Durham University

Machine Learning Assignment(Z0193145)

Assessment: Pokemon Data Analysis for Type Prediction

Course: Machine Learning (23/24)

Module Code: MATH42815_2023

1. Introduction

The main problem statement for this assignment is to create a model to predict the type of Pokemon based on the data given. This dataset has information about 810 Pokemon from all Seven Generations and 18 type1, the goal in this assignment is to use machine learning techniques to classify the type of Pokemon.

Each Pokemon has 41 features, which describes about their strengths, abilities, powers, and other characteristics. The dataset has so many features comprising of numerical and classification data which makes it bit challenging to analyse.

For this assignment I has chosen two modelling approaches. They are classification trees and random forests.

Classification Trees:
I have chosen this because this can handle the categorical variables effectively along with non-linear relationships which is important for this task. Apart from this classification trees has the capability to capture important features for classification as there are some many features to decide. They have the ability to partition the features based on the values of attributes which make me to choose this as first model

Random Forests:
The next model I have used Random forests. I have used this as it offers a technique that combines multiple decision trees to improve prediction accuracy and reduce overfitting. This constructs multiple decision trees and then average their predictions. By doing this random forests are enhancing the stability of the model. Apart form this, they can handle large datasets with lot of features, making them suitable for the dataset like Pokemon which is diverse in its features.

These two approaches were chosen because of their proven suitability for the classification data.

2.Data Cleaning and Exploratory Data Analysis

The first thing the program does is read data from the "pokemon.csv" CSV file that is stored on my local computer. Subsequently, I looked up the data's dimensions to have an understanding of its

features and length. To obtain a summary of the data, I used the summary function. To better understand the code, I must reorder the data, such as moving the name columns first.

Finding the columns with the missing values is the first thing I do. From there, I can see that the columns for height_m, percentage_male, and weight_kg have the missing values. In order to address these missing values, I have substituted the corresponding column's mode for each missing value. I have to make sure that every column has the correct data type. But for the column capture_rate when all values are just numerical values, I can tell that it is a string. To clear up any confusion, I printed their values. From there, I saw that one particular value was "30 (Meteorite)255 (Core)," whereas the others were all numerical numbers. I changed that value to 30 to improve the data's interpretability. Upon closer inspection, we can find that type 2 contains a large number of null values, making it unsuitable for interpretation. We may locate the mode there and use its value to substitute the null values. However, because there are so many null values, this leads to misunderstanding. For the time being, I have eliminated type 2 and combined type 1 and type 2 to form a new column type. Then, since it is irrelevant to the analysis of the data, I deleted this column from the data frame along with japanese_name and pokedex_number. I saw in the following step that there is a column with the abilities that contains many strings, the quantity of which varies according on the pokemon. As a result, I deleted this column and created a new one with its count.

This data contains some outliers. I created a graph by using the IQR method to identify outliers in each column. This demonstrates how numerous outliers there are in certain columns. These outliers are Pokemon powers that indicate which pokemon are exceptionally strong or resilient against a particular ability; they cannot be changed or eliminated. We can combine types 1 and 2 to get a column of type and build a model to find that type, albeit at a high computational expense. As a result, after eliminating type, I only used type 1 to forecast in this data.

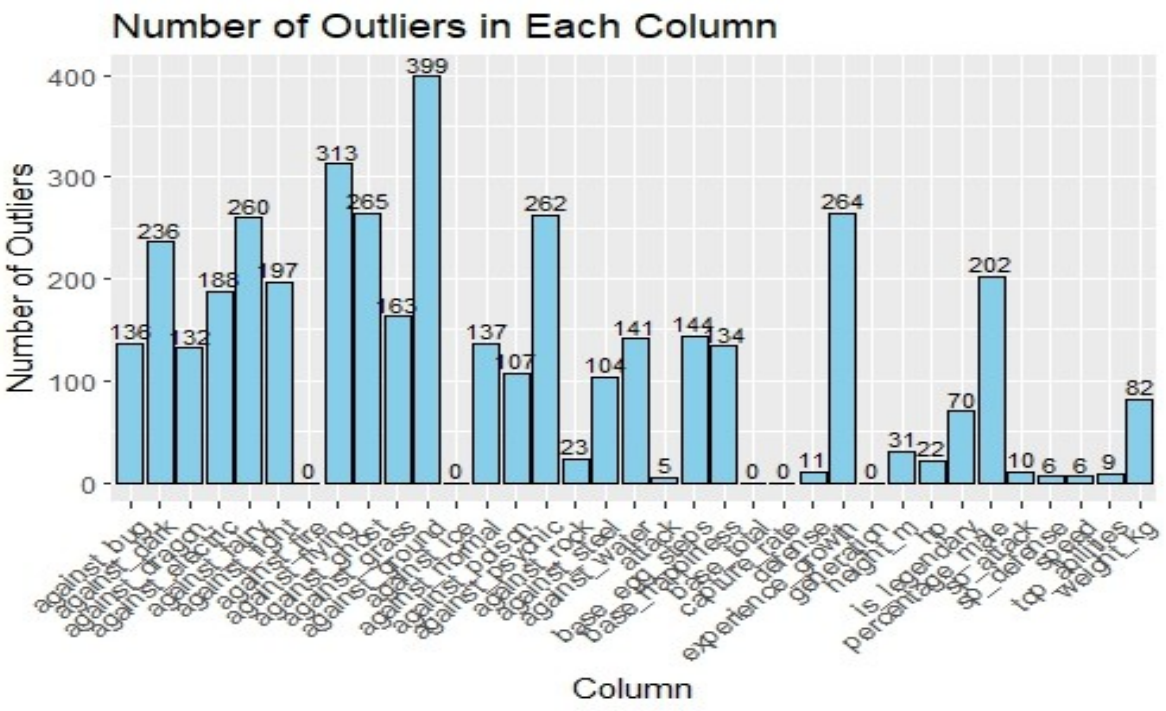


Fig 1:- Number of outliers in each column

Since type 1 is our primary focus, let's examine the number of each type 1. From this we can see that there are most pokemon are in type water, normal,grass, bug, psychic and fire from which the count falls.

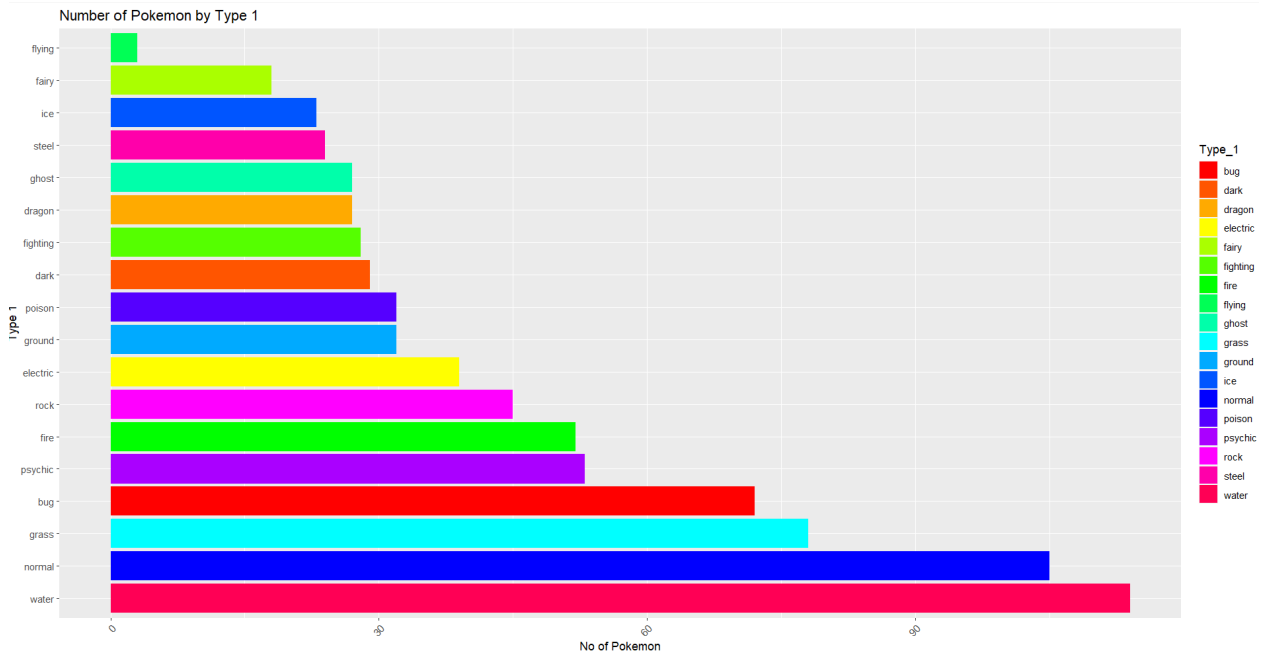


Fig2:Type vs No of pokemon

There are so many variables to be processed. It always better to remove some variables which are highly correlated as this may create overfitting. For this I have plotted corrplot and a heat plot which has showed all the correlated variables. I have removed all the variables which are correlated by more than 0.65. to other variables. These are "against_ghost" to "against_dark" , "base_total" to "attack", "is_legendary" to "base_egg_steps", "capture_rate" to "base_total" , "sp_attack" to "base_total" and "sp_defense" to "base_total" of which I have used only one from this each set of variables and removed other.

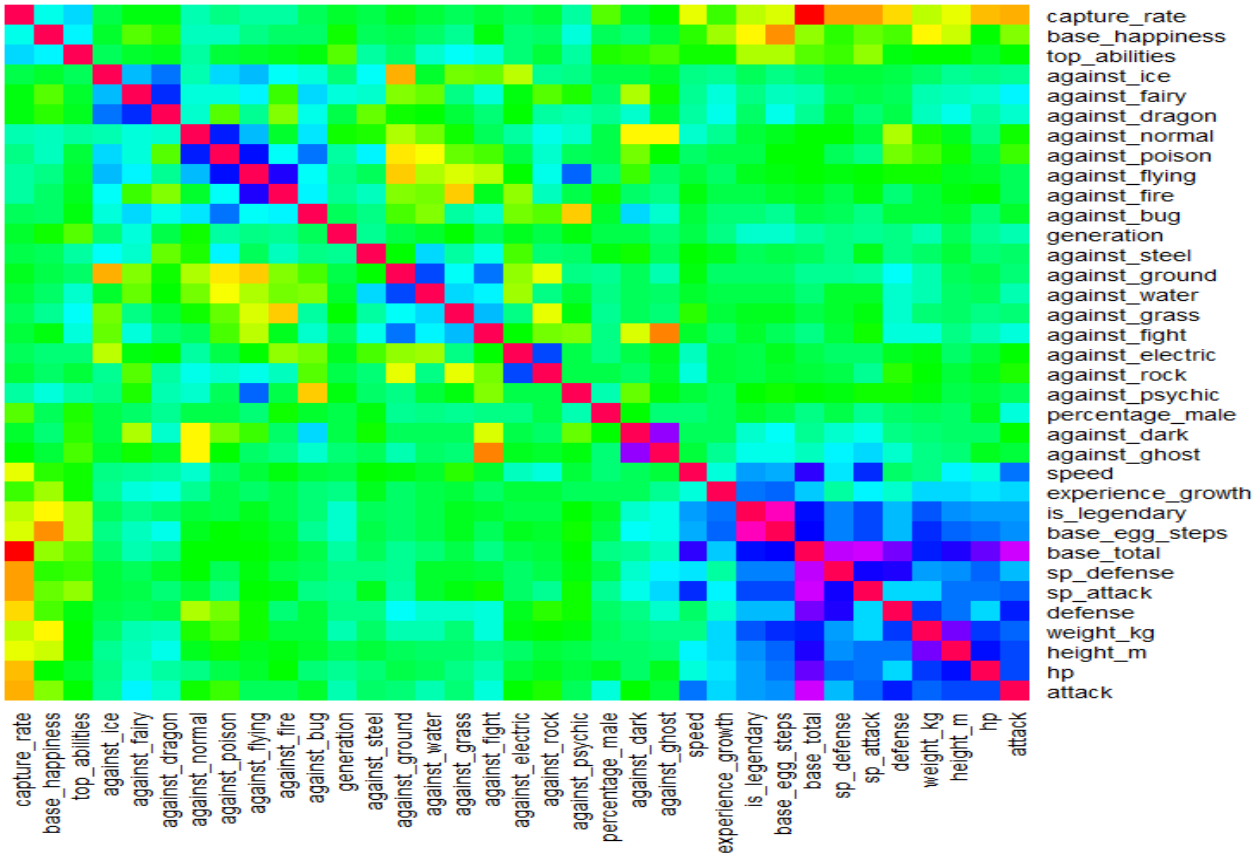


Fig3: **Heat map of correlation plot**

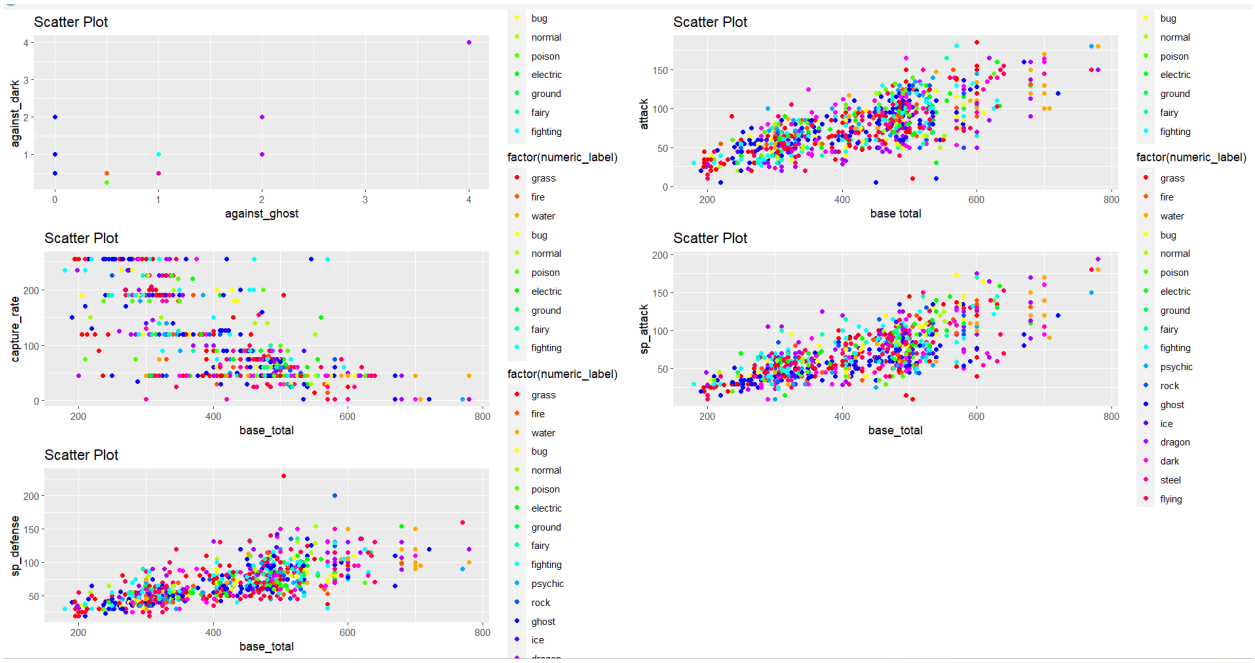


fig 4: plots of individual correlated variables

I have created new feature by combining the columns of the powers which start with the string “against_” as they looks same but not correlated and moreover I want to check how powerful they are against all possible things. For this I created a new column called “sum_column” which has the sum of all this. There are almost 600 types of classifications in this data. This column creates nothing but a confusion. So this column has been removed and the final data set has 31 columns to be processed.

After this I have normalized the data before using this data for a ML model.

3.Modelling and validation

In this task I have used classification trees and random forests over other methods. Here are the reasons:-

i)Random Forests are favored over other methods due to their resilience to variance and overfitting. Random Forests reduce the chance of individual decision tree of picking up some noise in the data by building several trees and combining their forecasts. This method works well with a variety of datasets that include complex correlation between features and the target variable. If we see this data it has lot of complex features to access and lot of data. Apart from that I need to divide data to 18 type1. So, random forest is a best choice to go with.

ii)On the other hand, classification trees provide decision-makers with simplicity and interpretability. It creates easily understandable trees that stack holders can understand by dividing features to hierarchical segments . The identification of actionable insights helps in decision-making processes. This clearly show how data is getting divided and apart from that as asked in the question I need to choose on from CART and regularization technique. In this CART out perform the regularization technique.

iii)I used these two because they have thoughtful consideration of bias and variation. Random Forests reduce variation by averaging predictions from several trees, whereas Classification Trees divide the feature space recursively according to Gini impurity or information gain to control bias. This equilibrium guarantees the resilience of the model while maintaining its interpretability and computational effectiveness. The size, complexity, and distribution of the dataset are taken into account while selecting between Random Forests and Classification Trees. These models work especially well with diverse data formats, varied feature importance, and nonlinear interactions in datasets. Their adaptability makes it possible to classify them effectively in a variety of fields, such as marketing, finance, healthcare, and more.

3.1.Model 1:- Using Classification Trees

1. Data Preparation:

The first step starts by splitting up the training and test data. I have used the sample() function to randomly select 75% of the data for training, and I have used set.seed(123) to produce reproducability.

2. Start of Model Development:

I have started the script by initiating the construction of a grid of parameters (param_grid) for the CART model. Parameters include minsplit, minbucket, and maxdepth, essential for controlling tree growth and preventing overfitting.

```
param_grid <- expand.grid(
  minsplit = c(10, 20, 30),
  minbucket = c(5, 10, 15),
  maxdepth = c(3,6,9,12,15)
)
```

I have used rpart() function which iteratively move across the various parameter combinations(param_grid) which was build by the CART model. I have used Nested loops in this

for cross-validation, where the dataset is partitioned into folds, trained, and evaluated iteratively to estimate model performance accurately.

3. Hyperparameter Tuning:

The accuracy of each parameter combination is assessed in order to determine which model performs the best. These accuracy levels are plotted by the script, where the highest accuracy level is indicated by a red point. The CART model's ideal hyperparameters are chosen with the help of this procedure. From this I have got the best model is at minsplit = 10, minbucket = 5, maxdepth = 9.

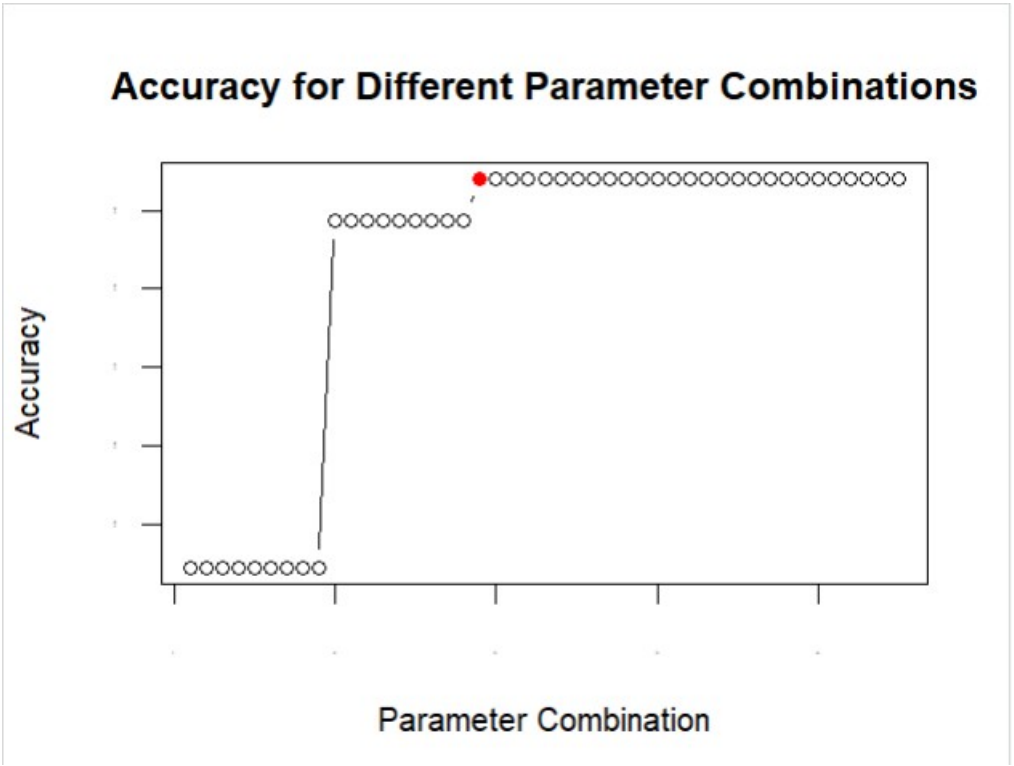


Fig 5: Accuracy for different parameter combinations

4. Finding best cp value

It's time to modify the model by determining the optimal cp value. By using the above parameters I have utilized range of cp values from 0.01 to 0.5 with interval of 0.01. In brief I built a function that sets up a range of complexity parameters cp for cross-validation and then trains the decision tree model using cross-validation with the specified parameters. This makes it possible to choose the ideal complexity parameter for the decision tree model by assessing the model's performance throughout the range of values we had provided for the complexity parameter.

5.Complete algorithm used:

1) Create a grid (param_grid) containing combinations of hyperparameter values: minsplit, minbucket, and maxdepth.

2)Set best_accuracy to 0 to track the best performance. Set best_model to NULL to store the best-performing model. Create an empty vector accuracies to store accuracy scores.

3)For each combination in param_grid, extract parameter values (minsplit_val, minbucket_val, maxdepth_val). Train a decision tree model (tree_model) using rpart() with specified parameters and control. Perform 5-fold cross-validation, splitting data into 5 folds, training and evaluating models on each fold, and computing accuracy. Average accuracy across folds and store it.

4)Plot accuracies to observe trends across different parameter combinations.

5)Find the index of the highest accuracy (best_index). Retrieve corresponding best parameters (best_params) from param_grid. Print best parameters and accuracy.

6)For pruning create a control object (cv_model) with a range of complexity parameter (cp) values. Train a tree (cv_results) using rpart() with cost-complexity pruning. Print cross-validation results to visualize pruning effects and select an optimal cp value.

7)Based on cross-validation results, choose the model with the best balance of accuracy and complexity (optimal cp). This final model represents the best-performing model for the given dataset.

