credit_risk_ensemble

September 24, 2019

[1]:
```
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
import numpy as np
import pandas as pd
from pathlib import Path
from collections import Counter
```

[3]:
```
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import confusion_matrix
from imblearn.metrics import classification_report_imbalanced
```

1 Read the CSV and Perform Basic Data Cleaning

[4]:
```
# https://help.lendingclub.com/hc/en-us/articles/
 ↪215488038-What-do-the-different-Note-statuses-mean-

columns = [
    "loan_amnt", "int_rate", "installment", "home_ownership",
    "annual_inc", "verification_status", "issue_d", "loan_status",
    "pymnt_plan", "dti", "delinq_2yrs", "inq_last_6mths",
    "open_acc", "pub_rec", "revol_bal", "total_acc",
    "initial_list_status", "out_prncp", "out_prncp_inv", "total_pymnt",
    "total_pymnt_inv", "total_rec_prncp", "total_rec_int", "total_rec_late_fee",
    "recoveries", "collection_recovery_fee", "last_pymnt_amnt", "next_pymnt_d",
    "collections_12_mths_ex_med", "policy_code", "application_type",␣
 ↪"acc_now_delinq",
    "tot_coll_amt", "tot_cur_bal", "open_acc_6m", "open_act_il",
    "open_il_12m", "open_il_24m", "mths_since_rcnt_il", "total_bal_il",
    "il_util", "open_rv_12m", "open_rv_24m", "max_bal_bc",
    "all_util", "total_rev_hi_lim", "inq_fi", "total_cu_tl",
    "inq_last_12m", "acc_open_past_24mths", "avg_cur_bal", "bc_open_to_buy",
    "bc_util", "chargeoff_within_12_mths", "delinq_amnt", "mo_sin_old_il_acct",
    "mo_sin_old_rev_tl_op", "mo_sin_rcnt_rev_tl_op", "mo_sin_rcnt_tl",␣
 ↪"mort_acc",
```

```
    "mths_since_recent_bc", "mths_since_recent_inq", "num_accts_ever_120_pd",␣
 ↪"num_actv_bc_tl",
    "num_actv_rev_tl", "num_bc_sats", "num_bc_tl", "num_il_tl",
    "num_op_rev_tl", "num_rev_accts", "num_rev_tl_bal_gt_0",
    "num_sats", "num_tl_120dpd_2m", "num_tl_30dpd", "num_tl_90g_dpd_24m",
    "num_tl_op_past_12m", "pct_tl_nvr_dlq", "percent_bc_gt_75",␣
 ↪"pub_rec_bankruptcies",
    "tax_liens", "tot_hi_cred_lim", "total_bal_ex_mort", "total_bc_limit",
    "total_il_high_credit_limit", "hardship_flag", "debt_settlement_flag"
]

target = ["loan_status"]
```

[5]:
```
# Load the data
file_path = Path('../Resources/LoanStats_2019Q1.csv.zip')
df = pd.read_csv(file_path, skiprows=1)[:-2]
df = df.loc[:, columns].copy()

# Drop the null columns where all values are null
df = df.dropna(axis='columns', how='all')

# Drop the null rows
df = df.dropna()

# Remove the `Issued` loan status
issued_mask = df['loan_status'] != 'Issued'
df = df.loc[issued_mask]

# convert interest rate to numerical
df['int_rate'] = df['int_rate'].str.replace('%', '')
df['int_rate'] = df['int_rate'].astype('float') / 100


# Convert the target column values to low_risk and high_risk based on their␣
 ↪values
x = {'Current': 'low_risk'}
df = df.replace(x)

x = dict.fromkeys(['Late (31-120 days)', 'Late (16-30 days)', 'Default', 'In␣
 ↪Grace Period'], 'high_risk')
df = df.replace(x)

df.reset_index(inplace=True, drop=True)

df.head()
```

[5]:
```
   loan_amnt  int_rate  installment home_ownership  annual_inc  \
0    10500.0    0.1719       375.35           RENT     66000.0
1    25000.0    0.2000       929.09       MORTGAGE    105000.0
2    20000.0    0.2000       529.88       MORTGAGE     56000.0
3    10000.0    0.1640       353.55           RENT     92000.0
4    22000.0    0.1474       520.39       MORTGAGE     52000.0

  verification_status   issue_d loan_status pymnt_plan    dti  ...   \
0     Source Verified  Mar-2019    low_risk          n  27.24  ...
1            Verified  Mar-2019    low_risk          n  20.23  ...
2            Verified  Mar-2019    low_risk          n  24.26  ...
3            Verified  Mar-2019    low_risk          n  31.44  ...
4        Not Verified  Mar-2019    low_risk          n  18.76  ...

   pct_tl_nvr_dlq  percent_bc_gt_75  pub_rec_bankruptcies  tax_liens  \
0            85.7             100.0                   0.0        0.0
1            91.2              50.0                   1.0        0.0
2            66.7              50.0                   0.0        0.0
3           100.0              50.0                   1.0        0.0
4           100.0               0.0                   0.0        0.0

   tot_hi_cred_lim  total_bal_ex_mort total_bc_limit  \
0          65687.0            38199.0         2000.0
1         271427.0            60641.0        41200.0
2          60644.0            45684.0         7500.0
3          99506.0            68784.0        19700.0
4         219750.0            25919.0        27600.0

   total_il_high_credit_limit  hardship_flag  debt_settlement_flag
0                     61987.0              N                     N
1                     49197.0              N                     N
2                     43144.0              N                     N
3                     76506.0              N                     N
4                     20000.0              N                     N

[5 rows x 86 columns]
```

2 Split the Data into Training and Testing

[6]:
```
# Create our features
X = # YOUR CODE HERE

# Create our target
y = # YOUR CODE HERE
```

[7]:
```
X.describe()
```

[7]:

```
          loan_amnt      int_rate   installment    annual_inc           dti  \
count  68817.000000  68817.000000  68817.000000  6.881700e+04  68817.000000
mean   16677.594562      0.127718    480.652863  8.821371e+04     21.778153
std    10277.348590      0.048130    288.062432  1.155800e+05     20.199244
min     1000.000000      0.060000     30.890000  4.000000e+01      0.000000
25%     9000.000000      0.088100    265.730000  5.000000e+04     13.890000
50%    15000.000000      0.118000    404.560000  7.300000e+04     19.760000
75%    24000.000000      0.155700    648.100000  1.040000e+05     26.660000
max    40000.000000      0.308400   1676.230000  8.797500e+06    999.000000

        delinq_2yrs  inq_last_6mths      open_acc       pub_rec  \
count  68817.000000    68817.000000  68817.000000  68817.000000
mean       0.217766        0.497697     12.587340      0.126030
std        0.718367        0.758122      6.022869      0.336797
min        0.000000        0.000000      2.000000      0.000000
25%        0.000000        0.000000      8.000000      0.000000
50%        0.000000        0.000000     11.000000      0.000000
75%        0.000000        1.000000     16.000000      0.000000
max       18.000000        5.000000     72.000000      4.000000

           revol_bal  ...  issue_d_Mar-2019  pymnt_plan_n  \
count   68817.000000  ...      68817.000000       68817.0
mean    17604.142828  ...          0.177238           1.0
std     21835.880400  ...          0.381873           0.0
min         0.000000  ...          0.000000           1.0
25%      6293.000000  ...          0.000000           1.0
50%     12068.000000  ...          0.000000           1.0
75%     21735.000000  ...          0.000000           1.0
max    587191.000000  ...          1.000000           1.0

       initial_list_status_f  initial_list_status_w  next_pymnt_d_Apr-2019  \
count           68817.000000           68817.000000           68817.000000
mean                0.123879               0.876121               0.383161
std                 0.329446               0.329446               0.486161
min                 0.000000               0.000000               0.000000
25%                 0.000000               1.000000               0.000000
50%                 0.000000               1.000000               0.000000
75%                 0.000000               1.000000               1.000000
max                 1.000000               1.000000               1.000000

       next_pymnt_d_May-2019  application_type_Individual  \
count           68817.000000                 68817.000000
mean                0.616839                     0.860340
std                 0.486161                     0.346637
min                 0.000000                     0.000000
25%                 0.000000                     1.000000
50%                 1.000000                     1.000000
```

```
75%                   1.000000                      1.000000
max                   1.000000                      1.000000

       application_type_Joint App  hardship_flag_N  debt_settlement_flag_N
count                  68817.000000          68817.0                 68817.0
mean                       0.139660              1.0                     1.0
std                        0.346637              0.0                     0.0
min                        0.000000              1.0                     1.0
25%                        0.000000              1.0                     1.0
50%                        0.000000              1.0                     1.0
75%                        0.000000              1.0                     1.0
max                        1.000000              1.0                     1.0

[8 rows x 95 columns]
```

[8]:
```
# Check the balance of our target values
y['loan_status'].value_counts()
```

[8]:
```
low_risk     68470
high_risk      347
Name: loan_status, dtype: int64
```

[9]:
```
# Split the X and y into X_train, X_test, y_train, y_test
# YOUR CODE HERE
```

3 Ensemble Learners

In this section, you will compare two ensemble algorithms to determine which algorithm results in the best performance. You will train a Balanced Random Forest Classifier and an Easy Ensemble AdaBoost classifier . For each algorithm, be sure to complete the folliowing steps:

1. Train the model using the training data.
2. Calculate the balanced accuracy score from sklearn.metrics.
3. Print the confusion matrix from sklearn.metrics.
4. Generate a classication report using the `imbalanced_classification_report` from imbalanced-learn.
5. For the Balanced Random Forest Classifier onely, print the feature importance sorted in descending order (most important feature to least important) along with the feature score

Note: Use a random state of 1 for each algorithm to ensure consistency between tests

3.0.1 Balanced Random Forest Classifier

[10]:
```
# Resample the training data with the RandomOversampler
# YOUR CODE HERE
```

[10]:
```
BalancedRandomForestClassifier(bootstrap=True, class_weight=None,
                               criterion='gini', max_depth=None, max_features='auto',
                               max_leaf_nodes=None, min_impurity_decrease=0.0,
```

```
                    min_samples_leaf=2, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                    oob_score=False, random_state=1, replacement=False,
                    sampling_strategy='auto', verbose=0, warm_start=False)
```

[11]:
```
# Calculated the balanced accuracy score
# YOUR CODE HERE
```

[11]:
```
0.7855052723466922
```

[12]:
```
# Display the confusion matrix
# YOUR CODE HERE
```

[12]:
```
array([[   68,    33],
       [ 1749, 15355]])
```

[13]:
```
# Print the imbalanced classification report
# YOUR CODE HERE
```

```
                   pre       rec       spe        f1       geo       iba
sup

  high_risk       0.04      0.67      0.90      0.07      0.78      0.59
101
   low_risk       1.00      0.90      0.67      0.95      0.78      0.62
17104

avg / total       0.99      0.90      0.67      0.94      0.78      0.62
17205
```

[14]:
```
# List the features sorted in descending order by feature importance
# YOUR CODE HERE
```

```
loan_amnt: (0.09175752102205247)
int_rate: (0.06410003199501778)
installment: (0.05764917485461809)
annual_inc: (0.05729679526683975)
dti: (0.05174788106507317)
delinq_2yrs: (0.031955619175665397)
inq_last_6mths: (0.02353678623968216)
open_acc: (0.017078915518993903)
pub_rec: (0.017014861224701222)
revol_bal: (0.016537957646730293)
total_acc: (0.016169718411077325)
out_prncp: (0.01607049983545137)
out_prncp_inv: (0.01599866290723441)
total_pymnt: (0.015775537221600675)
total_pymnt_inv: (0.01535560674178928)
```

```
total_rec_prncp: (0.015029265003541079)
total_rec_int: (0.014828006488636946)
total_rec_late_fee: (0.01464881608833323)
recoveries: (0.014402430445752665)
collection_recovery_fee: (0.014318832248876989)
last_pymnt_amnt: (0.013519867193755364)
collections_12_mths_ex_med: (0.013151520216882331)
policy_code: (0.013101578263049833)
acc_now_delinq: (0.012784600558682344)
tot_coll_amt: (0.012636608914961465)
tot_cur_bal: (0.012633464965390648)
open_acc_6m: (0.012406321468566728)
open_act_il: (0.011687404692448701)
open_il_12m: (0.01156494245653799)
open_il_24m: (0.011455878011762288)
mths_since_rcnt_il: (0.011409157520644688)
total_bal_il: (0.01073641504525053)
il_util: (0.010380085181706624)
open_rv_12m: (0.010097528131347774)
open_rv_24m: (0.00995373830638152)
max_bal_bc: (0.00991410213601043)
all_util: (0.009821715826953788)
total_rev_hi_lim: (0.009603648248133598)
inq_fi: (0.009537423049553)
total_cu_tl: (0.008976776055926955)
inq_last_12m: (0.008870623013604539)
acc_open_past_24mths: (0.008745106187024114)
avg_cur_bal: (0.008045578273709669)
bc_open_to_buy: (0.007906251501807723)
bc_util: (0.00782073260901301)
chargeoff_within_12_mths: (0.007798696767389274)
delinq_amnt: (0.007608045628523077)
mo_sin_old_il_acct: (0.0075861537897335815)
mo_sin_old_rev_tl_op: (0.007554511001273182)
mo_sin_rcnt_rev_tl_op: (0.007471884930172615)
mo_sin_rcnt_tl: (0.007273779915807858)
mort_acc: (0.006874845464745796)
mths_since_recent_bc: (0.006862142977394886)
mths_since_recent_inq: (0.006838718858820505)
num_accts_ever_120_pd: (0.006413554699909871)
num_actv_bc_tl: (0.006319439816216779)
num_actv_rev_tl: (0.006160469432535709)
num_bc_sats: (0.006066257227997291)
num_bc_tl: (0.005981472544437747)
num_il_tl: (0.0055301594524349495)
num_op_rev_tl: (0.004961823663836347)
num_rev_accts: (0.004685198497435334)
num_rev_tl_bal_gt_0: (0.0045872929977180356)
```

```
num_sats: (0.0041651633321967895)
num_tl_120dpd_2m: (0.004016461341161775)
num_tl_30dpd: (0.0032750717701661657)
num_tl_90g_dpd_24m: (0.0027565184136781346)
num_tl_op_past_12m: (0.0026174030074401656)
pct_tl_nvr_dlq: (0.002279671873697176)
percent_bc_gt_75: (0.0021899772867773103)
pub_rec_bankruptcies: (0.0020851101815353096)
tax_liens: (0.0018404849590376573)
tot_hi_cred_lim: (0.001736019018028134)
total_bal_ex_mort: (0.0015472230884974506)
total_bc_limit: (0.0012263315437383057)
total_il_high_credit_limit: (0.0012213148580230454)
home_ownership_ANY: (0.0012151288883862276)
home_ownership_MORTGAGE: (0.0008976722260399365)
home_ownership_OWN: (0.0008125182396705508)
home_ownership_RENT: (0.000573414997420326)
verification_status_Not Verified: (0.0005168345750594915)
verification_status_Source Verified: (0.0004192455022893127)
verification_status_Verified: (0.0)
issue_d_Feb-2019: (0.0)
issue_d_Jan-2019: (0.0)
issue_d_Mar-2019: (0.0)
pymnt_plan_n: (0.0)
initial_list_status_f: (0.0)
initial_list_status_w: (0.0)
next_pymnt_d_Apr-2019: (0.0)
next_pymnt_d_May-2019: (0.0)
application_type_Individual: (0.0)
application_type_Joint App: (0.0)
hardship_flag_N: (0.0)
debt_settlement_flag_N: (0.0)
```

3.0.2 Easy Ensemble AdaBoost Classifier

[15]:
```
# Train the Classifier
# YOUR CODE HERE
```

[15]:
```
EasyEnsembleClassifier(base_estimator=None, n_estimators=100, n_jobs=1,
                       random_state=1, replacement=False, sampling_strategy='auto',
                       verbose=0, warm_start=False)
```

[16]:
```
# Calculated the balanced accuracy score
# YOUR CODE HERE
```

[16]:
```
0.9316600714093861
```

[17]:
```
# Display the confusion matrix
# YOUR CODE HERE
```

[17]:
```
array([[   93,     8],
       [  983, 16121]])
```

[18]:
```
# Print the imbalanced classification report
# YOUR CODE HERE
```

```
                   pre       rec       spe        f1       geo       iba
sup

  high_risk       0.09      0.92      0.94      0.16      0.93      0.87
101
   low_risk       1.00      0.94      0.92      0.97      0.93      0.87
17104

avg / total       0.99      0.94      0.92      0.97      0.93      0.87
17205
```

[ ]:
```
```