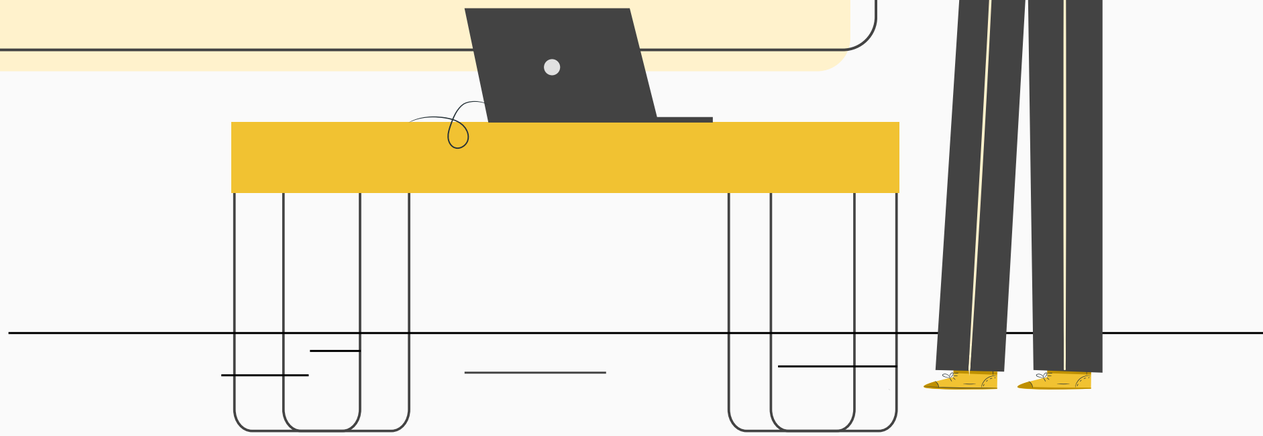
PREDIKSI ELIGIBILITAS PEMOHON/KANDIDAT PINJAMAN RUMAH MENGGUNAKAN DECISION TREE DAN RANDOM FOREST

Tugas Akhir Sains Data Kelompok 2

- Jessica Natama Napitupulu (2106726150)
- Nabila Parahita (2106726094)
- Ghina Rahadatul ‘aisy (2206024606)
- Kezia Declare Taniyara (2206025312)
- Muhammad Fasya Syaifullah (2206025496)

ABSTRAK

Tugas akhir ini bertujuan untuk mengetahui model klasifikasi menggunakan *Decision Tree* dan *Random Forest* berdasarkan data yang tersedia, serta akan diprediksi hasil terhadap persetujuan pinjaman rumah yang diperoleh dari model tersebut. Tugas ini dibantu oleh *problem statement* dan data dari kaggle. Kemudian, akan dibuat rancangan program yang diimplementasikan menggunakan Google Collaboratory sehingga dapat memenuhi tujuan dan manfaat dari tugas ini. Melalui proses analisis data dan pembangunan model menggunakan algoritma, diharapkan dapat menentukan penentuan kelayakan pinjaman dengan lebih efisien dan akurat.



LATAR BELAKANG

Memiliki rumah merupakan kebutuhan dasar manusia dan pinjaman rumah merupakan salah satu jenis pinjaman terbesar dan paling umum di seluruh dunia yang berkontribusi pada keuangan dan perekonomian suatu negara. Banyak permintaan pinjaman rumah, tetapi tidak semua permintaan akan disetujui. Oleh karena itu, pada tugas ini akan memecahkan masalah untuk menentukan persetujuan pinjaman rumah dengan menggunakan metode *Decision Tree* dan *Random Forest,* dan kemudian akan dibandingkan hasil prediksi dari metode tersebut. Banyaknya data yang tersedia dan dapat diakses secara publik tentang pinjaman rumah memudahkan kami dalam mengerjakan tugas ini.

