santander - ensemble methods

March 4, 2019

0.1 Create and consolidate predictions from various methods

```
In [1]: import pandas as pd
        from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
        from IPython.display import display
        import numpy as np
        from sklearn import metrics
        import matplotlib.pyplot as plt
        from sklearn.metrics import classification_report, roc_auc_score
        from sklearn.model_selection import train_test_split
        import xgboost as xgb
        from xgboost import XGBClassifier
        from sklearn.ensemble import AdaBoostClassifier
        import lightgbm
```

0.2 Data pre-processing

```
In [2]: train = pd.read_csv('train.csv')
        test = pd.read_csv('test.csv')
```

```
In [3]: train.target.value_counts()
```

```
Out[3]: 0    179902
        1     20098
        Name: target, dtype: int64
```

```
In [4]: #Drop the id column
        train = train.drop(['ID_code'], axis=1)
        test = test.drop(['ID_code'], axis=1)
```

```
In [5]: #Split train data into train and valid
        x_valid = train.iloc[100000:110000].copy()
        y_valid = x_valid['target']
        x_valid = x_valid.drop(['target'], axis=1)

        x_train = train.drop(x_valid.index).copy()
        y_train = x_train['target']
        x_train = x_train.drop(['target'], axis=1)
```

```
In [6]: x_train.shape, y_train.shape, x_valid.shape, y_valid.shape
```

```
Out[6]: ((190000, 200), (190000,), (10000, 200), (10000,))
```

```
In [7]: test.shape
```

```
Out[7]: (200000, 200)
```

0.3 Run models

```
In [8]: #Run Randomforest
        rf = RandomForestClassifier(n_jobs=-1, max_features= 23, min_samples_leaf= 10, n_estima
        rf.fit(x_train, y_train)
```

```
Out[8]: RandomForestClassifier(bootstrap=True, class_weight='balanced',
                    criterion='gini', max_depth=12, max_features=23,
                    max_leaf_nodes=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=10,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=10, n_jobs=-1, oob_score=False, random_state=None,
                    verbose=0, warm_start=False)
```

```
In [32]: #Use the feature importance to find the most important ones
         feature_importance = pd.DataFrame({'Feature' : x_train.columns, 'Importance' : rf.feat
         feature_importance.sort_values('Importance', ascending=False, inplace=True)
         feature_importance.head(30)
```

Out[32]:

	Feature	Importance
81	var_81	0.062372
139	var_139	0.042399
110	var_110	0.036165
12	var_12	0.032567
53	var_53	0.030094
26	var_26	0.029922
174	var_174	0.023696
6	var_6	0.023388
109	var_109	0.021580
22	var_22	0.021035
146	var_146	0.017432
133	var_133	0.016531
190	var_190	0.014535
148	var_148	0.014105
76	var_76	0.013605
166	var_166	0.013055
80	var_80	0.012766
21	var_21	0.012717
165	var_165	0.011777
198	var_198	0.011372
179	var_179	0.010742

44	var_44	0.010255
2	var_2	0.009981
99	var_99	0.009980
78	var_78	0.008830
1	var_1	0.008337
94	var_94	0.008320
170	var_170	0.008053
34	var_34	0.007728
115	var_115	0.007256

In [9]:
```
#XGBoost
xgbm = XGBClassifier(learning_rate =0.1, n_estimators=1000, max_depth=5, min_child_weig
xgbm.fit(x_train, y_train)
```

Out[9]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
       colsample_bytree=0.7, cv=5, gamma=0.1, iid=False, learning_rate=0.1,
       max_delta_step=0, max_depth=5, min_child_weight=3, missing=None,
       n_estimators=1000, n_jobs=4, nthread=4, objective='binary:logistic',
       random_state=0, reg_alpha=1, reg_lambda=1, scale_pos_weight=0.65,
       scoring='roc_auc', seed=27, silent=True, subsample=0.75)
```

In [31]:
```
#Use the feature importance to find the most important ones
feature_importance = pd.DataFrame({'Feature' : x_train.columns, 'Importance' : xgbm.fe
feature_importance.sort_values('Importance', ascending=False, inplace=True)
feature_importance.head(30)
```

Out[31]:

	Feature	Importance
2	var_2	0.007750
166	var_166	0.007428
26	var_26	0.007393
177	var_177	0.007250
139	var_139	0.007107
34	var_34	0.006964
165	var_165	0.006928
169	var_169	0.006893
1	var_1	0.006893
21	var_21	0.006821
13	var_13	0.006821
123	var_123	0.006678
192	var_192	0.006678
133	var_133	0.006643
198	var_198	0.006643
174	var_174	0.006643
127	var_127	0.006607
78	var_78	0.006571
148	var_148	0.006571
149	var_149	0.006571
12	var_12	0.006571
6	var_6	0.006535

```
173  var_173    0.006500
110  var_110    0.006500
118  var_118    0.006500
53    var_53    0.006500
108  var_108    0.006464
99    var_99    0.006464
109  var_109    0.006464
22    var_22    0.006464
```

In [10]:
```
#Adaboost
abc = AdaBoostClassifier(n_estimators=100, learning_rate=2)
abc.fit(x_train, y_train)
```

Out[10]:
```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=2,
          n_estimators=100, random_state=None)
```

In [28]:
```
abc.feature_importances_
```

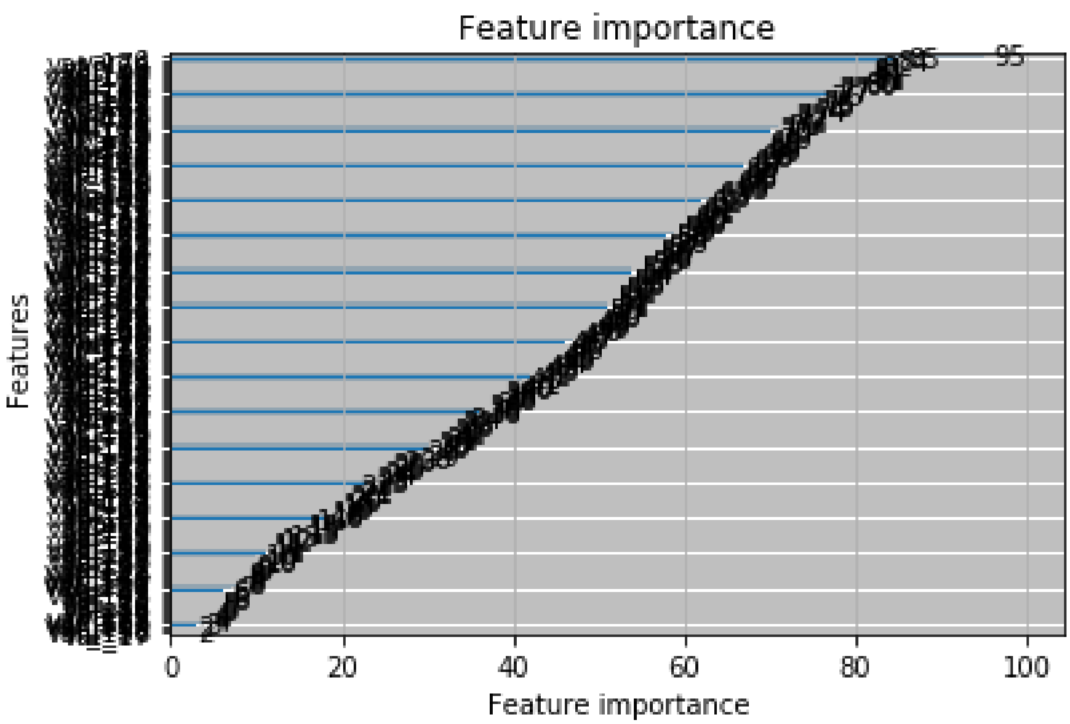
Out[28]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.,
       0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

In [55]:
```
#LightGBM
parameters = {
    'application': 'binary',
    'objective': 'binary',
    'metric': 'auc',
    'is_unbalance': 'true',
    'boosting': 'gbdt',
    'num_leaves': 7,
    'feature_fraction': 0.5,
    'bagging_fraction': 0.5,
    'bagging_freq': 20,
    'learning_rate': 0.05,
    'verbose': 0,
    'max_depth' : 3,
    'min_data_in_leaf': 5000
}

train_data = lightgbm.Dataset(x_train, label=y_train)
```

```
valid_data = lightgbm.Dataset(x_valid, label=y_valid)

model = lightgbm.train(parameters,
                       train_data,
                       valid_sets=valid_data,
                       num_boost_round=5000,
                       early_stopping_rounds=100, verbose_eval=False);
```

In [57]:
```
lightgbm.plot_importance(model)
```

Out[57]: <matplotlib.axes._subplots.AxesSubplot at 0x14e16afb400>



In [ ]:
```
assessMod()
```

In [27]:
```
train_data.feature_name[138]
```

Out[27]: 'var_138'

0.4 Validation prediction

In [13]:
```
#Prediction for rf validation data
rf_pred_df_valid = pd.DataFrame(rf.predict_proba(x_valid))
rf_pred_df_valid['target'] = rf.predict(x_valid)
rf_pred_df_valid.columns = ['rf_0', 'rf_1', 'rf_target']
rf_pred_df_valid['rf_1'].values
roc_auc_score(y_valid, rf_pred_df_valid['rf_1'].values)
```

```
Out[13]: 0.7192191641040887
```

```
In [14]: #Prediction for xgb validation data
         xgbm_pred_df_valid = pd.DataFrame(xgbm.predict_proba(x_valid))
         xgbm_pred_df_valid['target'] = xgbm.predict(x_valid)
         xgbm_pred_df_valid.columns = ['xgbm_0', 'xgbm_1', 'xgbm_target']
         xgbm_pred_df_valid['xgbm_1'].values
         roc_auc_score(y_valid, xgbm_pred_df_valid['xgbm_1'].values)
```

```
Out[14]: 0.8907805503995351
```

```
In [15]: #Prediction for abc validation data
         abc_pred_df_valid = pd.DataFrame(abc.predict_proba(x_valid))
         abc_pred_df_valid['target'] = abc.predict(x_valid)
         abc_pred_df_valid.columns = ['abc_0', 'abc_1', 'abc_target']
         abc_pred_df_valid['abc_1'].values
         roc_auc_score(y_valid, abc_pred_df_valid['abc_1'].values)
```

```
Out[15]: 0.4505475081953444
```

```
In [56]: #Prediction for validation data
         lgbm_pred_df_valid = pd.DataFrame(model.predict(x_valid.values), columns=['xgbm_prob']
         y_pred = []
         for i in lgbm_pred_df_valid.index:
             if lgbm_pred_df_valid.iloc[i,0] >= 0.5:       # setting threshold to .5
                y_pred.append(1)
             else:
                y_pred.append(0)

         lgbm_pred_df_valid['lgbm_1'] = y_pred
         lgbm_pred_df_valid.columns = ['lgbm_1', 'lgbm_target']
         roc_auc_score(y_valid, lgbm_pred_df_valid['lgbm_1'].values)
```

```
Out[56]: 0.8920777401120976
```

0.5 Test data

```
In [ ]: #Prediction for rf test data
        rf_pred_df_test = pd.DataFrame(rf.predict_proba(test))
        rf_pred_df_test['target'] = rf.predict(test)
        rf_pred_df_test.columns = ['rf_0', 'rf_1', 'rf_target']
```

```
In [ ]: #Prediction for xbg test data
        xgb_pred_df = pd.DataFrame(xgbm.predict_proba(test))
        xgb_pred_df['xgb_target'] = xgbm.predict(test)
        xgb_pred_df.columns = ['xgb_0', 'xgb_1', 'xgb_target']
```

```
In [ ]: #Prediction for abc test data
        abc_pred_df = pd.DataFrame(abc.predict_proba(test))
        abc_pred_df['abc_target'] = abc.predict(test)
        abc_pred_df.columns = ['abc_0', 'abc_1', 'abc_target']
```

```
In [ ]: #Prediction for lgbm test values
        lgbm_pred_df_test = pd.DataFrame(model.predict(test.values), columns=['xgbm_prob'])
        y_pred = []
        for i in lgbm_pred_df_test.index:
            if lgbm_pred_df_test.iloc[i,0] >= 0.5:       # setting threshold to .5
               y_pred.append(1)
            else:
               y_pred.append(0)

        lgbm_pred_df_test['lgbm_1'] = y_pred
        lgbm_pred_df_test.columns = ['lgbm_1', 'lgbm_target']
```

0.6 Consolidation

```
In [71]: final_pred = pd.concat([rf_pred_df, xgb_pred_df, abc_pred_df, lgbm_pred_df ], axis=1)

In [77]: final_pred.loc[0, 'rf_target']

Out[77]: 1

In [81]: from statistics import mode
         mode([final_pred.loc[0,'rf_target'], final_pred.loc[0,'xgb_target'], final_pred.loc[0


        ---------------------------------------------------------------------------

        StatisticsError                           Traceback (most recent call last)

        <ipython-input-81-d1280be51e26> in <module>
          1 from statistics import mode
    ----> 2 mode([final_pred.loc[0,'rf_target'], final_pred.loc[0,'xgb_target'], final_pred.loc


        ~\AppData\Local\Continuum\anaconda3\lib\statistics.py in mode(data)
        505     elif table:
        506         raise StatisticsError(
    --> 507                 'no unique mode; found %d equally common values' % len(table)
        508                 )
        509     else:


        StatisticsError: no unique mode; found 2 equally common values


In [82]: final_pred['lgbm_0'] = 1 - final_pred['lgbm_1']

In [86]: final_pred.shape

Out[86]: (200000, 13)
```

In [85]:
```
final_pred['avg_prob_0'] = (final_pred['rf_0'] + final_pred['xgb_0'] + final_pred['ab
```

In [87]:
```
final_pred['avg_prob_1'] = (final_pred['rf_1'] + final_pred['xgb_1'] + final_pred['ab
```

In [89]:
```
final_pred['ID_code'] = pd.read_csv('test.csv')['ID_code']
```

In [90]:
```
final_pred.head()
```

Out[90]:

	rf_0	rf_1	rf_target	xgb_0	xgb_1	xgb_target	abc_0	\
0	0.440911	0.559089	1	0.962129	0.037871	0	0.499997	
1	0.629574	0.370426	0	0.844926	0.155074	0	0.499997	
2	0.627094	0.372906	0	0.931649	0.068351	0	0.499997	
3	0.700227	0.299773	0	0.814178	0.185822	0	0.499997	
4	0.632039	0.367961	0	0.980460	0.019540	0	0.499997	

	abc_1	abc_target	lgbm_1	lgbm_target	lgbm_0	avg_prob_0	\
0	0.500003	1	0.299194	0	0.700806	0.650961	
1	0.500003	1	0.569884	1	0.430116	0.601153	
2	0.500003	1	0.564769	1	0.435231	0.623493	
3	0.500003	1	0.554634	1	0.445366	0.614942	
4	0.500003	1	0.177830	0	0.822170	0.733666	

	avg_prob_1	ID_code
0	0.349039	test_0
1	0.398847	test_1
2	0.376507	test_2
3	0.385058	test_3
4	0.266334	test_4

In [91]:
```
submission = final_pred[['ID_code', 'avg_prob_1']]
```

In [93]:
```
submission.to_csv('submission4.csv')
```

In [ ]: