 non-null values

lat\_rnd2 26769 non-null values

long\_lat\_rnd2 26769 non-null values

dayofweek 26769 non-null values

description\_length 26769 non-null values

description\_length\_log 26769 non-null values

description\_fg 26769 non-null values

tagtype\_fg 26769 non-null values

weekend\_fg 26769 non-null values

nbr\_longlat 26769 non-null values

dtypes: float64(8), int64(8), object(17)

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_views

FEATURES: ['hours\_range', 'neighborhood', 'summary\_count', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.64

==============================================================================================================

Temporal CV started at: 11-26-13 22:49

==============================================================================================================

Error Measure: 0.697628268306

Prediction mean: 4.325642

Prediction std dev: 3.160812

Prediction max/min: 65.320506/0.148430

==============================================================================================================

Temporal CV completed at: 11-26-13 22:49. Total runtime: 0:00:02.842000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2249--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 22:49

==============================================================================================================

Prediction mean: 4.230095

Prediction std dev: 3.122285

Prediction max/min: 64.716055/0.137073

==============================================================================================================

Predictions completed at: 11-26-13 22:49. Total runtime: 0:00:02.818000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2249--Rich Votes\_RF\_test.csv

>>> models[13].postprocess\_scalar=.68

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_views

FEATURES: ['hours\_range', 'neighborhood', 'summary\_count', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.68

==============================================================================================================

Temporal CV started at: 11-26-13 22:50

==============================================================================================================

Error Measure: 0.704968388723

Prediction mean: 4.463511

Prediction std dev: 3.303269

Prediction max/min: 68.430449/0.141732

==============================================================================================================

Temporal CV completed at: 11-26-13 22:50. Total runtime: 0:00:02.872000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2250--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.61

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_views

FEATURES: ['hours\_range', 'neighborhood', 'summary\_count', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.61

==============================================================================================================

Temporal CV started at: 11-26-13 22:50

==============================================================================================================

Error Measure: 0.682667148105

Prediction mean: 3.988099

Prediction std dev: 2.959829

Prediction max/min: 61.354068/0.122342

==============================================================================================================

Temporal CV completed at: 11-26-13 22:50. Total runtime: 0:00:02.849000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2250--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.58

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_views

FEATURES: ['hours\_range', 'neighborhood', 'summary\_count', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.58

==============================================================================================================

Temporal CV started at: 11-26-13 22:51

==============================================================================================================

Error Measure: 0.676961997249

Prediction mean: 3.818029

Prediction std dev: 2.826142

Prediction max/min: 58.495931/0.122629

==============================================================================================================

Temporal CV completed at: 11-26-13 22:51. Total runtime: 0:00:02.910000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2251--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.54

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_views

FEATURES: ['hours\_range', 'neighborhood', 'summary\_count', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.54

==============================================================================================================

Temporal CV started at: 11-26-13 22:51

==============================================================================================================

Error Measure: 0.671389850812

Prediction mean: 3.551781

Prediction std dev: 2.625338

Prediction max/min: 54.419469/0.111687

==============================================================================================================

Temporal CV completed at: 11-26-13 22:51. Total runtime: 0:00:02.940000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2251--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.50

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_views

FEATURES: ['hours\_range', 'neighborhood', 'summary\_count', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.5

==============================================================================================================

Temporal CV started at: 11-26-13 22:51

==============================================================================================================

Error Measure: 0.669057739793

Prediction mean: 3.349794

Prediction std dev: 2.460943

Prediction max/min: 50.863478/0.115649

==============================================================================================================

Temporal CV completed at: 11-26-13 22:51. Total runtime: 0:00:02.922000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2251--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.47

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_views

FEATURES: ['hours\_range', 'neighborhood', 'summary\_count', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.47

==============================================================================================================

Temporal CV started at: 11-26-13 22:51

==============================================================================================================

Error Measure: 0.672623267793

Prediction mean: 3.063486

Prediction std dev: 2.278430

Prediction max/min: 47.208532/0.095999

==============================================================================================================

Temporal CV completed at: 11-26-13 22:51. Total runtime: 0:00:02.883000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2251--Rich Votes\_RF\_CV\_list.csv

Predictions found. Using cached predictions.