TUJUAN PENULISAN

01

Mengetahui model klasifikasi menggunakan Decision Tree berdasarkan data yang tersedia

02

Mengetahui model klasifikasi menggunakan Random Forest berdasarkan data yang tersedia

03

Mengetahui hasil prediksi terhadap persetujuan pinjaman rumah yang diperoleh dari model klasifikasi Decision Tree dan Random Forest

PENJELASAN KASUS

Terdapat sebuah data Home Loan dengan 614 observasi dan 13 fitur, yaitu:

- Loan ID : kode unik peminjam
- Gender : jenis kelamin
- Married : status pernikahan
- Dependents : jumlah tanggungan
- Education : pendidikan terakhir
- Self Employed : wiraswasta
- Applicant Income : pemasukan peminjam
- Co Applicant Income : pemasukan pasangan peminjam
- Loan amount: jumlah pinjaman
- Loan Amount Term : jumlah waktu peminjaman
- Credit History : riwayat kredit
- Property Area : wilayah properti
- Loan Status : status pinjaman

Loan Status akan dipilih sebagai variabel target. Data tersebut didapat dari calon kandidat yang mengisi formulir. Tujuan utama dari kasus ini adalah untuk mengidentifikasi kandidat yang memenuhi syarat untuk pinjaman rumah. Akan digunakan dua metode yaitu Decision Tree dan Random forest untuk memprediksi kelayakan pinjaman rumah.

DECISION TREE

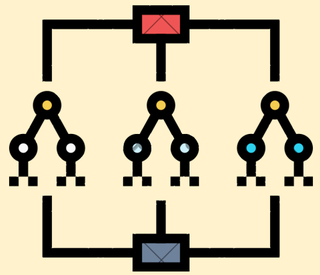
Decision Tree adalah model prediksi yang menggunakan suatu flowchart dengan struktur hierarki seperti pohon yang digunakan untuk membuat keputusan. Tiap node dalam Decision Tree merepresentasikan keputusan berdasarkan fitur tertentu. Berikut adalah perhitungan matematis dalam klasifikasi decision tree:

1. Menghitung nilai entropi dan gini dari setiap atribut: $$Entropy(S) = \sum_{i \in categorical}^{n} - p_i \cdot \log_2(p_i)$$
2. Menghitung nilai dari informasi: $$Gain(S, A) = Entropy(S) - \sum_{v \in feature}^{n} \frac{|S_v|}{|S|} \cdot Entropy(S_v)$$ (Sv adalah fitur bersesuaian)
3. Menghitung nilai informasi terpisah: $$SplitInfo(D) = - \sum_{j \in categorical}^{n} \frac{|D_1|}{D} \cdot \log_2(\frac{|D_j|}{|D|})$$
4. Menghitung nilai rasio Informasi: $$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

Root Node pada decision tree adalah atribut dengan rasio informasi terbesar. Setelah mendapat root node, akan dilakukan lagi penentuan child node yang akan dianggap sebagai root node yang baru tanpa menghitung atribut yang sudah terpilih sebelumnya. Atribut dengan rasio informasi tertinggi berikutnya akan dibuat cabang dari decision tree tersebut.

RANDOM FOREST

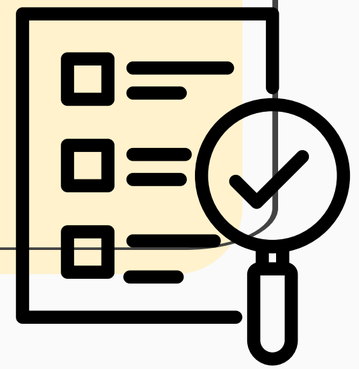
Algoritma Random Forest adalah algoritma yang menggabungkan hasil dari beberapa decision tree untuk mencapai satu hasil. Alasannya adalah karena decision tree sensitif terhadap variansi dari data yang dimiliki, alhasil model akan gagal untuk diperumum. Random forest memiliki struktur yang mirip dengan decision tree, tetapi karena beberapa decision tree yang tidak berkorelasi akan bekerja lebih baik sebagai kelompok dibandingkan individu, maka decision tree tersebut akan menjadi subtree. Hasil prediksi Random Forest didapatkan dari hasil terbanyak (voting) dari setiap decision tree yaitu:



