KIET GROUP OF INSTITUTIONS

Introduction to AI

MSE1

PROBLEM: Credit Score Prediction: Clean and transform financial data to improve credit risk assessment models.

NAME: NIVEDITA NIGAM

BRANCH: CSEAIML-B

ROLL NO.: 202401100400133

Methodology (Continued)

Data Cleaning and Preprocessing

1. **Handling Missing Values**: Missing values were imputed using the mean for numerical features.

2. **Outlier Detection**: Outliers were capped at the 99th percentile to reduce their impact on the model.

3. **Feature Engineering**: New features like `Debt-to-Income Ratio` were created to improve model performance.

Model Selection

The XGBoost algorithm was chosen for its ability to handle imbalanced datasets and its high performance in classification tasks.

Model Training

The dataset was split into training and testing sets (80% training, 20% testing). The model was trained on the training set and evaluated on the testing set.

Evaluation Metrics

The model's performance was evaluated using:

- **Accuracy**: Percentage of correct predictions.

- **ROC-AUC Score**: Measures the model's ability to distinguish between classes.

- **Classification Report**: Includes precision, recall, and F1-score.

Code

```python
# Import required libraries
import pandas as pd
from xgboost import XGBClassifier
```

```
import joblib
import matplotlib.pyplot as plt

# Load the pre-trained model
try:
    model = joblib.load('credit_score_model.pkl')
except FileNotFoundError:
    print("Error: Model not found. Please train the model first.")
    exit()

# Take input from the user
print("Enter your financial details to predict credit risk:")
age = int(input("Age: "))
income = float(input("Annual Income (₹): "))
debt = float(input("Total Debt (₹): "))
credit_utilization = float(input("Credit Utilization (%): "))
```

```
payment_history = int(input("Months of On-Time Payments: "))
num_credit_accounts = int(input("Number of Credit Accounts: "))
loan_amount = float(input("Loan Amount (₹): "))

# Calculate Debt-to-Income Ratio
debt_to_income_ratio = debt / income

# Create a DataFrame from user input
user_data = pd.DataFrame({
    'Age': [age],
    'Income (₹)': [income],
    'Credit Utilization (%)': [credit_utilization],
    'Payment History (months)': [payment_history],
    'Number of Credit Accounts': [num_credit_accounts],
    'Loan Amount (₹)': [loan_amount],
    'Debt-to-Income Ratio': [debt_to_income_ratio]
})
```

```
# Predict credit risk
prediction = model.predict(user_data)
result = "High Risk (Default)" if prediction[0] == 1 else "Low Risk (No Default)"

# Display the result
print(f"\nCredit Risk Prediction: {result}")

# Plot feature importance
feature_importance = model.feature_importances_
features = user_data.columns

# Create a DataFrame for visualization
importance_df = pd.DataFrame({'Feature': features, 'Importance': feature_importance})
importance_df = importance_df.sort_values(by='Importance', ascending=False)

# Plot the graph
```

```
plt.figure(figsize=(10, 6))
plt.barh(importance_df['Feature'], importance_df['Importance'], color='skyblue')
plt.xlabel('Importance')
plt.title('Feature Importance for Credit Risk Prediction')
plt.gca().invert_yaxis()  # Invert y-axis to show the most important feature at the top
plt.show()
```

#OUTPUTS