COMM053: Practical Business Analytics

Department of Computer Science

What factors are responsible for corporate employee attrition?

Group V

Anthony Awobasivwe	6442385
Ana Paniagua Cruz	6478023
Abdulazeez Mahmud-Ajeigbe	6778114
David Amusan	6763674
Humza Malik-Ramzan	6435376
Jason Jay Dookarun	6494278

Table of Contents

Table of Figures

Introduction

How much does it cost to lose an employee? Estimates indicate the cost of losing and replacing an employee to be approximately 1.5 - 2 times the employee's annual salary (McFeely & Wigert, 2019). The cost of replacement not only has a financial impact on the organisation but also disrupts business flow due to the time taken in hiring and training recruits. An employee leaving, otherwise known as 'attrition', can have detrimental effects on an organisation. This is why we would like to investigate which factors have a more significant impact on attrition and build a model to predict this through classification methods.

Attrition is a key area of interest due to its negative effects on the organisation as a whole. When employees decide to leave, their departure harms the whole team, this may present as lower engagement and efficiency, which could result in an organisation not meeting deadlines and in turn, damaging business productivity and reputation. Furthermore, it could also point to deeper issues such as discrimination in the workplace, if there is evidence that a particular demographic of people (women, older workers, ethnic minorities) leaving at a higher rate than the rest. These are direct consequences of attrition.

Both the high replacement cost and potential reputational damage drive the need for employee retention and motivate employers to keep their employee turnover low. But is there any way to predict who will leave the company, and to what level of accuracy can we do this?

To investigate the points mentioned above a dataset by IBM HR Analytics Employee Attrition and Performance dataset, published by IBM and Abhishek Kumar via Kaggle (IBM, n.d.) will be used. A public dataset repository platform. A Human Resources (HR) dataset was chosen because the main purpose of this department is to enhance employee experience and strengthen business operations. As a result, HR records a variety of data, giving great insight into what employees want from an organisation and more importantly, how to retain them.

This work will focus on a fictional dataset, namely WA-Fn-UseC_-HR-Employee-Attrition.csv. The dataset contains a single file on employees from various sectors, with the following attributes:

- Employee Details and Tenure
- Education Background of Employee
- Job Performance and Involvement
- Relationship Satisfaction
- Environment Satisfaction
- Work-Life Balance
- Marital Status and Satisfaction

Additionally, the dataset generated by IBM provides additional fields related to each of the categories mentioned above. For example, employee details include data on the following metrics:

- Job satisfaction
- Total Working Years

- Years in Current Role
- Years since Last Promotion

The data fields mentioned above are both discrete and continuous. A summary of each field, its type, and a brief description is provided in Appendix A.

The project aims to use the chosen dataset to investigate which factor has the greatest effect on attrition. We will do so based on the initial prediction that experience level, distance from home to the office, salary, and job satisfaction, will have the greatest effect on attrition. A detailed plan of our investigation is provided in Figure 1. Where each of the four factors listed, indicates an area of interest to be explored as a Key Performance Indicator for this project.

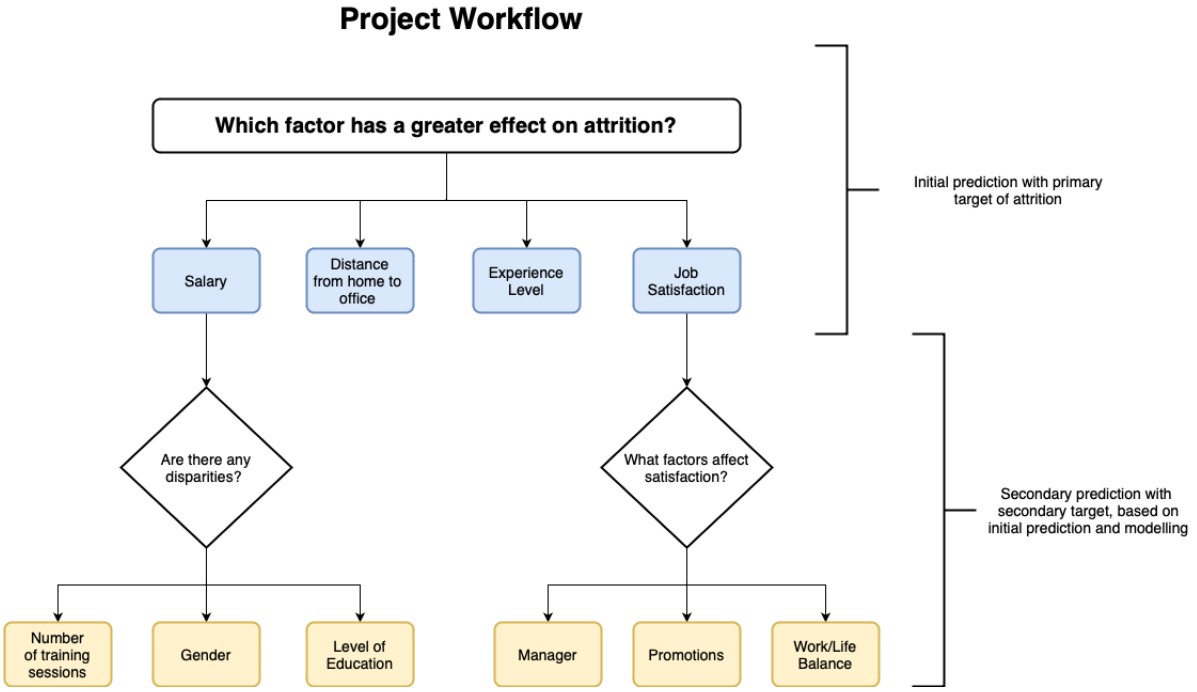


Figure 1: Key Performance Indicators (KPI)

Questions such as distance from the office, have become all the more relevant with rising fuel prices and as employees shift from working remotely, to going back into the office. As such, we predict distance from the office will be a big factor in predicting attrition.

Moreover, both salary and career growth are considered to be some of the most important qualities when applicants consider joining a company. This leads us to predict that these factors will also be significant as employees consider leaving a company.

Our results will inform a business' future actions in response to attrition. Once the business is aware that an employee is likely to leave, it can act accordingly (Fryre, 2018) by; increasing employee benefits to retain workers, assessing whether the rest of the team can absorb the role or if not, beginning the hiring process to fill that role. Our model will also serve as a tool within the hiring process, to ensure successful recruits will be more likely to stay with the company for a minimum period.

It is considered common knowledge that HR poses a significant factor when it comes to an individual's decision whether to leave or stay in a company. Hence researchers have tackled the question, 'what factor(s) affect company turnover the most'? Studies have proven that company turnover is influenced by a combination of variables including satisfaction, pay, performance and an individual's commitment to an organisation. Price and Mueller (1981, 1986) highlighted the importance of satisfaction in a workplace, and its indirect correlation with an employee's commitment within the organisation, leading to poor employee retention. This statement was taken further by Howard and Homma (2001), who found that job satisfaction as a singular variable does not influence company turnover job commitment should be an independent variable that influences company turnover. It all depends on the HR practices and an individual employee's characteristics, what motivates them and how much of a fit they are to the organisation's goals, this aids in their commitment to the role, in turn affecting their decision to stay or leave Kund S. et al. (2007).

Other pieces of literature concentrate on other variables. For instance, Griffeth et al (2000) indicated that the relationship between employee wage and performance greatly impacts employee turnover. High performers not receiving sufficient rewards could indicate a lack of progress within the company leaving individuals looking for opportunities elsewhere. The lack of wages could also indicate environmental conditions. Recent studies show that the work environment can have a large impact on employee turnover and should be carefully handled by employers Maxcalabro (2017).

Methods

Pre-Processing

To follow the CRISP-DM cross-industry standard, we conducted data pre-processing to produce a data set that could be used for modelling. The pre-processing stage involved activities such as: examining field attributes, encoding the data into the correct format (suitable to be used as input in the models), dimensionality reduction, feature scaling, as well as data sampling. While the data pre-processing stage traditionally (Chapman et al, 1999) takes place after the initial data exploration phase, as a group, we found it necessary to conduct some pre-processing *alongside* the data exploration. The choice to conduct pre-processing in tandem with the data exploration was made to ensure that the data would not only be in the right format to be used as inputs in the models but also in the right format to produce the preliminary visualisations we wished to create. A breakdown of the pre-processing activities and why we did them can be found below:

Examining Field Attributes

The first step in pre-processing was to examine the data type of each column. Assessing the data type allowed us to gain a better understanding of the data and determined what steps should be taken to encode that column.

Firstly, it was necessary to know if the data in the column was symbolic or numeric. If the column has a numeric data type, what type of numbers did it hold, ordinal (continuous) or discrete numbers? This is important information. Knowing that a field is numeric ordinal, means the data in that column can be sorted into appropriately sized bins. In turn, this binning of data makes it easier to interpret the information during EDA, modelling, and results.

Equally, fields that are symbolic or numeric discrete are categorical variables that require additional encoding (such as one-hot encoding or factor encoding) to be easily used as inputs when it comes to preliminary visualisations in EDA or as inputs in the models. Understanding which variables are categorical is critical, as it allows us to analyse the data set by categories of people within the organisation.

To examine the field attributes, we used the following R functions (courtesy of Prof. Nick F Ryman-Tubb and provided in the COMM053: Practical Business Analytics laboratory R scripts):

> **NPREPROCESSING_initialFieldType()** to determine whether a column was numeric or symbolic, this makes use of the R function **is.numeric()**.
>
> **NPREPROCESSING_discreteNumeric()** to determine if a numeric column was ordinal or discrete. This sorts the numeric data into bins to create a histogram and takes as input a user-specified cut-off value. The cut-off value is the number of empty bins in the histogram to classify the numeric data as discrete.

A summary of each field and its data type can be found in Appendix B.

Converting to Factor Data Structures

Part of the data exploration involved creating preliminary visualisations of the data to reveal any interesting trends, and underlying patterns, and to investigate the relationships between various columns. This Exploratory Data Analysis was conducted with the ggplot2 library due to its ability to produce a variety of plots, including density plots, box & whiskers plots, histograms, and bar graphs among others. However, instances, where plots should be coloured by categories to display the split between classes, proved difficult to plot with ggplot2.

This problem arose as the ggplot2 library expects categorical variables to be used as inputs when conducting these operations. This condition is easily met when the input is “Male” or “Female” as this input is a symbolic data type. However, discrete numeric inputs (classes “1”, “2”, “3”, “4” or one-hot-encoded {1,0,0}, {0,1,0}, {0,0,1} values) are not accepted once a plot is split into categorical variables. This is because plotting libraries do not recognise numeric data types as categorical variables, as numeric data types can be ordinal (continuous) or discrete (categorical).

The solution is to transform the data inputs into factors. A factor is a data structure that can store integers, strings, and vectors and will treat these as categories. As such, a decision was made to convert necessary fields (any discrete numeric columns as well as one-hot encoded columns) to factors before producing graphs, to ensure the data was in the correct format to be examined. Converting columns to factors was done using either the **as.factor()** function or the **factor()** function in R, the only difference being that the **factor()** function allows for the category “levels” and labels to be specified in the input.

Splitting Dataset to Make It Manageable

With consideration of the existing dataset, dividing the present features into numerous subsets and modules was undertaken. The present dataset (*IBM HR Analytics Employee Attrition & Performance*, 2016) consists of a total of 35 columns. This can be divided into modular categories for management. However, following a team decision, it was decided that it would be in the best interest to retain the data format in case any relationship overlaps. If the dataset was to provide/present multiple data sheets/files, we then believe that the division of the data would be most appropriate, if this case was applicable. However, post-data division via a categorical approach (as shown in Figure 1), we retained the data format. Nonetheless, data preparation has been applied in various forms, including feature removal and verification of missing data.

Outlier Detection and Replacement

Outliers are defined as any value that is a significant distance away from the majority of the data. These points normally represent anomalies in the data (which may have occurred in the data collection stage) and can introduce a skew in the data towards extreme values. As such, outliers are not representative of the vast majority of the data. If used as inputs in a model, outliers would generate, at best, an unsuccessful model and at worst, a model that worked and output false conclusions. Hence, it was important to identify any outliers and to treat them appropriately. This is to avoid them negatively impacting the data.

Firstly, we examined the summary data of any columns that were deemed interesting. Further investigating those columns relating to the KPIs (Key Performance Indicators). The summary data contained information relating to the Inter Quartile Range (IQR) which gave us an insight into the spread of the data. To gain a better understanding of the data, we created box & whiskers plots to show the distribution of values and identify any possible outliers.

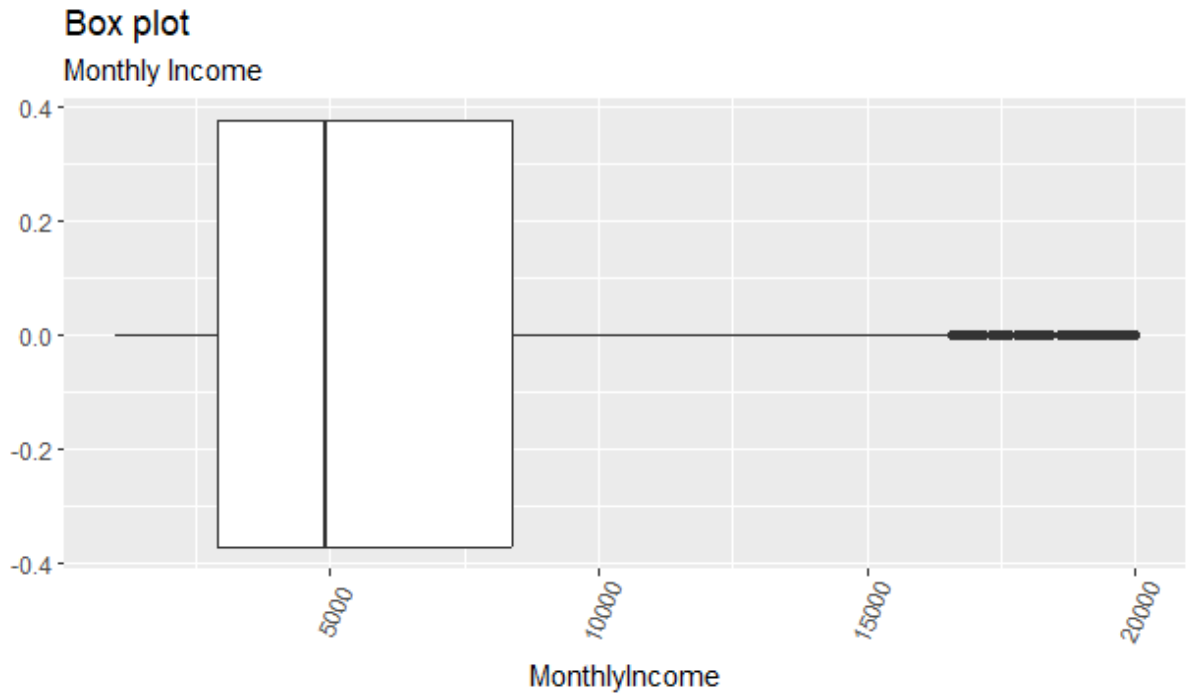


Figure 2: Box & Whiskers plot of Monthly Income

Using the IQR method, outliers are defined as:

Any value that is a distance of ($1.5 \times IQR$**) below Q1 or above Q3**.

Box & whisker plots are effective at revealing outliers because the "whiskers" on the plot are distance of $1.5 \times IQR$ away from Q1 and Q3. In the case of Monthly Income, it is visible in Figure 2 that this excludes a significant number of values.

Following this outlier method would imply that there are 114 outlier values out of a total of 1470 values – the equivalent of 7.8% of the total data values. In other words, too many outliers for the size of the data set. However, when examining the distribution of Monthly Income values, it became apparent that there is a majority of people are low-medium earners but there is also a non-insignificant group of high earners. Whilst the high-earning group is small and out-earns the majority of the employees by ~4x their salary, these are not outliers, but an accurate reflection of the income distribution within the UK (Office for National Statistics, 2020).

To include the high-earning group and to reduce the percentage of outliers in the data set to a more reasonable amount, we decided to instead consider only extreme IQR outliers (NIST, 2022), which are defined as:

Any value that is ($3 \times IQR$**) below Q1 or above Q3.**

This approach reduced the number of columns with outliers but did not remove them completely.

Treatment of outliers is often to remove them from the data set entirely or to replace them with an appropriate measure such as the mean, median or mode. As a group, a decision had to be made to *not* replace outliers with the mean. It should be indicated that the mean would have been influenced by the presence of outliers. With consideration of this information, it was decided outliers would be replaced with the Winsorized mean (Investopedia, 2022). This was implemented by replacing extreme outliers with the closest value that is not an outlier. This type of mean was chosen to best represent the original data, without impacting the inputs of the model. In practice, this meant:

- Extreme outliers above Q3 were replaced with the Q3 value.
- Extreme outliers below Q1 were replaced with the Q1 value.

To implement the outlier detection and replacement, we created three functions in R. A function to detect mild outliers, a function to detect extreme outliers, and a function to replace any extreme outliers. Our functions utilised the **quantile()** R function to calculate initial IQR statistics, as well as **if** statements to specify the treatment of outliers.

In practice, we only found extreme outliers in the YearsSinceLastPromotion column. Here, there were 32 extreme outlier values. Upon further inspection, values of YearsSinceLastPromotion ranged from 0 to 15 years, with the third quartile being (75% of the values falling under) 3 years. Using the extreme outlier method, an extreme outlier would be any value above 9 years since their last promotion. The values of the 32 outliers are shown in the output below:

```
In [17]: summary(df$YearsSinceLastPromotion)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   1.000   2.188   3.000  15.000

In [19]: extreme_yearsLastPromotion <- 3 + (3 * IQR(df$YearsSinceLastPromotion) )

In [28]: outlier_df <- filter(df,YearsSinceLastPromotion>extreme_yearsLastPromotion)

In [29]: outlier_df$YearsSinceLastPromotion

 15· 13· 15· 15· 15· 15· 13· 15· 13· 14· 13· 14· 15· 15· 13· 13· 15· 13· 15· 14· 15· 15· 15· 14· 13· 13·
 14· 14· 14· 13· 14· 14
```

Figure 3: Extreme Outliers in YearsSinceLastPromotion

In the context of the data, the extreme outlier values represent real people and the number of years since their last promotion - a value calculated by the HR department. As such, a value of ‘YearsSinceLastPromotion’ of 13, 14, or 15 years would represent employees that have been with the company for long periods, despite a lack of retention techniques by the company (no promotions or additional perks offered as incentives to stay). Considering that 15 years of experience within a role could likely result in job offers at other companies, and presumably the offer of a higher salary to switch companies, these “extreme outliers” are the company’s most loyal employees. Therefore, these values are of particular importance in the context of the project and will not be removed in this case.

Check Missing Data

Part of the process of cleaning a dataset and assessing its data quality involves checking for missing values. It is important to ensure that data is complete or in the case that it isn't complete, acknowledge this and determine how this might affect both the model and the findings.

We checked for missing values in the data by using the R function **is.na()** which determines whether values are Not Available. Applying **is.na()** to the data set will return a table with the data column headings and substitute the value of each element to be either: TRUE (the value is not available) or FALSE (the value is available). To ensure that the entire data set is FALSE, we applied the R function **any()** to determine if any element in the data set returned a TRUE Not Available value.

Figure 4: Checks implemented to find missing values

We found no missing values, as can be seen in Figure 2. This required no further action. However, it is important to note that whilst we performed little data cleaning to the data set, we did run computational checks to ensure that the data was indeed clean. Moreover, the little data cleaning required is a result of the hours spent finding an appropriate dataset to use in this project.

Dropping Irrelevant Features

Following examination of data through the use of missing values and EDA, it was concluded some features were deemed weak concerning interactions. Whilst others were deemed to be repetitive/non-relevant to the intended target. With references to Appendix E and the EDA heatmap, the data illustrates relationships and patterns amongst fields within the selected dataset. With collaboration to checking missing values, EDA, and analytics of data, we proceeded with the removal of the following features within the dataset:

- EmployeeNumber
- EmployeeCount
- Over18
- StandardHours
- Daily Rate
- Monthly Rate

To determine which features were to be dropped, a verification procedure was run on each of the discussed fields, accordingly, including checking if values were repeated or presented variations. By using the following, as demonstrated in the figure below, analytical skills were then applied to identify any varying patterns we present.

```
summary(df)
```

```
     Age         Attrition         BusinessTravel      DailyRate
Min.   :18.00   Length:1470        Length:1470        Min.   : 102.0
1st Qu.:30.00   Class :character   Class :character   1st Qu.: 465.0
Median :36.00   Mode  :character   Mode  :character   Median : 802.0
Mean   :36.92                                         Mean   : 802.5
3rd Qu.:43.00                                         3rd Qu.:1157.0
Max.   :60.00                                         Max.   :1499.0
 Department        DistanceFromHome   Education     EducationField
Length:1470        Min.   : 1.000   Min.   :1.000   Length:1470
Class :character   1st Qu.: 2.000   1st Qu.:2.000   Class :character
Mode  :character   Median : 7.000   Median :3.000   Mode  :character
                   Mean   : 9.193   Mean   :2.913
                   3rd Qu.:14.000   3rd Qu.:4.000
                   Max.   :29.000   Max.   :5.000
EmployeeCount EmployeeNumber   EnvironmentSatisfaction    Gender
Min.   :1     Min.   :   1.0   Min.   :1.000           Length:1470
1st Qu.:1     1st Qu.: 491.2   1st Qu.:2.000           Class :character
Median :1     Median :1020.5   Median :3.000           Mode  :character
Mean   :1     Mean   :1024.9   Mean   :2.722
3rd Qu.:1     3rd Qu.:1555.8   3rd Qu.:4.000
Max.   :1     Max.   :2068.0   Max.   :4.000
  HourlyRate     JobInvolvement    JobLevel       JobRole
Min.   : 30.00   Min.   :1.00   Min.   :1.000   Length:1470
1st Qu.: 48.00   1st Qu.:2.00   1st Qu.:1.000   Class :character
Median : 66.00   Median :3.00   Median :2.000   Mode  :character
Mean   : 65.89   Mean   :2.73   Mean   :2.064
3rd Qu.: 83.75   3rd Qu.:3.00   3rd Qu.:3.000
Max.   :100.00   Max.   :4.00   Max.   :5.000
```

Figure 5: Summary of Dataframe Pre-Transformation

Based on the data presented following the summary command, as shown in Figure 5. The data summary suggests that several fields have data with no variations. For instance, EmployeeCount, based on the data supplied and collected from summary(df) illustrates that both min, max and mean are at value 1. As a result of this pattern, the feature was dropped. Similarly, the field EmployeeNumber simply represented the value assigned to the individual upon joining the organisation. Despite its benefits when querying data, or querying patterns based on employees, the data did not present any aspects of interest, and thus was removed from the structure. Finally, patterns highlighted by StandardHours illustrate a singular variant of 80 hours for all rows recorded. Similar to EmployeeCount, due to the regular presence of the said pattern, it was appropriate to drop the field to proceed with modelling.

Randomisation

Additionally, to support the process of pre-processing and EDA, a randomisation function was applied to the dataset. Applying the randomisation command varied the data order, accordingly, ensuring no bias. data was not present among the mentioned. To achieve this, the following code was utilised.

```
df<-df[order(runif(nrow(df))),]
```

Figure 6: Randomisation of Data, Pre-Transformation

TrainTest Splits

Our IBM data set consists of 1470 rows of data. To successfully apply a model that can predict attrition, we must train the model on previously seen (training) data and test it on unseen (testing) data. In this project, the original dataset was split into train/test subsets with a ratio of 70/30 (Nguyen et al, 2021) for training and testing, respectively. Note that all splitting was done following the randomisation of the data. A larger training set was chosen to increase the probability that the models would be trained with at least one of every variable. However, careful consideration was taken when deciding the exact split of the data. A training set that is too large would perform extremely well on the data and its noise but would fail to capture

any underlying patterns in the data, resulting in “overfitting” and a model that would not generalise well on new unseen data. Conversely, a training set that is too small might not capture the patterns in the data which would result in “underfitting”.

Oftentimes, data might be split into three subsets; training, testing, and validation – a third subset to evaluate parameters during training. Due to the limited size of the original data set, we opted to *not* split the data set to include a validation subset. However, evaluating the performance of the models remained a priority which is why we have implemented k-fold cross-validation on the 70% of the data designated for training.

K fold cross-validation is a process by which a data sample is split into a user-specified number of “K” folds. In the project, we opted to split data into ten folds. Once the data is split into ten folds, the first section will be taken as the test data, and the remaining nine sections will be used as training data. The model will run, and evaluation statistics will be obtained. Next, the second section will be picked as the test data, and the remaining nine sections will act as the training data. The model will run and obtain evaluation statistics. This process will be repeated “K” several times until each fold has the test data and “K” evaluation statistics have been obtained. At the end of this process, evaluation statistics can be averaged to get a robust measure of model performance.

The train test split was implemented through the use of the **partition()** function in R. The K fold cross-validation resampling was implemented by the **trainControl()** function within the caret library – where the “number” input is the number of folds, in case 10.

Feature Scaling

Feature scaling is a method utilised to normalise independent variables or features present within the data. By doing so, it normalises the data to accordingly fit the aforementioned into the appropriate date range, namely adjusting this into the correct range, dependent on the technique utilised, in modelling processes (*Machine Learning: When to Perform a Feature Scaling?* n.d.). For this project, feature scaling has been utilised in KNN modelling. We proceeded with scaling to ensure the data was appropriately tailored to maximise accuracy with the data. Moreover, scaling would be deemed appropriate where distances are computed as part of a modelling process.

One-Hot Encoding

One-hot encoding is a method applied to categorical fields where unordered classes are transformed into a series of 0s and faults will uniquely identify the class. This method expands the number of columns, creating a column for every categorical option. For example, the Gender column becomes gender.male and gender.female. Next, all the option columns will be allocated a 0 (if the category does not apply) or a 1 (if the category applies). Following the example, a female would be represented as a {0,1}. In this way, one-hot encoding was used to transform categorical string inputs into binary inputs for the models.

Both the K-Nearest Neighbours model and the XGBoost model required binary inputs to be able to run successfully. As a result, the dataset used as an input for K-Nearest Neighbours was one-hot encoded using the **dummyVars()** function in the caret library.

New Features

New features were developed to reduce the dimensionality of the data set and to improve its efficiency. To implement this, the dataset was reviewed, and all the fields within pre-set categories were summarised in Appendix D. Next, any similar fields were identified and discussed to justify the merger of components. After deciding if creating a new component made sense in the context of the data, the new features were encoded.

The 'ManagerYearsLabel' column was created to examine whether an employee's manager had any effect on their satisfaction at work. This feature was created by categorising the 'YearsWithCurrManager' column into 3 categories: 'New Hire', 'Experienced Hire' or 'Veteran Hire'.

The 'WorkExperienceScore' column was created to reduce dimensionality and to replace 'Age' as a measure of professional experience. This feature was calculated as,

$$WorkExperienceScore = \frac{TotalWorkingYears}{Age}$$

where 'TotalWorkingYears' represents the years worked in a particular professional field.

The 'AttritionScore' column was created to indicate employees' attrition throughout their professional career, i.e., the number of companies an employee has worked for (and left) throughout their career. This feature was calculated as

$$AttritonScore = \frac{NumCompaniesWorked}{TotalWorkingYears' + 1}$$

Note the +1 avoids division by zero in the case that an employee has not held any roles in their professional careers (e.g., graduates or those people who might have had a recent career change)

Once new features were created and added to the data set, the 'YearsWithCurrManager', 'Age', and 'TotalWorkingYears' columns were removed.

Synthetic Minority Oversampling Technique (SMOTE)

When working with datasets that simulate real-life occurrences such as tax fraud, loan defaulting, and attrition. The target label for these cases tends to be highly imbalanced. Imbalanced datasets cause classification problems in the sense that there are too few instances of the minority class for the model to effectively determine the decision boundary so it can make accurate predictions. To counter this issue, we have implemented the Synthetic Minority Oversampling Technique (SMOTE)**.** SMOTE counters the imbalance problem during sampling by first selecting a minority class at random and then finding its k nearest minority class neighbour and then connecting them to form a line segment which becomes an instance of the sample.

```
[14]: library(performanceEstimation)
      train_smote <- smote(Attrition ~ . , train, perc.over = 2, k = 3, perc.under = 1)
```

Figure 7: Implementation of SMOTE

Smote takes 5 arguments, the dependent and independent columns. The dataset columns are in **perc.over**; determines the number of observations that would be generated relative to the number of observations we have. i.e., if we have 1000 observations, we will generate 2% of every 1000 new observations as new samples, therefore, having (1000+1000)* 0.02= 40 observations. **k** determines the k-nearest neighbour value and **perc.under** denotes how many extra cases of the majority class are generated from the minority class i.e 1040*0.01= 10.4. We chose these values for perc.under and perc.over because they provided the best outcome for the model (Determined through numerous Iterations). The results section explains the outcome of the implementation of smote in the data set.

Exploratory Data Analysis

EDA, Exploratory Data Analysis is a process undertaken to comprehend and understand patterns present within the existing data sets. By applying such a procedure, numerous advantages are identified within the project. These include identification of errors within the dataset, whether it may be of the incorrect format, increased understanding of the data set selected, comprehension of outliers, and comprehension of data set variables/attributes as well as any potential relationship among them (Telang, 2021). Moreover, by applying EDA, crucial information can be discovered and identified from the dataset, before any adjustments are applied, i.e., removal of features, scaling etc.

To start the process, we reviewed all the fields attached to the dataset and used the KPIs in Figure 1 to categorise the fields accordingly. By doing so, we were able to comprehend what fields we could compare against one another, including the target variable, namely attrition. As shown in Appendix E, a visual workflow was developed to categorise the said fields into a category. Following this procedure, we commenced reviewing relationships, from one field to another through the usage of visualisations, as demonstrated in the Figure below.

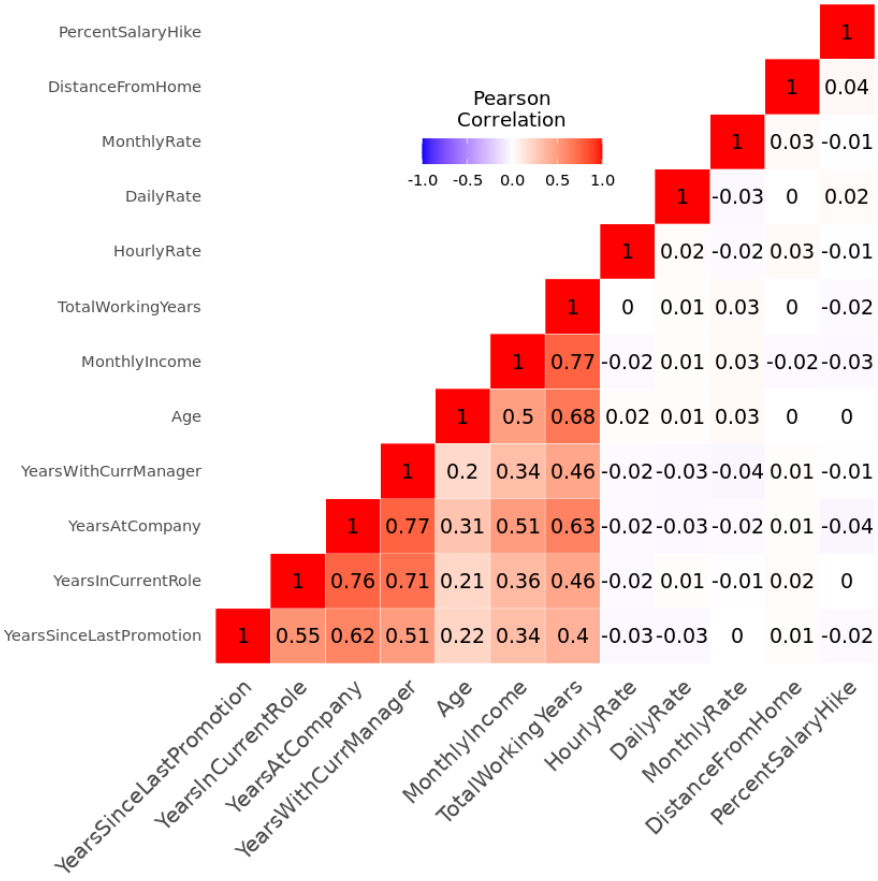


Figure 8: Investigation questions and hypotheses

As demonstrated by Figure 8, a correlation matrix was presented displaying the top 12 fields based on the correlation between one against the other. Based on the above, the results present a higher correlation between Managers v Years at Company, Manager v Years in Current Role and Age v Total Working Years.

The correlation present between the aforementioned fields is an expected pattern as Management in any firm would play a large role within the business and would correlate with paired fields based on the data present and logical components, such as a manager having greater experience within the said field, thus their position within the company.

We also noticed a relatively higher positive correlation between Monthly Income v Total Working Years, which could be indicating that the more years an employee spends working, the higher their salary increases over time.

Distance from Home to Office

In a similar manner, we monitored and attempted to identify the patterns between one's travel, against attrition, as attrition is the target variable against the identified hypothesis. We developed this intention, to comprehend if a pattern or relationship was present. To present this, we commenced with plotting the aforementioned via a bar graph and using both parameters of DistanceFromHome, and Attrition, presented below:

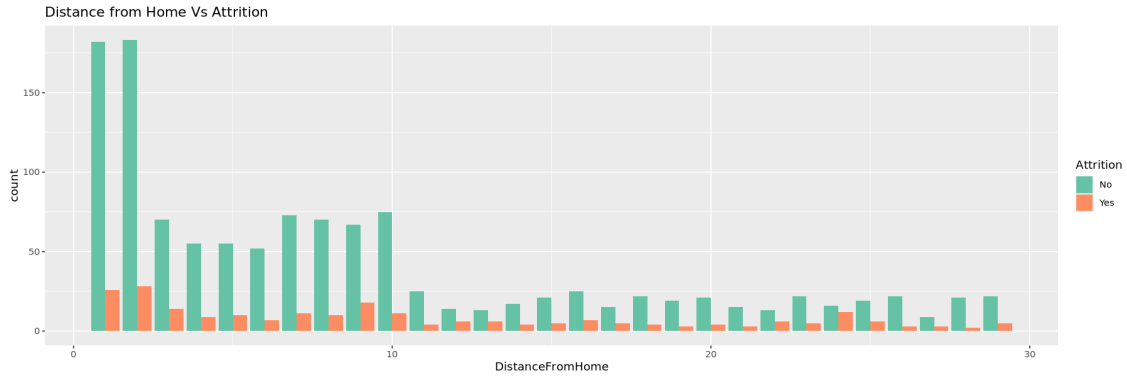


Figure 9: Distance from Home vs Attrition Plot

Based on the patterns present, this illustrates the attrition rate present among those who travel, based on the distance. The patterns prove that distance does not impact attrition as strongly as other parameters. However, based on the data, it can be seen that for those with a distance of 24, the balance between attrition, yes and no, is approximately a 60:40 balance.

Salary

There are many columns relating to salary. More specifically, these are: 'DailyRate', 'HourlyRate', and 'MonthlyIncome'. Note that 'Standard Hours', 'PercentSalaryHike', and 'MonthlyRate' also relate to salary however, these columns were already deemed redundant as a part of the pre-processing stage.

To investigate the relationship between attrition and the salary-related columns, visualisations of the data were created to view which column best describes salary. Figure 12 shows the density distribution of 'DailyRate' and 'HourlyRate' and 'MonthlyIncome'. These plots reveal little variation in 'DailyRate' values and no clear pattern upon first inspection. Similarly, the 'HourlyRate' distribution of values shows fairly constant values within the data set, and no significant patterns that require further investigation. On the other hand, 'MonthlyIncome' has

the most variation in its data, meaning it can best represent the variation in employee salary going forward. As a result, the 'DailyRate' and 'HourlyRate' columns have been made redundant.

