[Kelompok 12] - Final Presentation

[Nama Course : Data Science]

Challenge 1

Terdapat 2 hal penting yang terdapat pada challenge 1 yaitu:

1) Melakukan SQL Query
2) Membuat Dashboard

1) SQL Query

1. Jumlah total kasus Covid-19 aktif yang baru di setiap provinsi lalu diurutkan berdasarkan jumlah kasus paling besar

```sql
SELECT
  Province,
  SUM(New_Active_Cases) AS total_new_active_cases
FROM
  `fifth-flame-414914.kasus_covid_19_id.kasus_covid_19`
WHERE Location = 'DKI Jakarta'
GROUP BY
  Province
ORDER BY
  total_new_active_cases DESC;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON | EXECUTION DETAILS

Row	Province	total_new_active_cases
1	DKI Jakarta	10922

2. Mengambil 2 location iso code yang memiliki jumlah total kematian karena Covid-19 paling sedikit

```sql
--kami menggunakan New_Deaths karena data tersebut adalah data angka kematian baru tiap harinya, sehingga kami bisa mendapatkan total kematian hanya dijangka waktu maret 2020 hingga september 2022
--sebenernya ketika kami menggunakan where data tidak berpengaruh, karna kita ambil data yg terendah aja dan IDN bukan yang terendah
SELECT
  Location_ISO_Code,
  SUM(New_Deaths) as sum_total_deaths
FROM `fifth-flame-414914.kasus_covid_19_id.kasus_covid_19`
WHERE Location_ISO_Code <> 'IDN'
GROUP BY Location_ISO_Code
ORDER BY sum_total_deaths
LIMIT 2;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	Location_ISO_Code	sum_total_deaths
1	ID-MA	294
2	ID-MU	334

3. Data tentang tanggal-tanggal ketika rate kasus recovered di Indonesia paling tinggi beserta jumlah ratenya tinggi

```sql
--Kita memutuskan menggunakan LIMIT untuk menentukan nilai paling maksimal
SELECT
  Date,
  MAX(Case_Recovered_Rate) AS Highest_Recovery_Rate
FROM
  `my-project-challenge-binar.dataset_covid19indo.covid`
GROUP BY
  Date
ORDER BY
  Highest_Recovery_Rate DESC
LIMIT
  1;
```

Query results

JOB INFORMATION | RESULTS | CHART

Row	Date	Highest_Recovery_Rate
1	2020-03-06	111.0

4. Total case fatality rate dari masing-masing location iso code yang diurutkan dari data yang paling rendah

```sql
SELECT
  Location_ISO_Code,
  SUM(New_Deaths) / SUM(New_Cases) as total_case_fatality_rate,
FROM `fifth-flame-414914.kasus_covid_19_id.kasus_covid_19`
GROUP BY Location_ISO_Code
ORDER BY total_case_fatality_rate;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSON

Row	Location_ISO_Code	total_case_fatality_ra
1	ID-BT	0.008820666417...
2	ID-JK	0.010968697476...
3	ID-PA	0.011596931520...
4	ID-PB	0.011936586882...
5	ID-JB	0.013578068569...
6	ID-MA	0.015691716481...
7	ID-NT	0.016173277551...
8	ID-SN	0.017197945935...
9	ID-KB	0.017224296928...
10	ID-BE	0.017893257464...
11	ID-KU	0.018935640839...
12	ID-SU	0.020696687774...
13	ID-SG	0.022146109835...
14	ID-SB	0.022658639143...

4. Total case recovered rate dari masing-masing location iso code yang diurutkan dari data yang paling rendah

```
SELECT
  Location_ISO_Code,
  SUM(New_Recovered) / SUM(New_Cases) as total_case_recovered_rate
FROM `fifth-flame-414914.kasus_covid_19_id.kasus_covid_19`
GROUP BY Location_ISO_Code
ORDER BY total_case_recovered_rate;
```

Query results

JOB INFORMATION | RESULTS | CHART | JSO

Row	Location_ISO_Code	total_case_recovered
1	ID-LA	0.941551301583...
2	ID-JT	0.945156338141...
3	ID-JI	0.945359697041...
4	ID-AC	0.947181979199...
5	ID-SS	0.955120562544...
6	ID-GO	0.962870045158...
7	ID-SA	0.966325563767...
8	ID-KS	0.968722849695...
9	ID-BA	0.968800762448...
10	ID-RI	0.969367433572...
11	ID-KT	0.969373207138...
12	ID-ST	0.970163177793...

5. Data tentang tanggal-tanggal saat total kasus Covid-19 mulai menyentuh angka 30.000-an (30.001)

```
SELECT
  Date,
  Total_Cases
FROM
  `fifth-flame-414914.kasus_covid_19_id.kasus_covid_19`
WHERE
  Total_Cases > 30000;
```

Query results

JOB INFORMATION | RESULTS | CHAR

Row	Date	Total_Cases
1	2020-06-06	30514
2	2020-06-07	31186
3	2020-06-08	32033
4	2020-06-09	33075
5	2020-06-10	34316
6	2020-06-11	35295
7	2020-06-12	36406
8	2020-06-13	37420

6. Jumlah data yang tercatat ketika kasus Covid-19 lebih dari atau sama dengan 30.000

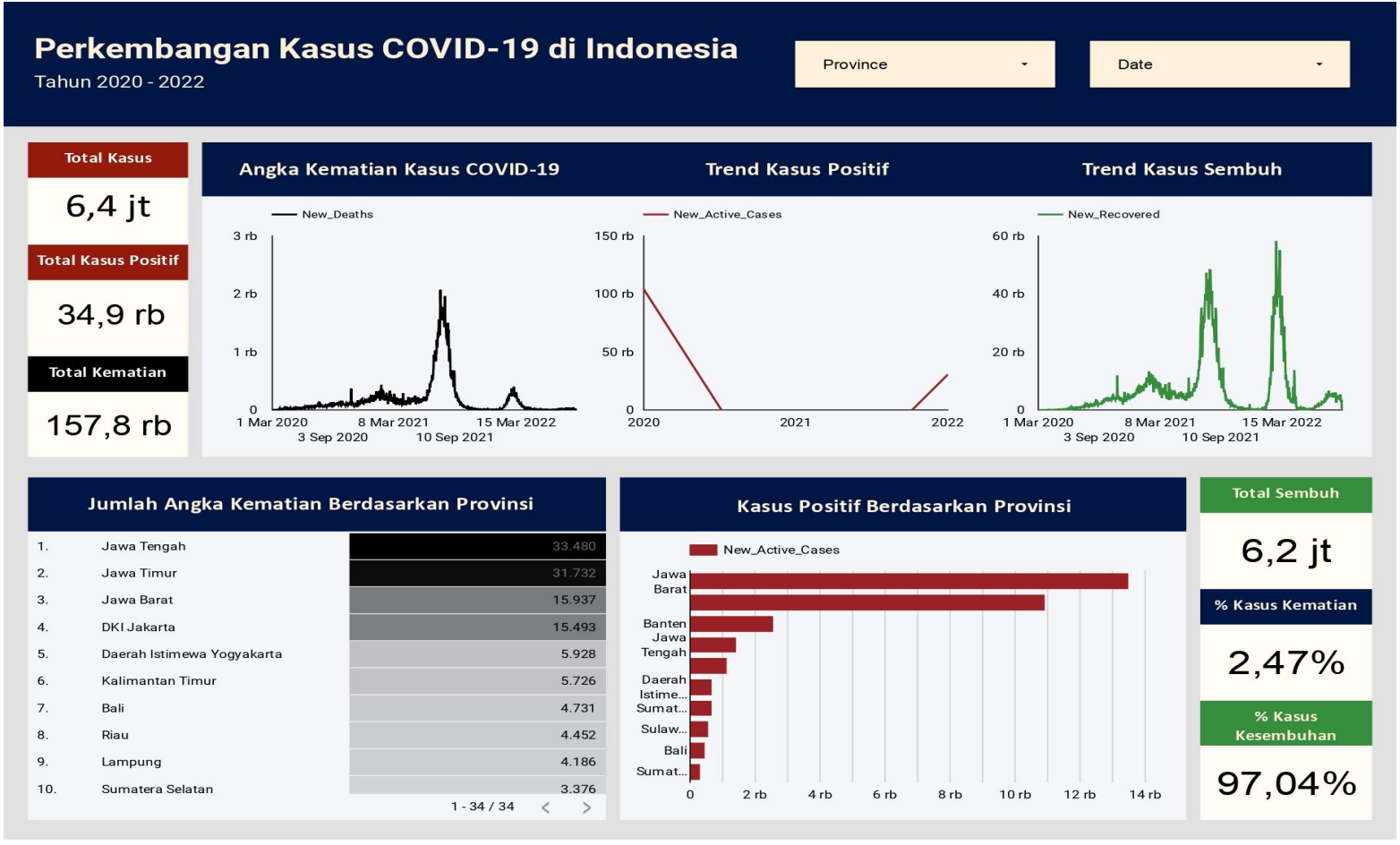
```
--Jumlah data menggunakan New_Cases untuk mendapatkan kasus covid-19 tiap harinya lebih dari atau sama dengan 30.000
SELECT
  COUNT(New_Cases) as total_recorded_cases
FROM `fifth-flame-414914.kasus_covid_19_id.kasus_covid_19`
WHERE New_Cases >= 30000;
```

Query results

JOB INFORMATION | RESULTS

Row	total_recorded_cases
1	61

2) Dashboard


Perkembangan Kasus COVID-19 di Indonesia
Tahun 2020 - 2022
Province
Date
Total Kasus
6,4 jt
Total Kasus Positif
34,9 rb
Total Kematian
157,8 rb
Angka Kematian Kasus COVID-19
New_Deaths
3 rb
2 rb
1 rb
0
1 Mar 2020
3 Sep 2020
8 Mar 2021
10 Sep 2021
15 Mar 2022
Trend Kasus Positif
New_Active_Cases
150 rb
100 rb
50 rb
0
2020
2021
2022
Trend Kasus Sembuh
New_Recovered
60 rb
40 rb
20 rb
0
1 Mar 2020
3 Sep 2020
8 Mar 2021
10 Sep 2021
15 Mar 2022
Jumlah Angka Kematian Berdasarkan Provinsi
1. Jawa Tengah 33.480
2. Jawa Timur 31.732
3. Jawa Barat 15.937
4. DKI Jakarta 15.493
5. Daerah Istimewa Yogyakarta 5.928
6. Kalimantan Timur 5.726
7. Bali 4.731
8. Riau 4.452
9. Lampung 4.186
10. Sumatera Selatan 3.376
1 - 34 / 34
Kasus Positif Berdasarkan Provinsi
New_Active_Cases
Jawa Barat
Banten
Jawa Tengah
Daerah Istime...
Sumat...
Sulaw...
Bali
Sumat...
0
2 rb
4 rb
6 rb
8 rb
10 rb
12 rb
14 rb
Total Sembuh
6,2 jt
% Kasus Kematian
2,47%
% Kasus Kesembuhan
97,04%

2) Dashboard

Pada seluruh dashboard kami menggunakan data:

- New_Deaths
- New_Active_Cases
- New_Recovered

Dan dua rumus yaitu :

- SUM(New_Deaths) / SUM(New_Cases) = %Kasus Kematian
- SUM(New_Recovered) / SUM(New_Cases) = %Kasus Kesembuhan

Penjelasan Lebih Lanjut:

- Pada grafik “Trend Kasus Positif” pada Tahun 2021 grafik menunjukkan penurunan hingga dibawah 0
- Kami melakukan pemeriksaan data menggunakan Query:

```
SELECT
  New_Active_Cases, Date
FROM
`fifth-flame-414914.kasus_covid_19_id.kasus_covid_19`
where Date >= '2021-01-01' and Date < '2022-01-01'
```

- Hal ini disebabkan :
  Terdapat data negatif mulai dari puluhan hingga ribuan sehingga menyebabkan grafik pada tahun 2021 turun hingga >0

Row	New_Active_Cases	Date
38	-142	2021-02-07
39	-5003	2021-02-08
40	-1937	2021-02-09
41	-935	2021-02-10
42	-1924	2021-02-11
43	-1406	2021-02-12
44	-3355	2021-02-13
45	-2719	2021-02-14
46	-514	2021-02-15

Pada seluruh dashboard kami menggunakan data:

- New_Deaths
- New_Active_Cases
- New_Recovered Dan dua rumus yaitu :
- SUM(New_Deaths) / SUM(New_Cases) = %Kasus Kematian
- SUM(New_Recovered) / SUM(New_Cases) = %Kasus Kesembuhan

Penjelasan Lebih Lanjut:

- Pada grafik "Trend Kasus Positif" pada Tahun 2021 grafik menunjukkan penurunan hingga dibawah 0
- Kami melakukan pemeriksaan data menggunakan

Query:

```
SELECT
  New_Active_Cases, Date
FROM
`fifth-flame-414914.kasus_covid_19_id.kasus_covid_19`
where Date >= '2021-01-01' and Date < '2022-01-01'
```

- Hal ini disebabkan :
  Terdapat data negatif mulai dari puluhan hingga ribuan sehingga menyebabkan grafik pada tahun 2021 turun hingga >0

Row	New_Active_Cases	Date
38	-142	2021-02-07
39	-5003	2021-02-08
40	-1937	2021-02-09
41	-935	2021-02-10
42	-1924	2021-02-11
43	-1406	2021-02-12
44	-3355	2021-02-13
45	-2719	2021-02-14
46	-514	2021-02-15

Challenge 2

FILE PENGUMPULAN:

Berikut file hasil analisis:

Link Google Colabs

Tahapan Machine Learning

Pada kesempatan kali ini kami akan melakukan studi kasus untuk memprediksi customer churn di perusahaan telekomunikasi.

Berikut dataset yang digunakan:
Data Train
Data Test

A. Import Data

1. Upload File

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

2. Baca Dataset Train

```
df = pd.read_csv('/content/Data Train (2).csv')
```

3. Import Library Yang Dibutuhkan

```
#import library
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from scipy import stats

#import library scikit-learn untuk proses statistika san meachine learning
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler

from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.metrics import f1_score
from sklearn.metrics import log_loss

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectKBest, chi2

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV
```

4. Melihat Informasi dari dataset

```
df.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
```

5. Melihat 5 Data teratas (dari sini kita sudah bisa melihat variabel dependentnya adalah 'churn'

```
df.head()
#churn = dependent
```

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	total_eve_minutes	total_eve_c
0	OH	107	area_code_415	no	yes	26	161.6	123	27.47	195.5	
1	NJ	137	area_code_415	no	no	0	243.4	114	41.38	121.2	
2	OH	84	area_code_408	yes	no	0	299.4	71	50.90	61.9	
3	OK	75	area_code_415	yes	no	0	166.7	113	28.34	148.3	
4	MA	121	area_code_510	no	yes	24	218.2	88	37.09	348.5	

B. Data Cleaning

1. Missing Value
2. Duplicated Value

```
df.isna().sum()
# Data bersih tidak memiliki null
# Data tidak memerlukan langkah fillna
```

```
df.duplicated().sum()
# data bersih tanpa duplicate
```

C. Data Prepocessing

1. Kategorisasi Data

