Bureau ID- Assignment

By Sarang Deb Saha

1. Approach Taken

a. Data Loading and Overview

I began by loading the training and test datasets from CSV files using the read_csv() function of the pandas library.

b. Data Preprocessing

Data preprocessing involved several steps:

I. **Data type Conversion**: The ‘APPLICATION LOGIN DATE’ was converted to datetime format to extract additional features like month, day, and day of the week. Then the ‘Cibil Score’ was converted to numeric from object type.
II. **Handling Missing Values**:
    A. Categorical missing values were filled with 'Unknown'.
    B. Boolean features were filled with 0 and converted to integers.
    C. Numerical missing values were imputed with the mean of respective columns.
III. **Feature Engineering**: New features derived from dates were added to the dataset.
IV. **Label Encoding**: Categorical variables were encoded to integers for compatibility with machine learning models.

c. Model Training

7 models were trained and evaluated, including:

- Logistic Regression
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Naive Bayes
- Random Forest
- Decision Tree
- Extra Trees Classifier

d. Model Evaluation

Models were evaluated based on accuracy, precision, recall, and F1-score. The training data was split into training and validation sets (80-20 split) for evaluation purposes.

2. Insights and Conclusions from Data

a. Model Performance

- **Random Forest** emerged as the top-performing model with an accuracy of 88.35% on the validation set and balanced performance across both classes.
- **Decision Tree** and **Extra Trees Classifier** also demonstrated strong performance, with accuracies of 86.55% and 85.55% respectively.
- **Logistic Regression**, **SVM**, **KNN**, and **Naive Bayes** struggled, particularly with class 1 (APPROVED), showing low recall and precision for this class.

b. Class Imbalance

The models struggled with detecting the positive class (1). This indicates a possible class imbalance issue, where the number of instances for class 1 is significantly lower compared to class 0.

c. Feature Impact

The feature engineering steps, including date extraction and label encoding, contributed to a better model understanding of the data. This preprocessing step was crucial for the performance of the machine learning models.

3. Performance on Training Dataset

a. Metrics

The following metrics were used to evaluate model performance:

- **Accuracy**: The proportion of correctly classified instances out of the total instances.
- **Precision**: The proportion of true positive predictions out of all positive predictions.
- **Recall**: The proportion of true positive predictions out of all actual positive instances.
- **F1-Score**: The harmonic mean of precision and recall, providing a single metric to evaluate the model's performance, especially when dealing with class imbalance.

b. Results

Model	Accuracy	Precision (Class 0)	Recall (Class 0)	F1-Score (Class 0)	Precision (Class 1)	Recall (Class 1)	F1-Score (Class 1)
Logistic Regression	66.35%	0.66	1.00	0.80	0.00	0.00	0.00
Support Vector Machine	66.35%	0.66	1.00	0.80	0.00	0.00	0.00

K-Nearest Neighbors	61.80%	0.67	0.82	0.74	0.38	0.22	0.28
Naive Bayes	66.35%	0.66	1.00	0.80	0.00	0.00	0.00
Random Forest	88.35%	0.92	0.91	0.91	0.82	0.84	0.83
Decision Tree	86.55%	0.90	0.90	0.90	0.80	0.81	0.80
Extra Trees Classifier	85.55%	0.94	0.83	0.88	0.73	0.90	0.81

4. Conclusion

Based on the evaluation metrics, **Random Forest** was identified as the best-performing model, offering a high accuracy and balanced performance across both classes. Despite this, class imbalance issues should be addressed to further improve the model's performance on the minority class.

Next Steps:

1. **Hyperparameter Tuning**: Fine-tune the Random Forest, Decision Tree and Extra Trees Classification models to further enhance performance.

   This may be achieved using GrindSearchCV to find optimal values for hyperparameters like-

   a. n_estimators
   b. max_depth
   c. max_features etc.
2. **Class Imbalance Techniques**: Implement techniques such as SMOTE or class weighting to improve recall and precision for the positive class.
3. **Cross-Validation**: Apply cross-validation to ensure robust model performance and avoid overfitting.

This comprehensive approach ensures that the chosen model is robust and generalizes well to unseen data, providing valuable predictions for the application status.