IBM HR Analytics Employee Attrition - Capstone

Martinez Maldonado Matías

Last compiled on marzo 19, 2023

Contents

1 Introduction

High performing employees are the investment of companies in their success and growth in future. Therefore it's important for businesses to find out about factors that are contributing more to employee attrition.

In this report, we'll try to find out which employees tend to leave the company, what factors are the stronger contributors to this behavior, which departments face the attrition problem and what type of measures could the company take in order to retain their employees.

The purpose of this project is to investigate the employee attrition of a company and predicting the potential employees who would leave the company. We will be working with a dataset extracted from Kaggle called "IBM HR Analytics Employee Attrition & Performance".

To do this all, first we'll do some EDA (Explanatory Data Analysis) to get some intuition about the data. We then try to build different models with different techniques and compare their scores with different evaluation metrics. We finally fine tune our model in the best performing technique.

2 Exploratory Data Analysis

We start the exploratory data analysis (EDA) to discover some patterns and correlations in data that may notice us why the employees stay in the company or leaves it.

First of all, we take a look at the dataset structure and make a statistical summary of the features that are into it.

2.1 Data Structure

After *str(employee)*, we know that features in the dataset are:

- **Age** `<integer>` that contains the age for each employee.
- **Attrition** `<character>` that contains if the employee stay at the company or leaves it *(target feature)*.
- **BussinesTravel** `<character>` that contains the travel frequency for each employee.
- **DailyRate** `<integer>` that contains the daily rate.
- **Department** `<character>` that contains the department for each employee.
- **DistanceFromHome** `<integer>` that contains the distance from home for each employee.
- **Education** `<integer>` that contains the education grade for each employee.
- **EducationField** `<character>` that contains the education field for each employee.
- **EmployeeCount** `<integer>`
- **EmployeeNumber** `<integer>` that contains the id number for each employee.
- **EnvironmentSatisfaction** `<integer>` that contains the environment satisfaction for each employee.
- **Gender** `<character>` that contains if the employee is male or female.
- **HourlyRate** `<integer>` that contains the hourly rate for each employee.
- **JobInvolvement** `<integer>` that contains the job involvement for each employee.
- **JobLevel** `<integer>` that contains the job level for each employee.
- **JobRole** `<character>` that contains the job role for each employee.
- **JobSatisfaction** `<integer>` that contains the job satisfaction for each employee.
- **MaritalStatus** `<character>` that contains if the employee is married, single or divorced.
- **MonthlyIncome** `<integer>` that contains the monthly income for each employee.
- **MonthlyRate** `<integer>` that contains the monthly rate for each employee.
- **NumCompaniesWorked** `<character>` that contains the number of companies in which an employee works.
- **Over18** `<character>` that contains if the employee is eighteen years or more.
- **OverTime** `<character>` that contains if the employee do overtime hours.
- **PercentSalaryHike** `<integer>` that contains if the employee.
- **PerformanceRating** `<integer>` that contains the performance rating for each employee.
- **RelationshipSatisfaction** `<integer>` that contains the satisfaction rating on employee relations at the company.
- **StandardHours** `<integer>` that contains the standard hours for each employee.
- **StockOptionLevel** `<integer>` that contains the level of some employees that are paid with company stocks. (0 if not)
- **TotalWorkingYears** `<integer>` that contains the amount of years each employee worked.
- **TrainingTimesLastYear** `<integer>` that contains the number of times the employee is trained.
- **WorkLifeBalance** `<integer>` that contains the work-life balance for each employee.

- **YearsAtCompany** `<integer>` that contains the number of years the employee worked at the company.
- **YearsInCurrentRole** `<integer>` that contains the number of years in current role for each employee.
- **YearsSinceLastPromotion** `<integer>` that contains the number of years since the last promotion for each employee.
- **YearsWithCurrManager** `<integer>` that contains the number of years the employee worked with the same manager (current manager).

2.2 Missing Values

Continuing with the data exploration we need to know if the dataset has missing values and how many values different are in each feature. In case of missing data we have to find the best approach to fill it.

Table 1: Missing Values

	x
Age	0
Attrition	0
BusinessTravel	0
DailyRate	0
Department	0
DistanceFromHome	0
Education	0
EducationField	0
EmployeeCount	0
EmployeeNumber	0
EnvironmentSatisfaction	0
Gender	0
HourlyRate	0
JobInvolvement	0
JobLevel	0
JobRole	0
JobSatisfaction	0
MaritalStatus	0
MonthlyIncome	0
MonthlyRate	0
NumCompaniesWorked	0
Over18	0
OverTime	0
PercentSalaryHike	0
PerformanceRating	0
RelationshipSatisfaction	0
StandardHours	0
StockOptionLevel	0
TotalWorkingYears	0
TrainingTimesLastYear	0
WorkLifeBalance	0
YearsAtCompany	0
YearsInCurrentRole	0
YearsSinceLastPromotion	0
YearsWithCurrManager	0

We are lucky, we don't have missing values! So let's continue.

Table 2: Number of Unique Values

	x
Age	43
Attrition	2
BusinessTravel	3
DailyRate	886
Department	3
DistanceFromHome	29
Education	5
EducationField	6
EmployeeCount	1
EmployeeNumber	1470
EnvironmentSatisfaction	4
Gender	2
HourlyRate	71
JobInvolvement	4
JobLevel	5
JobRole	9
JobSatisfaction	4
MaritalStatus	3
MonthlyIncome	1349
MonthlyRate	1427
NumCompaniesWorked	10
Over18	1
OverTime	2
PercentSalaryHike	15
PerformanceRating	2
RelationshipSatisfaction	4
StandardHours	1
StockOptionLevel	4
TotalWorkingYears	40
TrainingTimesLastYear	7
WorkLifeBalance	4
YearsAtCompany	37
YearsInCurrentRole	19
YearsSinceLastPromotion	16
YearsWithCurrManager	18

The following columns are "no use" variables; they does not add values to the predictive model that I am going to build:

1. EmployeeNumber (1470 unique values)
2. EmployeeCount (only 1 unique value)
3. Over18 (only 1 unique value)
4. StandardHours (only 1 unique value)

Our first plot will be a global map on all the variables, to take reference on the characteristics of the variables and their distributions. Let's start!

2.3 Histogram plots

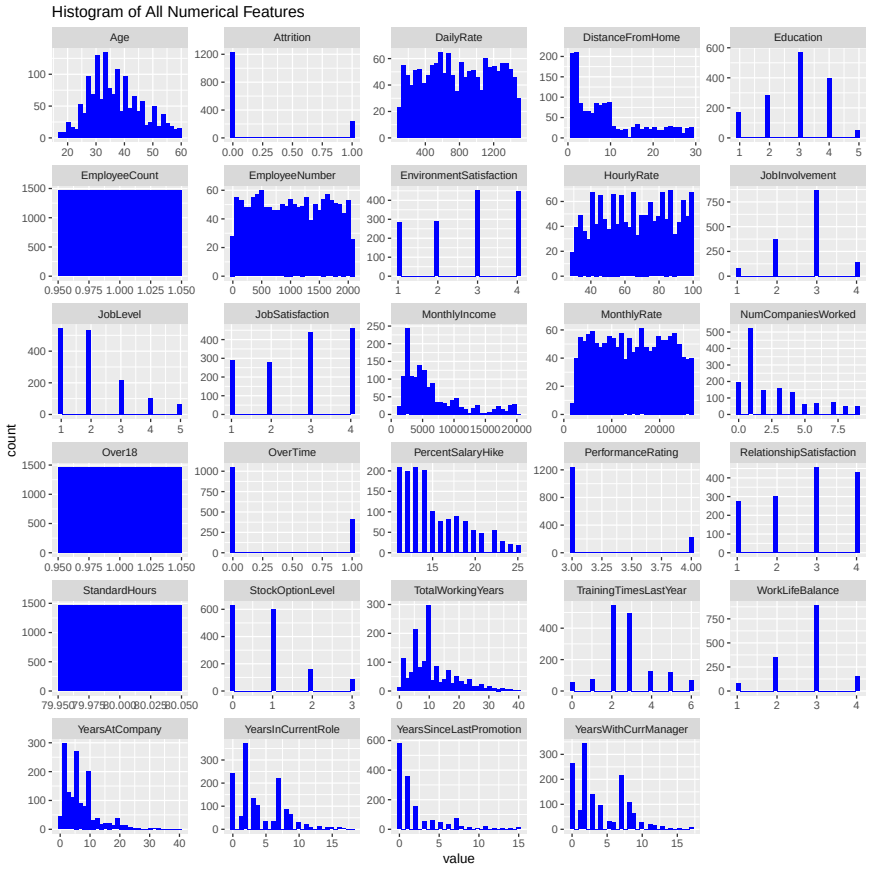

Figure 1: Histogram of All Numerical Features

In the previous plot we realize that the majority of the employees are in the range of between 20 and 30 years. We can realize by looking at the Attrition graph that the dataset is unbalanced, around 83% of the data belongs to employees who stayed in the company (the remaining 17% left it).

We can also see four long-tail distribution plots: the first in DistanceFromHome, MonthlyIncome, PercentSalaryHike and lastly YearsSinceLastPromotion. In "long-tailed" distributions a high-frequency or high-amplitude population is followed by a low-frequency or low-amplitude population which gradually "tails off" asymptotically. The events at the far end of the tail have a very low probability of occurrence. ("Long Tail," n.d.).

In the case of MonthlyIncome we notice some quite predictable, the most employees salary are lower than u$s5000 and as the salary is higher, there are fewer employee.

After this plot, we confirm what we previously stated about the following "no use" columns.

Hence we have to drop those features (remembering: EmployeeNumber, EmployeeCount, Over18 and StandardHours).

