lab9

November 1, 2016

1 EE 379K - Data Science Lab

2 Lab 9

3 Wenyang Fu and Rohan Nagar

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        %matplotlib inline

        from sklearn.model_selection import (cross_val_score, train_test_split,
                                             GridSearchCV, RandomizedSearchCV)
        from sklearn.preprocessing import Imputer

        %load_ext autoreload
        %autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

4 Question 1

```
In [3]: train_inclass = pd.read_csv('data/train_inclass.csv')
        test_inclass = pd.read_csv('data/test_inclass.csv')
```

```
In [4]: print(train_inclass['F3'].describe())
        print()
        print(train_inclass['F23'].describe())
```

```
count    49998.000000
mean         5.272668
std        224.530270
min         -0.372758
25%          0.038775
50%          0.186073
75%          0.563830
max      29110.040580
Name: F3, dtype: float64

count    49998.000000
```

```
mean         5.273124
std        224.529521
min          0.000000
25%          0.030389
50%          0.154672
75%          0.555344
max      29110.000000
Name: F23, dtype: float64
```

```
In [5]: difference = abs(train_inclass['F23'] - train_inclass['F3'])

        print('Mean of added noise: {}'.format(difference.mean()))
        print('Variance of added noise: {}'.format(difference.std()**2))
```

```
Mean of added noise: 0.07956681166550796
Variance of added noise: 0.003602256845676487
```

5 Question 2

As we explained in lecture, the InClass competition data came from https://www.kaggle.com/c/GiveMeSomeCredit

You can now double the training data and you have a new validation set using the leaderboard of this Kaggle competition.

You can also look at 'Data Dictionary.xls' to find what each of the features are exactly.

Train your models on the additional data and validate using the private LB of that competition. How do the optimal hyperparameters parameters change ? Are the winning XGB parameters still better? Report your Private LB score and include a screenshot of your submissions in your report.

```
In [6]: def write_preds(filename, preds):
            with open(filename, 'w') as f:
                f.write('Id,Probability\n')
                for num, pred in zip(range(1,101504), preds):
                    f.write('{},{}\n'.format(num, pred))
```

```
In [7]: train = pd.read_csv('data/cs-training.csv', index_col=0)
        test = pd.read_csv('data/cs-test.csv', index_col=0)

        SEED = 42
```

```
In [8]: # Drop dependent variable in test
        test = test.drop(['SeriousDlqin2yrs'], axis=1)
```

```
In [9]: # Fill missing with mean
        train = train.fillna(train.mean())
        test = test.fillna(test.mean())
```

```
In [10]: # Seperate dependent and independent
         X_train = train.drop(['SeriousDlqin2yrs'], axis=1)
         y_train = train['SeriousDlqin2yrs']
```

```
In [11]: from sklearn.preprocessing import FunctionTransformer

         # Perform a log transform on the data
         transformer = FunctionTransformer(np.log1p)
         X_train = transformer.transform(X_train)
         test = transformer.transform(test)
         X_test = test
```

```
In [12]: import xgboost as xgb


         # XGB, Raymond Wen's parameters
         # Raymond Wen's parameters
         params = {
             'n_estimators': 1000,
             'eta': 0.01,
             'max_depth': 4,
             'min_child_weight': 5,
             'subsample': 0.4,
             'gamma': 0.8,
             'colsample_bytree': 0.4,
             'lambda': 0.93,
             'alpha': 0.5,
             'eval_metric': 'auc',
             'objective': 'binary:logistic',
             # Increase this number if you have more cores.
             # Otherwise, remove it and it will default
             # to the maxium number.
             'nthread': 4,
             'booster': 'gbtree',
             'tree_method': 'exact',
             'silent': 1,
             'seed': SEED
         }
```

```
/Users/rohannagar/anaconda/lib/python3.5/site-packages/sklearn/cross_validation.py:44: DeprecationWarnin
  "This module will be removed in 0.20.", DeprecationWarning)
```

```
In [9]: # check model CV scores
        num_boost_round = int(params['n_estimators'])
        del params['n_estimators']
        dtrain = xgb.DMatrix(X_train, label=y_train)
        dtest = xgb.DMatrix(X_test)

        score_history = xgb.cv(params, dtrain, num_boost_round,
                               nfold=5, stratified=True,
                               early_stopping_rounds=250,
                               verbose_eval=500)

        # Only use scores from the final boosting round since that's the one
        # that performed the best.
        mean_final_round = score_history.tail(1).iloc[0, 0]
        std_final_round = score_history.tail(1).iloc[0, 1]
```

```
[0]        train-auc:0.747949+0.0466865        test-auc:0.744769+0.0521322
[500]        train-auc:0.868464+0.000509831        test-auc:0.864688+0.00180149
```

```
In [10]: print("\tMean Score: {0}\n".format(mean_final_round))
         print("\tStd Dev: {0}\n\n".format(std_final_round))
```

```
Mean Score: 0.8663118000000001

        Std Dev: 0.0018126794973187925
```

```
In [11]: # As of version 0.6, XGBoost returns a dataframe of the following form:
         # boosting iter | mean_test_err | mean_test_std | mean_train_err | mean_train_std
         # boost iter 1 mean_test_iter1 | mean_test_std1 | ... | ...
         # boost iter 2 mean_test_iter2 | mean_test_std2 | ... | ...
         # ...
         # boost iter n_estimators

         xg_booster = xgb.train(params, dtrain, num_boost_round)
         preds = xg_booster.predict(dtest)
         write_preds('submissions/xgb_raymond_{}.csv'.format(SEED), preds)
```

5.1 Raymond's hyperparameters achieved a private leaderboard score of $ 0.867641$

```
In [12]: import os
         import logging
         # Let OpenMP use 4 threads to evaluate models - may run into errors
         # if this is not set. Should be set before hyperopt import.
         os.environ['OMP_NUM_THREADS'] = '4'

         import hyperopt
         from hyperopt import STATUS_OK, Trials, fmin, hp, tpe
```

```
In [13]: logging.basicConfig(filename="logs/hyperopt_xgb.log", level=logging.INFO)
```

```
In [15]: # -----------------------------------------------------
         #                       HYPEROPT
         # -----------------------------------------------------

         def score(params):
             logging.info("Training with params: ")
             logging.info(params)
             # Delete 'n_estimators' because it's only a constructor param
             # when you're using  XGB's sklearn API.
             # Instead, we have to save 'n_estimators' (# of boosting rounds)
             # to xgb.cv().
             num_boost_round = int(params['n_estimators'])
             del params['n_estimators']
             dtrain = xgb.DMatrix(X_train, label=y_train)
             # As of version 0.6, XGBoost returns a dataframe of the following form:
             # boosting iter | mean_test_err | mean_test_std | mean_train_err | mean_train_std
             # boost iter 1 mean_test_iter1 | mean_test_std1 | ... | ...
             # boost iter 2 mean_test_iter2 | mean_test_std2 | ... | ...
             # ...
             # boost iter n_estimators

             score_history = xgb.cv(params, dtrain, num_boost_round,
                                    nfold=5, stratified=True,
                                    early_stopping_rounds=250,
                                    verbose_eval=500)
             # Only use scores from the final boosting round since that's the one
             # that performed the best.
             mean_final_round = score_history.tail(1).iloc[0, 0]
             std_final_round = score_history.tail(1).iloc[0, 1]
             logging.info("\tMean Score: {0}\n".format(mean_final_round))
```

```python
    logging.info("\tStd Dev: {0}\n\n".format(std_final_round))
    # score() needs to return the loss (1 - score)
    # since optimize() should be finding the minimum, and AUC
    # naturally finds the maximum.
    loss = 1 - mean_final_round
    return {'loss': loss, 'status': STATUS_OK}


def optimize(
    # trials,
        random_state=SEED):
    """
    This is the optimization function that given a space (space here) of
    hyperparameters and a scoring function (score here),
    finds the best hyperparameters.
    """

    space = {
        'n_estimators': hp.choice('n_estimators', [1000, 1100]),
        'eta': hp.quniform('eta', 0.01, 0.1, 0.025),
        'max_depth': hp.choice('max_depth', [4, 5, 7, 9, 17]),
        'min_child_weight': hp.choice('min_child_weight', [3, 5, 7]),
        'subsample': hp.choice('subsample', [0.4, 0.6, 0.8]),
        'gamma': hp.choice('gamma', [0.3, 0.4]),
        'colsample_bytree': hp.quniform('colsample_bytree', 0.4, 0.7, 0.1),
        'lambda': hp.choice('lambda', [0.01, 0.1, 0.9, 1.0]),
        'alpha': hp.choice('alpha', [0, 0.1, 0.5, 1.0]),
        'eval_metric': 'auc',
        'objective': 'binary:logistic',
        # Increase this number if you have more cores.
        # Otherwise, remove it and it will default
        # to the maxium number.
        'nthread': 4,
        'booster': 'gbtree',
        'tree_method': 'exact',
        'silent': 1,
        'seed': random_state
    }

    # Use the fmin function from Hyperopt to find the best hyperparameters
    best = fmin(score, space, algo=tpe.suggest,
                # trials=trials,
                max_evals=250)
    return best


best_hyperparams = optimize(
    # trials
)
print("The best hyperparameters are: ", "\n")
print(best_hyperparams)
```

```
[0]	train-auc:0.771441+0.0304771	test-auc:0.766047+0.0346485
[500]	train-auc:0.876783+0.000469591	test-auc:0.866306+0.00182939
```

```
[0]	train-auc:0.7669+0.0394836	test-auc:0.76161+0.0419847
[0]	train-auc:0.762717+0.0412081	test-auc:0.759824+0.0446267
[0]	train-auc:0.773133+0.0366549	test-auc:0.764708+0.0403129
[0]	train-auc:0.778639+0.0288682	test-auc:0.771273+0.0324843
[500]	train-auc:0.900989+0.000451365	test-auc:0.86548+0.00206645
[0]	train-auc:0.758615+0.0436854	test-auc:0.755917+0.0460673
[0]	train-auc:0.778721+0.0329956	test-auc:0.761332+0.0414229
[0]	train-auc:0.800929+0.025089	test-auc:0.730082+0.0452533
[0]	train-auc:0.756091+0.0431604	test-auc:0.752678+0.0475648
[0]	train-auc:0.769693+0.030857	test-auc:0.764447+0.0348932
[500]	train-auc:0.875552+0.000536211	test-auc:0.866435+0.00188262
[0]	train-auc:0.787605+0.03528	test-auc:0.768254+0.0419516
[0]	train-auc:0.776661+0.0376994	test-auc:0.767846+0.0428384
[0]	train-auc:0.755086+0.0446986	test-auc:0.752432+0.0469662
[500]	train-auc:0.881635+0.000709419	test-auc:0.866173+0.00224314
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.763539+0.0435745	test-auc:0.758604+0.0463404
[500]	train-auc:0.882679+0.0007133	test-auc:0.866585+0.0018844
[0]	train-auc:0.775558+0.0285237	test-auc:0.770177+0.0317029
[0]	train-auc:0.77713+0.0284574	test-auc:0.771147+0.0322218
[500]	train-auc:0.893024+0.000602726	test-auc:0.865761+0.00234567
[0]	train-auc:0.785937+0.0361956	test-auc:0.764466+0.04228
[0]	train-auc:0.752636+0.0421576	test-auc:0.749351+0.0472248
[0]	train-auc:0.752061+0.044355	test-auc:0.749117+0.0466121
[0]	train-auc:0.76853+0.0407507	test-auc:0.761894+0.0452379
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.760751+0.0435294	test-auc:0.757672+0.0475081
[500]	train-auc:0.882013+0.000677385	test-auc:0.86647+0.0018747
[0]	train-auc:0.76303+0.0411598	test-auc:0.757849+0.0445284
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.761165+0.0431798	test-auc:0.757668+0.0474485
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.768132+0.0389637	test-auc:0.758342+0.046246
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.760511+0.0433667	test-auc:0.75783+0.047517
[500]	train-auc:0.882074+0.000706871	test-auc:0.866504+0.00191711
[0]	train-auc:0.767084+0.0402917	test-auc:0.761037+0.0440676
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.755598+0.0439696	test-auc:0.752513+0.0471887
[500]	train-auc:0.875667+0.000537976	test-auc:0.866289+0.00189119
[1000]	train-auc:0.881881+0.000601151	test-auc:0.866363+0.00209384
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.770782+0.0374382	test-auc:0.764947+0.0409292
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.778953+0.0297636	test-auc:0.771732+0.0336069
[500]	train-auc:0.882128+0.000577257	test-auc:0.866511+0.00202866
[0]	train-auc:0.5+0	test-auc:0.5+0
```

```
[0]	train-auc:0.763505+0.043589	test-auc:0.758493+0.0462579
[500]	train-auc:0.881775+0.000600932	test-auc:0.866575+0.00193755
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.763505+0.043589	test-auc:0.758493+0.0462579
[500]	train-auc:0.881775+0.000600932	test-auc:0.866575+0.00193755
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.754393+0.0445669	test-auc:0.752324+0.0478946
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.780127+0.0349547	test-auc:0.765172+0.0407941
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.76311+0.0434524	test-auc:0.758846+0.0464165
[500]	train-auc:0.882042+0.000661047	test-auc:0.866633+0.00186998
[0]	train-auc:0.760719+0.0436015	test-auc:0.757942+0.047639
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.762684+0.0411832	test-auc:0.759936+0.0446773
[0]	train-auc:0.763108+0.0434704	test-auc:0.758842+0.046409
[500]	train-auc:0.882985+0.00066513	test-auc:0.866593+0.00187552
[0]	train-auc:0.763108+0.0434704	test-auc:0.758842+0.046409
[500]	train-auc:0.882985+0.00066513	test-auc:0.866593+0.00187552
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.763108+0.0434704	test-auc:0.758842+0.046409
[500]	train-auc:0.882985+0.00066513	test-auc:0.866593+0.00187552
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.763108+0.0434704	test-auc:0.758842+0.046409
[500]	train-auc:0.882985+0.00066513	test-auc:0.866593+0.00187552
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.763108+0.0434704	test-auc:0.758842+0.046409
[500]	train-auc:0.882985+0.00066513	test-auc:0.866593+0.00187552
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.777353+0.0358651	test-auc:0.766101+0.0407284
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.758245+0.0465703	test-auc:0.755692+0.0489982
[500]	train-auc:0.881623+0.000589681	test-auc:0.866511+0.00195397
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.760719+0.0436015	test-auc:0.757942+0.047639
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.759522+0.041753	test-auc:0.754926+0.0455971
[500]	train-auc:0.881847+0.000653877	test-auc:0.866436+0.00213884
[0]	train-auc:0.752594+0.0475628	test-auc:0.749975+0.0501285
[500]	train-auc:0.875307+0.000533532	test-auc:0.86638+0.00185262
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.763417+0.0438047	test-auc:0.759608+0.047114
```

```
[500]   train-auc:0.882645+0.000673949   test-auc:0.866432+0.00180865
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.754959+0.0439946    test-auc:0.75147+0.0472519
[500]   train-auc:0.87608+0.000573683   test-auc:0.866341+0.00181525
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.760555+0.0435942    test-auc:0.755476+0.0491714
[0]     train-auc:0.763474+0.0436002    test-auc:0.758573+0.0462787
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.754959+0.0439945    test-auc:0.751489+0.0472597
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.816456+0.0196411    test-auc:0.711439+0.0476554
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.754959+0.0439946    test-auc:0.75147+0.0472519
[500]   train-auc:0.875412+0.000609402   test-auc:0.866337+0.00188633
[0]     train-auc:0.761673+0.0439665    test-auc:0.758566+0.0462544
[0]     train-auc:0.75509+0.0419859    test-auc:0.751636+0.0461888
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.765478+0.0415685    test-auc:0.761118+0.0432056
[500]   train-auc:0.882771+0.000540355   test-auc:0.866557+0.00188783
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.776316+0.0361495    test-auc:0.766517+0.0410733
[0]     train-auc:0.763108+0.0434704    test-auc:0.758842+0.046409
[0]     train-auc:0.747235+0.0478879    test-auc:0.745245+0.0509949
[500]   train-auc:0.874397+0.000600934   test-auc:0.866306+0.00183929
[1000]  train-auc:0.880345+0.000644882   test-auc:0.866419+0.00201833
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.760103+0.0443194    test-auc:0.756384+0.0477751
[0]     train-auc:0.7631+0.043451    test-auc:0.758813+0.0464074
[500]   train-auc:0.881461+0.000629381   test-auc:0.866529+0.00196688
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.773724+0.0290227    test-auc:0.768064+0.0316364
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.751269+0.0455283    test-auc:0.749284+0.0488604
[500]   train-auc:0.881558+0.000518602   test-auc:0.866408+0.00227421
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.776262+0.036304    test-auc:0.76637+0.0407023
[0]     train-auc:0.768882+0.0383169    test-auc:0.762418+0.0411155
[0]     train-auc:0.5+0     test-auc:0.5+0
[0]     train-auc:0.76659+0.038973    test-auc:0.761855+0.0416062
[500]   train-auc:0.895728+0.000718801   test-auc:0.866175+0.00228806
[0]     train-auc:0.5+0     test-auc:0.5+0
```

```
[0]	train-auc:0.77373+0.0372875	test-auc:0.766155+0.0417738
[500]	train-auc:0.896431+0.000662105	test-auc:0.866047+0.00220631
[0]	train-auc:0.758738+0.0449404	test-auc:0.754872+0.0492701
[500]	train-auc:0.889977+0.000762277	test-auc:0.865467+0.00234209
[0]	train-auc:0.789471+0.0263721	test-auc:0.780179+0.0295967
[0]	train-auc:0.774275+0.037864	test-auc:0.766464+0.0418789
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.770836+0.0375671	test-auc:0.765024+0.0413325
[500]	train-auc:0.896191+0.000721614	test-auc:0.866064+0.00218483
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.766541+0.0390235	test-auc:0.761885+0.0415818
[500]	train-auc:0.895607+0.000706252	test-auc:0.866042+0.0022801
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.76659+0.038973	test-auc:0.761855+0.0416062
[500]	train-auc:0.895728+0.000718801	test-auc:0.866175+0.00228806
[0]	train-auc:0.774225+0.0372069	test-auc:0.765875+0.041614
[500]	train-auc:0.896547+0.000632156	test-auc:0.865968+0.00214934
[0]	train-auc:0.774201+0.0372002	test-auc:0.765942+0.0416204
[500]	train-auc:0.896483+0.000661713	test-auc:0.86611+0.00213591
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.774201+0.0372002	test-auc:0.765942+0.0416204
[500]	train-auc:0.896483+0.000661713	test-auc:0.86611+0.00213591
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.766406+0.0418873	test-auc:0.760552+0.0453045
[500]	train-auc:0.893464+0.000860053	test-auc:0.865694+0.00219457
[0]	train-auc:0.77373+0.0372875	test-auc:0.766155+0.0417738
[500]	train-auc:0.896431+0.000662105	test-auc:0.866047+0.00220631
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.747954+0.0441475	test-auc:0.745278+0.0475702
[500]	train-auc:0.874654+0.000629202	test-auc:0.866259+0.00202391
[0]	train-auc:0.767734+0.0406666	test-auc:0.762123+0.0451324
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.773061+0.0363113	test-auc:0.764742+0.0399449
[0]	train-auc:0.776687+0.0399863	test-auc:0.767415+0.0443809
[500]	train-auc:0.896375+0.000751826	test-auc:0.865717+0.00218438
[0]	train-auc:0.778824+0.0344754	test-auc:0.765758+0.0414967
[0]	train-auc:0.764368+0.0410277	test-auc:0.759834+0.0463478
```

```
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.77373+0.0372875	test-auc:0.766155+0.0417738
[500]	train-auc:0.896431+0.000662105	test-auc:0.866047+0.00220631
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.778511+0.0380653	test-auc:0.770153+0.0428036
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.776819+0.0361528	test-auc:0.766283+0.0409607
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.773555+0.0420757	test-auc:0.766221+0.045173
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.762061+0.0408762	test-auc:0.757106+0.0438786
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.755165+0.0438644	test-auc:0.751948+0.0470213
[500]	train-auc:0.87563+0.000577302	test-auc:0.866364+0.00190521
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.768373+0.0381835	test-auc:0.762412+0.0411692
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.776687+0.0399863	test-auc:0.767415+0.0443809
[500]	train-auc:0.896375+0.000751826	test-auc:0.865717+0.00218438
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.774146+0.0378243	test-auc:0.766361+0.0418834
[500]	train-auc:0.899157+0.000574355	test-auc:0.86601+0.00192838
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.762553+0.0436265	test-auc:0.758882+0.0457615
[0]	train-auc:0.76659+0.038973	test-auc:0.761855+0.0416062
[500]	train-auc:0.895728+0.000718801	test-auc:0.866175+0.00228806
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.752606+0.0448341	test-auc:0.749696+0.0475433
[500]	train-auc:0.881513+0.000527624	test-auc:0.866536+0.00236448
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.756175+0.0453363	test-auc:0.753169+0.0491988
[500]	train-auc:0.888698+0.000849572	test-auc:0.86578+0.00216512
[0]	train-auc:0.784403+0.0270008	test-auc:0.777012+0.0298264
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.768769+0.0402963	test-auc:0.764177+0.0429446
[500]	train-auc:0.894692+0.000866697	test-auc:0.865984+0.00221741
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.756495+0.0443989	test-auc:0.753335+0.0482404
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
[0]	train-auc:0.5+0	test-auc:0.5+0
```

```
[0]     train-auc:0.782104+0.0284431    test-auc:0.775044+0.0315727
[500]   train-auc:0.89732+0.00061171    test-auc:0.865836+0.0020987
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.752607+0.0448322    test-auc:0.749696+0.0475436
[500]   train-auc:0.875448+0.000528686  test-auc:0.866347+0.00190901
[1000]  train-auc:0.88203+0.000555938   test-auc:0.866409+0.00212919
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.76659+0.038973      test-auc:0.761855+0.0416062
[500]   train-auc:0.895728+0.000718801  test-auc:0.866175+0.00228806
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.766612+0.0389781    test-auc:0.761812+0.0415919
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.752606+0.0448341    test-auc:0.749696+0.0475433
[500]   train-auc:0.875184+0.000583308  test-auc:0.866345+0.00188909
[0]     train-auc:0.76659+0.038973      test-auc:0.761855+0.0416062
[500]   train-auc:0.895728+0.000718801  test-auc:0.866175+0.00228806
[0]     train-auc:0.772757+0.0371411    test-auc:0.766254+0.0413289
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.76659+0.038973      test-auc:0.761855+0.0416062
[500]   train-auc:0.895728+0.000718801  test-auc:0.866175+0.00228806
[0]     train-auc:0.768767+0.040246     test-auc:0.764111+0.0427932
[500]   train-auc:0.896967+0.000725382  test-auc:0.865802+0.00214231
[0]     train-auc:0.768224+0.0383059    test-auc:0.762535+0.0410486
[0]     train-auc:0.761712+0.0438702    test-auc:0.758767+0.0462556
[500]   train-auc:0.894399+0.000927674  test-auc:0.865792+0.00213055
[0]     train-auc:0.768475+0.0383223    test-auc:0.762548+0.040908
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.5+0 test-auc:0.5+0
[0]     train-auc:0.754959+0.0439945    test-auc:0.751489+0.0472597
[500]   train-auc:0.875379+0.000597828  test-auc:0.866333+0.00186733
[0]     train-auc:0.5+0 test-auc:0.5+0
The best hyperparameters are:

{'subsample': 2, 'lambda': 2, 'max_depth': 2, 'min_child_weight': 2, 'gamma': 0, 'eta': 0.025, 'alpha':
```

In [16]:
```python
params = {
        'silent': 1,
        'seed': 42,
        'subsample': 0.8,
        'eta': 0.025,
        'nthread': 4,
        'eval_metric': 'auc',
        'lambda': 0.9,
        'booster': 'gbtree',
        'alpha': 1.0,
        'colsample_bytree': 0.5,
```

```
        'objective': 'binary:logistic',
        'max_depth': 7,
        'min_child_weight': 7,
        'gamma': 0.3,
        'tree_method': 'exact',
        'n_estimators': 1100
    }

    # check model CV scores
    num_boost_round = int(params['n_estimators'])
    del params['n_estimators']
    dtrain = xgb.DMatrix(X_train, label=y_train)
    dtest = xgb.DMatrix(X_test)

    xg_booster = xgb.train(params, dtrain, num_boost_round)
    preds = xg_booster.predict(dtest)
    write_preds('submissions/xgb_mybest_{}.csv'.format(SEED), preds)
```

Final private score for these hyperparams: 0.866054, which is worse than Raymond's hyperparams. However, these obtained a higher CV score than Raymond's hyperparams. It's possible that we haven't discovered the optimal hyperparams for this dataset (didn't fully explore the search space), but that Raymond's parameters are still quite good.

6 Question 3

`Data Dictionary.xls` explains that you are making your decisions on giving loans using the Total balance on credit cards, the Monthly debt payments, the number of mortgage loands of the individual etc. You are now asked to tell a story from this dataset.

6.1 Part A

Fit a simple logistic regression model and report which features are important (and how they influence the deliquency chance). Discuss what is expected and what is surprising. See how regularization changes the importance of features.

Would you expect that the number of dependents to have a postive or negative effect in deliquency probability? Discuss what you think and what the data says.

```
In [13]: from sklearn.linear_model import LogisticRegression
         from sklearn.feature_selection import SelectFromModel

         log_clf = LogisticRegression().fit(X_train, y_train)
         model = SelectFromModel(log_clf, prefit=True)

         X_new = model.transform(X_train)
         print(X_train.shape)
         print(X_new.shape)
         print(model.threshold_)
         print(model.get_support())
```

```
(150000, 10)
(150000, 5)
0.514343479402
[ True  True  True False False False  True False  True False]
```

The features that were eliminated were: - DebtRatio - MonthlyIncome - NumberofOpenCreditLines - NumberofRealEstateLoansOrLines - NumberOfDependents.

The features that were important were: - RevolvingUtilizationOfUnsecuredLines - age - NumberOfTime30-59DaysPastDueNotWorse - NumberOfTimes90DaysLate - NumberOfTime60-89DaysPastDueNotWorse

In [14]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectFromModel

log_clf = LogisticRegression(C=0.0001).fit(X_train, y_train)
model = SelectFromModel(log_clf, prefit=True)

X_new = model.transform(X_train)
print(X_train.shape)
print(X_new.shape)
print(model.threshold_)
print(model.get_support())
```

```
(150000, 10)
(150000, 5)
0.148917013943
[ True  True  True False  True False  True False False False]
```

Regularization may be changing the relative feature importances, but SelectFromModel consistently selects the same 5 features across C=1, 0.5, 0.1, 0.01, 0.001, and 0.0001. Suffice to say that the regularization strength is not affecting feature selection very much, which I was surprised by. I'm also surprised by the fact that the model is discriminating based on age, even when it is illegal to do so.

In [15]:
```
train.corr().ix['NumberOfDependents', 'SeriousDlqin2yrs']
```

Out[15]:
```
0.045621089376376468
```

The Number of dependents and serious delinquency in 2 years is slightly positively correlated. I think that someone with more dependents is more likely to have trouble paying off their loans.

6.2 Part B

Look at your best models (in terms of LB AUC). Try to perform feature interpretability for them. Are the results consistent with interpreting a simple logistic regression?

In [35]:
```
import xgboost as xgb


# XGB, Raymond Wen's parameters
# Raymond Wen's parameters
params = {
    'n_estimators': 1000,
    'eta': 0.01,
    'max_depth': 4,
    'min_child_weight': 5,
    'subsample': 0.4,
    'gamma': 0.8,
    'colsample_bytree': 0.4,
    'lambda': 0.93,
    'alpha': 0.5,
    'eval_metric': 'auc',
```

```
        'objective': 'binary:logistic',
        # Increase this number if you have more cores.
        # Otherwise, remove it and it will default
        # to the maxium number.
        'nthread': 4,
        'booster': 'gbtree',
        'tree_method': 'exact',
        'silent': 1,
        'seed': SEED
    }

    xg_booster = xgb.train(params, dtrain, num_boost_round)
    preds = xg_booster.predict(dtest)
    write_preds('submissions/xgb_raymond_{}.csv'.format(SEED), preds)
```

```
In [38]: xg_booster.get_score()
```

```
Out[38]: {'f0': 2640,
          'f1': 1671,
          'f2': 1060,
          'f3': 2627,
          'f4': 1966,
          'f5': 1467,
          'f6': 959,
          'f7': 975,
          'f8': 777,
          'f9': 578}
```

Clearly, the results are different than performing a simple logistic regression.

7 Question 4

The Age Discrimination in Employment Act (ADEA) forbids age discrimination against people who are age 40 or older, see https://www.eeoc.gov/laws/types/age.cfm

Are your models considering age as a factor of influence?

Fit a model for people over 40 or 50 and a model for younger people. Are the two models different?

```
In [16]: train = pd.read_csv('data/cs-training.csv', index_col=0)
         test = pd.read_csv('data/cs-test.csv', index_col=0)
```

```
In [17]: # Drop dependent variable in test
         test = test.drop(['SeriousDlqin2yrs'], axis=1)
```

```
In [18]: # Fill missing with mean
         train = train.fillna(train.mean())
         test = test.fillna(test.mean())
```

```
In [19]: # Split on age
         train_young = train[train.age <= 40]
         train_old = train[train.age > 40]

         test_young = test[test.age <= 40]
         test_old = test[test.age > 40]

         # Seperate dependent and independent
```

```
X_train_young = train_young.drop(['SeriousDlqin2yrs'], axis=1)
y_train_young = train_young['SeriousDlqin2yrs']

X_train_old = train_old.drop(['SeriousDlqin2yrs'], axis=1)
y_train_old = train_old['SeriousDlqin2yrs']
```

In [45]:
```
# Young model
xg = xgb.XGBClassifier(max_depth=8, learning_rate=0.3, n_estimators=155,
                       min_child_weight=0.6, subsample=1.0, colsample_bytree=0.45)

score = cross_val_score(xg, X=X_train_young, y=y_train_young, scoring='roc_auc', cv=10, n_jobs
print(score)
print(score.mean())
```

```
[ 0.79526594  0.80453694  0.80701095  0.81433514  0.80898215  0.79669316
  0.81571334  0.82097823  0.8071783   0.81165422]
0.808234837985
```

In [46]:
```
# Old model
xg = xgb.XGBClassifier(max_depth=8, learning_rate=0.3, n_estimators=155,
                       min_child_weight=0.6, subsample=1.0, colsample_bytree=0.45)

score = cross_val_score(xg, X=X_train_old, y=y_train_old, scoring='roc_auc', cv=10, n_jobs=-1)
print(score)
print(score.mean())
```

```
[ 0.84046547  0.85704543  0.8401881   0.84075806  0.84354391  0.84919176
  0.84959688  0.84603375  0.85213222  0.86746431]
0.848641989413
```

7.0.1 Discussion

We can see that a model with the same parameters does much better on the set of older people than on the set of younger people. Age is clearly an influence factor in this dataset. We can use a RandomizedSearch to see if different parameters are selected for models.

In [47]:
```
def print_cv(model, name):
    print("Best parameter set found on {} model:\n".format(name))
    print(model.best_params_)
    print()
    for params, mean_score, scores in model.grid_scores_:
        print("{0:.3f} (+/-{1:.03f}) for {2}".format(mean_score, scores.std() * 2, params))
    print()
```

In [48]:
```
from sklearn.model_selection import RandomizedSearchCV

parameters = {
    'max_depth': [6, 8],
    'learning_rate': [0.1, 0.01],
    'n_estimators': [200],
    'min_child_weight': [1/(0.95**(1/2))],
    'colsample_bytree': [0.4, 0.5]
}

xg_clf = RandomizedSearchCV(xgb.XGBClassifier(), parameters, n_iter=5, cv=5, n_jobs=-1, scorin
xg_clf.fit(X_train_young, y_train_young)
```

```
print_cv(xg_clf, 'young')

xg_clf = RandomizedSearchCV(xgb.XGBClassifier(), parameters, n_iter=5, cv=5, n_jobs=-1, scoring
xg_clf.fit(X_train_old, y_train_old)
print_cv(xg_clf, 'old')
```

```
Best parameter set found on young model:

{'min_child_weight': 1.0259783520851542, 'learning_rate': 0.01, 'colsample_bytree': 0.5, 'max_depth': 8,

0.832 (+/-0.006) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.1, 'colsample_bytree': (
0.825 (+/-0.007) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.1, 'colsample_bytree': (
0.837 (+/-0.008) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.01, 'colsample_bytree':
0.825 (+/-0.008) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.1, 'colsample_bytree': (
0.831 (+/-0.006) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.1, 'colsample_bytree': (

/home/aetherzephyr/anaconda3/envs/datasci/lib/python3.5/site-packages/sklearn/model_selection/_search.py
  DeprecationWarning)

Best parameter set found on old model:

{'min_child_weight': 1.0259783520851542, 'learning_rate': 0.01, 'colsample_bytree': 0.5, 'max_depth': 8,

0.869 (+/-0.009) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.01, 'colsample_bytree':
0.863 (+/-0.010) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.1, 'colsample_bytree': (
0.868 (+/-0.009) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.01, 'colsample_bytree':
0.868 (+/-0.009) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.01, 'colsample_bytree':
0.863 (+/-0.010) for {'min_child_weight': 1.0259783520851542, 'learning_rate': 0.1, 'colsample_bytree': (

/home/aetherzephyr/anaconda3/envs/datasci/lib/python3.5/site-packages/sklearn/model_selection/_search.py
  DeprecationWarning)
```

7.0.2 Discussion

Again, we see that when seperated by age, the model with older people performs much better. Also, different paramters are selected. In the younger model, `max_depth` was chosen to be 6, while the older model chose `max_depth` as 8. If we had searched over more parameter values, the models would likely be completely different.

7.1 Part B

As a law-maker do you think that forcing age and number of dependents to be forbidden features is a good idea for this problem? Try to base your discussion on what you discover from the data.

7.2 Answer

I think that from a law point of view, those features should be forbidden no matter what the data says. Age should not be considered when deciding if a person can get a loan or not, because that does classify as age discrimination. Also, if we are to not discriminate for people 40 or older, we should not discriminate based on any age value.

According to the data, knowing the age can be valuable in predicting financial distress. This is clear from the work we did in part A. Since there is such a boost in performace for a model predicting on only people over the age of 40, this means that using their age is very helpful to the model. This may seem like a good idea since we get a better AUC ROC score, but in fact this is leading to age discrimination. As a law-maker, I would not feel comfortable knowing that we can predict so much better for people over age 40. This may be generalization and can lead to discrimination based on a person's age.

Because of this, I think that it would be a good idea (as a law-maker) to make age and the number of dependents to be forbidden features. However, from a data perspective (disregarding law), knowing the age can help your models a lot.

```
In [ ]:
```