



# MACHINE LEARNING



## PROJECT REPORT

- CRISP-DM Methodology
- Business Understanding and Data Understanding
- Data Preparation and Modelling
- Evaluation

<https://kmmi.kemdikbud.go.id>

## DAFTAR ISI

<b><i>CRISP-DM Methodology</i></b>	<b>2</b>
<b><i>Business Understanding</i></b>	<b>3</b>
<b><i>Data Understanding</i></b>	<b>4</b>
Data Exploration	6
<b><i>Data Preparation</i></b>	<b>8</b>
Missing Values	8
Data Transformation	8
Feature Engineering	9
<b><i>Modelling</i></b>	<b>13</b>
KNN	13
Gaussian-Naive Bayes	17
SVM	20
<b><i>Evaluation</i></b>	<b>24</b>
<b><i>Team Profiles</i></b>	<b>25</b>

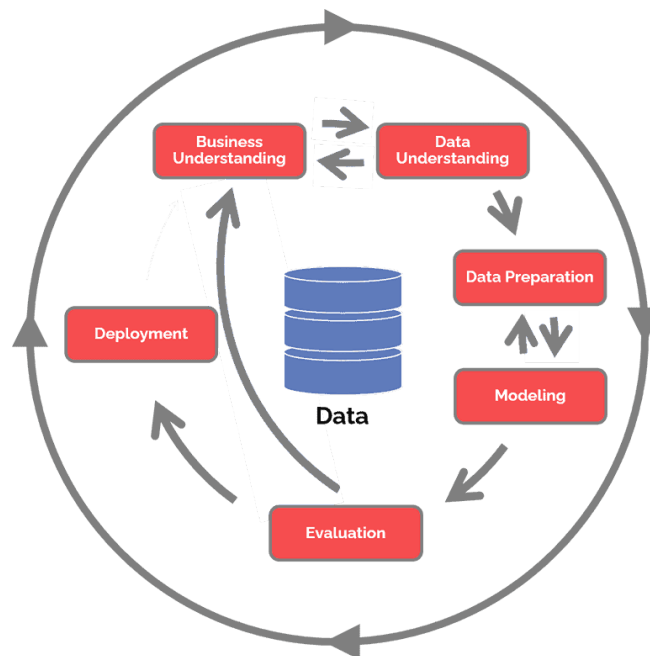
## CRISP-DM METHODOLOGY

Cross Industry Standard Process for Data Mining (CRISP-DM) adalah model proses dengan enam fase yang secara alami menggambarkan life cycle data science yang menggunakan Machine Learning. Metodologi ini akan membantu anda merencanakan, mengatur, dan mengimplementasikan proyek.

CRISP-DM dan Metodologi Data science dari IBM diawali dengan kegiatan Business Understanding yang merupakan proses pemahaman terhadap masalah yang akan diselesaikan. Di dalam kegiatan tersebut juga dilakukan proses pemetaan antara masalah bisnis dengan tugas analitik (tugas data science yang sesuai).

Berikutnya kegiatan pemahaman terhadap data (Data Understanding) yang meliputi penentuan kebutuhan data, pengumpulan data dan eksplorasi data. Pada Metodologi IBM masing-masing sub kegiatan dijadikan proses tersendiri.

Langkah berikutnya adalah Data Preparation yang dilakukan untuk memperbaiki kualitas data agar sesuai dengan proses Modeling yang akan dilakukan berikutnya. Kualitas model yang dihasilkan di evaluasi (Evaluation) sebelum di-deploy menjadi sistem operasional. Rangkaian kegiatan diakhiri dengan proses feedback dan pelaporan.



Gambar 0. Siklus Metode CRISP-DM

## BUSINESS UNDERSTANDING

Administrasi Bisnis Kecil (SBA) adalah sebuah badan pemerintah Amerika yang didirikan pada tahun 1953. Badan ini didirikan untuk mendorong dan mempromosikan ekonomi secara umum dengan memberikan bantuan kepada usaha kecil.

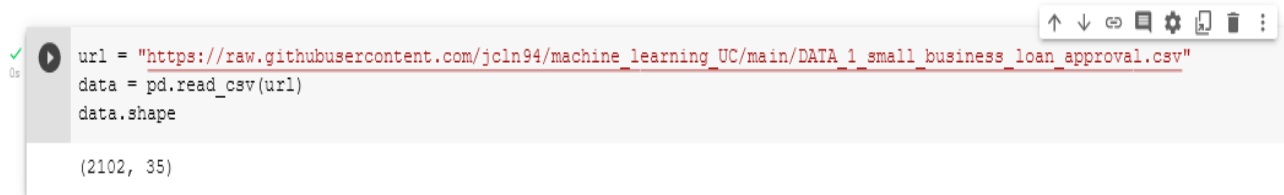
Permasalahan yang ingin diselesaikan adalah persetujuan pinjaman untuk bisnis kecil. Dalam memberikan pinjaman modal usaha, sebisa mungkin peminjam modal mengusahakan agar dapat terhindar dari kebangkrutan. Oleh karena itu kita perlu membuat adanya aturan atau regulasi yang baik, agar pinjaman modal tepat sasaran. Untuk mendapatkan pinjaman dari masing-masing bank, setiap usaha harus memenuhi kriteria berdasarkan variabel yang disediakan yang akan menentukan apakah usaha tersebut akan mendapatkan pinjaman atau tidak.

Kami bertujuan menganalisis dataset ini menggunakan model Machine Learning yang dapat menentukan terpilih atau tidaknya usaha untuk mendapat pinjaman berdasarkan beberapa variabel yang kami pilih melalui perhitungan korelasi. Model Machine Learning yang kami pilih adalah K-NN, Naive Bayes, dan SVM. Ketiga model tersebut kami lihat sesuai untuk digunakan dengan dataset kami dan akan dibahas lebih lanjut pada keterangan masing-masing model.

Untuk pemilihan evaluation metrics yang digunakan adalah F1-Score, nilai terbaiknya adalah 1 dan nilai terburuknya adalah 0 dan jika F1-Score memiliki hasil yang baik maka akan mengindikasikan model klasifikasi kami memiliki precision dan recall yang baik. Sebab nilai False Positive maupun False Negative akan berakibat buruk untuk model kami sehingga kami memutuskan untuk mengurangi kedua hasil tersebut, yang dapat ditunjukkan melalui evaluation metrics F1-Score.

## DATA UNDERSTANDING

Berdasarkan pengamatan kelompok, kami melihat bahwa data yang disediakan ini merupakan data riwayat badan usaha yang mendapatkan pinjaman modal atau tidak. Data ini menyimpan informasi sebanyak 35 Kolom dan 2102 baris data . Bisa dilihat pada Gambar1.A.



```
url = "https://raw.githubusercontent.com/jc1n94/machine_learning_UC/main/DATA_1_small_business_loan_approval.csv"
data = pd.read_csv(url)
data.shape

(2102, 35)
```

Gambar1.A

Beberapa penjelasan mengenai kolom yang ada pada dataset dapat dilihat pada Tabel1.

Variabel	Tipe Data	Deskripsi Variabel
Selected	Integer	= 1 jika data tersebut dipilih sebagai data training untuk membuat assignment, = 0 jika data tersebut dipilih sebagai data testing untuk memvalidasi model
LoanNr_ChkDgt	Integer	Pengidentifikasi
Name	String	Nama peminjam
City	String	Asal kota peminjam
State	String	Asal negara peminjam
Zip	Integer	Kode postal peminjam
Bank	String	Nama bank
BankState	String	Asal negara bank
NAICS	Integer	North American Industry Classification System code, kode yang digunakan peminjam untuk mendeskripsikan apa usahanya.
ApprovalDate	Integer	Tanggal pengajuan pinjaman (SBA)
ApprovalFY	Integer	Tahun fiskal komitmen
Term	Integer	Jangka peminjaman dalam waktu sebulan
NoEmp	Integer	Jumlah karyawan pada usaha
NewExist	Float	1 = usaha lama, 2 = usaha baru
CreateJob	Integer	Jumlah pekerjaan yang dibuat
RetainedJob	Integer	Jumlah pekerjaan yang dipertahankan
FranchiseCode	Integer	Kode franchise, (00000 atau 00001) = Tidak ada franchise

UrbanRural	Integer	1 = daerah perkotaan, 2 = daerah pedesaan, 0 = tidak diketahui
RevLineCr	String	Jalur kredit bergulir : Y = Yes, N = No
LowDoc	String	Program pinjaman LowDoc, Y = Iya (Yes), N = Tidak (No)
ChgOffDate	Float	Tanggal saat pinjaman dinyatakan wanprestasi
DisbursementDate	Float	Tanggal pencairan dana
DisbursementGross	Integer	Jumlah dana yang dicairkan
BalanceGross	Integer	Jumlah bruto yang beredar
MIS_Status	String	Status pinjaman (CHGOFF = dikenakan biaya / charged off; PIF = dibayarkan penuh / Paid In Full)
ChgOffPrinGr	Integer	Jumlah dana yang dibebankan
GrApprv	Integer	Jumlah bruto dana pinjaman yang disetujui oleh bank
SBA_Appv	Integer	Jumlah pinjaman yang disetujui SBA
New	Integer	= 1 apabila variabel NewExist = 2 (usaha baru), = 0 apabila variabel NewExist = 1 (usaha lama)
RealEstate	Integer	= 1 apabila pinjaman didukung real estat, = 0 jika tidak
Portion	Float	Proporsi jumlah kotor yang dijamin oleh SBA
Recession	Integer	=1 apabila pinjaman aktif pada waktu Resesi Besar, = 0 jika tidak
daysterm	Integer	= 1 jika MIS_Status=CHGOFF, = 0 Jika MIS_Status=PIF
xx	Float	variabel tambahan yang dihasilkan saat "Recession" dalam section
Default	Integer	variabel tambahan yang dihasilkan saat "Recession" dalam section

Tabel1

Kita bisa melihat gambaran mengenai dataset dengan cara memanggil fungsi head() untuk menampilkan 5 data teratas seperti Gambar1.B

```
[3] data.head()
```

	Selected	LoanNr_ChkDgt	Name	City	State	Zip	Bank	BankState	NAICS	ApprovalDate	ApprovalFY	Term	NoEmp	NewExist	CreateJob
0	0	1004285007	SIMPLEX OFFICE SOLUTIONS	ANAHEIM	CA	92801	CALIFORNIA BANK & TRUST	CA	532420	15074	2001	36	1	1.0	C
1	1	1004535010	DREAM HOME REALTY	TORRANCE	CA	90505	CALIFORNIA BANK & TRUST	CA	531210	15130	2001	56	1	1.0	C
2	0	1005005006	Winset, Inc. dba Bankers Hill	SAN DIEGO	CA	92103	CALIFORNIA BANK & TRUST	CA	531210	15188	2001	36	10	1.0	C
3	1	1005535001	Shiva Management	SAN DIEGO	CA	92108	CALIFORNIA BANK & TRUST	CA	531312	15719	2003	36	6	1.0	C
4	1	1005996006	GOLD CROWN HOME LOANS, INC	LOS ANGELES	CA	91345	SBA - EDF ENFORCEMENT ACTION	CO	531390	16840	2006	240	65	1.0	3

Gambar1.B

## DATA EXPLORATION

Selanjutnya kita ingin mengecek data mana saja yang memiliki missing values dan apa tipe datanya. Kita membuat fungsi sendiri untuk mengecek, seperti pada Gambar1.C dan hasil yang memiliki missing value, jumlah, beserta tipe datanya pada Gambar1.D

```
def cekna(df):
    col = []
    d_type = []
    na = []
    na_value = []

    for i in df.columns:
        col.append(i)
        d_type.append(df[i].dtypes)
        na.append(df[i].isna().any())
        na_value.append((df[i].isnull().sum() / len(df[i])) * 100)

    return pd.DataFrame({'Column': col, 'd_type': d_type, 'Is NA': na, 'NA_value':na_value})
```

Gambar1.C

6	Bank	object	True	0.142721
7	BankState	object	True	0.142721
13	NewExist	float64	True	0.047574
18	RevLineCr	object	True	0.095147
19	LowDoc	object	True	0.142721
20	ChgOffDate	float64	True	66.841104
21	DisbursementDate	float64	True	0.142721
33	xx	float64	True	0.142721

Gambar1.D

Untuk tipe data numerik, kita akan mengecek nilai distribusinya , yang berguna untuk menentukan nilai mean atau median yang akan kita pakai untuk mengisi nilai NA. Kita bisa membuat dataframe baru dengan nama datana yang berisi 'NewExist', 'ChgOffDate', 'DisbursementDate', 'xx'. Kemudian kita tampilkan histogramnya seperti Gambar1.E.

```

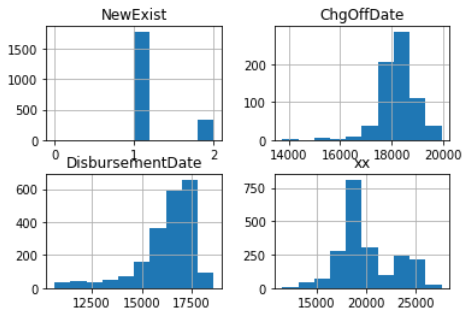
1 #dari data numerik yang memiliki NA , kita bisa membuat dataframe baru untuk mlihat histogramnya
2 #yang berguna untuk menentukan metoda apa yang akan digunakan untuk mengisi NA
3 datana = data[['NewExist', 'ChgOffDate', 'DisbursementDate', 'xx']]
4 datana.hist()

```

```

array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f23f722dfd0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f23f71b4510>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f23f71669d0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f23f711ac10>]],
      dtype=object)

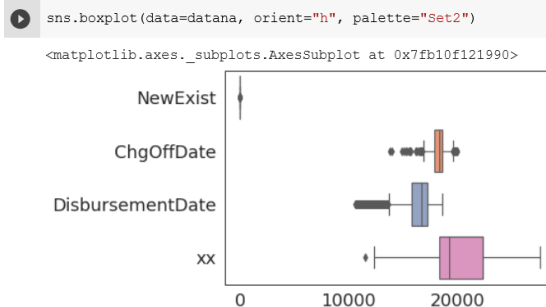
```



Gambar1.E

Dari informasi diatas, kita mengetahui bahwa dari masing-masing variabel ternyata memiliki nilai distribusi selain normal, oleh karena itu kita bisa mengisi NA dengan median dari masing masing kolom.

Selain mencari NA , kita juga perlu mengetahui apakah ada outlier pada data kita, yaitu data yang melenceng dari pengamatan. Untuk mempermudah proses, kita akan mencari apakah ada outlier dari dataframe datana yang merupakan data yang memiliki NA. Untuk itu bisa lihat Gambar1.F.



Gambar1.F

Kita juga bisa melihat nilai tendensi sentral dari masing-masing kolom dataset dengan menggunakan fungsi describe() seperti Gambar1.G.

```

data.describe()
#Untuk melihat deskripsi pada setiap kolom

```

	Selected	LoanNr_ChkDgt	Zip	NAICS	ApprovalDate	ApprovalFY	Term	NoEmp	NewExist	CreateJob	RetainedJob	Franc
count	2102.000000	2.102000e+03	2102.000000	2102.000000	2102.000000	2102.000000	2102.000000	2102.000000	2101.000000	2102.000000	2102.000000	2102.000000
mean	0.500000	4.469172e+09	92698.612274	531630.902950	16179.587060	2004.035680	126.980495	10.150809	1.153736	2.549952	5.803996	196.000000
std	0.500119	2.530069e+09	1878.208435	521.836986	1454.931276	4.006321	93.798944	34.402420	0.362099	8.010175	18.977996	112.000000
min	0.000000	1.004285e+09	65757.000000	531110.000000	10554.000000	1989.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	2.392978e+09	91402.000000	531210.000000	15695.750000	2003.000000	60.000000	2.000000	1.000000	0.000000	1.000000	0.000000
50%	0.500000	3.621730e+09	92559.500000	531312.000000	16556.000000	2005.000000	84.000000	3.000000	1.000000	0.000000	2.000000	0.000000
75%	1.000000	6.551607e+09	94127.750000	532230.000000	17149.750000	2007.000000	240.000000	8.000000	1.000000	2.000000	5.000000	0.000000
max	1.000000	9.958873e+09	96161.000000	533110.000000	18911.000000	2012.000000	306.000000	650.000000	2.000000	130.000000	535.000000	896.000000

Gambar1.G



## DATA PREPARATION

Setelah kita memahami data kita, langkah selanjutnya adalah melakukan pembersihan terhadap data kita. Kita bisa mulai dari missing values.

### MISSING VALUES

Pada dataset kita yang bernama data , kita sudah menyalin kolom kolom yang mempunyai missing values ke dalam dataframe yang bernama datana untuk diidentifikasi distribusinya. Langkah selanjutnya adalah kita mengisi kolom yang ada missing values pada dataframe kita yang bernama data sesuai dengan jenis distribusinya. Karena semua distribusi nya bukan distribusi normal , maka kita isi missing values nya menggunakan median, seperti Gambar2.A

```
# Fill Missing Values

data['NewExist'].fillna(data['NewExist'].median(), inplace=True)
data['ChgOffDate'].fillna(data['ChgOffDate'].median(), inplace=True)
data['DisbursementDate'].fillna(data['DisbursementDate'].median(), inplace=True)
data['xx'].fillna(data['xx'].median(), inplace=True)
```

Gambar2.A

### DATA TRANSFORMATION

Pertama-tama kita akan men-drop beberapa variabel yang dapat dihapuskan, seperti variabel identifikasi dan tanggal (Gambar 2.B).

Drop variabel identifikasi

```
1 data = data.drop(['LoanNr_ChkDgt', 'Selected', 'Name', 'City', 'State', 'Zip', 'Bank', 'BankState', 'NAICS',
2 | | | | | | | 'BalanceGross', 'UrbanRural'], axis=1)
```

Drop data tanggal

```
1 data = data.drop(['ApprovalDate', 'ChgOffDate', 'DisbursementDate'], axis=1)
```

Gambar2.B

Sesudah itu, kolom yang isinya masih berupa huruf / object akan kita encode menggunakan LabelEncoder. Sebelumnya, beberapa kolom seperti RevLineCr dan LowDoc memiliki data yang salah ketik sehingga kita akan menyelesaikannya terlebih dahulu. Untuk kolom FranchiseCode akan kita ubah menjadi 0 dan 1 sesuai dengan keberadaan franchise dari bisnis tersebut atau tidak (Gambar2.C). Setelah itu barulah data dapat kita encode (Gambar2.D).

```
1 # replacing values (other than Y will be N)
2 # FranchiseCode will be replaced as 1 (if business has franchise) and 0 (if business doesn't have franchise)
3 data.loc[data['RevLineCr'] != 'Y', 'RevLineCr'] = 'N'
4 data.loc[data['LowDoc'] != 'Y', 'LowDoc'] = 'N'
5 data.loc[data['FranchiseCode'] == 1, 'FranchiseCode'] = 0
6 data.loc[data['FranchiseCode'] != 0, 'FranchiseCode'] = 1
```

Gambar2.C

```

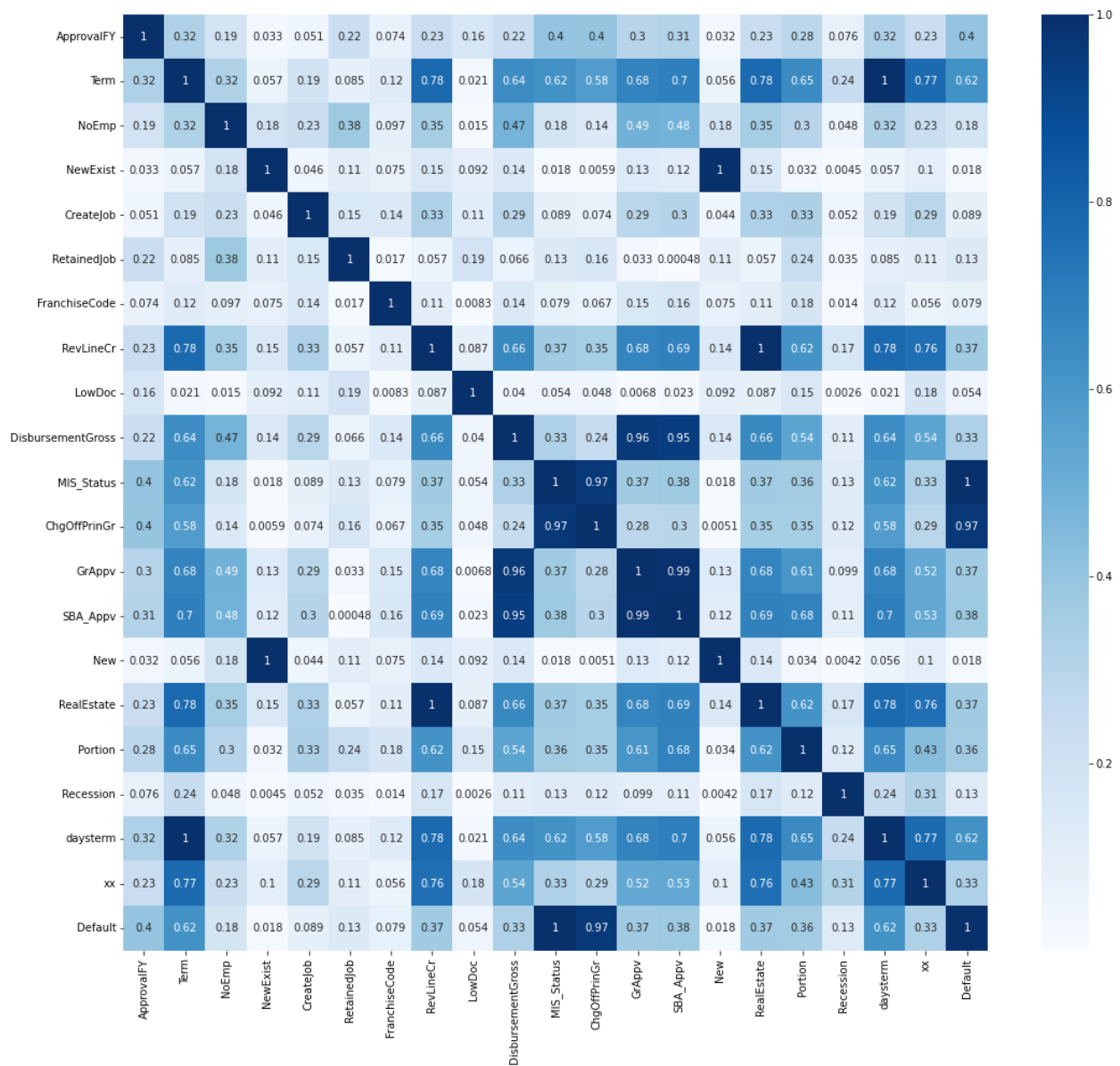
1 labelencoder = LabelEncoder()
2 data['RevLineCr'] = labelencoder.fit_transform(data['RealEstate'])
3 data['LowDoc'] = labelencoder.fit_transform(data['LowDoc'])
4 data['MIS_Status'] = labelencoder.fit_transform(data['MIS_Status'])

```

Gambar2.D

## FEATURE ENGINEERING

Dari pembersihan yang telah dilakukan kita tersisa dengan 21 variabel. Dari ke-21 variabel ini kita akan lihat korelasi antar variabelnya (Gambar2.E) dan men-drop variabel dengan threshold > 0.85 (Gambar2.F). Hal ini kami lakukan untuk mengurangi variabel yang berulang / redundant. Setelah itu kita dapat melihat korelasi antar kelas pada Gambar2.G.



Gambar2.E

```

1 def correlation(dataset, threshold):
2     col_corr = set()
3     corr_matrix = dataset.corr()
4     for i in range(len(corr_matrix.columns)):
5         for j in range(i):
6             if abs(corr_matrix.iloc[i, j]) > threshold:
7                 colname = corr_matrix.columns[j]
8                 col_corr.add(colname)
9     return col_corr

