Harrison Bacordo
Student ID: 300388687

Assignment 1 Report

Part 1

All models were created and analysed using python's scikit-learn library. Through analysing my dataset (appendicitis), it is realized that the data instances without appendicitis far outnumbers the data instances with appendicitis. Because of this, there will be more emphasis on each model's scores in precision (the ability to identify only the relevant data instances), recall (the ability to identify all the relevant data instances), and F1 (the average of precision and recall). Accuracy will not be as emphasised since it can potentially be a misleading metric and doesn't reveal as much information for a skewed dataset.

Symbolist Tribe

The model used to represent the symbolist tribe of AI was the decision tree classifier. The decision tree model involves building a logic binary tree on data based on an optimization formula which relies on inverse deduction (this is done with a variety of different metrics, most notably the Gini impurity formula). In a decision tree, a node represents a feature, an edge represents the value of the parent node, and a leaf represents a classification along with the accuracy of this classification.

Classification of a given data instance begins at the root node of the decision tree. The value for the feature at the root node is grabbed from the instance, and the instance will continue down the matching edge to the appropriate child node. This is repeated (usually in a recursive manner) until a leaf node is reached, which represents the model's prediction of this instance's class.

Accuracy: 1.0	Precision	Recall	F1	Support
False	1.0	1.0	1.0	85
True	1.0	1.0	1.0	21
Avg/Total	1.0	1.0	1.0	106

Though the results may look perfect, it should be realized that this is the result of the training set being used to build the tree is also test set. Since the tree is fitted specifically to the training set, and there is more than likely a lack of pruning and/or noise in the data, the results will always perfect 1.0's.

Connectionist Tribe

The model used to represent the connectionist tribe of AI was the multi-layer perceptron classifier. The multi-layer perceptron is a model that comes from the deep-learning branch of machine learning, which is largely represented by various neural network structures. It uses multiple layers of weighted artificial neurons and relies on the tweaking of hyperparameters through back propagation to increase the model's accuracy.

A standard multi-layer perceptron passes a data instance's features as a vector through the first layer of the network, calculating the dot product of the feature vector and the first layer's weights. An activation function (used for normalization) is then applied to the vector and passed to the next layer, where the process is repeated until the output layer is reached. Evaluation is then done by calculating the squared error of the prediction vector and the label vector. Optimization through back propagation (many different optimization algorithms, though gradient descent is the most notable) is then calculated using the squared error, and the process repeats for another data instance.

Accuracy: 0.86	Precision	Recall	F1	Support
False	0.86	0.99	0.92	85
True	0.88	0.33	0.48	21
Avg/Total	0.86	0.86	0.83	106

The most notable score this model received is likely to be the recall of True instances. It could be speculated that the score is the result of the weights not being exposed/tweaked enough to true data instances in comparison to false data instances. This causes a heavy bias network and will result in the minority class being more commonly identified as the majority class.

Another interesting observation is that the precision score for false data instances is lower than the precision score for true data instances. This may be the result of the recall score of the true data instances being so low, since precision and recall tend to be inverses of each other.

Evolutionist Tribe

The model used to represent the evolutionist tribe of AI was the Evolutionary Search. classifier. Evolutionary Search is represented by a genetic program that works by creating a population of a specific classifier (in this case, a support vector machine) initialized with random parameters. It then runs the data instances through the entire population and finds the best scoring individuals through a fitness evaluation. It then breeds and mutates the best scoring individuals through a process called genetic search to optimize the next generation of individuals. This repeats for a set amount of generations, and the best individual is returned.

Accuracy: 0.90	Precision	Recall	F1	Support
False	0.91	0.96	0.94	85
True	0.81	0.62	0.70	21
Avg/Total	0.89	0.90	0.89	106

Since this tribe of AI uses a single model repeatedly to find the optimal parameters, it can be expected that it will outperform the model itself if ran separately without evolutionary techniques. The scores are overall above average, tough the recall for true instances falters relative to the other scores.

Bayesian Tribe

The model used to represent the Bayesian tribe of AI was naïve Bayes classifier. Naïve Bayes is a model which applies Bayes' theorem on the dataset to an output classification. Naïve Bayes works by assuming conditional independence between all features, regardless of whether that is actually true.

Naïve Bayes is best represented using a graphical model, where a feature/event is represented by a node (usually accompanied by a probability table), and each feature's/event's relationship with one another is represented using an edge. Evaluation of a specific data instance is done by calculating the probability that each of the possible classes are true for the instance given its values for certain features (posterior probability). The optimization method for this model is probabilistic inference, which consists of finding the overall probability that each feature/event in the data instance takes the value that it does.

Accuracy: 0.86	Precision	Recall	F1	Support
False	0.93	0.89	0.91	85
True	0.62	0.71	0.67	21
Avg/Total	0.87	0.86	0.86	106

As expected, the precision, recall, and F1 score for true data instances are considerably lower in comparison to that of the false data instances. One interesting thing to note is that though the recall score of the true data instances is significantly better than that of the multi-layer perceptron, it still scores the same accuracy. This further demonstrates how accuracy isn't always the best metric to analyse models on, especially in skewed datasets.

Analogizer Tribe

The model used to represent the analogizers tribe of AI was the k-nearest neighbours classifier. The nearest neighbour classifier classifies a data point by looking for the k closest data points in the existing dataset and classifying it as the majority class of the k data points.