>>> models[13].postprocess\_scalar=.50

>>> del models[13].dfPredictions

>>> for model in models[13:]:

features\_list = (map(str,model.features.keys()))

features\_list.sort()

print "=============================================================================================================="

print "=============================================================================================================="

print "=============================================================================================================="

print "MODEL: %s SEGMENT: %s TARGET: %s " % (model.model\_name, model.segment, model.target)

print "FEATURES: %s" % features\_list

print "LEARNING ALGORITHM: %s POST-PROCESS SCALAR: %s " % (model.classifier\_name,model.postprocess\_scalar)

print "=============================================================================================================="

if settings['generate\_cv\_score'] == 'true':

#dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrnCV, dfTestCV, model.segment)

dfTrn\_Segment, dfTest\_Segment = munge.segment\_data(dfTrn, dfTest, model.segment)

dfTest\_Segment = dfTest\_Segment.merge(dfCV,on='id',how="inner")

#Vectorize each text, categorical, or boolean feature into a train and test matrix stored in model.features

features.vectors(dfTrn\_Segment, dfTest\_Segment, model.features)

#Transform or scale any numerical features and create feature vector

features.numerical(dfTrn\_Segment, dfTest\_Segment, model.features)

#Generate predictions

train.cross\_validate(model, settings, dfTrn\_Segment, dfTest\_Segment)

#Output each model's CV predictions to file (optional)

if settings['export\_cv\_predictions\_each\_model'] == 'true':

#Output individual model predictions

data\_io.save\_predictions(dfTest\_Segment,model,'CV\_list')

if settings['export\_cv\_predictions\_total'] == 'true':

if 'dfCVPredictions' not in locals():

dfCVPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfCVPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

#If there are no existing predictions already calculated AND predictions are needed then make predictions

if settings['export\_predictions\_each\_model'] == 'true' or settings['export\_predictions\_total'] == 'true':

if not hasattr(model,'dfPredictions'):

model.predict(dfTrn, dfTest)

else:

print 'Predictions found. Using cached predictions.'

#Merge each model's predictions with all test data for later export (optional)

if settings['export\_predictions\_total'] == 'true':

if 'dfTestPredictions' not in locals():

dfTestPredictions = pd.DataFrame(columns=['id'])

#If current model's predictions are for new ID's, then append the new ID's. Otherwise merge the predictions.

if len(dfTestPredictions.merge(model.dfPredictions,how='inner',on='id')) == 0:

dfTestPredictions = dfTestPredictions.append(model.dfPredictions)

elif model.target in dfTestPredictions.columns:

try:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]]).reindex\_like(dfTestPredictions)

except AssertionError:

dfTestPredictions[model.target] = pd.concat([dfTestPredictions[model.target].dropna(),

model.dfPredictions[model.target]], ignore\_index=True).reindex\_like(dfTestPredictions)

else:

dfTestPredictions = dfTestPredictions.merge(model.dfPredictions,how='left',on='id')

==============================================================================================================

==============================================================================================================

==============================================================================================================

MODEL: Rich Votes SEGMENT: Richmond TARGET: num\_views

FEATURES: ['hours\_range', 'neighborhood', 'summary\_count', 'weekend\_fg']

LEARNING ALGORITHM: RF POST-PROCESS SCALAR: 0.5

==============================================================================================================

Temporal CV started at: 11-26-13 22:54

==============================================================================================================

Error Measure: 0.669893770897

Prediction mean: 3.281863

Prediction std dev: 2.428972

Prediction max/min: 50.348876/0.105368

==============================================================================================================

Temporal CV completed at: 11-26-13 22:54. Total runtime: 0:00:02.816000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2254--Rich Votes\_RF\_CV\_list.csv

Model predictions started at: 11-26-13 22:54

==============================================================================================================

Prediction mean: 3.321674

Prediction std dev: 2.443056

Prediction max/min: 50.628178/0.107097

==============================================================================================================

Predictions completed at: 11-26-13 22:54. Total runtime: 0:00:02.801000

==============================================================================================================

Submission file saved as Submits/11-26-13\_2254--Rich Votes\_RF\_test.csv

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

#munge

for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Richmond"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(3):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

Model loaded into index 0

Model loaded into index 1

Model loaded into index 2

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-431-12e7c11f312d>", line 5, in <module>

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Richmond"].reset\_index())

File "C:\Python27\lib\site-packages\pandas\core\frame.py", line 2003, in \_\_getitem\_\_

return self.\_get\_item\_cache(key)

File "C:\Python27\lib\site-packages\pandas\core\generic.py", line 667, in \_get\_item\_cache

values = self.\_data.get(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1655, in get

\_, block = self.\_find\_block(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1935, in \_find\_block

self.\_check\_have(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1942, in \_check\_have

raise KeyError('no item named %s' % com.pprint\_thing(item))

KeyError: u'no item named Segment'

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-433-a00b641f29f8>", line 2, in <module>

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

IndexError: list index out of range

>>> ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

Model loaded into index 3

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

Model loaded into index 0

Model loaded into index 1

Model loaded into index 2

>>> for i in range(len(ensemble\_test.sub\_models)):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Richmond"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-439-12e7c11f312d>", line 5, in <module>

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Richmond"].reset\_index())

File "C:\Python27\lib\site-packages\pandas\core\frame.py", line 2003, in \_\_getitem\_\_

return self.\_get\_item\_cache(key)

File "C:\Python27\lib\site-packages\pandas\core\generic.py", line 667, in \_get\_item\_cache

values = self.\_data.get(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1655, in get

\_, block = self.\_find\_block(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1935, in \_find\_block

self.\_check\_have(item)

File "C:\Python27\lib\site-packages\pandas\core\internals.py", line 1942, in \_check\_have

raise KeyError('no item named %s' % com.pprint\_thing(item))

KeyError: u'no item named Segment'

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

#munge

for i in range(2):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Richmond"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(3):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

Model loaded into index 0

Model loaded into index 1

Model loaded into index 2

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-448-a00b641f29f8>", line 2, in <module>

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

IndexError: list index out of range

>>>

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

Model loaded into index 0

Model loaded into index 1

Model loaded into index 2

>>> for i in range(2):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Richmond"].reset\_index())

>>> dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(3):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-455-a00b641f29f8>", line 2, in <module>

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

IndexError: list index out of range

>>> range(3)

Out[456]: [0, 1, 2]

>>> ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[2])

>>> dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(3):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

>>> for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

True value for target variables successfully loaded into self.df\_true\_segment

>>>

>>> weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_comments":.3}]

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(3):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(3):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[ 0.40636881 0.10183584 0.4962559 ]]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 0.659118

Total RMSLE score: 0.659118

>>> weights

Out[473]:

[{'num\_comments': 0.7, 'num\_views': 0.40636880971905293, 'num\_votes': 0.45},

{'num\_comments': 0.3, 'num\_views': 0.1018358378343806, 'num\_votes': 0.55},

{'num\_comments': 0.3, 'num\_views': 0.49625590196381147}]

>>> weights = [{"num\_views":.0,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.7,"num\_comments":.3}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 0.663875

Total RMSLE score: 0.663875

>>> weights = [{"num\_views":.0,"num\_votes":.45,"num\_comments":.7},{"num\_views":.4,"num\_votes":.55,"num\_comments":.3},{"num\_views":.6,"num\_comments":.3}]

>>> weights = [{"num\_views":.0,"num\_votes":.45,"num\_comments":.7},{"num\_views":.4,"num\_votes":.55,"num\_comments":.3},{"num\_views":.6,"num\_comments":.3}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 0.664322