```

```

1 corr_features = correlation(df, 0.85)
2 corr_features

```

```

{'DisbursementGross',
 'GrAppv',
 'MIS_Status',
 'NewExist',
 'RealEstate',
 'RevLineCr',
 'Term',
 'daysterm'}

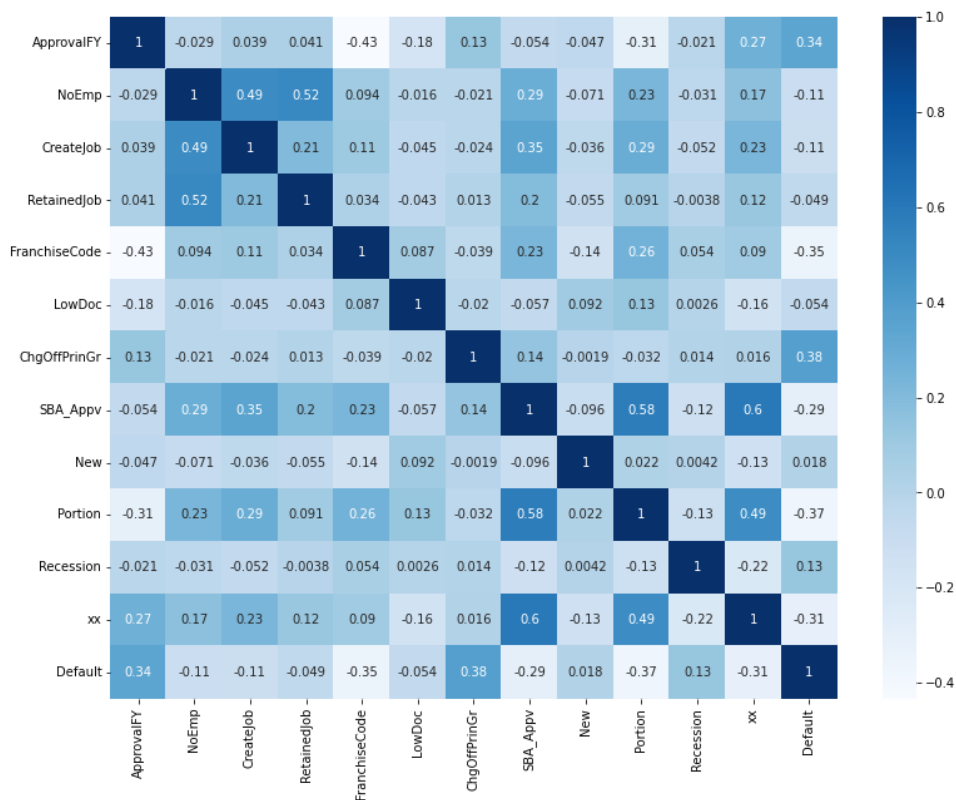
```

```

1 df = df.drop(corr_features, axis=1)

```

Gambar2.F



Gambar2.G

Langkah selanjutnya adalah *features selection*, dimana kita akan menentukan variabel terbaik untuk dibuatkan modelnya sebab semakin banyak variabel bisa membuat akurasi model kita buruk. Kelompok kami menggunakan *Univariate selection* untuk melihat variabel yang memiliki urutan 10 teratas untuk cocok dimasukkan kedalam model. Cara menggunakannya adalah dengan mengimport library dibawah ini.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
```

Untuk penggunaannya bisa kita lihat pada Gambar 15.

```
1 X,y = df.loc[:,df.columns != 'Default'], df.loc[:,'Default']
```

```
1 #apply SelectKBest class to extract top 10 best features
2 bestfeatures = SelectKBest(score_func=chi2, k=10)
3 fit = bestfeatures.fit(X,y)
4 dfscores = pd.DataFrame(fit.scores_)
5 dfcolumns = pd.DataFrame(X.columns)
6
7 #concat two dataframes for better visualization
8 featureScores = pd.concat([dfcolumns,dfscores],axis=1)
9 featureScores.columns = ['Specs','Score'] #naming the dataframe columns
10 print(featureScores.nlargest(10,'Score')) #print 10 best features
```

	Specs	Score
7	SBA_Appv	8.635022e+07
6	ChgOffPrinGr	8.627104e+07
11	xx	8.196460e+04
1	NoEmp	2.942854e+03
2	CreateJob	6.135137e+02
3	RetainedJob	3.070797e+02
10	Recession	3.235524e+01
9	Portion	1.471144e+01
4	FranchiseCode	1.258125e+01
5	LowDoc	6.043331e+00

Gambar2.H

Kelompok kami memilih 6 variabel teratas untuk dijadikan variabel independen yaitu SBA\_Appv, ChgOffPrinGr, xx, CreateJob, LowDoc, dan Recession , dan variabel Default sebagai variabel dependent nya.

Terakhir, kita akan membuat dataframe baru yang berisi 6 variabel independen teratas serta variabel dependen nya (Gambar2.I). Kemudian dataframe ini akan dipecah ke dalam x dan y serta akan dinormalisasikan menggunakan MinmaxScaler (Gambar2.J).

```
1 new_df = df[['SBA_Appv', 'ChgOffPrinGr', 'xx', 'CreateJob', 'LowDoc', 'Recession', 'Default']]
2 new_df.head()
```

	SBA_Appv	ChgOffPrinGr	xx	CreateJob	LowDoc	Recession	Default
0	15000	0	16175.0	0	0	0	0
1	15000	0	17658.0	0	0	1	0
2	15000	0	16298.0	0	0	0	0
3	25000	0	16816.0	0	0	0	0
4	343000	0	24103.0	3	0	0	0

Gambar2.I

```
1 X,y = new_df.loc[:, new_df.columns != 'Default'], new_df.loc[:, 'Default']
```

```
1 # Normalisasi data
2 from sklearn.preprocessing import MinMaxScaler
3
4 scaler = MinMaxScaler()
5 x_scaled = pd.DataFrame(scaler.fit_transform(X), columns = X.columns)
6 x_scaled.head()
```

	SBA_Appv	ChgOffPrinGr	xx	CreateJob	LowDoc	Recession
0	0.006035	0.0	0.289349	0.000000	0.0	0.0
1	0.006035	0.0	0.381610	0.000000	0.0	1.0
2	0.006035	0.0	0.297001	0.000000	0.0	0.0
3	0.010768	0.0	0.329227	0.000000	0.0	0.0
4	0.161283	0.0	0.782568	0.023077	0.0	0.0

Gambar2.J

## MODELLING

## 1. K-NN

Untuk pemilihan algoritma pertama kami menggunakan K-NN (K-Nearest Neighbor). K-NN adalah sebuah metode untuk melakukan klasifikasi terhadap objek yang berdasarkan dari data pembelajaran yang jaraknya paling dekat dengan objek tersebut. KNN merupakan algoritma supervised learning dimana hasil dari query instance yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada algoritma KNN

Tujuan dari algoritma k-NN adalah untuk mengklasifikasi objek baru berdasarkan atribut dan training samples. Dimana hasil dari sampel uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada k-NN. Pada proses pengklasifikasian, algoritma ini tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Algoritma k-NN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sampel uji yang baru.

Pertama-tama, import library yang diperlukan. Karena kami menggunakan K-NN, kami menggunakan library yang sudah disediakan dari scikit learn, yaitu **KNeighborsClassifier** untuk model algoritma K-NN itu sendiri. Untuk  $k = 4$  itu kami pakai karena aturan standar yang ada.

```
from sklearn.neighbors import KNeighborsClassifier
# model menggunakan KNeighborsClassifier dengan K = 4
knn = KNeighborsClassifier(n_neighbors = 4)
```

Selanjutnya kita melakukan fitting pada dataset trainingnya. Setelah itu, kita dapat melihat hasil prediksi serta nilai akurasinya.

```
# Lakukan training dengan dataset train
knn.fit(X_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                    weights='uniform')
```

```
1 # Lakukan prediksi menggunakan x_test lalu print class hasil prediksi
2 prediction_KNN = knn.predict(x_test)
3 print('prediction : {}'.format(prediction_KNN))
```

```
prediction : [0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0
1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 0 1
0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0 0 0 0]
```

```

1 # Print akurasi dari data testing menggunakan score
2 from sklearn.metrics import confusion_matrix, accuracy_score
3
4 print("Accuracy Score: ", knn.score(x_test, y_test))

```

Accuracy Score: 0.9619047619047619

```

1 # Model complexity
2 neig = np.arange(1,25)
3 train_accuracy = []
4 test_accuracy = []
5 # Loop over different values of k
6 for i, k in enumerate(neig):
7     # k from 1 to 25
8     knn = KNeighborsClassifier(n_neighbors=k)
9     #fit with knn
10    knn.fit(x_train, y_train)
11    #train accuracy
12    train_accuracy.append(knn.score(x_train, y_train))
13    #test accuracy
14    test_accuracy.append(knn.score(x_test, y_test))
15 # Plot
16 plt.figure(figsize=[13,8])
17 plt.plot(neig, test_accuracy, label= "Testing Accuracy")
18 plt.plot(neig, train_accuracy, label= "Training Accuracy")
19 plt.legend()
20 plt.title('-value vs accuracy')
21 plt.xlabel('number of neighbors')
22 plt.ylabel('accuracy')
23 plt.xticks(neig)
24 plt.savefig('graph.png')
25 plt.show()
26 print('Best accuracy is {} with k = {}'.format(np.max(test_accuracy), 1+test_accuracy.index(np.max(test_accuracy))))

```



Best accuracy is 0.969047619047619 with k = 6

Karena dikatakan bahwa akurasi terbaik dicapai dengan  $K = 6$ , kita akan mencoba membuat ulang model KNN dengan  $K = 6$ .

```
1 knn = KNeighborsClassifier(n_neighbors = 6)
```

```
1 # Lakukan training dengan dataset train
2 knn.fit(x_train, y_train)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=6, p=2,
                     weights='uniform')
```

```
1 # Print akurasi dari data testing menggunakan score
2 print("Accuracy Score: ", knn.score(x_test, y_test))
```

Accuracy Score: 0.969047619047619



Best accuracy is 0.969047619047619 with  $k = 6$

Dengan menggunakan  $K = 6$  kita dapat mendapat hasil akurasi yang sedikit lebih baik. Kemudian dibawah ini kami merangkum hasil prediksi menggunakan confusion matrix dan divisualisasikan dengan heatmap.

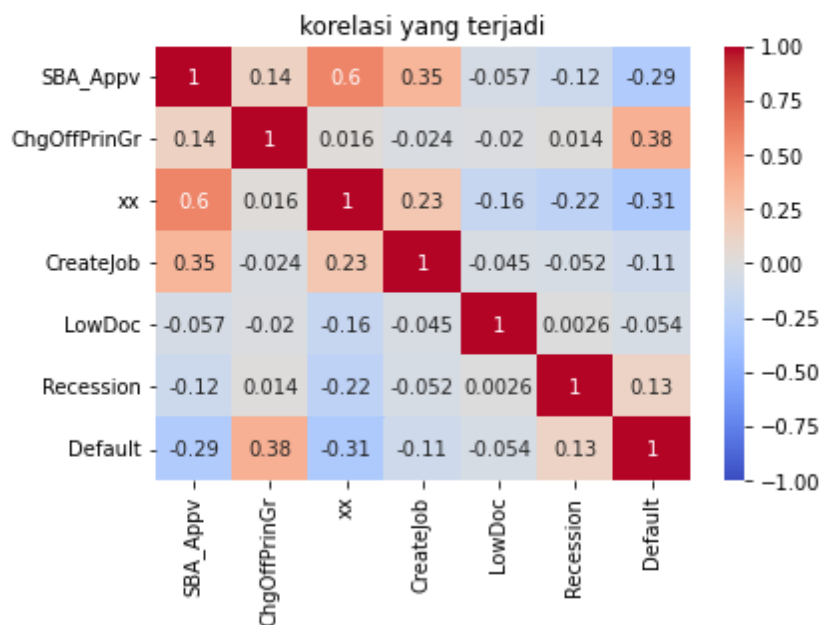


```
1 cm = confusion_matrix(y_test, prediction_KNN)
2 print("Confusion Matrix:\n", cm)
```

Confusion Matrix:

```
[[343  2]
 [ 14  61]]
```

```
1 # show heatmap
2 sns.heatmap(new_df.corr(), vmin=-1, vmax=1, cmap="coolwarm", annot=True)
3 plt.title('korelasi yang terjadi')
4 plt.show()
```



Terakhir, kami mencoba evaluasi menggunakan F1-Score dan simulasi sederhana dengan data dummy.

```
] 1 from sklearn.metrics import f1_score
2 print('F1-Score : {}'.format(f1_score(y_test, prediction)))
```

F1-Score : 0.9064748201438849

```
] 1 #dummy data 1
2 knn.predict([[4.164, 1.5, 21116, 30, 0, 0]])
```

array([0])

```
] 1 #dummy data 2
2 knn.predict([[2.120e+05, 1.209550e+06, 19250, 105, 1, 1]])
```

array([1])

## 2. Gaussian-Naive Bayes

Algoritma kedua yang kami pilih adalah Naive Bayes. Naive Bayes adalah salah satu algoritma dari supervised learning yang didasarkan oleh Bayes theorem. Algoritma ini digunakan untuk menyelesaikan masalah terkait pengklasifikasian dan cocok diterapkan pada dataset dengan variabel yang banyak / high-dimensional training dataset.

Naive Bayes sendiri terdiri dari beberapa jenis, yaitu Gaussian, Multinomial, dan Bernoulli. Algoritma Multinomial dan Bernoulli digunakan ketika datanya terdistribusi secara multinomial, sedangkan pada algoritma Gaussian data berdistribusi normal.

Langkah pertama yang harus dilakukan adalah mengimport library yang diperlukan. Kami menggunakan library yang sudah disediakan scikit learn, yaitu **KFold** untuk melakukan cross validation serta **GaussianNB** sebagai library untuk algoritma Naive Bayes.

```
] 1 from sklearn.naive_bayes import GaussianNB
   2 from sklearn.model_selection import KFold
```

Disini kami akan menggunakan metode KFold untuk cross validation. Untuk itu, kita harus membagi dataset ke dalam beberapa partisi agar data yang digunakan untuk training dan testingnya beragam. Pada kasus ini, kami membagi dataset ke dalam 5 partisi yang kemudian dilanjutkan dengan training dan testingnya.

```
1 k = 5
2 kf = KFold(n_splits=k, random_state=None)
```

```
1 GNB = GaussianNB()
2 GNB.fit(x_train, y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

```
1 for train_index, test_index in kf.split(x_scaled):
2     x_train, x_test = x_scaled.iloc[train_index, :], x_scaled.iloc[test_index, :]
3     y_train, y_test = y[train_index], y[test_index]
4
5 # Lakukan training dengan dataset train
6 pred_values = GNB.predict(x_test)
```

Berikut ini adalah hasil prediksi dari training dan testing dataset yang dilakukan pada langkah sebelumnya.

```
1 pred_values
```

```
array([0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0])
```

Dari langkah-langkah diatas, kami dapat menghasilkan model dengan tingkat akurasi dibawah ini.

```
1 accuracy_GNB = GNB.score(x_test, y_test)
2 accuracy_GNB
```

```
0.9833333333333333
```

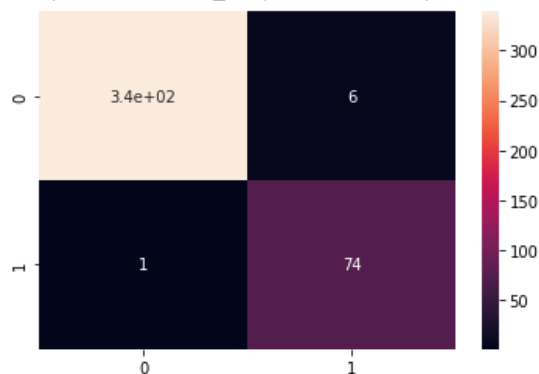
Setelah itu, kami merangkum hasil prediksi diatas ke dalam confusion matrix dan divisualisasikan menggunakan heatmap dari seaborn.

```
1 from sklearn.metrics import confusion_matrix
2 cf_matrix = confusion_matrix(y_test, pred_values_GNB)
3 cf_matrix
```

```
array([[339,  6],
       [ 1, 74]])
```

```
1 sns.heatmap(cf_matrix, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f509e9f3490>
```



Terakhir, kami mencoba evaluasi menggunakan F1-Score dan simulasi sederhana dengan data

dummy.

```
1 from sklearn.metrics import f1_score
2 print('F1-Score: {}'.format(f1_score(y_test, pred_values_GNB)))
```

F1-Score: 0.9548387096774195

```
1 #dummy data 1
2 GNB.predict([[2.3, 1.5, 22326, 2, 0, 0]])
```

array([0])

```
1 #dummy data 2
2 GNB.predict([[2.112e+04, 1.14320e+05, 21212, 90, 1, 0]])
```

array([1])

### 3. SVM

Berikutnya kami membuat model SVM untuk dataset ini. SVM (Support Vector Machine) adalah algoritma yang didasari oleh teori optimasi. Algoritma ini menggunakan *kernel trick*, dimana data akan ditransformasi dan mencari batasan optimal di antara kemungkinan output nya berdasarkan transformasi tersebut.

Metode SVM ini mempunyai beberapa jenis kernel namun pada problem ini kami akan mencoba 3 jenis kernel yaitu linear, radial, dan sigmoid. Berikut akan kami jabarkan tahapan untuk masing-masing kernel.

#### A. Kernel Linear

Langkah pertama adalah menyimpan function kernel yang akan digunakan pada variabel `svc_linear`. Setelah itu kita akan melakukan fitting dataset dengan variabel tersebut.

```
1 svc_linear = SVC(kernel='linear')
2 svc_linear.fit(x_train, y_train)
```

Setelah itu, hasil prediksi akan kita simpan pada variabel `yHat`. Dari hasil prediksi ini kita akan melihat akurasi dari training serta testing model SVM dengan kernel linear.

```
1 yHat = svc_linear.predict(x_test)
```

```
1 from sklearn import metrics
2 print("Training set accuracy = ", metrics.accuracy_score(y_train, svc_linear.predict(x_train)))
3 print("Test set accuracy = ", metrics.accuracy_score(y_test, yHat))
```

```
Training set accuracy = 0.6890606420927468
Test set accuracy = 0.8857142857142857
```

Hasil prediksi ini akan dirangkum dalam confusion matrix dan akan kita evaluasi menggunakan metrics F1-Score.

```
1 from sklearn.metrics import confusion_matrix
2 # Matriks confusion array using sklearn
3 cf_matrix = confusion_matrix(y_test, yHat)
4 cf_matrix
```

```
array([[340,  5],
       [ 43, 32]])
```

```
| 1 from sklearn.metrics import f1_score
  2 print('F1-Score : {}'.format(f1_score(y_test, yHat)))
```

```
F1-Score : 0.5714285714285715
```

Terakhir, data ini akan kita coba simulasikan dengan data dummy.

```
1 #dummy data 1
2 svc_linear.predict([[4.4212, 1.9, 16026, 20, 1, 0]])

array([0])
```

```
1 #dummy data 2
2 svc_linear.predict([[2.362e+04, 1.19550e+06, 20898, 120, 1, 1]])

array([1])
```

## B. Kernel Radial

Kernel kedua yang akan kita coba adalah Gaussian Radial Basis Function atau yang sering disebut RBF. Pertama-tama kita akan menyimpan function kernel yang akan dipakai pada variabel `svc_rbf` dan kemudian kita akan fitting data training nya.

```
1 svc_rbf = SVC(kernel='rbf')
2 svc_rbf.fit(x_train, y_train)
```

Setelah itu, prediksi dengan kernel rbf ini akan disimpan pada variabel `yHat`. Hasil akurasi dari training dan testing setnya dapat dilihat pada gambar dibawah.

```
1 yHat = svc_rbf.predict(x_test)
```

```
1 from sklearn import metrics
2 print("Training set accuracy = ", metrics.accuracy_score(y_train, svc_rbf.predict(x_train)))
3 print("Test set accuracy = ", metrics.accuracy_score(y_test, yHat))
```

```
Training set accuracy = 0.8781212841854935
Test set accuracy = 0.95
```

Setelah itu hasil prediksi ini akan dirangkum dalam confusion matrix dan akan kita evaluasi menggunakan metrics F1-Score.

```
1 from sklearn.metrics import confusion_matrix
2 # Matriks confusion array using sklearn
3 cf_matrix = confusion_matrix(y_test, yHat)
4 cf_matrix
```

```
array([[340,  5],
       [ 16, 59]])
```

```
1 from sklearn.metrics import f1_score
2 print('F1-Score : {}'.format(f1_score(y_test, yHat)))
```

```
F1-Score : 0.8489208633093526
```

Terakhir, data ini akan kita coba simulasikan dengan data dummy.

```
1 #dummy data 1
2 svc_rbf.predict([[4.4212, 1.9, 16026, 20, 1, 0]])

array([0])
```

```
1 #dummy data 2
2 svc_rbf.predict([[2.362e+04, 1.19550e+06, 20898, 120, 1, 1]])

array([0])
```

### C. Kernel Polynomial

Kernel ketiga yang akan kita coba adalah Polynomial. Pada sklearn, nama untuk kernel ini disingkat sebagai 'poly'. Untuk langkah pertamanya kita akan memasukan kernel yang digunakan pada variabel svc\_poly. Setelah itu training datasetnya akan kita fittingkan pada variabel tersebut.

```
1 svc_poly = SVC(kernel='poly')
2 svc_poly.fit(x_train, y_train)
```

Setelah itu, pada variabel yHat kita akan memasukan hasil prediksi dari dataset testing yang menggunakan kernel rbf ini. Kemudian nilai akurasi dari dataset training dan testing nya dapat terlihat pada gambar dibawah.

```
1 yHat = svc_poly.predict(x_test)
```

```
1 from sklearn import metrics
2 print("Training set accuracy = ", metrics.accuracy_score(y_train, svc_poly.predict(x_train)))
3 print("Test set accuracy = ", metrics.accuracy_score(y_test, yHat))
```

```
Training set accuracy = 0.7675386444708681
Test set accuracy = 0.9095238095238095
```

Setelah itu hasil prediksi ini akan dirangkum dalam confusion matrix dan akan kita evaluasi menggunakan metrics F1-Score.

```
1 from sklearn.metrics import confusion_matrix
2 # Matriks confusion array using sklearn
3 cf_matrix = confusion_matrix(y_test, yHat)
4 cf_matrix
```

```
array([[345,  0],
       [ 38, 37]])
```

```
1 from sklearn.metrics import f1_score
2 print('F1-Score : {}'.format(f1_score(y_test, yHat)))
```

```
F1-Score : 0.6607142857142857
```

Terakhir, data ini akan kita coba simulasikan dengan data dummy.

```
1 #dummy data 1
2 svc_poly.predict([[4.4212, 1.9, 16026, 20, 1, 0]])

array([0])
```

```
| 1 #dummy data 2
  2 svc_poly.predict([[2.362e+04, 1.19550e+06, 20898, 120, 1, 1]])

array([1])
```

Setelah mencoba ketiga kernel diatas, dapat disimpulkan bahwa menggunakan kernel RBF akan menghasilkan akurasi terbaik yaitu 0.95. Terbukti juga dari F1-Score nya kernel RBF memberikan nilai 0.84892086 yang termasuk cukup baik.



## EVALUATION

Berdasarkan dari 3 model diatas bisa kita simpulkan berdasarkan perbandingan hasil akurasinya yaitu,

1. Naive Bayes : 98.3%
2. KNN : 96.9%
3. SVM : 95%

dan untuk F1-Score setiap masing-masing model yaitu,

1. Naive Bayes : 95.4%
2. KNN : 90.6%
3. SVM : 84.8%

Dari 3 model tersebut kami dapat menyimpulkan bahwa 6 dari 10 variable teratas yang kami pilih untuk dianalisa, apakah variabel tersebut memiliki pengaruh terhadap persetujuan bisnis kecil tersebut akan mendapatkan pinjaman atau tidak dari bank. Hasil evaluasi setiap model yang kami gunakan menunjukan F1-Score diatas memiliki hasil yang baik.

## FEEDBACK:

Kekurangan:

1. Penjelasan variabel yang tidak begitu jelas.
2. Penulisan data yang tidak tepat dan juga ada yang kosong.
3. Waktu pengerjaan yang cukup singkat.
4. Pengerjaan yang tidak beratur.

Penyelesaian :

1. Browsing di internet terkait dataset yang serupa.
2. Melakukan pengecekan berulang kali secara lebih detail.
3. Pembagian dan penjadwalan tugas kepada kelompok.
4. Double check dan revisi beberapa kali.

Untuk kedepannya kami berharap memiliki waktu yang lebih lama setidaknya 14 hari untuk pengerjaan. Kami yang awalnya masih belum mengerti betul python dan pada akhirnya kami bisa menyelesaikan proyek ini merupakan sebuah pencapaian yang jujur kami bilang 'MANTAB'

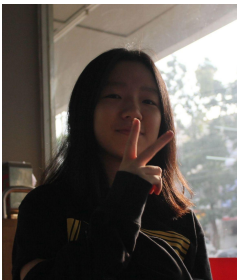
## TEAM PROFILES



Nama : Jonathan Kevin Marsiano  
NIM : 0706011910025  
Universitas : Ciputra University

Job Desc:

1. Model Naive Bayes
2. Evaluation
3. Business Understanding



Nama : Jocelyn Michelle Kho  
NIM : 2440039796  
Universitas : Bina Nusantara

Job Desc:

1. Data Preparation
2. Business Understanding



Nama : I Wayan Bayu Adnyana  
NIM : 0706021910019  
Universitas : Ciputra University

Job Desc:

1. Model SVM
2. Data Understanding



Nama : Arief Riski Indra Pratama  
NIM : 1810631170090  
Universitas : Universitas Singaperbangsa Karawang

Job Desc:

1. Data Understanding
2. Model KNN