Customer Transaction Prediction

RIHAD VARIAWA

26-02-2019

Overview

In Santander Customer Transaction Prediction competition we have a binary classification task. Train and test data have 200k samples each and we have 200 anonimyzed numerical columns. It would be interesting to try good models without overfitting and knowing the meaning of the features. In fact this competition seems to be similar to another current competition: don't overfit! In this kernel I'll write the following things:

- EDA on the features and trying to get some insights;
- Using permutation importance to select most impactful features;
- Comparing various models: linear models, tree based models and others;
- Trying various approaches to feature selection including taking top features from eli5;
- Hyperparameter optimization for models;
- Feature generation;
- Other things

In [0]:
```
# access kaggle datasets
!pip install kaggle

# math operations
!pip install numpy==1.15.0

!pip install eli5

!pip install shap

# for machine learning
!pip install catboost
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.6/dist-packa
ges (1.5.3)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/pytho
n3.6/dist-packages (from kaggle) (1.22)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.6/dist-pa
ckages (from kaggle) (1.11.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-pack
ages (from kaggle) (2018.11.29)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.6/d
ist-packages (from kaggle) (2.5.3)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-pac
kages (from kaggle) (2.18.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-package
s (from kaggle) (4.28.1)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.6/di
st-packages (from kaggle) (2.0.1)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/pytho
n3.6/dist-packages (from requests->kaggle) (3.0.4)
Requirement already satisfied: idna<2.7,>=2.5 in /usr/local/lib/python3.6/di
st-packages (from requests->kaggle) (2.6)
Requirement already satisfied: Unidecode>=0.04.16 in /usr/local/lib/python3.
6/dist-packages (from python-slugify->kaggle) (1.0.23)
Collecting numpy==1.15.0
  Downloading https://files.pythonhosted.org/packages/88/29/f4c845648ed23264
e986cdc5fbab5f8eace1be5e62144ef69ccc7189461d/numpy-1.15.0-cp36-cp36m-manylin
ux1_x86_64.whl (13.9MB)
    100% |████████████████████████████████| 13.9MB 3.6MB/s
torchvision 0.2.1 has requirement pillow>=4.1.1, but you'll have pillow 4.0.
0 which is incompatible.
thinc 6.12.1 has requirement wrapt<1.11.0,>=1.10.0, but you'll have wrapt 1.
11.1 which is incompatible.
pymc3 3.6 has requirement joblib<0.13.0, but you'll have joblib 0.13.2 which
is incompatible.
featuretools 0.4.1 has requirement pandas>=0.23.0, but you'll have pandas 0.
22.0 which is incompatible.
albumentations 0.1.12 has requirement imgaug<0.2.7,>=0.2.5, but you'll have
 imgaug 0.2.8 which is incompatible.
Installing collected packages: numpy
  Found existing installation: numpy 1.14.6
    Uninstalling numpy-1.14.6:
      Successfully uninstalled numpy-1.14.6
Successfully installed numpy-1.15.0

Collecting eli5
  Downloading https://files.pythonhosted.org/packages/8d/c8/04bed18dcce1d927
b0dd5fc3425777354b714d2e62d60ae301928b5a5bf8/eli5-0.8.1-py2.py3-none-any.whl
(98kB)
    100% |████████████████████████████████| 102kB 3.4MB/s
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.
6/dist-packages (from eli5) (0.20.2)
Requirement already satisfied: attrs>16.0.0 in /usr/local/lib/python3.6/dist
-packages (from eli5) (18.2.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages
(from eli5) (1.11.0)
Requirement already satisfied: numpy>=1.9.0 in /usr/local/lib/python3.6/dist
-packages (from eli5) (1.15.0)
Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-pac
kages (from eli5) (0.10.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.6/dist-packa
ges (from eli5) (2.10)
Requirement already satisfied: tabulate>=0.7.7 in /usr/local/lib/python3.6/d
ist-packages (from eli5) (0.8.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packag
es (from eli5) (1.1.0)
Requirement already satisfied: typing in /usr/local/lib/python3.6/dist-packa
ges (from eli5) (3.6.6)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.6/
dist-packages (from jinja2->eli5) (1.1.0)
Installing collected packages: eli5
Successfully installed eli5-0.8.1
Collecting shap
  Downloading https://files.pythonhosted.org/packages/30/b3/866b0101cbd18298
44c35964af68c14ba522a5cce7a1e8d0f7937411d910/shap-0.28.5.tar.gz (223kB)
    100% |████████████████████████████████| 225kB 9.2MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packag
es (from shap) (1.15.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packag
es (from shap) (1.1.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist
-packages (from shap) (0.20.2)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-p
ackages (from shap) (3.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packa
ges (from shap) (0.22.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-package
s (from shap) (4.28.1)
Requirement already satisfied: ipython in /usr/local/lib/python3.6/dist-pack
ages (from shap) (5.5.0)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.6/dist
-packages (from shap) (0.13.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /
usr/local/lib/python3.6/dist-packages (from matplotlib->shap) (2.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist
-packages (from matplotlib->shap) (0.10.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python
3.6/dist-packages (from matplotlib->shap) (2.5.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.
6/dist-packages (from matplotlib->shap) (1.0.1)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-
packages (from pandas->shap) (2018.9)
Requirement already satisfied: pexpect; sys_platform != "win32" in /usr/loca
l/lib/python3.6/dist-packages (from ipython->shap) (4.6.0)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/li
b/python3.6/dist-packages (from ipython->shap) (1.0.15)
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-pac
kages (from ipython->shap) (2.1.3)
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.
6/dist-packages (from ipython->shap) (0.8.1)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/di
st-packages (from ipython->shap) (4.3.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-
packages (from ipython->shap) (0.7.5)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.6/
dist-packages (from ipython->shap) (40.8.0)
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-pa
ckages (from ipython->shap) (4.3.2)
Requirement already satisfied: six>=1.7.3 in /usr/local/lib/python3.6/dist-p
ackages (from scikit-image->shap) (1.11.0)
Requirement already satisfied: networkx>=1.8 in /usr/local/lib/python3.6/dis
t-packages (from scikit-image->shap) (2.2)
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.
6/dist-packages (from scikit-image->shap) (1.0.1)
Requirement already satisfied: pillow>=2.1.0 in /usr/local/lib/python3.6/dis
t-packages (from scikit-image->shap) (4.0.0)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/d
ist-packages (from pexpect; sys_platform != "win32"->ipython->shap) (0.6.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-pack
ages (from prompt-toolkit<2.0.0,>=1.0.4->ipython->shap) (0.1.7)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.6/
dist-packages (from traitlets>=4.2->ipython->shap) (0.2.0)
Requirement already satisfied: olefile in /usr/local/lib/python3.6/dist-pack
ages (from pillow>=2.1.0->scikit-image->shap) (0.46)
Building wheels for collected packages: shap
  Building wheel for shap (setup.py) ... done
  Stored in directory: /root/.cache/pip/wheels/bf/26/bd/912db1314f1cef0171d9
b7f128dd01e8b8c92ed8d0062e632d
Successfully built shap
Installing collected packages: shap
Successfully installed shap-0.28.5
Collecting catboost
  Downloading https://files.pythonhosted.org/packages/98/03/777a0e1c12571a7f
3320a4fa6d5f123dba2dd7c0bca34f4f698a6396eb48/catboost-0.12.2-cp36-none-manyl
inux1_x86_64.whl (55.5MB)
    100% |████████████████████████████████| 55.5MB 818kB/s
Requirement already satisfied: pandas>=0.19.1 in /usr/local/lib/python3.6/di
st-packages (from catboost) (0.22.0)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages
(from catboost) (1.11.0)
Requirement already satisfied: enum34 in /usr/local/lib/python3.6/dist-packa
ges (from catboost) (1.1.6)
Requirement already satisfied: numpy>=1.11.1 in /usr/local/lib/python3.6/dis
t-packages (from catboost) (1.15.0)
Requirement already satisfied: python-dateutil>=2 in /usr/local/lib/python3.
6/dist-packages (from pandas>=0.19.1->catboost) (2.5.3)
Requirement already satisfied: pytz>=2011k in /usr/local/lib/python3.6/dist-
packages (from pandas>=0.19.1->catboost) (2018.9)
Installing collected packages: catboost
Successfully installed catboost-0.12.2
```