K-nearest neighbours uses multiple explicit data points, represented as support vectors, to derive its prediction. This differs greatly from the other four tribes of AI, which use a sort of blending of all the data to derive their predictions. Evaluation in nearest neighbour is done by adjusting the margin for the nearest neighbours collected, which is represented by k. The model is then optimized through constrained optimization.

Accuracy: 0.88	Precision	Recall	F1	Support
False	0.90	0.95	0.93	85
True	0.75	0.57	0.65	21
Avg/Total	0.87	0.88	0.87	106

As expected, the precision, recall, and F1 score for true data instances are considerably lower in comparison to that of the false data instances. Although analogizers tend to be one of the best tribes for small datasets and could even be argued to be a suitable tribe for skewed datasets, the scores for true data instances are still similar to the previous models. This is largely due to default number of neighbours (the k value) set by scikit-learn, which is 5. This is quite a large k value, better utilized in larger datasets. Decreasing this value to better suit the dataset's size should increase all the resulting scores for this model.

Part 2

Business Understanding

This dataset was gathered to better understand the medical measures that are common among patients with appendicitis. Using the dataset, medical professionals can better identify patients with appendicitis, and may be able to spot signs of appendicitis before it happens. This allows patients to be proactive and receive treatment sooner.

Data Understanding

The dataset is made up of 106 real-world classification instances, each comprising of eight columns: the first seven being the features, and the last being the class label. Each instance represents a patient being checked for appendicitis. Each feature represents a medical measure (the specific medical measure is unspecified for all features) that has been observed in the given patient. All features are in real number form and normalized to a range between 0 and 1. The class label represents if the patient has appendicitis (class label 1) or not (class label 0).

The nature of the data itself looks relatively clean, though an initial observation makes it known that the dataset is very small. Additionally, the proportion of data instances with appendicitis to those without appendicitis is very skewed. 82% of the data instances belong to class label 0, with the remaining 18% going to class label 1. This is difference is further amplified in a small dataset, where this results in only 21 data instances representing class label 1.

Although there are no explicit missing values in the dataset, there are some occurrences of 0.0 among the features. Without knowledge of the specific medical measure each feature represents, it is unsure whether a 0.0 is a realistic value for the measure;

There are a couple of instances of outliers within the data (e.g. line 14 in the data file shows a patient with appendicitis, though the patient's medical measures are extremely similar to that of a patient without appendicitis).

There are no mixed attribute types.

Data Preparation

A pipeline would help in separating the training data instances from the test and validation data instances. A pipeline can also help with giving the minority class label more equal prominence in the different splits. This can be done by ensuring the training, test, and validation sets each have a close-to-equal amount of the minority class so that it is equally represented across the split. Lastly, a pipeline would give the ability to generate synthetic instances of the minority class label, which can help drastically increase fair representation of the class labels.

Modelling

This pipeline will mostly suit the connectionist tribe; neurons are very prone to being affected by bias and rely on large amounts of data to be properly trained. This pipeline will aid tremendously in increasing the connectionist tribe's effectiveness on the model. Overall, all tribes will benefit from the generation of synthetic data to reduce bias and increase the number of data instances. However, the connectionist tribe will likely improve the most from the pipeline.

Evaluation

Yes, this pipeline does support more than one method to evaluate a solution. It uses multiple methods, such as splitting the data into training and test sets, under sampling with SMOTE (removing unnecessary data such as noise and outliers), and oversampling with Repeated Edited Nearest Neighbours (generating synthetic data to give a minority class more prominence). This pipeline also supports a feature selector method which uses a kNN classifier to keep the k most significant features and remove the rest.

Deployment

This pipeline will be easily deployable. The pipeline will do two main operations after training the model. Firstly, it will pickle the model and saves it into a folder that can be loaded to make predictions when needed. This allows for storing multiple classifiers that persist past the program's termination, making it production-ready. Secondly, the pipeline will write a classification report based on the trained model's performance on a test set. This gives information on the classifier's viability, which further decisions can be made from.

Part 3

Symbolist Tribe

Accuracy: 0.84	Precision	Recall	F1	Support
False	0.89	0.92	0.91	26
True	0.60	0.50	0.55	6
Avg/Total	0.83	0.84	0.84	32

Connectionist Tribe

Accuracy: 0.88	Precision	Recall	F1	Support
False	0.92	0.92	0.92	26
True	0.67	0.67	0.67	6
Avg/Total	0.88	0.88	0.88	32

Evolutionist Tribe

Accuracy: 0.88	Precision	Recall	F1	Support
False	0.89	0.96	0.93	26
True	0.75	0.50	0.60	6
Avg/Total	0.87	0.88	0.86	32

Bayesian Tribe

Accuracy: 0.84	Precision	Recall	F1	Support
False	0.92	0.88	0.90	26
True	0.57	0.67	0.62	6
Avg/Total	0.85	0.84	0.85	32

Analogizer Tribe

Accuracy: 0.86	Precision	Recall	F1	Support
False	0.88	0.88	0.88	26
True	0.50	0.50	0.50	6
Avg/Total	0.81	0.81	0.81	32

The overall scores of the classifiers has worsened after implementing the pipeline. This is likely due to replacing the test data (which before was the same data used to train the classifiers) with data instances that were not before seen by the classifier, as well as providing fewer instances than before. However, although the overall scores of the classifiers have lowered, it gives us a more reliable and legitimate insight into how the model will classify real-world problems that it hasn't been exposed to before.

Program code, classification reports, and pickled classifiers are stored in the Pipeline.py, classification_report.txt, and the directory "pickled_classifiers" respectively.