NUS
National University of Singapore

DBA3803
Final Project

Group 26

Name	Matriculation Number
Ang Wen Qi Steffi	A0190683X
Gloria Hung WenMin	A0187777A
Lim Fang Yi	A0183225J

Prepared for:
Long Zhao

22 November 2020

Contents

1. Introduction

1.1 About the Dataset

The IBM attrition dataset shows the different attributes of employees which could potentially affect whether they quit the company or not. We take on the role as members of the IBM HR department who aim to find out the underlying reasons behind why employees choose to quit, and the degree to which certain factors affect one's choice to quit the company. This act of quitting is referred to as attrition in our report.

We hope to explore a myriad of questions such as 'how important is an employee's distance from home by job role that affects attrition rates' or 'how might average monthly income by education affect attrition rates'. We started off with a total of 1470 employees in our dataset and 34 features to predict on 'Attrition'. We gradually dropped the number of features to 25 as we explored the dataset. The rationale for doing so will be discussed in later sections.

After feature engineering, our number of covariates increased to 38. However, we still fulfilled the rule of thumb for Logistic Regression as a benchmark model. We are assured that there would be no severe overfitting since #observations ≥ 10 * #covariates.

1.2 Problems and Importance to HR Department

Employees are the backbone of the company and the company's performance relies heavily on the quality of its employees. Employee attrition is a potential problem for IBM. While the dataset suggests a low attrition rate, which means that the number of employees who wish to quit the company are a minority, IBM should still keep track of why employees quit. The longer an employee works for the company, the deeper the knowledge of the industry and IBM's practices. This may be a problem when employees leave IBM to work for rival companies. We also need to consider the large sunk costs from the time and money invested by IBM on training employees over the years.

We hope to gain a better appreciation for Human Resource Analytics (HR Analytics) as an essential tool in helping companies like IBM to understand employees better. We believe that through a predictive model for employee attrition rates, the HR department will gain a better insight about the factors contributing to an employee's decision to leave the company. This would enable the department to be more productive in its efforts towards catering to the needs of its employees to avoid attrition where possible.

2. Preparing our Dataset

We started off with a total of 1470 employees assessed over 34 features in our dataset.

The table below records these features.

1. **Age:** Continuous ; Integer. *Indicates employee's age*
2. **BusinessTravel:** Categorical ; Ordinal. *Indicates level of business travel by employee. 3 levels: Non-Travel, Travel_Frequently, Travel_Rarely*
3. **DailyRate:** Continuous ; Integer. *Indicates employee's daily salary*
4. **Department:** Categorical, Nominal. *Indicates employee's working department. 3 levels: Sales, Research & Development, Human Resources*
5. **DistanceFromHome:** Continuous ; Integer. *Indicates distance of employee's office from home*
6. **Education:** Integer ; Ordinal. *Indicates education level of employees. 1: 'Below College' 2: 'College' 3: 'Bachelor' 4: 'Master' 5: 'Doctor'*
7. **EducationField:** Categorical ; Nominal. *Life Sciences, Other, Medical, ,Marketing, Technical Degree, Human Resources*
8. **EmployeeCount:** Integer.
9. **EmployeeNumber:** Integer. *Unique ID that identifies the employee.*
10. **EnvironmentSatisfaction:** Integer ; Ordinal. *Indicates how satisfied an employee is with his environment. 1 'Low', 2 'Medium', 3 'High', 4 'Very High'*
11. **Gender:** Boolean. Employee's gender
12. **HourlyRate:** Continuous ; Integer. *Indicates employee's hourly salary*
13. **JobInvolvement:** Integer ; Ordinal. *Indicates how involved the employee is with his job. 1: 'Low', 2: 'Medium', 3: 'High', 4: 'Very High'*
14. **JobLevel:** Integer, Ordinal. *Indicates job level of employee.*
15. **JobRole:** Categorical ; Nominal. *Indicates employee's job position. 9 levels: 'Sales Executive', 'Research Scientist', 'Laboratory Technician', 'Manufacturing Director', 'Healthcare Representative' ,'Manager', 'Sales Representative' ,'Research Director', 'Human Resources'*
16. **JobSatisfaction:** Integer ; Ordinal. *Indicates how satisfied an employee is with his job. 1 'Low', 2 'Medium', 3 'High', 4 'Very High'*
17. **MaritalStatus:** Categorical ; Nominal. *Indicates employee's marital status. 3 Levels: 'Single', 'Married', 'Divorced'*
18. **MonthlyIncome:** Continuous ; Integer. *Indicate employee's monthly income.*
19. **MonthlyRate:** Continuous ; Integer. *Indicates employee's monthly salary*
20. **NumCompaniesWorked:** Continuous ; Integer. *Indicates the number of companies the employee has worked for so far.*
21. **Over18:** Boolean.
22. **OverTime:** Boolean.
23. **PercentSalaryHike:** Continuous ; Integer. *Indicate percentage increase in employee's monthly pay from previous year's*
24. **PerformanceRating:** Integer ; Ordinal. *Indicates employee's performance. 1 'Low', 2 'Good', 3 'Excellent', 4 'Outstanding'*
25. **RelationshipSatisfaction:** Integer ; Ordinal. *Indicates how satisfied an employee is with his relationship with colleagues. 1: 'Low', 2: 'Medium', 3: 'High', 4: 'Very High'*
26. **StandardHours:** Continuous ; Integer. *Indicate number of hours employee works weekly*
27. **StockOptionLevel:** Integer ; Ordinal. *Indicates level of stock option benefit that employee is entitled to. 5 levels: with 1 being lowest and 5 being highest*
28. **TotalWorkingYears:** Continuous ; Integer. *Indicates employee's working*

experience in his lifetime

29. **TrainingTimesLastYear:** Discrete ; Integer. *Indicates employee's frequency of undergoing training in the year before*
30. **WorkLifeBalance:** Integer ; Ordinal. *Indicates how satisfied an employee is with his worklife. 1: 'Bad', 2: 'Good', 3: 'Better', 4: 'Best'*
31. **YearsAtCompany:** Continuous ; Integer. *Indicates employee's working duration in IBM*
32. **YearsInCurrentRole:** Continuous ; Integer. *Indicates employee's working years in his current role in IBM*
33. **YearsSinceLastPromotion:** Continuous ; Integer. *Indicates number of years from the previous promotion*
34. **YearsWithCurrManager:** Continuous ; Integer. *Indicates number of years one has worked with his current Manager*

The label that we will be using is:

