Credit Card Approval dataset from the UCI Machine Learning Repository

Name: Sahand Namvar
Group Number: 15

1) What are we trying to learn about the credit card data?

The aim of this project is to leverage machine learning techniques to automate the credit card application approval process. The dataset, sourced from the UCI Machine Learning Repository, comprises both numerical and categorical features, including variables like income levels, loan balances, and credit report inquiries. Initial steps involve data preprocessing to handle missing values and normalize feature scales. Subsequent exploratory data analysis will uncover insights into the relationships between these features and the target variable—whether an application is approved or rejected. By developing and evaluating supervised learning models, the objective is to create a robust predictive model similar to those used by commercial banks with high accuracy in predicting the outcome of credit card applications.

2) Why do we drop some of the features?

We drop some of the features, specifically 'DriversLicense' and 'ZipCode', because they are not relevant to the model's ability to predict credit card approvals. Features that do not contribute meaningful information to the prediction task can introduce noise and unnecessary complexity into the model. In this case, 'DriversLicense' and 'ZipCode' are unlikely to have any bearing on whether a credit card application is approved or not, based on domain knowledge and the context of credit card approval processes. Therefore, removing these features simplifies the dataset and improves the efficiency and accuracy of the machine learning models that will be built.

In addition, after replacing '?' values with NaNs, we observed that approximately 4.49% of the rows contain at least one missing value. Therefore, by dropping rows with missing values, we prioritize data quality and enhance the effectiveness of our machine learning models in predicting credit card approval outcomes.

3) List each supervised model that is explored and the accuracy. Which is best based on accuracy alone?

- Logistic Regression
    - Accuracy before grid search: Approximately 85.32%
    - Accuracy after grid search (best performing model): 85.9% using {'max_iter': 100, 'tol': 0.001}
- Decision Tree Classifier:
    - Accuracy: 78.4%
    - ***Note:*** In the notebook, it is stated that the Decision Tree classifier performed better than the Logistic Regression model at 87.7%. However, after rerunning the notebook and some trial and error, I was not able to get the *expected* 87.7%. Therefore, I will continue my answers based on my findings.
- Random Forest Classifier:
    - Accuracy: 86%

Therefore, with these accuracy scores:

- Random Forest Classifier is the top performer with an accuracy of 86%.
- Logistic Regression follows with an accuracy of 85.9% after grid search optimization.
- Decision Tree Classifier is the least performer with an accuracy of 78.4% (*as opposed to the presented 87.7% score in the notebook*).

Based on these accuracy scores alone, the Random Forest Classifier has the highest accuracy among the models explored, closely followed by the optimized Logistic Regression model after grid search.

4) What is overfitting? Are any of the models prone to overfitting?

Overfitting occurs when a machine learning model learns the details and noise in the training data to the extent that it negatively impacts the model's ability to generalize to unseen data. In essence, the model becomes too specific to the training data and fails to capture the underlying patterns that would allow it to perform well on new, unseen data.

Based on the models explored:

- Logistic Regression
    - Logistic Regression typically has a lower tendency to overfit because it's a relatively simple linear model with regularization, which helps prevent it from fitting the noise in the data too closely.
- Decision Tree Classifier:
    - Decision Trees have a higher tendency to overfit, especially when they are deep and have many branches. A deep tree can capture intricate details of the training data, including noise, which may not generalize well to new data.
- Random Forest Classifier:
    - Random Forests are ensemble methods built on Decision Trees. While Random Forests mitigate overfitting compared to single Decision Trees by averaging multiple trees, they can still overfit if the individual trees are allowed to grow deep and complex.

Given these points, the Decision Tree Classifier is more prone to overfitting among the models explored, especially if its depth is not controlled or pruned adequately. Logistic Regression, with its regularization and simplicity, is less prone to overfitting, while Random Forests strike a balance but can still overfit if not properly tuned. Regularization techniques, cross-validation, and careful tuning of hyperparameters can help mitigate overfitting in machine learning models.

5) Explain the confusion matrix associated with the Decision Tree Model with respect to TP, FP, TN, FN, Precision, Recall and Accuracy:

A confusion matrix for a binary classification problem is typically associated with a Decision Tree Model. The matrix itself is structured as follows: { [[TN FP] [FN TP]] ⇒ [[74, 18], [29, 97]] }

Given the matrix:

- **True Negatives (TN)** = 74
- **False Positives (FP)** = 18

- **False Negatives (FN)** = 29
- **True Positives (TP)** = 97

We can relate the terms to the metrics given in the classification report:

- **Precision**: Precision is the ratio of correctly predicted positive observations (TP) to the total predicted positive observations (TP + FP). It measures how precise the positive predictions are.
    - Precision = TP / (TP + FP) = 97 / (97 + 18) = 0.843
- **Recall (Sensitivity)**: Recall is the ratio of correctly predicted positive observations (TP) to all observations in the actual class (TP + FN). It measures the ability of the model to correctly identify positive instances.
    - Recall = TP / (TP + FN) = 97 / (97 + 29) = 0.770
- **F1-Score**: The F1-score is the harmonic mean of precision and recall, providing a single metric to evaluate the model's performance.
    - F1-score = 2 * (Precision * Recall) / (Precision + Recall)
    - F1-score = 2 * (0.843 * 0.770) / (0.843 + 0.770) = 0.805
- **Accuracy**: Accuracy measures the overall correctness of the model.
    - Accuracy = (TP + TN) / (TP + TN + FP + FN) = (97 + 74) / (97 + 74 + 18 + 29) = 0.784 ~ 78.4%

These metrics collectively describe the performance of the Decision Tree Model:

- Precision (for class 1) is 0.843, indicating that when the model predicts class 1, it is correct approximately 84.3% of the time.
- Recall (for class 1) is 0.770, indicating that the model correctly identifies about 77.0% of all actual class 1 instances.
- The F1-score (for class 1) is 0.805, providing a balanced measure of precision and recall.
- Accuracy is 0.784, showing the overall correct predictions made by the model across both classes.

6) Why is it important to add preprocessing steps to a pipeline such as the one included in this notebook?

We can use a pipeline in machine learning both with and without preprocessing steps, depending on the requirements of the model and data. In most machine learning workflows, preprocessing steps are crucial for preparing the data before training a model. Preprocessing might involve tasks such as:

- Handling missing values
- Scaling numerical features
- Encoding categorical variables
- Feature selection or extraction
- Any other data transformation needed to make the data suitable for the model

There are scenarios where preprocessing might not be necessary or where we explicitly want to skip preprocessing steps (e.g., when using models like decision trees that do not require scaled data). That said, adding preprocessing steps to a pipeline in machine learning is important because it

ensures that our data is properly prepared before feeding it into the model. Reasons for adding preprocessing steps to a pipeline include:

- **Consistency**
    - Preprocessing steps like scaling numerical values or encoding categorical variables need to be applied consistently across the data. A pipeline automates this process, ensuring that the same transformations are applied to both the training and testing data.
- **Avoiding Data Leakage**
- **Simplifying Deployment**
- **Cross-Validation**
- **Improving Model Performance**

7) What are ways we can improve on the accuracy of the top model(s)?

There are several techniques that can contribute to improving the accuracy of the top models. These include:

- Feature Engineering
    - Feature Selection
    - Feature Transformation
- Hyperparameter Tuning
    - Grid Search or Randomized Search
    - Bayesian Optimization
- Model Selection
    - Ensemble Methods (Combine predictions from multiple models (e.g., Random Forests, Gradient Boosting Machines) to improve accuracy and robustness)
- Improving Data Quality
    - Data Cleaning
    - Normalization and Standardization
- Data Imbalance
- Cross-Validation
- Domain Knowledge Integration

8) What did you learn about approving credit card applications? What more would you like to do?

Through the Credit Card Approval dataset project, I've learned that approving credit card applications involves analyzing various factors such as income levels, credit history, and personal information. Machine learning models can effectively predict approval decisions based on these factors, providing automation and efficiency in the application review process.

Moving forward, I would like to explore more advanced machine learning techniques to improve prediction accuracy further. Additionally, incorporating real-time data and monitoring systems could enhance the model's performance and adaptability to changing economic conditions and customer behaviors. This project has highlighted the potential of leveraging data-driven approaches to streamline financial services.