In [0]:
```
# Libraries
import numpy as np
import pandas as pd
pd.set_option('max_columns', None)
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import datetime
import lightgbm as lgb
from scipy import stats
from sklearn.model_selection import train_test_split, StratifiedKFold, KFold,
cross_val_score, GridSearchCV, RepeatedStratifiedKFold
from sklearn.preprocessing import StandardScaler
import os
import plotly.offline as py
py.init_notebook_mode(connected=True)
import plotly.graph_objs as go
import plotly.tools as tls
import xgboost as xgb
import lightgbm as lgb
from sklearn import model_selection
from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn import metrics
import json
import ast
import time
from sklearn import linear_model
import eli5
from eli5.sklearn import PermutationImportance
import shap
from tqdm import tqdm_notebook
from mlxtend.feature_selection import SequentialFeatureSelector as SFS
from mlxtend.plotting import plot_sequential_feature_selection as plot_sfs
from sklearn.neighbors import NearestNeighbors
from sklearn.feature_selection import GenericUnivariateSelect, SelectPercentil
e, SelectKBest, f_classif, mutual_info_classif, RFE
import statsmodels.api as sm
import warnings
warnings.filterwarnings('ignore')
from catboost import CatBoostClassifier
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/compat/pandas.py:56: Futu
reWarning:

The pandas.core.datetools module is deprecated and will be removed in a futu
re version. Please use the pandas.tseries module instead.
```

