PERBANDINGAN ALGORITMA RANDOM FOREST, NAÏVE BAYES & SUPPORT VECTOR MACHINE UNTUK MEMPREDIKSI KREDIT MACET

Abdurrahman Malik Karim
1314618011
Prodi Statistika, FMIPA UNJ, Jakarta

Abstrak

Pemberian kredit(pinjaman) dari bank ke nasabah merupakan kegiatan yang memiliki resiko tinggi bagi si pemberi kredit. Karena banyak nasabah yang sering membayar pinjamannya telat atau tidak tepat waktu (kredit macet). Hal ini bisa terjadi akibat analisis yang kurang cermat dari pihak bank. Untuk mencegah kredit macet, dibutuhkan suatu metode untuk memprediksi kreditnya secara akurat. Dalam hal ini, penulis menggunakan metode data mining.

Metode data mining yang penulis gunakan adalah klasifikasi. Klasifikasi sendiri memiliki banyak algoritma, tetapi untuk saat ini penulis hanya akan membandingkan tiga algoritma yaitu algoritma random forest, naïve bayes & support vector machine. Ketiga algoritma ini dikenal dengan model yang dihasilkannya memiliki akurasi yang tinggi.

Dari hasil pemilihan fitur dengan Random Forest terdapat 7 fitur yang kurang berpengaruh terhadap kredit dinyatakan lancar atau tidak. Sehingga 7 fitur tersebut tidak diikutkan dalam proses pembuatan model. Dan dari ketiga algoritma diatas, Support Vector Machine dengan menggunakan kernel Gaussian mendapat akurasi tertinggi sebesar 82,04% terhadap data training dan terhadap data testing akurasinya sebesar 81,37%.

Kata kunci*----kartu kredit, klasifikasi, random forest, naïve bayes, support vector machine*

1. Pendahuluan

Pemberian kredit(pinjaman) dari bank ke nasabah merupakan kegiatan yang meiliki resiko tinggi bagi si pemberi kredit. Karena banyak nasabah yang sering membayar pinjamannya telat atau tidak tepat waktu (kredit macet). Hal ini bisa terjadi akibat analisis yang kurang cermat dari pihak bank. Untuk mencegah kredit macet, dibutuhkan suatu metode untuk memprediksi kreditnya secara akurat. Dalam hal ini, penulis menggunakan metode *data mining*.

Data mining adalah praktik pencarian secara otomatis di dalam data yang besar untuk menemukan pola dan tren yang melampaui analisis sederhana (*What Is Data Mining?*, n.d.). *Data mining* menggunakan algoritma matematika yang canggih untuk mensegmentasi data dan mengevaluasi kemungkinan kejadian di masa depan. *Data mining* juga dikenal sebagai *Knowledge Discovery in Data* (KDD).

Salah satu metode dari *data mining* yang pada kesempatan kali ini penulis gunakan adalah klasifikasi. Klasifikasi adalah proses memprediksi kelas dari titik data yang diberikan (Asiri, 2018). Contoh yang mudah dipahami adalah mengklasifikasikan email sebagai "spam" atau "bukan spam". Pada kesempatan kali ini, penulis akan membandingkan tiga algoritma klasifikasi yaitu *Random Forest, Naïve Bayes* dan *Support Vector Machine*. Kenapa tiga

algoritma ini yang digunakan? Karena ketiga algoritma diatas merupakan algoritma yang dikenal dengan model yang dihasilkannya memiliki akurasi yang tinggi.

2. Metode Penelitian

Untuk memprediksi kredit dengan menggunakan metode klasifikasi, penulis menggunakan 30000 data nasabah kredit negara Taiwan dengan jangka waktu dari bulan April 2005 sampai September 2005. Data diambil dari *https://www.kaggle.com/uciml/default-of-credit-card-clients-dataset.*

Berikut penjelasan variabel-variabel dari dataset yang digunakan :

ID	ID dari setiap klien
LIMIT_BAL	Jumlah kredit yang diberikan dalam dollar NT
SEX	Jenis kelamin (1 = laki-laki, 2 = perempuan)
EDUCATION	Pendidikan (1 = sekolah pascasarjana, 2 = universitas, 3 = SMA, 4 = lainnya, 5 = tidak diketahui, 6 = tidak diketahui)
MARRIAGE	Status perkawinan (1 = menikah, 2 = lajang, 3 = lainnya)
AGE	Umur dalam tahun
PAY_0	Status pelunasan pada bulan September 2005 (-1 = bayar tepat waktu, 1 = penundaan pembayaran untuk satu bulan, 2 = penundaan pembayaran selama dua bulan, ... ,8 = penundaan pembayaran selama delapan bulan, 9 = penundaan pembayaran selama Sembilan bulan dan lebih)
PAY_2	Status pelunasan pada bulan Agustus 2005 (skala sama seperti diatas)
PAY_3	Status pelunasan pada bulan Juli 2005 (skala sama seperti diatas)
PAY_4	Status pelunasan pada bulan Juni 2005 (skala sama seperti diatas)
PAY_5	Status pelunasan pada bulan Mei 2005 (skala sama seperti diatas)
PAY_6	Status pelunasan pada bulan April 2005 (skala sama seperti diatas)
BILL_AMT1	Jumlah tagihan pada bulan September 2005 (Dollar NT)
BILL_AMT2	Jumlah tagihan pada bulan Agustus 2005 (Dollar NT)
BILL_AMT3	Jumlah tagihan pada bulan Juli 2005 (Dollar NT)
BILL_AMT4	Jumlah tagihan pada bulan Juni 2005 (Dollar NT)
BILL_AMT5	Jumlah tagihan pada bulan Mei 2005 (Dollar NT)
BILL_AMT6	Jumlah tagihan pada bulan April 2005 (Dollar NT)
PAY_AMT1	Jumlah pembayaran sebelumnya pada bulan September 2005 (Dollar NT)
PAY_AMT2	Jumlah pembayaran sebelumnya pada bulan Agustus 2005 (Dollar NT)
PAY_AMT3	Jumlah pembayaran sebelumnya pada bulan Juli 2005 (Dollar NT)

PAY_AMT4	Jumlah pembayaran sebelumnya pada bulan Juni 2005 (Dollar NT)
PAY_AMT5	Jumlah pembayaran sebelumnya pada bulan Mei 2005 (Dollar NT)
PAY_AMT6	Jumlah pembayaran sebelumnya pada bulan April 2005 (Dollar NT)
default.payment.next.month	Pembayaran kredit akan macet (1 = iya, 0 = tidak)

Bagaimana cara algoritma-algoritma diatas bekerja?

Random Forest	Naïve Bayes	Support Vector Machine
1. Memilih sampel acak dari dataset yang telah diberikan	1. Menghitung probabilitas sebelumnya untuk label kelas yang diberikan	1. Menghasilkan *hyperplanes* yang memisahkan kelas dengan cara terbaik
2. Membuat *decision tree* untuk setiap sampel dan mendapatkan prediksi hasil dari setiap *decision tree*	2. Mencari probabilitas *Likelihood* dengan setiap atribut untuk setiap kelas	2. Memilih *hyperplane* yang tepat dengan pemisahan maksimum dari data terdekat
3. Melakukan *vote* untuk setiap hasil prediksi	3. Memasukkan nilai-nilai ini ke rumus *Bayes* dan menghitung peluang *posterior*	
4. Memilih hasil prediksi yang memiliki suara terbanyak sebagai prediksi akhir	4. Lihat kelas mana yang memiliki peluang lebih tinggi	

2.1 Persiapan Data

Pertama penulis akan mengubah nama variabel ‘PAY_0’ dan ‘default.payment.next.month’ menjadi ‘PAY_1’ dan ‘def_pay’ agar lebih mudah dalam proses pengerjaan.

```
# rename columns
dataset = dataset.rename(columns={'default.payment.next.month': 'def_pay',
                                  'PAY_0': 'PAY_1'})
dataset.head()
```

Lalu ketika melihat statistika deskriptif dari dataset tersebut terdapat data yang ‘aneh’.

