Challenge 2 FGA X Binar Academy

Customer Churn Classification

Kelompok P :

- Ilham Ramadhan Mu'taz
- Imam Ahmad Qusyairi

Outline

01 **Pendahuluan**
Pengantar

02 ***EDA***
Analisis data secara mendalam

03 ***Data Modelling***
Membangun model prediksi

04 **Kesimpulan**
Intisari projek

01 Pendahuluan

- Problem Statement
- Pengenalan Data
- Model yang Akan Digunakan

Problem Statement

- Perusahaan X menghadapi penurunan revenue di Q4 yang disebabkan oleh tingginya angka customer churn
- Kita ditugaskan untuk menganalisis data perusahaan X dan membangun model machine learning untuk memprediksi customer churn

Target / Objectives

- Berhasil menyiapkan & membersihkan data
- Berhasil melakukan EDA
- Berhasil membangun model dengan performa yang **memuaskan*

* Performa model berdasarkan ragam metode evaluasi mempunyai nilai yang tinggi (> 85 %)

Model ML yang Digunakan

- Random Forest Classifier
- Support Vector Classifier (SVC)
- Light Gradient Boosting Machine (LGBM)
- Bagging Classifier
- Ada Boost Classifier
- XGBoost

02 Exploratory Data Analysis

- Dataset Overview
- Customer Churn
- International Plan
- Voice Mail Plan
- Area Code
- Number Customer Call
- Total Day Minutes, Calls, dan Charge
- Korelasi

Dataset Overview

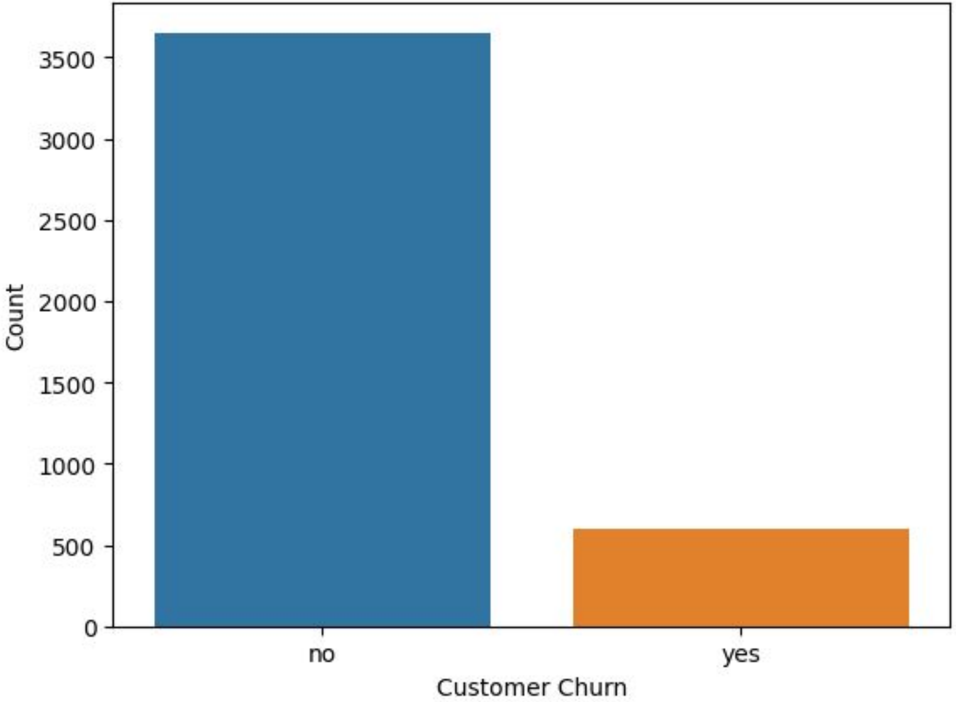
Variabel	Keterangan
state	US State
account_length	Total bulan customer menjadi user telco provider
area_code	Kode area
international_plan	Customer memiliki international plan
voice_mail_plan	Customer memiliki plan voice mail
number_vmail_messages	Total pesan voice mail
total_day_minutes/calls/charge	Total minutes/calls/charge pada day calls
total_eve_minutes/calls/charger	Total minutes/calls/charge pada evening calls
total_night_minutes/calls/charger	Total minutes/calls/charge pada night calls
total_int_minutes/calls/charger	Total minutes/calls/charge pada international calls
number_customer_service_calls	Total call kepada customer service
churn	Customer Churn

Jumlah Baris

4250

Customer Churn

Distribusi Customer Churn



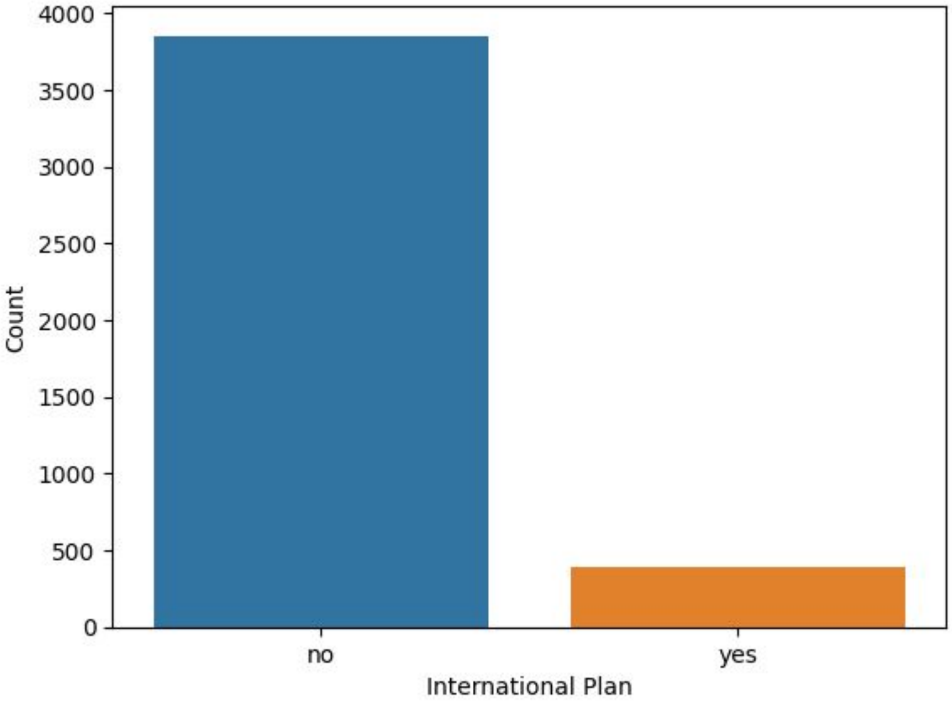
- Memiliki Custumer yang masih aktif sebesar 3652 dan yang telah hilang sebesar 598

Customer Churn
: Kode

```
sns.countplot(x = train['churn'])
plt.xlabel('Customer Churn', size = 10)
plt.ylabel('Count', size = 10)
```

International Plan

Customer yang mempunyai dan tidak International Plan



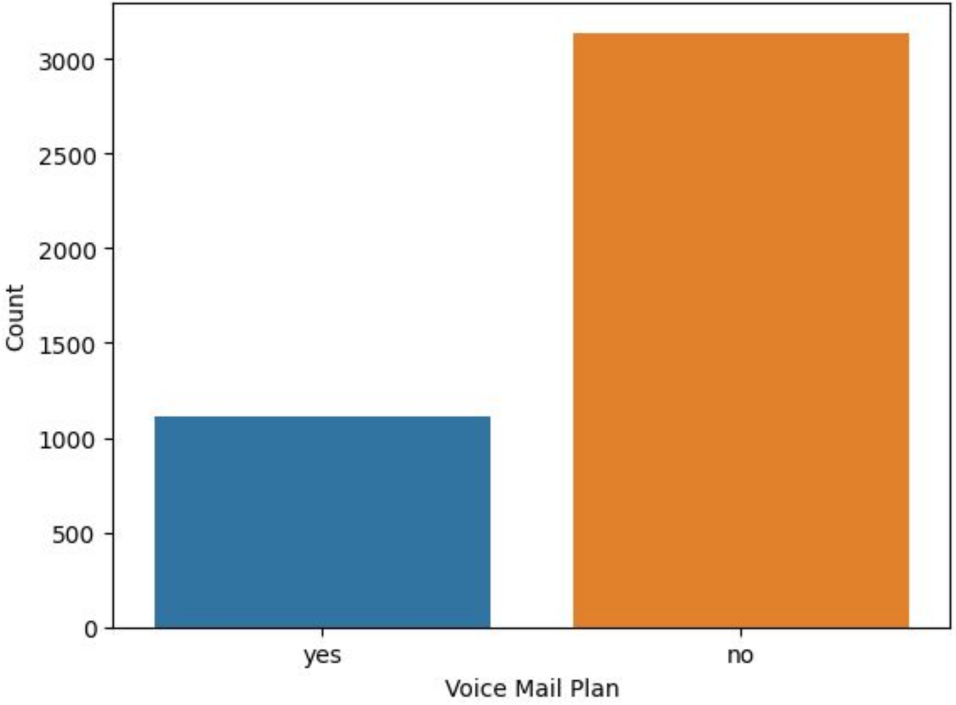
- Custumer yang memiliki International Plan sebanyak 3138 dan yang tidak memiliki International Plan sebanyak 1112.

International Plan
: Kode

```
[13] sns.countplot(x = train['international_plan'])
     plt.xlabel('International Plan', size = 10)
     plt.ylabel('Count', size = 10)
```

Voice Mail Plan

Customer yang mempunyai dan tida Voice Mail Calls

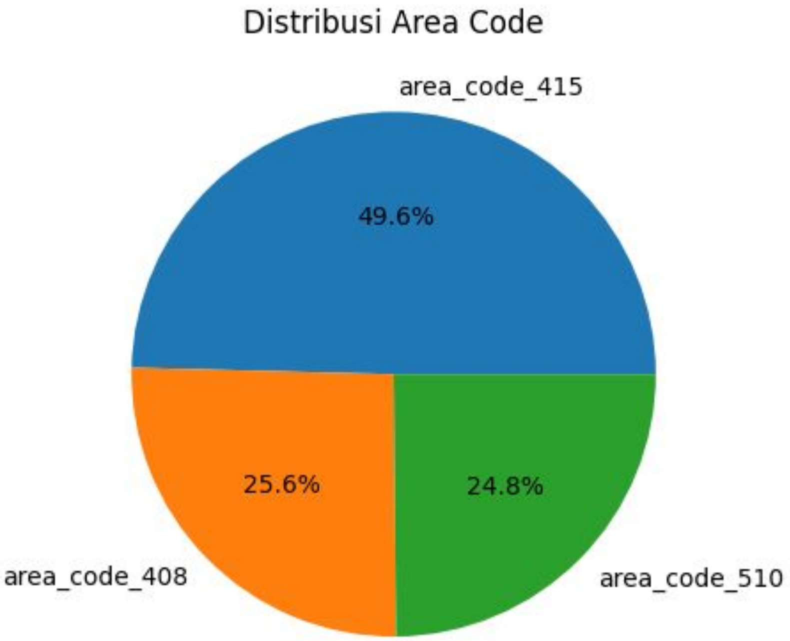


- Custumer yang memiliki Voice Mail Plan sebanyak 3854 dan yang tidak memiliki Voice Mail Plan sebanyak 396.

Voice Mail Plan
: Kode

```
[ ]  sns.countplot(x = train['voice_mail_plan'])
     plt.xlabel('Voice Mail Plan', size = 10)
     plt.ylabel('Count', size = 10)
```

Distribusi Area Code



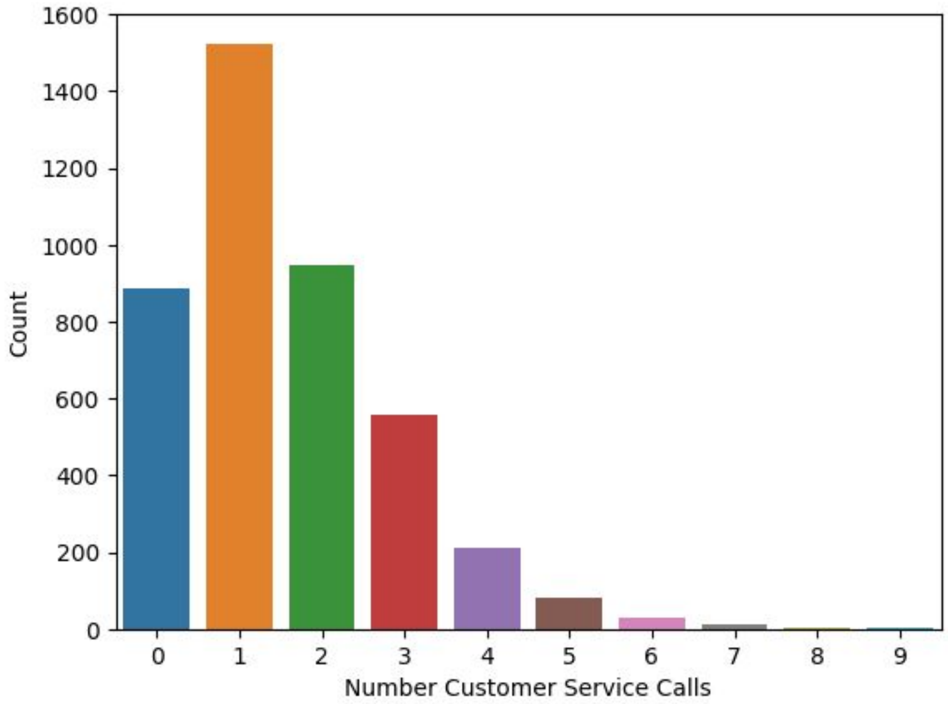
- Ditribusi Area Code yaitu terletak pada Area Code 415 sebanyak 49,6% atau 2108, terletak pada Area Code 408 sebanyak 25,6% atau 1088, dan terletak pada Area Code 510 sebanyak 24,8% atau 1056.

Distribusi Area Code : Kode

```
counts = train['area_code'].value_counts()
plt.pie(counts, labels=counts.index, autopct='%1.1f%%')
plt.title('Distribusi Area Code')
```

Number Customer Service Calls

Distribusi pada Number Customer Service Calls



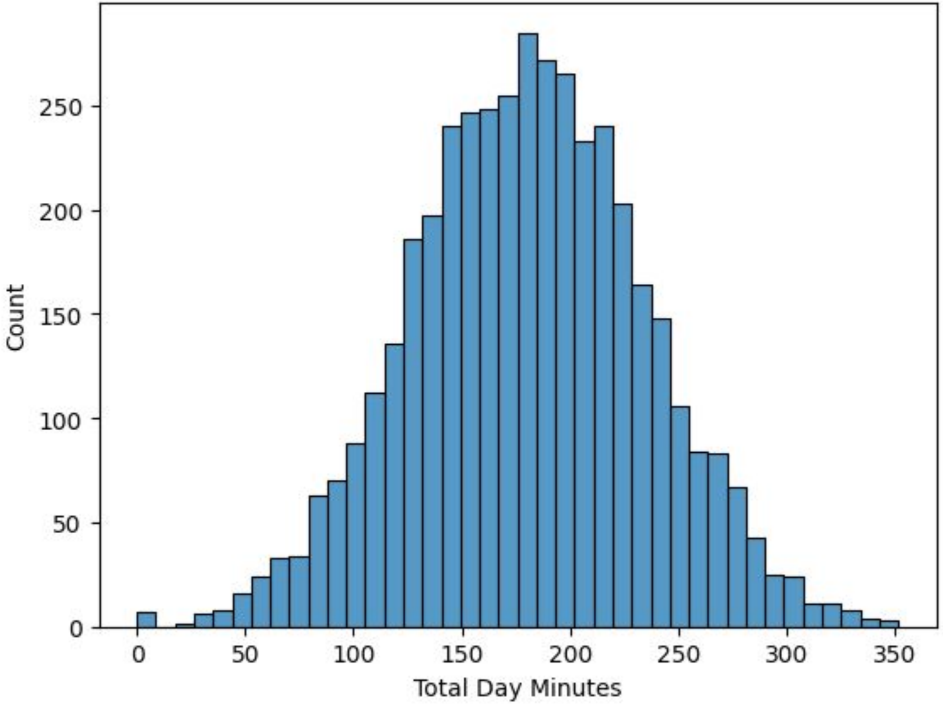
- Terdapat 79% Customer perna melakukan pelayanan panggilan dan masih terdapat Customer yang belum pernah melakukan layanan panggilan.

Number Customer Service Calls : Kode

```
sns.countplot(x = train['number_customer_service_calls']);
plt.xlabel('Number Customer Service Calls', size = 10)
plt.ylabel('Count', size = 10)
```

Total Day Minutes

Distribusi pada Total Day Minutes



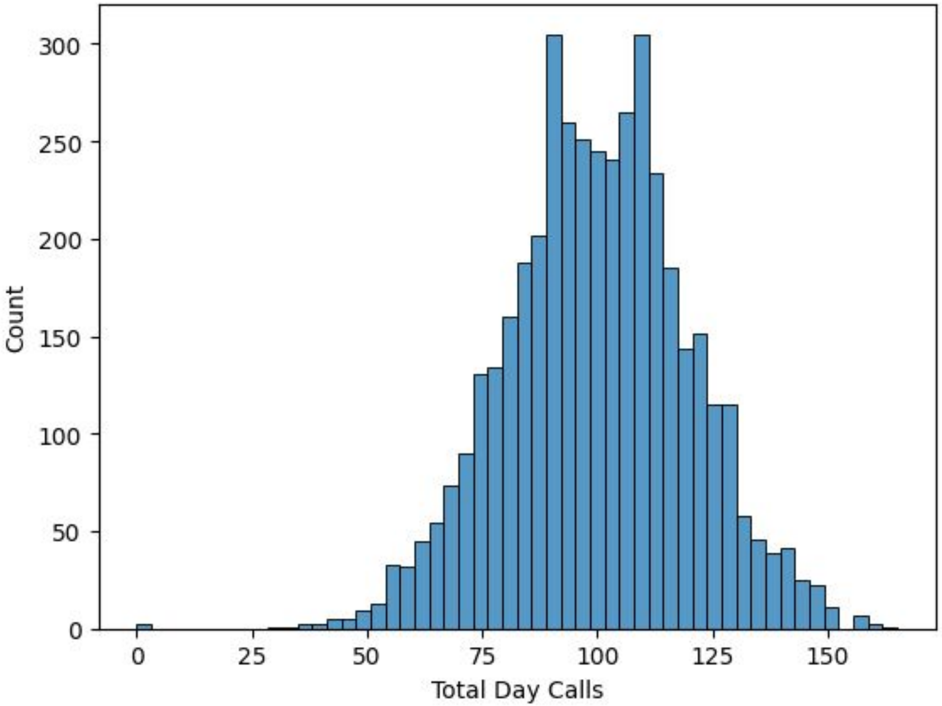
- Total day Calls memiliki nilai mean 180,26 dengan standar deviasinya sebesar 54,01.
- Memiliki median 180,45 dan modus sebesar 189,3.

Total Day Minutes
: Kode

```
[ ]  sns.histplot(x = train['total_day_minutes'])
     plt.xlabel('Total Day Minutes', size = 10)
     plt.ylabel('Count', size = 10)
```

Total Day Calls

Distribusi pada Total Day Calls



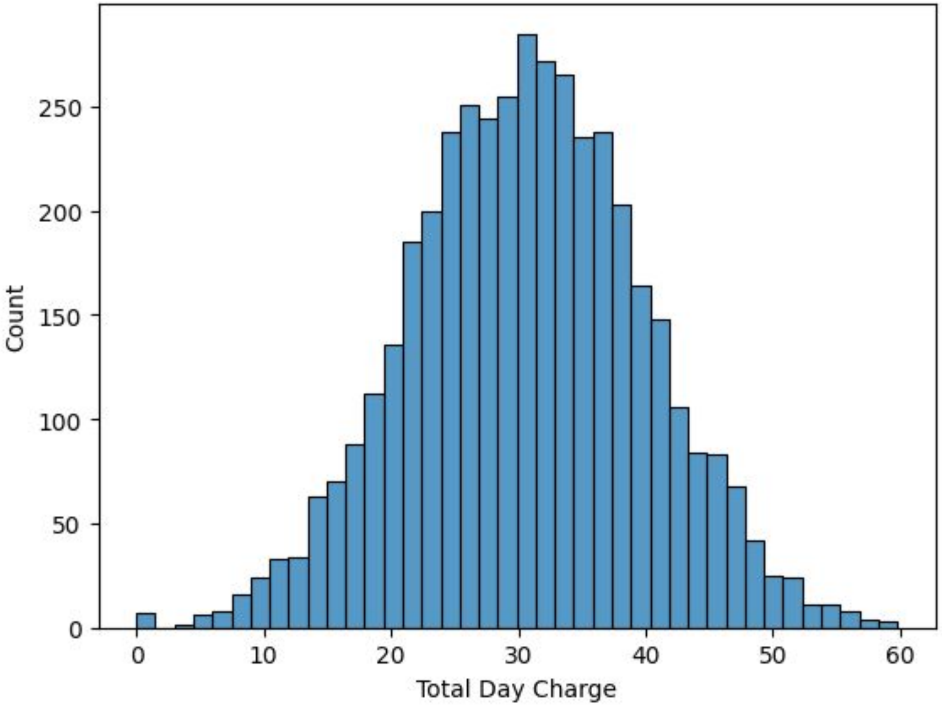
- Total day Calls memiliki nilai mean 99,91 dengan standar deviasinya sebesar 19,85.
- Memiliki median 100 dan modus sebesar 105.

Total Day Calls
: Kode

```
[ ]  sns.histplot(x = train['total_day_calls'])
     plt.xlabel('Total Day Calls', size = 10)
     plt.ylabel('Count', size = 10)
```

Total Day Charge

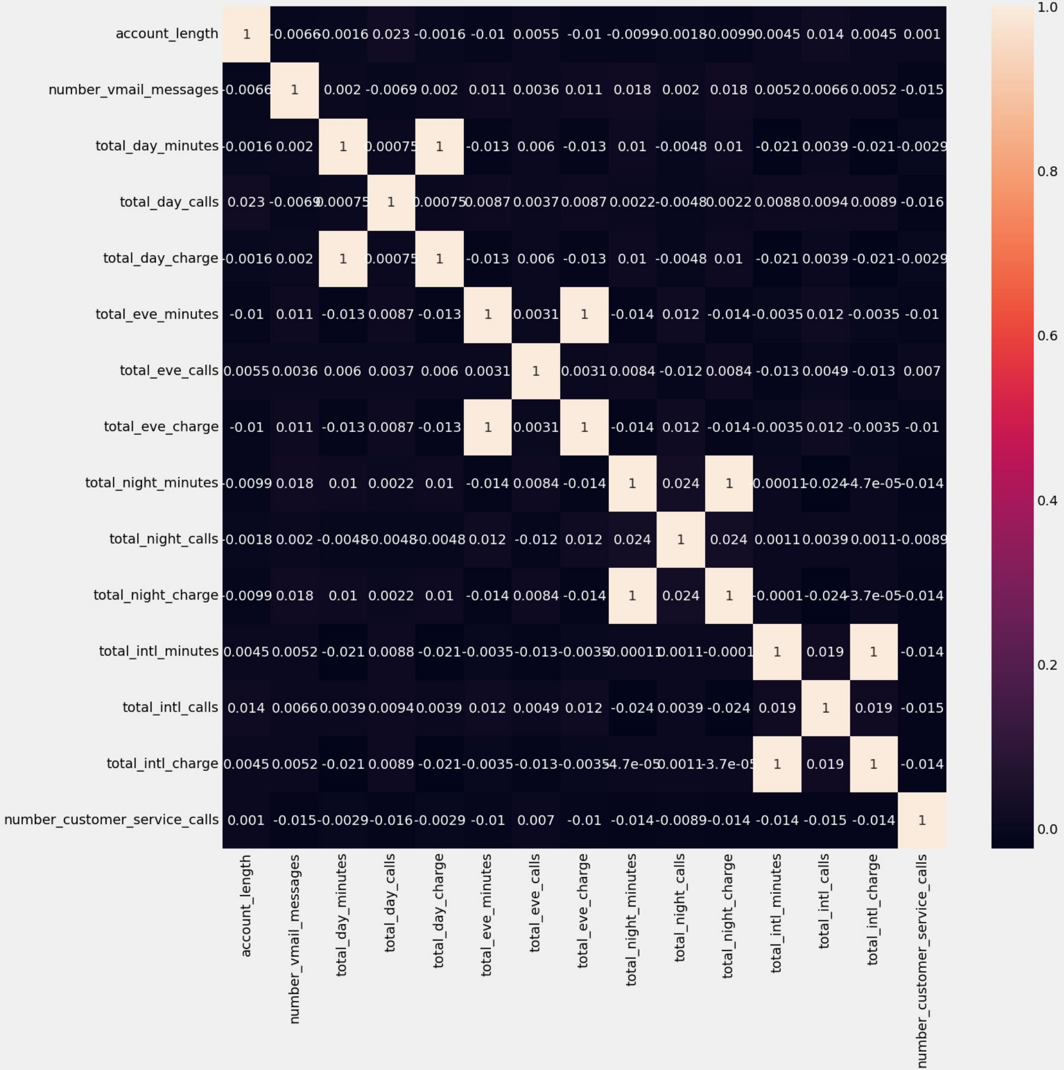
Distribusi pada Total Day Charge



- Total day Calls memiliki nilai mean 30,64 dengan standar deviasinya sebesar 9,18.
- Memiliki median 30,68 dan modus sebesar 32,18.

Total Day Charge
: Kode

```
[ ]  sns.histplot(x = train['total_day_charge'])
     plt.xlabel('Total Day Charge', size = 10)
     plt.ylabel('Count', size = 10)
```



Korelasi

Distribusi Korelasi pada setiap komponen.

Korelasi
: Kode

```
# penampilan korelasi antar kolom dengan heatmap
plt.figure(figsize=(15,15))
sns.heatmap(train.corr(), annot=True)
```

03 Model Building

- Pemrosesan Data
- Pelatihan Model
- Hyperparameter Tuning dengan Optuna
- Evaluasi Model

Pemrosesan Data

- Identifikasi nilai kosong
- Melihat Distribusi nilai ekstrem
- Data Splitting
- Feature Encoding
- Class-Balancing Operations

Pemrosesan Data

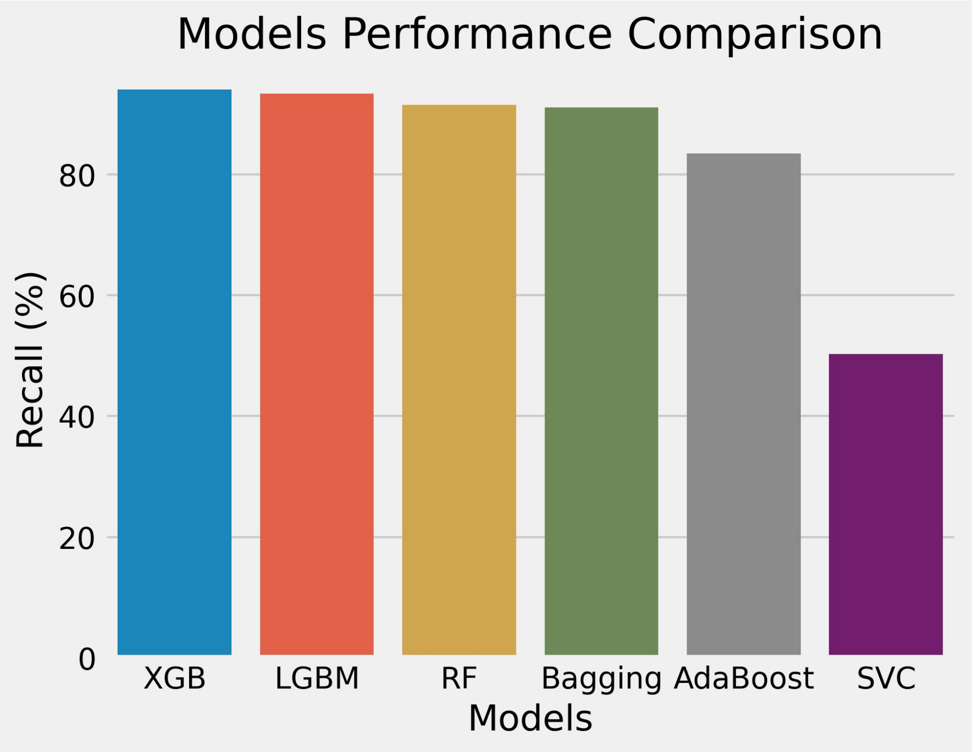
1. **Missing Values**
   Data bersih dari missing values
2. **Nilai Ekstrem**
   1 feature memiliki jumlah outliers tinggi. Diganti dengan median
3. **Data Splitting**
   Membagi data menjadi train data dan validation data.
4. ***Feature Encoding***
   Mengubah data kategorik menjadi data numerik.
5. ***Class Balancing Operations***
   Menstabilkan kelas dominan pada target dengan SMOTE

Pelatihan Model

- Melatih banyak model secara stimultan
- Validasi performa menggunakan *cross-validation*
- Memilih 3 model berperforma terbaik

Pelatihan Model

Melatih Banyak Model Secara Stimultan dengan Cross Validation



- 3 model terbaik : XGB, LGBM dan RF
- Performa model terpilih akan diitingkatkan dengan Hyperparameter Tuning

Pelatihan Model : Kode

```
# train all model
xgb = XGBClassifier(random_state = 42).fit(X_train, y_train)
lgb = lgb.LGBMClassifier(random_state=42).fit(X_train, y_train)
bagging = BaggingClassifier(random_state=42).fit(X_train, y_train)
rf = RandomForestClassifier(random_state=42).fit(X_train, y_train)
ada = AdaBoostClassifier(random_state=42).fit(X_train, y_train)
svm = SVC(random_state=42).fit(X_train, y_train )
```

```
models = [svm, rf, ada, bagging, lgb, xgb]
cv = StratifiedKFold(n_splits = 5, random_state=42, shuffle=True)

for model in models:
    crosval = cross_val_score(model, X_train, y_train, cv = cv, scoring = 'recall')
    print(f'Model Name {model}')
    print(f'Recall Score : {mean(crosval)}')
```

Hyperparameter Tuning dengan Optuna

- Mencari parameter ideal dari 3 model yang dipilih sebelumnya
- Menggunakan *module python optuna*
- Model baru akan dibangun berdasarkan parameter terbaik

Hyperparameter Tuning

*Untuk reproduksibilitas, random_state = 0

Parameter XGBoost

```
▾                         XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=0.9, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=0.1, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=0.04, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=7, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=700, n_jobs=None, num_parallel_tree=None,
              predictor=None, random_state=0, ...)
```

Hyperparameter Tuning : Kode

```
# hyperparameter for xgboost
# Define objective function for Optuna
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 1000, step=50),
        'max_depth': trial.suggest_int('max_depth', 3, 10),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3, step=0.01),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.1, 1.0, step=0.1),
        'subsample': trial.suggest_float('subsample', 0.1, 1.0, step=0.1),
        'gamma': trial.suggest_float('gamma', 0.1, 1.0, step=0.1),
        'reg_alpha': trial.suggest_float('reg_alpha', 0.1, 10.0, step=0.1),
        'reg_lambda': trial.suggest_float('reg_lambda', 0.1, 10.0, step=0.1),
        'random_state': 0,
        'objective': 'binary:logistic',
        'eval_metric': 'auc',
    }
    # Train XGBoost model with early stopping
    model = xgb.XGBClassifier(**params)
    eval_set = [(X_val, y_val)]
    model.fit(X_train, y_train, eval_set=eval_set, early_stopping_rounds=50, verbose=False)

    # Predict on validation set
    y_pred = model.predict(X_val)

    # Evaluate on test set and return AUC score
    recall = recall_score(y_val, y_pred)
    return recall

# Define study and optimize hyperparameters with Optuna
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)

# Print best hyperparameters and score
print('Best hyperparameters: ', study.best_params)
print('Best score: ', study.best_value)
```

Hyperparameter Tuning

*Untuk reproduksibilitas, random_state = 0

Parameter LGBM

```
▾                                LGBMClassifier
LGBMClassifier(learning_rate=0.15000000000000002, max_depth=9, n_estimators=750,
               num_leaves=214, random_state=0, reg_alpha=8.2, reg_lambda=7.3,
               subsample=0.2)
```

Parameter Random Forest

```
▾                          RandomForestClassifier
RandomForestClassifier(max_depth=10, max_features='log2', min_samples_leaf=7,
                       min_samples_split=16, n_estimators=50, random_state=0)
```

Hyperparameter Tuning : Kode

```python
# hyperparameter optimization for lgbm
def objective(trial):
    params = {
        'objective': 'binary',
        'metric': 'auc',
        'boosting_type': 'gbdt',
        'n_estimators': trial.suggest_int('n_estimators', 50, 1000, step=50),
        'max_depth': trial.suggest_int('max_depth', 3, 10),
        'learning_rate': trial.suggest_float('learning_rate', 0.01, 0.3, step=0.01),
        'colsample_bytree': trial.suggest_float('colsample_bytree', 0.1, 1.0, step=0.1),
        'subsample': trial.suggest_float('subsample', 0.1, 1.0, step=0.1),
        'reg_alpha': trial.suggest_float('reg_alpha', 0.1, 10.0, step=0.1),
        'reg_lambda': trial.suggest_float('reg_lambda', 0.1, 10.0, step=0.1),
        'num_leaves': trial.suggest_int('num_leaves', 10, 300),
        'random_state': 0,
    }

    # Train LightGBM model
    model = lgb.LGBMClassifier(**params)
    model.fit(X_train, y_train, eval_set=eval_set, early_stopping_rounds=50, verbose=False)

    # Predict on validation set
    y_pred = model.predict(X_val)

    # Evaluate on test set and return AUC score
    recall = recall_score(y_val, y_pred)
    return recall

# Define study and optimize hyperparameters with Optuna
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)

# Print best hyperparameters and score
print('Best hyperparameters: ', study.best_params)
print('Best score: ', study.best_value)
```

Hyperparameter Tuning : Kode

```python
# hyperparameter optimization for random forest classifier

# Define objective function for Optuna
def objective(trial):
    params = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 1000, step=50),
        'max_depth': trial.suggest_int('max_depth', 3, 10),
        'min_samples_split': trial.suggest_int('min_samples_split', 2, 20),
        'min_samples_leaf': trial.suggest_int('min_samples_leaf', 1, 20),
        'max_features': trial.suggest_categorical('max_features', ['sqrt', 'log2', None]),
        'random_state': 0,
    }

    # Train Random Forest model
    model = RandomForestClassifier(**params)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_val)

    # Evaluate on test set and return accuracy score
    recall = recall_score(y_val, y_pred)
    return recall

# Define study and optimize hyperparameters with Optuna
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=100)

# Print best hyperparameters and score
print('Best hyperparameters: ', study.best_params)
print('Best score: ', study.best_value)
```

Evaluasi Model

- Classification report
- Confusion matrix
- ROC Curve

Classification Report

Random Forest

	precision	recall	f1-score	support
0	0.86	0.97	0.91	730
1	0.97	0.84	0.90	730
accuracy			0.91	1460
macro avg	0.91	0.91	0.91	1460
weighted avg	0.91	0.91	0.91	1460

- Random forest memiliki performa yang baik
- Recall terhadap kelas positif mencapai 84 %

Classification Report

LGBM

	precision	recall	f1-score	support
0	0.91	0.97	0.94	730
1	0.97	0.90	0.93	730
accuracy			0.93	1460
macro avg	0.94	0.93	0.93	1460
weighted avg	0.94	0.93	0.93	1460

- LGBM memiliki performa yang relatif lebih baik dari RF
- Recall terhadap kelas positif mencapai 90%

Classification Report

XGBoost

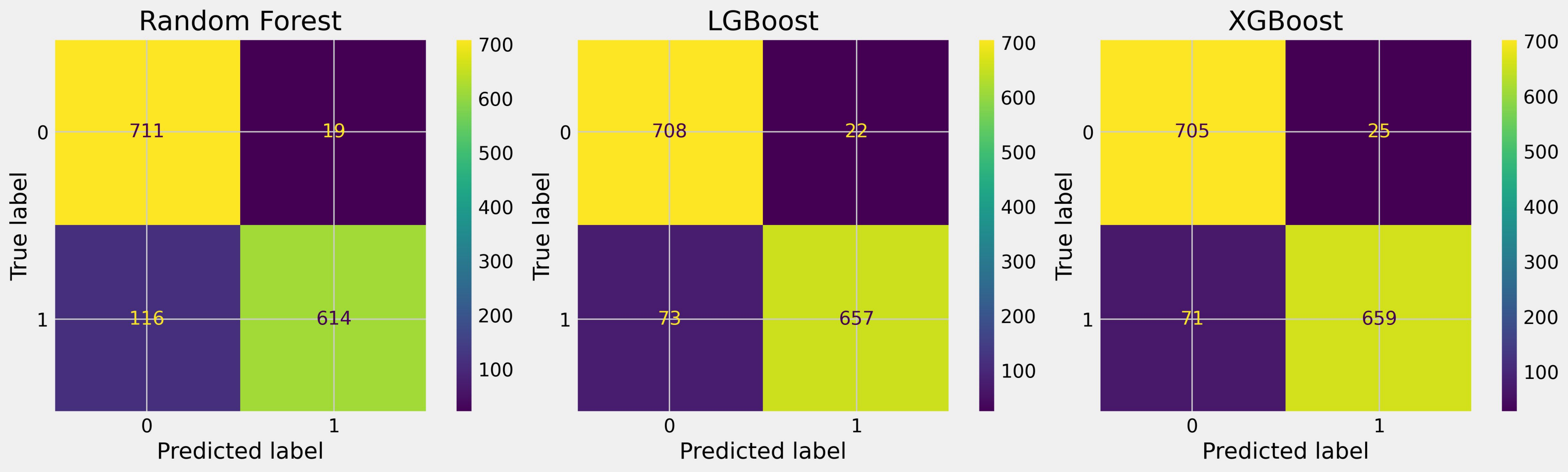
	precision	recall	f1-score	support
0	0.91	0.97	0.94	730
1	0.96	0.90	0.93	730
accuracy			0.93	1460
macro avg	0.94	0.93	0.93	1460
weighted avg	0.94	0.93	0.93	1460

- XGBoost memiliki performa yang mirip dengan LGBM
- Recall terhadap kelas positif mencapai 90%

