Supplementary Material – Brenner Swenson – 190046406

GLOSSARY:

AUC – Area underneath the ROC curve. Measures the entire two-dimensional area underneath the curve. Allows for a high-level combination of multiple performance metrics.

Bootstrap Aggregation (bagging) – A statistical method for estimating a quantity from a sample of data. Used to estimate descriptive stats in many predictive algorithms.

Convolutional Neural Network – A neural network that uses convolutional operations. Compared to a regular multilayer perceptron network, convolutional neural networks replace matrix multiplication with convolution in at least once layer.

F1 Score – A measure of a test's accuracy, calculated as the harmonic mean of both recall and precision divided by the total number of test samples. The balance of precision and recall can be modified depending on the application.

Kernel Smoother – A technique of estimating a function as the weighted average of close-by observed data points. Various kernel functions can be used, including Gaussian, Triangular, and Epanechnikov. Gaussian is the most widely used.

Max Number of Splits – In a decision tree, the maximum number of decision splits per tree. Ultimately decides how complex (deep) a tree can be.

Minimum Leaf Size – A decision tree hyperparameter that specifies the minimum number of examples per leaf node before it can be split again. This parameter is primarily used to prevent overfitting and promote generalisation.

Posterior Probability – (In Bayesian Statistics): the conditional probability of an event that is computed after relevant information or prior distribution is applied.

Precision - A metric calculated by dividing the number of true positives by the sum of true positives and false positives. Bounded between 0 and 1, precision conveys the proportion of the relevant examples found by a model that are indeed relevant.

Pruning – A method to reduce the complexity and size of tree-based algorithms by removing or "pruning" the branches of the tree that are deemed to be either non-critical to performance or redundant.

Principal Component Analysis (PCA) – The process of calculating the principal components of a dataset, or collection of datapoints. PCA is primarily used to reduce the dimensionality of a dataset and capture as much variance as possible in as few principal components as possible. Principal components can be interpreted as eigenvectors of the dataset's covariance matrix.

Recall – A metric calculated by dividing the number of true positives by the sum of true positives plus the number of false negatives. Bounded between 0 and 1, recall conveys the model's ability to locate all relevant examples.

Recurrent Neural Networks – A subset of neural networks that contain connections between nodes that form a connected graph along a temporal sequence.

ROC – Receiver operating characteristic curve. A plot that exemplifies the classification power of a binary classifier.

RUSBoost – An algorithm that remedies the issue of class imbalance. The algorithm mixes data sampling methods and boosting, ultimately resulting in a simple and efficient method that improves both binary and multiclass classification performance on imbalanced datasets.

SMOTE – Synthetic Minority Oversampling Technique. An oversampling technique that utilises k-nearest neighbour algorithm to produce synthetic data points as opposed to more general techniques where the minority class is simply duplicated.

Support Vector Machine (SVM) – A supervised learning technique, or a discriminative classifier, that uses hyperplanes to separate data in high dimensional space by finding the line that best separates data and optimizes the distance between points for the greatest separation possible. Many algorithms can find a line to separate data, but SVM's will find the best line.

Intermediate Results:

Random forest was first investigated as a classifier and has a built-in under-sampling option in MATLAB whereas Naïve Bayes does not. Without using SMOTE and instead using RUSBoost as a training parameter for the RF model, an F1 score of 0.673 was achieved. Naïve Bayes using the same unsampled dataset with optimised hyperparameters achieved a 0.552 F1 score. After applying SMOTE from the Python library imbalance-learn, RF improved even more while NB remained essentially unchanged.

When initially developing, it was appealing to use MATLAB to determine the train/test splits of the data for convenience, but Python proved to be a more appealing option. This stems from the ability to save the data down as a .csv as train/test files for not only reproducibility, but also to ensure that there is no data leakage between training and testing data when developing the models.

Min max normalisation was investigated first as a pre-processing step, but due to the large presence of outliers in the dataset, min max resulted in very skewed and small values for certain features. To mitigate this, standard scaling was used.

The number of neighbours to use in the SMOTE oversampling application was expected to have an effect on the resulting dataset and ultimately on model performance but increasing the number of neighbours from 4 to 10 incrementally did not have a material effect on F1 or accuracy.

Initially, nearly all hyperparameter options were included in the grid as possible options for searching, but ultimately had to be refined by initial testing using smaller grids to observe those that had the largest sensitivity to performance metrics. For example, the split criterion used to determine leaf-node splits was used in grid search but was found to have essentially zero effect on performance and only added to the time it took to search.

It was anticipated that a higher number of folds in K-Fold cross validation would help to improve generalisation, but the only observed marginal effect was that on overall training time, an expected side-effect.

Implementation Details:

All the categorical features in the dataset were one-hot encoded using Python Pandas functionalities; the one-hot encoding took place prior to the train test split. Each grid search was performed in MATLAB, with the results iteratively saved to an array and subsequently written to a .mat file. The Python library Scipy was then utilised to load in the .mat file via Python for further analysis and visualisation using Matplotlib and Altair.

The Python package imbalanced-learn was utilised to apply SMOTE as well as under-sampling. The SMOTE sampling_strategy parameter corresponds to the desired ratio of the number of samples in the minority class over the number of samples in the majority class after resampling. 0.5 was used. For under-sampling, the same parameter and value combination was used. This combination results in a 2:1 ratio of negative to positive labels in the modified dataset.

Each model has three MATLAB files: one file to execute the grid search (do not need to run this as it will take hours), a file to train the model with optimised hyperparameters, and another file to load the trained model and evaluate on a test set. It is only necessary to load the trained models and then run on the test dataset (that has not been modified other than one-hot encoding).

For example, for Random Forest, only the file titled ‘random_forest_test.m’ needs to be ran, as long as the current working directory has the ‘test.csv’ file in it as well as the ‘best_rf_trained.mat’ file. The model should then run smoothly, and variables appear on the right side with performance metrics. For Naïve Bayes, simply replicate the process but with the ‘bayes_test.m’ file.