Analisis dan Prediksi Default Kartu Kredit dengan Model Machine Learning

Donny[#1], Hendra Bunyamin[*2]

[#]*Program Studi S1 Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Maranatha*
Jln. Prof. Drg. Surya Sumantri No. 65, Sukawarna, Bandung, Indonesia
[1]1972027@maranatha.ac.id
[2]hendra.bunyamin@it.maranatha.edu

Abstract — **This research aims to study dataset partitioning, prediction, and model selection with the best performance in the case of credit card default classification. The trained models include logistic regression, k-nearest neighbor, artificial neural networks, decision trees, support vector machines, random forests, and gradient boosting. The objective is to determine the best model and the most suitable model for credit card default classification. The research involves dataset visualization, dataset preprocessing, model selection, parameter tuning for each model, and using machine learning models to predict the test dataset. Dataset visualization is conducted to understand dataset characteristics and identify patterns among features. Dataset preprocessing is performed to handle missing values, feature normalization or standardization, outlier removal, and encoding or transformation of categorical features. Model selection is carried out to choose the most appropriate machine learning model for credit card default classification. Parameter tuning for each model aims to improve model performance and generalization. Finally, the trained machine learning models are utilized to predict the test dataset and provide the final evaluation of model performance, specifically the F1 score.**

Keywords— **Dataset, F1 score, Classification, Model, Prediction**

I. PENDAHULUAN

Default kartu kredit terjadi setelah customer menunggak pembayaran kartu kreditnya selama enam bulan. Status default kartu kredit tidak hanya mempengaruhi penerbit kartu kredit, tetapi juga mempengaruhi customer untuk mendapatkan persetujuan kartu kredit, pinjaman, dan layanan berbasis kredit lainnya. Contohnya, customer setuju untuk melakukan pembayaran minimum pada tanggal jatuh tempo yang tercantum di akunnya. Jika customer tidak melakukan pembayaran kartu kredit minimal enam bulan berturut-turut, kartu kredit akan dinyatakan default. Selanjutnya, penerbit kartu kredit akan menutup akun dan melaporkan status default kartu kredit ke biro kredit.

Penelitian ini bertujuan untuk membantu penerbit kartu kredit dalam memprediksi kemungkinan default kartu kredit oleh para pelanggan. Dalam penelitian ini, terdapat beberapa fitur yang berpotensi mempengaruhi kemungkinan default kartu kredit, seperti batas kredit (limit_bal), jenis kelamin (sex), pendidikan (education), status pernikahan (marriage), usia (age), serta status pembayaran (pay_0 hingga pay_6), jumlah tagihan (bill_amt1 hingga bill_amt6), dan jumlah pembayaran (pay_amt1 hingga pay_amt6).

Dengan adanya penelitian ini, diharapkan penerbit kartu kredit dapat mengoptimalkan proses pengambilan keputusan kredit mereka dan mengidentifikasi pelanggan yang berpotensi melakukan default kartu kredit dengan lebih efektif. Hal ini dapat membantu mengurangi risiko kredit, meningkatkan manajemen risiko, dan mengoptimalkan keuntungan bagi penerbit kartu kredit.

II. KAJIAN TEORI

A. Machine Learning

Machine learning merupakan bagian dari penerapan matematika dan ilmu komputer. Machine learning menggunakan tools dari matematika seperti probabilitas, statistik, dan teori optimasi untuk mengekstrak pola dari data. Hal yang dibutuhkan untuk belajar machine learning adalah kumpulan data dan khusus untuk setiap item input yang memiliki output yang diinginkan maka algoritma machine learning akan disebut supervised learning. Proses ini sangat berbeda dari rekayasa perangkat lunak tradisional. Umumnya, bahasa pemrograman machine learning menggunakan Python dan R.

1) Logistic Regression

Logistic regression adalah jenis analisis statistik yang sering digunakan seorang data analis untuk pemodelan prediktif [2].

2) K-Nearest Neighbor

k-Nearest Neighbor (k-NN) adalah algoritma yang berfungsi untuk melakukan klasifikasi suatu data berdasarkan data pembelajaran yang diambil dari k tetangga terdekatnya. Bila dalam kasus terdapat lebih dari 2 variabel independen maka menghitung jaraknya dapat menggunakan rumus Euclidean Distance.

3) Artificial Neural Network

Artificial neural network (ANN) adalah upaya untuk mensimulasikan jaringan neuron yang membentuk otak manusia sehingga komputer akan dapat mempelajari berbagai hal dan membuat keputusan dengan cara yang mirip manusia. ANN dibuat dengan memprogram komputer biasa untuk berperilaku seolah-olah mereka merupakan sel-sel otak yang saling berhubungan [4].

4) Decision Trees

Decision trees (DT) merupakan suatu struktur yang digunakan untuk membantu proses pengambilan keputusan**.** DT disebut sebagai "tree" karena struktur ini menyerupai sebuah pohon dengan akar, batang, dan percabangannya.

5) Support Vector Machines

Support Vector Machines (SVM) adalah suatu teknik untuk melakukan prediksi, baik untuk kasus klasifikasi maupun regresi**.** SVM berada dalam satu kelas dengan Artificial Neural Network (ANN) dalam hal fungsi dan kondisi permasalahan yang dapat diselesaikan. Keduanya masuk dalam kelas supervised learning. SVM memiliki manfaat sebagai pemisah untuk memaksimalkan jarak antar kelas data dengan mengukur margin dan mencari titik maksimal.

6) Random Forest

Random Forests adalah teknik ansamble yang menggabungkan banyak decision tree. Random forests biasanya memiliki kinerja generalisasi yang lebih baik daripada decision tree individu karena keacakannya membantu mengurangi model yang bervariasi. Random forests memiliki keuntungan lain seperti kurang sensitif terhadap outlier dalam kumpulan data dan tidak memerlukan banyak penyetelan parameter [3].

7) Gradient Boosting

Gradient Boosting adalah salah satu algoritma yang paling populer. Gradient boosting bekerja dengan menambahkan prediktor secara berurutan ke ansamble. Gradient boosting memiliki manfaat lain seperti menyesuaikan prediktor baru dengan residual kesalahan yang dilakukan prediktor sebelumnya [3].

B. Matriks Pengukuran Kinerja

Model machine learning perlu diukur kinerjanya supaya kinerja model memberikan hasil yang baik. Pengukuran kinerja tersebut dapat dihitung dengan confusion matrix, akurasi, precision, recall, dan F1 score.

1) F1 Score

F1 score adalah rata-rata harmonik dari *precision* dan *recall*, dan F1 score menggabungkan keduanya dalam satu angka. F1 score digunakan ketika *precision* dan *recall* lebih diutamakan. Dalam penelitian ini, F1 score akan digunakan.

III. ANALISIS DAN RANCANGAN SISTEM

A. Tahapan dalam Pembuatan Model Machine Learning

Pembuatan model machine learning terdiri dari beberapa tahap, yaitu : visualisasi dataset, pemrosesan dataset, pemilihan model-model machine learning, melakukan fine-tuning pada setiap model machine learning, dan menggunakan model machine learning untuk memprediksi test set.

B. Visualisasi Dataset

Dataset yang akan digunakan adalah data default kartu kredit di Taiwan pada tahun 2005. Fitur-fitur pada dataset tersebut adalah limit_bal, sex, education, marriage, age, pay_0, pay_2, pay_3, pay_4, pay_5, pay_6, bill_amt1, bill_amt2, bill_amt3, bill_amt4, bill_amt5, bill_amt6, pay_amt1, pay_amt2, pay_amt3, pay_amt4, pay_amt5, pay_amt6.

C. Melakukan Fine-Tuning Setiap Model Machine Learning

Pada penelitian ini, fine tuning yang akan dilakukan adalah dengan menggunakan grid search. Grid search adalah fine tuning dengan cara mencoba hyperparameter secara manual, sampai menemukan kombinasi dari nilai hyperparameter yang memberikan kinerja terbaik.

D. Menggunakan Model Machine Learning untuk Memprediksi Test Set

Model machine learning yang memberikan kinerja terbaik adalah model yang memberikan nilai F1 score tertinggi. Selanjutnya, analisis model machine learning terbaik akan menggunakan model global model-agnostic dan local model-agnostic.

IV. IMPLEMENTASI

A. Visualisasi Dataset

Dalam tahap visualisasi dataset, langkah awal yang dilakukan adalah memuat dataset agar dapat diakses. Sebelum memuat dataset, modul yang dibutuhkan seperti pandas, matplotlib, dan drive perlu diimpor terlebih dahulu untuk menjamin kelancaran proses analisis data.

```
import pandas as pd
import matplotlib.pyplot as plt

from google.colab import drive

drive.mount('/content/drive')
```

Kode 1 Melakukan Impor Modul pandas, matplotlib, dan drive

B. Memuat Dataset

Pada modul pandas, peneliti dapat memuat dataset dengan menggunakan fungsi pd.read_csv(). Fungsi ini memungkinkan peneliti untuk membaca file dataset dan memuatnya ke dalam sebuah dataframe. Selanjutnya, untuk memudahkan peneliti dalam memahami struktur data, dapat digunakan fungsi df.head(). Fungsi ini akan menampilkan lima baris pertama dari dataset sehingga peneliti dapat memeriksa struktur dan format data dengan lebih jelas.

```
df = pd.read_csv('DatasetTA-1.csv')
df.head()
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	P.
0	20000	2	2	1	24	2	2	-1	-1	
1	120000	2	2	2	26	-1	2	0	0	
2	90000	2	2	2	34	0	0	0	0	
3	50000	2	2	1	37	0	0	0	0	
4	50000	1	2	1	57	-1	0	-1	0	

5 rows × 24 columns

Kode 2 Melakukan Load Dataset dan Menampilkan Lima Baris Pertama

C. Melakukan Pemisahan Data menjadi Training Set dan Test Set

Pada tahap ini, dilakukan pemisahan dataset menjadi training set dan test set menggunakan train_test_split. Fitur-fitur dataset yang tidak termasuk dalam kolom target disimpan dalam variabel X, sedangkan target disimpan dalam variabel y. Dataset dibagi menjadi training set dan test set dengan ukuran masing-masing 80% dan 20% dari keseluruhan dataset. Metode smote digunakan untuk oversampling pada data pelatihan.

```
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE

X = defaulters.drop('default payment next month', axis=1)
y = defaulters['default payment next month']

smote = SMOTE()
x_smote, y_smote = smote.fit_resample(X,y)

X_train, X_test, y_train, y_test = train_test_split(x_smote, y_smote,
test_size=0.2, random_state=42) X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.1,

random_state=42)
```

Kode 3 Pembagian Dataset Setelah Penerapan SMOTE

Smote adalah sebuah metode yang digunakan dalam analisis data untuk mengatasi ketidakseimbangan kelas, khususnya dalam kasus default kartu kredit. Pada konteks ini, ketidakseimbangan kelas merujuk pada perbedaan jumlah sampel antara kelas mayoritas (non-default) dan kelas minoritas (default). Smote bekerja dengan menciptakan sampel sintetis untuk kelas minoritas dengan cara menggabungkan fitur-fitur dari sampel minoritas yang ada. Dengan menggunakan smote, model yang akan dibangun memiliki data yang lebih seimbang, sehingga membantu meningkatkan kinerja prediksi pada kasus default kartu kredit.

D. Melakukan Fine Tuning pada Setiap Model Machine Learning

Setelah melakukan visualisasi dan pemisahan pada dataset, langkah berikutnya adalah mencari parameter terbaik dari hyperparameter yang telah ditentukan. Dalam penelitian ini, akan diuji beberapa model machine learning seperti logistic regression, k-nearest neighbor, artificial neural networks, decision trees, support vector machines, random forests, dan gradient boosting untuk menemukan kinerja model terbaik yang dapat digunakan.

a. Logistic Regression

Pada model ini, dilakukan tuning hyperparameter menggunakan GridSearchCV. Parameter C dan solver diuji dengan menggunakan nilai yang telah ditentukan. C adalah parameter regulasi yang digunakan untuk mengontrol kompleksitas model, sedangkan solver adalah parameter yang digunakan untuk menentukan bobot terbaik.

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, f1_score

lr = LogisticRegression()

param_grid = {
    'C': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
}

grid_search = GridSearchCV(lr, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best score: ", grid_search.best_score_)
```

```
Best parameters:  {'C': 9, 'solver': 'newton-cg'}
Best score:  0.7252689521273479
```

Kode 4 Hasil Pencarian Parameter Terbaik pada Model Logistic Regression

Model logistic regression yang sudah di tuning dengan GridSearchCV digunakan untuk melakukan prediksi pada test set. Hasil prediksi tersebut kemudian dicetak dalam bentuk classification report dan F1 score. F1 score digunakan untuk mengukur perfoma model dalam memprediksi test set dengan nilai antara 0 hingga 1.

```
lr_best=LogisticRegression(C=grid_search.best_params_['C'],
                           solver=grid_search.best_params_['solver'])

lr_best.fit(X_train, y_train)
y_pred = lr_best.predict(X_test)

print(classification_report(y_test, y_pred))
print("F1 score: ", f1_score(y_test, y_pred))
```

```
                precision    recall  f1-score   support

             0       0.70      0.75      0.73      4664
             1       0.73      0.69      0.71      4682

      accuracy                           0.72      9346
     macro avg       0.72      0.72      0.72      9346
  weighted avg       0.72      0.72      0.72      9346

  F1 score:  0.7096703054361011
```

Kode 5 Hasil Evaluasi pada Model Logistic Regression

b. K-Nearest Neighbor (KNN)

Model KNN akan mencari parameter terbaik dengan menggunakan GridSearchCV. Parameter yang akan diuji meliputi jumlah tetangga terdekat (n_neighbors), bobot yang diberikan pada tetangga (weights), algoritma yang digunakan untuk menghitung tetangga terdekat (algorithm), dan parameter jarak (p). GridSearchCV akan mencoba kombinasi parameter yang berbeda-beda untuk menemukan parameter terbaik dengan *cross validation* (CV) sebesar 5. Setelah parameter terbaik ditemukan, hasilnya dicetak dengan mencetak parameter terbaik dan skor terbaik.

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

param_grid = {
    'n_neighbors': list(range(1, 11)),
    'weights': ['uniform', 'distance'],
    'algorithm': ['ball_tree', 'kd_tree', 'brute'],
    'p': [1, 2]
}

grid_search = GridSearchCV(knn, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

```
  Best parameters: {'algorithm': 'brute', 'n_neighbors': 1, 'p':
  1, 'weights': 'uniform'}
  Best Score:  0.8073404490016933
```

Kode 6 Hasil Pencarian Parameter Terbaik pada Model KNN

Selanjutnya, melakukan fitting pada model KNN dengan parameter terbaik yang didapatkan dari GridSearchCV. Setelah melakukan fitting, model tersebut akan digunakan untuk melakukan prediksi pada test set.

```
knn_best = KNeighborsClassifier(
     n_neighbors=grid_search.best_params_['n_neighbors'],
     weights=grid_search.best_params_['weights'],
     algorithm=grid_search.best_params_['algorithm'],
     p=grid_search.best_params_['p']
     )

knn_best.fit(X_train, y_train)
y_pred = knn_best.predict(X_test)

print(classification_report(y_test, y_pred))
print("F1 Score: ", f1_score(y_test, y_pred))
```

```
                precision    recall  f1-score   support

             0       0.91      0.72      0.81      4664
             1       0.77      0.93      0.84      4682

      accuracy                           0.83      9346
     macro avg       0.84      0.83      0.83      9346
  weighted avg       0.84      0.83      0.83      9346

  F1 Score:  0.8437257939581719
```

Kode 7 Hasil Evaluasi pada Model KNN

c. Artificial Neural Networks (ANN)

Parameter yang digunakan pada model ANN adalah hidden_layer_sizes untuk menguji berbagai kombinasi ukuran hidden layer, activation untuk menguji berbagai fungsi aktivasi, dan learning_rate untuk menguji berbagai metode pembelajaran. Model ini juga menggunakan GridSearchCV untuk mencari parameter terbaik dari hyperparameter yang digunakan.

```
from sklearn.neural_network import MLPClassifier

ann = MLPClassifier()

param_grid = {
    'hidden_layer_sizes': [(10,), (20,), (30,), (40,), (50,),
    (60,), (70,), (80,), (90,), (100,)],
    'activation': ['logistic', 'tanh', 'relu'],
    'learning_rate': ['constant', 'invscaling', 'adaptive']}

grid_search = GridSearchCV(ann, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

```
  Best parameters: {'activation': 'relu', 'hidden_layer_sizes': (1
  00,), 'learning_rate': 'invscaling'}
  Best Score: 0.6276828573561025
```

Kode 8 Hasil Pencarian Parameter Terbaik pada Model ANN

Setelah mendapatkan parameter terbaik dari model ANN, parameter terbaik akan dilatih pada keseluruhan training set. Kemudian, melakukan prediksi pada test set dan melakukan evaluasi perfoma menggunakan classification report dan F1 score.

```
ann_best = MLPClassifier(
  hidden_layer_sizes=grid_search.best_params_['hidden_layer_sizes'],
  activation=grid_search.best_params_['activation'],
  learning_rate=grid_search.best_params_['learning_rate'])

ann_best.fit(X_train, y_train)
y_pred = ann_best.predict(X_test)

print(classification_report(y_test, y_pred))
print("F1 Score:", f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.27      0.39      4664
           1       0.55      0.89      0.68      4682

    accuracy                           0.58      9346
   macro avg       0.64      0.58      0.54      9346
weighted avg       0.63      0.58      0.54      9346

F1 Score: 0.682477587612062
```

Kode 9 Hasil Evaluasi pada Model ANN

d. Decision Trees

Pada model decision trees, parameter yang digunakan adalah criterion untuk menguji berbagai metode pengukuran dan splitter untuk menguji berbagai metode pemilihan pemisah pada node. Perhitungan parameter terbaik dan skor terbaik dari hasil tuning dilakukan menggunakan GridSearchCV.

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier()

param_grid = {
    'criterion': ['gini', 'entropy', 'log_loss'],
    'splitter': ['best', 'random']
    }

grid_search = GridSearchCV(dtc, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

```
Best parameters:  {'criterion': 'entropy', 'splitter': 'best'}
Best Score:  0.7487293975825848
```

Kode 10 Hasil Pencarian Parameter Terbaik pada Model Decision Trees

Proses fitting dilakukan dengan metode fit() yang akan dilatih dengan training test. Selanjutnya, melakukan prediksi pada test set dengan metode predict(). Hasil model akan ditampilkan dalam bentuk classification report dan F1 score.

```
dtc_best = DecisionTreeClassifier(
  criterion=grid_search.best_params_['criterion'],
  splitter=grid_search.best_params_['splitter'])

dtc_best.fit(X_train, y_train)
y_pred = dtc_best.predict(X_test)

print(classification_report(y_test, y_pred))
print("F1 Score: ", f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.77      0.75      0.76      4664
           1       0.75      0.77      0.76      4682

    accuracy                           0.76      9346
   macro avg       0.76      0.76      0.76      9346
weighted avg       0.76      0.76      0.76      9346

F1 Score:  0.7637015177065768
```

Kode 11 Hasil Evaluasi pada Model Decision Trees

e. Support Vector Machines (SVM)

Model SVM melakukan proses tuning parameter menggunakan GridSearchCV untuk mendapatkan kombinasi parameter terbaik yang menghasilkan perfoma model yang optimal.

```
from sklearn.svm import SVC

param_grid = {
              'kernel': ['linear', 'poly', 'rbf', 'sigmoid']
             }

svc = SVC()

grid_search = GridSearchCV(svc, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

```
Best parameters:  {'kernel': 'rbf'}
Best Score:  0.8201481481481482
```

Kode 12 Hasil Pencarian Parameter Terbaik pada Model SVM

Setelah tuning parameter selesai dilakukan, hal yang harus dilakukan selanjutnya adalah fitting pada training set. Kemudian, menggunakan model yang telah di-fit untuk melakukan prediksi pada test set. Hasil model akan ditampilkan dalam bentuk classification report dan F1 score.

```
svc_best = SVC(kernel=grid_search.best_params_['kernel'])

svc_best.fit(X_train, y_train)
y_pred = svc_best.predict(X_test)

print(classification_report(y_test, y_pred))
print("F1 Score: ", f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.95      0.89      2341
           1       0.68      0.35      0.46       659

    accuracy                           0.82      3000
   macro avg       0.76      0.65      0.68      3000
weighted avg       0.80      0.82      0.80      3000

F1 Score:  0.4598393574297189
```

Kode 13 Hasil Evaluasi pada Model SVM

f. Random Forest

Parameter yang akan dicari pada model random forest adalah n_estimators untuk menguji berbagai jumlah pohon dalam random forest, max_depth untuk menguji berbagai kedalaman maksimum pohon, min_samples_split untuk menguji berbagai jumlah minimum sampel untuk split, dan min_samples_leaf untuk menguji berbagai jumlah minimum sampel untuk leaf node. Model ini akan menggunakan GridSearchCV untuk secara otomatis mencari parameter terbaik.

```
from sklearn.ensemble import RandomForestClassifier

param_grid = {
          'n_estimators': [50, 100, 200],
          'max_depth': [2, 4, 6, 8],
          'min_samples_split': [2, 4, 6],
          'min_samples_leaf': [1, 2, 4]
          }

rfc = RandomForestClassifier()

grid_search = GridSearchCV(rfc, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

```
Best parameters:  {'max_depth': 8, 'min_samples_leaf': 1, 'min_s
amples_split': 4, 'n_estimators': 200}
Best Score:  0.7745438535138935
```

Kode 14 Hasil Pencarian Parameter Terbaik pada Model Random Forest

Setelah mendapatkan parameter terbaik dari model random forest, parameter terbaik dari model akan dilatih pada keseluruhan training set dan melakukan prediksi pada test set. Selanjutnya, mencetak laporan dengan classification report dan F1 score dari model random forest.

```
rfc_best = RandomForestClassifier(
      n_estimators=grid_search.best_params_['n_estimators'],
      max_depth=grid_search.best_params_['max_depth'],
      min_samples_split=grid_search.best_params_[
      'min_samples_split'],
      min_samples_leaf=grid_search.best_params_[
      'min_samples_leaf']
      )

rfc_best.fit(X_train, y_train)
y_pred = rfc_best.predict(X_test)

print(classification_report(y_test, y_pred))
print("F1 score: ", f1_score(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.74      0.83      0.78      4664
           1       0.81      0.71      0.75      4682

    accuracy                           0.77      9346
   macro avg       0.77      0.77      0.77      9346
weighted avg       0.77      0.77      0.77      9346

F1 score:  0.7534947153085579
```

Kode 15 Hasil Evaluasi pada Model Random Forest

g. Gradient Boosting

Parameter yang akan dicari pada model gradient boosting adalah n_estimators untuk menguji berbagai jumlah pohon dalam gradient boosting, learning_rate untuk menguji berbagai tingkat learning rate, max_depth untuk menguji berbagai kedalaman maksimum pohon, min_samples_split untuk menguji berbagai jumlah minimum sampel untuk split, min_samples_leaf untuk menguji berbagai jumlah minimum sampel untuk leaf node. Model ini akan menggunakan GridSearchCV untuk mencari parameter terbaik.

```
from sklearn.ensemble import GradientBoostingClassifier

param_grid = {
              'n_estimators': [50, 100, 200],
              'learning_rate': [0.01, 0.1, 1],
              'max_depth': [2, 4, 6],
              'min_samples_split': [2, 4, 6],
              'min_samples_leaf': [1, 2, 4]
             }

gbc = GradientBoostingClassifier()
grid_search = GridSearchCV(gbc, param_grid=param_grid, cv=5)
grid_search.fit(X_train, y_train)

print("Best parameters: ", grid_search.best_params_)
print("Best Score: ", grid_search.best_score_)
```

```
Best parameters:  {'learning_rate': 0.1, 'max_depth': 6, 'min_sa
mples_leaf': 1, 'min_samples_split': 6, 'n_estimators': 200}
Best Score:  0.8249692493756312
```

Kode 16 Hasil Pencarian Parameter Terbaik pada Model Gradient Boosting

Setelah mendapatkan parameter terbaik dari model *gradient boosting*, parameter terbaik akan dilatih pada keseluruhan *training set*. Kemudian, melakukan prediksi pada *test set* dan menghasilkan laporan klasifikasi beserta nilai F1 score.

```
gbc_best = GradientBoostingClassifier(
      n_estimators=grid_search.best_params_['n_estimators'],
      learning_rate=grid_search.best_params_['learning_rate'],
      max_depth=grid_search.best_params_['max_depth'],
      min_samples_split=grid_search.best_params_[
      'min_samples_split'],
      min_samples_leaf=grid_search.best_params_[
      'min_samples_leaf']
      )

gbc_best.fit(X_train, y_train)
y_pred = gbc_best.predict(X_test)

print(classification_report(y_test, y_pred))
print("F1 Score: ", f1_score(y_test, y_pred))
```

```
                precision    recall  f1-score   support

             0       0.80      0.86      0.83      4664
             1       0.85      0.79      0.82      4682

      accuracy                           0.82      9346
     macro avg       0.83      0.82      0.82      9346
  weighted avg       0.83      0.82      0.82      9346

  F1 Score:  0.8180204172214824
```

Kode 17 Hasil Evaluasi Kinerja Model Gradient Boosting

E. Membuat Plot untuk Meihat Kontribusi Fitur Model Terbaik

Tabel 1 Hyperparameter dan Nilai F_1 score dari Setiap Model

Model	Hyperparameter Terbaik	F_1 *score*
Logistic regression	'C': 9, 'solver': 'newton-cg'	70%
K-Nearest neighbor	'algorithm': 'brute', 'n_neighbors': 1, 'p': 1, 'weights': 'uniform'	**84%**
Artificial neural network	'activation': 'relu', 'hidden_layer_sizes': (100), 'learning_rate': 'invscaling'	68%
Decision trees	'criterion': 'entropy', 'splitter': 'best'	75%
Support vector machines	'kernel': 'rbf'	46%
Random forest	'max_depth': 8, 'min_samples_leaf': 1, 'min_samples_split': 4, 'n_estimators': 200	75%
Gradient Boosting	'n_estimators': 200, 'learning_rate': 0.1, 'max_depth': 6, 'min_samples_leaf': 1, 'min_samples_split': 6	**82%**

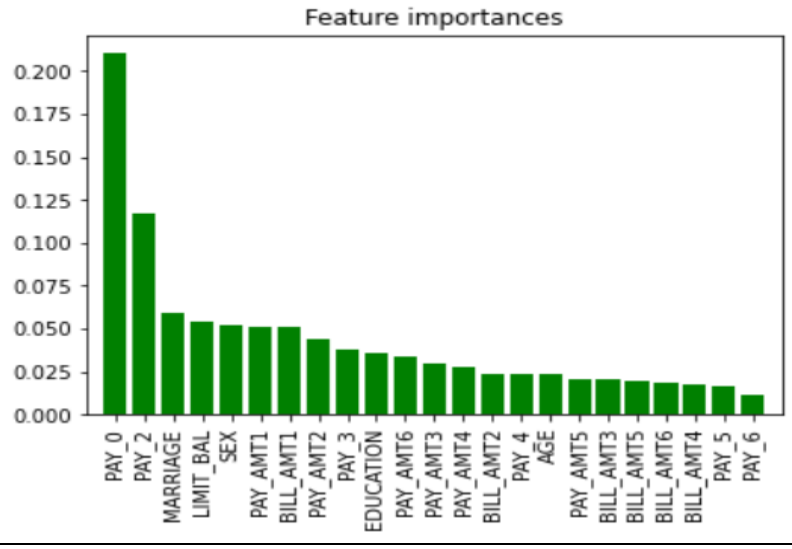
Pada tahap membuat model-model machine learning sebelumnya, terdapat model dengan F1 score terbaik yaitu KNN dan gradient boosting. Plot yang akan dibuat bertujuan untuk menunjukkan nilai kontribusi dari masing-masing fitur.

Pertama, untuk menghitung tingkat pentingnya setiap fitur pada model adalah dengan menggunakan fungsi feature_importance_. Fitur-fitur tersebut akan ditampilkan secara berurutan berdasarkan tingkat pentingnya fitur menggunakan np.argsort dan mengubah urutannya secara terbalik dengan [::-1]. Kemudian, tingkat pentingnya setiap fitur akan direpresentasikan dalam bentuk diagram batang menggunakan plt.bar.

```
import matplotlib.pyplot as plt
import numpy as np

importances = gbc_best.feature_importances_
indices = np.argsort(importances)[::-1]

plt.figure()
plt.title("Feature importances")
plt.bar(range(X_train.shape[1]), importances[indices], color="green")
plt.xticks(range(X_train.shape[1]), X_train.columns[indices], rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.show()
```



Kode 3 Feature Importance pada Model Gradient Boosting

Pada saat mencari nilai F1 score dari model gradient boosting, hasil yang diperoleh adalah 82%. Hasil yang didapatkan mendekati model terbaik yaitu KNN. Fitur yang paling penting pada model gradient boosting adalah PAY_0 dengan nilai kontribusi lebih dari 0,200.

V. KESIMPULAN DAN SARAN

A. Simpulan

Berdasarkan hasil penelitian ini, model KNN menunjukkan kinerja terbaik dibandingkan dengan model lainnya. Meskipun gradient boosting juga memberikan kinerja yang baik, F1 score yang dihasilkan lebih kecil dibandingkan dengan KNN.

Dari hasil plot feature importance, pay_0 adalah fitur yang paling berkontribusi dengan nilai 0,200. Pay_0 merepresentasikan riwayat pembayaran kredit pada bulan sebelumnya dan menjadi fitur paling penting karena memberikan informasi tentang perilaku pembayaran kredit dari customer. Hasil penelitian menunjukkan bahwa riwayat pembayaran yang buruk atau tidak teratur pada pay_0 dapat menjadi indikasi kuat bahwa customer akan mengalami kesulitan dalam membayar kredit di masa depan.

B. Saran

Pada penelitian berikutnya, sebaiknya dilakukan analisis lebih mendalam untuk meningkatkan perfoma model, terutama dalam meningkatkan nilai F1 score. Kemudian, dipertimbangkan untuk menambahkan fitur atau variabel lainnya yang mungkin dapat meningkatkan performa model, seperti durasi pembayaran, jumlah pembayaran minimum, dan sebagainya. Selain itu, disarankan menganalisis lebih mendalam mengenai faktor-faktor lain yang mempengaruhi default kartu kredit, seperti faktor ekonomi, sosial, atau demografi.

DAFTAR PUSTAKA

[1] A. Grigorev, Machine Learning Bookcamp, Shelter Island: Manning Publications, 2021.

[2] T. M. Mitchell, Naive Bayes and Logistic Regression, 2017.

[3] S. Raschka and V. Mirjalili, Python Machine Learning, Birmingham: Packt Publishing, 2019.

[4] A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, Canada: O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2019.

[5] R. Garreta and G. Moncecchi, Learning scikit-learn: Machine Learning in Python, BIRMINGHAM - MUMBAI: Packt Publishing, 2013.

[6] C. Molnar, Interpretable Machine Learning, Yvonne Doinel, 2022.

[7] s.-l. developers, "sklearn.linear_model.LogisticRegression," 2007. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.

[8] s.-l. developers, "sklearn.neural_network.MLPClassifier," 2007. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.

[9] s.-l. developers, "sklearn.neighbors.KNeighborsClassifier," 2007. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html.

[10] s.-l. developers, "sklearn.tree.DecisionTreeClassifier," 2007. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[11] s.-l. developers, "sklearn.svm.SVC," 2007. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[12] s.-l. developers, "sklearn.ensemble.RandomForestClassifier," 2007. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier.

[13] s.-l. developers, "sklearn.ensemble.GradientBoostingClassifier," 2007. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier.