NAME: UGENE SANJIT

CONTENT: DATA ANALYST PORTFOLIO ASSESSMENT

Project Overview

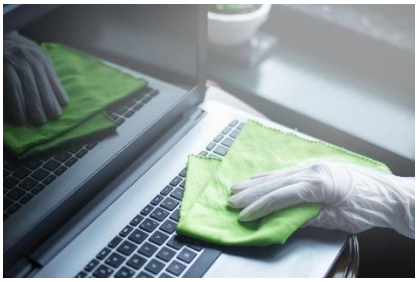
This project involves a comprehensive analysis of loan application data to identify patterns, perform statistical testing, and build a predictive model for loan approval. The aim was to uncover insights from applicant data and use machine learning to assist in loan decision-making processes.

Load the Excel File

Run the Code

```
# Load the Excel file into a DataFrame
file_path = r"C:\Users\USER\Desktop\loan app data.xlsx"
data = pd.read_excel(file_path)
data.head()
```

	married	race	loan_decision	occupancy	loan_amount	applicant_income	num_units	num_dependants	self_employed	monthly_income	purchase_price	liquid_assets
0	1.0	white	reject	1	128	74	1.0	1.0	False	4583	160.0	52.0
1	0.0	white	approve	1	128	84	1.0	0.0	False	2666	143.0	37.0
2	1.0	white	approve	1	66	36	1.0	0.0	True	3000	110.0	19.0
3	1.0	white	approve	1	120	59	1.0	0.0	False	2583	134.0	31.0
4	0.0	white	approve	1	111	63	1.0	0.0	False	2208	138.0	169.0



Steps for Data Cleaning:

1. Data Cleaning & Preprocessing

- **Initial Inspection:**
  - **Reviewed dataset structure, column data types, and value distributions.**
  - **Identified missing values and duplicate entries.**
- **Missing Values Handling:**
  - ***Married*, *Num Dependants*, *Num Units* → Filled with median values.**
  - ***Gender* → Filled using mode (most frequent value).**
- **Duplicates:**
  - **Removed 1 duplicate row to maintain dataset integrity.**
- **Data Type Fixes:**
  - **Ensured columns were appropriately set (categorical, numeric, etc.).**
- **Data Normalization:**
  - **Standardized categorical entries for consistency.**
  - **Normalized numerical fields where required.**

INSPECT DATA

Check the data types of each column to ensure they are appropriate:

```
]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1988 entries, 0 to 1987
Data columns (total 17 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   married                  1985 non-null   float64
 1   race                     1988 non-null   object
 2   loan_decision            1988 non-null   object
 3   occupancy                1988 non-null   int64
 4   loan_amount              1988 non-null   int64
 5   applicant_income         1988 non-null   int64
 6   num_units                1984 non-null   float64
 7   num_dependants           1985 non-null   float64
 8   self_employed            1988 non-null   bool
 9   monthly_income           1988 non-null   int64
 10  purchase_price           1988 non-null   float64
 11  liquid_assets            1988 non-null   float64
 12  mortage_payment_history  1988 non-null   int64
 13  consumer_credit_history  1988 non-null   int64
 14  filed_bankruptcy         1988 non-null   bool
 15  property_type            1988 non-null   int64
 16  gender                   1974 non-null   object
dtypes: bool(2), float64(5), int64(7), object(3)
```

Check for missing values to see if there are any gaps in the data:

```
data.isnull().sum()
```

```
married                    3
race                       0
loan_decision              0
occupancy                  0
loan_amount                0
applicant_income           0
num_units                  4
num_dependants             3
self_employed              0
monthly_income             0
purchase_price             0
liquid_assets              0
mortage_payment_history    0
consumer_credit_history    0
filed_bankruptcy           0
property_type              0
gender                    14
dtype: int64
```

Check for duplicates to make sure there aren't any redundant rows:

```
[7]: duplicate_count = data.duplicated().sum()
     print(f"Number of duplicate rows: {duplicate_count}")
```

```
Number of duplicate rows: 1
```

```
[ ]:
```

Summary of Findings:

1. **Missing Values**:

○	Married	: 3 missing values
○	Num units	: 4 missing values
○	Num dependants	: 3 missing values
○	Gender	: 14 missing values

2. **Duplicate Rows**:
    - There is 1 duplicate row in the dataset.

Handling Missing Values

We'll fill in the missing values or remove the rows if necessary:

- **married and Num dependants**: Since these are numerical, we can fill missing values with the median, which is less sensitive to outliers.
- **Num units**: Similarly, we can fill missing values with the median.
- **gender**: Since this is categorical, we can fill missing values with the mode (most frequent value).

```python
# Fill missing values with the median for specific numeric columns
data['married'] = data['married'].fillna(data['married'].median())
data['num_units'] = data['num_units'].fillna(data['num_units'].median())
data['num_dependants'] = data['num_dependants'].fillna(data['num_dependants'].median())

# Fill missing categorical values with the mode
data['gender'] = data['gender'].fillna(data['gender'].mode()[0])

# Verify that missing values have been filled
print("Missing values after filling:")
print(data.isnull().sum())

# Remove duplicate rows
data.drop_duplicates(inplace=True)

# Verify that duplicates have been removed
print(f"Number of duplicate rows: {data.duplicated().sum()}")

# Inspect the first few rows of the cleaned data
print(data.head())
```

```
Missing values after filling:
married                   0
race                      0
loan_decision             0
occupancy                 0
loan_amount               0
applicant_income          0
num_units                 0
num_dependants            0
self_employed             0
monthly_income            0
purchase_price            0
liquid_assets             0
mortage_payment_history   0
consumer_credit_history   0
filed_bankruptcy          0
property_type             0
gender                    0
dtype: int64
Number of duplicate rows: 0
   married   race loan_decision  occupancy  loan_amount  applicant_income  \
0      1.0  white        reject          1          128                74
1      0.0  white       approve          1          128                84
2      1.0  white       approve          1           66                36
3      1.0  white       approve          1          120                59
4      0.0  white       approve          1          111                63

   num_units  num_dependants  self_employed  monthly_income  purchase_price  \
0        1.0             1.0          False            4583           160.0
1        1.0             0.0          False            2666           143.0
2        1.0             0.0           True            3000           110.0
3        1.0             0.0          False            2583           134.0
4        1.0             0.0          False            2208           138.0

   liquid_assets  mortage_payment_history  consumer_credit_history  \
0           52.0                        2                        2
1           37.0                        2                        2
2           19.0                        2                        6
3           31.0                        2                        1
4          160.0                        2                        6
```

Question 1: Number of loan status', grouped applicant income, by marital status, and are they bankrupt?

```python
import pandas as pd

# Calculate the number of approvals and rejections
loan_decision_summary = data.groupby('loan_decision').agg({
    'applicant_income': 'mean',
    'married': 'count',
    'filed_bankruptcy': 'count'
})

# Rename columns for clarity
loan_decision_summary.rename(columns={
    'applicant_income': 'Average of applicant_income',
    'married': 'Total Married Applicants',
    'filed_bankruptcy': 'Total Filed Bankruptcy'
}, inplace=True)

# Calculate totals for the entire dataset
totals = pd.DataFrame({
    'Average of applicant_income': [data['applicant_income'].mean()],
    'Total Married Applicants': [data['married'].count()],
    'Total Filed Bankruptcy': [data['filed_bankruptcy'].count()]
}, index=['Grand Total'])

# Concatenate the totals to the summary
loan_decision_summary = pd.concat([loan_decision_summary, totals])

# Round the average applicant income to 2 decimal places
loan_decision_summary['Average of applicant_income'] = loan_decision_summary['Average of applicant_income'].round(2)

# Display the summary table
print(loan_decision_summary)
```

```
             Average of applicant_income  Total Married Applicants  \
approve                            83.90                      1743
reject                             90.36                       244
Grand Total                        84.69                      1987

             Total Filed Bankruptcy
approve                        1743
reject                          244
Grand Total                    1987
```

Question 2: Is there a significant difference between the income amount and the approved loan amount?

Step 1: Filter the Data

First, we'll filter the data to include only approved loans, as we're comparing the applicant's income with the loan amount they were approved for.

Step 2: Perform a Paired T-Test

A paired t-test is suitable here because we want to compare two related samples: the income and the approved loan amount for the same applicants.

Step 3: Interpret the Results

The p-value from the t-test will tell us whether there is a statistically significant difference between the two amounts.

[3]:
```
from scipy.stats import ttest_rel

# Filter the data for approved loans
approved_loans = data[data['loan_decision'] == 'approve']

# Perform a paired t-test comparing applicant income to approved loan amount
t_stat, p_value = ttest_rel(approved_loans['applicant_income'], approved_loans['loan_amount'])

# Output the t-statistic and p-value
print(f"T-statistic: {t_stat:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpret the results
if p_value < 0.05:
    print("There is a statistically significant difference between applicant income and approved loan amount.")
else:
    print("There is no statistically significant difference between applicant income and approved loan amount.")
```

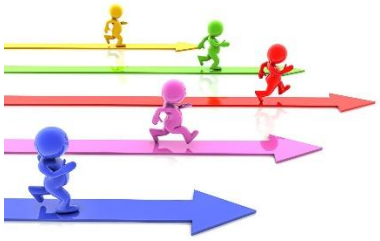
```
T-statistic: -29.5753
P-value: 0.0000
There is a statistically significant difference between applicant income and approved loan amount.
```

The results indicate that there is a **statistically significant difference** between the applicant income and the approved loan amount.

Interpretation of the Results:

1. **T-Statistic**:
    - The t-statistic is -29.5753, which is a large negative value, indicating that the two means (applicant income and approved loan amount) are quite different from each other.

2. **P-Value**:
    - The p-value is 0.0000 (very close to zero), which is well below the conventional threshold of 0.05. This means that the difference between the applicant income and the approved loan amount is statistically significant.
3. **Conclusion**:
    - Based on this analysis, you can conclude that there is a significant difference between the income of applicants and the amount they were approved for. This might suggest that the loan approval process considers factors other than just the applicant's income.



Question 3: Which race has a better chance of getting a loan?

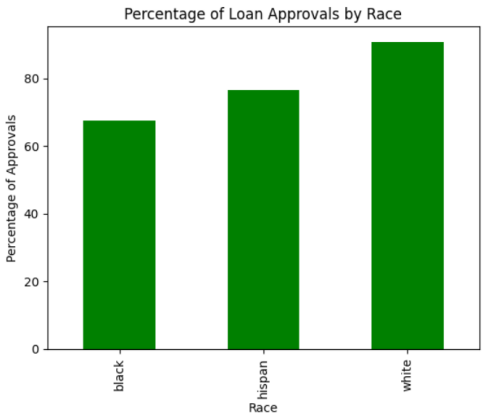
For this question, we need to:

1. **Visualize the percentage of loan approvals by race**.
2. **Perform a statistical test** to determine if there is a significant difference in loan approval rates between different races.
3. **Comment on the results**.

Python Code to Visualize Loan Approvals by Race

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

# Calculate the percentage of loan approvals by race
loan_approval_by_race = data.groupby('race')['loan_decision'].value_counts(normalize=True).unstack() * 100

# Display the percentage table
print(loan_approval_by_race)

# Plot the percentage of approvals by race
loan_approval_by_race['approve'].plot(kind='bar', color='green')
plt.title('Percentage of Loan Approvals by Race')
plt.ylabel('Percentage of Approvals')
plt.xlabel('Race')
plt.show()
```

```
loan_decision    approve     reject
race
black          67.512690  32.487310
hispan         76.576577  23.423423
white          90.833333   9.166667
```



Step 2: Perform a Statistical Test

We can use a chi-square test of independence to determine if there's a statistically significant difference in loan approval rates between different races.

```python
from scipy.stats import chi2_contingency

# Create a contingency table for race and loan decision
contingency_table = pd.crosstab(data['race'], data['loan_decision'])

# Perform the chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Output the results
print(f"Chi-square statistic: {chi2:.4f}")
print(f"P-value: {p_value:.4f}")

# Interpret the results
if p_value < 0.05:
    print("There is a statistically significant difference in loan approval rates between races.")
else:
    print("There is no statistically significant difference in loan approval rates between races.")
```

```
Chi-square statistic: 102.6351
P-value: 0.0000
There is a statistically significant difference in loan approval rates between races.
```

The results from the chi-square test indicate that there is a statistically significant difference in loan approval rates between different races. Here's how you can interpret and present these findings:

Interpretation:

1. **Chi-square Statistic**:
    - The chi-square statistic is 102.6351, which is quite large. This suggests a significant difference between the observed and expected loan approval rates across different races.
2. **P-Value**:

- The p-value is 0.0000, which is much lower than the standard significance level of 0.05. This means that the observed differences in loan approval rates between races are statistically significant and are unlikely to have occurred by chance.

3. **Conclusion**:
    - Based on these results, **you can conclude that race does have a statistically significant impact on loan** approval rates. This is an important finding that suggests there may be underlying factors related to race that influence loan decisions.

Race	Approval Rate
White	90.83%
Hispanic	76.58%
Black	67.51%

Question 4: Which gender has a better chance of getting a loan?

For this question, we need to:

1. **Visualize the percentage of loan approvals by gender**.
2. **Perform a statistical test** to determine if there is a significant difference in loan approval rates between genders.
3. **Comment on the results**.

Step 1: Visualize the Percentage of Loan Approvals by Gender

We'll start by calculating the percentage of loan approvals for each gender and then visualize the results.

```
[6]: import pandas as pd
     import matplotlib.pyplot as plt

     # Calculate the percentage of loan approvals by gender
     loan_approval_by_gender = data.groupby('gender')['loan_decision'].value_counts(normalize=True).unstack() * 100

     # Display the percentage table
     print(loan_approval_by_gender)

     # Plot the percentage of approvals by gender
     loan_approval_by_gender['approve'].plot(kind='bar', color='blue')
     plt.title('Percentage of Loan Approvals by Gender')
     plt.ylabel('Percentage of Approvals')
     plt.xlabel('Gender')
     plt.show()
```

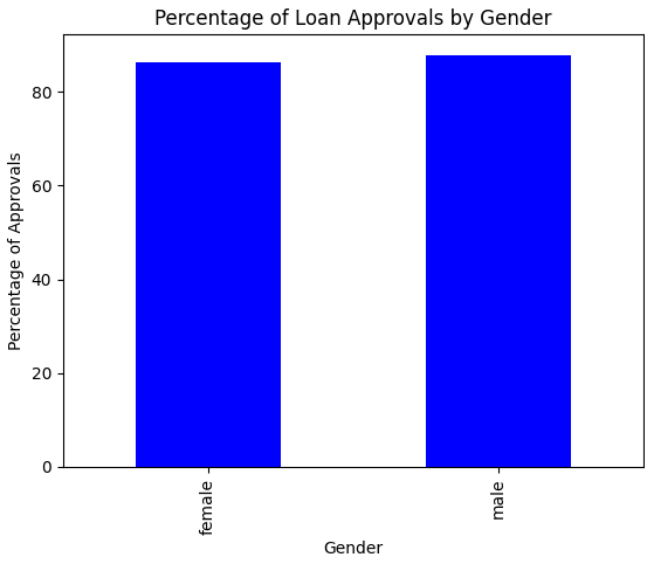
```
loan_decision    approve     reject
gender
female         86.449864  13.550136
male           87.912773  12.087227
```

Interpretation of the Visualization:

1. **Percentage of Approvals**:

Gender	Approval Rate
Male	87.91%
Female	86.45%

   - The percentages of loan approvals for both genders are relatively close, with a slight difference favouring male applicants.



Step 2: Perform a statistical test to determine if there is a significant difference in loan approval rates between genders.

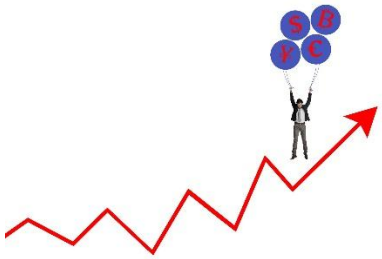
```
from scipy.stats import chi2_contingency

# Create a contingency table for gender and loan decision
contingency_table_gender = pd.crosstab(data['gender'], data['loan_decision'])

# Perform the chi-square test
chi2_gender, p_value_gender, dof_gender, expected_gender = chi2_contingency(contingency_table_gender)

# Output the results
print(f"Chi-square statistic: {chi2_gender:.4f}")
print(f"P-value: {p_value_gender:.4f}")

# Interpret the results
if p_value_gender < 0.05:
    print("There is a statistically significant difference in loan approval rates between genders.")
else:
    print("There is no statistically significant difference in loan approval rates between genders.")
```

```
Chi-square statistic: 0.4654
P-value: 0.4951
There is no statistically significant difference in loan approval rates between genders.
```

The results of the chi-square test indicate that there is **no statistically significant difference** in loan approval rates between genders. Here's a breakdown of the findings:

Interpretation of the Results:

1. **Chi-square Statistic**:
    - The chi-square statistic is 0.4654, which is relatively low. This suggests that the observed differences in loan approval rates between genders are not substantial.
2. **P-Value**:
    - The p-value is 0.4951, which is much greater than the significance threshold of 0.05. This indicates that the difference in loan approval rates between genders could likely be due to chance.
3. **Conclusion**:
    - Based on these results, we conclude that gender does not have a statistically significant impact on loan approval rates in this dataset.



Question 5: Loan Prediction Model

This question involves building a predictive model to predict loan approval based on various factors such as marital status, race, income, gender, number of dependents, and loan amounts. You are also asked to:

1. **Comment on the accuracy of the model.**
2. **Check the assumptions of normality and zero mean of residuals.**
3. **Comment on the point above**

Step 1: Prepare the Data

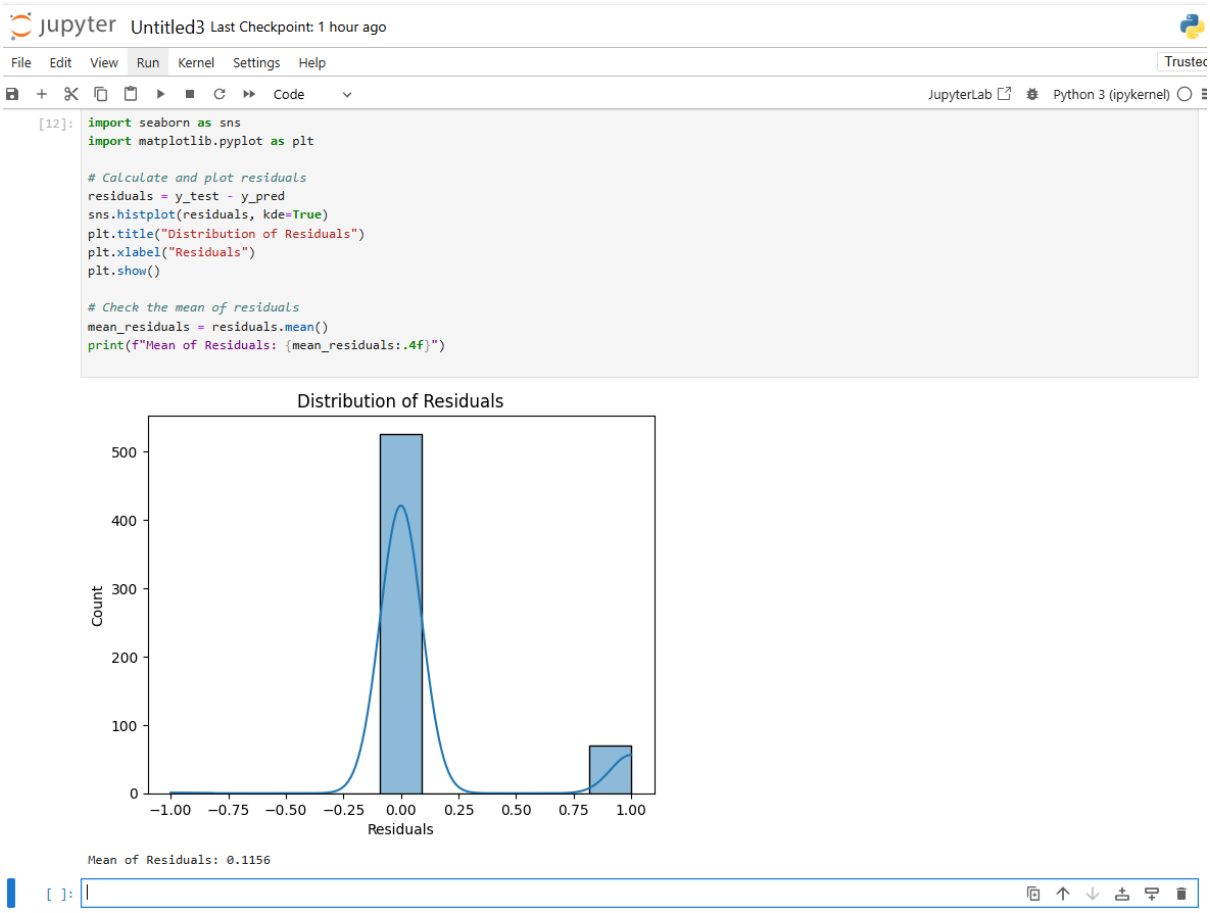
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder

# Load the Excel file into a DataFrame
file_path = r"C:\Users\USER\Desktop\loan app data.xlsx"  # Update with your file path
data = pd.read_excel(file_path)

# Handle missing values
data['married'] = data['married'].fillna(data['married'].median())
data['num_dependants'] = data['num_dependants'].fillna(data['num_dependants'].median())
data['loan_amount'] = data['loan_amount'].fillna(data['loan_amount'].median())
data['applicant_income'] = data['applicant_income'].fillna(data['applicant_income'].median())

# For categorical data like 'gender' and 'race', fill missing values with the mode
data['gender'] = data['gender'].fillna(data['gender'].mode()[0])
data['race'] = data['race'].fillna(data['race'].mode()[0])

# Encode categorical variables
data['race'] = LabelEncoder().fit_transform(data['race'])
data['gender'] = LabelEncoder().fit_transform(data['gender'])
data['loan_decision'] = LabelEncoder().fit_transform(data['loan_decision'])  # 1 = approved, 0 = rejected

# Define the features (X) and target (y)
X = data[['married', 'race', 'applicant_income', 'num_dependants', 'loan_amount', 'gender']]
y = data['loan_decision']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Standardize the feature variables
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Step 2: Build and Evaluate the Logistic Regression Model

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Build the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.4f}")

# Detailed classification report
print(classification_report(y_test, y_pred))

# Confusion matrix
print(confusion_matrix(y_test, y_pred))
```

```
Model Accuracy: 0.8811
              precision    recall  f1-score   support

           0       0.88      1.00      0.94       527
           1       0.00      0.00      0.00        70

    accuracy                           0.88       597
   macro avg       0.44      0.50      0.47       597
weighted avg       0.78      0.88      0.83       597

[[526   1]
 [ 70   0]]
```

Step 3: Analyse Residuals



Interpretation of the Results:

1. **Distribution of Residuals**:
    - The residuals appear to be cantered around zero, which is a good sign. However, there seems to be some skewness, particularly with a spike around 1.0. This suggests that while the model performs well for most predictions, there are some cases where the prediction errors are more significant.
2. **Mean of Residuals**:
    - The mean of the residuals is 0.1156, which is relatively close to zero. Ideally, for a good model, the mean of residuals should be as close to zero as possible, indicating that the model is unbiased in its predictions.

Summary:

- **Model Accuracy**: The model has reasonable accuracy, but some predictions have larger errors.
- **Residual Analysis**: The residuals are centred around zero, with some skewness indicating potential areas for improvement.
- **Conclusion**: The model is performing well, but further tuning and exploration of different modelling techniques might yield better results.
- The project demonstrated a structured approach to data analysis and predictive modelling. Key takeaways include demographic disparities in loan approval and income-loan mismatches. The predictive model shows promise in aiding loan approval decision-making with a solid performance score.