# *Final report - HR analysis case study*

Sicheng Yang
DSI, Brown University
https://github.com/kkarzyang/data1030_project

## Introduction

This report will mainly focus on a machine learning problem which is about predicting whether a given employee is going to be promoted or not. This study will play an important role in modern sociology research, as scientists can study people's behavior in the company. Junior employees or even students currently in school may also be interested in this topic because they want to know a way to help them get promoted. This is a binary classification problem and the dataset is from *Kaggle*. There are 13 features in total in this dataset. There are also some previous works we can check as references. *Tarun Thakur*[1] gave a decent EDA and tried some models which gave a pretty good result. The models he tried get around 96-99% accuracy on the training set. *Ashwin viswanathan*[2] prepares pair plots for the feature of interesting in his work

## EDA

First of all, the employee_id column is dropped because it is not useful. Then we separate the target variable from the original data frame. The dataset has 54808 rows in total.

The following are the features before preprocessing.

| Type | Features |
|---|---|
| Categorical | department, region, gender, recruitment_channel, KPIs_met >80%, awards_won? |
| Numerical | no_of_trainings, length_of_service, avg_training_score, age |
| Ordinal | education, previous_year_rating |

Table 1. Summary of features before preprocessing

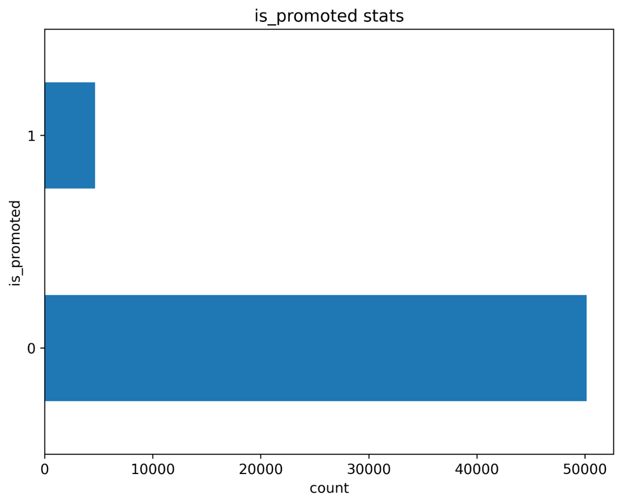

Figure 1. The distribution of the target variable

From the above figure, we can see that our data set is imbalanced.

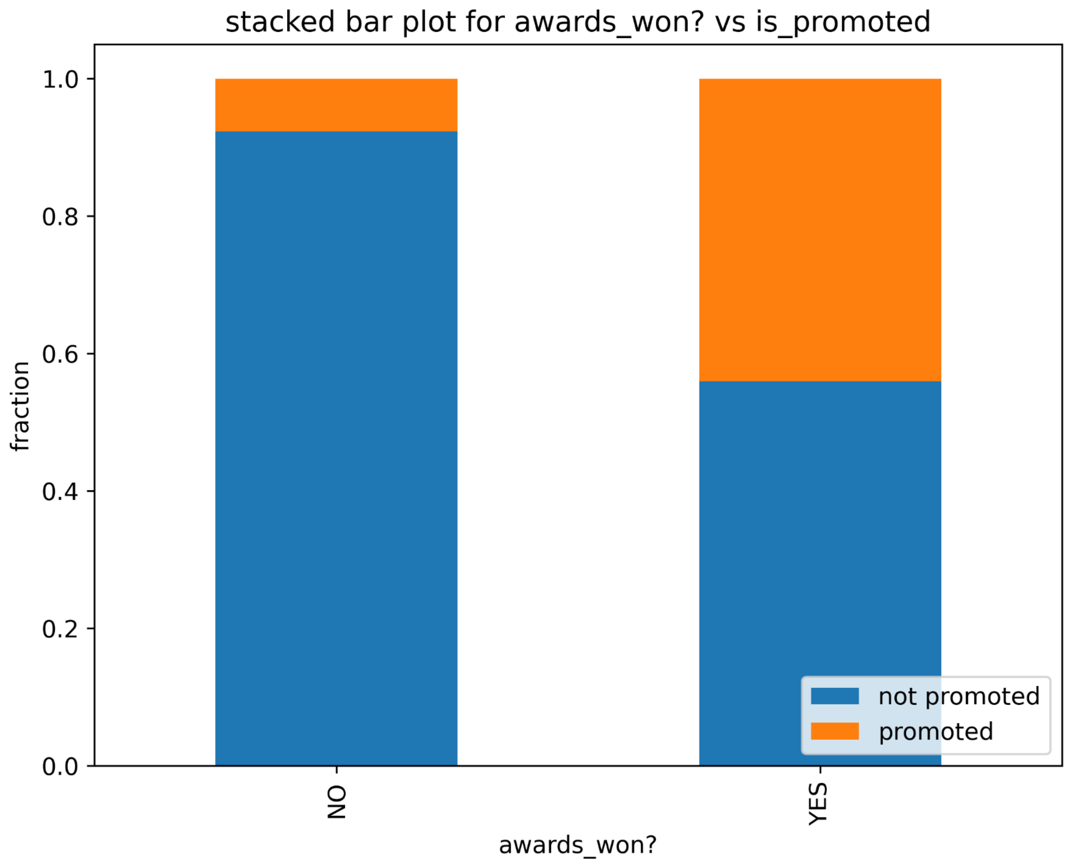

Figure 2. The stacked bar plot for the feature 'awards_won?' and the target variable

From the above figure, we can see that nearly fifty percent of people who win an award get promoted, while less than 10 percent of people who do not win an award get promoted.

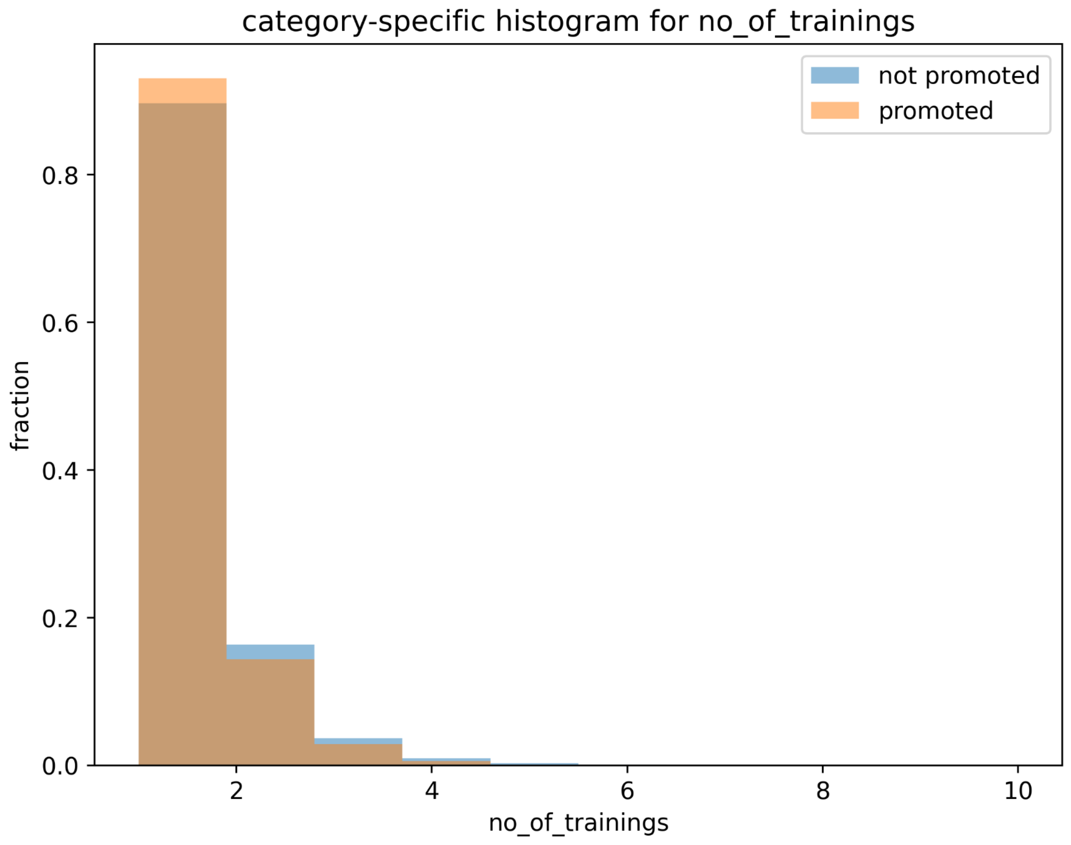

Figure 3. The category-specific histogram for feature ‘no_of_trainings’ and the target variable

From the Figure above, we can see that the shape of distribution for no_of_training is pretty similar between people who get promoted and people who do not get promoted. The majority of people have 0-4 training sessions.

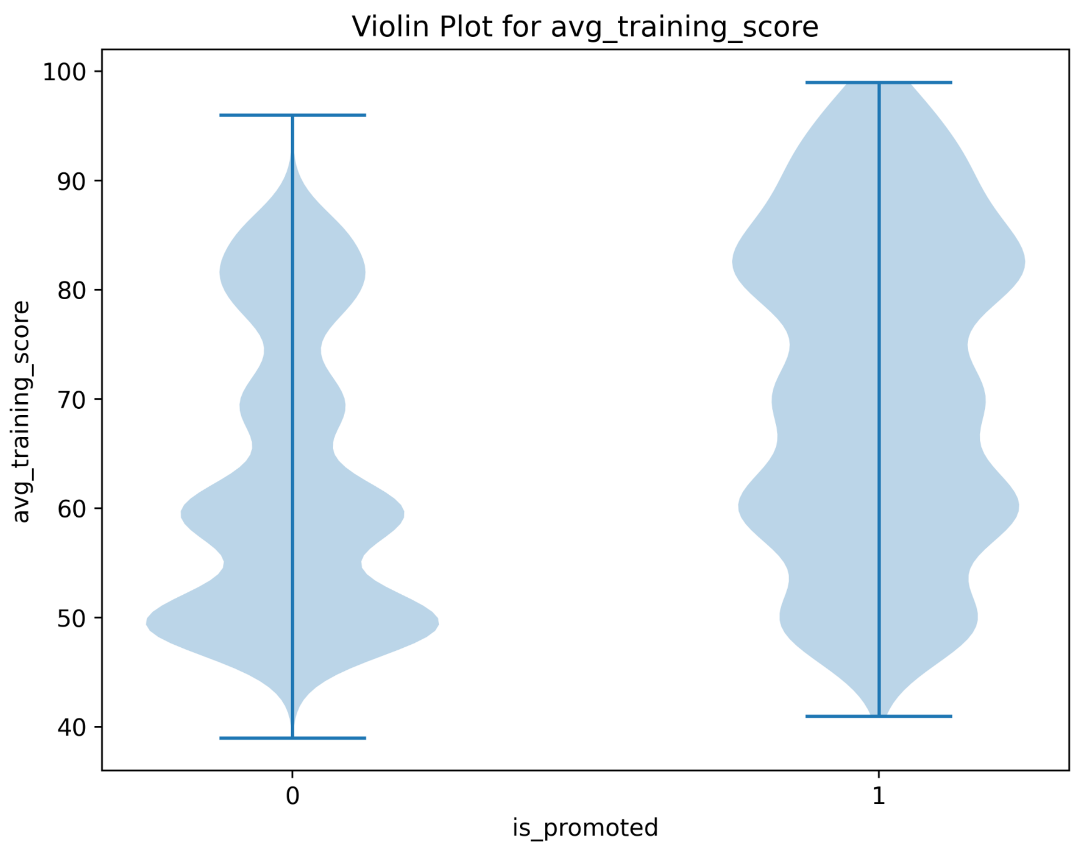

Figure 4. The Violin Plot for feature 'avg_training_score' and the target variable

From the figure above, we can see there is a higher proportion of people that have a high avg_training_score among promoted employees than among employees that do not get promoted.

