BINAR

DS4-16 - Presentation

Data Science

Anggota Tim

TIM	NAMA
DS4-16	Aulia Nur Adib Phasya
	Muhammad Tibri Syofyan

List of Contents

BINAR

Apa saja yang akan dibahas di sini?

- Big Query & Dashboard (Case Study: COVID-19 in Indonesia)
  - BigQuery (SQL)
  - Dashboard
- Customer Churn Prediction
  - Problem Statement
  - Exploratory Data Analysis (EDA)
  - Data Preprocessing
  - Classification Analysis
  - Conclusion

BINAR

Creating Query Using BigQuery

BINAR

Query Ke-1

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

Jawaban Query

```
SELECT province, SUM(New_Active_Cases)
total_kasus_aktif_baru
FROM fga-ds-project.challenge01.case_covid_19_in
WHERE province IS NOT NULL
```

Hasil Output Query

Row	province	total_kasus_aktif_bar
1	Jawa Barat	13496
2	DKI Jakarta	10922
3	Banten	2558
4	Jawa Tengah	1423
5	Jawa Timur	1136
6	Daerah Istimewa Yogyakarta	669
7	Sumatera Utara	664
8	Sulawesi Utara	565
9	Bali	474
10	Sumatera Selatan	313
11	Kalimantan Timur	272
12	Papua	237
13	Lampung	226
14	Riau	224

BINAR

Query Ke-2

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

Jawaban Query

```
SELECT province, SUM(New_Active_Cases)
total_kasus_aktif_baru
FROM fga-ds-project.challenge01.case_covid_19_in
WHERE province IS NOT NULL
GROUP BY 1
ORDER BY 2 DESC
```

Hasil Output Query

Row	Location_ISO_Code	total_kematian
1	ID-MA	147196
2	ID-MU	167511

BINAR

Query Ke-3

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

Jawaban Query

```
SELECT case_covid_19_in.Date Tanggal,
MAX(Case_Recovered_Rate)
case_recovered_rate_tertinggi
FROM fga-ds-project.challenge01.case_covid_19_in
GROUP BY 1
ORDER BY 2 DESC
```

Hasil Output Query

Row	Tanggal	case_recovered_rate
1	2020-03-06	111.0
2	2020-03-07	111.0
3	2020-03-12	66.0
4	2020-03-11	65.0
5	2020-03-10	61.0
6	2020-03-03	60.0
7	2020-03-08	57.0
8	2020-03-09	57.0
9	2020-03-13	44.0
10	2020-03-14	33.5
11	2020-03-04	30.0
12	2020-03-26	28.0
13	2020-03-15	27.8

BINAR

Query Ke-4

Total case fatality rate dan case recovered rate dari masing-masing location iso code yang diurutkan dari data yang paling rendah

Jawaban Query

```
SELECT Location_ISO_Code, AVG(Case_Fatality_Rate)
jumlah_case_fatality_rate,
AVG(Case_Recovered_Rate)
jumlah_case_recovered_rate
FROM fga-ds-project.challenge01.case_covid_19_in
GROUP BY 1
ORDER BY 2 ASC, 3 ASC
```

Hasil Output Query

Row	Location_ISO_Code	jumlah_case_fatality	jumlah_case_recover
1	ID-KU	0.015837028824...	0.813444124168...
2	ID-NT	0.017903932584...	0.787439101123...
3	ID-PA	0.018607158590...	0.669859691629...
4	ID-JA	0.019040439560...	0.835746373626...
5	ID-SG	0.021379021739...	0.806156956521...
6	ID-KB	0.022820199778...	0.856352719200...
7	ID-SR	0.024146059933...	0.813398779134...
8	ID-SN	0.024651372118...	0.851039956092...
9	ID-SB	0.026560066371...	0.834350774336...
10	ID-PB	0.026948062015...	0.838536766334...
11	ID-MU	0.027153252480...	0.792092723263...
12	ID-KI	0.027508624454...	0.826606659388...
13	ID-BB	0.027768777777...	0.857679222222...

BINAR

Query Ke-5

Data tentang tanggal-tanggal saat total kasus Covid-19 mulai menyentuh angka 30.000-an

Jawaban Query

```
SELECT case_covid_19_in.Date Tanggal,
SUM(New_Cases) total_kasus_baru
FROM fga-ds-project.challenge01.case_covid_19_in
GROUP BY 1
HAVING total_kasus_baru >= 30000
ORDER BY 1 ASC
```

Hasil Output Query

Row	Tanggal	total_kasus_baru
1	2021-06-23	30504
2	2021-06-24	39957
3	2021-06-25	37409
4	2021-06-26	41917
5	2021-06-27	42723
6	2021-06-28	40716
7	2021-06-29	41264
8	2021-06-30	43943
9	2021-07-01	48920
10	2021-07-02	52400
11	2021-07-03	56354
12	2021-07-04	55142

BINAR

Query Ke-6

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

Jawaban Query

```
SELECT count(Total_Cases) total_kasus
FROM fga-ds-project.challenge01.case_covid_19_in
WHERE Total_Cases >= 30000
```

Hasil Output Query

Row	total_kasus
1	14399

BINAR

Dashboard

COVID-19 in Indonesia

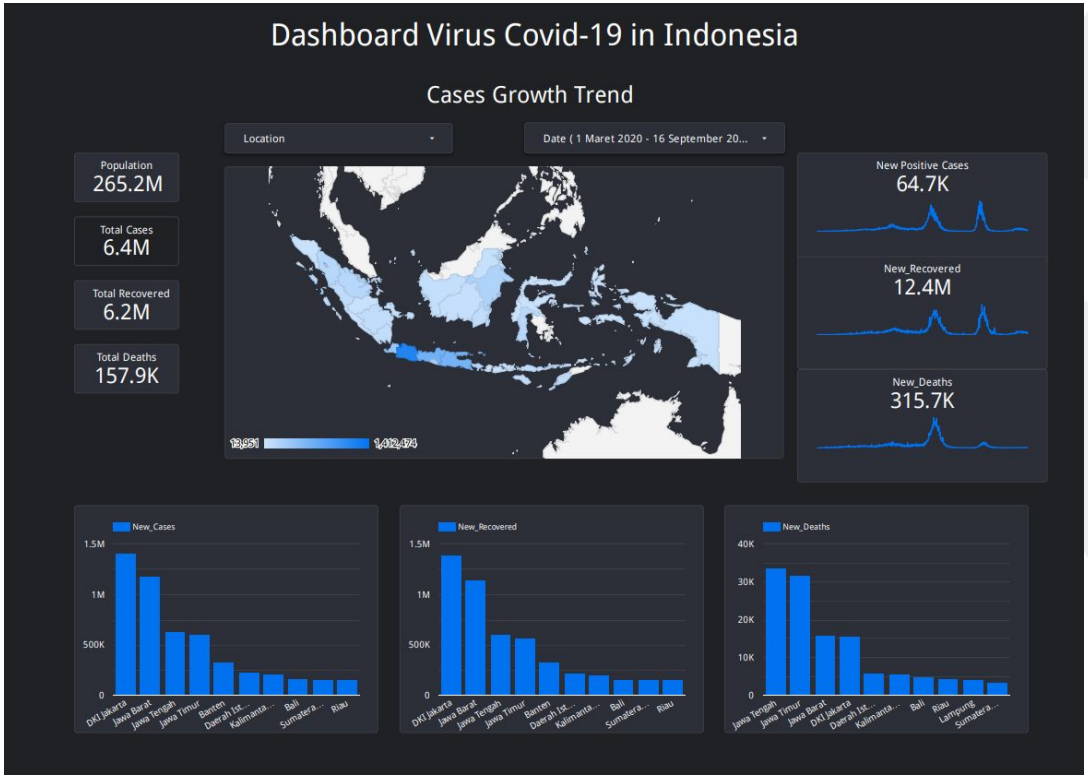
BINAR

Link Dashboard

https://lookerstudio.google.com/s/t8_T5q1S9Qo

BINAR

Visualisasi Dashboard



BINAR

Penjelasan tampilan dari dashboard

Dashboard menampilkan data umum yang isinya sebagai berikut.

- Populasi
- Total kasus COVID-19
- Total kasus yang sembuh
- Total kasus yang berakhir dengan kematian
- Peta geografis yang menunjukkan persebaran kasus COVID-19 di seluruh Indonesia
- Tiga buah *scorecard* yang masing-masing adalah data jumlah kasus baru yang muncul, kasus baru yang sembuh dan kasus baru yang meninggal, ditampilkan dengan grafik *time-series*
- Tiga buah diagram *bar-chart* berisi total kasus baru, jumlah kasus baru yang sembuh dan jumlah kasus baru yang meninggal, terurut berdasarkan provinsi dengan jumlah kasus terbanyak.

BINAR

Informasi yang bisa diambil dari dashboard?

- Provinsi DKI Jakarta dan Jawa Barat menjadi provinsi dengan jumlah kasus baru COVID-19 terbanyak di Indonesia. Dan sekaligus menjadi provinsi dengan jumlah kasus baru sembuh yang paling banyak juga.
- Provinsi Jawa Tengah dan Jawa Timur menjadi provinsi dengan jumlah kematian kasus baru COVID-19 terbanyak. Dilihat dari rasio jumlah kematian dan total kasusnya, jumlah kematian dari dua provinsi tersebut termasuk tinggi dan menjadi indikasi adanya penanganan yang kurang tepat.

Customer Churn Prediction

Problem Statement

BINAR

Customer Churn

Perusahaan telekomunikasi X sudah beroperasi selama puluhan tahun. Namun di masa sekarang, teknologi berkembang sangat pesat dan secara tidak langsung memunculkan banyak perusahaan telekomunikasi baru yang menimbulkan persaingan antar perusahaan telekomunikasi untuk memperoleh pelanggan baru.

Churn menjadi ancaman besar bagi perusahaan karena dapat mengurangi jumlah pendapatan dan *revenue* yang dihasilkan oleh perusahaan. Maka dari itu, diperlukan usaha dari perusahaan untuk **memprediksi pelanggan mana yang berpotensi *churn*** atau pindah ke *provider* lain. Selain itu, perlu untuk mengidentifikasi **faktor apa saja yang mempengaruhi terjadinya churn**.

Exploratory Data Analysis

BINAR

Library

Berikut merupakan *library* Python yang digunakan untuk pengerjaan *challenge* kali ini.

```
# Import library
import io
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

BINAR

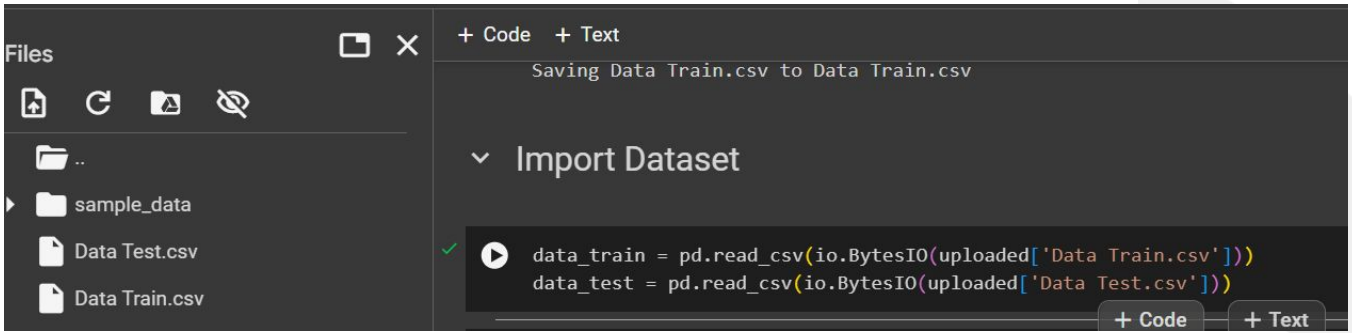
Import Dataset

Syntax untuk meng-upload data terdapat pada gambar di samping kanan.

```
# Upload data
uploaded = files.upload()
```

Kode untuk menampilkan data yang berformat csv terdapat pada gambar di bawah ini.

.



BINAR

Top 5 Train Data & Test Data

Syntax untuk menampilkan 5 data teratas dari data train dan data test ada pada di bawah ini beserta hasilnya.

```
data_train.head()
```

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

```
[5] data_test.head()
```

	id	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge	t
0	1	KS	128	area_code_415	no	yes	25	265.1	110	45.07	
1	2	AL	118	area_code_510	yes	no	0	223.4	98	37.98	
2	3	IA	62	area_code_415	no	no	0	120.7	70	20.52	
3	4	VT	93	area_code_510	no	no	0	190.7	114	32.42	
4	5	NE	174	area_code_415	no	no	0	124.3	76	21.13	

BINAR

Check Data Null dan Tipe Datanya

Syntax untuk melakukan cek ada tidaknya data yang nilainya null beserta tipe datanya ada pada gambar disamping kanan.

Hasilnya terdapat pada gambar disamping kanan untuk data train.

```
data_train.info(), data_test.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
```

BINAR

Check Data Null dan Tipe Datanya

Hasilnya terdapat pada gambar disamping kanan untuk data test.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 750 entries, 0 to 749
Data columns (total 20 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   id                            750 non-null    int64
 1   state                         750 non-null    object
 2   account_length                750 non-null    int64
 3   area_code                     750 non-null    object
 4   international_plan            750 non-null    object
 5   voice_mail_plan               750 non-null    object
 6   number_vmail_messages         750 non-null    int64
 7   total_day_minutes             750 non-null    float64
 8   total_day_calls               750 non-null    int64
 9   total_day_charge              750 non-null    float64
 10  total_eve_minutes             750 non-null    float64
 11  total_eve_calls               750 non-null    int64
 12  total_eve_charge              750 non-null    float64
 13  total_night_minutes           750 non-null    float64
 14  total_night_calls             750 non-null    int64
 15  total_night_charge            750 non-null    float64
 16  total_intl_minutes            750 non-null    float64
 17  total_intl_calls              750 non-null    int64
 18  total_intl_charge             750 non-null    float64
 19  number_customer_service_calls  750 non-null    int64
dtypes: float64(8), int64(8), object(4)
memory usage: 117.3+ KB
(None, None)
```

BINAR

Data Descriptive

Syntax untuk memisahkan variabel yang datanya numerikal dan variabel yang datanya kategorikal ada pada gambar di bawah ini..

Data descriptive

```
numerical = ['account_length','number_vmail_messages','total_day_minutes','total_day_calls','total_day_charge','total_eve_minutes','total_eve_calls','total_eve_
             'total_night_minutes','total_night_calls','total_night_charge','total_intl_minutes','total_intl_calls','total_intl_charge','number_customer_service_
categorical = ['state','area_code','international_plan','voice_mail_plan', 'churn']
```

BINAR

Data Descriptive

Syntax untuk mengetahui persebaran data untuk variabel yang nilainya kategorikal ada pada gambar di kanan. Hasilnya terdapat pada gambar di bawah ini..

```
for col in categorical:
  print(data_train[col].value_counts())
```

```
NE      73
DC      72
SC      72
AR      71
LA      69
PA      67
ND      67
GA      64
IA      62
AK      61
CA      39
Name: state, dtype: int64
area_code_415    2108
area_code_408    1086
area_code_510    1056
Name: area_code, dtype: int64
no      3854
yes      396
Name: international_plan, dtype: int64
no      3138
yes     1112
Name: voice_mail_plan, dtype: int64
no      3652
yes      598
Name: churn, dtype: int64
```

BINAR

Churn Consumer based on States

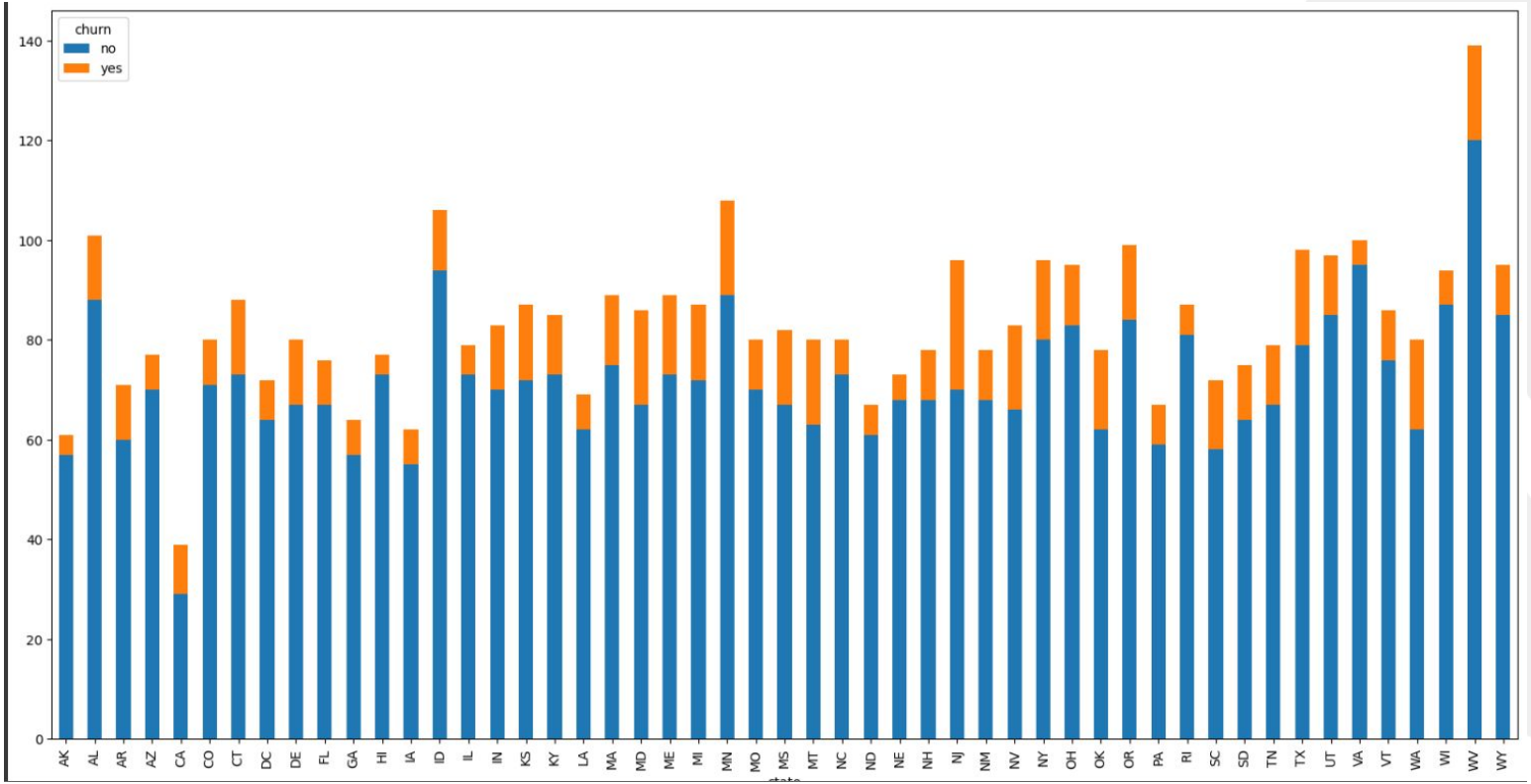
Syntax untuk mengukur persebaran data jumlah konsumen yang *churn* untuk masing-masing negara bagian atau *states* dimana konsumer tinggal.

```
#Ukur persebaran data jumlah konsumen (baik yang churn maupun tidak) untuk masing-masing state/negara
data_train.groupby(['state','churn']).size().unstack().plot(kind='bar', stacked=True, figsize=(20,10))
```

Untuk hasilnya berupa visualisasi bar chart di slide selanjutnya.

BINAR

Churn Consumer based on States



Churn Consumer based on States

Dari bar chart pada slide sebelumnya, jumlah konsumen terbanyak berada di negara bagian West Virginia. Tiga negara bagian lain setelahnya dengan jumlah konsumen terbanyak adalah Minnesota, Idaho dan Alabama.

BINAR

Churn Consumer based on Total Call to Customer Service

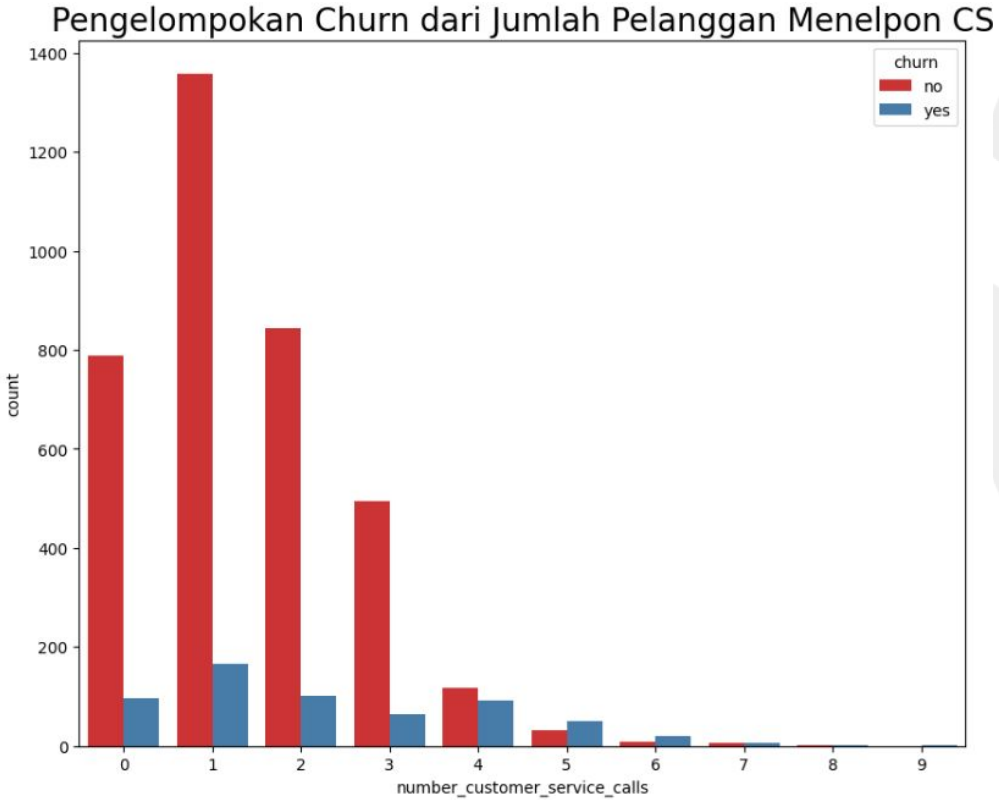
Syntax untuk mengukur persebaran data jumlah konsumen yang *churn* berdasarkan jumlah panggilan ke *customer service*.

```
plt.figure(figsize=(10,8))
plt.title('Pengelompokan Churn dari Jumlah Pelanggan Menelpon CS', fontsize=20)
sns.countplot(x='number_customer_service_calls', hue='churn', data=data_train, palette="Set1")
```

