MEMBUAT MODEL MACHINE LEARNING UNTUK MEMPREDIKSI RESIKO KREDIT PINJAMAN

Created by Mouriverd Laurent

Pendahuluan

- Nama Dataset yang digunakan adalah " loan_data_2007_2014.csv", terdiri atas 466285 baris & 75 kolom.
- Dataset ini berisi data pinjaman dari sebuah perusahaan pemberi pinjaman (multifinance).
- Dataset ini akan di gunakan untuk membuat model machine learning untuk memprediksi resiko kredit pinjaman dengan outcome penilaian resiko kredit dengan 3 kategori: ***Low-risk, Medium-risk*** dan ***High-Risk***, yang dapat digunakan oleh stake-holder sebagai referensi untuk menyetujui atau menolak pengajuan pinjaman.
- Tools yang digunakan: Phyton/ Jupyter Notebook/ Google Colaboratory

Metodology Analysis

Berikut tahap-tahap yang dilakukan di dalam pengembangan model di dalam project ini.

Data Understanding	01
EDA | Data Cleaning	02
EDA |Data Visualization	03
Feature Engineering	04
Build Machine Learning & Model Evaluation	05
Penilaian Resiko Kredit	06
Insight & Recommendation	07

01. Data Understanding

01 Dataset terdiri dari 53 Kolom numerik & 22 kolom kategorik dengan type data integer, float dan object.

02 Tidak ditemukan adanya duplikasi data.

03 Ditemukan banyak missing value , termasuk 17 kolom yang semua barisnya berisi “NaN”.

04 Ditemukan nilai outlier di banyak kolom.

02. EDA | Data Cleaning

1. Drop Column.

Drop Kolom yang memiliki missing value > 40%, yaitu 31 Kolom.

2. Data Imputation.

Dari boxplot masing-masing kolom terlihat banyak nilai outlier , sehingga dilakukan imputasi kolom numerik dengan median dan kolom kategorik dengan modus.

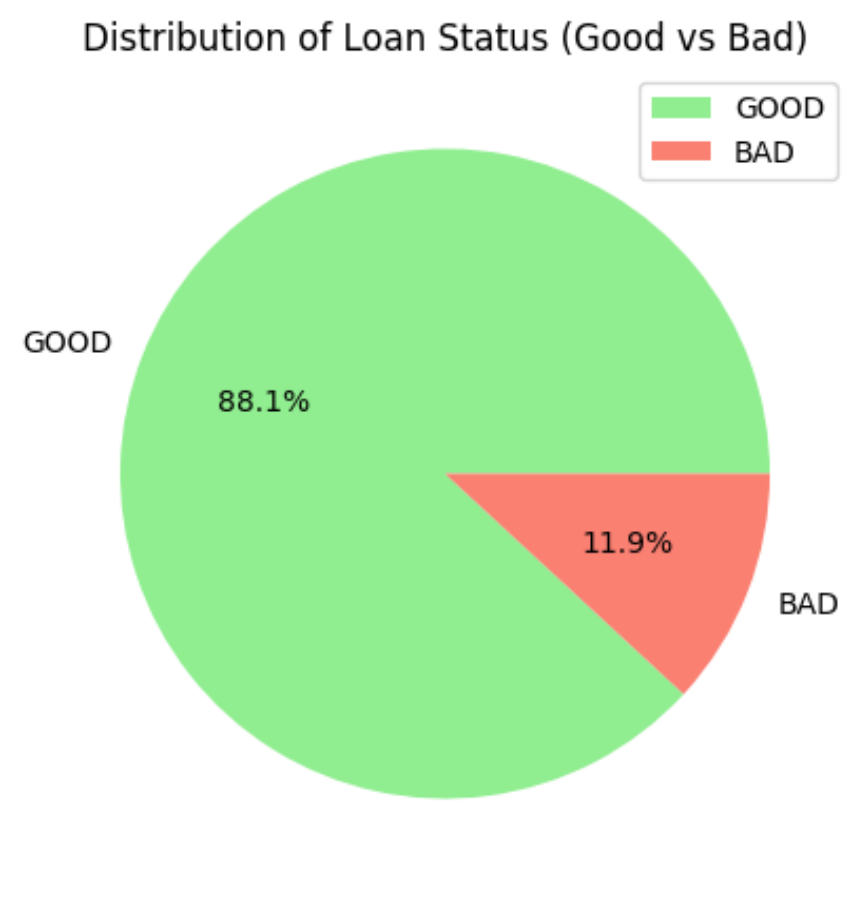
3. Labeling

Memberikan label Good dan Bad untuk menunjukkan status kredit baik atau buruk.

Setelah di lakukkan data cleaning, Kolom tersisa adalah 44 kolom dan sudah tidak di temukan adanya missing value.

03. EDA | Data Visualization

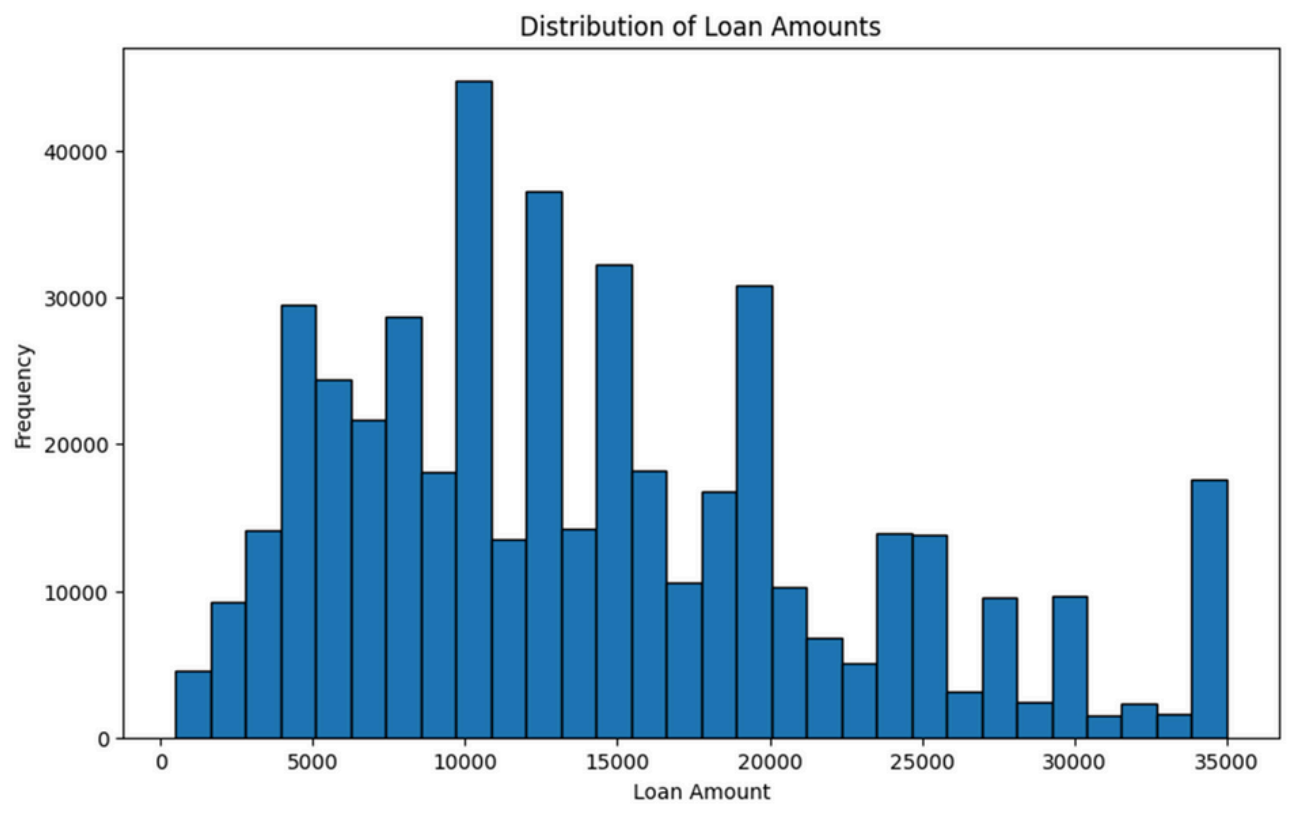
Distribusi Loan Status



Data “Loan Status” terdistribusi **Imbalance,** sehingga perlu dilakukan Teknik SMOTE (Synthetic Minority Over-sampling Technique).

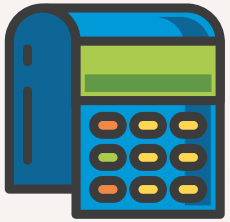
03. EDA | Data Visualization

Distribusi Jumlah Pinjaman



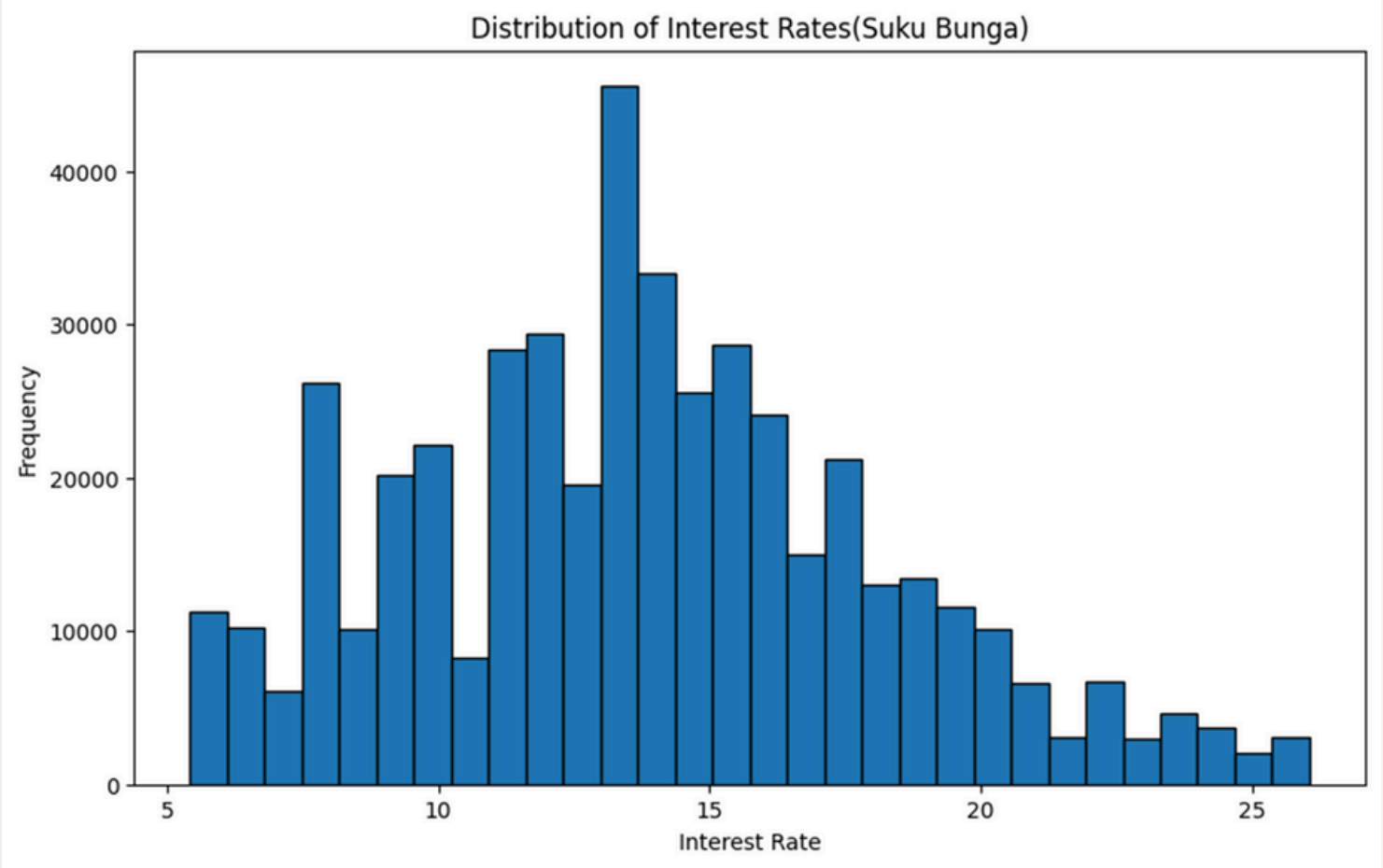
Distribusi Jumlah pinjaman memperlihatkan Skew positif dan sebagian besar pinjaman berada di kisaran 10.000 - 15.000$.

Distribusi data tidak simetris, maka amputasi data missing value di replace dengan median.



03. EDA | Data Visualization

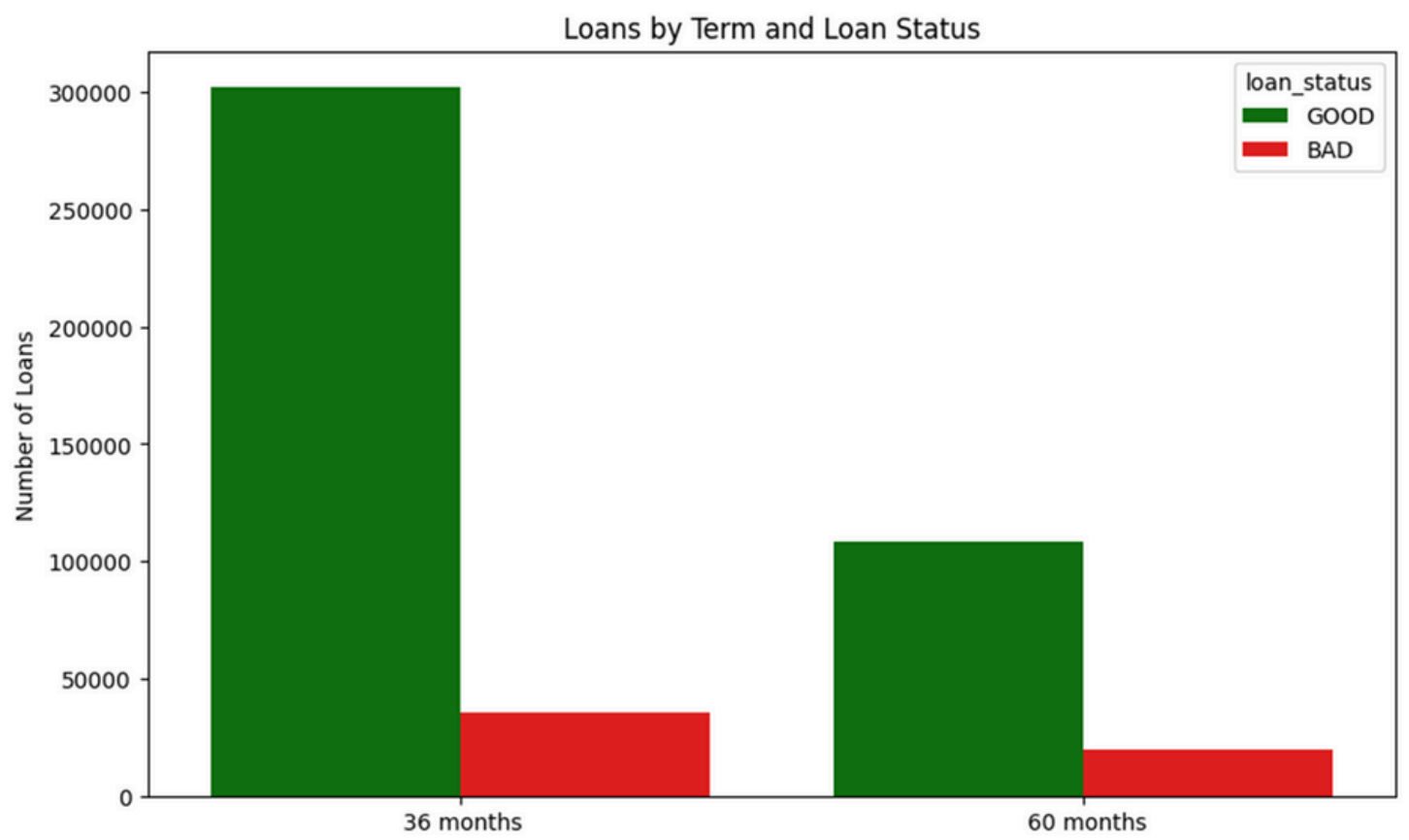
Distribusi Suku Bunga



Suku bunga yang paling umum diberikan kepada peminjam adalah antara 10% sampai 15%. Dan sangat sedikit yang tertarik dengan pinjaman dengan suku bunga tinggi.

03. EDA | Data Visualization

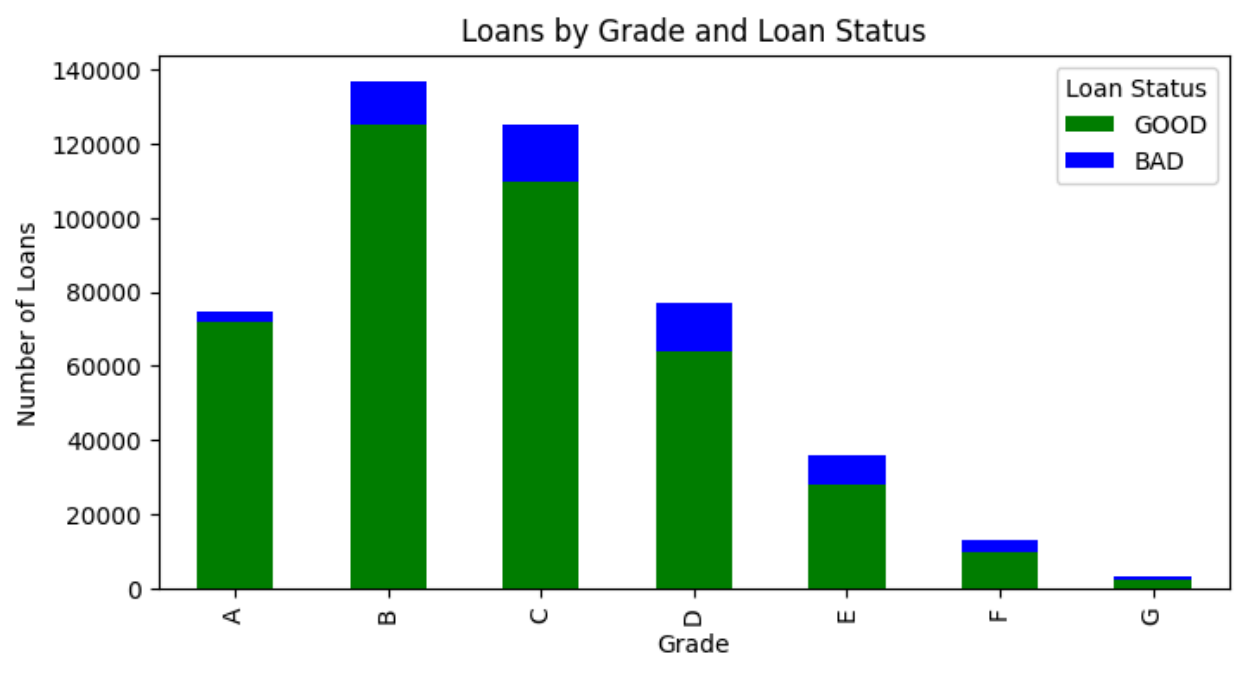
Distribusi Jumlah Pinjaman
(berdasarkan Jangka waktu cicilan and Status Pinjaman)

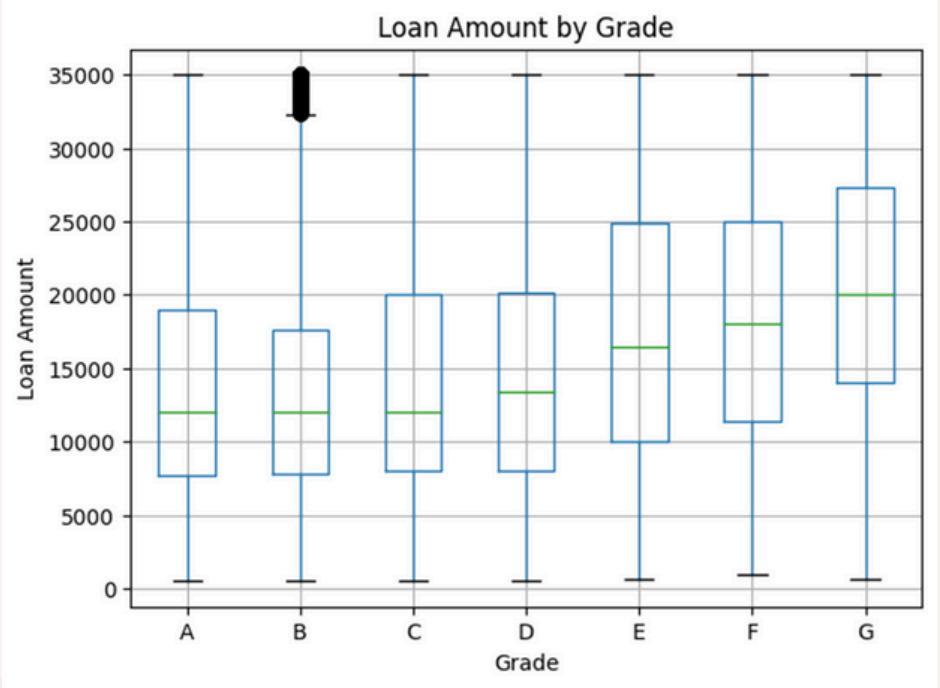


- Sebagian besar peminjam lebih memilih jangka waktu yang lebih pendek, yaitu 36 bulan, menunjukkan bahwa peminjam mampu melunasi pinjaman dalam jangka waktu yang lebih pendek.
- Mayoritas pinjaman berhasil dilunasi oleh peminjam.

03. EDA | Data Visualization

Distribusi Jumlah Pinjaman
(berdasarkan tingkat resiko and Status Pinjaman)

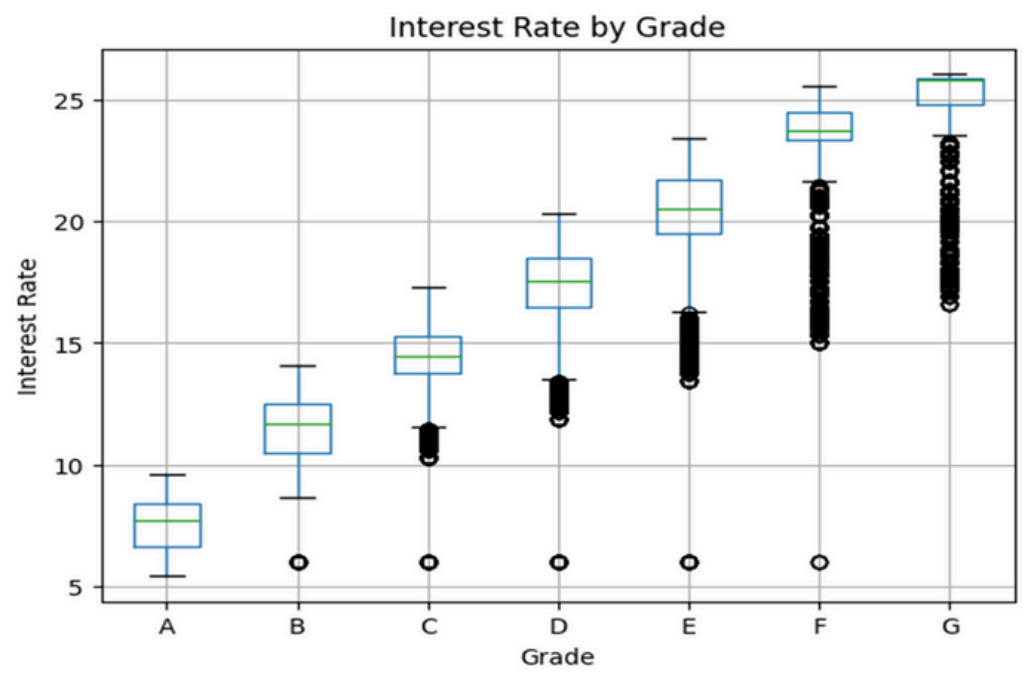




- Grade menunjukkan tingkat risiko peminjam. Semakin besar jumlah pinjaman, maka akan semakin besar pula tingkat resikonya.
- Pada Grade B, ada anomali pinjaman sangat besar, namun dengan tingkat resiko yang lebih kecil.

03. EDA | Data Visualization

Distribusi Suku Bunga
(berdasarkan Tingkat Resiko)



Suku bunga meningkat secara konsisten dari grade A ke grade G, menunjukkan bahwa peminjam dengan risiko lebih tinggi dikenakan suku bunga yang lebih tinggi sebagai kompensasi atas risiko tambahan.

04. Feature Engineering

Drop fitur yang tidak diperlukan

Konversi kolom kategorik ke kolom numerik

Label Encoding ‘loan_status’ sebagai target variable into 0 and 1

Feature engineering menghasilkan 21 kolom dengan tipe numerik, selanjutnya di split menjadi 20 feature dan 1 target variable yang akan digunakan di dalam pemodelan.

05. Build Machine Learning & Model Evaluation

01 Split data for features and target variable

02 Dataset Splitting into Training and Testing Sets + Feature Scaling

03 Model Building

04 Model Evaluation

05 SMOTE (Synthetic Minority Over-sampling Technique)

Classification Report						
ML Model	Class	Accuracy %	Precision %	Recall %	F1-Score %	ROC-AUC %
Logistic Regression	0	93	93	100	96	91
	1		96	45	61	
Random Forest	0	96	95	100	98	94
	1		99	64	78	
Logistic Regression (SMOTE train data)	0	84	97	85	91	
	1		42	79	55	
Random Forest (SMOTE train data)	0	95	96	99	97	
	1		91	69	78	
Logistic Regression (SMOTE all data)	0	88	86	91	89	
	1		90	86	88	
Random Forest (SMOTE all data)	0	97	95	100	97	
	1		100	94	97	

Untuk case prediksi kredit score, metrik yang menjadi fokus utama adalah Recall dan F1-score.

Maka dari hasil classification report dari 2 model yang di latih, maka ***Random forest memiliki performance yang paling baik setelah di lakukan oversampling dengan SMOTE.***, karena memiliki performance keseluruhan yang lebih baik.

Coefficient Model

	Feature	Coefficient
0	loan_amnt	-0.773327
1	term	0.042157
2	int_rate	1.118730
3	installment	2.186655
4	grade	-0.821525
5	emp_length	-0.041372
6	home_ownership	0.002908
7	annual_inc	-0.175437
8	verification_status	0.095559
9	purpose	0.021983
10	dti	-0.067010
11	open_acc	-0.007915
12	revol_bal	0.142113
13	revol_util	-0.026390
14	total_acc	0.093748
15	total_pymnt	-2.968347
16	total_rec_int	0.385251
17	collection_recovery_fee	58.014382
18	last_pymnt_amnt	-3.226983
19	tot_coll_amt	-0.857203
20	tot_cur_bal	-0.089274
21	total_rev_hi_lim	-0.265964
22	credit_status_encoded	-0.116578

- Model Koefisien tertinggi adalah fitur "***collection recovery fee***" dan "***installment***".
- Hal dikarenakan variabel tersebut berkaitan langsung dengan jumlah uang yang harus dibayar oleh peminjam, baik sebagai biaya tambahan akibat gagal bayar (collection recovery fee) maupun sebagai komponen utama dari pembayaran pinjaman (installment). Oleh karena itu, fluktuasi dalam variabel-variabel ini akan secara langsung mempengaruhi nilai target dalam model.

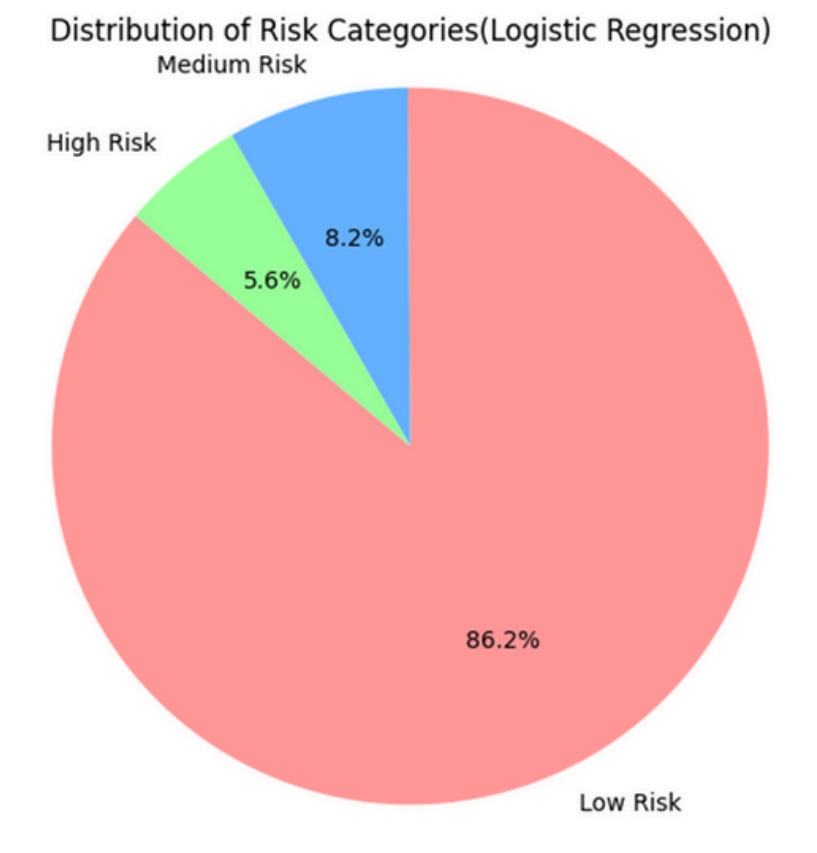
06. Penilaian Resiko Kredit

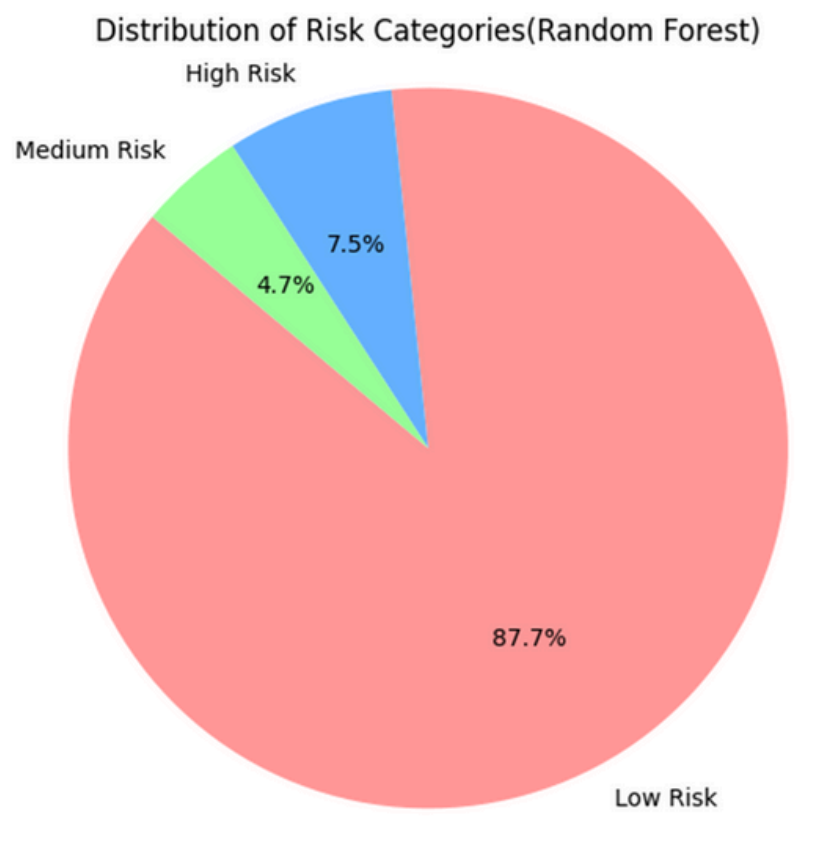
01 Prediksiprobabilitas pada data baru (X_test).

02 Menambahkan kolom baru "risk_score" pada dataset.

03 Mengelompokkan peminjam berdasarkan risk_score dengan memberi label Low_Risk (risk_score <0.2), Medium_Risk (0.2 <= risk score < 0.5), High_Risk (risk_score >0.5).

Distribusi Risk Category





•Berdasarkan hasil dari distribusi Risk category, maka dapat dilihat bahwa High risk dan Medium Risk lebih banyak di tangkap oleh

•Model machine Learning Logistic Regression dibandingkan dengan random Forest.

07. Insight & Recommandation

Semakin besar jumlah pinjaman, maka akan semakin besar pula grade resikonya. karena mendapatkan suku bunga yang lebih tinggi sebagai kompensasi tambahan. Namun, terdapat anomali pada Grade B di mana ada pinjaman sangat besar dengan tingkat resiko yang lebih kecil.

Dari model logistic regression yang sudah coba di bangun, di temukan bahwa fitur yang paling berpengaruh terhadap resiko peminjaman adalah "Collection Recovery Fee" dan "Installment". Hal ini dikarenakan varibel tersebut berkaitan langsung dengan jumlah yang harus di bayar oleh peminjam , baik sebagai biaya tambahan/denda karena gagal/telat bayar, maupun sebagai komponen utama dari pembayaran pinjaman.

Dari 2 Model klasifikasi yang sudah bangun, yaitu Logistic Regression dan Random Forest. Yang memiliki performa terbaik adalah Random forest setelah di lakukan oversampling dengan SMOTE. karena memiliki performance keseluruhan yang lebih baik, walaupun untuk menangkap kasus "Bad", logistic regression memiliki Recall yang lebih tinggi.

Berdasarkan penilaian resiko kredit, maka dapat dilihat bahwa High risk dan Medium Risk lebih banyak di tangkap oleh Model Logistic Regression dibandingkan dengan random Forest.

Untuk mengurangi resiko pinjaman karena gagal bayar, sebaiknya perlu mengidentifikasi peminjam beresiko tinggi sejak awal dan mengambil Langkah-Langkah preventif, seperti penawaran program bantuan keuangan atau penyesuaian persyaratan pinjaman.

CODES

import phyton libraries & data prepation

```
[1] #import library
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
```

```
[2] from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Import dataset

```
[3] #Import Dataset & Show 5 baris pertama
    df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Intern Data Science Rakamin/Final Task/loan_data_2007_2014.csv')
    df.sample(5)
```

```
<ipython-input-3-a45944b33c24>:2: DtypeWarning: Columns (20) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Intern Data Science Rakamin/Final Task/loan_data_2007_2014.csv')
```

	Unnamed: 0	id	member_id	loan_amnt	funded_amnt	funded_amnt_inv	term	int_rate	installment	grade	...	total_bal_il	il_util	open_rv_12m	open_rv_24m	max_bal_bc	all_util	total_rev_hi_l
11071	11071	821787	1030172	7800	7800	7800.0	36 months	11.99	259.04	B	...	NaN	NaN	NaN	NaN	NaN	NaN	Na
306695	306695	27700869	30213965	14000	14000	14000.0	36 months	8.39	441.24	A	...	NaN	NaN	NaN	NaN	NaN	NaN	1540(
266264	266264	32179067	34792301	24650	24650	24650.0	60 months	16.99	612.49	D	...	NaN	NaN	NaN	NaN	NaN	NaN	2940(
221984	221984	1210846	1452014	6400	6400	6400.0	36 ..	9.76	205.79	B	...	NaN	NaN	NaN	NaN	NaN	NaN	Na

Handle missing value & Labeling “Good” or “Bad”

```
[12] # Drop kolom dengan missing value > 40%
     threshold = 40
     cols_to_drop = missing_values[missing_values > threshold].index
     df.drop(cols_to_drop, axis=1, inplace=True)
```

```
[13] # Menampilkan kolom-kolom yang di-drop
     print(f"Kolom-kolom yang di-drop (missing value > {threshold}%):")
     print(cols_to_drop)
```

```
Kolom-kolom yang di-drop (missing value > 40%):
Index(['emp_title', 'emp_length', 'desc', 'mths_since_last_delinq',
       'mths_since_last_record', 'revol_util', 'last_pymnt_d', 'next_pymnt_d',
       'last_credit_pull_d', 'collections_12_mths_ex_med',
       'mths_since_last_major_derog', 'annual_inc_joint', 'dti_joint',
       'verification_status_joint', 'tot_coll_amt', 'tot_cur_bal',
       'open_acc_6m', 'open_il_6m', 'open_il_12m', 'open_il_24m',
       'mths_since_rcnt_il', 'total_bal_il', 'il_util', 'open_rv_12m',
       'open_rv_24m', 'max_bal_bc', 'all_util', 'total_rev_hi_lim', 'inq_fi',
       'total_cu_tl', 'inq_last_12m'],
      dtype='object')
```

```
[14] # Menampilkan jumlah kolom yang di-drop
     print(len(cols_to_drop))
```

```
31
```

Total Jumlah kolom yang di drop due to missing value >40% adalah 31 kolom.

Handle Missing Value dan Duplikasi Data

```
[10] #Melihat duplikat data
     df.duplicated().sum()
```

```
0
```

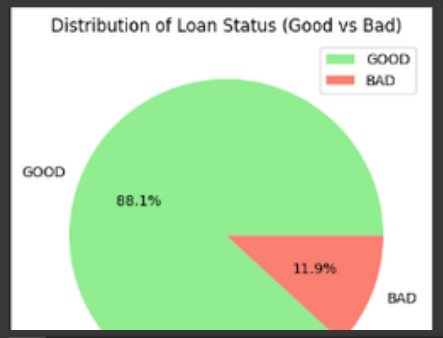
```
[17] # Imputasi kolom numerik dengan median dan kolom kategorik dengan modus
     from sklearn.impute import SimpleImputer

     # Imputasi untuk kolom numerik
     imputer_numerik = SimpleImputer(strategy='median')
     kolom_numerik = df.select_dtypes(include=['number']).columns
     df[kolom_numerik] = imputer_numerik.fit_transform(df[kolom_numerik])

