Research Project: A Comparison of Logistic Regression and Random Forests on Predicting Default of Credit Cards

Supplementary Information

1. Glossary of Main Terms

- **Binary Classification:** A type of classification task where the target variable has two possible outcomes (in this case: default or no default).
- **Imbalanced Dataset:** A dataset where one class is significantly larger than the other, which can make standard accuracy metrics misleading.
- **ROC-AUC:** A metric that measures the ability of a model to distinguish between classes, ranging from 0 to 1.
- **SMOTE (Synthetic Minority Oversampling Technique) :** A resampling technique used to balance classes by generating synthetic samples for the minority class.
- **Recursive Feature Elimination (RFE)-** technique that removes the least important features to improve model performance
- **Misclassification Rate-** proportion of incorrect predictions made by the model
- **Accuracy-** percentage of correct predictions made by the model out of all predictions
- **Precision-** proportion of true positive predictions out of all positive predictions
- **Recall** - proportion of true positive predictions out of all actual positive cases
- **F1 Score** - mean of precision and recall
- **Log Loss-** how well the model's predicted probabilities match the actual class labels, lower value means better predictive accuracy.

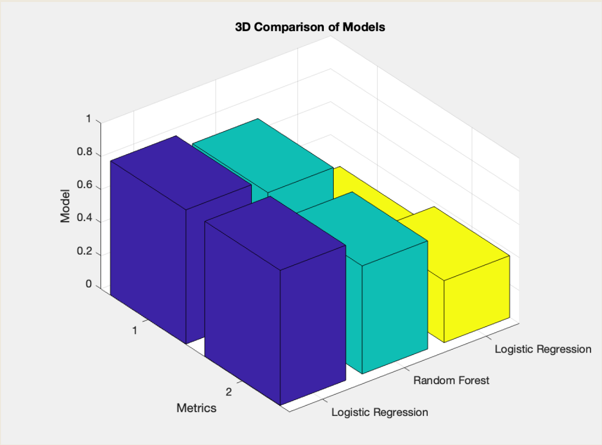
2. Issues Encountered

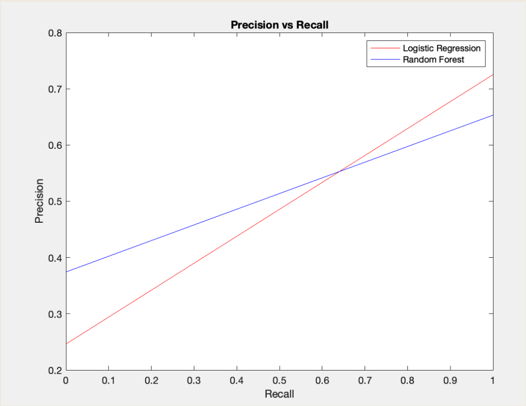
1. Originally, I had not split my target variables correctly and did not exclude the necessary columns.
2. There were many issues when adding in log loss to the code and I had to troubleshoot quite a bit to figure out how to do this. The matlab documentation only gave me a 'loss' function. With the help of online resources and GenAI I was able to apply the log loss function to matlab with epsilon correctly.
3. Another issue I encountered was determining the correct sequence of steps with my project. For example, I had originally standardized my data before splitting the data. Later, I realized that it is best to standardize after splitting the data to avoid data leakage.
4. I was unable to correctly use the SMOTE toolbox and came across numerous errors. I decided to implement this technique later and possibly do it in python instead of matlab.

3. Implementation Details

1. **Libraries and Tools:**
    - Use PYTHON for some initial EDA as it's the language I am most familiar with to carry out these tasks.
    - Used MATLAB for modeling, visualization, and evaluation.
        i. Matlab Libraries: Statistics and Machine Learning Toolbox.
2. **Data Preprocessing:**
    - Label encoded ordinal and binary features (e.g., EDUCATION, MARRIAGE, SEX) to prepare them for model input.
    - Standardized numerical features for Logistic Regression to ensure proper convergence.

4. Additional Graphs





1. **3D Graph:**
    - I did not use this graph in my project as it was not as easy to interpret and I figured that separate bar graphs that show each metric divided by model would be easier to examine.
2. **Precision vs Recall:**
    - This graph was one from the initial stages of my project. I ended up going with an ROC curve instead.

5. Models and Architectures Considered but Discarded

1. **Decision Tree:**
    - I was debating on going with decision trees vs logistic regression but after reading a few papers decided to select a more robust model like Random Forest. I am glad I went with this choice as it was the one model that I had not seen other researchers use when working with this particular dataset.

2. **ANNs:**
    - I was going to implement ANN models similar to the research paper that I reference throughout my project ([12]), but decided not to replicate the study and try another robust model like Random Forest and compare that to Logistic Regression instead.
3. **SMOTE:**
    - I tried SMOTE , but I was not able to successfully implement this even after downloading the necessary Matlab toolbox and consulting the founder's Github documentation. In the future, I would like to implement this with the help of python instead of matlab.
4. **PCA:**
    - I was going to implement PCA but decided to see how the results would hold up without doing so. I wanted to see how the models would behave with more raw data. In the future, I would love to build upon this project and see how things pan out after applying techniques like SMOTE and PCA.

6. Lessons Learned

- Random Forest models show better generalization, especially for imbalanced datasets, but tuning the number of trees and leaf sizes is important to prevent overfitting.
- In Logistic Regression, the choice of solver had a significant impact on convergence speed. The lbfgs solver performed the best in terms of speed and accuracy.
- Standardizing the features for Logistic Regression ensured faster convergence.
- Random Forest took significantly longer to tune than Logistic Regression.
- Logistic Regression showed faster convergence and was computationally less expensive, but it had lower recall, especially for identifying defaults (the minority class).
- Some models perform better than others when it comes to class imbalance.

7. Intermediate Result/Change:

I didnt go with this approach in favor of random search with cross-validation because those methods are more systematic and scalable when tuning hyperparameters. I also realized that relying on just a few performance metrics and not using cross-validation could lead to overfitting, so using more metrics and ensuring proper train-test splits would give me more reliable and accurate results. I used this thinking for random forest too and ended up scrapping this initial code that i had:

```
% manually select hyperparameters for Logistic Regression
lambda_values = [0.01, 0.1, 1];  % Arbitrary regularization strengths
solvers = {'lbfgs', 'sgd'};  % Limited set of solvers

best_mcr = inf;  % To track best misclassification rate
best_lambda = 0;
best_solver = '';

% Manual tuning of the dataset
for i = 1:length(lambda_values)
   for j = 1:length(solvers)
      fprintf('Trying Lambda = %.4f, Solver = %s\n', lambda_values(i), solvers{j});

      % Performing  a  train-validation split on the dataset
      X_train_split = X_train(1:floor(0.8*end), :);  % First 80% for training
      Y_train_split = Y_train(1:floor(0.8*end));
      X_val_split = X_train(floor(0.8*end)+1:end, :);  % Last 20% for validation
      Y_val_split = Y_train(floor(0.8*end)+1:end);

      % Training of the  logistic regression model
      model = fitclinear(X_train_split, Y_train_split, 'Learner', 'logistic', ...
         'Lambda', lambda_values(i), 'Solver', solvers{j});

      % Calculate misclassification rate on validation data
      pred = predict(model, X_val_split);
      mcr = mean(pred ~= Y_val_split);

      % Track best performing model
      if mcr < best_mcr
         best_mcr = mcr;
         best_lambda = lambda_values(i);
         best_solver = solvers{j};
      end
   end
end
```