Import Dataset from Kaggle

In [0]:
```
# Colab's file access feature
from google.colab import files

# retrieve uploaded file
uploaded = files.upload()

# print results
for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))

# then move kaggle.json into the folder where the API expects to find it.
!mkdir -p ~/.kaggle/ && mv kaggle.json ~/.kaggle/ && chmod 600 ~/.kaggle/kaggl
e.json
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving kaggle.json to kaggle.json
User uploaded file "kaggle.json" with length 62 bytes
```

In [0]:
```
# list files
!ls
```

```
sample_data  sample_submission.csv.zip  test.csv.zip  train.csv.zip
```

In [0]:
```
# unzip training data for usage
!unzip train.parquet.zip
!unzip test.parquet.zip
!unzip test.csv.zip
!unzip train.csv.zip
!unzip sample_submission.csv.zip
```

```
unzip:  cannot find or open train.parquet.zip, train.parquet.zip.zip or trai
n.parquet.zip.ZIP.
unzip:  cannot find or open test.parquet.zip, test.parquet.zip.zip or test.p
arquet.zip.ZIP.
Archive:  test.csv.zip
  inflating: test.csv
Archive:  train.csv.zip
  inflating: train.csv
Archive:  sample_submission.csv.zip
  inflating: sample_submission.csv
```

In [0]:
```
# download santander customer transaction dataset
!kaggle competitions download -c santander-customer-transaction-prediction
```

```
train.csv.zip: Skipping, found more recently modified local copy (use --forc
e to force download)
sample_submission.csv.zip: Skipping, found more recently modified local copy
(use --force to force download)
test.csv.zip: Skipping, found more recently modified local copy (use --force
to force download)
```

In [0]:
```
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
train.shape, test.shape
```

Out[0]:
```
((200000, 202), (200000, 201))
```

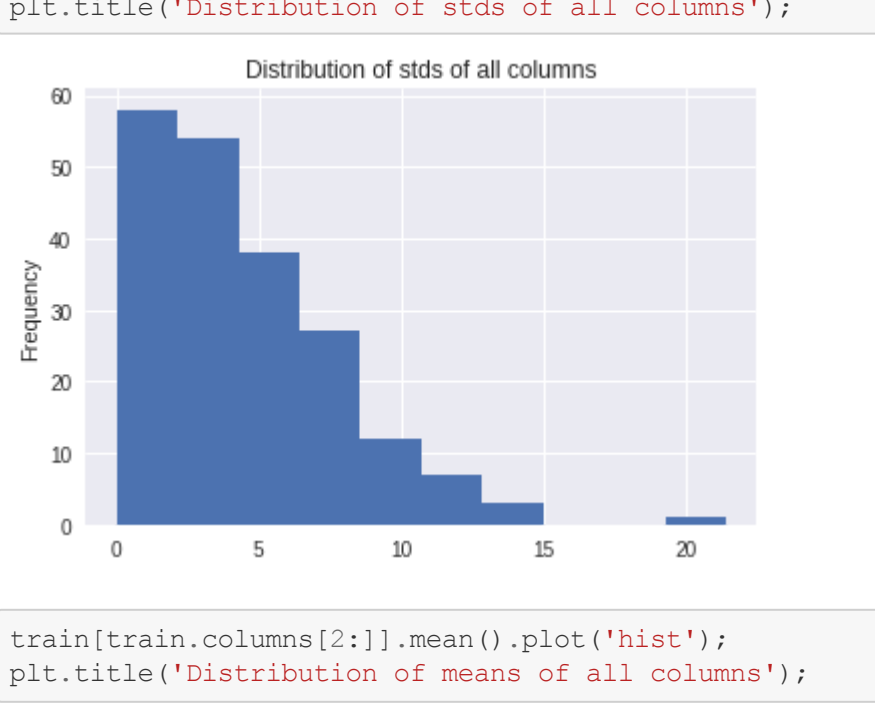
Data exploration

In [0]:
```
train.head()
```

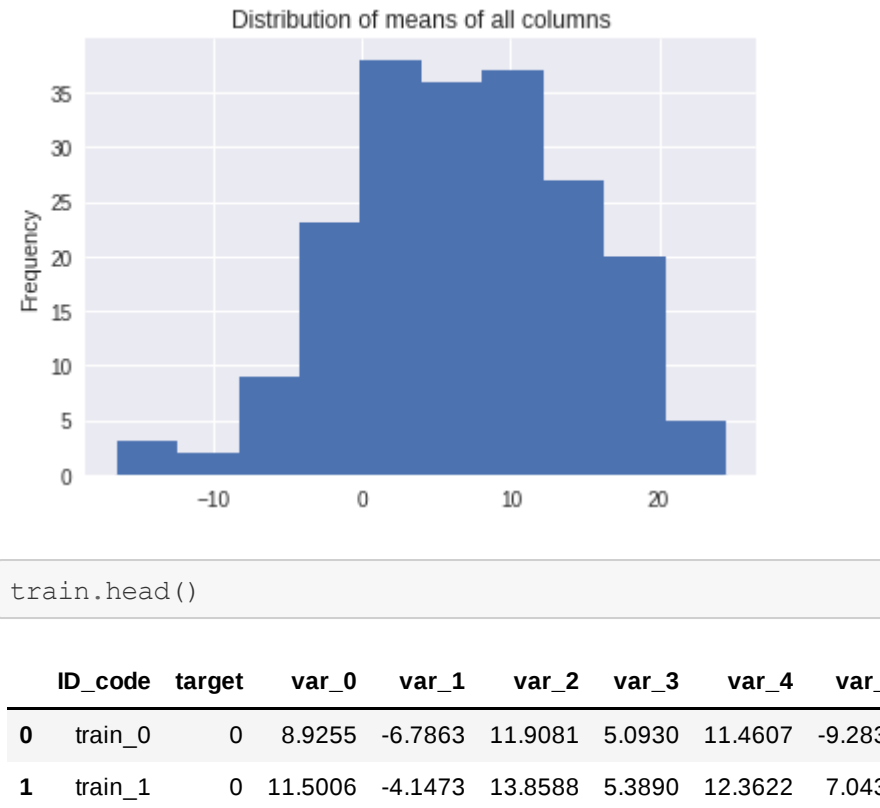
Out[0]:

	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	-4.9200
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	3.1468
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	-4.9193
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	-5.8609
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	6.2654

In [0]:
```
train[train.columns[2:]].std().plot('hist');
plt.title('Distribution of stds of all columns');
```



In [0]:
```
train[train.columns[2:]].mean().plot('hist');
plt.title('Distribution of means of all columns');
```



In [0]:
```
train.head()
```

Out[0]:

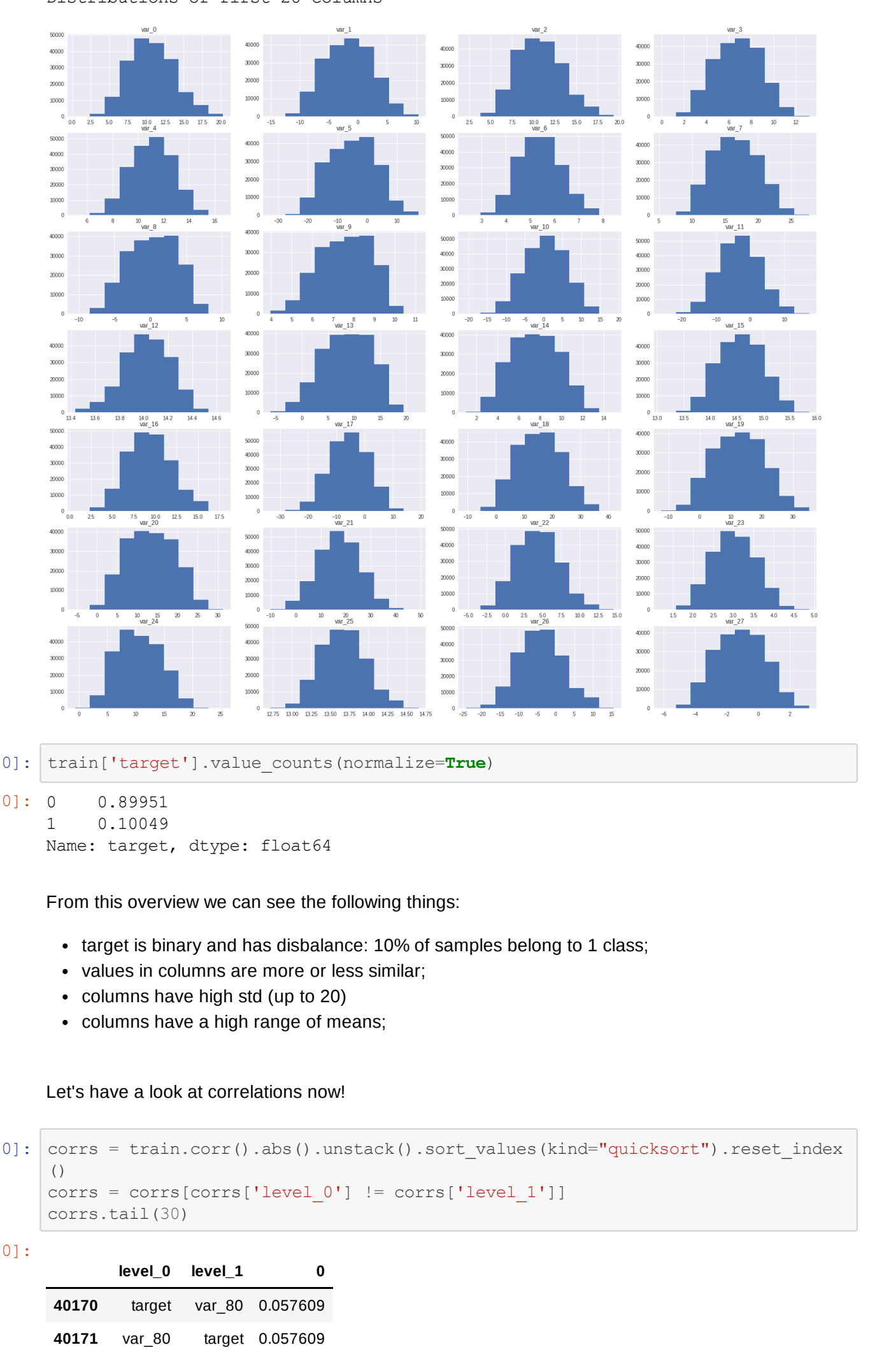
	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8
0	train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	-4.9200
1	train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	3.1468
2	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	-4.9193
3	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	-5.8609
4	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	6.2654

In [0]:
```
# we have no missing values
train.isnull().any().any()
```

Out[0]:
```
False
```

In [0]:
```
print('Distributions of first 28 columns')
plt.figure(figsize=(26, 24))
for i, col in enumerate(list(train.columns)[2:30]):
    plt.subplot(7, 4, i + 1)
    plt.hist(train[col])
    plt.title(col)
```

```
Distributions of first 28 columns
```



In [0]:
```
train['target'].value_counts(normalize=True)
```

Out[0]:
```
0    0.89951
1    0.10049
Name: target, dtype: float64
```

From this overview we can see the following things:

- target is binary and has disbalance: 10% of samples belong to 1 class;
- values in columns are more or less similar;
- columns have high std (up to 20)
- columns have a high range of means;

Let's have a look at correlations now!

In [0]:
```
corrs = train.corr().abs().unstack().sort_values(kind="quicksort").reset_index
()
corrs = corrs[corrs['level_0'] != corrs['level_1']]
corrs.tail(30)
```

Out[0]:

	level_0	level_1	0
40170	target	var_80	0.057609
40171	var_80	target	0.057609
40172	var_166	target	0.057773
40173	target	var_166	0.057773
40174	var_99	target	0.058367
40175	target	var_99	0.058367
40176	target	var_21	0.058483
40177	var_21	target	0.058483
40178	target	var_22	0.060558
40179	var_22	target	0.060558
40180	target	var_174	0.061669
40181	var_174	target	0.061669
40182	var_76	target	0.061917
40183	target	var_76	0.061917
40184	target	var_26	0.062422
40185	var_26	target	0.062422
40186	var_53	target	0.063399
40187	target	var_53	0.063399
40188	var_146	target	0.063644
40189	target	var_146	0.063644
40190	var_110	target	0.064275
40191	target	var_110	0.064275
40192	var_6	target	0.066731
40193	target	var_6	0.066731
40194	var_12	target	0.069489
40195	target	var_12	0.069489
40196	target	var_139	0.074080
40197	var_139	target	0.074080
40198	target	var_81	0.080917
40199	var_81	target	0.080917

In [0]:
```
corrs.head()
```

Out[0]:

	level_0	level_1	0
0	var_191	var_75	2.703975e-08
1	var_75	var_191	2.703975e-08
2	var_173	var_6	5.942735e-08
3	var_6	var_173	5.942735e-08
4	var_126	var_109	1.313947e-07

We can see that all features have a low correlation with target. So we have no highly correlated features which we could drop, on the other hand we could drop some columns with have little correlation with the target.

Basic modelling

In [0]:
```
X = train.drop(['ID_code', 'target'], axis=1)
y = train['target']
X_test = test.drop(['ID_code'], axis=1)
n_fold = 5
folds = StratifiedKFold(n_splits=n_fold, shuffle=True, random_state=42)
repeated_folds = RepeatedStratifiedKFold(n_splits=10, n_repeats=20, random_sta
te=42)

# scaler = StandardScaler()
# X_train = scaler.fit_transform(X_train)
# X_test = scaler.transform(X_test)
```

In [0]:
```
def train_model(X, X_test, y, params, folds, model_type='lgb', plot_feature_im
portance=False, averaging='usual', model=None):
    oof = np.zeros(len(X))
    prediction = np.zeros(len(X_test))
    scores = []
    feature_importance = pd.DataFrame()
    for fold_n, (train_index, valid_index) in enumerate(folds.split(X, y)):
        print('Fold', fold_n, 'started at', time.ctime())
        X_train, X_valid = X.loc[train_index], X.loc[valid_index]
        y_train, y_valid = y[train_index], y[valid_index]

        if model_type == 'lgb':
            train_data = lgb.Dataset(X_train, label=y_train)
            valid_data = lgb.Dataset(X_valid, label=y_valid)

            model = lgb.train(params,
                    train_data,
                    num_boost_round=20000,
                    valid_sets = [train_data, valid_data],
                    verbose_eval=1000,
                    early_stopping_rounds = 200)

            y_pred_valid = model.predict(X_valid)
            y_pred = model.predict(X_test, num_iteration=model.best_iteration)

        if model_type == 'xgb':
            train_data = xgb.DMatrix(data=X_train, label=y_train, feature_name
s=X_train.columns)
            valid_data = xgb.DMatrix(data=X_valid, label=y_valid, feature_name
s=X_train.columns)

            watchlist = [(train_data, 'train'), (valid_data, 'valid_data')]
            model = xgb.train(dtrain=train_data, num_boost_round=20000, evals=
watchlist, early_stopping_rounds=200, verbose_eval=500, params=params)
            y_pred_valid = model.predict(xgb.DMatrix(X_valid, feature_names=X_
train.columns), ntree_limit=model.best_ntree_limit)
            y_pred = model.predict(xgb.DMatrix(X_test, feature_names=X_train.c
olumns), ntree_limit=model.best_ntree_limit)

        if model_type == 'sklearn':
            model = model
            model.fit(X_train, y_train)
            y_pred_valid = model.predict_proba(X_valid).reshape(-1,)
            score = roc_auc_score(y_valid, y_pred_valid)
            # print(f'Fold {fold_n}. AUC: {score:.4f}.')
            # print('')

            y_pred = model.predict_proba(X_test)[:, 1]

        if model_type == 'glm':
            model = sm.GLM(y_train, X_train, family=sm.families.Binomial())
            model_results = model.fit()
            model_results.predict(X_test)
            y_pred_valid = model_results.predict(X_valid).reshape(-1,)
            score = roc_auc_score(y_valid, y_pred_valid)

            y_pred = model_results.predict(X_test)

        if model_type == 'cat':
            model = CatBoostClassifier(iterations=20000, learning_rate=0.05, l
oss_function='Logloss',  eval_metric='AUC', **params)
            model.fit(X_train, y_train, eval_set=(X_valid, y_valid), cat_featu
res=[], use_best_model=True, verbose=False)

            y_pred_valid = model.predict_proba(X_valid)[:, 1]
            y_pred = model.predict_proba(X_test)[:, 1]

        oof[valid_index] = y_pred_valid.reshape(-1,)
        scores.append(roc_auc_score(y_valid, y_pred_valid))

        if averaging == 'usual':
            prediction += y_pred
        elif averaging == 'rank':
            prediction += pd.Series(y_pred).rank().values

        if model_type == 'lgb':
            # feature importance
            fold_importance = pd.DataFrame()
            fold_importance["feature"] = X.columns
            fold_importance["importance"] = model.feature_importance()
            fold_importance["fold"] = fold_n + 1
            feature_importance = pd.concat([feature_importance, fold_importanc
e], axis=0)

    prediction /= n_fold

    print('CV mean score: {0:.4f}, std: {1:.4f}.'.format(np.mean(scores), np.s
td(scores)))

    if model_type == 'lgb':
        feature_importance["importance"] /= n_fold
        if plot_feature_importance:
            cols = feature_importance[["feature", "importance"]].groupby("feat
ure").mean().sort_values(
                by="importance", ascending=False)[:50].index

            best_features = feature_importance.loc[feature_importance.feature.
isin(cols)]

            plt.figure(figsize=(16, 12));
            sns.barplot(x="importance", y="feature", data=best_features.sort_v
alues(by="importance", ascending=False));
            plt.title('LGB Features (avg over folds)');

            return oof, prediction, feature_importance
        return oof, prediction, scores

    else:
        return oof, prediction, scores
```

