BINAR

DS2-6 - Final Presentation

Data Science

Anggota Tim

Kelas	TIM	NAMA
DS 2	**DS2-6**	ROBBY SUTIAWAN
		MUHAMMAD RIZKI FADHILLA

Business Understanding

Problem Statement

Perkembangan industri telekomunikasi sekarang sangat cepat. Hal ini dapat dilihat dari perilaku masyarakat yang menggunakan internet dalam berkomunikasi. Perilaku ini menyebabkan banyaknya perusahaan telekomunikasi dan meningkatnya internet service provider yang dapat menimbulkan persaingan antar provider.

Pelanggan memiliki hak dalam memilih provider yang sesuai dan dapat beralih dari provider sebelumnya yang diartikan sebagai Customer Churn. Peralihan ini dapat menyebabkan berkurangnya pendapatan bagi perusahaan telekomunikasi sehingga penting untuk ditangani.

Tujuan Analisis

Melakukan analisis klasifikasi data customer dengan bantuan Google Colab dalam memprediksi peluang customer yang akan beralih ke provider lain (churn) dengan memanfaatkan data historis yang berhubungan dengan perilaku customer.

Data Understanding

Deskripsi Data

Dataset yang digunakan adalah data customer churn yang bersumber dari laman Kaggle. Dataset ini terdiri dari 2 dataset yaitu data training dan data testing. Data training terdiri 4250 sampel. Setiap sampel terdiri 19 features dan 1 variabel boolean "churn" sebagai variabel target. Sedangkan data testing memiliki feature yang sama dengan data training namun tidak memiliki variabel target dan hanya terdiri 750 sampel. Adapun deskripsi feature tersebut yaitu:

Features	Tipe Data	Deskripsi
state	string	Kode 2 huruf negara bagian USA
account_length	numerical	Lama bulan pemakaian
area_code	string	Kode Area
international_plan	(yes/no)	Customer memiliki international plan
voice_mail_plan	(yes/no)	Customer memiliki voice mail plan
number_vmail_messages	numerical	Jumlah pesan voice-mail
total_day_minutes	numerical	Total menit panggilan siang hari
total_day_calls	numerical	Jumlah panggilan siang hari
total_day_charge	numerical	Total biaya panggilan siang hari
total_eve_minutes	numerical	Total menit panggilan sore hari
total_eve_calls	numerical	Jumlah panggilan sore hari
total_eve_charge	numerical	Total biaya panggilan siang hari
total_night_minutes	numerical	Total menit panggilan malam hari
total_night_calls	numerical	Total panggilan malam hari
total_night_charge	numerical	Total biaya panggilan malam hari
total_intl_minutes	numerical	Total menit panggilan internasional
total_intl_calls	numerical	Jumlah panggilan internasional
total_intl_charge	numerical	Total biaya panggilan internasional
number_customer_service_calls	numerical	jumlah panggilan ke customer service
churn	(yes/no)	Target variable

BINAR

Library yang digunakan

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
sns.set_style('whitegrid')
from google.colab import files

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

BINAR

Import Dataset Training

```
from google.colab import files
uploaded = files.upload()
```

Choose Files Data Train.csv

- **Data Train.csv**(text/csv) - 387621 bytes, last modified: 3/12/2024 - 100% done

```
Saving Data Train.csv to Data Train.csv
```

```
df = pd.read_csv("Data Train.csv")
```

```
df.head()
```

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes
0	OH	107	area_code_415	no	yes	26	161.6
1	NJ	137	area_code_415	no	no	0	243.4
2	OH	84	area_code_408	yes	no	0	299.4
3	OK	75	area_code_415	yes	no	0	166.7
4	MA	121	area_code_510	no	yes	24	218.2

BINAR

Cek Tipe Data dan Kolom

```
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4250 entries, 0 to 4249
Data columns (total 20 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   state                           4250 non-null   object 
 1   account_length                  4250 non-null   int64  
 2   area_code                       4250 non-null   object 
 3   international_plan              4250 non-null   object 
 4   voice_mail_plan                 4250 non-null   object 
 5   number_vmail_messages           4250 non-null   int64  
 6   total_day_minutes               4250 non-null   float64
 7   total_day_calls                 4250 non-null   int64  
 8   total_day_charge                4250 non-null   float64
 9   total_eve_minutes               4250 non-null   float64
 10  total_eve_calls                 4250 non-null   int64  
 11  total_eve_charge                4250 non-null   float64
 12  total_night_minutes             4250 non-null   float64
 13  total_night_calls               4250 non-null   int64  
 14  total_night_charge              4250 non-null   float64
 15  total_intl_minutes              4250 non-null   float64
 16  total_intl_calls                4250 non-null   int64  
 17  total_intl_charge               4250 non-null   float64
 18  number_customer_service_calls   4250 non-null   int64  
 19  churn                           4250 non-null   object 
dtypes: float64(8), int64(7), object(5)
memory usage: 664.2+ KB
```

Dari informasi di samping dapat dilihat:

- Ukuran data adalah (4250, 20)
- Seluruh variabel (kolom) tidak memiliki missing value
- Terdapat 5 variabel yang bertipe kategori, dan 15 numerik

BINAR

Statistik Data Numerik

```
df_train.describe().T
```

	count	mean	std	min	25%	50%	75%	max
account_length	4250.0	100.236235	39.698401	1.0	73.0000	100.00	127.0000	243.00
number_vmail_messages	4250.0	7.631765	13.439882	0.0	0.0000	0.00	16.0000	52.00
total_day_minutes	4250.0	180.259600	54.012373	0.0	143.3250	180.45	216.2000	351.50
total_day_calls	4250.0	99.907294	19.850817	0.0	87.0000	100.00	113.0000	165.00
total_day_charge	4250.0	30.644682	9.182096	0.0	24.3650	30.68	36.7500	59.76
total_eve_minutes	4250.0	200.173906	50.249518	0.0	165.9250	200.70	233.7750	359.30
total_eve_calls	4250.0	100.176471	19.908591	0.0	87.0000	100.00	114.0000	170.00
total_eve_charge	4250.0	17.015012	4.271212	0.0	14.1025	17.06	19.8675	30.54
total_night_minutes	4250.0	200.527882	50.353548	0.0	167.2250	200.45	234.7000	395.00
total_night_calls	4250.0	99.839529	20.093220	0.0	86.0000	100.00	113.0000	175.00
total_night_charge	4250.0	9.023892	2.265922	0.0	7.5225	9.02	10.5600	17.77
total_intl_minutes	4250.0	10.256071	2.760102	0.0	8.5000	10.30	12.0000	20.00
total_intl_calls	4250.0	4.426353	2.463069	0.0	3.0000	4.00	6.0000	20.00
total_intl_charge	4250.0	2.769654	0.745204	0.0	2.3000	2.78	3.2400	5.40
number_customer_service_calls	4250.0	1.559059	1.311434	0.0	1.0000	1.00	2.0000	9.00

