Introduction

Banks earn a major revenue from lending loans. But it is often associated with risk. The borrowers may default on the loan. To mitigate this issue, the banks have decided to use Machine Learning to overcome this issue. They have collected past data on the loan borrowers & would like you to develop a strong ML Model to classify if any new borrower is likely to default or not.

The dataset is enormous & consists of multiple deterministic factors like borrowe's income, Credit score, loan purpose etc. Can you overcome these factors & build a strong classifier to predict defaulters?

Objective:

- Understand the Dataset & cleanup (if required).
- Build classification model to predict weather the loan borrower will default or not.
- Also fine-tune the hyperparameters & compare the evaluation metrics of various classification algorithms.

Dataset Introduction

LoanID	Age	Income	LoanAmount	CreditScore	MonthsEmployed	NumCreditLines	InterestRate	LoanTerm	DTIRatio	Education	EmploymentType	MaritalStatus	HasMortgage	HasDependents	LoanPurpose	HasCoSigner	Default
I38PQUQS96	56	85994	50587	520	80	4	15.23	36	0.44	Bachelor's	Full-time	Divorced	Yes	Yes	Other	Yes	0
HPSK72WA7R	69	50432	124440	458	15	1	4.81	60	0.68	Master's	Full-time	Married	No	No	Other	Yes	0
C1OZ6DPJ8Y	46	84208	129188	451	26	3	21.17	24	0.31	Master's	Unemployed	Divorced	Yes	Yes	Auto	No	1
V2KKSFM3UN	32	31713	44799	743	0	3	7.07	24	0.23	High School	Full-time	Married	No	No	Business	No	0
EY08JDHTZP	60	20437	9139	633	8	4	6.51	48	0.73	Bachelor's	Unemployed	Divorced	No	Yes	Auto	No	0
A9S62RQ7US	25	90298	90448	720	18	2	22.72	24	0.1	High School	Unemployed	Single	Yes	No	Business	Yes	1
H8GXPAOS71	38	111188	177025	429	80	1	19.11	12	0.16	Bachelor's	Unemployed	Single	Yes	No	Home	Yes	0
0HGZQKJ36W	56	126802	155511	531	67	4	8.15	60	0.43	PhD	Full-time	Married	No	No	Home	Yes	0
1R0N3LGNRJ	36	42053	92357	827	83	1	23.94	48	0.2	Bachelor's	Self-employed	Divorced	Yes	No	Education	No	1
CM9L1GTT2P	40	132784	228510	480	114	4	9.09	48	0.33	High School	Self-employed	Married	Yes	No	Other	Yes	0
IA35XVH6ZO	28	140466	163781	652	94	2	9.08	48	0.23	High School	Unemployed	Married	No	No	Education	No	0
Y8UETC3LSG	28	149227	139759	375	56	3	5.84	36	0.8	PhD	Full-time	Divorced	No	No	Education	Yes	1
RM6QSRHIYP	41	23265	63527	829	87	4	9.73	60	0.45	Master's	Full-time	Divorced	Yes	No	Auto	Yes	0
GX5YQOGROM	53	117550	95744	395	112	4	3.58	24	0.73	High School	Unemployed	Single	No	No	Auto	Yes	0
X0BVPZLDC0	57	139699	88143	635	112	4	5.63	48	0.2	Master's	Part-time	Divorced	No	No	Home	No	0
O5DM5MPPNA	41	74064	230883	432	31	2	5	60	0.89	Master's	Unemployed	Married	Yes	No	Auto	No	0
ZDDRGVTEXS	20	119704	25697	313	49	1	9.63	24	0.28	PhD	Unemployed	Single	Yes	No	Home	No	0
9V0FJW7QPB	39	33015	10889	811	106	2	13.56	60	0.66	Master's	Self-employed	Single	Yes	No	Other	No	0
O1IKKLC69B	19	40718	78515	319	119	2	14	24	0.17	Bachelor's	Self-employed	Divorced	Yes	No	Education	No	1
F7487UU2BF	41	123419	161146	376	65	4	16.96	60	0.39	High School	Self-employed	Single	Yes	No	Other	Yes	0
7ASF0IHRIT	61	30142	133714	429	96	1	15.58	12	0.65	PhD	Part-time	Divorced	No	Yes	Business	No	0
A22KI1B6SE	47	146113	100621	419	55	1	9.32	12	0.38	Bachelor's	Unemployed	Married	Yes	Yes	Business	No	0
1MUSHWD9TW	55	132058	130912	583	48	4	5.82	60	0.47	High School	Unemployed	Married	No	Yes	Business	Yes	0

