

Covid-19 Dashboard And Customers Churn Analysis

Dewandiaksa Syahda Marhaendra



...

Covid-19 Dashboard

1. Dashboard
2. Dashboard Implement



...

Customers Churn Analysis

1. Business Understanding
2. Data Understanding (EDA)
3. Data Preparation
4. Modeling
5. Evaluation
6. Implementation



Covid-19 Dashboard



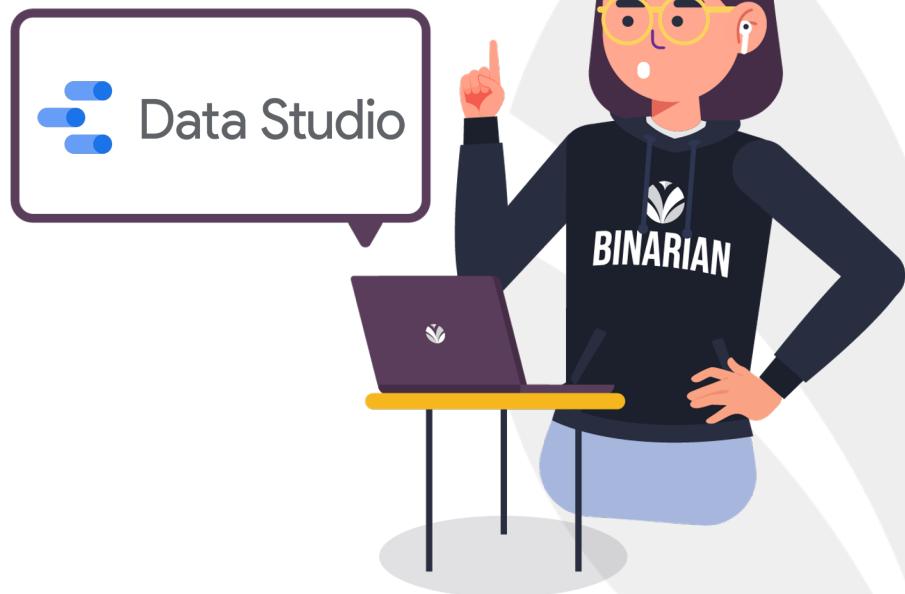
Pendahuluan

Pandemi Covid-19 telah melanda dunia sejak tahun 2020. Hingga sekarang, tren dari kasus covid di Indonesia sempat mengalami proses naik dan turun.

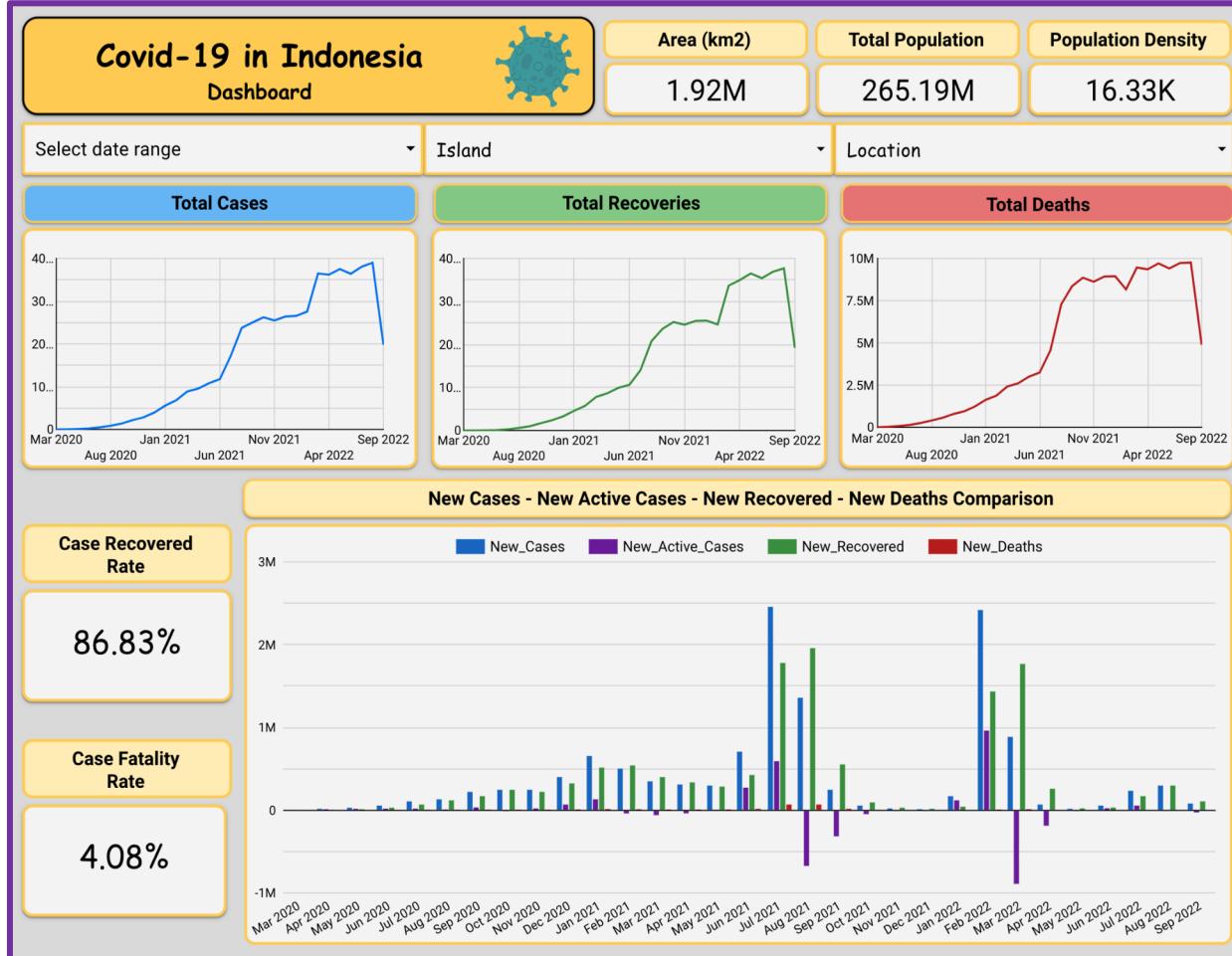
Dashboard yang dibuat ditujukan agar orang awam dapat mengetahui tren kasus covid yang ada di Indonesia sejak awal hingga saat ini.

