Kelompok 12 DS 8 - Challenge Chapter 2

Data Science

Anggota Tim

TIM	NAMA
Kelompok 12 DS 8	Hanosi Wazri
	Inocentius Reynaldo Bhoka Tola

Introduction

Di zaman yang semakin canggih perkembangan industri telekomunikasi sekarang sangat cepat, ya, gengs. 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.

Studi Kasus :
Memprediksi customer churn

Oleh karena itu, kamu yang baru aja direkrut sebagai junior data scientist di perusahaan telekomunikasi diminta untuk melakukan prediksi customer churn.

Prediksi ini penting diketahui oleh perusahaan agar bisa memetakan strategi bisnis untuk mempertahankan pelanggan.

Langkah yang harus dilakukan:

- Membuat model machine learning dengan algoritma klasifikasi
- Melakukan prediksi customer churn

BINAR

❖Data Manipulation

- **`pandas`**: Library untuk manipulasi dan analisis data.
- **`numpy`**: Library untuk komputasi numerik yang efisien.

❖Data Visualization

- **`seaborn`**: Library untuk visualisasi data berbasis matplotlib, memberikan tampilan yang lebih menarik dan informatif.
- **`matplotlib.pyplot`**: Library untuk membuat visualisasi seperti grafik dan plot.
- **`plotly.graph_objs`**: Library untuk membuat visualisasi interaktif dan visualisasi data dengan plotly.
- **`plotly.express`**: Menyediakan antarmuka tingkat tinggi untuk membuat visualisasi data interaktif dengan plotly.

❖Modeling

- **`sklearn.model_selection`**: Menyediakan alat untuk pemisahan data, validasi silang, dan pencarian hiperparameter.
- **`imblearn.over_sampling`**: Library untuk penanganan ketidakseimbangan kelas dalam dataset.
- **`sklearn.metrics`**: Menyediakan berbagai metrik evaluasi kinerja model.
- **`scipy.stats`**: Library untuk analisis statistik.
- **`sklearn.preprocessing.LabelEncoder`**: Untuk mengkodekan label target dengan nilai antara 0 dan n_classes-1.
- **`sklearn.preprocessing.MinMaxScaler`**: Untuk penskalaan fitur ke rentang yang diberikan.

BINAR

❖Algorithms for Supervised Learning Methods

- **`sklearn.tree.DecisionTreeClassifier`**: Algoritma Decision Tree untuk klasifikasi.
- **`sklearn.ensemble.RandomForestClassifier`**: Algoritma Random Forest untuk klasifikasi.

❖Filtering Future Warnings

- **`warnings`**: Library untuk mengatur tindakan yang diambil oleh interpreter terhadap peringatan. Dalam kasus ini, mengabaikan peringatan yang dihasilkan.

BINAR

❖Model Script

```
# Data manipulation
import pandas as pd
import numpy as np

# Data visualization
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objs as go
import plotly.express as px

# Modeling
from sklearn.model_selection import train_test_split,cross_val_score,GridSearchCV
from imblearn.over_sampling import SMOTE,SMOTENC
from sklearn.metrics import f1_score,recall_score,precision_score,confusion_matrix,roc_curve,roc_auc_score,classification_report,accuracy_score, auc # performance metrics
from scipy import stats
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

# Algorithms for supervised learning methods
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
import xgboost as xgb
from xgboost import XGBClassifier

# Filtering future warnings
import warnings
warnings.filterwarnings('ignore')
```

BINAR

1. DATA UNDERSTANDING
2. DATA PREPARATION
3. MODELING
4. MODEL EVALUATION

DATA UNDERSTANDING

Untuk dataset yang digunakan sudah disediakan dalam format csv yaitu train= 'Data Train.csv', test= 'Data Test.csv' , menggunakan perpustakaan pandas (pd). Data dari kedua file CSV tersebut dibaca dan dimuat ke dalam dua dataframe terpisah: 'test' dan 'train'. Selanjutnya, data dari dataframe 'train' disalin ke dataframe baru yang disebut 'data'. Pemanggilan 'data.head()' digunakan untuk menampilkan lima baris pertama dari dataframe 'data', yang berguna untuk memberikan gambaran awal tentang struktur dan isinya.

```
test = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Data Test.csv')
train = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Data Train.csv')
```

```
data = train
```

```
data.head()
```

e_charge	total_night_minutes	total_night_calls	total_night_charge	total_intl_minutes	total_intl_calls	total_intl_charge	number_customer_service_calls	churn
16.62	254.4	103	11.45	13.7	3	3.70	1	no
10.30	162.6	104	7.32	12.2	5	3.29	0	no
5.26	196.9	89	8.86	6.6	7	1.78	2	no
12.61	186.9	121	8.41	10.1	3	2.73	3	no
29.62	212.6	118	9.57	7.5	7	2.03	3	no

BINAR

Script berikut memiliki dua tujuan utama. Yang pertama adalah untuk mencetak jumlah baris dan kolom dari dataframe 'data', di mana ".shape[0]" digunakan untuk mendapatkan jumlah baris dan ".shape[1]" untuk mendapatkan jumlah kolom. Informasi ini memberikan gambaran singkat tentang ukuran data yang sedang diproses. Yang kedua adalah untuk memberikan statistik deskriptif ringkas dari dataframe 'data' menggunakan metode '.describe()'. Statistik ini mencakup jumlah, rata-rata, standar deviasi, nilai minimum, kuartil, dan nilai maksimum dari kolom-kolom numerik dalam dataframe

```
#Shape of the dataframe
print("The number of rows: {}".format(data.shape[0]))
print("The number of columns:{}".format(data.shape[1]))
```

```
The number of rows: 4250
The number of columns:20
```

```
data.describe()
```

	account_length	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total_eve_calls	total_eve_charge	total_ni
count	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	
mean	100.236235	7.631765	180.259600	99.907294	30.644682	200.173906	100.176471	17.015012	
std	39.698401	13.439882	54.012373	19.850817	9.182096	50.249518	19.908591	4.271212	
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	73.000000	0.000000	143.325000	87.000000	24.365000	165.925000	87.000000	14.102500	
50%	100.000000	0.000000	180.450000	100.000000	30.680000	200.700000	100.000000	17.060000	
75%	127.000000	16.000000	216.200000	113.000000	36.750000	233.775000	114.000000	19.867500	
max	243.000000	52.000000	351.500000	165.000000	59.760000	359.300000	170.000000	30.540000	

metode 'info()' digunakan untuk mencetak informasi ringkas tentang dataframe, termasuk jumlah non-null values dan tipe data dari setiap kolom. Kemudian, dilakukan pemisahan kolom-kolom numerik dan kategorikal menggunakan metode 'select_dtypes()' dengan parameter 'include' yang sesuai dengan tipe data yang diinginkan, yaitu 'number' untuk kolom numerik dan 'object' untuk kolom kategorikal. Setelah itu, hasilnya dicetak dalam bentuk daftar kolom numerik dan kategorikal. Terakhir, menggunakan metode 'isnull().sum()', dilakukan pengecekan terhadap jumlah nilai null pada setiap kolom dalam dataframe 'data'. Hal ini memberikan informasi awal tentang keberadaan nilai null dalam dataset yang mungkin memerlukan penanganan lebih lanjut sebelum melakukan analisis.

```
data.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
```

```
# Numerical Columns
print(f"Numerical Columns: {data.select_dtypes(include='number').columns}\n")
# Categorical Columns
print(f"Categorical Columns: {data.select_dtypes(include='object').columns}")
```

```
Numerical Columns: Index(['account_length', 'number_vmail_messages', 'total_day_minutes',
       'total_day_calls', 'total_day_charge', 'total_eve_minutes',
       'total_eve_calls', 'total_eve_charge', 'total_night_minutes',
       'total_night_calls', 'total_night_charge', 'total_intl_minutes',
       'total_intl_calls', 'total_intl_charge',
       'number_customer_service_calls'],
      dtype='object')

Categorical Columns: Index(['state', 'area_code', 'international_plan', 'voice_mail_plan', 'churn'], dtype='object')
```

```
data.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 Cleaning

Script ini merupakan bagian dari proses pembersihan data (data cleaning) yang bertujuan untuk memeriksa keberadaan nilai yang hilang (missing values) dan duplikat dalam sebuah DataFrame. Fungsi 'check_missing_values(df)' digunakan untuk memeriksa nilai yang hilang dalam DataFrame 'df'. Jika tidak ada nilai yang hilang, akan dikembalikan pesan "No missing values found.". Namun, jika ada nilai yang hilang, fungsi akan menghitung jumlah dan persentase nilai yang hilang untuk setiap kolom, kemudian mengembalikan DataFrame yang berisi informasi tersebut. Sementara itu, fungsi 'check_duplicates(df)' digunakan untuk memeriksa duplikat dalam DataFrame 'df'. Jika duplikat ditemukan, akan dicetak pesan yang berisi indeks duplikat. Namun, jika tidak ada duplikat, akan dicetak pesan "No duplicates"

```python
def check_missing_values(df):
    """
    A function to check for missing values in a DataFrame
    """
    missing_values = df.isnull().sum().sort_values(ascending=False)
    if missing_values.sum() == 0:
        return "No missing values found."
    else:
        missing_percent = round(missing_values/len(df)*100,2)
        missing_values = pd.concat([missing_values, missing_percent], axis=1, keys=['Number of Missing Values', 'Percentage of Missing Values'])
        return missing_values

def check_duplicates(df):
    """
    Function to check for duplicates in a DataFrame
    """
    duplicates = df.duplicated()
    if duplicates.any():
        print("Duplicates found:{}", duplicates)
    else:
        print("No duplicates")
```