```
categorical_col = ['state', 'area_code', 'international_plan', 'voice_mail_plan', 'churn']
numeric_col = ['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']
```

2. Unik Value

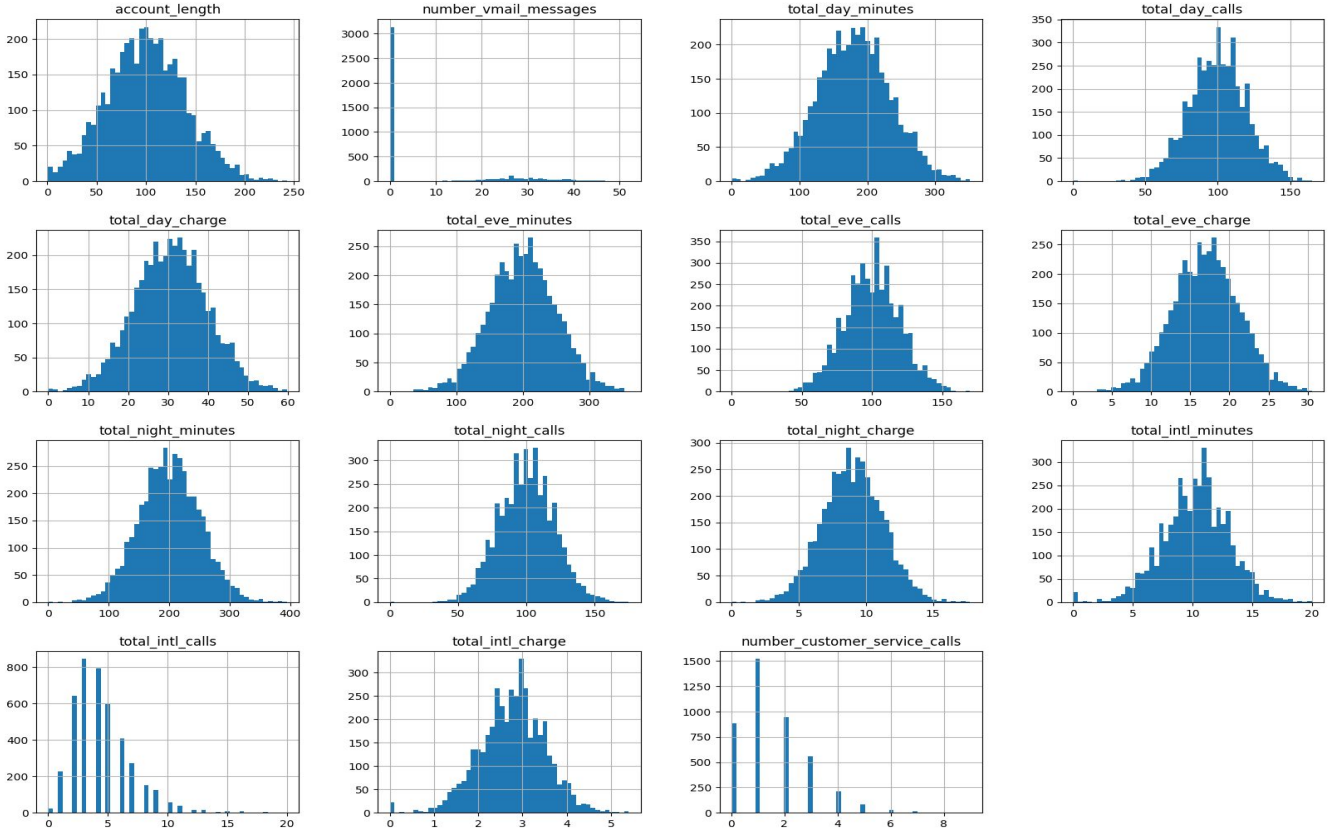
```
df.nunique().sort_values()
```

3. Cek Outlier

```
df[numeric_col].hist(bins=50, figsize=(20,15))
```

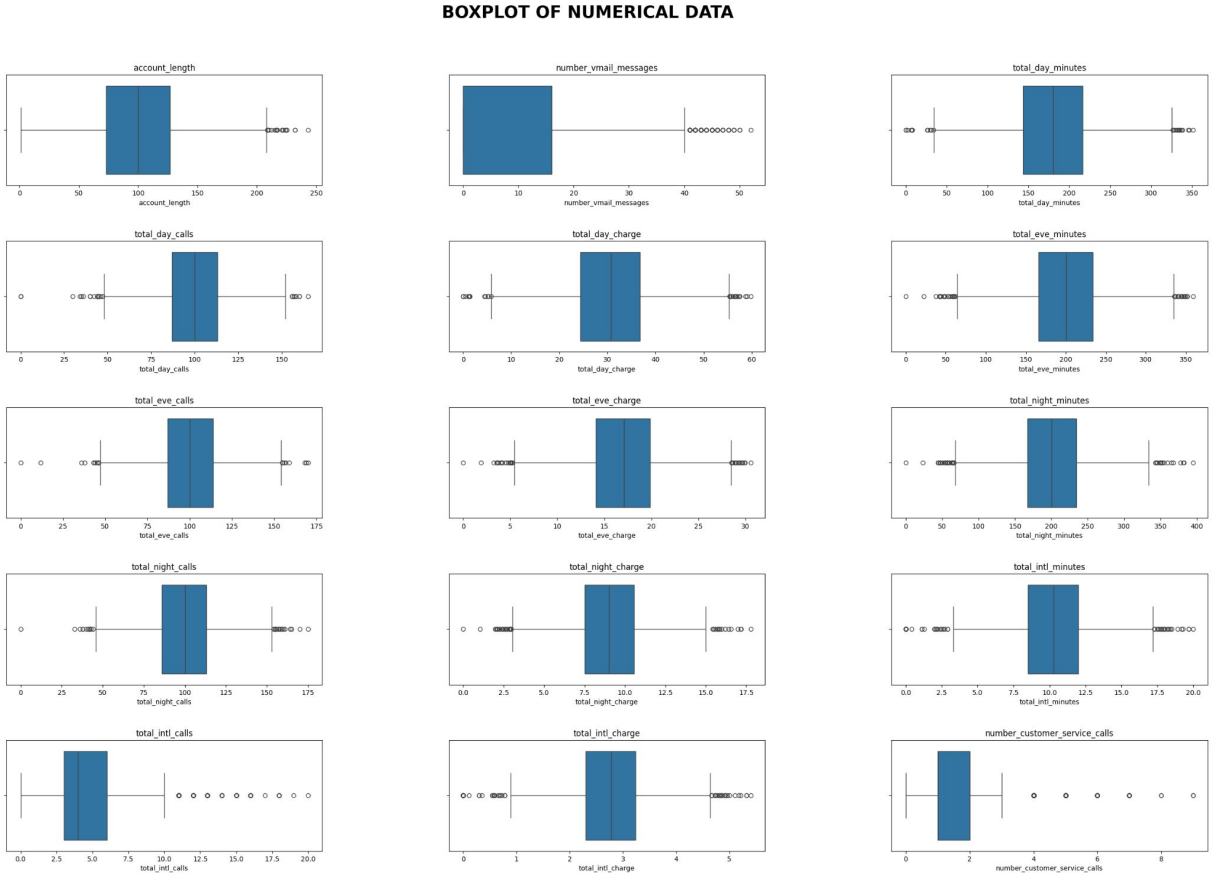
```
#Boxplot untuk mengindentifikasi outliers
fig = plt.figure(figsize=[32,24])
fig.suptitle('BOXPLOT OF NUMERICAL DATA', fontsize=25, fontweight='bold')
fig.subplots_adjust(top=0.92);
fig.subplots_adjust(hspace=0.5, wspace=0.4);
for i ,column in enumerate(numeric_col):
    ax1 = fig.add_subplot(6,3, i+1);
    ax1 = sns.boxplot(data = df, x=column);

