LAPORAN PROJECT BASED
CII-2M3 MACHINE LEARNING
AHY - AGUS HARTOYO, S.T., M.Sc., Ph.D.
SEMESTER GANJIL 2022/2023

Oleh :
Ahmad Alfarel - 1301200081
Muhammad Daffa' Ibrahim - 1301204051
Rahmalia Rahadi - 1301201591
Tiara Firdausa Abdillah - 1301204039

IF-44-06

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
TAHUN 2023

BAGIAN I
FORMULASI MASALAH

	Regresi (dataset: autos MPG)	**Klasifikasi (dataset: German credit)**
Bagging	Tugas tipe 0	Tugas tipe 1
Boosting	Tugas tipe 2	Tugas tipe 3

Tipe tugas yang kami kerjakan adalah tipe 1 karena NIM terkecil di kelompok kami adalah 1301200081 modulo 4 = 1. Maka kami mengerjakan klasifikasi pada dataset German credit menggunakan bagging.

Menebak kelas/kategori dari calon kreditur (apakah akan menjadi kreditur yang baik atau buruk) berdasarkan profil calon kreditur yang diberikan yang diwakili oleh atribut-atribut-atribut seperti status pekerjaan, status perkawinan, tujuan kredit, usia, jenis kelamin, dll. Dataset German credit beserta deskripsi atribut-atributnya bisa diakses di tautan berikut (login SSO):
https://drive.google.com/drive/folders/1P7DBFKFoIr1CCZbJp6uFg9hCsjMb7Ssk?usp=share_link
Hal pertama yang dilakukan yaitu mengunggah data set yang akan digunakan pada file google collab. Selanjutnya melihat isi dataset *german_credit.xslx* dalam bentuk tabel menggunakan syntax df. Berikut adalah data dari german credit:

	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex	other_debtors	...	property
0	no checking account	18	all credits at this bank paid back duly	car (used)	1049	unknown/no savings account	< 1 yr	< 20	female : non-single or male : single	none	...	car or other
1	no checking account	9	all credits at this bank paid back duly	others	2799	unknown/no savings account	1 <= ... < 4 yrs	25 <= ... < 35	male : married/widowed	none	...	unknown / no property
2	... < 0 DM	12	no credits taken/all credits paid back duly	retraining	841	... < 100 DM	4 <= ... < 7 yrs	25 <= ... < 35	female : non-single or male : single	none	...	unknown / no property
3	no checking account	12	all credits at this bank paid back duly	others	2122	unknown/no savings account	1 <= ... < 4 yrs	20 <= ... < 25	male : married/widowed	none	...	unknown / no property
4	no checking account	12	all credits at this bank paid back duly	others	2171	unknown/no savings account	1 <= ... < 4 yrs	< 20	male : married/widowed	none	...	car or other
...	...	...	...	...	...	...	...	...	...	...	...	...
995	no checking account	24	no credits taken/all credits paid back duly	furniture/equipment	1987	unknown/no savings account	1 <= ... < 4 yrs	25 <= ... < 35	male : married/widowed	none	...	unknown / no property
996	no checking account	24	no credits taken/all credits paid back duly	others	2303	unknown/no savings account	>= 7 yrs	< 20	male : married/widowed	co-applicant	...	unknown / no property

BAGIAN II
EKSPLORASI DAN PRA-PEMROSESAN DATA

2.1 Missing Value

Missing value merupakan kondisi dimana adanya data yang hilang atau tidak lengkap di dalam dataset. Pada dataset german credit ini tidak terdapat missing value.

```
status                    0
duration                  0
credit_history            0
purpose                   0
amount                    0
savings                   0
employment_duration       0
installment_rate          0
personal_status_sex       0
other_debtors             0
present_residence         0
property                  0
age                       0
other_installment_plans   0
housing                   0
number_credits            0
job                       0
people_liable             0
telephone                 0
foreign_worker            0
credit_risk               0
dtype: int64
```

