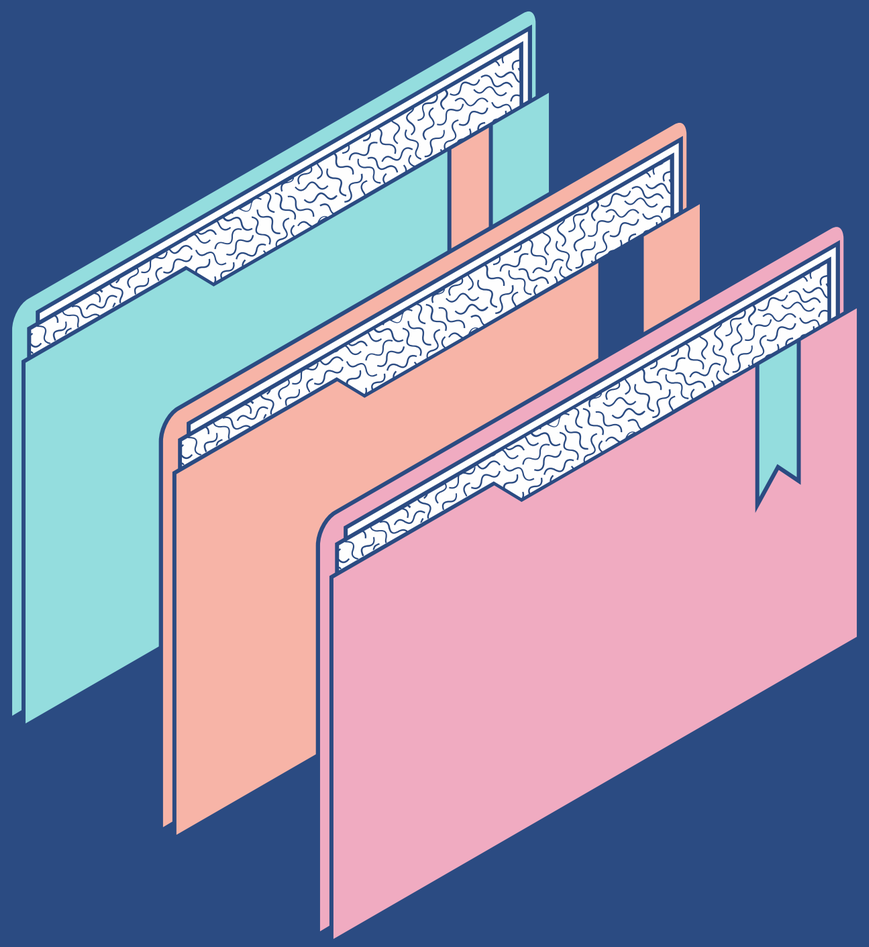
FRESH GRADUATE ACADEMY (FGA)
X
BINAR ACADEMY

Data Visualization & Telecommunication Customer Churn Prediction

Muhammad Randa Yandika



Data Visualization

Indonesia's Covid-19 Dashboard

Dashboard Overview

From this dashboard we can find out things like:

- Number of new and total cases, new and total deaths, new and total recovered, new and total active cases from covid 19 data
- Total number of cases in 2022 in each province
- Monthly chart for new cases, new death and new recovered from march 2020 to september 2022
- Percentage chart for recovered cases in each province
- Table of total active case data in each month in 2022

Data Covid-19 Indonesia

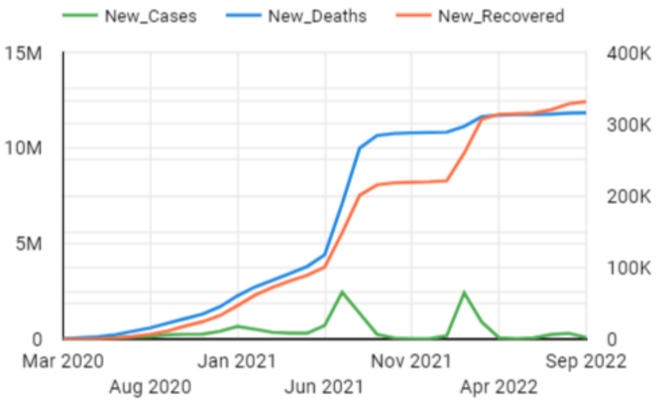
Total Kasus Covid-19 Tahun 2022

Province	Total_Cases
Type to search	
DKI Jakarta	312.1M
Jawa Barat	268M
Jawa Tengah	154.3M
Jawa Timur	140.5M
Banten	69.9M
Daerah Istimewa Yog...	53.4M
Kalimantan Timur	50.6M
Bali	39.1M
Riau	37.7M
Sumatera Utara	37.5M
Sulawesi Selatan	35.3M
Sumatera Barat	26M
Nusa Tenggara Timur	22.4M
Kalimantan Selatan	21.2M
Sumatera Selatan	19.8M
Lampung	17.8M
Kepulauan Riau	17.2M