Tools yang Digunakan

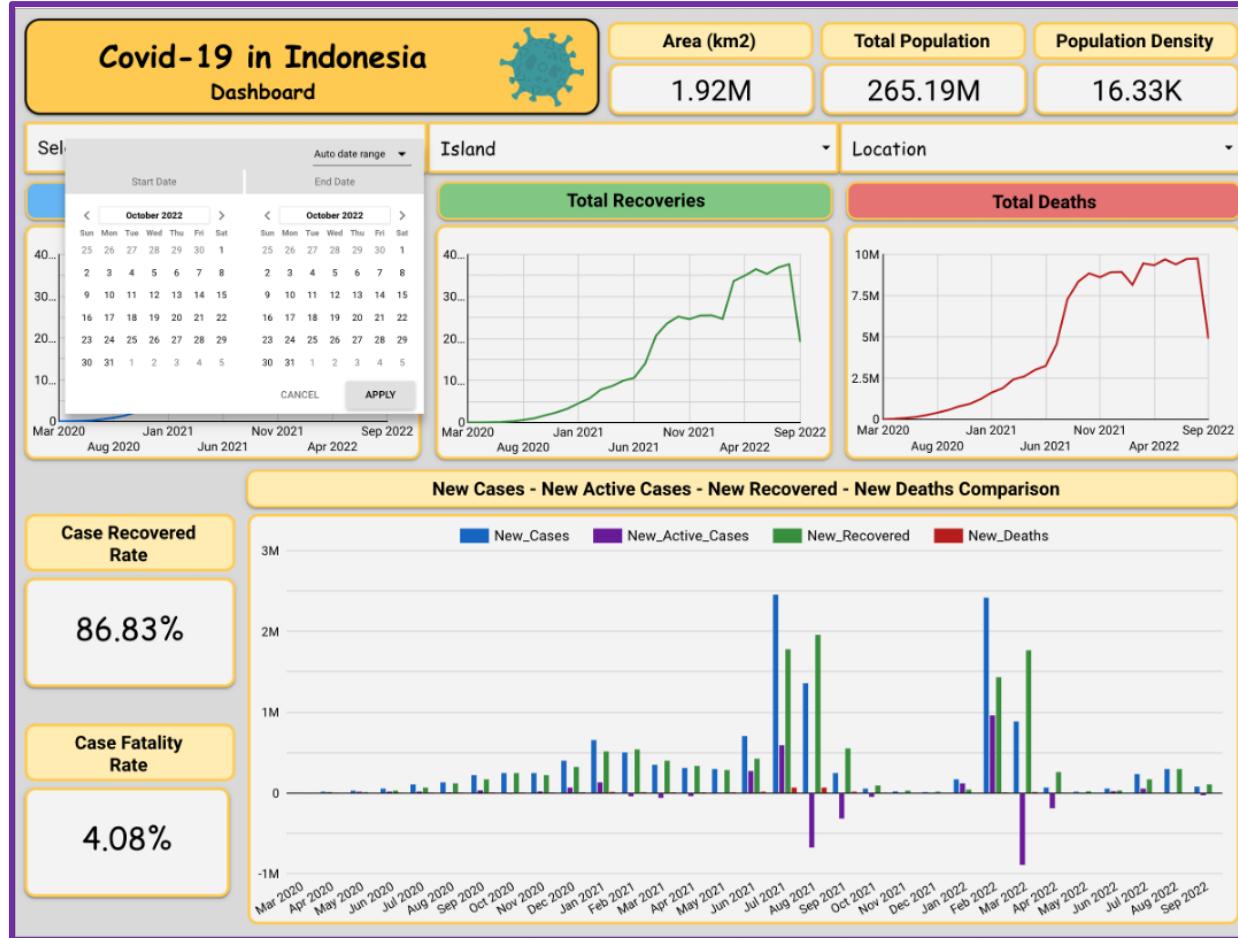
- Google Data Studio adalah BI tools yang dikelola oleh google
- Tools ini digunakan untuk membuat sebuah visualisasi data ke dalam bentuk report dan dashboard