$$l(y) = argmax_c(\sum_{i=1}^{n} lh_i(y) = c)$$

PROSES PENGOLAHAN DATA

1. Pengecekan data mentah:
   - menentukan variabel target yaitu Loan_Status
   - memperhatikan jumlah variabel kategorik dan numerik
   - mengecek entri yang hilang
2. Evaluasi data:
   - menghilangkan fitur yang tidak berpengaruh yaitu ‘Loan_ID’
   - penghapusan observasi jika ada lebih dari 3 entri kosong
   - binning fitur numerik yang seharusnya kategorik yaitu Credit_History dan Loan_Amount_Term
   - imputasi data dengan metode terbaik yaitu mean untuk fitur numerik dan modus untuk fitur kategorik
3. Feature engineering:
   - mengubah variabel kategorik menjadi numerik (integer)
   - membagi data menjadi data training dan data testing dengan perbandingan 80:20



PENGECEKAN DATA MENTAH

Diberikan dataset **loan_sanction,** di sini ingin diprediksi apakah seseorang masuk dalam kategori orang yang dapat diberikan pinjaman atau tidak. Variabel target yang digunakan adalah 'Loan_Status' yaitu status pinjaman.
Berikut informasi yang dapat diambil dari tabel tersebut.

1. Terdapat 6 variabel independent bertipe **kategorik** dan semuanya termasuk dalam data nominal. Variabel tersebut adalah Gender, Married, Dependents, Education, Self_Employed, dan Property_Area.
2. Terdapat 5 variabel independent bertipe **numerik** dan semuanya diskrit. Variabel tersebut adalah ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term, dan Credit_History.
3. Fitur **Loan_ID tidak dapat digunakan** karena fitur tersebut hanya memberikan informasi identitas peminjam yang pasti tidak berpengaruh terhadap status peminjaman.

```
X.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Gender             614 non-null    int64  
 1   Married            614 non-null    int64  
 2   Dependents         614 non-null    int64  
 3   Education          614 non-null    int64  
 4   Self_Employed      614 non-null    int64  
 5   ApplicantIncome    614 non-null    int64  
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         614 non-null    float64
 8   Loan_Amount_Term   614 non-null    float64
 9   Credit_History     614 non-null    float64
 10  Property_Area      614 non-null    int64  
 11  Total_Income       614 non-null    float64
dtypes: float64(5), int64(7)
memory usage: 57.7 KB
```

CEK ENTRI YANG HILANG

Check for Missing Values

```
[ ] print(f"Ukuran data adalah: {df.shape}")
    print("="*75)
    print("Entri data yang hilang: ")
    print(df.isna().sum())
    print("="*75)
    print("Presentase data yang hilang: ")
    print(df.isnull().mean())
```

```
Ukuran data adalah: (614, 13)
===========================================================================
Entri data yang hilang:
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

```
===========================================
Presentase data yang hilang:
Loan_ID               0.000000
Gender                0.021173
Married               0.004886
Dependents            0.024430
Education             0.000000
Self_Employed         0.052117
ApplicantIncome       0.000000
CoapplicantIncome     0.000000
LoanAmount            0.035831
Loan_Amount_Term      0.022801
Credit_History        0.081433
Property_Area         0.000000
Loan_Status           0.000000
dtype: float64
```

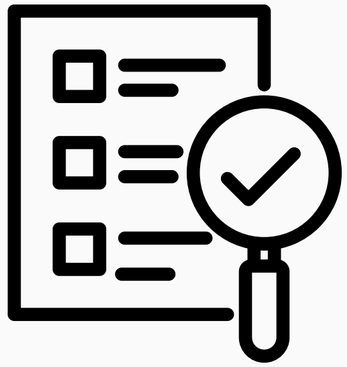
Berdasarkan informasi tersebut, terdapat 6 fitur yang memiliki entri data yang kosong. Namun, persentase dari entri hilang tersebut masih di bawah 9 persen. Karenanya masih dapat dilakukan imputasi untuk dataset tersebut.

```
[ ]  df = df.drop(columns=['Loan_ID'])
```

Fitur ‘Loan_ID’ akan dihilangkan sepenuhnya karena tidak berpengaruh pada variabel target.

```
[ ]  missing_counts = df.isna().sum(axis=1)
     df_empty_3 = df[missing_counts > 3]
     df_empty_3
```

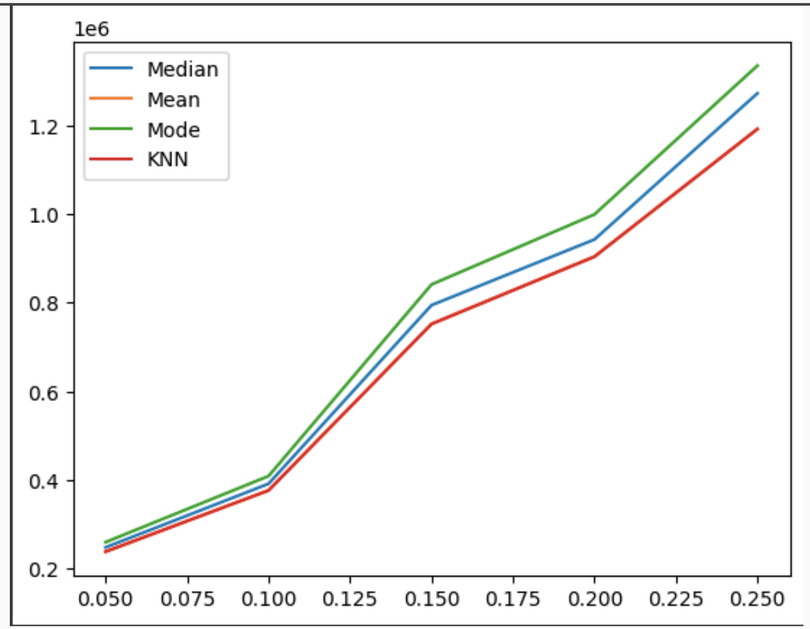
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term



Cek apakah ada observasi pada data frame yang memiliki entri kosong lebih dari 3. Jika ada dapat dipertimbangkan untuk dibuang sepenuhnya.
Karena tidak ada observasi pada data yang memiliki lebih dari 3 entri yang hilang maka tidak perlu adanya penghapusan baris pada dataframe.

IMPUTASI DATA

```
[ ] from sklearn.impute import SimpleImputer
    from sklearn.impute import KNNImputer
    median_imputer = SimpleImputer(missing_values=np.nan, strategy='median')
    mean_imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
    mode_imputer = SimpleImputer(missing_values=np.nan, strategy="most_frequent")
    KNN_imputer = KNNImputer(missing_values=np.nan, n_neighbors=3)
    list_persen = [0.05, 0.10, 0.15, 0.20, 0.25]
    results_1 = compare_imputation(df_num, [median_imputer, mean_imputer, mode_imputer, KNN_imputer], list_persen)
    results_2 = compare_imputation(df_num, [median_imputer, mean_imputer, mode_imputer, KNN_imputer], list_persen)
    results_3 = compare_imputation(df_num, [median_imputer, mean_imputer, mode_imputer, KNN_imputer], list_persen)
    results_4 = compare_imputation(df_num, [median_imputer, mean_imputer, mode_imputer, KNN_imputer], list_persen)
    results_5 = compare_imputation(df_num, [median_imputer, mean_imputer, mode_imputer, KNN_imputer], list_persen)

    results = (np.array(results_1) + np.array(results_2) + np.array(results_3) + np.array(results_4) + np.array(results_5))

    plt.plot(list_persen, results[0])
    plt.plot(list_persen, results[1])
    plt.plot(list_persen, results[2])
    plt.plot(list_persen, results[3])

    plt.legend(["Median", "Mean", "Mode", "KNN"])
    plt.show()
```



Sebelum melakukan imputasi, akan dilakukan perbandingan metode imputasi untuk mencari metode mana yang terbaik.

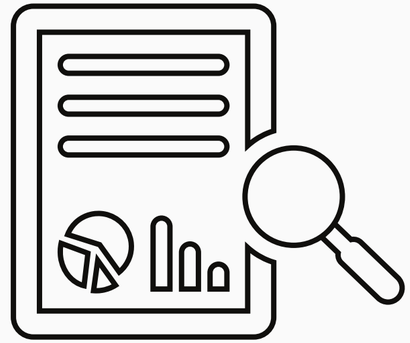
Untuk fitur numerik, akan dilakukan strategi imputasi mean, median, modus, dan KNN.

Imputasi oleh KNN dan Mean menunjukkan hasil yang sama dengan memperoleh error terkecil. Untuk simplisitas akan digunakan mean sebagai imputasi entri data yang hilang.

IMPUTASI DATA

```
[ ]  df[['LoanAmount']] = mean_imputer.fit_transform(df[['LoanAmount']])
     # imputasi selesai
```

```
[ ]  mode_impute = SimpleImputer(strategy='most_frequent')
     df[['Credit_History']] = mode_impute.fit_transform(df[['Credit_History']])
     df[['Loan_Amount_Term']] = mode_impute.fit_transform(df[['Loan_Amount_Term']])
     df[['Married']] = mode_impute.fit_transform(df[['Married']])
```



```
[ ]  # Random Sample
     import pandas as pd
     import numpy as np

     def random_sample_imputation(df, columns):
         df_imputed = df.copy()

         for column in columns:
             missing = df_imputed[column].isnull()
             n_missing = missing.sum()

             if n_missing > 0:
                 sampled_values = df_imputed.loc[~missing, column].sample(n=n_missing, replace=True).values
                 df_imputed.loc[missing, column] = sampled_values

         return df_imputed
```

```
[ ]  df = random_sample_imputation(df, ['Gender', 'Self_Employed', 'Dependents'])
```

Untuk fitur kategorik, akan dilakukan random sampel random (untuk fitur ‘Credit_History’, ‘Loan_Amount_Status’, dan ‘Married’) dan modus (untuk fitur ‘Gender’, ‘Self_Employed’, dan ‘Dependents’.

```
[ ] X.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Gender             614 non-null    int64
 1   Married            614 non-null    int64
 2   Dependents         614 non-null    int64
 3   Education          614 non-null    int64
 4   Self_Employed      614 non-null    int64
 5   ApplicantIncome    614 non-null    int64
 6   CoapplicantIncome  614 non-null    float64
 7   LoanAmount         614 non-null    float64
 8   Loan_Amount_Term   614 non-null    float64
 9   Credit_History     614 non-null    float64
 10  Property_Area      614 non-null    int64
 11  Total_Income       614 non-null    float64
dtypes: float64(5), int64(7)
memory usage: 57.7 KB
```

Feature Engineering
Variabel kategorik akan diubah menjadi numerik (integer).

Selanjutnya, data akan dibagi menjadi 20% data testing dan 80% data training.

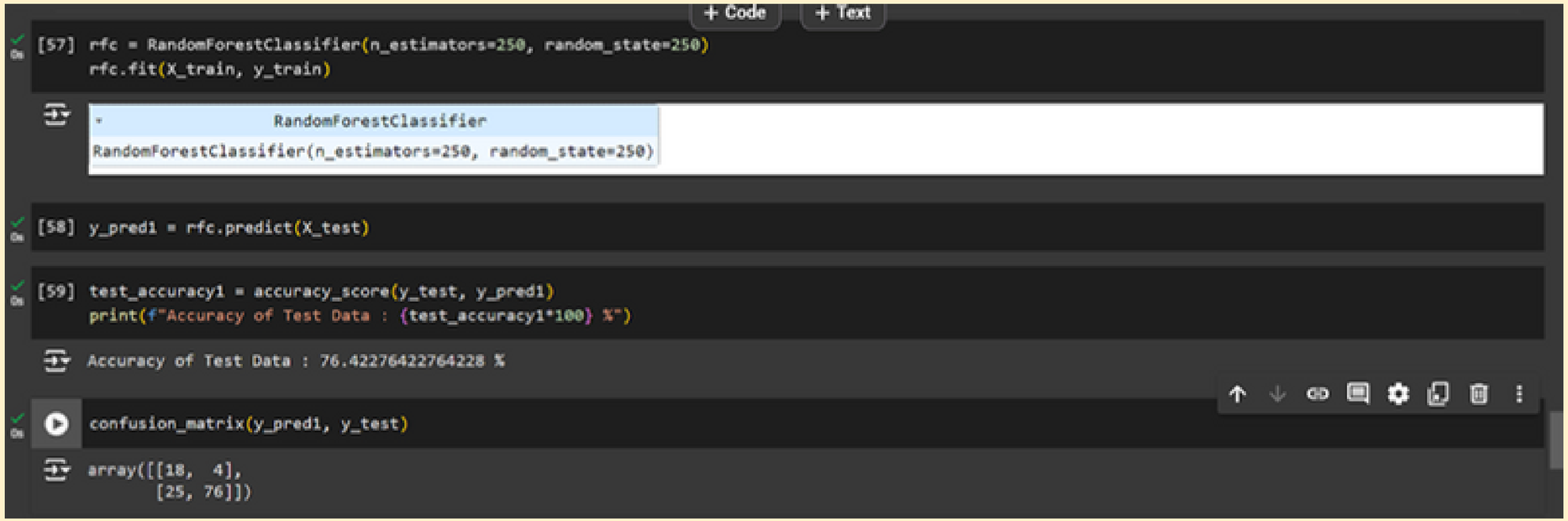
```
[ ] # train-test-split
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

