Project Title: **A Deep Dive on Decision Trees**

Team number: 1

Team members: Leela Manas Bayireddy, Karthik Nerella, Asha Latha Nagathota, Sai Keerthi Madhuvani Perubhotla, Ashwik Sagi

1 Introduction

As part of our research on traditional methods in Machine Learning, Decision Trees being one of the oldest classification algorithms, piqued our interest. In psychology, decision tree methods were used to model the human concept of learning. Along the way, researchers discovered the algorithm was useful for programming. As part of this project, we want to perform an in-depth analysis and comprehensive review on Decision Trees.

1.1 Motivation

Decision trees are widely-used algorithms, which are flexible models and with the recent advancements in Neural Nets, combining the Deep Neural networks with traditional Decision trees has proven to achieve improved accuracy and interpretability. These ideas and vast applications of decision trees in the industies like Finance, Medicine, Banking inspired us to do a detailed review on Decision trees.

2 Project Description

2.1 Goal

Our goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. As part of this project, we intend to dive deep into understanding the architecture of decision trees, types of algorithms, their applications, performance improvements and also their limitations.

2.2 Objectives

A detailed analysis and implementation on decision tree algorithms like ID3, C4.5, CART, CHAID on various datasets, Ensemble Learning techniques and strategies to improve performance of decision trees (Boosting). Dive deep into promising future works on decision trees including HSM, Born-Again Tree Ensembles, One-vs-rest, Binary relevance, NBDT, IDT and implement a few of these.

3 Contributions

Topic	Name of the Individual
Introduction of Decision Trees, Exploratory Data Analysis	Ashwik Sagi
ID3 Algorithm Analysis and Implementation	Leela Manas Bayireddy
C4.5 Algorithm Analysis and Implementation	Leela Manas Bayireddy
CART Classification Algorithm Analysis and Implementation	Asha Latha Nagathota
CART Regression Algorithm Analysis and Implementation	Asha Latha Nagathota
CHAID Algorithm Analysis; Limitations of Decision Trees	Sai Keerthi Madhuvani Perubhotla
Ensemble Learning	Karthik Nerella
Random Forests Classification	Karthik Nerella, Ashwik Sagi
Random Forests Regression	Asha Latha Nagathota
Boosting - Ada-Boost, Gradient Boosting, XGBoost	Karthik Nerella, Ashwik Sagi
Hierarchical SoftMax	Asha Latha Nagathota
Born-Again Tree Ensembles Review	Sai Keerthi Madhuvani Perubhotla
One-vs-rest & Binary relevance	Leela Manas Bayireddy
Neural-Backed Decision Trees Topic Introduction	Sai Keerthi Madhuvani Perubhotla
Neural-Backed Decision Trees Analysis & Implementation	Asha Latha Nagathota
Improved Decision Tree Review	Sai Keerthi Madhuvani Perubhotla

4 Decision Trees

4.1 Introduction

A Decision Tree is one of the most popular and effective supervised learning technique for classification problem that equally works well with both categorical and quantitative variables. It is a graphical representation of all the possible solution to a decision that is based on a certain condition. In this algorithm, the training sample points are split into two or more sets based on the split condition over input variables.

4.2 Hyper parameters of Decision Trees

Maximum Depth: The maximum depth of the decision tree is simply the largest length between the root to leaf. A tree of maximum length k can have at most 2**k leaves.
Minimum number of samples per leaf: While splitting a node, one could run into the problem of having 990 samples in one of them, and 10 on the other. This will not take us too far in our process and would be a waste of resources and time. If we want to avoid this, we can set a minimum for the number of samples we allow on each leaf.
Maximum number of the feature: We can have too many features to build a decision tree. While splitting, in every split, we have to check the entire data-set on each of the features. This can be very expensive. A solution for this is to limit the number of features that one looks for in each split.

4.3 Exploratory Data Analysis

We have done exploratory data analysis on two of the important datasets we have commonly used across algorithms. The details are in these notebooks: heart.ipynb.credit.ipynb and also in the appendix[7]

5 Various Decision Tree Algorithms

In this section, we will detail out the work we have done in exploring and implementing various decision tree algorithms.

5.1 ID3

ID3 Algorithm was initially developed by Quinlan[1] and is one of the oldest decision tree algorithms. ID3, which stands for Iterative Dichotomiser 3 tries to iteratively split the decision tree based on an attribute that has highest information gain. First we need to calculate the entropy of an attribute for each of its values using the formula mentioned here.

$$Entropy = \frac{-p}{p+n} \cdot \log_2\left(\frac{p}{p+n}\right) - \frac{n}{p+n} \cdot \log_2\left(\frac{n}{p+n}\right)$$

where p and n are the number of samples with positive and negative values of the target variable respectively. The entropy becomes zero when you have all positive values of a target variable or all negative values. Then we calculate the average information/entropy of the attribute, across all its values V_i as follows.

$$I\left(Attribute\right) = \sum \frac{p_i + n_i}{p+n} \cdot Entropy(V_i)$$

We calculate Information Gain of attribute by subtracting Average Information from the entropy of the entire dataset S, using this formula:

$$InformationGain = Entropy\left(S\right) - I\left(Attribute\right)$$

We calculate this value for each of the attributes and pick the attribute with highest information gain as the next tree node. We stop our recursion when Entropy becomes zero, getting a pure subset with a single value of our target variable. We have implemented this algorithm from scratch here: ID3_Algorithm.ipynb

5.1.1 Performance on datasets

To better understand the advantages and limitations of this algorithm, we ran it on various types of datasets and analysed the results. We picked up the Heart Disease dataset, Credit Defaultor dataset, Zoo dataset[2]. Both Heart and Credit datasets required binomial classification whereas Zoo dataset requires multinomial classification. The results are as follows.

Dataset Name	Accuracy%	Recall%	Specificity%	Precision%	F1 Score%
Heart Disease Dataset	28.39	31.91	23.52	36.58	34.09
Credit Defaultor Dataset	61.00	37.20	70.56	33.68	35.35
Zoo Dataset	96.77	96.77	N/A	100.00	98.36

The constructed decision trees can be found here: ID3_Algorithm.ipynb. In the heart dataset, we have a continuous variable 'sc' with a wide range of values, due to which tree branches out very widely in the initial step and provides poor performance. This is one of the drawbacks of this algorithm. It performs poorly if the considered(descriptive) features in the data set are widely distributed continuous variables. In the credit dataset, there is a similar issue but on pruning such features(age,amount) the accuracy improved to 61% (from the previous 45 %). We have also done multi class classification on this zoo dataset from UCI repository where target variable has values ranging from 1 to 7, and got a good accuracy of 96 percent because our descriptive features are discrete variables with smaller range of values and our constructed tree could efficiently differentiate them. To improve its performance on multivalued attributes like 'sc' in heart dataset, we instead used 'Information Gain Ratio' here which reduces bias on such attributes and improved its accuracy to 34% but nevertheless, the accuracy could not be improved for the other datasets.

5.1.2 C4.5 Algorithm

C4.5 Algorithm is an improvement of ID3, which uses Normalized Information Gain Ration as its splitting criteria. One of the major drawbacks of ID3 was its inability to handle continuous variables as observed in the heart dataset. To be able to deal with continuous variables, this algorithm defines a threshold and then differentiates the values into two lists, one in which attribute value is above the threshold and one with values below it. Also, C4.5 can handle data with missing ('?') values, it optimizes the tree after creation by pruning unnecessary branches. To observe the gain in performance, we ran this algorithm using this code, utilizing chefboost library, on two of previous datasets: Heart and Zoo dataset. The results are as follows.

Dataset Name	Accuracy%	Recall%	Specificity%	Precision%	F1 Score%
Heart Disease Dataset	93.82	97.87	88.23	92.00	94.84
Zoo Dataset	100.00	100.00	N/A	100.00	100.00

From the results, we can see that its accuracy on heart dataset improved significantly to 93.82% from ID3's 23.39%. Hence, C4.5 is an efficient algorithm in handling data with continuous attributes. We also tried to implement this on credit dataset but were facing some issues in the code. For the zoo dataset, we got accuracy of 100 percent because the dataset is smaller and our chefboost has overfitted. Otherwise, an accuracy of 100 percent is not realistic on actual datasets.

5.2 CART

CART stands for Classification and Regression Trees, and it is the modern name for the Decision Tree algorithm. Leo Breiman coined the term "classification and regression trees (CART)" to refer to Decision Tree algorithms that can be used to solve classification and regression modeling problems. This algorithm provides a foundation for advanced algorithms like bagged decision trees, random forest and boosted decision trees. The algorithm works repeatedly in the following steps:
1. Finds the best split for each feature. For each feature with K different values there exist K-1 possible splits.
2. Among the best splits from Step 1 finds the one, which maximizes the splitting criterion.
3. Splits the node using the best node split from Step 2 and repeats Step 1 until the stopping criterion is satisfied.
CART is a binary tree and tries to split the decision tree recursively based on Gini index for classification and Mean Square Error (MSE) for regression. We calculate splitting criterion for each

of the attributes and pick the attribute with lowest value(Gini Index for classification and MSE for Regression) as the next tree node. We terminate the recursion when stopping criteria is met. The most common stopping criterion is to use a minimum count on the number of training instances assigned to each leaf node. Another approach is when the depth of the tree reaches a maximum value. The code implementation of classification and regression algorithms from scratch and evaluation on various datasets can be found here: CART_Classification.ipynb , CART_Regression.ipynb

5.2.1 Performance on DataSets

CART classification algorithm is run on different data sets and the results are analyzed. The algorithm is tested on the Heart Disease, Credit Defaulter, Classifying Gender based on Voice data sets for binary classification. The results of custom implementation of CART classification versus SCIKIT library are as follows.

Performance Metrics	Heart Disease Dataset		Credit Defaulter Dataset		Gender Voice Dataset	
	CUSTOM	SCIKIT	CUSTOM	SCIKIT	CUSTOM	SCIKIT
Accuracy%	70.37	76.54	75.00	74.33	96.53	96.85
Recall%	74.47	82.98	46.51	46.51	96.16	97.37
Specificity%	64.71	67.65	86.45	85.51	96.93	96.27
Precision%	74.47	78.00	57.97	56.34	97.14	96.59
F1 Score%	74.47	80.41	51.61	50.96	96.65	96.98

We tried to improve the accuracy of the CUSTOM implementation. Although couldn't outperform the SCIKIT implementation achieved performance metrics almost similar to SCIKIT library. Though it is not a significant improvement, dropping the feature "thal" from the heart dataset, increased the performance metrics by around 5 percent and can be found here CART_Classification.ipynb . Observed that the decision trees are deterministic and output vary by adding or removing a row or more, and they tend to over fit. Further explored the performance of decision tree for multi class classification problem on Fruit dataset with integer values as labels using SCIKIT library and can be found here Fruit_Classification_Comparision.ipynb .Below is the comparison study with other predictive algorithms

Algorithm	Accuracy%	Recall%	Specificity%	Precision%	F1 Score%
SVM	99.36	99.36	99.36	99.36	99.36
KNN	95.88	95.88	95.88	95.88	95.88
Decision Tree	74.38	74.38	74.38	74.38	74.38
Random Forest	93.87	93.87	93.87	93.87	93.87

From the above comparison, compared to Decision tree SVM, KNN and Random Forest performed better. Random Forest is suitable when we have a large dataset and the performance of decision trees can be improved using Hierarchical softmax, which will be explored in the next sections.

The results of custom implementation of **CART regression versus SCIKIT** library is as follows.

Life Expectancy (WHO) DataSet				
	Mean Squared Error	Root Mean Squared Error	Mean Absolute Squared Error	r2_score
CUSTOM	6.829	2.613	1.682	0.923
SCIKIT	6.872	2.613	1.689	0.922

From the metrics, we can see that the custom decision tree regression performance is similar to SCIKIT library. The decision trees are great for building the non-linear relationships between input features and the target variable and for the above dataset considering R2 score, decision tree regression did a pretty decent performance.

5.3 CHAID

CHAID(Chi-square Automatic Interaction Detector) analysis is a Decision tree algorithm that is used for discovering relationships between a categorical response variable and another categorical predictor variables. It is of great use when are trying to look for patterns in datasets with more number

of categorical variables and is a convenient way of summarizing the data as the relationships can be easily visualized. In case of Classification problems where the dependent variable is categorical, as indicated in the name, CHAID uses Person's chi-squared test of independence to determine the best split at each step. It test for an association between two categorical variables. In case of Regression problems, where the dependent variable is continuous, CHAID uses F-test.

5.3.1 Chi-square test

Chi-Square is a statistical measure to find the difference between child and parent nodes. To calculate this we find the difference between observed and expected counts of target variable for each node and the squared sum of these standardized differences will give us the Chi-square value. The chi-square test is non parametric. That means this test does not make any assumption about the distribution of the data i.e., it is a non-parametric statistical method of free distribution. Steps to calculate Chi-square for a split are as follows:
1. Calculate Chi-square for individual node by calculating the deviation for Success and Failure both.
2. Calculate Chi-square of Split using Sum of all Chi-square of success and Failure of each node of the split.

5.3.2 F-test

F-tests (instead of Chi-square tests) to calculate the difference between two population means. If the F-test is significant, a new partition (child node) is created (which means that the partition is statistically different from the parent node). On the other hand, if the result of the F-test between target means is not significant, the categories are merged into a single node. The F test is a parametric test. It assumes that data are normally distributed and that samples are independent from one another.

5.3.3 Key Components of the CHAID process

1. Preparing the predictor variables
2. Merging categories.
3. Selecting the best split.

5.3.4 Advantages of CHAID

1. Data pre-processing is easier. There is no need for data normalization and missing value handling.
2. The branching is non-binary which means wider decision trees are possible. It is easy to manage, flexible, fast and useful.

5.3.5 Limitations and Challenges of CHAID

1. Older Decision Tree Algorithm and it does not have pruning function.
2. Data must be either ordinal, nominal or interval, and not metric. Response or dependent variable is a mandate.
3. Can't perform better analysis with continuous dependent variables compared to that of other algorithms and not suitable for large data sets which has more than about 1000 rows. 4. If the decision tree gets more wider,it may be unrealistically short and hard to relate to real business conditions.
5. Difficult to validate as it is not available on skicit-learn.
6. Chi-square formula is not consistent as the formula varied in different sources.

5.4 CART vs CHAID

1. While CART is helpful for binary splits, CHAID supports multi-class(non-binary) splits as well.
2. CHAID is more suitable for Descriptive analysis where as CART is more suitable for Predictive analysis.
3. In case of missing data, CART performs better than CHAID because missing values are taken as a separate group in CART and it'll only be combined with any other category if there is actually a high similarity between the two categories, unlike CHAID which combines the missing values with the nearest group.

6 Limitations of Decision Tree

Decision trees may become in stable and tend to over fit, which can be reduced using pruning, ensemble learning techniques discussed in the next section

6.1 Pruning

Pruning may be defined as shortening the branches of the tree. The process of reducing the size of the tree by turning some branch node into a leaf node and removing the leaf node under the original branch.

7 Ensemble Learning

Ensemble learning gives credence to the idea of the "wisdom of crowds," which suggests that the decision-making of a larger group of people is typically better than that of an individual expert. Similarly, ensemble learning refers to a group (or ensemble) of base learners, or models, which work collectively to achieve a better final prediction.A single model, also known as a base or weak learner, may not perform well individually due to high variance or high bias. However, when weak learners are aggregated, they can form a strong learner, as their combination reduces bias or variance, yielding better model performance.

7.1 Bagging : Bootstrap Aggregation

The working principle is to build several base models independently and then to average them for final predictions. Refer to Appendix section 4 for Bootstrap aggregation workflow

Advantages: Ease of Implementation and Reduction of Variance

Disadvantages: Loss of interoperability,Computationally expensive and Less flexible

7.1.1 Random Forest

Random forest changes the algorithm for the way that the sub-trees are learned so that the resulting predictions from all of the subtrees have less correlation. It is a simple tweak. In CART, when selecting a split point, the learning algorithm is allowed to look through all variables and all variable values in order to select the most optimal split-point. The random forest algorithm changes this procedure so that the learning algorithm is limited to a random sample of features of which to search. The number of features that can be searched at each split point (m) must be specified as a parameter to the algorithm. You can try different values and tune it using cross validation.

For classification a good default is: m = sqrt(p) .For regression a good default is: m = p/3. Where m is the number of randomly selected features that can be searched at a split point and p is the number of input variables.

7.1.2 Random Forest Regression

Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction than a single model.A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees. Let's compare r2_score of regression models like decision tree (DT), Random Forest(RF) and linear regression(LR) using SCIKIT libraries and the code can be found here Models_Regression.ipynb .

Metric	Life Expectancy			Car Price			Student's marks			Heart Disease		
	DT	RF	LR	DT	RF	LR	DT	RF	LR	DT	RF	LR
r2_score	0.93	0.96	0.82	0.61	0.89	0.66	0.97	0.98	0.96	0.05	0.08	0.1

From the above results, Random Forest Regression performed better than CART Decision tree regression and linear regression for all datasets except heart disease. Although manually changed the hyper parameters like n_estimators and max_depth could not improve the metric for heart disease.

7.2 BOOSTING

Boosting is an ensemble learning method that combines a set of weak learners into a strong learner to minimize training errors. In boosting, a random sample of data is selected, fitted with a model and then trained sequentially—that is, each model tries to compensate for the weaknesses of its predecessor. With each iteration, the weak rules from each individual classifier are combined to form one, strong prediction rule. Refer to section 5 in Appedix for Boosting workflow.

7.2.1 ADA-BOOST

This method operates iteratively, identifying miss classified data points and adjusting their weights to minimize the training error. The model continues optimize in a sequential fashion until it yields the strongest predictor.

7.2.2 Gradient boosting

Unlike, Ada-boosting algorithm, the base estimator in the gradient boosting algorithm cannot be mentioned by us. The base estimator for the Gradient Boost algorithm is fixed and i.e. Decision Stump. Like, Ada-Boost, we can tune the n_estimator of the gradient boosting algorithm. However, if we do not mention the value of n_estimator, the default value of n_estimator for this algorithm is 100.

Gradient boosting algorithm can be used for predicting not only continuous target variable (as a Regressor) but also categorical target variable (as a Classifier). When it is used as a regressor, the cost function is Mean Square Error (MSE) and when it is used as a classifier then the cost function is Log loss.

7.2.3 Extreme gradient boosting or XGBoost

XGBoost is an implementation of gradient boosting that's designed for computational speed and scale. XGBoost leverages multiple cores on the CPU, allowing for learning to occur in parallel during training

Advantages: Ease of Implementation, Reduction of bias and Computational Efficiency

Disadvantages: Intense computation

7.2.4 Performance on DataSets

The results of implementing various classification algorithms utilizing SCIKIT libraries on Heart Disease, Credit Defaulter, and Gender Voice datasets considering accuracy as a metric are as follows. Dataset analysis related code links can be found at Credit.ipynb , Heart.ipynb ,Gender_Voice.ipynb

Algorithm	Heart Disease DataSet	Credit Defaulter DataSet	Gender Voice DataSet
ID3	28.39	61.00	NA
C4.5	93.82	NA	NA
CART	76.54	74.33	96.85
RANDOM FOREST	83.95	77.66	98.21
ADA BOOST	77.78	74	97.48
GRADIENT BOOSTING	76.75	75	97.69

8 On-going advancements in Decision Trees

In this section, we will detail out latest improvements built on the decision trees

8.1 Hierarchical SoftMax

Hierarchical softmax (HSM) is designed for multi-class classification. For multi class problems there is one and only one label assigned to each instance and the marginal probabilities sum up to 1. The HSM classifier h(x) takes form of a label tree and encode all labels using a prefix code and such code can be used to form a tree from the root to a leaf node.By using logistic loss and a linear model in each

node zi for estimating P(zi | zi-1, x), we obtain the formulation of HSM. To observe the improved performance, we ran the HSM algorithm on the fruit data set for multi class classification using the code Fruit_Classification_Comparision.ipynb , utilizing napkinxc library. There is a tremendous improvement in performance using HSM with an accuracy of 99.02 compared to CART decision tree with an accuracy of 74.38.

8.2 Born-Again Tree Ensembles

The process of constructing a single decision tree of minimum size that reproduces the exact same behavior as a given tree ensemble in its entire feature space. To find such a tree, we develop a dynamic-programming based algorithm that exploits sophisticated pruning and bounding rules to reduce the number of recursive calls. As a perspective for future work, it is recommended to progress further on solution techniques for the born-again tree ensembles problem, proposing new optimal algorithms to effectively handle larger datasets as well as fast and accurate heuristics. Heuristic upper bounds can also be jointly exploited with mathematical programming techniques to eliminate candidate hyperplanes and recursions. Another interesting research line concerns the combination of the dynamic programming algorithm for the construction of the born-again tree with active pruning during construction,leading to a different definition of the recursion and to different base-case evaluations. Finally, it is recommended to pursue the investigation of the structural properties of tree ensembles in light of this new born-again tree representation.

8.3 One-vs-rest & Binary relevance

One-vs-rest (OvR) is a heuristic method for using binary classification algorithms for multi-class classification. It involves splitting the multi-class dataset into multiple binary classification problems. Binary relevance is the intuitive way of learning the multi-class problems. It works by decomposing the multi-class learning task into a number of independent binary learning tasks (one per class label).Explored the performance using these heuristic on ZOO dataset using napkinxc library and code can be found at ZOO_Classification_OVR.ipynb . These algorithms efficiently improved the performance of multi-classification problem to an accuracy around 96-100 percentage.

8.4 Neural-Backed Decision Trees

Neural-Backed Decision Trees (NBDT) returns sequential decisions leading to a prediction. NBDT replace a neural network's final linear layer with a differentiable sequence of decisions and a surrogate loss.This forces the model to learn high-level concepts and lessens reliance on highly-uncertain decisions with improved accuracy and interpretability.NBDT uses a Tree Supervision Loss, proving that the hierarchical softmax is not the only way to train a hierarchical classifier. With the help of the pretrained model 'SoftNBDT' on CIFAR10 dataset from the NBDT library, which can be found at NBDT.ipynb , tested the performance of the model with random images from 'pixabay' and observed that even though the dataset does not contain the images of 'bear', 'zebra','wolf', 'elephant', 'fish' the model predicted them as animals ('dog','horse','deer' etc.) instead of automobile or airplane. Tried to create a new model replacing the 'ResNet18' with 'VGG' from 'TORCHVISION.MODELS' to run on FRUIT dataset but with the code errors and time constraint could not succeed to the completion.

8.5 Improved Decision Tree(IDT)

Classification algorithms are tested on social networking data. Here, a new classification algorithm called the improved Decision Tree (IDT) is introduced. This model provides better classification accuracy than the existing systems for classifying the social network dataset. The performance of some familiar classification algorithms like SVM, Naive Bayes, KNN etc., with respect to accuracy is compared with this IDT algorithm. This algorithm provided much better accuracy compared to that of the remaining traditional algorithms. In future, it is recommended to enhance this algorithm to consider other factors in the confusion matrix as well along with accuracy.

Appendix

[1] Gini Impurity

Gini impurity is a metric to describe the purity or homogeneity of a node. A node is pure if the Gini impurity is zero i.e., all the samples belong to the same class. And the total Gini impurity is calculated as the weighted average of Gini impurities.

$$GiniImpurity = 1 - Gini = (1 - \sum p_i^2)$$

where p[i] is the fraction of samples belonging to ith class.

[2] MSE

$$MSE(t) = \frac{1}{N_t} \sum y^{(i)} - y_t^{\wedge}$$

Here, Nt is the number of training samples at node t, Dt is the training subset at node t, y(i) is the true target value, and ŷt. is the predicted target value (sample mean):

$$y_t^{\wedge} = \frac{1}{N_t} \sum y^{(i)}$$

[3] R2 score

R2 score determines how the model is fitted to data by comparing it to average line of dependent variable. If the score is closer to 1, then it indicates that the model performs well.

[4] Bootstrap aggregation workflow

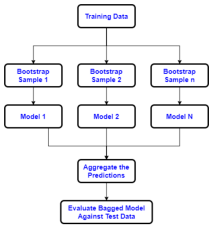

Figure 1: Bootstrap aggregation workflow

[5] Boosting workflow

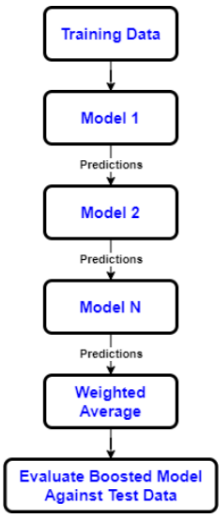

Figure 2: Boosting workflow

[6] Chi-square

Higher the value of Chi-Square, higher the statistical significance of differences between sub-node and Parent node.

Chi-square = ((Actual — Expected)² / Expected)¹/2

[7] Exploratory Data Analysis Strategies

[7.1]Information about the dataset

Find the shape of the data,data type of individual columns

```
data.info()
```

Figure 3: Information about the Dataset

[7.2]Descriptive statistics of numerical columns

This function is implemented for calculating some statistical data like percentile, mean and standard deviation of the numerical values of the Series or DataFrame.

```
In [6]: df.describe(percentiles=[0.10,0.25,0.50,0.75,0.90]).T   # five point summary of the continuous
                                                               #additional percentiles
```

Figure 4: Descriptive Statistics including 5 point summary

[7.3]Checking for Missing Values

This function is implemented to check for any missing values in the dataset

```
#null value counts in each column
data.isnull().sum()
```

Figure 5: Missing values check

[7.4]Univariate Analysis

Here We have just defined a function so that I can use it through out the code instead of repeating again.This function plots count plot for categorical variables and distribution and boxplots for numerical variables

```
In [13]: def unilateralAnalysis(attributename, categorical=False):
             # For categorical attributes, plot a countplot
             if categorical:
                 sns.countplot(df[attributename])
                 plt.show()
             else:
                 plt.subplot(2,2,1)
                 # Distribution Plot
                 sns.distplot(df[attributename])
                 plt.subplot(2,2,2)
                 # Boxplot
                 sns.boxplot(df[attributename])
                 plt.show()
                 print("Mean: {:.2f}".format(df[attributename].mean()))
```

Figure 6: Univariate Analysis

[7.5]Bivariate Analysis

Since we have understood the univariate analysis, let's continue to understand the effect they have on the target variable and their relationship between each other.

```
In [15]: def CategoricalAnalysis(attributename, countplot=False, variable2 = None):
             if countplot:
                 sns.countplot(x=attributename,data=df,hue='Output Variable')
                 plt.show()
             else:
                 # Boxplot
                 sns.boxplot(x=attributename,y=variable2,hue='Output Variable',data=df)
                 plt.show()
```

Figure 7: Bivariate Analysis

[7.6]Correlation Analysis

This function is implemented to compute the pairwise correlation of columns, excluding NA/null values. It returns correlation matrix DataFrame.

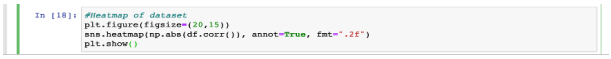

Figure 8: Correlation Analysis

[7.7]Pairwise relationships in a Dataset

This function is implemented inorder to understand the pairwise relationships in a dataset.

Figure 9: pairwise relationships in a dataset

References

[1] Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81–106

[2] UCI Zoo Dataset - https://archive.ics.uci.edu/ml/datasets/zoo

[3] ID3 Algorithm to predict weather https://iq.opengenus.org/id3-algorithm/

[4] A Step By Step C4.5 Decision Tree Example https://sefiks.com/2018/05/13/a-step-by-step-c4-5-decision-tree-example/

[5] CART Classification-
https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning
https://towardsdatascience.com/decision-tree-algorithm-in-python-from-scratch-8c43f0e40173
https://www.kaggle.com/code/prashant111/decision-tree-classifier-tutorial/notebook

[6] CART Regression-
https://towardsdatascience.com/https-medium-com-lorrli-classification-and-regression-analysis-with-decision-trees-c43cdbc58054

[7] Random Forest Regression-
https://levelup.gitconnected.com/random-forest-regression-209c0f354c84

[8] napkinxc Models-
https://napkinxc.readthedocs.io/en/0.4.2/python_api/napkinxc.models.HSM.html
https://napkinxc.readthedocs.io/en/0.4.2/python_api/napkinxc.models.OVR.html
https://napkinxc.readthedocs.io/en/0.4.2/python_api/napkinxc.models.BR.html

[9] Research paper on Hierarchical softmax -
https://proceedings.neurips.cc/paper/2018/file/8b8388180314a337c9aa3c5aa8e2f37a-Paper.pdf

[10] Research paper on Binary Relevance-
https://link.springer.com/article/10.1007/s11704-017-7031-7

[11]NBDT Research paper-
https://research.alvinwan.com/neural-backed-decision-trees

[12] NBDT Pretrained models-
https://github.com/alvinwan/neural-backed-decision-trees

[13] DATASETS for CART Classification-
https://www.kaggle.com/datasets/khushboosrivastava2/gender-voice-predictiondecision-tree-modeling
https://www.kaggle.com/datasets/moltean/fruits

[14] DATASETS for CART & Random Forest Regression-
https://www.kaggle.com/datasets/yasserh/student-marks-dataset/code
https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who
https://www.kaggle.com/datasets/hellbuoy/car-price-prediction
https://www.kaggle.com/datasets/navink25/framingham

[15] Ensemble learning - https://en.wikipedia.org/wiki/Ensemble_learning

[16]Bagging - https://www.ibm.com/cloud/learn/bagging

[17]Random Forest - https://medium.com/@harshdeepsingh_35448/understanding-random-forests-aa0ccecdbbbb

[18]Bosting Algorithm- https://towardsdatascience.com/boosting-algorithms-explained-d38f56ef3f30