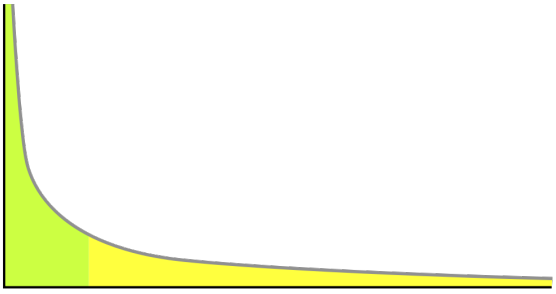

Figure 2: Long-Tail Image. Source: Wikipedia

2.4 Correlation Matrix

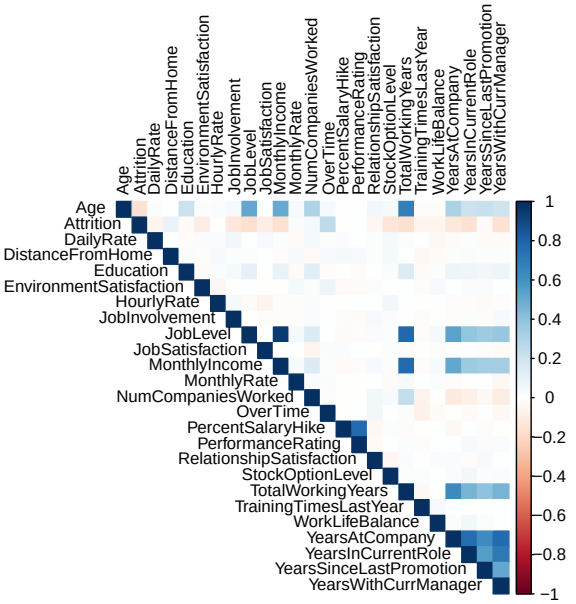

Figure 3: Correlation Matrix

Plot Summary:

1. Age are positively correlated with JobLavel, MonthlyIncome and TotalWorkingYears.
2. JobLevel are positively correlated with Monthly Income, TotalWorkingYears and YearsAtCompany.
3. MonthyIncome are positively correlated with JobLevel, TotalWorkingYears and YearsAtCompany.
4. PrecentSalaryHike are positively correlated with PerformanceRating.
5. The last four features in the dataset (YearsAtCompany, YearsInCurrentRole, YearsSinceLastPromotion and YearsWithCurrManager) are positively correlated with each other.

We could consider that if the employee is older than others, they will have a better JobLevel, in turn, due to the high correlation, a higher salary. Could it be that this employee has a better chance

of staying in the company?.

As well, if the employee have better performance in his job the percent salary hike is higher.

2.5 Attrition

As we see in this plot, between 18 and 21 years the employee is most likely to leave the company. But according as the ages are increasing, the probability what employee left the company reduces substantially. The attrition peek is around 28 and 32 years.

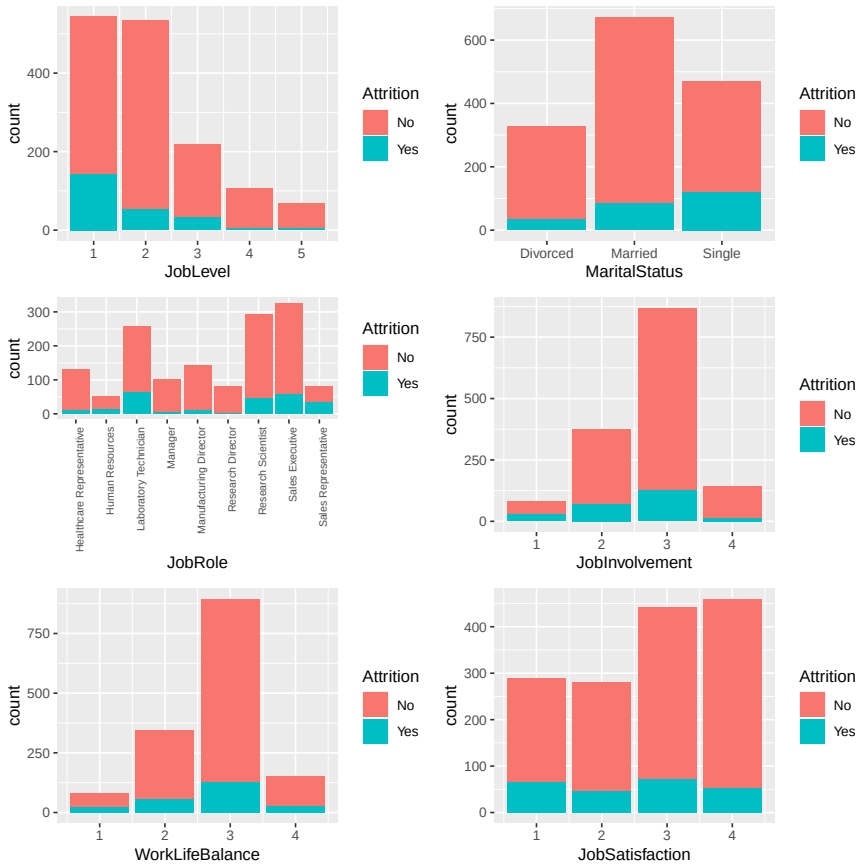

Figure 4: Some Features vs Attrition

From this plot we could consider:

1. When JobLevel is low (1, 2 or 3) the employee is most likely to leave the company.
2. A single employee is most likely to leave the company.
3. Sales Representative employees are most likely to leave the company compared to other roles. Laboratory Technician are also an important ratio of attrition.
4. When the employee are more involved in his job is less likely to leave the company.

2.6 Density Plots

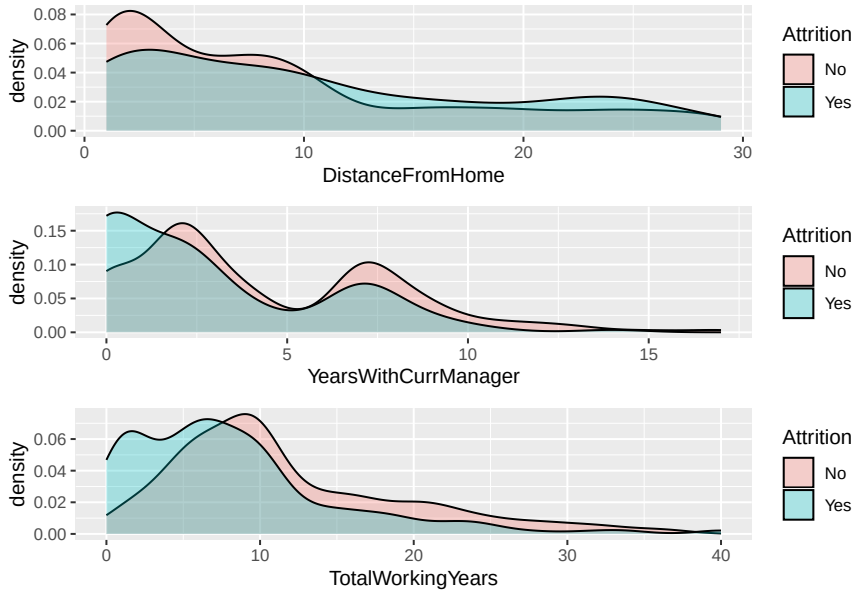

Figure 5: DistanceFromHome, YearsWithCurrManager and TotalWorkingYears vs Attrition (Densities)

Here, we see that if the distance between the employee's work and home is greater, it is more likely that he will leave the company.

We also see that if the employee has fewer years in the world of work, he is more likely to leave the company.

2.7 MonthlyIncome

2.7.1 MonthlyIncome vs Gender

Before we finalizing the Exploration Data Analysis, we need to analyse the controversial relationship between gender and monthly incomes. For this purpose, we make a boxplot. Let's see!

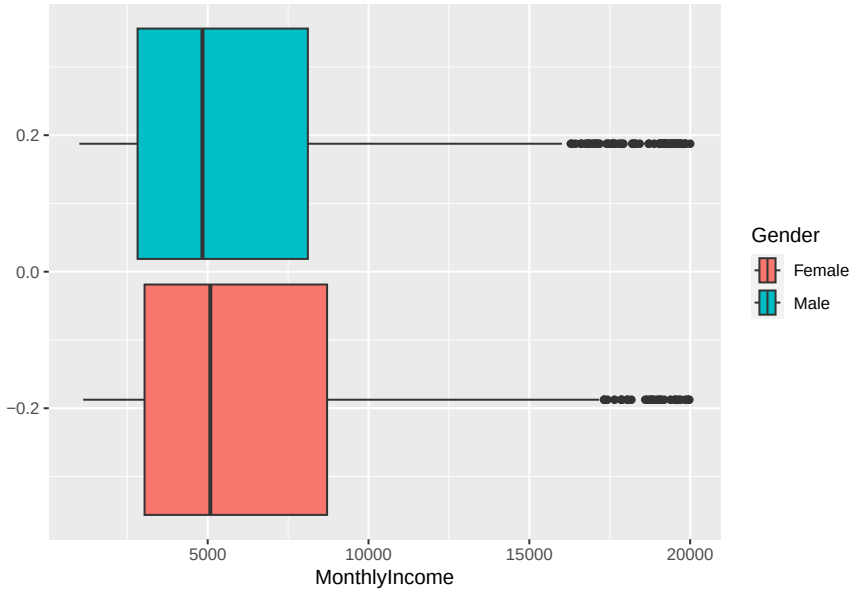

Figure 6: MonthlyIncome vs Gender

Curiosity in this company females 1st(25% of data), 2nd(median) and 3th(75% of data) quartiles are greater than males. In this company, there isn't discrimination by gender based on salary because the range of salaries are pretty close between both gender.

2.7.2 MonthlyIncome vs JobRole

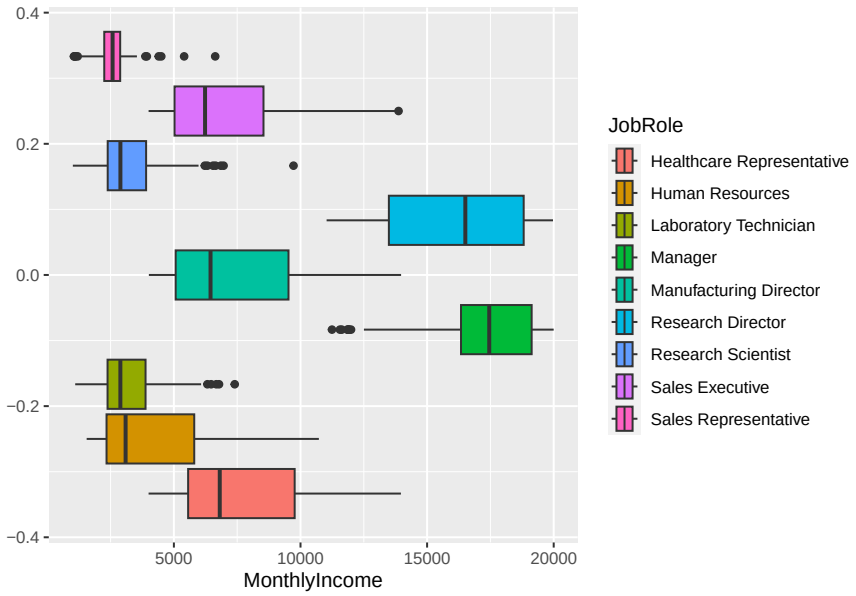

Figure 7: MonthlyIncome vs JobRole

Hence this plot, we could say that the best salaries are for Managers and Research Directors.

The worst salaries are for Sales Representative, Research Scientist and Laboratory Technician. Maybe, as we see earlier, the salary is the reason for high attrition in this roles.

2.7.3 MonthlyIncome vs Overtime grouped by Attrition

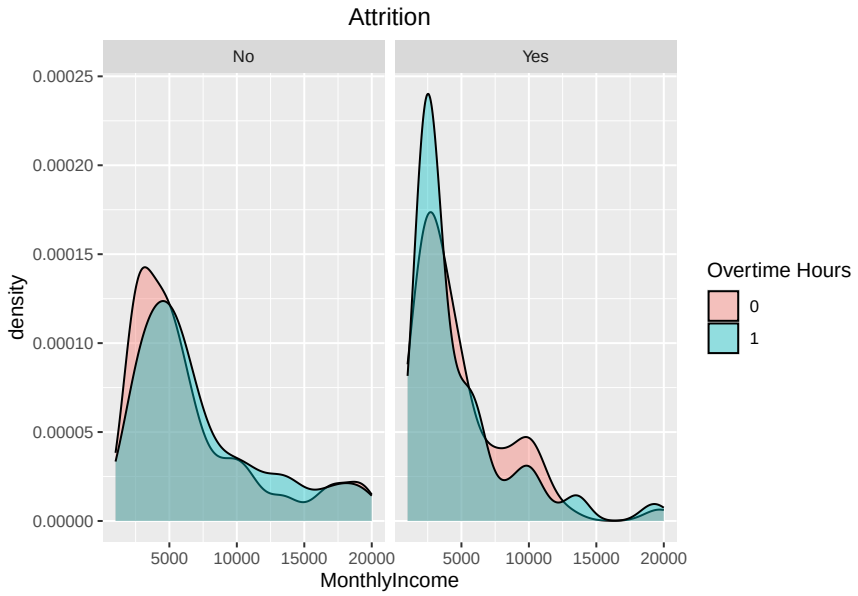

Figure 8: MonthlyIncome vs Overtime grouped by Attrition

In this case the employees who work overtime are not accompanied with higher remuneration. if we see the box of attrition, we can see that people who work overtime are the least paid and leave the company on top of that.

Here we can clearly see an effort-reward imbalance. The company should change its overtime policy.

3 Pre-Processing Data

The performance of a machine learning model not only depends on the model and the hyperparameters but also on how we process and feed different types of variables to the model. Since most machine learning models only accept numerical variables, preprocessing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model is able to understand and extract valuable information.

Identified categorical variables that need one of following encodings: binary encoding, ordinal encoding or one-hot encoding. Action: Encode categorical variables to numerical values.

3.1 Binary Encoding

Binary Encoding transform categorical variables with two values to binary values.

- Attrition (2 values): needs binary encoding to encode categorical values ('Yes'/'No') to numerical values.
- Gender (2 values): needs binary encoding.
- OverTime (2 values): needs binary encoding.

3.2 Ordinal Encoding

Ordinal encoding is similar to binary encoding, but here categorical variables have more than two values which follow an ordered series. The numbers we change for those variables also follow this order.

- BusinessTravel (3 values): is an ordinal variable and needs ordinal encoding. Here we set 0 to "Non-Travel"; 1 to "Travel_Rarely"; and 2 to "Travel_Frequently"

3.3 One-Hot Encoding

In one hot encoding, for each level of a categorical feature, we create a new variable. Each category is mapped with a binary variable containing either 0 or 1. Here, 0 represents the absence, and 1 represents the presence of that category.

These newly created binary features are known as Dummy variables. The number of dummy variables depends on the levels present in the categorical variable.

- Department (3 values): is a nominal variable and needs one-hot encoding.
- MaritalStatus (3 values): is a nominal variable and needs other one-hot encoding.
- EducationField (6 values): is a nominal variable and needs other categorical encoding.
- JobRole (9 values): is a nominal variable and needs other categorical encoding.

3.4 Data Normalization

The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model.

3.4.1 Scaling to a range

Scaling means converting floating-point feature values from their natural range (for example, 100 to 900) into a standard range—usually 0 and 1 (or sometimes -1 to +1). Use the following simple formula to scale to a range: ("Normalization" Last updated 2022)

$$x' = (x - x_{min})/(x_{max} - x_{min})$$

4 Machine Learning Algorithms

The term 'machine learning' is often, incorrectly, interchanged with Artificial Intelligence, but machine learning is actually a sub field/type of AI. Machine learning is also often referred to as predictive analytics, or predictive modelling.

Coined by American computer scientist Arthur Samuel in 1959, the term 'machine learning' is defined as a "computer's ability to learn without being explicitly programmed".

At its most basic, machine learning uses programmed algorithms that receive and analyse input data to predict output values within an acceptable range. As new data is fed to these algorithms, they learn and optimize their operations to improve performance, developing 'intelligence' over time.

In these case we are dealing with a specific type of machine learning algorithm called "Classification". In classification tasks, the machine learning program must draw a conclusion from observed values and determine to what category new observations belong.

4.1 Create Data Partition

Before we start creating the machine learning algorithms, we need to split the data into *train_set* and *test_set*. We use train_set to create the algorithm in question and then we would use test_set to set the optimal parameters and get the performance metrics.

Here our pre-processed dataset is divided into: 70% of data for training and 30% of data for testing.

4.2 Imbalanced Data and ROC

As we see in EDA section, this dataset has 1233 employees that stay at the company and 237 employees that left the company which means our dataset is imbalanced.

Imbalanced data refers to those types of datasets where the target class has an uneven distribution of observations, i.e one class label has a very high number of observations and the other has a very low number of observations.

Hence, if we predict all possible outcomes as employees that stay at the company we obtain a high accuracy but low specificity. Low specificity means that our algorithm fails predicting when an employee leaves the company.

Our goal in this section is to increase the specificity.

Table 3: Naive Approach

Method	Accuracy	Sensitivity	Specificity	AUC
Naive Approach	0.8371041	1	0	0.5

Since the data set is unbalanced to measure the efficiency of our algorithms we will use AUC metric instead of precision. AUC - ROC curve is a performance measurement for the classification problems at various threshold settings. ROC is a probability curve and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes. Higher the AUC, the better the model is at predicting 0 classes as 0 and 1 classes as 1.

The ROC curve is plotted with TPR against the FPR where TPR is on the y-axis and FPR is on the x-axis.

True Positive Rate (TPR) is a synonym for recall and is therefore defined as follows:

$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate (FPR) is defined as follows:

$$FPR = \frac{FP}{FP + TN}$$

4.3 Logistic Regression

Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Since the outcome is a probability, the dependent variable is bounded between 0 and 1. In logistic regression, a logit transformation is applied on the odds—that is, the probability of success divided by the probability of failure. This is also commonly known as the log odds, or the natural logarithm of odds, and this logistic function is represented by the following formulas:

$$Logit(\pi) = \frac{1}{1 + exp(-\pi)}$$

Table 4: Logistic Regression

Method	Accuracy	Sensitivity	Specificity	AUC
Naive Approach	0.8371041	1.0000000	0.0000000	0.5000000
Logistic Regression	0.8710407	0.9810811	0.3055556	0.6433183

4.4 Random Forest

Random forests are a very popular machine learning approach that addresses the shortcomings of decision trees using a clever idea. The goal is to improve prediction performance and reduce instability by averaging multiple decision trees (a forest of trees constructed with randomness). It has two features that help accomplish this.

The first step is bootstrap aggregation or bagging. The general idea is to generate many predictors, each using regression or classification trees, and then forming a final prediction based on the average prediction of all these trees. To assure that the individual trees are not the same, we use the bootstrap to induce randomness. These two features combined explain the name: the bootstrap makes the individual trees randomly different, and the combination of trees is the forest. (Irizarry 2019)

Table 5: Logistic Regression

Method	Accuracy	Sensitivity	Specificity	AUC
Naive Approach	0.8371041	1.0000000	0.0000000	0.5000000
Logistic Regression	0.8710407	0.9810811	0.3055556	0.6433183
Random Forest	0.8438914	0.9756757	0.1666667	0.5711712

4.5 K-Nearest Neighbor

The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph.

There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice. (Harrison 2018)

We implement this algorithm in our data. Let's see!

Table 6: K-Nearest Neighbor

Method	Accuracy	Sensitivity	Specificity	AUC
Naive Approach	0.8371041	1.0000000	0.0000000	0.5000000
Logistic Regression	0.8710407	0.9810811	0.3055556	0.6433183
Random Forest	0.8438914	0.9756757	0.1666667	0.5711712
K-Nearest Neighbor	0.8506787	0.9918919	0.1250000	0.5584459

4.6 Gradient Boosting Machines

Gradient boosting machines (GBMs) are an extremely popular machine learning algorithm that have proven successful across many domains and is one of the leading methods for winning Kaggle competitions. Whereas random forests build an ensemble of deep independent trees, GBMs build an ensemble of shallow trees in sequence with each tree learning and improving on the previous one. Although shallow trees by themselves are rather weak predictive models, they can be "boosted" to produce a powerful "committee" that, when appropriately tuned, is often hard to beat with other algorithms.

Table 7: GBM

Method	Accuracy	Sensitivity	Specificity	AUC
Naive Approach	0.8371041	1.0000000	0.0000000	0.5000000
Logistic Regression	0.8710407	0.9810811	0.3055556	0.6433183
Random Forest	0.8438914	0.9756757	0.1666667	0.5711712
K-Nearest Neighbor	0.8506787	0.9918919	0.1250000	0.5584459
GBM	0.8778281	0.9810811	0.3472222	0.6641517

As we see in the table above, GBM performs better than Random Forest and Knn. Therefore, let us try to improve the GBM model by addressing the problem of unbalanced data.

Two approaches to make a balanced dataset out of an imbalanced one are under-sampling and over-sampling.

4.6.1 Under-sampling

Under-sampling balances the dataset by reducing the size of the abundant class. This method is used when quantity of data is sufficient. By keeping all samples in the rare class and randomly selecting an equal number of samples in the abundant class.

4.6.2 Over-sampling

On the contrary, oversampling is used when the quantity of data is insufficient. It tries to balance dataset by increasing the size of rare samples. Rather than getting rid of abundant samples, new rare samples are generated by using e.g. repetition, bootstrapping or SMOTE (Synthetic Minority Over-Sampling Technique) .

4.6.3 Weighting

In both previous cases, we manually use a different population to train the model on than what we naturally had, with the express purpose of affecting the model we're going to build. Weighting is kind of like this, but instead of duplicating or removing records, we assign different weights to each record as a separate column.

Table 8: GBM Imbalanced Correction

Method	Accuracy	Sensitivity	Specificity	AUC
GBM Weighted	0.8257919	0.8594595	0.6527778	0.7561186
GBM Under-Sampling	0.7420814	0.7324324	0.7916667	0.7620495
GBM Over-Sampling	0.7850679	0.8027027	0.6944444	0.7485736

After fitting the model to fill the gap in the data, we can see that the results in AUC are significantly better than previously obtained. Let's see how the table looks with all the algorithms and their respective metrics.

Table 9: Final Results

Method	Accuracy	Sensitivity	Specificity	AUC
Naive Approach	0.8371041	1.0000000	0.0000000	0.5000000
Logistic Regression	0.8710407	0.9810811	0.3055556	0.6433183
Random Forest	0.8438914	0.9756757	0.1666667	0.5711712
K-Nearest Neighbor	0.8506787	0.9918919	0.1250000	0.5584459
GBM	0.8778281	0.9810811	0.3472222	0.6641517
GBM Under-Sampling	0.7420814	0.7324324	0.7916667	0.7620495

5 Final results

After testing several machine learning algorithms, we can see that the best value in the AUC is the GBM model. Despite this, our model was still inefficient in predicting employees leaving the company (specificity). For this reason, we decided to apply three important techniques for handling unbalanced datasets (weighted, under-sampling and over-sampling).

As we saw in Table 9, our GBM model with down-sampling is the one that yields the best AUC values with 0.7620495. This model, although it reduces the precision compared to the previous ones, manages to considerably increase the specificity with a value of 0.7916667 surpassing the GBM model (which was the best of the previous ones) by approximately 45 points (around 127.95% better).

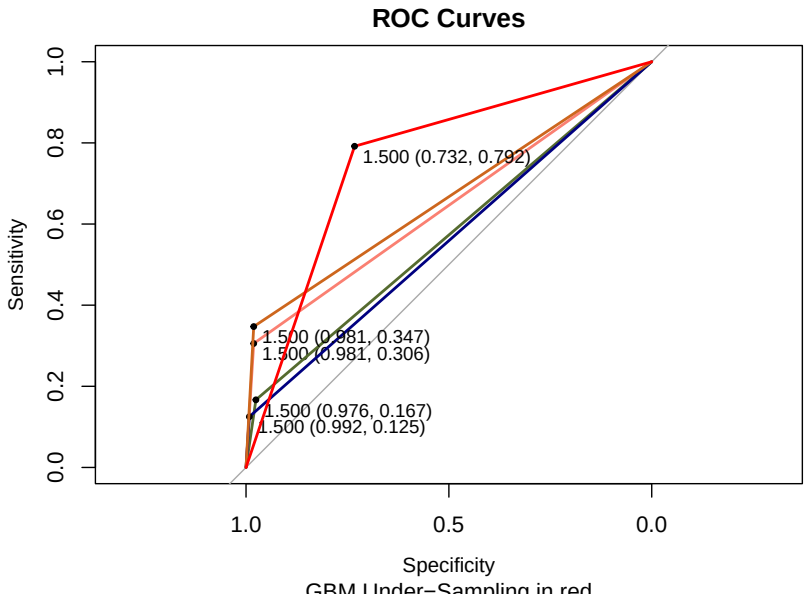

Figure 9: ROC Curves

6 Conclusion

After creating an algorithm that can predict around 80% of employees leaving the company, we can extract the most important variables that were used to build the model. Let's make a plot!

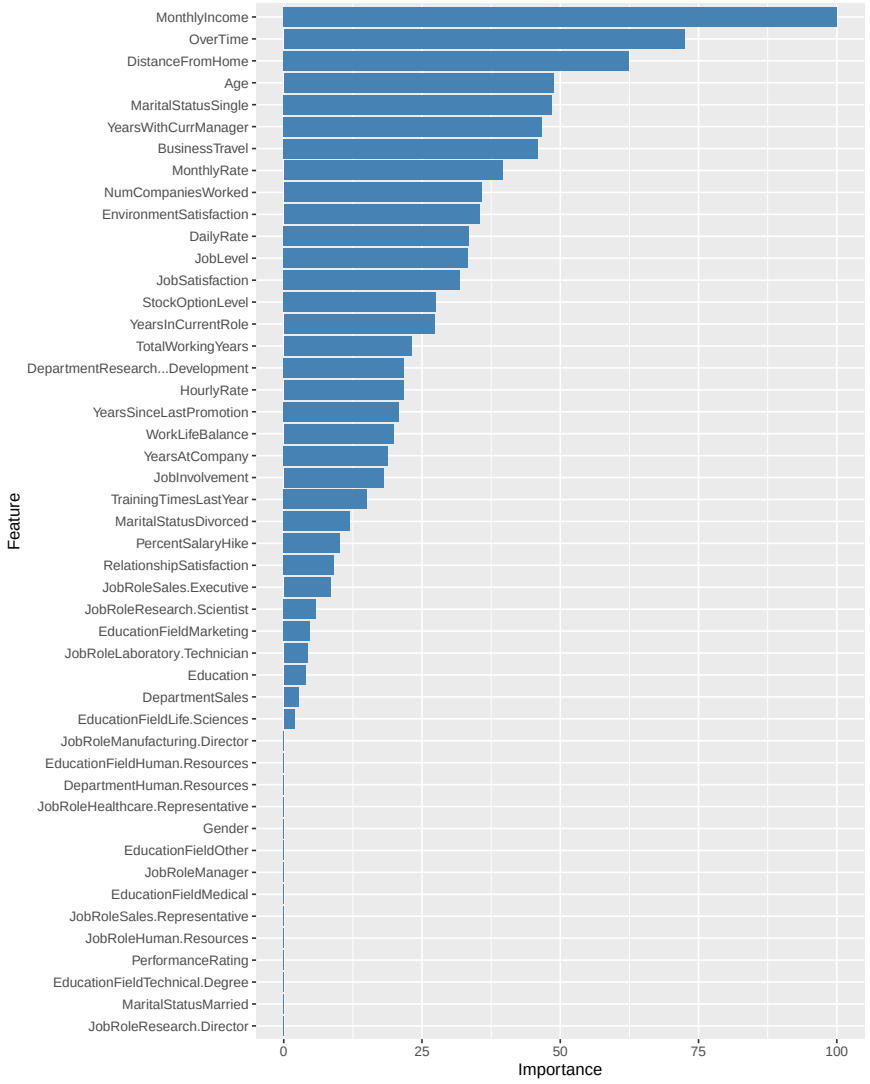

Figure 10: Variable importance According to GBM Under-Sampling

As we see the top 5 important variables are: MonthlyIncome, Overtime, DistanceFromHome, Age and MaritalStatusSingle. This make perfect sense.

Suppose we work for a company that pays us a low salary, surely we would think about changing our job, especially if even doing overtime the salary does not increase as we wish (as we saw in the EDA section). In addition to this, a person who lives far from work is also prone to leave it and look for something closer to home, due to all the transport problems that this entails, in addition to the obvious loss of time.

Following this logic, we can say that the last two variables are related to each other. An older

person is generally more likely to marry and have children, which means changing jobs is not the best idea, assuming you have a family to support.

To finish the report, we can say that we created a model that can make good predictions. But beware! This work could be improved, perhaps working on the imbalance of the data from another perspective, using other types of algorithms to predict, or perhaps we could experiment with neural networks. We'll leave it for next time!

References

Harrison, Onel. 2018. "Machine Learning Basics with the k-Nearest Neighbors Algorithm," September.

Irizarry, Rafael A. 2019. *Introduction to Data Science: Data Analysis and Prediction Algorithms with r*. CRC Press.

"Long Tail." n.d. Wikipedia. https://en.wikipedia.org/wiki/Long_tail.

"Normalization." Last updated 2022. GoogleDevelopers. https://developers.google.com/machine-learning/data-prep/transform/normalization?hl=es-419.