```
check_missing_values(data)
```

```
'No missing values found.'
```

```
check_duplicates(data)
```

```
No duplicates
```

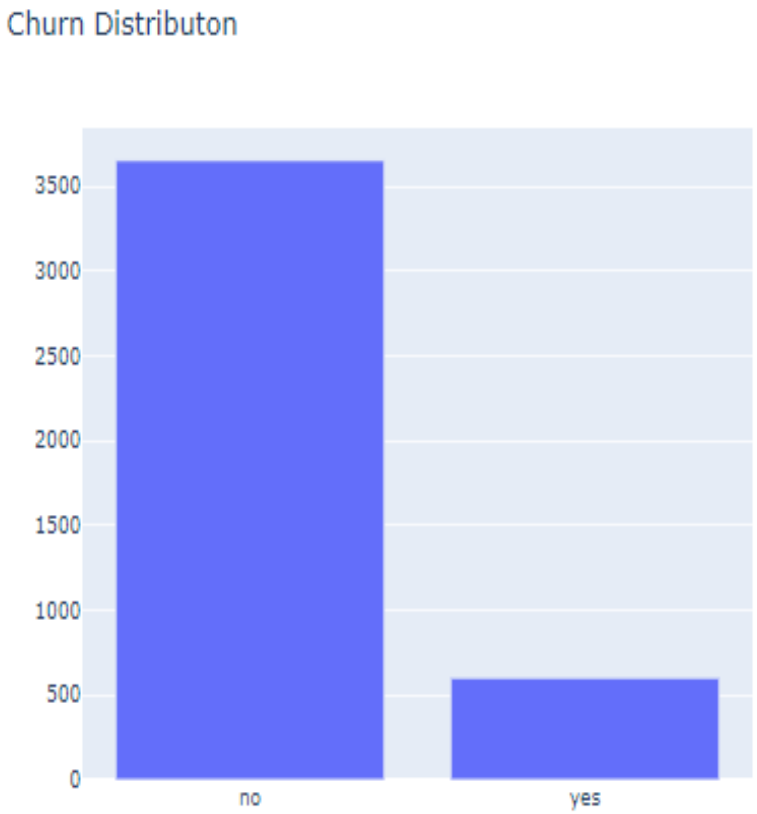
BINAR

❖EXPLORATORY DATA

CHURN FEATURE:

Fungsi 'figs(ds)' digunakan untuk memvisualisasikan distribusi variabel target 'churn'. Pertama, dilakukan penghitungan jumlah kemunculan setiap nilai dalam variabel target menggunakan metode 'value_counts()'. Selanjutnya, hasilnya digunakan untuk membuat diagram batang menggunakan Plotly, di mana sumbu x menunjukkan nilai-nilai variabel target dan sumbu y menunjukkan jumlah kemunculan masing-masing nilai. Judul diagram disesuaikan dengan variabel target yang sedang diproses, dengan mengganti tanda underscore (_) dengan spasi. Diagram yang dihasilkan kemudian ditampilkan menggunakan metode 'show()'.

```
def figs(ds) :
  #plotting the target variable distribution
  class_counts = data[ds].value_counts()

  # Create a bar chart of the value counts using Plotly
  fig = go.Figure(
      data=[go.Bar(x=class_counts.index, y=class_counts.values)],
      layout=go.Layout(title=ds.title().replace('_',' ')+' Distributon',
            hovermode = 'closest',width=600)
  )

  # Show the chart
  fig.show()

figs('churn')
```



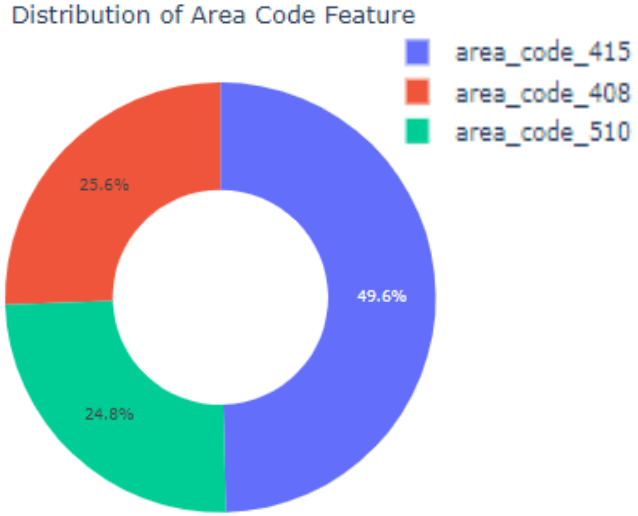
BINAR

❖EXPLORATORY DATA

AREA CODE FEATURE:

Pada bagian ini, dibuat sebuah pie chart untuk memvisualisasikan distribusi fitur 'area_code'. Pertama, dilakukan penghitungan jumlah kemunculan setiap kode area menggunakan metode 'value_counts()'. Selanjutnya, menggunakan Plotly, dibuat pie chart dengan jumlah dan label yang sesuai. Diagram ini membantu dalam memahami sebaran kode area pada dataset dengan cara visual.

```
# Pie chart of area code feature
area = data['area_code'].value_counts()
transanction = area.index
quantity = area.values

# plot pie circle with plotly
figure = px.pie(data,
                values = quantity,
                names = transanction,
                hole = .5,
                title = 'Distribution of Area Code Feature')
figure.show()
```



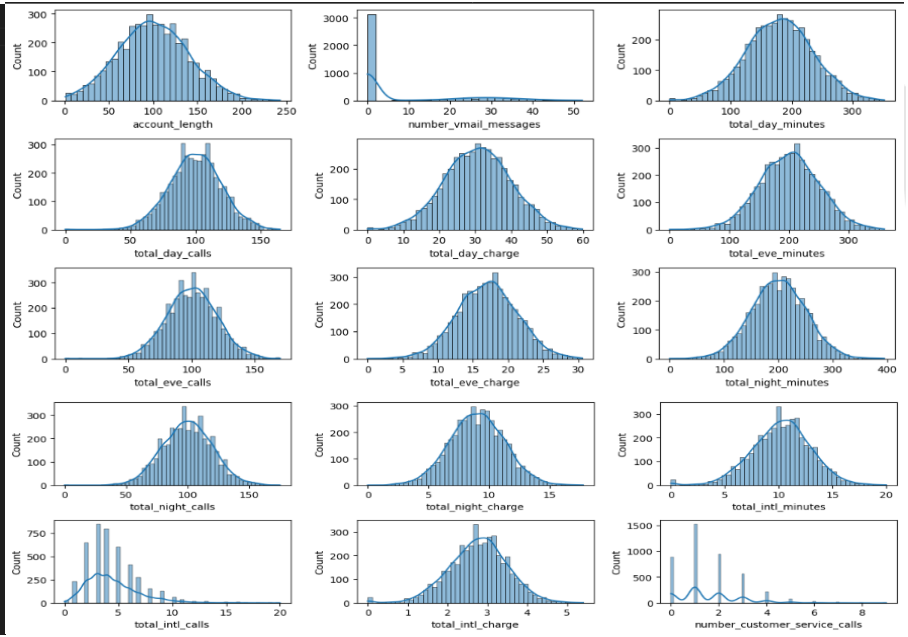
BINAR

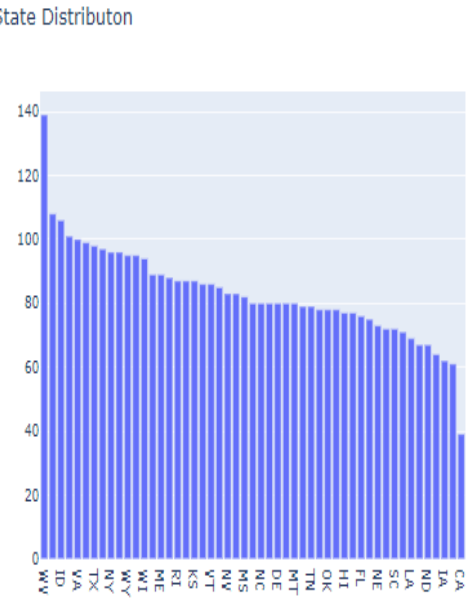
❖EXPLORATORY DATA

NUMERICAL FEATURE:

Script ini bertujuan untuk memeriksa distribusi fitur-fitur numerik dalam dataset. Pertama, didefinisikan daftar fitur numerik yang akan diperiksa. Selanjutnya, dilakukan pengaturan subplot untuk menampilkan distribusi fitur-fitur tersebut dalam bentuk histogram. Histogram digunakan untuk menunjukkan distribusi frekuensi dari setiap nilai dalam fitur-fitur numerik. Proses ini membantu dalam memahami pola distribusi dan karakteristik fitur-fitur numerik.

```python
#checking for distribution of the numeric features
numeric_features = ['account_length', 'number_vmail_messages', 'total_day_minutes',
       'total_day_calls', 'total_day_charge', 'total_eve_minutes',
       'total_eve_calls', 'total_eve_charge', 'total_night_minutes',
       'total_night_calls', 'total_night_charge', 'total_intl_minutes',
       'total_intl_calls', 'total_intl_charge',
       'number_customer_service_calls']

# Calculate the number of rows and columns for subplots
nrows = (len(numeric_features) - 1) // 3 + 1
ncols = min(3, len(numeric_features))

# Create subplots
fig, axes = plt.subplots(nrows=nrows, ncols=ncols, figsize=(12, 10))

# Flatten axes if necessary
axes = axes.flatten() if nrows > 1 else [axes]

# Plot numeric features
for i, feature in enumerate(numeric_features):
    ax = axes[i]
    sns.histplot(data[feature], kde=True, ax=ax)
    ax.set_xlabel(feature)
    ax.set_ylabel("Count")

# Remove empty subplots
if len(numeric_features) < nrows * ncols:
    for i in range(len(numeric_features), nrows * ncols):
        fig.delaxes(axes[i])

# Adjust subplot spacing
fig.tight_layout()

# Display the plot
plt.show()
```

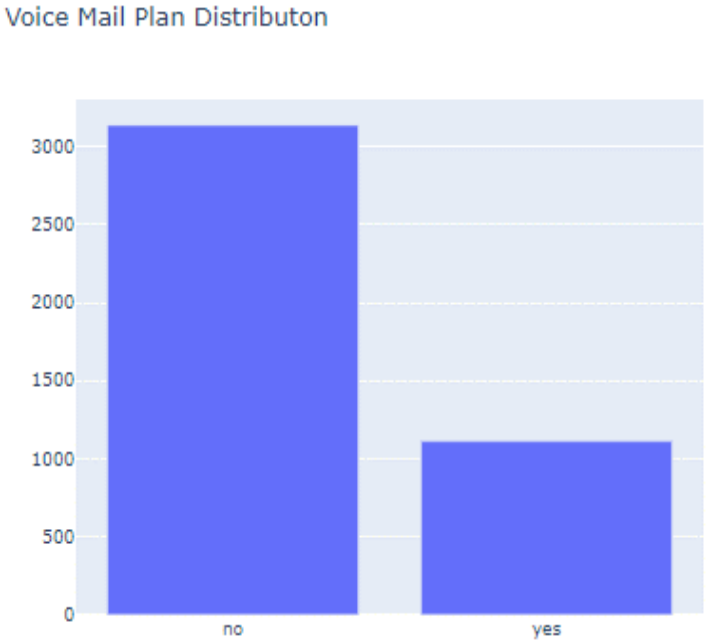




❖EXPLORATORY DATA

CATEGORICAL FEATURE:

Bagian ini digunakan untuk memvisualisasikan distribusi fitur-fitur kategorikal seperti 'international_plan' dan 'voice_mail_plan'. Fungsi 'figs(ds)' digunakan untuk memplot distribusi nilai pada fitur-fitur kategorikal. Ini membantu dalam memahami sebaran nilai-nilai pada fitur-fitur kategorikal tertentu dalam dataset.



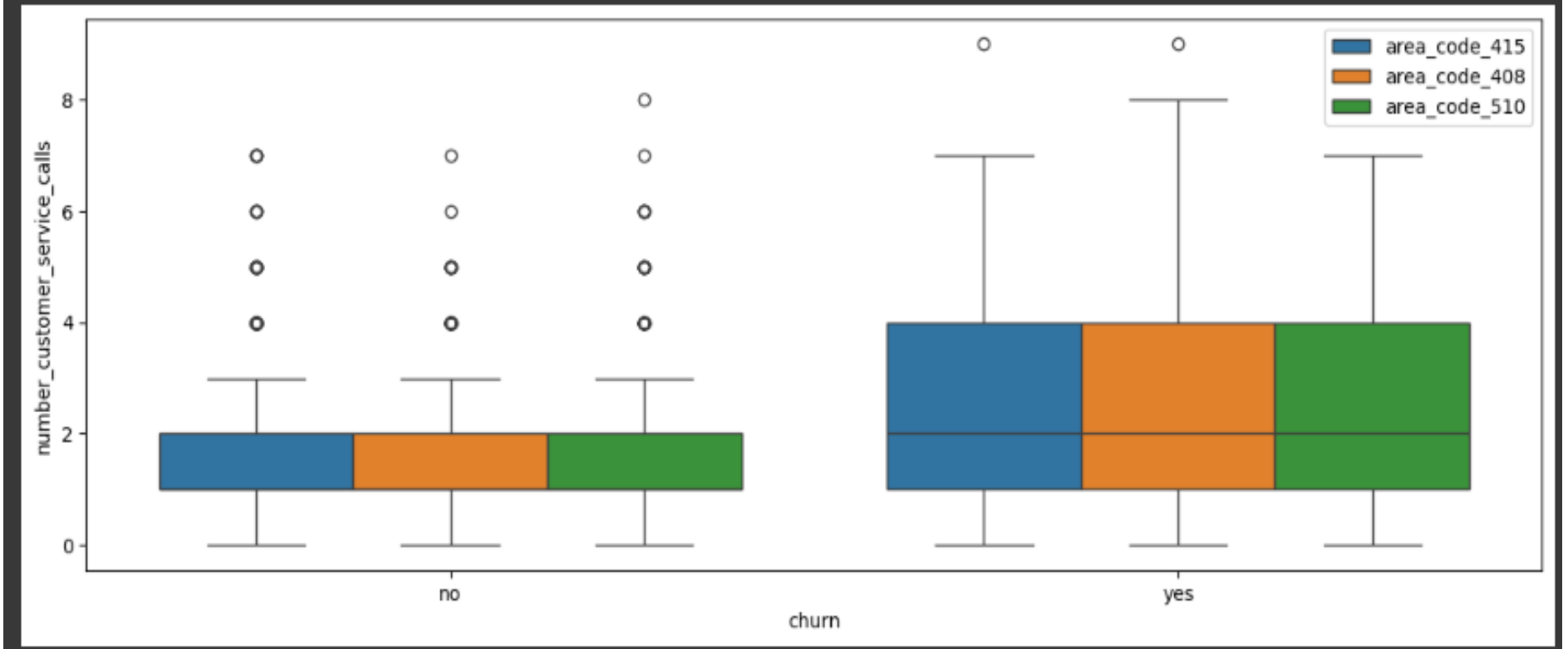
❖EXPLORATORY DATA

BOX PLOT:

Pada bagian ini, digunakan boxplot untuk mengeksplorasi hubungan antara variabel 'area_code' dengan variabel target 'churn', dengan fokus pada fitur 'number_customer_service_calls'. Boxplot adalah jenis plot yang memvisualisasikan distribusi nilai-nilai numerik dalam satu atau lebih kelompok kategori. Dalam skrip ini, setiap kotak pada plot mewakili kuartil pertama hingga kuartil ketiga dari distribusi data, dengan garis di dalamnya menunjukkan median. Garis-garis yang menghubungkan kotak dengan ujung-ujungnya (whiskers) mengindikasikan sebaran data di luar kuartil. Pemisahan plot berdasarkan 'churn' memungkinkan pengamatan perbedaan distribusi 'number_customer_service_calls' antara pelanggan yang berhenti berlangganan dan yang tidak, sementara pemisahan berdasarkan 'area_code' memungkinkan identifikasi apakah pola tersebut bervariasi tergantung pada kode area tertentu. Fungsi 'plot_categorical_distribution' digunakan untuk memplot distribusi fitur-fitur kategorikal berdasarkan variabel target 'churn', sedangkan fungsi 'plot_churn_kde' digunakan untuk memplot distribusi fitur-fitur numerik berdasarkan variabel target 'churn' menggunakan KDE plot. Ini membantu dalam pemahaman tentang bagaimana variabel 'area_code' dan fitur lainnya berkontribusi terhadap perilaku churn pelanggan melalui analisis fitur yang berbeda.

BOX PLOT - Script dan Hasil Running

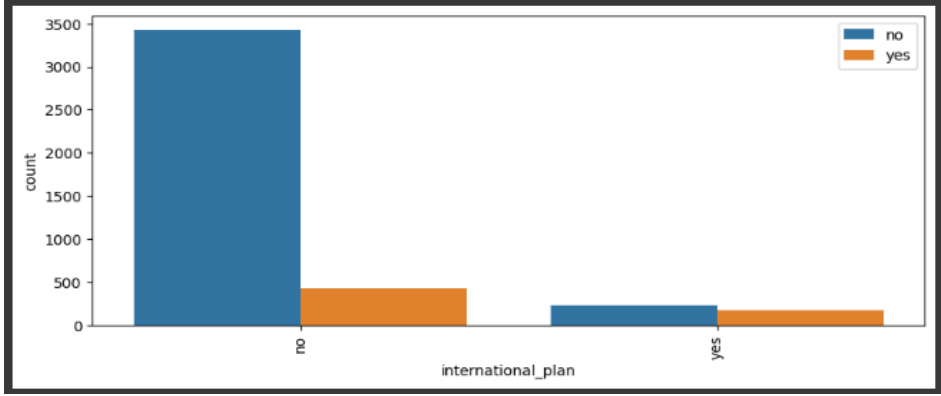
```
# Boxplot to see which area code has the highest churn
plt.figure(figsize=(14,5))
sns.boxplot(data=data,x='churn',y='number_customer_service_calls',hue='area_code');
plt.legend(loc='upper right');
```



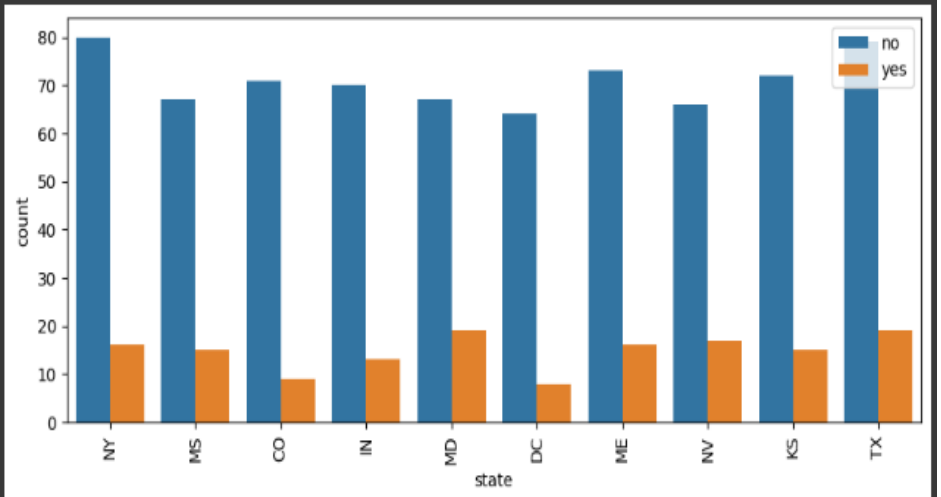
BOX PLOT - Script dan Hasil Running

```python
#Checking the distribution of categorical features based on churn rate
def plot_categorical_distribution(data, feature):
    """
    Plots the distribution of a categorical feature in the given data.
    """
    plt.figure(figsize=(10, 4))
    churn_counts = data.groupby(feature)["churn"].sum().sort_values(ascending=False)
    top_10_categories = churn_counts.head(10).index.tolist()
    sns.countplot(x=feature, hue="churn", data=data, order=top_10_categories)
    plt.xticks(rotation=90)
    plt.legend(loc="upper right")
    plt.show()
```

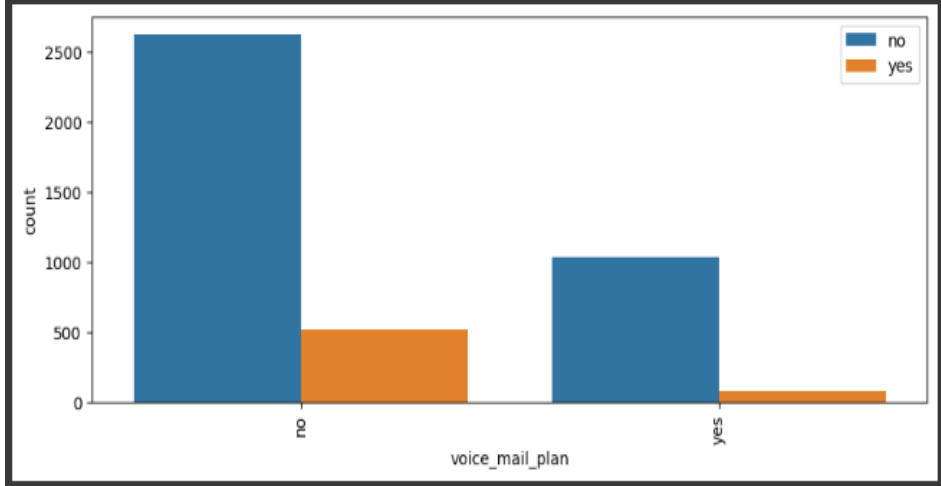
```python
plot_categorical_distribution(data, 'international_plan')
```




```python
plot_categorical_distribution(data, 'state')
```

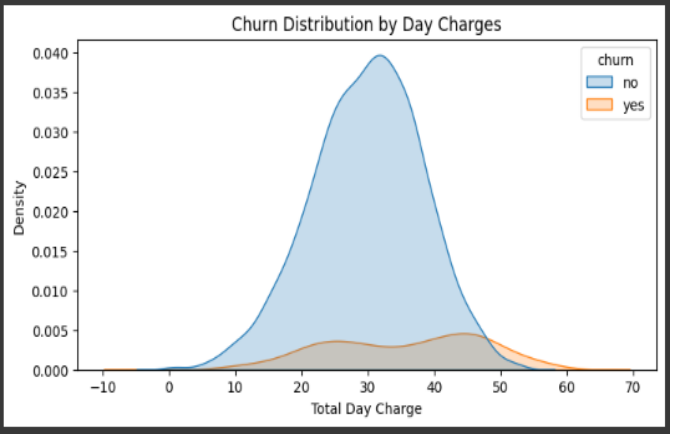



```python
plot_categorical_distribution(data, 'voice_mail_plan')
```

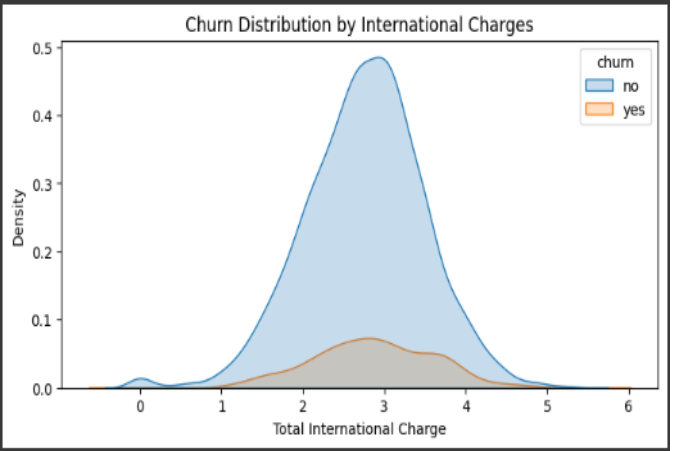


BOX PLOT - Script dan Hasil Running

```
# Churn by day charges
plot_churn_kde(data, 'total_day_charge', 'Day')
```

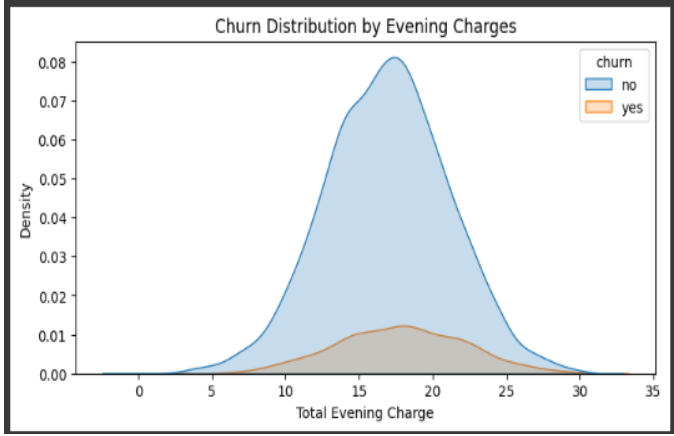



```
plot_churn_kde(data, 'total_intl_charge', 'International')
```

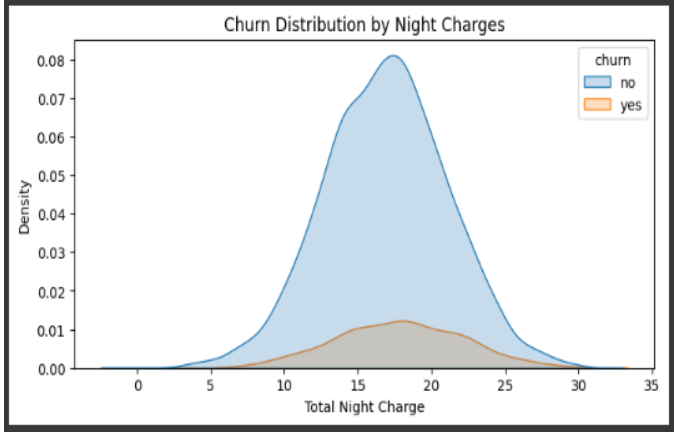



```
def plot_churn_kde(data, x_column, charge_type):
    """
    A function to plot features based on churn rate
    """
    plt.figure(figsize=(8, 4))
    sns.kdeplot(data=data, x=x_column, hue='churn', fill=True)
    plt.xlabel(f'Total {charge_type} Charge')
    plt.ylabel('Density')
    plt.title(f'Churn Distribution by {charge_type} Charges')
    plt.show()
```

```
# Churn by evening charges
plot_churn_kde(data, 'total_eve_charge', 'Evening')
```




```
# Churn by night charges
plot_churn_kde(data, 'total_eve_charge', 'Night')
```



BINAR

❖OUTLIER

Script ini digunakan untuk menghapus outlier dari fitur-fitur numerik dalam dataset. Fungsi 'drop_numerical_outliers(df, z_thresh=3)' menerima sebuah dataframe 'df' dan ambang z-score 'z_thresh' sebagai argumen. Dalam skrip ini, outlier diidentifikasi dengan menggunakan metode z-score, di mana setiap fitur numerik diubah menjadi z-score dan nilai absolutnya dibandingkan dengan ambang tertentu. Jika z-score lebih besar dari ambang tersebut, data dianggap sebagai outlier dan akan dihapus. Setelah pemrosesan, ukuran dataframe dicetak sebagai output, menunjukkan jumlah baris dan kolom setelah penghapusan outlier dilakukan.

```
def drop_numerical_outliers(df, z_thresh=3):
    constrains = df.select_dtypes(include=[np.number]).apply(lambda x: np.abs(stats.zscore(x)) < z_thresh).all(axis=1)
    df.drop(df.index[~constrains], inplace=True)

drop_numerical_outliers(data)
print(data.shape)
```

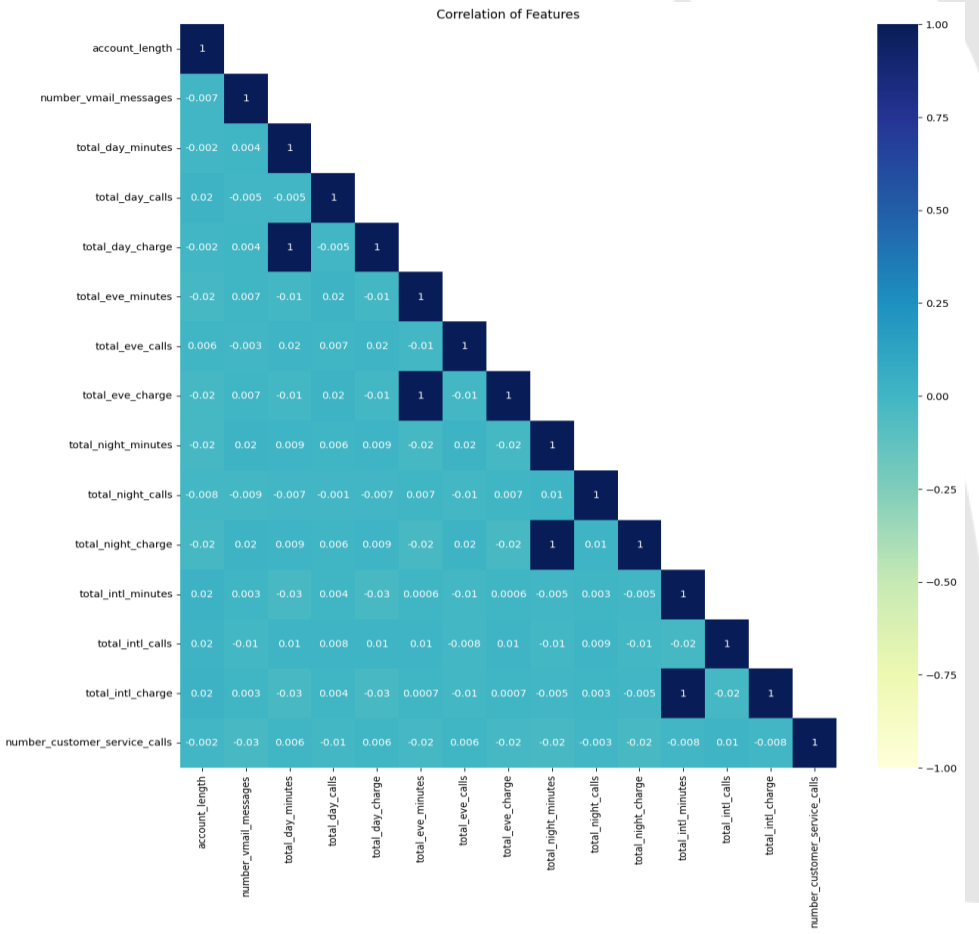
```
(4031, 20)
```

BINAR

❖FEATURE CORRELATION

Bagian ini bertujuan untuk mengeksplorasi korelasi antara fitur-fitur dalam dataset menggunakan heatmap. Fungsi 'corrmatrix(df)' digunakan untuk menghasilkan matriks korelasi antara fitur-fitur numerik dalam dataframe 'df'. Matriks korelasi kemudian ditampilkan dalam bentuk heatmap dengan menggunakan pustaka seaborn (sns), di mana warna-warna yang berbeda menunjukkan tingkat korelasi antara pasangan fitur. Anotasi pada heatmap menampilkan nilai korelasi untuk setiap pasangan fitur. Ini membantu dalam memahami hubungan linier antara fitur-fitur dalam dataset.

```
# Finding correlation between features using a heatmap
def corrmatrix(df):
    ''' This function plots a correlation matrix for a given dataframe '''
    plt.figure(figsize=(14,14))

    corr = df.corr()

    # Generate a mask to only show the bottom triangle
    corr_tri = corr.where(np.tril(np.ones(corr.shape)).astype(np.bool_))

    sns.heatmap(data = corr_tri, center = 0, cmap = "YlGnBu", annot = True, fmt='.1g',vmin=-1);
    plt.title('Correlation of Features')
    plt.show()

corrmatrix(data)
```



❖MULTICOLLINEARITY CHECK

Pada bagian ini, dilakukan pemeriksaan multikolinearitas antara fitur-fitur dalam dataset. Pertama, korelasi antara fitur-fitur numerik dihitung dan kemudian dicari pasangan fitur yang memiliki korelasi tinggi (r > 0.90). Fitur-fitur yang memiliki korelasi tinggi tersebut kemudian dihapus dari dataset untuk mengurangi redundansi. Ini dilakukan dengan membuat mask untuk memfilter pasangan fitur yang memiliki korelasi tinggi, dan kemudian menghapus fitur-fitur tersebut dari dataset. Langkah ini membantu dalam mengurangi masalah multikolinearitas yang dapat mempengaruhi hasil analisis.

```
# Calculate the correlation matrix and take the absolute value
corr_matrix = data.corr().abs()

# Create a True/False mask and apply it
mask = np.triu(np.ones_like(corr_matrix, dtype=bool))
tri_df = corr_matrix.mask(mask)

# List column names of highly correlated features (r > 0.90)
to_drop = [c for c in tri_df.columns if any(tri_df[c] >  0.90)]

data = data.drop(to_drop, axis=1) # Drop the features
data
```

voice_mail_plan	number_vmail_messages	total_day_calls	total_day_charge	total_eve_calls	total_eve_charge	total_night_calls	total_night_charge	total_intl_calls	total_intl_charge	number_customer_service_calls	churn
yes	26	123	27.47	103	16.62	103	11.45	3	3.70	1	no
no	0	114	41.38	110	10.30	104	7.32	5	3.29	0	no
no	0	71	50.90	88	5.26	89	8.86	7	1.78	2	no
no	0	113	28.34	122	12.61	121	8.41	3	2.73	3	no
yes	24	88	37.09	108	29.62	118	9.57	7	2.03	3	no
...	...	...	...	...	...	...	...	...	...	...	...
no	0	89	42.94	91	28.93	67	11.54	5	2.38	1	yes
no	0	70	32.01	88	20.72	79	9.62	6	2.78	0	no
no	0	89	30.24	82	11.15	89	8.38	6	3.11	3	no
no	0	101	29.02	126	16.41	104	5.81	7	1.86	1	no
yes	40	127	40.07	126	18.96	116	13.39	5	2.67	2	no

BINAR

❖FEATURE ENGINEERING

Bagian ini mencakup beberapa teknik feature engineering, yaitu label encoding, one-hot encoding, dan scaling data. Pertama, fitur 'churn' yang memiliki nilai 'yes' atau 'no' diubah menjadi nilai biner menggunakan LabelEncoder. Selanjutnya, fitur-fitur kategorikal lainnya diubah menjadi bentuk one-hot encoding menggunakan pd.get_dummies(). Terakhir, dilakukan scaling pada fitur-fitur numerik menggunakan MinMaxScaler agar memiliki rentang nilai yang seragam. Langkah-langkah ini membantu dalam mempersiapkan data untuk pemodelan, meningkatkan kualitas dan interpretabilitas model yang akan dibangun.

LABEL ENCODING

```
# Convert columns with 'yes' or 'no' to binary using LabelEncoder
label_encoder = LabelEncoder()
data['churn'] = label_encoder.fit_transform(data['churn'])
```

ONE HOT ENCODING

```
data = pd.get_dummies(data,columns = ['state', 'area_code','international_plan','voice_mail_plan'])
data.head()
```

	account_length	number_vmail_messages	total_day_calls	total_day_charge	total_eve_calls	total_eve_ch
0	107	26	123	27.47	103	
1	137	0	114	41.38	110	
2	84	0	71	50.90	88	
3	75	0	113	28.34	122	
4	121	24	88	37.09	108	

SCALING DATA

```
scaler = MinMaxScaler()

def scaling(columns):
    return scaler.fit_transform(data[columns].values.reshape(-1,1))

for i in data.select_dtypes(include=[np.number]).columns:
    data[i] = scaling(i)
data.head()
```

ntl_charge	...	state_WI	state_WV	state_WY	area_code_area_code_408	area_code_area_code_415	area_code_area_code_510	international_plan_no	international_plan_yes	voice_mail_plan_no	voice_mail_plan_yes
0.708520	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	1.0
0.616592	...	0.0	0.0	0.0	0.0	1.0	0.0	1.0	0.0	1.0	0.0
0.278027	...	0.0	0.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0
0.491031	...	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0
0.334081	...	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	1.0

MODELING

Bagian ini berfokus pada proses pemodelan, di mana data dipersiapkan untuk dilatih dan diuji menggunakan berbagai algoritma pembelajaran mesin. Pertama, data dibagi menjadi fitur (X) dan target (y). Kemudian, data dibagi menjadi set pelatihan dan set pengujian menggunakan train_test_split() dengan proporsi pengujian sebesar 25%. Setelah itu, dilakukan oversampling menggunakan SMOTENC untuk menangani ketidakseimbangan kelas pada target 'churn'. Fungsi plot_confusion_matrix() digunakan untuk memvisualisasikan matriks kebingungan (confusion matrix) yang menunjukkan performa model dalam memprediksi kelas target. Setelahnya, dilakukan pemodelan menggunakan DecisionTreeClassifier dan RandomForestClassifier. Untuk setiap model, dilakukan pelatihan pada data pelatihan yang sudah diselaraskan (resampled) dan dilakukan prediksi pada data uji. Selain itu, dilakukan analisis fitur penting (feature importance) menggunakan grafik batang untuk mengetahui kontribusi masing-masing fitur dalam pembuatan keputusan oleh model. Hal ini membantu dalam memahami kinerja dan interpretasi model, serta dalam identifikasi fitur yang paling berpengaruh dalam memprediksi churn.

```
#Defining X and y
X = data.drop("churn", axis=1)
y = data["churn"]
```

```
#splitting the data in to train and test sets
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.25, random_state=123)
```

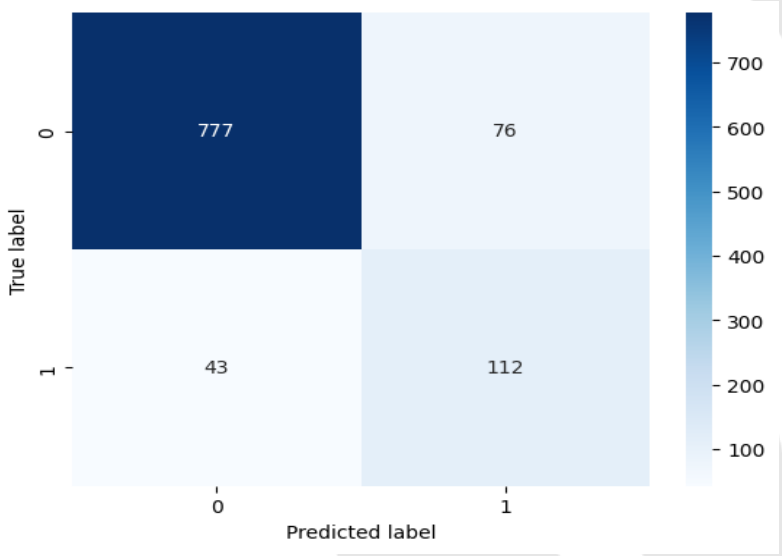
```
#instantiate SMOTENC
from imblearn.over_sampling import SMOTE, SMOTENC

smote = SMOTENC(categorical_features = [1,2],random_state = 123)
resampled_X_train, resampled_y_train = smote.fit_resample(X_train,y_train)
```

```
def plot_confusion_matrix(y_true, y_pred, classes):
    """
    Plots a confusion matrix.
    """
    cm = confusion_matrix(y_true, y_pred)
    plt.figure()
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes, yticklabels=classes)
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.show()
```

BINAR

❖DECISION TREE CLASSIFIER

Bagian ini fokus pada pemodelan menggunakan algoritma Decision Tree Classifier. Pertama, dilakukan instansiasi DecisionTreeClassifier dengan menggunakan random_state=123 untuk memastikan reproduktibilitas hasil. Kemudian, model dipelajari (fit) pada data pelatihan yang telah diselaraskan (resampled) dengan SMOTENC. Setelahnya, model dipakai untuk memprediksi kelas target pada data uji. Selanjutnya, dilakukan visualisasi matriks kebingungan menggunakan fungsi plot_confusion_matrix() dan dicetak laporan klasifikasi menggunakan classification_report(), yang memberikan informasi lebih lanjut tentang performa model. Terakhir, dilakukan analisis fitur penting dengan memplot feature importances dari model untuk memahami kontribusi masing-masing fitur dalam pembuatan keputusan.

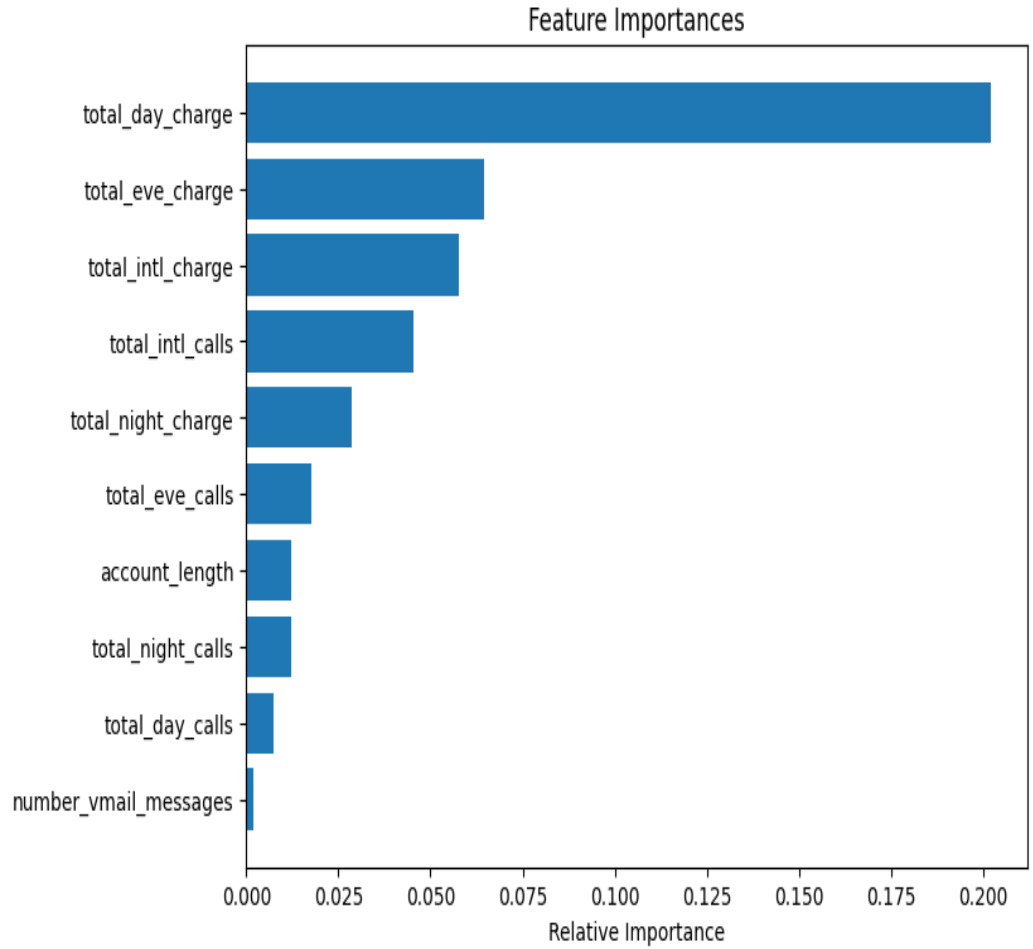



```
#Instantiate DecisionTreeClassifier
dt_clf = DecisionTreeClassifier(random_state=123)

#Fit on the training data
dt_clf.fit(resampled_X_train,resampled_y_train)

#predict on the test set
y_pred_dt = dt_clf.predict(X_test)

plot_confusion_matrix(y_test, y_pred_dt, [0,1])
```

```
print(classification_report(y_test,y_pred_dt))
```

	precision	recall	f1-score	support
0.0	0.95	0.91	0.93	853
1.0	0.60	0.72	0.65	155
accuracy			0.88	1008
macro avg	0.77	0.82	0.79	1008
weighted avg	0.89	0.88	0.89	1008

DECISION TREE CLASSIFIER - Script dan Hasil Running Feature Importances

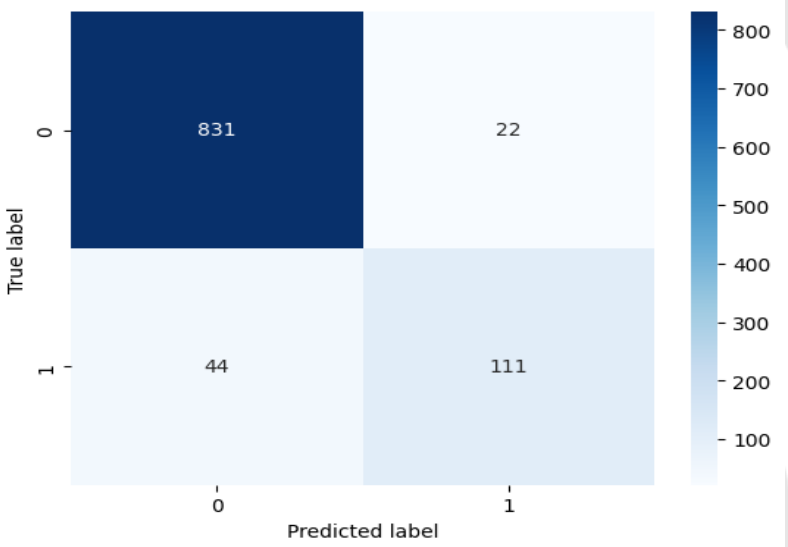



```python
# Feature Importances
feature_names = list(resampled_X_train.columns)
importances = dt_clf.feature_importances_[0:10]
indices = np.argsort(importances)

plt.figure(figsize=(8,6))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], align='center')
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```

BINAR

❖RANDOM FOREST CLASIFIER

Bagian ini mirip dengan pemodelan menggunakan Decision Tree Classifier, namun menggunakan algoritma Random Forest Classifier. Prosesnya mirip, di mana terlebih dahulu dilakukan instansiasi RandomForestClassifier, lalu dilakukan pelatihan pada data pelatihan yang telah diselaraskan, dan dilakukan prediksi pada data uji. Setelahnya, matriks kebingungan dan laporan klasifikasi juga dievaluasi untuk menilai performa model. Selain itu, dilakukan analisis fitur penting untuk memahami kontribusi fitur-fitur dalam model Random Forest. Langkah-langkah ini membantu dalam membandingkan performa dua jenis model yang berbeda dan dalam mengidentifikasi fitur yang paling berpengaruh dalam memprediksi churn.




```
#Instantiate the classifier
rf_clf= RandomForestClassifier(random_state=123)

#Fit on the training data
rf_clf.fit(resampled_X_train,resampled_y_train)
```

```
RandomForestClassifier(random_state=123)
```

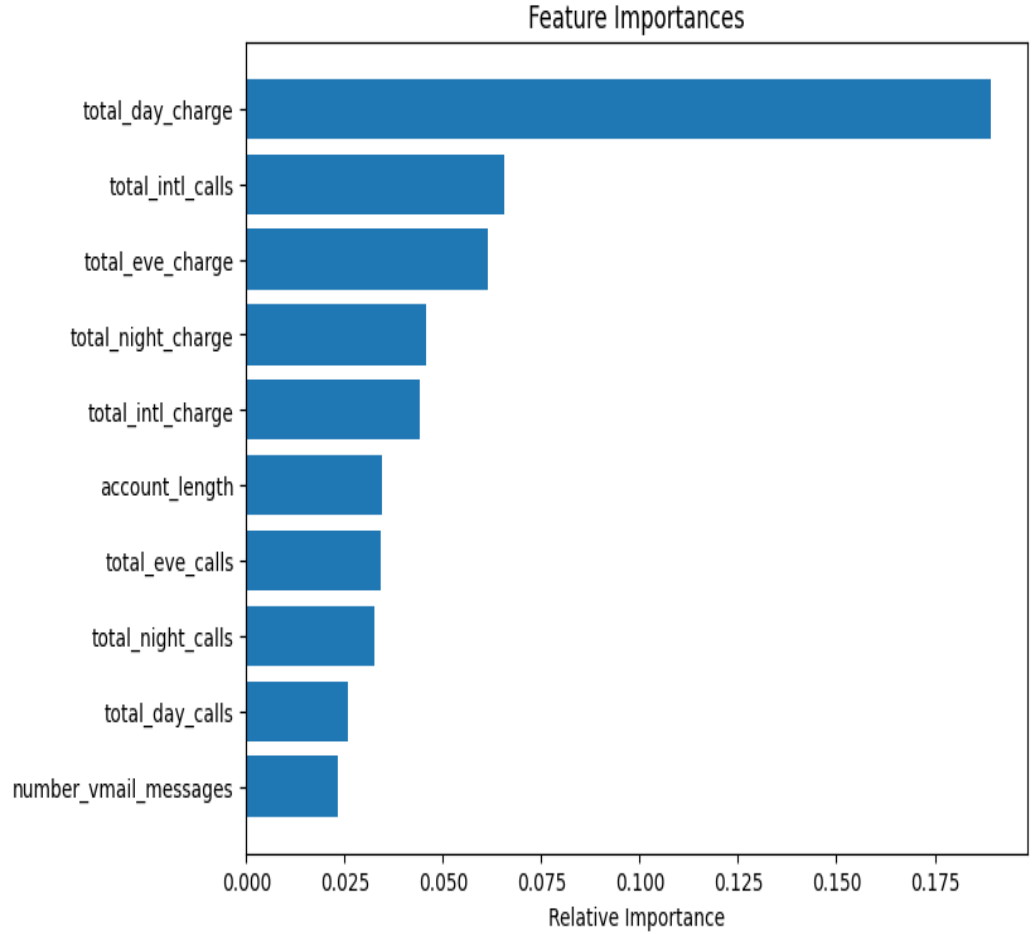
```
#predict on the test data
y_pred_rf = rf_clf.predict(X_test)
```

```
plot_confusion_matrix(y_test, y_pred_rf, [0,1])
```

```
print(classification_report(y_test,y_pred_rf))
```

	precision	recall	f1-score	support
0.0	0.95	0.97	0.96	853
1.0	0.83	0.72	0.77	155
accuracy			0.93	1008
macro avg	0.89	0.85	0.87	1008
weighted avg	0.93	0.93	0.93	1008

RANDOM FOREST CLASIFIER - Script dan Hasil Running Feature Importances




```
feature_names = list(resampled_X_train.columns)
importances = rf_clf.feature_importances_[0:10]
indices = np.argsort(importances)

plt.figure(figsize=(8,6))
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], align='center')
plt.yticks(range(len(indices)), [feature_names[i] for i in indices])
plt.xlabel('Relative Importance')
plt.show()
```

