Loan Approval Prediction Using Machine Learning

Project Overview

This project is designed to predict whether a loan will be approved or not using machine learning algorithms. The dataset used is based on various customer features like gender, marital status, income, education, loan amount, and credit history. The objective is to build an accurate model to automate the loan approval process and reduce the dependency on manual review.

Dataset

File Used: Loan dataset(loan.csv)

- Contains 13 columns and 614 rows (after removing nulls).
- Target Variable: Loan_Status (Y/N)
- Features: Categorical and numerical values such as Gender, Married, Education, ApplicantIncome, Credit_History, etc.

Data Preprocessing

Missing Values Handling:

- Filled missing categorical values (Gender, Married, Self_Employed, Dependents, Credit_History) with mode.
- Filled missing numerical values (LoanAmount, Loan_Amount_Term) with mean.
- Used df.fillna(df[column].mode()[0]) pattern and avoided chained assignment warnings.

Feature Engineering:

- Added LoanAmount_log column using np.log1p(LoanAmount) for normalization (optional).
- Removed rows with still missing values after fill operations using df.dropna() (if required).

Label Encoding:

- Encoded categorical variables using LabelEncoder() from sklearn.preprocessing.
- Applied encoding to both feature columns (x) and target variable (y).

Feature Selection

Selected relevant features using:

```
python:
x = df.iloc[:, np.r_[1:5, 9:11, 13:15]].values
y = df.iloc[:, 12].values
```

This includes features like:

- Gender
- Married

- Education
- Self_Employed
- Credit_History
- Property_Area
- LoanAmount
- Loan_Amount_Term

Train-Test Split

Split the dataset:

python:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

Feature Scaling

Used StandardScaler to normalize feature values:

python:

```
ss = StandardScaler()
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

Tools & Technologies

- Python
- Pandas
- NumPy
- Scikit-learn (sklearn)
- Jupyter Notebook

Machine Learning Models Building

Random Forest Classifier: python:

```
from sklearn.ensemble import RandomForestClassifier
rf_clf = RandomForestClassifier()
rf_clf.fit(X_train, y_train)
y_pred = rf_clf.predict(X_test)
```

Gaussian Naive Bayes: python:

```
from sklearn.naive_bayes import GaussianNB
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
y_pred_nb = nb_classifier.predict(X_test)
```

Evaluation Metrics: Used accuracy score to evaluate models:

```
python:
from sklearn import metrics
print("Accuracy of Random Forest:", metrics.accuracy_score(y_test, y_pred))
print("Accuracy of GaussianNB:", metrics.accuracy_score(y_test, y_pred_nb))
```

Conclusion

- The Random Forest model performed with higher accuracy compared to GaussianNB.
- Data preprocessing played a crucial role in ensuring model reliability.
- This project demonstrates an end-to-end pipeline for solving a binary classification problem using real-world financial data.

~ **AKASH KUMAR RAJAK**

~ **Email Id:** akashkumarrajak200@gmail.com

~ **Github:** Link