DECISION TREE

```
[53] clf = DecisionTreeClassifier(ccp_alpha=0.01, splitter='best')
     clf.fit(X_train, y_train)

     DecisionTreeClassifier
DecisionTreeClassifier(ccp_alpha=0.01)

[54] y_pred = clf.predict(X_test)
     test_accuracy = accuracy_score(y_test, y_pred)
     print(f"Accuracy of Test Data : {test_accuracy*100} %")

Accuracy of Test Data : 78.86178861788618 %

     confusion_matrix(y_pred, y_test)

array([[18,  1],
       [25, 79]])
```

Dari program tersebut didapatkan output akurasi dari model Decision Tree sebesar 76.86178861788618 % dengan confusion matrix yang digunakan untuk mengevaluasi kinerja model klasifikasi, terdapat 18 prediksi yang benar untuk kelas pertama, 79 prediksi yang benar untuk kelas kedua, 1 prediksi yang salah untuk kelas pertama, dan 25 prediksi yang salah untuk kelas kedua.

RANDOM FOREST



Dari program tersebut didapatkan output akurasi dari model Random Forest sebesar 76.42276422764228 % dengan confusion matrix yang digunakan untuk mengevaluasi kinerja model klasifikasi, terdapat 18 prediksi yang benar untuk kelas pertama, 76 prediksi yang benar untuk kelas kedua, 4 prediksi yang salah untuk kelas pertama, dan 25 prediksi yang salah untuk kelas kedua.

KESIMPULAN

- Telah dipelajari penggunaan model klasifikasi Decision Tree dan Random Forest berdasarkan data yang tersedia, mengetahui hasil prediksi terhadap persetujuan pinjaman rumah, serta membandingkan akurasi kedua model
- Data diimplementasikan menggunakan google colaboratory dengan model Decision Tree dan Random Forest.
- Kedua model memiliki tingkat akurasi yang cukup bagus. Namun, akurasi Decision Tree sedikit lebih tinggi daripada Random Forest.
- Decision Tree dapat menentukan penentuan kelayakan pinjaman dengan lebih efisien dan akurat.

DAFTAR PUSTAKA

1. Helmud, E., Helmud, E., Fitriyani, & Romadiana, P. (2024). *Classification Comparison Performance of Supervised Machine Learning Random Forest and Decision Tree Algorithms Using Confusion Matrix.* Jurnal SISFOKOM (Sistem Informasi dan Komputer), 13(01), 92-97.
2. *Decision Tree*. Diakses dari https://revou.co/kosakata/decision-tree
3. *Random Forest*. Diakses dari https://revou.co/kosakata/random-forest
4. Viswanatha, V., Ramachandra, A. C., Vishwas, K. N., & Adithya, G. (2023). Prediction of Loan Approval in Banks using Machine Learning Approach. International Journal of Engineering and Management Research, 13(4), 7-19.