	SEX	EDUCATION	MARRIAGE
count	30000.000000	30000.000000	30000.000000
mean	1.603733	1.853133	1.551867
std	0.489129	0.790349	0.521970
min	1.000000	0.000000	0.000000
25%	1.000000	1.000000	1.000000
50%	2.000000	2.000000	2.000000
75%	2.000000	2.000000	2.000000
max	2.000000	6.000000	3.000000

Disitu terlihat variabel ‘EDUCATION’ memiliki nilai 0, 5 dan 6 yang tidak diketahui artinya apa dan variabel ‘MARRIAGE’ memiliki nilai 0 yang juga tidak diketahui artinya apa. Oleh karena itu penulis memasukkan nilai-nilai diatas ke nilai yang memiliki arti ‘lainnya’ yaitu nilai 4 dan 3.

```
fil = (dataset.EDUCATION == 5) | (dataset.EDUCATION == 6) | (dataset.EDUCATION == 0)
dataset.loc[fil, 'EDUCATION'] = 4
print(dataset.EDUCATION.value_counts())
```

```
2    14030
1    10585
3     4917
4      468
Name: EDUCATION, dtype: int64
```

```
dataset.loc[dataset.MARRIAGE == 0, 'MARRIAGE'] = 3
print(dataset.MARRIAGE.value_counts())
```

```
2    15964
1    13659
3      377
Name: MARRIAGE, dtype: int64
```

Pada variabel ‘PAY_*’ pun juga terdapat keanehan.

	PAY_1	PAY_2	...	PAY_5	PAY_6
count	30000.000000	30000.000000	...	30000.000000	30000.000000
mean	-0.016700	-0.133767	...	-0.266200	-0.291100
std	1.123802	1.197186	...	1.133187	1.149988
min	-2.000000	-2.000000	...	-2.000000	-2.000000
25%	-1.000000	-1.000000	...	-1.000000	-1.000000
50%	0.000000	0.000000	...	0.000000	0.000000
75%	0.000000	0.000000	...	0.000000	0.000000
max	8.000000	8.000000	...	8.000000	8.000000

[8 rows x 6 columns]

Disitu terlihat ada nilai -2. Jika 1, 2, 3, dst menujukkan berapa bulan dia menunda pembayaran, maka nilai 0 menujukkan dia bayar tepat waktu dan nilai negatif diubah nilainya menjadi 0 juga.

```
#Change value to zero
fil = (dataset.PAY_1 == -2) | (dataset.PAY_1 == -1) | (dataset.PAY_1 == 0)
dataset.loc[fil, 'PAY_1'] = 0
fil = (dataset.PAY_2 == -2) | (dataset.PAY_2 == -1) | (dataset.PAY_2 == 0)
dataset.loc[fil, 'PAY_2'] = 0
fil = (dataset.PAY_3 == -2) | (dataset.PAY_3 == -1) | (dataset.PAY_3 == 0)
dataset.loc[fil, 'PAY_3'] = 0
fil = (dataset.PAY_4 == -2) | (dataset.PAY_4 == -1) | (dataset.PAY_4 == 0)
dataset.loc[fil, 'PAY_4'] = 0
fil = (dataset.PAY_5 == -2) | (dataset.PAY_5 == -1) | (dataset.PAY_5 == 0)
dataset.loc[fil, 'PAY_5'] = 0
fil = (dataset.PAY_6 == -2) | (dataset.PAY_6 == -1) | (dataset.PAY_6 == 0)
dataset.loc[fil, 'PAY_6'] = 0
```

```
              PAY_1         PAY_2  ...         PAY_5         PAY_6
count  30000.000000  30000.000000  ...  30000.00000  30000.000000
mean       0.356767      0.320033  ...      0.22150      0.226567
std        0.760594      0.801727  ...      0.71772      0.715438
min        0.000000      0.000000  ...      0.00000      0.000000
25%        0.000000      0.000000  ...      0.00000      0.000000
50%        0.000000      0.000000  ...      0.00000      0.000000
75%        0.000000      0.000000  ...      0.00000      0.000000
max        8.000000      8.000000  ...      8.00000      8.000000

[8 rows x 6 columns]
```

