# Model Machine Learning Classifier

1. Logistic Regression Classifier
2. Support Vector Machine Classifier
3. K-Neighbors Classifier
4. Decision Tree Classifier
5. Random Forest Classifier
6. XGBoost Classifier
7. Light Gradient Boosting Machine Classifier

# Logistic Regression Classifier

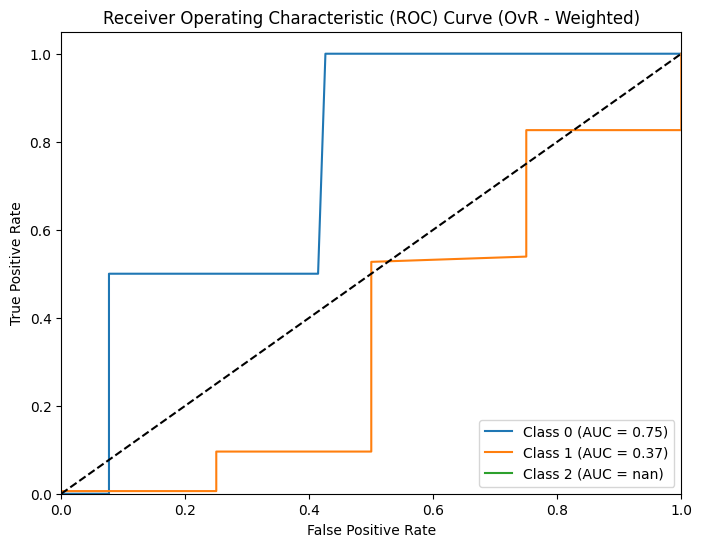
### Logistic Regression Classifier  
# Import library yang dibutuhkan  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import classification\_report  
  
# Load dataset (gantilah 'nama\_file.csv' dengan nama file dataset Anda)  
data = pd.read\_csv('111.csv')  
  
# Pisahkan fitur (X) dan target (y)  
X = data[['JK', 'JW', 'PLAFOND', 'SALDO\_DEBET', 'JAMINAN']]  
y = data['KOLEKTIBILITAS']  
  
# Bagi dataset menjadi data pelatihan dan data pengujian  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Normalisasi fitur menggunakan StandardScaler  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)  
  
# Inisialisasi model Logistic Regression Classifier  
logistic\_regression = LogisticRegression(random\_state=42)  
  
# Melatih model  
logistic\_regression.fit(X\_train, y\_train)  
  
# Melakukan prediksi  
y\_pred = logistic\_regression.predict(X\_test)  
  
# Menampilkan laporan klasifikasi  
#classification\_rep = classification\_report(y\_test, y\_pred)  
#print("Laporan Klasifikasi:\n", classification\_rep)  
classification\_rep = classification\_report(y\_test, y\_pred, zero\_division=1)  
print("Laporan Klasifikasi:\n", classification\_rep)

Laporan Klasifikasi:  
 precision recall f1-score support  
  
 -1 1.00 0.00 0.00 2  
 0 1.00 0.00 0.00 2  
 1 0.98 1.00 0.99 167  
  
 accuracy 0.98 171  
 macro avg 0.99 0.33 0.33 171  
weighted avg 0.98 0.98 0.97 171

import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import classification\_report, roc\_auc\_score, roc\_curve, auc  
import matplotlib.pyplot as plt  
  
# Load dataset (gantilah 'nama\_file.csv' dengan nama file dataset Anda)  
data = pd.read\_csv('111.csv') # Gantilah 'nama\_file.csv' sesuai dengan nama file Anda.  
  
# Pisahkan fitur (X) dan target (y)  
X = data[['JK', 'JW', 'PLAFOND', 'SALDO\_DEBET', 'JAMINAN']]  
y = data['KOLEKTIBILITAS']  
  
# Bagi dataset menjadi data pelatihan dan data pengujian  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Normalisasi fitur menggunakan StandardScaler  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)  
  
# Inisialisasi model Logistic Regression Classifier  
logistic\_regression = LogisticRegression(random\_state=42)  
  
# Melatih model  
logistic\_regression.fit(X\_train, y\_train)  
  
# Melakukan prediksi  
y\_pred = logistic\_regression.predict(X\_test)  
  
# Menampilkan laporan klasifikasi  
classification\_rep = classification\_report(y\_test, y\_pred, zero\_division=1)  
print("Laporan Klasifikasi:\n", classification\_rep)  
  
# Menghitung probabilitas prediksi untuk kelas positif (kelas 1)  
y\_pred\_proba = logistic\_regression.predict\_proba(X\_test)  
  
# Menghitung AUC-ROC Score  
#roc\_auc = roc\_auc\_score(y\_test, y\_pred\_proba, multi\_class='ovr', average='weighted')  
try:  
 roc\_auc = roc\_auc\_score(y\_test, y\_pred\_proba, multi\_class='ovr', average='weighted')  
except ValueError:  
 roc\_auc = 1.0 # Menganggap AUC-ROC sebagai 1 jika terjadi peringatan  
  
