Penanganan Imbalanced Dataset dan Seleksi Fitur dalam Prediksi Customer Churn

1. Pendahuluan

Customer churn merupakan permasalahan penting dalam industri layanan pelanggan. Untuk memprediksi churn, kita menggunakan model Logistic Regression. Tantangan utama dalam kasus ini adalah ketidakseimbangan kelas (class imbalance), yaitu jumlah pelanggan yang churn jauh lebih sedikit daripada yang tidak. Oleh karena itu, dilakukan beberapa pendekatan penanganan class imbalance dan seleksi fitur.

Untuk source code dan dataset bisa di akses di [repository github].

2. Dataset

Dataset yang digunakan adalah data pelanggan dari perusahaan telekomunikasi, di mana target variabel adalah churn_num (0 = tidak churn, 1 = churn). Dataset telah melalui tahap encoding dan preprocessing.

3. Penanganan Imbalanced Class

Berikut adalah empat pendekatan yang digunakan:

3.1. Method 1: Logistic Regression dengan class_weight='balanced'

```python
logreg_balanced = LogisticRegression(**params)
logreg_balanced.fit(X_train, y_train)
y_pred_balanced = logreg_balanced.predict(X_test)
logit_roc_auc_balanced = roc_auc_score(y_test, y_pred_balanced)
print(classification_report(y_test, y_pred))
print(f"The area under the curve is: {logit_roc_auc_balanced}")

with mlflow.start_run():
    mlflow.log_params(params)
    mlflow.log_metric("area_under_curve", logit_roc_auc_balanced)
    mlflow.set_tag("Training Info", "Basic Logistic regression model with balanced class")
    signature_balanced = infer_signature(X_train, logreg_balanced.predict(X_test))
    model_info_balanced = mlflow.sklearn.log_model(
        sk_model=logreg_balanced,
        artifact_path="balanced_log_reg",
        signature=signature_balanced,
        input_example=X_train,
        registered_model_name="balanced_log_reg",
    )
```

Pendekatan ini mengatur bobot kelas secara otomatis berdasarkan distribusi data

3.2. Method 2: Custom Class Weights

```python
params = {
    "random_state" : 13,
    "class_weight" : {0:90, 1:10}
}
logreg_custom_ = LogisticRegression(**params)
logreg_custom_.fit(X_train, y_train)
y_pred_custom_ = logreg_custom_.predict(X_test)
logit_roc_auc_custom_ = roc_auc_score(y_test, y_pred_custom_)
print(classification_report(y_test, y_pred_custom_))
print(f"The area under the curve is: {logit_roc_auc_custom_}")

with mlflow.start_run():
    mlflow.log_params(params)
    mlflow.log_metric("area_under_curve", logit_roc_auc_custom_)
    mlflow.set_tag("Training Info", "Basic Logistic regression model with custom class weights")
    signature_balanced = infer_signature(X_train, logreg_balanced.predict(X_test))
    model_info_balanced = mlflow.sklearn.log_model(
        sk_model=logreg_custom_,
        artifact_path="custom_class_weights_log_reg",
        signature=signature_balanced,
        input_example=X_train,
        registered_model_name="custom_class_weights_log_reg",
    )
```

Bobot disesuaikan secara manual untuk memperbesar kontribusi minoritas.

3.3. Method 3: Resampling Data

```python
params = {
    "random_state":13
}
logreg_resampled = LogisticRegression(**params)

X=resampled_df[resampled_df[resampled_df.select_dtypes(include=np.number).
columns.tolist()].columns.difference(['customer_id','churn_num'])]
y=resampled_df['churn_num']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=0)
```

Minoritas (kelas 1) di-upsample agar jumlahnya sama dengan mayoritas.

3.4. Method 4: SMOTE (Synthetic Minority Over-sampling Technique)

```python
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=0, sampling_strategy=1.0)

X=df_enc_sel[df_enc_sel[df_enc_sel.select_dtypes(include=np.number).columns.tolist()].columns.difference(['customer_id','churn_num'])]
y=df_enc_sel['churn_num']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

X_train_smote, y_train_smote = sm.fit_resample(X_train, y_train)
params = {
    "random_state":13
}
logreg_smote = LogisticRegression(**params)
logreg_smote.fit(X_train_smote, y_train_smote)
y_pred_smote = logreg_smote.predict(X_test)
logit_roc_auc_smote = roc_auc_score(y_test, y_pred_smote)
print(classification_report(y_test, y_pred_smote))
print(f"The area under the curve is: {logit_roc_auc_smote}")

with mlflow.start_run():
    mlflow.log_params(params)
    mlflow.log_metric("area_under_curve", logit_roc_auc_smote)
    mlflow.set_tag("Training Info", "Logistic regression model with SMOTE to fix imbalanced data")
    signature_smote = infer_signature(X_train_smote, logreg_balanced.predict(X_test))
    model_info_smote = mlflow.sklearn.log_model(
        sk_model=logreg_smote,
        artifact_path="logreg_smote",
        signature=signature_smote,
        input_example=X_train_smote,
        registered_model_name="logreg_smote",
    )
```

SMOTE membuat sampel sintetis berdasarkan tetangga terdekat dari kelas minoritas.

4. Seleksi Fitur

4.1. Variance Threshold

Fitur dengan variansi rendah dihapus, kemudian training dilakukan dengan SMOTE dan Logistic Regression.

4.2. Recursive Feature Elimination (RFE)

```
from sklearn.feature_selection import RFE
rfe = RFE(model)
rfe = rfe.fit(X, y)
selected_cols_rfe = X.columns[rfe.support_]
print(list(selected_cols_rfe))

X_rfe_cols=df_enc_sel[['SeniorCitizen', 'contract_code', 'dependents_code', 'multiple_lines_code', 'online_backup_code',
                'online_security_code', 'paperless_billing_code', 'phone_service_code', 'tech_support_code']]
y=df_enc_sel['churn_num']
X_train__rfe, X_test_rfe, y_train_rfe, y_test_rfe = train_test_split(X_rfe_cols, y, test_size=0.3, random_state=0)
sm = SMOTE(random_state=0, sampling_strategy=1.0)
X_train__rfe, y_train_rfe = sm.fit_resample(X_train__rfe, y_train_rfe)
params = {
    "random_state":13
}
```

RFE memilih fitur paling penting berdasarkan performa model.

5. Evaluasi Model

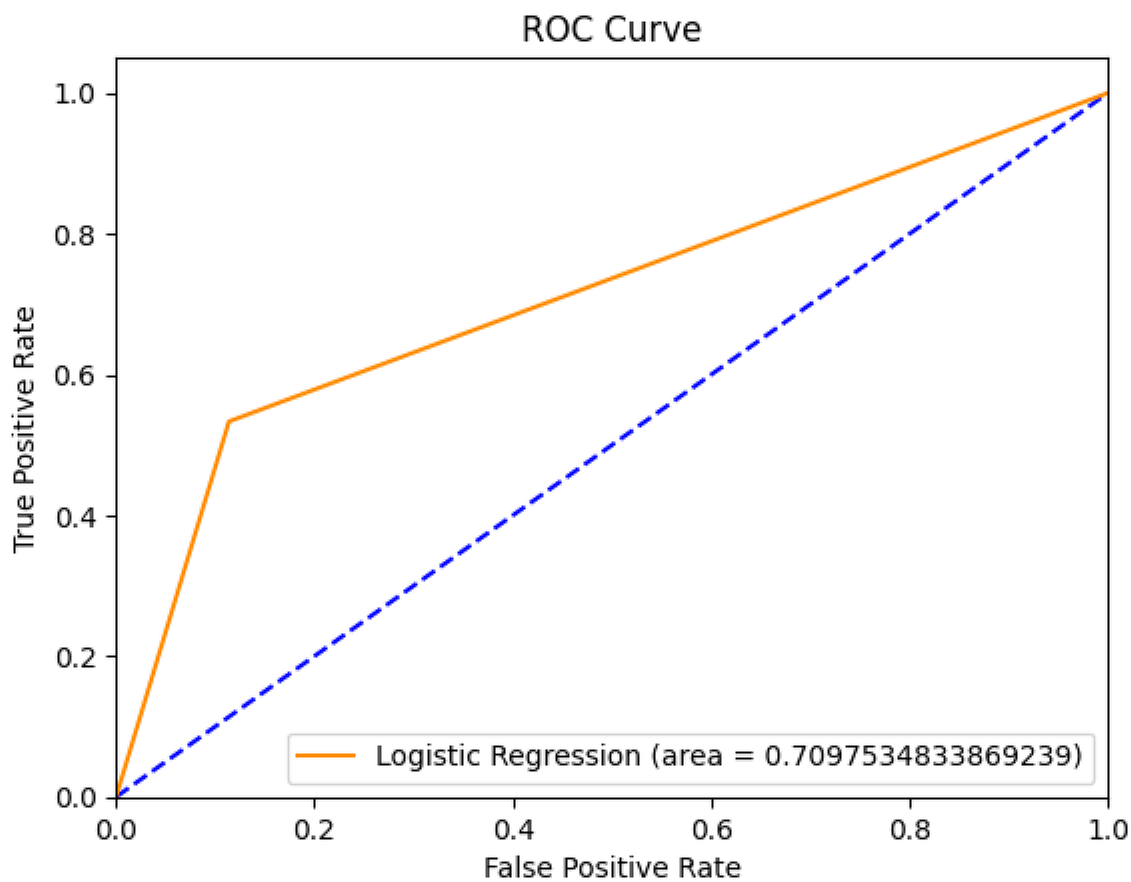
Evaluasi dilakukan menggunakan klasifikasi report dan ROC AUC score. Berikut beberapa hasil:

```
['SeniorCitizen', 'contract_code', 'dependents_code', 'multiple_lines_code', 'online_backup_code',
              precision    recall  f1-score   support

         0.0       0.91      0.69      0.78      1555
         1.0       0.48      0.81      0.61       555

    accuracy                           0.72      2110
   macro avg       0.70      0.75      0.69      2110
weighted avg       0.80      0.72      0.74      2110

The area under the curve is: 0.7506792966600041
Successfully registered model 'logreg_rfe_t'.
2025/04/28 19:31:41 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for
Created version '1' of model 'logreg_rfe_t'.
```

```
              precision    recall  f1-score   support

         0.0       0.83      0.61      0.71      1555
         1.0       0.38      0.66      0.48       555

    accuracy                           0.62      2110
   macro avg       0.60      0.63      0.59      2110
weighted avg       0.71      0.62      0.65      2110

The area under the curve is: 0.6337157092784103
Successfully registered model 'logreg_smote_t'.
2025/04/28 19:31:13 INFO mlflow.store.model_registry.abstract_store: Waiting up to 300 seconds for
Created version '1' of model 'logreg_smote_t'.
```

6. Visualisasi

Confusion Matrix dan ROC Curve (Contoh)



7. Logging dengan MLflow

Semua model dilog menggunakan MLflow untuk tracking parameter, metrik, dan artefak model.

```
with mlflow.start_run():
    mlflow.log_params(params)
    mlflow.log_metric("area_under_curve", logit_roc_auc_rfe_t)
    mlflow.set_tag("Training Info", "Logistic regression model with SMOTE & RFE for feature selection")
    signature_rfe_t = infer_signature(X_train__rfe, logreg_balanced.predict(X_test))
    model_info_rfe_t = mlflow.sklearn.log_model(
        sk_model=logreg_rfe_t,
        artifact_path="logreg_rfe_t",
        signature=signature_rfe_t,
        input_example=X_train__rfe,
        registered_model_name="logreg_rfe_t",
    )
```

8. Kesimpulan

SMOTE dan penggunaan class_weight='balanced' terbukti membantu meningkatkan performa model dalam kasus data tidak seimbang. Rekomendasi terbaik adalah menggabungkan teknik balancing dengan seleksi fitur seperti RFE untuk hasil optimal.