    ax1.set_title(f'{column}')
    ax1.set xlabel(f'{column}')
```



semua terdistribusi normal kecuali total_intl_calls, number_customer_service_calls, dan number_vmail_messages

Kita juga menggunakan boxplot untuk mengidentifikasi outlier



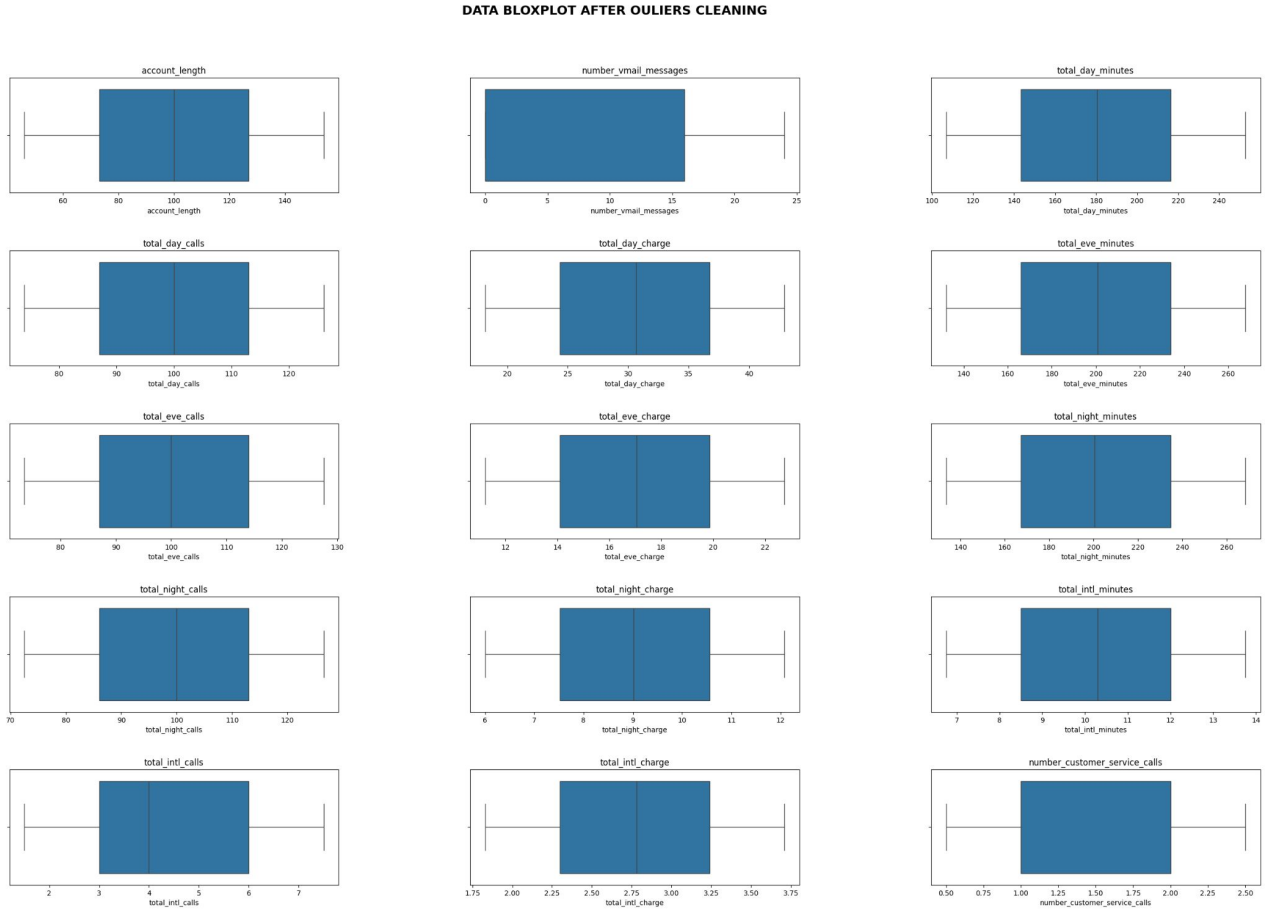
4. Mengatasi Otliers

```
#Penangan jika numerical data terdapat Outliers menggunakan IQR
# Dan menggunakan capping (mengubah nilai outliers dengan nilai upper / lower limit)
dict = {}
for col in numeric_col:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR  = Q3 - Q1
    upper_limit = Q1 + 1.5 * IQR
    lower_limit = Q3 - 1.5 * IQR
    dict['upper_limit' + col] = upper_limit
    dict['lower_limit' + col] = lower_limit
for col in numeric_col:
    df[col] = np.where(
        df[col] > dict['upper_limit' + col],
        dict['upper_limit' + col],
        np.where(
            df[col] < dict['lower_limit' + col],
            dict['lower_limit' + col],
            df[col]
        )
    )
```

```
#Boxplot setelah cleaning outlier
fig = plt.figure(figsize=[32,24])
fig.suptitle('DATA BLOXPLOT AFTER OULIERS CLEANING', fontsize=18, fontweight='bold')
fig.subplots_adjust(top=0.92);
fig.subplots_adjust(hspace=0.5, wspace=0.4);
for i ,col in enumerate(numeric_col):
    ax1 = fig.add_subplot(6,3, i+1);
    ax1 = sns.boxplot(data = df, x=col);

    ax1.set_title(f'{col}')
    ax1.set_xlabel(f'{col}')
```

Boxplot Setelah Outliers ditangani



D. Exploratory Data Analysis

1. Measure Of Central Tendency
```
df.describe()
```
2. Melihat Unik Value
```
df.nunique().sort_values()
```
3. Analisa Multivariat
```
plt.figure(figsize=(30,15))
corr_matrix = df.corr()

sns.heatmap(corr_matrix, annot=True, cmap='viridis')
plt.show()
```
4. Analisa Univariat
```
#Mengelompokkan dataset (kategorikal dan numerikal)
numerical_col = ['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']
categorical_col = ['state', 'area_code', 'international_plan', 'voice_mail_plan', 'churn']
```

Presentase Churn Kostumer

```
# Melihat berapa banyak churn dilakukan oleh pelanggan
df.groupby('churn').size().plot(kind='pie', autopct = '%.1f%%', radius=1)
```

Demografi

```
# area_code, international_plan, voice_mail_plan yang melakukan churn
fig, axs = plt.subplots(1, 3, figsize=(14,5))

# Count plot AREA CODE
sns.countplot(x='area_code', hue='churn', data=df, ax=axs[0])
axs[0].set_title('Churn Area Code')

# Count plot INTERNATIONAL PLAN
sns.countplot(x='international_plan', hue='churn', data=df, ax=axs[1])
axs[1].set_title('Churn International Plan')

# Count plot for VOICE MAIL PLAN
sns.countplot(x='voice_mail_plan', hue='churn', data=df, ax=axs[2])
axs[2].set_title('Churn Voice Mail Plan')

plt.tight_layout()
plt.show()
```

Hubungan Numerical Data dengan Churn

```
categorical_col = ['state', 'area_code', 'international_plan', 'voice_mail_plan', 'churn']
numeric_col = ['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']
```

Features

```
total_durasi= df['total_day_minutes'] + df['total_eve_minutes'] + df['total_night_minutes']
total_charge = df['total_day_charge'] + df['total_eve_charge'] + df['total_night_charge']
total_calls = df['total_day_calls'] + df['total_eve_calls'] + df['total_night_calls']

df['total_durasi'] = total_durasi
df['total_charge'] = total_charge
df['total_calls'] = total_calls
```

Q1 : Bagaimana pengaruh lama panggilan terhadap harga

A1: Positif Correlation

```
[41] plt.figure(figsize=(5,5))
     plt.scatter(x='total_durasi',y='total_charge',data=df, color='yellow')
     plt.xlabel('Durasi')
     plt.ylabel('Price')
```

```
Text(0, 0.5, 'Price')
```

Insight

dapat dilakukan promo panggilan, atau gratis panggilan berdasarkan durasi

```
[42] sns.histplot(df.total_durasi,bins=10)
```

Q2 : Pengaruh jumlah panggilan terhadap harga

A2 : No Correlation

```
[43] plt.figure(figsize=(5,5))
     plt.scatter(x='total_calls',y='total_charge',data=df, color='yellow')
     plt.xlabel('Panggilan')
     plt.ylabel('Price')

     Text(0, 0.5, 'Price')
```

```
[44] sns.histplot(df.total_calls,bins=10)
```

`Total calls` tidak memiliki korelasi terhadap `Total_charge`. Sehingga penentuan total tagihan tidak berdasarkan jumlah panggilan melainkan jumlah durasi, mayoritas pelanggan melakukan panggilan sebanyak kurang lebih 270 hingga 350 setiap periode langganan.

E. Data Preprocessing

1. Features and mapping

```
[46] # Dropping column yang tidak digunakan
     df.drop(columns=['account_length', 'state', 'total_day_minutes', 'total_day_charge', 'total_eve_minutes', 'total_eve_charge', 'total_night_minutes',
                      'total_night_charge', 'total_intl_minutes', 'total_day_calls', 'total_eve_calls', 'total_night_calls', 'total_intl_calls'], inplace=True)
```

```
#Mengganti string 'yes' 'no'menjadi dummy
df['international_plan'] = df['international_plan'].map({'yes': 1, 'no': 0})
df['voice_mail_plan'] = df['voice_mail_plan'].map({'yes': 1, 'no': 0})
df['area_code'] = df['area_code'].map({'area_code_415': 1, 'area_code_408':2, 'area_code_510': 3})
df['churn'] = df['churn'].map({'yes': 1, 'no': 0})
```

```
sns.heatmap(df.corr())
```

```
#Melakukan treatment terhadap multikolinieritas
df.drop(columns=['number_vmail_messages'], inplace=True)
```

```
[50] sns.heatmap(df.corr())
```

```
[51] df.hist(bins=50, figsize=(20,15))
```

2. Data Modelling

```
[52] #Drop Target Variable
     X = df.drop('churn' , 1 )
     y = df['churn']
```

```
[53] #Train-test Split
     #Kami menggunakan test size 20%, dan random_state pada d123
     # Split data ke training(80%) dan testing(2%).
     from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.2, random_state =0)
```

Model yang kami gunakan:

- Logistic Regression
- Decision Tree
- Random Forest

Feature Scaling

```
[60] sc_X = StandardScaler()
     X_train = sc_X.fit_transform(X_train)
     X_test = sc_X.transform(X_test)
```

Model Logistic Regression

```
[61] #Feature Scaling
     from sklearn.preprocessing import StandardScaler
     from sklearn.linear_model import LogisticRegression

     #Fitting Logistic Regression di Training Set
     logreg = LogisticRegression(random_state = 0)
     logreg.fit(X_train, y_train)
```

```
#Predicting Test Result
y_pred = logreg.predict(X_test)
y_pred
```

```
[63] # Evaluating Model
     from sklearn.metrics import confusion_matrix
     conmat = confusion_matrix(y_test, y_pred)
     conmat

     array([[726,   9],
            [100,  15]])
```

```
#Accuracy dan Missclassification Rate
accuracy1 = (726 + 15) / 100
print('Accuracy rate dari model Logistic Regression adalah: ' + str(accuracy1))
missclass1 = (9 + 100) / 100
print('Missclassification rate dari model Logistic Regression adalah: ' + str(missclass1))
```

```
Accuracy rate dari model Logistic Regression adalah: 7.41
Missclassification rate dari model Logistic Regression adalah: 1.09
```

Model Decision Tree

```
dtc = DecisionTreeClassifier()
dtc.fit(X_train, y_train)
```

```
[66] # Predict Model
     predictions_tree = dtc.predict(X_test)
     predictions_tree
```

```
# Evaluasi Model
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test, predictions_tree))
```

```
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       735
           1       0.55      0.61      0.58       115

    accuracy                           0.88       850
   macro avg       0.74      0.77      0.75       850
weighted avg       0.89      0.88      0.88       850
```

```
print(confusion_matrix(y_test, predictions_tree))
```

```
[[678  57]
 [ 46  69]]
```

```
#Accuracy dan Missclassification Rate
accuracy2 = (675 + 69) / 100
print('Accuracy rate dari model Decisions Tree adalah: ' + str(accuracy2))
missclass2 = (60 + 46) / 100
print('Missclassification rate dari model Decisions Tree adalah: ' + str(missclass2))
```

```
Accuracy rate dari model Decisions Tree adalah: 7.44
Missclassification rate dari model Decisions Tree adalah: 1.06
```

Model Random Forests

```
from sklearn.ensemble import RandomForestClassifier
rforest = RandomForestClassifier(n_estimators=100)
rforest.fit(X_train, y_train)
```

```
rforest_pred = rforest.predict(X_test)
```

```
[71] print(classification_report(y_test, rforest_pred))
```

```
                  precision    recall  f1-score   support

               0       0.94      0.98      0.96       735
               1       0.82      0.58      0.68       115

        accuracy                           0.93       850
       macro avg       0.88      0.78      0.82       850
    weighted avg       0.92      0.93      0.92       850
```

```
[72] print(confusion_matrix(y_test, rforest_pred))
```

```
[[720  15]
 [ 48  67]]
```

```
[73] #Accuracy dan Missclassification Rate
     accuracy3 = (722 + 68) / 100
     print('Accuracy rate dari model Random Forest adalah: ' + str(accuracy3))
     missclass3 = (13 + 47) / 100
     print('Missclassification rate dari model Random Forest adalah: ' + str(missclass3))
```

```
Accuracy rate dari model Random Forest adalah: 7.9
Missclassification rate dari model Random Forest adalah: 0.6
```

Kesimpulan

Random Forest memiliki accuracy paling tinggi

```
[78] print('----Logistic Regression----')
     print('Accuracy rate dari model Logistic Regression adalah: ' + str(accuracy1))
     print('Missclassification rate dari model Logistic Regression adalah: ' + str(missclass1))
     print('----Decisions Tree----')
     print('Accuracy rate dari model Decisions Tree adalah: ' + str(accuracy2))
     print('Missclassification rate dari model Decisions Tree adalah: ' + str(missclass2))
     print('----Random Forest----')
     print('Accuracy rate dari model Random Forest adalah: ' + str(accuracy3))
     print('Missclassification rate dari model Random Forest adalah: ' + str(missclass3))
     print('----KNN----')
     print('Accuracy rate dari model KNN adalah: ' + str(accuracy4))
     print('Missclassification rate dari model KNN adalah: ' + str(missclass4))
```

```
----Logistic Regression----
Accuracy rate dari model Logistic Regression adalah: 7.41
Missclassification rate dari model Logistic Regression adalah: 1.09
----Decisions Tree----
Accuracy rate dari model Decisions Tree adalah: 7.44
Missclassification rate dari model Decisions Tree adalah: 1.06
----Random Forest----
Accuracy rate dari model Random Forest adalah: 7.9
Missclassification rate dari model Random Forest adalah: 0.6
----KNN----
Accuracy rate dari model KNN adalah: 7.66
Missclassification rate dari model KNN adalah: 0.84
```

Simpan model kemudian upload data test

```
[79] import pickle
     rforest = RandomForestClassifier(n_estimators=100)
     rforest = rforest.fit(X_train, y_train)
     pickle.dump(rforest, open('forest.pkl', 'wb'))
```

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

```
[82] df = pd.read_csv('/content/Data Test.csv')
```

```
[83] df.head()
```

```
[84] #Membuat fungsi preprocessing untuk label encoding data yang baru masuk
     def Preprocess(data):
        # Assign new variable
        total_durasi= df['total_day_minutes'] + df['total_eve_minutes'] + df['total_night_minutes']
        total_charge = df['total_day_charge'] + df['total_eve_charge'] + df['total_night_charge']
        total_calls = df['total_day_calls'] + df['total_eve_calls'] + df['total_night_calls']

        df['total_durasi'] = total_durasi
        df['total_charge'] = total_charge
        df['total_calls'] = total_calls
        #Mengganti string 'yes' 'no'menjadi dummy
        df['international_plan'] = df['international_plan'].map({'yes': 1, 'no': 0})
        df['voice_mail_plan'] = df['voice_mail_plan'].map({'yes': 1, 'no': 0})
        df['area_code'] = df['area_code'].map({'area_code_415': 1, 'area_code_408':2, 'area_code_510': 3})
        # Dropping column yang tidak digunakan
        df.drop(columns=['account_length', 'state', 'total_day_minutes', 'total_day_charge', 'total_eve_minutes', 'total_eve_charge', 'total_night_minutes',
                         'total_night_charge', 'total_intl_minutes', 'total_day_calls', 'total_eve_calls', 'total_night_calls', 'total_intl_calls'], inplace=True)
        #Melakukan treatment terhadap multikolinieritas
        df.drop(columns=['number_vmail_messages'], inplace=True)
        #Variable X
        X = ['area_code', 'international_plan', 'voice_mail_plan', 'total_intl_charge', 'number_customer_service_calls','total_durasi','total_charge','total_calls']
        return data[X]
```

```
[85] df_test = Preprocess(df)
     df_test
```

	area_code	international_plan	voice_mail_plan	total_intl_charge	number_customer_service_calls	total_durasi	total_charge	total_calls
0	1	0	1	2.70	1	707.2	72.86	300
1	3	1	0	1.70	0	647.9	65.91	317
2	1	0	0	3.54	4	630.9	55.77	245
3	3	0	0	2.19	3	538.5	56.80	346
4	1	0	0	4.19	3	652.1	55.96	303
...	...	...	...	...	...	...	...	...
745	1	0	0	3.05	0	548.4	48.66	307
746	2	0	0	3.32	3	689.5	63.99	314
747	1	0	0	3.97	3	654.6	62.75	276
748	1	0	0	3.67	1	525.8	48.15	314
749	3	0	0	2.30	0	584.9	56.79	248

750 rows × 8 columns

```
[86] #Memanggil Model dan melakukan prediksi
     rforest_model = pickle.load(open('/content/forest.pkl', 'rb'))

     r_forest_test_predictions = rforest_model.predict(df_test)
```

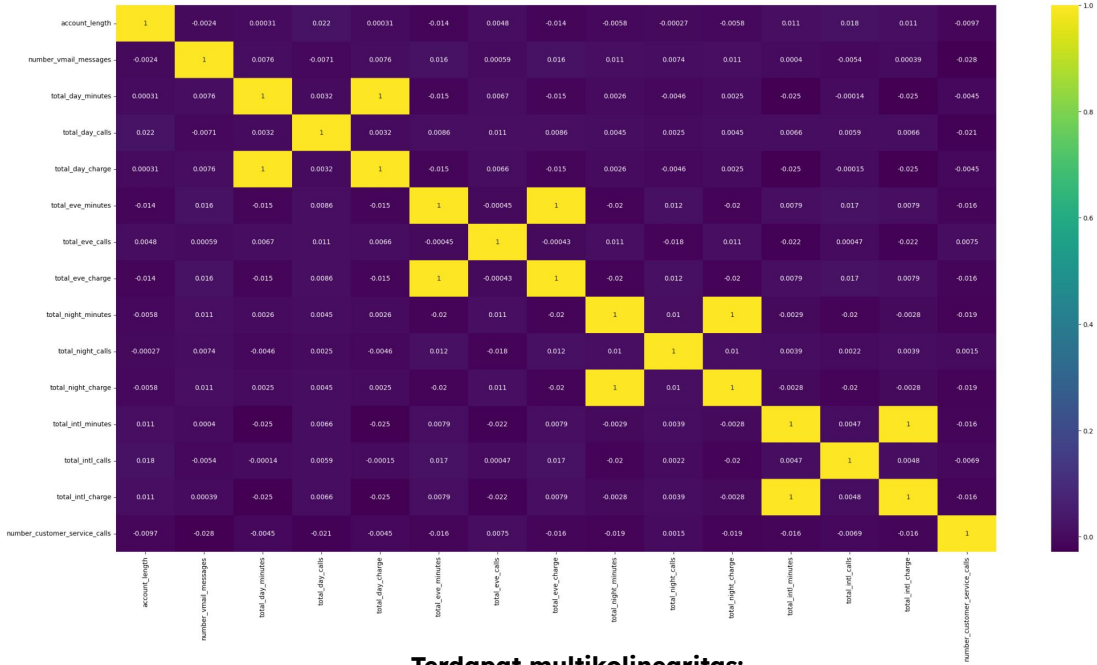
```
[87] print("Predictions:")
     print(r_forest_test_predictions)
```

```
Predictions:
[0 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 1 0 0 1 1 1 1 1 1 0 1 0 0 1 1 1 0 0
 1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1
 0 1 1 0 0 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 0 0
 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 0
 1 1 1 1 0 1 1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 1
 1 1 1 0 1 1 1 0 1 0 0 1 0 1 1 0 0 1 0 1 1 0 1 0 0 1 1 1 0 1 0 1 1 0 0 0 1
 1 0 0 1 1 0 0 1 1 1 0 1 0 1 1 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 1 1 0 1
 1 0 1 0 1 1 1 0 1 1 1 1 1 1 0 0 1 1 0 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1 0
 0 1 1 0 1 0 1 1 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0 0 0 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 0 0 0 1 1 1 1
 1 1 1 0 1 1 1 1 0 1 0 0 1 1 1 1 1 0 0 0 1 1 0 1 0 1 1 1 1 1 0 1 1 0 0 0 1
 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 1 0 1 1 1 0 1
 0 1 1 1 1 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0 1 0 1 1 1 1
 1 1 1 0 1 1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 1 1 1 1 1 1 0 0 0 1
 0 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 0
 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 1 1 1 0 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1
 1 0 0 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 1 0 1 1 0 0
 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1]
```

Interpretasi:

EDA (Exploratory Data Analysis)

1. Analisa Multivariat



Terdapat multikolinearitas:

1. total_day_minutes and total_day_charge
2. total_eve_minutes and total_eve_charge
3. total_night_minutes and total_night_charge
4. total_intl_minutes and total_intl_charge

Treatment:

1. melakukan kombinasi variable yang terindikasi multikolinieritas
   - **total_charge =** total_day_charge + total_eve_charge + total_night_charge
   - **total_minutes =** total_day_minutes+total_eve_minutes+ total_night_minutes
2. mengeliminasi salah satu variable yang memiliki korelasi tinggi tersebut

   total_intl_minutes & **intl_charge,**

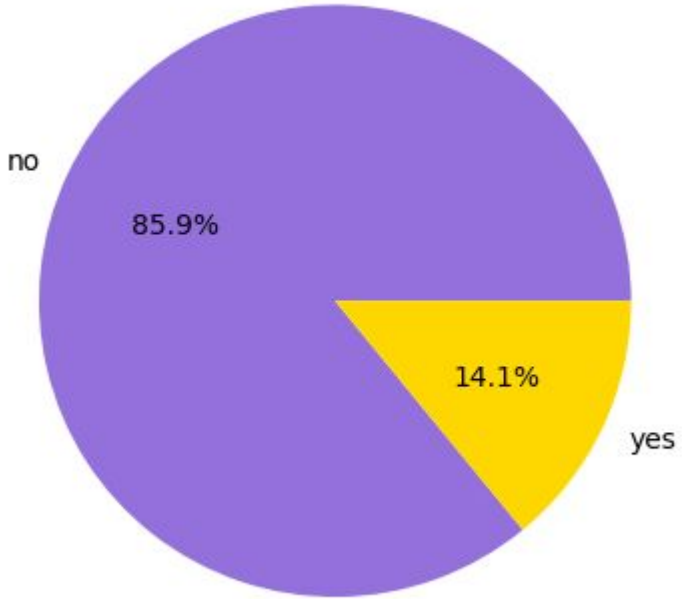
   karena dalam data kami mengamati pengaruh international plan terhadap churn maka, kami memberi treatment pada multikolinieritas ini dengan *drop total* **total_intl_minutes**

EDA (Exploratory Data Analysis)

2. Analisa Univariat

Problem Business

terdapat 14.1% kosumen yang melakukan churn, beralih menggunakan produk lain, maka perlu dilakukan peningkatan peforma pelayanan kepada konsumen berdasarkan hasil analisa yang menargetkan konsumen untuk tetap menggunakan layanan perusahaan ini (loyalitas)

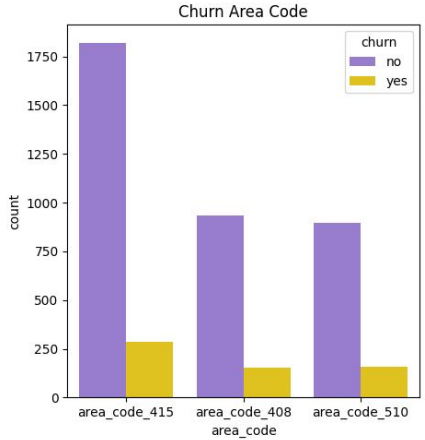


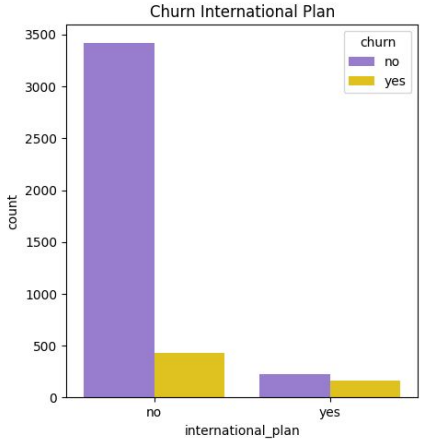
EDA (Exploratory Data Analysis)

2. Analisa Univariat

Pelanggan paling banyak berasal dari area code 415, diikuti oleh area code 408 dan 510

Mayoritas pelanggan tidak menggunakan layanan international, dan tidak menggunakan voice mail



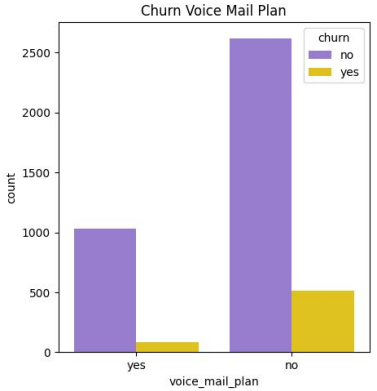


Insight :

- Area 415 memiliki tingkat churn paling tinggi
- Pelanggan yang tidak melakukan international plan lebih cenderung mempertahankan untuk berlangganan dibanding pelanggan yang melakukan langganan international

PERLU PENINGKATAN DI INTERNATIONAL PLAN

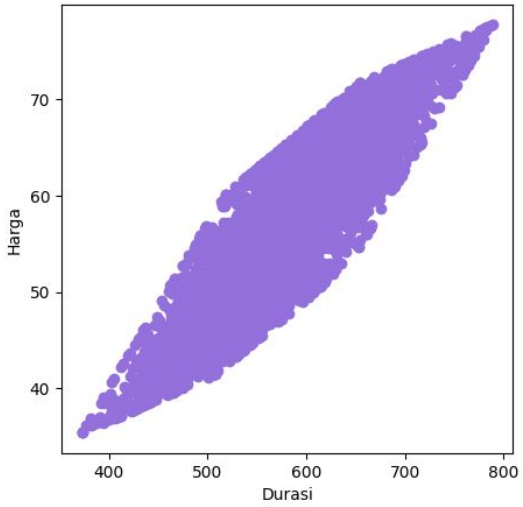
- Pelanggan yang tidak melakukan langganan voice mail cenderung melakukan churn dibandingkan yang melakukan langganan voice mail



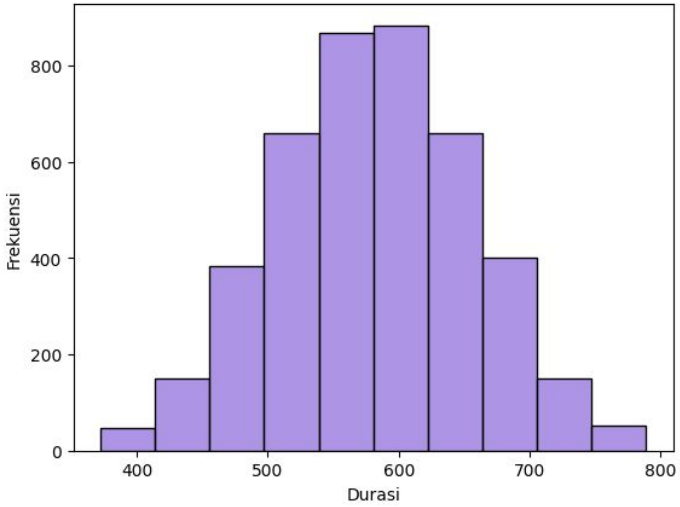
PERLU PROMOSI TENTANG HARGA VOICE MAIL YANG TERJANGKAU, SEHINGGA PELANGGAN YANG BELUM MELAKUKAN LANGGANAN VOICE MAIL DAPAT MERASAKAN LAYANAN INI

EDA (Exploratory Data Analysis)

Q1: Bagaimana pengaruh lama panggilan terhadap harga



Total_durasi dan **Total Charge** memiliki korelasi positif, dimana semakin meningkat total panggilan harga tagihan akan semakin tinggi



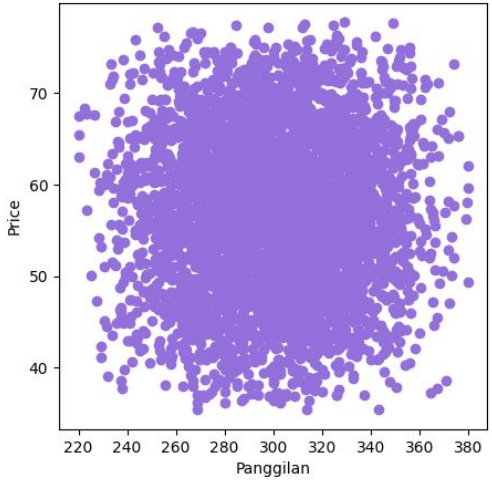
Dapat dilihat bahwa mayoritas dari pelanggan melakukan total durasi panggilan selama 500 hingga kurang lebih 650 menit dan mendapatkan charge sebesar 50 hingga 65.

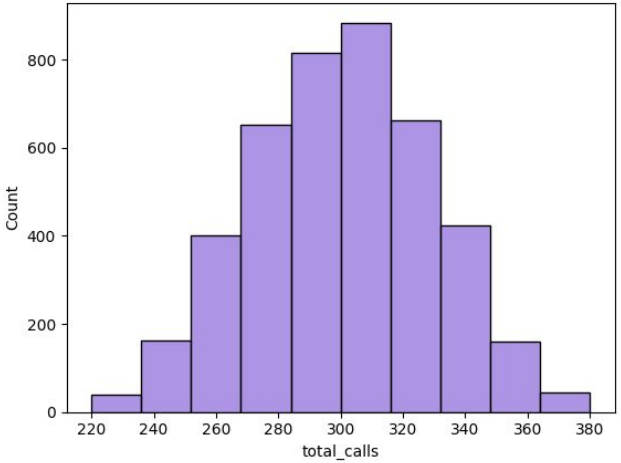
Insight:

dapat dilakukan promo panggilan, atau gratis panggilan berdasarkan durasi

EDA (Exploratory Data Analysis)

Q2: Pengaruh jumlah panggilan terhadap harga



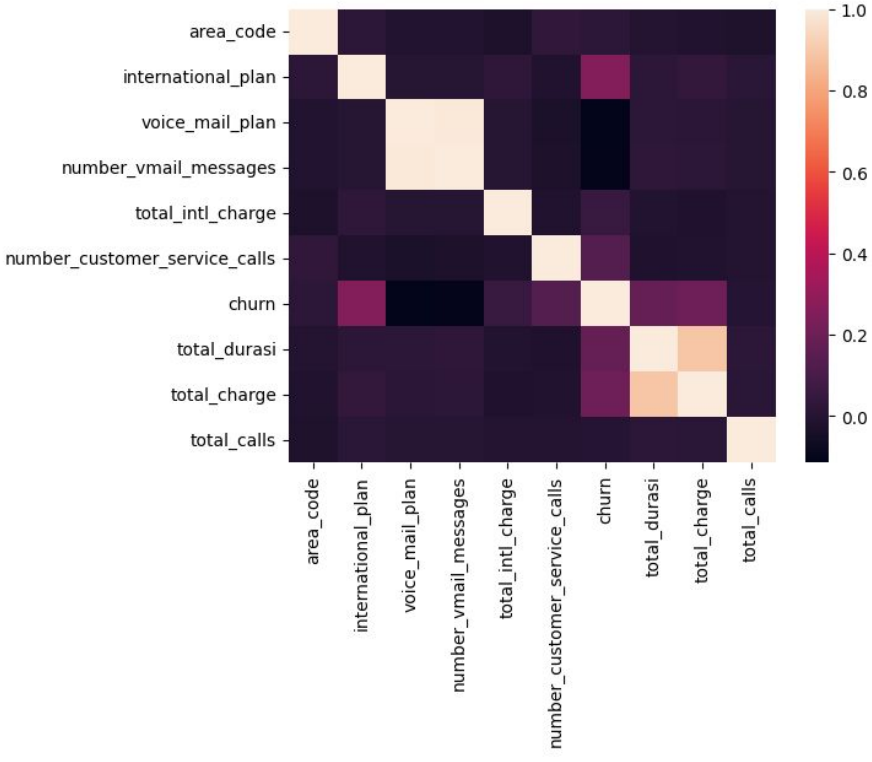


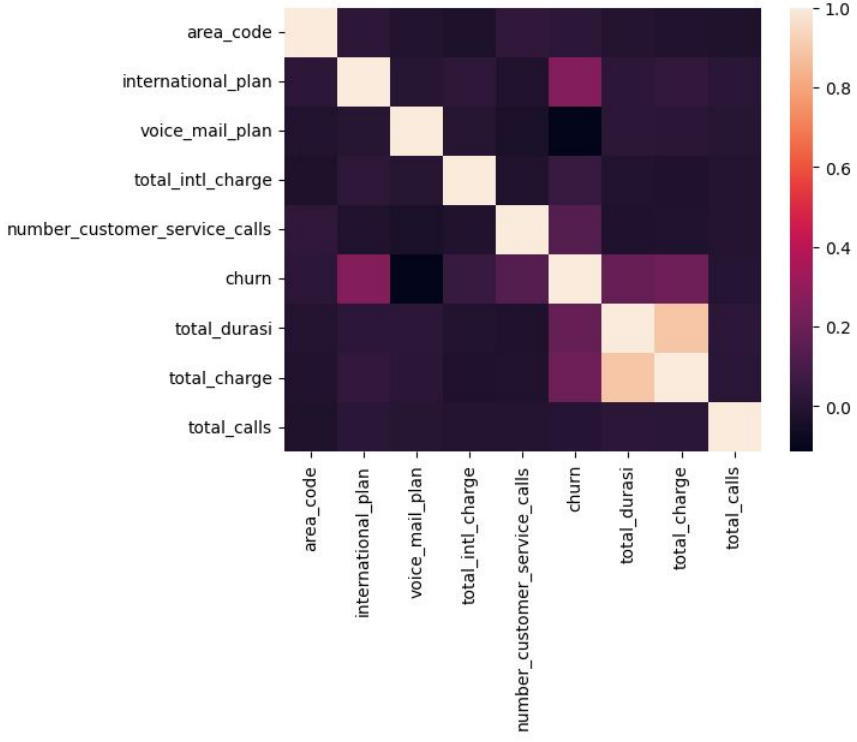
Total calls tidak memiliki korelasi terhadap **Total_charge**. Sehingga penentuan total tagihan tidak berdasarkan jumlah panggilan melainkan jumlah durasi, mayoritas pelanggan melakukan panggilan sebanyak kurang lebih 270 hingga 350 setiap periode langganan.

Data Pre-Processing

Pengecekan Corr()

drop number_vmail_messages





Hasil Model

Random Forest

1.Logistic Regression

Accuracy rate dari model Logistic Regression adalah: (87%)

2. Decisions Tree

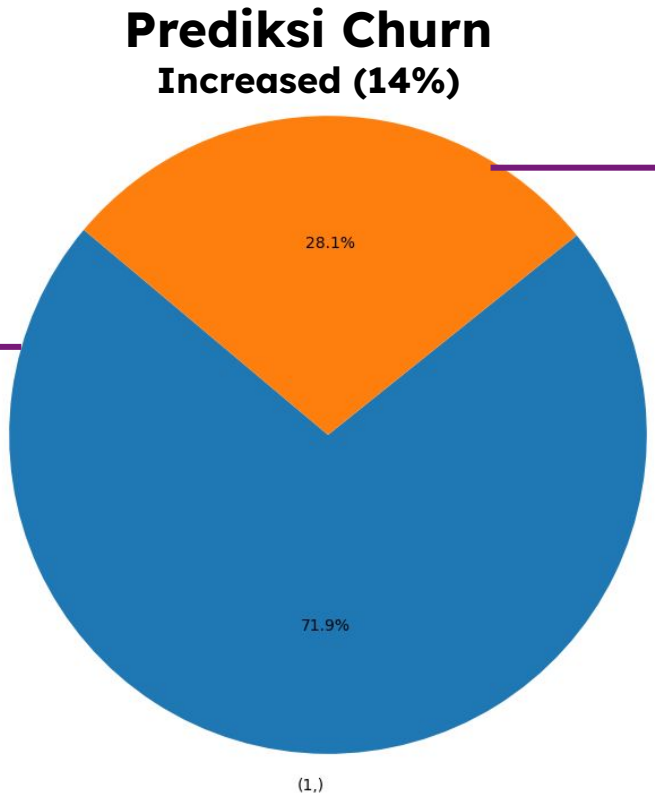
Accuracy rate dari model Decisions Tree adalah: (87%)

3. Random Forest

Accuracy rate dari model Random Forest adalah: (93%)

Hasil Model

Random Forest



Orange (Churn) = 28.1%

Blue (Tidak Churn) = 71.9%

Business Insights:

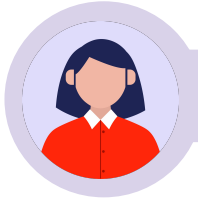
Business Insight

1. Harga

1. Pemberian potongan harga dan gratis panggilan

   kepada konsumen loyal sebagai bentuk penghargaan. Mengingat tidak banyak konsumen yang melakukan churn, maka **loyalitas konsumen menjadi prioritas**.

2. pemberian promo khusus dan reward untuk **daerah 415**, untuk menekan angka churn, dengan meningkatkan loyalitas kepada brand



2. Loyalitas

1. Meningkatkan tawaran layanan program **langganan international**
2. Melakukan promosi **harga voice mail / tawaran layanan voice mail gratis untuk periode tertentu, a**gar konsumen mau mencoba layanan ini

Mengingat konsumen yang menggunakan layanan **voice mail dan international** memiliki t**otal churn lebih rendah,** menandakan bahwa konsumen memiliki **kepuasan** terhadap layanan ini dan menjadi salah satu faktor konsumen memutuskan untuk **tetap loyal pada brand**

3. Model

Berdasarkan hasil analisa model, maka dapat disimpulkan bahwa model yang paling mampu menghasilkan hasil paling akurat diantara tiga model yang diuji adalah **model Random Forest**