Untuk hasilnya berupa visualisasi bar chart di slide selanjutnya.

BINAR

Churn Consumer based on Total Call to Customer Service



BINAR

Churn Consumer based on Total Call to Customer Service

Dari bar chart pada slide sebelumnya, konsumen yang telah melakukan panggilan ke CS lebih dari 3 kali memiliki peluang untuk churn lebih besar dibanding konsumen yang melakukan panggilan ke CS kurang dari atau sama dengan 3 kali.

BINAR

Churn Consumer based on Total Total Subscription

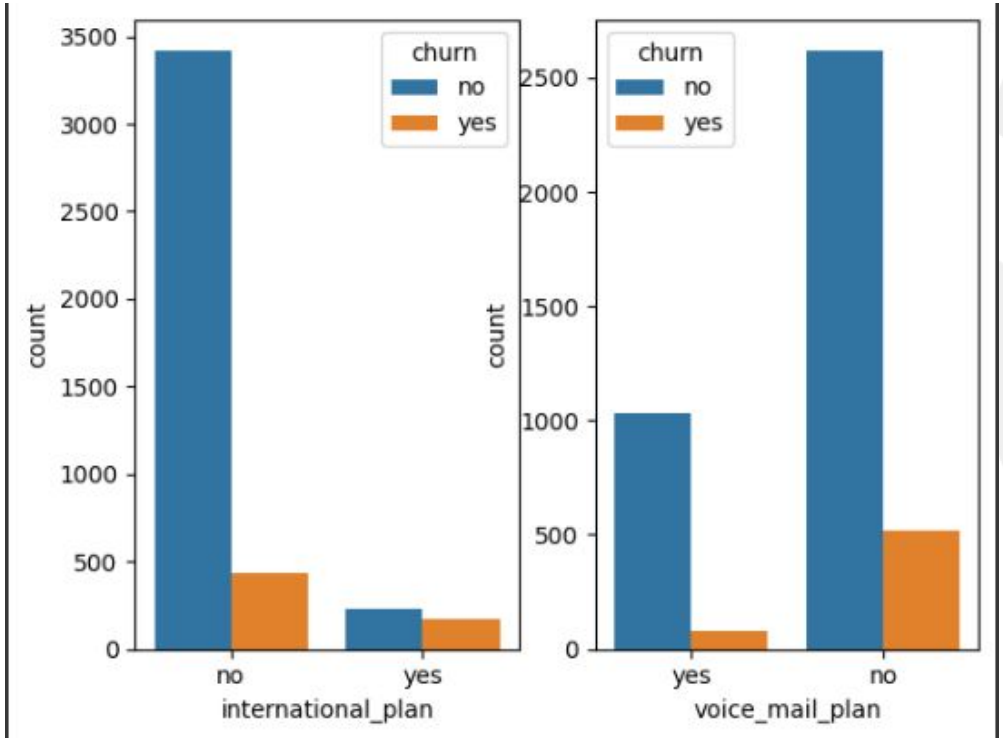
Syntax untuk mengukur persebaran data jumlah konsumen yang *churn* berdasarkan jumlah panggilan ke *customer service*.

```
fig, ax = plt.subplots(nrows=1, ncols=2)
sns.countplot(x='international_plan', hue='churn', data=data_train, ax=ax[0])
sns.countplot(x='voice_mail_plan', hue='churn', data=data_train, ax=ax[1])
```

Untuk hasilnya berupa visualisasi bar chart di slide selanjutnya.

BINAR

Churn Consumer based on Total Total Subscription

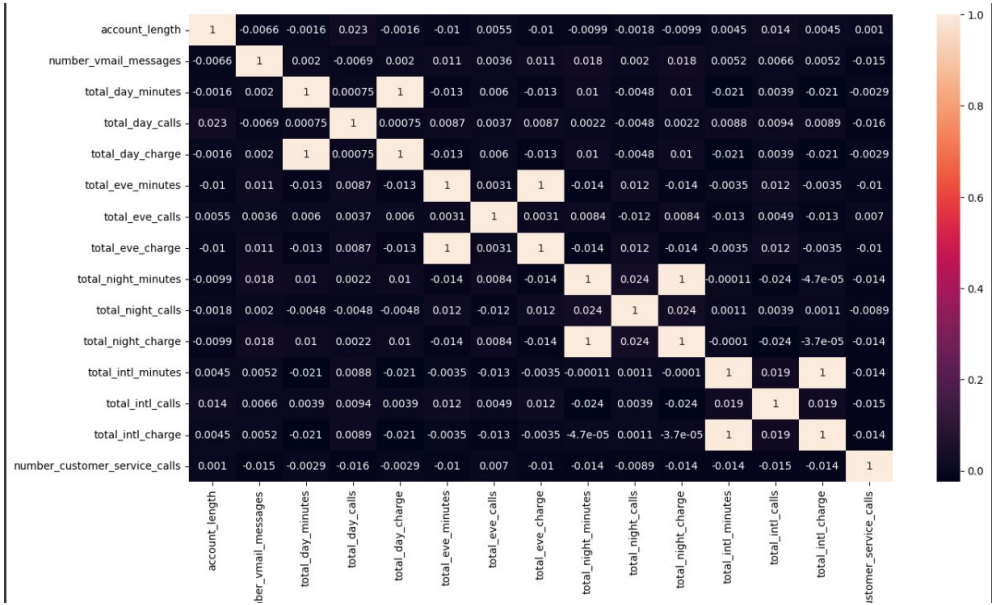


BINAR

Korelasi

Syntax untuk melihat korelasi atau hubungan antar variabel terdapat pada gambar di kanan dan hasilnya pada gambar di bawah ini..

```
plt.figure(figsize=(15,8))
sns.heatmap(data_train.corr(),annot=True, )
plt.show()
```



BINAR

Data Preprocessing

BINAR

Checking Null and Duplicate Data

Syntax untuk cek data yang null dan data yang duplikat ada di gambar samping kanan ini beserta hasilnya.

Terlihat bahwa tidak ada data yang nilainya null dan tidak ada data yang duplikat.

Checking Missing Value

```
data_train.isna().sum()
```

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

Checking Data Duplicates

```
data_train.duplicated().sum()
```

```
0
```

BINAR

Feature Selection & Encoding

Syntax untuk melakukan *feature selection* ada pada gambar disamping kanan.

Syntax untuk melakukan *feature encoding* ada pada gambar disamping kanan.

Feature Selection

```
data_train.drop(['area_code'], axis = 1, inplace = True)
data_test.drop(['id'], axis = 1, inplace = True)
data_test.drop(['area_code'], axis = 1, inplace = True)
```

Feature Encoding

```
[ ] data_train['churn']= LabelEncoder().fit_transform(data_train['churn'])
    data_train['international_plan']= LabelEncoder().fit_transform(data_train
    data_train['voice_mail_plan']= LabelEncoder().fit_transform(data_train['v
    dummy_train= pd.get_dummies(data_train['state'], prefix = 'state')
    dummy_test= pd.get_dummies(data_test['state'], prefix = 'state')
    data_train.drop(['state'], axis = 1, inplace = True)
    data_test.drop(['state'], axis = 1, inplace = True)
    data_train = data_train.join(dummy_train)
    data_test = data_test.join(dummy_test)
```

BINAR

Feature Selection & Encoding

Syntax untuk melakukan *feature selection* ada pada gambar disamping kanan.

Syntax untuk melakukan *feature encoding* ada pada gambar disamping kanan.

Feature Selection

```
data_train.drop(['area_code'], axis = 1, inplace = True)
data_test.drop(['id'], axis = 1, inplace = True)
data_test.drop(['area_code'], axis = 1, inplace = True)
```

Feature Encoding

```
[ ] data_train['churn']= LabelEncoder().fit_transform(data_train['churn'])
    data_train['international_plan']= LabelEncoder().fit_transform(data_train
    data_train['voice_mail_plan']= LabelEncoder().fit_transform(data_train['v
    dummy_train= pd.get_dummies(data_train['state'], prefix = 'state')
    dummy_test= pd.get_dummies(data_test['state'], prefix = 'state')
    data_train.drop(['state'], axis = 1, inplace = True)
    data_test.drop(['state'], axis = 1, inplace = True)
    data_train = data_train.join(dummy_train)
    data_test = data_test.join(dummy_test)
```

BINAR

Data Outlier

Syntax untuk melakukan *feature selection* ada pada gambar dibawah ini beserta hasilnya. Setelah dijalankan, data *outlier* sudah dihapus,

Checking outlier

```
[ ]  #mengecek outlier dengan IQR
     filtered_entries = np.array([True] * len(data_train))
     for col in numerical:
         Q1 = data_train[col].quantile(0.25)
         Q3 = data_train[col].quantile(0.75)
         IQR = Q3 - Q1
         low_limit = Q1 - (IQR * 1.5)
         high_limit = Q3 + (IQR * 1.5)

         filtered_entries = ((data_train[col] < low_limit) & (data_train[col] > high_limit)) & filtered_entries

     data_outlier = data_train[filtered_entries]
```

```
[ ]  data_outlier
```

account_length	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_

0 rows × 69 columns

BINAR

Normalization

Syntax untuk melakukan normalisasi menggunakan *library sklearn* ada pada gambar di bawah ini.