Statistik Data Kategorik

```
[ ] df_train.describe(include='O').T
```

	count	unique	top	freq
state	4250	51	WV	139
area_code	4250	3	area_code_415	2108
international_plan	4250	2	no	3854
voice_mail_plan	4250	2	no	3138
churn	4250	2	no	3652

BINAR

Import Dataset Testing

```
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available whe
this cell to enable.
Saving Data Test.csv to Data Test.csv

```
df_test = pd.read_csv('Data Test.csv')
df_test.head()
```

	id	state	account_length	area_code	international_plan	voice_mail_plan	number_vma
0	1	KS	128	area_code_415	no	yes	
1	2	AL	118	area_code_510	yes	no	
2	3	IA	62	area_code_415	no	no	
3	4	VT	93	area_code_510	no	no	
4	5	NE	174	area_code_415	no	no	

BINAR

```
[ ] # membuang kolom id karena merupakan index dan tidak memiliki arti
    df_test = df_test.drop('id', axis=1)
```

```
df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 750 entries, 0 to 749
Data columns (total 19 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   state                         750 non-null    object
 1   account_length                750 non-null    int64
 2   area_code                     750 non-null    object
 3   international_plan            750 non-null    object
 4   voice_mail_plan               750 non-null    object
 5   number_vmail_messages         750 non-null    int64
 6   total_day_minutes             750 non-null    float64
 7   total_day_calls               750 non-null    int64
 8   total_day_charge              750 non-null    float64
 9   total_eve_minutes             750 non-null    float64
 10  total_eve_calls               750 non-null    int64
 11  total_eve_charge              750 non-null    float64
 12  total_night_minutes           750 non-null    float64