1. **Attrition:** Boolean. *Indicates whether employee has quit IBM*

2.1 Exploratory Data Analysis

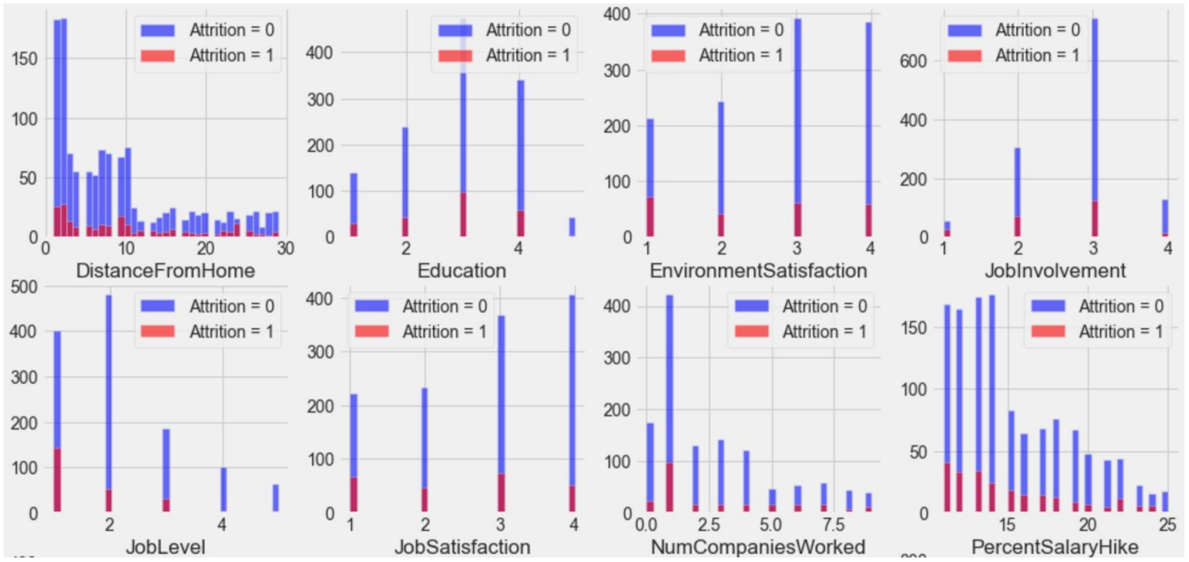
We dropped the following 8 variables which we felt would not value-add to our predictive model:

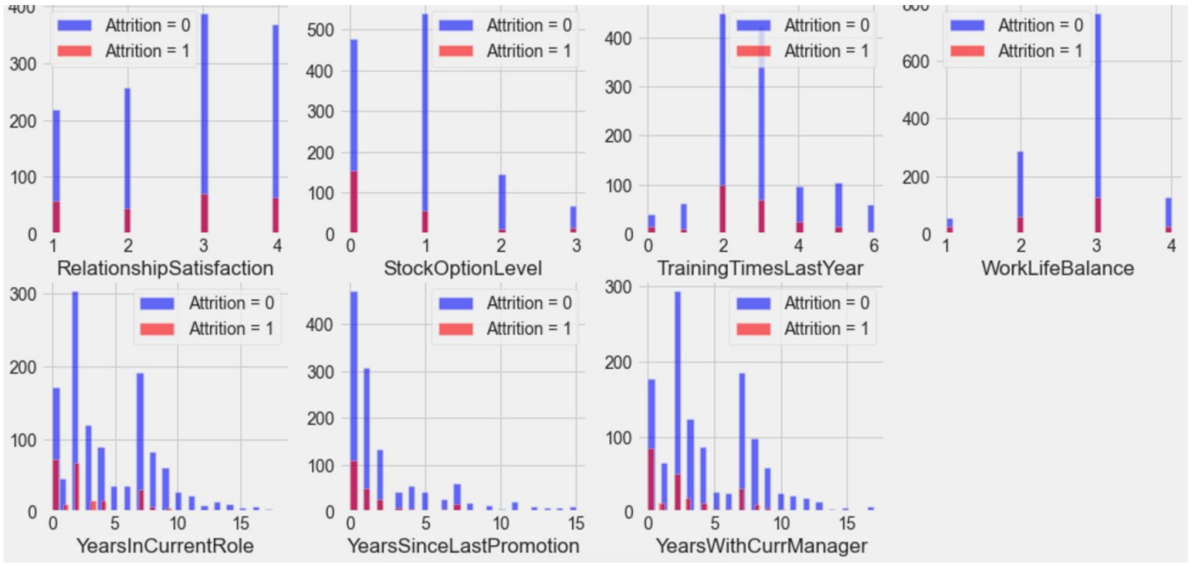
1. **DailyRate, HourlyRate, MonthlyRate** - *Did not tally with the monthly income and standard hours*
2. **Over18** - *Essentially a constant since all values = True*
3. **EmployeeNumber** - *Only an ID which identifies the employee*
4. **PerformanceRating** - *Ratings were essentially level 3 and 4 only, which were 'Excellent' and 'Outstanding' which does not tell us much*
5. **EmployeeCount** - *Essentially a constant since all values = 1*
6. **StandardHours** - *Essentially a constant since all values = 80*

2.1.1 Distributions and Categories

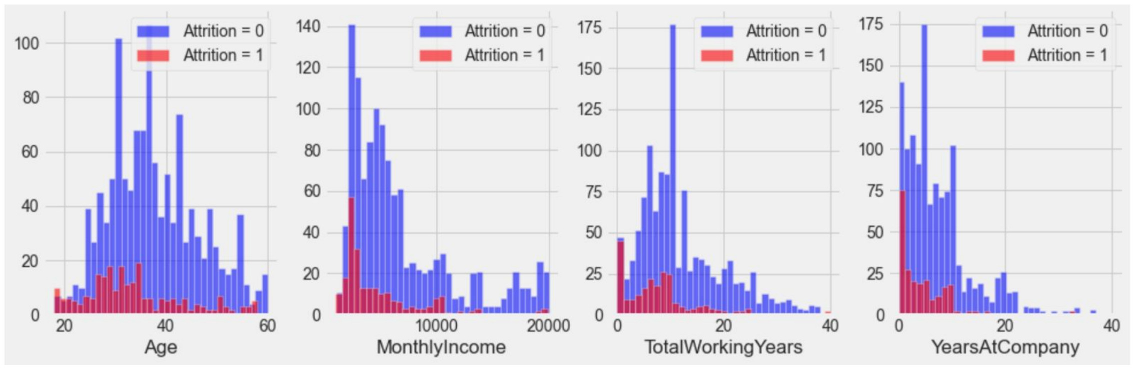
We also sought to understand the distributions and categories of each feature after stratifying them by **Attrition** = 0 and **Attrition** = 1. This helps us to prepare for our feature encoding and scaling methods.

The graphs plotted below were initially filtered as discrete variables. However, we can see that features such as **PercentSalaryHike** are actually continuous and have to be scaled. Without plotting these graphs, we would not have known our subsequent steps for data pre-processing.

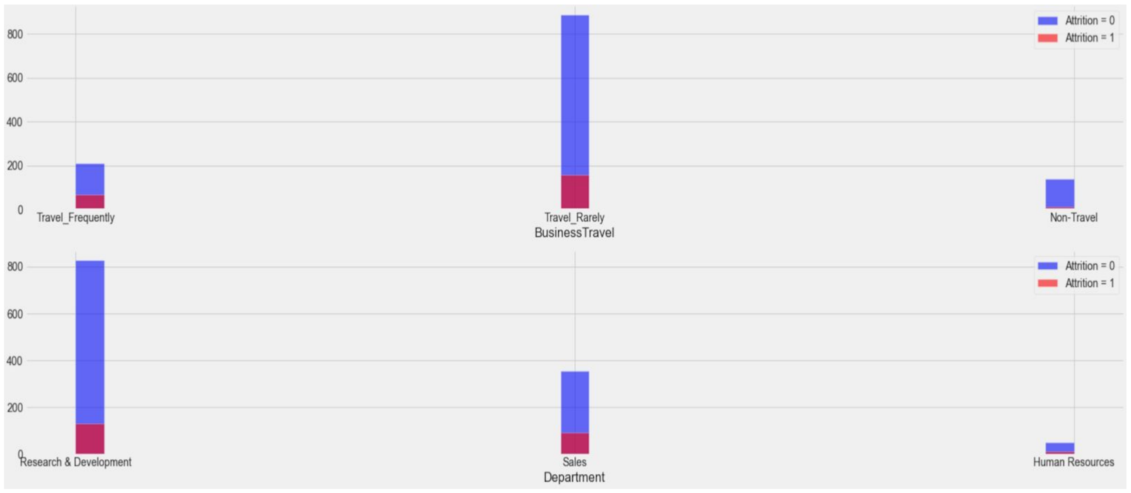




On the other hand, there are also categories which we already expected to be numerical continuous. Hence, we would have to scale them. These are **Age**, **MonthlyIncome**, **TotalWorkingYears**, **YearsAtCompany**.