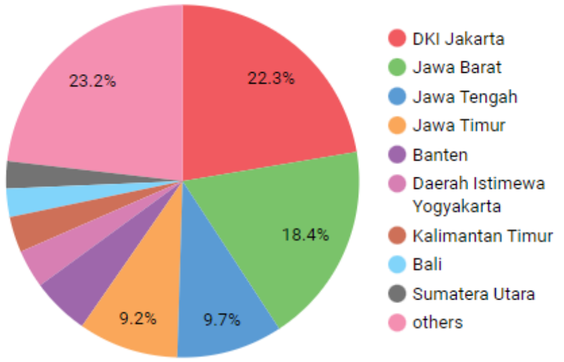
Total_Cases	New_Active_Cases	New_Cases	New_Deaths
5,074,017,827	63,397	12,802,353	315,695

Total_Recovered	Total_Active_Cases	New_Recovered	Total_Deaths
4,749,798,246	178,960,004	12,423,261	145,259,577

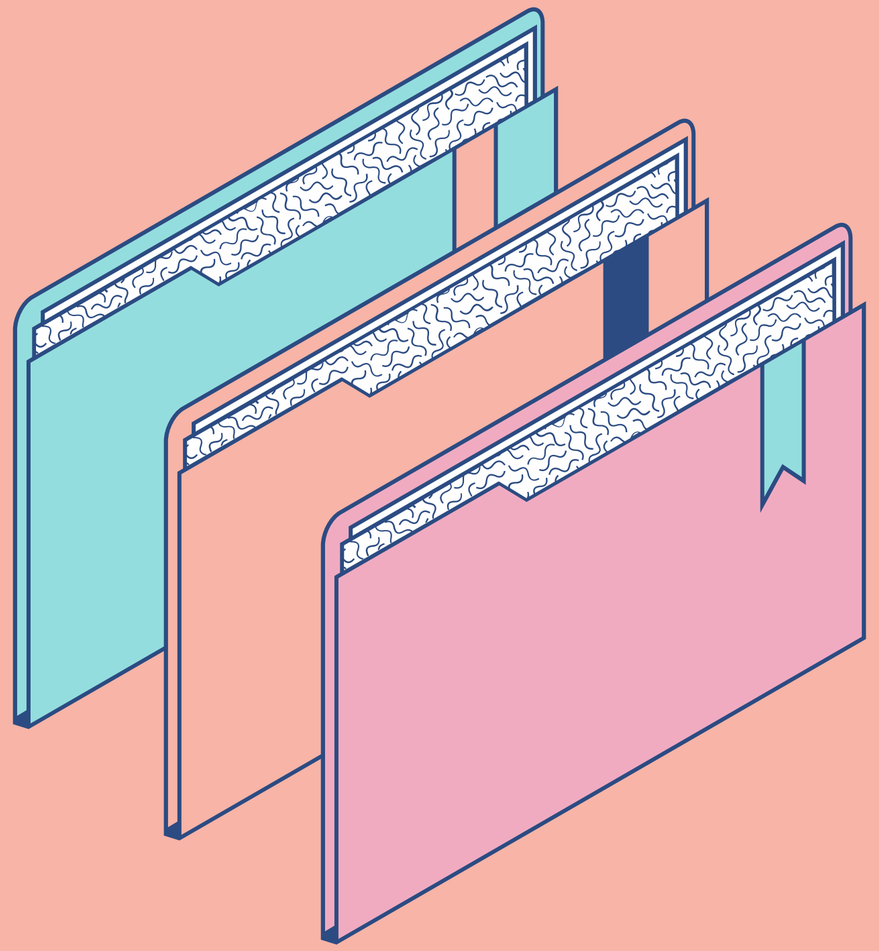
Grafik Bulanan



Persentase Kasus Pulih Covid-19



	Date (Month) / Total_Active_Cases							
Province	January	February	March	April	May	June	July	Augu
DKI Jakarta	62	56	93	90	93	90	93	
Riau	62	56	92	90	93	90	93	
Jawa Barat	62	56	91	90	93	90	93	
Banten	62	56	88	90	93	90	93	
Jawa Tengah	62	56	86	90	93	90	93	
Sulawesi Ten...	62	56	84	90	93	90	93	
Bali	62	56	83	90	93	90	93	
Kalimantan Ti...	62	56	80	90	93	90	93	



Machine Learning Model

Customer Churn Prediction

Table of Content

Data Understanding

Customer churn is losing customers from a business. Churn is calculated by how many customers have left your business in a certain time

This customer churn dataset has 20 rows and 4250 columns

```
[44] import numpy as np
     import pandas as pd
     # Import and read dataset
     df = pd.read_csv('/content/train.csv')
     df.head(5)
```

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

```
[46] 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
```

There is no any duplicated and null data

```
[58] df.duplicated().sum()

     0
```

```
# Checking if any rows are missing any data.
df.isnull().sum()
```

```
account_length                  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
```

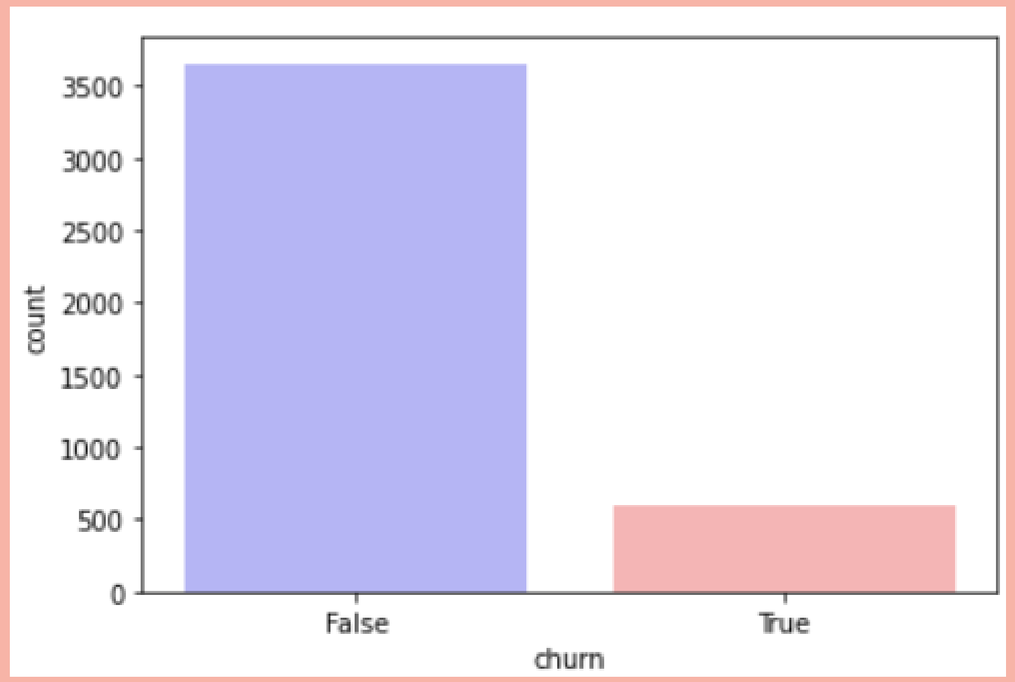
We can see count of unique value for each columns in the dataset

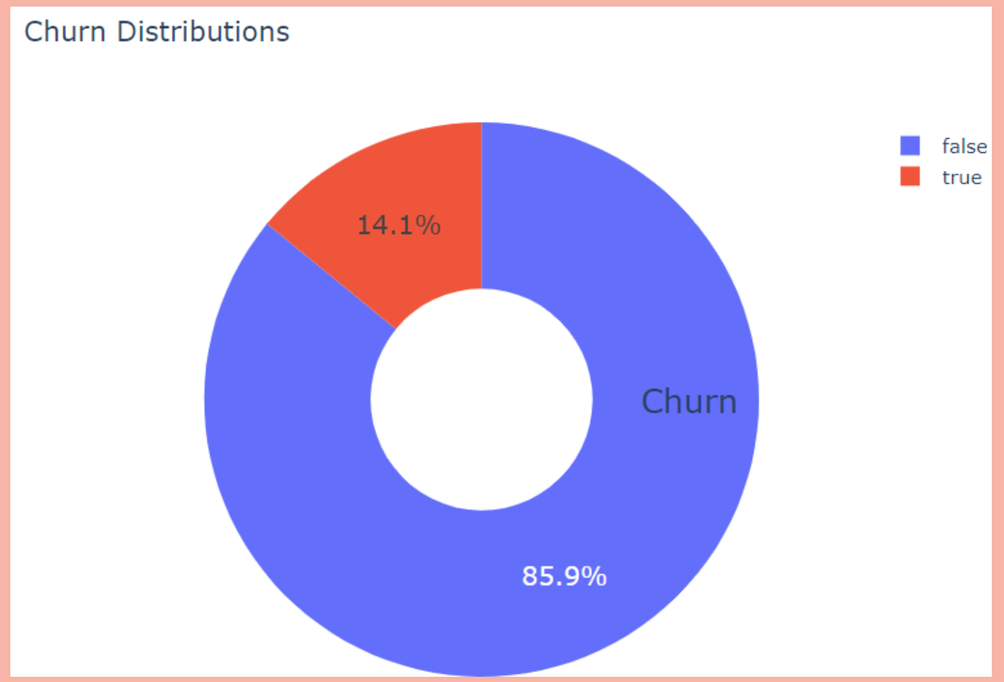
```
df.nunique()
```

```
state                           51
account_length                 215
area_code                        3
international_plan               2
voice_mail_plan                  2
number_vmail_messages           46
total_day_minutes             1843
total_day_calls                120
total_day_charge              1843
total_eve_minutes             1773
total_eve_calls                123
total_eve_charge              1572
total_night_minutes           1757
total_night_calls              128
total_night_charge             992
total_intl_minutes             168
total_intl_calls                21
total_intl_charge              168
number_customer_service_calls   10
churn                            2
dtype: int64
```

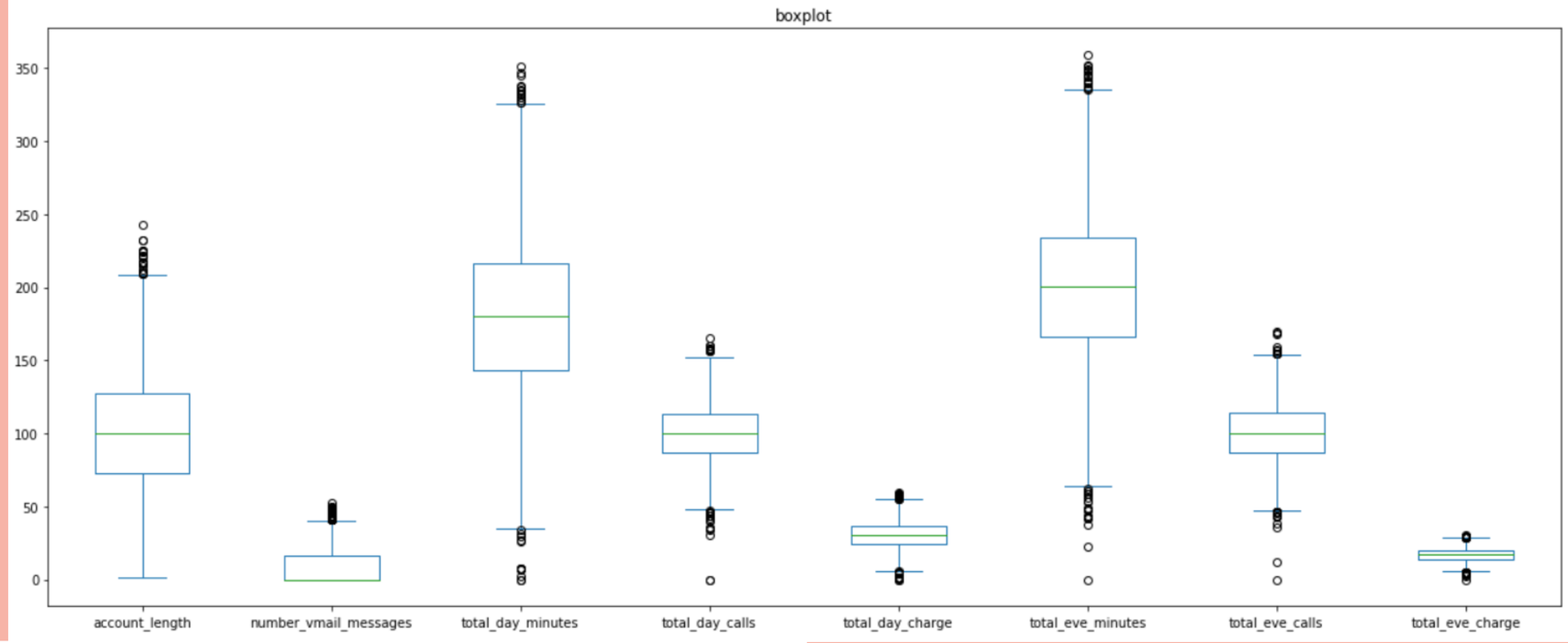
Exploratory Data Analysis

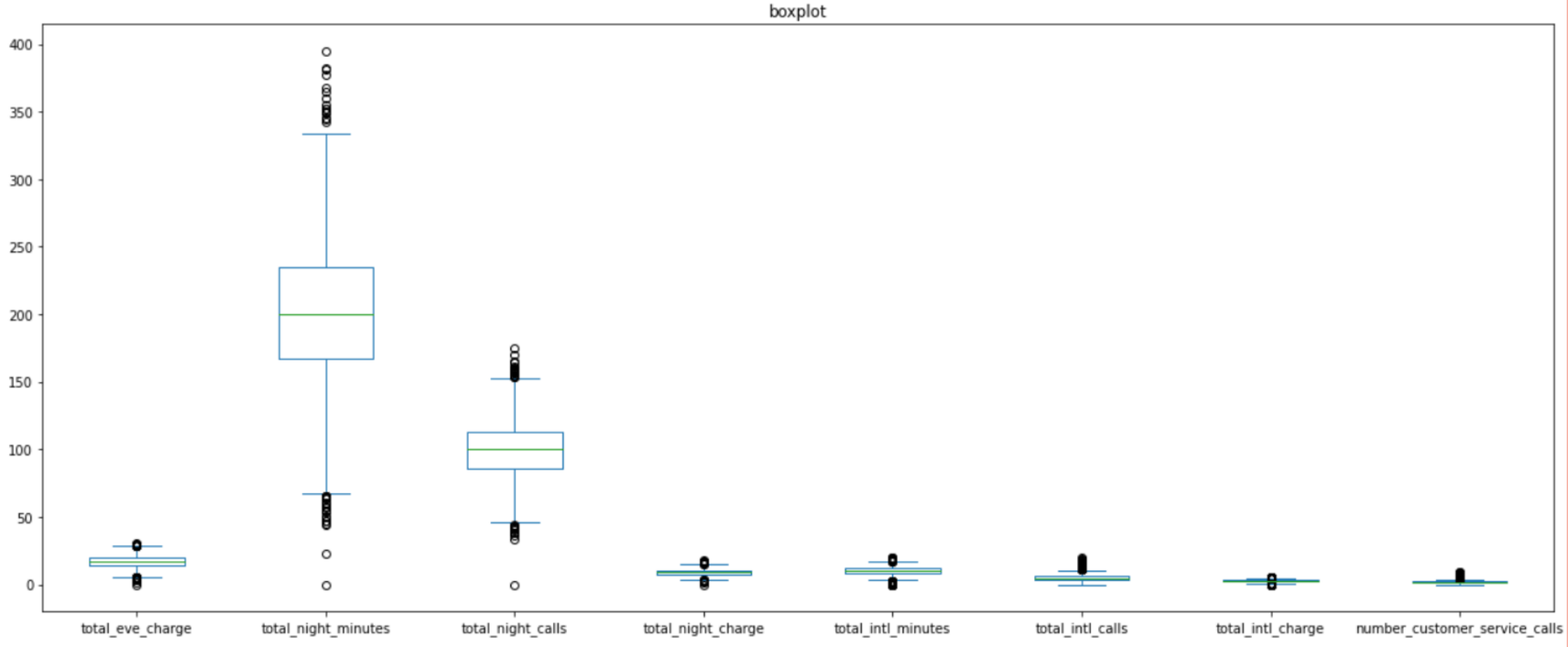
In this chart, seen that amount percentage of lost customers in this data is smaller than customers who are still using this provider