 13  total_night_calls             750 non-null    int64
 14  total_night_charge            750 non-null    float64
 15  total_intl_minutes            750 non-null    float64
 16  total_intl_calls              750 non-null    int64
 17  total_intl_charge             750 non-null    float64
 18  number_customer_service_calls  750 non-null    int64
dtypes: float64(8), int64(7), object(4)
memory usage: 111.5+ KB
```

Dari informasi di samping dapat dilihat:

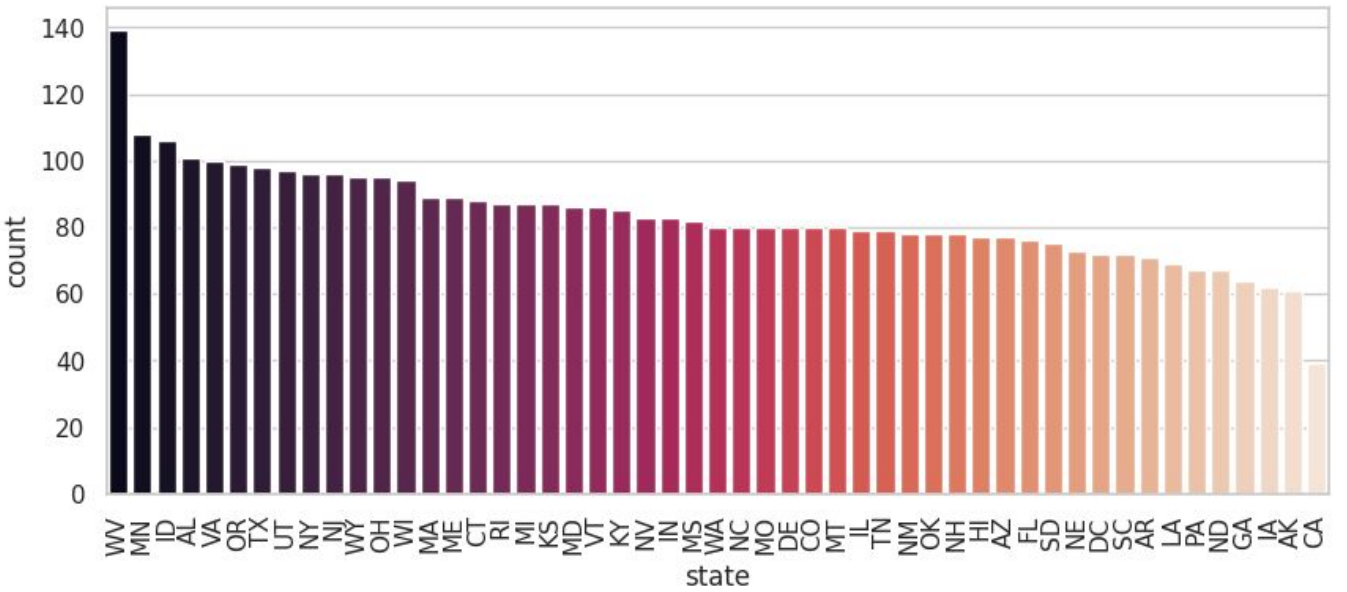
- Ukuran data testing adalah (750, 19)
- Seluruh variabel (kolom) tidak memiliki missing value
- Terdapat 4 variabel yang bertipe kategori, dan 15 numerik

Exploratory Data Analysis

Pada exploratory data analisis, analisis hanya dilakukan pada data training

Data Categorical

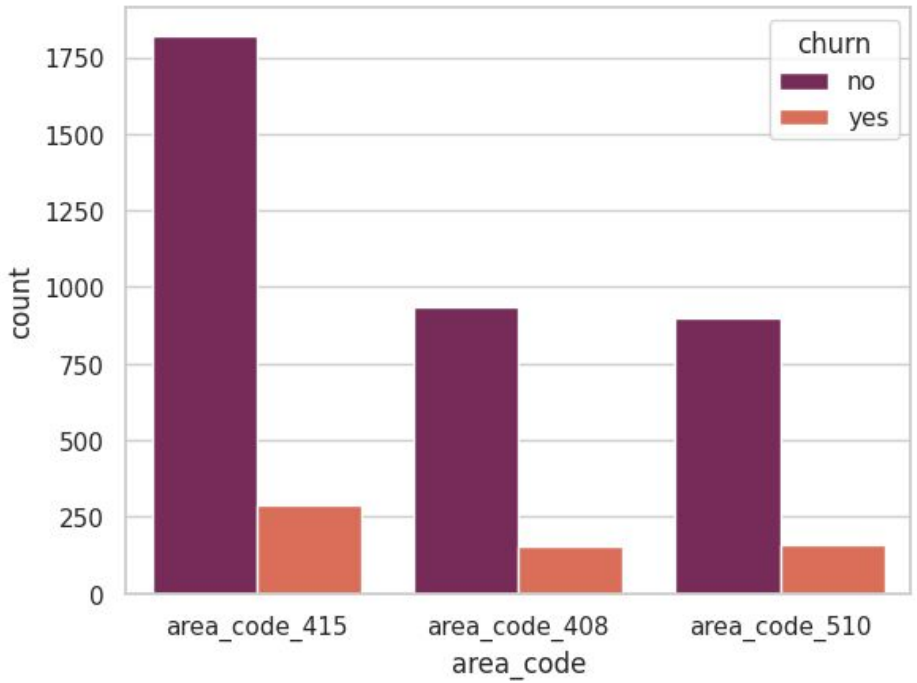
Persebaran Customer berdasar State



Dari distribusi customer berdasarkan State diketahui bahwa customer paling banyak tinggal di State Virginia Barat (WV). Sedangkan State yang paling sedikit ditinggali oleh customer adalah California (CA)

Data Categorical

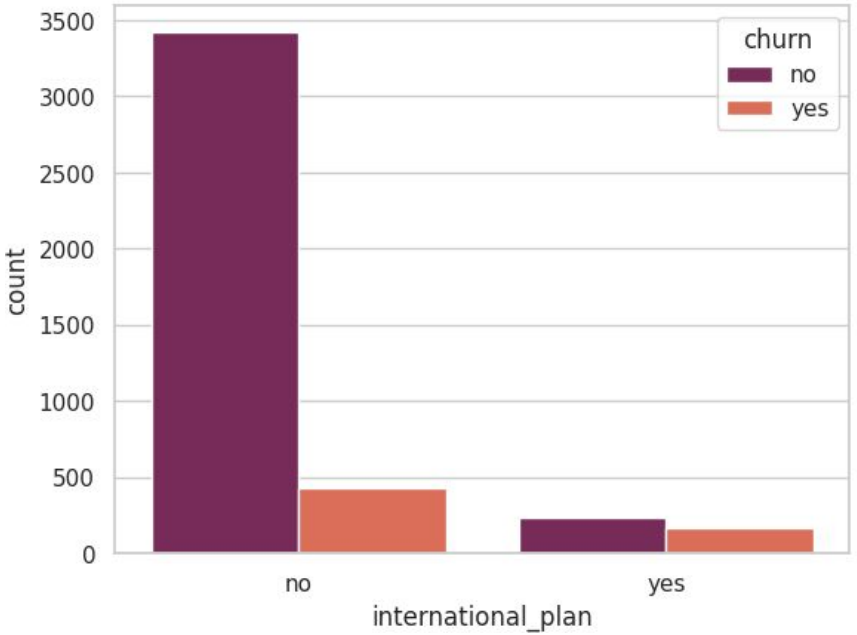
Persebaran customer berdasar kode area



Hampir separuh customer provider tinggal di area code 415

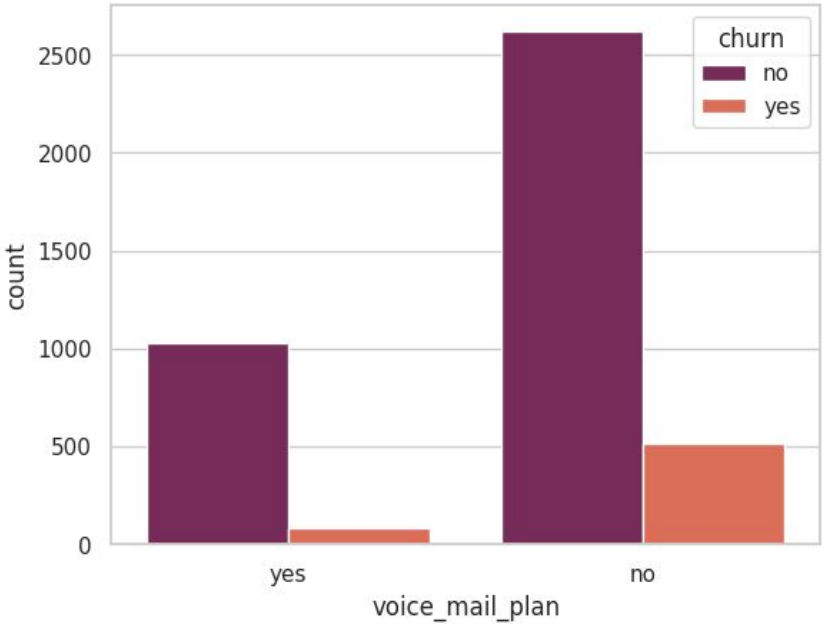
Data Categorical

Customer yang memiliki international plan



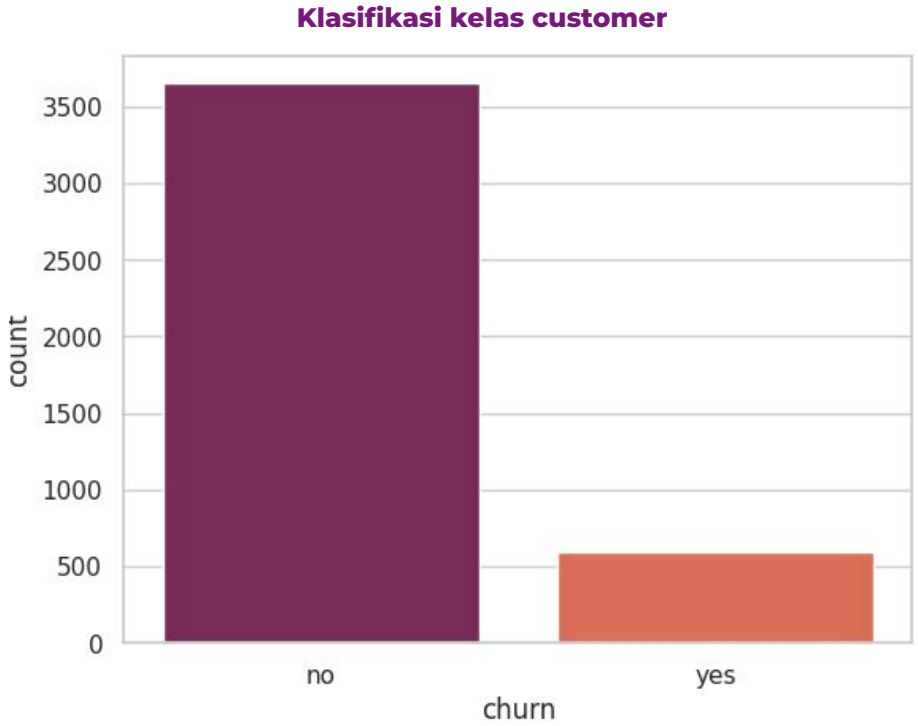
customer yang beralih dari provider ini (churn) hampir separuhnya memiliki international plan.

Customer yang memiliki voice mail plan



Sebagian besar customer tidak memiliki voice mail plan

Data Categorical

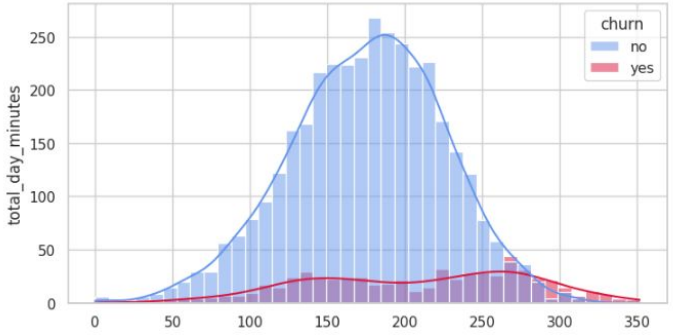


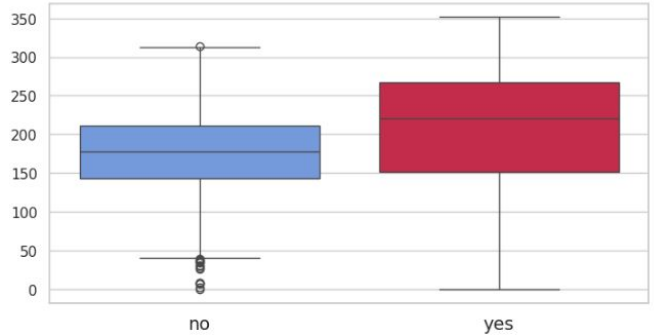
Pada variable target, class customer yang tidak churn jauh lebih banyak dibandingkan dengan customer yang churn. Karena ketimpangan ini, maka data ini disebut data imbalance.

