# IBM  HR Analytics Employee Attrition and Performance

Isabela Caetano

July 2021

## I. Background

With the rise of Data Science over the past few years, companies are finding more ways to apply machine learning methods within their own establishments. One such way is People Analytics. People Analytics is a data driven  way for companies to better understand and manage their employees. It comes as no surprise that IBM is one company that focuses on People Analytics.

The focus of this project was on figuring out what causes IBM employees to feel attrited and creating a predictive model that will classify attrited employees versus non attrited employees. The dataset used came from Kaggle, but originally provided by IBM data scientists to demonstrate their Watson analytics tool.

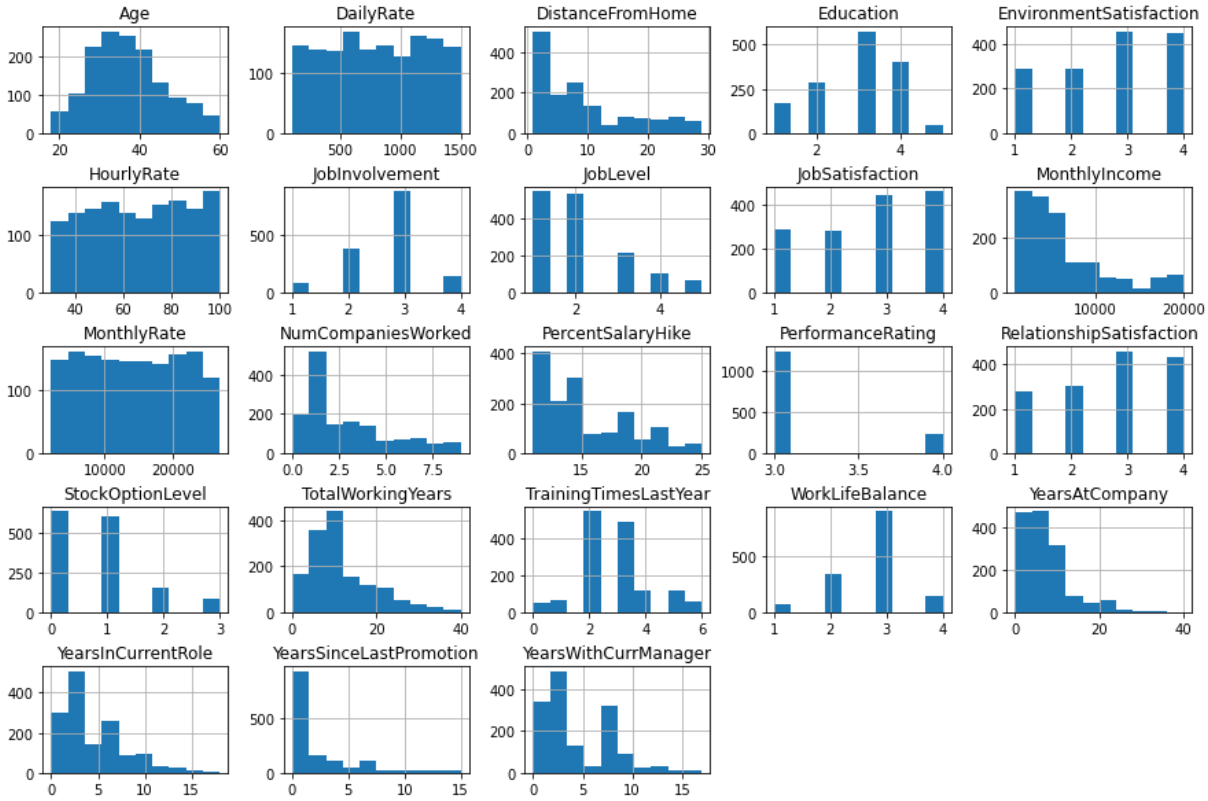
## II. Data Wrangling

The original data set had 1470 records with 35 different features, so it was very workable in terms of size. Luckily, it also contained no duplicates or missing values. However, there was a mix of categorical and numeric features, which was dealt with later in preprocessing.

Additionally, there was one feature found that had the same value for every record which was `Over18`. This entire column was dropped and the size of the data after wrangling wsa 1470  x 34.

## III. Data Exploratory Analysis

The first thing that was explored was distributions of each of the features.

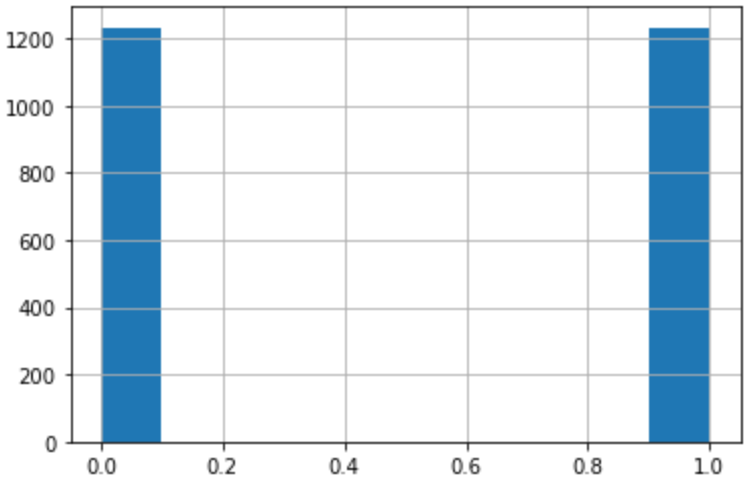



It was clear that there were a few skewed distributions such as `TotalWorkingYears`, `PercentSalaryHike`, `MonthlyIncome`, `YearsAtCompany`, `YearsInCurrentRole` and `YearsSinceLastPromotion`. These skewed distributions were not surprising since they have to do with time at the company or salary.

## IV. Preprocessing

The features of this dataset are a mix of categorical and numerical data types. So for preprocessing, it was important to encode the categorical features. Applying a simple `LabelEncoder` did the trick for some of the features, but for others such as `BusinessTravel`, an `OrdinalEncoder` was needed. The last part of encoding was using a OneHot encoder. This was applied to the features that had a relatively small set of distinct values, to not overwhelm the size of the dataset. After all the encoding was done, the total number of columns was 41.

The next step in preprocessing was dealing with the imbalance of the data. To balance the data, a SMOTE technique was used.




Balanced Data

The last part of preprocessing was to scale the data. Since all the features are on a different scale, a `MinMax` scaler was used to transform the data. After the data was transformed, the data was split 70-30 for testing and training.

## V. Modelling

The final step for this project was modelling. Five models, plus one dummy model, were chosen: KNN, Decision Tree, Logistic Regression, Gradient Boosting and Random Forest.

The dummy model chosen was a sklearn `DummyClassifier` with the strategy set to stratified. Below are the the results of the base model:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.53 | 0.55 | 0.54 | 370 |
| 1 | 0.54 | 0.52 | 0.53 | 370 |
| accuracy | | | 0.54 | 740 |
| macro avg | 0.54 | 0.54 | 0.53 | 740 |
| weighted avg | 0.54 | 0.54 | 0.53 | 740 |

For remaining models, hyperparameter tuning with a 5 fold cross validation was used. To tune each model, a `RandomizedSearchCV` was used to select the hyperparameters. Below are the f1 scores for each of the models:

## KNN

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.84 | 0.87 | 370 |
| 1 | 0.85 | 0.91 | 0.88 | 370 |
| accuracy | | | 0.88 | 740 |
| macro avg | 0.88 | 0.88 | 0.88 | 740 |
| weighted avg | 0.88 | 0.88 | 0.88 | 740 |

## Decision Tree

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.83 | 0.81 | 0.82 | 370 |
| 1 | 0.82 | 0.84 | 0.83 | 370 |
| accuracy | | | 0.82 | 740 |
| macro avg | 0.82 | 0.82 | 0.82 | 740 |
| weighted avg | 0.82 | 0.82 | 0.82 | 740 |

## Logistic Regression

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.89 | 0.95 | 0.92 | 370 |
| 1 | 0.94 | 0.88 | 0.91 | 370 |
| accuracy | | | 0.91 | 740 |
| macro avg | 0.92 | 0.91 | 0.91 | 740 |
| weighted avg | 0.92 | 0.91 | 0.91 | 740 |

## Random Forest

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.92 | 0.91 | 370 |
| 1 | 0.92 | 0.90 | 0.91 | 370 |
| accuracy | | | 0.91 | 740 |
| macro avg | 0.91 | 0.91 | 0.91 | 740 |
| weighted avg | 0.91 | 0.91 | 0.91 | 740 |

## Gradient Boosting

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.90 | 0.93 | 0.92 | 370 |
| 1 | 0.93 | 0.90 | 0.91 | 370 |
| accuracy | | | 0.91 | 740 |
| macro avg | 0.92 | 0.91 | 0.91 | 740 |
| weighted avg | 0.92 | 0.91 | 0.91 | 740 |

Each of the models performed fairly well, with f1 scores all above 0.80. In the end, the model selected was the random forest for accuracy and speed reasons. From the chosen random forest model, the top feature importances were found to be related to marital status: