porto insurance car

May 9, 2018

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
np.random.seed(10)
from tensorflow import set_random_seed
set_random_seed(15)

from keras.models import Sequential
from keras.layers import Dense, Activation, Merge, Reshape, Dropout
from keras.layers.embeddings import Embedding

from sklearn.model_selection import StratifiedKFold
```

```
/home/nikhil/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversio
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.
```

In [3]:
```
df_train = pd.read_csv('input/portotrain.csv')
df_test = pd.read_csv('input/portotest.csv')

X_train, y_train = df_train.iloc[:,2:], df_train.target
X_test = df_test.iloc[:,1:]

cols_use = [c for c in X_train.columns if (not c.startswith('ps_calc_'))]

X_train = X_train[cols_use]
X_test = X_test[cols_use]

col_vals_dict = {c: list(X_train[c].unique()) for c in X_train.columns if c.endswith('_c

embed_cols = []
for c in col_vals_dict:
    if len(col_vals_dict[c])>2:
        embed_cols.append(c)
        print(c + ': %d values' % len(col_vals_dict[c])) #look at value counts to know t

print('\n')
```

```
ps_ind_02_cat: 5 values
ps_ind_04_cat: 3 values
ps_ind_05_cat: 8 values
ps_car_01_cat: 13 values
ps_car_02_cat: 3 values
ps_car_03_cat: 3 values
ps_car_04_cat: 10 values
ps_car_05_cat: 3 values
ps_car_06_cat: 18 values
ps_car_07_cat: 3 values
ps_car_09_cat: 6 values
ps_car_10_cat: 3 values
ps_car_11_cat: 104 values
```

```
In [4]: def build_embedding_network():

            models = []

            model_ps_ind_02_cat = Sequential()
            model_ps_ind_02_cat.add(Embedding(5, 3, input_length=1))
            model_ps_ind_02_cat.add(Reshape(target_shape=(3,)))
            models.append(model_ps_ind_02_cat)

            model_ps_ind_04_cat = Sequential()
            model_ps_ind_04_cat.add(Embedding(3, 2, input_length=1))
            model_ps_ind_04_cat.add(Reshape(target_shape=(2,)))
            models.append(model_ps_ind_04_cat)

            model_ps_ind_05_cat = Sequential()
            model_ps_ind_05_cat.add(Embedding(8, 5, input_length=1))
            model_ps_ind_05_cat.add(Reshape(target_shape=(5,)))
            models.append(model_ps_ind_05_cat)

            model_ps_car_01_cat = Sequential()
            model_ps_car_01_cat.add(Embedding(13, 7, input_length=1))
            model_ps_car_01_cat.add(Reshape(target_shape=(7,)))
            models.append(model_ps_car_01_cat)

            model_ps_car_02_cat = Sequential()
            model_ps_car_02_cat.add(Embedding(3, 2, input_length=1))
            model_ps_car_02_cat.add(Reshape(target_shape=(2,)))
            models.append(model_ps_car_02_cat)

            model_ps_car_03_cat = Sequential()
            model_ps_car_03_cat.add(Embedding(3, 2, input_length=1))
```

```
model_ps_car_03_cat.add(Reshape(target_shape=(2,)))
models.append(model_ps_car_03_cat)

model_ps_car_04_cat = Sequential()
model_ps_car_04_cat.add(Embedding(10, 5, input_length=1))
model_ps_car_04_cat.add(Reshape(target_shape=(5,)))
models.append(model_ps_car_04_cat)

model_ps_car_05_cat = Sequential()
model_ps_car_05_cat.add(Embedding(3, 2, input_length=1))
model_ps_car_05_cat.add(Reshape(target_shape=(2,)))
models.append(model_ps_car_05_cat)

model_ps_car_06_cat = Sequential()
model_ps_car_06_cat.add(Embedding(18, 8, input_length=1))
model_ps_car_06_cat.add(Reshape(target_shape=(8,)))
models.append(model_ps_car_06_cat)

model_ps_car_07_cat = Sequential()
model_ps_car_07_cat.add(Embedding(3, 2, input_length=1))
model_ps_car_07_cat.add(Reshape(target_shape=(2,)))
models.append(model_ps_car_07_cat)

model_ps_car_09_cat = Sequential()
model_ps_car_09_cat.add(Embedding(6, 3, input_length=1))
model_ps_car_09_cat.add(Reshape(target_shape=(3,)))
models.append(model_ps_car_09_cat)

model_ps_car_10_cat = Sequential()
model_ps_car_10_cat.add(Embedding(3, 2, input_length=1))
model_ps_car_10_cat.add(Reshape(target_shape=(2,)))
models.append(model_ps_car_10_cat)

model_ps_car_11_cat = Sequential()
model_ps_car_11_cat.add(Embedding(104, 10, input_length=1))
model_ps_car_11_cat.add(Reshape(target_shape=(10,)))
models.append(model_ps_car_11_cat)

model_rest = Sequential()
model_rest.add(Dense(16, input_dim=24))
models.append(model_rest)

model = Sequential()
model.add(Merge(models, mode='concat'))
model.add(Dense(80))
model.add(Activation('relu'))
model.add(Dropout(.35))
model.add(Dense(20))
```

```
    model.add(Activation('relu'))
    model.add(Dropout(.15))
    model.add(Dense(10))
    model.add(Activation('relu'))
    model.add(Dropout(.15))
    model.add(Dense(1))
    model.add(Activation('sigmoid'))

    model.compile(loss='binary_crossentropy', optimizer='adam')

    return model
```

In [5]:
```
def preproc(X_train, X_val, X_test):

    input_list_train = []
    input_list_val = []
    input_list_test = []

    #the cols to be embedded: rescaling to range [0, # values)
    for c in embed_cols:
        raw_vals = np.unique(X_train[c])
        val_map = {}
        for i in range(len(raw_vals)):
            val_map[raw_vals[i]] = i
        input_list_train.append(X_train[c].map(val_map).values)
        input_list_val.append(X_val[c].map(val_map).fillna(0).values)
        input_list_test.append(X_test[c].map(val_map).fillna(0).values)

    #the rest of the columns
    other_cols = [c for c in X_train.columns if (not c in embed_cols)]
    input_list_train.append(X_train[other_cols].values)
    input_list_val.append(X_val[other_cols].values)
    input_list_test.append(X_test[other_cols].values)

    return input_list_train, input_list_val, input_list_test
```

In [6]:
```
#https://www.kaggle.com/tezdhar/faster-gini-calculation
def ginic(actual, pred):
    n = len(actual)
    a_s = actual[np.argsort(pred)]
    a_c = a_s.cumsum()
    giniSum = a_c.sum() / a_c[-1] - (n + 1) / 2.0
    return giniSum / n
```

In [7]:
```
def gini_normalizedc(a, p):
    return ginic(a, p) / ginic(a, a)
```

In [8]:
```
K = 8
runs_per_fold = 3
```

```
n_epochs = 15

cv_ginis = []
full_val_preds = np.zeros(np.shape(X_train)[0])
y_preds = np.zeros((np.shape(X_test)[0],K))

kfold = StratifiedKFold(n_splits = K,
                            random_state = 231,
                            shuffle = True)

for i, (f_ind, outf_ind) in enumerate(kfold.split(X_train, y_train)):

    X_train_f, X_val_f = X_train.loc[f_ind].copy(), X_train.loc[outf_ind].copy()
    y_train_f, y_val_f = y_train[f_ind], y_train[outf_ind]

    X_test_f = X_test.copy()

    #upsampling adapted from kernel:
    #https://www.kaggle.com/ogrellier/xgb-classifier-upsampling-lb-0-283
    pos = (pd.Series(y_train_f == 1))

    # Add positive examples
    X_train_f = pd.concat([X_train_f, X_train_f.loc[pos]], axis=0)
    y_train_f = pd.concat([y_train_f, y_train_f.loc[pos]], axis=0)

    # Shuffle data
    idx = np.arange(len(X_train_f))
    np.random.shuffle(idx)
    X_train_f = X_train_f.iloc[idx]
    y_train_f = y_train_f.iloc[idx]

    #preprocessing
    proc_X_train_f, proc_X_val_f, proc_X_test_f = preproc(X_train_f, X_val_f, X_test_f)

    #track oof prediction for cv scores
    val_preds = 0

    for j in range(runs_per_fold):

        NN = build_embedding_network()
        NN.fit(proc_X_train_f, y_train_f.values, epochs=n_epochs, batch_size=4096, verbo

        val_preds += NN.predict(proc_X_val_f)[:,0] / runs_per_fold
        y_preds[:,i] += NN.predict(proc_X_test_f)[:,0] / runs_per_fold

    full_val_preds[outf_ind] += val_preds

    cv_gini = gini_normalizedc(y_val_f.values, val_preds)
```

```
            cv_ginis.append(cv_gini)
            print ('\nFold %i prediction cv gini: %.5f\n' %(i,cv_gini))

        print('Mean out of fold gini: %.5f' % np.mean(cv_ginis))
        print('Full validation gini: %.5f' % gini_normalizedc(y_train.values, full_val_preds))

        y_pred_final = np.mean(y_preds, axis=1)

        df_sub = pd.DataFrame({'id' : df_test.id,
                               'target' : y_pred_final},
                               columns = ['id','target'])
        df_sub.to_csv('NN_EntityEmbed_10fold-sub.csv', index=False)

        pd.DataFrame(full_val_preds).to_csv('NN_EntityEmbed_10fold-val_preds.csv',index=False)
```

```
/home/nikhil/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:76: UserWarning: The `M
```

```
Fold 0 prediction cv gini: 0.28934


Fold 1 prediction cv gini: 0.25602


Fold 2 prediction cv gini: 0.28685


Fold 3 prediction cv gini: 0.27863


Fold 4 prediction cv gini: 0.26646


Fold 5 prediction cv gini: 0.28944


Fold 6 prediction cv gini: 0.27869


Fold 7 prediction cv gini: 0.27919

Mean out of fold gini: 0.27808
Full validation gini: 0.27713
```