Total RMSLE score: 0.664322

>>> weights = [{"num\_views":.0,"num\_votes":.45,"num\_comments":.7},{"num\_views":.2,"num\_votes":.55,"num\_comments":.3},{"num\_views":.8,"num\_comments":.3}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 0.664625

Total RMSLE score: 0.664625

>>> weights = [{"num\_views":.0,"num\_votes":.45,"num\_comments":.32},{"num\_views":.2,"num\_votes":.55,"num\_comments":.3},{"num\_views":.68,"num\_comments":.3}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 0.684660

Total RMSLE score: 0.684660

>>> weights = [{"num\_views":.0,"num\_votes":.45,"num\_comments":.32},{"num\_views":.32,"num\_votes":.55,"num\_comments":.3},{"num\_views":.68,"num\_comments":.3}]

>>> ensemble\_test.create\_ensemble\_segment([0,1,2],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 0.663868

Total RMSLE score: 0.663868

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

#munge

for i in range(2):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Chicago\_Other"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(3):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

Model loaded into index 0

Model loaded into index 1

Model loaded into index 2

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-507-a00b641f29f8>", line 2, in <module>

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

IndexError: list index out of range

>>>

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

#ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

#munge

for i in range(2):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Chicago\_Other"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(2):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

Model loaded into index 0

Model loaded into index 1

>>>

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

#Set weights

#Remote\_API: weights = [{"num\_views":.16,"num\_votes":.3,"num\_comments":.9},{"num\_views":.84,"num\_votes":.7,"num\_comments":.1}]

#Richmond: weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_comments":.3}]

#Oakland weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.7},{"num\_views":.8,"num\_votes":.9,"num\_comments":.3}]

weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.6},{"num\_views":.8,"num\_votes":.9,"num\_comments":.4}]

True value for target variables successfully loaded into self.df\_true\_segment

>>>

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(3):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(3):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-529-cb4ad3ddff06>", line 5, in <module>

,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

IndexError: list index out of range

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[ 0.83087286 0.30992011]]

>>> ensemble\_test.create\_ensemble\_segment([0,1],weights)

#Score the ensemble

#ensemble\_CV.score\_rmsle(ensemble\_CV.sub\_models\_segment[0], df\_true=ensemble\_CV.df\_true\_segment)

ensemble\_test.score\_rmsle(ensemble\_test.df\_ensemble\_segment, df\_true=ensemble\_test.df\_true\_segment)

RMSLE score for num\_views: 0.700647

Total RMSLE score: 0.700647

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_comments"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

#ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

#munge

for i in range(2):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Chicago\_Other"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(2):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

#Set weights

#Remote\_API: weights = [{"num\_views":.16,"num\_votes":.3,"num\_comments":.9},{"num\_views":.84,"num\_votes":.7,"num\_comments":.1}]

#Richmond: weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_comments":.3}]

#Oakland weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.7},{"num\_views":.8,"num\_votes":.9,"num\_comments":.3}]

weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.6},{"num\_views":.8,"num\_votes":.9,"num\_comments":.4}]

Model loaded into index 0

Model loaded into index 1

True value for target variables successfully loaded into self.df\_true\_segment

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[ 0.32825437 1.03945282]]

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

#ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

#munge

for i in range(2):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Oakland"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(2):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

#Set weights

#Remote\_API: weights = [{"num\_views":.16,"num\_votes":.3,"num\_comments":.9},{"num\_views":.84,"num\_votes":.7,"num\_comments":.1}]

#Richmond: weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_comments":.3}]

#Oakland weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.7},{"num\_views":.8,"num\_votes":.9,"num\_comments":.3}]

weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.6},{"num\_views":.8,"num\_votes":.9,"num\_comments":.4}]

Model loaded into index 0

Model loaded into index 1

True value for target variables successfully loaded into self.df\_true\_segment

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

Traceback (most recent call last):

File "C:\Python27\lib\site-packages\IPython\core\interactiveshell.py", line 2745, in run\_code

exec code\_obj in self.user\_global\_ns, self.user\_ns

File "<ipython-input-578-428dd84d6ab1>", line 8, in <module>

clf.fit(train\_2,target\_2)

File "C:\Python27\lib\site-packages\sklearn\linear\_model\base.py", line 379, in fit

linalg.lstsq(X, y)

File "C:\Python27\lib\site-packages\scipy\linalg\basic.py", line 535, in lstsq

work = gelss(a1, b1, lwork=-1)[4]

ValueError: failed to create intent(cache|hide)|optional array-- must have defined dimensions but got (0,)

>>> train\_2

Out[579]: array([], shape=(0L, 2L), dtype=float64)

>>> reload(ensembles);ensemble\_test = ensembles.EnsembleAvg(targets=["num\_views"],id="id")

ensemble\_test.load\_models\_csv(filepath="Submits/BryanModel-Updated.csv")

ensemble\_test.load\_models\_csv(filepath="Submits/ridge\_38\_test.csv")

#ensemble\_test.load\_models\_csv(filepath="Submits/rich\_views2\_knnnbr8\_sgd.csv")

#munge

for i in range(2):

ensemble\_test.sub\_models[i] = ensemble\_test.sub\_models[i].merge(dfCV.ix[:,['id']],on='id',how="inner")

#Parse segments

ensemble\_test.sub\_models\_segment.append\

(ensemble\_test.sub\_models[i][ensemble\_test.sub\_models[i]['Segment'] == "Oakland\_Other"].reset\_index())

dfSegTest = dfCV.merge(ensemble\_test.sub\_models\_segment[0].ix[:,['id']],on='id',how="inner").dropna()

for i in range(2):

ensemble\_test.sub\_models\_segment[i] = ensemble\_test.sub\_models\_segment[i].merge(dfSegTest.ix[:,['id']],on='id',how="inner")

#Transform CV targets back to normal

for target in ensemble\_test.targets:

dfSegTest[target]=np.exp(dfSegTest[target])-1

#Load groundtruth values for CV

ensemble\_test.load\_df\_true\_segment(dfSegTest)

#Sort all dataframes by ID for easy comparison

ensemble\_test.sort\_dataframes("id")

#Transform predictions to log space for averaging

ensemble\_test.transform\_targets\_log()

#Set weights

#Remote\_API: weights = [{"num\_views":.16,"num\_votes":.3,"num\_comments":.9},{"num\_views":.84,"num\_votes":.7,"num\_comments":.1}]

#Richmond: weights = [{"num\_views":.7,"num\_votes":.45,"num\_comments":.7},{"num\_views":.3,"num\_votes":.55,"num\_comments":.3},{"num\_views":.4,"num\_comments":.3}]

#Oakland weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.7},{"num\_views":.8,"num\_votes":.9,"num\_comments":.3}]

weights = [{"num\_views":.2,"num\_votes":.1,"num\_comments":.6},{"num\_views":.8,"num\_votes":.9,"num\_comments":.4}]

Model loaded into index 0

Model loaded into index 1

True value for target variables successfully loaded into self.df\_true\_segment

>>>

>>>

>>> for target\_label in ensemble\_test.targets:

clf.fit\_intercept=False

train\_2 = np.hstack((ensemble\_test.sub\_models\_segment[0].ix[:,[target\_label]].as\_matrix(),

ensemble\_test.sub\_models\_segment[1].ix[:,[target\_label]].as\_matrix()

#,ensemble\_test.sub\_models\_segment[2].ix[:,[target\_label]].as\_matrix()

))

target\_2 = ensemble\_test.df\_true\_segment.ix[:,[target\_label]].as\_matrix()

clf.fit(train\_2,target\_2)

try:

for i in range(2):

weights[i][target\_label]=clf.coef\_[i]

except:

for i in range(2):

weights[i][target\_label]=clf.coef\_[0][i]

print clf.coef\_

[[ 0.58900098 0.487576 ]]