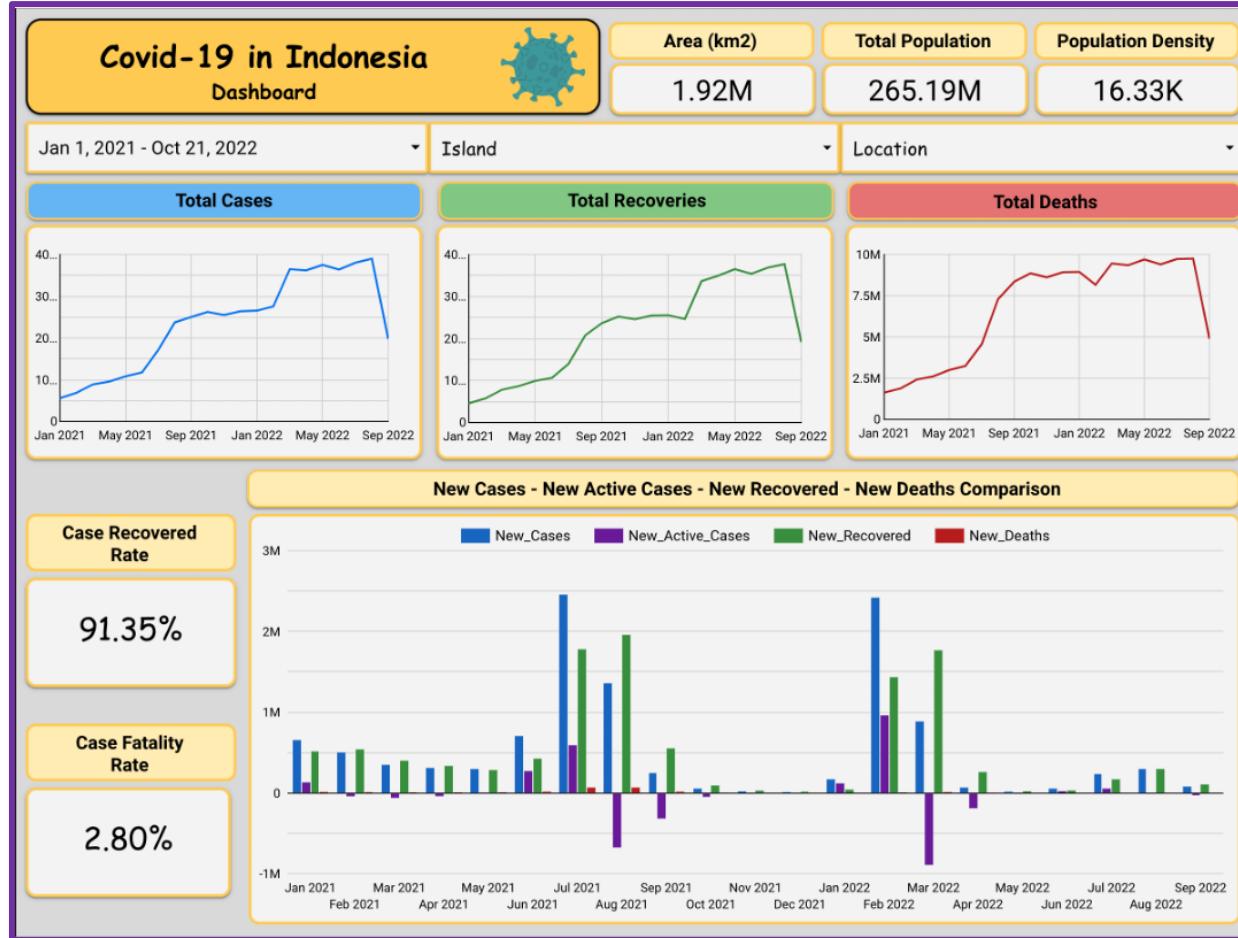
Dashboard



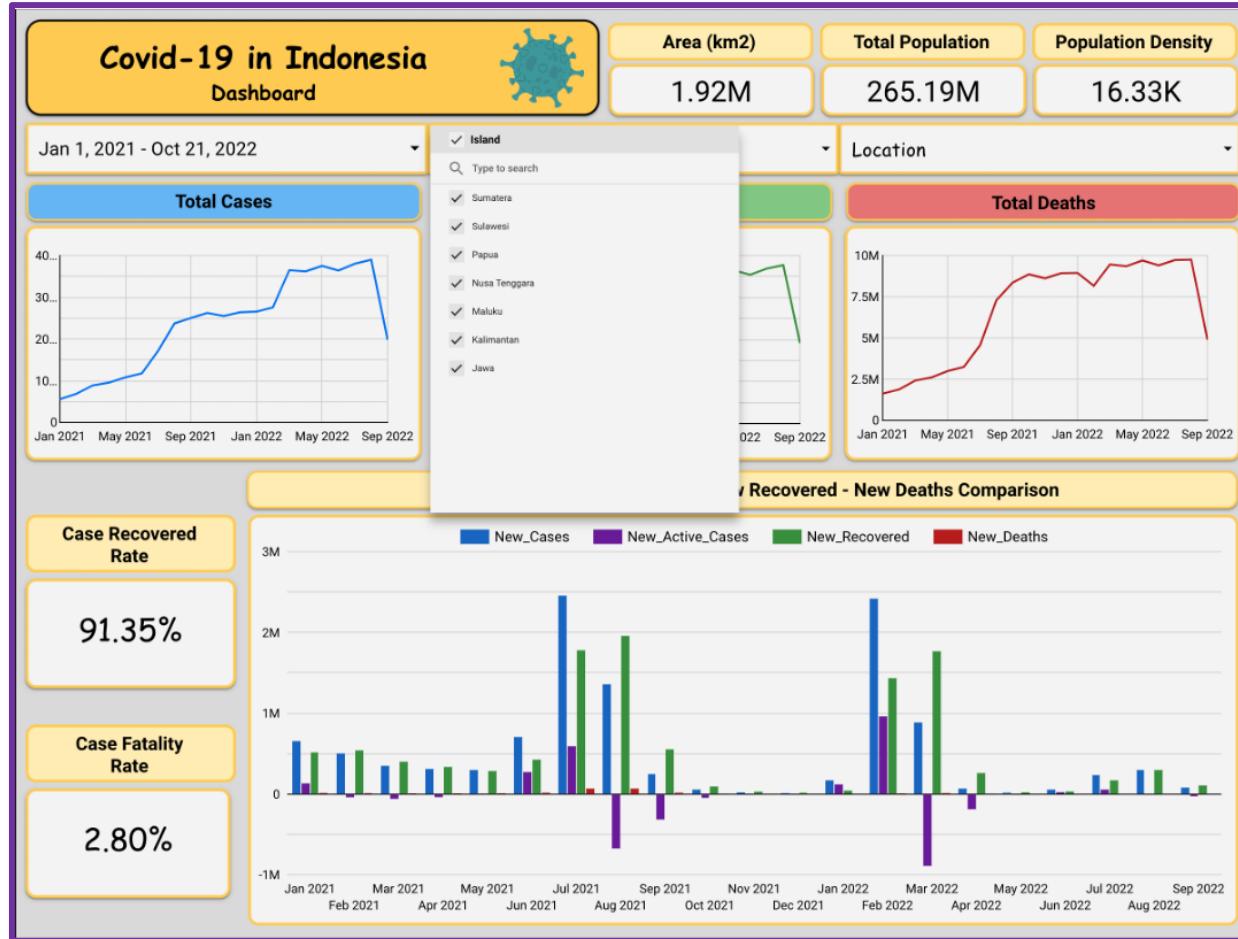
Dashboard Implement



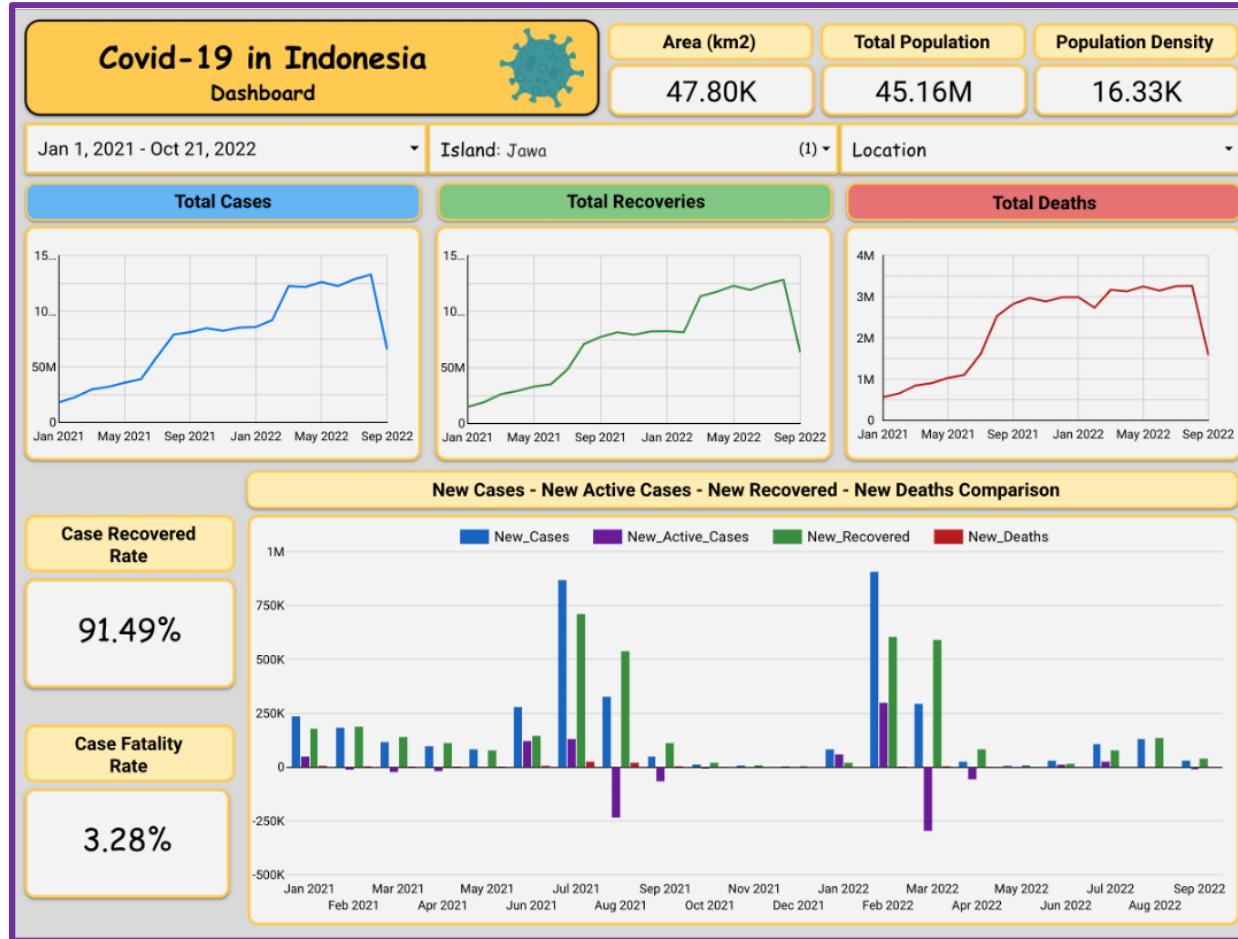
Dashboard Implement



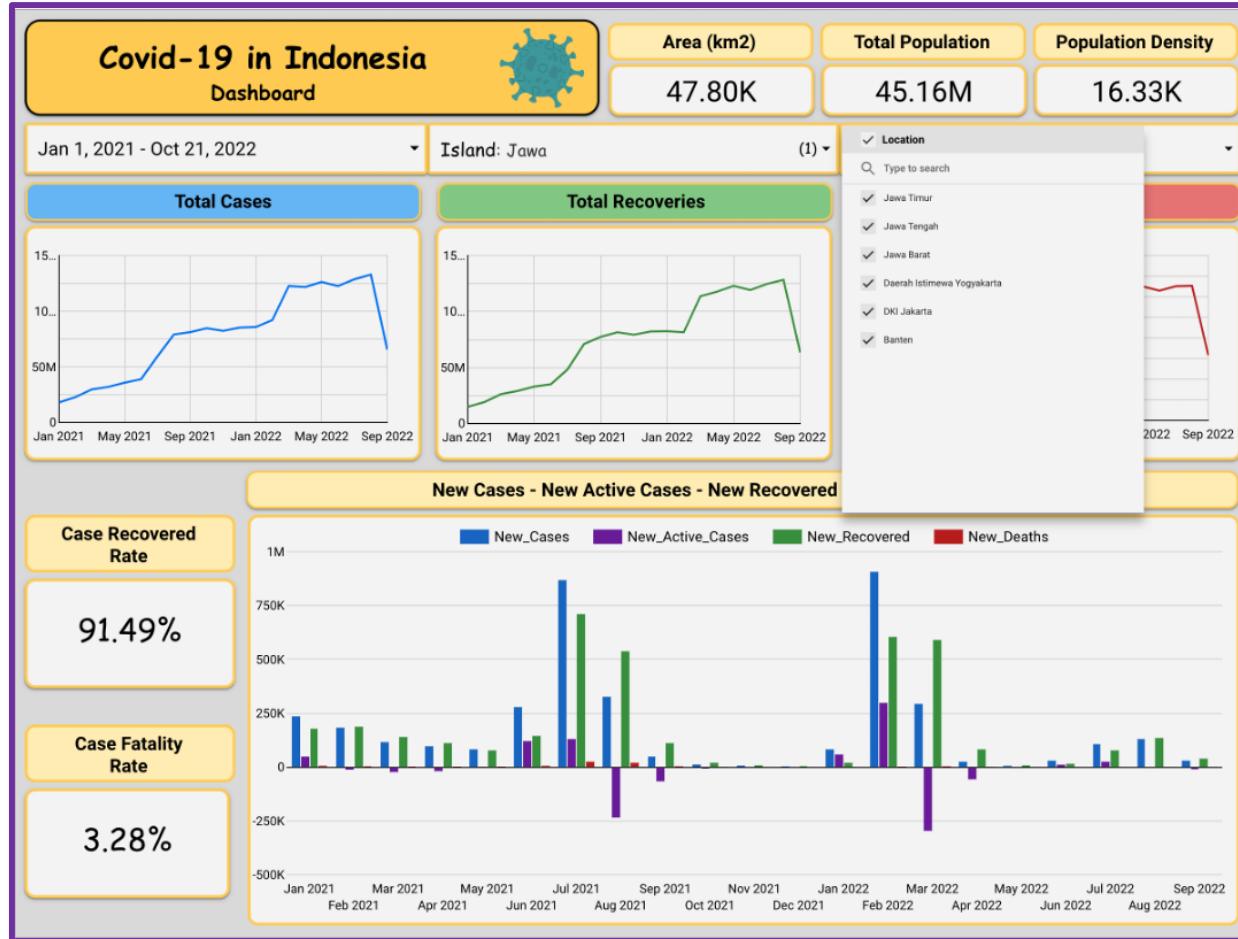
Dashboard Implement



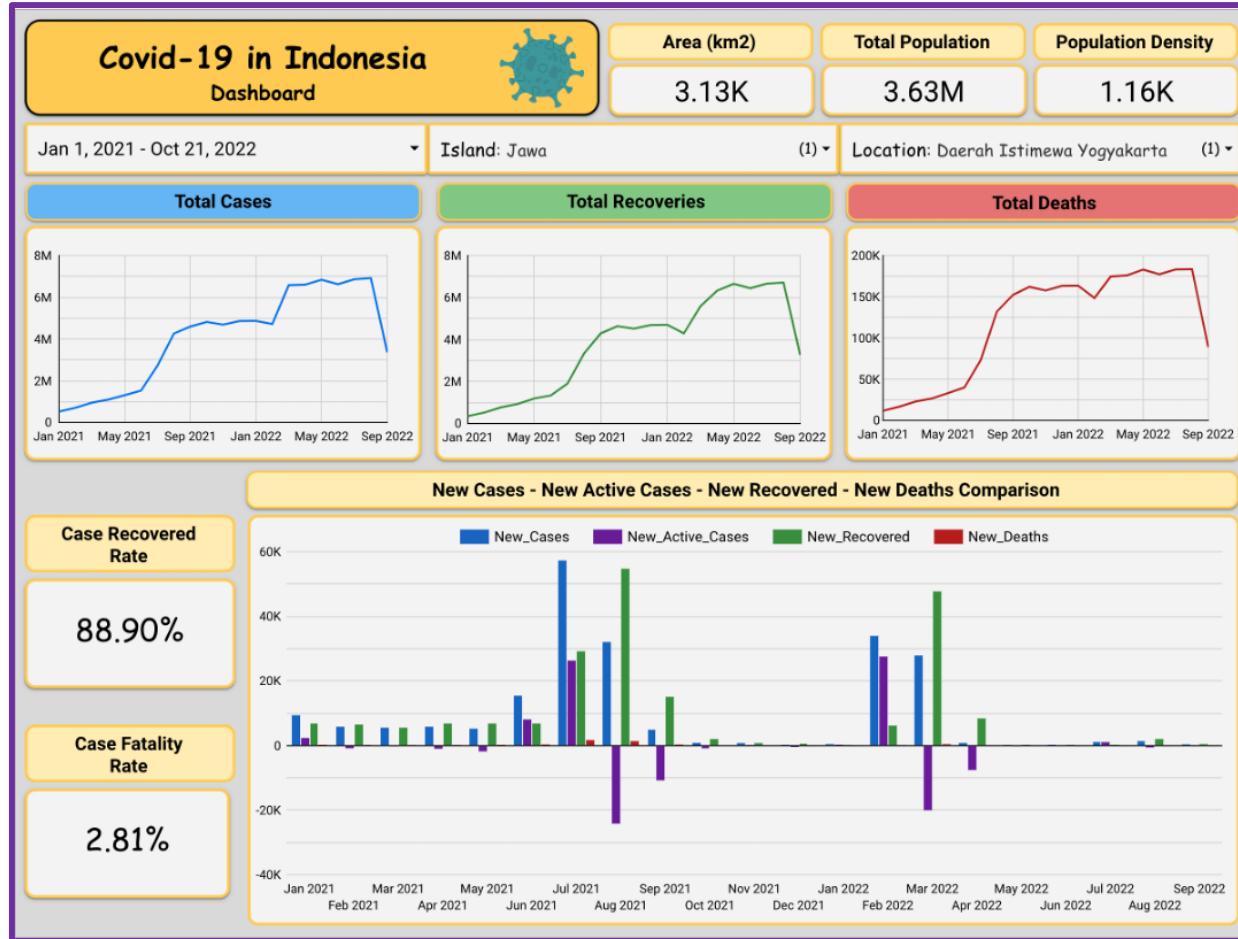
Dashboard Implement



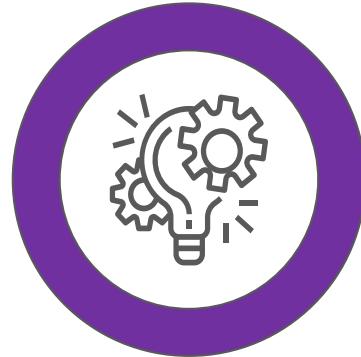
Dashboard Implement



Dashboard Implement

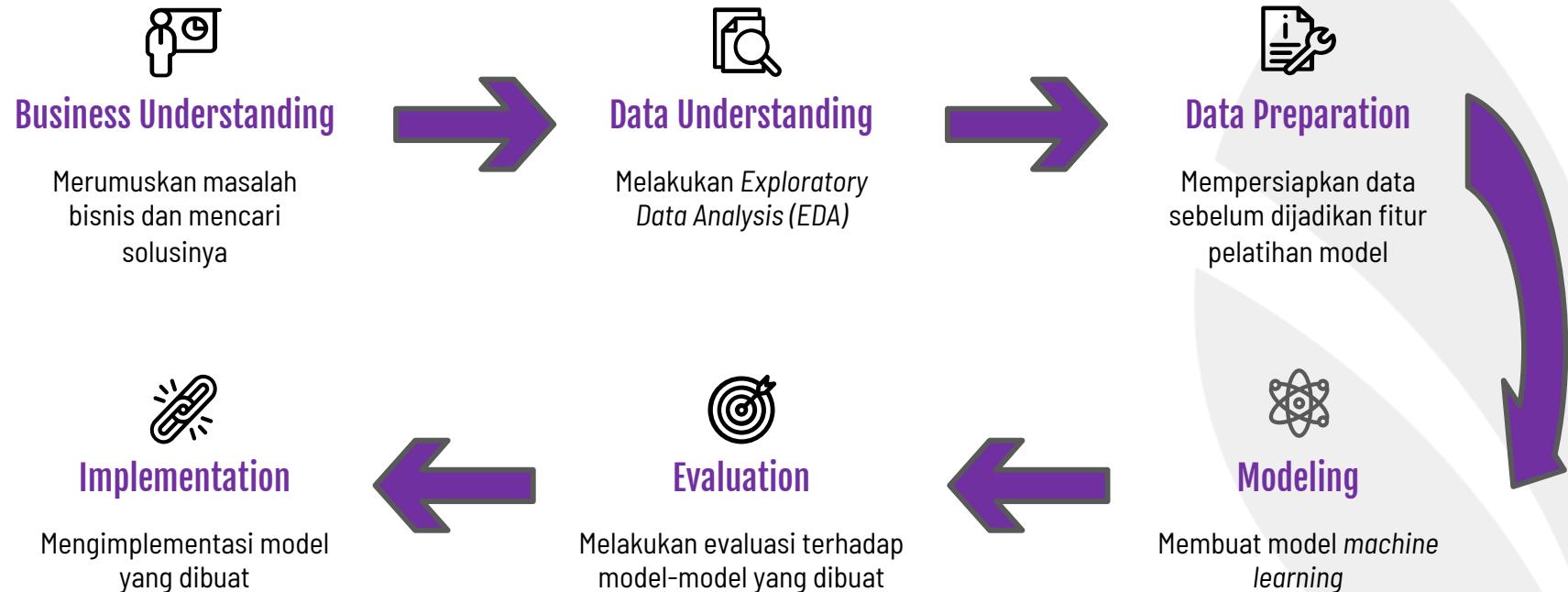


Customers Churn Analysis



Pendahuluan

Pada challenge ini, saya diberi tugas untuk membuat sebuah model untuk mengklasifikasi customer churn. Data yang diberikan pada challenge ini ada dua yaitu data train dan data test.



Business Understanding



Problem and Solution

- Perkembangan industri telekomunikasi yang semakin cepat membuat setiap provider berlomba-lomba untuk memberikan fasilitas terbaiknya.
- Persaingan ini nantinya dapat menimbulkan seorang pelanggan berpindah dari salah satu provider telekomunikasi ke provider lain. Hal ini dapat disebut sebagai **customer churn**.
- Pembuatan model machine learning untuk memprediksi apakah seorang pelanggan akan mengganti provider yang dia gunakan saat ini atau tidak.
- Nantinya, provider dapat memperkirakan fitur apa saja yang dapat mereka perbaiki atau tingkatkan untuk menjaga pelanggan mereka tetap menggunakan providernya.



Data Understanding (EDA)

Data Train

- Dataset disamping adalah dataset train. Dataset ini terdiri dari 4250 baris dan 20 kolom.
- Komposisi dari dataset ini mulai dari total null value hingga jumlah unique valuenya dapat dilihat pada gambar disamping
- Selain itu, dataset ini tidak memiliki baris yang terduplikasi, sehingga dataset ini dapat dikatakan cukup bersih.

	Total_Null	Percentage_Null	Unique_Value
state	0	0.0%	51
account_length	0	0.0%	215
area_code	0	0.0%	3
international_plan	0	0.0%	2
voice_mail_plan	0	0.0%	2
number_vmail_messages	0	0.0%	46
total_day_minutes	0	0.0%	1843
total_day_calls	0	0.0%	120
total_day_charge	0	0.0%	1843
total_eve_minutes	0	0.0%	1773
total_eve_calls	0	0.0%	123
total_eve_charge	0	0.0%	1572
total_night_minutes	0	0.0%	1757
total_night_calls	0	0.0%	128
total_night_charge	0	0.0%	992
total_intl_minutes	0	0.0%	168
total_intl_calls	0	0.0%	21
total_intl_charge	0	0.0%	168
number_customer_service_calls	0	0.0%	10
churn	0	0.0%	2

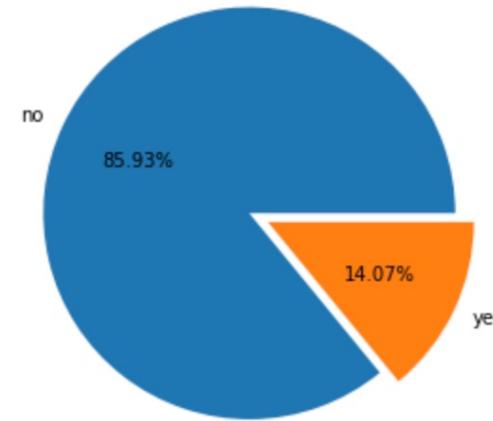
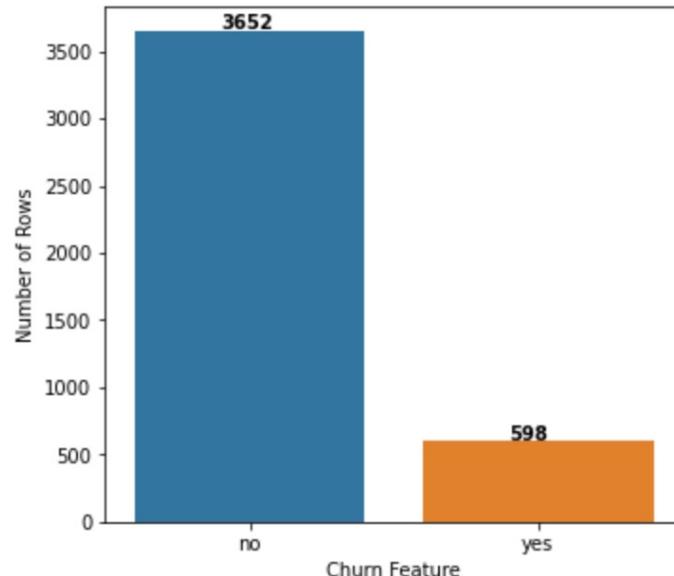
	Total_Null	Percentage_Null	Unique_Value
id	0	0.0%	750
state	0	0.0%	51
account_length	0	0.0%	175
area_code	0	0.0%	3
international_plan	0	0.0%	2
voice_mail_plan	0	0.0%	2
number_vmail_messages	0	0.0%	39
total_day_minutes	0	0.0%	619
total_day_calls	0	0.0%	100
total_day_charge	0	0.0%	619
total_eve_minutes	0	0.0%	611
total_eve_calls	0	0.0%	102
total_eve_charge	0	0.0%	584
total_night_minutes	0	0.0%	628
total_night_calls	0	0.0%	97
total_night_charge	0	0.0%	502
total_intl_minutes	0	0.0%	135
total_intl_calls	0	0.0%	17
total_intl_charge	0	0.0%	135
number_customer_service_calls	0	0.0%	7

Data Test

- Dataset disamping adalah dataset test. Dataset ini terdiri dari 750 baris dan 19 kolom. Perbedaan dataset train dan test adalah kolom customer churn pada dataset ini tidak ada
- Dataset ini tidak memiliki nilai null dan duplikat. Untuk pesebaran dataset ini dapat dilihat pada gambar di samping.

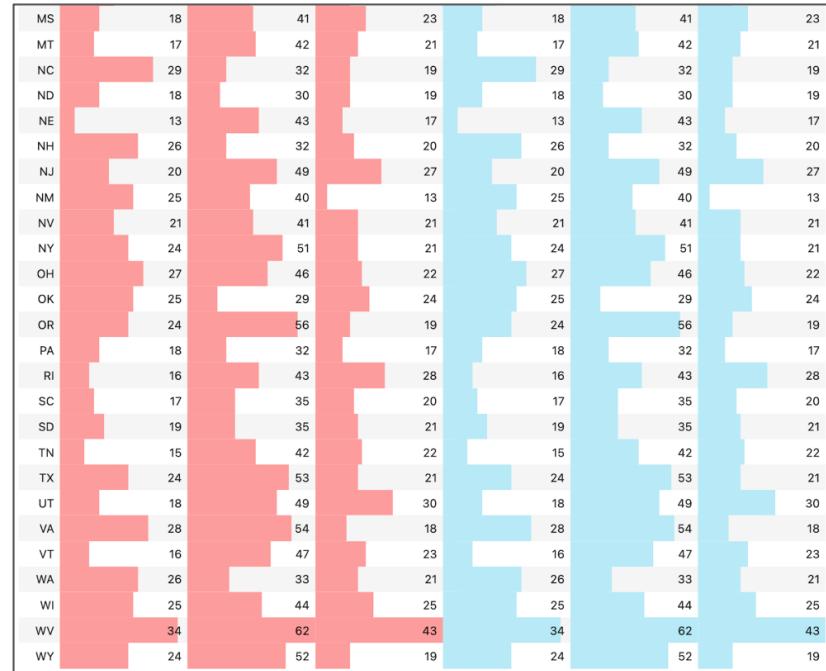
Kolom Output

Output Feature (Churn Column) Value Distribution

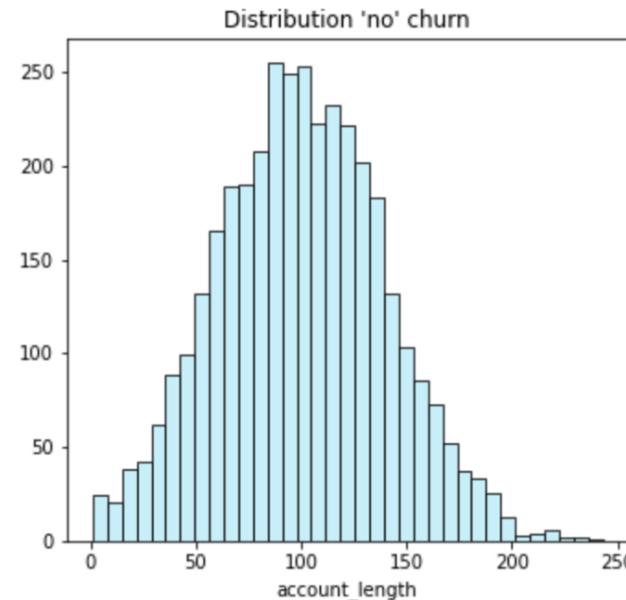
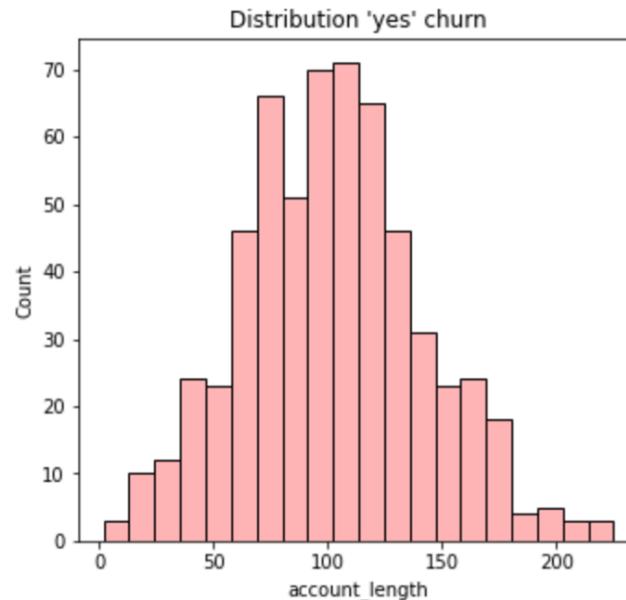


Hal-Hal yang Harus Kamu Lewati

Kolom State dan Area_Code



Kolom Account_Length



Kolom Number_Customer_Service_Calls

number_customer_service_calls	churn	no	yes
	0	789	97
1	1358	166	
2	845	102	
3	495	63	
4	117	92	
5	32	49	
6	9	19	
7	6	7	
8	1	1	
9	0	2	

- Secara persentase terhadap jumlah data per kolom customer churn, customer churn 'yes' cenderung lebih banyak menelpon customer service selama berlanggan dibandingkan customer churn 'no'.
- Di sisi lain, terdapat data yang memiliki nilai tinggi (jumlah telfon ≥ 8) namun hanya berjumlah sedikit

Data Preparation



```
df_train.drop(df_train[df_train['number_customer_service_calls'] >= 8].index, inplace=True)
```

Menghapus baris data pelanggan yang menelfon customer service lebih atau sama dengan 8x

```
del_idx = [ x for x in df_bill[(df_bill['day_ratio'] < .5) |  
                               (df_bill['eve_ratio'] < .5) |  
                               (df_bill['night_ratio'] < .5) |  
                               (df_bill['intl_ratio'] < .5)  
                           ]  
           [[x for x in cols]].index  
       ]  
  
df_train.drop(index=[x for x in del_idx], inplace=True)
```

Menghapus baris data pelanggan yang memiliki rata-rata waktu saat menelfon kurang dari 0.5 menit atau 30 detik



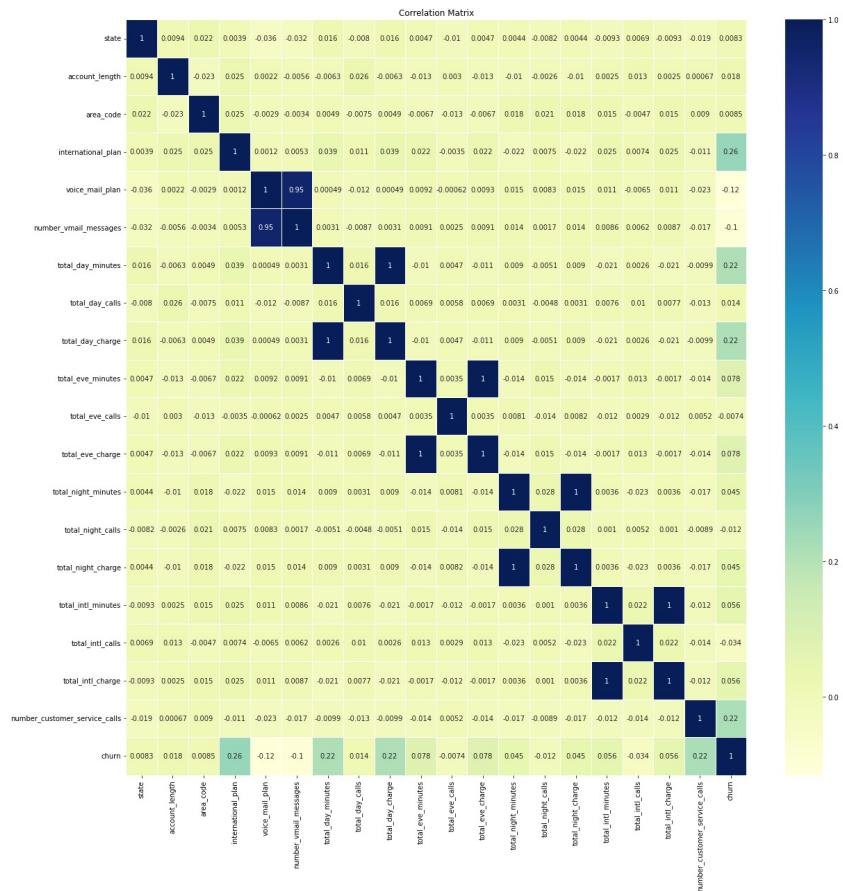
Encoding Data

- Membuat data categorical menjadi data numerik. Hal ini dilakukan agar kolom categorical dapat dijadikan fitur untuk mentraining data. Selain itu, hal ini dilakukan agar dapat dibuat sebuah visualisasi yang memperlihatkan korelasi antar fitur.

```
le = LabelEncoder()

for col in df_train.columns:
    if df_train[col].dtype == 'O':
        df_train[col] = le.fit_transform(df_train[col])
```

Customer Churn Analysis



Features Selection

Fitur yang dipilih adalah:

1. 'account_length',
2. 'area_code',
3. 'international_plan',
4. 'total_day_minutes',
5. 'total_day_calls',
6. 'total_eve_minutes',
7. 'total_eve_calls',
8. 'total_night_minutes',
9. 'total_night_calls',
10. 'total_intl_minutes',
11. 'total_intl_calls', dan
12. 'number_customer_service_calls'

Splitting and Features Scaling

- Membagi data menjadi data train (80%) dan data test (20%)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=.8, random_state=123)

print('x train dimension : {}'.format(X_train.shape))
print('x test dimension : {}'.format(X_test.shape))
print()
print('y train dimension : {}'.format(y_train.shape))
print('y test dimension : {}'.format(y_test.shape))

x train dimension : (3355, 12)
x test dimension : (839, 12)

y train dimension : (3355,)
y test dimension : (839,)
```

- Menscaling data menggunakan Standarisasi

```
:
stand = StandardScaler()

X_train = stand.fit_transform(X_train)
X_test = stand.transform(X_test)
```



Modeling

XGBoost Algorithm

```
# Fit and train the model
xgb_model = XGBClassifier()
xgb_model.fit(X_train,y_train)
xgb_predict = xgb_model.predict(X_test)
```

Naïve Bayes Algorithm

```
# Fit and train the model
nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_predict = nb_model.predict(X_test)
```

SVM Algorithm

```
# Fit and train the model
svm_model = SVC()
svm_model.fit(X_train,y_train)
svm_predict = svm_model.predict(X_test)
```

K-Nearest Neighbor Algorithm

```
# Fit and train the model
knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
knn_predict = knn_model.predict(X_test)
```

Logistic Algorithm

```
# Fit and train the model
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
lr_predict = lr_model.predict(X_test)
```

LDA Algorithm

```
# Fit and train the model
lda_model = LinearDiscriminantAnalysis()
lda_model.fit(X_train, y_train)
lda_predict = lda_model.predict(X_test)
```

Decision Tree Algorithm

```
# Fit and train the model
dtc_model = DecisionTreeClassifier()
dtc_model.fit(X_train, y_train)
dtc_predict = dtc_model.predict(X_test)
```

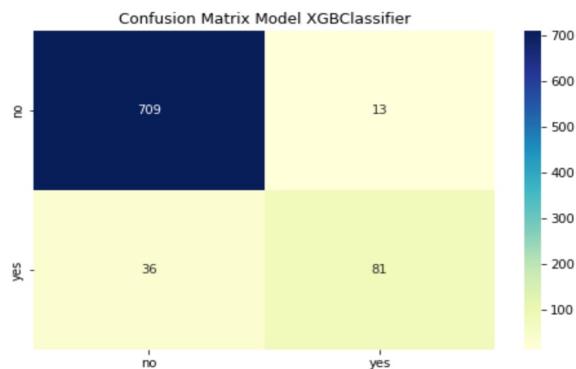
Evaluation



```
# This function is for showing the evaluation metrics for the trained model and its confusion matrix
def eval_model(model, y_test, y_predict):

    print(metrics.classification_report(y_test, y_predict))
    print()
    fig, ax = plt.subplots(figsize=(8,5))
    sns.heatmap(metrics.confusion_matrix(y_test, y_predict), annot=True, cmap="YlGnBu", fmt='g', ax=ax)
    ax.set_title('Confusion Matrix Model {}'.format(model.__class__.__name__))
    ax.xaxis.set_ticklabels(['no', 'yes'])
    ax.yaxis.set_ticklabels(['no', 'yes'])
```

	precision	recall	f1-score	support
0	0.95	0.98	0.97	722
1	0.86	0.69	0.77	117
accuracy			0.94	839
macro avg	0.91	0.84	0.87	839
weighted avg	0.94	0.94	0.94	839



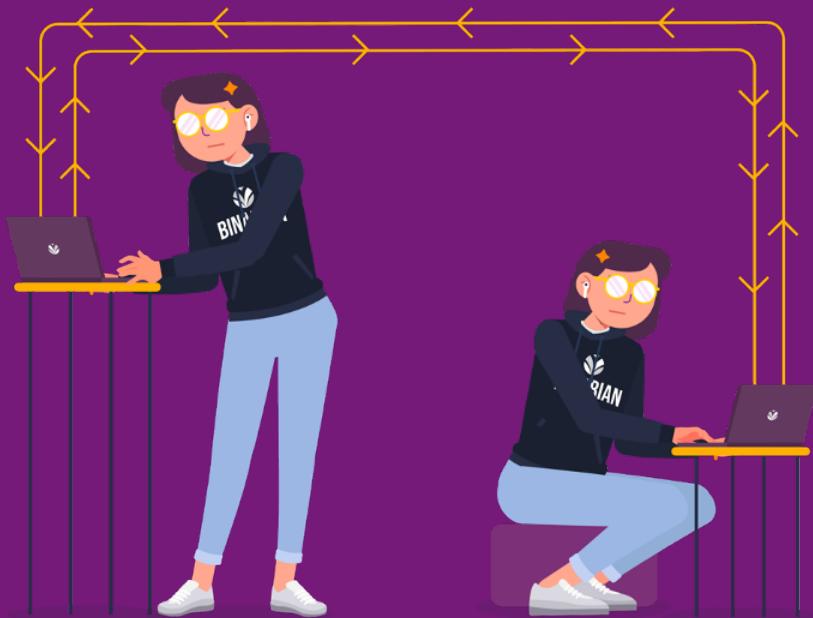
Untuk melakukan evaluasi individu per model, saya membuat sebuah fungsi bernama '**eval_model**'.

Fungsi ini digunakan untuk menampilkan beberapa skor metrik dari model dan confusion matrix.

Evaluation

- Saya melakukan evaluasi model berdasarkan beberapa metrik seperti akurasi, presisi, recall, skor f1 dan AUC. Visualisasi dari model ini dapat dilihat pada gambar di bawah.
- Berdasarkan visualisasi tersebut, didapat jika model terbaik adalah model yang dilatih dengan menggunakan algoritma XGBoost dengan nilai skor dari setiap metriknya termasuk ke dalam skor tertinggi.

	accuracy	precision_no	precision_yes	recall_no	recall_yes	f1_no	f1_yes	roc_auc
XGBoost	0.941597	0.951678	0.861702	0.981994	0.692308	0.966599	0.767773	0.837151
Support_Vector_Machine	0.902265	0.912371	0.777778	0.980609	0.418803	0.945260	0.544444	0.699706
Logistic	0.866508	0.880299	0.567568	0.977839	0.179487	0.926509	0.272727	0.578663
Decision_Tree	0.898689	0.945455	0.629032	0.936288	0.666667	0.940849	0.647303	0.801477
Naive_Bayes	0.851013	0.901750	0.458333	0.927978	0.376068	0.914676	0.413146	0.652023
KNN	0.873659	0.883085	0.657143	0.983380	0.196581	0.930537	0.302632	0.589980
Linear_Discriminant_Analysis	0.855781	0.882803	0.462963	0.959834	0.213675	0.919708	0.292398	0.586755



Implementation

Implementation

- Untuk menggunakan model, saya membuat sebuah fungsi bernama '**use_model**'. Fungsi ini nantinya dapat digunakan untuk melakukan prediksi data baru berdasarkan model XGBoost yang telah saya buat sebelumnya.
- Fungsi ini nantinya akan secara otomatis mengambil fitur yang diperlukan, melabel kolom categorical, melakukan standarisasi data, dan mengimplementasikan model XGBoost.

```
def use_model(df, label_encoder, standarization):
    features = ['account_length', 'area_code', 'international_plan', 'total_day_minutes', 'total_day_calls', ''

    X = df.loc[:,[x for x in features]]
    for col in X.columns:
        if X[col].dtype == 'O':
            X[col] = label_encoder.fit_transform(X[col])
    X = standarization.transform(X)
    df['churn'] = nb_model.predict(X)
    return df
```

Implementation

- Selanjutnya saya menggunakan fungsi model tadi dengan input berupa dataset test. Hasil dari fungsi ini dapat dilihat dibawah ini.
- Saya juga menyimpan dataset baru yang sudah diprediksi customer churnnya dengan nama 'customer_churn_predict.csv'.

```
output = use_model(df_test, le, stand)
print('Predicted column distribution:\n{}\n{}'.format(output['churn'].value_counts()))
print()
output
```

```
Predicted column distribution:
0    654
1     96
Name: churn, dtype: int64
```

0	1	KS	128	area_code_415	no	yes	25	265.1	110
1	2	AL	118	area_code_510	yes	no	0	223.4	98
2	3	IA	62	area_code_415	no	no	0	120.7	70
3	4	VT	93	area_code_510	no	no	0	190.7	114
4	5	NE	174	area_code_415	no	no	0	124.3	76
...
745	746	GA	130	area_code_415	no	no	0	119.4	99
746	747	WA	73	area_code_408	no	no	0	177.2	118
747	748	WV	152	area_code_415	no	no	0	184.2	90
748	749	DC	61	area_code_415	no	no	0	140.6	89
749	750	DC	109	area_code_510	no	no	0	188.8	67

750 rows × 21 columns

Terima Kasih