Normalization

```
[ ] data_train['account_length'] = MinMaxScaler().fit_transform(data_train['account_length'].values.reshape(len(data_train), 1))
    data_train['number_vmail_messages'] = MinMaxScaler().fit_transform(data_train['number_vmail_messages'].values.reshape(len(data_train)
    data_train['total_day_minutes'] = MinMaxScaler().fit_transform(data_train['total_day_minutes'].values.reshape(len(data_train), 1))
    data_train['total_day_calls'] = MinMaxScaler().fit_transform(data_train['total_day_calls'].values.reshape(len(data_train), 1))
    data_train['total_day_charge'] = MinMaxScaler().fit_transform(data_train['total_day_charge'].values.reshape(len(data_train), 1))
    data_train['total_eve_minutes'] = MinMaxScaler().fit_transform(data_train['total_eve_minutes'].values.reshape(len(data_train), 1))
    data_train['total_eve_calls'] = MinMaxScaler().fit_transform(data_train['total_eve_calls'].values.reshape(len(data_train), 1))
    data_train['total_eve_charge'] = MinMaxScaler().fit_transform(data_train['total_eve_charge'].values.reshape(len(data_train), 1))
    data_train['total_night_minutes'] = MinMaxScaler().fit_transform(data_train['total_night_minutes'].values.reshape(len(data_train), 1
    data_train['total_night_calls'] = MinMaxScaler().fit_transform(data_train['total_night_calls'].values.reshape(len(data_train), 1))
    data_train['total_night_charge'] = MinMaxScaler().fit_transform(data_train['total_night_charge'].values.reshape(len(data_train), 1))
    data_train['total_intl_minutes'] = MinMaxScaler().fit_transform(data_train['total_intl_minutes'].values.reshape(len(data_train), 1))
    data_train['total_intl_calls'] = MinMaxScaler().fit_transform(data_train['total_intl_calls'].values.reshape(len(data_train), 1))
    data_train['total_intl_charge'] = MinMaxScaler().fit_transform(data_train['total_intl_charge'].values.reshape(len(data_train), 1))
```

BINAR

Split Data

Syntax untuk melakukan split data ada pada gambar di bawah ini.

Split Data

```
y = data_train['churn']
x = data_train.drop(['churn'], axis = 1)

size = 0.3
seed = 243
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = size, random_state = seed)
```

```
len(xtrain), len(ytrain), len(ytest), len(xtest)
```

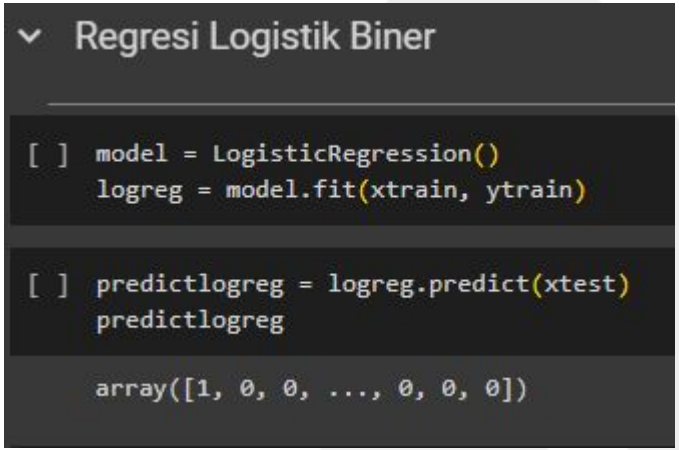
```
(2975, 2975, 1275, 1275)
```

Classification Analysis

BINAR

Logistic Regression

Syntax implementasi model menggunakan algoritma Logistic Regression ada pada gambar di kanan ini.



BINAR

Logistic Regression

Syntax pemodelan menggunakan algoritma Logistic Regression ada pada gambar di kanan beserta hasil evaluasinya.

```
# Confusion Matrix
CM_logreg = confusion_matrix(ytest, predictlogreg)
CM_logreg
# TP = 1042
# FP = 35
# FN = 160
# TN = 38
```

```
array([[1042,   35],
       [ 160,   38]])
```

```
print('Akurasi : ', accuracy_score(ytest, predictlogreg)*100,' %')
print(classification_report(ytest, predictlogreg))
```

```
Akurasi :  84.70588235294117  %
              precision    recall  f1-score   support

           0       0.87      0.97      0.91      1077
           1       0.52      0.19      0.28       198

    accuracy                           0.85      1275
   macro avg       0.69      0.58      0.60      1275
weighted avg       0.81      0.85      0.82      1275
```

BINAR

Support Vector Machine

Syntax implementasi model menggunakan algoritma Support Vector Machine ada pada gambar di kanan ini.

Support Vector Machine

```
model = SVC(kernel='linear',random_state=24,probab
svc = model.fit(xtrain,ytrain)
```

```
predictsvc = svc.predict(xtest)
predictsvc
```

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

BINAR

Support Vector Machine

Syntax pemodelan menggunakan algoritma Support Vector Machine ada pada gambar di kanan beserta hasil evaluasinya.

```
# Confusion Matrix
CM_svc = confusion_matrix(ytest, predictsvc)
CM_svc
# TP = 1077
# FP = 0
# FN = 198
# TN = 0
```

```
array([[1077,    0],
       [ 198,    0]])
```

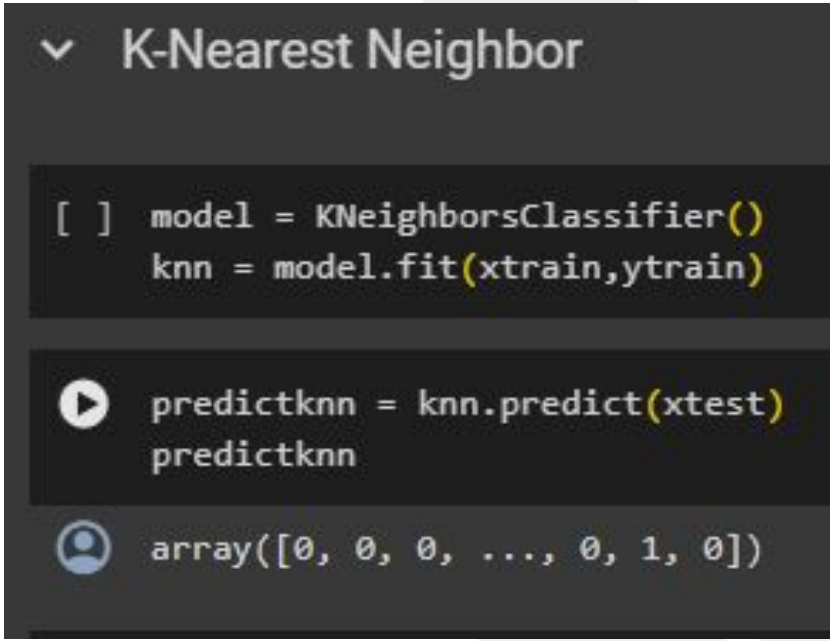
```
print('Akurasi : ', accuracy_score(ytest, predictsvc)*100,' %')
print(classification_report(ytest, predictsvc))
```

```
Akurasi :  84.47058823529412  %
              precision    recall  f1-score   support

           0       0.84      1.00      0.92      1077
           1       0.00      0.00      0.00       198

    accuracy                           0.84      1275
   macro avg       0.42      0.50      0.46      1275
weighted avg       0.71      0.84      0.77      1275
```

BINAR

K-Nearest Neighbour

Syntax implementasi model menggunakan algoritma K-Nearest Neighbour ada pada gambar di kanan ini.



BINAR

K-Nearest Neighbour

Syntax pemodelan menggunakan algoritma K-Nearest Neighbour ada pada gambar di kanan beserta hasil evaluasinya.

```
# Confusion Matrix
CM_knn = confusion_matrix(ytest, predictknn)
CM_knn
# TP = 1052
# FP = 25
# FN = 168
# TN = 30
```

```
array([[1052,   25],
       [ 168,   30]])
```

```
print('Akurasi : ', accuracy_score(ytest, predictknn)*100,
print(classification_report(ytest, predictknn))
```

```
Akurasi :  84.86274509803921  %
              precision    recall  f1-score   support

           0       0.86      0.98      0.92      1077
           1       0.55      0.15      0.24       198

    accuracy                           0.85      1275
   macro avg       0.70      0.56      0.58      1275
weighted avg       0.81      0.85      0.81      1275
```

Random Forest

Syntax implementasi model menggunakan algoritma Random Forest ada pada gambar di bawah ini.

Random Forest Classifier Model

```
[ ]  model = RandomForestClassifier(n_estimators=100, criterion='entropy',random_state=48)
     rf = model.fit(xtrain,ytrain)
```

```
[ ]  predictrf = rf.predict(xtest)
     predictrf
```

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

BINAR

Random Forest

Syntax pemodelan menggunakan algoritma Random Forest ada pada gambar di kanan beserta hasil evaluasinya.

```
# Confusion Matrix
CM_rf = confusion_matrix(ytest, predictrf)
CM_rf
# TP = 1072
# FP = 5
# FN = 71
# TN = 127
```

```
array([[1072,    5],
       [  71,  127]])
```

```
print('Akurasi : ', accuracy_score(ytest, predictrf)*100,' %')
print(classification_report(ytest, predictrf))
```

```
Akurasi :  94.03921568627452  %
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      1077
           1       0.96      0.64      0.77       198

    accuracy                           0.94      1275
   macro avg       0.95      0.82      0.87      1275
weighted avg       0.94      0.94      0.94      1275
```

Conclusion

Conclusion

- Dari hasil EDA, jumlah panggilan ke *customer service* dari pelanggan berpengaruh terhadap *churn*.
- Pelanggan yang berlangganan paket *international plan* memiliki peluang lebih besar untuk *churn* dibanding yang tidak berlangganan paket tersebut
- Dari hasil klasifikasi menggunakan 4 buah algoritma (**Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbour** dan **Random Forest**), algoritma **Random Forest** memiliki akurasi yang paling tinggi dibanding algoritma lainnya (**94%**)

Sumber bacaan

https://medium.com/@lekileki/customer-churn-predictions-using-classification-analysis-e0f0a038f2fd

Report Pembagian Tugas

Report Pembagian Tugas

Nama	Tasklist/Deliverable
Aulia Nur Adib Phasya	**Challenge 1** ● Penanggungjawab misi pertama untuk soal 1, 3, dan 5 ● Mengerjakan semua misi pertama ● Diskusi dan mengerjakan pembuatan dashboard bersama **Challenge 2 + Deck Presentation** ● EDA + Data Preprocessing ● Pembuatan 2 model algoritma klasifikasi ● Pembuatan presentasi
Muhammad Tibri Syofyan	**Challenge 1** ● Penanggungjawab misi pertama untuk soal nomor 2, 4, dan 6 ● Mengerjakan semua misi kedua ● Diskusi dan mengerjakan pembuatan dashboard bersama **Challenge 2 + Deck Presentation** ● EDA + Data Preprocessing ● Pembuatan 4 model algoritma klasifikasi ● Pembuatan presentasi

BINAR

Thank You