Lalu karena variabel ‘ID’ tidak akan digunakan pada proses ini, maka penulis menghapusnya dari dataset dan memberi nama dataset yang siap digunakan sebagai ‘dat’.

```
#Remove "ID" column
dat = dataset.drop(columns = "ID")
```

2.2 Membuat Model dari Dataset

Yang pertama, mendefinisikan variabel X dan Y.

```
#Mendefinisikan x dan y
X = dat.iloc[:, :-1].values
Y = dat.iloc[:, -1].values
```

Lalu, penulis me-rescale dataset tersebut ke interval 0 sampai 1. Karena kebanyakan algoritma *machine learning* dapat bekerja lebih baik dengan proses rescale ini (Brownlee, 2016).

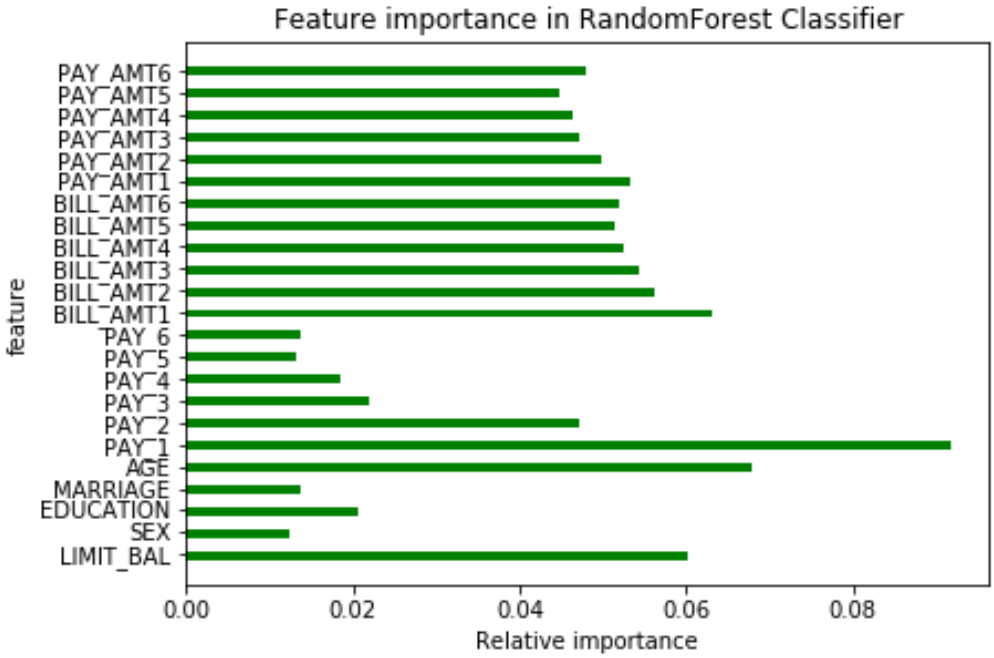
```
#Rescale Data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX = scaler.fit_transform(X)
```

Setelah itu, penulis membagi dataset menjadi *data training* dan *data testing*. Masing-masing sebesar 80% dan 20%.

```
import sklearn.model_selection as model_selection

X_train, X_test, Y_train, Y_test = model_selection.train_test_split(rescaledX, Y, test_size=0.2, random_state=35)
```

Karena menurut penulis 23 variabel bebas itu banyak dan dapat memperlambat proses pada komputer, maka penulis melakukan pemilihan fitur dengan *Random Forest*. Mengapa *Random Forest*? Karena sistem pohon yang digunakan *Random Forest* secara alami membuat peringkat berdasarkan seberapa baik mereka meningkatkan kemurnian *node*. Ini berarti terjadinya pengurangan *impurity* diseluruh pohon (disebut *gini impurity*) (Albon, 2017).

```
#Finding feature importances with Random Forest
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=100, random_state=35)
clf.fit(X_train, Y_train)

for feature in zip(feat_names, clf.feature_importances_):
    print(feature)
```



Disini penulis menggunakan *threshold* sebesar 0.043478 dimana nilai tersebut adalah *mean* dari semua nilai *feature importance* untuk memilih variabel-variabel yang akan digunakan saat proses pembuatan model.

```
#Memilih fitur yang paling berpengaruh
sfm = SelectFromModel(clf, threshold=0.043478)
sfm.fit(X_train, Y_train)

for feature_list_index in sfm.get_support(indices=True):
    print(feat_names[feature_list_index])
```

```
LIMIT_BAL
AGE
PAY_1
PAY_2
BILL_AMT1
BILL_AMT2
BILL_AMT3
BILL_AMT4
BILL_AMT5
BILL_AMT6
PAY_AMT1
PAY_AMT2
PAY_AMT3
PAY_AMT4
PAY_AMT5
PAY_AMT6
```

Karena sudah dilakukan pemilihan fitur, maka kita harus mendefinisikan ulang variabel X nya.

```
#Mendefinisikan x baru
X_important_train = sfm.transform(X_train)
X_important_test = sfm.transform(X_test)
```

Untuk mendapatkan hasil terbaik, penulis menggunakan *10-fold cross validation* untuk mengestimasi akurasi. Ini akan membagi dataset menjadi 10 bagian, 9 bagian untuk proses *training* dan 1 bagian untuk proses *testing* dan akan mengulang untuk semua kombinasi dari hasil pembagian *train-test* yang telah dilakukan sebelumnya. Penulis menggunakan akurasi untuk mengevaluasi model. Akurasi adalah rasio dari jumlah yang diprediksi tepat dibagi dengan jumlah seluruh data.

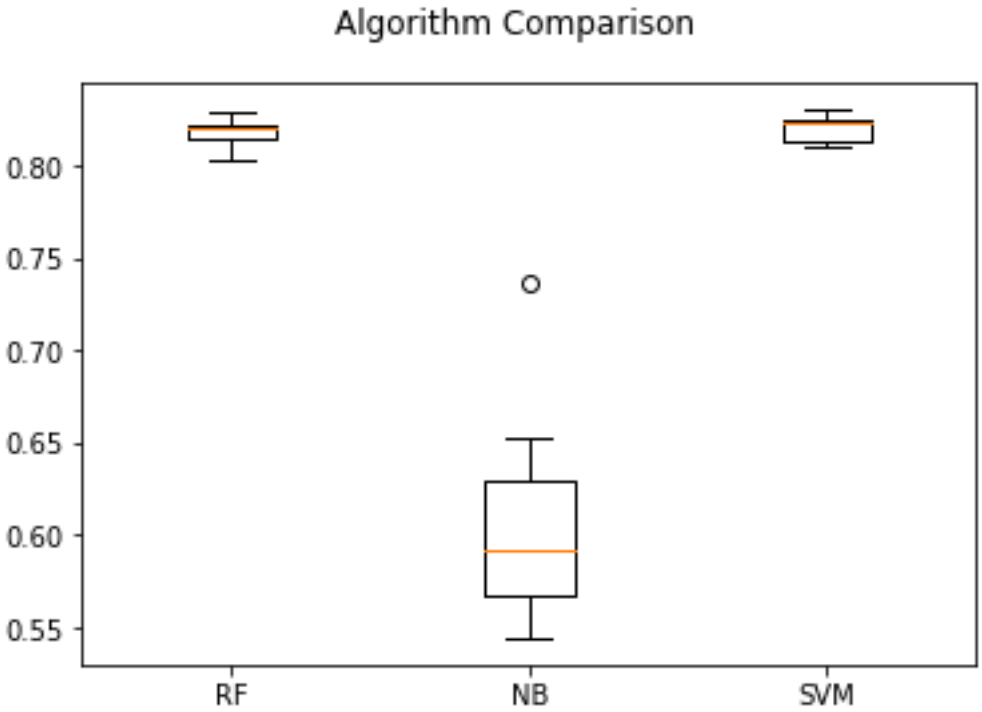
Karena tujuan dari artikel ini adalah untuk membandingkan algoritma random forest, naïve bayes & support vector machine, maka dalam perintahnya dijalankan secara bersamaan.

```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC

# Spot-Check Algorithms
models = []
models.append(('RF', RandomForestClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(kernel='rbf')))

# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = KFold(n_splits=10, random_state=7)
    cv_results = cross_val_score(model, X_important_train, Y_train, cv=kfold,
    scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = " %s: %f " % (name, cv_results.mean())
    print(msg)
```

3. Hasil dan Pembahasan

Untuk *Support Vector Machine* penulis menggunakan kernel Gaussian karena itu yang memberikan akurasi tertinggi dibanding kernel yang lain.

```
RF: 0.817000
 NB: 0.605292
 SVM: 0.820375
```



Dari hasil diatas, terlihat model yang memiliki estimasi akurasi tertinggi adalah *Support Vector Machine* sebesar 82.0375%. Jika dilihat dari *Box and Whisker Plot*, *Box* dan *Whisker* milik SVM lebih kecil dan sedikit lebih tinggi dari *Box* dan *Whisker* milik RF. Oleh karena itu, model yang akan digunakan untuk data *testing* adalah model *Support Vector Machine*.

```
# Make predictions on validation dataset

classifier = SVC(kernel='rbf')
classifier.fit(X_important_train, Y_train)
y_pred = classifier.predict(X_important_test)
print(accuracy_score(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))
```

```
0.8136666666666666
[[4457  169]
 [ 949  425]]
```

Setelah menggunakan *Support Vector Machine* untuk memprediksi data *testing*, didapatkan akurasi sebesar 81,366% atau model tersebut dapat memprediksi 4882 data secara tepat dan 1118 data diprediksi salah.

4. Keterbatasan

Penelitian ini pun tetap ada kekurangannya. Seperti penulis tidak tau apa yang menyebabkan algoritma *Naïve Bayes* memberikan akurasi yang sangat rendah. Pemilihan nilai *random_state* juga memengaruhi tingkat akurasi tetapi itu tetap tidak memengaruhi hasil akhir bahwa SVM adalah model dengan akurasi terbaik. Penulis tidak mengetahui apakah itu sebuah kebetulan dalam pemilihan nilainya atau memang *random_state* tidak memengaruhi hasil akhir. Penulis berharap untuk penelitian kedepannya agar keterbatasan-keterbatasan yang penulis alami dapat di atasi dan hasil menjadi lebih akurat.

5. Kesimpulan

Jadi, menurut penelitian di atas untuk mencegah terjadinya kredit macet dapat menggunakan metode klasifikasi algoritma *Support Vector Machine*. Karena dapat memberikan tingkat akurasi yang lebih tinggi dibandingkan algoritma *Random Forest* dan *Naïve Bayes*. Penggunaan komputer juga lebih disarankan daripada kita harus menghitung secara manual mengingat variabel-variabel yang diperhitungkan cukup banyak.

DAFTAR PUSTAKA

Albon, C. (2017). *Feature Selection Using Random Forest*. https://chrisalbon.com/machine_learning/trees_and_forests/feature_selection_using_random_forest/

Asiri, S. (2018). *Machine Learning Classifiers - Towards Data Science*. https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

Brownlee, J. (2016). *Machine Learning Mastery With Python* (1.4).

What Is Data Mining? (n.d.). Retrieved June 14, 2020, from https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/process.htm#CHDFGCIJ

LAMPIRAN

1. Hasil Turnitin

2. Program

Running Time : 321.609375 seconds

==========

```
from time import process_time

a = process_time()

# Load dataset
import pandas as pd

filename = 'UCI_Credit_Card.csv'
dataset = pd.read_csv(filename)

# head
print(dataset.head())

# descriptions
print(dataset.describe())
```

```
# rename columns
dataset = dataset.rename(columns={'default.payment.next.month': 'def_pay',
                    'PAY_0': 'PAY_1'})

#Moving unlabeled category to fourth category
fil = (dataset.EDUCATION == 5) | (dataset.EDUCATION == 6) |
(dataset.EDUCATION == 0)
dataset.loc[fil, 'EDUCATION'] = 4
print(dataset.EDUCATION.value_counts())

#Moving unlabeled category to third category
dataset.loc[dataset.MARRIAGE == 0, 'MARRIAGE'] = 3
print(dataset.MARRIAGE.value_counts())

#Change value to zero
fil = (dataset.PAY_1 == -2) | (dataset.PAY_1 == -1) | (dataset.PAY_1 == 0)
dataset.loc[fil, 'PAY_1'] = 0
fil = (dataset.PAY_2 == -2) | (dataset.PAY_2 == -1) | (dataset.PAY_2 == 0)
dataset.loc[fil, 'PAY_2'] = 0
fil = (dataset.PAY_3 == -2) | (dataset.PAY_3 == -1) | (dataset.PAY_3 == 0)
dataset.loc[fil, 'PAY_3'] = 0
fil = (dataset.PAY_4 == -2) | (dataset.PAY_4 == -1) | (dataset.PAY_4 == 0)
dataset.loc[fil, 'PAY_4'] = 0
fil = (dataset.PAY_5 == -2) | (dataset.PAY_5 == -1) | (dataset.PAY_5 == 0)
dataset.loc[fil, 'PAY_5'] = 0
fil = (dataset.PAY_6 == -2) | (dataset.PAY_6 == -1) | (dataset.PAY_6 == 0)
dataset.loc[fil, 'PAY_6'] = 0

#Remove "ID" column
dat = dataset.drop(columns = "ID")
feat_names =
["LIMIT_BAL","SEX","EDUCATION","MARRIAGE","AGE","PAY_1","PA
Y_2","PAY_3","PAY_4","PAY_5","PAY_6","BILL_AMT1","BILL_AMT2","
BILL_AMT3","BILL_AMT4","BILL_AMT5","BILL_AMT6","PAY_AMT1",
"PAY_AMT2","PAY_AMT3","PAY_AMT4","PAY_AMT5","PAY_AMT6","
def_pay"]

#Mendefinisikan x dan y
X = dat.iloc[:, :-1].values
Y = dat.iloc[:, -1].values
```

```
#Rescale Data
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(0, 1))
rescaledX = scaler.fit_transform(X)

# Split-out validation dataset
import sklearn.model_selection as model_selection

X_train, X_test, Y_train, Y_test = model_selection.train_test_split(rescaledX,
Y, test_size=0.2, random_state=35)

#Finding feature importances with Random Forest
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=100, random_state=35)
clf.fit(X_train, Y_train)

for feature in zip(feat_names, clf.feature_importances_):
    print(feature)

#Visualisasi Pengaruh Fitur Terhadap Label

import matplotlib.pyplot as plt
import numpy as np

yv = clf.feature_importances_
fig, ax = plt.subplots()
width = 0.4 # the width of the bars
ind = np.arange(len(yv)) # the x locations for the groups
ax.barh(ind, yv, width, color='green')
ax.set_yticks(ind+width/10)
ax.set_yticklabels(feat_names, minor=False)
plt.title('Feature importance in RandomForest Classifier')
plt.xlabel('Relative importance')
plt.ylabel('feature')
plt.figure(figsize=(5,5))
fig.set_size_inches(6.5, 4.5, forward=True)
```

```
#Memilih fitur yang paling berpengaruh
sfm = SelectFromModel(clf, threshold=0.043478)
sfm.fit(X_train, Y_train)

for feature_list_index in sfm.get_support(indices=True):
   print(feat_names[feature_list_index])

#Mendefinisikan x baru
X_important_train = sfm.transform(X_train)
X_important_test = sfm.transform(X_test)

from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
# Spot-Check Algorithms
models = []
models.append(('RF', RandomForestClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(kernel='rbf')))

# evaluate each model in turn
results = []
names = []
for name, model in models:
   kfold = KFold(n_splits=10, random_state=7)
   cv_results = cross_val_score(model, X_important_train, Y_train, cv=kfold,
   scoring='accuracy')
   results.append(cv_results)
   names.append(name)
   msg = " %s: %f " % (name, cv_results.mean())
   print(msg)

# Compare Algorithms

fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
```

```
# Make predictions on validation dataset

classifier = SVC(kernel='rbf')
classifier.fit(X_important_train, Y_train)
y_pred = classifier.predict(X_important_test)
print(accuracy_score(Y_test, y_pred))
print(confusion_matrix(Y_test, y_pred))

b = process_time()

Total_time = b-a
```