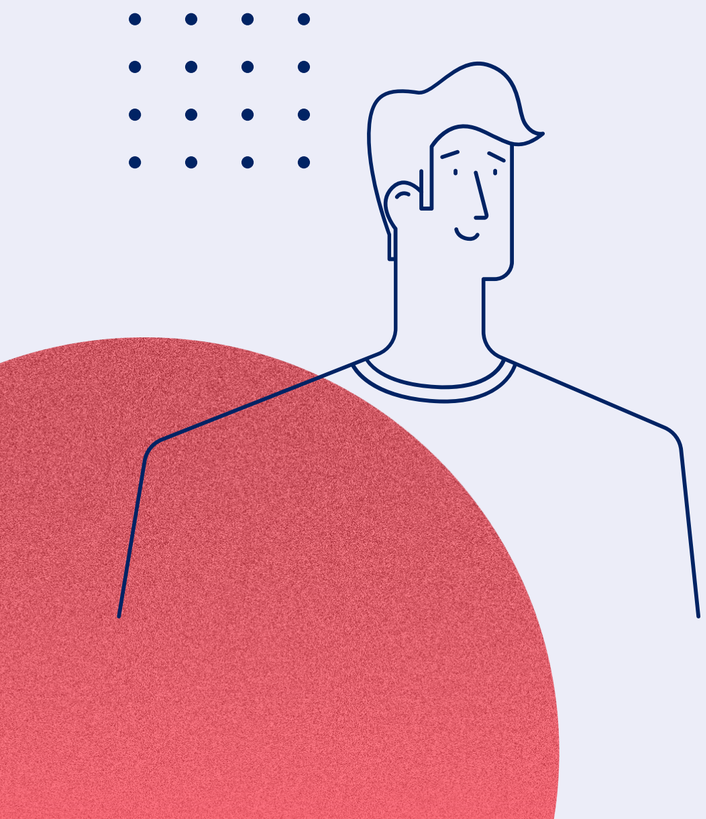
CLASSIFICATION

CUSTOMER CHURN PADA PROVIDER

- Melakukan klasifikasi pada data train
- Data Customer yang menggunakan Provider
- Terdiri dari 20 Kolom dan 4250 Record
- Data dependence yaitu kolom churn

PREPROCESSING

MISSING VALUE & DUPLICATED



```
df.isna().sum()
```

```
state                           0
account_length                  0
area_code                       0
international_plan              0
voice_mail_plan                 0
number_vmail_messages           0
total_day_minutes               0
total_day_calls                 0
total_day_charge                0
total_eve_minutes               0
total_eve_calls                 0
total_eve_charge                0
total_night_minutes             0
total_night_calls               0
total_night_charge              0
total_intl_minutes              0
total_intl_calls                0
total_intl_charge               0
number_customer_service_calls   0
churn                           0
dtype: int64
```

Tidak ada Missing value pada data train

```
df.duplicated().sum()
```

```
0
```

