IE School of Human Sciences and Technology
Master in Business Analytics and Big Data
Machine Learning 2
Professor Jesus Salvador Renero Quintero

Machine Learning 2: Assignment 1

Muhammad Furqan

Table of Contents

Executive Summary

The aim of this report is to present a model that can predict probability of attrition of each employee and highlight variables that play the most important role with respect to attrition. The model is built using Logistic Regression with the aim of attaining optimal accuracy score on past data. The model can predict whether an employee would leave the company or not with **95.4%** accuracy. Variables that impact the attrition rate are **work accident**, **promotion last five years**, **department**, **satisfaction level**, **last evaluation**, **number of projects**, **average monthly hours**, and **time spend company**. Each variable is further divided into range of scores and the model identifies which range of score impacts attrition the most.

Data Loading and Summary Statistics

The dataset comprises of information about 15000 employees. Each employee record contains data for ten different information points. The first five columns are numeric columns, whereas the last five columns have either boolean values or strings. Columns with strings have salaries categorized in three levels, low, medium, and high, and departments categorized as ten different department names.

Summary statistics for every column is computed to understand the shape of data and detect presence of columns that have a relatively bigger scale. Number_project, Average_monthly_hours, and Time_spend_company are on a much larger scale compared to other columns, therefore, data will have to be scaled to a common range. Furthermore, data does not contain any Null values.

Data Transformation

Encoding: The columns that were detected to have string, namely, salary and department column were divided into the number of unique strings present in each column. The salary column had three unique strings and the department column had ten unique strings.

In order to encode the columns that had strings, a custom function was used to detect categorical columns, divide them into the number of unique values found in every categorical column and replace the string with 1/0. This operation increased the number of columns from ten to twenty one.

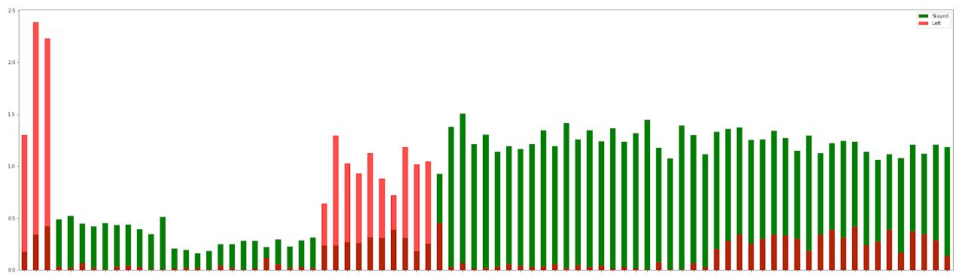
Scaling: Summary statistics on the data set showed that numerical columns did not have a compatible scale. Some columns had values between zero and one, whereas, other columns had values in hundreds. In order to standardize the scale of numerical columns, python's built-in MinMaxScaler function was used on the following columns: satisfaction_level, last_evaluation, number_project, average_montly_hours, and time_spend_company.

Fix Data Type: The datatype of every column was checked with a python built-in function 'dtypes'. Upon inspection, it was found that all the columns with boolean values were either ‘int64’ or ‘uint8’. In order to fix the data type of miscategorized columns, panda’s ‘categorical’ function was used.

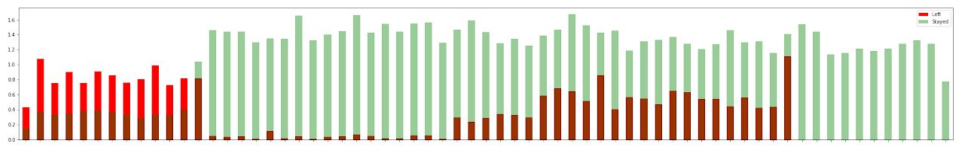
Data Exploration

The main aim of data exploration is to find hidden insights and use them to tweak the dataset. In order to find patterns in data, some variables were mapped to the target variable, ‘left’. Mapping helped in highlighting the range of scores of a particular variable associated with people leaving the company and the range of scores of a variable associated with people not leaving the company. This exploratory data analysis was conducted for the following variables:

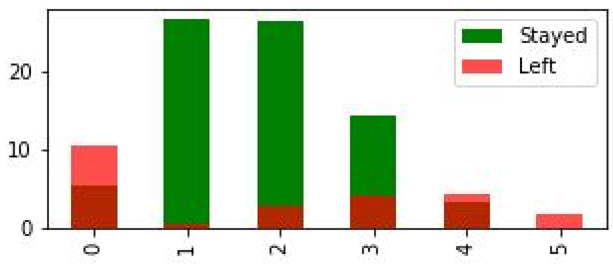
Satisfaction level: The red bars in the graph below show that for a particular satisfaction level, more people leave the company than those who stay. Without going into details, it can be observed that most employees with low satisfaction levels and mid range scores for satisfaction leave the company.



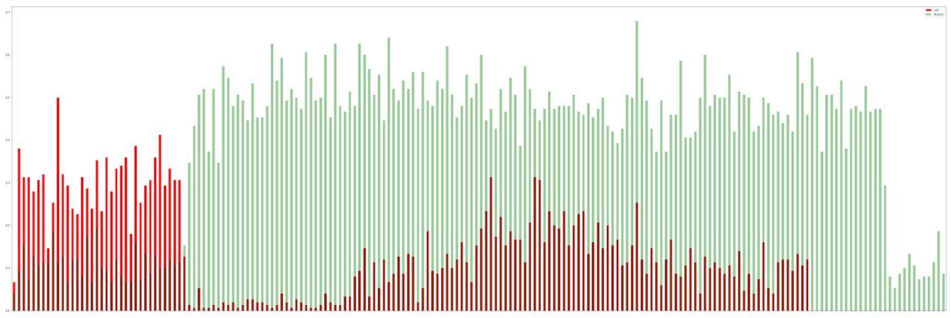
Last evaluation: Similarly, for last evaluation scores, from the employees who were evaluated poorly, a lot more left the company than those who stayed back. Average last evaluation score also resulted in attritions but the number was not as big as for those who had low last evaluation scores.



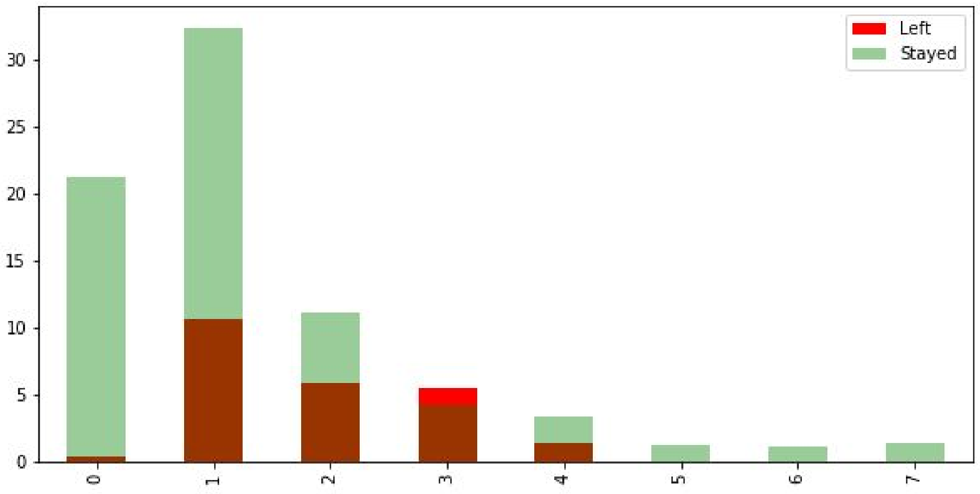
Number of Projects: The graph shows that fewer employees stay with the company for number of projects score of 0,4, and 5 than employees who leave the company.



Average Monthly Hours: The red bars in the graph below show that for a particular number of monthly hours, more people leave the company than those who stay. Without going into details, it can be observed that most employees who work less hours leave the company. Whereas, for hours in the mid range, attritions are less than number of employees who stay with the company.



Time spent with the company: There are more employees who leave after working with the company for three years than there are employees who stay with the company after working for three years. Alot of employees also leave after working for 1 or 2



Baseline model

At this point, the transformed data was used to build a baseline model. Python's built-in 'LogisticRegressionCV' function was used with k=5. The value of K was chosen to be 5 in order to have a 80/20 split between train and test data in each iteration. The mean accuracy score was **78.4**.

Feature Engineering

Findings from the data exploration exercise formed the basis for feature engineering. Columns were binned according to pattern in the graphs. For example, employees who scored their satisfaction between 0.0 and 0.022 left the company, therefore a seperate column was made for employees with satisfaction level between 0.0 and 0.22 and filled with

0/1 implying whether an employee stayed or left. Following functions were defined to bin and dummify the columns:

Sats_trans: Binned satisfaction level as follows: [0.0, 0.022, 0.286, 0.396, 0.91]. Dummified each column and dropped the original satisfaction level column

Eval_trans: Binned 'Last evaluation' column as follows: [0.0,0.171875,0.453125,0.828125,1.0]. Dummified each column and dropped the original 'Last evaluation' column.

Proj_trans: Replaced the numerical values in 'Number of Projects' column with the dictionary values mentioned below and dummified the column as per the labels. For example, all the employees with low number of projects had 1's for low and 0's for medium, high, and very high.
dict = { 0.20000000000000007: "low", 0.4 : "medium", 0.6 : "medium", 0.8000000000000002: "high", 0.0: "high", 1: "Very high"}

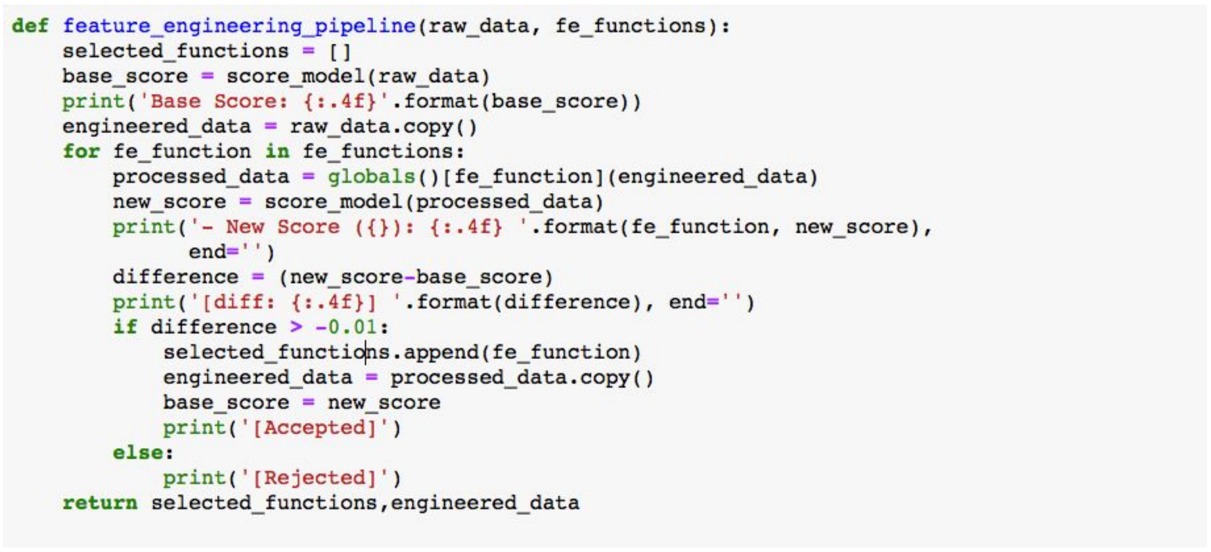
Hour_trans: Binned 'Average monthly hours' columns as follows: [0.0, 0.1589, 0.322, 0.7617,0.8925]. Dummified each column and dropped the original 'Average monthly hours' column.

Year_trans: Replaced the numerical values in 'Time spend company' column with the dictionary values mentioned below and dummified the column as per the labels. For example, all the employees with low time spend company had 1's for low and 0's for very low, medium, and high.
dict = { 0.0: "low", 0.125 : "medium", 0.25 : "medium", 0.5 : "medium", 0.375: "high", 0.625: "very low", 0.75: "very low", 1.0: "very low"}

Pipeline

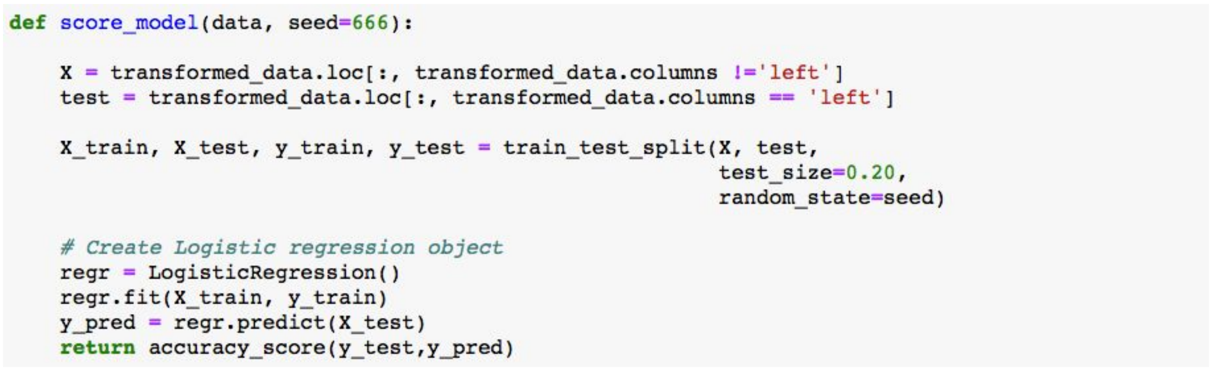
All the transformations were vetted before being accepted. Figure 1.0 shows the detailed pipeline that was used to test whether a transformation is useful or not. Each transformation was applied to the dataset and the dataset was split into train and test set before being scored by Score model function, shown in Figure 1.1. If the difference in accuracy between the baseline model and transformed model was greater than -0.01, transformation was accepted permanently and dataset was changed accordingly. All the transformations listed above were accepted by the pipeline.

After all the transformations were accepted, engineered dataset was used for Logistic regression with five fold cross validation. Accuracy score for the engineered data was **93.2%**, which is a **14.8%** increase over the baseline model.



```python
def feature_engineering_pipeline(raw_data, fe_functions):
    selected_functions = []
    base_score = score_model(raw_data)
    print('Base Score: {:.4f}'.format(base_score))
    engineered_data = raw_data.copy()
    for fe_function in fe_functions:
        processed_data = globals()[fe_function](engineered_data)
        new_score = score_model(processed_data)
        print('- New Score ({}): {:.4f} '.format(fe_function, new_score),
              end='')
        difference = (new_score-base_score)
        print('[diff: {:.4f}] '.format(difference), end='')
        if difference > -0.01:
            selected_functions.append(fe_function)
            engineered_data = processed_data.copy()
            base_score = new_score
            print('[Accepted]')
        else:
            print('[Rejected]')
    return selected_functions,engineered_data
```

Figure 1.0



```python
def score_model(data, seed=666):

    X = transformed_data.loc[:, transformed_data.columns !='left']
    test = transformed_data.loc[:, transformed_data.columns == 'left']

    X_train, X_test, y_train, y_test = train_test_split(X, test,
                                                        test_size=0.20,
                                                        random_state=seed)

    # Create Logistic regression object
    regr = LogisticRegression()
    regr.fit(X_train, y_train)
    y_pred = regr.predict(X_test)
    return accuracy_score(y_test,y_pred)
```

Figure 1.1

Feature Selection

The final step in this project was to test whether same or better accuracy can be achieved with lesser features. In order to perform feature selection, Recursive Feature Elimination (RFE) method was used. RFE method works by recursively removing attributes and building a model on those attributes that remain. It uses accuracy metric to rank the feature according to their importance. The RFE method takes the Logistic Regression model and the number of required features as input. It outputs the optimal number of features and the accuracy score using those features. The model achieved an accuracy of **93.6%**, which is **0.4%** higher than the previous attempt with only **22 features** instead of **36** that were used for the last model.

Polynomial Features

Finally, after reducing the dataset to 22 features, polynomial features with a degree of 2 were computed for every variable. Dataset with polynomials was used for modelling with five fold cross validation, which produced an accuracy of **95.4%.**