2.2 Penyelidikan Kualitas Data

Untuk memahami dataset diatas dilakukan penyelidikan kualitas data mengenai dataset tersebut. Pencarian informasi dapat dilakukan dengan syntax df.info() seperti gambar di bawah ini:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 21 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   status                  1000 non-null   object
 1   duration                1000 non-null   int64
 2   credit_history          1000 non-null   object
 3   purpose                 1000 non-null   object
 4   amount                  1000 non-null   int64
 5   savings                 1000 non-null   object
 6   employment_duration     1000 non-null   object
 7   installment_rate        1000 non-null   object
 8   personal_status_sex     1000 non-null   object
 9   other_debtors           1000 non-null   object
 10  present_residence       1000 non-null   object
 11  property                1000 non-null   object
 12  age                     1000 non-null   int64
 13  other_installment_plans 1000 non-null   object
 14  housing                 1000 non-null   object
 15  number_credits          1000 non-null   object
 16  job                     1000 non-null   object
 17  people_liable           1000 non-null   object
 18  telephone               1000 non-null   object
 19  foreign_worker          1000 non-null   object
 20  credit_risk             1000 non-null   object
dtypes: int64(3), object(18)
memory usage: 164.2+ KB
```

df.info() berfungsi untuk menampilkan informasi dari dataset. Informasi tersebut berisi jangkauan index yaitu 0 - 999, total 1000 entri data, total kolom sebanyak 21 dimulai dari 0 - 20, keseluruhan data yang bersifat non-null artinya tidak kosong, tipe data yang terdiri dari, int64 sebanyak 3, dan object sebanyak 18.

2.3 Replace

Replace bertujuan untuk mengganti kemunculan karakter substring tertentu dalam string dengan karakter substring tertentu. Penggantian karakter ini bertujuan agar memudahkan pengerjaan klasifikasi dataset. Fungsi replace() tidak menggantikan string yang sebenarnya, tetapi membuat copyan string, dan menggantikan instance dari string yang ditentukan dengan karakter baru.

```
df['status'] = df['status'].replace(['... < 0 DM', '0<= ... < 200 DM', '... >= 200 DM / salary for at least 1 year', 'no checking account'], [0, 1, 2, 3])

df['credit_history'] = df['credit_history'].replace(['no credits taken/all credits paid back duly', 'all credits at this bank paid back duly', 'existing credits paid back duly till now',

df['purpose'] = df['purpose'].replace(['car (new)', 'car (used)', 'furniture/equipment', 'radio/television', 'domestic appliances', 'repairs', 'education','vacation', 'retraining', 'busi

df['savings'] = df['savings'].replace(['... <  100 DM', '100 <= ... <  500 DM', '500 <= ... < 1000 DM', '... >= 1000 DM', 'unknown/no savings account'], [1, 2, 3, 4, 5])

df['employment_duration'] = df['employment_duration'].replace(['unemployed', '< 1 yr', '1 <= ... < 4 yrs', '4 <= ... < 7 yrs', '>= 7 yrs'],[1, 2, 3, 4, 5])

df['installment_rate'] = df['installment_rate'].replace(['< 20', '20 <= ... < 25', '25 <= ... < 35', '>= 35'], [0, 1, 2, 3])

df['personal_status_sex'] = df['personal_status_sex'].replace(['male : divorced/separated','female : non-single or male : single', 'male : married/widowed', 'female : single', ], [0, 1,

df['other_debtors'] = df['other_debtors'].replace(['none','guarantor', 'co-applicant'], [0, 2, 1])

df['present_residence'] = df['present_residence'].replace(['< 1 yr','1 <= ... < 4 yrs', '4 <= ... < 7 yrs', '>= 7 yrs'],[0, 1, 2, 3])

df['property'] = df['property'].replace(['car or other','unknown / no property','building soc. savings agr./life insurance', 'real estate'], [2, 3, 1, 0])

df['other_installment_plans'] = df['other_installment_plans'].replace(['none', 'bank', 'stores'], [2,0,1])