Classification Report : Kode

```
models = [rf, lgb, xgb]
for model in models:
    y_pred = model.predict(X_val)
    print(model)
    print(classification_report(y_val, y_pred))
```

Confusion Matrix



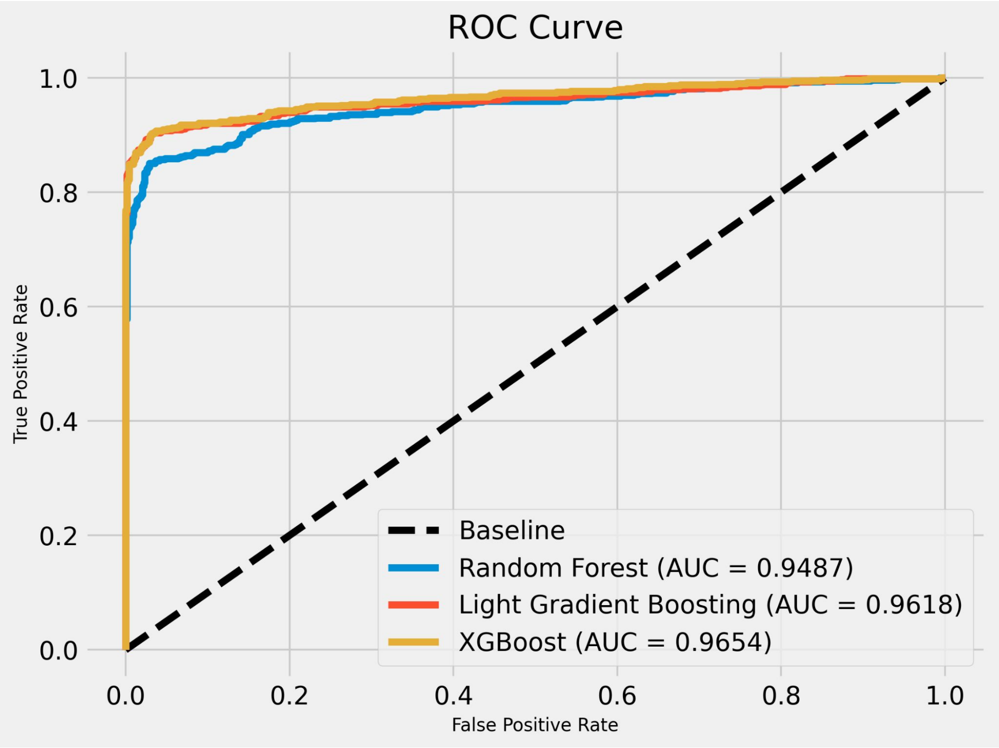
XGBoost memiliki nilai True Positive (TP) yang paling tinggi

Confusion Matrix
: Kode

```
names = ['dummy', 'Random Forest', 'LGBoost', 'XGBoost']
i = 0
models = [rf, lgb, xgb]
for i, model in enumerate(models):
    y_pred = model.predict(X_val)
    print(model)
    cm = confusion_matrix(y_val, y_pred)
    cm_display = ConfusionMatrixDisplay(cm)

    cm_display.plot()
    plt.title(names[i+1])
    plt.savefig(f'{names[i+1]}.png', dpi=300, bbox_inches='tight')
    plt.show()
```

ROC Curve



- XGBoost memiliki nilai AUC paling tinggi dengan 0.9654
- Performa XGboost tidak berbeda jauh dengan LGB

ROC Curve : Kode

```python
def roc_comparison(model, name):
    y_pred = model.predict(X_val)
    y_probs = model.predict_proba(X_val)
    y_probs = y_probs[:, 1]

    fpr, tpr, _ = roc_curve(y_val, y_probs)
    auc = round(roc_auc_score(y_val, y_probs), 4)
    plt.plot(fpr, tpr, label = (f'{name} (AUC = {auc:.4f})'))
    plt.legend(loc='best')
```

```python
# create a plot
ax, fig = plt.subplots(figsize=(8,6))

# create a baseline
plt.plot([0,1], [0,1], linestyle='--', label='Baseline', color = 'black')

roc_comparison(rf, 'Random Forest')
roc_comparison(lgb, 'Light Gradient Boosting')
roc_comparison(xgb, 'XGBoost')

# set plot axises
plt.ylabel('True Positive Rate', size = 10)
plt.xlabel('False Positive Rate', size = 10)
plt.title('ROC Curve', size=18)
plt.legend()

# show the plot
plt.savefig('ROC Curve.png', dpi=300, bbox_inches='tight')
plt.show()
```

04 Kesimpulan

- Intisari projek data dan saran untuk data model

Simpulan

Dalam project modeling Customer Churn ini, telah dibuat tiga model yaitu XGBClassifier, Random Forest Classifier, dan LGB dengan menggunakan data pelanggan suatu perusahaan. Evaluasi model dilakukan dengan menggunakan metrik AUC, akurasi, presisi, recall, dan F1-score.

Berdasarkan hasil evaluasi, dapat disimpulkan bahwa model XGB dan LGB lebih unggul dalam memprediksi Customer Churn dibandingkan dengan model RandomForestClassifier. Model XGB dianggap sebagai model terbaik karena memiliki nilai AUC yang lebih tinggi dan Recall yang mencapai 0.96.

Model ini dapat digunakan oleh perusahaan untuk mengidentifikasi Customer Churn dan mengambil tindakan untuk mempertahankan pelanggan tersebut. Namun, perlu mempertimbangkan faktor bisnis yang terkait dengan tindakan yang harus diambil berdasarkan prediksi model.

Thanks!

Check produced python code for this project by clicking this [link !]