6. finally building the model

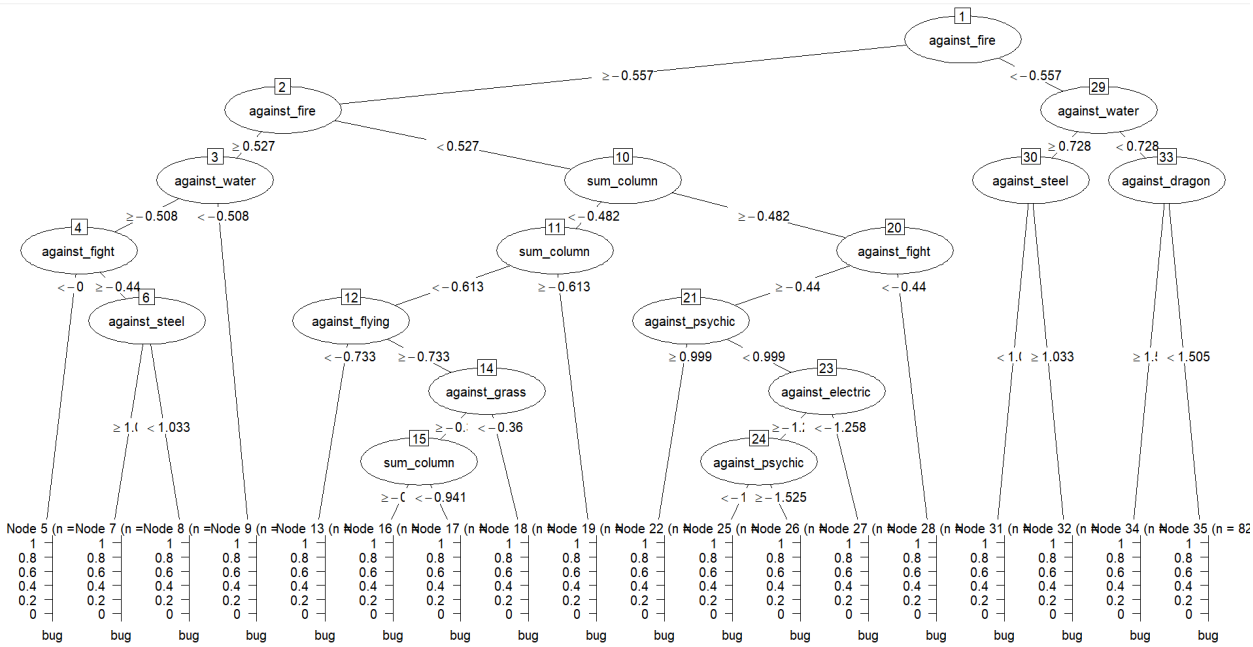
after getting all the best parameters I have build the model.

```
tree_model <- rpart(type1 ~ .,
 data = train_data,
 method = "class",
 control = rpart.control(cp = 0.01, minsplit = 10, minbucket = 5, maxdepth = 9))
```

At last, I built a model. I set the parameter cp to 0.01 since I don't want the tree to be very complicated which I got from above steps. This keeps the tree more straightforward. I also established some guidelines for when the tree should cease to split. For instance, minsplit = 10 requires at least 10 data points to make a choice, and minbucket = 5 requires at least 5 data points for each group it forms which I have determined from above steps. Furthermore, I assert that the tree shall not go below nine levels (maxdepth = 9). This keeps it from becoming overly technical.

7. Model evaluation and visualization.

Here I have to test the model as we have already seperated the test data, i use that to test our model. Below are the pictures of the model and confusion matrix. This model has the accuracy of 80.6%



this has the overall accuracy of :80.6%

```
Overall Statistics

               Accuracy : 0.806
                 95% CI : (0.7444, 0.8582)
    No Information Rate : 0.1642
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7869

 Mcnemar's Test P-Value : NA
```

confusion matrix

```
> conf_matrixtable
          Reference
Prediction bug dark dragon electric fairy fighting fire flying ghost grass ground ice normal poison psychic rock steel water
  bug       10    0      0        0     0        0    0      0     0     0      0   1      0      0       0    0     2     0
  dark       0    3      0        0     0        1    0      0     0     0      0   0      0      0       0    0     0     0
  dragon     0    2      8        0     0        0    0      0     1     0      1   0      0      0       0    0     0     2
  electric   0    0      0        7     0        0    0      0     0     0      0   0      0      0       0    0     0     0
  fairy      0    0      0        0     4        0    0      0     0     0      0   0      0      0       0    0     0     0
  fighting   0    0      0        0     0        4    0      0     0     0      0   0      1      0       0    0     0     0
  fire       0    0      0        0     0        0   10      0     1     0      0   0      0      0       1    0     0     0
  flying     0    0      0        0     0        0    0      0     0     0      0   0      0      0       0    0     0     0
  ghost      0    0      0        0     0        0    0      0     2     0      0   0      0      0       0    0     0     0
  grass      1    0      0        0     0        0    0      0     1    33      0   0      1      0       0    0     0     0
  ground     0    0      0        0     0        0    0      0     0     0      5   0      0      0       0    0     0     0
  ice        0    1      0        0     0        0    0      0     0     0      0   5      0      0       0    0     0     0
  normal     0    0      0        0     0        0    0      0     0     0      0   0     17      0       0    0     0     0
  poison     0    0      0        0     0        0    0      0     2     0      0   0      0      4       0    0     0     0
  psychic    2    0      0        0     0        1    0      1     2     0      1   0      0      2       9    0     0     0
  rock       1    0      0        0     0        0    0      0     0     0      0   0      1      0       0    9     0     2
  steel      0    1      0        1     0        1    0      0     0     0      0   0      0      0       0    0     4     0
  water      0    0      0        0     0        0    0      0     0     0      0   0      0      1       0    3     0    28
```

From the confusion matrix and the accuracy it is clear that the model is working good. The classification model's evaluation yields an overall accuracy of 80.6%, greatly above the No Information Rate (NIR) of 16.42%, with a 95% confidence interval between 74.44% and 85.82%. The Kappa coefficient, which gauges agreement above random variation, is 0.7869, indicating strong agreement. The McNemar test did not apply. The model's sensitivity and specificity vary depending on the class, with some showing higher detection rates than others. Across classes, balanced accuracy falls between 0.61111 to 0.9911. Although class performance varies, overall model performance is good, indicating that it can identify between many classes with a noteworthy overall accuracy of more than 80%.

3.2.Model 2:- Using random forests

1. Random Forest Model Training:

In the second model I have used Random Forest model by using the randomForest() function. When compared with CART, Random Forests has capability to learn method that constructs multiple decision trees and combines their predictions to improve accuracy and mitigate overfitting.
rf_model <- randomForest(factor(type1) ~., data = train_data)

the above model has 500 trees and has the OOB estimate of error rate: 9.5%.

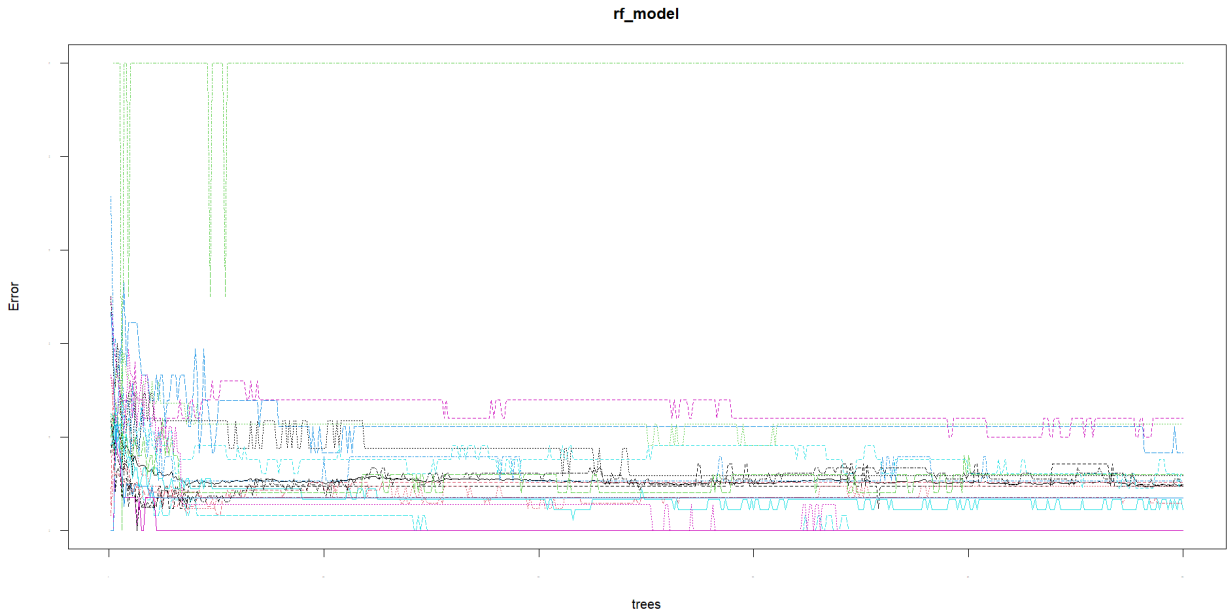


Fig 6: rf_model Error Vs Trees

the above graph shows error rate over number of trees for the rf_model I had built.

2 Model Evaluation and Visualization:

It's important to assess the Random Forest model's performance using a separate dataset, known as the test set, after it has been trained. As I have already created the test_data, the given script evaluates the Random Forest model's performance.

predictions <- predict(rf_model, newdata = test_data)

```
Overall Statistics

               Accuracy : 0.9303
                 95% CI : (0.8859, 0.9614)
    No Information Rate : 0.1642
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9235

 Mcnemar's Test P-Value : NA
```

Prediction \ Reference	bug	dark	dragon	electric	fairy	fighting	fire	flying	ghost	grass	ground	ice	normal	poison	psychic	rock	steel	water
bug	12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
dark	0	7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
dragon	0	0	8	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
electric	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
fairy	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0
fighting	0	0	0	0	0	5	0	0	0	0	0	0	1	0	0	0	0	0
fire	0	0	0	0	0	0	10	0	0	0	0	0	0	0	1	0	0	0
flying	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ghost	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0
grass	0	0	0	0	0	0	0	0	0	33	0	0	1	0	0	0	0	0
ground	0	0	0	0	0	0	0	0	1	0	7	0	0	0	0	0	1	0
ice	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0
normal	0	0	0	0	0	0	0	1	0	0	0	0	18	0	0	0	0	0
poison	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0
psychic	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0
rock	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0
steel	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	0
water	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	32

3.3. Final result:

From the result of random forest I can conclude that it has the best prediction accuracy. The model works perfectly well and able to predict with the accuracy of 93.03. A thorough analysis of the model's classification performance in several categories is given by the confusion matrix. The actual class is represented by each column, while the anticipated class is represented by each row. For example, the model accurately identified 12 'bug' and 7 'dark' cases. A few instances were also misclassified, as seen by the off-diagonal parts. With a 95% confidence interval between 88.59% and 96.14%, the aggregate data demonstrate a high accuracy of 93.03%, which is much better than the No Information Rate (NIR) of 16.42%. Significant agreement is indicated by the Kappa coefficient of 0.9235, which measures agreement beyond chance.

Additionally, the model's accuracy in detecting instances inside each class is shown by the sensitivity and specificity values for each class. For the majority of classes, the model exhibits good sensitivity and specificity, and its balanced accuracy ranges from 0.500 to 1.000 in various categories. This suggests that the model performs a generally good job of differentiating between the different classes.

All things considered, the model performs admirably in a variety of classes, attaining high accuracy and showcasing a noteworthy capacity to accurately identify occurrences inside each category.

Classification tree Vs Random Forest From above graphs, accuracy and confusion matrix, it is obviously clear that random forest out perform. In terms of accuracy (93.03% vs. 80.6%) and Kappa coefficient (0.9235 vs. 0.7869), Model 2 outperforms Model 1, suggesting superior agreement. More accurate estimates are provided by Model 2, which has a narrower confidence interval. Additionally, it provides specificity and sensitivity values for every class, enabling in-depth evaluation. Mcnemar's Test P-Value is absent from both models. All things considered, Model 2

performs better overall on a number of assessment parameters, making it the recommended option.

4.conclusion:-

To conclude, I have picked classification trees and random forests for this Pokemon data analysis because they're good at handling when we have lots of different information. Classification trees help us understand how the data is split up into different groups, making it easier to see why Pokemon are categorized the way they are. Random forests are like a bunch of trees working together, which helps them make really good predictions. I cleaned up this data and looked at it closely to make sure it was ready before starting this model. Then I have used the data to train the classification tree and random forest models. These models helped me to figure out which features are important for predicting Pokemon types. In the end, I found that the random forest model was the most accurate in predicting Pokemon types. These techniques are helpful for understanding data and making decisions, by using this model I can find the pokemon type. This data has so many features and it is so hard to choose which features are the best. Even then I have removed some irrelevant features and made some new features if necessary. One limitation about this data is that it has so many outliers as these outliers are the powers of the pokemon I just considered that might be some pokemon are more powerful than others. This work could be further developed in such a way that the classification can be done by based on combining type1 and type2 by merging them. As this creates more classes in type variable this could be computationally pricy.

5.Bibliography:-

Banik, R. (2017) *The Complete Pokemon Dataset*, *Kaggle*. Available at: https://www.kaggle.com/datasets/rounakbanik/pokemon (Accessed: 12 February 2024).