Fig 1: Dataset

Description of Columns:

Column Name	Description
Loan ID:	Unique number given to loan.
Age:	Age of Person.
Income:	Annual Income of a person.
Loan Amount:	Loan taken by person.
Credit Score:	A credit score is a prediction of your credit behaviour, such as how likely you are to pay loan back on time, based on information from your credit reports.
Months Employed:	From how many months a person being employed.
No. of Credit Lines:	
Interest rate:	An interest rate is the percentage of the principal that a bank charges a borrower.
Loan Term	A loan term is the length of time it takes to repay the loan.
DTI ratio:	A debt-to-income (DTI) ratio compares how much you owe each month to how much you earn.
Education:	What is your education.
Employment Type:	Type of employment you have.
Marital Status:	Your marital status, eg. Married or Divorced or Single.
Has Mortgage:	Mortgage means pledging a piece of land, a home, or any type of property to get loan.
Has Dependants:	
Has Co-signer:	A co-signer is a person such as a parent, family member, etc. who adds their information, including income and credit record, to the loan application and pledges to pay back the loan if you`re unable to pay
Default:	Target Column.

Project Pipeline:

1. Data Preprocessing and Exploration:

- **_Data Loading:_ ** The dataset "Loan_default.csv" was loaded into a Pandas DataFrame.

- **_Data Overview:_ ** The basic properties of the dataset, such as shape, column names, and data types, were explored.

- **_Data Cleaning:_ ** Duplicate rows were identified and dropped, and the null values were checked.

2. Exploratory Data Analysis (EDA):

- **_Box Plots:_ ** Visualized the distribution and outliers of numerical features using box plots.

- **_Heatmap_: ** Explored the correlation between numerical features using a heatmap.

- **_Histograms:_ ** Examined the distribution of individual features using histograms.

3. Feature Engineering:

- _Label Encoding:_ - Converted categorical variables using Label Encoding. This step is essential for models that require numerical input.

4. Handling Class Imbalance:

- _Random Oversampling:_ - Balanced the dataset using the RandomOverSampler from the imbalanced-learn library.

5. Train-Test Split:

- The dataset was split into training and testing sets.

6. Data Scaling:

- *Standard Scaling:* Applied StandardScaler to scale the features to a standard Gaussian distribution.

7. Model Training:

- Trained five different models: Random Forest, Naive Bayes, Decision Tree, Logistic Regression, and Support Vector Machine.

8. Model Evaluation:

- Evaluated each model using accuracy, confusion matrix, and classification report.

9. Results:

Each model's performance metrics were printed, allowing for a comparison of their effectiveness.

This code appears to cover various aspects of a machine learning project, including data exploration, preprocessing, model training, and evaluation. The inclusion of multiple models allows for a comparative analysis of their performance. Further improvements could involve hyperparameter tuning, feature selection, and a more in-depth analysis of model results.

Code Explain:

Let us go through the code line by line:

```
# Importing necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from imblearn.over_sampling import RandomOverSampler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

The code begins by importing the required libraries for data manipulation, visualization, preprocessing, and machine learning models.

```
# Loading the dataset
data = pd.read_csv(r"C:\Users\q\Desktop\ProjectPawanSir\Loan_default.csv")
df = data.copy()
```

The dataset is loaded from a CSV file, and a copy is created to ensure that modifications don't affect the original data.

```
# Checking for null values in the dataset
df.isnull().sum()
```

This line checks for null values in the dataset.

Removing duplicates from the dataset

df.drop_duplicates(inplace=True)

Duplicates are removed from the dataset.

Displaying the shape of the dataset

df.shape

This line displays the shape (number of rows and columns) of the dataset.

Displaying the column names in the dataset

df.columns

#This line displays the names of the columns in the dataset.

Displaying information about the dataset (data types, non-null counts)

df.info()

This line displays information about the dataset, including data types and non-null counts.

Displaying descriptive statistics of the dataset

df.describe()

#Descriptive statistics (mean, std, min, max, etc.) of the dataset are displayed.

Box plot for selected numerical columns

```
columns = df.columns
fig, axs = plt.subplots(5, 2, figsize=(10, 10))

for i in range(1, 10):  # Assuming columns 1 to 9 are numerical columns
    sns.boxplot(df.iloc[:, i], ax=axs[(i - 1) // 2, (i - 1) % 2], orient='h').set_title(columns[i])
fig.tight_layout()
plt.show()
```

```
# Displaying a correlation heatmap
plt.figure(figsize=(10, 8))
corr = df[['Age', 'Income', 'LoanAmount', 'CreditScore', 'MonthsEmployed',
        'NumCreditLines', 'InterestRate', 'LoanTerm', 'DTIRatio', 'Default']].corr()
sns.heatmap(corr, vmin=-0.25, annot=True, vmax=0.6)
plt.title('Correlation Heatmap')
```

These lines create box plots for selected numerical columns and a correlation heatmap using seaborn.

```
# Histogram plot for selected numerical columns
# Get column names
columns = df.columns
# Set up the subplots
fig, axs = plt.subplots(5, 2, figsize=(10, 10))
# Flatten the axs array for easy iteration
axs_flat = axs.flatten()
# Loop through columns and create histograms
for i in range(10):  # Assuming you want to plot the first 10 columns
   sns.histplot(df.iloc[:, i+1], ax=axs_flat[i], kde=True).set_title(columns[i+1])
# Adjust layout
fig.tight_layout()
# Show the plots
plt.show()
```

```
# Label encoding for categorical columns using a loop
label_encoder = LabelEncoder()
columns_to_encode = df[["Education", "EmploymentType", "MaritalStatus", "HasMortgage",
                        "HasDependents", "LoanPurpose", "HasCoSigner"]]
for i in columns_to_encode:
    df[i] = label_encoder.fit_transform(df[i])
df
```

#certain categorical columns using a loop.

```
# Dropping the 'LoanID' column
df.drop(["LoanID"], axis=1, inplace=True)
```

The 'LoanID' column is dropped from the dataset.

```
# Displaying the correlation heatmap after label encoding
plt.figure(figsize=(30, 30))
corr = df.corr()
sns.heatmap(corr, vmin=-0.25, annot=True, vmax=0.6)
plt.title('Correlation Heatmap')
```

A correlation heatmap is displayed after label encoding.

```
# Checking the distribution of values in the 'Default' column
df["Default"].value_counts()
```

#The distribution of values in the 'Default' column is checked to determine if the dataset is #balanced or imbalanced.

```
# Applying random oversampling to balance the dataset
X = df
y = df["Default"]
X.drop(["Default"], axis=1, inplace=True)
over_sampler = RandomOverSampler(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = over_sampler.fit_resample(X, y)
Random oversampling is applied to balance the dataset.
```

```
# Splitting the dataset into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=53)
print('x_train:', x_train.shape)
print('y_train:', y_train.shape)
print('x_test:', x_test.shape)
print('y_test:', y_test.shape)
# The dataset is split into training and testing sets.
```

```
# Standard scaling of the dataset
ss = StandardScaler()
x_train = ss.fit_transform(x_train)
x_test = ss.fit_transform(x_test)
The dataset is scaled using StandardScaler.
```

