CCE5225 Assignment 1

Methodology

The 3 classifiers were trained on the dataset using the method outlined in the requirements:

- Grid search over set of hyperparameters per-model.
- 5-fold cross validation.
- Dataset split into a training and test set with a ratio of 5:1.
- Assigning of class labels (0 for signal and 1 for background)
- Scaling and shuffling of test data.

For implementation, sklearn was used for its simplicity and rapid prototyping.

Hyperparameters selected

The following hyperparameters were selected for the classifiers selected, with their official name and the name used in the notebook. Each one uses a selection of values as present in the notebook.

Multi-Layer Perceptron

- Hidden layer configuration – hidden_layer_size
- Max iterations – max_iter
- Activation function – activation
- Max iterations without significant change - n_iter_no_change
- Epsilon, minimal value to divide in place of zero – epsilon

Of the above parameters, it is expected that hidden layer size and max iterations will have the greatest impact on performance as they determine how much there is to be trained and how much training the model receives. Max iterations limits should help optimize for fits which reach plateau in terms of accuracy. Epsilon however, is not expected to make a significant difference in performance.

Linear Support Vector Machine

- Penalty - penalty
- Optimize for primality or duality – dual
- Stopping tolerance – tol
- Per-class weights – class_weight
- Max iterations – max_iter

Penalty selection would determine how estimates are regularized, with an l1 penalty being expected to produce the best results. Duality in this case should be preferred since it is favored in scenarios when the number of samples outweighs the number of features. Stopping tolerance – equivalent to epsilon in the MLP model – defines when to stop training the model. Per-class weights should boost training for a favored class over the other (unless set to 0). Since there is a large proportion of background neutrinos in the dataset to signal ones, a balanced weighting would favor the signal class over the background class. Max iterations will help ensure that the model reaches the best accuracy with the given configuration.

Random Forest

- Total trees - n_estimators
- Quality measure of a branch split – criterion
- Minimum samples before splitting – min_samples_split
- Minimum samples to become a leaf node – min_samples_leaf
- Per-class weights – class_weight
- Total features for split evaluation – max_features
- Use out-of-bag samples – oob_score

Total trees should help identify the best performance as it would allow for more trees to be created by the model. min_samples_split, min_samples_leaf and max_features will help optimise trees by expanding on the number of end classifications, whereas quality measure and per-class weights experiment with preferring signal over background neutrinos and vice-versa. Out-of-bag validation will help ensure the accuracy of created

Results

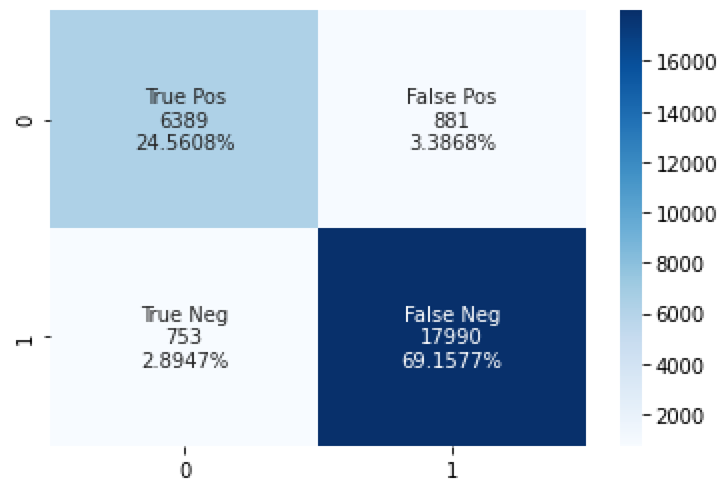
Execution was performed on 8 CPU cores and 24GB of memory. Execution was parallelized with a maximum of 7 jobs at a time and in a Windows (WSL2) environment with a Python 3.8.5 kernel. The following results were obtained:

MLP

Total fits: 480

Total training time: 79.4min

Average prediction time: 0.003ms

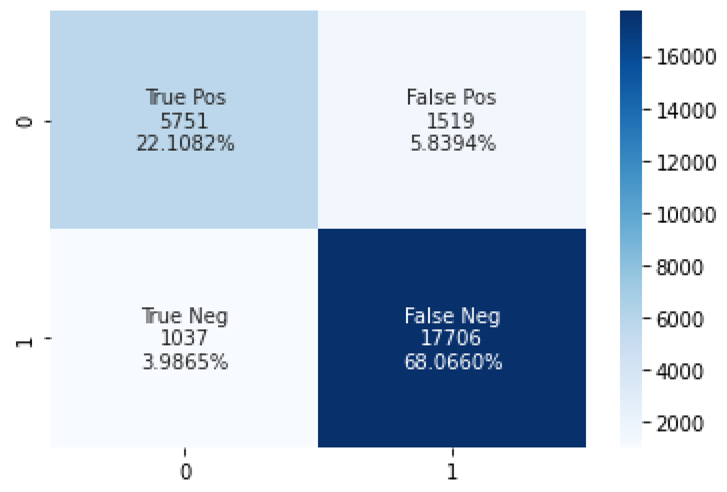


SVM

`Total fits: 600`

`Total training time: 43.9min`

`Average prediction time: 0.0004ms`

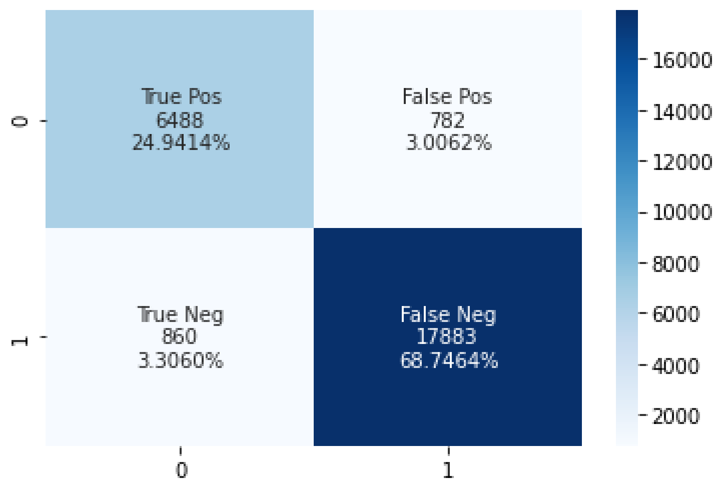


RF

Total fits: 1440

Total training time: 295.7min

Average prediction time: 0.032ms



Evaluation

Based on the above matrices, a classification report was produced for each one, resulting in the below metrics:

	Accuracy	Sensitivity	Precision	Specificity
MLP	0.94	0.91	0.92	0.95
SVM	0.90	0.79	0.88	0.94
RF	0.94	0.89	0.92	0.95

According to the above, the best-performing model across the four measures is the MLP which scored the best amongst all four measures. SVMs performed the most poorly in the above, barely matching the other models as far as classifying background (1’s) neutrinos. The Random Forest model almost tied with the MLP model, being slightly worse at classifying signal (0’s) neutrinos than MLP. As outlined above, it is believed that the MLP model is the best performant model from the 3 models trained on the basis that it is more sensitive to classifying samples as signals over background noise which would be favored in this use-case.