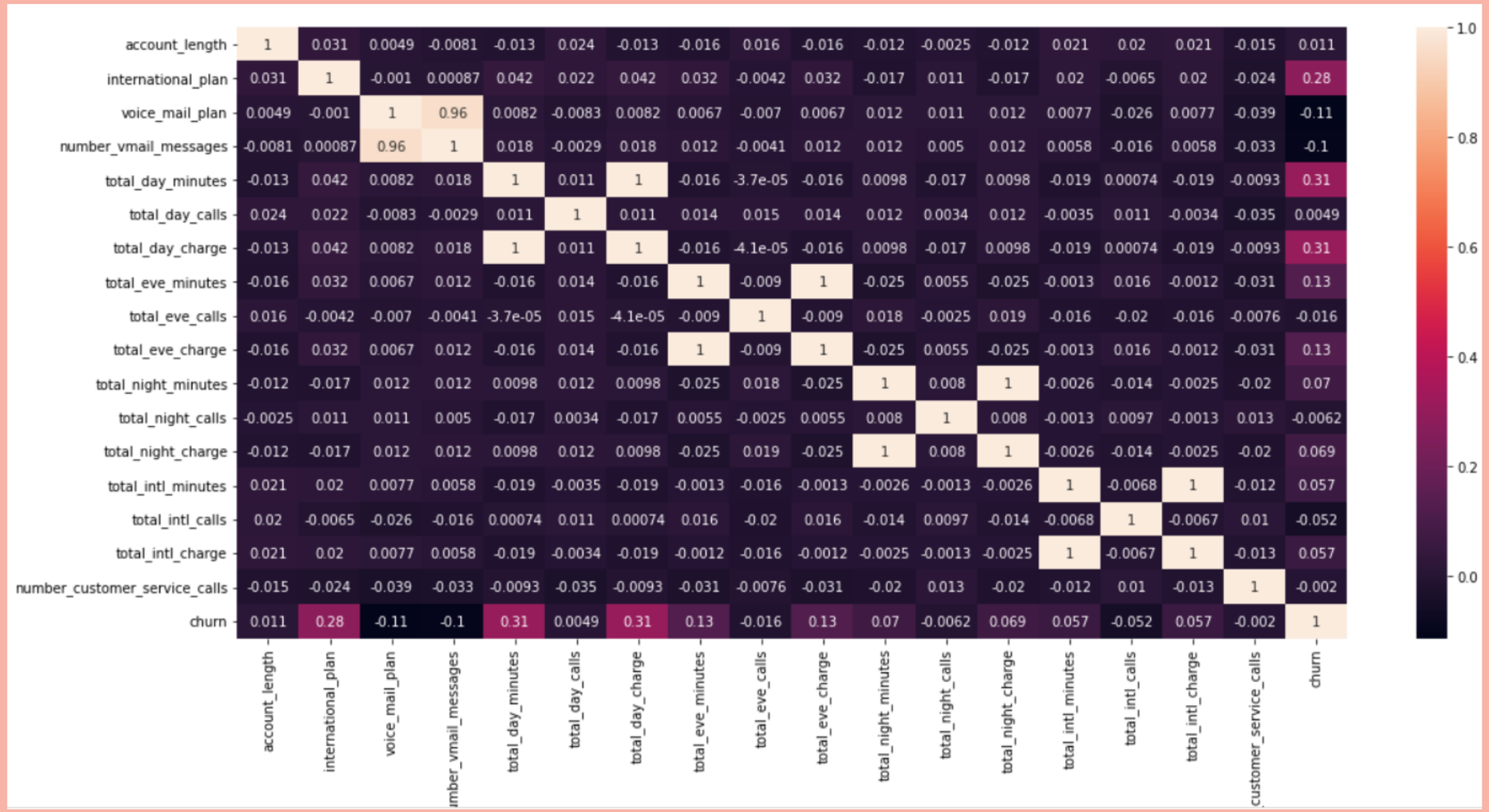


In this boxplot, we can see the outlier in each column





In this heatmap, show the correlation of data



Data Preprocessing

We must split between numerical data and categorical data

```
[178] numerikal =  df.select_dtypes(include=[np.number])
      kategorikal = df.select_dtypes(exclude=[np.number])
```

```
[179] numerikal.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')
```

```
[180] kategorikal.columns
```

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

Data Preprocessing

Deleting ouliers in every column using IQR, form 4250 to 3501 column

```
print(f'Jumlah Baris Sebelum Outlier Dihapus: {len(df)}')
filtered_entries = np.array([True] * len(df))
for col in['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']:

    q1=df[col].quantile(0.25)
    q3=df[col].quantile(0.75)
    iqr=q3-q1

    min_IQR = q1 - (1.5 * iqr)
    max_IQR = q3 + (1.5 * iqr)

    filtered_entries=((df[col]>=min_IQR) & (df[col]<=max_IQR)) & filtered_entries
    df=df[filtered_entries]

print(f'Jumlah Baris Sebelum Outlier Dihapus: {len(df)}')
```

```
Jumlah Baris Sebelum Outlier Dihapus: 4250
Jumlah Baris Sebelum Outlier Dihapus: 3501
```

