Prediksi Churn Customer
Logistic Regression dan XGBoost

Memahami faktor-faktor apa yang memengaruhi keputusan pelanggan untuk berhenti berlangganan (churn) atau tetap berlangganan,

Yang memungkinkan perusahaan untuk mengidentifikasi pola perilaku yang berkaitan dengan churn dan mengambil tindakan yang tepat untuk mempertahankan pelanggan yang ada dan mencegah churn di masa depan.

Data Understanding

BINAR

Mengecek Missing Value dan Duplicat data, Serta memahami isi dari setiap kolom nya

Hasilnya masih aman tidak ada missing dan duplicat data serta isi kolom-kolom tidak ada yang aneh masih normal

```
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
```

```
df.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
```

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

```
0
```

```
[5] df.describe()
```

	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_c
count	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.000000	4250.0
mean	100.236235	7.631765	180.259600	99.907294	30.644682	200.173906	100.176471	17.015012	200.527882	99.839529	9.0
std	39.698401	13.439882	54.012373	19.850817	9.182096	50.249518	19.908591	4.271212	50.353548	20.093220	2.2
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	73.000000	0.000000	143.325000	87.000000	24.365000	165.925000	87.000000	14.102500	167.225000	86.000000	7.5
50%	100.000000	0.000000	180.450000	100.000000	30.680000	200.700000	100.000000	17.060000	200.450000	100.000000	9.0
75%	127.000000	16.000000	216.200000	113.000000	36.750000	233.775000	114.000000	19.867500	234.700000	113.000000	10.5
max	243.000000	52.000000	351.500000	165.000000	59.760000	359.300000	170.000000	30.540000	395.000000	175.000000	17.7

Data Understanding

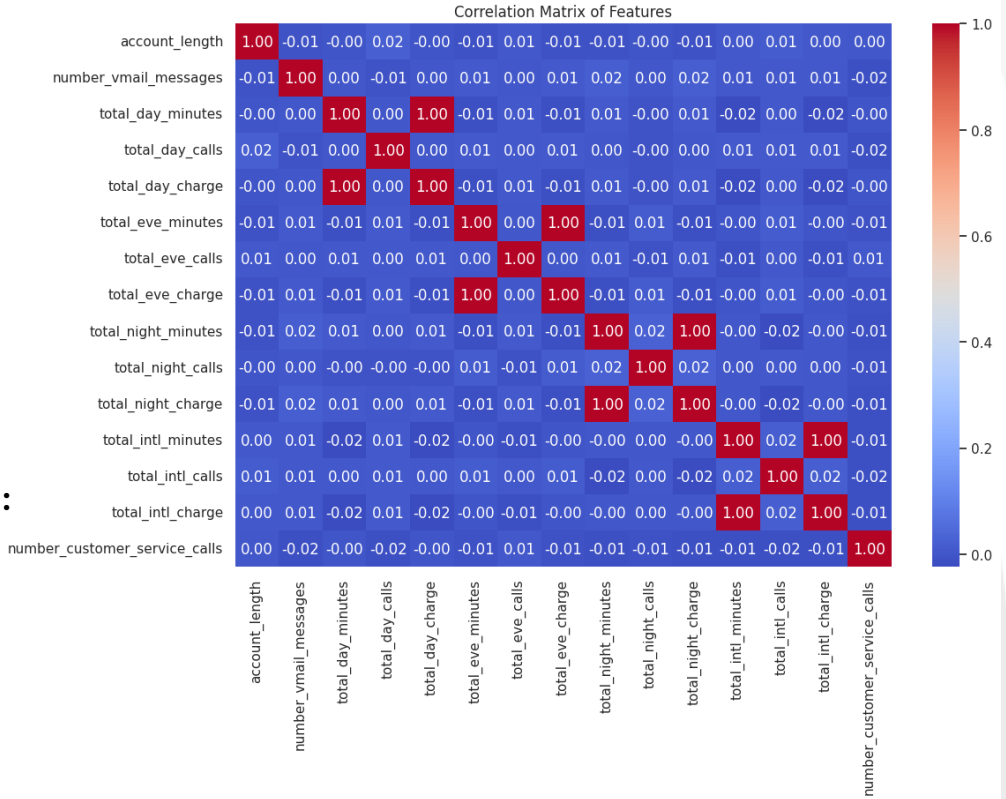
Fitur-fitur yang memiliki korelasi yang sangat tinggi dengan fitur lainnya :

- 'total_day_minutes' dan 'total_day_charge'
- 'total_eve_minutes' dan 'total_eve_charge'
- 'total_night_minutes' dan 'total_night_charge'
- 'total_intl_minutes' dan 'total_intl_charge'

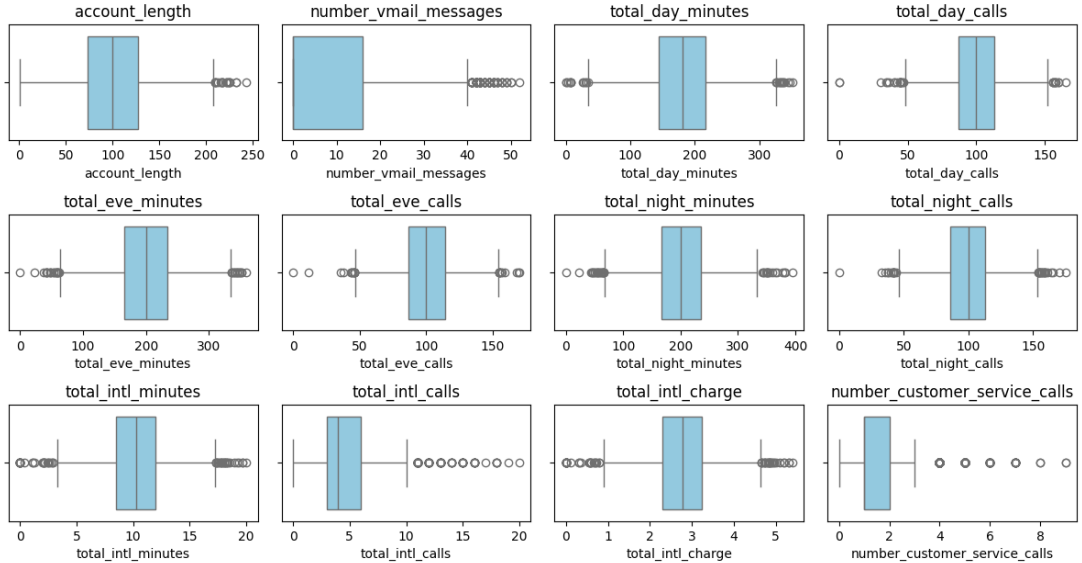
Fitur biaya panggilan dihasilkan dari informasi jumlah menit yang digunakan dalam setiap periode waktu, sehingga fitur biaya panggilan dianggap redundan.

Oleh karena itu, kami memilih untuk menghapus fitur :
- ‘total_day_charge'
- 'total_eve_charge’
- 'total_night_charge’
- ‘total_intl_charge’

agar:
- Mengurangi dimensi dataset,
- Menghilangkan informasi yang tidak diperlukan, dan
- Mempertahankan informasi asli yang diwakili oleh jumlah menit yang digunakan.



Data Understanding




```python
for column in df.select_dtypes(include=['int64', 'float64']).columns:
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    print(f"Outliers di kolom '{column}': {outliers.shape[0]}")
```

```
Outliers di kolom 'account_length': 20
Outliers di kolom 'number_vmail_messages': 86
Outliers di kolom 'total_day_minutes': 25
Outliers di kolom 'total_day_calls': 28
Outliers di kolom 'total_eve_minutes': 34
Outliers di kolom 'total_eve_calls': 24
Outliers di kolom 'total_night_minutes': 37
Outliers di kolom 'total_night_calls': 33
Outliers di kolom 'total_intl_minutes': 62
Outliers di kolom 'total_intl_calls': 100
Outliers di kolom 'number_customer_service_calls': 335
```

Disini saya mempertahankan outlier karena masih dianggap normal dan bisa saja benar terjadi dan dapat membantu mencerminkan variasi alami dalam dataset.

BINAR

Check Categorical

Fitur yang sangat didominasi oleh salah satu nilai saja akan dibuang pada tahap ini

yaitu kolom 'international_plan'

```
for col in df.select_dtypes(include='object').columns.tolist():
    print(df[col].value_counts(normalize=True)*100)
    print('\n')
```

```
area_code_415    49.600000
area_code_408    25.552941
area_code_510    24.847059
Name: area_code, dtype: float64


no     90.682353
yes     9.317647
Name: international_plan, dtype: float64


no     73.835294
yes    26.164706
Name: voice_mail_plan, dtype: float64


no     85.929412
yes    14.070588
Name: churn, dtype: float64
```