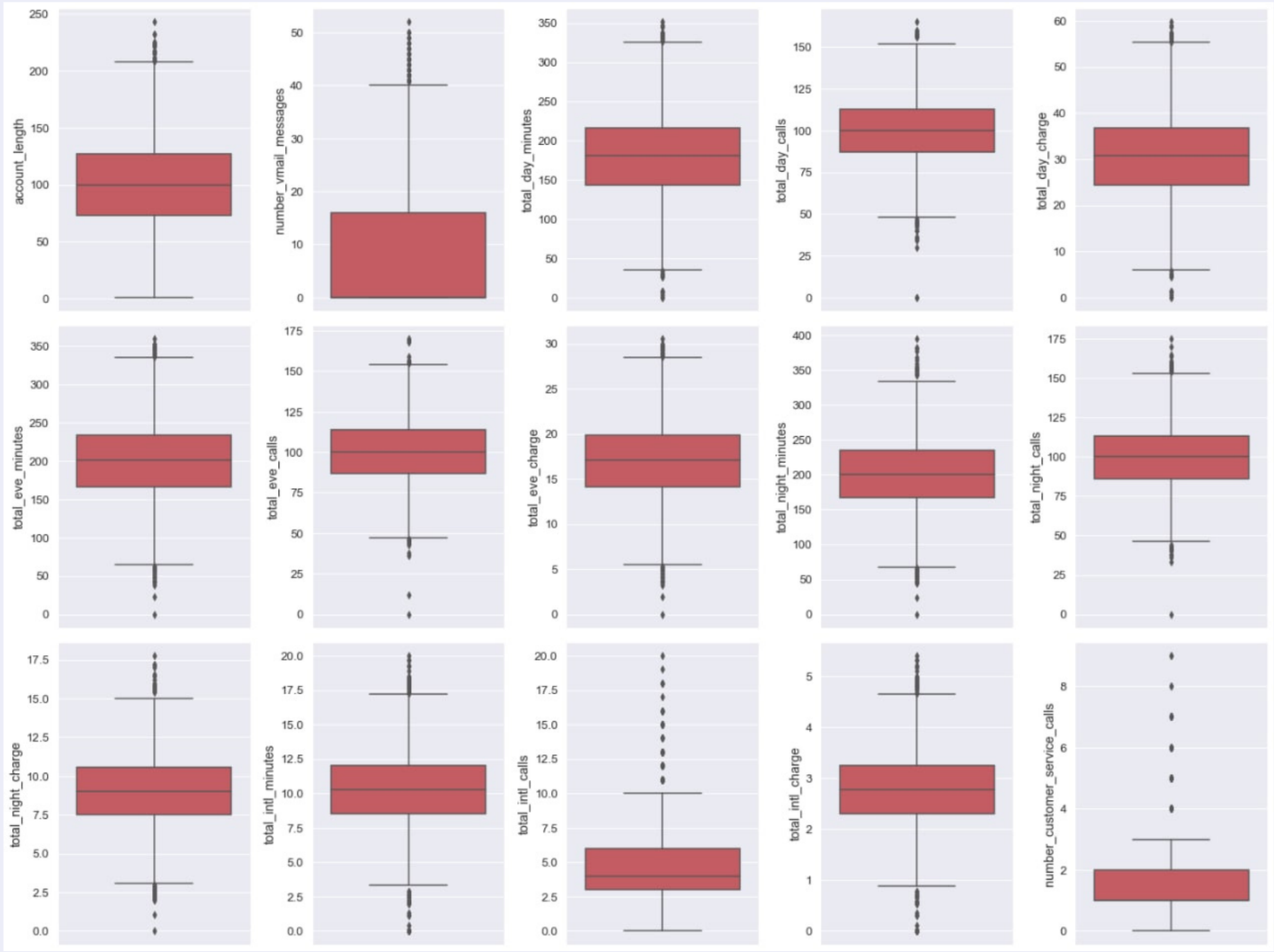
Missing Value pada setiap kolom tidak ada

Tidak ada Duplikasi data

PREPROCESSING

OUTLIER

Cek setiap outlier pada data numerik



PREPROCESSING

OUTLIER

Filtering data yang outlier dengan batas interquartile dari setiap kolom

```
outliers = [] #untuk liat outliernya, yg outlier disimpen disini
fill = np.array([True]*len(df)) #untuk filtering dataframe train yg outlier dengan bool
#cari nilai interquartile
for col in numerik:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    low_limit = Q1 - (IQR * 1.5)
    high_limit = Q3 + (IQR * 1.5)

    outliers = df[(df[col] < low_limit) | (df[col] > high_limit)]
    fill = ((df[col] >= low_limit) & (df[col] <= high_limit)) & fill
```

```
df2 = df[fill]#filtering outlier
```

```
(4250, 20)
(3515, 20)
```

```
print("Data yang dibuang : %.2f%%" % ((df2.shape[0]-df.shape[0])/df.shape[0]*100))
```

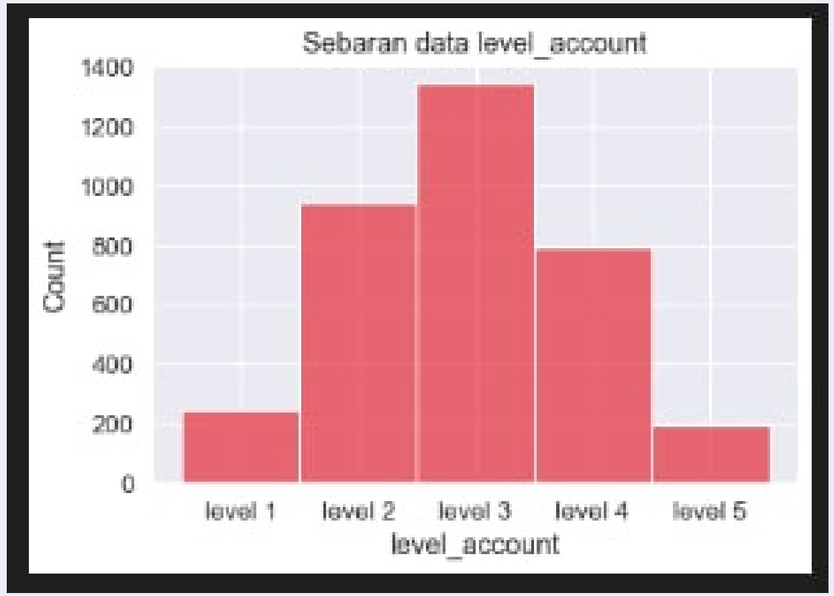
```
Data yang dibuang : -17.29%
```

PREPROCESSING

BINNING DATA

```
lvl = np.linspace(min(df2['account_length']), max(df2['account_length']), 6)
lvl_acc = ['level 1', 'level 2', 'level 3', 'level 4', 'level 5']
df2['level_account'] = pd.cut(df2['account_length'], lvl, labels=lvl_acc, include_lowest=True)
#membagi account_length menjadi 5 kategori level dengan 6 pembatas
```

Mengkategori kolom account_length



PREPROCESSING

FEATURE ENGINE

```
Tot_min = (df2['total_day_minutes'] + df2['total_eve_minutes'] + df2['total_night_minutes'] + df2['total_intl_minutes'])
Tot_call = (df2['total_day_calls'] + df2['total_eve_calls'] + df2['total_night_calls'] + df2['total_intl_calls'])
Tot_charge = (df2['total_day_charge'] + df2['total_eve_charge'] + df2['total_night_charge'] + df2['total_intl_charge'])
#feature baru dari total minute call dan charge pada setiap waktunya
```

```
avg_call_charge = Tot_charge/Tot_call
avg_call_min = Tot_min/Tot_call
#feature dari rata2 setiap charge dan minute dari total telpon
```

Membuat kolom baru dari setiap total minute, call dan charge kemudian, membuat rata-rata charge dan minute

level_account	Total_minute	Total_call	Total_charge	avg_call_charge	avg_call_minute
level 3	625.2	332	59.24	0.178434	1.883133
level 4	539.4	333	62.29	0.187057	1.619820
level 2	512.0	359	52.09	0.145097	1.426184
level 4	479.0	275	46.90	0.170545	1.741818
level 4	818.2	297	80.54	0.271178	2.754882

PREPROCESSING

NORMALISASI

```
kategori = ['state','area_code','international_plan','voice_mail_plan', 'churn', 'level_account']
numerik = ['number_vmail_messages','number_customer_service_calls',
           'Total_minute','Total_call','Total_charge', 'avg_call_minute', 'avg_call_charge']
#bagi kategori dan numerik column untuk cleaning dll
```

```
for col in numerik:
    df3[col] = MinMaxScaler().fit_transform(df2[col].values.reshape(len(df2), 1)) #normalisasi data n
```

Normalisasi dengan MinMaxScaler pada data numerik

```
df3.describe()
```

	number_vmail_messages	number_customer_service_calls	Total_minute	Total_call	Total_charge	avg_call_charge	avg_call_minute
count	3515.000000	3515.000000	3515.000000	3515.000000	3515.000000	3515.000000	3515.000000
mean	0.173492	0.434519	0.514090	0.518319	0.516806	0.374938	0.400457
std	0.313365	0.325224	0.143626	0.155066	0.160466	0.121048	0.125020
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.333333	0.418465	0.412844	0.405392	0.293170	0.314648
50%	0.000000	0.333333	0.513631	0.518349	0.518028	0.364663	0.391530
75%	0.000000	0.666667	0.609466	0.623853	0.622208	0.449836	0.477054
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

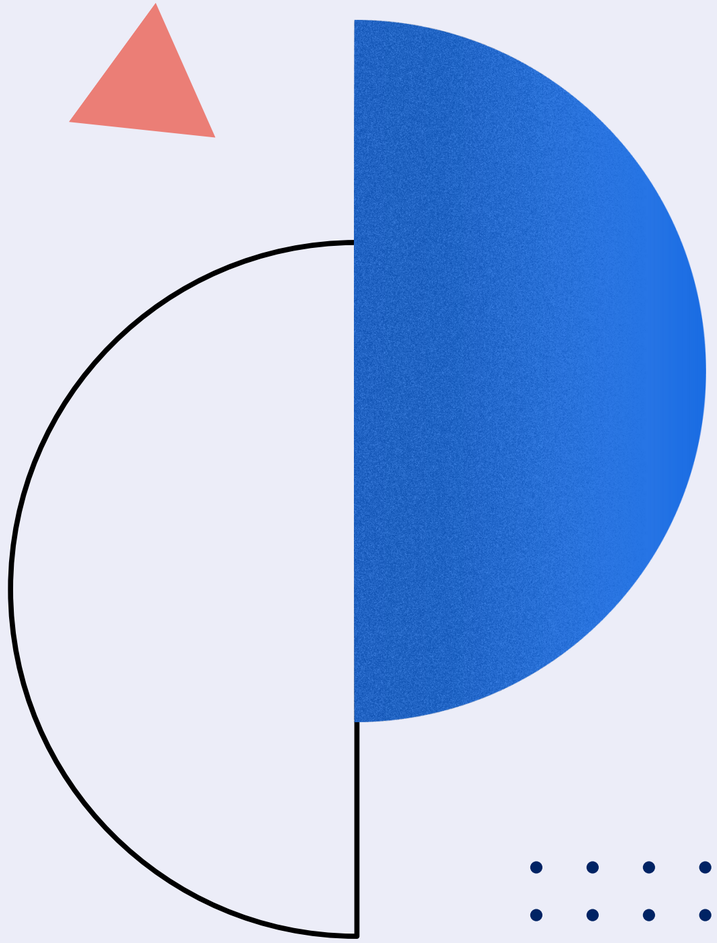
PREPROCESSING

FEATURE ENCODING

Mengubah data kotegori dengan LabelEncoder

```
#label encoder
df4 = df3.copy()
lbl_encoder = preprocessing.LabelEncoder()
for cat in kategori:
    df4[cat] = lbl_encoder.fit_transform(df4[cat])
df4.head()
#transforming pada data bentuk kategorik agar bisa dolah dengan
```

	state	area_code	international_plan	voice_mail_plan	number_vmail_m
0	35	1	0	1	
1	31	1	0	0	
3	36	1	1	0	
5	24	1	1	0	
7	49	1	1	1	



CORELATION

Melihat hubungan dari setiap feature terhadap target

```
df4.corr().iloc[6].sort_values(ascending=False)
#nilai korelasi palingtinggi dengan target
```

```
churn                           1.000000
Total_charge                    0.345670
Total_minute                    0.300893
avg_call_charge                 0.293355
international_plan              0.279414
avg_call_minute                 0.245193
state                           0.012978
area_code                       0.005712
level_account                   0.002612
number_customer_service_calls  -0.004485
Total_call                     -0.014945
number_vmail_messages          -0.105567
voice_mail_plan                -0.115205
Name: churn, dtype: float64
```

FEATURE SELECTION

```
#jika manual select, berdasarkan korelasi yg kuat
features_selec = ['Total_charge','Total_minute','avg_call_charge','international_plan']
Y = df4['churn']
X_manual = df4[features_selec]
```

Memilih feature yang paling berhubungan kuat dengan target

SPLIT DATA

```
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X_manual_res, Y_res, test_size=test_sz, random_state=seed)
```

Membagi data train dan data test dengan perbandingan 80:20

BALANCING

```
Y.value_counts()
```

```
0    3147
1     368
```

→

```
Y_res.value_co
```

```
0    3147
1    1573
```

```
over = SMOTE(sampling_strategy=0.3, random_state=seed) #smote untuk sampling
under = RandomOverSampler(sampling_strategy=0.5, random_state=seed) #RandomO
stp = [('o', over),('u', under)]
pipe = Pipeline(steps=stp)  #satuin functionnya dengan pipeline
```

```
X_manual_res, Y_res = pipe.fit_resample(X_manual, Y) #fiting sample dengan p
```

MODELING

```
deci = DecisionTreeClassifier(random_state=seed)
deci.fit(X_train, Y_train)
y_pred = deci.predict(X_test)
#penerapan algoritma DecisionteeClassifier pada  dat
```

DecisionTreeClassifier

SupportVectorMachine Classifier

```
svc = SVC(random_state=seed)
svc.fit(X_train, Y_train)
y_pred2 = svc.predict(X_test)
#penerapan algoritma SVM Classicfier pad
```

EVALUASI

DECISION TREE CLASSIFIER

```
Accuracy: 90.678%
Recall: 90.064%
Precission: 83.136%
F1-Score: 86.462%
ROC-AUC: 90.523%
```

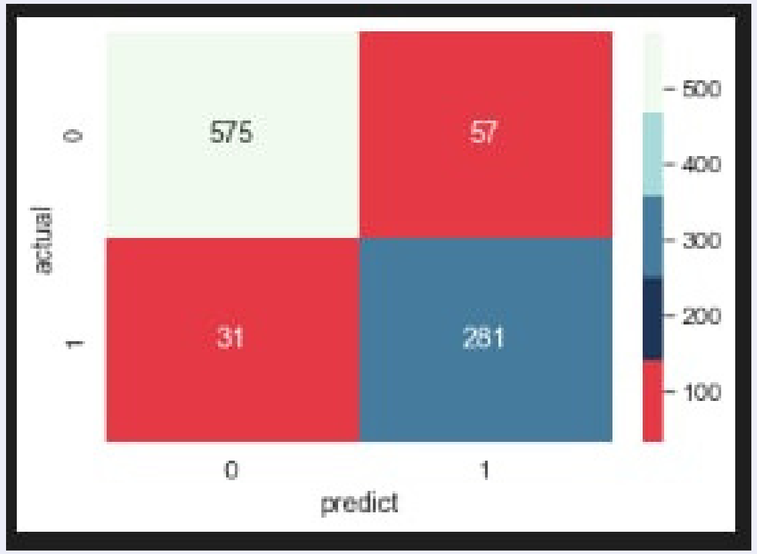
K-fold

```
Accuracy: 91.483%(1.398%)
```

Classification Report

	precision	recall	f1-score	support
0	0.95	0.91	0.93	632
1	0.83	0.90	0.86	312
accuracy			0.91	944
macro avg	0.89	0.91	0.90	944
weighted avg	0.91	0.91	0.91	944

ConfusionMatrix



EVALUASI

SUPPORT VECTOR MACHINE

```
Accuracy: 88.136%
Recall: 84.615%
Precission: 80.488%
F1-Score: 82.500%
ROC-AUC: 87.244%
```

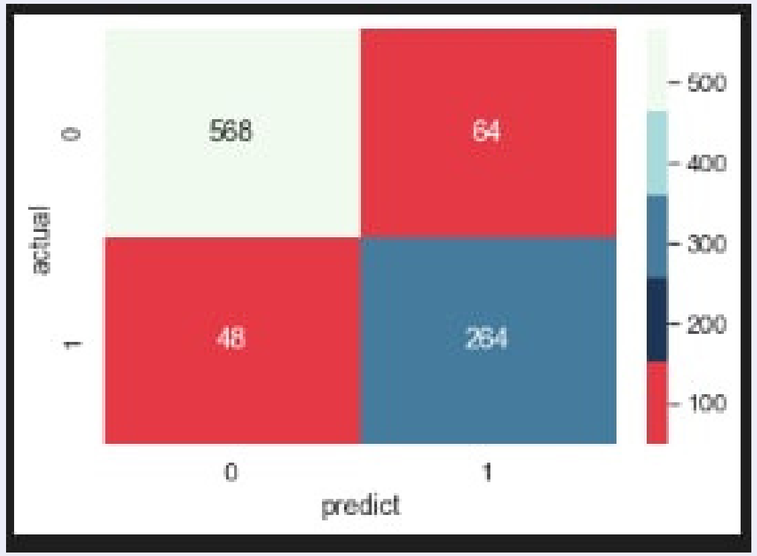
K-fold

```
Accuracy: 86.843%(1.687%)
```

Classification Report

	precision	recall	f1-score	support
0	0.92	0.90	0.91	632
1	0.80	0.85	0.83	312
accuracy			0.88	944
macro avg	0.86	0.87	0.87	944
weighted avg	0.88	0.88	0.88	944

ConfusionMatrix



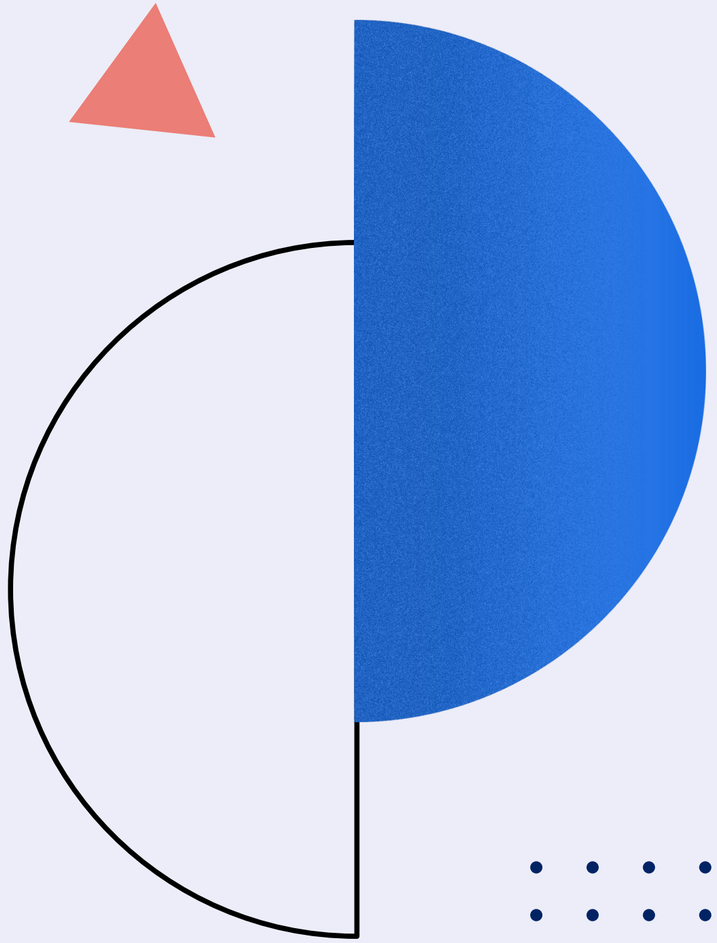
HYPERTUNING

Hasil GridSearchCV DecisionTree dengan hasil
Max_depth 25

```
Accuracy after Tuning: 90.678%
Recall after Tuning: 90.064%
Precission after Tuning: 83.136%
F1-Score after Tuning: 86.462%
ROC-AUC after Tuning: 90.523%
```

Hasil GridSearchCV SVM dengan hasil
C 100 dan gamma 1

```
Accuracy after Tuning: 88.242%
Recall after Tuning: 84.615%
Precission after Tuning: 80.734%
F1-Score after Tuning: 82.629%
ROC-AUC after Tuning: 87.324%
```



HASIL PREDIKSI

DecisionTreeClassifier

```
dt_test_prediction["chrun_predict"]

0    625
1    125
Name: chrun_predict, dtype: int64
```

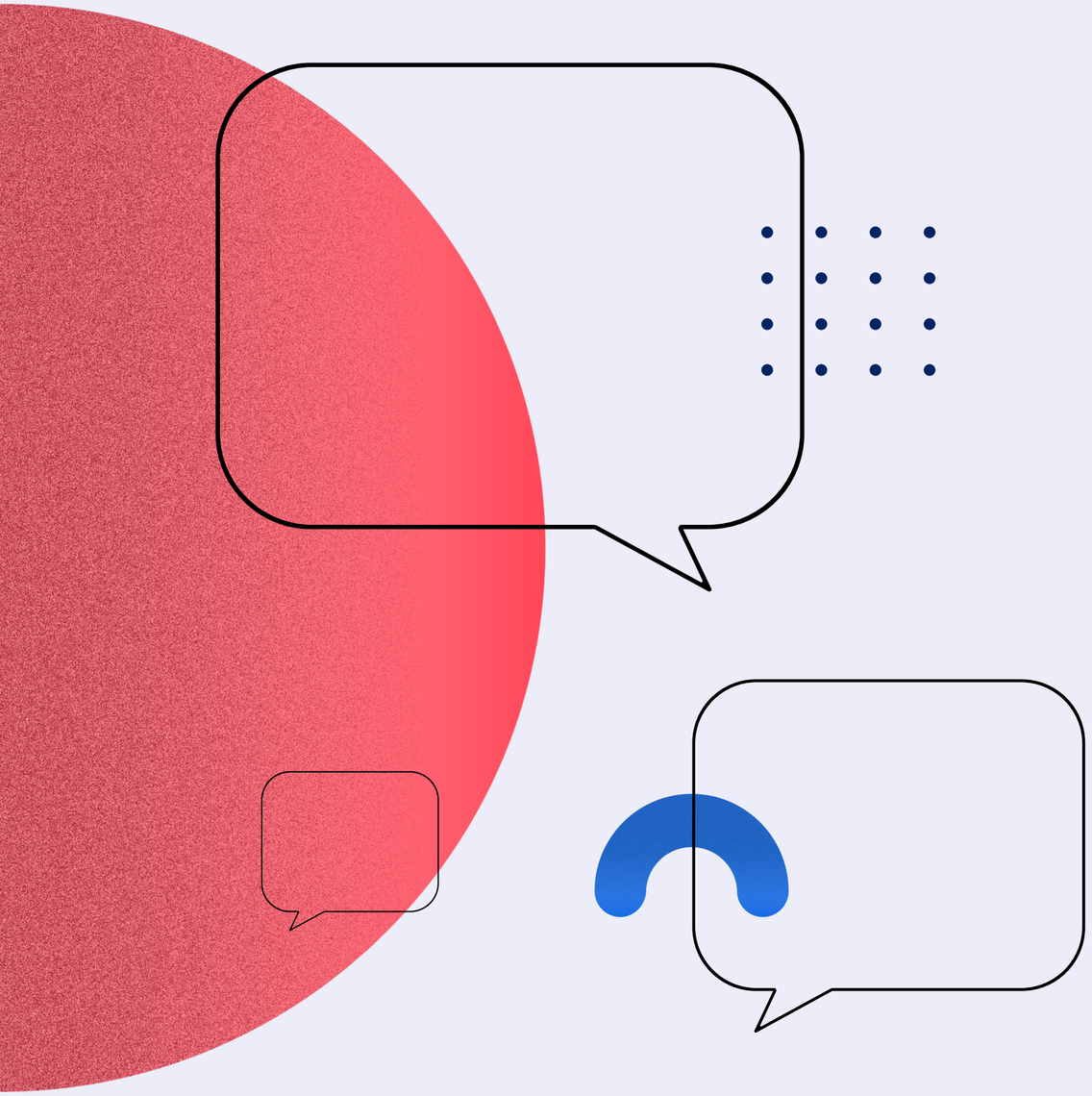
SupportVectorMachine

```
dt_test_prediction2["chrun_predict"]

0    529
1    221
Name: chrun_predict, dtype: int64
```

Model SVM memprediksi data negative (terjadi chrun) lebih bnyak dari Model Deci dalam memprediksi data negative(terjadi churn),

Model Deci memprediksi data positive (tidak terjadi churn) lebih banyak dari Model SVM dalam memprediksi data positive(tidak terjadi churn)



KESIMPULAN

- Hasil akurasi dari DecissionTree lebih baik dari SVM
- Recall pada kedua Algoritma lebih tinggi dari Precisionnya
- Hasil predict pada test.csv di DecissionTree nilai 0 lebih banyak dari pada SVM, sedangkan pada SVM nilai 1 lebih banyak dari DecissionTree
- Menambahkan feature dengan korelasi >= 0.2 membuat hasil model tidak dapat dipercaya
- Hypertuning cukup diterapkan di SVM untuk menaikkan nilai evaluasinya

“

OUTLIER MAKE YOUR DATA MAD
OVERTHINK MAKE YOURSELF MINDER

Thank You by Zain