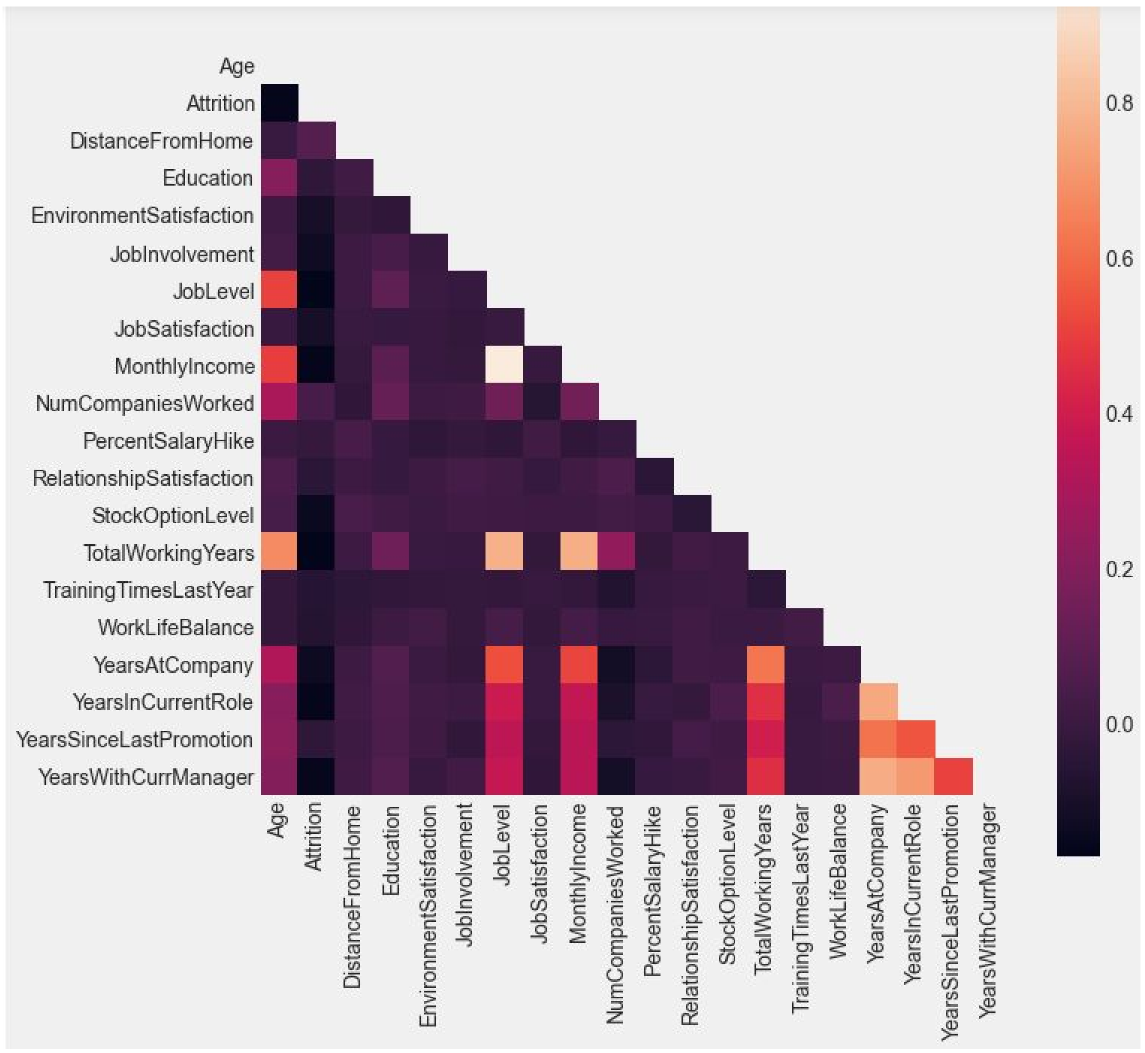


We also plot graphs for the categorical features to find out whether there are any ordinal categorical features which we might need to account for during feature encoding. We found one, which is **BusinessTravel**.



2.1.2 Correlation Matrix

We did a correlation plot after dropping the irrelevant variables.



From our correlation matrix, **MonthlyIncome** and **JobLevel** are the 2 variables that seem to be almost perfectly correlated. It makes sense that a person who has a higher **JobLevel** would be receiving a higher **MonthlyIncome**. We decided to remove **MonthlyIncome** from our dataset, and we continue to keep **JobLevel**.

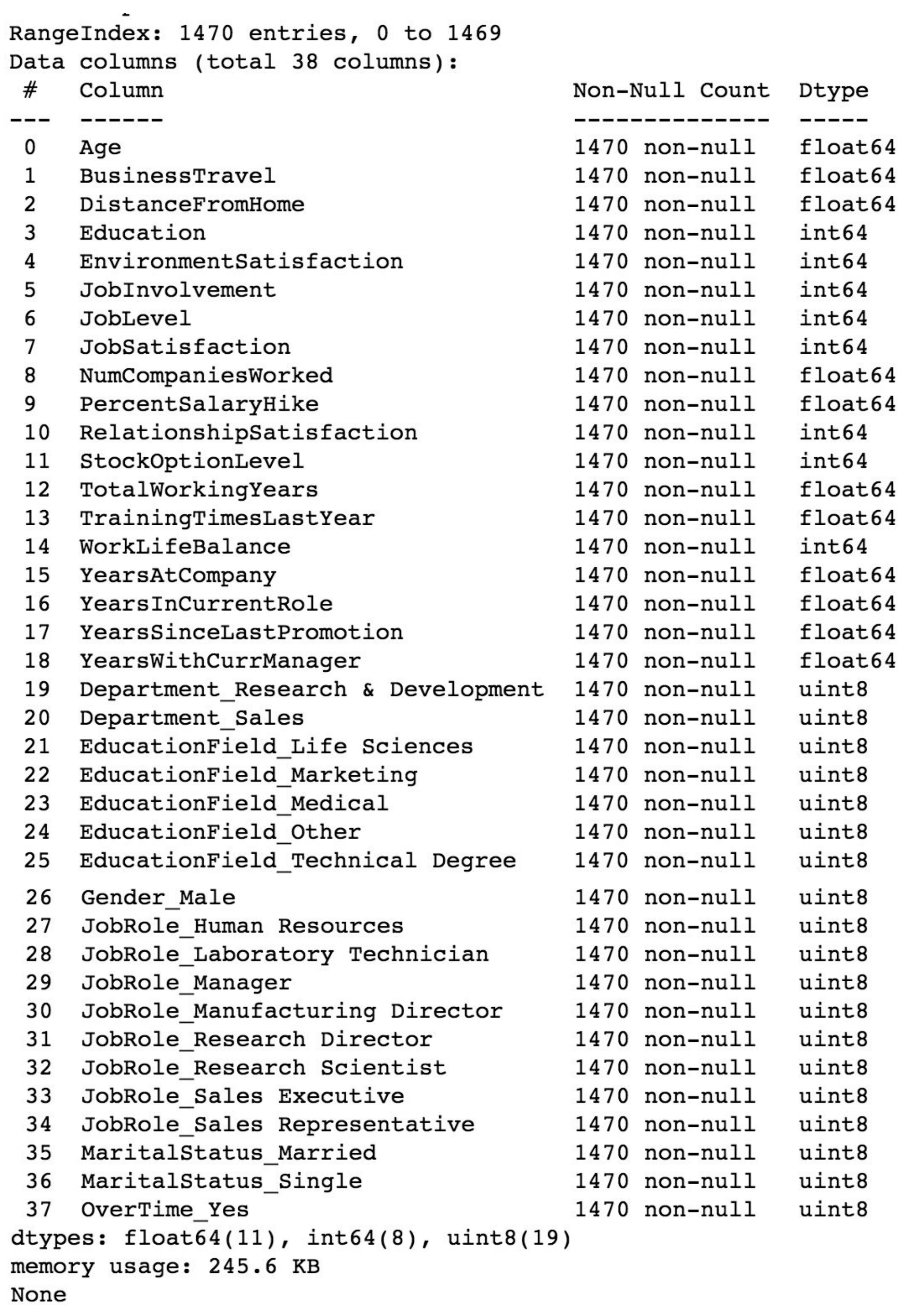
After we drop the 9th variable of **MonthlyIncome**, we are left with 25 features.

1. **MonthlyIncome:** Continuous ; Integer. *Indicate employee's monthly income.*

2.2 Data Pre-processing

We applied `StandardScaler()` to our continuous variables. `StandardScaler()` transforms the variable to approximate a standard normal distribution with mean = 0 and variance = 1.

We also performed Ordinal Encoding `OrdinalEncoder()` on the BusinessTravel category since it contains ordinal data and One-Hot Encoding `pd.get_dummies()` on non-ordinal categorical variables. We removed **Attrition** which is our target and applied `LabelEncoder()` to it. As a result, we have 38 covariates.



```
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 38 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Age                                1470 non-null   float64
 1   BusinessTravel                     1470 non-null   float64
 2   DistanceFromHome                   1470 non-null   float64
 3   Education                          1470 non-null   int64
 4   EnvironmentSatisfaction            1470 non-null   int64
 5   JobInvolvement                     1470 non-null   int64
 6   JobLevel                           1470 non-null   int64
 7   JobSatisfaction                    1470 non-null   int64
 8   NumCompaniesWorked                 1470 non-null   float64
 9   PercentSalaryHike                  1470 non-null   float64
 10  RelationshipSatisfaction           1470 non-null   int64
 11  StockOptionLevel                   1470 non-null   int64
 12  TotalWorkingYears                  1470 non-null   float64
 13  TrainingTimesLastYear              1470 non-null   float64
 14  WorkLifeBalance                    1470 non-null   int64
 15  YearsAtCompany                     1470 non-null   float64
 16  YearsInCurrentRole                 1470 non-null   float64
 17  YearsSinceLastPromotion            1470 non-null   float64
 18  YearsWithCurrManager               1470 non-null   float64
 19  Department_Research & Development  1470 non-null   uint8
 20  Department_Sales                   1470 non-null   uint8
 21  EducationField_Life Sciences       1470 non-null   uint8
 22  EducationField_Marketing           1470 non-null   uint8
 23  EducationField_Medical             1470 non-null   uint8
 24  EducationField_Other               1470 non-null   uint8
 25  EducationField_Technical Degree    1470 non-null   uint8
 26  Gender_Male                        1470 non-null   uint8
 27  JobRole_Human Resources            1470 non-null   uint8
 28  JobRole_Laboratory Technician      1470 non-null   uint8
 29  JobRole_Manager                    1470 non-null   uint8
 30  JobRole_Manufacturing Director     1470 non-null   uint8
 31  JobRole_Research Director          1470 non-null   uint8
 32  JobRole_Research Scientist         1470 non-null   uint8
 33  JobRole_Sales Executive            1470 non-null   uint8
 34  JobRole_Sales Representative       1470 non-null   uint8
 35  MaritalStatus_Married              1470 non-null   uint8
 36  MaritalStatus_Single               1470 non-null   uint8
 37  OverTime_Yes                       1470 non-null   uint8
dtypes: float64(11), int64(8), uint8(19)
memory usage: 245.6 KB
None
```

We used `train_test_split()` twice so that we have a training set size = 72.25%, a validation set size = 12.75% and a test set size = 15% based on our pre-processed dataset.

3. <u>Models and Performance</u>

3.1 Overview of Models Used and Process of Hyperparameter Tuning

We identified our problem to be a classification problem as our target **Attrition** is discrete categorical. We tried 4 different models - Random Forest, LightGBM, Logistic Regression and Support Vector Machine. Logistic Regression is our benchmark model which uses traditional statistical methods. The other models utilise Machine Learning algorithms. We used LightGBM which applies the Gradient Boosting trees framework which is known to be efficient to run with low memory usage[1].

Since we experienced imbalanced classification of our target **Attrition**, we used `SMOTE()` in our pipeline to solve the imbalance and oversample the minority class - that employee actually quits. We fit the training set in a 5-fold Grid Search Cross-Validation via `GridSearchCV()` to optimize the hyperparameters of each of our 4 models.

Grid Search partitions the parameter space into small grids and check values of parameters on those intersection points. To prevent information leakage (data points appear in both subset of training and hold-out sets during Cross Validation), the resampling of the minority class of **Attrition** = 1 is only done on the training set after splitting by 5-fold Grid Search Cross-Validation. This prevents the subset of training data from duplicating entries with the hold-out set and causing our chosen model to see some of the hold-out data inside the training set during Grid Search.

The trade-off that we had to deal with is the fact that it is computationally expensive to run `GridSearchCV()`. This is especially so for the Random Forest algorithm. To reduce our running time, we reduced the search space for each hyperparameter before allowing `GridSearchCV()` to optimize across all hyperparameters.

3.2 Overview of Performance Measures Used

We considered the confusion matrix performance measures of Precision, Recall, F_1 score, Specificity, AUC-ROC, Accuracy. Eventually, we decided to choose the **AUC-ROC** score as our performance measurement for comparison. However, we also paid attention to the **Recall** measure which is equivalent to the True Positive Rate.

[1] LightGBM Documentation https://lightgbm.readthedocs.io/en/latest/

	Actual Attrition = 1	Actual Attrition = 0
Predicted Attrition = 1	True Positive: TP Employees who quit & predicted to quit	False Positive: FP Employees who do not quit & predicted to quit
Predicted Attrition = 0	False Negative: FN Employees who quit & predicted not to quit	True Negative: TN Employees who do not quit & predicted not to quit

The confusion matrix gives us the metrics as follows:

1) Precision = $\frac{TP}{TP+FP}$

This measures the proportion of positive predictions who actually quit.

