

AlwandiaRW - Visualization & Machine Learning

Challenge
Chapter 1 & Chapter 2





Challenge 1 - Visualization

Dashboard Covid-19 Indonesia



INDONESIA COVID-19 DISTRIBUTION DASHBOARD

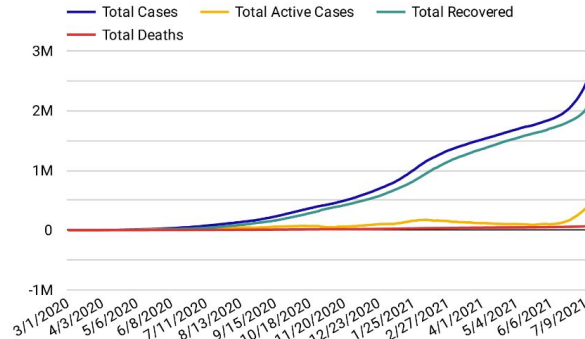
Total Cases
5,074,017,827

Total Active Cases
178,960,004

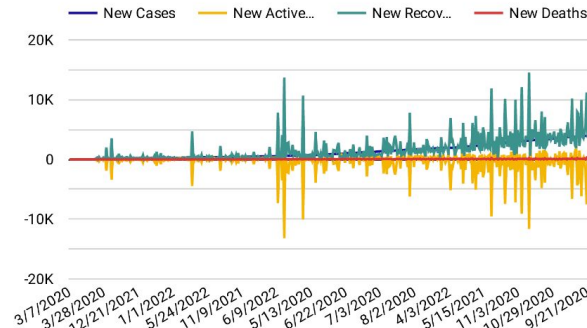
Total Recovered
4,749,798,246

Total Deaths
145,259,577

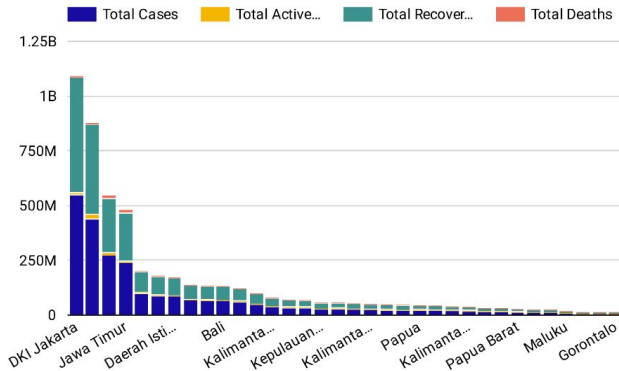
DAILY CASES



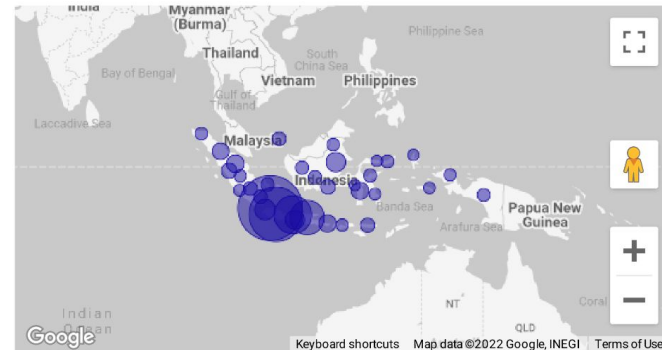
DAILY NEW CASES



CASES PER PROVINCE



CASES DISTRIBUTION MAP



Total Cases 6,746,166 • 545,806,047



Challenge 2

Case Study - Problem and Task



Case Study - Problem

Perkembangan industri telekomunikasi sangatlah cepat, hal ini dapat dilihat dari perilaku masyarakat yang menggunakan internet dalam berkomunikasi.

Perilaku ini menyebabkan banyaknya perusahaan telekomunikasi dan meningkatnya *internet service provider* yang dapat menimbulkan persaingan antar provider.

Pelanggan memiliki hak dalam memilih provider yang sesuai dan dapat beralih dari provider sebelumnya yang diartikan sebagai **Customer Churn**.

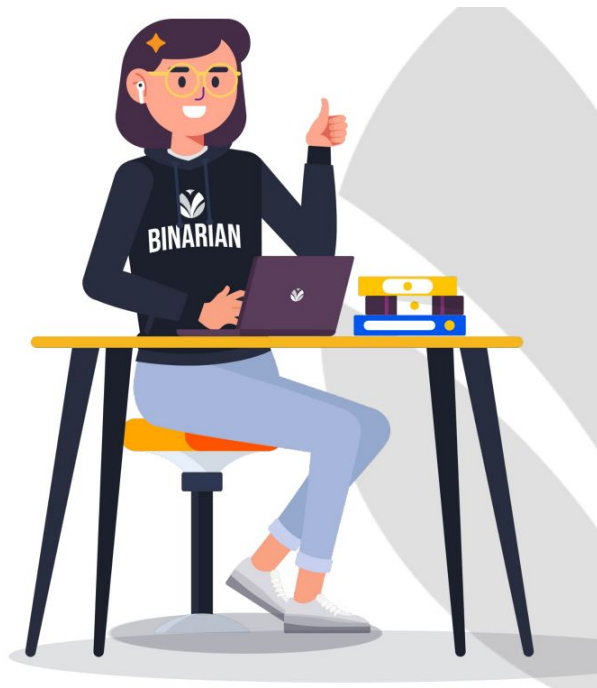
Peralihan ini dapat menyebabkan berkurangnya pendapatan bagi perusahaan telekomunikasi sehingga penting untuk ditangani.

Kolom	Definisi
state	US State
account_length	Total bulan customer menjadi user telco provider
area_code	Kode Area
international_plan	Customer memiliki plan international
voice_mail_plan	Customer memiliki plan voice mail
number_vmail_messages	Total pesan voice mail
total_day_minutes	Total minutes pada day calls
total_day_calls	Total day calls
total_day_charge	Total charge dari day calls
total_eve_minutes	Total menit pada evening call
total_eve_calls	Total evening call
total_eve_charge	Total charge pada evening call
total_night_minutes	Total menit pada night call
total_night_calls	Total night call
total_night_charge	Total charge pada night call
total_intl_minutes	Total menit pada international call
total_intl_calls	Total international call
total_intl_charge	Total charge pada international call
number_customer_service_calls	Total call kepada customer service
churn	Customer churn

Case Study - Task

Tugas:

1. Buat model machine learning dengan algoritma klasifikasi (supervised learning) menggunakan [data train.csv](#)
2. Lakukan prediksi customer yang churn dari hasil model (poin 1) menggunakan [data test.csv](#)
3. Kumpulkan code dalam bentuk file .ipynb/Google Colab dan hasil interpretasi dalam bentuk ppt / pdf mengenai *step by step* penyelesaian dan hasil yang sudah dilakukan menggunakan form submission yang disediakan oleh Tim Binar maksimal **14 Oktober 2022 pukul 23.59 WIB**





Step by Step

Import - EDA - Preprocessing - Modelling -
Evaluating & Tuning - Prediction Test



Step by Step - Import Data

Hal yang pertama kali dilakukan untuk melakukan suatu pemodelan machine learning dalam memprediksi Customer Churn adalah dengan mengimport **data_train.csv** dan **data_test.csv**.