```
# Training a Random Forest classifier
model = RandomForestClassifier()
model = model.fit(x_train, y_train)
y_pred = model.predict(x_test)
#A Random Forest classifier is trained and predictions are made on the test set.
```

```
# Training a Naive Bayes classifier
nb_clf = GaussianNB()
model = nb_clf.fit(x_train, y_train)
y_pred1 = nb_clf.predict(x_test)
#A Gaussian Naive Bayes classifier is trained and predictions are made on the test set.
```

```
# Training a Decision Tree classifier
dt = DecisionTreeClassifier()
model = dt.fit(x_train, y_train)
y_pred2 = dt.predict(x_test)
# A Decision Tree classifier is trained, and predictions are made on the test set.
```

```
# Training a Logistic Regression classifier
Lr = LogisticRegression()
model = Lr.fit(x_train, y_train)
y_pred3 = Lr.predict(x_test)
#A Logistic Regression classifier is trained, and predictions are made on the test set.
```

```
# Training a Support Vector Machine (SVM) classifier
svm_classifier = SVC(kernel='linear', C=1.0)
model = svm_classifier.fit(x_train, y_train)
y_pred4 = svm_classifier.predict(x_test)


#A Support Vector Machine (SVM) classifier with a linear kernel is trained, and predictions
#are made on the test set.
```

```
# Evaluating the Random Forest classifier
accuracy_RF = accuracy_score(y_test, y_pred)
conf_matrix_RF = confusion_matrix(y_test, y_pred)
classification_rep_RF = classification_report(y_test, y_pred)


# Print the evaluation metrics
print("Accuracy:", accuracy_RF)
print("\nConfusion Matrix:")
print(conf_matrix_RF)
print("\nClassification Report:")
print(classification_rep_RF)
```

```
# Evaluating the Naive Bayes classifier
accuracy_NB = accuracy_score(y_test, y_pred1)
conf_matrix_NB = confusion_matrix(y_test, y_pred1)
classification_rep_NB = classification_report(y_test, y_pred1)

# Print the evaluation metrics
print("Accuracy:", accuracy_NB)
print("\nConfusion Matrix:")
print(conf_matrix_NB)
print("\nClassification Report:")
print(classification_rep_NB)
```

```
# Evaluating the Decision Tree classifier
accuracy_DTC = accuracy_score(y_test, y_pred2)
conf_matrix_DTC = confusion_matrix(y_test, y_pred2)
classification_rep_DTC = classification_report(y_test, y_pred2)

# Print the evaluation metrics
print("Accuracy:", accuracy_DTC)
print("\nConfusion Matrix:")
print(conf_matrix_DTC)
print("\nClassification Report:")
print(classification_rep_DTC)
```

```
# Evaluating the Logistic Regression classifier
accuracy_LR = accuracy_score(y_test, y_pred3)
conf_matrix_LR = confusion_matrix(y_test, y_pred3)
classification_rep_LR = classification_report(y_test, y_pred3)

# Print the evaluation metrics
print("Accuracy:", accuracy_SVM)
print("\nConfusion Matrix:")
print(conf_matrix_SVM)
print("\nClassification Report:")
print(classification_rep_SVM)
```

```
# Evaluating the Support Vector Machine (SVM) classifier
accuracy_SVM = accuracy_score(y_test, y_pred4)
conf_matrix_SVM = confusion_matrix(y_test, y_pred4)
classification_rep_SVM = classification_report(y_test, y_pred4)

# Print the evaluation metrics
print("Accuracy:", accuracy_SVM)
print("\nConfusion Matrix:")
print(conf_matrix_SVM)
print("\nClassification Report:")
print(classification_rep_SVM)
```

The classifiers are evaluated using accuracy, confusion matrix, and classification report.

Conclusion:

In conclusion, the provided code implements a comprehensive machine learning pipeline for a binary classification problem related to loan default prediction. The pipeline includes data preprocessing, exploratory data analysis (EDA), feature engineering, handling class imbalance, train-test split, data scaling, model training, and model evaluation.