2) **Recall** (a.k.a. True Positive Rate, Sensitivity) = $\frac{TP}{TP+FN}$

This measures the proportion of employees who quit that are correctly detected by the classifier models. It is an important metric for us because we do not want to waste business dollars on a model that wrongly detects employees who quit.

3) F_1 score = 2 x $\frac{Precision \times Recall}{Precision + Recall}$ = $\frac{TP}{TP+\frac{FN+FP}{2}}$

This metric combines precision and recall and compares between 2 or more classifiers. The metric is a harmonic mean of precision and recall that gives more weight to low values. a classifier will get a high F_1 score if both recall and precision are high.

In our case, we do not require a high F_1 score. We only require models to have fairly high recall even if precision is low. The HR department can have a snapshot of the employees who do intend to quit.

4) Specificity (a.k.a. True Negative Rate) = $\frac{TN}{TN+FP}$

This metric measures the proportion of employees that are correctly classified as not going to quit. This is not the most useful measure in our context.

5) **AUC-ROC** (Area under the Receiver Operating Characteristic Curve)

The ROC curve plots the True Positive Rate (TPR a.k.a. Recall) against the False Positive Rate (FPR = 1 - True Negative Rate). Each point plotted represents the TPR and FPR for a threshold value whose range lies in [0, 1]. Many points representing the TPR and FPR for different threshold values are plotted and joined to form the ROC curve. The curve shows the performance of a classification model at all threshold values.

The AUC measures the area under the ROC curve, which is the probability that a model ranks a random positive example more highly than a random negative one. In our case, AUC will rank an employee will quit more highly than one who is not going to quit.

A straight line where FPR = TPR represents a purely random guess and AUC = 0.5. A perfect classifier has a curve with AUC = 1. Our classifier should have a curve with AUC close to 1.

6) Accuracy

We chose not to use accuracy as the dataset is heavily skewed. There are many more entries with **Attrition** = 0 than **Attrition** = 1. High accuracy means that models will pick out employees whose **Attrition** = 0 but fail to pick out employees of **Attrition** = 1. This fails to achieve the objective of finding out which employees are likely to quit.

3.3 Models

3.3.1 Logistic Regression

Logistic Regression is a binary classifier that is used to estimate probability that an instance belongs to a particular class. When the estimated probability for a particular instance is greater than 50%, the model will predict that instance belongs to the positive class, labeled "1") Else, the model predicts that instance belongs to the negative class, labeled "0".

We intend to use logistic regression as a benchmark model against the other 3 models since it has good interpretability.

The logistic regression model computes a weighted sum of covariates and a bias term and returns the logistic of this weighted sum, as seen in the following equation:

$$log\left(\frac{p}{1-p}\right) = f(\mathbf{x}) = b_0 + b_1x_1 + \cdots + b_rx_r$$

The logistic regression function $p(\mathbf{x})$ is the sigmoid function of $f(\mathbf{x})$:

$$p(\mathbf{x}) = \frac{1}{(1 + exp(-())).}$$

The function $p(\mathbf{x})$ is the predicted probability that the output (employee's attrition) for a given $\mathbf{x}$ (i.e. employee's attributes) is equal to 1, and $1 - p(\mathbf{x})$ is the probability that the output is 0.

In summary, Logistic Regression measures the relationship between our target **Attrition** and the 38 covariates by estimating probabilities using it's underlying logistic function and transforming to a binary value in order to make predictions. It is applicable as our target variable, **Attrition**, is a binary variable. Furthemore, by removing highly correlated variables in our datasets prior to fitting to models, we attempted to remove or minimize multicollinearity which Logistic Regression assumes to be absent. We also assume observations in the data are independent from one another.

We use Logistic Regression as a benchmark model against the other 3 models since it has good interpretability[2].

Pipeline Inputs	`'sampling', SMOTE(random_state = 42)` `'lr', LogisticRegression(penalty = 'l2', random_state = 42, n_jobs = -1)`
GridSearchCV() Inputs	`'lr__C' : [0.001, 0.01, 0.1, 1, 10, 100, 1000]`
Optimal Hyperparameters	`'lr__C': 1`

	AUC	**Recall**
Performance on training set	0.867	0.766
Performance on validation set	0.839	0.833

There are no signs of overfitting under the classic Logistic Regression (i.e. model is able to generalize well onto unseen data; in sample error and out-of-sample error difference not too wide) as performances on training and validation set have a small gap.

3.3.2 Random Forest

Random Forest is an ensemble method that aggregates the predictions of individual Decision Trees. Each Decision Tree is trained on a different random subset of the training data set of employees. Bagging is incorporated to sample with replacement a

[2] Logistic Regression Pros and Cons
https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

subset of the training data set.

We chose Random Forest over Decision Tree because of the “wisdom of the crowd” idea, whereby gathering the aggregation of more answers is better than just one answer. Decision Tree tends to overfit with high variance and low bias on training dataset and requires pruning. That means Decision Tree learns patterns in the training data set well but cannot generalize well on unseen data, achieving worse performance.

Pipeline Inputs	`'sampling', SMOTE(random_state = 42)`
	`'clf', RandomForestClassifier(criterion = 'gini', random_state = 42, n_jobs = -1)`

GridSearchCV() Inputs	`'clf__max_depth': [2, 4, 6],`
	`'clf__max_features': ['sqrt', 'log2', None],`
	`'clf__min_samples_leaf': [30, 40, 50],`
	`'clf__n_estimators': [500, 1000, 1500]`

Optimal Hyperparameters	`'clf__max_depth': 6,`
	`'clf__min_samples_leaf': 30,`
	`'clf__max_features': 'log2',`
	`'clf__n_estimators': 1000`

	AUC	**Recall**
Performance on training set	0.868	0.754
Performance on validation set	0.755	0.6

Unfortunately, the random forest model seemed to have overfitted as the performance measures under training and validation sets are rather wide. This would eliminate Random Forest in our candidate set of models.

Random Forest also measures the relative importance of each feature. Our plot indicates that **TotalWorkingYears**, **OverTime_Yes**, **Age**, **YearsAtCompany**, **YearsWithCurrManager** are our top 5 most important features.

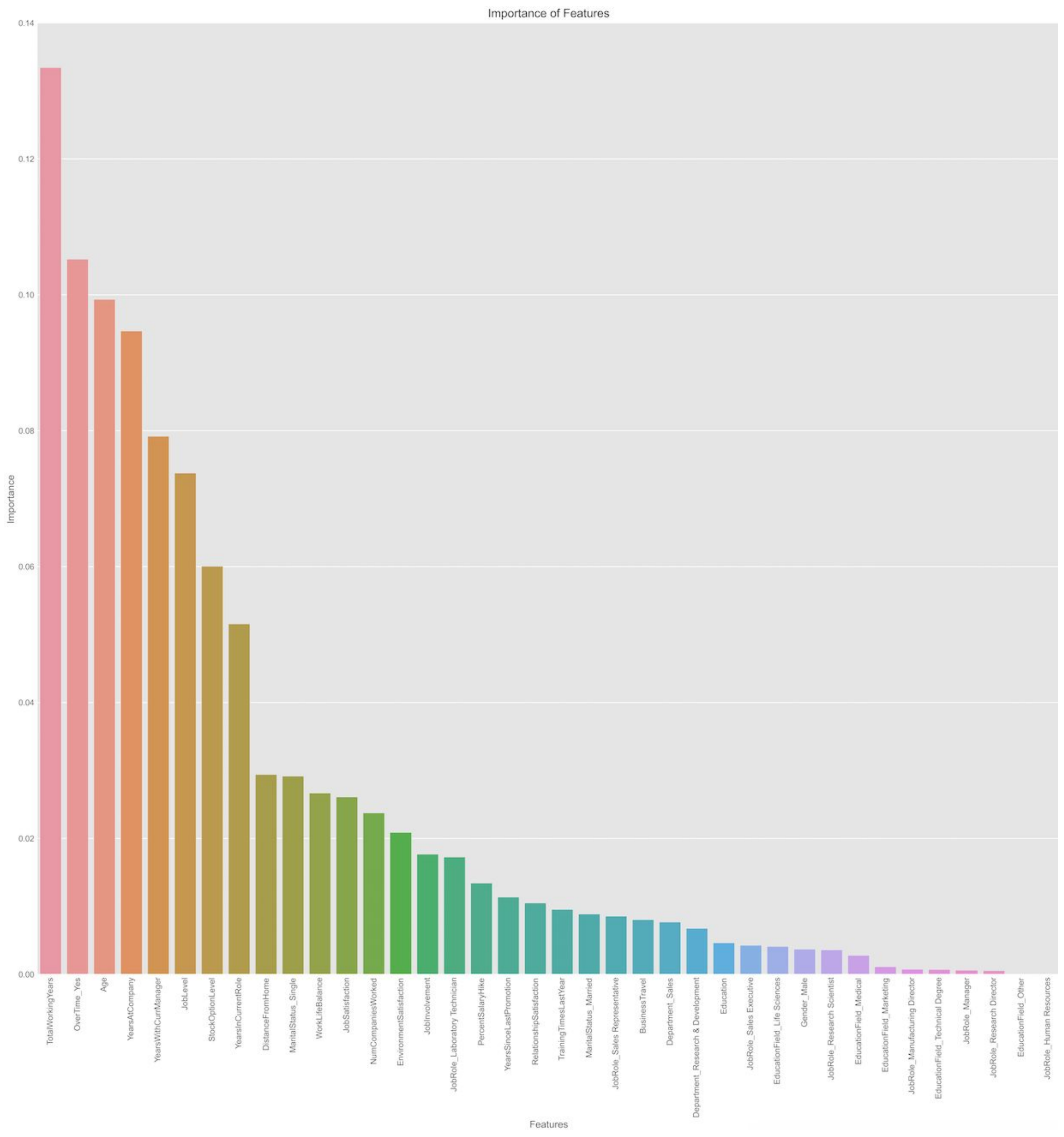
Intuitively, they make sense. **TotalWorkingYears**, **Age**, **YearsAtCompany** could mean that older employees who have worked for many years quit because they wish to retire. A higher value for **YearsWithCurrManager** is also an indicator of a more experienced and older employee who might not be able to advance their career further at IBM, or is about to retire. **OverTime_Yes** indicates that overworked workers are more likely to quit, and this is in line with existing HR knowledge on reasons why employees quit[3].

On the other hand, features of little importance are mostly related to **EducationField** and **JobRole**. This may indicate that the hiring process of IBM is rigorous enough such that most employees are in a relevant job role related to what they studied and which they enjoy working on. Hence, these are not features that lead to an employee's choice to quit IBM.

TotalWorkingYears	**0.129722 : Most Important Feature**
OverTime_Yes	0.098244
YearsAtCompany	0.097317
Age	0.095687
YearsWithCurrManager	0.076972
JobLevel	0.072326
StockOptionLevel	0.062694
YearsInCurrentRole	0.062007
DistanceFromHome	0.030708
MaritalStatus_Single	0.027082
WorkLifeBalance	0.025927
JobSatisfaction	0.025061
NumCompaniesWorked	0.023240
EnvironmentSatisfaction	0.021153
JobRole_Laboratory Technician	0.019871
JobInvolvement	0.017391
PercentSalaryHike	0.014390
JobRole_Sales Representative	0.010346
YearsSinceLastPromotion	0.010179
RelationshipSatisfaction	0.010058
TrainingTimesLastYear	0.009239
MaritalStatus_Married	0.008206
Department_Sales	0.008202
BusinessTravel	0.007382
Department_Research & Development	0.007366
JobRole_Sales Executive	0.006364
Education	0.004144
JobRole_Research Scientist	0.003942
EducationField_Life Sciences	0.003612
Gender_Male	0.003590
EducationField_Medical	0.003348
EducationField_Technical Degree	0.001231
JobRole_Research Director	0.001035
EducationField_Marketing	0.000894
JobRole_Manufacturing Director	0.000542

[3] Overworked Employees Quit
https://www.forbes.com/sites/ashiraprossack1/2018/11/30/6-reasons-your-best-employees-quit/?sh=34a0a5512d74

JobRole_Manager	0.000527
EducationField_Other	0.000000
JobRole_Human Resources	**0.000000 : Least Important Feature**



Unfortunately, Random Forest does not have the best AUC performance so we will have to try out other models.

3.3.3 Support Vector Machine

A Support Vector Machine (SVM) performs linear and non-linear classification. SVM can also detect outliers. In a linear SVM classification, the classes of labels are linearly separable by a hyperplane in n-dimensions. For example, n = 2 dimensions would refer to a line and n = 3 dimensions would refer to a plane. The SVM classifier separates the classes clearly by creating decision boundaries that maximize the margins between employees who quit and do not quit.

Pipeline Inputs	`'sampling', SMOTE(random_state = 42)`
	`'svc', SVC()`

GridSearchCV() Inputs	`'SVC__C': [0.1, 0.5, 1, 5, 10, 50, 100],`
	`'SVC__gamma': ['scale', 'auto'],`
	`'SVC__kernel': ['linear', 'rbf', 'poly', 'sigmoid']`

Optimal Hyperparameters	`'SVC__C': 1,`
	`'SVC__gamma': 'scale',`
	`'SVC__kernel': 'linear'`

	AUC	**Recall**
Performance on training set	0.862	0.760
Performance on validation set	0.839	0.767

Hyperparameter tuning suggested that no kernel trick was required.

The SVM algorithm has a similar AUC performance with our benchmark model Logistic Regression. However, the SVM's Recall performance on validation set is significantly worse than the Logistic Regression. This means that the proportion of employees who quit that are correctly detected by SVM is much lower. Hence, we conclude the SVM is outclassed by the Logistic Regression model.

3.3.4 Gradient Boosting Method (Light Gradient Boosted Machine, LightGBM)

Boosting is an ensemble method that combines several weak learners into strong learners by adding predictors first. New predictors are then fitted to the residuals made by the previous predictor, and this process is repeated. When boosting, we aim to minimize the loss function.

The Gradient Boosted Decision Trees are complex trees that avoid overfitting. It relies on a sequence of simple trees. Each simple tree has a prediction ability marginally better than random guessing and learns the most noticeable pattern in the current iteration. Since simple trees tend to underfit, the sequence of simple trees would eventually help to avoid overfitting.

LightGBM has efficient implementation and is fast in training compared to `sklearn`’s GradientBoostingClassifier. LightGBM has also been shown to minimize more loss than other Gradient Boosting algorithms by growing leaves with higher reductions in loss.

Unlike our other 3 models, we did not use `SMOTE()` to resample the unbalanced data as LightGBM allows for the parameter `‘is_balanced’` which performs the same function.

Pipeline Inputs	`'Classifier', LGBMClassifier(objective = 'binary', random_state = 42, n_jobs = -1)`

GridSearchCV() Inputs	`'lgbmclassifier__learning_rate': [0.2, 0.25, 0.3],`
	`'lgbmclassifier__n_estimators': [700, 1000, 1300],`
	`'lgbmclassifier__num_leaves': [2, 4, 6],`
	`'lgbmclassifier__colsample_bytree': [0.4, 0.6, 0.8],`
	`'lgbmclassifier__is_unbalanced': ['true']`

Optimal Hyperparameters	`'lgbmclassifier__learning_rate': 0.2,`
	`'lgbmclassifier__n_estimators': 700,`
	`'lgbmclassifier__num_leaves': 2,`
	`'lgbmclassifier__colsample_bytree': 0.8,`
	`'lgbmclassifier__is_unbalanced': 'true'`

	AUC	**Recall**
Performance on training set	0.923	0.825
Performance on validation set	0.862	0.733

If we look at AUC difference between training and validation set purely, we conclude there is not much signs of overfitting by the LightGBM algorithm. Purely AUC wise, However, the recall did not perform as well on the validation set compared to the Logistic Regression model.

3.4 Chosen Model and Performance

We compared the validation set ROC_AUC scores amongst the 4 models, with Logistic Regression as the benchmark model. We realized LightGBM performed the best with a AUC score of 0.862.

	AUC	**Recall**
Logistic Regression	0.839	0.833

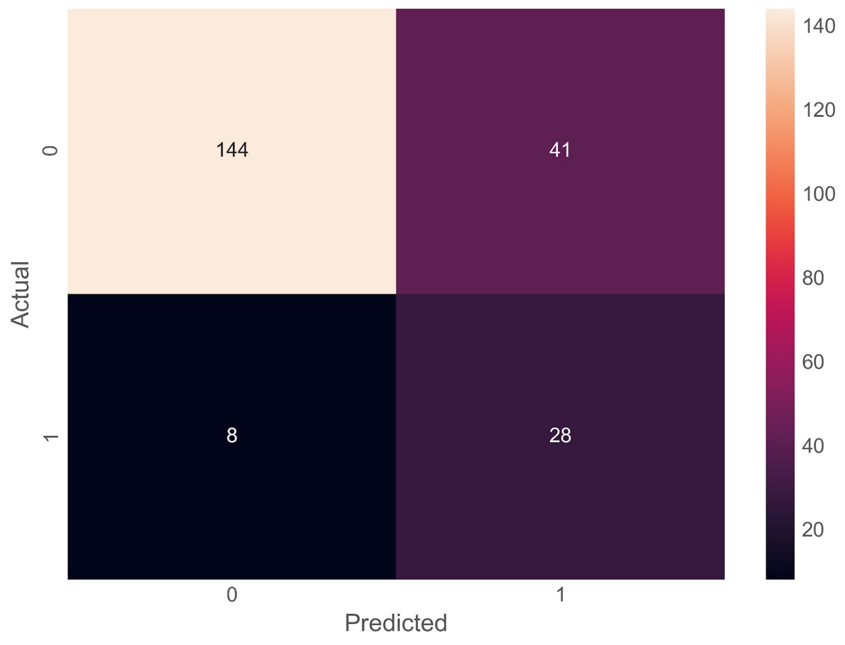
Random Forest	0.755	0.6
Support Vector Machine	0.839	0.767
LightGBM	**0.862**	**0.733**

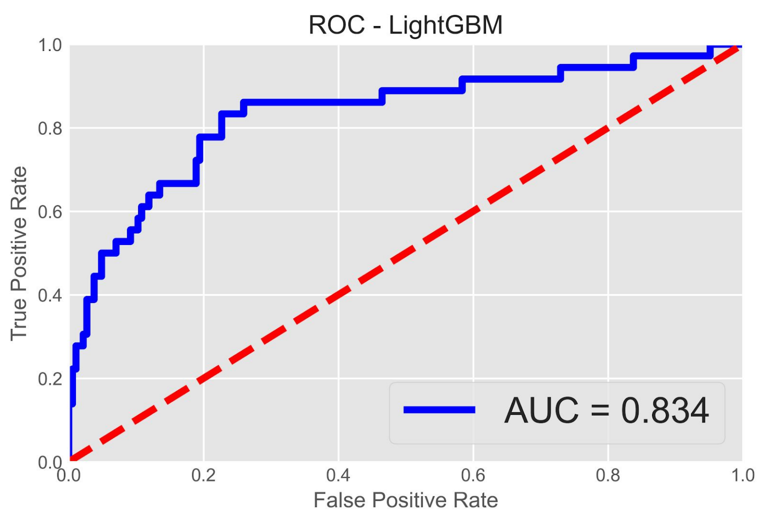
Hence, we chose LightGBM as our model.

We implemented the optimal hyperparameters recommended by `GridSearchCV()` in Section 3.3.4 Gradient Boosting Machine (LightGBM) on our test set.

	AUC	**Recall**
Performance on test set	0.834	0.778

Again, with test set performance not differing too much from validation set performance, we conclude there were not severe overfitting concerns by the LightGBM model. Hence, it should do reasonably well in generalizing to new instances (new employee data that model has not been trained on) - detecting employees who are likely to quit fairly well.

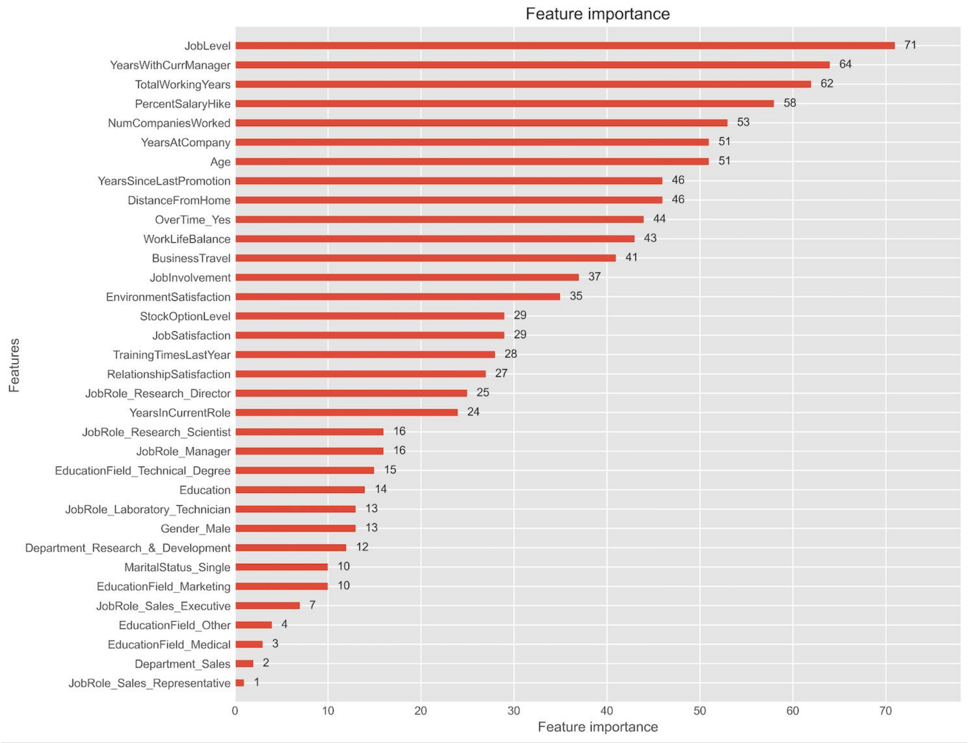




4. Feature Importance and Targeted Recommendations

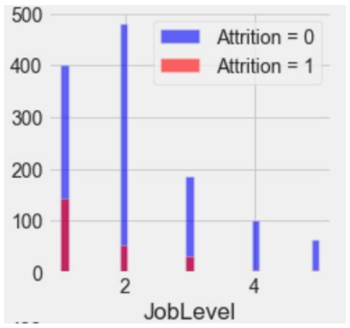
We were also able to visualise the most important features as provided by `lightgbm.plot_importance()`. If we compare this plot to the one we did in using our Random Forest model in Section 3.3.2, we get rather similar results for our top features, although the rankings may have changed by 1 or 2 positions.

In the larger scheme of things, our final model of LightGBM is indeed doing a relatively good job at predicting the factors that may result in an employee's decision to quit.



Given that our predictive model seems to work well, we can then allocate our HR budget efficiently to plan out initiatives to target the top potential features that lead to employee attrition. We decided to focus on our top 3 features **- JobLevel**, **YearsWithCurrManager** and **TotalWorkingYears**.

JobLevel

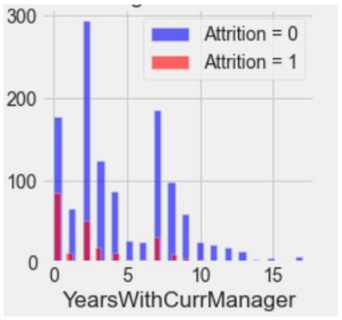


We observed that **Attrition** is high when **JobLevel** is low. When a job is too easy, employees may lose interest and fulfilment. Hence, employees lose interest and fulfilment in their line of work. This nudges them towards other companies that allow these employees to take on more challenging work. Another take on this observation is that when **JobLevel** is low, **MonthlyIncome** is low. This corresponds to our findings in our correlation matrix in Section 2.1.2. This might signal to an individual that they should leave to find a higher level job with a higher pay instead of staying at IBM.

Our first recommendation would be for IBM to set higher goals and reward employees with higher bonuses when goals are achieved. Giving more challenging work can increase motivation and overall work productivity for IBM.

Our second recommendation would be for IBM to ensure appropriate job allocation for different workers. If an employee seems to be achieving their set targets, IBM should consider promoting him/her. This also signals to other employees that they will be rewarded accordingly for the work that they do. This will also ensure that employee’s strengths are being used in the right departments to value-add to the company.

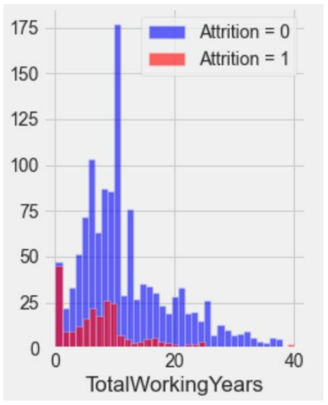
YearsWithCurrManager



We observed that **Attrition** is high when **YearsWithCurrManager** is low. When employees are unable to work well with their current manager, this causes workplace unhappiness that might encourage them to leave the job. However, if more years are spent with the current manager, the good camaraderie between employee and manager might reduce the likelihood of an employee quitting IBM.

Our recommendation is for IBM to pick the right managers. According to Harvard Business Review[4], the qualities of a good manager include identifying and capitalizing on each person's uniqueness, motivating colleagues, giving recognition to team members and tailoring to the different learning styles of each individual they lead. Employees are more likely to stay on with managers that they can build deep and strong bonds with.

TotalWorkingYears



We observed that **Attrition** is high when **TotalWorkingYears** is low. This might be because the employee has yet to form a strong attachment to IBM, hence an employee is more willing to quit the company with fewer strings attached. We also observed that when **TotalWorkingYears** = 0, **Attrition** is almost always definite. This indicates that employees almost always decide whether they want to stay or leave within a year of entering IBM.

[4] HBR: Why People Quit Their Jobs https://hbr.org/2018/01/why-people-really-quit-their-jobs

We recommend that IBM pay more attention to new hires. Before employing a new hire, the onboarding process should enable these new hires to be well-aware of the company's culture, working prospects, and job expectations. This can reduce any unnecessary conflict from arising. Also, there should also be stricter contractual agreements to demotivate the employees from leaving after a short period of time. This will decentive employees from job hopping within a year or two of entering IBM.

5. Limitations and Opportunities

5.1 Limitations

Our approach to tune hyperparameters before performing `GridSearchCV()` was to run each hyperparameter space on selected values that might improve the AUC score until further increases do not cause any of our model performance to change rapidly. After going through these values, we then tune our selected range of hyperparameters via `GridSearchCV()`. Admittedly, this was not the most time efficient way of obtaining the optimal hyperparameters.

We could have considered the Bayesian Optimization approach which is a more systematic way to find a good combination of hyperparameters. This approach is much more efficient in time-complexity. It also keeps track of past evaluation results by utilizing hyperparameter values that have already been sampled to form a probabilistic model that maps these values to a probability of a score on the objective function[5].

In addition, we assume the data of all employees are independent and identically distributed. We assume these employees in the data are the population of IBM employees. With no specific details on how the data is collected, these assumptions may not be true.

Lastly, we considered that the process of data collection impacts the predictions and feature importance plot. Variables like **JobInvolvement** and **EnvironmentSatisfaction** are self-reported by employees. This would result in self-reporting bias as they might fear the repercussions of giving a negative answer. Hence, these variables may be over-reported to be positive.

5.2 Opportunities

Since we did not give a monetary value which is at the company's discretion to how much loss is prevented by successfully retaining an employee on average and the cost of speaking to employees who are predicted to quit, we do not know how much benefit the company would get from deploying this model in production.

[5] How Bayesian Model Optimization Works
https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f

As more data about employees are collected, big data frameworks and techniques such as Hadoop, MapReduce could be deployed to process the data set and train models by an ‘online-system’ over several machines.

We believe that the findings from this dataset presents valuable insights for IBM to find out why employees may quit. From our findings, the HR department can go about considering new initiatives or revising existing policies to retain valuable employees. In the long run, this will help IBM save costs, minimizing loss of sunk cost in training top talents and continue to make it a productive company.

The framework that we have created does not just have to be used in IBM alone. It would be a useful predictive tool framework for other companies to use, with their own employee dataset.

6. <u>Conclusion</u>

This has been an insightful project into the potentials of HR Analytics as a predictive tool. The model can not only be used to predict whether an employee would quit but also provide feature importances which suggest to the HR department on what efforts should be channeled to which kind of attributes of employees at priority, by using `feature_importances()`.

There is much potential to disrupt how the HR department runs. We hope that our findings will make IBM’s HR more productive in identifying the factors closely related to employees that are likely to quit, and in planning initiatives to target these factors.