In [0]:
```
# %%time
# model = linear_model.LogisticRegression(class_weight='balanced', penalty='l
2', C=0.1)
# oof_lr, prediction_lr, scores = train_model(X, X_test, y, params=None, folds
=folds, model_type='sklearn', model=model)
```

In [0]:
```
params = {'num_leaves': 8,
         'min_data_in_leaf': 42,
         'objective': 'binary',
         'max_depth': 16,
         'learning_rate': 0.0123,
         'boosting': 'gbdt',
         'bagging_freq': 5,
         'feature_fraction': 0.8201,
         'bagging_seed': 11,
         'reg_alpha': 1.728910519108444,
         'reg_lambda': 4.9847051755586085,
         'random_state': 42,
         'metric': 'auc',
         'verbosity': -1,
         'subsample': 0.81,
         'min_gain_to_split': 0.01077313523861969,
         'min_child_weight': 19.428902804238373,
         'num_threads': 4}
oof_lgb, prediction_lgb, scores = train_model(X, X_test, y, params=params, fol
ds=folds, model_type='lgb', plot_feature_importance=True)
```

```
Fold 0 started at Wed Feb 27 07:27:43 2019
Training until validation scores don't improve for 200 rounds.
[1000]  training's auc: 0.885014        valid_1's auc: 0.864492
[2000]  training's auc: 0.909721        valid_1's auc: 0.882867
[3000]  training's auc: 0.921202        valid_1's auc: 0.89042
[4000]  training's auc: 0.928088        valid_1's auc: 0.89445
[5000]  training's auc: 0.932895        valid_1's auc: 0.896647
[6000]  training's auc: 0.936977        valid_1's auc: 0.897681
```

In [0]:
```
sub = pd.read_csv('sample_submission.csv')
sub['target'] = prediction_lgb
sub.to_csv('lgb.csv', index=False)
```

ELI5

In [0]:
```
model = lgb.LGBMClassifier(**params, n_estimators = 20000, n_jobs = -1)
X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, str
atify=y)
model.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_valid, y_valid)],
verbose=1000, early_stopping_rounds=200)
```

In [0]:
```
eli5.show_weights(model, targets=[0, 1], feature_names=list(X_train.columns),
top=40, feature_filter=lambda x: x != '<BIAS>')
```

ELI5 didn't help up to eliminate features, but let's at least try to take top-100 and see how it helps.

In [0]:
```
top_features = [i for i in eli5.formatters.as_dataframe.explain_weights_df(mod
el).feature if 'BIAS' not in i][:100]
X1 = X[top_features]
X_train, X_valid, y_train, y_valid = train_test_split(X1, y, test_size=0.2, st
ratify=y)
model.fit(X_train, y_train, eval_set=[(X_train, y_train), (X_valid, y_valid)],
verbose=1000, early_stopping_rounds=200)
```

In [0]:
```
def calculate_metrics(model, X_train: pd.DataFrame() = None, y_train: pd.DataF
rame() = None, X_valid: pd.DataFrame() = None,
                      y_valid: pd.DataFrame() = None, columns: list = []) -> p
d.DataFrame():
    columns = columns if len(columns) > 0 else list(X_train.columns)
    train_pred = model.predict_proba(X_train[columns])
    valid_pred = model.predict_proba(X_valid[columns])
    f1 = 0
    best_t = 0
    for t in np.arange(0.1, 1, 0.05):
        valid_pr = (valid_pred[:, 1] > t).astype(int)
        valid_f1 = metrics.f1_score(y_valid, valid_pr)
        if valid_f1 > f1:
            f1 = valid_f1
            best_t = t

    t = best_t
    train_pr = (train_pred[:, 1] > t).astype(int)
    valid_pr = (valid_pred[:, 1] > t).astype(int)
    train_f1 = metrics.f1_score(y_train, train_pr)
    valid_f1 = metrics.f1_score(y_valid, valid_pr)
    score_df = []
    print(f'Best threshold: {t:.2f}. Train f1: {train_f1:.4f}. Valid f1: {vali
d_f1:.4f}.')
    score_df.append(['F1', np.round(train_f1, 4), np.round(valid_f1, 4)])
    train_r = metrics.recall_score(y_train, train_pr)
    valid_r = metrics.recall_score(y_valid, valid_pr)

    score_df.append(['Recall', np.round(train_r, 4), np.round(valid_r, 4)])
    train_p = metrics.precision_score(y_train, train_pr)
    valid_p = metrics.precision_score(y_valid, valid_pr)

    score_df.append(['Precision', np.round(train_p, 4), np.round(valid_p, 4)])
    train_roc = metrics.roc_auc_score(y_train, train_pred[:, 1])
    valid_roc = metrics.roc_auc_score(y_valid, valid_pred[:, 1])

    score_df.append(['ROCAUC', np.round(train_roc, 4), np.round(valid_roc, 4
)])
    train_apc = metrics.average_precision_score(y_train, train_pred[:, 1])
    valid_apc = metrics.average_precision_score(y_valid, valid_pred[:, 1])

    score_df.append(['APC', np.round(train_apc, 4), np.round(valid_apc, 4)])
    print(metrics.confusion_matrix(y_valid, valid_pr))
    score_df = pd.DataFrame(score_df, columns=['Metric', 'Train', 'Valid'])
    print(score_df)

    return score_df, t
```

In [0]:
```
_ = calculate_metrics(model, X_train, y_train, X_valid, y_valid)
```

Feature generation

Feature interaction

In [0]:
```
X = train.drop(['ID_code', 'target'], axis=1)
X_test = test.drop(['ID_code'], axis=1)

columns = top_features = [i for i in eli5.formatters.as_dataframe.explain_weig
hts_df(model).feature if 'BIAS' not in i][:20]
for col1 in tqdm_notebook(columns):
    for col2 in columns:
        X[col1 + '_' + col2] = X[col1] * X[col2]
        X_test[col1 + '_' + col2] = X_test[col1] * X_test[col2]
```

In [0]:
```
# oof_lgb, prediction_lgb_inter, scores = train_model(X, X_test, y, params=par
ams, folds=folds, model_type='lgb', plot_feature_importance=True)
```

In [0]:
```
# sub = pd.read_csv('sample_submission.csv')
# sub['target'] = prediction_lgb_inter
# sub.to_csv('lgb_inter.csv', index=False)
```

Scaling

! **Notice** scaling severely decreases score

In [0]:
```
X = train.drop(['ID_code', 'target'], axis=1)
X_test = test.drop(['ID_code'], axis=1)
scaler = StandardScaler()
X_train[X_train.columns] = scaler.fit_transform(X_train[X_train.columns])
X_test[X_train.columns] = scaler.transform(X_test[X_train.columns])
# oof_lgb, prediction_lgb_scaled, scores = train_model(X, X_test, y, params=pa
rams, folds=folds, model_type='lgb', plot_feature_importance=True)
# sub = pd.read_csv('sample_submission.csv')
# sub['target'] = prediction_lgb_scaled
# sub.to_csv('lgb_scaled.csv', index=False)
```

Statistics

In [0]:
```
# X = train.drop(['ID_code', 'target'], axis=1)
# X_test = test.drop(['ID_code'], axis=1)

# X['std'] = X.std(1)
# X_test['std'] = X_test.std(1)

# X['mean'] = X.mean(1)
# X_test['mean'] = X_test.mean(1)
# oof_lgb, prediction_lgb_stats, scores = train_model(X, X_test, y, params=par
ams, folds=folds, model_type='lgb', plot_feature_importance=True)
# sub = pd.read_csv('sample_submission.csv')
# sub['target'] = prediction_lgb_stats
# sub.to_csv('lgb_stats.csv', index=False)
```

Training with these features gives the same score on LB: 0.899

NN features

In [0]:
```
# %%time
# X = train.drop(['ID_code', 'target'], axis=1)
# X_test = test.drop(['ID_code'], axis=1)
# neigh = NearestNeighbors(3, n_jobs=-1)
# neigh.fit(X)

# dists, _ = neigh.kneighbors(X, n_neighbors=3)
# mean_dist = dists.mean(axis=1)
# max_dist = dists.max(axis=1)
# min_dist = dists.min(axis=1)

# X['mean_dist'] = mean_dist
# X['max_dist'] = max_dist
# X['min_dist'] = min_dist

# test_dists, _ = neigh.kneighbors(X_test, n_neighbors=3)

# test_mean_dist = test_dists.mean(axis=1)
# test_max_dist = test_dists.max(axis=1)
# test_min_dist = test_dists.min(axis=1)

# X_test['mean_dist'] = test_mean_dist
# X_test['max_dist'] = test_max_dist
# X_test['min_dist'] = test_min_dist

# oof_lgb, prediction_lgb_dist, scores = train_model(X, X_test, y, params=para
ms, folds=folds, model_type='lgb', plot_feature_importance=True)
# sub = pd.read_csv('sample_submission.csv')
# sub['target'] = prediction_lgb_dist
# sub.to_csv('lgb_dist.csv', index=False)
```

Blend

In [0]:
```
xgb_params = {'eta': 0.05, 'max_depth': 3, 'subsample': 0.9, 'colsample_bytre
e': 0.9,
          'objective': 'binary:logistic', 'eval_metric': 'auc', 'silent': True
```