     # Imputasi untuk kolom kategorikal
     imputer_kategorik = SimpleImputer(strategy='most_frequent')
     kolom_kategorik = df.select_dtypes(include=['object']).columns
     df[kolom_kategorik] = imputer_kategorik.fit_transform(df[kolom_kategorik])
```

```
[18] #Check missing value setelah dilakukan imputasi.
     missing_values_after = df.isnull().sum()
     print(missing_values_after)
```

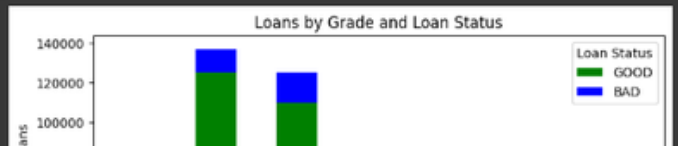
```
[21] # Menambahkan kolom baru 'loan_status_label' dengan label 'GOOD' atau 'BAD'
     df['loan_status_label'] = df['loan_status'].apply(lambda x: 'GOOD' if x in ['Current', 'Fully Paid', 'Does not meet the credit policy. Status:Fully Paid'] else 'BAD')

     # Menampilkan beberapa baris dari dataset untuk memastikan label telah ditambahkan
     print(df[['loan_status', 'loan_status_label']].head(10))
```

Data Visualization & get insight

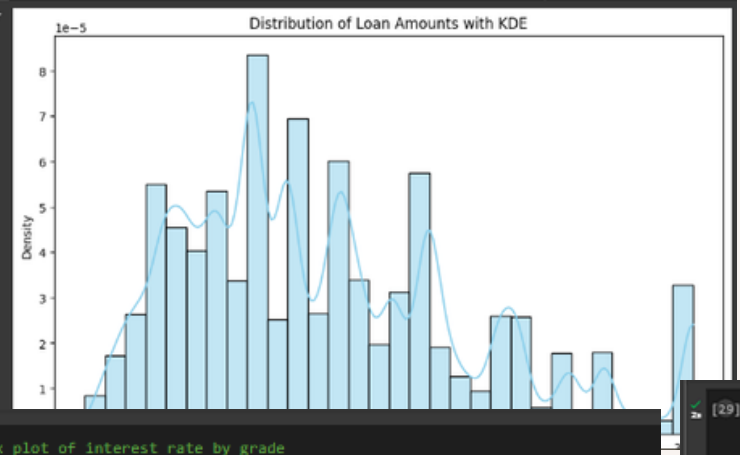
```
# Menghitung distribusi "GOOD" dan "BAD" dari kolom loan_status_label
loan_status_label_counts = df['loan_status_label'].value_counts()

# Plot distribusi Loan Status
plt.figure(figsize=(5, 5))
loan_status_label_counts.plot(kind='pie', autopct='%1.1f%%', colors=['lightgreen', 'salmon'])
plt.title('Distribution of Loan Status (Good vs Bad)')
plt.ylabel('')  # Hide the y-label
plt.legend(labels=loan_status_label_counts.index, loc="best")
plt.show()
```



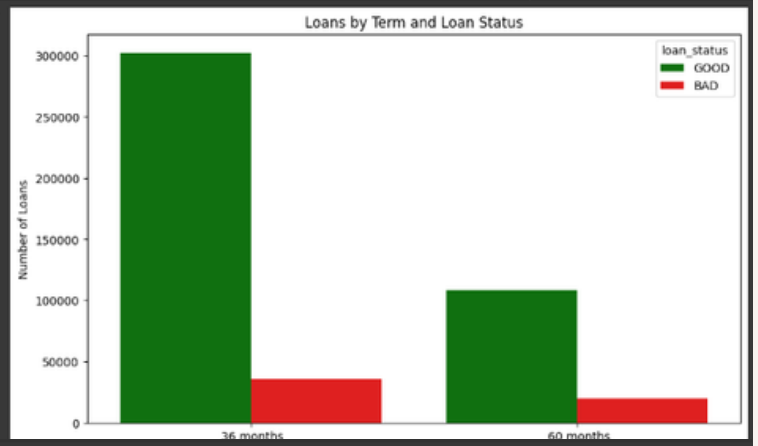

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Misalnya, kita punya DataFrame df dengan kolom 'grade' dan 'loan_status_label'
# df = pd.read_csv('your_data_file.csv')  # Baca data dari file

# Mengurutkan grade dari A ke G
grade_order = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
df['grade'] = pd.Categorical(df['grade'], categories=grade_order, ordered=True)

# Mengelompokkan data dan menghitung jumlah pinjaman berdasarkan grade dan status pinjaman
loan_counts = df.groupby(['grade', 'loan_status_label']).size().unstack(fill_value=0)

# Mengurutkan kolom agar 'bad' ada di atas
if 'bad' in loan_counts.columns:
    loan_counts = loan_counts[['good', 'bad']]
else:
    loan_counts = loan_counts[loan_counts.columns[::-1]]

# Plotting the stacked bar chart
loan_counts.plot(kind='bar', stacked=True, color=['green', 'blue'], figsize=(8, 4))

plt.title('Loans by Grade and Loan Status')
plt.xlabel('Grade')
plt.ylabel('Number of Loans')
plt.legend(title='Loan Status')
plt.show()
```




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

# Plotting the distribution of loan amounts with KDE
plt.figure(figsize=(10, 6))
sns.histplot(df['loan_amnt'], bins=30, kde=True, color='skyblue', edgecolor='k', stat="density")

plt.title('Distribution of Loan Amounts with KDE')
plt.xlabel('Loan Amount')
plt.ylabel('Density')

plt.show()
```

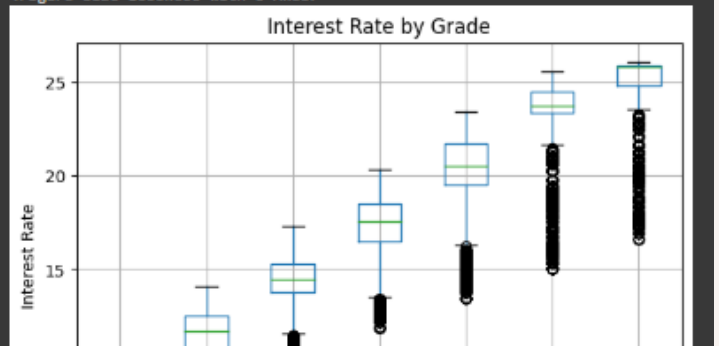



```
[25] # Plotting the count of loans by Term and loan status
plt.figure(figsize=(10, 6))
sns.countplot(data=df, x='term', hue='loan_status_label', palette=['green', 'red'])
plt.title('Loans by Term and Loan Status')
plt.xlabel('Term')
plt.ylabel('Number of Loans')
plt.legend(title='loan_status')
plt.show()
```




```
[28] # Box plot of interest rate by grade
plt.figure(figsize=(10, 6))
df.boxplot(column='int_rate', by='grade')
plt.title('Interest Rate by Grade')
plt.xlabel('Grade')
plt.ylabel('Interest Rate')
plt.suptitle('')
plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```




```
[29] # Plotting loan amount vs interest rate
plt.figure(figsize=(10, 6))
plt.scatter(df['loan_amnt'], df['int_rate'], alpha=0.01, edgecolor='k', linewidth=0.01)
plt.title('Loan Amount vs. Interest Rate')
plt.xlabel('Loan Amount')
plt.ylabel('Interest Rate')
plt.show()
```

Drop un-used Column, Convert categorical to numerical & label encoding

```
[30] # Daftar kolom yang akan dihapus
     columns_to_drop = [
         'id', 'member_id', 'url', 'title', 'zip_code', 'addr_state',
         'funded_amnt','funded_amnt_inv','sub_grade','issue_d','loan_status','earliest_cr_line','inq_last_6mths','initial_list_status',
         'Unnamed: 0', 'pymnt_plan', 'application_type', 'policy_code', 'total_pymnt_inv', 'total_rec_prncp', 'recoveries', 'total_rec_late_fee', 'out_prncp', 'out_prncp_inv'
     ]
     # Menghapus kolom-kolom yang tidak diperlukan
     df = df.drop(columns=columns_to_drop)
```

```
[32] # Konversi kolom kategori ke numerik
     df['term'] = df['term'].apply(lambda x: int(x.strip().split(' ')[0]))
     df['grade'] = df['grade'].astype('category').cat.codes
     df['home_ownership'] = df['home_ownership'].astype('category').cat.codes
     df['verification_status'] = df['verification_status'].astype('category').cat.codes
     df['purpose'] = df['purpose'].astype('category').cat.codes
```

Label Encoding

```
[33] # Labeling kolom loan_status_label
     df['loan_status_encoded'] = df['loan_status_label'].map({'BAD':1,'GOOD' :0})

     # Daftar kolom yang sudah di encode akan didrop
     columns_to_drop = [
         'loan_status_label'
     ]
     # drop kolom-kolom yang sudah di encode.
     df = df.drop(columns=columns_to_drop)
```

Dataset Splitting into Training and Testing Sets + Standard Scaler

Data Splitting for Features and Target Variable

```
[47] # Memisahkan fitur dan target
     X = df.drop(columns=['loan_status_encoded'])
     y = df['loan_status_encoded']
```

Dataset Splitting into Training and Testing Sets + Feature Scaling

```
[48] # Split the data into training and testing sets
     from sklearn.model_selection import train_test_split #Split data ke train dan test
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[49] # Feature Scaling dengan StandardScaler
     from sklearn.preprocessing import StandardScaler
     scaler = StandardScaler()
     X_train = scaler.fit_transform(X_train)
     X_test = scaler.transform(X_test)
```

ML Model 1 - Logistic regression & AUC-ROC

Logistic Regression

```
[ ] # Melatih model Logistic Regression
    from sklearn.linear_model import LogisticRegression
    model = LogisticRegression(random_state=42, solver='lbfgs', max_iter=1000)
    model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=1000, random_state=42)
```

```
[ ] # Memprediksi pada test set
    y_pred = model.predict(X_test)
```

```
[ ] # Evaluasi model
    from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
    accuracy = accuracy_score(y_test, y_pred)
    conf_matrix = confusion_matrix(y_test, y_pred)
    class_report = classification_report(y_test, y_pred)
```

```
[ ] print("Accuracy:", accuracy)
    print("Confusion Matrix:\n", conf_matrix)
    print("Classification Report:\n", class_report)
```

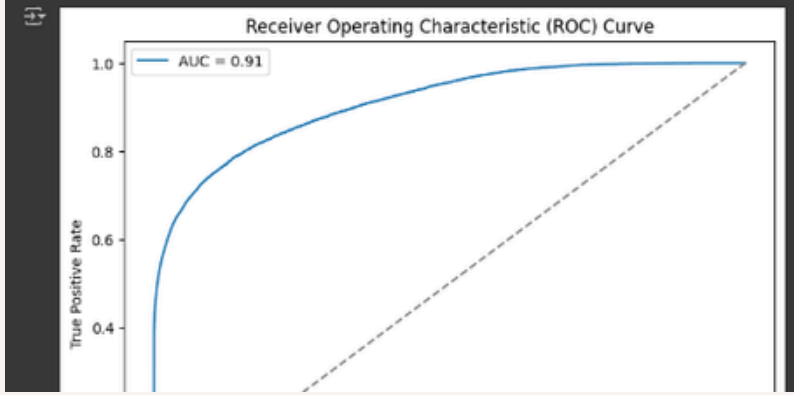
```
Accuracy: 0.9327771641807049
Confusion Matrix:
 [[81983   194]
 [ 6075  5005]]
Classification Report:
              precision    recall  f1-score   support

           0       0.93      1.00      0.96     82177
           1       0.96      0.45      0.61     11080

    accuracy                           0.93     93257
   macro avg       0.95      0.72      0.79     93257
weighted avg       0.93      0.93      0.92     93257
```

AUC-ROC Score

```
[ ] from sklearn.metrics import roc_auc_score, roc_curve

    y_probabilities = model.predict_proba(X_test)[:, 1]
    roc_auc = roc_auc_score(y_test, y_probabilities)
    print("AUC-ROC Score:", roc_auc)
```

```
AUC-ROC Score: 0.9101923496209577
```

```
[ ] fpr, tpr, thresholds = roc_curve(y_test, y_probabilities)
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
    plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend()
    plt.show()
```



ML Model 2 - Random Forest & AUC-ROC

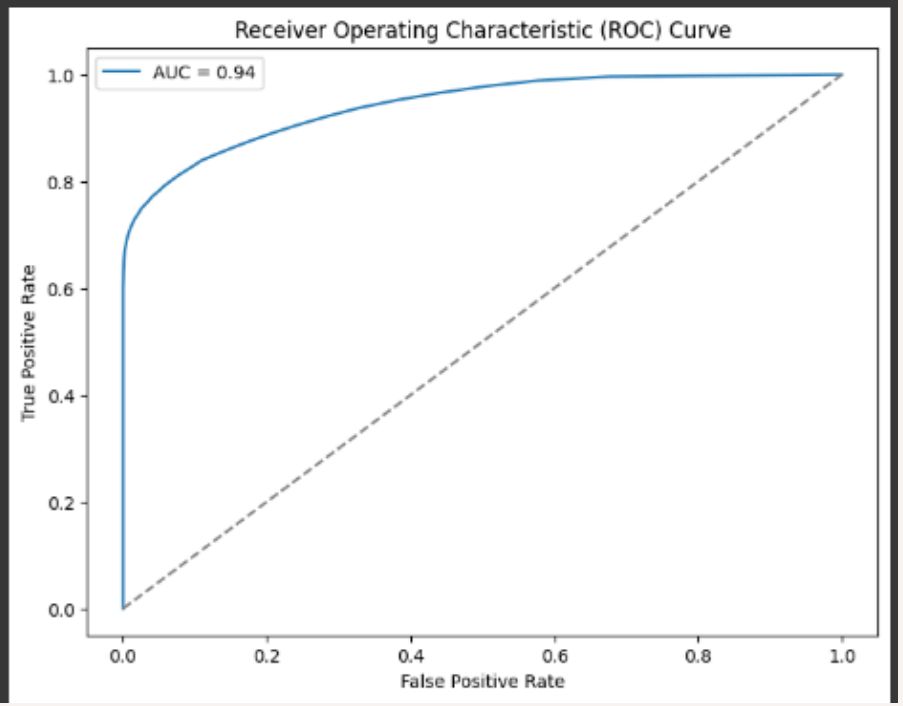
Random Forest

```
[40] #Inisialisasi model Random Forest
     from sklearn.ensemble import RandomForestClassifier
     model = RandomForestClassifier(random_state=42)
     model.fit(X_train, y_train)

     y_pred = model.predict(X_test)# Prediksi pada set pengujian
```

```
[42] # Cross-validation score
     from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
     from sklearn.model_selection import cross_val_score #Model Evaluation dengan cross-validasi
     cv_score = cross_val_score(model, X_train, y_train, cv=5)
     print("Cross-Validation Score:", np.mean(cv_score))

     # Model Evaluation Metrics
     accuracy = accuracy_score(y_test, y_pred)
     print("Accuracy:", accuracy)

     conf_matrix = confusion_matrix(y_test, y_pred)
     print("Confusion Matrix:\n", conf_matrix)

     class_report = classification_report(y_test, y_pred)
     print("Classification Report for Random Forest:\n", class_report)
```

```
Cross-Validation Score: 0.9560461961578908
Accuracy: 0.9562713790921861
Confusion Matrix:
 [[82100    77]
 [ 4001  7079]]
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       0.95      1.00      0.98     82177
           1       0.99      0.64      0.78     11080

    accuracy                           0.96     93257
   macro avg       0.97      0.82      0.88     93257
```

```
[44] fpr, tpr, thresholds = roc_curve(y_test, y_probabilities)
     plt.figure(figsize=(8, 6))
     plt.plot(fpr, tpr, label=f'AUC = {roc_auc:.2f}')
     plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Receiver Operating Characteristic (ROC) Curve')
     plt.legend()
     plt.show()
```



SMOTE at data train

SMOTE pada data Train

```
[ ]  from imblearn.over_sampling import SMOTE

     smote = SMOTE(random_state=42)
     X_resampled, y_resampled = smote.fit_resample(X_train, y_train)
     scaler = StandardScaler()
     X_train_scaled = scaler.fit_transform(X_resampled)
     X_test_scaled = scaler.transform(X_test)
```

Logistic Regration (SMOTE pada data Train)

```
[ ]  model = LogisticRegression(random_state=42, solver='lbfgs', max_iter=1000)
     model.fit(X_train_scaled, y_resampled)
     y_test_pred = model.predict(X_test_scaled)
     print(classification_report(y_test, y_test_pred))
```

```
                precision    recall  f1-score   support

             0       0.97      0.85      0.91     82177
             1       0.42      0.79      0.55     11080

      accuracy                           0.84     93257
     macro avg       0.69      0.82      0.73     93257
  weighted avg       0.90      0.84      0.86     93257
```

Random Forest (SMOTE pada data Train)

```
[ ]  model = RandomForestClassifier()
     model.fit(X_train_scaled, y_resampled)
     y_test_pred = model.predict(X_test_scaled)
     print(classification_report(y_test, y_test_pred))
```

```
                precision    recall  f1-score   support

             0       0.96      0.99      0.97     82177
             1       0.91      0.69      0.78     11080

      accuracy                           0.95     93257
```

SMOTE at all data as comparation

SMOTE terhadap keseluruhan data

```
[50] from imblearn.over_sampling import SMOTE

     # Menggunakan SMOTE untuk oversampling pada keseluruhan data
     smote = SMOTE(random_state=42)
     X_resampled, y_resampled = smote.fit_resample(X, y)

     # Scaling pada keseluruhan data yang telah di-resampling
     scaler = StandardScaler()
     X_resampled_scaled = scaler.fit_transform(X_resampled)

     # Membagi data yang telah di-resampling dan di-scaling menjadi train dan test
     X_train_resampled, X_test_resampled, y_train_resampled, y_test_resampled = train_test_split(
         X_resampled_scaled, y_resampled, test_size=0.3, random_state=42, stratify=y_resampled)
```

Logistic Regration (SMOTE data keseluruhan)

```
[ ]  # Melatih model
     model = LogisticRegression(random_state=42, solver='lbfgs', max_iter=1000)
     model.fit(X_train_resampled, y_train_resampled)

     # Memprediksi dan mengevaluasi model
     y_test_resampled_pred = model.predict(X_test_resampled)
     print("Classification Report for SMOTE on Entire Data:")
     print(classification_report(y_test_resampled, y_test_resampled_pred))
```

```
Classification Report for SMOTE on Entire Data:
              precision    recall  f1-score   support

           0       0.86      0.91      0.89    123286
           1       0.90      0.86      0.88    123286

    accuracy                           0.88    246572
   macro avg       0.88      0.88      0.88    246572
weighted avg       0.88      0.88      0.88    246572
```

Random Forest (SMOTE data keseluruhan)

```
[51] model = RandomForestClassifier(random_state=42)
     model.fit(X_train_resampled, y_train_resampled)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
[52] # Memprediksi dan mengevaluasi model
     y_test_resampled_pred = model.predict(X_test_resampled)
     print("Classification Report for SMOTE on Entire Data with Random Forest:")
     print(classification_report(y_test_resampled, y_test_resampled_pred))
```

```
Classification Report for SMOTE on Entire Data with Random Forest:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97    123286
           1       1.00      0.94      0.97    123286

    accuracy                           0.97    246572
   macro avg       0.97      0.97      0.97    246572
weighted avg       0.97      0.97      0.97    246572
```

Risk credit score and classification into Low risk, medium risk, high risk (Random forest)

```
[ ]  #Inisialisasi model Random Forest
     from sklearn.ensemble import RandomForestClassifier
     model = RandomForestClassifier(random_state=42)
     model.fit(X_train, y_train)
```

```
RandomForestClassifier(random_state=42)
```

```
[ ]  # Prediksi probabilitas pada data baru (X_test)
     predictions = model.predict_proba(X_test)[:, 1]
```

```
[ ]  # Menambahkan kolom skor risiko pada dataset
     X_test = pd.DataFrame(X_test)
     X_test['risk_score'] = predictions
```

```
[ ]  # Mengelompokkan peminjam berdasarkan skor risiko
     def risk_category(score):
         if score < 0.2:
             return 'Low Risk'
         elif 0.2 <= score < 0.5:
             return 'Medium Risk'
         else:
             return 'High Risk'

     X_test['Risk Category'] = X_test['risk_score'].apply(risk_category)

     # Menampilkan jumlah peminjam di setiap kategori risiko
     risk_category_counts = X_test['Risk Category'].value_counts()
     print(risk_category_counts)
```

```
Risk Category
Low Risk       81872
High Risk       7186
Medium Risk     4199
Name: count, dtype: int64
```

```
[ ]  # Plotting Pie chart Risk Category
     plt.figure(figsize=(6, 6))
     plt.pie(risk_category_counts, labels=risk_category_counts.index, autopct='%1.1f%%', startangle=140, colors=['#ff9999','#66b3ff','#99ff99'])
     plt.title('Distribution of Risk Categories(Random Forest)')
     plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
     plt.show()
```

Risk credit score and classification into Low risk, medium risk, high risk (Logistic regression)

Penilaian Resiko Kredit dengan Logistic Regression

```
[ ] #Inisialisasi model LogisticRegression
    model = LogisticRegression(random_state=42, solver='lbfgs', max_iter=1000)
    model.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

```
[ ] # Prediksi probabilitas pada data baru (X_test)
    predictions = model.predict_proba(X_test)[:, 1]
```

```
[ ] # Menambahkan kolom skor risiko pada dataset
    X_test = pd.DataFrame(X_test)
    X_test['risk_score'] = predictions
```

```
[ ] # Menampilkan data yang sudah diprioritaskan
    print(X_test.head())
```

```
# Mengelompokkan peminjam berdasarkan skor risiko
def risk_category(score):
    if score < 0.2:
        return 'Low Risk'
    elif 0.2 <= score < 0.5:
        return 'Medium Risk'
    else:
        return 'High Risk'

X_test['Risk Category'] = X_test['risk_score'].apply(risk_category)

# Menampilkan jumlah peminjam di setiap kategori risiko
risk_category_counts = X_test['Risk Category'].value_counts()
print(risk_category_counts)
```

```
Risk Category
Low Risk       80491
Medium Risk     7567
High Risk       5199
Name: count, dtype: int64
```

```
[ ] # Plotting Pie chart Risk Category
    plt.figure(figsize=(6, 6))
    plt.pie(risk_category_counts, labels=risk_category_counts.index, autopct='%1.1f%%', startangle=140, colors=['#ff9999','#66b3ff','#99ff99'])
    plt.title('Distribution of Risk Categories(Logistic Regression)')
    plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
    plt.show()
```

Thank You