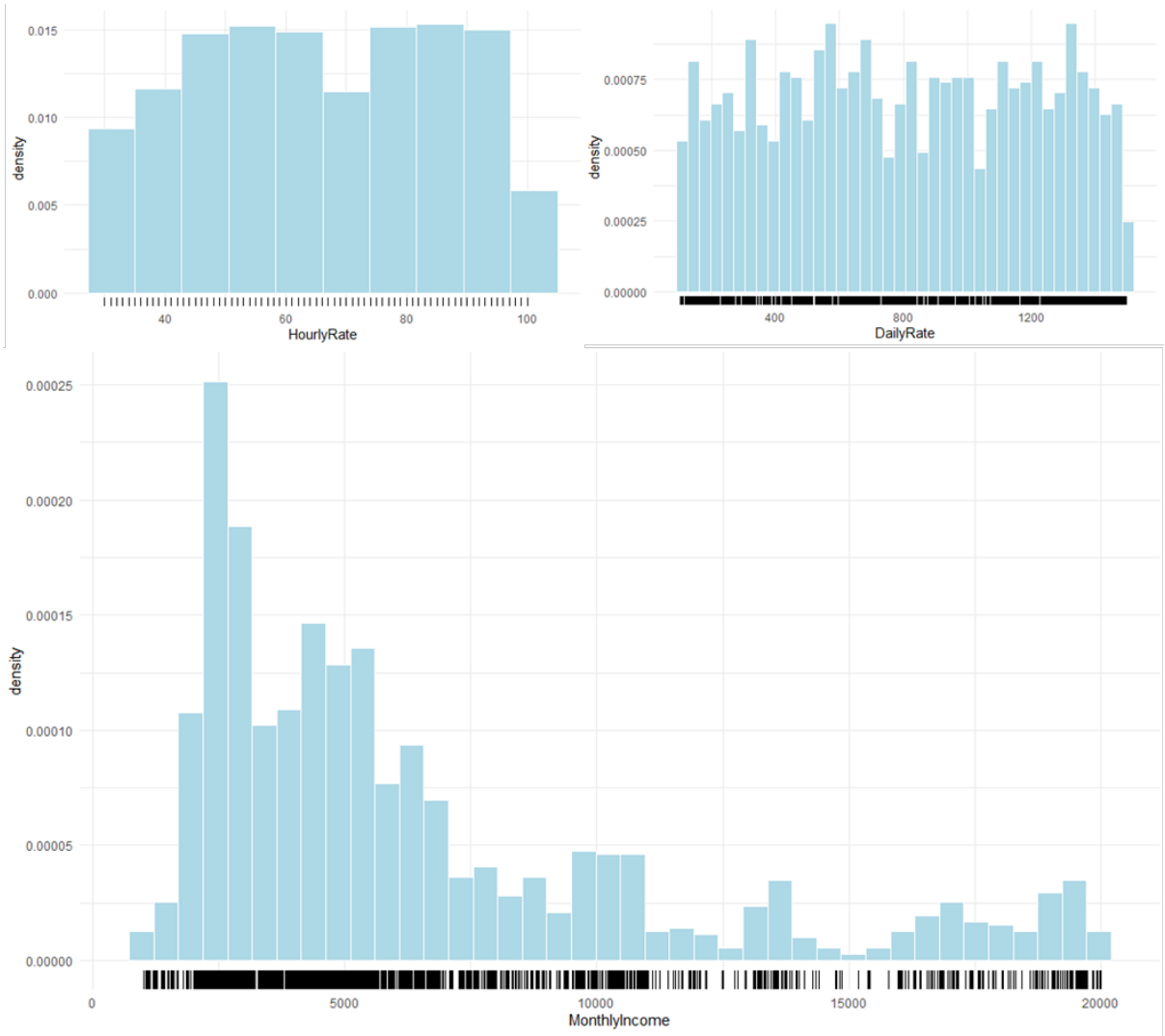


Figure 10: Density Distributions of 'DailyRate', 'HourlyRate', 'MonthlyIncome'

To present the division and relationship between monthly income and attrition, we felt that a histogram would be deemed the most appropriate. This would allow for stakeholders or external partners to understand, analyse, and learn the patterns attached between these fields during the process of EDA before removing fields deemed unnecessary. Figure 10 shows a histogram of salary and attrition.

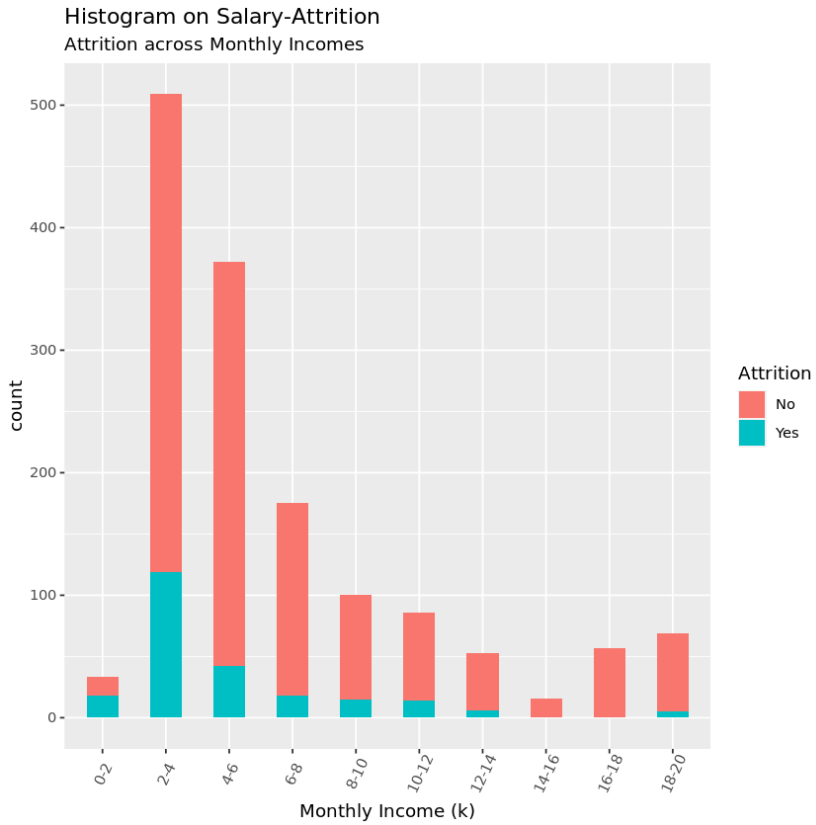


Figure 11: Histogram on Salary-Attrition

There is a definite trend correlating monthly income to attrition, clearly showing that as income increases, the number of employees leaving the company decreases. At the lowest monthly income, employees were equally as likely as to stay in the company as they were to leave. In the 2-4k bracket, roughly a quarter of employees left the company. As salary increased from 2-14k, attrition decreased steadily, eventually reaching zero. Though this behaviour might be partly explained by the shape of the income distribution, which shows that there are significantly fewer people earning medium-high incomes (the distribution is clustered around low-medium earners and very high earners). Since attrition makes up 16% of the data set, it is reasonable to expect that when the number of employees is low, no attrition would take place.

To further support the above investigation, a box plot was utilised. By using a box plot of this activity and task, further information was uncovered effectivity. A box plot allows one to comprehend the distribution between the present values within the dataset or data frame, followed by visuals of statistical values, including lower quartile, median, upper quartile, and maximum score. Additionally, by using a box plot, developers, stakeholders, and partners can learn and identify whether the data is said to share a normal distribution, positive or negative skew (McLeod, 2019). This is to further elaborate on the above comparison between monthly income and attrition, as presented below:

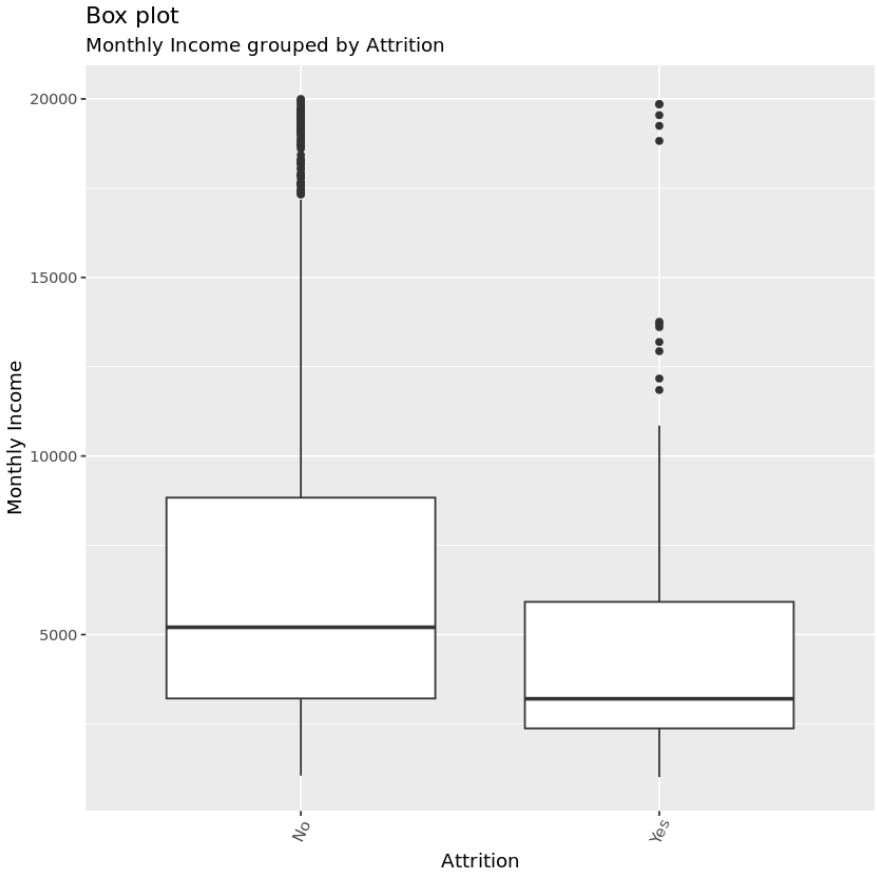


Figure 12: Box Plot for Monthly Income, Grouped by Attrition

Based on the visual presented above, including comparison and consideration to the initial visualisation presented in Figure 12, the data suggests that those who have a selected attrition value of yes are within a medium of under 5000, with an upper quartile of approximately 6000. Similarly, the data for attrition value of no suggests that the median is in the higher range of 5000, with a greater upper quartile of under 10000. Additionally, when analysing both statistics, the data suggests a positive skew of data. By having a positive skew of data, as illustrated in the above figure, the data suggests that the higher frequencies are present and that the distribution is greater towards the right-hand side of data (Sharma, 2020), equally creating a greater mode, median and mean effectively.

Employee Satisfaction

There are many columns relating to satisfaction. Namely: ‘EnvironmentSatisfaction’, ‘JobSatisfaction’, ‘RelationshipSatisfaction’, and ‘WorkLifeBalance’.

The columns of ‘EnvironmentSatisfaction’, ‘JobSatisfaction’ and ‘RelationshipSatisfaction’ are ratings of 1-4 where: 1 'Low', 2 'Medium', 3 'High', 4 'Very High'. Similarly, the ‘WorkLifeBalance’ column contains ratings from 1-4 where: 1 'Bad', 2 'Good', 3 'Better', and 4 'Best'.

To best display the relationship between the satisfaction columns and attrition, a bar graph of cases where an employee leaf (attrition = Yes) was created, for each satisfaction column.

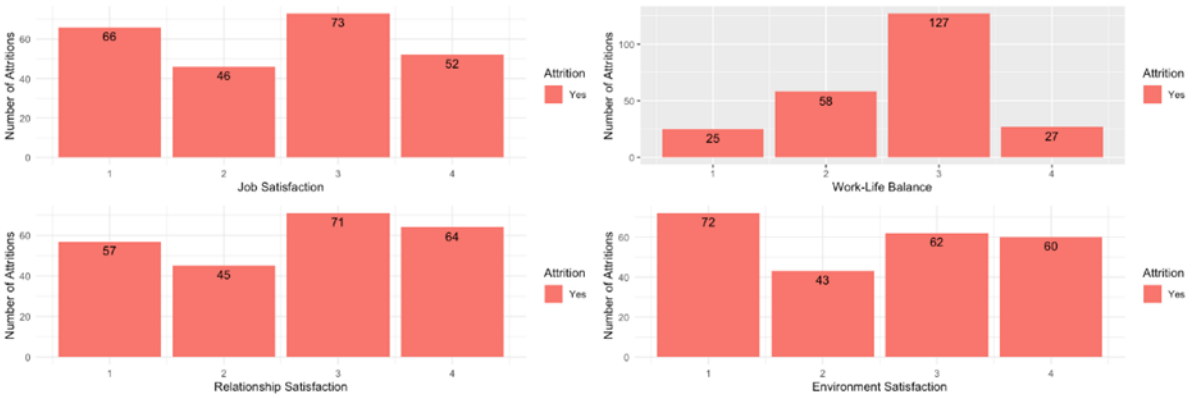


Figure 13: Bar Graphs of Attrition = Yes against 'JobSatisfaction' (top left), 'WorkLifeBalance' (top right), 'RelationshipSatisfaction' (bottom left), 'EnvironmentSatisfaction' (bottom right)

As seen in Figure 13, 'JobSatisfaction' and 'RelationshipSatisfaction' have a very similar relationship with attrition. This implies it is not necessary to keep both the 'JobSatisfaction' and 'RelationshipSatisfaction' columns in the data set to predict attrition. It is recommended that one of these columns can be made redundant to reduce the dimensionality of the data set.

Furthermore, while 'EnvironmentSatisfaction' appears to have a different relationship to attrition, the number of cases where employees leave the company (attrition = Yes) appears to be fairly constant at 60 cases, regardless of whether the environment satisfaction was rated as 1, 2, 3, 4. The fact that there is little variation in attrition over the entire range of values of 'EnvironmentSatisfaction' suggests that the 'EnvironmentSatisfaction' column will not be a good predictor of attrition and that this is not a significant relationship.

Conversely, there is a clear relationship between the number of employees who decide to leave and their 'WorkLifeBalance' satisfaction scores. When there is little work-life balance attrition is low. When there is a better work-life balance, attrition is at its highest. This may suggest that those employees who are busier with work and are spending more time in the workplace, perhaps do not have the time or desire to look for another job. The 'WorkLifeBalance' attrition plot also implies that employees are most likely to leave when the work-life balance is better. If an employee's work-life balance is better, this might suggest they are at work from 9-5 only i.e., they are spending little extra time at the office and leaving promptly - which might be a symptom of job dissatisfaction. Additionally, those employees with the best work-life balance are very unlikely to leave, presumably because they are most content personally and professionally.

Experience Level

There are many columns relating to job experience, such as: 'TotalWorkingYears', 'Department', and 'YearsUnderCurrentManager'. A range of plots was produced to investigate which job experience column has the greatest effect on attrition if any.

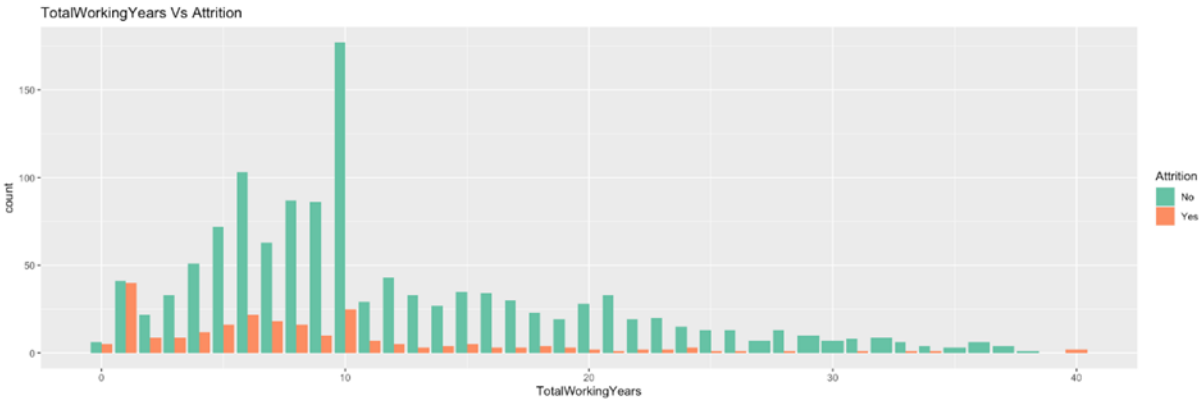


Figure 14: Bar Chart of 'Attrition' over 'TotalWorkingYears'

Figure 14 illustrates the attrition for 0-40 total working years. The bar plot produced shows a clear trend where employees are most likely to leave the company within their first year of work, with a roughly equal number of employees leaving and staying with the company. A higher proportion of employees appear to leave within the first 10 years of employment. After ten years of working with the company attrition levels dropped significantly to rates close to zero. These findings indicate that the 'TotalWorkingYears' column might have a direct impact on attrition, therefore it should be investigated further.

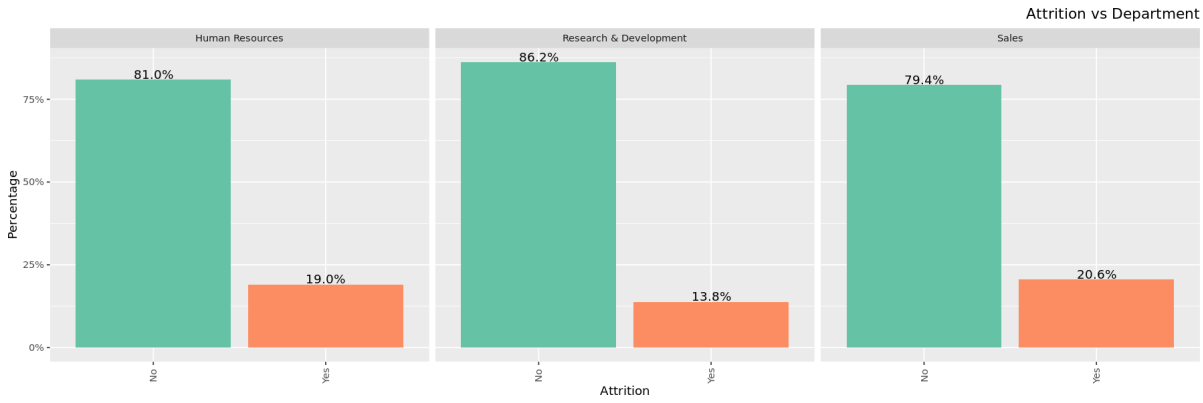


Figure 15: Bar Chart of 'Attrition' over Departments

The percentage of attrition by the department is displayed in Figure 15. This plot shows that all departments have similar attrition rates with the sales department having the highest rate of 1 out of every 5 people leaving. Having only a slight variation from department to department implies that the 'Department' column does not heavily impact attrition.

By using such visualisations to analyse each of our KPI questions of interest, we as a project team were able to successfully identify any non-declared patterns present within the dataset. As a result of this, and in collaboration with the data present from the visualisations, namely the correlation plot, the data suggests which features are deemed to be most valuable.

To conclude, after pre-processing and EDA it was discovered that the IBM HR dataset was relatively small, consisting of just 1470 rows. Additionally, the majority consists of discrete data and binary classifications. There is no missing data or duplicate data to consider within the dataset however, there were some outliers (none that would drastically affect the overall

results), which were replaced with winsorized mean. It should be mentioned that the target label (attrition) is imbalanced, 84/16 with an advantage to no.

Modelling

Plans were slightly deviated after analysing the dataset. We found the target label Attrition was highly imbalanced, with only a limited sample size of people leaving the organisation. To counter this problem, we trained the model using XGBoost, a model which specialises in dealing with imbalanced datasets. We also trained the model using oversampling techniques (smote) to further counter the imbalance problem.

After extensive research on what type of models would work best based on the points from the EDA, it was concluded that the following three models were most appropriate for the dataset and business needs:

K Nearest Neighbours

K Nearest Neighbours (KNN) is a non-parametric supervised machine learning technique used to solve both regression and classification tasks. KNN algorithms work based on the assumption that data points closest to each other fall into the same class. Distance metrics such as Euclidean and Manhattan distance can be used to determine the distance between data points. We use a k value to determine how many data points are needed to classify a new data value. Depending on the value of k, the model then makes a classification. A drawback of the k nearest neighbours' algorithm is its computational inefficiency, distances need to be computed for all training points. As there is also no clear means to know what type of distance metric will increase model performance, inefficiency further increases if the model is trained using different distance metrics. This model was ultimately a good choice for the project as the data set is relatively small and the KNN algorithm won't suffer too much when it comes to efficiency.

K Nearest Neighbour Implementation

When implementing KNN, it was decided not to run a base model as the only hyper we used to be the k value. Before training the model, we did one hot encoding on all values as the k nearest neighbour's algorithm works better with just numerical features. As KNN is a distance-based algorithm, it is sensitive to data points of different scales, so we normalise the data before training the model. We then trained the model using hyper tuning and cross-validation and then ran it on the base data and oversampled data.

Hyper-Tuned Cross-Validated K Nearest Neighbours

To find the optimal value for k, we ran a grid search containing k values from 1 to 20. We then performed a 10-fold cross-validation, which was repeated 3 times for each fold. The following parameters were used for hyper-tuning and cross-validation.

Hyper-Tune parameters:

- K Value - Determines the number of data points needed in order to make a new classification.

Cross-validation parameters:

- Method - Determines cross-validation method to be used.

- Number – Determines the nth fold of cross-validation to be done.
- Repeat - Repeats cross-validation nth times for each fold.

Elbow Plot Using Accuracy on Base Data

From the plot, we see the model accuracy has a steep rise when trained with k values between 1 and 5. The accuracy score reaches its peak at a k value of 7 before slightly dropping off again and remaining at scores above 84 percent for k values between 7 and 20.

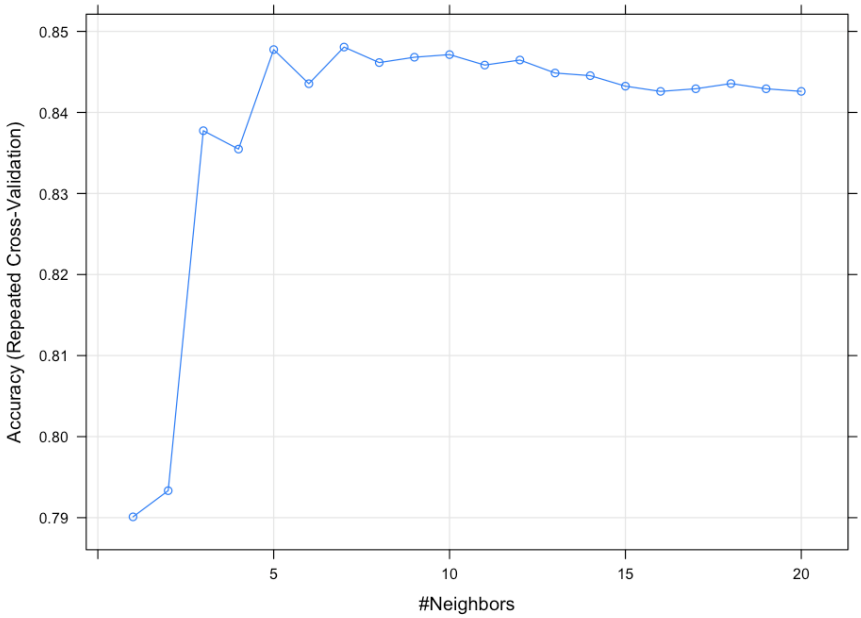


Figure 16: Elbow Plot Using Accuracy

Elbow Plot Using Accuracy on Oversampled Data

From the model we see the highest accuracy is observed right at the start when k is equal to the model accuracy. This then drops significantly across all other k values possibly from underfitting on the training set for the higher k values.

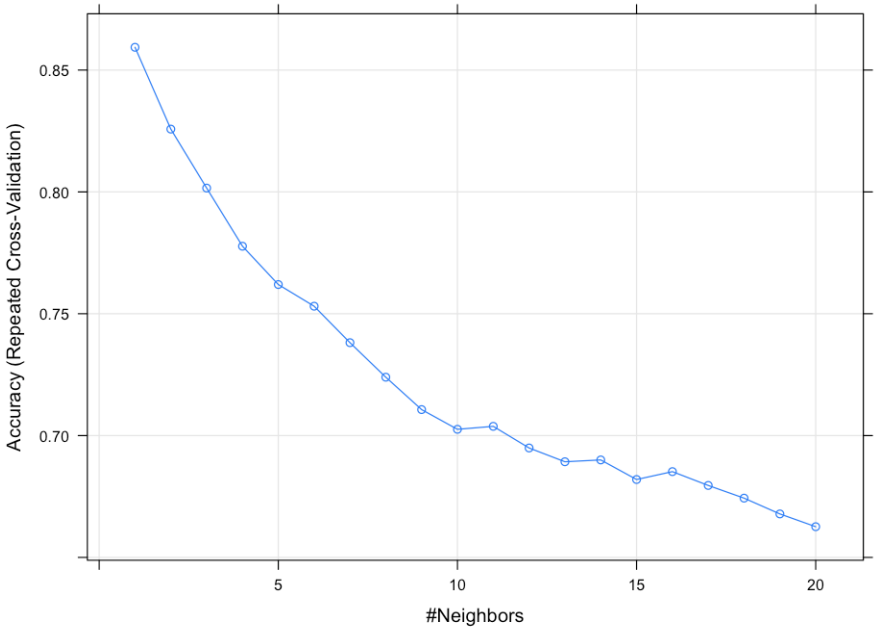


Figure 17:Plot of Neighbours vs Accuracy

Decision Tree

A decision tree is a tree-based machine-learning technique used to solve both classification and regression problems. Hence, known to be strong when working with discrete data and binary classifications, both of which are frequently seen within the IBM HR dataset. Although a downside of the decision tree could be its nature to produce complex and irrelevant data, it can be solved with the pruning procedure. After pruning the model can easily display the most important factors, even to non-technical stakeholders.

For this project, we will be using a classification tree as the target label is discrete. Decision trees follow a set of conditions asked to classify data according to that condition. Metrics such as Gini impurity are used to calculate impurity at each section of the tree to decide the order in which nodes appear in the tree. Every decision tree is made up of three node types:

- **Root Node** - Also known as the parent node, it is the topmost part of the decision tree. The root node is the feature which best splits the dataset when modelling and the lowest Gini impurity in the tree.
- **Decision Node** - Also known as the intermediate node, these nodes have branches coming and going out of them. The features of these nodes are evaluated and then branched out to either another decision node or a leaf node.
- **Leaf Node** - Also known as the terminal node, these are the endpoint nodes of any decision tree where predictions are made.

Decision Tree Implementation

When implementing the decision tree, hyper-tuning and cross-validation were utilised in hopes of improving the performance of the model. As decision trees are non-variant and aren't sensitive to data of different scales, it was decided *not* to do feature scaling or one hot encoding before training the model. This was advantageous as it saved time in data pre-processing.

The model was trained on three different decision trees: post-pruned, pre-prone and untouched decision trees, using the base dataset and the oversampled data set.

The first implementation of the decision tree was a base model with default parameters set and no cross-validation. The complexity parameter (cp) value, a metric used by decision trees to control the size and optimization of the tree, was set to 0 to maximise tree size for pruning.

When the tree is allowed to grow out fully, it ends up with a max depth of 9. Work experience score is the root node which then branches off on both sides to overtime decision nodes. The tree then makes further splits based on other decision nodes before ending up at the leaf nodes. In total, this tree had 27 leaf nodes, with most classifications being made in cases where attrition was no.

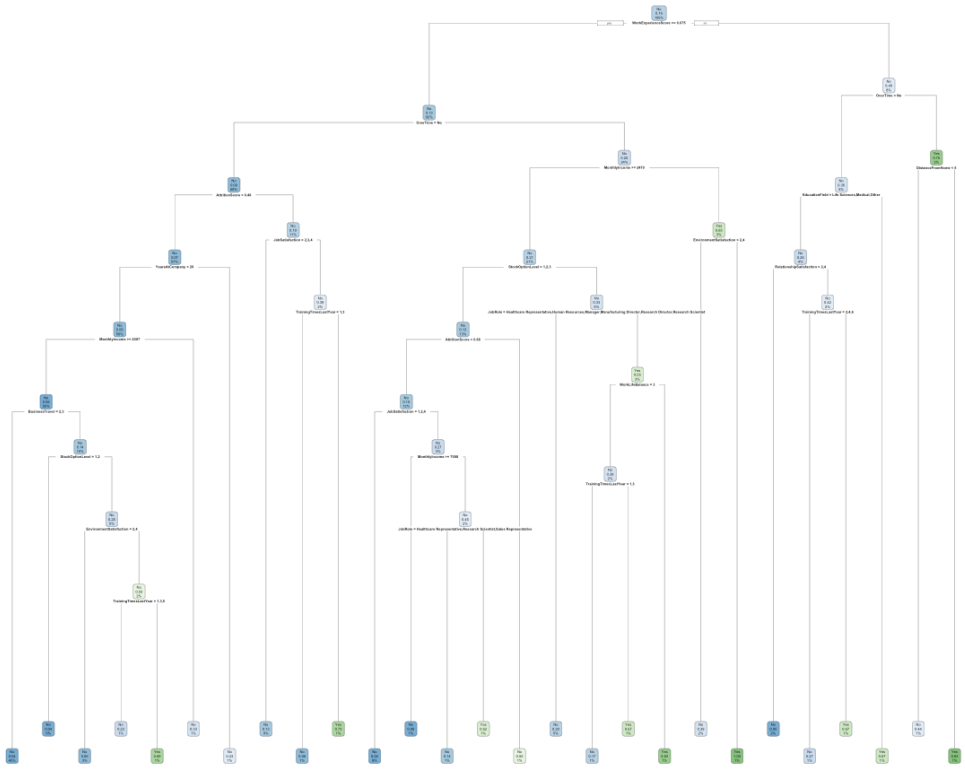


Figure 18: Base Decision Tree Plot

When the tree is allowed to grow out fully as seen in Figure 18, it ends up with a max depth of 9. The overtime node is the root node which then branches to the stock options level and job role decision nodes. The tree then makes further splits based on other decision nodes before

ending up at the leaf nodes. In total, this tree had 24 leaf nodes with an even representation of attrition=yes and attrition=no cases being classified.

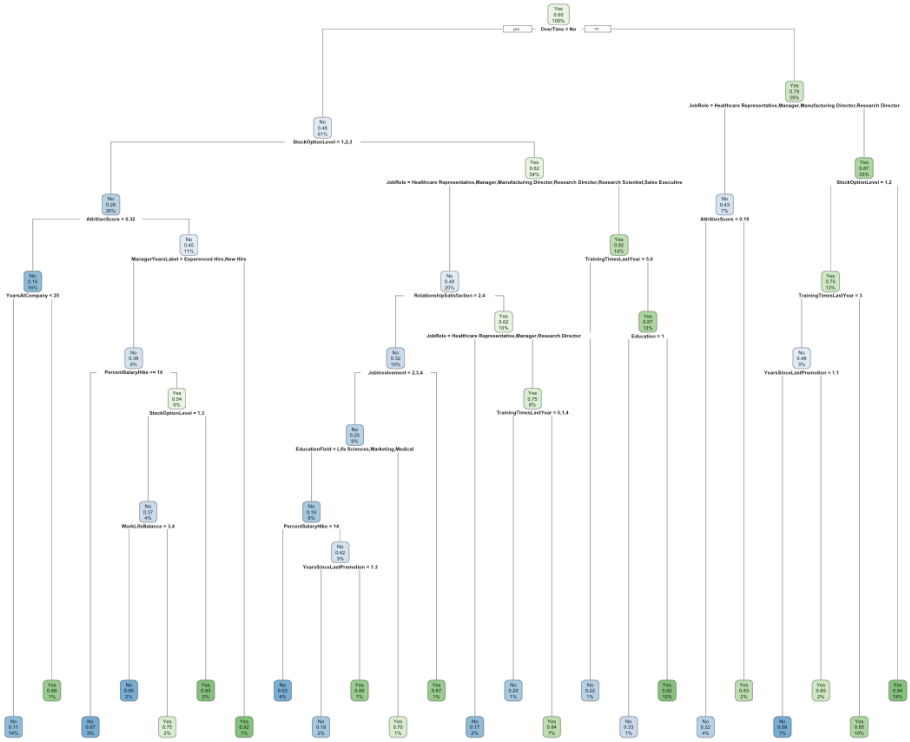


Figure 19: Oversampled Data Decision Tree Plot

1. **Pre-Pruned Decision Tree** - In the second implementation of the decision tree, we used pre-pruning in the hope of improving performance from the base tree model. Pre-pruning is a pruning technique which stops the growth of non-significant branches through methods such as hyper tuning and cross-validation. We performed 10-fold cross-validation when training the model. The hyper tune parameters we use when pre-pruning was:
    - **Min Split** - The minimum number of observations in the parent node which can be further split up.
    - **Complexity Parameter**(cp) - the metric used by decision trees to control the size and optimization of the tree.
    - **Min Bucket** - The minimum number of observations a parent node is allowed to have.
    - **Max Depth** - The maximum depth the tree is allowed to grow to.

    The cross-validation parameters we used when pre-pruning was:
    - **Iters** - Sets total folds for cross-validation.
    - **maxit** - Sets the number of iterations for the random search.

To improve the performance time when training the model, we ran the hyper tuning and cross-validation in parallel, using the parallel and parallel map packages from R. Using the rpart.plot library, we then plotted out the pre-pruned tree for both the base data and oversampled data.

As seen in Figure 20, this tree has a maximum depth of 5. Here, the work experience score is the root node and the most important node for tree building. The tree then splits based on the work experience score is less than or above the value of 0.075. Both sides of the tree then branch out to the overtime decision node, which indicates this tree had overtime being the second most important node. The tree then branches off further based on other conditions stated in Figure 20 until it reaches the leaf nodes. The decision tree has a total of 10 leaf nodes with the majority of classifications made, classified as no attrition.

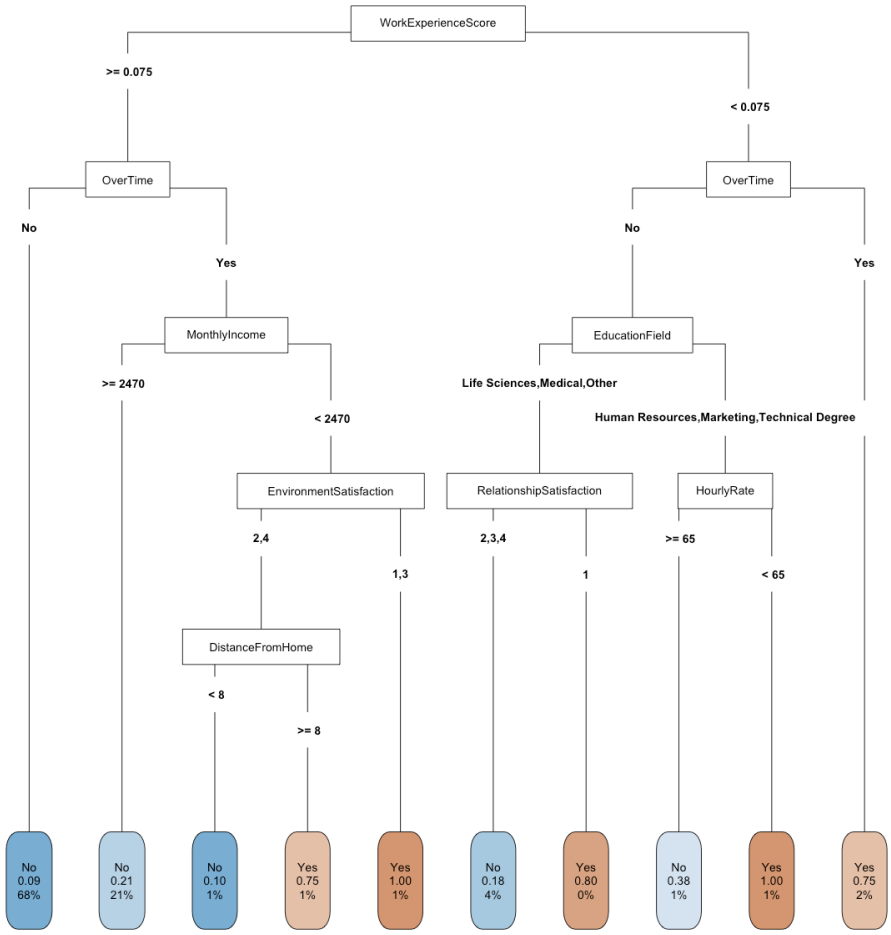


Figure 20: Base Data Decision Tree

From Figure 21 we see this tree has a maximum depth of 5. Here, overtime is the root node and the most important node for tree building. The tree then splits up based on overtime. In cases of no overtime, the tree branches out to the stock options level decision node. In cases of yes overtime, the tree branches out to the job role decision node. The tree then branches off further based on other conditions stated in Figure 21 until it reaches the leaf nodes. The decision tree has a total of 13 leaf nodes with a more balanced representation of yes and no attrition cases.

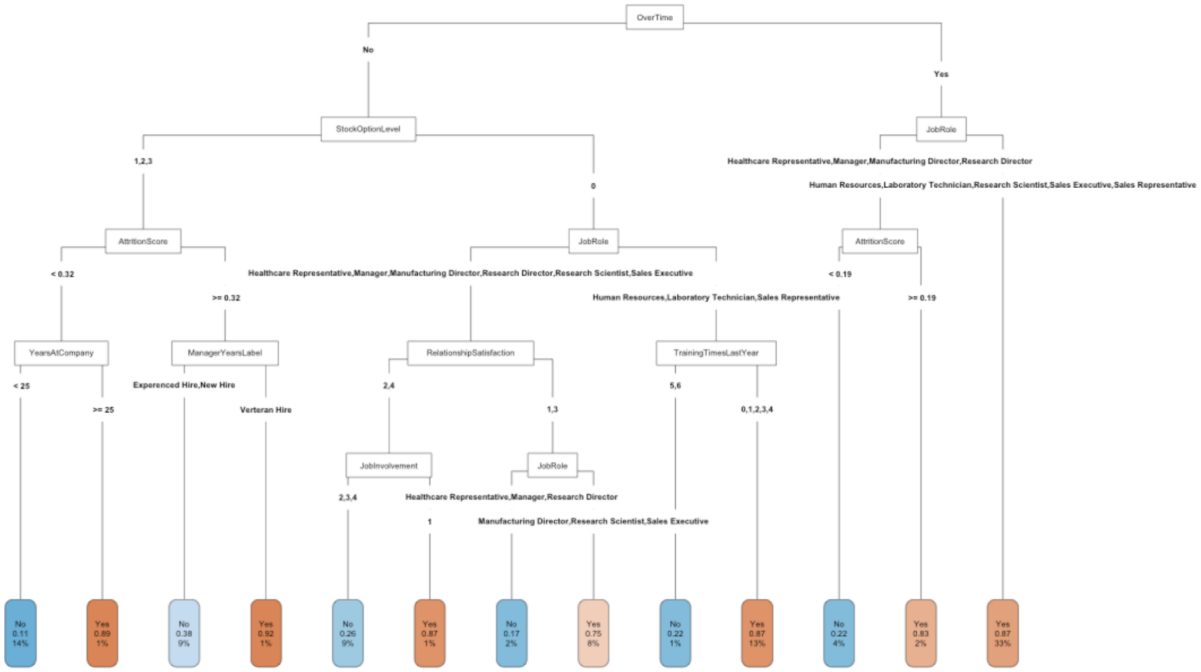


Figure 21: Oversampled Decision Decision Tree

2. **Post-Pruned Base Decision Tree** - The third implementation of the decision follows the base decision tree. Post-pruning works by first generating the full tree and then removing unnecessary branches. We manually tested out different complexity parameter values and found out that a cp value of 0.04 gave us an almost identical performance to the pre-pruned decision tree which also has cross-validation and hyper tuning. Using the rpart.plot library, we then plotted out the post-pruned tree as seen below:

Base Data Post Pruned Tree

From Figure 22 we see this tree has a maximum depth of 3. Here, the work experience score is the root node and the most important node for tree building. For cases where the work experience score was greater than or equal to 0.075, the tree immediately branches to the leaf node and classifies those cases as having no attrition. For cases of a work experience score greater than 0.075, the tree then splits up to an overtime decision node. In cases of no overtime, the tree branches out to the education field decision node before making a final classification. In cases of yes attrition, the tree immediately makes a yes classification. The has a total of 4 leaf nodes with most of its leaf nodes being classified as no attrition.

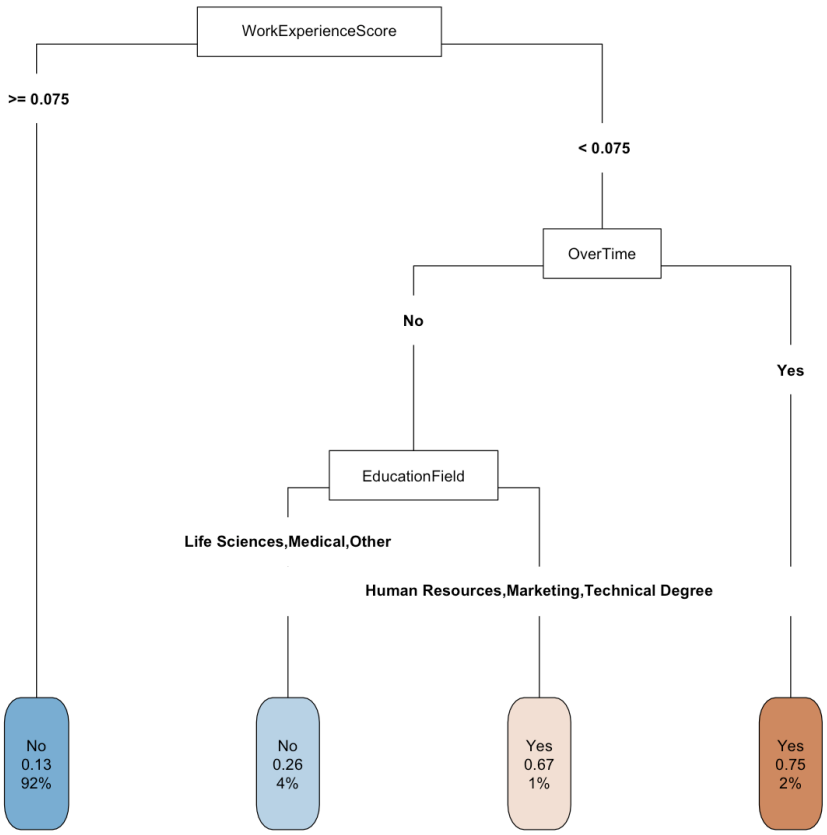


Figure 22: Base Data Post-Pruned Tree

From Figure 23 we see this tree has a maximum depth of 5. Similarly, to the pre-pruned tree, overtime is the root node and the most important node for tree building. The tree then splits up based on overtime. In cases of no overtime, the tree branches out to the stock options level decision node. In cases of yes overtime, the tree branches out to the job role decision node. The tree then branches off further based on other conditions stated in Figure 21 until it reaches the leaf nodes. The decision tree has a total of 8 leaf nodes with a more balanced representation of yes and no attrition cases.

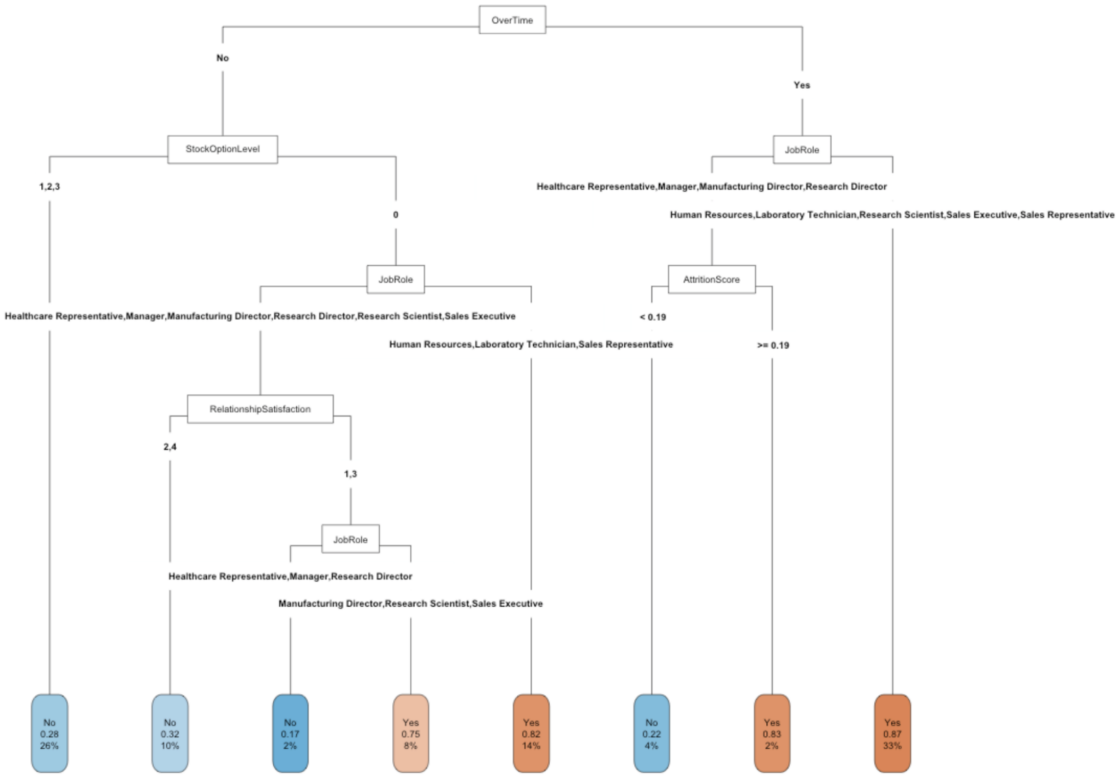


Figure 23: Oversampled Data Post-Pruned Tree

XGBoost

Extreme Gradient Boosting (XGBoost) is an advanced ensemble machine-learning technique built from gradient boosting. It is used to solve both classification and regression tasks. It is a good choice when handling the IBM HR dataset as higher weights are given to the minority class during model classification. Given that the dataset is imbalanced, this model should perform well.

XGBoost was built to be an improved version of gradient boosting as it uses regularised methods in order to prevent overfitting. Like gradient boosting, XGBoost works by combining weak learners to form strong learners. Several different trees are built in parallel with each tree trained on different subsets of the dataset. When training is complete, predictions are made by combining the results obtained from each tree to form a prediction. A drawback to XGBoost is it induces sparsity into the dataset since, unlike decision trees, it needs one hot encoding to work. The algorithm views dummy variables as independent of each other which can affect gain values when splitting off dummy variables.

XGBoost Implementation

Before implementing the model, we used one-hot encoding as the XGBoost algorithm is sensitive to non-numeric data. XGBoost isn't sensitive to data of different scales so we opted *not* to do feature scaling before training the model. To improve memory efficiency and training

speed, the data was converted to a matrix before training. When implementing XGBoost, we trained the model on two separate models using the base dataset and the oversampled data set.

Base XGBoost

The first implementation of the XGBoost was a base model with default parameters set and no cross-validation. The number of trees in the final model was set to 79. Using the diagram R library, we then plotted out the first tree for the base data and oversampled data.

Base Data XGBOOST Plot

This XGBoost model was made up of 45 trees. For practical purposes, we will be analysing the performance on just the first tree.

From Fig 24, we see this tree has a maximum depth of 6. Work experience score was used to make the first split and is the root node for tree building. It has the highest gain value of 8.82 which shows it was the most important feature when building the first tree. Its cover value of 1028 tells us it has a high number of observations related to it. The second highest gain value for this tree was overtime yes which had a gain of 5.43. The tree has a total of 41 leaf nodes with a high number of cases being probabilities where attrition was no

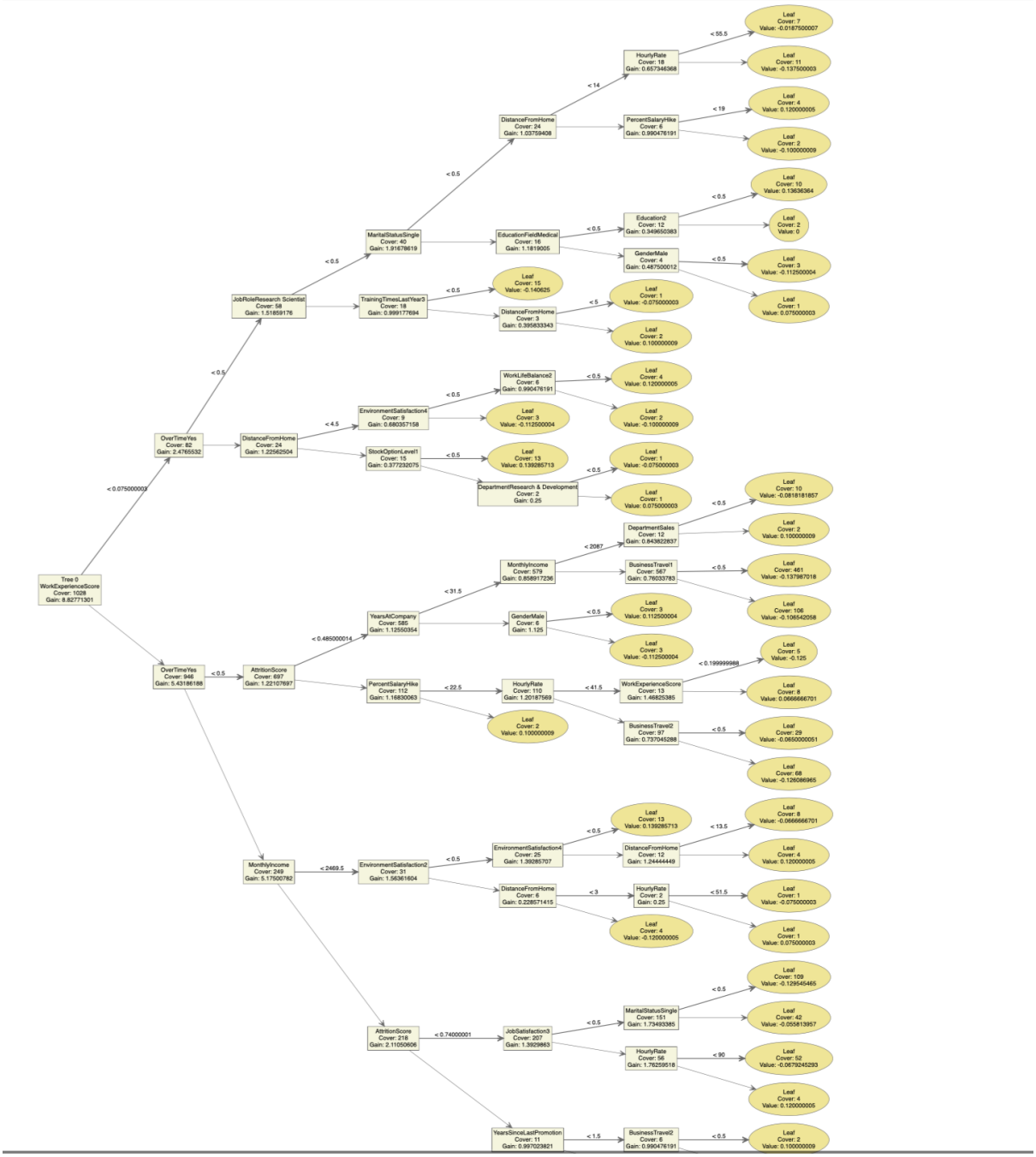


Figure 24: Base XGBoost Plot

Smote Data XGBOOST Plot

This XGBoost model was made up of 27 trees. For practical purposes, we will be analysing the performance on just the first tree.

From Figure 25, we see this tree has a maximum depth of 6. Overtime yes is used to make the first split and is the root node for tree building. It has the highest gain value of 19.84 which shows it was the most important feature when building the first XGBoost tree. The cover value of 825 tells us the relative number of observations related to the overtime yes feature. The

second highest gain value for this tree was attrition score followed by work experience. The tree has a total of 36 leaf nodes with a more balanced representation of yes and no attrition cases.

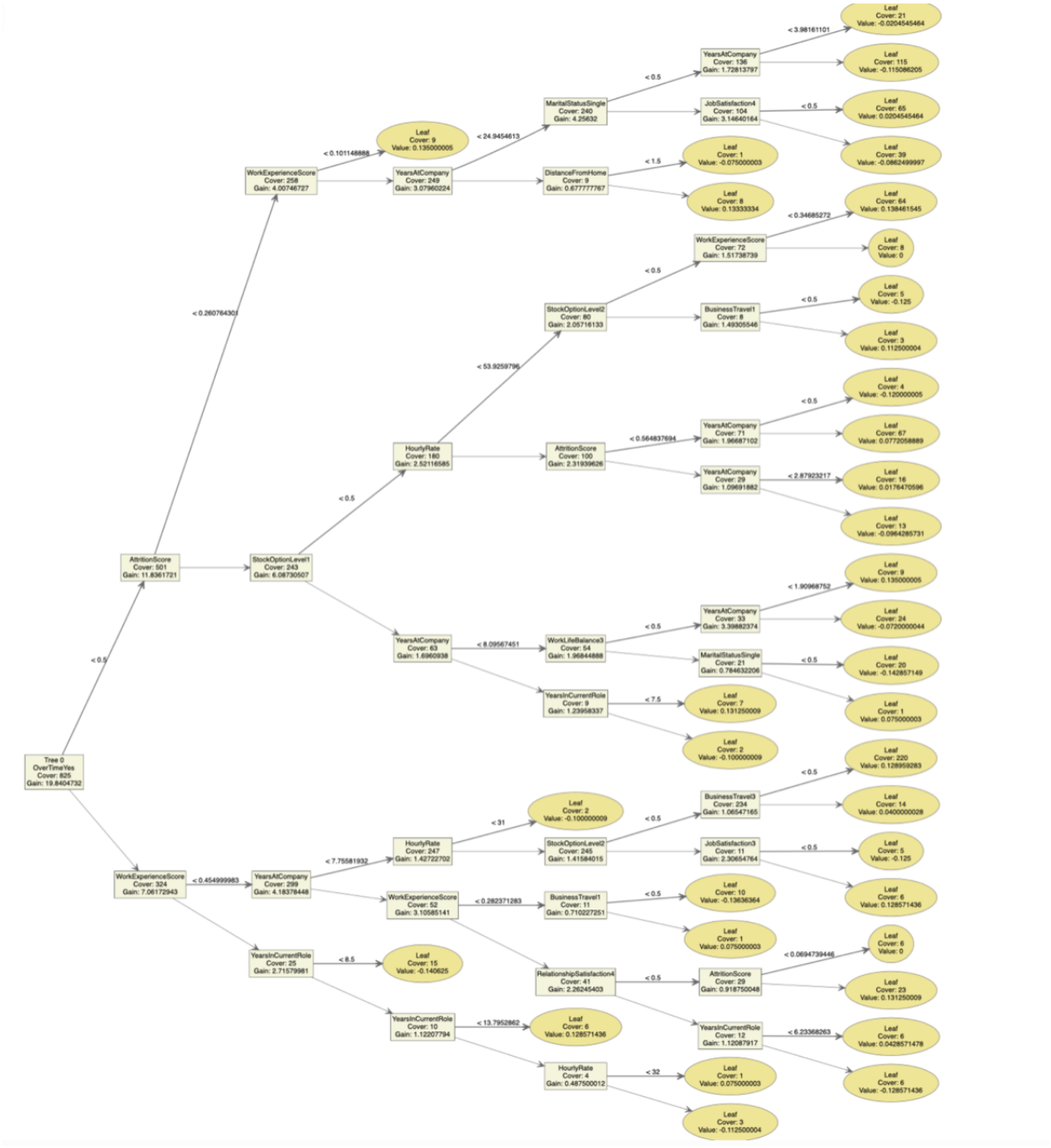


Figure 25: SMOTE Data XGBoost Plot

Hyper-Tuned XGBoost

Building on the base model we retrained the model using hyper turning and cross-validation. When performing the random search, we set to value to a maximum of 10 iterations. To maintain the proportions of observations in each training set concerning the original dataset we used a stratified resampling approach. We trained the model using 5-fold cross-validation.

The hyper-tuned parameters used when building the model were:

1. **Max_depth** - Maximum tree depth
2. **Min_child_weight** -The minimum sum of instance weight needed in a child node
3. **Subsample** - Subsample ratio of the training instances
4. **Colsample_bytree** - the ratio of subsamples used for columns during tree construction

Base Data Hyper-Tuned XGBOOST plot

This XGBoost model was made up of 73 trees. For practical purposes, we will be analysing the first and last trees.

From Figure 26, we see the first tree has a maximum depth of 5. Overtime yes is used to make the first split and is the root node for tree building. The gain value of 23.25 tells us that overtime was the most important feature used when building the tree. The cover value of 133 tells us the relative number of observations related to the overtime yes feature. The second highest gain and cover value for this tree was the attrition score for cases of overtime greater than 0.5 with a much lower gain of 5.93. The tree has a total of 11 leaf nodes with a high number of cases being probabilities where attrition was no.

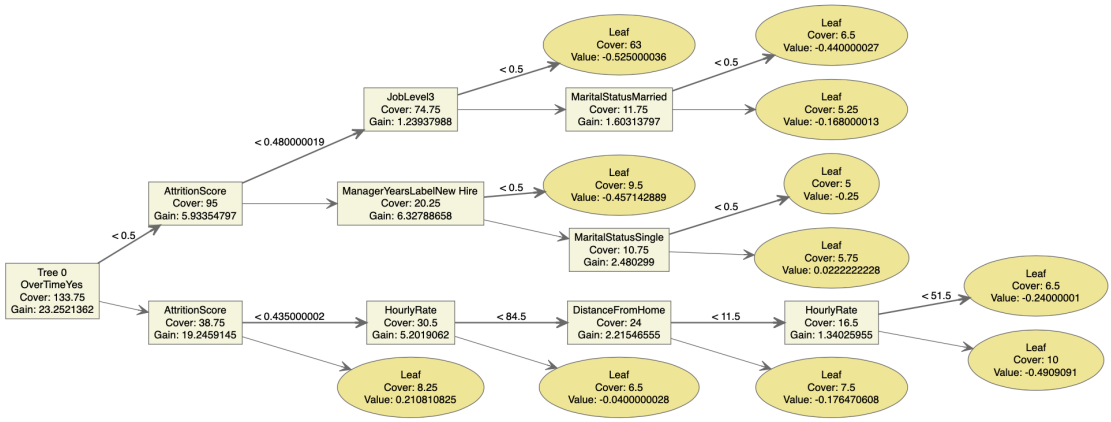


Figure 26: Base Data Hyper-Tuned XGBoost Plot

From Figure 27, we see the last tree has a maximum depth of 4. Hourly rate is used to make the first split and is the root node for tree building. The gain value of 2.6 tells us that the hourly rate was the most important feature used when building the tree. The cover value of 37 tells us the relative number of observations related to the hourly rate feature. The second highest gain value for this tree was the distance from home. The tree has a total of 6 leaf nodes with

a high number of cases being probabilities where attrition was no.

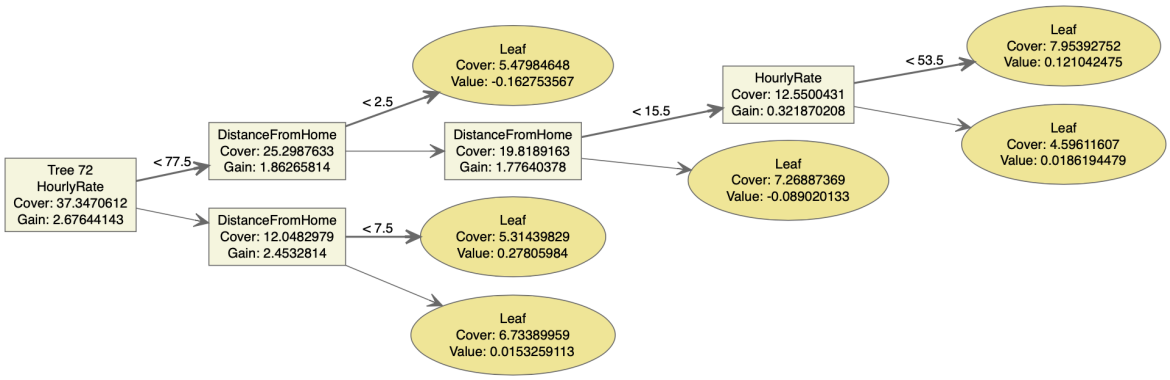


Figure 27: Base Data Hyper-Tuned XGBoost Plot

Oversampled Data Hyper-Tuned XGBOOST Plot

This XGBoost model was made up of 53 trees. For practical purposes, we will be analysing the first and last trees.

From figure 28 we see the first tree has a maximum depth of 7. Overtime yes is used to make the first split and is the root node for tree building. The gain value of 69.76 tells us that overtime was the most important feature used when building the tree. The cover value of 128 tells us the relative number of observations related to the overtime yes feature. The second highest gain and cover value for this tree were years at the company for cases of overtime greater than 0.5 with a much lower gain of 37.86. The tree has a total of 23 leaf nodes with a more balanced representation of the probabilities for yes and no cases.

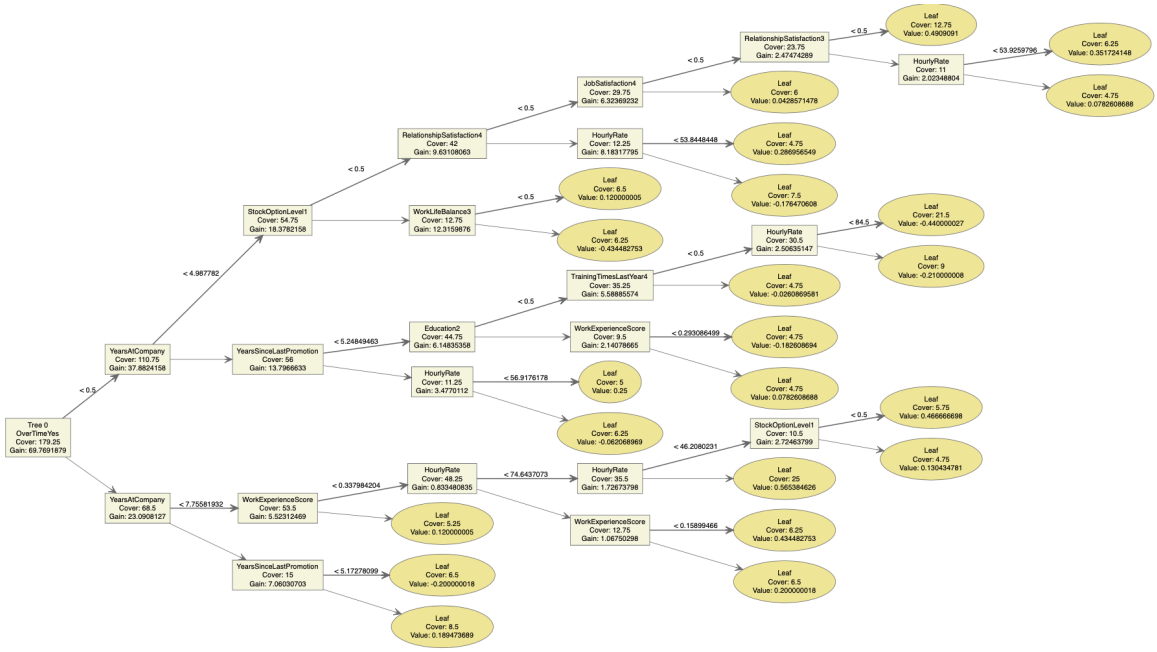


Figure 28: Base Data Hyper-Tuned XGBoost Plot

From Figure 29 we see the last tree has a maximum depth of 4. At this point gain values have dropped significantly from the first tree. Years after the last promotion the root node has a gain value of 0.49 which shows its lower importance for making predictions with the overtime yes feature. The total number of decision nodes used on this tree also dropped significantly to 5. The tree has a total of 7 leaf nodes with a more balanced representation of the probabilities for yes and no cases.

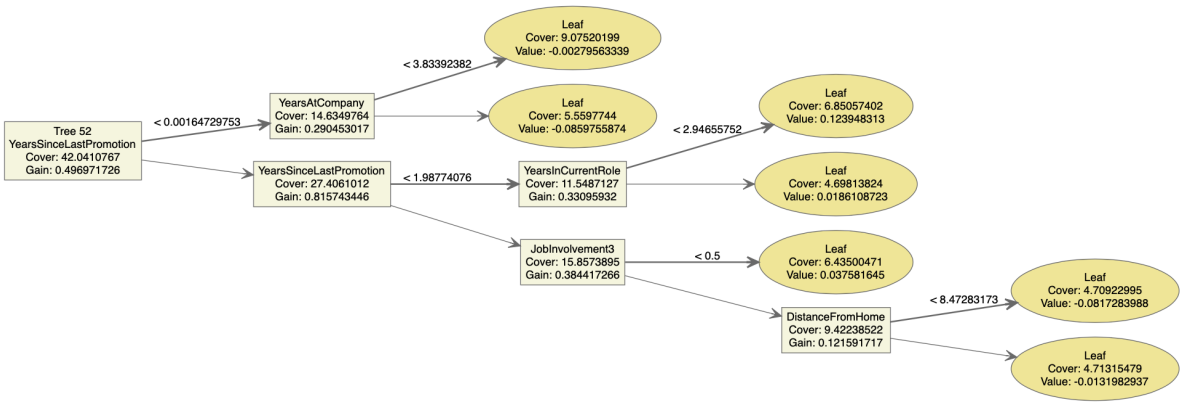


Figure 29: Base Data Hyper-Tuned XGBoost Plot

Evaluation Models

Each metric has its strengths and weaknesses, that is where using multiple is advantageous, each metric covers the informational gap left by the previous, providing more insight into the model's performance. We have used multiple metrics to evaluate the success of the models, these include accuracy, recall, precision, and F1 Score.

Many of these evaluation metrics can be represented in a confusion matrix, as seen in Figure XXX. Confusion matrices are a useful diagram to evaluate performance metrics at a glance. In this case, columns represent the results of the trained model, and rows represent the actual result according to the test data. The number of columns and rows corresponds to the number of possible categories being investigated, in this case, attrition could only be Yes or No hence this is a 2x2 matrix:

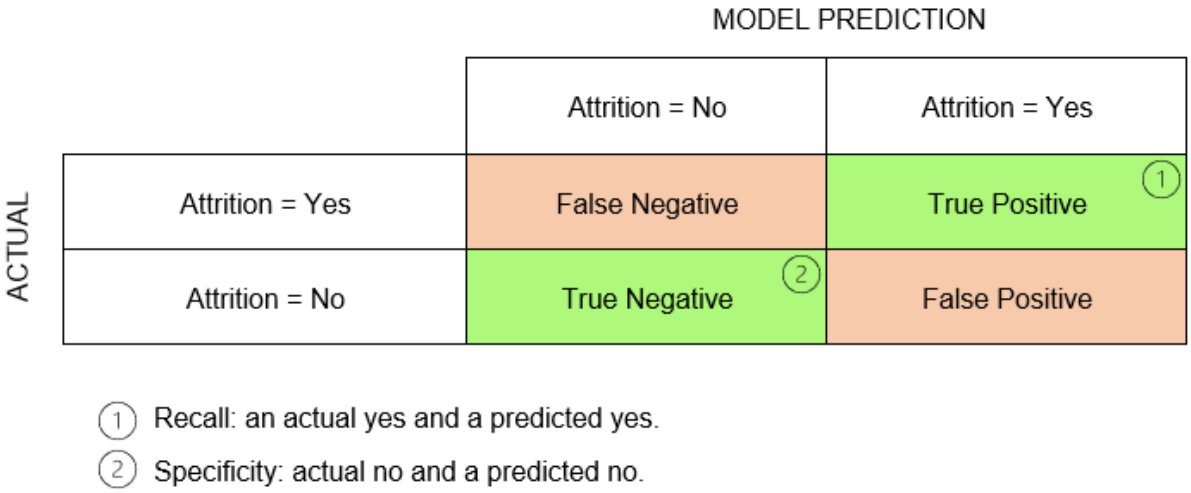


Figure 30: Attrition Confusion Matrix

The recall of a model is also known as the true positive, and it is a measure of how many instances the models predicted attrition correctly. Conversely, specificity represents the true negatives, i.e., how many instances the model correctly predicted that attrition would be No.

Accuracy is the simplest metric used for classification models. It is a general metric that is computed as the ratio of the total number of true predictions with the total number of predictions. This provides an output that indicates how well the model has been trained. Whilst conducting EDA, it was discovered that the dataset was heavily imbalanced. The dataset in question has a higher ratio of No attritions when compared to Yes attritions, 237:1470 or 16% to be exact. This imbalance in the dataset would normally impede the evaluation results, however, with the usage of SMOTE (Synthetic Minority Oversampling Technique) this can be rectified and improve the accuracy of the model.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + False\ Positive + True\ Negative + False\ Negative}$$

We want the model to correctly predict whether an employee will leave an organisation (attrition = Yes) or not (attrition = No), in this situation accuracy becomes a very useful evaluation metric. Accuracy counts the true positives and false positives. By observing the number of events that are accurately predicted, the model's performance can be determined.

The model must be able to recall and precisely identify the positive cases (employees leaving), and only the positive cases. This would be the case in an ideal world, but this is not an ideal world. Precision is useful when the false positives (employee was predicted to attrite but did not) outweigh the false negatives (employee was predicted to not attrite but did) in terms of cost to the business. Whereas recall is useful when the importance of false negatives is higher.

The model's task is to predict correctly whether an employee leaves or not. The metrics task is to evaluate how well the model can perform in its task. Taking this into consideration, recall is the best evaluation metric to score model performance as it can evaluate how well the model can predict the employee's leaving the company. Precision should still not be discarded as it can provide valuable insight into how well the model has been trained. In this scenario it is

best to use a precision-recall trade-off, tweaking the model to weigh recall more than precision. This way we can get closer to the ideal world results and produce more information on how well the models have been trained.

Additionally, precision and recall are great metrics when dealing with imbalanced datasets as they will only consider cases where an employee is leaving the organisation.

F1 Score is a general evaluation metric that combines precision and recall and computes those averages (harmonic mean). To summarise, it equally weighs both precision and recall. This will be an important metric to evaluate the model's performance. Using the F1 score means that the model has low false positives and low false negatives, which is a generalised case where the weight of false positives and false negatives is relatively equal.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

To conclude, accuracy and F1 Score are generalised performance metrics that do not tailor towards a certain niche and tend to work well as metrics in situations where the importance of identifying false positives and false negatives is relatively equal. Precision is a metric that tends to work well for problems involving the reduction of false positives. Whereas the main metric would be Recall, seeing as Recall tends to work well in situations where we are trying to reduce the number of false negatives i.e., a case where we predicted a customer would not attrite but in actuality, they did leave the company. Seeing as the prior mentioned case would cost the company more compared to the opposite case where we predict an employee is attrited, but they didn't in reality, we believe Recall would be the most important metric to use for assessing the models.

Results

We used three models, decision tree, KNN and XGBoost. Additionally, hyper-tuning was implemented to potentially improve the results. Figure 31 indicates the metric scores obtained from base models and the hyper-tuned models.

BaseModel	Accuracy	Recall	Precision	F1Score
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
Base Decision Tree	0.84	0.65	0.71	0.68
HyperTuned/PrePruned Decision Tree	0.85	0.56	0.77	0.63
PostPruned Base Decison Tree	0.84	0.49	0.77	0.56
Base XGBOOST	0.87	0.67	0.80	0.72
HyperTuned XGBOOST	0.86	0.65	0.78	0.70
HyperTuned KNN	0.85	0.47	0.94	0.55

Figure 31: Table showing performance metrics for all models (Without SMOTE)

We can see that the accuracy scores for all the models are relatively the same, with all of them being over 80%, the highest being 87% (Base XGBoost model). This indicates that the hyper-tuning of the parameters is not affecting the model very much.

In the recall column, we noticed a similar trend of performance across all models. The best-performing models are the base models. However, this is unexpected because hyper tuning the model parameters is meant to improve the performance of the base model.

The reason behind the hyper-tuned models performing worse than the base models could be a side effect of not implementing any oversampling techniques (SMOTE) on an imbalanced dataset.

SmoteModel	Smote.Accuracy	Smote.Recall	Smote.Precision	Smote.F1Score
<chr>	<dbl>	<dbl>	<dbl>	<dbl>
Base Decision Tree	0.63	0.81	0.44	0.55
HyperTuned/PrePruned Decision Tree	0.65	0.82	0.46	0.58
PostPruned Base Decison Tree	0.67	0.81	0.47	0.59
Base XGBOOST	0.67	0.81	0.47	0.59
HyperTuned XGBOOST	0.73	0.83	0.53	0.64
HyperTuned KNN	0.54	0.75	0.37	0.48

Figure 32: Table showing performance metrics for all models (With SMOTE)

Figure 33 shows the metric scores when SMOTE is implemented. With SMOTE, both the base and hyper tuned models performed well. However, this time, the hyper tuned models performed the best, especially the hyper tuned XGBoost scoring a 0.83 in the recall.

In terms of overall performance, the hyper tuned XGBoost model is the best, scoring the best in all metrics when compared to other models. We see a massive spike in recall with many models scoring over 80% and hyper tuned XGBoost being the highest with 83%. This is a significant improvement to what the recall was before SMOTE was applied to the dataset.

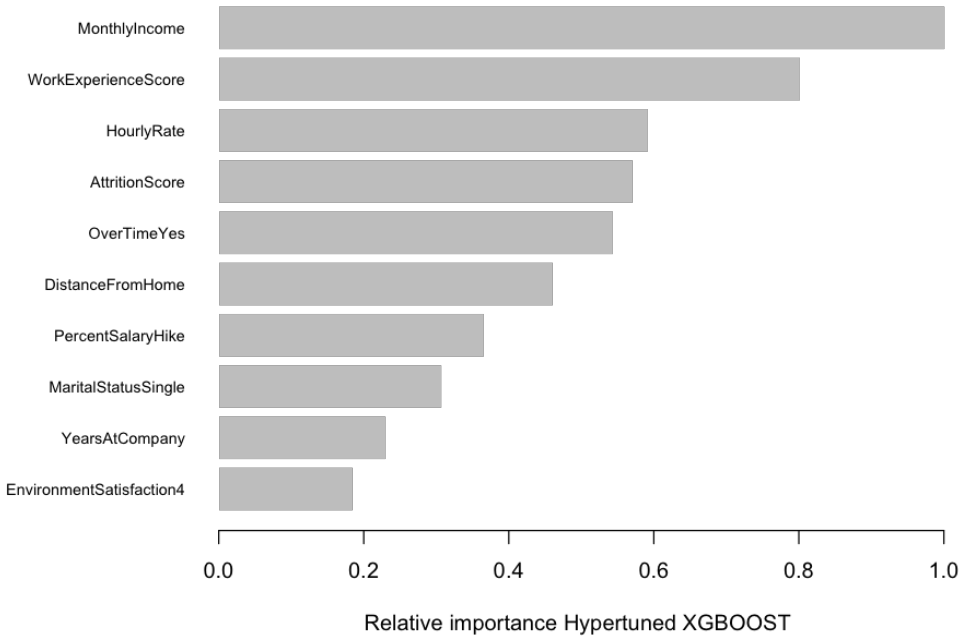


Figure 33: Table showing important features for Hyper-Tuned XGBoost (Without SMOTE)

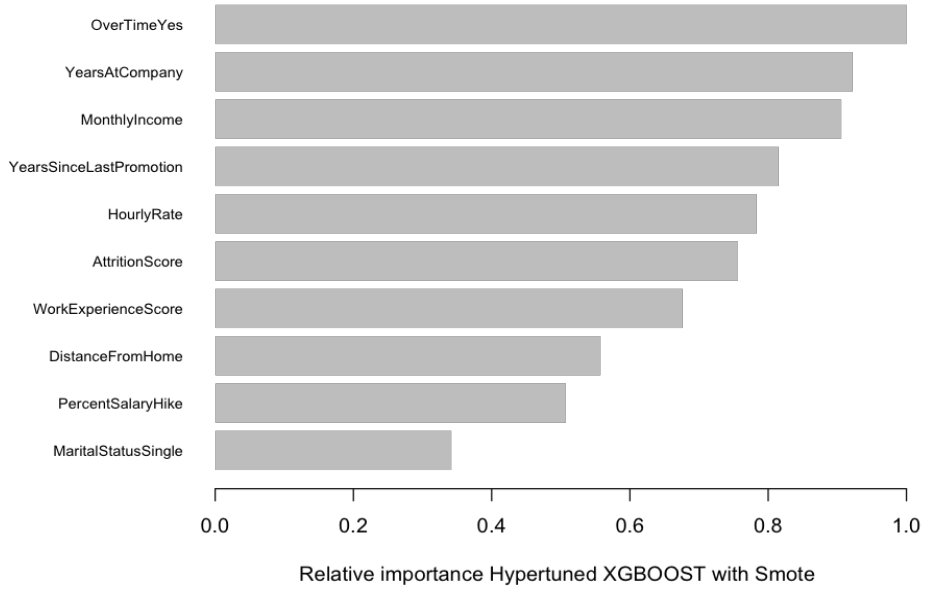


Figure 34: Table showing important features for Hypertuned XGBoost (With SMOTE)

The best performing model with smote and without smote was used to represent the importance of using oversampling techniques, and the significant impact it leaves when interpreting the results. Taking into consideration the imbalance of the dataset it should be known that the combination of hypertuned and SMOTE outputted the more reliable outcome, this was because hyper tuning reacted as expected with SMOTE rather than without.

Reviewing the aforementioned results, the Hypertuned XGBoost model can be deemed the main model as it can explain 83% of the data accurately. As a base model, this stands firm because, as the dataset grows, there would be more data points to work with. Thus, allowing the model to become more accurate and make better predictions.

Performances of the models on specific class cases were also considered. This included attrition yes or no.

SmoteModelAttritionYes	Smote.RecallYes	Smote.PrecisionYes	Smote.F1_ScoreYes
<chr>	<dbl>	<dbl>	<dbl>
Base Decision Tree	0.71	0.26	0.38
HyperTuned/PrePruned Decision Tree	0.72	0.28	0.40
PostPruned Base Decison Tree	0.71	0.29	0.41
Base XGBOOST	0.71	0.29	0.41
HyperTuned XGBOOST	0.74	0.34	0.47
HyperTuned KNN	0.62	0.21	0.31

Figure 35: Evaluation metric with a specific class (Attrition Yes) and SMOTE

SmoteModelAttritionNo	Smote.RecallNo	Smote.PrecisionNo	Smote.F1_ScoreNo
<chr>	<dbl>	<dbl>	<dbl>
Base Decision Tree	0.92	0.61	0.73
HyperTuned/PrePruned Decision Tree	0.92	0.64	0.75
PostPruned Base Decison Tree	0.92	0.66	0.77
Base XGBOOST	0.92	0.66	0.77
HyperTuned XGBOOST	0.93	0.72	0.81
HyperTuned KNN	0.88	0.53	0.66

Figure 36:Evaluation metric with a specific class (Attrition No) and SMOTE

Predicting the positive/negative cases in the Attrition is represented by figures 33 and 34. The positive cases in the dataset indicate the case where an employee left the company i.e., the value in the attrition column was a ‘yes’ and the models did relatively well with those cases with an average (F1 Score) across all models of 70%. Unsurprisingly, hyper tuned XGBoost gives the highest recall of 93%.

Here we can see that the recall for most of the models is above 90%, with the hyper tuned XGBoost model giving a recall of 93%. This indicates that the model can correctly predict the cases where an employee did not leave accurately 93% of the time which is a really good performance. These results make sense seeing as we have a large number of ‘No’ cases

compared to the ‘Yes’ cases, so the model has a larger sample space to train on and make more accurate predictions.

To conclude, the results obtained from the Smote hyper-tuned XGBOOST gave the best value for false negatives which satisfied the evaluation metric criteria previously stated. From figure(36) we predicted a total of 287 people to stay with the company with only 19 people(6%) out of the 287 leaving the company. We then looked at the total accuracy, recall and precision to get further insight into the results of the chosen model.

	SmoteModel	Smote.Accuracy	Smote.Recall	Smote.Precision	Smote.F1Score
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
5	HyperTuned XGBOOST	0.73	0.83	0.53	0.64

Figure 37: Table Showing Performance Metrics for Hyper-Tuned XGBoost

Accuracy - Our model had a total accuracy of 73 percent. This shows the model had fairly high performance when classifying true negatives and positives. which means we were able to correctly classify 73 percent of the people who either stayed or left the company.

Recall - Our model had a total recall of 83 percent. This is obtained by adding the recall values of people attrition and not attrition. This shows the model was able to minimise false negatives when it came to predicting employees’ choices regarding organisation commitment. However, this is not a true indication of the model's performance concerning the evaluation as the recall value obtained might be skewed to either of the two scenarios(Attrition vs not Attrition).

Precision - The model had a total precision of 53 percent. This is obtained by adding the precision values of people attrition vs not attrition. This shows the model's predictive power struggled when minimising false positives. Additionally, it is a true indication of the model’s performance concerning predictions. It should be noted that values obtained may be highly skewed to either of the two scenarios (attrition vs not attrition).

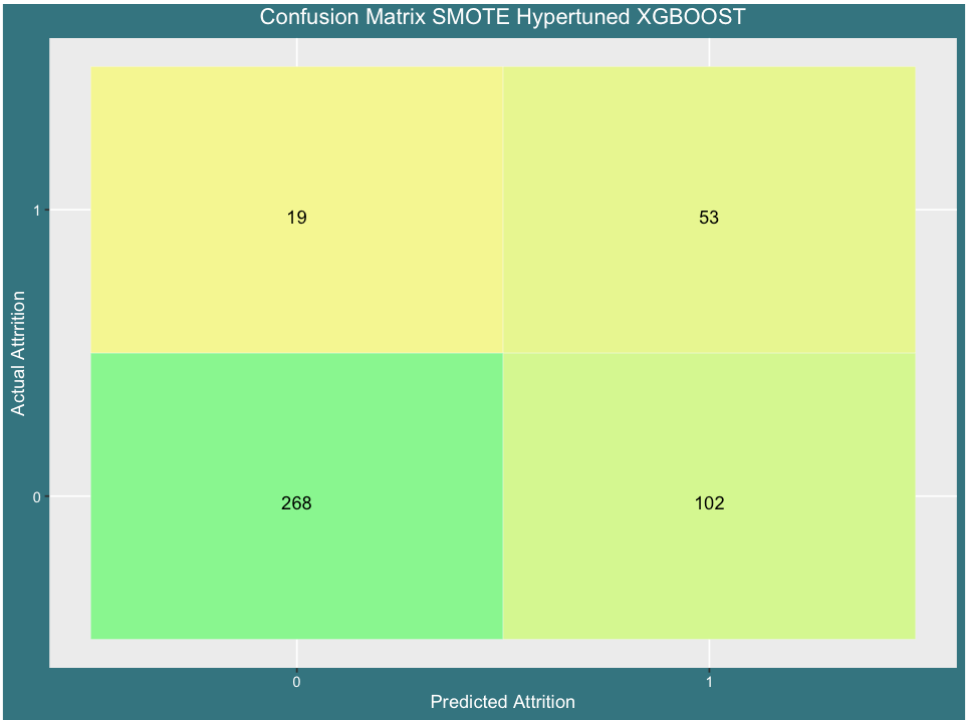


Figure 38: Confusion Matrix

To get a better understanding of hyper-tuning the XGBoost model's performance the data was split into two scenarios (attrition vs not attrition).

	SmoteAttritionYes	Smote.RecallYes	Smote.PrecisionYes	Smote.F1_ScoreYes
	<chr>	<dbl>	<dbl>	<dbl>
5	HyperTuned XGBOOST	0.74	0.34	0.47

Figure 39: Hypertuning XGBoost (Attrition Yes Table Prediction and SMOTE)

	SmoteAttritionNo	Smote.RecallNo	Smote.PrecisionNo	Smote.F1_ScoreNo
	<chr>	<dbl>	<dbl>	<dbl>
5	HyperTuned XGBOOST	0.93	0.72	0.81

Figure 40: Hypertuning XGBoost (Attrition No Table Prediction and Smote)

Attrition Yes Performance

When focused solely on cases where people chose to attrition the model performed well in certain areas and not so well in others. A recall score of 0.74 shows that the model correctly predicted 74 percent of the people who chose to leave the company. A precision score of 0.34 shows for the total predictions made when predicting cases where people attrition we were able to correctly predict a person staying with the company 34 percent of the time. The overall performance wasn't ideal, especially when evaluating the F1 Score. However, this performance fits the business requirements as the first objective was reducing the number of false negatives.

Attrition No Performance

When focused solely on cases where people chose not to attrition the model performed well. A recall score of 0.93 shows that the model correctly predicted 93 percent of the people who chose to stay. A precision score of 0.72 shows that the total predictions made when predicting no attrition cases were able to correctly predict a person staying with the company 72 percent of the time.

Discussion

Overall performance was reasonable. However, we can reference the poor performance when predicting false positives to the quality of the dataset, particularly the imbalance. The model returned a high number of false negatives (employees predicted to leave when they chose to stay). In a business sense, this is beneficial to the organisation because they retain their employees, and in turn, do not lose money on lost productivity or hiring a replacement. On the other hand, false negatives decrease the performance of the model and damage future optimization and reliability.

As this was a randomly generated dataset, it is expected the model would struggle in some aspects. False positives are to be expected. Consider scenarios observed during EDA, such as cases where employees who live close to their workplace are leaving at a much higher rate. This could be down to other factors impacting an individual's decision.

The base model, in terms of false negatives, performed badly. This improved drastically once oversampling was applied to the dataset. To check how the model performed in terms of predicting the most important features, a secondary model trained with just the 10 most important features from the figure(39) was built to see if model performance will increase as complexity is now reduced with just 10 features.

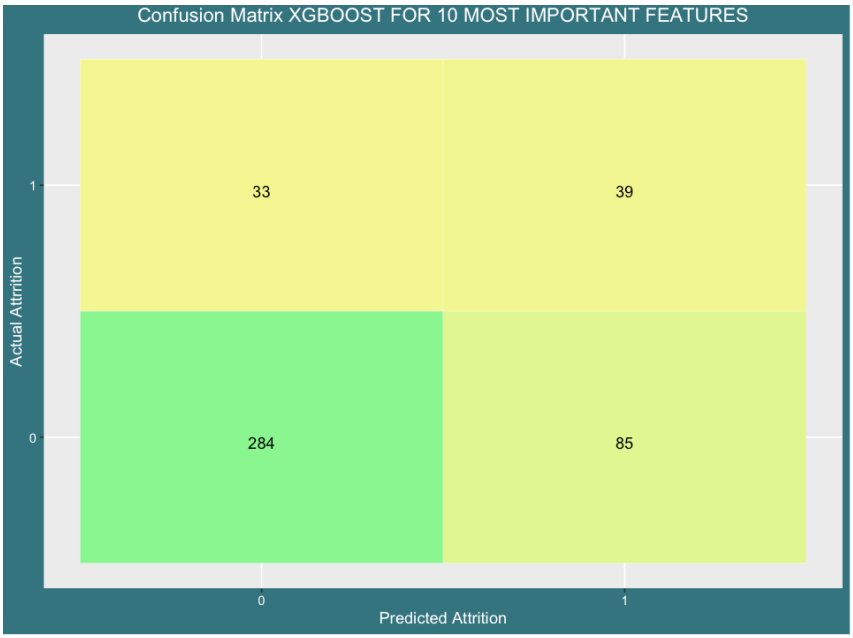


Figure 41: Confusion Matrix

In terms of predicting true negatives, model performance was similar. However, there was a trade-off between false positives, true positives and false negatives with false positive values and true positives values declining whilst false negatives showed an incline. An assumption for this would have to do with some of the lesser features still having some level of factor through combinations of several of them when determining the attrition.

The initial thoughts when observing the dataset were that job satisfaction and salary would become the major factors in impeding organisational commitment for employees. However,

this was superseded after completing the XGBoost model. As shown in Figure 8 overtime was the most important factor when discussing employee attrition within the IBM HR dataset. ‘OvertimeYes’ indicates if an employee is working overtime. This column being the most relevant is logical viewing from the perspective of an employee being more likely to leave due to the workload of regularly working overtime. The implications of overtime were noted during EDA, due to the significant impact it had on employee satisfaction, specifically environmental satisfaction.

Upon reviewing the model, ‘YearsAtCompany’ is identified as the second most significant factor when it comes to attrition rate. It indicates how long an employee has spent working for one company. This evidence can also be found in the EDA, where early hires were more prone in opting to leave the company within the first few years. It could be assumed that this is impacted by other sub-factors. For example, an individual not liking the environment, negative work relations, poor job involvement or salary pay.

A monthly income is tied to an individual's lifestyle hence, it is considered a big factor in employee attrition. The hypertuned XGBoost proves this by rating it third most influential to employee attrition. An individual's income can dramatically impact their motivations. This is taken further by the carrot and stick theory. This theory attempts to reward employees for their efforts in hopes to increase efficiency, and better connecting them to organisational goals Maria Frangieh (2021).

YearsSinceLastPromotion indicates the number of years since the last time an employee was promoted. Employees with a high number of YearsSinceLastPromotion, due to human nature, will feel stuck and stagnant and this would eventually lead to positive cases of attrition. Using a common-sense check on the model shows the indications on the dimensions of the graph aren't too far-fetched.

When initially observing the dataset the satisfaction attributes were considered to potentially be the most significant factor in employee attrition. However, this was proven incorrect after modelling they were not even in the top 10 when looking at figure 39. An assumption can be made that any of these satisfaction attributes are a sub-factor to employee attrition and overshadowed by other factors that can directly affect the probability of attrition being considered by employees.

Distance from home can be a low factor in employee attrition due the technological advancements such as version control, VPN, and shared documents. Additionally, with the Covid pandemic, it is considered a norm to work from home, and distance has become irrelevant and effortless to reduce in most workplace environments.

Conclusion

To conclude, although performance varied depending on the models used, every model was able to predict at least a small amount of attrition. Considering the company, we are investigating currently has no means of predicting employee departure, any insights our project provides can help to create strategies to mitigate the impact of attrition. In particular, models that were best at predicting true positives (the case an employee was predicted to leave and actually left), could be used to implement pre-emptive retention strategies. Retention strategies could include:

- Offering more employee benefits (such as company equipment/cars/trips).
- Providing the employee with more positive evaluations (a cost-effective way to improve their perception of the workplace).
- Offering a pay rise (to avoid losing valuable employees to the competition).
- Offering bonuses for every year spent working for the company.

Another strategy to mitigate the impact of attrition involves using the models that were able to predict attrition accurately some of the time, but also predicted a significant number of false negatives (the case that an employee was predicted to stay, but actually they left without warning). In this case, it would be financially unwise to apply retention strategies to all of the employees that are predicted to stay, just so that the company can be prepared in the rare event that these “settled” employees leave. In this instance, we recommend that no money be spent to encourage these employees to stay - instead, the company should begin to set aside money to ensure that there are funds available to train and hire new recruits in the near future. No money should be spent on advertising jobs or re-hiring straight away. However, the number of false negatives predicted, can serve as a valuable indication of how many employees may leave without warning, and provide an estimate of the funds that might be needed to replace them (so the company may start preparing the resources needed to recruit soon).

Considering our project’s results, we can conclude that ‘OverTime’ among employees in the IBM HR dataset has the most significant impact on attrition. This is followed by ‘MonthlyIncome’ and ‘YearSinceLastPromotion’. Although these are important factors in predicting employee attrition, this does not guarantee the models will always make correct predictions. These inaccurate predictions could be a by-product of extenuating factors (i.e., job offer, relocation, the decline in health) which cannot be predicted with the current dataset. Additionally, it was discovered that the satisfaction attributes had an indirect impact on employee attrition as they were interlinked with many other significant factors.

In terms of findings and conclusions, we can draw from EDA:

Examination of ‘WorkLifeBalance’ satisfaction scores revealed that employees who rate their work-life balance as 3 ‘Better’ are most likely to leave. This is because this group of employees spends the minimum amount of time at the office - a potential symptom of employee dissatisfaction. As such, a survey of employees' work-life balance might be a cost-effective way to narrow down what employees might leave the company.

Furthermore, both the ‘TotalWorkingYears’ and ‘MonthlyIncome’ data suggest that employees who have the lowest incomes (0-2k) are equally as likely to stay in the company as they are to leave the company. This true in their first year of employment. Given that the main aim of this project was to predict attrition, to avoid the unforeseen cost of replacing an employee, knowing that 50% of employees will leave in the first year is important information. This information can be exploited to hire graduates as the company may pay these employees the minimum company salary and prepare for the turnover of graduates to be high year on year.

Finally, inspection of the “extreme outliers” found during the investigation of ‘YearsSinceLastPromotion’, revealed that these outliers are actually the company’s most loyal employees. As such, we recommend that any employee who has been working for the company for 13 years or more should be interviewed, as these loyal employees might hold important information on how to best retain employees in the long term.

Future Work

If provided additional opportunities to re-attempt the project with an extension to the existing time constraint, there are numerous elements we would consider, as a plot of improvement to improve the outcome of the product from this project. Firstly, if given time, we could commence by learning and understanding hyper models more, in particular XGBoost. As XGBoost presents countless parameters for the model, we were only able to utilise 4, as they were the only ones we, as a team, were able to learn, with a total range of 30, and thus this would be a commencement point for future work. Secondly, we would also consider further train-test validation splits to prevent leakages. Data leakages are one of the biggest issues within the ML space when developing predictive models. This, as a result, can lead to invalidating the estimated performance of the mode being constructed (Brownlee, 2016). Thirdly, were there no time constraints on this project, we would have liked to use ROC/AUC curves to evaluate the performance of the models.

We would also implement principal component analysis in a situation where we were provided with a lot more dimensions and/or rows. The principal component analysis is a method of dimensionality reduction usually when working with large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

References

Average household income, UK: financial year 2020. (2021, January 21). Average household income, UK - Office for National Statistics. Retrieved November 27, 2022, from https://www.ons.gov.uk/peoplepopulationandcommunity/personalandhouseholdfinances/incomeandwealth/bulletins/householddisposableincomeandinequality/financialyear2020

Brownlee, J. (2016, August 2). *Data Leakage in Machine Learning - MachineLearningMastery.com*. Machine Learning Mastery. Retrieved November 24, 2022, from https://machinelearningmastery.com/data-leakage-machine-learning/

Brownlee, J. (2016, September 9). *A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning - MachineLearningMastery.com*. Machine Learning Mastery. Retrieved November 28, 2022, from https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

Brownlee, J. (2020, January 17). *SMOTE for Imbalanced Classification with Python - MachineLearningMastery.com*. Machine Learning Mastery. Retrieved November 27, 2022, from https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Chowdary, D. H. (2020, May 28). *Decision Trees Explained With a Practical Example*. Towards AI. Retrieved November 28, 2022, from https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53

The Effect of the Carrot and Stick Transactional Leadership style in Motivating Employees in SMEs. (n.d.). Revista de Management Comparat International. Retrieved November 28, 2022, from https://www.rmci.ase.ro/no22vol2/10.pdf

Guide, S. (2022, September 9). *How to Leverage KNN Algorithm in Machine Learning?* Simplilearn. Retrieved November 28, 2022, from https://www.simplilearn.com/tutorials/machine-learning-tutorial/knn-in-python

Hoare, J. (n.d.). *Pruning Decision Trees and Machine Learning*. Displayr. Retrieved November 28, 2022, from https://www.displayr.com/machine-learning-pruning-decision-trees/

Home. (n.d.). YouTube. Retrieved November 28, 2022, from https://www.anyscale.com/blog/what-is-hyper-tuning

How to find the optimal value of K in KNN? (2020, May 23). Towards Data Science. Retrieved November 28, 2022, from https://towardsdatascience.com/how-to-find-the-optimal-value-of-k-in-knn-35d936e554eb

IBM HR Analytics Employee Attrition & Performance. (2016). Kaggle. Retrieved November 23, 2022, from https://www.kaggle.com/datasets/uniabhi/ibm-hr-analytics-employee-attrition-performance

Machine Learning: When to perform a Feature Scaling? (n.d.). atoti. Retrieved November 23, 2022, from https://www.atoti.io/articles/when-to-perform-a-feature-scaling/

McLeod, S. (2019). *Box Plot Explained: Interpretation, Examples, & Comparison*. Simply Psychology. Retrieved November 23, 2022, from https://www.simplypsychology.org/boxplots.html

Nguyen, Q. H., Ly, H., Ho, L. S., Al-Ansari, N., Le, H. V., Tran, V. Q., Prakash, I., & Pham, B. T. (2021). Influence of Data Splitting on Performance of Machine Learning Models in Prediction of Shear Strength of Soil. *Mathematical Problems in Engineering Journal*. https://doi.org/10.1155/2021/4832864

Orellana, E. (n.d.). *SMOTE. A technique to overcome class imbalance… | by Emilia Orellana | Medium*. Emilia Orellana. Retrieved November 27, 2022, from https://emilia-orellana44.medium.com/smote-2acd5dd09948

Orellana, E. (n.d.). *SMOTE. A technique to overcome class imbalance… | by Emilia Orellana | Medium*. Emilia Orellana. Retrieved November 28, 2022, from https://emilia-orellana44.medium.com/smote-2acd5dd09948

Post-Pruning and Pre-Pruning in Decision Tree | by akhil anand | Analytics Vidhya. (2020, December 10). Medium. Retrieved November 28, 2022, from https://medium.com/analytics-vidhya/post-pruning-and-pre-pruning-in-decision-tree-561f3df73e65

7.1.6. What are outliers in the data? (n.d.). Information Technology Laboratory. Retrieved November 27, 2022, from https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm

Sharma, A. (2020, July 6). *What is Skewness in Statistics? | Statistics for Data Science*. Analytics Vidhya. Retrieved November 23, 2022, from https://www.analyticsvidhya.com/blog/2020/07/what-is-skewness-statistics/

SMOTE function. (n.d.). RDocumentation. Retrieved November 27, 2022, from https://www.rdocumentation.org/packages/DMwR/versions/0.4.1/topics/SMOTE

Telang, P. (2021, November 26). *What Is The Objective Of Exploratory Data Analysis? - Techcanvass*. Business Analysis Blog. Retrieved November 22, 2022, from https://businessanalyst.techcanvass.com/objective-of-exploratory-data-analysis/

Understanding XGBoost Algorithm | What is XGBoost Algorithm? (n.d.). Great Learning. Retrieved November 28, 2022, from https://www.mygreatlearning.com/blog/xgboost-algorithm/

Winsorized Mean Definition. (n.d.). Investopedia. Retrieved November 27, 2022, from https://www.investopedia.com/terms/w/winsorized_mean.asp

XGBoost Parameters — xgboost 1.7.1 documentation. (n.d.). XGBoost Documentation. Retrieved November 28, 2022, from https://xgboost.readthedocs.io/en/stable/parameter.html

Appendix A: IBM HR Analytics Employee Attrition & Performance Field Rundown

The following table describes the fields present (pre-processing) within the selected dataset for this project. Each field is further analysed with the type of data followed by a description. Any information provided within the description field references initial documentation provided via the source (IBM, n.d.).

Field Name	Field Type	Description
Age	Int	Representation of employee's age
Attrition	Boolean	Representation of an employee's departure from the organisation for any reason.
BusinessTravel	String	Brief description of how often an employee travels, categorised as travel_rarely, travel_frequently
DailyRate	Int	Calculation of how much an individual is paid, currency not defined.
Department	String	Department Name
DistanceFromHome	Int	Distance of employee's home address to the workplace (no measurement mentioned)
Education	Int	Education is ranked in a categorical scale, whereby 1: Below College 2. College 3. Bachelors 4. Masters 5. Doctors
EducationField	String	Education field is mentioned as a string value.
EmployeeCount	Int	The value represents the employee referred to, and

		how many times present in the database.
EmployeeNumber	Int	Unique employee ID for user
EnvironmentSatisfaction	Int	Environment satisfaction is ranked on a scale of 1-5, whereby: 1 'Low' 2 'Medium' 3 'High' 4 'Very High'
Gender	Boolean	Marked as Male **OR** Female
HourlyRate	HourlyRate	Description of how much an individual is paid on an hourly rate. Currency is not defined.
JobInvolvement	Int	1 'Low' 2 'Medium' 3 'High' 4 'Very High'
JobLevel	Int	Integer based on a scale of 1 to 5. Scale is not provided in the data dictionary of the dataset.
JobRole	String	Description of individual's job title
JobSatisfaction		1 'Low' 2 'Medium' 3 'High' 4 'Very High'
MaritalStatus	String	Statement of an individual's marital status
MonthlyIncome	Int	Value of individual's monthly income. Currency not defined.
MonthlyRate	Int	Value of individual's monthly rate.

NumCompaniesWorked	Int	Value of individual's prior experience based on the number of previous companies.
Over18	Boolean	Boolean value based on individual's age.
OverTime	Boolean	Boolean value of T or F representing if individual partakes in overtime.
PercentSalaryHike	Int	Percentage representation of salary increase. No definition was provided for the period discussed for the salary hike.
PerformanceRating	Int	1 'Low' 2 'Good' 3 'Excellent' 4 'Outstanding'
RelationshipSatisfaction	Int	1 'Low' 2 'Medium' 3 'High' 4 'Very High'
StandardHours	Int	Integer representation of scheduled hours per individual.
StockOptionLevel	Int	Integer representation of stock level enrolment, scale representation not provided in the data dictionary.
TotalWorkingYears	Int	Total working years within IBM per individual.
TrainingTimesLastYear	Int	Integer description on how often an individual has partaken in training over the previous year.
WorkLifeBalance	Int	1 'Bad' 2 'Good' 3 'Better' 4 'Best'

YearsAtCompany	Int	Value describing the number of years at IBM.
YearsInCurrentRole	Int	Value describes the number of years in an existing role.
YearsSinceLastPromotion	Int	Value describing the number of years since the last promotion.
YearsWithCurrManager	Int	Value describing the number of years with the current manager.

Source: IBM HR Analytics Employee Attrition and Performance Dataset (IBM, n.d.)

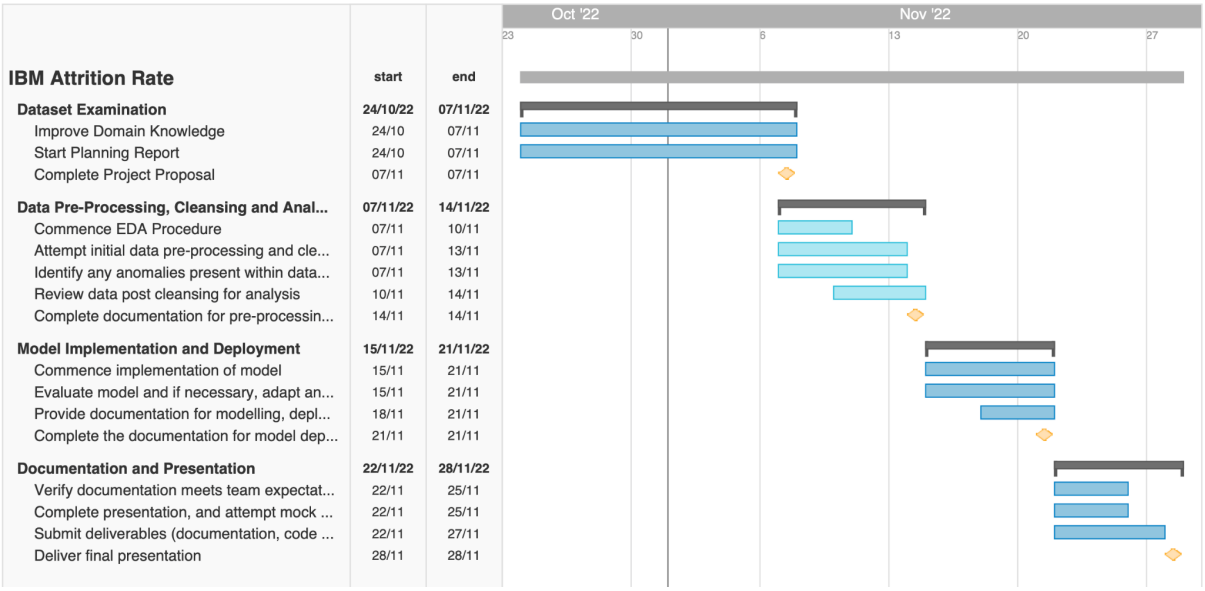
Appendix B: Results of Data Type Examination

Field Name	Initial Type	Data Type
Age	NUMERIC	ORDINAL
Attrition	SYMBOLIC	SYMBOLIC
BusinessTravel	SYMBOLIC	SYMBOLIC
DailyRate	NUMERIC	ORDINAL
Department	SYMBOLIC	SYMBOLIC
DistanceFromHome	NUMERIC	ORDINAL
Education	SYMBOLIC	SYMBOLIC
EducationField	SYMBOLIC	SYMBOLIC
EmployeeCount	NUMERIC	ORDINAL
EmployeeNumber	NUMERIC	ORDINAL
EnvironmentSatisfaction	NUMERIC	DISCRETE
Gender	SYMBOLIC	SYMBOLIC
HourlyRate	NUMERIC	ORDINAL
JobInvolvement	NUMERIC	DISCRETE
JobLevel	NUMERIC	DISCRETE
JobRole	SYMBOLIC	SYMBOLIC
JobSatisfaction	NUMERIC	DISCRETE
MaritalStatus	SYMBOLIC	SYMBOLIC

MonthlyIncome	NUMERIC	ORDINAL
MonthlyRate	NUMERIC	ORDINAL
NumCompaniesWorked	NUMERIC	ORDINAL
Over18	SYMBOLIC	SYMBOLIC
OverTime	SYMBOLIC	SYMBOLIC
PercentSalaryHike	NUMERIC	ORDINAL
PerformanceRating	NUMERIC	DISCRETE
RelationshipSatisfaction	NUMERIC	DISCRETE
StandardHours	NUMERIC	ORDINAL
StockOptionLevel	NUMERIC	DISCRETE
TotalWorkingYears	NUMERIC	ORDINAL
TrainingTimesLastYear	SYMBOLIC	SYMBOLIC
WorkLifeBalance	NUMERIC	DISCRETE
YearsAtCompany	NUMERIC	DISCRETE
YearsInCurrentRole	NUMERIC	ORDINAL
YearsSinceLastPromotion	NUMERIC	ORDINAL
YearsWithCurrManager	NUMERIC	DISCRETE

Appendix C: Gantt Chart

The following diagram illustrates a visual representation of how Team 5 intended on achieving the desired tasks as derived in the Project Plan. To support this understanding among the team, a Gantt chart was developed for visual feedback and understanding of future progress.

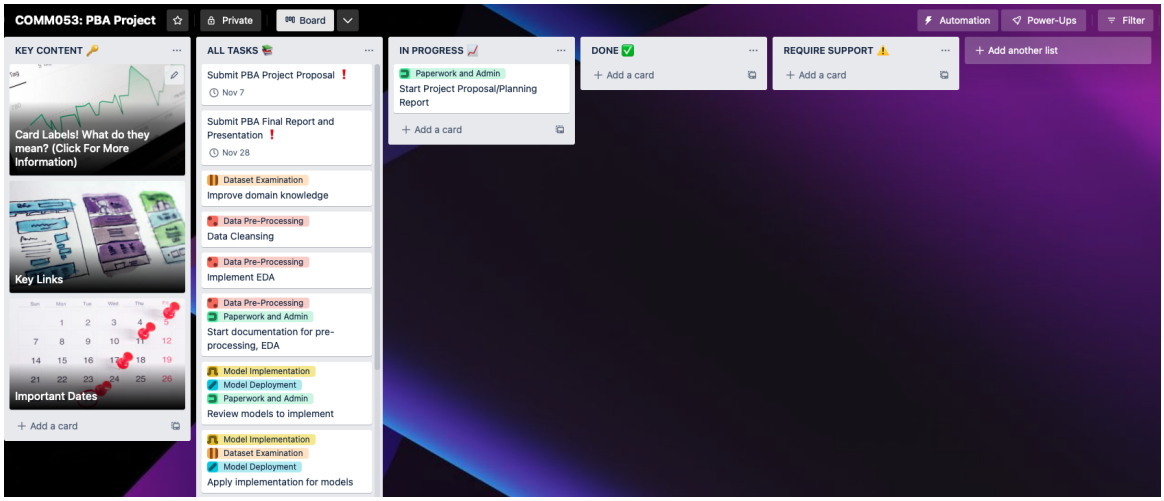


Appendix D: Trello/Kanban Board

As discussed within the project planning section of the following document, a Trello/Kanban board has been developed to support communication among the team, and to maximise performance. Below includes the shareable link to the said board.

Link:
https://trello.com/invite/b/4x4p7BgG/ATTI27a68a6089323c6c38692613d5888d121560AD11/comm053-pba-project

Visual Representation of Board:



QR Code for Access:

SCAN ME

Appendix E: Dataset Triage Pre-Categorisation

The following worksheet represents a triage attempted by the team to comprehend the data-set fields attached prior to performing EDA. This triaging was also later used for the purposes of understanding which columns require removal to proceed with modelling.

Field Names	Travel	Salary	Job Satisfaction	Experience Level	DROPPED OR RETAINED
Age					
Attrition					
BusinessTravel					
DailyRate					
Department					
DistanceFromHome					
Education					
EducationField					
EmployeeCount					
EmployeeNumber					
EnvironmentSatisfaction					
Gender					
HourlyRate					
JobInvolvement					
JobLevel					
JobRole					
JobSatisfaction					
MaritalStatus					
MonthlyIncome					
MonthlyRate					
NumCompaniesWorked					
Over18					
OverTime					
PercentSalaryHike					
PerformanceRating					
RelationshipSatisfaction					
StandardHours					
StockOptionLevel					
TotalWorkingYears					
TrainingTimesLastYear					
WorkLifeBalance					
YearsAtCompany					
YearsInCurrentRole					
YearsSinceLastPromotion					
YearsWithCurrManager					

NOTES AND LEGEND (COPY PASTE BELOW)

NOTE: Each category has been colour coded: RED: Travel, YELLOW: Salary, GREEN: Job Satisfaction Cyan: Experience Level. Please use this to make it easier for everyone :)

	Experience Level
	Job Satisfaction
	Salary
	Travel
	DROP
	REVIEW
	KEEP