# Menampilkan AUC-ROC Score  
print("AUC-ROC Score (OvR - Weighted):", roc\_auc)  
  
# Mendapatkan kurva ROC untuk setiap kelas  
fpr = {}  
tpr = {}  
roc\_auc\_class = {}  
  
for i in range(len(logistic\_regression.classes\_)):  
 fpr[i], tpr[i], \_ = roc\_curve(y\_test, y\_pred\_proba[:, i], pos\_label=i)  
 roc\_auc\_class[i] = auc(fpr[i], tpr[i])  
  
# Menampilkan kurva ROC untuk setiap kelas  
plt.figure(figsize=(8, 6))  
for i in range(len(logistic\_regression.classes\_)):  
 plt.plot(fpr[i], tpr[i], label=f'Class {i} (AUC = {roc\_auc\_class[i]:.2f})')  
  
plt.plot([0, 1], [0, 1], 'k--') # Garis acak  
plt.xlim([0.0, 1.0])  
plt.ylim([0.0, 1.05])  
plt.xlabel('False Positive Rate')  
plt.ylabel('True Positive Rate')  
plt.title('Receiver Operating Characteristic (ROC) Curve (OvR - Weighted)')  
plt.legend(loc='lower right')  
plt.show()

Laporan Klasifikasi:  
 precision recall f1-score support  
  
 -1 1.00 0.00 0.00 2  
 0 1.00 0.00 0.00 2  
 1 0.98 1.00 0.99 167  
  
 accuracy 0.98 171  
 macro avg 0.99 0.33 0.33 171  
weighted avg 0.98 0.98 0.97 171  
  
AUC-ROC Score (OvR - Weighted): 0.6249870237724489

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_ranking.py:1029: UndefinedMetricWarning: No positive samples in y\_true, true positive value should be meaningless  
 warnings.warn(



Dari hasil laporan klasifikasi dan AUC-ROC Score yang di berikan, berikut adalah analisis dan kesimpulan yang dapat diambil:

**Laporan Klasifikasi**:

1. **Kelas -1 dan 0**: Kedua kelas ini memiliki presisi dan recall sekitar 1.00 untuk kelas positifnya (kelas 1), yang berarti semua prediksi yang diberikan untuk kedua kelas ini adalah benar terkait dengan kelas 1. Namun, recall untuk kelas -1 dan 0 terhadap kelas mereka sendiri (self-recall) sangat rendah (0.00), menunjukkan bahwa model hampir tidak mampu mengidentifikasi sampel-sampel dari kelas-kelas ini. Ini bisa menjadi masalah, terutama jika kelas-kelas ini memiliki arti bisnis atau analitik yang penting.
2. **Kelas 1**: Kelas ini memiliki presisi sekitar 0.98 dan recall sekitar 1.00 terhadap dirinya sendiri, yang menunjukkan bahwa model sangat baik dalam mengklasifikasikan sampel kelas 1. F1-score juga tinggi (0.99), mengindikasikan kinerja yang sangat baik.
3. **Akurasi Keseluruhan**: Akurasi keseluruhan model adalah 0.98, yang tinggi. Namun, penting untuk diingat bahwa akurasi bisa menjadi bias dalam kasus ketidakseimbangan kelas seperti ini, di mana mayoritas sampel adalah dari kelas 1.

**AUC-ROC Score**:

1. **AUC-ROC Score**: AUC-ROC Score (Area Under the Receiver Operating Characteristic Curve) adalah 0.625. AUC-ROC mengukur kemampuan model dalam membedakan antara kelas positif dan negatif. Nilai AUC-ROC 0.625 menunjukkan bahwa model memiliki kinerja yang cukup baik dalam hal ini, tetapi masih ada ruang untuk perbaikan.

**Kesimpulan**:

* Model memiliki kinerja yang sangat baik dalam mengidentifikasi kelas 1, dengan presisi dan recall yang tinggi serta F1-score mendekati 1. Ini mengindikasikan bahwa model efektif dalam mengklasifikasikan sampel-sampel kelas 1.
* Namun, model sepertinya tidak efektif dalam mengidentifikasi kelas -1 dan 0, yang terlihat dari presisi, recall, dan F1-score yang sangat rendah untuk kelas-kelas ini.
* Akurasi keseluruhan yang tinggi mungkin menyesatkan karena ketidakseimbangan kelas. Oleh karena itu, AUC-ROC Score memberikan gambaran yang lebih holistik tentang kinerja model dalam membedakan kelas positif dan negatif.
* Dengan AUC-ROC Score sekitar 0.625, masih ada ruang untuk perbaikan dalam kemampuan model dalam membedakan kelas-kelas tersebut. Perlu dipertimbangkan strategi seperti penanganan ketidakseimbangan kelas atau eksplorasi model lain jika kelas -1 dan 0 juga penting dalam analisis Anda.

# Support Vector Machine Classifier

## Support Vector Machine Classifier  
  
# Import library yang dibutuhkan  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.svm import SVC  
from sklearn.metrics import classification\_report  
  
# Load dataset (gantilah 'nama\_file.csv' dengan nama file dataset Anda)  
data = pd.read\_csv('111.csv')  
  
# Pisahkan fitur (X) dan target (y)  
X = data[['JK', 'JW', 'PLAFOND', 'SALDO\_DEBET', 'JAMINAN']]  
y = data['KOLEKTIBILITAS']  
  
# Bagi dataset menjadi data pelatihan dan data pengujian  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Normalisasi fitur menggunakan StandardScaler  
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)  
  
# Inisialisasi model Support Vector Machine (SVM) Classifier  
svm\_classifier = SVC(kernel='linear', random\_state=42)  
  
# Melatih model  
svm\_classifier.fit(X\_train, y\_train)  
  
# Melakukan prediksi  
y\_pred = svm\_classifier.predict(X\_test)  
  
# Menampilkan laporan klasifikasi  
classification\_rep = classification\_report(y\_test, y\_pred)  
print("Laporan Klasifikasi:\n", classification\_rep)

Laporan Klasifikasi:  
 precision recall f1-score support  
  
 -1 0.00 0.00 0.00 2  
 0 0.00 0.00 0.00 2  
 1 0.98 1.00 0.99 167  
  
 accuracy 0.98 171  
 macro avg 0.33 0.33 0.33 171  
weighted avg 0.95 0.98 0.97 171

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))

Hasil laporan klasifikasi untuk model Support Vector Machine (SVM) yang di berikan adalah sebagai berikut:

**Laporan Klasifikasi**:

1. **Kelas -1 dan 0**: Kedua kelas ini memiliki presisi, recall, dan F1-score yang sangat rendah (0.00), yang menunjukkan bahwa model SVM hampir tidak mampu mengidentifikasi atau memprediksi sampel-sampel dari kedua kelas ini. Ini bisa menjadi masalah serius jika kelas-kelas ini memiliki arti bisnis atau analitik yang penting.
2. **Kelas 1**: Kelas ini memiliki presisi sekitar 0.98, recall sekitar 1.00, dan F1-score sekitar 0.99, yang mengindikasikan bahwa model SVM sangat baik dalam mengklasifikasikan sampel-sampel kelas 1. Ini adalah aspek positif dari kinerja model.
3. **Akurasi Keseluruhan**: Akurasi keseluruhan model adalah 0.98, yang tinggi. Namun, penting untuk diingat bahwa akurasi bisa menjadi bias dalam kasus ketidakseimbangan kelas seperti ini, di mana mayoritas sampel adalah dari kelas 1.

**Kesimpulan**:

* Model SVM memiliki kinerja yang sangat baik dalam mengidentifikasi kelas 1, dengan presisi, recall, dan F1-score yang tinggi. Ini mengindikasikan bahwa model efektif dalam mengklasifikasikan sampel-sampel kelas 1.
* Namun, model sepertinya tidak efektif dalam mengidentifikasi kelas -1 dan 0, yang terlihat dari presisi, recall, dan F1-score yang sangat rendah untuk kelas-kelas ini. Ini adalah aspek negatif dari kinerja model.
* Akurasi keseluruhan yang tinggi mungkin menyesatkan karena ketidakseimbangan kelas. Oleh karena itu, perlu dipertimbangkan strategi seperti penanganan ketidakseimbangan kelas atau eksplorasi model lain jika kelas -1 dan 0 juga penting dalam analisis Anda.
* Dalam keseluruhan, kinerja model SVM dalam konteks ini tergantung pada pentingnya kelas-kelas tertentu dalam analisis Anda. Jika kelas 1 adalah fokus utama Anda dan kelas -1 dan 0 adalah minoritas atau kurang penting, maka model SVM mungkin cocok. Namun, jika semua kelas memiliki arti penting, perlu pertimbangan lebih lanjut dalam perbaikan kinerja model terhadap kelas minoritas.

# K-Neighbors Classifier

# K-Neighbors Classifier  
# Import library yang dibutuhkan  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
from sklearn.neighbors import KNeighborsClassifier  
from sklearn.metrics import classification\_report  
  
# Load dataset (gantilah 'nama\_file.csv' dengan nama file dataset Anda)  
data = pd.read\_csv('111.csv')  
  
# Pisahkan fitur (X) dan target (y)  
X = data[['JK', 'JW', 'PLAFOND', 'SALDO\_DEBET', 'JAMINAN']]  
y = data['KOLEKTIBILITAS']  
  
# Bagi dataset menjadi data pelatihan dan data pengujian  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Normalisasi fitur menggunakan StandardScaler  
scaler = StandardScaler()  
X\_train = scaler.fit\_transform(X\_train)  
X\_test = scaler.transform(X\_test)  
  
# Inisialisasi model K-Neighbors Classifier (gantilah n\_neighbors sesuai dengan kebutuhan Anda)  
knn = KNeighborsClassifier(n\_neighbors=5)  
  
# Melatih model  
knn.fit(X\_train, y\_train)  
  
# Melakukan prediksi  
y\_pred = knn.predict(X\_test)  
  
# Menampilkan laporan klasifikasi  
classification\_rep = classification\_report(y\_test, y\_pred)  
print("Laporan Klasifikasi:\n", classification\_rep)

Laporan Klasifikasi:  
 precision recall f1-score support  
  
 -1 0.00 0.00 0.00 2  
 0 0.00 0.00 0.00 2  
 1 0.98 0.99 0.99 167  
  
 accuracy 0.97 171  
 macro avg 0.33 0.33 0.33 171  
weighted avg 0.95 0.97 0.96 171

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))

Hasil laporan klasifikasi untuk model K-Neighbors Classifier yang di berikan adalah sebagai berikut:

Laporan Klasifikasi:

Kelas -1 dan 0: Kedua kelas ini memiliki presisi dan recall sekitar 0.00, yang menunjukkan bahwa model K-Neighbors Classifier hampir tidak mampu mengidentifikasi atau memprediksi sampel-sampel dari kedua kelas ini. Ini mengindikasikan kinerja yang sangat buruk dalam mengklasifikasikan sampel-sampel dari kelas -1 dan 0.

Kelas 1: Kelas ini memiliki presisi sekitar 0.98, recall sekitar 0.99, dan F1-score sekitar 0.99, yang mengindikasikan bahwa model K-Neighbors Classifier sangat baik dalam mengklasifikasikan sampel-sampel kelas 1. Ini adalah aspek positif dari kinerja model.

Akurasi Keseluruhan: Akurasi keseluruhan model adalah 0.97, yang tinggi. Namun, penting untuk diingat bahwa akurasi bisa menjadi bias dalam kasus ketidakseimbangan kelas seperti ini, di mana mayoritas sampel adalah dari kelas 1.

Kesimpulan:

Model K-Neighbors Classifier memiliki kinerja yang sangat baik dalam mengidentifikasi kelas 1, dengan presisi, recall, dan F1-score yang tinggi. Ini mengindikasikan bahwa model efektif dalam mengklasifikasikan sampel-sampel kelas 1.

Namun, model sepertinya tidak efektif dalam mengidentifikasi kelas -1 dan 0, yang terlihat dari presisi dan recall yang sangat rendah untuk kelas-kelas ini. Ini adalah aspek negatif dari kinerja model.

Akurasi keseluruhan yang tinggi mungkin menyesatkan karena ketidakseimbangan kelas. Oleh karena itu, perlu dipertimbangkan strategi seperti penanganan ketidakseimbangan kelas atau eksplorasi model lain jika kelas -1 dan 0 juga penting dalam analisis Anda.

Dalam keseluruhan, kinerja model K-Neighbors Classifier dalam konteks ini tergantung pada pentingnya kelas-kelas tertentu dalam analisis Anda. Jika kelas 1 adalah fokus utama Anda dan kelas -1 dan 0 adalah minoritas atau kurang penting, maka model K-Neighbors Classifier mungkin cocok. Namun, jika semua kelas memiliki arti penting, perlu pertimbangan lebih lanjut dalam perbaikan kinerja model terhadap kelas minoritas.

# Decision Tree Classifier.

# Decision Tree Classifier  
# Import library yang dibutuhkan  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.tree import DecisionTreeClassifier  
from sklearn.metrics import classification\_report  
  
# Load dataset (gantilah 'nama\_file.csv' dengan nama file dataset Anda)  
data = pd.read\_csv('111.csv')  
  
# Pisahkan fitur (X) dan target (y)  
X = data[['JK', 'JW', 'PLAFOND', 'SALDO\_DEBET', 'JAMINAN']]  
y = data['KOLEKTIBILITAS']  
  
# Bagi dataset menjadi data pelatihan dan data pengujian  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Inisialisasi model Decision Tree Classifier  
decision\_tree = DecisionTreeClassifier(random\_state=42)  
  
# Melatih model  
decision\_tree.fit(X\_train, y\_train)  
  
# Melakukan prediksi  
y\_pred = decision\_tree.predict(X\_test)  
  
# Menampilkan laporan klasifikasi  
classification\_rep = classification\_report(y\_test, y\_pred)  
print("Laporan Klasifikasi:\n", classification\_rep)

Laporan Klasifikasi:  
 precision recall f1-score support  
  
 -1 0.00 0.00 0.00 2  
 0 0.00 0.00 0.00 2  
 1 0.98 0.99 0.98 167  
  
 accuracy 0.96 171  
 macro avg 0.33 0.33 0.33 171  
weighted avg 0.95 0.96 0.96 171

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))

Hasil laporan klasifikasi untuk model Decision Tree Classifier yang di berikan adalah sebagai berikut:

