MULTIVARIATE DECISION TREES AND THEIR COMPARISON TO NEURAL NETWORKS AND TRADITIONAL TREE-BASED METHODS

Shivam Mundada, Soumil Ghosh, Rishabh Khandelwal

1. PROBLEM STATEMENT

Decision trees are popular and widely used models in predictive analytics due to their simplicity, interpretability, and ability to handle both categorical and numerical data. However, traditional decision trees, often referred to as vanilla decision trees, which partition the feature space along axes, resulting in axis-aligned decision boundaries, have limitations in capturing complex relationships and interactions between features. This project investigates the advantages of multivariate decision trees, which utilise linear combinations of features, over vanilla decision trees in various predictive modelling tasks.

Vanilla decision trees have several shortcomings compared to multivariate decision trees:

- **Difficulty in Capturing Interactions:** Vanilla decision trees may fail to capture interactions between features since they split the feature space independently at each node. Multivariate decision trees can potentially capture interactions between features by using linear combinations, allowing for more flexible decision boundaries.

- **Sensitive to Feature Scaling:** Vanilla decision trees are sensitive to the scale of features. Large-scale features may dominate the splitting process, leading to biased decisions.

- **Limited Ability to Handle Noise:** Vanilla decision trees are susceptible to noise in the data, as they may split into irrelevant features or outliers. Multivariate decision trees can potentially be more robust to noise by considering linear combinations of features, which may smooth out noisy patterns in the data.

Through empirical analysis of various datasets, this project aims to explore and compare the performance of multivariate decision trees against vanilla decision trees, random forest, boosting and neural network models and test the hypothesis that multivariate decision trees outperform these classes of models at classification tasks.

2. METHODOLOGY

2.1. MODEL CONSTRUCTION

Three versions of the multivariate decision trees with linear splits were constructed, with increasing levels of complexity, as follows:

- **Decision Tree with Random Features chosen for the Linear Split(Method A):** This decision randomly selects two features at each node. It then uses logistic regression to compute a linear combination of these two randomly selected features. A split threshold is computed based on the linear combination that maximises information gain. Although

the method might seem simple to implement, randomly selecting features may not always lead to the most informative splits. It may lead to suboptimal decisions.

- **Decision Tree with Lasso Regularisation and Logistic Regression(Method B):** This Utilises Lasso regularisation to automatically select the most significant features and reduce the remaining coefficients to zero. The randomness introduced by np.random.choice in selecting features (feat_idxs) during each tree growth iteration can indeed lead to different prediction values each time you fit the model. Like Method A , this also Fits logistic regression on the selected features to compute the optimal linear combination and computes a split threshold based on the linear combination that maximises information gain. It performs better than Method A since it incorporates Lasso regularisation for feature selection, potentially leading to better split decisions compared to the Method A.

- **Decision Tree with Lasso/Ridge Regularisation and Logistic Regression(Method C):** Similar to Method B, Method C also utilises Lasso and Ridge regularisation for automatic feature selection. It's different from Method B because it is performing feature selection using the SelectKBest method from Scikit-learn instead of selecting random features using np.random.choice. It fits logistic regression on the selected features to compute the optimal linear combination and computes a split threshold based on the linear combination that maximises information gain. It offers more flexibility by considering all features rather than a fixed number. Automatically selects the most relevant features, potentially leading to better-split decisions and model performance compared to both Method A and Method B. It sometimes performs worse than Method B since it tends to overfit into the data because we are using the SelectKBest method.

2.2. MODEL TESTING AND PERFORMANCE EVALUATION

Two experiments were performed to evaluate our models against traditional tree-based models and neural networks.

2.2.1. EXPERIMENT 1: Comparison to other tree-based methods based on binary Classification using the Scikit-learn Breast Cancer dataset

This experiment evaluated the performance of Multivariate trees relative to traditional tree-based methods in a binary classification task.

The Scikit Learn Breast Cancer dataset consists of breast cancer data from 569 samples with Binary (0/1) labels for the presence of breast cancer in the patient, along with data based on 30 features. The performance of all 3 versions of the multivariate tree (Method A, better and best), along with vanilla decision trees, boosted trees, and random forest models, were evaluated.

2.2.2. EXPERIMENT 2: Comparison to neural networks based on classification of larger and more complex datasets

The goal was to compare the performance of our multivariate tree with a more advanced model, such as a neural network, when the task dealt with a larger dataset with many features. The neural network used was an ANN with one input layer, one output layer, and 4 hidden layers. 2 datasets were used for comparison:

A. Microorganisms: The dataset had 24 features, which were measurable attributes of the organisms which define their shape, such as solidity, eccentricity, minor/major axis length, area and perimeter, and 10 class labels from 0 to 9. The set contained about 30000 data points.

B. Coffee Beans: The dataset had 16 features, comprising physical attributes of the coffee beans defining their shape and class labels from 0 to 6. The set contained about 13000 data points.

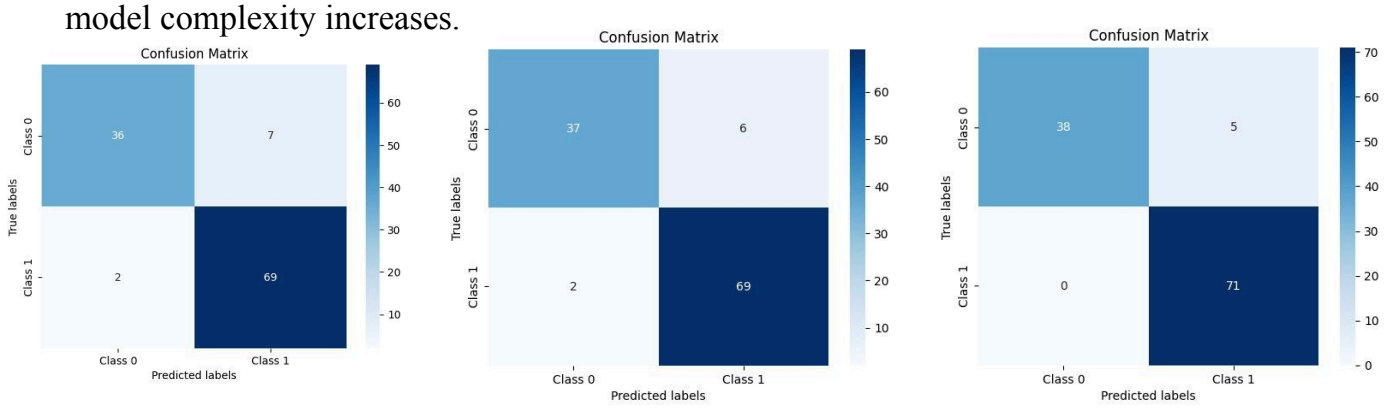
3. EXPERIMENTAL RESULTS AND VALIDATION

3.1. RESULTS OF EXPERIMENT 1: CLASSIFICATION OF BREAST CANCER DATA SET

The table below shows the accuracy scores of the different models. While Method A and better versions of the multivariate tree fail to outperform vanilla trees, the best version can do so, despite eventually falling short of boosting and random forest models.

MODEL	ACCURACY SCORE
MULTIVARIATE TREE (METHOD A)	0.9263157895
MULTIVARIATE TREE (METHOD B)	0.9364912281
MULTIVARIATE TREE (METHOD C)	0.9523895080
VANILLA DECISION TREE	0.9425980298
RANDOM FOREST	0.9649122807
GRADIENT BOOSTING	0.9561403509

Given below, from left to right, are the confusion matrices obtained from the Method A, better and best models, respectively, which show a decreasing number of misclassifications as the model complexity increases.



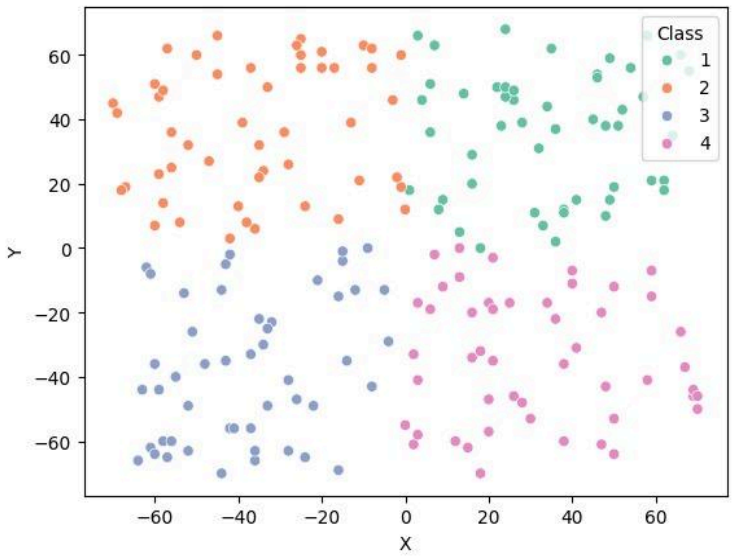
3.2. RESULTS OF EXPERIMENT 2: COMPARISON OF MULTIVARIATE TREES WITH NEURAL NETWORK

- **Microorganisms:** When a neural network is fit on the data we obtain a accuracy of 83.93% whereas, our multivariate tree gives us 64.002% accuracy. As it is a large dataset with 24 features, this result is to be expected. Decision trees tend to underperform on such complex datasets where the relations between features are complex and hard to capture.
- **Coffee Beans:** XGBoost and RF provide relatively straightforward interpretations of their predictions, making them suitable for tasks where interpretability is important. It can be difficult to interpret the predictions of neural networks, particularly deep neural networks, since they are sometimes thought of as "black box" models. On the CoffeeBeans dataset, we notice that our multivariate outperforms a neural network setup. Our ridge regression model provided 89.019% prediction accuracy whereas, the best accuracy we could achieve with a NN model is around 45% after changing and optimizing the number of hidden layers and nodes per layer. Another reason for this could be noisy data as neural networks might struggle to learn meaningful patterns. Random Forest and XGBoost models gave a prediciton accuracy of 92.47% and 92.84% respectively. This show that the dataset might be better suited for a decision tree setup.

3.3. SPECIFIC DATASETS WHERE LINEAR SPLIT TREES UNDERPERFORM OR OVER PERFORM RELATIVE TO VANILLA TREES

When we take axes-aligned data, Vanilla Decision Trees perform extremely well, as expected, whereas our custom (linear split) decision tree seems to be struggling. Maybe if we take a more complex dataset, our custom tree could come closer in performance.

```
Vanilla accuracy score:  96.66666666666667
              precision    recall  f1-score   support

           1       0.80      1.00      0.89         4
           2       1.00      0.83      0.91         6
           3       1.00      1.00      1.00        12
           4       1.00      1.00      1.00         8

    accuracy                           0.97        30
   macro avg       0.95      0.96      0.95        30
weighted avg       0.97      0.97      0.97        30

Custom accuracy score:  76.66666666666667
              precision    recall  f1-score   support

           1       0.50      1.00      0.67         4
           2       0.50      0.33      0.40         6
           3       0.92      1.00      0.96        12
           4       1.00      0.62      0.77         8

    accuracy                           0.77        30
   macro avg       0.73      0.74      0.70        30
weighted avg       0.80      0.77      0.76        30
```



Whereas, when we rotate the same data so that the axes are now a linear combination of each other, we notice that the Vanilla decision tree drops in accuracy. Our custom(linear split) decision tree performs much better as it favours splits based on a linear combination of data.

```
Vanilla accuracy score:  87.5
              precision    recall  f1-score   support

           1       0.86      1.00      0.92         6
```

4. CONCLUSION AND FUTURE WORK

Multivariate decision trees may prove to be a useful class of models which capture complex relationships between features in data that ordinary trees cannot.
We see through our experiments that for simple binary classification tasks, the best version of our multivariate tree can outperform vanilla trees and convincingly keep up with random forest and boosting models in terms of prediction accuracy.
However, for more complex classification tasks, the multivariate tree performs poorly relative to neural networks, which are known for their robustness and ability to handle large datasets with many features. A likely reason that neural networks outperform the multivariate tree is that the presence of multiple hidden layers and a sigmoid activation function introduces nonlinearity in the model, possibly allowing for more accurate detection of complex relationships between different features.
Possible future work includes finetuning our multivariate tree algorithm to be able to handle larger datasets more accurately and provide us with a meaningful relationship between features to allow a complex linear split.

We Also performed Cross Validation and came up with the following results:
Decision Tree with Lasso Regularisation and Logistic Regression: ***Mean cross-validation accuracy: 0.9351032448377582***
Decision Tree with Lasso Regularisation and Logistic Regression(K best features): ***Mean cross-validation accuracy: 0.9336283185840708***
Decision Tree with Ridge Regularisation and Logistic Regression(K best features): ***Mean cross-validation accuracy: 0.9238938053097344***

We would expect **Decision Tree with Lasso/Ridge Regularisation and Logistic Regression(K best features)** to perform better on the cross-validation sets but SelectKBest, especially when using metrics like f_regression, focuses on selecting features based on their individual relationship with the target variable. It does not inherently consider the interrelatedness or multicollinearity between features.
If features are highly correlated or interrelated, SelectKBest may prioritise one feature over another based on its individual predictive power, potentially ignoring the redundant information captured by correlated features. Certain features might have low importance individually but have high significance together, leading to them not being considered for the linear split. This is something we would like to look into in the future and work on by tuning the Regularisation methods and improving feature selection.