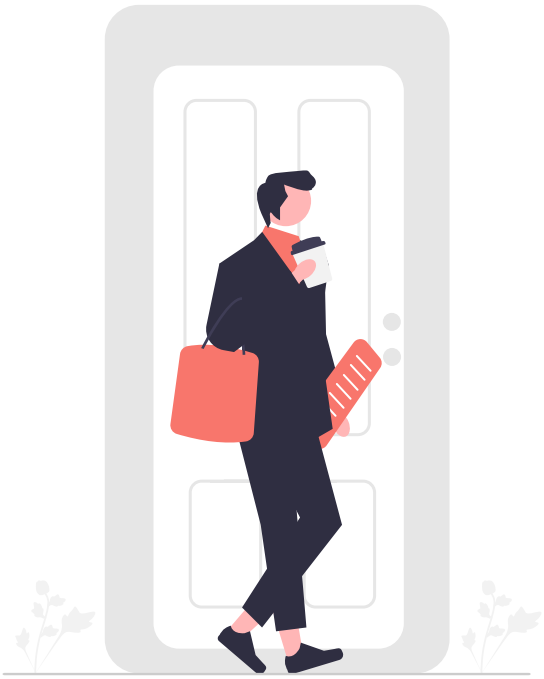
Draft



Reducing Attrition at Mini IBM

Harry Ayre, Anna Carter, Zion Ejechi, Christopher Harland, Melric Lazaro

January 2024

Contents

1 Introduction

1.1 Background

Mini IBM is a medium sized business that operate in multiple industries. They employ over 1000 people to work in different parts of the company. Mini IBM aspire to achieve growth within their company, primarily in increasing their number of employees so that they are able to work on bigger projects in the future more easily. To do this, lots of time and money has to be spent on searching for new hires, along with allocating time and resources towards other parts of the hiring process such as interviews and negotiations, which could have been spent elsewhere within the company.

Mini IBM has noticed that expenses for hiring employees are high, however they have also noticed that many newly hired employees do not stay in the company for long period of time and leave shortly for greener pastures. The company does not understand why new hires shortly leave and would like to reduce the number of employees leaving.

To measure this, Mini IBM can look at the amount of attrition they have within the business. Attrition is defined as the departure of employees from the organization for any given reason whether this is due to voluntary reasons or involuntary reasons. It is important to understand that attrition should be something that should ideally be reduced to zero in this situation.

Having a large attrition is bad for Mini IBM as it indicates that many employees are unhappy with the organisation for several reasons, this could be due to pay disputes, working conditions, relationships between staff and managers among various other reasons. Many people leaving will result in the organisation having to spend more time and money on rehiring which affects current projects along with increasing their yearly expenses. Additionally, employees leaving unhappy may result in the reputation of Mini IBM worsening and may cause further difficulties in hiring employees and will amplify costs even further.

1.2 Problem statement and objectives

1.2.1 Statement

The problem we are trying to solve is the following:

> *What combinations of factors correlate the most to employee attrition, and how can machine learning be used to help so that the business can improve upon these for their employees?*

This is important for the business as it will allow Mini IBM to identify the leading factors that cause employees to leave the organization. Identifying these factors means improvements can be made into these key areas which will result in employees staying for longer and reducing overall attrition. This will in turn reduce overall costs for business and allow them to be more productive in the long term.

By creating a machine learning model, we will also be able to predict who is most likely to leave and Mini IBM can target these individuals by negotiating with them for better contracts for example in the hope that they won't leave the business. This will create a better working relationship between the individual and the organization and can potentially improve work ethic and performance.

1.2.2 Objectives

One objective to create a good machine learning model is to follow the CRISP-DM approach for analytics. This involves a good understanding of the problem domain and what factors are important to us and what

this means for the business. Understanding the dataset provided is another key step to creating an effective model. This is where we analyse the quality of the dataset by performing data exploration to see mean, mode and median values of fields along with any existing trends between data that we can exploit. This step also involves identifying if we have any missing or redundant data along with highlighting outliers which may skew the results. Once these are identified, we can then prepare the data for modelling appropriately and then create initial models to interact with this dataset. The evaluation of the model provides key insights into how the dataset works and may provide more knowledge into other fields importance to predicting attrition that may have not been know before. This creates a feedback loop where the results of the evaluation of a previous model will be used to start the cycle again and create even better models by repeating the same steps we took before to create our initial models.

After following the CRISP-DM approach to create our models we wish to identify a series of items that relate to our problem statement.

One objective is to identify what model out of our set provides the greatest price to performance ratio for our business. We define price as in the cost to create a model from scratch using the dataset and long with running the model. This price can refer to factors such as electrical cost, quantity and quality of data required to run the model along with model complexity. We define performance in this scenario to be accuracy and other evaluation metrics to judge the model. Ideally, the best model will have the best performance for the smallest cost on Mini IBM.

To meet our initial statement, we also wish to identify what columns contribute the most to the result of attrition for the given employee. Upon identification of these factors and their importance, this gives an idea to Mini IBM on key aspects of their business that they should focus on that will improve employee attrition and will result in less wasted time and money on rehiring employees such that they can focus on their original goal of internal growth.

1.3 Dataset

The dataset that we utilise in our work is a synthetic dataset generated by IBM to simulate the conditions and lives of employees with up to 35 qualitative and quantitative characteristics for 1470 employees. The characteristics included consist of information ranging from the age of the employee, their salary, hourly rates, and department they work in, to more personal characteristics such as relationship status, business travel data and their distance away from home. Importantly, we can also see whether or not the related employee has left the company or not as stated by their `Attrition` field being either *Yes* or *No*. Due to the quantity and range of the supplied dataset, we are not required to source a secondary dataset to combine and link with the synthetic Mini IBM dataset for additional information as we already have enough data to examine employee performance, financials and how job roles affect employee attrition.

As our dataset is synthetic and was professionally generated by IBM, it may not completely reflective of real-life standards and qualities and therefore we work within the scope of the Mini IBM dataset for this reason. This may also potentially impact our findings in our evaluation as they are based on how well the synthetic data was generated.

1.3.1 Exploration

Deciding on a dataset meant analysing and assessing the quality of the explored datasets, encompassing various criteria. Firstly, we began by scrutinising the accuracy of the fields and values within the dataset. We chose a dataset sourced from IBM published in 2020, a reputable company with current and dependable information, which installed confidence in the correctness of the data. Therefore, our choice was verified in the reliability of IBM as a data source, ensuring the accuracy of the information we had at hand. Additionally, an important step of data exploration involves looking for missing data to ensure the dataset is complete as lots of missing data can lead to inaccurate results. As we do not have missing data and most of the columns supplied provide meaningful insights to us, we consider that the quality of the dataset is good enough to be used directly without crucial additions. We define quality being good as the dataset having enough columns

and rows to be able to create insights and core understanding of Mini IBM that can be related back to the problem statement and can be used in a machine learning model.

While we do not have any missing data, we do have redundant columns. We manually analysed the relevance of individual fields within the dataset. While we have 35 columns, not all of these may be of good quality for use in a machine learning model and we wanted to identify these fields so we can prune them and reduce the overall dimensionality of our data so our models can work with the data better. Out of the 35, we decided that there were at least 12 columns that were considered redundant. We consider these columns as ones that contain qualities such as being too vague to use, not being directly related to the problem, overly subjective along with columns that do not provide any insight to us (see inclusion criteria for more). For example. having only one unique row in the column is useless as we cannot learn from it. Furthermore, not enough unique rows and columns can lead to overfitting. Examples of such columns include the `Over18` field where the only value in each row is Yes, `Standard Hours` where every row is 80, `Employee Count` where every row is 1 or `Performance Rating` where the data is not diverse enough to get meaningful insights. We removed these fields from the dataset in the `cleanData()` function, shown below. This assessment led us to identify 23 relevant fields, forming approximately 66% of the entire dataset. Focusing on these variables enabled us to narrow our exploration to fields crucial for our analysis objectives.

`cleanData()` function in `dataPrep.R`

```r
cleanData<-function(dataset, remove = list()){
  print("cleaning data")
  cleanedData <- dataset
  markedColumns <- list()


  #find the columns from the given list to remove
  for(field in remove){
    print(paste("removing ", field))
    markedColumns <- append(markedColumns, field)
  }

  # find columns that have the same value in every row
  #For every field in our dataset
  for(field in 1:(ncol(dataset))){
    # Convert into factors. A level for each unique string
    ffield<-factor(dataset[,field])
    # Check if just one value!

    # nlevels returns the number of unique rows in the column
    # If there is only one unique row in the column it is useless as we cannot learn from it
    if (nlevels(ffield) ==1) {
      #mark column for removal
      markedColumns <- append(markedColumns, names(dataset[field]))
      print(paste("removing ", names(dataset[field])))
    }
  }


  #remove columns
  cleanedData <- dataset[,!(names(dataset) %in% markedColumns)]

  #identify numeric columns for removeRedundantFields
  numericColumns <- sapply(cleanedData, is.numeric)

  #apply removeRedundantFields only to numeric columns
  if (any(numericColumns)) {
    cleanedData[, numericColumns] <- removeRedundantFields(cleanedData[, numericColumns], 0.95)
    print("removed redundant numeric fields")
  } else {
```

```
    print("no numeric columns for removeRedundantFields")
  }



  print("removed redundant fields")
  print("cleaned data")
  return (cleanedData)
}
```

A further step involved us investigating the presence of duplicate rows within the dataset. Calculating only (?) duplicate rows allowed us to retain (?)% of the dataset which provided a solid foundation for analysis. Additionally, we create a `prettyDataset()` function to explore the main characteristics of the relevant fields by performing statistical analysis, computing the mean, median, and mode below.

`prettyDataset()` function in `dataPrep.R`

```
prettyDataset<-function(dataset,...){
  params <- list(...)

  tidyTable<-data.frame(Field=names(dataset),
                        Categorical=FALSE,
                        Symbols=0,
                        Name=0,
                        Min=0.0,
                        Mean=0.0,
                        Max=0.0,
                        Skew=0.0,
                        stringsAsFactors = FALSE)

  if (length(params)>0){
    names(tidyTable)[1]<-params[1]
  }

  for (i in 1:ncol(dataset)){
    isFieldAfactor<-!is.numeric(dataset[,i])
    tidyTable$Categorical[i]<-isFieldAfactor
    if (isFieldAfactor){
      tidyTable$Symbols[i]<-length(unique(dataset[,i]))  #Number of symbols in categorical
      #Gets the count of each unique symbol
      symbolTable<-sapply(unique(dataset[,i]),function(x) length(which(dataset[,i]==x)))
      majoritySymbolPC<-round((sort(symbolTable,decreasing = TRUE)[1]/nrow(dataset))*100,digits=0)
      tidyTable$Name[i]<-paste(names(majoritySymbolPC),"(",majoritySymbolPC,"%)",sep="")
    } else
    {
      tidyTable$Max[i]<-round(max(dataset[,i]),2)
      tidyTable$Mean[i]<-round(mean(dataset[,i]),2)
      tidyTable$Min[i]<-round(min(dataset[,i]),2)
      tidyTable$Skew[i]<-round(PerformanceAnalytics::skewness(dataset[,i],method="moment"),2)
      # add Median and Mode
      tidyTable$Median[i] <- round(median(dataset[, i]), 2)

      get_mode <- function(v) {
        uniqv <- unique(v)
        uniqv[which.max(tabulate(match(v, uniqv)))]
      }
      tidyTable$Mode[i] <- get_mode(dataset[, i])
    }
  }

  #Sort table so that all numerics are first
```

```r
  t<-formattable::formattable(tidyTable[order(tidyTable$Categorical),],
                              list(Categorical = formatter("span",style = x ~ style(color =
                              ↪  ifelse(x,"green", "red")),
                                                           x ~ icontext(ifelse(x, "ok", "remove"), ifelse(x,
                                                             ↪  "Yes", "No"))),
                                   Symbols = formatter("span",style = x ~ style(color = "black"),x ~
                                   ↪  ifelse(x==0,"-",sprintf("%d", x))),
                                   Min = formatter("span",style = x ~ style(color = "black"), ~
                                   ↪  ifelse(Categorical,"-",format(Min, nsmall=2, big.mark=","))),
                                   Mean = formatter("span",style = x ~ style(color = "black"),~
                                   ↪  ifelse(Categorical,"-",format(Mean, nsmall=2, big.mark=","))),
                                   Max = formatter("span",style = x ~ style(color = "black"), ~
                                   ↪  ifelse(Categorical,"-",format(Max, nsmall=2, big.mark=","))),
                                   Skew = formatter("span",style = x ~ style(color = "black"),~
                                   ↪  ifelse(Categorical,"-",sprintf("%.2f", Skew)))
                              ))
  print(t)
}
```

Job satisfaction This variable is related to employee sentiment and so is a crucial indicator for predicting attrition. The mean job satisfaction score is 2.73 which suggests a moderate level of satisfaction among respondents. The median of 3 indicates a central tendency towards satisfaction, with the mode being 4. This suggests a significant number of respondents reporting relatively high job satisfaction.

Work life balance On average, employees rated their work-life balance at 2.76, indicating that it falls below a satisfactory level. The median of 3 suggests variations in responses, while the mode of 3 emphasizes a substantial number of respondents specifically rating their work-life balance as average.

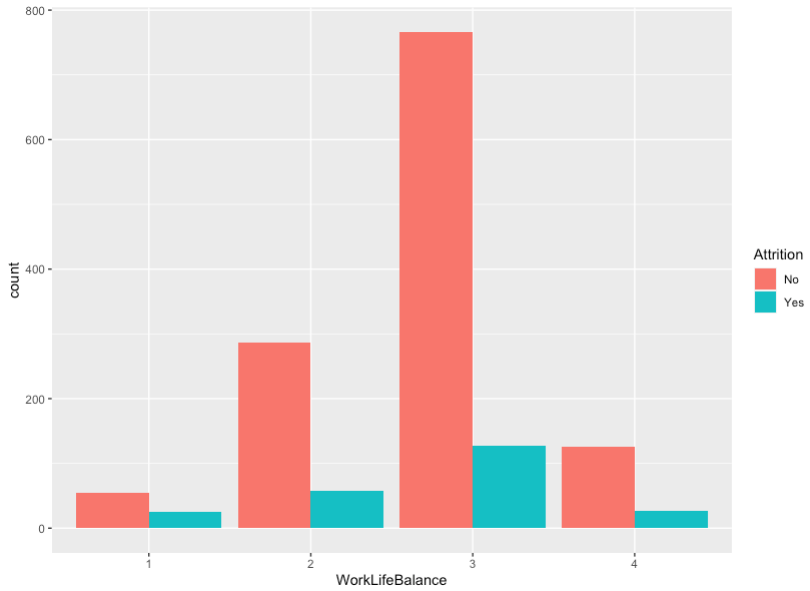


Figure 1.1: Work-life balance vs attrition

Years in current role Employees in the dataset have an average period in their current roles of 4.23 years, with a median of 3, suggesting the distribution is right skewed. The mode of 2 indicates a there is a higher concentration of employees with a 2-year tenure which suggests the most common duration time for employees in their role.

Years at company The mean duration at the company is 7.01 years, which is a relatively long-term commitment from employees. A median of 5, highlights the distribution is slightly right skewed which indicates there is a portion of employees with shorter durations. The mode of 5 reinforces the fact that there is a greater number of employees with a 5-year tenure.

plotOutliers() function in dataPrep.R

```
# function to plot a scatter plot of field values and colour the outliers in red
# inputs:
# sorted - vector - points to plot as literal values
# outliers - vector - list of above points that are considered outliers
# fieldName - string - name of field to plot
plotOutliers<-function(sorted,outliers,fieldName){

  plot(1:length(sorted),sorted,pch=1,xlab="Unique records",ylab=paste("Sorted values",fieldName),bty="n")
  if (length(outliers)>0)
    points(outliers,sorted[outliers],col="red",pch=19)
}
```

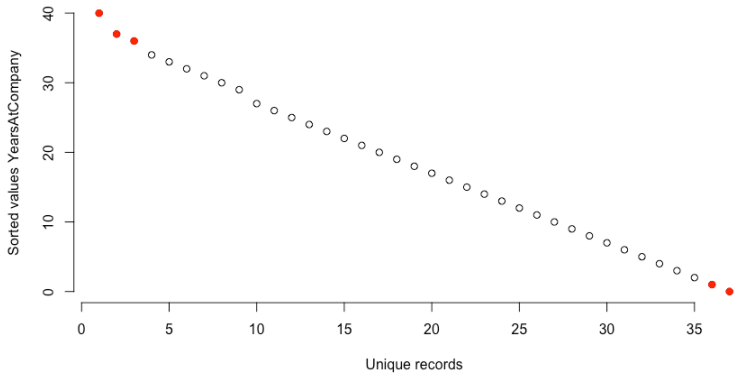


Figure 1.2: Outliers

Gender The dataset shows a balanced gender distribution, with 40% female and 60% male employees. This balanced representation may positively impact diversity and inclusivity initiatives when we use this dataset in our model evaluations.

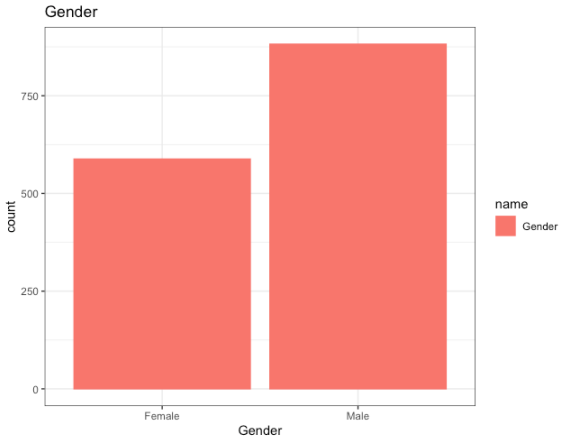


Figure 1.3: Gender distribution in our dataset

Overtime A significant majority of employees, 72%, responded favourably to working overtime which indicates a satisfactory environment within the workforce. This insight may also have implications for factors such as job satisfaction and work-life balance.

For Mini IBM to understand their business problem and how they it affects them, we also needed to see the current levels of attrition within the company. We did this by creating a simple bar chart to identify the amount of *Yes* and *No* for `Attrition` we have in our current dataset.

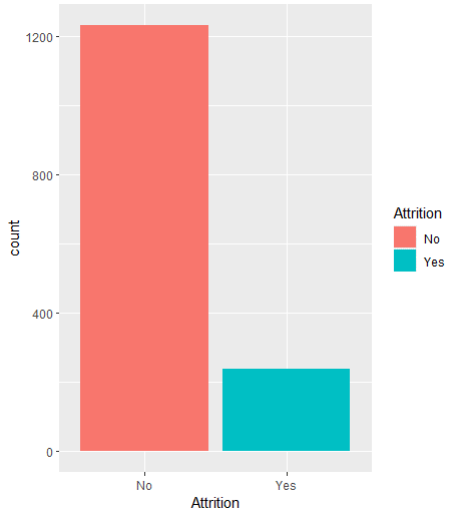


Figure 1.4:

Our observation of this shows that we have a much greater amount of people staying within the company compared to people leaving. While this seems like a good thing at first, we must remember that attrition is the exception and not the norm and for Mini IBM to achieve their target for growth, they must try to reduce this number further.

Interestingly, we can see a large class imbalance between the two attrition types. As we have many more people who have not left the company, subsequent models may be biased towards learning towards this class and will suffer more on attempting to predict the positive class. This means we should implement some sort of class balancing to allow the model to see an equal amount of data while training, giving the model a better chance at predicting attrition more accurately. This would give Mini IBM a better chance at identifying employees who are at risk of leaving.

We also began the data exploration process by plotting each field against attrition. Three plots were produced for each field.

Using these plots, we can identify any trends between the chosen variable in association with attrition. This makes it easier to make an educated guess to see what variables are directly related to changing the outcome of attrition and we can put a greater emphasis on these variables.

We can see that there are a few graphs that indicate trends between variables, such as…

From our dataset, we can also see that the average age for employees who have left is 33.6, indicating that people who are more likely to leave tend to be younger and do not have as much experience as seen with the average companies worked at previously column which also is lower. Additionally, we can also see that employees who have left had longer commutes to work according to the mean distance field and do not tend to be paid as much other employees to offset these values.

1.3.2 Processing

To process our data within R, there were a number of steps we needed to take. This involves taking the original CSV file containing our records and putting them into a dataframe. This allows us to perform R functions on our data such as generating plots easily for each column against attrition.

As we convert our CSV file to the dataframe in R, we must also correctly set the field types for our data as this will affect the training of our models in the future. To determine if a column is discrete, continuous, symbolic or ordered categorical we use a combination of manually going through the data and setting them appropriate, with more robust and mechanical methods such as x.

We also decided to derive new fields that can create more insights through the combination of existing columns. One example of this involves the age when an employee joined Mini IBM. This helps to identify if attrition is related to the age of the employee when they first started working here and can point out individuals who appeared to be loyal and show growth while they have been at Mini IBM compared to those who are older and have more experience and subsequently left because of it. It also helps to identify what age Mini IBM should target for hiring employees if they want to reduce attrition.

1.4 Models

We used a variety of machine learning models to predict the correct values for attrition be aid in our identifying if an employee is at risk of leaving along with identifying what columns contribute the most to this.

Considering that our objective is to identify possible attrition for employees in our dataset, we are working with a binary classification problem using a supervised method as we have ground truth labels to influence the models. This means that regression models such as linear and logistic regression are not as suitable for our tasks and therefore, we must focus on creating and tuning models that are more suitable for binary classification.

1.4.1 Model design

To meet our objectives, we focus on utilising six different models. Our core 5 models consist of:

- Logistic Regression
- Decision Tree
- Random Forest
- Naïve Bayes
- Neural Network

These 5 models are used to emphasise how selecting the correct machine learning model affects predictions of attrition and its accuracy, which is important for Mini IBM to successfully reduce attrition. Each model also has a complexity and cost associated with it, providing insights to how more complex models affect performance against predicting a testing dataset.

Our sixth model involves using a support vector machine (SVM) with different compositions of the original dataset to emphasise how the dataset the model is trained on affects its hyperparameters and as a result, how it affects overall performance of the model.

1.4.2 Measuring performance

To ensure that the results from each method are as fair as possible, a holdout value of 70% is used. Performance will be measured using MCC, defined as

$$MCC = \frac{TN \times TP - FN \times FP}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{1.1}$$

2 Data preprocessing and cleaning

2.1 Inclusion criteria

Our inclusion criteria are outlined below:

IC1. Fields must be something that the business can directly improve on.

IC2. Fields must not be subjective.

IC3. Fields must be clearly defined.

IC4. Fields must be something we can learn from. Also need to be heterogenous

This led us to exclude the fields listed below from our dataset. The inclusion criteria violated for each field are identified, with additional comments alongside.

Table 2.1: Fields that violate the inclusion criteria.

Field	IC1	IC2	IC3	IC4	Comments
JobInvolvement		✔	✔		
Over18				✔	One unique value (*Y*). Additionally, this field can be easily derived.
EmployeeCount				✔	One unique value (1)
EmployeeNumber				✔	Unique identifier for an employee
PerformanceRating				✔	All employees are between 3 and 4, meaning good and very good. Need 1s and 2s for the field to be useful. (non-heterogeneous)
PercentSalaryHike			✔		
RelationshipSatisfaction	✔				
StandardHours	✔			✔	
YearsWithCurrManager			✔		There could be several reasons for a change in manager.
MonthlyIncome				✔	
MonthlyRate				✔	
DailyRate				✔	

2.2 Steps

2.2.1 Removal of outliers

We needed to remove any outliers from our dataset which was done with our `removeOutliers` function[1]. For each continuous field, we order it then outliers are identified based on a confidence value. If the confidence is positive then the outliers are replaced with the mean, and if its negative they are not replaced, just identified.

[1] `dataPrep.R`, line 392

2.2.2 Field transformation

2.2.3 Dimensionality reduction

In order to reduce the number of columns in our dataset we manually set certain columns to be considered as continuous. This meant that columns that would have otherwise been 1-hot-encoded into many more columns (increasing dimensionality) remain as just a single column. This makes it easier for models to learn patterns in the data. We did this using our `getFieldTypes` function[2]. This function lets our override which columns should be considered as what type.

2.2.4 Derivation of new fields – age and years at company

To further help with dimensionality reduction and to remove redundant data in place of more useful information, we derived and combined new columns from already existing columns in the dataset. We did this using our `combineOrDeriveFields` function[3]. This function lets you specify to columns in the dataset, a function to apply to the two columns to create the new column and then a boolean to specify whether you want to remove the original columns or keep them. We used this to first divide (and remove) `YearsSinceLastPromotion` from `YearsWithCurrManager` to create a new combined column `PerformanceWithCurrentManager`. This gives an insite into the impact a manager had on an employes performance, which might help to see if employees are leaving due to an employer. We skip this calculation for any employees that have been with their manager less than a year because its not enough time to evaluate the performance. Their value will be NA in the new column but obviously we don't want NA values in our dataset, this causes problems down the line, so we replaced these with the mean once all performances with current managers had finished (see figures (?) and (?) and (?)).

We then took away `YearsAtCompany` from `Age` to create a derived column `AgeJoined`, keeping the original two columns. We kept the original columns in this case because there are multiple combinations of `YearsAtCompany` and `Age` that can lead to the same `AgeJoined` value, and so having all three columns helps distinguish between them whilst providing more information than before. Having this column shows if employees who started earlier stayed at the company which may suggest loyalty and satisfaction to/with the company.

`divide` function in `main.R`

```
253 # formula to divide two columns and obtain a ratio
254 # to understand the average time between promotions relative to the time spent with the current manager.
255 # this ratio could provide insights into the frequency of promotions in relation to the duration of the
256 # current managerial relationship. For example, a higher ratio might suggest that employees tend to receive
257 # promotions more frequently in comparison to the time they spend with their current manager.
258 divide <- function(colName1, colName2, dataframe) {
259   results <- colName1 / colName2
260   results[is.infinite(results) | is.nan(results)] <- 0
261   return(results)
262 }
```

`CombineOrDeriveFields` function in `dataPrep.R`

```
678 combineOrDeriveFields <- function(dataframe, colName1, colName2, fun, newColName, combine = FALSE) {
679
680   column1 <- dataframe[[colName1]]
681   column2 <- dataframe[[colName2]]
682
683   # apply the function to create the new column
```

[2] dataPrep.R, line 88
[3] dataPrep.R, line 670

```
  newColumn <- fun(column1, column2)

  # add the new column to the dataframe
  dataframe[[newColName]] <- newColumn

  # if (combine) {
  #   print("Deleting old columns...")
  #   #dataset <- subset(dataset, select = -c(column1, column2))
  #   dataset <- select(dataset, -c(column1, column2))
  # }

  if (combine) {
    print("Deleting...")
    #dataframe <- subset(dataframe, select = -c(column1, column2))
    dataframe <- dataframe[, -which(names(dataframe) %in% c(colName1, colName2))]
  }

  return(dataframe)
}
```

2.2.5 Field encoding

Field encoding is done depending on the type of the field in the dataset. For continuous data it has its outliers removed and then, it is z-scaled followed by scaling into the range 0–1 using our `rescaleDataFrame` function[4]. Scaling is extremely useful and practically a necessity for most models such as neural networks, but it does have its drawbacks. The first is that it's difficult to effectively scale a value that is un-bounded into a finite range. The second is that once scaled, the distances between the values can sometimes not be representative of the original data which can mislead models.

To encode un-ordered categorical data we used 1-hot-encoding. We do this using our `oneHotEncode` function[5]. For each un-ordered categorical field, we convert it into a series of columns where each column represents one possible value for that field, and then the values for each column are set to 0, and 1 for the column representing the original value. Doing this represents categorical data in a numerical format allowing the models to learn from it without implying any kind of order.

Finally, to encode ordered categorical data we use our `encodeOrderedCategorical` function[6]. This function, for each ordered categorical field, factors the values to give each unique value a new value starting from 1 and increasing. So if originally a field had two unique values 10 and 20, it gets factored to 1 and 2. Then for each of these factors, it divides by the total number of factors. So you would end up with 1/2 and 2/2. This scales each of the values to the range 0-1, and maintains the ordering to allow the models to learn from the order.

2.2.6 Normalisation

2.2.7 Class balancing

We rebalance the training data using our `rebalance` function[7]. This function can use, undersampling, oversampling or both to try and balance a dataset, in our case we used both methods. To do oversampling and undersampling, the `rebalance` function uses the `ovun.sample` function which can randomly replicate data observations to create synthetic data.

[4] dataPrep.R, line 32
[5] dataPrep.R, line 171
[6] dataPrep.R, line 220
[7] dataPrep.R, line 714

2.3 Data dictionary

This data dictionary contains the final list of fields used in our models and is based on descriptions provided with the dataset.

Table 2.2: Data dictionary containing the fields used in our models.

Field	Description	Type
Age	The age of the employee.	CONTINUOUS
Attrition	Whether or not the employee has left the organisation.	SYMBOLIC
BusinessTravel	The frequency of business travel for the employee.	ORDERED_CATEGORICAL
Department	The department the employee works in.	SYMBOLIC
DistanceFromHome	The distance from home in miles for the employee.	CONTINUOUS
Education	The level of education achieved by the employee.	ORDERED_CATEGORICAL
EducationField	The field of study for the employee's education.	SYMBOLIC
EnvironmentSatisfaction	The employee's satisfaction with their work environment.	ORDERED_CATEGORICAL
Gender	The gender of the employee.	SYMBOLIC
HourlyRate	The hourly rate of pay for the employee.	CONTINUOUS
JobLevel	The job level of the employee.	ORDERED_CATEGORICAL
JobRole	The role of the employee in the organisation.	SYMBOLIC
JobSatisfaction	The employee's satisfaction with their job.	ORDERED_CATEGORICAL
NumCompaniesWorked	The number of companies the employee has worked for.	DISCRETE
OverTime	Whether or not the employee works overtime.	SYMBOLIC
StockOptionLevel	The stock option level of the employee.	DISCRETE
TotalWorkingYears	The total number of years the employee has worked.	DISCRETE
TrainingTimesLastYear	The number of times the employee was taken for training in the last year.	DISCRETE
WorkLifeBalance	The number of times the employee was taken for training in the last year.	ORDERED_CATEGORICAL
YearsAtCompany	The number of years the employee has been with the company.	DISCRETE
YearsInCurrentRole	The number of years the employee has been in their current role.	DISCRETE
YearsSinceLastPromotion	The number of years since the employee's last promotion.	DISCRETE

3 Modelling

3.1 Anna

3.2 Chris

3.3 Harry

3.4 Melric

3.5 Zion

4 Results

4.1 Anna

4.2 Chris

4.3 Harry

4.4 Melric

4.5 Zion

5 Discussion

6 Author contribution statement

6.1 Anna

6.2 Chris

6.3 Harry

6.4 Melric

6.5 Zion