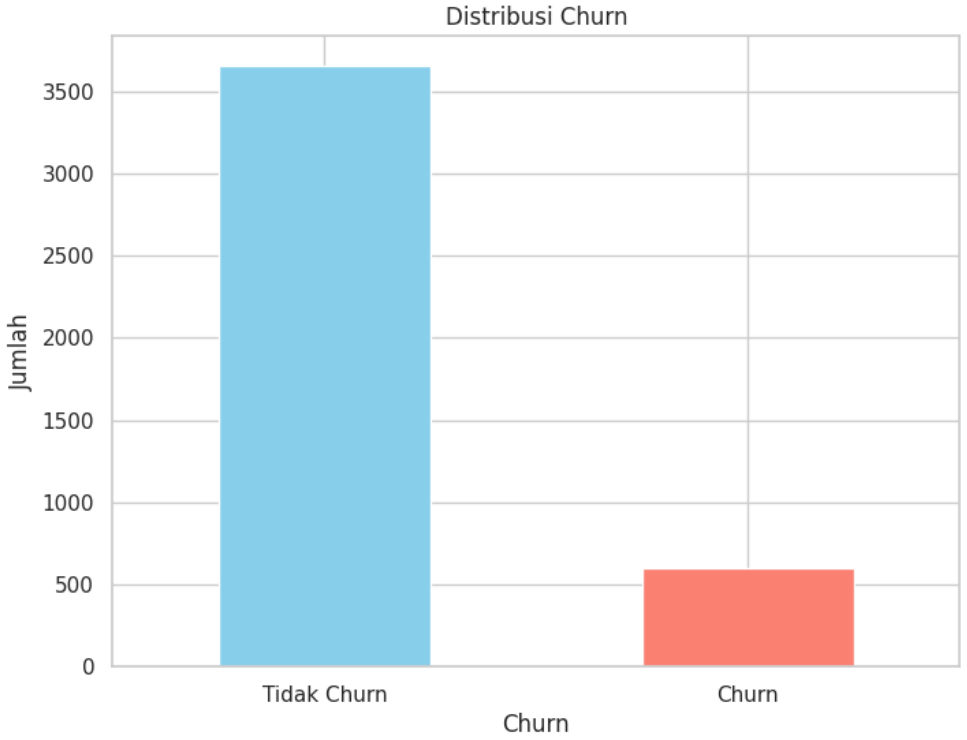
Fitur yang sangat didominasi oleh salah satu nilai saja akan dibuang pada tahap ini yaitu kolom '**international_plan**'

```
[ ] df.drop('international_plan', axis=1, inplace=True)
    df.head()
```

Dari analisis distribusi target, ditemukan bahwa terdapat **ketidakseimbangan** yang signifikan antara kategori 'Tidak Churn' dan 'Churn'.

Jumlah entri:

- Tidak churn = 3652
- Churn = 598



Data Preparation

- Kolom target 'churn' diubah menjadi bentuk numerik terlebih dahulu menggunakan LabelEncoder
- Pembagian data pelatihan dan pengujian dengan proporsi pengujian sebesar 80:20.
- Stratifikasi dilakukan terhadap label target untuk memastikan distribusi kelas yang seimbang di kedua set data.
- Penggunaan StandardScaler untuk menormalkan data numerik
- Encoder one-hot untuk mengonversi data kategorikal menjadi bentuk yang dapat dimengerti oleh model.

```
label_encoder = LabelEncoder()
df['churn'] = label_encoder.fit_transform(df['churn'])
```

```
[ ] X = df.drop(columns=['churn'])
    y = df['churn']

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
    X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((3400, 14), (850, 14), (3400,), (850,))
```

```
[ ] numerical_cols = ['account_length', 'number_vmail_messages', 'total_day_minutes', 'total_day_calls',
                      'total_eve_minutes', 'total_eve_calls', 'total_night_minutes', 'total_night_calls',
                      'total_intl_minutes', 'total_intl_calls', 'number_customer_service_calls']

    categorical_cols = ['state', 'area_code', 'voice_mail_plan']

    # Preprocessing untuk data numerik
    numerical_transformer = Pipeline(steps=[
        ('scaler', StandardScaler())
    ])

    # Preprocessing untuk data kategorikal
    categorical_transformer = Pipeline(steps=[
        ('onehot', OneHotEncoder(handle_unknown='ignore'))
    ])

    # Gabungkan preprocessor
    preprocessor = ColumnTransformer(
        transformers=[
            ('num', numerical_transformer, numerical_cols),
            ('cat', categorical_transformer, categorical_cols)
        ])
```

Modeling Logistic Regression

Parameter yang di tuning:

- C: Parameter regularisasi yang mengontrol kekuatan regularisasi dalam model.
- fit_intercept: Parameter yang menentukan apakah model akan mempelajari intercept atau tidak.

Dua parameter ini digunakan bersamaan GridSearchCV untuk menemukan kombinasi parameter terbaik yang menghasilkan kinerja model optimal.

```
# Pipeline untuk model regresi logistik
logistic_model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(solver='liblinear', max_iter=1000, penalty='l2', class_weight='balanced'))
])

# Menambahkan parameter yang akan diuji
param_space_lr = {
    "classifier__C": np.logspace(-3, 3, 7),
    "classifier__fit_intercept": [True, False],
}

# Melatih model dengan teknik RandomizedSearchCV
model_lr = GridSearchCV(logistic_model, param_grid=param_space_lr, cv=3)

# Fitting model ke data pelatihan
model_lr.fit(X_train, y_train)

# Tampilkan hasil regresi logistik
print("Best Parameters (Logistic Regression):", model_lr.best_params_)
print("Training Accuracy (Logistic Regression):", model_lr.score(X_train, y_train))
print("Model Best Score (Logistic Regression):", model_lr.best_score_)
print("Test Accuracy (Logistic Regression):", model_lr.score(X_test, y_test))
```

Modeling XGBoost

Pada model XGBoost, dilakukan penelusuran parameter menggunakan GridSearchCV dengan kombinasi parameter berikut:

- n_estimators: Jumlah pohon keputusan yang akan dibuat dalam model.
- max_depth: Kedalaman maksimum dari setiap pohon keputusan dalam model.
- gamma: Minimum penurunan kehilangan yang diperlukan untuk membagi node selama pembentukan pohon

```
# Pipeline untuk model XGBoost
xgb_model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', XGBClassifier())
])

# Menambahkan parameter yang akan diuji
param_space_xgb = {
    'classifier__n_estimators': [100, 500, 1000],
    'classifier__max_depth': [3, 5, 7],
    'classifier__gamma': [0, 0.1, 0.2]
}

# Melatih model dengan teknik GridSearchCV
model_xgb = GridSearchCV(xgb_model, param_grid=param_space_xgb, cv=3)

# Fitting model ke data pelatihan
model_xgb.fit(X_train, y_train)

# Tampilkan hasil XGBoost
print("Best Parameters (XGBoost):", model_xgb.best_params_)
print("Training Accuracy (XGBoost):", model_xgb.score(X_train, y_train))
print("Model Best Score (XGBoost):", model_xgb.best_score_)
print("Test Accuracy (XGBoost):", model_xgb.score(X_test, y_test))
```

Evaluation Logistic Regression

```
Best Parameters (Logistic Regression): {'classifier__C': 0.1, 'classifier__fit_intercept': True}
Training Accuracy (Logistic Regression): 0.7302941176470589
Model Best Score (Logistic Regression): 0.7238208872513079
Test Accuracy (Logistic Regression): 0.7070588235294117
```

- Parameter Terbaik: C = 0.1, dengan fit_intercept = True.
- Akurasi Pelatihan: 73.03%
- Skor Terbaik: 72.38%
- Akurasi Pengujian: 70.71%

Confusion Matrix:
  - TN: 513, FP: 217, FN: 32, TP: 88.

Classification Report:
  - Precision: Tidak churn (0.94), Churn (0.29)
  - Recall: Tidak churn (0.70), Churn (0.73)
  - F1-score: Tidak churn (0.80), Churn (0.41)

```
Confusion Matrix Logistic Regression:
[[513 217]
 [ 32  88]]

Classification Report Logistic Regression:
              precision    recall  f1-score   support

           0       0.94      0.70      0.80       730
           1       0.29      0.73      0.41       120

    accuracy                           0.71       850
   macro avg       0.61      0.72      0.61       850
weighted avg       0.85      0.71      0.75       850
```

Evaluation XGBoost

```
Best Parameters (XGBoost): {'classifier__gamma': 0.1, 'classifier__max_depth': 7, 'classifier__n_estimators': 100}
Training Accuracy (XGBoost): 1.0
Model Best Score (XGBoost): 0.9261801764498636
Test Accuracy (XGBoost): 0.9305882352941176
```

- Parameter Terbaik: gamma = 0.1, max_depth = 7, n_estimators = 100.
- Akurasi Pelatihan: 100%
- Skor Terbaik: 92.62%
- Akurasi Pengujian: 93.06%

Confusion Matrix:
- TN: 723, FP: 7, FN: 52, TP: 68.

Classification Report:
- Precision: Tidak churn (0.93), Churn (0.91)
- Recall: Tidak churn (0.99), Churn (0.57)
- F1-score: Tidak churn (0.96), Churn (0.70)

```
Confusion Matrix XGBoost:
[[723   7]
 [ 52  68]]
```

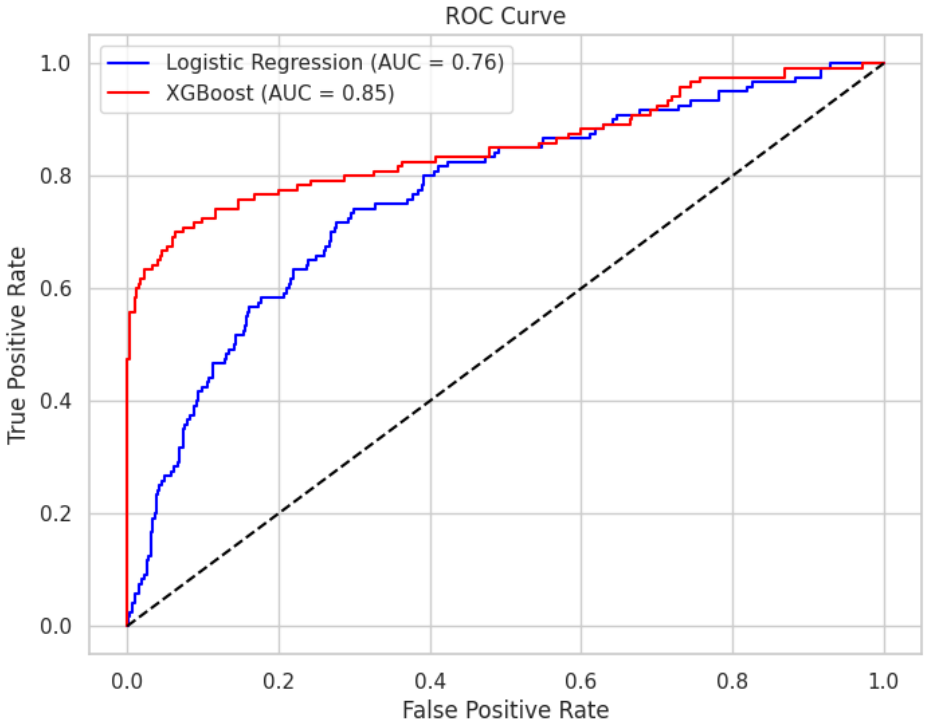
Classification Report XGBoost:

	precision	recall	f1-score	support
0	0.93	0.99	0.96	730
1	0.91	0.57	0.70	120
accuracy			0.93	850
macro avg	0.92	0.78	0.83	850
weighted avg	0.93	0.93	0.92	850

Evaluation ROC

Dari hasil evaluasi, terlihat bahwa model XGBoost memiliki nilai Area Under the Curve (AUC) yang lebih tinggi, yaitu sebesar 0.85, dibandingkan dengan model logistic regression yang memiliki nilai AUC sebesar 0.76.

Dengan demikian, berdasarkan metrik AUC, model XGBoost dapat dianggap lebih baik daripada model logistic regression.



Deployment

```
# Memprediksi churn menggunakan model regresi logistik
y_pred_lr = model_lr.predict(df_test)

# Memprediksi churn menggunakan model XGBoost
y_pred_xgb = model_xgb.predict(df_test)

# Menambahkan hasil prediksi sebagai kolom baru pada DataFrame df_test
df_test['Churn Prediction (Logistic Regression)'] = y_pred_lr
df_test['Churn Prediction (XGBoost)'] = y_pred_xgb

# Menampilkan lima baris pertama DataFrame df_test yang telah diperbarui
df_test.head()
```

:al_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	Churn Prediction (Logistic Regression)	Churn Prediction (XGBoost)
197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	0	0
220.6	101	18.75	203.9	118	9.18	6.3	6	1.70	0	0	0
307.2	76	26.11	203.0	99	9.14	13.1	6	3.54	4	1	1
218.2	111	18.55	129.6	121	5.83	8.1	3	2.19	3	1	0
277.1	112	23.55	250.7	115	11.28	15.5	5	4.19	3	1	0

```
# Save hasil prediksi
df_test.to_csv('/content/drive/MyDrive/FGA - Binar Academy/Dataset/predictions.csv', index=False)
```

Mempredict dataset Test dari kedua model tersebut kemudian data predict tersebut di simpan di dalam drive dan berformat csv

BINAR

Thank You