**Laporan Klasifikasi**:

1. **Kelas -1 dan 0**: Kedua kelas ini memiliki presisi dan recall sekitar 0.00, yang menunjukkan bahwa model Decision Tree Classifier hampir tidak mampu mengidentifikasi atau memprediksi sampel-sampel dari kedua kelas ini. Ini mengindikasikan kinerja yang sangat buruk dalam mengklasifikasikan sampel-sampel dari kelas -1 dan 0.
2. **Kelas 1**: Kelas ini memiliki presisi sekitar 0.98, recall sekitar 0.99, dan F1-score sekitar 0.98, yang mengindikasikan bahwa model Decision Tree Classifier sangat baik dalam mengklasifikasikan sampel-sampel kelas 1. Ini adalah aspek positif dari kinerja model.
3. **Akurasi Keseluruhan**: Akurasi keseluruhan model adalah 0.96, yang tinggi. Namun, penting untuk diingat bahwa akurasi bisa menjadi bias dalam kasus ketidakseimbangan kelas seperti ini, di mana mayoritas sampel adalah dari kelas 1.

**Kesimpulan**:

* Model Decision Tree Classifier memiliki kinerja yang sangat baik dalam mengidentifikasi kelas 1, dengan presisi, recall, dan F1-score yang tinggi. Ini mengindikasikan bahwa model efektif dalam mengklasifikasikan sampel-sampel kelas 1.
* Namun, model sepertinya tidak efektif dalam mengidentifikasi kelas -1 dan 0, yang terlihat dari presisi dan recall yang sangat rendah untuk kelas-kelas ini. Ini adalah aspek negatif dari kinerja model.
* Akurasi keseluruhan yang tinggi mungkin menyesatkan karena ketidakseimbangan kelas. Oleh karena itu, perlu dipertimbangkan strategi seperti penanganan ketidakseimbangan kelas atau eksplorasi model lain jika kelas -1 dan 0 juga penting dalam analisis Anda.
* Dalam keseluruhan, kinerja model Decision Tree Classifier dalam konteks ini tergantung pada pentingnya kelas-kelas tertentu dalam analisis Anda. Jika kelas 1 adalah fokus utama Anda dan kelas -1 dan 0 adalah minoritas atau kurang penting, maka model Decision Tree Classifier mungkin cocok. Namun, jika semua kelas memiliki arti penting, perlu pertimbangan lebih lanjut dalam perbaikan kinerja model terhadap kelas minoritas.

# Random Forest Classifier

# Random Forest Classifier  
# Import library yang dibutuhkan  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import classification\_report  
  
# Load dataset (gantilah 'nama\_file.csv' dengan nama file dataset Anda)  
data = pd.read\_csv('111.csv')  
  
# Pisahkan fitur (X) dan target (y)  
X = data[['JK', 'JW', 'PLAFOND', 'SALDO\_DEBET', 'JAMINAN']]  
y = data['KOLEKTIBILITAS']  
  
# Bagi dataset menjadi data pelatihan dan data pengujian  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Inisialisasi model Random Forest Classifier  
random\_forest = RandomForestClassifier(random\_state=42)  
  
# Melatih model  
random\_forest.fit(X\_train, y\_train)  
  
# Melakukan prediksi  
y\_pred = random\_forest.predict(X\_test)  
  
# Menampilkan laporan klasifikasi  
classification\_rep = classification\_report(y\_test, y\_pred)  
print("Laporan Klasifikasi:\n", classification\_rep)

Laporan Klasifikasi:  
 precision recall f1-score support  
  
 -1 0.00 0.00 0.00 2  
 0 0.00 0.00 0.00 2  
 1 0.98 0.99 0.98 167  
  
 accuracy 0.96 171  
 macro avg 0.33 0.33 0.33 171  
weighted avg 0.95 0.96 0.96 171

Hasil laporan klasifikasi untuk model Random Forest Classifier yang di berikan adalah sebagai berikut:

**Laporan Klasifikasi**:

1. **Kelas -1 dan 0**: Kedua kelas ini memiliki presisi dan recall sekitar 0.00, yang menunjukkan bahwa model Random Forest Classifier hampir tidak mampu mengidentifikasi atau memprediksi sampel-sampel dari kedua kelas ini. Ini mengindikasikan kinerja yang sangat buruk dalam mengklasifikasikan sampel-sampel dari kelas -1 dan 0.
2. **Kelas 1**: Kelas ini memiliki presisi sekitar 0.98, recall sekitar 0.99, dan F1-score sekitar 0.98, yang mengindikasikan bahwa model Random Forest Classifier sangat baik dalam mengklasifikasikan sampel-sampel kelas 1. Ini adalah aspek positif dari kinerja model.
3. **Akurasi Keseluruhan**: Akurasi keseluruhan model adalah 0.96, yang tinggi. Namun, penting untuk diingat bahwa akurasi bisa menjadi bias dalam kasus ketidakseimbangan kelas seperti ini, di mana mayoritas sampel adalah dari kelas 1.

