ML Final Project Report

SARVESH 2020B5A71839G

1 Discuss your insights about the data.

- There are 30 features which are numerical and of type float.
- The mean values and standard deviations are quite varied across different features.
- There are no null values in any of the columns, indicating that the dataset is clean and does not require handling of missing data.

2 Discuss any preprocessing and feature engineering steps you might have employed.

- The mean values and standard deviations are quite varied across different features, so I used sklearn's StandardScaler function to bring them onto a common scale. This step is essential for algorithms that are sensitive to the magnitude of the input data, such as neural networks and gradient descent-based algorithms, as it can speed up convergence.
- I also tried to handle the outliers if any of them exist using a method called "capping" or "winsorizing"
  - It involves calculating the Interquartile Range (IQR) for each feature, which is the range between the first quartile (25th percentile) and the third quartile (75th percentile). Outliers are then defined as any values below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR. These outliers are then "capped" or trimmed to the nearest boundary value, either the lower or upper bound, thus minimizing their impact on the dataset.
- I have created a correlation matrix to assess the level of correlation between pairs of features. The analysis indicated that there is no significant correlation between the features.

3 Describe the ML model you have used and the reasoning behind your choice.

I tested three models for this binary classification task.

- The first model I used was the Random Forest classifier, chosen for its efficacy in handling large datasets. It achieved an accuracy of approximately 71%.
- To enhance performance, I integrated AdaBoost with the Random Forest, which slightly improved accuracy to 72%.
- Subsequently, I applied a Gradient Boosting classifier, which delivered similar accuracy results, ranging between 71% and 72%.
- Lastly, I deployed a neural network that outperformed the previous models, reaching an accuracy of roughly 74%.
- To reduce the risk of overfitting, I incorporated L1 and L2 regularization techniques.
- Additionally, I utilized batch normalization to optimize the neural network's training efficiency.

4 Discuss evaluation metric(s) you have employed to measure the performance.

- **Accuracy:** Considering that our Kaggle evaluation criterion is accuracy, I partitioned the training data into training and validation subsets. This division enabled the evaluation of my model's accuracy on the validation set to gauge its predictive performance.
- **Cross-Entropy Loss:** In the process of training a Neural Network, I employed cross-entropy loss to examine the model's learning progression during each epoch. This assessment was pivotal for fine-tuning the hyperparameters, which in turn, improved the model's accuracy.

5 Discuss any other approaches that you would want to try out in the future.

Based on what I have read from online sources, here are few approaches I am considering for future exploration:

- **XGBoost**: XGBoost is a more efficient and powerful implementation of gradient boosting with additional features for regularization, which can help in preventing overfitting. It's known for its performance on structured data.
- **LightGBM**: Similar to XGBoost, LightGBM is another gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with faster training speed and higher efficiency, lower memory usage, and better accuracy.
- **Ensemble Methods**: Beyond AdaBoost, I want to try other ensemble techniques like stacking or blending where it combines the predictions of multiple different models to make a final prediction, potentially improving performance over any single model.
- **Hyperparameter Tuning**: For all the models I have tried, hyperparameter tuning can be critical. Using methods like grid search, random search, or Bayesian optimization can help in finding the best settings for your models.
- **Dimensionality Reduction**: Techniques like PCA or t-SNE could be used before training models to reduce the feature space, which might help in improving model performance by removing noise and redundancy.