Appendix - Submitting the Prediction

September 24, 2016

Now that the two pieces of the puzzle are complete - predicting which houses own EVs, and predicting when those houses are charging - it's possible to submit predictions for the test data. The approach is simple - first use the auto-generated 2D KNN classifier to pick out which houses own EVs, then use a trained Markov Model to compute probabilities of charging at each timestep for the houses which are reported to own EVs. All the data in the training files will be used to train the classifiers and Markov Model, while the uncertainties will be estimated using the results from earlier. It is assumed the usages in the test files follow the same distribution as the usages in the train files.

The expected accuracy of the submission is given by the accuracies found in parts II and III.
The accuracy of which houses own EVs is given in part III, which is 84% (unc of 2%).
The accuracy of when a house is charging is given in part II, which is 95.56% (unc of .04%).

In [1]:
```
%matplotlib inline

%load_ext autoreload
%autoreload
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as pl
import matplotlib.colors as colors
import numpy as np
import scipy
import scipy.stats as stats
import scipy.stats.mstats as mstats
import scipy.interpolate as interpolate
import sklearn.neighbors as neighbors
import sklearn
import random

import util
import markov


train_usages = pd.read_csv('./data/EV_train.csv', index_col='House ID').transpose().dropna()
train_when_charging = pd.read_csv('./data/EV_train_labels.csv', index_col='House ID').transpose
test_usages = pd.read_csv('./data/EV_test.csv', index_col='House ID').transpose()



train_usages.index = range(len(train_usages.index))
train_when_charging.index = range(len(train_when_charging.index))

original_index = test_usages.index
test_usages.index = range(len(test_usages.index))
```

```
        train_houses = train_usages.columns
        test_houses = test_usages.columns

In [2]: train_derivatives = util.derivatives(train_usages)

        normed_train_usages = util.normalize_usages(train_usages)
        normed_train_derivatives = util.derivatives(normed_train_usages)

        train_house_stats = pd.DataFrame(index = train_houses)
        train_house_stats['owns_ev'] = train_when_charging.sum() > 0
        train_house_stats['d1_k5'] = train_derivatives[1].apply(np.abs).apply(lambda x: np.power(x, 5))

In [3]: test_derivatives = util.derivatives(test_usages.dropna())

        normed_test_usages = util.normalize_usages(test_usages.dropna())
        normed_test_derivatives = util.derivatives(normed_test_usages)

        test_house_stats = pd.DataFrame(index = test_houses)
        test_house_stats['d1_k5'] = test_derivatives[1].apply(np.abs).apply(lambda x: np.power(x, 5)).ap

In [4]: from sklearn.neighbors import KNeighborsClassifier

        train_X = train_house_stats['d1_k5'].values
        train_y = train_house_stats['owns_ev']

        clf = KNeighborsClassifier()
        clf.fit(np.array([train_X]).T,train_y)

        test_X = test_house_stats['d1_k5'].values
        test_y = clf.predict(np.array([test_X]).T)
        owns_ev_pred = pd.Series(test_y.astype('int'), index=test_usages.columns).astype('float')

In [5]: train_with_ev = train_houses[train_when_charging.sum() > 0]
        m = markov.EVMarkovModel(train_usages[train_with_ev], train_when_charging[train_with_ev])

In [6]: m.train(train_with_ev)

In []: test_with_ev = owns_ev_pred[owns_ev_pred > 0].index

In []: when_charging_prob  = pd.DataFrame(0, index=test_usages.index, columns = test_usages.columns)
       for i,house in enumerate(test_with_ev):
           #print "house %i out of %i" % (i+1, len(test_with_ev))
           when_charging_prob[house] = m.run_usage(test_usages[house])

In []: when_charging_prob.index = original_index
       when_charging_prob.T.to_csv('./data/submission.csv')
```