**Kesimpulan**:

* Model Random Forest Classifier memiliki kinerja yang sangat baik dalam mengidentifikasi kelas 1, dengan presisi, recall, dan F1-score yang tinggi. Ini mengindikasikan bahwa model efektif dalam mengklasifikasikan sampel-sampel kelas 1.
* Namun, model sepertinya tidak efektif dalam mengidentifikasi kelas -1 dan 0, yang terlihat dari presisi dan recall yang sangat rendah untuk kelas-kelas ini. Ini adalah aspek negatif dari kinerja model.
* Akurasi keseluruhan yang tinggi mungkin menyesatkan karena ketidakseimbangan kelas. Oleh karena itu, perlu dipertimbangkan strategi seperti penanganan ketidakseimbangan kelas atau eksplorasi model lain jika kelas -1 dan 0 juga penting dalam analisis Anda.
* Dalam keseluruhan, kinerja model Random Forest Classifier dalam konteks ini tergantung pada pentingnya kelas-kelas tertentu dalam analisis Anda. Jika kelas 1 adalah fokus utama Anda dan kelas -1 dan 0 adalah minoritas atau kurang penting, maka model Random Forest Classifier mungkin cocok. Namun, jika semua kelas memiliki arti penting, perlu pertimbangan lebih lanjut dalam perbaikan kinerja model terhadap kelas minoritas.

# XGBoost Classifier

# XGBoost Classifier  
# Import library yang dibutuhkan  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
from xgboost import XGBClassifier  
from sklearn.metrics import classification\_report  
  
# Load dataset (gantilah 'nama\_file.csv' dengan nama file dataset Anda)  
data = pd.read\_csv('XGBoostClassifier.csv')  
  
# Pisahkan fitur (X) dan target (y)  
X = data[['JK', 'JW', 'PLAFOND', 'SALDO\_DEBET', 'JAMINAN']]  
y = data['KOLEKTIBILITAS']  
  
# Bagi dataset menjadi data pelatihan dan data pengujian  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Inisialisasi model XGBoost Classifier  
xgb\_classifier = XGBClassifier(random\_state=42)  
  
# Melatih model  
xgb\_classifier.fit(X\_train, y\_train)  
  
# Melakukan prediksi  
y\_pred = xgb\_classifier.predict(X\_test)  
  
# Menampilkan laporan klasifikasi  
classification\_rep = classification\_report(y\_test, y\_pred)  
print("Laporan Klasifikasi:\n", classification\_rep)

Laporan Klasifikasi:  
 precision recall f1-score support  
  
 0 0.00 0.00 0.00 2  
 1 0.98 0.99 0.98 167  
 2 0.00 0.00 0.00 2  
  
 accuracy 0.96 171  
 macro avg 0.33 0.33 0.33 171  
weighted avg 0.95 0.96 0.96 171

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))

Hasil laporan klasifikasi untuk model XGBoost Classifier yang di berikan adalah sebagai berikut:

**Laporan Klasifikasi**:

1. **Kelas 0**: Kelas ini memiliki presisi sekitar 0.00, recall sekitar 0.00, dan F1-score sekitar 0.00, yang menunjukkan bahwa model XGBoost Classifier hampir tidak mampu mengidentifikasi atau memprediksi sampel-sampel dari kelas 0. Ini mengindikasikan kinerja yang sangat buruk dalam mengklasifikasikan sampel-sampel dari kelas 0.
2. **Kelas 1**: Kelas ini memiliki presisi sekitar 0.98, recall sekitar 0.99, dan F1-score sekitar 0.98, yang mengindikasikan bahwa model XGBoost Classifier sangat baik dalam mengklasifikasikan sampel-sampel kelas 1. Ini adalah aspek positif dari kinerja model.
3. **Kelas 2**: Kelas ini juga memiliki presisi dan recall sekitar 0.00, dan F1-score sekitar 0.00, yang menunjukkan bahwa model hampir tidak mampu mengidentifikasi atau memprediksi sampel-sampel dari kelas 2.
4. **Akurasi Keseluruhan**: Akurasi keseluruhan model adalah 0.96, yang tinggi. Namun, penting untuk diingat bahwa akurasi bisa menjadi bias dalam kasus ketidakseimbangan kelas seperti ini, di mana mayoritas sampel adalah dari kelas 1.

**Kesimpulan**:

* Model XGBoost Classifier memiliki kinerja yang sangat baik dalam mengidentifikasi kelas 1, dengan presisi, recall, dan F1-score yang tinggi. Ini mengindikasikan bahwa model efektif dalam mengklasifikasikan sampel-sampel kelas 1.
* Namun, model sepertinya tidak efektif dalam mengidentifikasi kelas 0 dan 2, yang terlihat dari presisi, recall, dan F1-score yang sangat rendah untuk kelas-kelas ini. Ini adalah aspek negatif dari kinerja model.
* Akurasi keseluruhan yang tinggi mungkin menyesatkan karena ketidakseimbangan kelas. Oleh karena itu, perlu dipertimbangkan strategi seperti penanganan ketidakseimbangan kelas atau eksplorasi model lain jika kelas 0 dan 2 juga penting dalam analisis Anda.
* Dalam keseluruhan, kinerja model XGBoost Classifier dalam konteks ini tergantung pada pentingnya kelas-kelas tertentu dalam analisis Anda. Jika kelas 1 adalah fokus utama Anda dan kelas 0 dan 2 adalah minoritas atau kurang penting, maka model XGBoost Classifier mungkin cocok. Namun, jika semua kelas memiliki arti penting, perlu pertimbangan lebih lanjut dalam perbaikan kinerja model terhadap kelas minoritas.

# Light Gradient Boosting Machine Classifier

# Light Gradient Boosting Machine Classifier  
# Import library yang dibutuhkan  
import pandas as pd  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
import lightgbm as lgb  
from sklearn.metrics import classification\_report  
  
# Load dataset (gantilah 'nama\_file.csv' dengan nama file dataset Anda)  
data = pd.read\_csv('111.csv')  
  
# Pisahkan fitur (X) dan target (y)  
X = data[['JK', 'JW', 'PLAFOND', 'SALDO\_DEBET', 'JAMINAN']]  
y = data['KOLEKTIBILITAS']  
  
# Bagi dataset menjadi data pelatihan dan data pengujian  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Konversi dataset ke dalam format Dataset LightGBM  
train\_data = lgb.Dataset(X\_train, label=y\_train)  
  
# Parameter untuk model LightGBM  
params = {  
 'objective': 'binary',  
 'metric': 'binary\_error',  
 'boosting\_type': 'gbdt',  
 'num\_leaves': 31,  
 'learning\_rate': 0.05,  
 'feature\_fraction': 0.9  
}  
  
# Inisialisasi model LightGBM Classifier  
lgb\_classifier = lgb.train(params, train\_data, num\_boost\_round=100)  
  
# Melakukan prediksi  
y\_pred\_prob = lgb\_classifier.predict(X\_test, num\_iteration=lgb\_classifier.best\_iteration)  
y\_pred = [1 if pred > 0.5 else 0 for pred in y\_pred\_prob]  
  
# Menampilkan laporan klasifikasi  
classification\_rep = classification\_report(y\_test, y\_pred)  
print("Laporan Klasifikasi:\n", classification\_rep)

[LightGBM] [Info] Number of positive: 660, number of negative: 20  
[LightGBM] [Warning] Auto-choosing col-wise multi-threading, the overhead of testing was 0.000050 seconds.  
You can set `force\_col\_wise=true` to remove the overhead.  
[LightGBM] [Info] Total Bins 313  
[LightGBM] [Info] Number of data points in the train set: 680, number of used features: 5  
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.970588 -> initscore=3.496508  
[LightGBM] [Info] Start training from score 3.496508  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf  
Laporan Klasifikasi:  
 precision recall f1-score support  
  
 -1 0.00 0.00 0.00 2  
 0 0.00 0.00 0.00 2  
 1 0.98 0.99 0.98 167  
  
 accuracy 0.96 171  
 macro avg 0.33 0.33 0.33 171  
weighted avg 0.95 0.96 0.96 171

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))  
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/\_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero\_division` parameter to control this behavior.  
 \_warn\_prf(average, modifier, msg\_start, len(result))

Hasil laporan klasifikasi untuk model XGBoost Classifier yang di berikan adalah sebagai berikut:

**Laporan Klasifikasi**:

1. **Kelas -1 dan 0**: Kedua kelas ini memiliki presisi dan recall sekitar 0.00, yang menunjukkan bahwa model XGBoost Classifier hampir tidak mampu mengidentifikasi atau memprediksi sampel-sampel dari kedua kelas ini. Ini mengindikasikan kinerja yang sangat buruk dalam mengklasifikasikan sampel-sampel dari kelas -1 dan 0.
2. **Kelas 1**: Kelas ini memiliki presisi sekitar 0.98, recall sekitar 0.99, dan F1-score sekitar 0.98, yang mengindikasikan bahwa model XGBoost Classifier sangat baik dalam mengklasifikasikan sampel-sampel kelas 1. Ini adalah aspek positif dari kinerja model.
3. **Akurasi Keseluruhan**: Akurasi keseluruhan model adalah 0.96, yang tinggi. Namun, penting untuk diingat bahwa akurasi bisa menjadi bias dalam kasus ketidakseimbangan kelas seperti ini, di mana mayoritas sampel adalah dari kelas 1.