BINAR

❖MODEL COMPARISON - RECALL SCORE

Dalam bagian ini, dilakukan perbandingan performa model berdasarkan skor recall. Dua jenis model, yaitu RandomForestClassifier dan DecisionTreeClassifier, diinisialisasi dan dilatih pada data pelatihan yang telah diselaraskan menggunakan SMOTENC. Kemudian, model-model tersebut dipakai untuk memprediksi kelas target pada data uji, dan skor recall untuk setiap model dicatat. Hasilnya kemudian disajikan dalam sebuah DataFrame yang menampilkan nama kelasifier dan skor recall masing-masing model.

```
np.random.seed(123)

classifiers = [RandomForestClassifier(),
               DecisionTreeClassifier()]

# Define a result table as a DataFrame
result_table = pd.DataFrame(columns=['classifiers', 'recall'])

# Train the models and record the results
for cls in classifiers:
    model = cls.fit(resampled_X_train, resampled_y_train)
    y_pred = model.predict(X_test)

    recall = recall_score(y_test, y_pred)

    result_table = result_table.append({'classifiers': cls.__class__.__name__,
                                        'recall': recall}, ignore_index=True)

# Set name of the classifiers as index labels
result_table.set_index('classifiers', inplace=True)

result_table
```

classifiers	recall
RandomForestClassifier	0.716129
DecisionTreeClassifier	0.709677

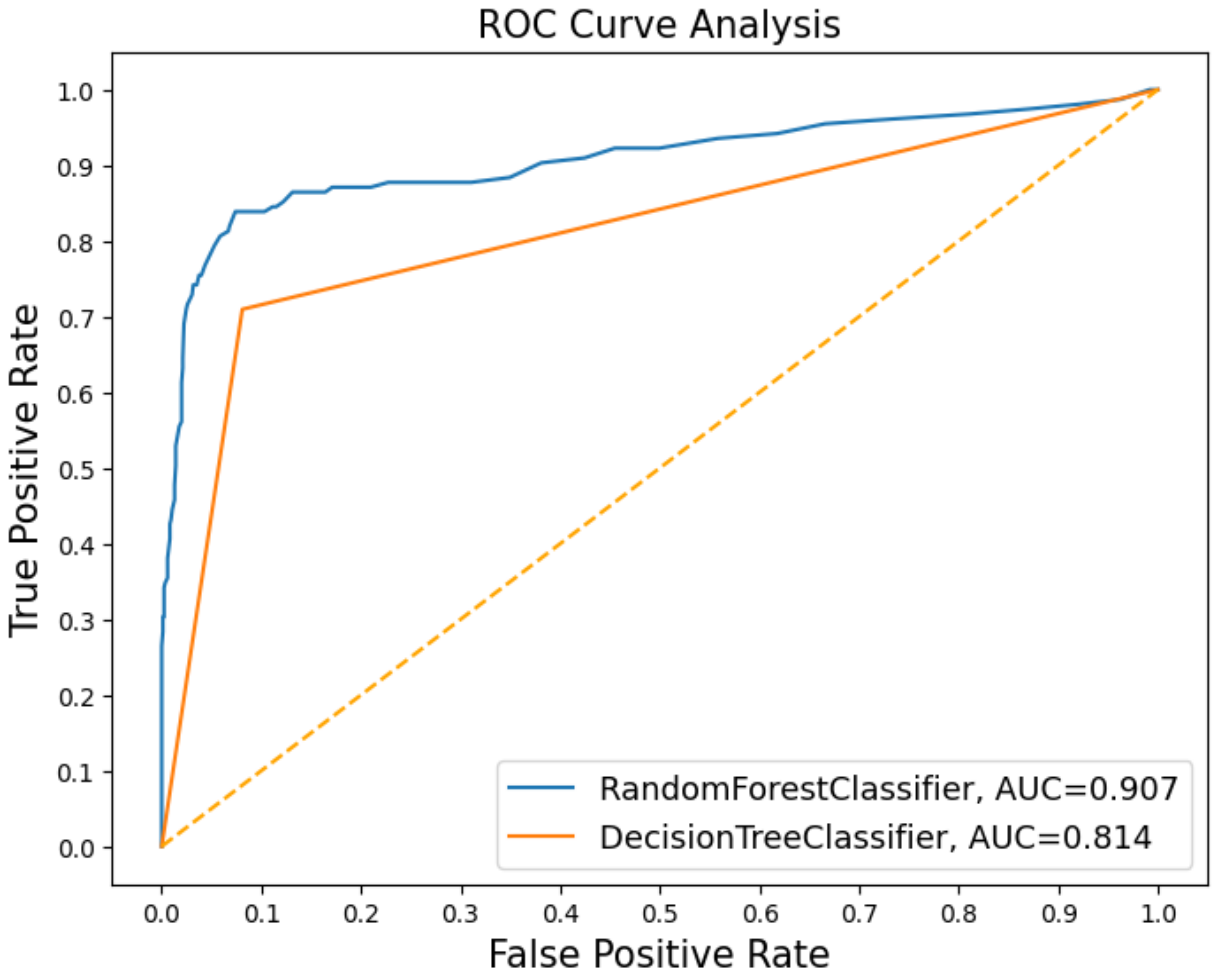
BINAR

❖MODEL COMPARISON - ROC CURVE

Bagian ini bertujuan untuk membandingkan performa model menggunakan kurva ROC (Receiver Operating Characteristic). Model-model RandomForestClassifier dan DecisionTreeClassifier dilatih dan dipakai untuk memprediksi probabilitas kelas target pada data uji. Selanjutnya, dilakukan perhitungan nilai False Positive Rate (FPR) dan True Positive Rate (TPR) untuk setiap model, yang kemudian digunakan untuk menggambar kurva ROC. Kurva ROC digunakan untuk menggambarkan hubungan antara sensitivitas dan spesifisitas model. Hasilnya disajikan dalam sebuah plot yang menunjukkan kurva ROC untuk setiap model beserta nilai AUC (Area Under Curve) sebagai metrik evaluasi.

```
np.random.seed(123)
classifiers = [RandomForestClassifier(),
               DecisionTreeClassifier()]


# Define a result table as a DataFrame
result_table = pd.DataFrame(columns=['classifiers', 'fpr','tpr','auc'])

# Train the models and record the results
for cls in classifiers:
    model = cls.fit(resampled_X_train, resampled_y_train)
    yproba = model.predict_proba(X_test)[::,1]

    fpr, tpr, _ = roc_curve(y_test,  yproba)
    auc = roc_auc_score(y_test, yproba)

    result_table = result_table.append({'classifiers':cls.__class__.__name__,
                                        'fpr':fpr,
                                        'tpr':tpr,
                                        'auc':auc}, ignore_index=True)

# Set name of the classifiers as index labels
result_table.set_index('classifiers', inplace=True)

fig = plt.figure(figsize=(8,6))

for i in result_table.index:
    plt.plot(result_table.loc[i]['fpr'],
             result_table.loc[i]['tpr'],
             label="{}, AUC={:.3f}".format(i, result_table.loc[i]['auc']))

plt.plot([0,1], [0,1], color='orange', linestyle='--')

plt.xticks(np.arange(0.0, 1.1, step=0.1))
plt.xlabel("False Positive Rate", fontsize=15)
plt.yticks(np.arange(0.0, 1.1, step=0.1))
plt.ylabel("True Positive Rate", fontsize=15)

plt.title('ROC Curve Analysis', fontsize=15)
plt.legend(prop={'size':13}, loc='lower right')

plt.show()
```

MODEL COMPARISON - ROC CURVE



❖TUNING RANDOM FOREST

Pada bagian ini, dilakukan penyetelan parameter (hyperparameter tuning) untuk model RandomForestClassifier menggunakan GridSearchCV. Sejumlah kombinasi parameter diuji untuk menemukan kombinasi terbaik yang menghasilkan model dengan performa optimal. Setelah penyetelan, model yang dihasilkan kemudian dipakai untuk melatih ulang data pelatihan dan dievaluasi performanya pada data uji.

```
# Tune Random Forest
param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf = RandomForestClassifier(random_state=42)
grid_search_rf = GridSearchCV(estimator=rf, param_grid=param_grid_rf, cv=5)
grid_search_rf.fit(X_train, y_train)

print("Best Parameters for Random Forest:", grid_search_rf.best_params_)
```

```
Best Parameters for Random Forest: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 100}
```

```
# Train RF
best_rf = grid_search_rf.best_estimator_
best_rf.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(min_samples_split=5, random_state=42)
```

BINAR

❖TUNING DECISIONTREE CLASSIFIER

Bagian ini mirip dengan sebelumnya, namun dilakukan untuk model DecisionTreeClassifier. Parameter-parameter model diuji untuk menemukan kombinasi terbaik yang menghasilkan performa optimal. Setelah penyetelan, model terbaik dipakai untuk melatih ulang data pelatihan dan dievaluasi pada data uji.

```
# Tune Decisiontree
param_grid_dt = {
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

dt = DecisionTreeClassifier(random_state=42)
grid_search_dt = GridSearchCV(estimator=dt, param_grid=param_grid_dt, cv=5)
grid_search_dt.fit(X_train, y_train)

print("Best Parameters for Decision Tree:", grid_search_dt.best_params_)
```

```
Best Parameters for Decision Tree: {'max_depth': 10, 'min_samples_leaf': 2, 'min_samples_split': 10}
```

```
# Train Decisiontree
best_dt = grid_search_dt.best_estimator_
best_dt.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(max_depth=10, min_samples_leaf=2, min_samples_split=10,
                       random_state=42)
```

MODEL ACCURACY AND ROC

Bagian ini berfokus pada evaluasi akurasi dan kurva ROC dari model-model yang telah dituning. Pertama, dilakukan prediksi pada data uji menggunakan model RandomForestClassifier dan DecisionTreeClassifier yang telah dituning. Selanjutnya, dihitung nilai akurasi dari kedua model tersebut. Hasil akurasi kemudian dicetak sebagai output. Selain itu, dilakukan juga pencetakan laporan klasifikasi (classification report) yang memberikan informasi lebih rinci tentang performa model dalam memprediksi setiap kelas target. Terakhir, dilakukan visualisasi kurva ROC untuk membandingkan performa kedua model dan memvisualisasikan feature importance dari model DecisionTreeClassifier yang dituning untuk memahami kontribusi masing-masing fitur dalam pembuatan keputusan model.

```
# Predictions
y_pred_rf = best_rf.predict(X_test)
y_pred_dt = best_dt.predict(X_test)

# Accuracy
accuracy_rf = accuracy_score(y_test, y_pred_rf)
accuracy_dt = accuracy_score(y_test, y_pred_dt)

print("Accuracy of Random Forest:", accuracy_rf)
print("Accuracy of Decision Tree:", accuracy_dt)
```

```
Accuracy of Random Forest: 0.952912019826518
Accuracy of Decision Tree: 0.952912019826518
```

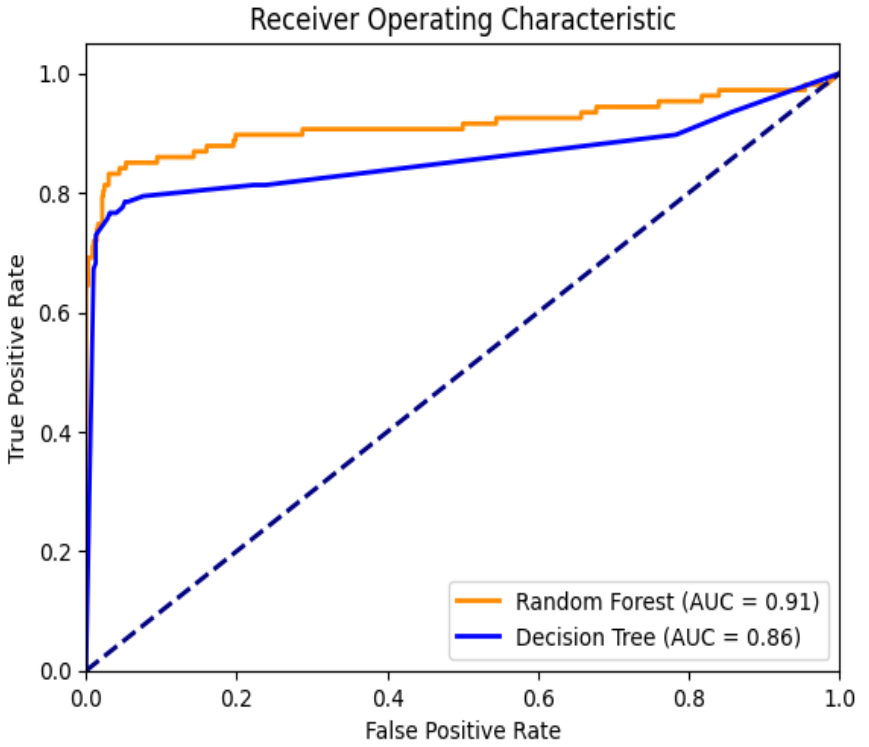
```
# Klasifikasi Report
print("Classification Report for Random Forest:")
print(classification_report(y_test, y_pred_rf))

print("Classification Report for Decision Tree:")
print(classification_report(y_test, y_pred_dt))
```

```
Classification Report for Random Forest:
              precision    recall  f1-score   support

         0.0       0.95      1.00      0.97       700
         1.0       0.97      0.66      0.79       107

    accuracy                           0.95       807
   macro avg       0.96      0.83      0.88       807
weighted avg       0.95      0.95      0.95       807

Classification Report for Decision Tree:
              precision    recall  f1-score   support

         0.0       0.96      0.99      0.97       700
         1.0       0.90      0.73      0.80       107

    accuracy                           0.95       807
   macro avg       0.93      0.86      0.89       807
weighted avg       0.95      0.95      0.95       807
```

MODEL ACCURACY AND ROC

BINAR

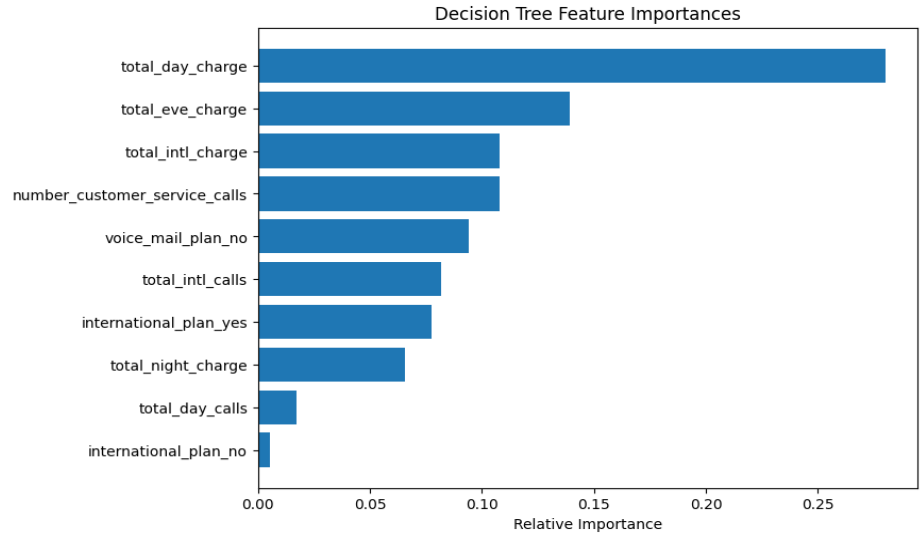
```
# Random Forest
y_prob_rf = best_rf.predict_proba(X_test)[:, 1]
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_prob_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)

# Decision Tree
y_prob_dt = best_dt.predict_proba(X_test)[:, 1]
fpr_dt, tpr_dt, _ = roc_curve(y_test, y_prob_dt)
roc_auc_dt = auc(fpr_dt, tpr_dt)

# Plot ROC Curve
plt.figure()
plt.plot(fpr_rf, tpr_rf, color='darkorange', lw=2, label='Random Forest (AUC = %0.2f)' % roc_auc_rf)
plt.plot(fpr_dt, tpr_dt, color='blue', lw=2, label='Decision Tree (AUC = %0.2f)' % roc_auc_dt)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



BINAR

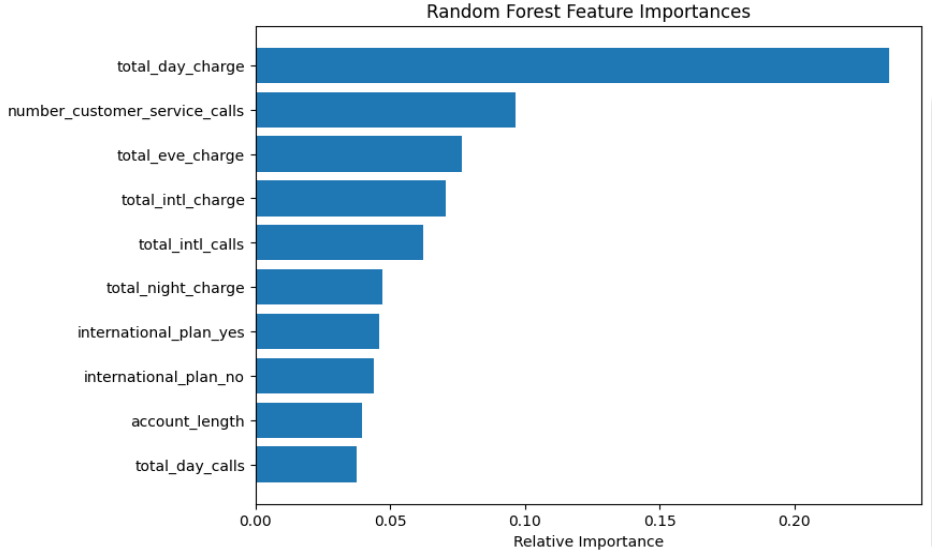
Decision Tree Feature Importance Visualization

```python
# Decision Tree Feature Importance Visualization
feature_names_dt = list(X_train.columns)
importances_dt = best_dt.feature_importances_
indices_dt = np.argsort(importances_dt)[-10:]  # Top 10 features

plt.figure(figsize=(8, 6))
plt.title('Decision Tree Feature Importances')
plt.barh(range(len(indices_dt)), importances_dt[indices_dt], align='center')
plt.yticks(range(len(indices_dt)), [feature_names_dt[i] for i in indices_dt])
plt.xlabel('Relative Importance')
plt.show()
```



Random Forest Feature Importance Visualization

```python
# Random Forest Feature Importance Visualization
feature_names_rf = list(X_train.columns)
importances_rf = best_rf.feature_importances_
indices_rf = np.argsort(importances_rf)[-10:]  # Top 10 features

plt.figure(figsize=(8, 6))
plt.title('Random Forest Feature Importances')
plt.barh(range(len(indices_rf)), importances_rf[indices_rf], align='center')
plt.yticks(range(len(indices_rf)), [feature_names_rf[i] for i in indices_rf])
plt.xlabel('Relative Importance')
plt.show()
```



CONCLUSION

Skor penarikan kembali pengklasifikasi adalah 73%. Meskipun model ini masih merupakan model prediktif yang baik.

Recomendations

- Tawarkan diskon atau penawaran promosi kepada pelanggan di kode area 415 dan 510, karena area ini memiliki tingkat churn yang lebih tinggi. Hal ini dapat membantu memberi insentif kepada pelanggan untuk tetap bersama perusahaan

- Meningkatkan kualitas layanan pelanggan dan mengurangi jumlah panggilan layanan pelanggan. Meningkatkan program pelatihan bagi perwakilan layanan pelanggan untuk memastikan penyelesaian masalah pelanggan dengan cepat dan efektif, sehingga menghasilkan kepuasan pelanggan yang lebih tinggi dan mengurangi churn.

- Evaluasi struktur harga untuk tarif siang, malam, malam, dan internasional. Pertimbangkan untuk menyesuaikan paket harga atau memperkenalkan paket diskon untuk mengatasi biaya lebih tinggi yang terkait dengan pelanggan yang melakukan churn.

- Fokus pada strategi retensi pelanggan di negara-negara dengan tingkat churn yang lebih tinggi, seperti Texas, New Jersey, Maryland, Miami, dan New York. Hal ini dapat melibatkan kampanye pemasaran yang ditargetkan, penawaran yang dipersonalisasi, atau peningkatan dukungan pelanggan yang disesuaikan dengan kebutuhan dan preferensi spesifik pelanggan di negara bagian tersebut.

- Meningkatkan proposisi nilai rencana pesan suara untuk meningkatkan adopsi di kalangan pelanggan. Soroti manfaat dan kenyamanan layanan pesan suara, dan pertimbangkan untuk menawarkan fitur atau diskon tambahan untuk mendorong pelanggan mendaftar.

Report Pembagian Tugas

Report Pembagian Tugas

Nama	Tasklist/Deliverable
Hanosi Wazri	Data Preparation, Modeling, Model Evaluation
Inocentius Reynaldo Bhoka Tola	Data Preparation, Modeling, Model Evaluation

BINAR

Thank You