Introduction

In this document I present a summary of the main elements of my capstone project 1. Please refer to the accompanying Jupyter notebook for additional comments, results and code.

Problem and Dataset

The goal of this project is to build a model that could predict employees at risk for turnover. The dataset contains information about employees of a certain company classified into two groups: Group of employees that left the company and a group of employees that are still working for the company. The company is facing high turnover.

Definitions of Positive and Negative Classes

Positive Class: This class consists of employees who left the company and is represented using boolean value '1' throughout the project documentation.

Negative Class: This class consists of employees that did not leave the company and is represented using boolean value '0' throughout the project documentation.

Quantitative and Continuous

- Satisfaction_Level
- Last_Evaluation
- Average_Monthly_Hours

Quantitative and Discrete

- Average_Monthly_Hours
- Time_Spend_Company
- Number_of_Projects

Qualitative and Binary

- Work_Accident
- Left
- Promotion_Last_5_Years

Qualitative and Nominal (Unordered)

- Department

Qualitative and Ordinal Ordered)

- Salary

I used methods like head(), tail () to preview the data and to be able to define data columns into the above categories.

Approach

1. Data Cleaning

This step is critical in Identifying any gaps and abnormalities in data. Once identified any missing or abnormal data needs to be cleaned so that the data is ready for analysis, visualization and model building. In the dataset there are about 15000 rows of data with employee information. Firstly, I made sure the data met the criteria used to define 'Tidy data'

- Each variable must have its own column.
    - All metrics associated with an employee are in their own column
- Each observation must have its own row.
    - Each employee has a separate row
- Each value must have its own cell.
    - Each feature about an employee in this dataset is represented as a single value in its own cell

The next step I took is to check for missing values. DataFrame's info() methods provides a quick summary of the non-null (NaN) values in the dataframe. This method still does not provide information on spaces or other invalid character types.

For qualitative variables and quantitative variables with discrete values, I used dataframe column's value_counts() method to see all the unique values within a given column. This method helps understand if there are any abnormal or invalid values inside the data. This method is useful if the number of unique values within a column are not too large.

Dataframe's describe() function is also useful to understand the summary statistics of the data like count, min, max, mean, standard deviation etc.

For quantitative variables with continuous values I used regular expressions (regex) to check if all values met the expected format. For example, for average monthly hours the expected format is 123.23. Regex validation helped confirm that there were no abnormalities within the data.

2. Exploratory Data Analysis and Data Visualization

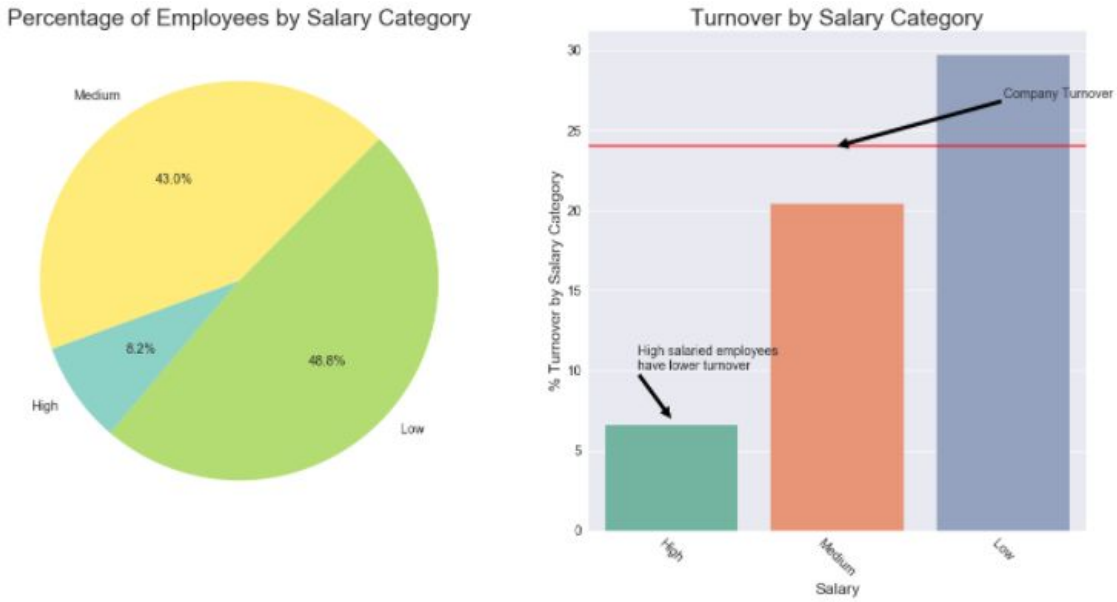
Before building any model it is important to spend time in exploratory data analysis, visualizing and summarizing the data.

Since the Employee turnover is the problem in question, I calculated the overall turnover by using the dataframe's groupby() function and representing it using a Matplotlib Pie chart which gives an inbuilt parameter to show the values as a percentage. The overall turnover is 24%

While exploring the different features, I identified trends among the two groups and their differentiating qualities.

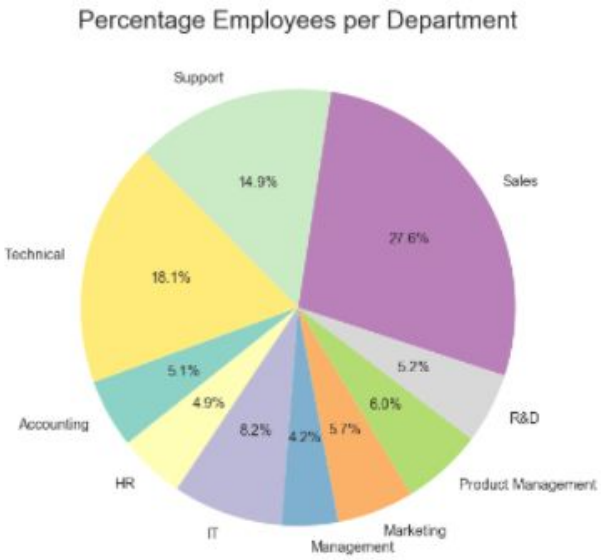
Salary

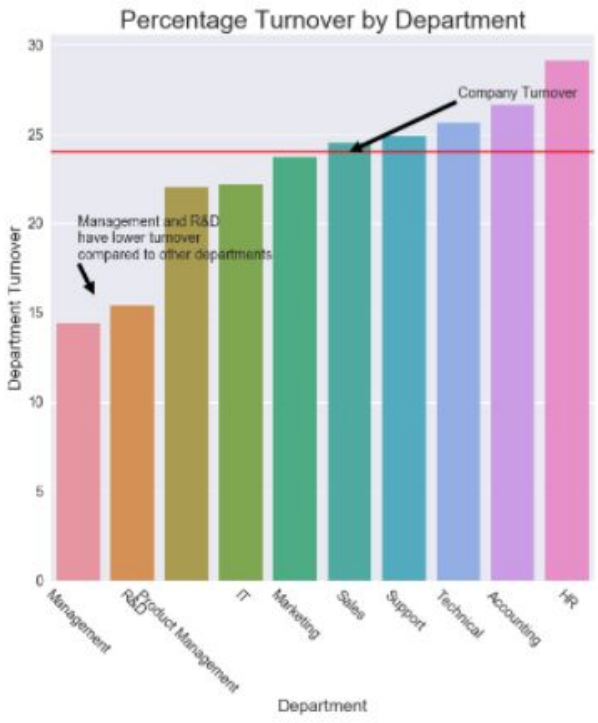
I added two subplots to a Matplotlib figure to show the distribution of salary categories within the employees in the given dataset as a pie chart (bar chart would also be a good alternative) and representation of turnover by salary category. I used group by functions and the lambda functions to calculate the turnover by each salary category. Annotate() of the Matplotlib axis was useful to add pointers and text to highlight interesting observations. Most of the employees fell into Low or Medium categories and those categories showed much higher turnover as compared to the High Salaried employees.



Departments

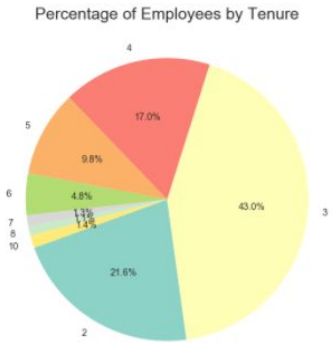
I used an approach similar to the one used for Salary to observe the distribution of employees across departments. Sales, Support and Technical have higher number of employees. In terms of turnover HR and accounting had higher turnover. R&D and Management have lower turnover compared to other departments.

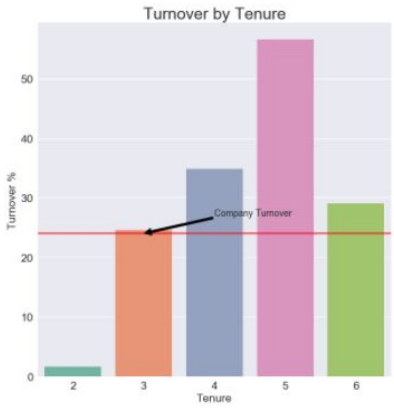


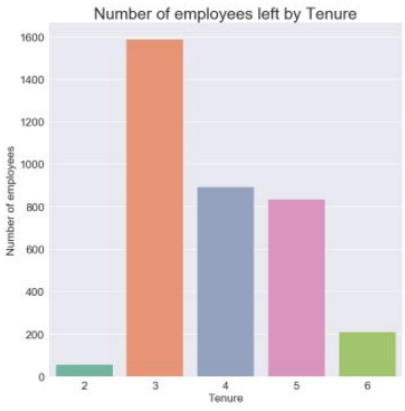


Company Tenure

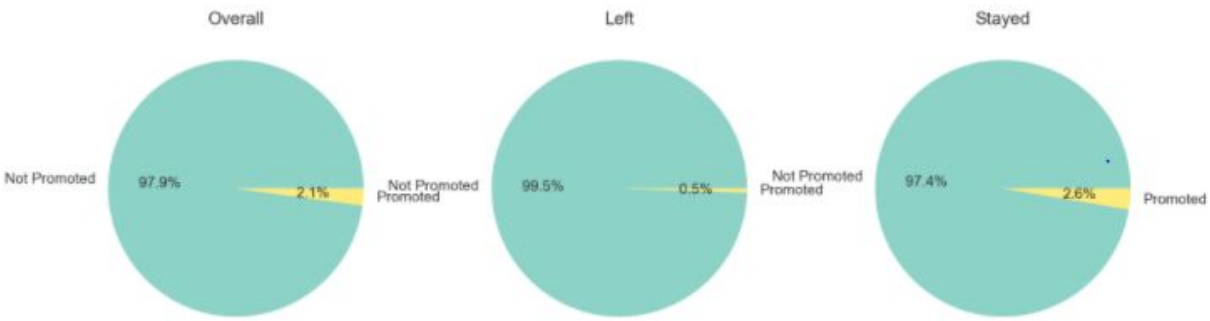
Most employees have been employed with the company for less than 3 years. Turnover begins around 3 years of service with the highest being around 5 years of service. There are very few employees that stay with the company beyond 5 years of service. I used different charts to look at the employee distribution, turnover and actual number by tenure to identify the trends.







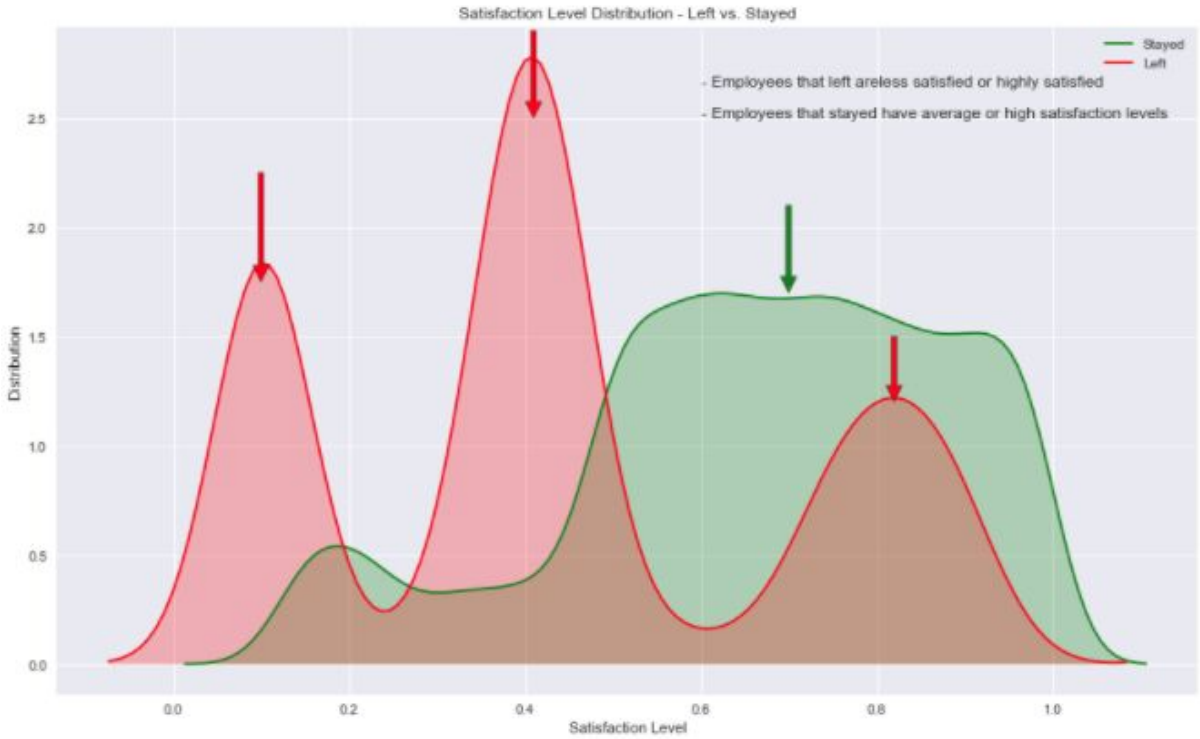
Promotions in last 5 years



Only 2.1% of all employees were promoted in the last 5 years. Most of the employees that left were never promoted. I used pie charts to show the promotions.

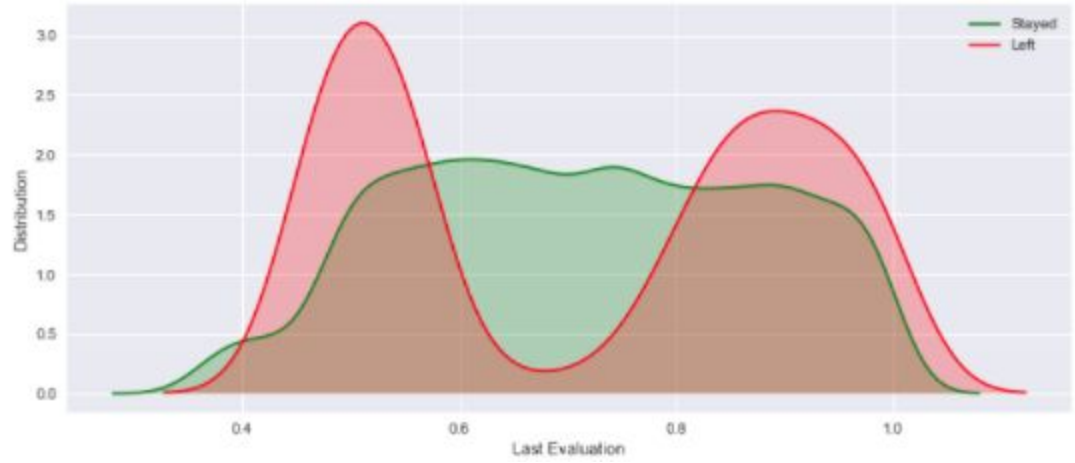
Satisfaction Level

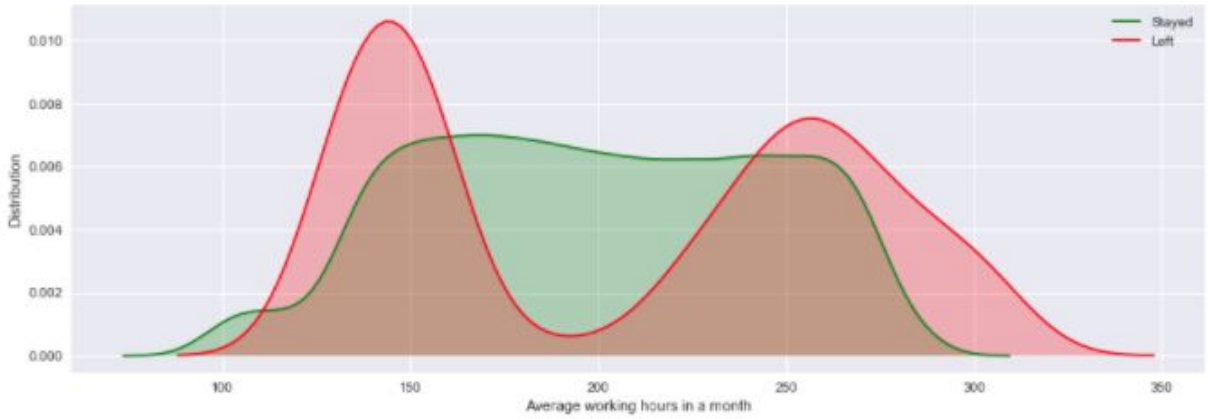
As Satisfaction levels were continuous variables, I chose kde plots to show the distribution of of satisfaction levels across both the groups. Most employees that left had lower satisfaction levels.



Job Performance & Average Monthly Working Hours

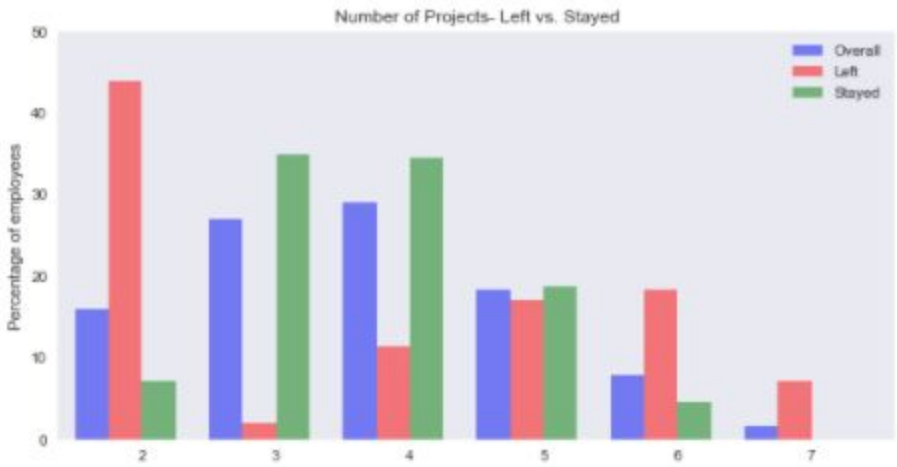
I used a similar approach as above to show the distribution of job performance within the two groups using kde plots. There were two distinct groups among the group of employees that left.





Number of projects

The dataset is imbalanced with more number of records belonging to the employees that were still employed. I did not think stacked bar would be a good choice to show the breakdown of employees. Instead I chose to represent percentages of employees. The blue bar shows the percentage of all employees that worked on the given number of projects. The orange and green bars show the percentage of employees among all the employees that left and all the employees that did not leave respectively.



3. Application of Inferential Statistics

For comparing categorical variables I used Chi-squared tests. For example, I used the test to prove the hypothesis that the difference in the number of employees leaving among groups of employees based on salary was statistically significant. Salary is a categorical variable having three categories: High, Medium and Low. The boolean variable used to indicate if an employee left or not can also be considered a categorical variable. In addition to chi-squared test I used Pairwise Chi-Squared test (McNemar tests) as there are more than two explanatory variables (Low, Medium, High among Salary categories). This helps understand if there are any particular pairs of categories where the difference is statistically significant. In most cases the the threshold value(popularly called P-value) is set at 0.05 or 5%. While doing multiple pairwise hypothesis tests, the chances of incorrectly rejecting a true null hypothesis become higher. So I used the Bonferroni correction to make the p-value smaller making it all the more difficult to reject the hypothesis.

I used the same approach for testing statistical significance of difference in turnover across departments. R&D and Management departments had a lower turnover compared to other departments. Using pairwise chi-squared test, I was able to prove that the difference in the turnover is statistically significant.

Similarly I divided the employees into two groups based on their satisfaction levels, one with low level of satisfaction and the other with high level of satisfaction. The difference in turnover between these two groups among the employees that left against their respective counterparts in the groups of employees that stayed, is statistically significant.

I divided the employees based on their job performance into three categories Low, Average and High performers. Using the similar approach, I was able to prove that the difference among groups compared to their counterparts among the employees that did not leave is statistically significant.

4. Machine Learning models

Baseline model using Logistic Regression

After completing the steps of data cleaning and exploration, I created a baseline model using Logistic Regression. The test and training accuracy scores were close to 80%. There was not a significant difference between the training and test accuracy scores, which meant that the model did not overfit to the training set. As an additional test to ensure there were no large gaps between the training and test accuracy score, I split the training and test sets randomly about 1000 times and ran them through the model. After plotting the results, we can observe that the test accuracy score was actually consistently better than the training accuracy score.

Accuracy score alone is not sufficient to evaluate model performance. In the section below, I've explained different ways to evaluate and understand the model better.

Baseline model performance evaluation methods

1. Classification Report and Confusion Matrix

Confusion matrix gives numbers of false positives and negatives as well as number of true positives and negatives. Below is a screen capture of the Test Confusion matrix

```
[Test confusion matrix:]
[[2659  198]
 [ 534  359]]
```

As highlighted, the number of false negatives for the positive class (employees who left) is very high which means the model is classifying majority of employees that did not leave and having left.

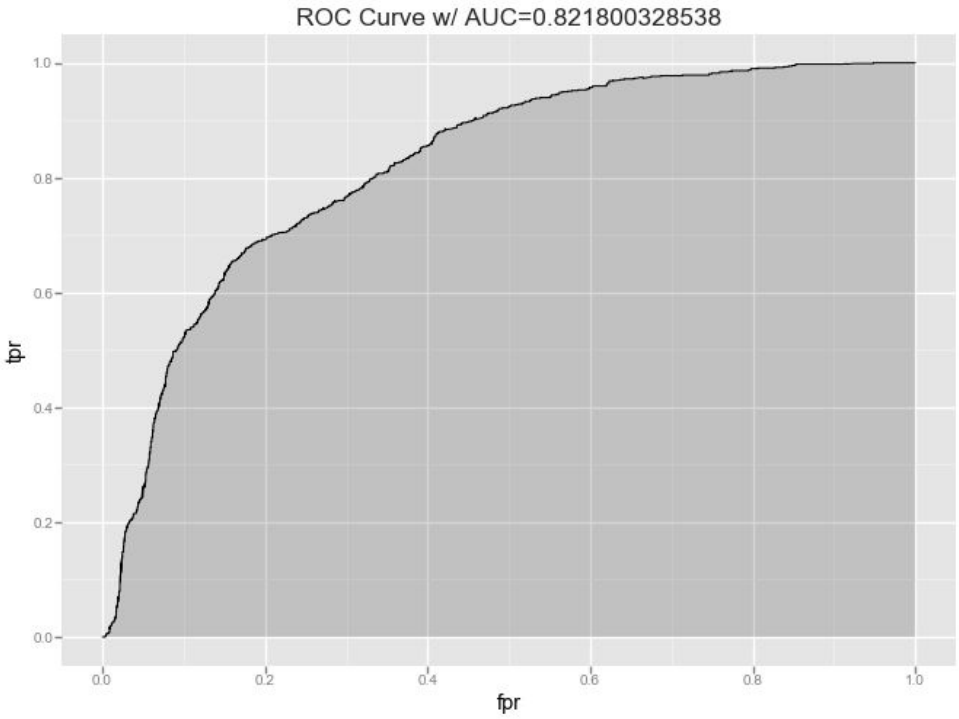
Precision and recall are also good ways to understand the model performance.

```
[Test Classification Report:]
             precision    recall  f1-score   support

          0       0.83      0.93      0.88      2857
          1       0.64      0.40      0.50       893

avg / total       0.79      0.80      0.79      3750
```

As highlighted, the Recall of positive class is also low at 0.4 and Precision is 0.65. The majority class with 75% of observations is the negative class representing the employees that did not leave the company. Low recall indicates that the model was possibly “cheating” by classifying all records as negative and still maintaining an accuracy score of ~80%. This behavior is due to class imbalance and I have presented ways to tackle this problem later in this document.

2. ROC Curve and AUC

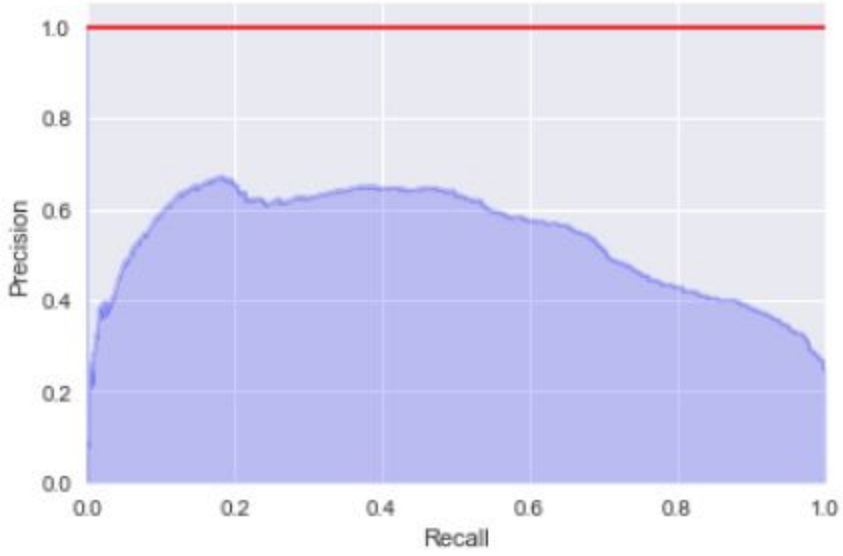
ROC curve can be observed by plotting the false positive rate (on x-axis) against the true positive rate (on y-axis). The ideal would be a horizontal line where the true positive rate is always 1. A good model would have a curve as close to this line as possible while maximizing the area under the curve (AUC).



ROC curve and the AUC look quite good for the baseline model, however we cannot draw any conclusions based on this metric alone especially because the precision and recall for the positive class are low.

3. Precision vs. Recall

The ideal scenario would be to have precision as 1 regardless of recall.



For the baseline model , the model precision does not seem to improve no matter what the recall is.

Below is the summary of results for the baseline model using Logistic regression.

Baseline model performance results

Method	Training Accuracy Score	Test Accuracy Score	Precision of negative class in Test Set (Employees who did not leave)	Precision of positive class in Test Set (Employees who left)	Recall of negative class in Test Set (Employees who did not leave)	Recall of positive class in Test Set (Employees who left)
Baseline using Logistic Regression	79.87	80.48	0.83	0.64	0.93	0.4

Applying Regularization on baseline model with Logistic regression

Typically regularization is applied to handle the problem of overfitting. Even though there are no signs of overfitting in the baseline model, I created additional models by applying L1 (Lasso) and L2 (Ridge) regression to help with any future needs that could arise as new data becomes available. The accuracy score did not change significantly after applying regularization.

Below is the summary of results for the Logistic regression model after applying L1 Regularization

Method	Training Accuracy Score	Test Accuracy Score	Precision of negative class in Test Set (Employees who did not leave)	Precision of positive class in Test Set (Employees who left)	Recall of negative class in Test Set (Employees who did not leave)	Recall of positive class in Test Set (Employees who left)
Logistic Regression with L1 Regularization	79.69	80.51	0.83	0.63	0.93	0.38

Below is the summary of results for the Logistic regression model after applying L2 Regularization

Method	Training Accuracy Score	Test Accuracy Score	Precision of negative class in Test Set (Employees who did not leave)	Precision of positive class in Test Set (Employees who left)	Recall of negative class in Test Set (Employees who did not leave)	Recall of positive class in Test Set (Employees who left)
Logistic Regression with L1 Regularization	79.65	80.59	0.83	0.63	0.93	0.38

As noted, regularization did not improve the accuracy scores, precision and recall from the baseline model using Logistic Regression

Baseline using Random Forest Classifier

Below is the summary of results from the Random Forest Classification Model

Method	Training Accuracy Score	Test Accuracy Score	Precision of negative class in Test Set (Employees who did not leave)	Precision of positive class in Test Set (Employees who left)	Recall of negative class in Test Set (Employees who did not leave)	Recall of positive class in Test Set (Employees who left)
Random Forest Classifier	99.97	98.96	0.99	0.98	0.99	0.97

The results were almost perfect with Random Forest Classifier. The model evaluation methods and the regularization that I used for the baseline model can also be used here. For the sake of brevity, I chose not to the same analysis here.

Even before tackling the class imbalance the model provides very good predictions.

Baseline using XGBoost Classifier

Below is the summary of results from the XGBoost Classification Model

Method	Training Accuracy Score	Test Accuracy Score	Precision of negative class in Test Set (Employees who did not leave)	Precision of positive class in Test Set (Employees who left)	Recall of negative class in Test Set (Employees who did not leave)	Recall of positive class in Test Set (Employees who left)
XGBoost Classifier	99.97	98.96	0.99	0.98	0.99	0.97

The results were almost perfect with XGBoost classier as well. The model evaluation methods and the regularization that I used for the baseline model can also be used here. For the sake of brevity, I chose not to the same analysis here.

Even before tackling the class imbalance the model provides very good prediction rate.

Applying sampling techniques for class imbalance

I applied all sampling methods on training data only. Applying sampling before the train-test split could lead to oversampled data to “leak” into the test data. Usually the intent is to keep the test set as a true representation of the original dataset. Splitting after oversampling/undersampling defeats this purpose.

Oversampling using SMOTE (Synthetic Minority Oversampling Technique)

I used SMOTE method from the Scikit-learn library to oversample the training data. Below is the summary of results after applying Logistic Regression, Random Forest Classifier and XGBoost algorithms on over sampled data.

Method	Training Accuracy Score	Test Accuracy Score	Precision of negative class in Test Set (Employees who did not leave)	Precision of positive class in Test Set (Employees who left)	Recall of negative class in Test Set (Employees who did not leave)	Recall of positive class in Test Set (Employees who left)
Baseline using Logistic Regression	79.87	80.48	0.83	0.64	0.93	0.4
SMOTE+ Logistic Regression	75.87	75.68	0.92	0.49	0.74	0.80
Random Forest Classifier	99.97	98.96	0.99	0.98	0.99	0.97
SMOTE+ Random Forest	99.97	98.83	0.99	0.98	0.99	0.97
XGBoost Classifier	99.97	98.96	0.99	0.98	0.99	0.97
SMOTE+ XGBoost	97.27	97.2	0.98	0.95	0.93	0.97

For ease of comparison, I've included the baseline model results and highlighted them in blue.

In case of Logistic regression, the accuracy scores have actually declined. For the positive class, the recall improved from 0.40 to 0.80 which means the false negatives have reduced. But the precision has reduced which means the false positives have increased. The algorithm after oversampling is definitely not "cheating" as there is no majority class in the data.

The accuracy scores, precision and recall for Random Forest classifier remained relatively unaffected after oversampling. Random Forest is a very good classifier for this use case.

XGBoost performance lowered slightly after oversampling.

Oversampling using ADASYN (Adaptive Synthetic sampling method)

I used ADASYN method from the Scikit-learn library to oversample the training data. Below is the summary of results after applying Logistic Regression, Random Forest Classifier and XGBoost algorithms on the over sampled data.

Method	Training Accuracy Score	Test Accuracy Score	Precision of negative class in Test Set (Employees who did not leave)	Precision of positive class in Test Set (Employees who left)	Recall of negative class in Test Set (Employees who did not leave)	Recall of positive class in Test Set (Employees who left)
Baseline using Logistic Regression	79.87	80.48	0.83	0.64	0.93	0.4
SMOTE+ Logistic Regression	75.87	75.68	0.92	0.49	0.74	0.80
ADASYN+ Logistic Regression	74.76	75.23	0.94	0.48	0.71	0.86
Random Forest Classifier	99.97	98.96	0.99	0.98	0.99	0.97
SMOTE+ Random Forest	99.97	98.83	0.99	0.98	0.99	0.97
ADASYN+ Random Forest	99.97	98.53	0.99	0.97	0.99	0.97
XGBoost Classifier	99.97	98.96	0.99	0.98	0.99	0.97
SMOTE+ XGBoost	97.27	97.2	0.98	0.95	0.93	0.97
ADASYN+ XGBoost	97.27	97.2	0.98	0.96	0.99	0.93

For ease of comparison, I've included the baseline model results and highlighted them in blue and the results from SMOTE oversampling in magenta.

The comments for model performances after oversampling training data using ADASYN would be similar to the comments for model performances after oversampling training data using SMOTE

Undersampling using Random Under Sampler

I used Random Under Sampler method from the Scikit-learn library to undersample the training data. Below is the summary of results after applying Logistic Regression, Random Forest Classifier and XGBoost algorithms on the undersample sampled data.

Method	Training Accuracy Score	Test Accuracy Score	Precision of negative class in Test Set (Employees who did not leave)	Precision of positive class in Test Set (Employees who left)	Recall of negative class in Test Set (Employees who did not leave)	Recall of positive class in Test Set (Employees who left)
Baseline using Logistic Regression	79.87	80.48	0.83	0.64	0.93	0.4
SMOTE+ Logistic Regression	75.87	75.68	0.92	0.49	0.74	0.80
ADASYN+ Logistic Regression	74.76	75.23	0.94	0.48	0.71	0.86
RandomUnder Sampler+ Logistic Regression	74.76	75.23	0.95	0.49	0.71	0.89
Random Forest Classifier	99.97	98.96	0.99	0.98	0.99	0.97
SMOTE+ Random Forest	99.97	98.83	0.99	0.98	0.99	0.97

ADASYN+ Random Forest	99.97	98.53	0.99	0.97	0.99	0.97
RandomUnder Sampler+ Random Forest	99.97	98.53	0.99	0.97	0.99	0.97
XGBoost Classifier	99.97	98.96	0.99	0.98	0.99	0.97
SMOTE+ XGBoost	97.27	97.2	0.98	0.95	0.93	0.97
ADASYN+ XGBoost	97.27	97.2	0.98	0.95	0.98	0.93
RandomUnder Sampler+ XGBoost	97.27	97.2	0.98	0.95	0.98	0.93

For ease of comparison, I've included the baseline model results and highlighted them in blue, results from SMOTE oversampling in magenta, results from ADASYN oversampling in orange.

The comments for model performances after under sampling training data using Random Under Sampling would be similar to the comments for model performances after SMOTE and ADASYN.

Findings

Higher turnover is observed in the following scenarios

- During 3 to 5 years of service.
- Employees that worked too many hours or worked too few hours
- Employees that were less satisfied
- High performers or low performers
- Low to medium salaries
- Among departments, Sales, Technical and Support are have the highest Turnover.

With given data Random Forest and XGBoost have the best performance

Recommendations

- Place emphasis on employees that are in their first few years of employment i.e. 1 to 5 years of service
- Ensure employees are balanced in their workload in projects, number of hours worked etc.
- High performers and low performers are at the highest risk for turnover. Low performers need to be put on a performance improvement plan. High performers need to be incentivized with promotions, better pay and challenging projects.
- Evaluate pay structures especially for low and medium salaried employees. Ensure salary and benefits are not the reason for high turnover.
- Create more incentives and opportunities in the departments with high turnover: Sales, Technical and Support
- Choose RandomForest or XGBoost classifier to predict employee turnover. The code also contains models that tackle class imbalances and hyper parameter tuning to prevent overfitting. As new data becomes available, evaluate the models and choose the one with best performance.