There are missing values in the dataset. There are 4897 rows with missing values, which is 11.17% of the total.

| | Fraction of missing value in features | Type of feature |
|---|---|---|
| education | 0.043333 | Ordinal |
| previous_year_rating | 0.075218 | Ordinal |

Table 2. Summary of features with missing values

## Methods

Because the dataset is an imbalanced dataset, choosing methods with stratification will be a better choice. In terms of splitting the data, I first use the general train_test_split method with stratification from *sklearn* and have the size of the test set equal to 20%. For the other 80% data (OTHER), they go into StratifiedKFold with the number of folds equal to 4 (which means 0.75 * OTHER go for train and 0.25 * OTHER go for cross validation each time). Kfold related methods can always effectively utilize data.

Data preprocessing is necessary for this dataset. For numerical features that have a general upper and lower bound (age), minmaxScaler is applied. For the other numerical features (no_of_trainings, length_of_service, avg_training_score), StandardScaler is applied. For the categorical features (department, region, gender, recruitment_channel, KPIs_met >80%, awards_won?), OneHotEncoder is applied. For the ordinal features (education, previous_year_rating), since both of them have missing values, we first impute the missing values with 'NA' and 0.0 respectively and order them among all categories. For example: ["NA","Below Secondary","Bachelor's","Master's & above"] & [0.0,1.0,2.0,3.0,4.0,5.0]. Then OrdinalEncoder is applied. Before preprocessing, there are 12 columns in the dataframe. After preprocessing, there are 58 columns in the dataframe.

The ML pipeline in this project includes two parts: 1) preprocessor (see above) 2) classifier we want to use (which will be discussed below)

I try both accuracy and f1-score as a metric to evaluate my models' performance because these two metrics are for classification problems (Theoretically speaking, f1 score is better for an imbalanced dataset compared to accuracy). Four different ML algorithms are used here: 1) K nearest neighbor 2) Random Forest 3) Xgboost 4) Logistic Regression classifier with elasticnet. Different hyperparameters are tuned for these four algorithms.

For K nearest neighbor classifier, I tune parameter *n_neighbors* with values: [3, 5, 10, 15, 30, 50] , parameter *weights* with values: ['uniform', 'distance']

For Random Forest classifier, I tune parameter *max_depth* with values: [None, 1, 3, 5, 10, 15], parameter *max_features* with values: [None, 0.25, 0.5, 1.0], parameter *n_estimators* with values: [1, 3, 10, 30]

For Xgboost, I tune parameter *learning_rate* with values: [0.01, 0.1, 0.2], parameter *max_depth* with values: [3, 4, 5], parameter *min_child_weight* with values: [1, 3, 5], parameter *subsample* with values: [0.8, 0.9, 1.0]

For Logistic Regression classifier with elasticnet, I tune parameter *C* with values:, parameter *l1_ratio* with values: [1e3, 3.16e1, 1e0, 3.16e-2, 1e-3]

Because there are uncertainties of evaluation metric due to splitting and due to non-deterministic ML methods, 10 random states are chosen and fixed (multiple of 42, from 0 to 10). To take the uncertainties into consideration, we will report the mean and standard deviation of the scores among these 10 random states, for all four algorithms.

## Results

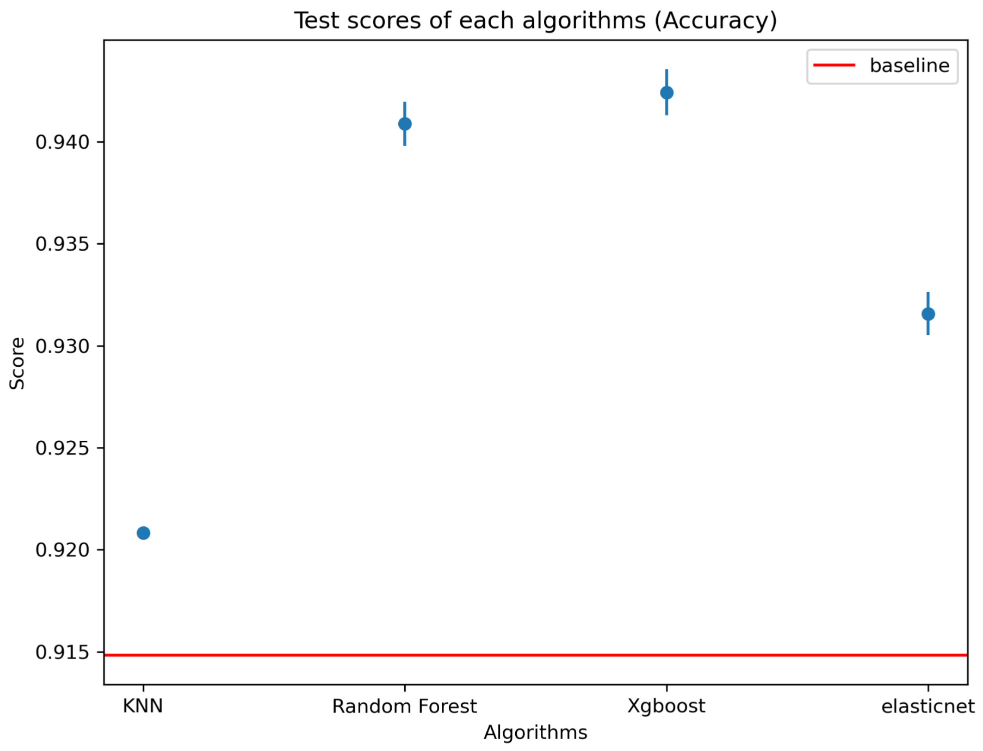

Figure 5. Test score for 4 different algorithms on the dataset with accuracy as metric