df['housing'] = df['housing'].replace(['for free','rent','own'], [0,2,1])
```

Setelah dilakukan replace

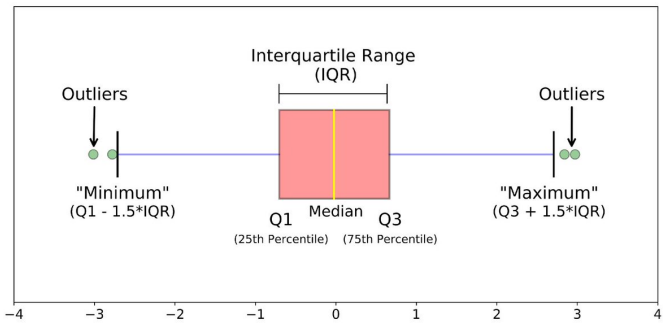
	status	duration	credit_history	purpose	amount	savings	employment_duration	installment_rate	personal_status_sex	other_debtors	...	propert
0	3	18	1	1	1049	5	2	0	1	0	...	
1	3	9	1	10	2799	5	3	2	2	0	...	
2	0	12	0	8	841	1	4	2	1	0	...	
3	3	12	1	10	2122	5	3	1	2	0	...	
4	3	12	1	10	2171	5	3	0	2	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
995	3	24	0	2	1987	5	3	2	2	0	...	
996	3	24	0	10	2303	5	5	0	2	1	...	
997	2	21	1	10	12680	4	5	0	2	0	...	
998	0	12	0	2	6468	4	1	2	2	0	...	
999	3	30	0	1	6350	4	5	0	2	0	...	

1000 rows × 21 columns

2.4 Handle Outlier

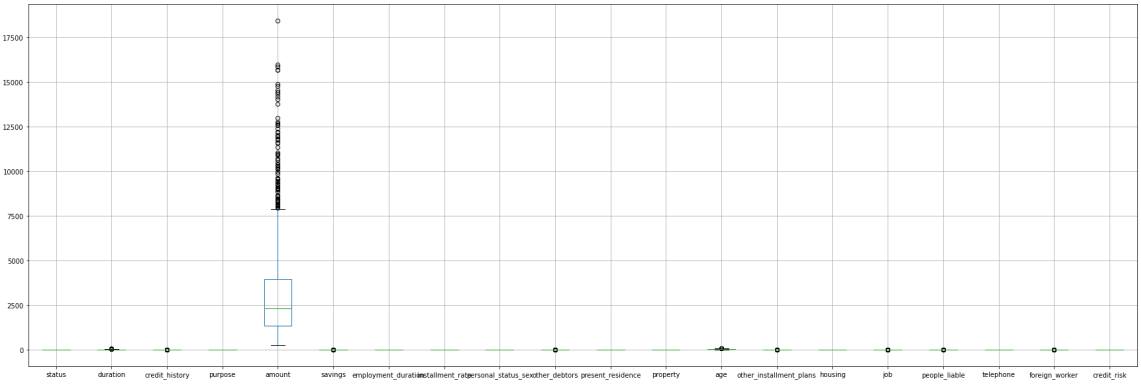
Data Outlier adalah suatu data hasil observasi yang memunculkan nilai-nilai yang berlebihan atau melebihi batas dan jauh berbeda dengan data-data yang masih masuk dalam satu sub set data maka dari itu data ini harus dihilangkan agar hasil analisis lebih akurat. Penanganan outlier pada project based kali ini menggunakan rumus IQR atau Interquartile Range adalah selisih dari kuartil ketiga (persentil 75) dengan kuartil pertama (persentil 25). Jika ditulis dalam formula **IQR = Q3 – Q1**. Implementasi formula tersebut dalam code dapat dituliskan:

```
def handling_outliers(data):
    for x in data.columns:
      Q1 = data[x].quantile(0.25)
      Q3 = data[x].quantile(0.75)
      intahquartile = Q3-Q1

      kicik = Q1 - (intahquartile *1.5)
      badag = Q3 + (intahquartile * 1.5)

      data.loc[data[x] > badag , x] = badag
      data.loc[data[x] < kicik , x] = kicik

handling_outliers(df)
```

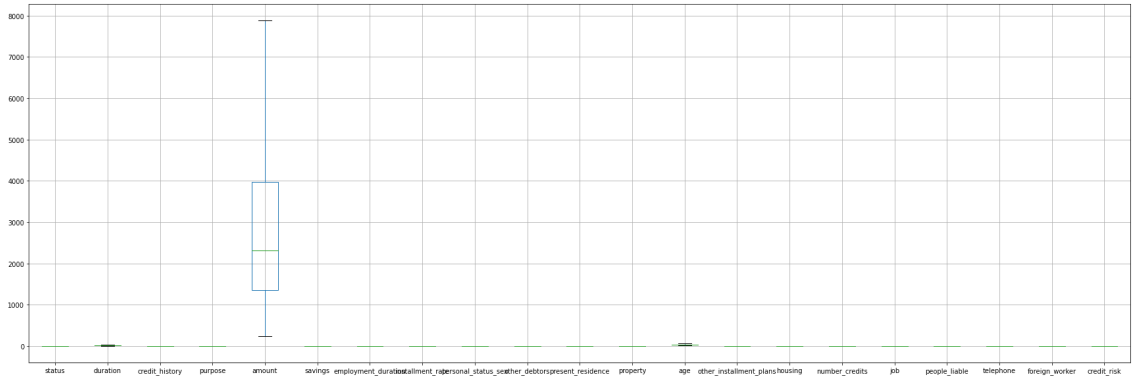


Variable Q1 digunakan untuk menampung quartil 1 yaitu 25% dan variable Q2 digunakan untuk menampung quartil 2 yaitu 75%. Masukkan Q3 - Q1 pada variable IQR. Selanjutnya, kalikan IQR dengan 1.5 (konstanta untuk menemukan outliers) untuk mendapatkan nilai IQR minimum dan maksimum. Untuk IQR minimum, kurangi Q1 dengan 1.5 x IQR, sedangkan untuk IQR maksimum, tambahkan Q3 dengan 1.5 x IQR. Untuk pembuktian bahwa outlier telah hilang dilakukan visualisasi data dalam bentuk bloxpot sebagai berikut:

Data dengan outlier



Data tanpa outlier



2.5 Data Split

Data splitting atau pemisahan data adalah metode membagi data menjadi dua bagian atau lebih yang membentuk subhimpunan data. Pada dasarnya data splitting dibagi menjadi dua bagian, yakni data training dan data testing. Data training atau data latih digunakan untuk melatih dan mengembangkan model. Kumpulan data training biasanya digunakan untuk mengestimasikan parameter yang berbeda atau untuk membandingkan kinerja model yang berbeda.

```
x = df.iloc[:, 0:-1]
y = df.iloc[:, -1]

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

- x_train: Untuk menampung *data source* yang akan dilatih.
- x_test: Untuk menampung *data target* yang akan dilatih.
- y_train: Untuk menampung *data source* yang akan digunakan untuk testing.
- y_test: Untuk menampung *data target* yang akan digunakan untuk testing.

Parameter test_size digunakan untuk mendefinisikan ukuran data testing. Dalam contoh di atas, test_size=0.25 berarti data yang digunakan sebagai data testing adalah sebesar 25% dari keseluruhan dataset.

BAGIAN III
PEMODELAN

Random Forest

Random forest merupakan salah satu algoritma yang digunakan untuk pengklasifikasian dataset dalam jumlah besar. Dengan menggunakan random forest pada klasifikasi data maka, akan menghasilkan vote yang paling baik. Pada saat proses klasifikasi selesai dilakukan, inisialisasi dilakukan dengan sebanyak data berdasarkan nilai akurasinya.

Keuntungan penggunaan random forest yaitu mampu mengklasifikasi data yang memiliki atribut yang tidak lengkap, dapat digunakan untuk klasifikasi dan regresi akan tetapi tidak terlalu bagus untuk regresi, lebih cocok untuk pengklasifikasian data serta dapat digunakan untuk menangani data sampel yang banyak.

```
from sklearn.metrics import
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
y_pred_train = clf.predict(x_train)
print("ACCURACY OF THE MODEL: ", metrics.accuracy_score(y_test, y_pred))
```

Mendefinisikan variabel classifier dengan menggunakan fungsi RandomForestClassifier dan parameter n_estimator = 100 yang artinya kita membuat 100 kelompok bootstrap. Lalu mendefinisikan y_pred untuk memprediksi hasil model random forest classification ke test set.