Kemudian, data yang telah diimport atau diupload disimpan pada suatu variabel, lalu menambahkan variabel baru sebagai **copy dari variabel data asli**.

Hal ini bertujuan apabila terdapat kesalahan dalam proses modelling, **data asli tidak berubah dan dapat dilakukan run ulang**.

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


Choose Files No file chosen
cell to enable.
Saving test.csv to test.csv

```
1 import pandas as pd
2 dtrain = pd.read_csv('/content/train.csv')
3 dtest = pd.read_csv('/content/test.csv')
4
5 df_train = dtrain.copy()
6 df_test = dtest.copy()
```




EDA (Exploratory Data Analysis)

info(), head(), & describe()



Step by Step - EDA (info, head, describe)

Selanjutnya tahap **EDA (Exploratory Data Analysis)** dilakukan untuk **investigasi awal** pada data dengan tujuan menemukan pola, anomali, menguji hipotesis dan dapat memeriksa asumsi dengan bantuan statistik ringkasan kemudian representasi grafis (visualisasi).

Pada data_train di samping dilakukan **df_train.info()** sehingga diketahui terdapat 20 fitur atau kolom dengan fitur kategori sebanyak 4 buah ditandai dengan Dtype=Object.

```
1 df_train.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
```



Step by Step - EDA (info, head, describe)

```
1 df_train.head()
```

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge
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

Perlu dilakukan juga **head()** untuk melihat isi data berupa apa sehingga dapat dilakukan kategorisasi dan menentukan hubungan variabel.

Step by Step - EDA (info, head, describe)

`info()` juga dijalankan pada `data_test` sehingga diketahui terdapat 20 fitur seperti pada data train. Namun, terdapat satu fitur yang berbeda dan dihilangkan, yakni fitur `churn` pada data train diganti menjadi fitur `id` pada data test.

Fitur `churn` merupakan target yang akan diprediksi sehingga jelas pada data test tidak terdapat fitur tersebut. Namun, sebagai ganti fitur `churn`, fitur yang unik atau `id` ditambahkan.

```
1 df_test.info()

<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
```



Step by Step - EDA (info, head, describe)

```
1 df_train.head()
```

	state	account_length	area_code	international_plan	voice_mail_plan	number_vmail_messages	total_day_minutes	total_day_calls	total_day_charge
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

Perlu dilakukan juga **head()** pada data test untuk melihat isi data berupa apa sehingga dapat dilakukan prediksi menggunakan machine learning.

Step by Step - EDA (info, head, describe)

describe() dijalankan untuk melihat mulai dari nilai min dan max, quartile, standar deviasi, dan sebagainya. Hal ini untuk melihat rentang data dan persebaran data.

Pada hasil **describe()** data train persebaran data pada tiap total calls cukup lebar kecuali pada total **international calls** dan **number customer service calls**. Ini dapat menunjukkan ada sesuatu yang menarik pada fitur international calls termasuk **international plan** dan number customer service calls terhadap fitur **churn**.

Fitur international plan termasuk karena hanya customer yang memiliki fitur tersebut yang melakukan international calls.

```
1 df_train.describe().T
```

	count	mean	std	min	25%	50%	75%	max
account_length	4250.0	100.236235	39.698401	1.0	73.0000	100.00	127.0000	243.00
number_vmail_messages	4250.0	7.631765	13.439882	0.0	0.0000	0.00	16.0000	52.00
total_day_minutes	4250.0	180.259600	54.012373	0.0	143.3250	180.45	216.2000	351.50
total_day_calls	4250.0	99.907294	19.850817	0.0	87.0000	100.00	113.0000	165.00
total_day_charge	4250.0	30.644682	9.182096	0.0	24.3650	30.68	36.7500	59.76
total_eve_minutes	4250.0	200.173906	50.249518	0.0	165.9250	200.70	233.7750	359.30
total_eve_calls	4250.0	100.176471	19.908591	0.0	87.0000	100.00	114.0000	170.00
total_eve_charge	4250.0	17.015012	4.271212	0.0	14.1025	17.06	19.8675	30.54
total_night_minutes	4250.0	200.527882	50.353548	0.0	167.2250	200.45	234.7000	395.00
total_night_calls	4250.0	99.839529	20.093220	0.0	86.0000	100.00	113.0000	175.00
total_night_charge	4250.0	9.023892	2.265922	0.0	7.5225	9.02	10.5600	17.77
total_intl_minutes	4250.0	10.256071	2.760102	0.0	8.5000	10.30	12.0000	20.00
total_intl_calls	4250.0	4.426353	2.463069	0.0	3.0000	4.00	6.0000	20.00
total_intl_charge	4250.0	2.769654	0.745204	0.0	2.3000	2.78	3.2400	5.40
number_customer_service_calls	4250.0	1.559059	1.311434	0.0	1.0000	1.00	2.0000	9.00

Step by Step - EDA (info, head, describe)

describe() pada data test dilakukan untuk melihat apakah persebaran data tersebut serupa dengan data train.

Pada data test yang diberikan terlihat persebaran data serupa dengan data train terlihat dari standar deviasi pada fitur-fitur yang ada.

```
1 df_test.describe().T
```

	count	mean	std	min	25%	50%	75%	max
id	750.0	375.500000	216.650640	1.00	188.2500	375.500	562.7500	750.00
account_length	750.0	100.385333	39.699029	1.00	74.0000	101.000	126.0000	238.00
number_vmail_messages	750.0	8.454667	14.123712	0.00	0.0000	0.000	21.0000	51.00
total_day_minutes	750.0	180.454933	53.258337	12.50	146.6250	178.200	215.9750	350.80
total_day_calls	750.0	100.721333	19.718539	39.00	88.0000	101.000	114.0000	163.00
total_day_charge	750.0	30.677920	9.053756	2.13	24.9250	30.295	36.7150	59.64
total_eve_minutes	750.0	203.258267	52.185471	31.20	166.8000	203.350	235.9750	363.70
total_eve_calls	750.0	100.273333	19.367535	37.00	87.0000	101.000	113.0000	164.00
total_eve_charge	750.0	17.277080	4.435638	2.65	14.1775	17.285	20.0575	30.91
total_night_minutes	750.0	199.619467	51.531351	50.90	164.4750	199.450	234.8000	364.30
total_night_calls	750.0	100.370667	19.185238	12.00	88.0000	100.500	113.0000	168.00
total_night_charge	750.0	8.982827	2.318920	2.29	7.4025	8.975	10.5650	16.39
total_intl_minutes	750.0	10.294133	2.770340	0.00	8.5250	10.300	12.1000	18.90
total_intl_calls	750.0	4.485333	2.421901	0.00	3.0000	4.000	6.0000	19.00
total_intl_charge	750.0	2.779933	0.747704	0.00	2.3050	2.780	3.2700	5.10
number_customer_service_calls	750.0	1.634667	1.276207	0.00	1.0000	1.000	2.0000	6.00



EDA (Exploratory Data Analysis)

Cleaning



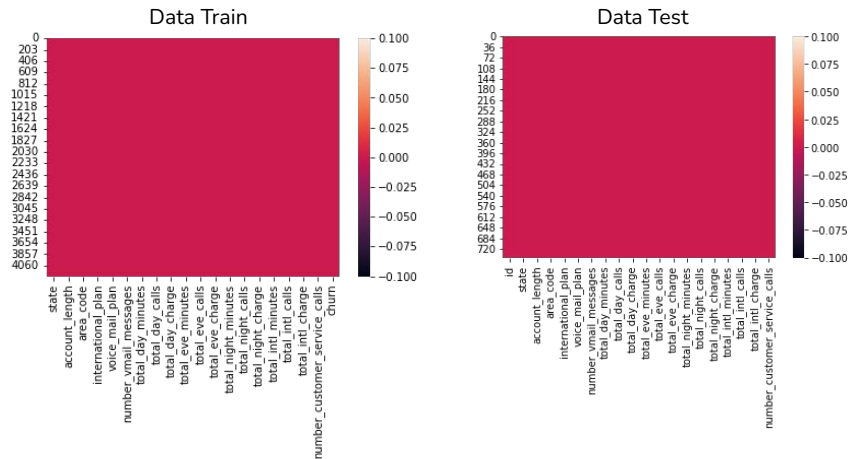
Step by Step - EDA (Cleaning)

Data Cleaning dilakukan untuk mengisi atau membuang nilai-nilai yang kosong, membuang data duplikat, dan memperbaiki format data.

Pada **head()** sebelumnya tidak terdapat format data yang salah atau ambigu pada data train dan test sehingga pada data hanya dilakukan pengecekan terhadap missing value dan duplikat data.

Perlakuan terhadap missing value didasarkan terhadap kategorisasi variabel dan target.

Setelah dilakukan pengecekan, ternyata **tidak terdapat data yang hilang pada data train dan test** sehingga langkah selanjutnya, yakni mengisi atau membuang nilai yang kosong tidak dilakukan.





Step by Step - EDA (Cleaning)

Selanjutnya, dilakukan pengecekan terhadap data duplikat dan hasilnya **tidak terdapat data duplikat** pada data train dan data test.

```
1 print(df_train.duplicated().sum())  
2 # df_train.drop_duplicates(inplace=True)  
3 # print(df_train.duplicated().sum())
```

```
0
```

```
[ ] 1 print(df_test.duplicated().sum())  
2 # df_test.drop_duplicates(inplace=True)  
3 # print(df_test.duplicated().sum())
```

```
0
```



EDA (Exploratory Data Analysis)

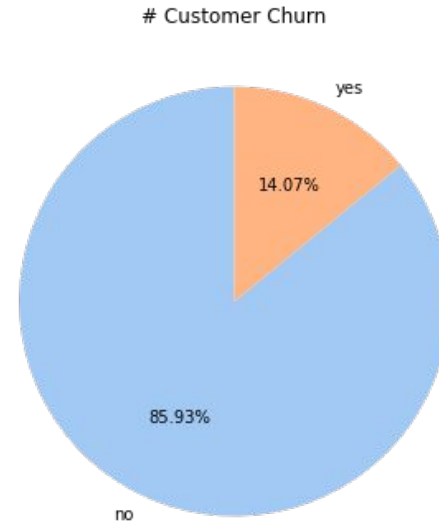
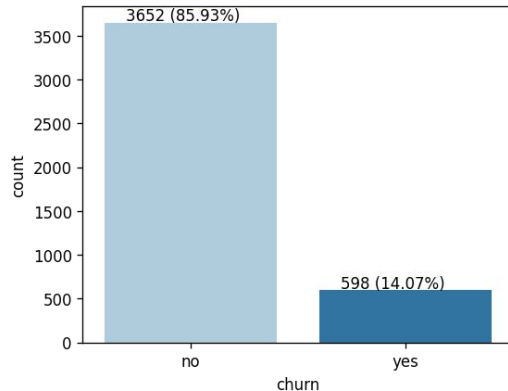
Data Exploring - Visualization



Step by Step - EDA (Visualization)

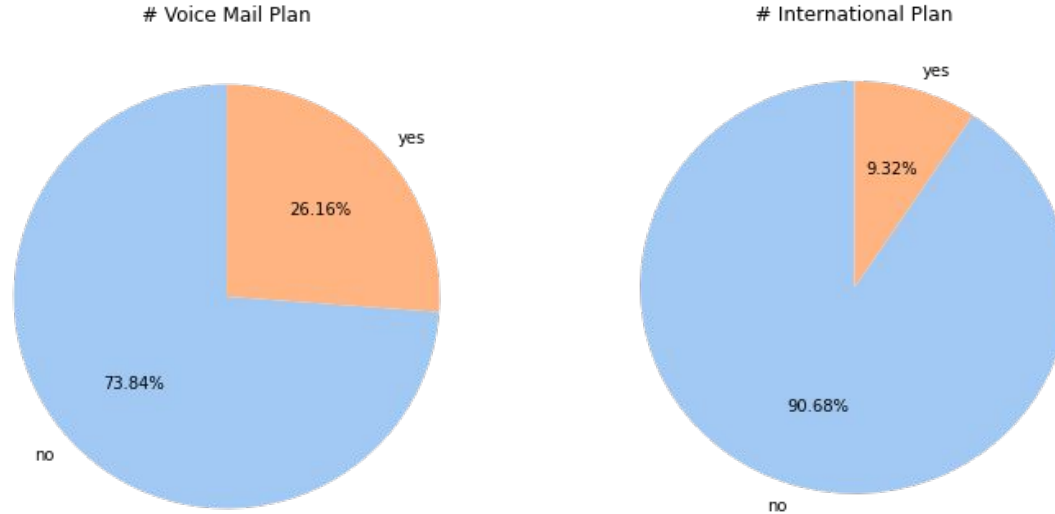
Visualisasi dibutuhkan untuk melihat data secara lebih jelas dan mudah serta untuk mengetahui pola dan asumsi yang ada.

Berikut merupakan hasil data exploring visualisasi pada banyaknya customer churn.





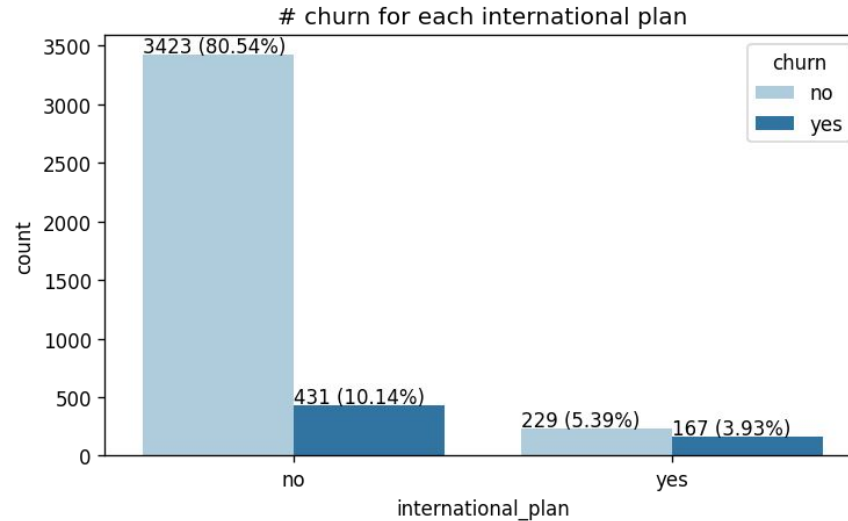
Step by Step - EDA (Visualization)



Berikut merupakan hasil data exploring visualisasi pada banyaknya customer yang memiliki international plan dan voice mail plan. Hasil yang dipaparkan ternyata customer yang memiliki international plan sangat sedikit sebesar 9.32% dan yang memiliki voice mail plan lebih dari $\frac{1}{4}$ total customer.



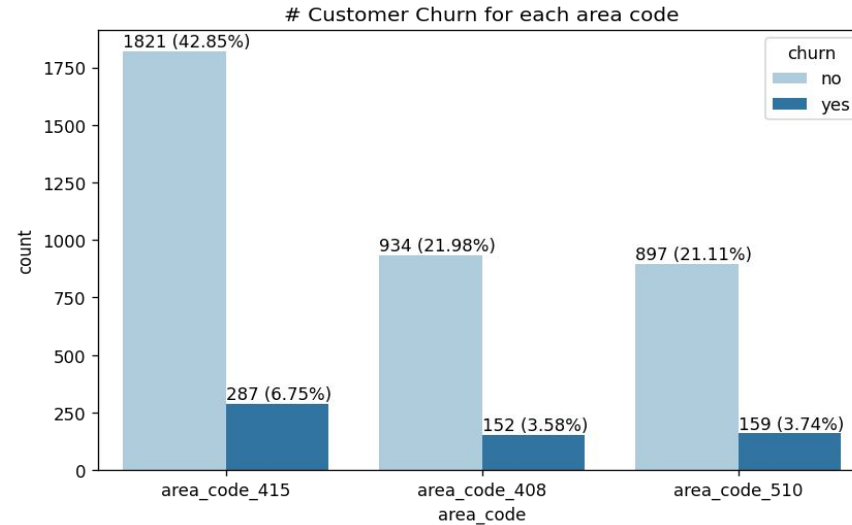
Step by Step - EDA (Visualization)



Berikut merupakan hasil data exploring visualisasi pada banyaknya customer churn pada tiap international plan. Hasil yang dipaparkan ternyata persentase churn pada customer yang tidak memiliki international plan sebesar 11.18% dan persentase churn pada customer yang memiliki international plan sebesar 42.17%. Hal ini dapat menandakan bahwa service pada international plan kurang baik.



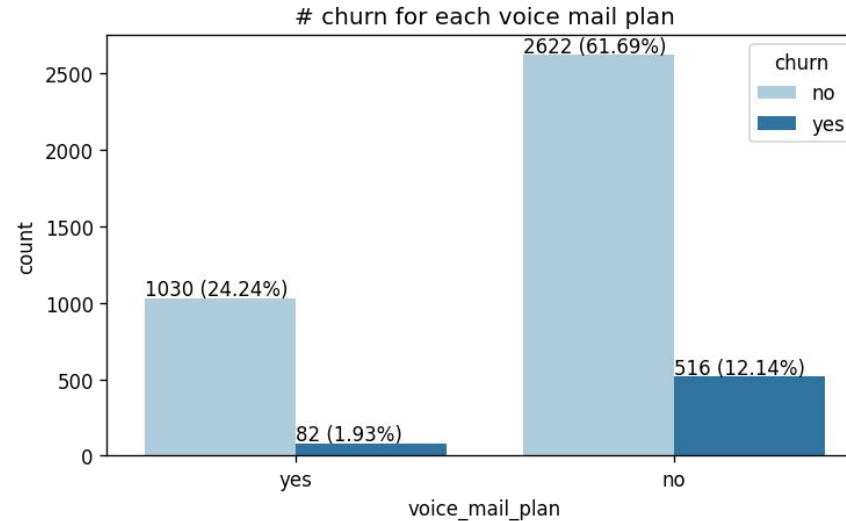
Step by Step - EDA (Visualization)



Berikut merupakan hasil data exploring visualisasi pada banyaknya customer churn pada tiap area code. Hasil yang dipaparkan ternyata persentase customer churn pada area code 415 sebesar 13.61%, persentase customer churn pada area code 408 sebesar 14.00%, dan persentase customer churn pada area code 510 sebesar 15.06%. Hal ini dapat menandakan bahwa service pada area code 510 kurang baik dibandingkan area lain sebanyak +- 2%



Step by Step - EDA (Visualization)



Berikut merupakan hasil data exploring visualisasi pada banyaknya customer churn pada tiap voice mail plan. Hasil yang dipaparkan ternyata persentase churn pada customer yang tidak memiliki voice mail plan sebesar 16.44%, persentase churn pada customer yang memiliki voice mail plan sebesar 7.37%. Hal ini dapat menandakan bahwa service pada voice mail plan dapat dikatakan cukup baik sehingga customer yang memilih voice mail plan lebih bertahan dibandingkan yang tidak memilih voice mail plan.



EDA (Exploratory Data Analysis)

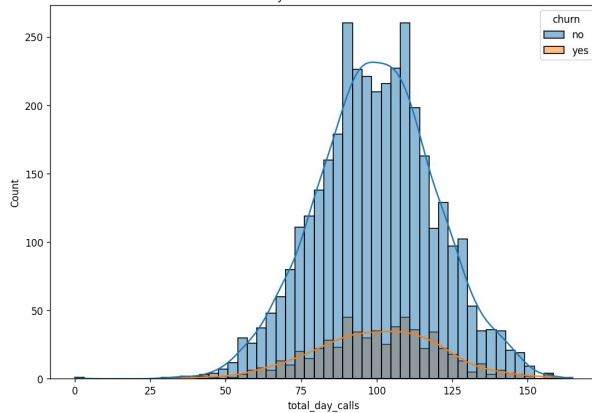
Data Exploring - Data Distribution



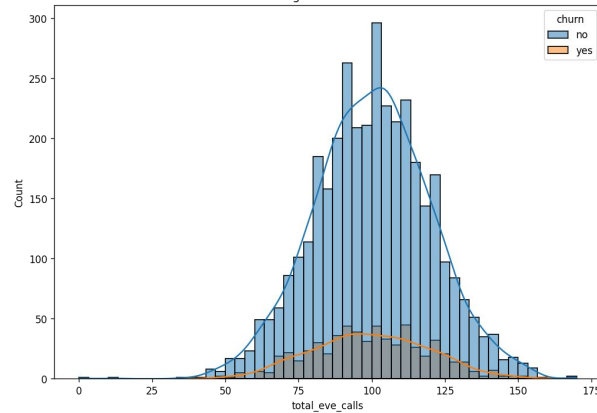


Step by Step - EDA (Data Distribution)

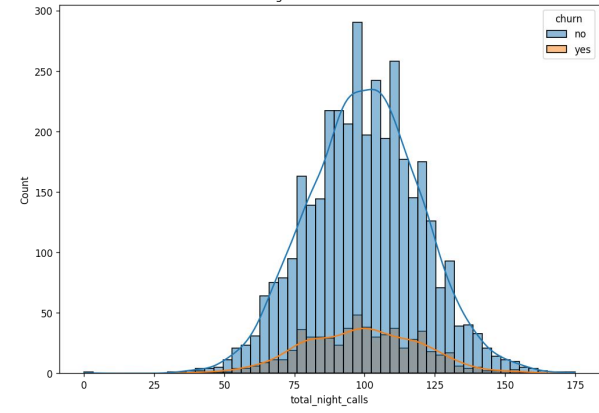
Day Calls distribution



Evening Calls distribution

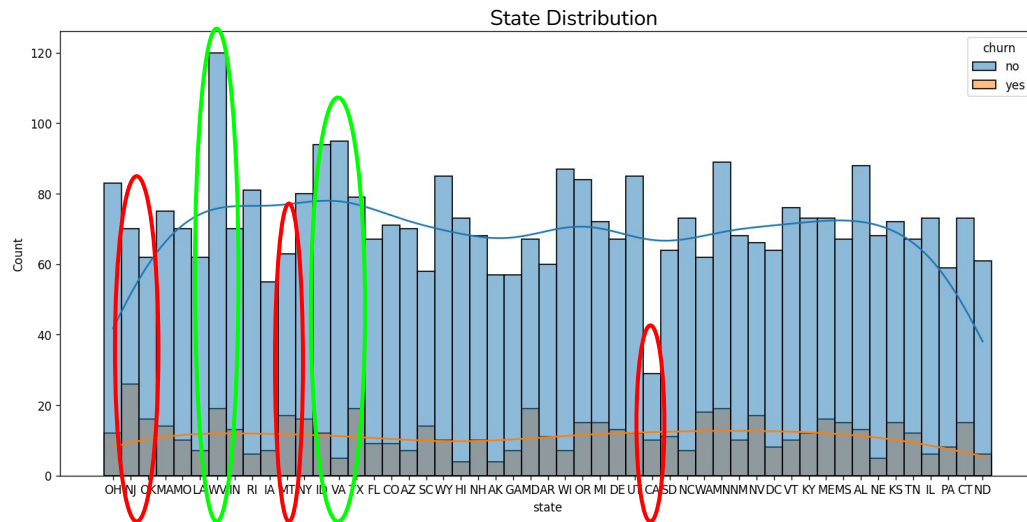


Night Calls distribution



Berikut merupakan hasil data exploring data distribution pada tiap waktu calls. Data yang ditampilkan menunjukkan data cukup terdistribusi secara normal. Selain itu, dapat dilihat jika churn banyak terjadi pada rentang 50 - 75 dan 125 - 150 dibandingkan total calls pada rentang tersebut.

Step by Step - EDA (Data Distribution)



Berikut merupakan hasil data exploring data distribution pada tiap state. Hal yang menarik bahwa terdapat beberapa state yang tingkat churn-nya cukup tinggi dan juga beberapa state yang tingkat churn-nya rendah.

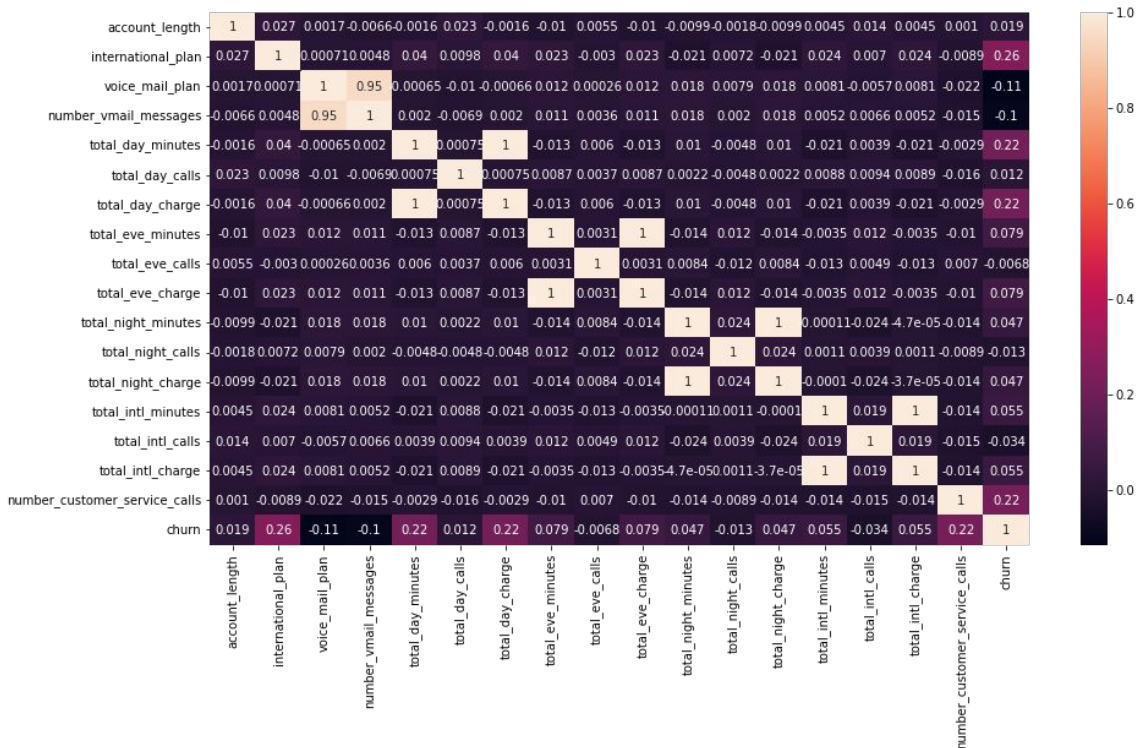


EDA (Exploratory Data Analysis)

Data Exploring - Heatmap



Step by Step - EDA (Heatmap)



Heatmap berfungsi untuk mengetahui relasi antar variabel. Sebelum dilakukan heatmap, terlebih dahulu variabel category diubah menjadi numerik. Pada pemodelan kali ini menggunakan **label encoding** pada **international plan**, **churn**, dan **voice mail plan**.

Berikut merupakan hasil data exploring heatmap. Hasil yang dipaparkan diketahui sesuai asumsi pada visualisasi sebelumnya bahwa fitur **international plan**, **total day minutes**, **total day charge**, dan **number customer service calls** memiliki **relasi positif** terhadap **churn**



Data Preprocessing

Encoding - Outliers - Drop Target Var - Normalization
- Train Test Split





Step by Step - Data Preprocessing

```
[ ] 1 df_train = pd.get_dummies(df_train)
```

Hal pertama yang dilakukan pada data preprocessing adalah mengubah fitur category menjadi fitur numerik atau encoding karena mesin hanya dapat membaca data dalam bentuk angka.

Pada kasus kali ini digunakan **one hot encoding**. Dimana semua kategori diubah menjadi data 0 dan 1 sesuai dengan fitur atau kolom pada kategori.

Pada data train yang berubah yakni **state** dan **area code**. Sedangkan, churn dan lainnya telah diubah menjadi label encoding.

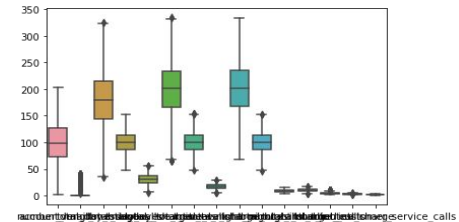
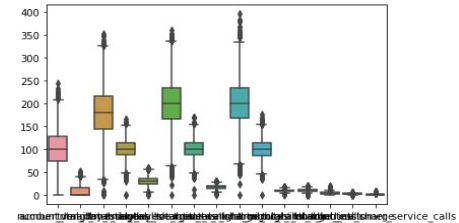
Step by Step - Data Preprocessing

Selanjutnya **mendeteksi outliers** berfungsi agar nilai-nilai ekstrem terbuang.

Pada kasus ini digunakan metode Interquartile atau **IQR** dengan Q1 sama dengan 25% dengan dan Q3 sama dengan 75%. Fitur yang dideteksi adalah semua fitur numerik.

```
1 import numpy as np
2 print(f'Jumlah baris sebelum memfilter outlier: {len(df_train)}')
3
4 filtered_entries = np.array([True] * len(df_train))
5 for col in num_cols:
6     Q1 = df_train[col].quantile(0.25)
7     Q3 = df_train[col].quantile(0.75)
8     IQR = Q3 - Q1
9     low_limit = Q1 - (IQR * 1.5)
10    high_limit = Q3 + (IQR * 1.5)
11
12    filtered_entries = ((df_train[col] >= low_limit) & (df_train[col] <= high_limit)) & filtered_entries
13
14 df_train = df_train[filtered_entries]
15
16 print(f'Jumlah baris setelah memfilter outlier: {len(df_train)}')
```

Jumlah baris sebelum memfilter outlier: 4250
Jumlah baris setelah memfilter outlier: 3515



Step by Step - Data Preprocessing

Setelah itu perlu dilakukan drop target variabel sehingga seakan-akan mesin belum mengetahui nilai target.

```
1 X = df_train.drop(['churn'], 1)
2 y = df_train['churn']
```

Selanjutnya dilakukan Normalisasi agar data lebih terdistribusi normal menggunakan min max scaler

```
1 from sklearn.preprocessing import MinMaxScaler
2
3 scaler = MinMaxScaler()
4 X_transform = scaler.fit_transform(X)

[ ] 1 X_transform = pd.DataFrame(X_transform, columns = X.columns)
```

Terakhir pada tahap ini adalah memisahkan train dan test berdasarkan target variabel sebelumnya. Pada kasus ini proporsi test sebesar 30%

```
1 from sklearn.model_selection import train_test_split
2
3 X_train,X_test,y_train,y_test = train_test_split(X_transform, y, test_size = 0.3, random_state = 42)
```



Modelling





Step by Step - Modelling

Pada kasus ini digunakan model ML classification dengan minimal 2 algoritma. Algoritma yang digunakan adalah:

1. Random Forest Classifier
2. Logistic Regression

Lalu, dibuat pula list untuk evaluation

```
1 from sklearn.linear_model import LogisticRegression
[REDACTED]
5 from sklearn.ensemble import RandomForestClassifier
[REDACTED]
9
10
11 from sklearn.metrics import confusion_matrix, classification_report
12 from sklearn.metrics import roc_auc_score, accuracy_score, f1_score, precision_score
```

```
1 model_list = []
2 accuracy_list = []
3 precision_list = []
4 f1_score_list = []
5 roc_auc_list = []
```

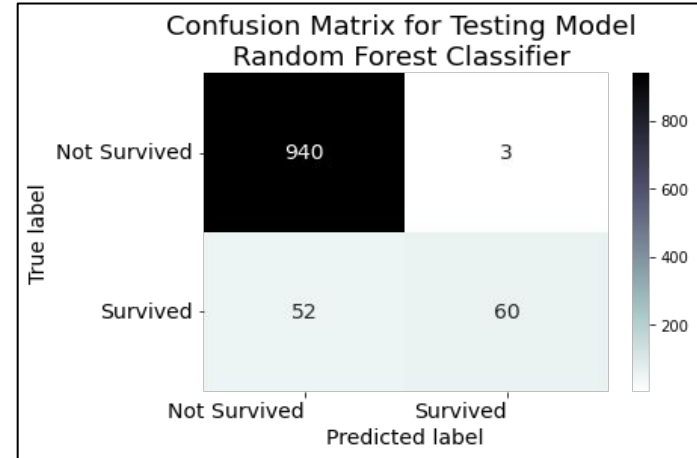
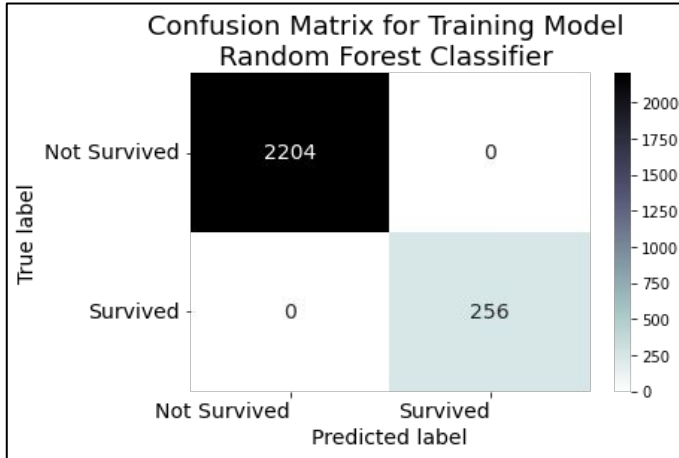


Step by Step - Modelling

```
1 def ML (alg, name_model): #y_train_alg, y_test_alg):
2
3     alg_model = alg().fit(X_train, y_train)
4     print(alg_model)
5     y_train_alg = alg_model.predict(X_train)
6     y_test_alg = alg_model.predict(X_test)
7
8     pred(X_train, y_train, alg_model, 'Training', name_model, y_train_alg)
9     pred(X_test, y_test, alg_model, 'Testing', name_model, y_test_alg)
10
11     accuracy, precision, f1_s, roc_auc_sc = round(accuracy_score(y_test , y_test_alg) , 3),
12     round(precision_score(y_test , y_test_alg) , 3),
13     round(f1_score(y_test , y_test_alg) , 3),
14     round(roc_auc_score(y_test, y_test_alg),3)
15
16     print(f'Test Accuracy Score is :{accuracy}')
17     print(f'Test Precision Score is :{precision}')
18     print(f'f1 Score is :{f1_s}')
19     print(f'ROC AUC Score is :{roc_auc_sc}')
20     model_list.append(name_model)
21     accuracy_list.append(accuracy)
22     precision_list.append(precision)
23     f1_score_list.append(f1_s)
24     roc_auc_list.append(roc_auc_sc)
25     return(alg_model)
```

```
1 def pred (x_data, y_data, alg_model, type, name_model, y_pred_alg):
2
3     # print classification report
4     print('\nClassification Report %s' % type, ' Model:')
5     print(classification_report(y_data, y_pred_alg))
6
7     # form confusion matrix as a dataframe
8     confusion_matrix_alg = pd.DataFrame((confusion_matrix(y_data, y_pred_alg)),
9                                         ('Not churn', 'churn'),('Not churn', 'churn'))
10
11     # plot confusion matrix
12     plt.figure()
13     heatmap = sns.heatmap(confusion_matrix_alg, annot=True,
14                           annot_kws={'size': 14}, fmt='d', cmap='bone_r')
15     heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(),
16                                  rotation=0, ha='right', fontsize=14)
17     heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(),
18                                  rotation=0, ha='right', fontsize=14)
19
20     plt.title('Confusion Matrix for %s Model \n%s' % (type,name_model),
21              fontsize=18, color='black')
22     plt.ylabel('True label', fontsize=14)
23     plt.xlabel('Predicted label', fontsize=14)
24     plt.show()
```

Step by Step - Modelling

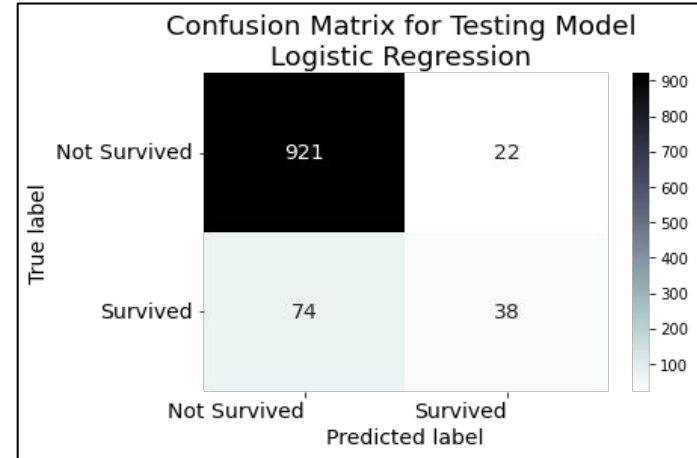
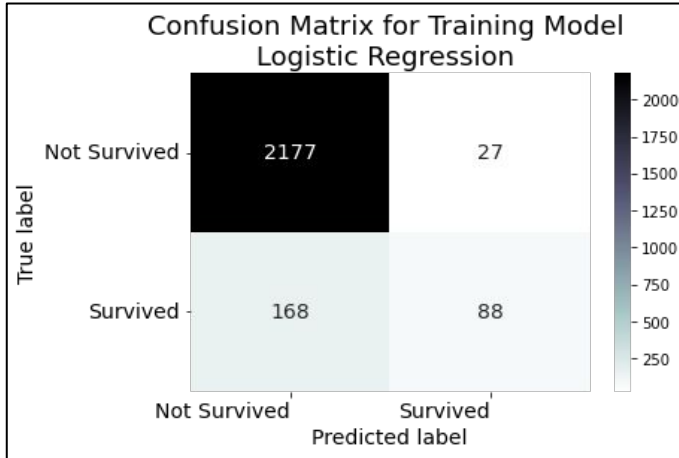


Berikut merupakan hasil training dan testing pada model algoritma Random Forest Classifier. Pada training diketahui akurasi mencapai 100% dan pada testing +/- 95%



```
1 model_rf = ML(RandomForestClassifier, 'Random Forest Classifier')
```

Step by Step - Modelling



Berikut merupakan hasil training dan testing pada model algoritma Logistic Regression. Pada training diketahui akurasi mencapai lebih dari 90% dan testing juga mencapai lebih dari 90%



```
1 model_lr = ML(LogisticRegression, 'Logistic Regression')
```



Evaluation & Tuning





Step by Step - Evaluation & Tuning

	Model	Accuracy	Precision	f1_score	ROC_AUC_score
0	Random Forest Classifier	0.953	0.970	0.719	0.785
1	Logistic Regression	0.909	0.633	0.442	0.658

Hasil Evaluasi menunjukkan bahwa model algoritma Random Forest Classifier memiliki tingkat akurasi, presisi, f1 score, dan roc_auc_score lebih tinggi dibandingkan model algoritma Logistic Regression. Selanjutnya akan dilakukan tuning pada kedua algoritma



Step by Step - Evaluation & Tuning

```
[234] 1 params = {'max_depth':[3,5,7,9,11,'max']} # hati-hati pemilihan hyperparameter jangan terlalu banyak kombinasinya  
      2 params_lr = {'C':[0.01,0.05,0.1,0.5,0.7,1,2,3]}
```

```
▶ 1 lr = LogisticRegression()  
  2 rf = RandomForestClassifier()  
  3 grid = GridSearchCV(  
  4     estimator=lr,  
  5     param_grid=params_lr,  
  6     scoring = 'accuracy',  
  7     n_jobs = 10, # core cpu yang digunakan  
  8     cv = 10 # 10-fold cross validation (artinya kita melakukan iterasi model sebanyak 3 kali)  
  9 )
```



Step by Step - Evaluation & Tuning

Hasil tuning menunjukkan bahwa akurasi dari model algoritma Logistic Regression tidak berubah dari awalnya 0.909 atau 90,9% dengan best parameter 'C': 2

```
[ ] 1 grid.best_params_
```

```
{'C': 2}
```



```
1 # membuat fungsi evaluasi model
2 def evaluasi_model(model, X_test, y_test):
3     y_pred = model.predict(X_test)
4     return accuracy_score(y_test, y_pred)
```

```
[ ] 1 evaluasi_model(grid,X_test,y_test)
```

```
0.9090047393364928
```



Step by Step - Evaluation & Tuning

Hasil tuning menunjukkan bahwa akurasi dari model algoritma Random Forest Classifier sedikit berubah dari awalnya 95.3% menjadi 93.6% dengan best parameter 'max_depth': 11

```
[255] 1 grid_rf.best_params_
```

```
{'max_depth': 11}
```



```
1 evaluasi_model(grid_rf,X_test,y_test)
```



```
0.9364928909952607
```



Prediction Test





Step by Step - Prediction Test

```
[242] 1 df_test['international_plan'] = intl_le.fit_transform(df_test['international_plan'])  
      2 df_test['voice_mail_plan'] = vmail_le.fit_transform(df_test['voice_mail_plan'])
```

```
▶ 1 df_test = pd.get_dummies(df_test)
```

Pada Prediction Test, data test terlebih dahulu dilakukan encoding seperti pada data train. Selanjutnya, menyimpan fitur 'id' untuk menggabungkan diakhir nanti dan drop fitur 'id' untuk dilakukan prediksi.

```
▶ 1 cus_id = df_test['id']  
  2 df_test = df_test.drop(columns='id')
```



Step by Step - Prediction Test

```
[247] 1 df_test_columns = df_test

[248] 1 df_test = scaler.fit_transform(df_test)

[250] 1 df_test = pd.DataFrame(df_test, columns = df_test_columns.columns)
```

Selanjutnya normalisasi pada data test perlu dilakukan seperti pada data train. Setelah itu, dilakukan prediksi berdasarkan model algoritma yang dipilih



Step by Step - Prediction Test

```
[258] 1 prediction_lr = model_lr.predict(df_test)

[259] 1 prediction_rf = model_rf.predict(df_test)

[260] 1 submission_lr = pd.DataFrame({'CustomerId':cus_id,'churn':prediction_lr})

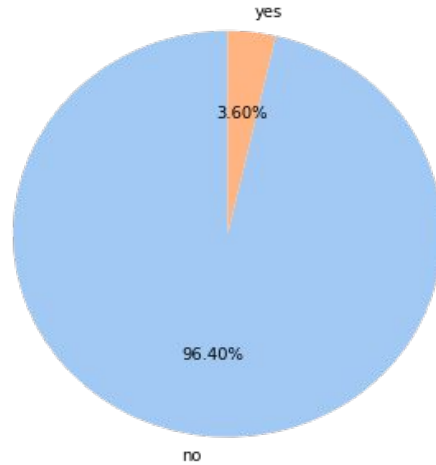
[261] 1 submission_rf = pd.DataFrame({'CustomerId':cus_id,'churn':prediction_rf})
```

Pada kasus ini dipilih model algoritma Logistic Regression dan Random Forest Classifier **sebelum dituning**. Hal ini melihat hasil tuning pada Logistic Regression memiliki tingkat akurasi yang sama dengan sebelum dituning dan tingkat akurasi Random Forest Classifier yang lebih kecil dibandingkan sebelum dituning.

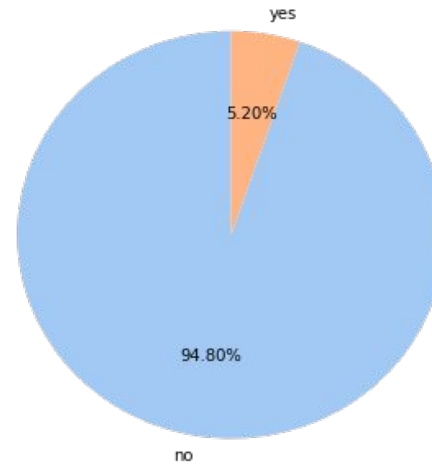


Step by Step - Prediction Test

Customer Churn Predicted LogReg



Customer Churn Predicted RanFor



Hasil prediksi menunjukkan algoritma Logistic Regression memprediksi tingkat Customer Churn sebesar 3.6% dari data test dan algoritma Random Forest Classifier memprediksi tingkat Customer Churn sebesar 5.2%



Ada Pertanyaan?





Kesimpulan

1. Visualisasi data Covid-19 Indonesia difokuskan pada **jumlah**, **peningkatan**, dan **persebaran** virus di Indonesia.
2. **Tipe data** perlu disesuaikan terhadap visualisasi data yang akan ditampilkan.
3. **Exploratory Data Analysis** dilakukan untuk **investigasi awal** pada data dengan tujuan menemukan pola, anomali, menguji hipotesis dan dapat memeriksa asumsi dengan bantuan statistik ringkasan kemudian representasi grafis (visualisasi).
4. Studi Kasus pada pemodelan Machine Learning ini menggunakan 2 Encoding, yakni **Label Encoding** pada fitur churn, international_plan, dan voice_mail_plan serta **One Hot Encoding** pada fitur state dan area_code.
5. **Sampel uji** pada pemodelan Machine Learning sebesar **30%** dan model Algoritma Klasifikasi yang digunakan adalah **Random Forest Classifier** dan **Logistic Regression**.
6. Random Forest Classifier memiliki akurasi **95.3%**, sedangkan Logistic Regression memiliki akurasi **90.9%**.
7. Berdasarkan hasil prediction test, **Logistic Regression** mendeteksi Customer Churn sebesar **3.6%** dari data test. Sedangkan, **Random Forest Classifier** mendeteksi 1.6% lebih tinggi, yakni **5.2%** Customer Churn.

Terima Kasih

Jangan Semangat dan Tetap Putus Asa