MCS: Masters's Degree in Business Analytics and Big Data

Machine Learning II

PREDICTING EMPLOYEE ATTRITION

Prepared by

PIERRE-LOUIS BERLEMONT | FRANCESCO DESCALZI

HASSAN EL MOKDAD | SACHIN NAIR | ALESSANDRO NERI

ANDREA PEREZ | JORGE ZABALLA

Professor:

ANGEL CASTELLANOS GONZALEZ

IE School of Human Sciences and Technology

Madrid, Spain

March 2023

I. Introduction

Statement of the problem and objectives

Employee attrition, or the process of employees leaving a company over a period of time, is a critical concern for many businesses. It is not only costly to replace a departing employee but employee attrition can also result in the loss of institutional knowledge and experience that can negatively impact a company's performance. As a result, companies need to be able to predict and mitigate employee attrition as accurately as possible.

Predicting *employee attrition* can be a complex task as it depends on a variety of factors such as job satisfaction, salary, work-life balance, years spent in the same role, among others. Furthermore, the reasons behind an employee's decision to leave can be personal or professional, making it challenging for employers to identify and address these factors.

However, machine learning and data analytics advances have enabled companies to gain deeper insights into employee behavior, including predicting the likelihood of attrition. By analyzing various factors that contribute to employee turnover, companies can develop targeted strategies to reduce attrition rates and improve employee retention.

In this paper, we will explore the **Data Attrition Dataset** from *Kaggle*, attempt to find an optimal Classification algorithm that can classify the binary target 'Attrition' well, and optimize those algorithms to find the best possible predictors for the target.

Our Classification solution should not only be accurate at predicting whether or not an employee is likely to leave the company, but should also aid the company in taking preventive measures to avoid attrition.

Overview of the approach

To ensure that we address the business problem effectively, we adopt a business-centric approach rather than treating machine learning as an isolated activity. This helps us prioritize the solution over achieving a perfect model that may not necessarily align with the business needs.

Following standard practice, we begin by conducting a detailed exploratory data analysis and feature engineering process. This involves understanding the data and its associated features, and transforming them appropriately to prepare for model training.

Next, we move onto model selection and evaluation, considering a range of classification algorithms such as Logistic Regression, RandomForestClassifier, XGBClassifier, and SVC, based on their suitability to the problem context. To prevent overfitting, we perform cross-validation and compare the performance of different models.

Once we have selected the best model, we use it to generate predictions and evaluate our results using various scoring metrics. We then translate these predictions into a business strategy that can effectively solve the problem at hand. By taking a holistic approach, we ensure that our machine learning efforts are directed towards achieving the desired business outcomes.

II. Exploratory Data Analysis

Data visualization and summary statistics

Prior to conducting the data cleaning and engineering stages of the machine learning process, our team sought to gain a comprehensive understanding of the data. We performed various data visualizations, primarily utilizing bar charts to compare the target variable (Attrition) against categorical features and box plots to compare it against continuous features. Our team discovered intriguing insights from these visualizations, which we will delve into below. For a complete overview of the plots, please refer to Appendices I and II.

Numerical Features vs Target (*Attrition)*

Our team discovered a noteworthy insight regarding the relationship between employee age and attrition. Specifically, we found that individuals who left the company (Attrition = 1) tended to be younger than those who remained (Attrition = 0). This observation was derived from analysing the age distributions of both target classes. A graphical representation of this comparison can be found in **Figure 1**:

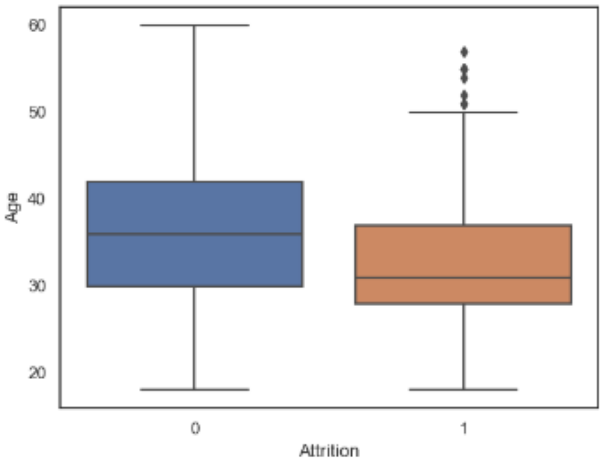


Figure 1 - *Age* Distribution vs *Attrition* Class

The team also uncovered a significant finding during the visual analysis of continuous features, specifically in relation to the MonthlyIncome variable. Our analysis revealed that employees with lower income levels had a higher likelihood of leaving the company than

those with higher income levels. This insight, visualised on **Figure 2,** sheds light on the potential impact of financial compensation on employee retention.

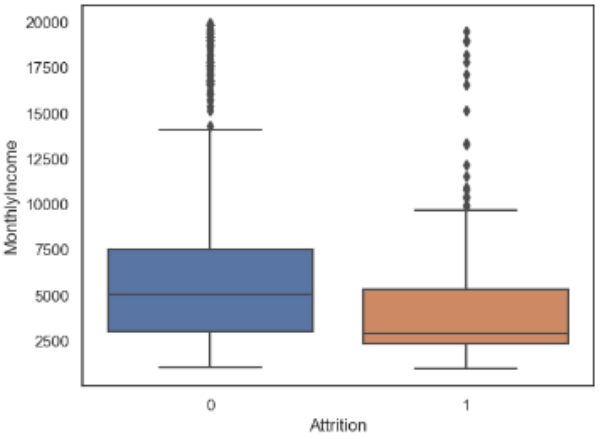


Figure 2 - *MonthlyIncome* vs *Attrition* Class

The insights gained from our data visualization analysis enabled our team to identify key numerical features that could potentially impact our model. The observed correlation between employee age, income levels, and attrition rates aligns with common industry knowledge, namely that younger employees tend to change jobs more frequently and that lower incomes may prompt individuals to seek out better-paying opportunities. While no other significant relationships were uncovered between continuous features and the target variable, these findings provide valuable context for our modeling decisions. For additional visualizations and insights related to categorical features, please refer to Appendix I.

Categorical Features vs Target (*Attrition)*

In order to better understand how our target variable behaved across different categorical features, we created a series of bar charts showing the count of each category and whether or not employees in that category had quit the company. To provide a more meaningful comparison between categories, we also included the proportion of employees in each category on top of each chart. This allowed us to easily identify the employee categories that had a more significant impact on the target feature *Attrition*. By visualizing how the target variable varied across categorical features, we gained valuable insights that can help inform our machine learning model selection and feature engineering decisions.

After visualizing the count and proportion of Attrition by each categorical feature, we focused on the most significant insights. One notable finding was the difference in Attrition rate between genders. Among all employees who quit the company (11.93% of the total dataset), 69% were men and 31% were women. Specifically, of the all employees in the dataset, 8.23% who quit were male, compared to 3.7% of female employees. This indicates that male employees are more likely to leave the company than female employees. This finding could have important implications for identifying the factors that contribute to Attrition and developing strategies to retain male employees. See **Figure 3:**

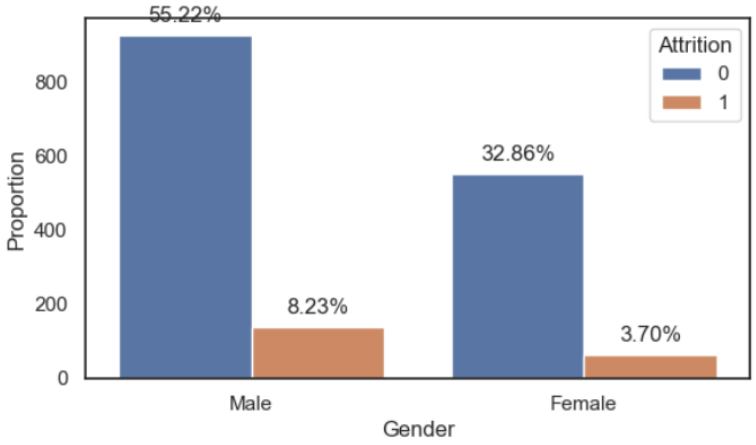
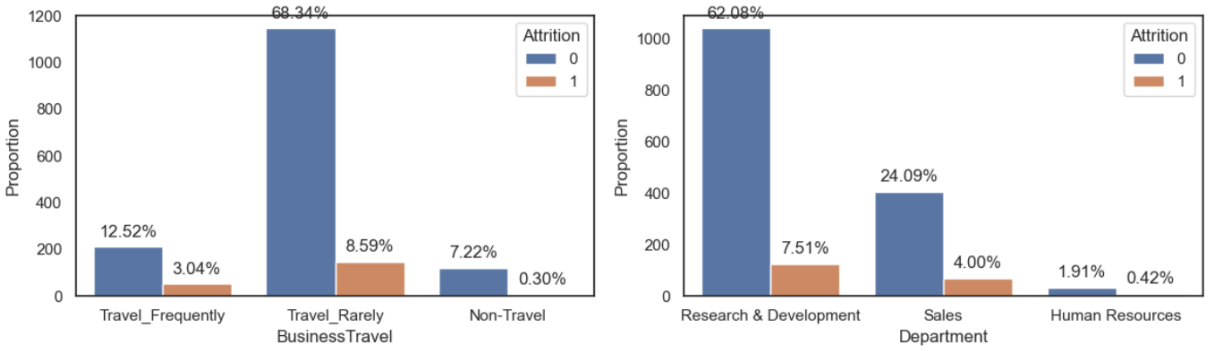


Figure 3 - *Gender* Count and Proportion *Attrition*

The other interesting insight drawn from this exercise is the majority (72%) of 'Quitters' (*Attrition = 1)* belong to the "Travel_Rarely" Category under the *BusinessTravel* feature, and also are from the R&D Department (63% of *Attrition*) as shown in **Figures 4 and 5** .



Figures 4 and 5 - *BusinessTravel* and *Department* Count and Proportion vs *Attrition*

Analysing the figures of our dataset provides valuable insights into the behaviour of both numerical and categorical features and their relationship with the target variable. This information can guide us toward making informed decisions on potential feature engineering strategies throughout the process. By identifying the features that have strong correlations with the target, we can focus our efforts on improving these features to enhance the performance of our machine-learning model.

In addition to helping us with the model, understanding these relationships helps business to understand what factors seem to be contributing to the problem of Attrition within the company. They can then build strategies to tackle the problem by focusing on such relevant categorical and numerical features.

III. Feature Engineering

Data cleaning and preparation

Before conducting exploratory data analysis, the dataset was cleaned and prepared to ensure data quality and consistency. The initial cleaning process included checking duplicates, handling missing values, and correcting data type errors. Luckily the dataset came with no duplicates or missing values but there were some data types that needed to be recast using the following code:

```
data = data.astype({"Education" : "object",
           "EnvironmentSatisfaction" : "object",
           "JobLevel" : "object",
           "JobInvolvement" : "object",
           "JobSatisfaction" : "object",
           "PerformanceRating" : "object",
           "RelationshipSatisfaction" : "object",
           "StockOptionLevel" : "object",
           "WorkLifeBalance" : "object"})
```

Even though some of these features were “correctly” identified as numerical (int64), we realized after further inspection that they were representing categorical features. We will later be calling these categorical features to encode them, and hence it is better to have them all in the ‘object’ data type group so they can be easily called later.

After the data types were corrected, we proceeded to check for other possible transformations that the data set would need before proceeding with visualization and statistical analysis. Using functions like `data.describe()` and `.value_counts()` we were able to identify some columns that could be removed or that had some values that demonstrated imputation mistakes:

FEATURE	FINDING	DECISION
Over18	Same value for all instances	Drop feature
EmployeeCount	Same value for all instances	Drop feature
Education	1 instance with value 15 (rest of values 1-5)	Drop instance
JobLevel	1 instance with value 7 (rest of values 1-5)	Drop instance

Feature Engineering

With the EDA and visualization exercise earlier, we were able to understand some of the relationships that exist between our explanatory features and the target. The team wanted to create some new features stemming from the original features to see if they helped our classifiers understand these relationships better.

To accomplish this, we employed a binning technique, which involved categorizing continuous features into discrete ones using custom functions with conditional statements. This approach allowed us to extract relevant information from the continuous features and create new columns that better capture the underlying relationships with the target variable. For a detailed view of the functions used, please refer to Appendix III, but here we can look at a quick example:

```
def cat_age(value):
    if value <= 20:
        return "Between 18-20 years"
    elif (value > 20) & (value <= 30):
        return "Between 21-30 years"
    elif (value > 31) & (value <= 40):
        return "Between 31-40 years"
    elif (value > 41) & (value <= 50):
        return "Between 41-50 years"
    else:
        return "Above 50 years"

data['Cat_Age'] = data['Age'].apply(cat_age)
```

After incorporating the newly engineered features into our machine learning models, we did not observe a significant improvement in performance. Despite adding multiple features, they did not contribute significantly to the classification exercise. In the interest of simplicity and to improve the overall performance of our models, we decided to proceed without these features.

It is worth noting that we ran our models both with and without these features and concluded that including them in the final models would negatively impact the accuracy scores. Therefore, we have chosen to focus on the most relevant features and avoid overfitting the models.

Dealing with Outliers

We ran a custom function to detect outliers for our different numeric features and get the percentage of outliers per feature. The highest percentages belonged to categorical features

so we did not look at those directly. The non-categorical features with the highest percentage of outliers were *TrainingTimesLastYear* (12.6%) and *YearsSinceLastPromotion* (13.4%). Our next step was to decide what to do with these outliers.

Since our dataset is already quite small, we needed to be very careful about whether or not to drop these instances. Going deeper into each of these features we realize that they represent real-world scenarios and hence dropping them would not be the best step to take. There are employees that are trained more often than others, as well as employees that last much longer in companies than most. We believe these features to be good predictors for whether or not an employee will leave his/her current company and so we decided to keep them in the model.

Distribution of target variable

The target variable *Attrition*, is a binary target (1, 0) for whether or not that employee has left the company. The distribution of our instances is as follows:

- Attrition - Yes (1) = 11.93%
- Attrition - No (0) = 88.08%

This tells us that our training data is highly unbalanced, with the data skewing towards the majority class *No (0).* This is important to bear in mind because the Machine Learning models we will implement may be biased to lean toward the majority class, perform poorly over the minority class, and could cause the model to not generalize well once it is put under production. In order to account for this imbalance in the target variable we need to look at metrics that are robust for this specific situation such as the F-1 score or ROC-AUC. Also, we will put importance to the cost of a false negative such as Recall. Indeed, it measures the proportion of true positives amongst all actual positives and therefore indicates how many of the positive cases were correctly identified.

Correlation analysis

A correlation analysis was performed to identify the relationship between different variables in the dataset. The objective was to find those features that have a high correlation and decide to keep them or drop them.

A correlation matrix was calculated and visualised using a heatmap using the following code:

```
corr = data.corr()
corr.style.background_gradient(
    cmap='coolwarm',
    axis = None).format(
                    precision = 2) #using this code to create a gradient
background for the matrix
```

The output was too large to include in the report and can be found on the Jupyter notebook or in Appendix 4, however, there were some critical conclusions that the team derived from this analysis. For the purposes of this exercise, we assume any correlation > 0.7 to be high and remove certain features on the basis of this:

FEATURE	CORRELATED WITH	DECISION
YearsAtCompany	*YearsInCurrentRole*	Drop *YearsAtCompany*
YearsInCurrentRole	*YearsWithCurrManager*	Drop *YearsWithCurrManager*
TotalWorkingYears	*MonthlyIncome*	Drop *TotalWorkingYears*

Overall, the exploratory data analysis provided insights into the dataset and identified potential factors that may contribute to employee attrition. These insights were used to inform the machine learning process's feature engineering and modeling stages.

Feature scaling and encoding

With the data-type reclassification code we discussed earlier in this paper it was quite simple to manipulate the different categories that needed scaling and encoding. We identified 9 columns that are of Categorical nature and that needed to be One-Hot Encoded and named them `data_cat`:

```
["Education", "EnvironmentSatisfaction","JobLevel", "JobInvolvement",
"JobSatisfaction","PerformanceRating","RelationshipSatisfaction",
"StockOptionLevel","WorkLifeBalance"]
```

We also identified 15 numerical columns that needed to be scaled using the `StandardScaling(),` and named them `data_num` .

```
['Age', 'DailyRate', 'DistanceFromHome', 'HourlyRate', 'MonthlyIncome',
 'MonthlyRate', 'NumCompaniesWorked', 'PercentSalaryHike',
 'StandardHours', 'TotalWorkingYears', 'TrainingTimesLastYear',
 'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
 'YearsWithCurrManager']
```

Having identified these columns, we could now run the appropriate `ColumnTransformer()` function below to prepare the data for feeding it to the different models. The remaining columns were kept in a different variable and concatenated afterward.

```
col_transformer = ColumnTransformer(
    [("standard_scaler",StandardScaler(),data_num.columns),
     ("one_hot_encoder", OneHotEncoder(),data_cat.columns)],
    remainder='drop')
```

These transformations yielded our `X_train_final` dataset with a shape of 1675 rows and 47 columns (originally 35 columns).

IV. Modeling

Splitting the data into training, validation, and testing sets

The datasets that were provided to our team had already been split into Train and Test therefore we found no need to further split the dataset. However, we will be using Cross Validation techniques within each of our models to get an approximate idea of how they are performing with newly added data. For this strategy we performed a split on our train data:

```
X_train_split, X_val, y_train_split, y_val = train_test_split(X_train,
                                                              y_train,
                                                              test_size = 0.2,
                                                              random_state = 42
                                                                     )
```

Selection of appropriate models

We decided to run a few classifiers to try and get the best-performing model possible. In order to determine what our best performance is, we decided to look at 4 key indicators that would weigh on the decision of “best performance”:

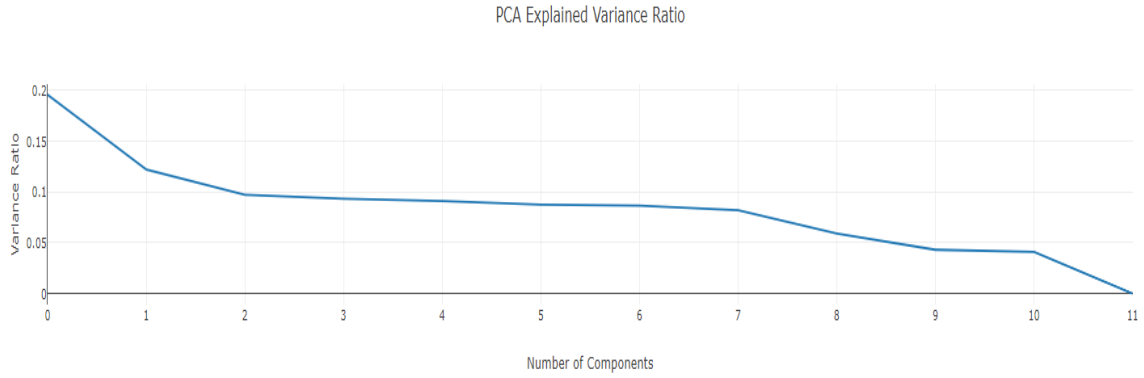
1. **ROC-AUC Score** - since we are interested in our model’s capabilities to correctly predict the positive class vs the negative class, the ROC score was determined to be our main driver for performance.
2. **Recall Score -** We are using this metric as well because, from the business context of this problem, the costs associated with not retaining employees are higher than the cost of misclassifying employees as leaving even when they do not.
3. **Variability between ROC Test Scores -** since we ran a Cross-Validation on every model to find the best possible parameters, we wanted to avoid overfitting. We also wanted to be weary of the mean difference between the Train vs Test scores of all validation folds. If we found that a model predicted high scores on the Training set and then dropped significantly on the validation test, this would show that the model was overfitting and performing poorly when fed unseen data.

4. **Interpretability / Explainability of Results -** Since our main audience is an executive audience, we wanted to choose a model that can be easily interpreted and explained by this audience. Yes, we could run a PCA / LDA analysis and come up with fantastic principal components and a great LDA that predicts accurately. But if our audience later wanted to know which features are the ones that affect Attrition the most so that they could strategize preventive measures, we would lose a lot of that business applicability.

Principal Component Analysis

We decided to start with PCA to check if dimensionality reduction could be useful. We run this analysis using only the 11 numerical features since finding correlations using categorical features is not appropriate.

After running the PCA analysis, the number of principal components did not help us reduce the complexity of our dataset as the new variables were less interpretable and provided as much variance from the original dataset as the 11 original numerical variables.



There is no clear set of reduced components that explain the variance in numerical features. Despite the fact that we have the same number of components as the original features, we decided to run a RandomForest with PCA (please refer to the notebook) and the scores did not improve. Hence, we decided to proceed with the original numerical features.

Random Forest with GridSearchCV

The next model we attempted was a `RandomForestClassifier().` We ran a `GridSearchCV()` function in order to get the best parameters with the scoring parameter set to 'roc_auc'.

These were our results:

```
Best hyperparameters:  {'max_depth': 5, 'max_features': 10, 'min_samples_leaf': 4, 'min_samples_split': 2, 'n_estimators':
100}
Best auc score:  0.810079113203129
Mean Train Score:  0.9126217188008523
Mean Test Score:  0.7884462273197743
```

Here we can see that it seems our model is overfitting as there is a big difference between the mean train and test scores. Nevertheless, we will cross-validate this score on our validation set that we defined earlier to make sure of that.

This was our best result:

```
Threshold of 0.15:  auc:  0.8059981255857546 recall :  0.8181818181818182 precision:  0.375
```

After having checked the AUC scores per threshold for the predictions on our validation set we determine that our optimal threshold is 15% with an AUC on the validation set of 80.6%. Moreover, we can see that with this threshold we have a relatively high recall of 81.81%, meaning we are able to correctly predict employees that will leave while not correctly predicting around 18% of employees that will indeed leave.

Logistic Regression with GridSearchCV

Even though our RandomForest model seems to have very positive results and has shown to be an adequate predictor for attrition, we proceeded to look at more models to see if there was a better-performing model. The next model we ran was a `LogisticRegression().` We ran a `GridSearchCV()` function in order to get the best parameters with the scoring parameter set to 'roc_auc'.

These were our results:

```
Best hyperparameters:  {'C': 1, 'max_iter': 500}
Best auc score:  0.8215584037241074
Mean Train Score:  0.8468368289336179
Mean Test Score:  0.8092873381098707
```

We can see that it seems the Logistic Regression is overfitting less than the RandomForestClassifier as expected. We will then cross-validate this score on our validation set that we defined earlier to check the performance of this model on unknown data.

This was our best result:

```
Threshold of 0.2:  auc:  0.7738987816307404 recall :  0.6818181818181818 precision:  0.43478260869565216
```

Here we see that using a LogisticRegression gives us validation scores lower than the best threshold used in our RandomForestClassifier. Indeed, we get lower AUC scores and lower recalls using the LogisticRegression. We will therefore discard it and keep our best model being RandomForestClassifier.

Support Vector Machine with GridSearchCV

The next model we ran was an `SVC().` We ran a `GridSearchCV()` function in order to get the best parameters with the scoring parameters set to 'roc_auc'.

These were our results:

```
Best hyperparameters:  {'C': 0.1, 'class_weight': 'balanced', 'kernel': 'linear'}
Best auc score:  0.8240050706733753
Mean Train Score:  0.8947277921481359
Mean Test Score:  0.7815907672714526
```

Again, we can observe that the SVC has a high difference between the mean test score and the mean train score, which could indicate overfitting during the cross-validation. We will cross-validate this score on our validation set to check the performance of this model on unknown data.

This was our best result:

```
Threshold of 0.15:  auc:  0.7275070290534207 recall :  0.6818181818181818 precision:  0.3125
```

Here we see that using a SupportVectorClassifier gives us validation scores lower than the best threshold used in our RandomForestClassifier. Indeed, we get lower AUC scores and lower recalls using SVC. We will therefore discard the use of the SupportVectorClassifier and keep our best model being RandomForestClassifier.

XGBClassifier with GridSearchCV

The last model we ran will be an `XGBClassifier().` We ran a `GridSearchCV()` function in order to get the best parameters with the scoring parameters set to 'roc_auc'.

These were our results:

```
Best hyperparameters:  {'colsample_bytree': 0.4, 'gamma': 0.9, 'learning_rate': 0.2, 'max_depth': 3, 'n_estimators': 20}
Best auc score:  0.8270345431693554
Mean Train Score:  0.8790592862956564
Mean Test Score:  0.7875896402538897
```

Again we can see that it seems our model is overfitting as there is a big difference between the mean train and test scores. We will cross-validate this score on our validation to check that.

This was our best result:

```
Threshold of 0.2:  auc:  0.7876444860980942 recall :  0.6818181818181818 precision:  0.4918032786885246
```

Here we see that using an XGBClassifier gives us validation scores higher than the SVC but lower than the best threshold used in our RandomForestClassifier. Indeed, we get lower AUC scores and lower recalls using XGBClassifier. We will therefore discard it and keep our best model being RandomForestClassifier.

VII. Conclusion

The business problem of Attrition is something that can be better tackled using Machine Learning approaches as we have described above. Understanding historical data and figuring out patterns would be the first step to tackling this problem.

We have decided to proceed with the RandomForestClassifier as it gives us the best AUC score and Recall score on our validation set as demonstrated above. What this means from a business perspective is that we have prioritized the model that helps reduce the error in not capturing employees that actually leave the company as compared to misclassifying them as leaving even when they do not.

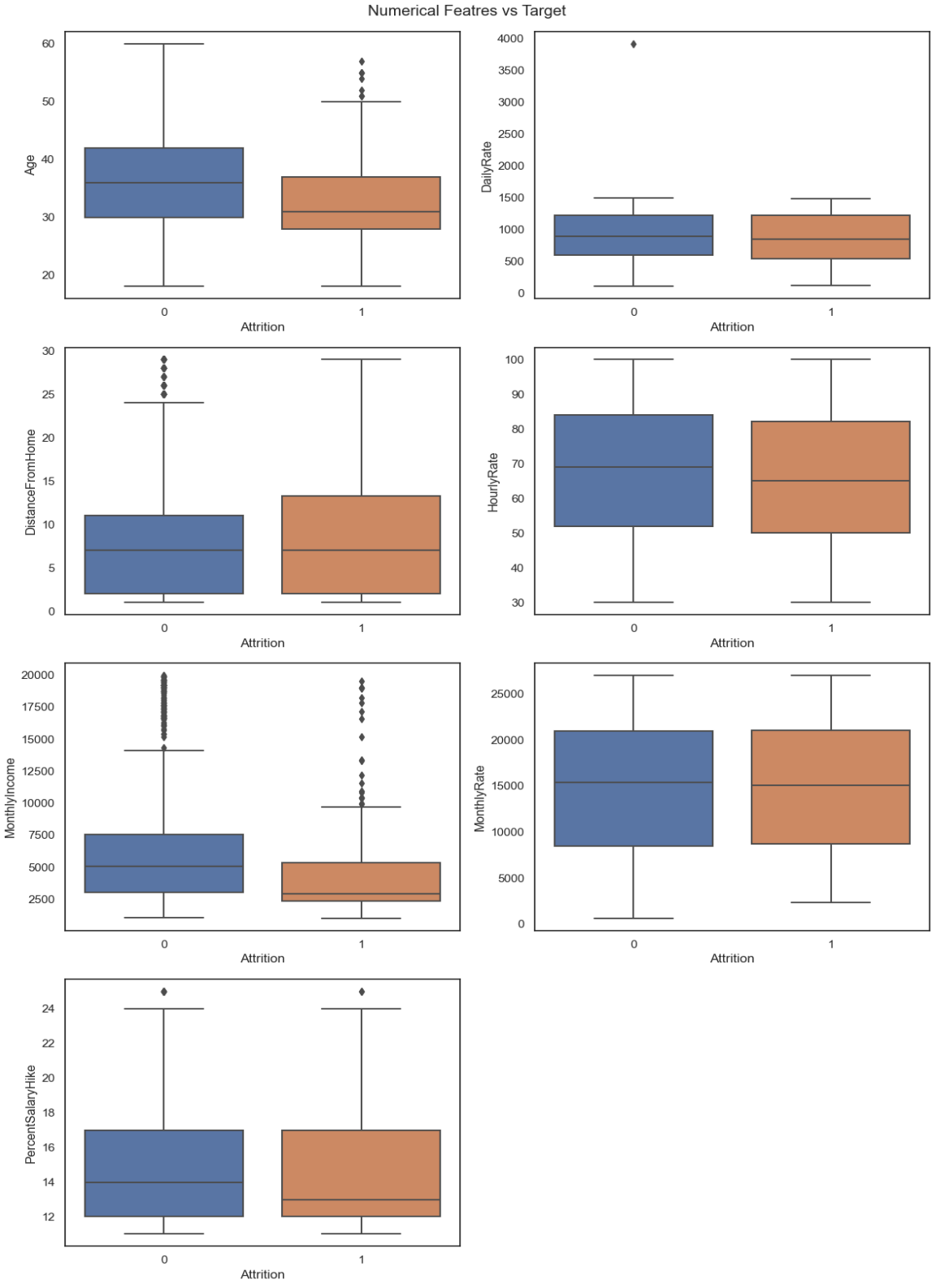
We need to be able to translate the model outputs into business measures so that other stakeholders can understand the implications and make strategic decisions in tandem with what the model showcases.

Future work and areas for improvement

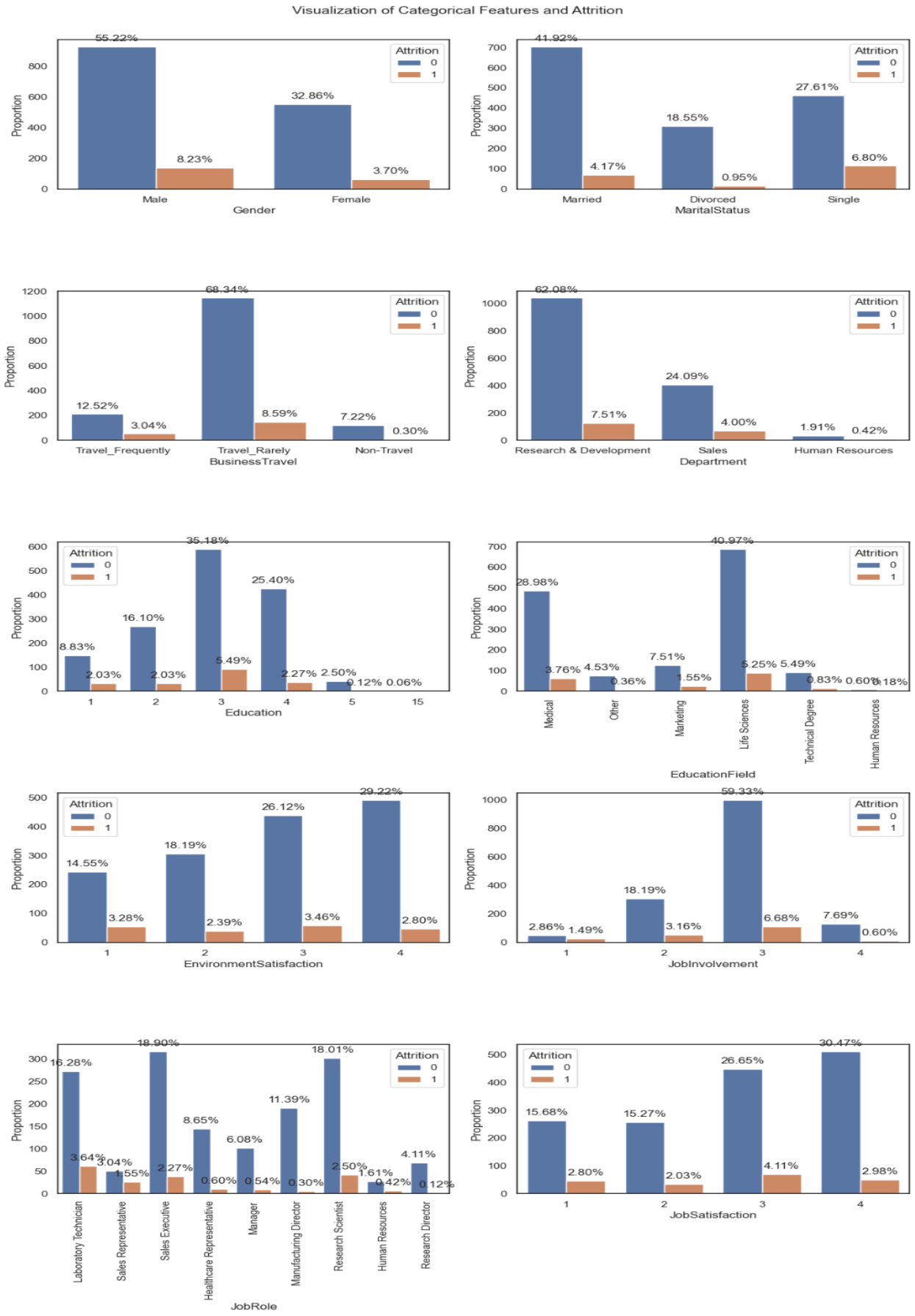
- Keeping an eye out for Target imbalance and how to include such imbalances through metrics such as recall, precision etc

IX. Appendices

Appendix 1 . Numerical Features vs Target (Boxplot)



Appendix 2. Categorical Features vs Target



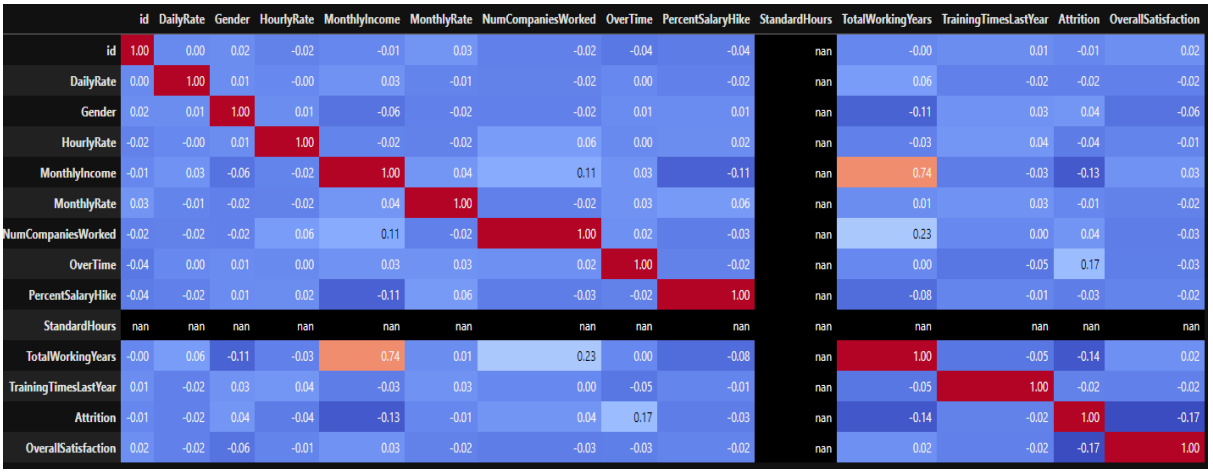
Appendix 3. Features Engineering Code

```
# some extra feature engineering
def cat_years_current_manager(value):
    if value < 5:
        return "Less than 5 years"
    elif (value >= 5) & (value <= 10):
        return "Between 5-10 years"
    else:
        return "More than 10 years"

def cat_years_at_company(value):
    if value < 5:
        return "Less than 5 years"
    elif (value >= 5) & (value <= 10):
        return "Between 5-10 years"
    elif (value >= 11) & (value <= 20):
        return "Between 11-20 years"
    elif (value >= 21) & (value <= 30):
        return "Between 21-30 years"
    else:
        return "More than 30 years"

def cat_years_current_role(value):
    if value < 5:
        return "Less than 5 years"
    elif (value >=5) & (value <= 10):
        return "Between 5-10 years"
    else:
        return "More than 10 years"

def cat_years_since_last_promotion(value):
    if value < 5:
        return "Less than 5 years"
    elif (value >=5) & (value <= 10):
        return "Between 5-10 years"
    else:
        return "More than 10 years"

def cat_distance_home(value):
    if value < 5:
        return "Less than 5 km"
```

```
    elif (value >=5) & (value <= 10):
        return "Between 5-10 km"
    elif (value >= 11) & (value <= 20):
        return "Between 11-20 km"
    else:
        return "Above 20 km"

def cat_age(value):
    if value <= 20:
        return "Between 18-20 years"
    elif (value > 20) & (value <= 30):
        return "Between 21-30 years"
    elif (value > 31) & (value <= 40):
        return "Between 31-40 years"
    elif (value > 41) & (value <= 50):
        return "Between 41-50 years"
    else:
        return "Above 50 years"

data['Cat_YearsWithCurrManager'] =
data['YearsWithCurrManager'].apply(cat_years_current_manager)

data['Cat_YearsAtCompany'] =
data['YearsAtCompany'].apply(cat_years_at_company)

data['Cat_YearsInCurrentRole'] =
data['YearsInCurrentRole'].apply(cat_years_current_role)

data['Cat_YearsSinceLastPromotion'] =
data['YearsSinceLastPromotion'].apply(cat_years_since_last_promoti
on)

data['Cat_DistanceFromHome'] =
data['DistanceFromHome'].apply(cat_distance_home)

data['OverallSatisfaction'] = ((data['EnvironmentSatisfaction'] +
data['JobSatisfaction']
                               + data['JobInvolvement'] +
data['RelationshipSatisfaction']
                                                +
data['WorkLifeBalance'])/5).astype('int32')
```

```
data['Cat_Age'] = data['Age'].apply(cat_age)

# drop columns that were used for the feature engineering

data.drop(columns=['YearsWithCurrManager', 'YearsAtCompany',
'YearsInCurrentRole',
                   'YearsSinceLastPromotion', 'DistanceFromHome',
'Age'], inplace=True)
```

Appendix 4. Correlation Matrix



	id	DailyRate	Gender	HourlyRate	MonthlyIncome	MonthlyRate	NumCompaniesWorked	OverTime	PercentSalaryHike	StandardHours	TotalWorkingYears	TrainingTimesLastYear	Attrition	OverallSatisfaction
id	1.00	0.00	0.02	-0.02	-0.01	0.03	-0.02	-0.04	-0.04	nan	-0.00	0.01	-0.01	0.02
DailyRate	0.00	1.00	0.01	-0.00	0.03	-0.01	-0.02	0.00	-0.02	nan	0.06	-0.02	-0.02	-0.02
Gender	0.02	0.01	1.00	0.01	-0.06	-0.02	-0.02	0.01	0.01	nan	-0.11	0.03	0.04	-0.06
HourlyRate	-0.02	-0.00	0.01	1.00	-0.02	-0.02	0.06	0.00	0.02	nan	-0.03	0.04	-0.04	-0.01
MonthlyIncome	-0.01	0.03	-0.06	-0.02	1.00	0.04	0.11	0.03	-0.11	nan	0.74	-0.03	-0.13	0.03
MonthlyRate	0.03	-0.01	-0.02	-0.02	0.04	1.00	-0.02	0.03	0.06	nan	0.01	0.03	-0.01	-0.02
NumCompaniesWorked	-0.02	-0.02	-0.02	0.06	0.11	-0.02	1.00	0.02	-0.03	nan	0.23	0.00	0.04	-0.03
OverTime	-0.04	0.00	0.01	0.00	0.03	0.03	0.02	1.00	-0.02	nan	0.00	-0.05	0.17	-0.03
PercentSalaryHike	-0.04	-0.02	0.01	0.02	-0.11	0.06	-0.03	-0.02	1.00	nan	-0.08	-0.01	-0.03	-0.02
StandardHours	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
TotalWorkingYears	-0.00	0.06	-0.11	-0.03	0.74	0.01	0.23	0.00	-0.08	nan	1.00	-0.05	-0.14	0.02
TrainingTimesLastYear	0.01	-0.02	0.03	0.04	-0.03	0.03	0.00	-0.05	-0.01	nan	-0.05	1.00	-0.02	-0.02
Attrition	-0.01	-0.02	0.04	-0.04	-0.13	-0.01	0.04	0.17	-0.03	nan	-0.14	-0.02	1.00	-0.17
OverallSatisfaction	0.02	-0.02	-0.06	-0.01	0.03	-0.02	-0.03	-0.03	-0.02	nan	0.02	-0.02	-0.17	1.00