BAGIAN IV
EVALUASI

Confusion Matrix

Confusion Matrix adalah pengukuran performa untuk masalah klasifikasi machine learning dimana keluaran dapat berupa dua kelas atau lebih. Confusion Matrix adalah tabel dengan 4 kombinasi berbeda dari nilai prediksi dan nilai aktual. Ada empat istilah yang merupakan representasi hasil proses klasifikasi pada confusion matrix yaitu True Positif, True Negatif, False Positif, dan False Negatif. Serta menampilkan dan membandingkan nilai aktual atau nilai sebenarnya dengan nilai hasil prediksi model yang dapat digunakan untuk menghasilkan metrik evaluasi seperti *Accuracy* (akurasi), *Precision*, *Recall*, dan *F1-Score* atau *F-Measure*.

```
[[ 31  44]
 [ 16 159]]
Classification Report
              precision    recall  f1-score   support

         0.0       0.66      0.41      0.51        75
         1.0       0.78      0.91      0.84       175

    accuracy                           0.76       250
   macro avg       0.72      0.66      0.67       250
weighted avg       0.75      0.76      0.74       250
```

BAGIAN V
EKSPERIMEN

5. 1 Data Train 75% dan Data Test 25%

5.1.1 Jumlah n = 10

```
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier(n_estimators = 10)
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

print(f'accuracy : {accuracy_score(y_test,y_pred)*100}%')
```

```
accuracy : 70.39999999999999%
```

5.1.2 Jumlah n = 50

```
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier(n_estimators = 50)
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

print(f'accuracy : {accuracy_score(y_test,y_pred)*100}%')
```

```
accuracy : 72.39999999999999%
```

5.1.3 Jumlah n = 100

```
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

print(f'accuracy : {accuracy_score(y_test,y_pred)*100}%')
```

```
accuracy : 74.4%
```

5.2 Data Train 80% dan Data Test 20%

5.2.1 Jumlah n = 20

```
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier(n_estimators = 20)
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

print(f'accuracy : {accuracy_score(y_test,y_pred)*100}%')
```

```
accuracy : 77.5%
```

5.2.2 Jumlah n = 50

```
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier(n_estimators = 50)
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

print(f'accuracy : {accuracy_score(y_test,y_pred)*100}%')
```

```
accuracy : 76.0%
```

5.2.3 Jumlah n = 100

```
from sklearn.metrics import accuracy_score
clf = RandomForestClassifier(n_estimators = 100)
clf.fit(x_train,y_train)
y_pred = clf.predict(x_test)

print(f'accuracy : {accuracy_score(y_test,y_pred)*100}%')
```

```
accuracy : 76.5%
```

BAGIAN VI
KESIMPULAN

Setelah melakukan pemodelan dengan menggunakan algoritma random forest yang merupakan salah satu metode bagging serta beberapa eksperimen yang telah dilakukan, dapat disimpulkan bahwa terdapat beberapa faktor yang dapat mempengaruhi hasil akurasi dari prediksi yang dilakukan. Pemilihan metode pada preprocessing data dapat mempengaruhi hasil akurasi seperti perbandingan antara jumlah data uji dengan jumlah data latih. Selain itu, jumlah kelompok pada saat melakukan bootstrap pun dapat mempengaruhi nilai akurasi yang didapatkan. Pemilihan algoritma yang berbeda akan menghasilkan proses bagging serta hasil yang berbeda.

LAMPIRAN

Source code (Google Collab):
https://colab.research.google.com/drive/1HU4q-IUu1eYuhX9FnRrA8xBH8kkSAUXb?usp=sharing

Video presentasi:
https://drive.google.com/file/d/1pvkh0ngLkfULlL0rQKMEPctYMofxb0Qr/view?usp=drivesdk