Problem 1

September 11, 2018

0.0.1 Author: Yasir Abdurrahman

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
df_train = pd.read_csv('dataset1/train1.csv')
```

0.1 Dataset

0.1.1 Description

Dataset berikut merupakan data demografi 1556 nasabah bank pada kuarter ke-4 tahun 2017, data ini kemudian dibagi menjadi data train dan test. Variabel target pada data ini adalah `not_paid`, variabel biner yang menjadi indikasi suatu loan/ pinjaman lunas (berhasil dilunasi pembayarannya) atau tidak. Suatu pinjaman dikatakan not_paid (not paid = 1) jika terjadi default (gagal bayar), Charged Off, atau lewat batas akhir pembayaran (Grace Period).

0.1.2 Variables Glossary

- initial_list_status: indikasi loan termasuk ke dalam kategori w (whole) atau f (fractional).
- purpose: tujuan peminjaman (loan) terbagi atas 5 kategori yaitu untuk credit_card, debt_consolidation, home_improvement, major_purchase, dan small_bussiness
- int_rate: interest rate(suku bunga) dalam prosentase
- installment: banyaknya installment/uang bulanan yang dibayarkan peminjam
- annual_inc: income/pemasukan tahunan peminjam sesuai yang tertulis saat proses pengajuan pinjaman
- dti: rasio antara pinjaman bulanan yang wajib dibayarkan peminjam dengan gaji/pemasukan peminjam sesuai report
- verification_status: status verifikasi report pemasukan/gaji peminjam, terbagi atas kategori income_verified, not verified, atau source was verified
- grade: grade load berdasarkan software
- revol_bal: total kredit dalam revolving balance (pinjaman yang tidak terbayarkan)
- inq_last_12m: banyaknya kredit/pinjaman pada akhir bulan 12
- delinq_2yrs: banyaknya hari telat bayar untuk kriteria 30+ pada history peminjam selama 2 tahun terakhir
- home_ownership: kategori kepemilikan rumah peminjam meliputi MORTGAGE, OWN, atau RENT
- log_inc: log dari annual_inc

- verified: 0 untuk not_verified masih dibawah status verifikasi, 1 lainnya
- grdCtoA: 1 untuk grade kredit A, B atau C; 0 untuk grade load lainnya
- not_paid: 1 jika gagal bayar *charge off/grace period*, 0 lainnya (TARGET)

```
In [3]: df_train.head()
```

```
Out[3]:    Unnamed: 0 initial_list_status             purpose  int_rate  installment  \
        0        1495                   w  debt_consolidation     21.45       955.75
        1         266                   w  debt_consolidation     18.06       289.47
        2         309                   w    home_improvement      9.44       838.91
        3         239                   w    home_improvement     10.42       214.55
        4         136                   f  debt_consolidation     11.99      1024.52

           annual_inc    dti verification_status grade  revol_bal  inq_last_12m  \
        0     90000.0  20.91            Verified     D      23448             4
        1     65000.0  12.74     Source Verified     D      13362             2
        2     97400.0  12.64     Source Verified     B       2372             1
        3     60000.0   2.38        Not Verified     B       4705             2
        4    150000.0  20.84        Not Verified     B      14342             0

           delinq_2yrs home_ownership  not_paid    log_inc  verified  grdCtoA
        0            0       MORTGAGE         1  11.407565         1        0
        1            0       MORTGAGE         1  11.082143         1        0
        2            0       MORTGAGE         1  11.486581         1        1
        3            0            OWN         0  11.002100         0        1
        4            0       MORTGAGE         0  11.918391         0        1
```

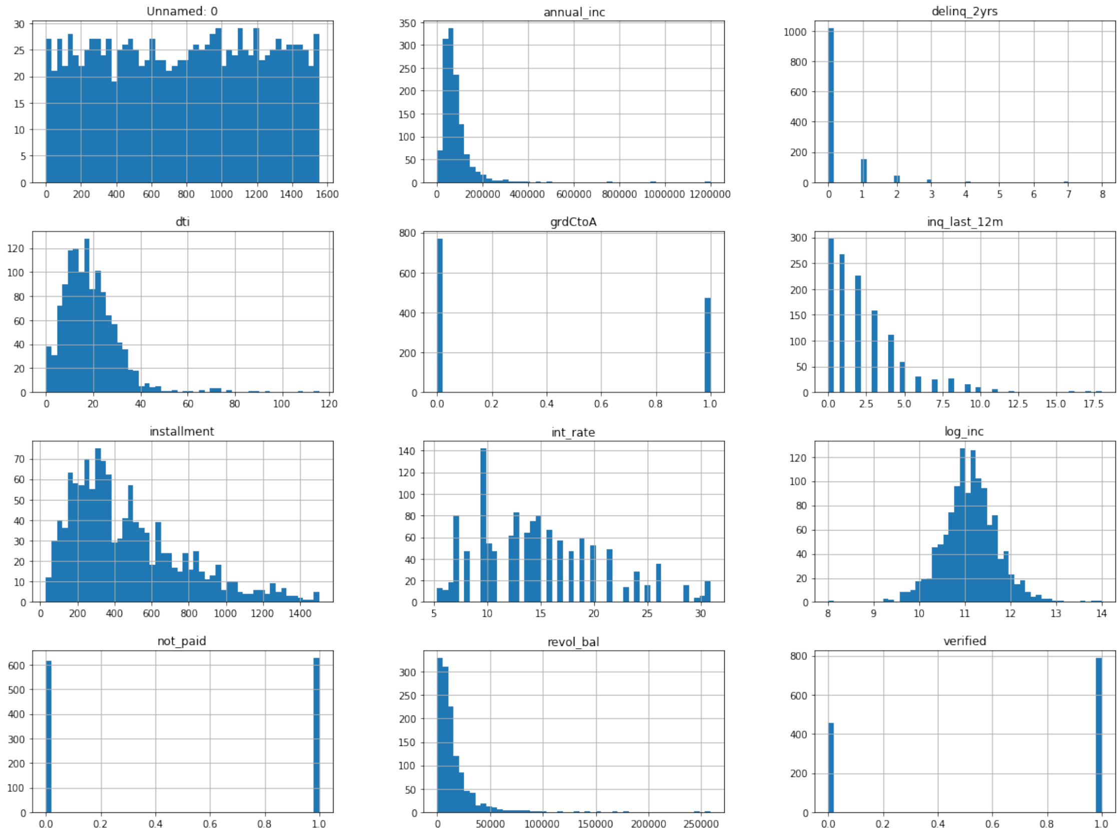
1 Explore Dataset

```
In [4]: df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1244 entries, 0 to 1243
Data columns (total 17 columns):
Unnamed: 0             1244 non-null int64
initial_list_status    1244 non-null object
purpose                1244 non-null object
int_rate               1244 non-null float64
installment            1244 non-null float64
annual_inc             1244 non-null float64
dti                    1244 non-null float64
verification_status    1244 non-null object
grade                  1244 non-null object
revol_bal              1244 non-null int64
inq_last_12m           1244 non-null int64
delinq_2yrs            1244 non-null int64
home_ownership         1244 non-null object
not_paid               1244 non-null int64
```

```
log_inc                1244 non-null float64
verified               1244 non-null int64
grdCtoA                1244 non-null int64
dtypes: float64(5), int64(7), object(5)
memory usage: 165.3+ KB
```

In [5]:
```
%matplotlib inline
df_train.hist(bins=50, figsize=(20,15))
plt.show()
```



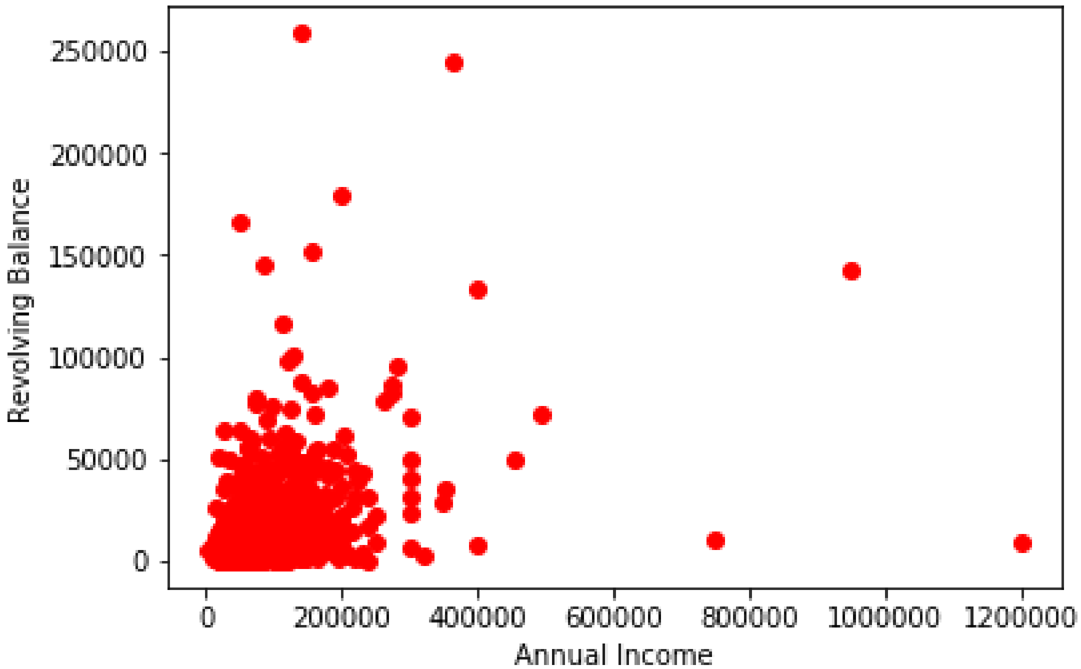
1.1 Question 1:

Bagaimana anda mendeskripsikan hubungan antara `annual_inc` dan `revol_bal`?

Answer: Tidak ada hubungan linier

In [6]:
```
# plot 'annual_inc' dan 'revol_bal'
plt.plot(df_train['annual_inc'], df_train['revol_bal'], 'ro')
plt.xlabel('Annual Income')
plt.ylabel('Revolving Balance')
```

Out[6]: `Text(0,0.5,'Revolving Balance')`



1.2 Question 2:

Berdasarkan kategori purpose (tujuan pinjaman) yang paling banyak ditemukan nasabah mengalami gagal bayar(not_paid=1), berapa banyak nasabah yang mengalami gagal bayar pada kategori tersebut?

Answer: debt_consolidation -> 420

```
In [7]: purpose_not_paid = []

        for index, row in df_train.iterrows():
            if row['not_paid'] == 1:
                purpose_not_paid.append(row['purpose'])
```

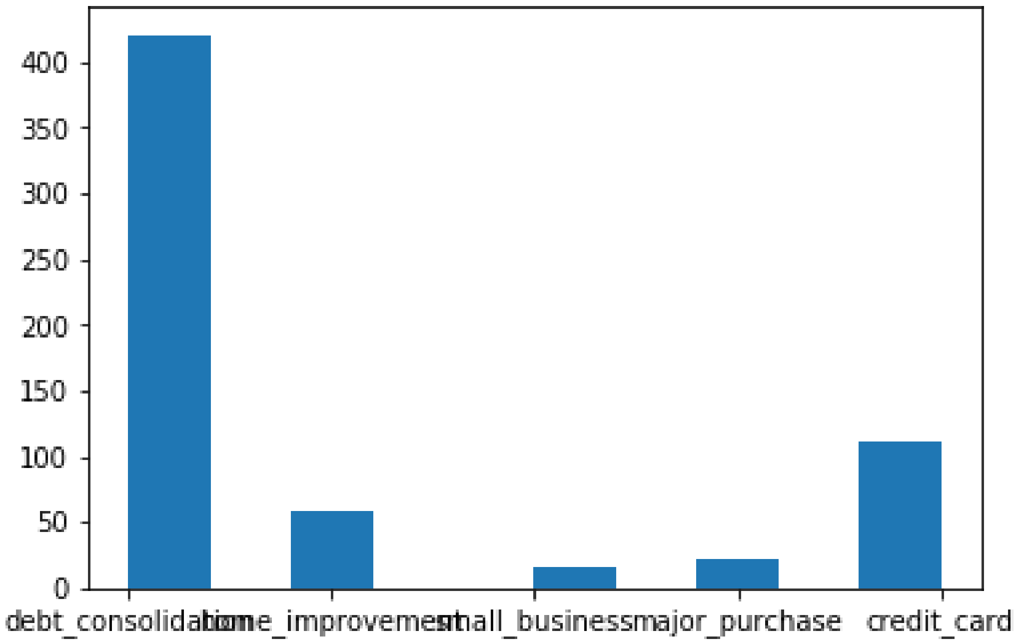
```
In [8]: from collections import Counter

        Counter(purpose_not_paid)
```

```
Out[8]: Counter({'debt_consolidation': 420,
                  'home_improvement': 58,
                  'small_business': 15,
                  'major_purchase': 22,
                  'credit_card': 112})
```

```
In [9]: plt.hist(purpose_not_paid)
```

```
Out[9]: (array([420.,   0.,  58.,   0.,   0.,  15.,   0.,  22.,   0., 112.]),
         array([0. , 0.4, 0.8, 1.2, 1.6, 2. , 2.4, 2.8, 3.2, 3.6, 4. ]),
         <a list of 10 Patch objects>)
```

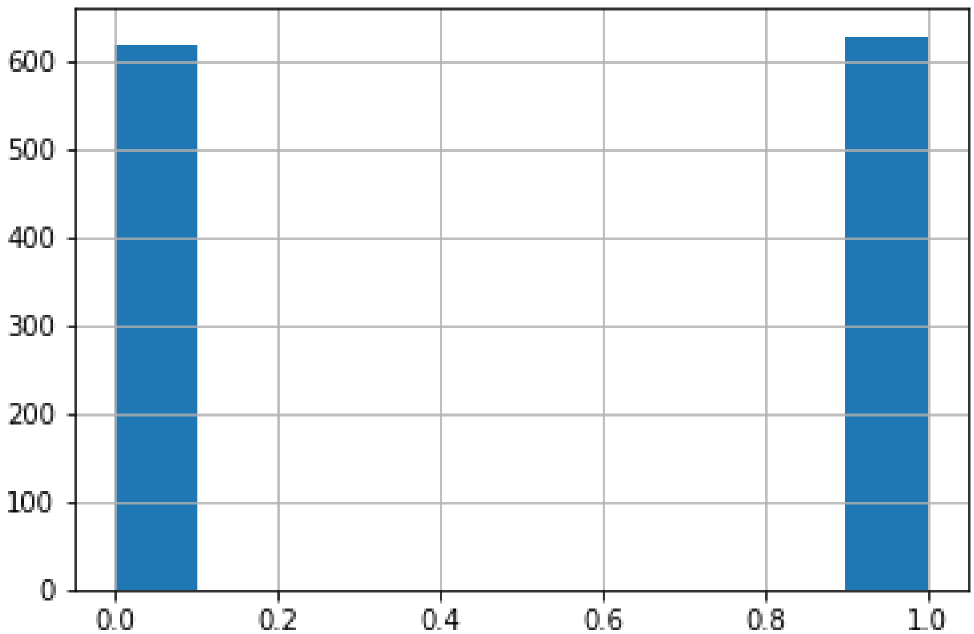


1.3 Question 3:

Apakah data loan (dataset 1) tersebut dikategorikan sebagai data yang akan mengalami masalah 'imbalanced class'?
Answer: Tidak

```
In [10]: df_train['not_paid'].hist()
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x22ccd141f28>
```



Menggunakan train dataset 1, buat sebuah model regresi logistik untuk memprediksi peluang default (not_paid) nasabah dengan variabel `purpose`, `int_rate`, `installment`, `annual_inc`, `verified`, `home_ownership` dan `grdCtoA` sebagai variabel prediktornya.

1.4 Question 4:

Variabel manakah yang memiliki korelasi negatif terhadap kenaikan odds default (not_paid=1)?

Answer:

```
In [11]: used_variables = ['purpose', 'int_rate', 'installment', 'annual_inc', 'verified', 'hor

         nasabah = df_train[used_variables]
```

```
In [12]: corr_matrix = nasabah.corr()
         corr_matrix['not_paid'].sort_values(ascending=False)
```

```
Out[12]: not_paid       1.000000
         int_rate       0.160727
         installment    0.148875
         verified       0.101174
         annual_inc    -0.058832
         grdCtoA       -0.153679
         Name: not_paid, dtype: float64
```

2 Preprocessing

```
In [13]: nasabah.head()
```

```
Out[13]:                purpose  int_rate  installment  annual_inc  verified  \
         0  debt_consolidation     21.45       955.75     90000.0         1
         1  debt_consolidation     18.06       289.47     65000.0         1
         2    home_improvement      9.44       838.91     97400.0         1
         3    home_improvement     10.42       214.55     60000.0         0
         4  debt_consolidation     11.99      1024.52    150000.0         0

           home_ownership  grdCtoA  not_paid
         0       MORTGAGE        0         1
         1       MORTGAGE        0         1
         2       MORTGAGE        1         1
         3            OWN        1         0
         4       MORTGAGE        1         0
```

```
In [14]: # check missing values
         nasabah.isnull().sum()
```

```
Out[14]: purpose           0
         int_rate          0
         installment       0
         annual_inc        0
         verified          0
         home_ownership    0
         grdCtoA           0
         not_paid          0
         dtype: int64
```

```
In [15]: nasabah_num = nasabah.drop(['purpose', 'home_ownership'], axis=1)
         nasabah_num.head()
```

```
Out[15]:    int_rate  installment  annual_inc  verified  grdCtoA  not_paid
         0     21.45       955.75     90000.0         1        0         1
         1     18.06       289.47     65000.0         1        0         1
         2      9.44       838.91     97400.0         1        1         1
         3     10.42       214.55     60000.0         0        1         0
         4     11.99      1024.52    150000.0         0        1         0
```

```
In [16]: nasabah_cat = nasabah[['purpose','home_ownership']]
         nasabah_cat.head()
```

```
Out[16]:                purpose home_ownership
         0  debt_consolidation       MORTGAGE
         1  debt_consolidation       MORTGAGE
         2    home_improvement       MORTGAGE
         3    home_improvement            OWN
         4  debt_consolidation       MORTGAGE
```

```
In [17]: nasabah_cat['purpose'].value_counts()
```

Out[17]: debt_consolidation   808
         credit_card   247
         home_improvement   126
         major_purchase   40
         small_business   23
         Name: purpose, dtype: int64

In [18]: nasabah_cat['home_ownership'].value_counts()

Out[18]: MORTGAGE   633
         RENT   458
         OWN   153
         Name: home_ownership, dtype: int64

2.1 Change categorical using One Hot Encoding

```
In [19]: from future_encoders import OneHotEncoder
```

```
In [20]: purpose_cat = nasabah_cat[['purpose']]
         purpose_cat.head()
```

```
Out[20]:               purpose
         0  debt_consolidation
         1  debt_consolidation
         2    home_improvement
         3    home_improvement
         4  debt_consolidation
```

```
In [21]: encoder = OneHotEncoder(sparse=False)
         purpose_cat_1hot = encoder.fit_transform(purpose_cat)
         purpose_cat_1hot
```

```
Out[21]: array([[0., 1., 0., 0., 0.],
                [0., 1., 0., 0., 0.],
                [0., 0., 1., 0., 0.],
                ...,
                [1., 0., 0., 0., 0.],
                [0., 1., 0., 0., 0.],
                [0., 1., 0., 0., 0.]])
```

```
In [22]: encoder.categories_
```

```
Out[22]: [array(['credit_card', 'debt_consolidation', 'home_improvement',
                 'major_purchase', 'small_business'], dtype=object)]
```

2.2 Pipeline

```
In [23]: from sklearn.pipeline import Pipeline
         from sklearn.preprocessing import StandardScaler
         from sklearn.preprocessing import Imputer
         from future_encoders import OneHotEncoder, ColumnTransformer
```

In [24]:
```
nasabah_num.head()
```

Out[24]:

	int_rate	installment	annual_inc	verified	grdCtoA	not_paid
0	21.45	955.75	90000.0	1	0	1
1	18.06	289.47	65000.0	1	0	1
2	9.44	838.91	97400.0	1	1	1
3	10.42	214.55	60000.0	0	1	0
4	11.99	1024.52	150000.0	0	1	0

In [25]:
```
num_attribs = ['int_rate', 'installment', 'annual_inc']
bool_attribs = ['verified', 'grdCtoA']
cat_attribs = ['purpose', 'home_ownership']

full_pipeline = ColumnTransformer([
    ("num", StandardScaler(), num_attribs),
    ("bool", Imputer(strategy="median"), bool_attribs),
    ("cat", OneHotEncoder(sparse=False), cat_attribs)
])

nasabah_prepared = full_pipeline.fit_transform(nasabah)
```

In [26]:
```
nasabah_prepared.shape
```

Out[26]:
```
(1244, 13)
```

In [27]:
```
X_train = nasabah_prepared
y_train = nasabah['not_paid']
```

3 Logistic Regression Model

In [28]:
```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression(random_state=42)
log_reg.fit(X_train, y_train)
```

Out[28]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=42, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

4 Fine-tune Model

In [29]:
```
from sklearn.model_selection import GridSearchCV
```

In [51]:
```
# regularization penalty space
penalty = ['l1', 'l2']

# regularization hyperparameter space
```

```
C = np.logspace(0, 1, 10)

# create hyperparameter options
hyperparameters = dict(C=C, penalty=penalty)
```

In [52]:
```
grid_search = GridSearchCV(log_reg, hyperparameters, cv=5, scoring='neg_mean_squared_e

grid_search.fit(X_train, y_train)
```

Out[52]:
```
GridSearchCV(cv=5, error_score='raise',
       estimator=LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercep
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=42, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid={'C': array([ 1.      ,  1.29155,  1.6681 ,  2.15443,  2.78256,  3.59
        4.64159,  5.99484,  7.74264, 10.      ]), 'penalty': ['l1', 'l2']},
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='neg_mean_squared_error', verbose=0)
```

In [53]:
```
grid_search.best_params_
```

Out[53]:
```
{'C': 1.0, 'penalty': 'l1'}
```

In [54]:
```
grid_search.best_estimator_
```

Out[54]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l1', random_state=42, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [55]:
```
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
0.6452896356105269 {'C': 1.0, 'penalty': 'l1'}
0.6502534751876141 {'C': 1.0, 'penalty': 'l2'}
0.6477763100870298 {'C': 1.2915496650148839, 'penalty': 'l1'}
0.6508712933556599 {'C': 1.2915496650148839, 'penalty': 'l2'}
0.6490160744941288 {'C': 1.6681005372000588, 'penalty': 'l1'}
0.6508712933556599 {'C': 1.6681005372000588, 'penalty': 'l2'}
0.6490160744941288 {'C': 2.154434690031884, 'penalty': 'l1'}
0.6502534751876141 {'C': 2.154434690031884, 'penalty': 'l2'}
0.6490160744941288 {'C': 2.7825594022071245, 'penalty': 'l1'}
0.6502534751876141 {'C': 2.7825594022071245, 'penalty': 'l2'}
0.6502534751876141 {'C': 3.5938136638046276, 'penalty': 'l1'}
0.6502534751876141 {'C': 3.5938136638046276, 'penalty': 'l2'}
0.6508712933556599 {'C': 4.641588833612778, 'penalty': 'l1'}
0.6502534751876141 {'C': 4.641588833612778, 'penalty': 'l2'}
```

```
0.6502534751876141 {'C': 5.994842503189409, 'penalty': 'l1'}
0.6502534751876141 {'C': 5.994842503189409, 'penalty': 'l2'}
0.6502534751876141 {'C': 7.742636826811269, 'penalty': 'l1'}
0.6502534751876141 {'C': 7.742636826811269, 'penalty': 'l2'}
0.6502534751876141 {'C': 10.0, 'penalty': 'l1'}
0.6502534751876141 {'C': 10.0, 'penalty': 'l2'}
```

5 Test the model

```
In [35]: df_test = pd.read_csv('dataset1/test1.csv')
```

```
In [36]: df_test.head()
```

```
Out[36]:    Unnamed: 0 initial_list_status             purpose  int_rate  installment  \
        0           6                   w  debt_consolidation     10.91       130.79
        1           8                   w         credit_card     10.91       915.50
        2           9                   w    home_improvement     17.09       713.96
        3          10                   w  debt_consolidation     18.06       408.73
        4          26                   w  debt_consolidation     18.06       578.93

           annual_inc    dti verification_status grade  revol_bal  inq_last_12m  \
        0     49000.0   5.12        Not Verified     B       2016             5
        1     95000.0  33.11        Not Verified     B      27588             1
        2    150000.0  14.26     Source Verified     D      27024             8
        3     85000.0  17.66            Verified     D      11719             1
        4     40000.0  25.32     Source Verified     D      15264             2

           delinq_2yrs home_ownership  not_paid    log_inc  verified  grdCtoA
        0            0       MORTGAGE         1  10.799576         0        1
        1            0           RENT         1  11.461632         0        1
        2            0       MORTGAGE         1  11.918391         1        0
        3            0           RENT         0  11.350407         1        0
        4            0           RENT         1  10.596635         1        0
```

```
In [37]: df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 312 entries, 0 to 311
Data columns (total 17 columns):
Unnamed: 0             312 non-null int64
initial_list_status    312 non-null object
purpose                312 non-null object
int_rate               312 non-null float64
installment            312 non-null float64
annual_inc             312 non-null float64
dti                    312 non-null float64
verification_status    312 non-null object
```

```
grade                    312 non-null object
revol_bal                312 non-null int64
inq_last_12m             312 non-null int64
delinq_2yrs              312 non-null int64
home_ownership           312 non-null object
not_paid                 312 non-null int64
log_inc                  312 non-null float64
verified                 312 non-null int64
grdCtoA                  312 non-null int64
dtypes: float64(5), int64(7), object(5)
memory usage: 41.5+ KB
```

```
In [38]: nasabah_test = df_test[used_variables]
         nasabah_test.head()
```

```
Out[38]:               purpose  int_rate  installment  annual_inc  verified  \
         0  debt_consolidation     10.91       130.79     49000.0         0
         1         credit_card     10.91       915.50     95000.0         0
         2    home_improvement     17.09       713.96    150000.0         1
         3  debt_consolidation     18.06       408.73     85000.0         1
         4  debt_consolidation     18.06       578.93     40000.0         1

           home_ownership  grdCtoA  not_paid
         0       MORTGAGE        1         1
         1           RENT        1         1
         2       MORTGAGE        0         1
         3           RENT        0         0
         4           RENT        0         1
```

```
In [39]: X_test = full_pipeline.fit_transform(nasabah_test)
```

```
In [40]: X_test.shape
```

```
Out[40]: (312, 13)
```

6 Accuracy Score

```
In [41]: from sklearn.metrics import accuracy_score
```

```
In [42]: y_true = nasabah_test['not_paid']
```

```
In [43]: final_model = grid_search.best_estimator_
         predict = final_model.predict(X_test)
```

```
In [44]: accuracy_score(y_true, predict)
```

```
Out[44]: 0.5993589743589743
```

```
In [45]: from sklearn.metrics import confusion_matrix

         tn, fp, fn, tp = confusion_matrix(y_true, predict).ravel()

         print(tn, fp, fn, tp)
```

```
91 70 55 96
```

6.1 Question 5:

Jika pada model regresi logistik diperoleh koefisien dari variabel gradeCtoA adalah -0.3298. Dengan mengasumsikan variabel lain konstan, berapa odds default (not_paid=1) untuk nasabah yang memiliki grad A-C (gradeCtoA=1) dibandingkan dengan nasabah yang memiliki grade loan lain? bulatkan hasil anda hingga 3 angka dibelakang koma (contoh : 4.323 atau 16.423)

Answer:

```
In [46]: pass
```

6.2 Question 6

Diberikan cross tabulasi hasil perbandingan nilai aktual dan prediksi menggunakan model regresi logistik berikut :

		actual	
		0	1
predicted	0	93	54
	1	68	97

Berapa nilai recall/sensitivity berdasarkan confusion matrix diatas?

Answer: 97/(97+54) = 0.6423841059602649

```
In [56]: recall = tp/(tp+fn)
         recall
```

```
Out[56]: 0.6357615894039735
```