**Kesimpulan**:

* Model XGBoost Classifier memiliki kinerja yang sangat baik dalam mengidentifikasi kelas 1, dengan presisi, recall, dan F1-score yang tinggi. Ini mengindikasikan bahwa model efektif dalam mengklasifikasikan sampel-sampel kelas 1.
* Namun, model sepertinya tidak efektif dalam mengidentifikasi kelas -1 dan 0, yang terlihat dari presisi dan recall yang sangat rendah untuk kelas-kelas ini. Ini adalah aspek negatif dari kinerja model.
* Akurasi keseluruhan yang tinggi mungkin menyesatkan karena ketidakseimbangan kelas. Oleh karena itu, perlu dipertimbangkan strategi seperti penanganan ketidakseimbangan kelas atau eksplorasi model lain jika kelas -1 dan 0 juga penting dalam analisis Anda.
* Dalam keseluruhan, kinerja model XGBoost Classifier dalam konteks ini tergantung pada pentingnya kelas-kelas tertentu dalam analisis Anda. Jika kelas 1 adalah fokus utama Anda dan kelas -1 dan 0 adalah minoritas atau kurang penting, maka model XGBoost Classifier mungkin cocok. Namun, jika semua kelas memiliki arti penting, perlu pertimbangan lebih lanjut dalam perbaikan kinerja model terhadap kelas minoritas.

# Kesimpulan

Dalam analisis hasil dari 7 model klasifikasi berdasarkan kategorisasi KOLEKTIBILITAS ('Lancar', 'Macet', 'Diragukan'), kita akan mempertimbangkan beberapa metrik evaluasi utama, yaitu presisi (precision), recall, dan f1-score, serta akurasi secara umum. Tujuan kita adalah untuk memahami sejauh mana setiap model mampu mengklasifikasikan KOLEKTIBILITAS dengan baik. Berikut adalah hasil analisis dan kesimpulan:

**Logistic Regression Classifier:**

* Klasifikasi KOLEKTIBILITAS 'Lancar' memiliki presisi, recall, dan f1-score yang sangat tinggi (0.98, 1.00, 0.99), serta akurasi 0.98.
* Klasifikasi KOLEKTIBILITAS 'Macet' dan 'Diragukan' memiliki presisi dan recall yang rendah, bahkan mendekati 0.
* Model ini baik dalam mengidentifikasi KOLEKTIBILITAS 'Lancar' tetapi tidak efektif dalam mengklasifikasikan yang lain.

**Support Vector Machine (SVM) Classifier:**

* Hasilnya serupa dengan Logistic Regression. Klasifikasi KOLEKTIBILITAS 'Lancar' memiliki presisi, recall, dan f1-score yang sangat tinggi (0.98, 1.00, 0.99), serta akurasi 0.98.
* Klasifikasi KOLEKTIBILITAS 'Macet' dan 'Diragukan' juga memiliki presisi dan recall yang rendah, mendekati 0.
* Seperti Logistic Regression, model ini efektif dalam mengidentifikasi KOLEKTIBILITAS 'Lancar' tetapi kurang baik dalam mengklasifikasikan yang lain.

**K-Neighbors Classifier:**

* Klasifikasi KOLEKTIBILITAS 'Lancar' memiliki presisi, recall, dan f1-score yang tinggi (0.98, 0.99, 0.99), serta akurasi 0.97.
* Model ini efektif dalam mengidentifikasi KOLEKTIBILITAS 'Lancar' dan memiliki hasil yang lebih baik daripada Logistic Regression dan SVM.
* Namun, seperti model sebelumnya, klasifikasi KOLEKTIBILITAS 'Macet' dan 'Diragukan' memiliki presisi dan recall yang rendah, mendekati 0.

**Decision Tree Classifier:**

* Model ini memiliki hasil yang kurang baik dibandingkan dengan tiga model sebelumnya. Klasifikasi KOLEKTIBILITAS 'Lancar' memiliki presisi, recall, dan f1-score yang tinggi (0.98, 0.99, 0.98), dengan akurasi 0.96.
* Klasifikasi KOLEKTIBILITAS 'Macet' dan 'Diragukan' juga memiliki presisi dan recall yang rendah.

**Random Forest Classifier, XGBoost Classifier, Light Gradient Boosting Machine Classifier:**

* Ketiga model ini memberikan hasil yang serupa dengan Decision Tree Classifier, dengan kinerja yang kurang baik dalam mengklasifikasikan KOLEKTIBILITAS selain 'Lancar'.

**Kesimpulan:** Berdasarkan hasil di atas, Logistic Regression, SVM, dan K-Neighbors Classifier adalah model-model yang memiliki performa relatif lebih baik dalam mengidentifikasi KOLEKTIBILITAS 'Lancar'. Namun, semua model memiliki kesulitan dalam mengklasifikasikan KOLEKTIBILITAS lainnya ('Macet' dan 'Diragukan'), dengan presisi dan recall yang rendah. Oleh karena itu, dalam konteks ini, tidak ada model yang secara konsisten baik dalam mengklasifikasikan semua kategori KOLEKTIBILITAS. Pemilihan model terbaik harus mempertimbangkan prioritas bisnis dan tingkat kesalahan yang dapat diterima dalam masing-masing kategori KOLEKTIBILITAS.