BCA FINHACKS

Credit Scoring Modeling

Indra Pratama Putra MR, data_paradise team, 07.10.2018

1. Introduction

Objective: Creating a model to predict wheather a loan applicant will be 'flag_kredit_macet' or not.

Metrics to be optimized: AUC (Recall as an additional metric).

2. Load Data and Outlier Checking

2.1 Load Data

In [1]:

```
import pandas as pd
import numpy as np
import re
import seaborn as sns
import matplotlib.pyplot as plt
import xgboost as xgb
import random
import time
import datetime as dt
%matplotlib inline

from collections import Counter
from operator import itemgetter
from xgboost import plot_importance
from numpy import genfromtxt

from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, cross_val_score, StratifiedKFold, learning_curve
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier, ExtraTr
eesClassifier, VotingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.cross_validation import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.grid_search import GridSearchCV
from sklearn.metrics import average_precision_score
from sklearn import preprocessing
from sklearn.metrics import roc_curve, auc,recall_score,precision_score
```

```
/home/indraputramr/anaconda3/lib/python3.6/site-packages/sklearn/cross_validation.py:41: Deprec
ationWarning: This module was deprecated in version 0.18 in favor of the model_selection module
 into which all the refactored classes and functions are moved. Also note that the interface of
 the new CV iterators are different from that of this module. This module will be removed in 0.
20.
  "This module will be removed in 0.20.", DeprecationWarning)
/home/indraputramr/anaconda3/lib/python3.6/site-packages/sklearn/grid_search.py:42: Deprecation
Warning: This module was deprecated in version 0.18 in favor of the model_selection module into
 which all the refactored classes and functions are moved. This module will be removed in 0.20.
  DeprecationWarning)
```

In [2]:

```
pd.set_option('display.max_columns', 500)
```

In [3]:

```
train = (pd.read_csv('npl_train.csv')).rename(columns={'X': 'id'})
```

In [4]:

```
train.shape
```

Out[4]:

```
(15493, 24)
```

In [5]:

```
train.flag_kredit_macet.value_counts(normalize=True)
```

Out[5]:

```
0    0.912283
1    0.087717
Name: flag_kredit_macet, dtype: float64
```

In [6]:

```
train.dtypes
```

Out[6]:

```
id                                   int64
jumlah_kartu                         int64
outstanding                          int64
limit_kredit                       float64
tagihan                            float64
total_pemakaian_tunai              float64
total_pemakaian_retail             float64
sisa_tagihan_tidak_terbayar        float64
kode_cabang                         object
rasio_pembayaran                   float64
persentasi_overlimit               float64
rasio_pembayaran_3bulan            float64
rasio_pembayaran_6bulan            float64
skor_delikuensi                      int64
flag_kredit_macet                    int64
jumlah_tahun_sejak_pembukaan_kredit    float64
total_pemakaian                    float64
sisa_tagihan_per_jumlah_kartu      float64
sisa_tagihan_per_limit             float64
total_pemakaian_per_limit          float64
pemakaian_3bln_per_limit           float64
pemakaian_6bln_per_limit           float64
utilisasi_3bulan                   float64
utilisasi_6bulan                   float64
dtype: object
```

2.2 Outlier Checking

In [7]:

```
def detect_outliers(df,n,features):
    """
    Takes a dataframe df of features and returns a list of the indices
    corresponding to the observations containing more than n outliers according
    to the Tukey method.
    """
    outlier_id = []
    features_upper_outlier = []
    features_bottom_outlier = []

    # iterate over features(columns)
    for col in features:
        # 1st quartile (25%)
        Q1 = np.percentile(df[col], 25)
        # 3rd quartile (75%)
        Q3 = np.percentile(df[col],75)
        # Interquartile range (IQR)
        IQR = Q3 - Q1

        # outlier step
        outlier_step = 1.5 * IQR

        features_upper_outlier.append(Q3 + outlier_step)
        features_bottom_outlier.append(Q1 - outlier_step)

        # Determine a list of indices of outliers for feature col
        outlier_list_col = df[(df[col] < Q1 - outlier_step) | (df[col] > Q3 + outlier_step)].id

        # append the found outlier indices for col to the list of outlier indices
        outlier_id.extend(outlier_list_col)

    # select observations containing more than 2 outliers
    outlier_id_counter = Counter(outlier_id)
    multiple_outliers = list( k for k, v in outlier_id_counter.items() if v > n )

    return multiple_outliers
```

In [8]:

```
features_basic_numerical = list(train.drop(['id', 'flag_kredit_macet'], axis=1).select_dtypes(exclude=['obje
ct']))

outliers_to_drop = detect_outliers(train,3,features_basic_numerical)
```

Since outliers can have a dramatic effect on the prediction, they will be managed.

Tukey method (Tukey JW., 1977) is used to detect ouliers which defines an interquartile range comprised between the 1st and 3rd quartile of the distribution values (IQR). An outlier is a row that have a feature value outside the (IQR +- an outlier step).

The numerical values features (exclude 'kode_cabang') will be utilized to find outlier. For this case, also, since recall metric will be optimized, then ids with outliers number more than 3 and have 0 'flag kredit macet' will be removed.

In [9]:

```
id_to_drop = train[(train.id.isin(outliers_to_drop)) & (train.flag_kredit_macet == 0)].id.tolist()
```

In [10]:

```
train = train[~train.id.isin(id_to_drop)]
```

In [11]:

```
train.flag_kredit_macet.value_counts(normalize=True)
```

Out[11]:

```
0    0.899027
1    0.100973
Name: flag_kredit_macet, dtype: float64
```

By doing id removal from outlier analysis, the proportion of "flag_kredit_macet" becomes around 90% of 0 and 10% of 1, compared to previous condition, around 91% of 0 and 9% of 1.

2.3. Train-Test-Dataset

In [12]:

```
train_id = train.id.tolist()
train_target = train.flag_kredit_macet.tolist()

test = (pd.read_csv('npl_test.csv')).rename(columns={'X': 'id'})
test_id = test.id.tolist()

dataset = pd.concat([train, test])

features_basic = list(dataset.drop(['id', 'flag_kredit_macet'], axis=1))
features_basic_numerical = list(dataset.drop(['id', 'flag_kredit_macet'], axis=1).select_dtypes(exclude=['ob
ject']))
features_basic_object = list(dataset.drop(['id', 'flag_kredit_macet'], axis=1).select_dtypes(include=['objec
t']))

del dataset['flag_kredit_macet']
```

In [13]:

```
train.shape
```

Out[13]:

```
(13459, 24)
```

In [14]:

```
test.shape
```

Out[14]:

```
(2214, 23)
```

In [15]:

```
dataset.shape
```

Out[15]:

```
(15673, 23)
```

Dataset (train data + test data) will be utilized to analyze categorical features (kode_cabang).

3. Checking Missing Value & Categorical Feature Handling

In [16]:

```
dataset.isnull().sum()
```

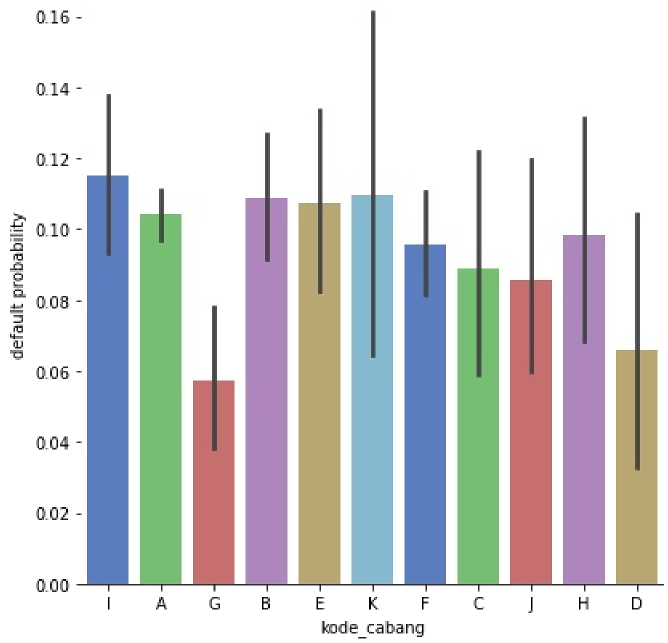
Out[16]:

```
id                                   0
jumlah_kartu                         0
jumlah_tahun_sejak_pembukaan_kredit  0
kode_cabang                         96
limit_kredit                         0
outstanding                          0
pemakaian_3bln_per_limit             0
pemakaian_6bln_per_limit             0
persentasi_overlimit                 0
rasio_pembayaran                     0
rasio_pembayaran_3bulan              0
rasio_pembayaran_6bulan              0
sisa_tagihan_per_jumlah_kartu        0
sisa_tagihan_per_limit               0
sisa_tagihan_tidak_terbayar          0
skor_delikuensi                      0
tagihan                              0
total_pemakaian                      0
total_pemakaian_per_limit            0
total_pemakaian_retail               0
total_pemakaian_tunai                0
utilisasi_3bulan                     0
utilisasi_6bulan                     0
dtype: int64
```

Since there are 96 missing values on kode_cabang, it will be handled by looking the train distribution of kode_cabang.

In [17]:

```
g = sns.factorplot(x="kode_cabang",y="flag_kredit_macet",data=train,kind="bar", size = 6, palette = "muted")
g.despine(left=True)
g = g.set_ylabels("default probability")
```



The highest probability to be "flag_kredit_macet" is given by I kode_cabang. Then missing value will be imputed by I kode_cabang.

In [18]:

```
dataset = dataset.fillna('I')
```

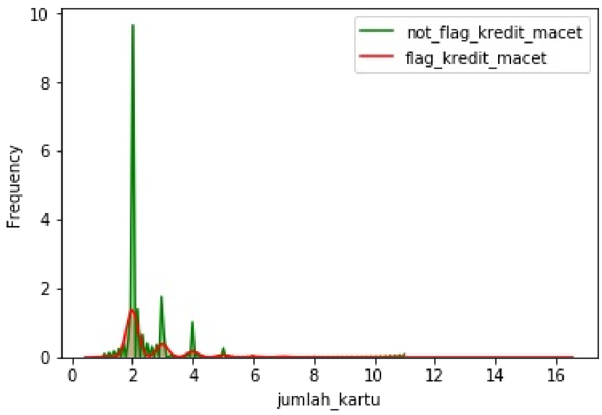
4. Feature engineering

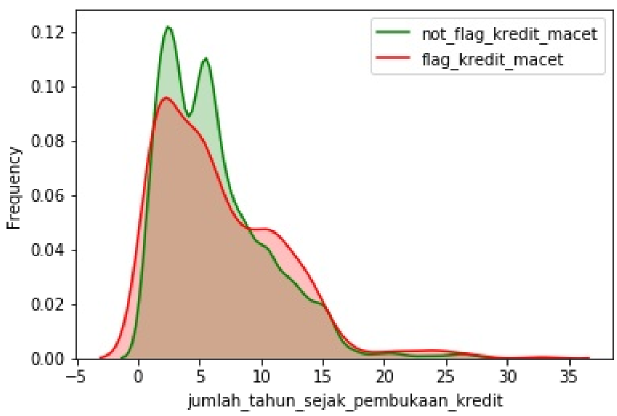
In [19]:

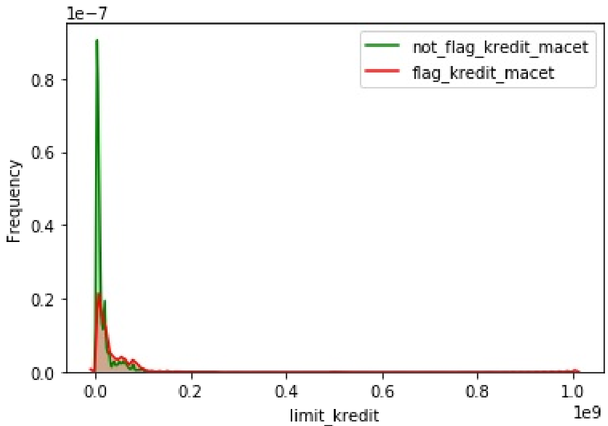
```
train_numerical = train.select_dtypes(exclude=['object'])
```

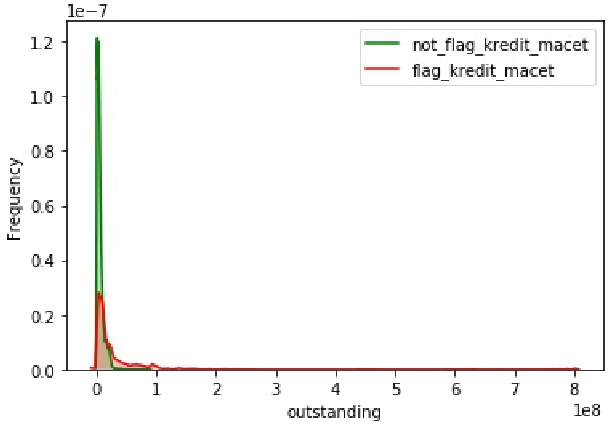
In [20]:

```
for i in features_basic:
    if i != 'flag_kredit_macet' and i != 'kode_cabang':
        g = sns.kdeplot(train_numerical[i][(train_numerical["flag_kredit_macet"] == 0) & (train_numerical[i]
.notnull())], color="Green", shade = True)
        g = sns.kdeplot(train_numerical[i][(train_numerical["flag_kredit_macet"] == 1) & (train_numerical[i]
.notnull())], ax =g, color="Red", shade= True)
        g.set_xlabel(i)
        g.set_ylabel("Frequency")
        g = g.legend(["not_flag_kredit_macet","flag_kredit_macet"])
        plt.show()
```

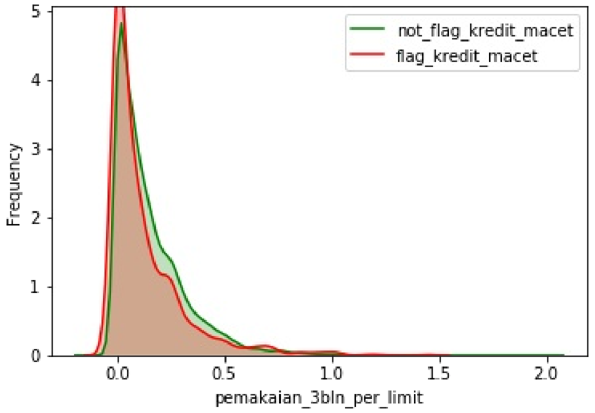




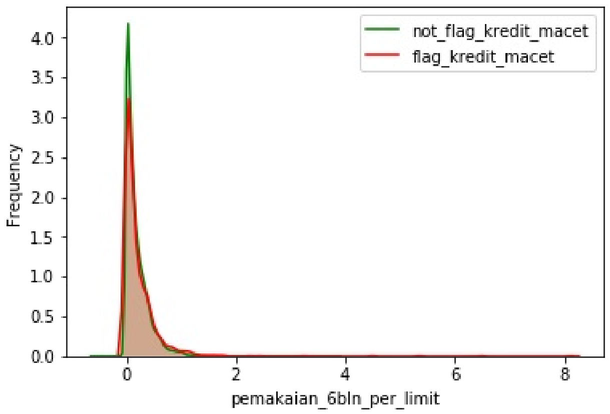




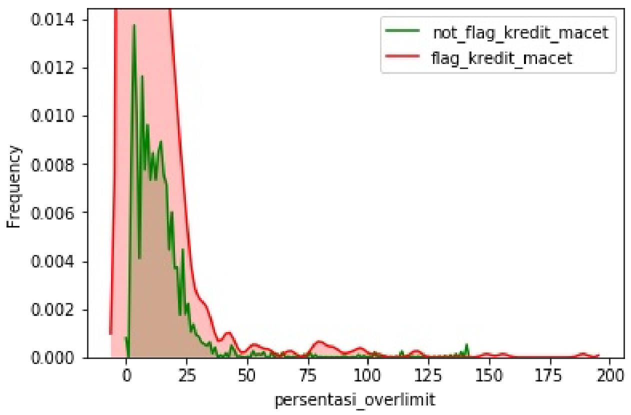
1e−7
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Frequency
not_flag_kredit_macet
flag_kredit_macet
0
1
2
3
4
5
6
7
8
1e8
outstanding



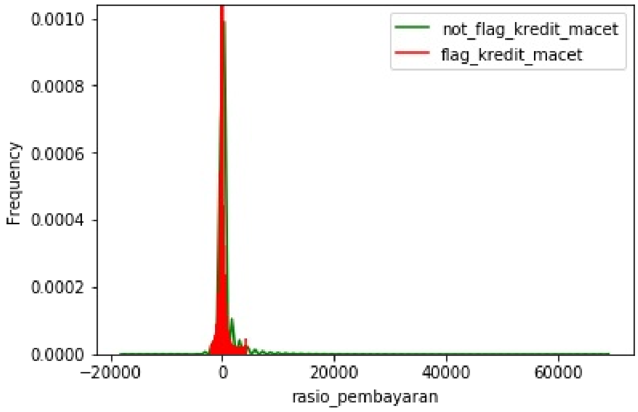
5
4
3
2
1
0
Frequency
not_flag_kredit_macet
flag_kredit_macet
0.0
0.5
1.0
1.5
2.0
pemakaian_3bln_per_limit

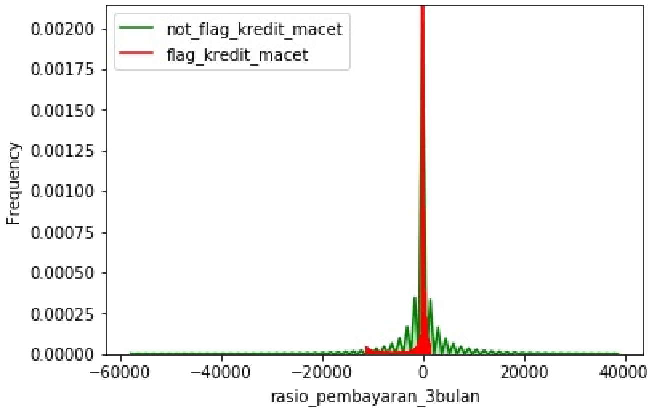


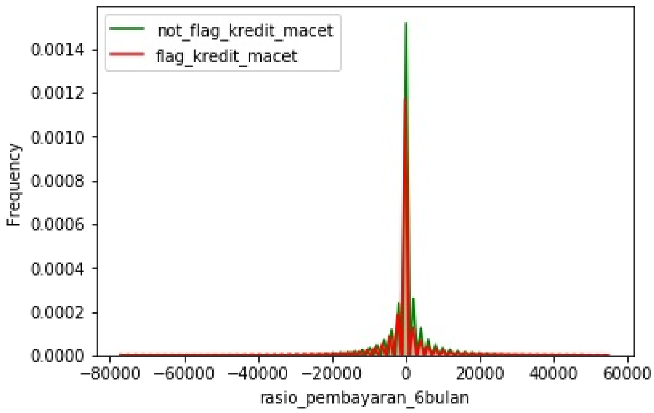
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Frequency
not_flag_kredit_macet
flag_kredit_macet
0
2
4
6
8
pemakaian_6bln_per_limit

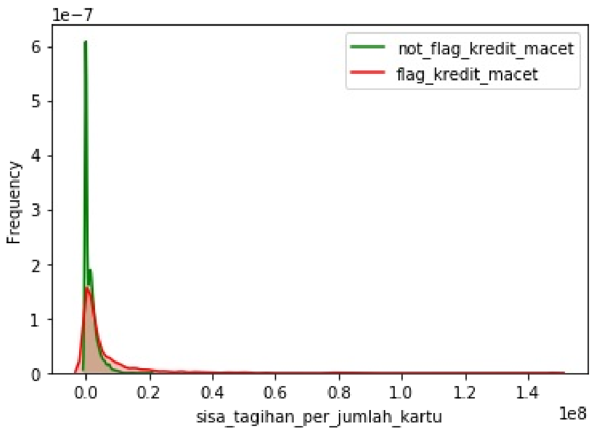


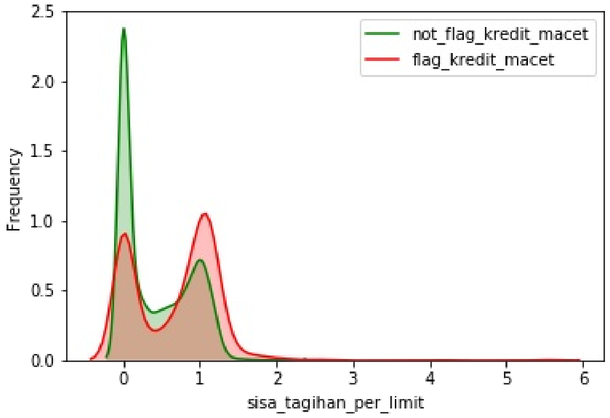
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Frequency
not_flag_kredit_macet
flag_kredit_macet
0
25
50
75
100
125
150
175
200
persentasi_overlimit

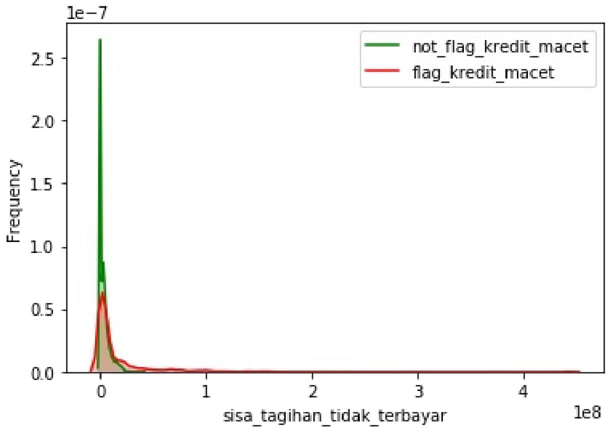

not_flag_kredit_macet
flag_kredit_macet
Frequency
rasio_pembayaran

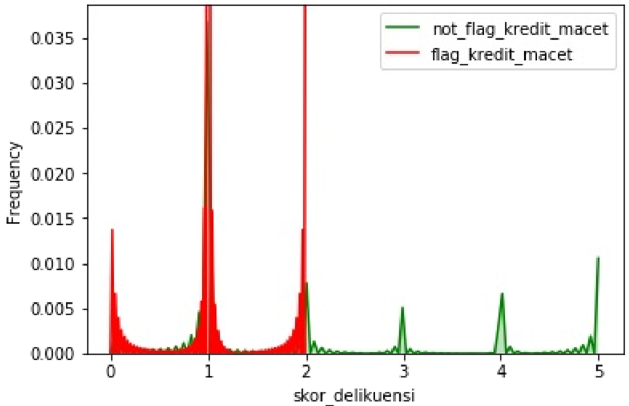

not_flag_kredit_macet
flag_kredit_macet
Frequency
rasio_pembayaran_3bulan

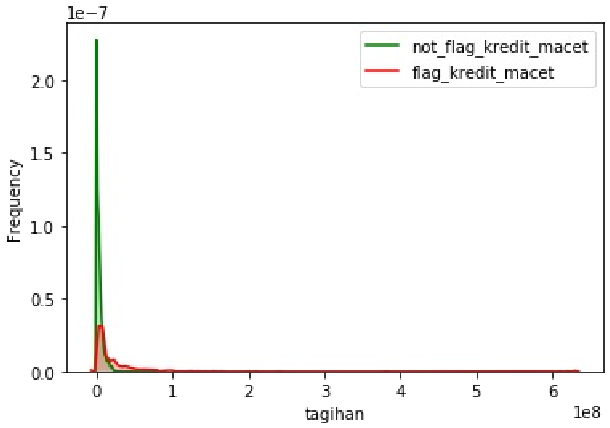

not_flag_kredit_macet
flag_kredit_macet
Frequency
rasio_pembayaran_6bulan

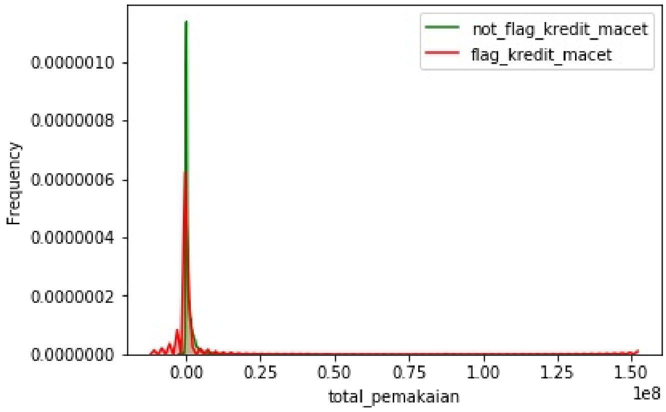

not_flag_kredit_macet
flag_kredit_macet
Frequency
sisa_tagihan_per_jumlah_kartu

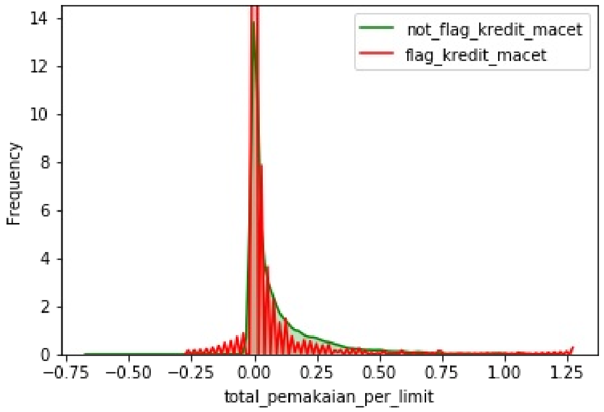

not_flag_kredit_macet
flag_kredit_macet
Frequency
sisa_tagihan_per_limit

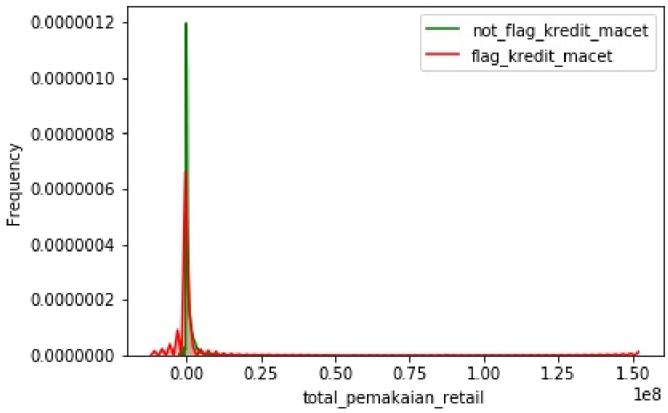

1e−7
not_flag_kredit_macet
flag_kredit_macet
Frequency
sisa_tagihan_tidak_terbayar
1e8

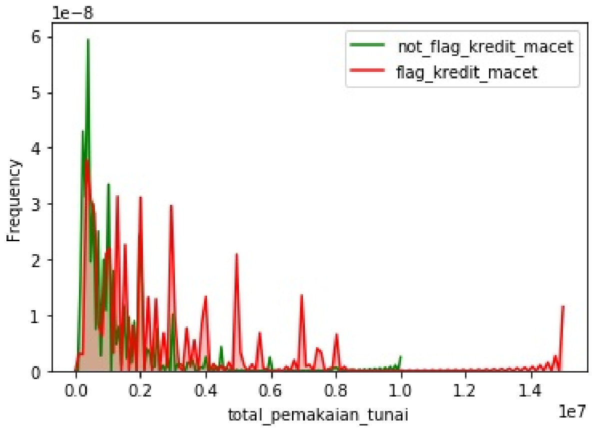

not_flag_kredit_macet
flag_kredit_macet
Frequency
skor_delikuensi

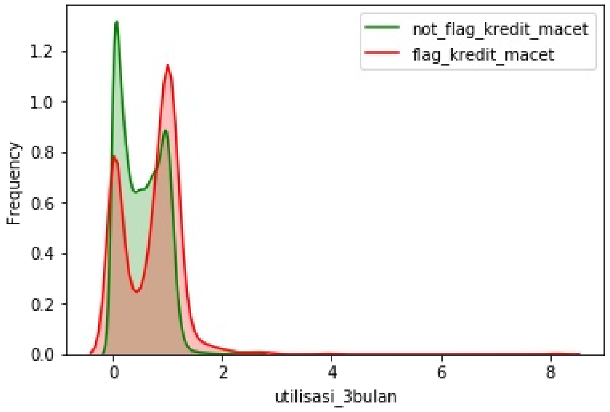

1e−7
not_flag_kredit_macet
flag_kredit_macet
Frequency
tagihan
1e8

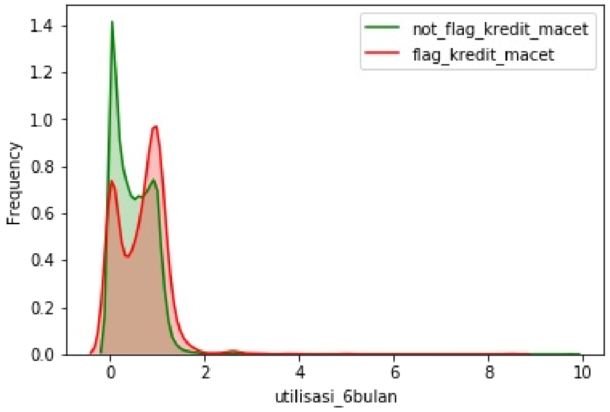

not_flag_kredit_macet
flag_kredit_macet
Frequency
total_pemakaian
1e8


not_flag_kredit_macet
flag_kredit_macet
Frequency
total_pemakaian_per_limit


not_flag_kredit_macet
flag_kredit_macet
Frequency
total_pemakaian_retail
1e8


1e−8
not_flag_kredit_macet
flag_kredit_macet
Frequency
total_pemakaian_tunai
1e7





Based on eyeballing observation through all graph and domain knowledge in credit product, several features are established as follows:

In [21]:

```
def feature_engineering_jumlah_kartu(df):
    df['jumlah_kartu_total_sisa_tagihan_semua_kartu'] = df.apply(lambda x: x['jumlah_kartu'] * x['sisa_tagih
an_per_jumlah_kartu'], axis=1)
    return df

def feature_engineering_outstanding(df):
    df['jumlah_kartu_total_sisa_tagihan_semua_kartu'] = df.apply(lambda x: x['jumlah_kartu'] * x['sisa_tagih
an_per_jumlah_kartu'], axis=1)
    return df

def feature_engineering_limit_kredit(df):
    df['limit_kredit_per_jumlah_kartu'] = df.apply(lambda x: x['limit_kredit'] / x['jumlah_kartu'], axis=1)
    df['limit_kredit_overlimit_maksimum'] = df.apply(lambda x: x['limit_kredit'] + (x['limit_kredit'] * x['p
ersentasi_overlimit']), axis=1)
    df['limit_kredit_total_sisa_tagihan_semua_limit'] = df.apply(lambda x: x['limit_kredit'] * x['sisa_tagih
an_per_limit'], axis=1)
    return df

def feature_engineering_jumlah_tahun_sejak_pembukaan_kredit(df):
    df['jumlah_tahun_sejak_pembukaan_kredit_average_quarter_utilization'] = df.apply(lambda x: x['jumlah_tah
un_sejak_pembukaan_kredit'] * 4 * x['utilisasi_3bulan'], axis=1)
    df['jumlah_tahun_sejak_pembukaan_kredit_average_semester_utilization'] = df.apply(lambda x: x['jumlah_ta
hun_sejak_pembukaan_kredit'] * 2 * x['utilisasi_6bulan'], axis=1)
    df['jumlah_tahun_sejak_pembukaan_kredit_average_quarter_utilization_per_limit'] = df.apply(lambda x: x['
jumlah_tahun_sejak_pembukaan_kredit'] * 4 * x['pemakaian_3bln_per_limit'], axis=1)
    df['jumlah_tahun_sejak_pembukaan_kredit_average_semester_utilization_per_limit'] = df.apply(lambda x: x[
'jumlah_tahun_sejak_pembukaan_kredit'] * 4 * x['pemakaian_6bln_per_limit'], axis=1)
    df['jumlah_tahun_sejak_pembukaan_kredit_rata_rata_waktu_pembukaan_kredit'] = df.apply(lambda x: x['jumla
h_tahun_sejak_pembukaan_kredit'] / x['jumlah_kartu'], axis=1)
    return df

def feature_engineering_persentasi_overlimit(df):
    df['persentasi_overlimit_excess'] = df.apply(lambda x: x['persentasi_overlimit'] * x['limit_kredit'], ax
is=1)
    return df

def feature_engineering_total_pemakaian(df):
    df['total_pemakaian_tunai_retail'] = df.apply(lambda x: x['total_pemakaian_tunai'] + x['total_pemakaian_
retail'], axis=1)
    df['total_pemakaian_per_limit_kredit'] = df.apply(lambda x: x['total_pemakaian'] / x['limit_kredit'], ax
is=1)
    df['total_pemakaian_per_jumlah_kartu'] = df.apply(lambda x: x['total_pemakaian'] / x['jumlah_kartu'], ax
is=1)
    df['total_pemakaian_unexpected'] = df.apply(lambda x: 1 if x['total_pemakaian'] <= 0.00 and x['total_pem
akaian_tunai'] <= 0.00 and x['total_pemakaian_retail'] <= 0.00 else 0, axis=1)
    return df

def feature_engineering_tagihan(df):
    df['tagihan_terbayar'] = df.apply(lambda x: x['tagihan'] - x['sisa_tagihan_tidak_terbayar'], axis=1)
    df['tagihan_per_limit_kredit'] = df.apply(lambda x: x['tagihan'] / x['limit_kredit'], axis=1)
    return df

def feature_engineering_sisa_tagihan(df):
    df['tagihan_per_jumlah_kartu'] = df.apply(lambda x: x['tagihan'] / x['jumlah_kartu'], axis=1)
    return df

def feature_engineering_utilisasi_3bulan(df):
    df['utilisasi_3bln_per_jumlah_kartu'] = df.apply(lambda x: x['utilisasi_3bulan'] / x['jumlah_kartu'], ax
is=1)
    return df

def feature_engineering_utilisasi_6bulan(df):
    df['utilisasi_6bln_per_jumlah_kartu'] = df.apply(lambda x: x['utilisasi_6bulan'] / x['jumlah_kartu'], ax
is=1)
    return df
```

In [22]:

```
dataset = feature_engineering_jumlah_kartu(dataset)
dataset = feature_engineering_outstanding(dataset)
dataset = feature_engineering_limit_kredit(dataset)
dataset = feature_engineering_jumlah_tahun_sejak_pembukaan_kredit(dataset)
dataset = feature_engineering_persentasi_overlimit(dataset)
dataset = feature_engineering_total_pemakaian(dataset)
dataset = feature_engineering_tagihan(dataset)
dataset = feature_engineering_sisa_tagihan(dataset)
dataset = feature_engineering_utilisasi_3bulan(dataset)
dataset = feature_engineering_utilisasi_6bulan(dataset)
```

5. One-Hot Encoding for Categorical Feature

In [23]:

```
kode_cabang_dummy = pd.get_dummies(dataset['kode_cabang'])
```

In [24]:

```
brach_name_list = []
for branch in list(kode_cabang_dummy):
    brach_name_list.append('kode_cabang_' + branch)
```

In [25]:

```
kode_cabang_dummy.columns = brach_name_list
```

In [26]:

```
dataset_new = pd.concat([dataset, kode_cabang_dummy], axis=1)
```

In [27]:

```
del dataset_new['kode_cabang']
```

In [28]:

```
dataset_new.isnull().sum()
```

Out[28]:

```
id                                                                    0
jumlah_kartu                                                          0
jumlah_tahun_sejak_pembukaan_kredit                                   0
limit_kredit                                                          0
outstanding                                                           0
pemakaian_3bln_per_limit                                              0
pemakaian_6bln_per_limit                                              0
persentasi_overlimit                                                  0
rasio_pembayaran                                                      0
rasio_pembayaran_3bulan                                               0
rasio_pembayaran_6bulan                                               0
sisa_tagihan_per_jumlah_kartu                                         0
sisa_tagihan_per_limit                                                0
sisa_tagihan_tidak_terbayar                                           0
skor_delikuensi                                                       0
tagihan                                                               0
total_pemakaian                                                       0
total_pemakaian_per_limit                                             0
total_pemakaian_retail                                                0
total_pemakaian_tunai                                                 0
utilisasi_3bulan                                                      0
utilisasi_6bulan                                                      0
jumlah_kartu_total_sisa_tagihan_semua_kartu                           0
limit_kredit_per_jumlah_kartu                                         0
limit_kredit_overlimit_maksimum                                       0
limit_kredit_total_sisa_tagihan_semua_limit                           0
jumlah_tahun_sejak_pembukaan_kredit_average_quarter_utilization       0
jumlah_tahun_sejak_pembukaan_kredit_average_semester_utilization      0
jumlah_tahun_sejak_pembukaan_kredit_average_quarter_utilization_per_limit     0
jumlah_tahun_sejak_pembukaan_kredit_average_semester_utilization_per_limit    0
jumlah_tahun_sejak_pembukaan_kredit_rata_rata_waktu_pembukaan_kredit          0
persentasi_overlimit_excess                                           0
total_pemakaian_tunai_retail                                          0
total_pemakaian_per_limit_kredit                                      0
total_pemakaian_per_jumlah_kartu                                      0
total_pemakaian_unexpected                                            0
tagihan_terbayar                                                      0
tagihan_per_limit_kredit                                              0
tagihan_per_jumlah_kartu                                              0
utilisasi_3bln_per_jumlah_kartu                                       0
utilisasi_6bln_per_jumlah_kartu                                       0
kode_cabang_A                                                         0
kode_cabang_B                                                         0
kode_cabang_C                                                         0
kode_cabang_D                                                         0
kode_cabang_E                                                         0
kode_cabang_F                                                         0
kode_cabang_G                                                         0
kode_cabang_H                                                         0
kode_cabang_I                                                         0
kode_cabang_J                                                         0
kode_cabang_K                                                         0
dtype: int64
```

6. Train-Test Split for Modeling

In [29]:

```python
train = dataset_new[dataset_new.id.isin(train_id)]
test = dataset_new[dataset_new.id.isin(test_id)]

del train['id']
del test['id']
```

In [30]:

```python
train.shape
```

Out[30]:

```
(13459, 51)
```

In [31]:

```
test.shape
```

Out[31]:

```
(2214, 51)
```

7. Modeling

7.1 First Modeling - Sklearn ML Package Compilation

Before use complex model, data will be trained using simple model, Sklearn ML package compilation.

In [32]:

```
kfold = StratifiedKFold(n_splits=10)
```

In [33]:

```
random_state = 2
classifiers = []
classifiers.append(SVC(random_state=random_state))
classifiers.append(DecisionTreeClassifier(random_state=random_state))
classifiers.append(AdaBoostClassifier(DecisionTreeClassifier(random_state=random_state),random_state=random_
state,learning_rate=0.1))
classifiers.append(RandomForestClassifier(random_state=random_state))
classifiers.append(ExtraTreesClassifier(random_state=random_state))
classifiers.append(GradientBoostingClassifier(random_state=random_state))
classifiers.append(MLPClassifier(random_state=random_state))
classifiers.append(KNeighborsClassifier())
classifiers.append(LogisticRegression(random_state = random_state))
classifiers.append(LinearDiscriminantAnalysis())
```

In [34]:

```
cv_results = []
for classifier in classifiers :
    cv_results.append(cross_val_score(classifier, train, y = train_target, scoring = "recall", cv = kfold, n
_jobs=4))
```

```
/home/indraputramr/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388:
UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
/home/indraputramr/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388:
UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
/home/indraputramr/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388:
UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
/home/indraputramr/anaconda3/lib/python3.6/site-packages/sklearn/discriminant_analysis.py:388:
UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
```

In [35]:

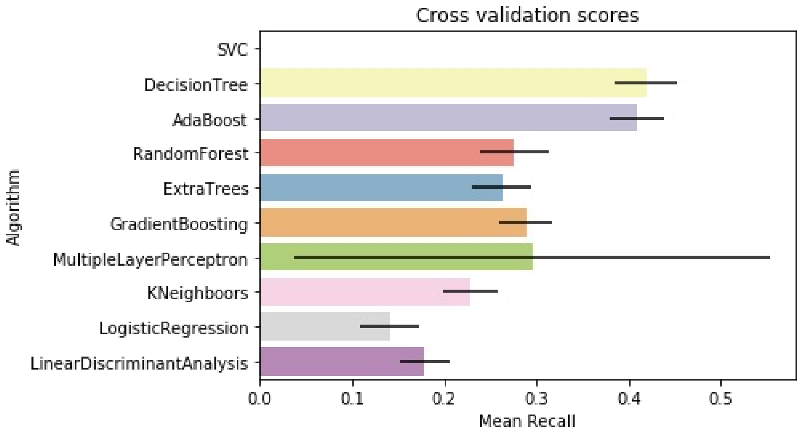
```
cv_means = []
cv_std = []
for cv_result in cv_results:
    cv_means.append(cv_result.mean())
    cv_std.append(cv_result.std())
```

In [36]:

```
cv_res = pd.DataFrame({"CrossValMeans":cv_means,"CrossValerrors": cv_std,"Algorithm":["SVC","DecisionTree","
AdaBoost",
"RandomForest","ExtraTrees","GradientBoosting","MultipleLayerPerceptron","KNeighboors","LogisticRegression",
"LinearDiscriminantAnalysis"]})
```

In [37]:

```
g = sns.barplot("CrossValMeans","Algorithm",data = cv_res, palette="Set3",orient = "h",**{'xerr':cv_std})
g.set_xlabel("Mean Recall")
g = g.set_title("Cross validation scores")
```



Based on those cross validation scores diagram, tree based algorithms give the highest mean recall. Then, next modeling will use another complex and solid algorithm of tree base algorithm.

7.2 Second Modeling - XGboost in action

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way.

In [38]:

```
def create_roc_curve(values, check):
    fpr, tpr, _ = roc_curve(values, check)
    roc_auc = auc(fpr, tpr)
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange',
             lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([-0.02, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC curve')
    plt.legend(loc="lower right")
    plt.show()
    print('')

def create_feature_map(features):
    outfile = open('xgb.fmap', 'w')
    for i, feat in enumerate(features):
        outfile.write('{0}\t{1}\tq\n'.format(i, feat))
    outfile.close()

def get_importance(gbm, features):
    create_feature_map(features)
    importance = gbm.get_fscore()
    importance = sorted(importance.items(), key=itemgetter(1), reverse=True)
    return importance

def run_single(train, test, features, target, random_state):
    eta = 0.1
    max_depth= 6
    subsample = 1
    colsample_bytree = 1
    min_chil_weight=1
    start_time = time.time()

    print('XGBoost params. ETA: {}, MAX_DEPTH: {}, SUBSAMPLE: {}, COLSAMPLE_BY_TREE: {}'.format(eta, max_dep
th, subsample, colsample_bytree))
    params = {
        "objective": "binary:logistic",
        "booster" : "gbtree",
        "eval_metric": "auc",
```

```
        "eval_metric": "auc",
        "eta": eta,
        "tree_method": 'exact',
        "max_depth": max_depth,
        "subsample": subsample,
        "colsample_bytree": colsample_bytree,
        "silent": 1,
        "min_chil_weight":min_chil_weight,
        "seed": random_state
    }
    num_boost_round = 500
    early_stopping_rounds = 10
    test_size = 0.1

    ### Creating XGB model

    X_train, X_valid, y_train, y_valid = train_test_split(train[features],
                                                          train[target],
                                                          stratify=train[target],
                                                          test_size=test_size,
                                                          random_state=random_state)

    print('Length train:', len(X_train.index))
    print('Length valid:', len(X_valid.index))

    dtrain = xgb.DMatrix(X_train, y_train)
    dvalid = xgb.DMatrix(X_valid, y_valid)

    watchlist = [(dtrain, 'train'), (dvalid, 'eval')]
    model = xgb.train(params, dtrain, num_boost_round, evals=watchlist, early_stopping_rounds=early_stopping
_rounds, verbose_eval=True)

    print("Validating...")
    check = model.predict(xgb.DMatrix(X_valid[features]), ntree_limit=model.best_iteration+1)

    imp = get_importance(model, features)

    print('##### ROC Curve for trained data #####')
    create_roc_curve(y_valid, check)

    print('Cross validation process for 5 k-folds...')
    cv = xgb.cv(params, dtrain, 5000, nfold=5, early_stopping_rounds=10, verbose_eval=1)
    print('Cross validation auc: '+str(cv['test-auc-mean'].tolist()[-1]))

    ##################################################

    print('Training time: {} minutes'.format(round((time.time() - start_time)/60, 2)))
    print('')

    print("Predict test set... ")
    X_test = test
    y_test = X_test[target]
    print('Length test:', len(X_test.index))
    dtest = xgb.DMatrix(X_test[features], y_test)
    test_prediction = model.predict(dtest, ntree_limit=model.best_iteration+1)

    ############################################ ROC Curve

    # Compute micro-average ROC curve and ROC area
    print('##### ROC Curve for test data #####')
    create_roc_curve(X_test[target].values, test_prediction)

    return test_prediction, imp, model.best_iteration+1, cv, check, model
```

In [39]:

```
X_train, X_oos, y_train, y_oos = train_test_split(train, train_target,
                                                  stratify=train_target,
                                                  test_size=0.1,
                                                  random_state=42)

df_train = pd.DataFrame(X_train)
df_train['flag_kredit_macet'] = y_train

df_oos = pd.DataFrame(X_oos)
df_oos['flag_kredit_macet'] = y_oos
```

In [40]:

```
df_train.shape
```

Out[40]:

```
(12113, 52)
```

In [41]:

```
df_oos.shape
```

Out[41]:

```
(1346, 52)
```

In [42]:

```
df_train.flag_kredit_macet.value_counts(normalize=True)
```

Out[42]:

```
0    0.899034
1    0.100966
Name: flag_kredit_macet, dtype: float64
```

In [43]:

```
df_oos.flag_kredit_macet.value_counts(normalize=True)
```

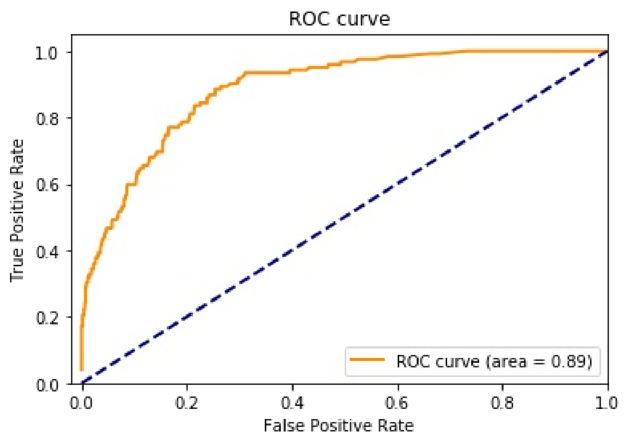
Out[43]:

```
0    0.89896
1    0.10104
Name: flag_kredit_macet, dtype: float64
```

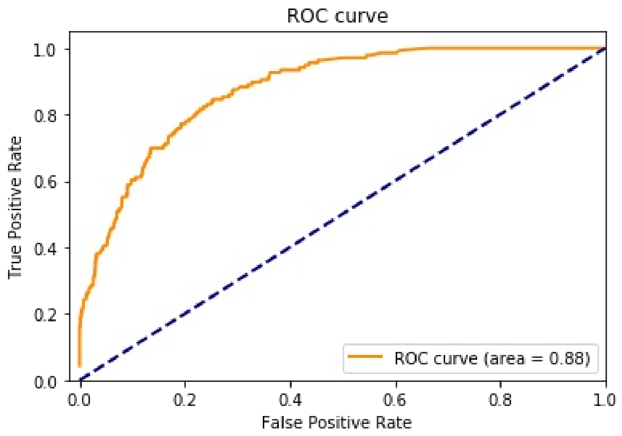
In [44]:

```
start_time = dt.datetime.now()
print("Start time: ",start_time)

features = list(df_train)
features.remove('flag_kredit_macet')

print("Building model.. ",dt.datetime.now()-start_time)
preds, imp, num_boost_rounds, cv, init, model = run_single(df_train, df_oos, features, 'flag_kredit_macet',
42)

print(dt.datetime.now()-start_time)
```

```
Start time:  2018-10-08 01:50:03.478891
Building model..  0:00:00.002882
XGBoost params. ETA: 0.1, MAX_DEPTH: 6, SUBSAMPLE: 1, COLSAMPLE_BY_TREE: 1
Length train: 10901
Length valid: 1212
[0]	train-auc:0.848978	eval-auc:0.841927
Multiple eval metrics have been passed: 'eval-auc' will be used for early stopping.

Will train until eval-auc hasn't improved in 10 rounds.
[1]	train-auc:0.87909	eval-auc:0.865209
[2]	train-auc:0.886651	eval-auc:0.870755
[3]	train-auc:0.887403	eval-auc:0.870379
[4]	train-auc:0.890501	eval-auc:0.870875
[5]	train-auc:0.893393	eval-auc:0.872142
[6]	train-auc:0.899272	eval-auc:0.876282
[7]	train-auc:0.901909	eval-auc:0.877399
[8]	train-auc:0.903931	eval-auc:0.879128
[9]	train-auc:0.904797	eval-auc:0.881636
[10]	train-auc:0.906655	eval-auc:0.882027
[11]	train-auc:0.90779	eval-auc:0.882599
[12]	train-auc:0.909675	eval-auc:0.883061
[13]	train-auc:0.910423	eval-auc:0.882287
[14]	train-auc:0.913692	eval-auc:0.884381
[15]	train-auc:0.915957	eval-auc:0.884689
[16]	train-auc:0.918448	eval-auc:0.886118
[17]	train-auc:0.920943	eval-auc:0.886806
[18]	train-auc:0.922068	eval-auc:0.886276
[19]	train-auc:0.923755	eval-auc:0.885114
[20]	train-auc:0.925509	eval-auc:0.884332
[21]	train-auc:0.928395	eval-auc:0.884344
[22]	train-auc:0.929488	eval-auc:0.883633
[23]	train-auc:0.931069	eval-auc:0.88452
[24]	train-auc:0.933214	eval-auc:0.884667
[25]	train-auc:0.935547	eval-auc:0.886066
[26]	train-auc:0.937041	eval-auc:0.88552
[27]	train-auc:0.937905	eval-auc:0.885035
Stopping. Best iteration:
[17]	train-auc:0.920943	eval-auc:0.886806

Validating...
##### ROC Curve for trained data #####
```



```
Cross validation process for 5 k-folds...
[0]	train-auc:0.856072+0.0107729	test-auc:0.827939+0.0139379
[1]	train-auc:0.874915+0.00591428	test-auc:0.834609+0.0142979
[2]	train-auc:0.882407+0.00755215	test-auc:0.842434+0.0122805
[3]	train-auc:0.888847+0.00358189	test-auc:0.847596+0.0120812
[4]	train-auc:0.892717+0.00387444	test-auc:0.851196+0.0102341
[5]	train-auc:0.895877+0.00334812	test-auc:0.854931+0.0105712
[6]	train-auc:0.899072+0.0031591	test-auc:0.856882+0.0117833
[7]	train-auc:0.9035+0.00315103	test-auc:0.860988+0.0121779
[8]	train-auc:0.906891+0.00396754	test-auc:0.863595+0.0125207
[9]	train-auc:0.908764+0.00360717	test-auc:0.862985+0.0125438
[10]	train-auc:0.911302+0.00385844	test-auc:0.864149+0.0118979
[11]	train-auc:0.912824+0.00387228	test-auc:0.864137+0.0122613
[12]	train-auc:0.914454+0.00361125	test-auc:0.864147+0.0124274
[13]	train-auc:0.916123+0.00342914	test-auc:0.864684+0.0120051
[14]	train-auc:0.917739+0.00353183	test-auc:0.86548+0.0127766
[15]	train-auc:0.920705+0.00266666	test-auc:0.866897+0.0117855
[16]	train-auc:0.922808+0.00207233	test-auc:0.868112+0.0124384
[17]	train-auc:0.925178+0.00248249	test-auc:0.869497+0.0117333
[18]	train-auc:0.927206+0.00214243	test-auc:0.869347+0.0117348
[19]	train-auc:0.92954+0.00195601	test-auc:0.86956+0.0113149
[20]	train-auc:0.930877+0.00200559	test-auc:0.870501+0.00963598
[21]	train-auc:0.93252+0.00190381	test-auc:0.871089+0.0101818
[22]	train-auc:0.934792+0.0011988	test-auc:0.87182+0.0098242
[23]	train-auc:0.936068+0.00140486	test-auc:0.872122+0.00938682
[24]	train-auc:0.938143+0.00129843	test-auc:0.873131+0.00863239
[25]	train-auc:0.939902+0.00164424	test-auc:0.873712+0.00837721
[26]	train-auc:0.941486+0.00152294	test-auc:0.873802+0.00814276
[27]	train-auc:0.943645+0.00102396	test-auc:0.874102+0.00803386
[28]	train-auc:0.945344+0.00114987	test-auc:0.874163+0.00803382
[29]	train-auc:0.946668+0.0014136	test-auc:0.874619+0.00778573
[30]	train-auc:0.948475+0.000814141	test-auc:0.875098+0.00787224
[31]	train-auc:0.950642+0.00102395	test-auc:0.875277+0.00811969
[32]	train-auc:0.951678+0.000945267	test-auc:0.87513+0.00877421
[33]	train-auc:0.952942+0.000893152	test-auc:0.875286+0.00856071
[34]	train-auc:0.954301+0.000902498	test-auc:0.875769+0.00860208
[35]	train-auc:0.955631+0.00102636	test-auc:0.875609+0.0087833
[36]	train-auc:0.957074+0.00098838	test-auc:0.875774+0.00875809
[37]	train-auc:0.958088+0.0010152	test-auc:0.875772+0.00875621
[38]	train-auc:0.95902+0.00117828	test-auc:0.875903+0.00906792
[39]	train-auc:0.959949+0.000861385	test-auc:0.875972+0.00897348
[40]	train-auc:0.96137+0.000759586	test-auc:0.876153+0.00899297
[41]	train-auc:0.962186+0.000695913	test-auc:0.87632+0.0093635
[42]	train-auc:0.963189+0.000853755	test-auc:0.876389+0.00934854
[43]	train-auc:0.963917+0.000730281	test-auc:0.876457+0.00917902
[44]	train-auc:0.965053+0.000744107	test-auc:0.876432+0.00936675
[45]	train-auc:0.966038+0.00109105	test-auc:0.876365+0.00953971
[46]	train-auc:0.967148+0.00102805	test-auc:0.876748+0.00916639
[47]	train-auc:0.96841+0.00135783	test-auc:0.876425+0.00916359
[48]	train-auc:0.969112+0.00147249	test-auc:0.876391+0.0090753
[49]	train-auc:0.970067+0.00140885	test-auc:0.876461+0.009335
[50]	train-auc:0.970759+0.00147267	test-auc:0.876445+0.00937878
[51]	train-auc:0.971377+0.00142249	test-auc:0.876372+0.00929729
[52]	train-auc:0.972017+0.00128958	test-auc:0.876537+0.00930044
[53]	train-auc:0.97244+0.00129447	test-auc:0.876466+0.00924897
[54]	train-auc:0.972909+0.00128371	test-auc:0.876465+0.0093292
[55]	train-auc:0.973354+0.00124918	test-auc:0.876377+0.00939072
Cross validation auc: 0.8767475999999998
Training time: 0.12 minutes

Predict test set...
Length test: 1346
##### ROC Curve for test data #####
```




```
0:00:07.309268
```

1. At this training, data is splited into train and test data. Train data is splited into X_train and X_valid data.
2. XGB model is built using X_train.
3. Test AUC is gained from test data which shooted by XGB model.
4. Cross validation AUC is gained from cross validation process within X_train. The algorithm choose train-test chunk within X_train by itself.
5. Cross validation AUC is 0.87. Test AUC is 0.88.

In [45]:

```
feature_importance = pd.DataFrame(imp)
feature_importance.columns = ['feature', 'importance_score']
```

In [46]:

```
feature_importance.shape
```

Out[46]:

```
(43, 2)
```

In [47]:

```
feature_importance.head()
```

Out[47]:

	feature	importance_score
0	outstanding	78
1	rasio_pembayaran_3bulan	76
2	rasio_pembayaran_6bulan	63
3	tagihan	62
4	jumlah_tahun_sejak_pembukaan_kredit_average_qu...	61

Top 5 features importances are shown as above.

8. Threshold Analysis

Threshold will be determined based on p0 of test distribution, rate of 'flag_kredit_macet' and recall percentage.

1. p0 Analysis

In [48]:

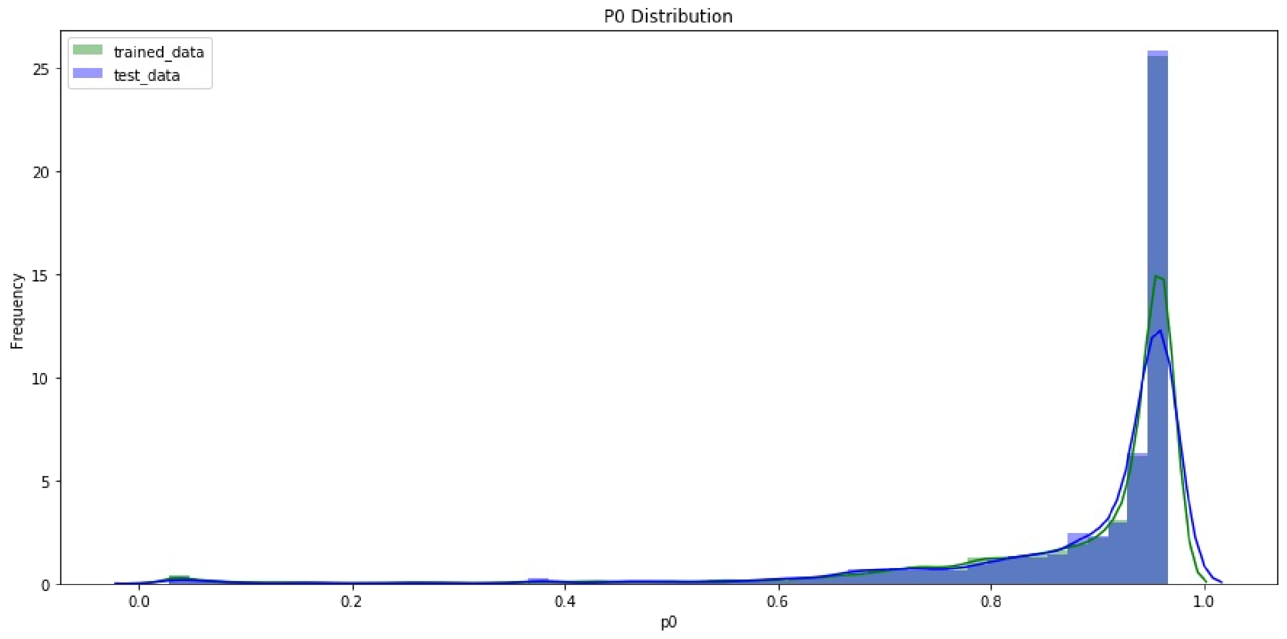
```
dtest = xgb.DMatrix(df_train[features])
preds_raw = model.predict(dtest)

preds = []
for i in preds_raw:
    preds.append(1-i)

dtest = xgb.DMatrix(df_oos[features])
check_raw = model.predict(dtest)

check = []
for i in check_raw:
    check.append(1-i)
```

In [49]:

```
fig, ax1 = plt.subplots(nrows=1, ncols=1, figsize=(15, 7))

sns.distplot(preds, ax = ax1, color="Green", hist=True)
sns.distplot(check, ax = ax1, color="Blue", hist=True)
plt.xlabel("p0")
plt.ylabel("Frequency")
plt.title("P0 Distribution")
plt.legend(["trained_data","test_data"])
```

Out[49]:

```
<matplotlib.legend.Legend at 0x7fb9f13fcfd0>
```



Based on p0 distribution, it can be seen that when both of them are superimposed, they are close. It meanse that model distribution on trained data is similar on test data.

2. Test vs NPL-Recall

In [50]:

```
bin_num = 41
bin_border_bottom = 0.4
bin_border_top = 0.96
bin_size = np.linspace(bin_border_bottom, bin_border_top, num=bin_num, retstep=True)[0].tolist()
bin_interval_bottom = bin_size[:-1]
bin_interval_top = bin_size[1:]
df_bin = pd.DataFrame()
df_bin['interval_num'] = list(range(bin_num-1))
df_bin['bin_bottom'] = [np.nan] + bin_interval_bottom[1:]
df_bin['bin_top'] = bin_interval_top[:-1] + [np.nan]

diff = df_bin.bin_top.iloc[1] - df_bin.bin_bottom.iloc[1]
```

In [51]:

```
df_test_pop = pd.DataFrame()
df_test_pop['p0'] = check
df_test_pop['flag_kredit_macet'] = y_oos
```

In [52]:

```
df_test_pop.shape
```

Out[52]:

```
(1346, 2)
```

In [53]:

```
bin_count = []
default_true_count = []
for i in range(0, len(df_test_pop)):
    if bin_border_bottom + diff > df_test_pop.p0.iloc[i]:
        bin_count.append(0)
        default_true_count.append(df_test_pop.flag_kredit_macet.iloc[i])
    elif bin_border_top - diff <= df_test_pop.p0.iloc[i]:
        bin_count.append(len(df_bin)-1)
        default_true_count.append(df_test_pop.flag_kredit_macet.iloc[i])
    else:
        for j in range(1, len(df_bin)-1):
            if df_bin.bin_bottom.iloc[j] <= df_test_pop.p0.iloc[i] and df_bin.bin_top.iloc[j] > df_test_pop.
p0.iloc[i]:
                bin_count.append(j)
                default_true_count.append(df_test_pop.flag_kredit_macet.iloc[i])
                break

df_bin_count = pd.DataFrame()
df_bin_count['bin_count'] = bin_count
df_bin_count['default_true_count'] = default_true_count

tmp_df_1 = df_bin_count.groupby('bin_count').count().reset_index()
tmp_df_1.columns = ['interval_num', 'flag_kredit_macet_count']
tmp_df_2 = df_bin_count.groupby('bin_count').sum().reset_index()
tmp_df_2.columns = ['interval_num', 'flag_kredit_macet_sum']

df_bin = df_bin.merge(tmp_df_1, on='interval_num', how='left')
df_bin = df_bin.merge(tmp_df_2, on='interval_num', how='left')
```

In [54]:

```
df_bin.head()
```

Out[54]:

	interval_num	bin_bottom	bin_top	flag_kredit_macet_count	flag_kredit_macet_sum
0	0	NaN	0.414	31.0	28.0
1	1	0.414	0.428	NaN	NaN
2	2	0.428	0.442	1.0	1.0
3	3	0.442	0.456	2.0	0.0
4	4	0.456	0.470	2.0	0.0

In [55]:

```
df_bin['count_id_test_%'] = df_bin['flag_kredit_macet_count'].apply(lambda x: np.nan if str(x) == 'nan' else
 x/len(df_test_pop))
df_bin['flag_test_%'] = df_bin.apply(lambda x: np.nan if str(x['flag_kredit_macet_sum']) == 'nan' else x['fl
ag_kredit_macet_sum']/x['flag_kredit_macet_count'], axis=1)
```

In [56]:

```
df_bin.head()
```

Out[56]:

	interval_num	bin_bottom	bin_top	flag_kredit_macet_count	flag_kredit_macet_sum	count_id_test_%	flag_test_%
0	0	NaN	0.414	31.0	28.0	0.023031	0.90322
1	1	0.414	0.428	NaN	NaN	NaN	Na
2	2	0.428	0.442	1.0	1.0	0.000743	1.00000
3	3	0.442	0.456	2.0	0.0	0.001486	0.00000
4	4	0.456	0.470	2.0	0.0	0.001486	0.00000

In [ ]:

```
recall = []
for i in df_bin.bin_top[:-1]:
    true_positive = 0
    true_negative = 0
    false_positive = 0
    false_negative = 0

    obs_df = df_test_pop[df_test_pop.p0 < i]
    obs_df['predict'] = [1] * len(obs_df)
    true_positive += len(obs_df[(obs_df.flag_kredit_macet == 1) & (obs_df.predict == 1)])
    true_negative += len(obs_df[(obs_df.flag_kredit_macet == 0) & (obs_df.predict == 0)])
    false_positive += len(obs_df[(obs_df.flag_kredit_macet == 0) & (obs_df.predict == 1)])
    false_negative += len(obs_df[(obs_df.flag_kredit_macet == 1) & (obs_df.predict == 0)])

    obs_df = df_test_pop[df_test_pop.p0 >= i]
    obs_df_2 = len(obs_df)
    obs_df['predict'] = [0] * len(obs_df)
    true_positive += len(obs_df[(obs_df.flag_kredit_macet == 1) & (obs_df.predict == 1)])
    true_negative += len(obs_df[(obs_df.flag_kredit_macet == 0) & (obs_df.predict == 0)])
    false_positive += len(obs_df[(obs_df.flag_kredit_macet == 0) & (obs_df.predict == 1)])
    false_negative += len(obs_df[(obs_df.flag_kredit_macet == 1) & (obs_df.predict == 0)])

    recall.append(true_positive / (true_positive + false_negative))

obs_df = df_test_pop[df_test_pop.p0 < 1]
obs_df['predict'] = [1] * len(obs_df)
true_positive += len(obs_df[(obs_df.flag_kredit_macet == 1) & (obs_df.predict == 1)])
true_negative += len(obs_df[(obs_df.flag_kredit_macet == 0) & (obs_df.predict == 0)])
false_positive += len(obs_df[(obs_df.flag_kredit_macet == 0) & (obs_df.predict == 1)])
false_negative += len(obs_df[(obs_df.flag_kredit_macet == 1) & (obs_df.predict == 0)])

recall.append(true_positive / (true_positive + false_negative))
```

```
/home/indraputramr/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:9: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
#indexing-view-versus-copy
/home/indraputramr/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:17: SettingWithC
opyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
#indexing-view-versus-copy
```

In [ ]:

```
df_bin['recall_%'] = recall
obs_result = df_bin[['bin_top', 'flag_test_%', 'recall_%']].sort_values(by=['flag_test_%', 'recall_%'], asce
nding=False)
```

In [ ]:

```
df_bin = df_bin.fillna(0)
```

In [ ]:

```
df_bin.head()
```

In [ ]:

```
fig, ax1 = plt.subplots(nrows=1, ncols=1, figsize=(15, 7))

sns.barplot(x=df_bin.interval_num, y=df_bin.flag_kredit_macet_count, palette="rocket", ax=ax1)
ax1.twinx()
plt.plot(df_bin.interval_num, df_bin['flag_test_%'], linewidth=1.0)
plt.plot(df_bin.interval_num, df_bin['recall_%'], linewidth=3.0)
plt.ylabel("flag_test_% | recall_%")
```

In [ ]:

```
df_bin[['bin_bottom', 'bin_top']].transpose()
```

In [229]:

```
df_bin.iloc[22].to_frame().transpose()
```

Out[229]:

	interval_num	bin_bottom	bin_top	flag_kredit_macet_count	flag_kredit_macet_sum	count_id_test_%	flag_test_
22	22.0	0.708	0.722	18.0	8.0	0.013373	0.4444

Based on eyeballing process on the graph, intersection between the highest recall_% curve and flag*test*% curve are between 0.708 and 0.722. Then, the threshold is 0.722. It means that id which get p0 prediction less than 0.722 will be considered as 1 or flagged as 'kredit_macet'. Otherwise, it will be considered as 0.

9. Prediction to Test Data

In [230]:

```
dtest = xgb.DMatrix(test[features])
probability_one = model.predict(dtest)

probability_zero = []
for i in probability_one:
    probability_zero.append(1-i)
```

In [231]:

```
df_final = pd.DataFrame()
df_final['test_id'] = test_id
df_final['p_zero'] = probability_zero
df_final['p_one'] = probability_one
df_final['prediction'] = df_final['p_zero'].apply(lambda x: 1 if x <= 0.722 else 0)
```

In [232]:

```
df_final.prediction.value_counts(normalize=True)
```

Out[232]:

```
0    0.823848
1    0.176152
Name: prediction, dtype: float64
```

In [233]:

```
prediction_result = df_final[['test_id', 'prediction', 'p_one']]
prediction_result.columns = ['X', 'prediction', 'probability']
```

In [234]:

```
prediction_result.head()
```

Out[234]:

	X	prediction	probability
0	15494	0	0.102171
1	15495	1	0.970043
2	15496	0	0.048079
3	15497	0	0.034377
4	15498	0	0.050254

In [236]:

```
prediction_result.to_csv('prediction.csv', index=False)
```