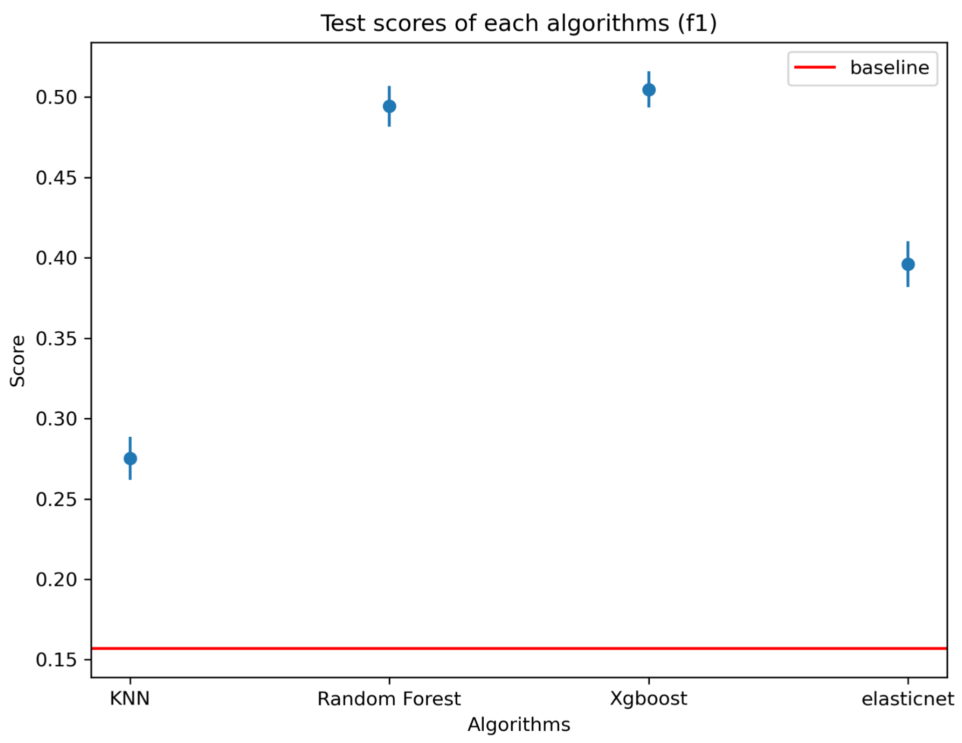

Figure 6. Test score for 4 different algorithms on the dataset with F1 score as metric

From the two figures above, we can see that all the four algorithms we choose overperform the baseline model on this dataset. The baseline accuracy of the test set is 0.9148 and the baseline F1 score of the test set is 0.1570. In the context of both accuracy and F1 score, Xgboost performs the best on the test set. Random Forest method has a similar score to Xgboost and ranks second in both figures. KNN performs the worst and has the lowest scores among these four methods. Xgboost is the most predictive ML model on the test set.

We can also calculate how many standard deviations above the baseline(s) our models are. A higher score and a lower standard deviation indicate the model is better. We can see that Xgboost still overperforms the other three methods. After an overall consideration of robustness and predictive power of the four models we have, we will continue with Xgboost as our main algorithm for the discussion later.

|  | Accuracy | F1 Score |
|---|---|---|
| KNN | 21.8 | 8.9 |
| Random Forest | 24.0 | 26.5 |
| Xgboost | 24.4 | 30.8 |
| Elasticnet | 15.7 | 16.8 |

Table 3. Number of standard deviations above the baseline(s) for four different algorithms

In the next step, we will take a look at the different global feature importance.

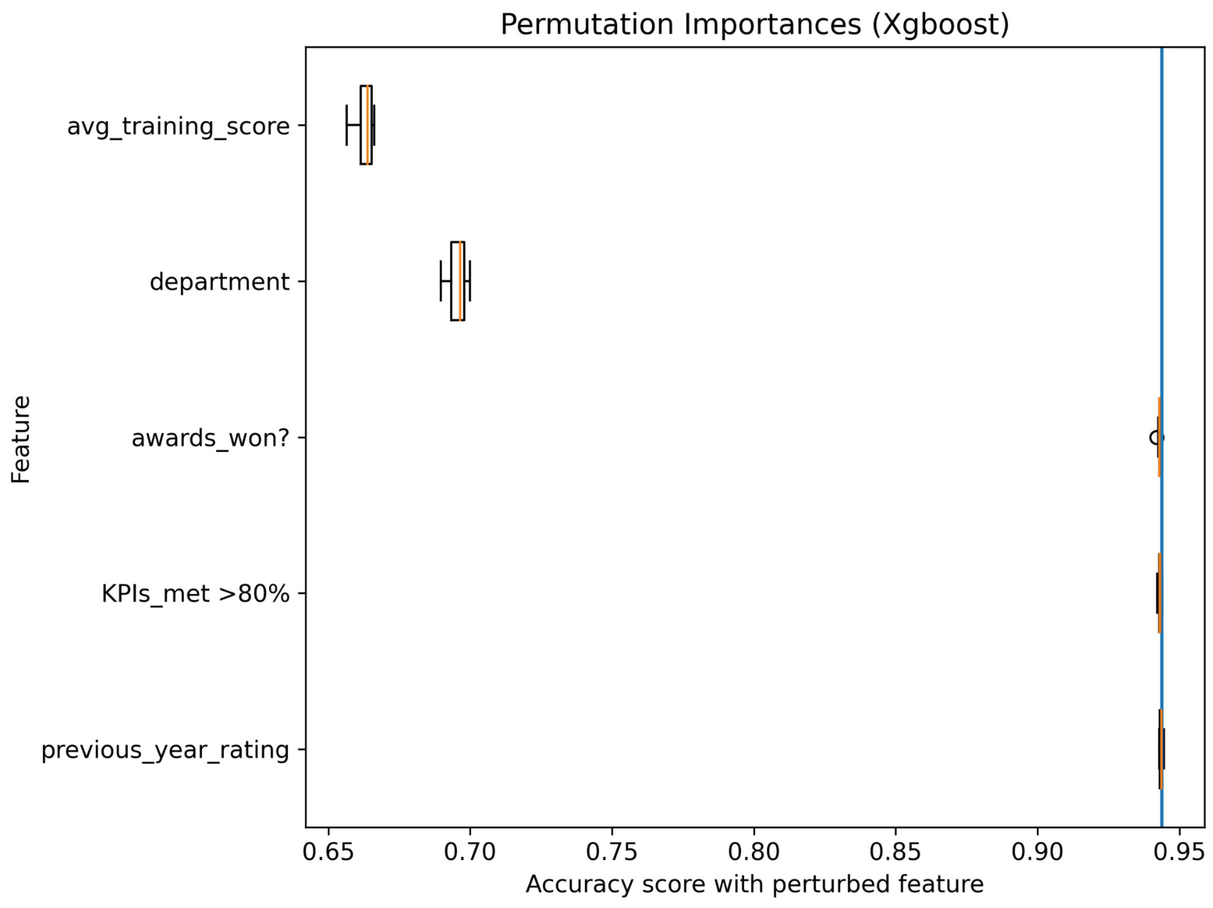

Figure 7. Top 5 most important features with Permutation Importances, accuracy as metric

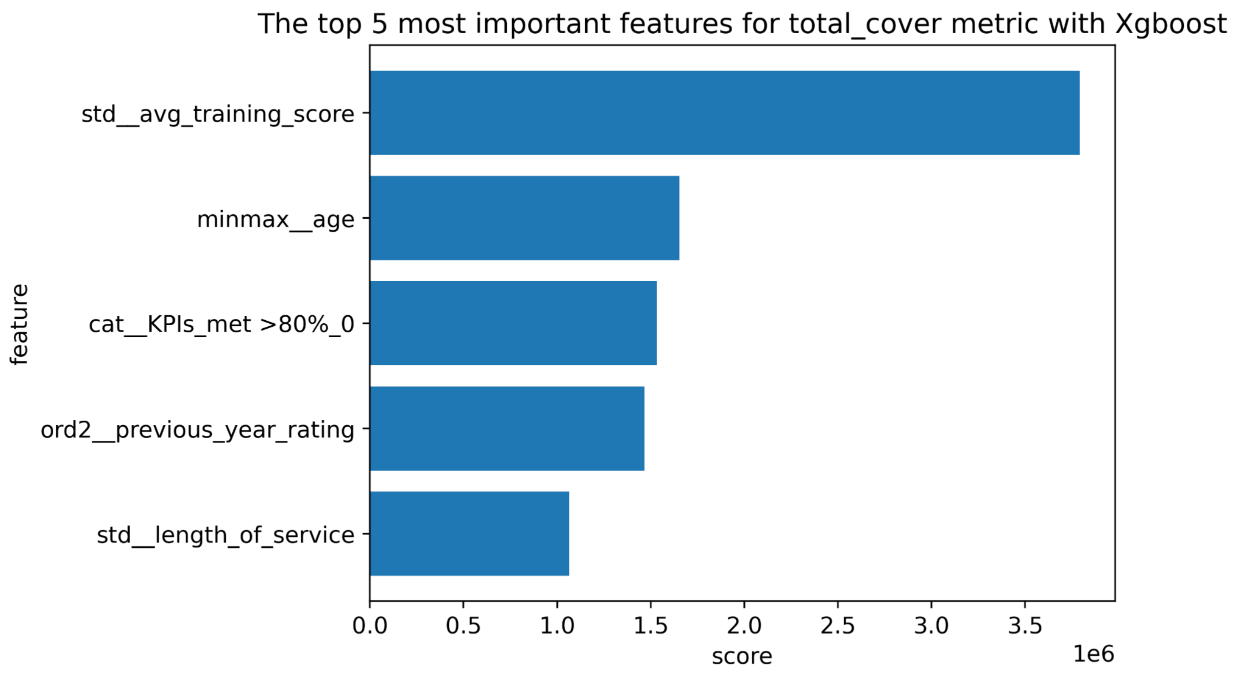

Figure 8. Top 5 most important features with Xgboost relative importance measure - total_cover

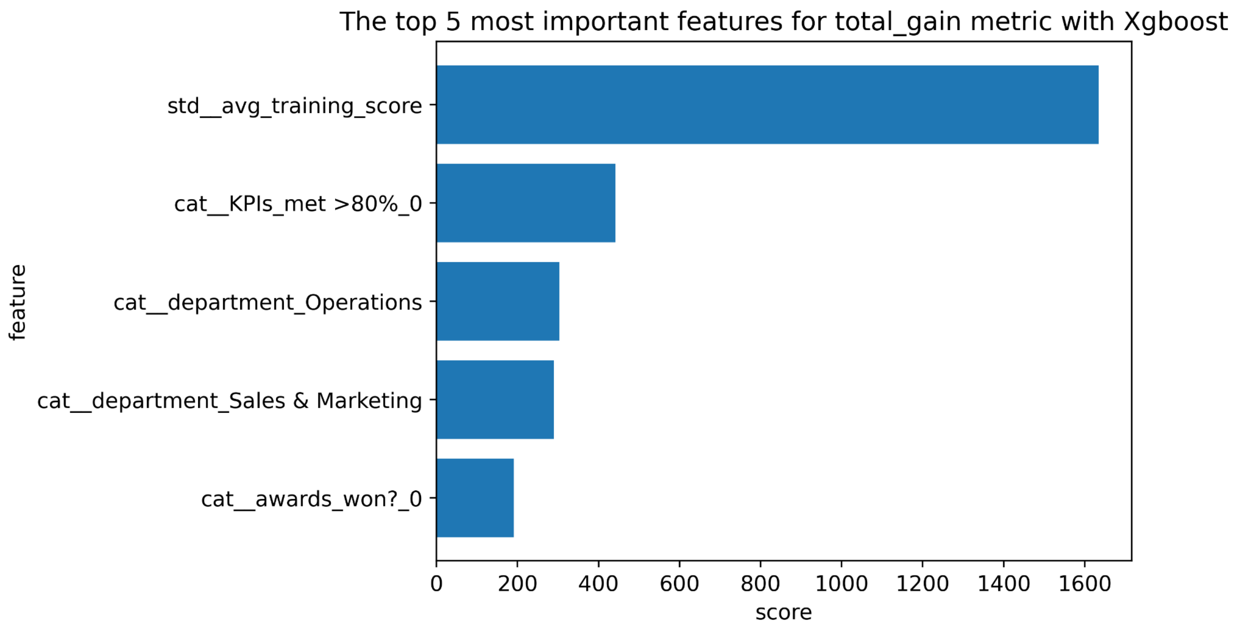

Figure 9. Top 5 most important features with Xgboost relative importance measure - total_gain

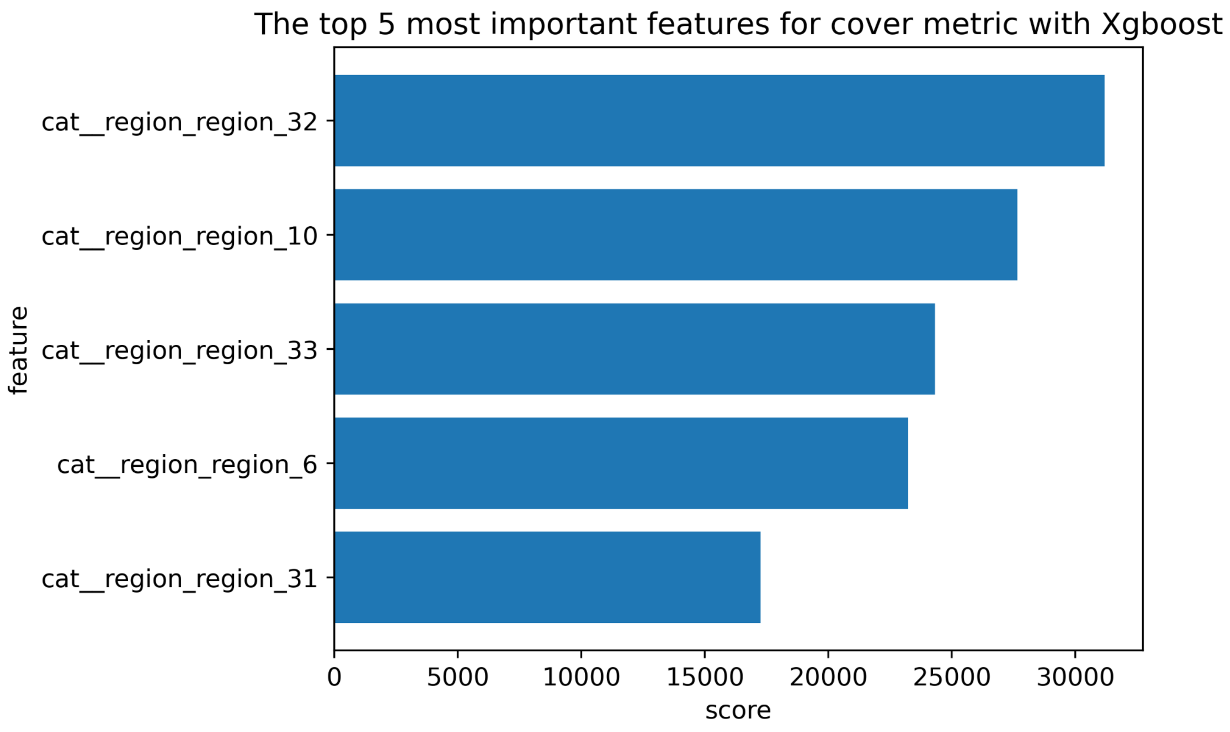

Figure 10. Top 5 most important features with Xgboost relative importance measure - cover

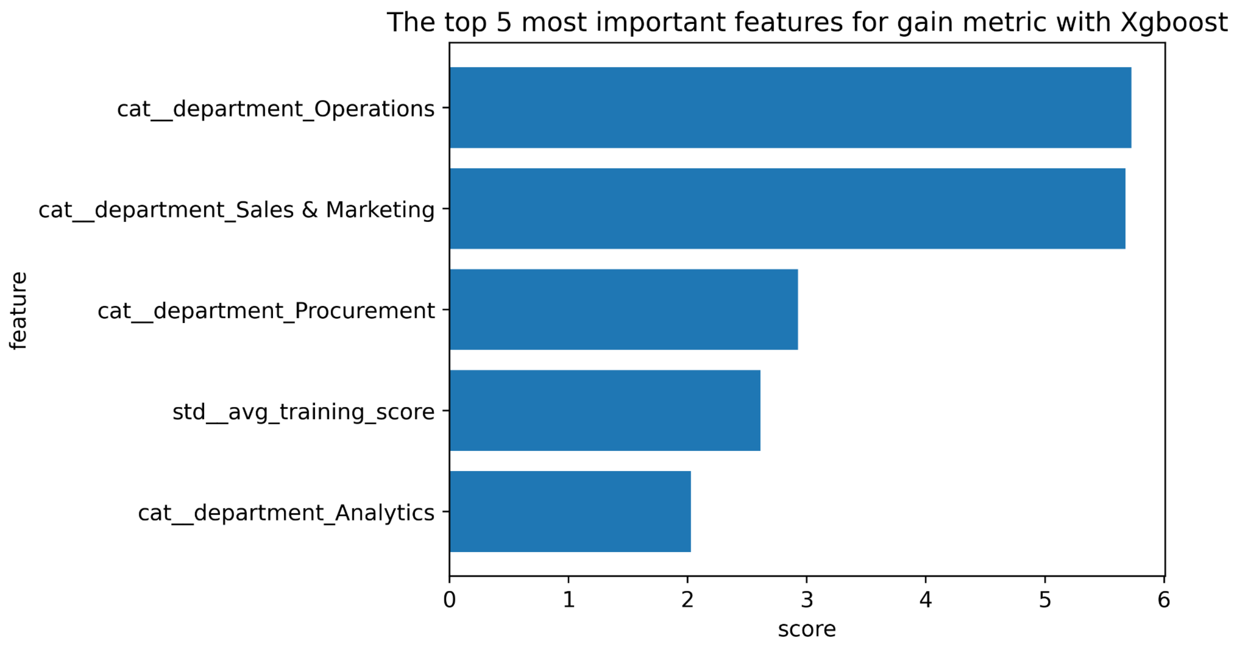

Figure 11. Top 5 most important features with Xgboost relative importance measure - gain

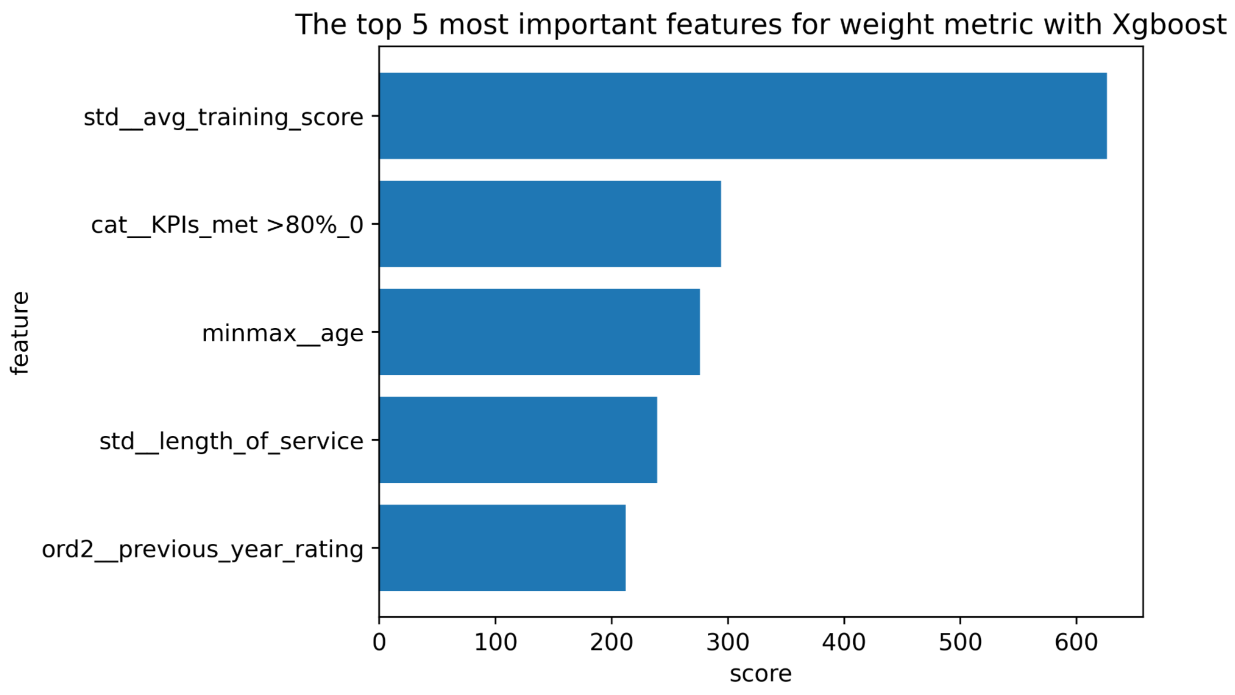

Figure 12. Top 5 most important features with Xgboost relative importance measure - weight

From the figures above, we can see that avg_training_score is always in the top 5 most important features for different global feature importances (only not in 'cover'). In many cases avg_training_score is the most important feature (rank 1st) and its score. We can easily interpret this as avg_training_score plays the most important role when deciding whether an employee is going to be promoted or not. Companies will record the training scores for each employee and make decisions based on the scores. This prevents the HR department from being biased on employees' age and length of service in the company, only focusing on numerical scores they collected. Also the department employee stays at, whether the employee has won an award or whether the employee meets KPI for the most of time seems to be important, which aligns with the common ssense. Some departments may be assigned with more important jobs and employees in these departments can be promoted easily. The other features that are not mentioned in the figures are not that important. It will be hard and meaningless to figure out the least important feature in this case (the scores are close to each other). Surprisingly, number of training is not an important feature when inspecting global feature importance.

In the next step, we take a look at local feature importance with *SHAP*.

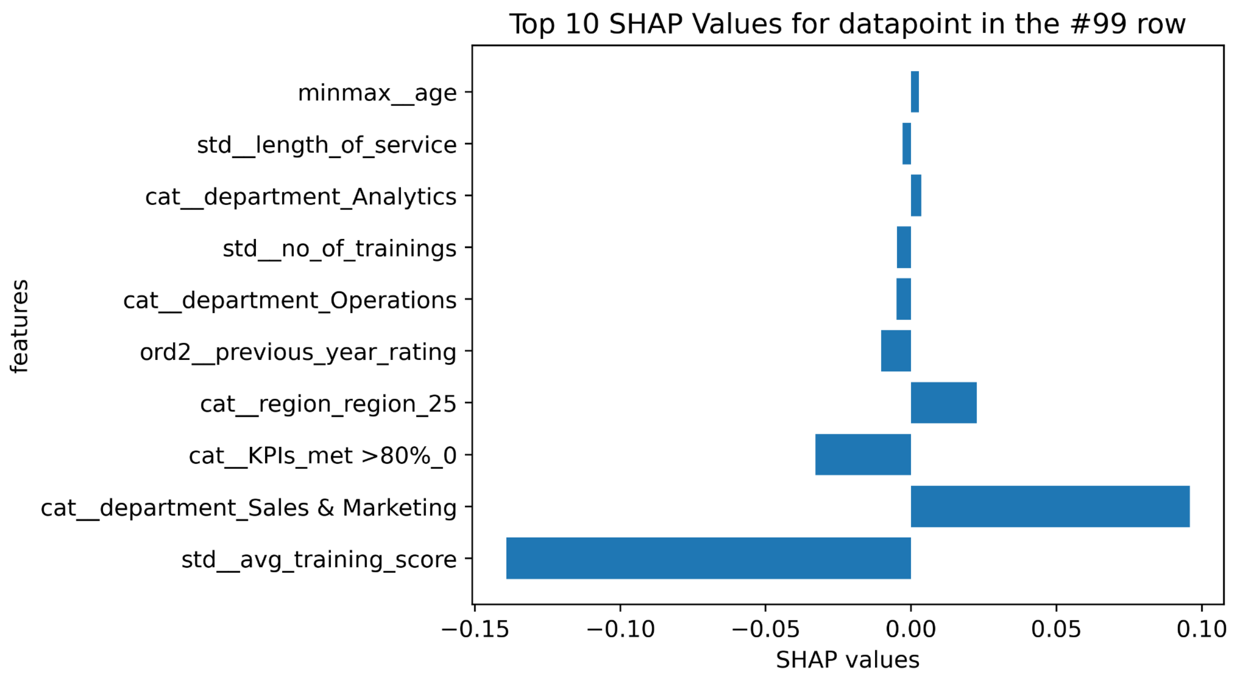

Figure 13. Features with top 10 highest SHAP values for local feature importance at data point on the 99th row of test set

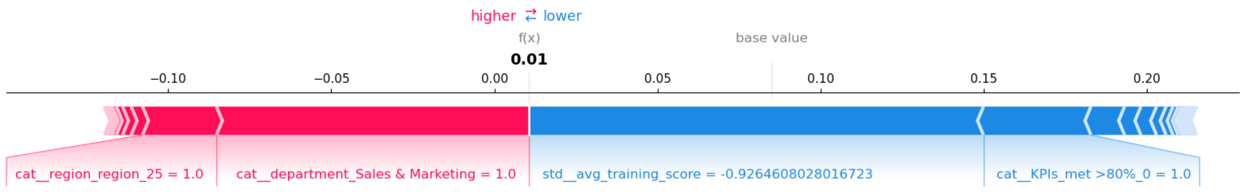

Figure 14. The Force Plot for the data point on the 99th row of test set

From the above two figures we can see that avg_training_score still plays an important role in terms of local feature important. In the force plot we can see that being in Sales & Marketing department contributes positively to the prediction result and the feature average training score contributes negatively to the prediction result. The predicted result is non-promoted as we can see from the numerical value under f(x). We can interpret it as the average training score of this employee is not that good and it results in the employee not getting promoted ,where the magnitude of the impact is large . The KPI feature is acting in a similar way but with a less magnitude of impact. Cat__gender_m (one-hot encoded) is the least important feature for this data point.

## Outlook

Due to the limit of computation power of the laptop, the range of values for hyperparameters being tuned is not that large. In the future, it is possible to try to tune other hyperparameters or a wider range of hyperparameters. In order to get a better performance in the future, we are also inspired to try other models. For example, support vector machines or deep neural networks. Deep neural networks nowadays are very powerful and we can expect to get a different view of this project when using it. Another way that can potentially help with the performance is to resample/augment the dataset. We can apply SMOTE (Synthetic Minority Over-sampling Technique) on this dataset.

One of the potential weak spots of my modeling approach is that I only consider missing value cases for ordinal data. If the model is finally deployed and there is a data point with missing values in numerical features, the model will fail to work. In the future, there should be more details handling cases like this. We can add imputers[3] in advance to stop this from happening. We can also collect employees mariage status, and previous employers' information  to improve model performance

## References

1. https://www.kaggle.com/code/tarunsolanki/hr-analysis-eda-models/notebook
2. https://www.kaggle.com/code/ashwin4kaggle/hr-analysis-data-cleaning-eda-visualization
3. S. van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate Imputation by Chained Equations in R", J. Stat. Soft., vol. 45, no. 3, pp. 1–67, Dec. 2011.