Exploratory Data Analysis

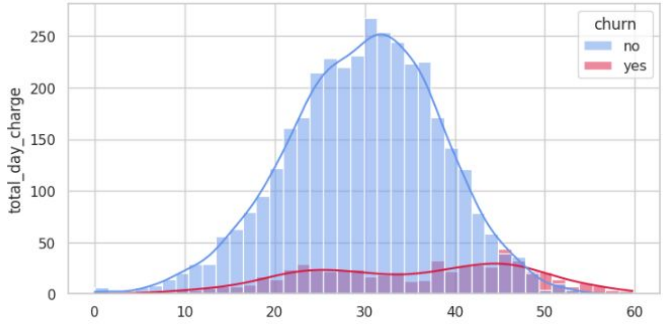
Data Numerical

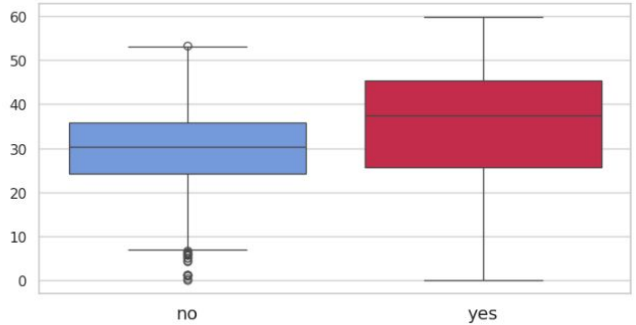
Distribusi total durasi panggilan (menit) di siang hari





Distribusi total biaya panggilan di siang hari



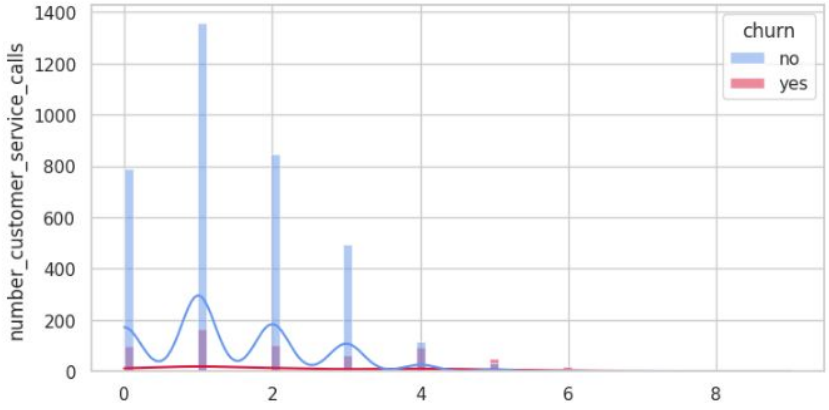


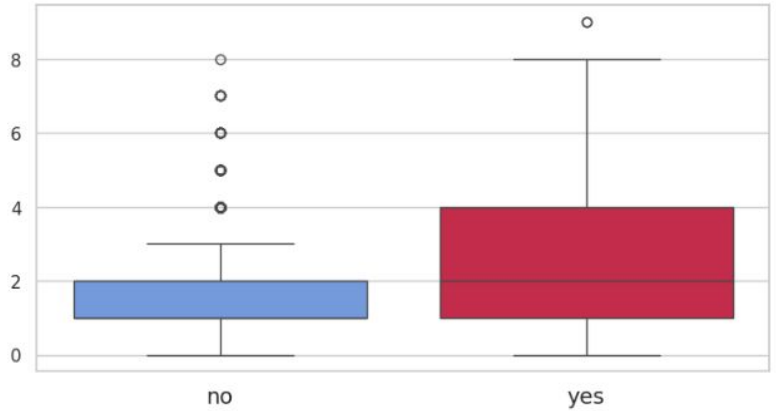
- Terlihat kedua data ini memiliki hubungan, yaitu semakin lama melakukan panggilan maka akan semakin besar biayanya
- Customer yang diindikasikan churn cenderung memiliki durasi panggilan yang lebih lama dan menghabiskan biaya panggilan yang lebih tinggi di siang hari

Exploratory Data Analysis

Data Numerical

Distribusi jumlah panggilan ke customer service





Customer yang diindikasikan churn cenderung lebih sering menghubungi customer service dibanding yang tidak churn

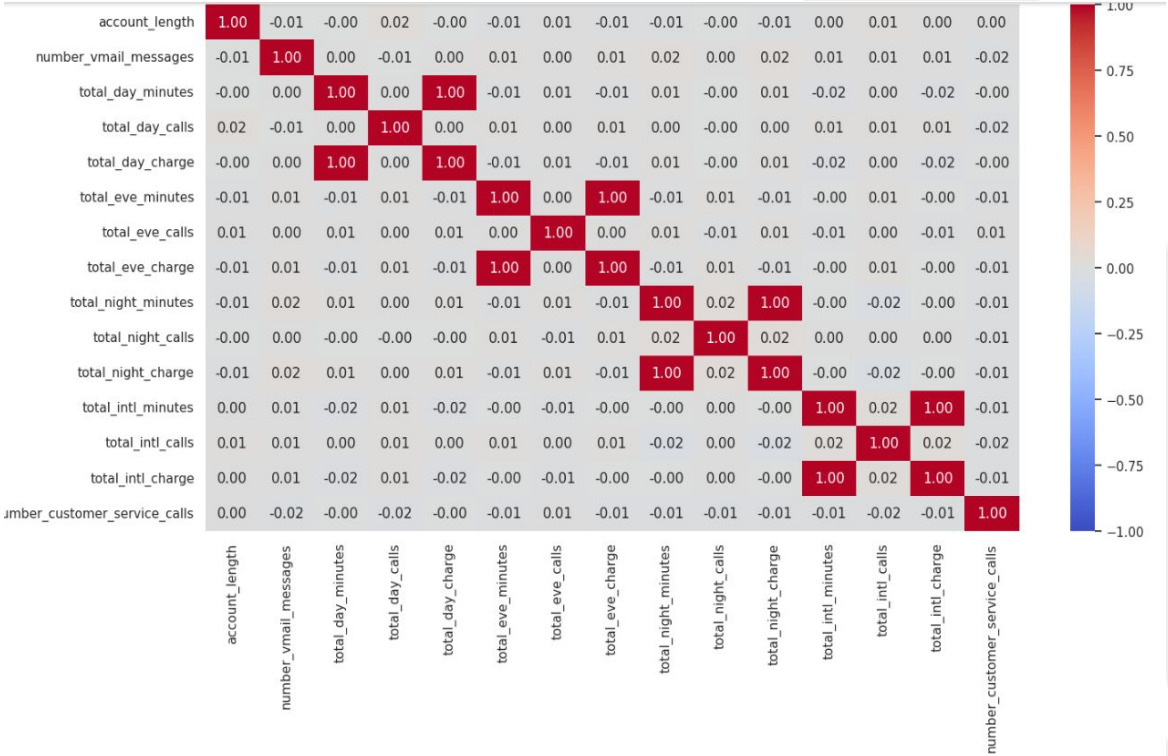
Korelasi

Pada heatmap di samping hanya dilakukan pada variabel yang bertipe numerik (integer dan float). Dapat dilihat nilai korelasi pada masing-masing variabel, terdapat 4 pasang variabel yang memiliki korelasi positif yang sangat kuat yaitu:

- Total durasi (menit) panggilan internasional dengan biaya yang dikeluarkan
- Total durasi (menit) panggilan pada malam hari dengan biaya yang dikeluarkan
- Total durasi (menit) panggilan pada sore hari dengan biaya yang dikeluarkan
- Total durasi (menit) panggilan pada siang hari dengan biaya yang dikeluarkan

Korelasi positif kuat mengindikasikan bahwa pada pasangan variabel tersebut memiliki pengaruh satu sama lain. Semakin lama customer melakukan panggilan maka akan semakin besar biaya yang harus dikeluarkan.

Korelasi antar variabel numerik



Data Preparation

Cek Missing Value

BINAR

Data Training

```
[ ] df_train.isnull().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
```

Data training tidak memiliki missing value

Data Testing

```
df_test.isnull().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
dtype: int64
```

data testing tidak memiliki missing value

Cek Data Duplikat

BINAR

Data Training

```
[ ] df_train.duplicated().sum()
```

```
0
```

Data tidak memiliki data duplikat, artinya setiap baris memiliki karakteristik yang unik.

Data Testing

```
[ ] df_test.duplicated().sum()
```

```
0
```

Data testing tidak memiliki data duplikat

Mengubah data kategorik menjadi numerik

Untuk dapat dianalisis lebih lanjut data kategori diubah menjadi data numerik dengan treatment yang berbeda.

- state: class terlalu banyak dan tidak begitu berpengaruh, untuk menghindari overfitting maka state dihapus
- area_code: One-hot Encoding
- international_plan: Label Encoding
- voice_mail_plan: Label Encoding
- churn: Label Encoding

Mengubah Data Kategorik Menjadi Numerik

Mengubah data kategorik menjadi numerik pada data training

```
# menghapus kolom state
df_clean = df_train.drop('state', axis=1)

# membuat one hot encoding pada kolom area code
area_code = pd.get_dummies(df_train['area_code'], dtype=int, prefix='is')
df_clean = pd.concat([df_clean,area_code], axis=1).drop('area_code', axis=1)

# membuat label encoding pada kolom international_plan, voice_mail_plan, dan churn
label_encoding = {"no" : 0, "yes" : 1}
df_clean['international_plan'] = df_clean['international_plan'].map(label_encoding)
df_clean['voice_mail_plan'] = df_clean['voice_mail_plan'].map(label_encoding)
df_clean['churn'] = df_clean['churn'].map(label_encoding)
```

```
df_clean.head()
```

	account_length	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls
0	107	0	1	26	161.6	123
1	137	0	0	0	243.4	114
2	84	1	0	0	299.4	71
3	75	1	0	0	166.7	113
4	121	0	1	24	218.2	88

5 rows × 21 columns

Mengubah Data Kategorik Menjadi Numerik

Mengubah data kategorik menjadi numerik pada data testing

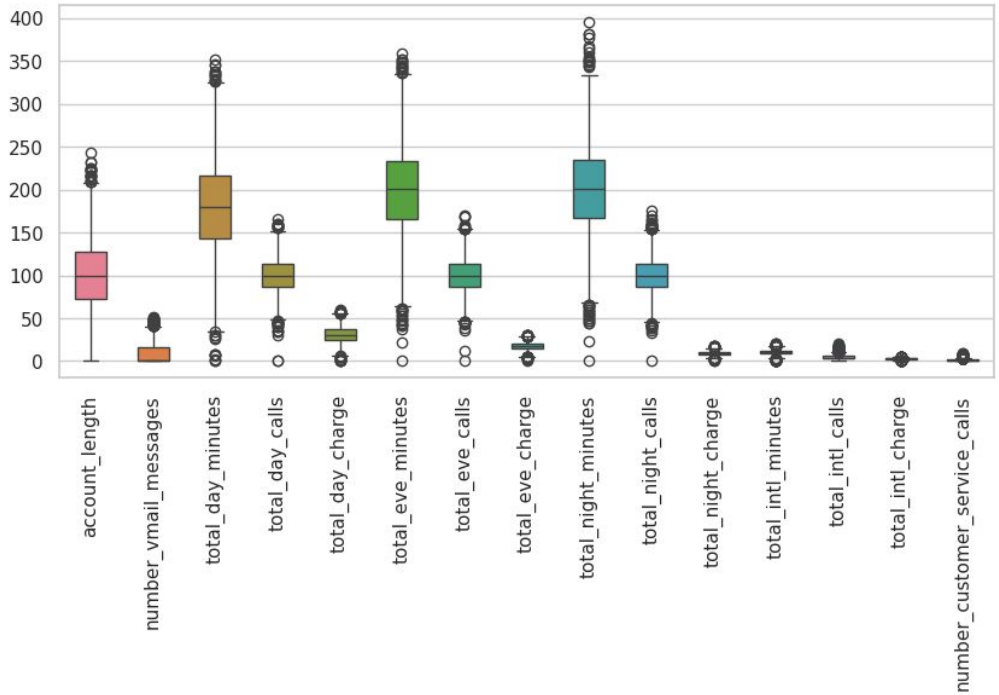
```
# menghapus kolom state
df_clean_test = df_test.drop('state', axis=1)

# membuat one hot encoding pada kolom area code
area_code = pd.get_dummies(df_test['area_code'], dtype=int, prefix='is')
df_clean_test = pd.concat([df_clean_test,area_code], axis=1).drop('area_code', axis=1)

# membuat label encoding pada kolom international_plan, voice_mail_plan, dan churn
label_encoding = {"no" : 0, "yes" : 1}
df_clean_test['international_plan'] = df_clean_test['international_plan'].map(label_encoding)
df_clean_test['voice_mail_plan'] = df_clean_test['voice_mail_plan'].map(label_encoding)
```

```
df_clean_test.head()
```

	account_length	international_plan	voice_mail_plan	number_vmail_messages	total_day_
0	128	0	1	25	
1	118	1	0	0	
2	62	0	0	0	
3	93	0	0	0	
4	174	0	0	0	

Deteksi outlier

Dalam analisis outlier kita hanya menggunakan data numerik pada data training



Pada boxplot di samping dapat dilihat bahwa semua variabel numerik memiliki nilai outlier. Outlier akan sangat berdampak ketika akan membuat modeling. Ada dua treatment yang bisa dilakukan, membuang outlier atau menggantikan outlier dengan upper dan lower bound.

Karena data yang digunakan imbalance, dan kemungkinan besar akan lebih banyak data yang terbuang, maka pada analisis ini kami akan mempertahankannya dengan mengganti menjadi nilai upper bound dan lower bound pada tiap variabel.

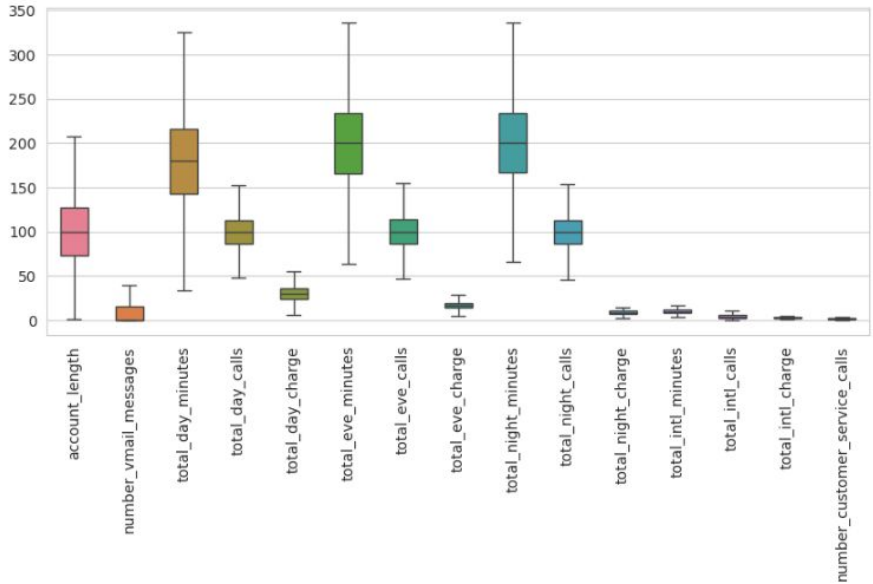
Deteksi & Treatment Outlier

Mengganti nilai outlier dengan nilai upper bound dan lower bound

```
def replace_outliers(df,columns):
    for col in columns:
        q1 = df[col].quantile(0.25)
        q3 = df[col].quantile(0.75)
        iqr = q3 - q1
        upper_bound = q3 + 1.5 * iqr
        lower_bound = q1 - 1.5 * iqr
        df[col] = df[col].mask(df[col]< lower_bound, lower_bound,axis=0)
        df[col] = df[col].mask(df[col]> upper_bound, upper_bound,axis=0)

    return df
```

```
df_clean = replace_outliers(df_clean, numeric_column)
```

Distribusi data setelah mengganti nilai outlier dengan upper dan lower bound



BINAR

Membagi data train menjadi data train dan data validasi

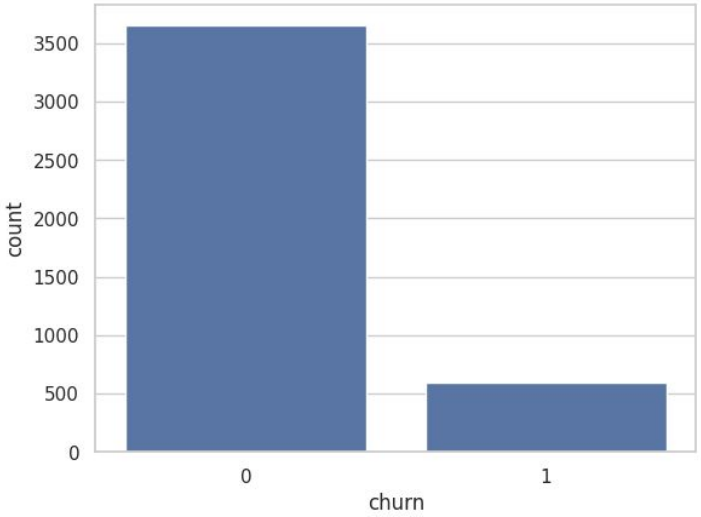
```
# Train Test Split
X = df_clean.drop('churn',axis=1)
y = df_clean['churn']
# Membagi data train menjadi data train dan data validasi
X_train, X_val , y_train, y_val = train_test_split(X, y , test_size=0.3, random_state=101)
```

Membagi data menjadi 2 yaitu nilai X yang terdiri seluruh feature kecuali feature churn, dan nilai y yaitu variabel churn sebagai targetnya. Kemudian data itu dibagi secara acak, dengan 30% data menjadi data validasi.

Handling Imbalance Data

BINAR

Kelas dari variabel target Churn yang tidak seimbang

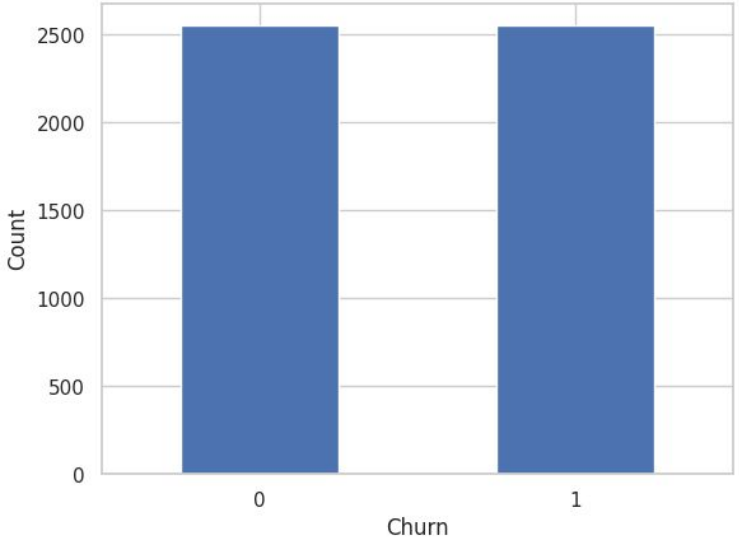


Melakukan over sampling dengan Smote

```
from imblearn.over_sampling import SMOTE
```

```
# Minority Over Sampling Technique
sm = SMOTE(sampling_strategy = 1, random_state=1)
X_train_smote, y_train_smote = sm.fit_resample(X_train, y_train.ravel())
```

Kelas dari variabel target Churn setelah dilakukan over sampling



Transformasi Data Menggunakan MinMax

Transformasi pada data training

```
# X_train
scaler = MinMaxScaler()
X_train_transform = scaler.fit_transform(X_train_smote)
X_train_transform = pd.DataFrame(X_train_transform, columns=X_train.columns)
```

```
# X_val
scaler = MinMaxScaler()
X_val_transform = scaler.fit_transform(X_val)
X_val_transform = pd.DataFrame(X_val_transform, columns=X_val.columns)
```

```
X_train_transform.head()
```

	account_length	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes
0	0.449275	0.0	0.0	0.000	0.552616
1	0.599034	0.0	0.0	0.000	0.317967
2	0.473430	0.0	1.0	0.525	0.501501
3	0.816425	0.0	0.0	0.000	0.728602
4	0.299517	0.0	0.0	0.000	0.502530

```
X_val_transform.head()
```

	account_length	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	t
0	0.507246	0.0	0.0	0.0	0.390695	
1	0.323671	0.0	0.0	0.0	0.375943	
2	0.574879	0.0	0.0	0.0	0.206818	
3	0.676329	0.0	0.0	0.0	0.318654	
4	0.657005	0.0	1.0	1.0	0.523113	

Transformasi Data Menggunakan MinMax

Transformasi pada data testing

```
df_test_transform = scaler.fit_transform(df_clean_test)
df_test_transform = pd.DataFrame(df_test_transform, columns=df_clean_test.columns)
```

```
df_test_transform.head()
```

	account_length	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls
0	0.535865	0.0	1.0	0.490196	0.746675	0.572581
1	0.493671	1.0	0.0	0.000000	0.623411	0.475806
2	0.257384	0.0	0.0	0.000000	0.319834	0.250000
3	0.388186	0.0	0.0	0.000000	0.526751	0.604839
4	0.729958	0.0	0.0	0.000000	0.330476	0.298387

BINAR

Modeling

Regresi Logistik

Melakukan klasifikasi dengan model regresi logistik

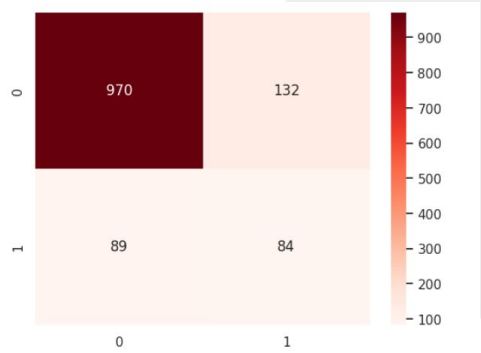
```
logmodel = LogisticRegression()
```

```
logmodel.fit(X_train_transform, y_train_smote)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
log_predict = logmodel.predict(X_val_transform)
```

Confusion Matrix



Evaluasi hasil klasifikasi model regresi logistik

	Accuracy	Precision	Recall	f1
Regresi Logistik	0.827	0.389	0.486	0.432

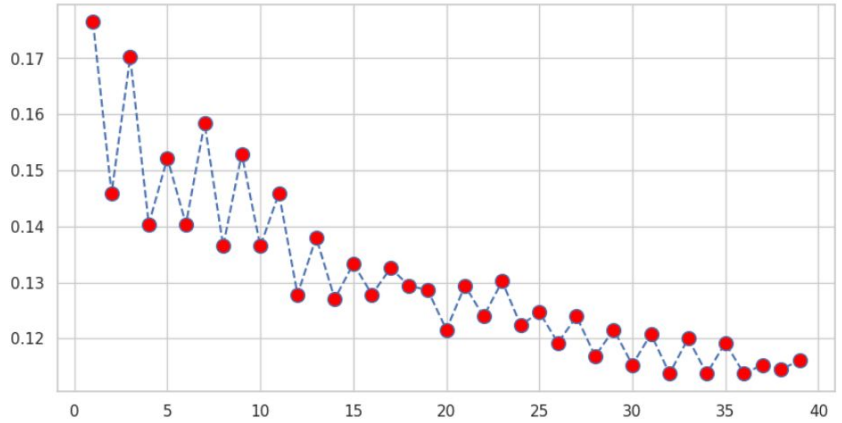
Secara general, akurasi model regresi logistik sudah cukup baik yaitu 83%. Namun, nilai recall cukup rendah yaitu hanya 49%.

K Neighbors Classifier

Membuat simulasi error rate untuk mencari nilai k optimal

```
error_rate = []

for i in range(1,40):

    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train_transform, y_train_smote)
    pred_i = knn.predict(X_val_transform)
    error_rate.append(np.mean(pred_i != y_val))
```

```
sns.set_style('whitegrid')
plt.figure(figsize=(10,5))
plt.plot(range(1,40), error_rate, linestyle='dashed', marker='o', markerfacecolor='red', markersize=10);
```



Berdasarkan simulasi didapatkan nilai k=36 dengan error rate paling rendah.

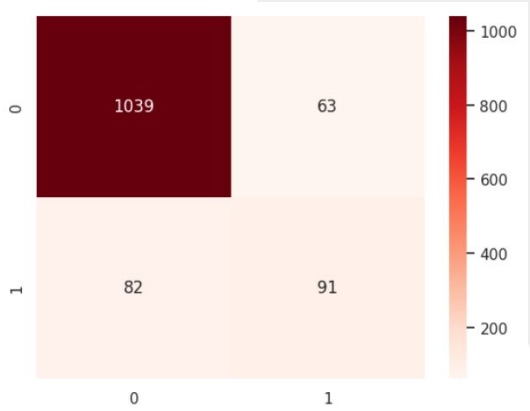
Maka akan digunakan nilai k=36 dalam pembuatan model

K Neighbors Classifier

Melakukan klasifikasi dengan model K Neighbors Classifier (KNN)

```
knn = KNeighborsClassifier(n_neighbors=36)
knn.fit(X_train_transform, y_train_smote)
knn_predict = knn.predict(X_val_transform)
```

Confusion Matrix



Evaluasi hasil klasifikasi model KNN

	Accuracy	Precision	Recall	f1
KNN	0.886	0.591	0.526	0.557

Secara general, akurasi model knn sudah cukup baik yaitu 89% dan nilai recall yaitu 52%.

Decision Tree

Melakukan klasifikasi dengan model decision tree

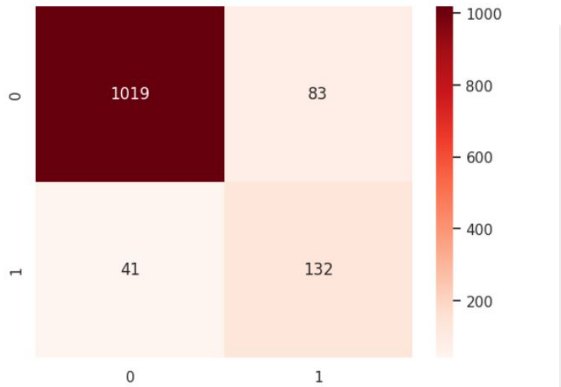
```
dtc = DecisionTreeClassifier()
dtc.fit(X_train_transform, y_train_smote)
dtc_predict = dtc.predict(X_val_transform)
```

Evaluasi hasil klasifikasi model decision tree

	Accuracy	Precision	Recall	f1
Decision Tree	0.903	0.614	0.763	0.68

Secara general, akurasi model decision tree sudah cukup baik yaitu 90% begitu juga dengan nilai recall mencapai 76%.

Confusion Matrix



Random Forest

Melakukan klasifikasi dengan model Random Forest

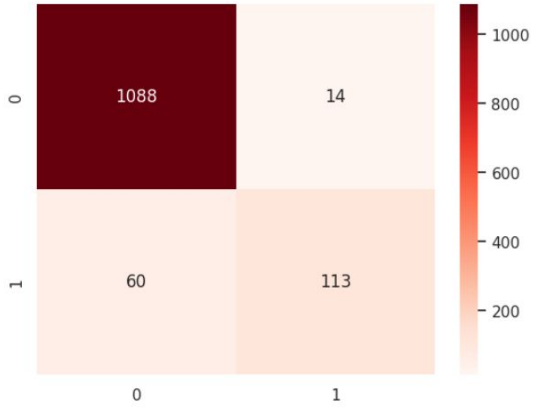
```
rf = RandomForestClassifier()
rf.fit(X_train_transform, y_train_smote)
rf_predict = rf.predict(X_val_transform)
```

Evaluasi hasil klasifikasi model random forest

	Accuracy	Precision	Recall	f1
Random Forest	0.942	0.89	0.653	0.753

Secara general, akurasi model random forest sudah sangat baik yaitu 94% dan nilai recall yaitu 65%.

Confusion Matrix



Evaluation

Evaluasi Model

BINAR

Melakukan Evaluasi pada keempat model

	Accuracy	Precision	Recall	f1
Regresi Logistik	0.827	0.389	0.486	0.432
KNN	0.886	0.591	0.526	0.557
Decision Tree	0.903	0.614	0.763	0.680
Random Forest	0.942	0.890	0.653	0.753

Berdasarkan hasil evaluasi berbagai model, model yang memiliki ketepatan prediksi terbaik adalah model Random Forest. Sehingga model random forest dipilih untuk melakukan prediksi pada data test.

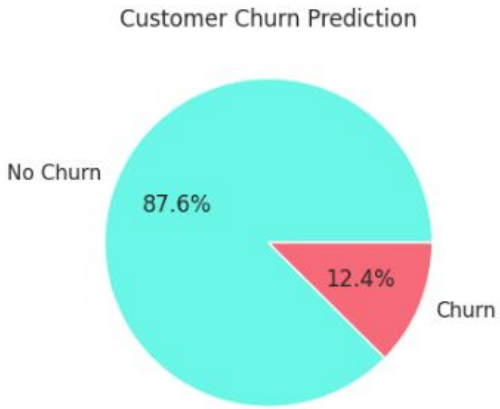
BINAR

Predictions

Prediksi Pada Data Testing

Melakukan prediksi data test dengan model random forest sebagai model terbaik

```
[ ]  predict = pd.Series(rf.predict(df_test_transform))
```



Dari sampel data test, diprediksi bahwa akan ada 12,4% customer yang akan pindah ke provider lain(churn), sedangkan sisanya 87,6% masih menjadi pelanggan setia provider

Kesimpulan dan Saran

BINAR

Kesimpulan

- Customer churn cenderung lebih banyak melakukan panggilan ke customer service dibanding yang tidak churn
- Customer churn banyak menghabiskan waktu dan biaya menelepon di siang hari
- Hampir separuh customer churn memiliki international plan
- Dari berbagai model klasifikasi customer churn, model random forest memiliki akurasi yang terbaik yaitu 94,2%
- Dari sampel data testing, diprediksi sebanyak 12,4% customer akan beralih provider (churn)

Saran yang dapat dipertimbangkan untuk perusahaan

- Perusahaan perlu meningkatkan pelayanan mereka agar dapat menyelesaikan masalah kurang dari 3 kali customer service call
- Memberikan promosi harga/paket promosi pada customer yang intens melakukan panggilan di siang hari
- Perusahaan dapat meningkatkan kualitas jaringan terkhusus pada jaringan internasional (Roaming)

BINAR

Link Google Colab

BINAR

Thank You