Data Preprocessing

Drop feature state and area code, because we will not use in modelling process, and change data type of international plan, voice mail plan and churn from object to boolean.

```
[184] df.drop('state', axis=1, inplace=True)
      df.drop('area_code', axis=1, inplace=True)
```

```
[185] df['international_plan'] = df['international_plan'].map(
                          {'yes':True ,'no':False})
      df['voice_mail_plan'] = df['voice_mail_plan'].map(
                          {'yes':True ,'no':False})
      df['churn'] = df['churn'].map(
                          {'yes':True ,'no':False})
```

Data Preprocessing

We set x as column data to be trained and y as label

Split data into train and test 80:20

```
X = df.drop(columns='churn')
X.head()
```

	account_length	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls
0	107	False	True	26	161.6	123
1	137	False	False	0	243.4	114
3	75	True	False	0	166.7	113
5	147	True	False	0	157.0	79
7	141	True	True	37	258.6	84

```
y = df['churn']
y.head()
```

```
0    False
1    False
3    False
5    False
7    False
Name: churn, dtype: bool
```

```
from sklearn.model_selection import train_test_split,cross_validate
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

Machine Learning - Naive Bayes

```
[193] from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score, roc_curve
      from sklearn.metrics import classification_report

      print("confussion matrix")
      print(confusion_matrix(y_test, y_pred))
      print("--------------------------------------------")
      print("Accuracy of Naive Bayes: {:.2f}%".format(akurasi))
      print(classification_report(y_test,nbpred))
      print(f'ROC_AUC score: {roc_auc_score(y_test, nbpred)}.')
```

```
confussion matrix
[[607  27]
 [ 20  47]]
--------------------------------------------
Accuracy of Naive Bayes: 93.30%
              precision    recall  f1-score   support

       False       0.97      0.96      0.96       634
        True       0.64      0.70      0.67        67

    accuracy                           0.93       701
   macro avg       0.80      0.83      0.81       701
weighted avg       0.94      0.93      0.93       701

ROC_AUC score: 0.8294528932623947.
```

Machine Learning - KNN

```
[195] print("confussion matrix")
      print(confusion_matrix(y_test, predicted_knn))
      print("----------------------------------------")
      print("Accuracy of KNN: {:.2f}%".format(akurasi_knn))
      print(classification_report(y_test,predicted_knn))
      print(f'ROC_AUC score: {roc_auc_score(y_test, predicted_knn)}.')
```

```
confussion matrix
[[629   5]
 [ 41  26]]
----------------------------------------
Accuracy of KNN: 93.44%
              precision    recall  f1-score   support

       False       0.94      0.99      0.96       634
        True       0.84      0.39      0.53        67

    accuracy                           0.93       701
   macro avg       0.89      0.69      0.75       701
weighted avg       0.93      0.93      0.92       701

ROC_AUC score: 0.6900866330806535.
```

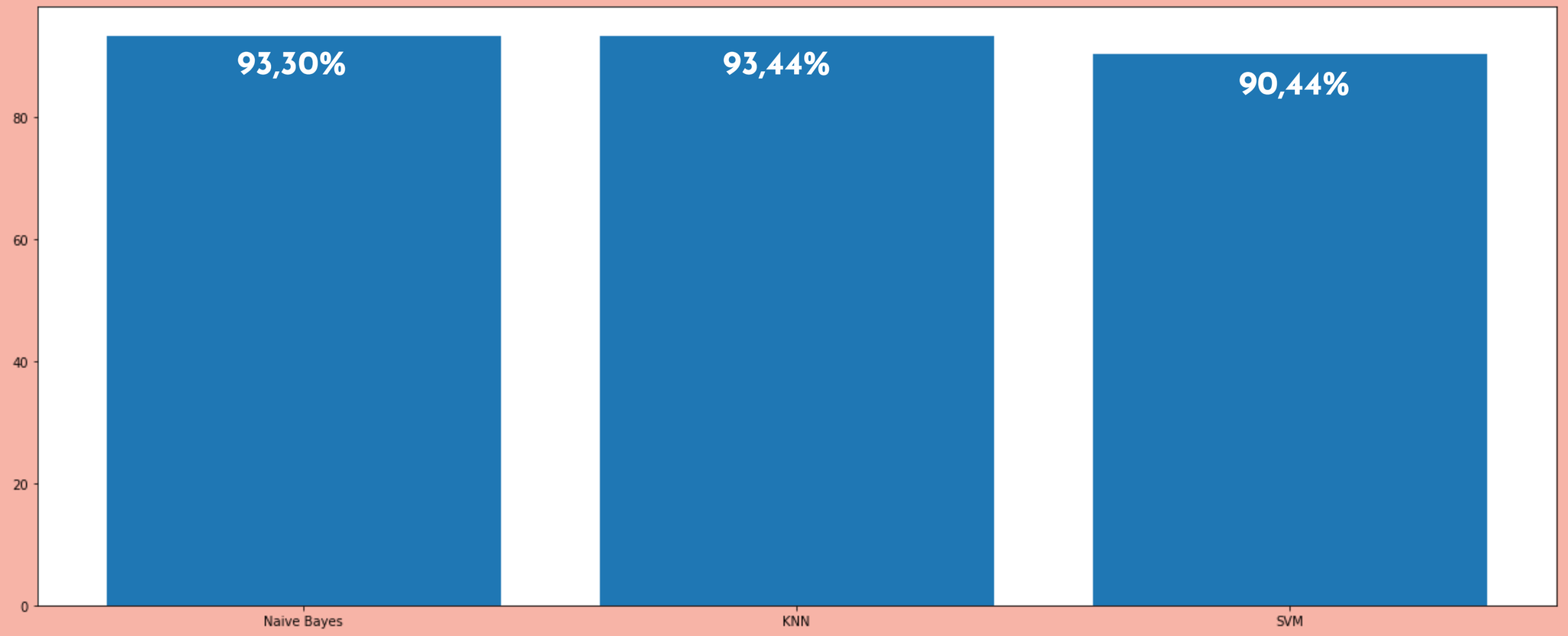
Machine Learning - SVM

```
[197] print("confussion matrix")
      print(confusion_matrix(y_test, predicted_svm))
      print("-----------------------------------------")
      print("Accuracy of SVM: {:.2f}%".format(akurasi_svm))
      print(classification_report(y_test,predicted_svm))
      print(f'ROC_AUC score: {roc_auc_score(y_test, predicted_svm)}.')
```

```
confussion matrix
[[634   0]
 [ 67   0]]
-----------------------------------------
Accuracy of SVM: 90.44%
              precision    recall  f1-score   support

       False       0.90      1.00      0.95       634
        True       0.00      0.00      0.00        67

    accuracy                           0.90       701
   macro avg       0.45      0.50      0.47       701
weighted avg       0.82      0.90      0.86       701

ROC_AUC score: 0.5.
```

Model Result

From 3 different method, final result is KNN has highest accuracy with 93,44%, Naive Bayes has 93,3% and SVM has 90,44%



Data Test Label Prediction

With 3 model using Naive Bayes, KNN, and SVM previously made, we predict data test with same feature to know customer is churn or not.

Data Test preprocessing step is same as data train before.

Naive Bayes Prediction

NAIVE BAYES

```
[259] predicted_baru = nb.predict(X_baru)
```

```
[260] #Hasil prediksi untuk klasifikasi untuk data yang labelnya belum diketahui belum diketahui
      y_pred_baru = nb.predict(X_baru)
      y_pred_baru
```

```
array([False,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False, False,  True, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False,  True,
       False, False, False,  True, False, False, False, False, False,
        True, False, False, False, False, False,  True, False, False,
       False,  True, False, False, False, False, False, False,  True,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
        True,  True, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False,  True, False, False,  True, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False,  True, False, False, False, False, False, False,
       False, False, False, False,  True, False, False, False, False,
        True, False, False, False, False, False, False, False, False,
```

Prediction Result

```
a=y_pred_baru
unique, counts = np.unique(a, return_counts=True)
dict(zip(unique, counts))
```

```
{False: 544, True: 67}
```

Churn

False = 544

True = 67

More Customers leave the provider

KNN Prediction

KNN

```
[263] prediksi_baru = knn_model.predict(X_baru)
```

```
[264] y_prediksi_baru = knn_model.predict(X_baru)
      y_prediksi_baru
```

```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False,  True, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

Prediction Result

```
[265] b=y_prediksi_baru
      unique, counts = np.unique(b, return_counts=True)
      dict(zip(unique, counts))
```

```
{False: 593, True: 18}
```

Churn

False = 593

True = 18

More Customers leave the provider

SVM Prediction

SVM

```
[267] svm_pred = clf.predict(X_baru)
```

```
[268] y_prediksi_svm = clf.predict(X_baru)
      y_prediksi_svm
```

```
array([False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
       False, False, False, False, False, False, False, False, False,
```

Prediction Result

```
[269] c=y_prediksi_svm
      unique, counts = np.unique(c, return_counts=True)
      dict(zip(unique, counts))

      {False: 611}
```

Churn
False = 611
True = 0
More Customers leave the provider

Thank you..