Helping Managers and HR save time using Machine Learning

Introduction to the competition problem:

Link: https://datahack.analyticsvidhya.com/contest/wns-analytics-hackathon-2018-1/

Our client is a large MNC, WNS Analytics, and they have 9 broad verticals across the organisation. One of the problems they face is identifying the right people for promotion (only for manager positions and below), and preparing them in time. Currently the process they are following is:

1. They first identify a set of employees based on recommendations and past performance.
2. Selected employees go through the separate training and evaluation program for each vertical. These programs are based on the required skill of each vertical.
3. At the end of the program, based on various factors such as training performance, KPI completion (only employees with KPIs completed greater than 60% are considered) etc., employee gets promotion

As we can see, the whole process takes a lot of time. We can help the company save time and find the right candidate by determining the most suitable candidates at the checkpoint. We will employ a data-driven approach to predict if a particular employee should be promoted or not. Our predictions will be based on a couple of features related to the employees.

The problem of competition is formulated as a binary classification problem. The evaluation metric for the competition is F1 score.

Understanding and Preprocessing the Data:

Initially, the train dataset has 13 columns. Test dataset has the same columns except for is_promoted. Predictions on this test dataset are submitted to the competition. Following are the descriptions of each of the train data columns:

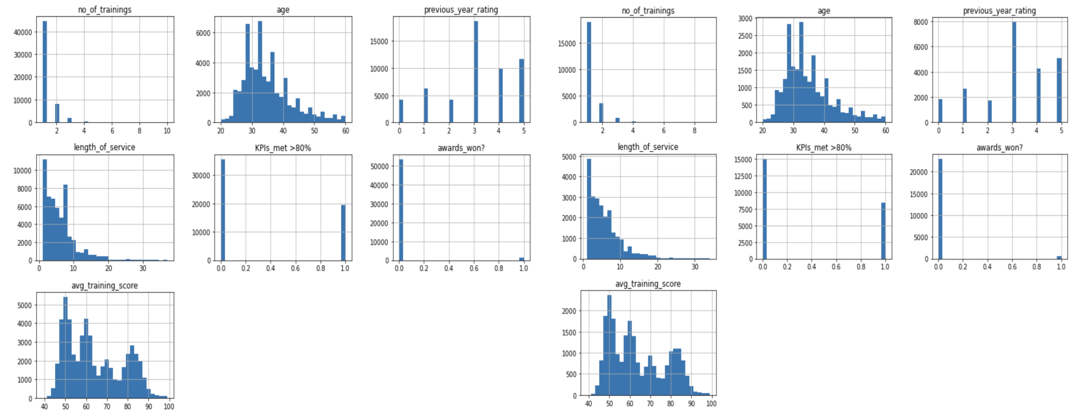
Variable	Description
employee_id	unique id for employee
department	department of employee
region	region of employment (unordered)
education	education level
gender	gender of employee
recruitment_channel	channel of recruitment for employee
no_of_trainings	number of other training completed in the previous year on soft skills, technical skills etc.
age	age of employee
previous_year_rating	employee rating for the previous year
length_of_service	length of service in years
kpis_met>80%	if percent of kpis(key performance indicators) > 80% then 1 else 0
awards_won?	if awards won during previous year then 1 else 0
avg_training_score	average score in current training evaluations
is_promoted	(target) recommended for promotion

First and foremost, we observe the distribution of our target variable in the train set i.e. is_promoted. We observe that there is a huge imbalance with 50140 points belonging to one class and only 4668 points belonging to the other class. This makes sense as only a minority of the employees get promoted.

First and foremost, we check for duplicates and missing values. Fortunately, we don't find any duplicates in the dataset. However, there are 2409 and 4124 missing values in 'education' and 'previous_year_rating' columns respectively. Real data is important so we can't drop these rows. We have come up with ways to impute missing values for both of the columns. For missing values in the 'education' column, we have imputed 'Bachelor's' because this is the mode for 'education' column by a large degree. As far as ''previous_year_rating'' is concerned, we have imputed missing values with a 0. The assumption we are making here is that this employee has just joined, and they don't have a previous year rating.

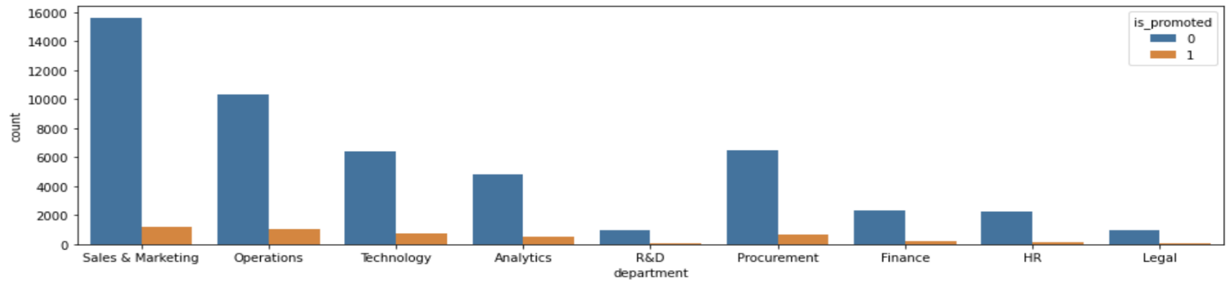
Next, we plot distributions of numerical columns for both train and test set. Distributions look similar, which is a positive sign.

Train: Test:

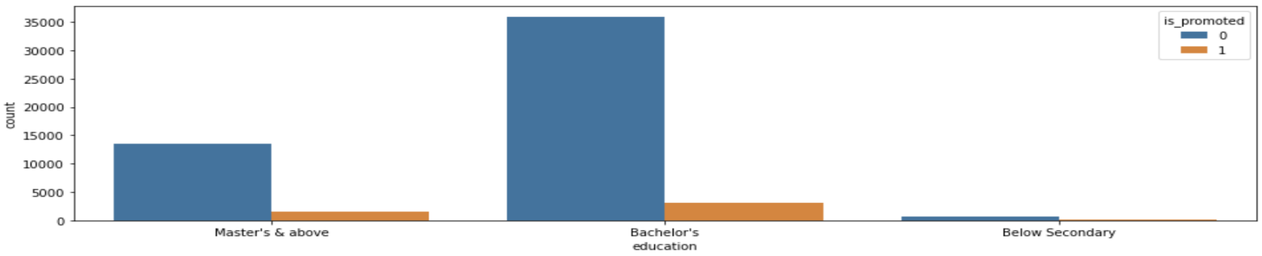


After that, we count plots for all categorical variables, which give us interesting insights for feature engineering and missing value imputation (mode of education for missing).

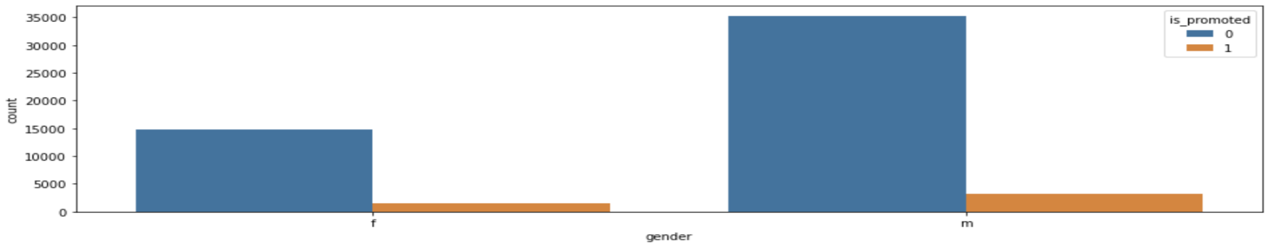
Department:



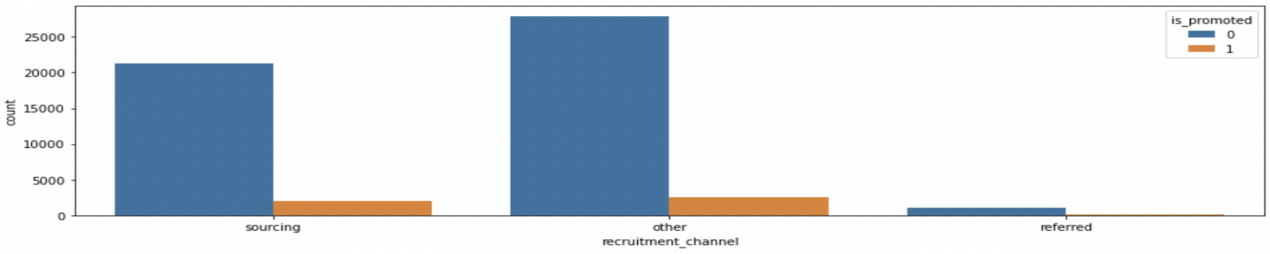
Education:



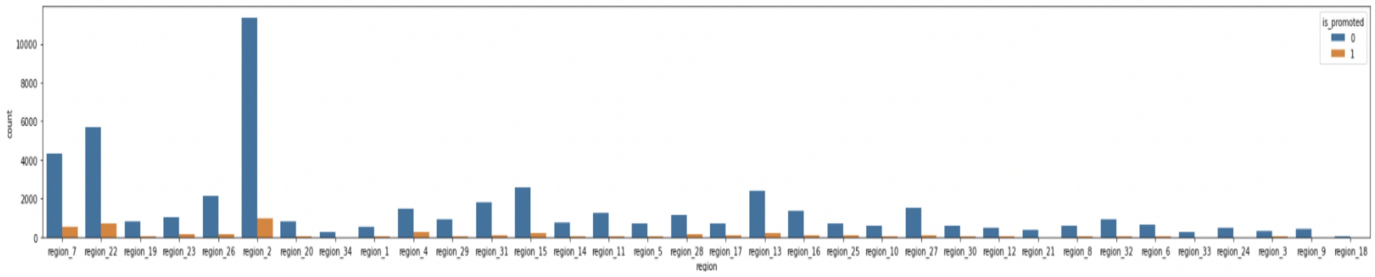
Gender:



Recruitment_Channel:



Region:



We need to encode categorical columns to numbers. We apply one-hot encoding to columns ‘department’, ‘gender’ and ‘recruitment_channel’. We haven’t encoded the ‘region’ column as the number of categories is quite large and one-hot encoding can lead to high memory consumption and curse of dimensionality. Instead, utilizing the count plot, we have engineered three columns from the ‘region’ column namely ‘region_2’, ‘region_22’ and ‘region_7’ as these regions dominate our dataset by a huge degree, and including other regions might be redundant. As far as the ‘education’ column is concerned, we have encoded it using Label Encoding as the data is ordinal in nature for this column.

As we previously mentioned that there is a huge imbalance, we need to generate synthetic data in order to create a class balance for training. We have employed Synthetic Minority Oversampling Technique (SMOTE). SMOTE works by randomly picking a point from the minority class and computing the k nearest neighbors for this point. The synthetic points are then added between the chosen point and its neighbors. The dataset is balanced now, and we can move to model building and evaluation.

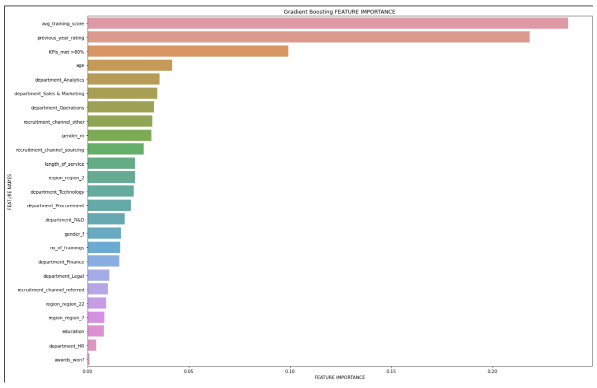
Model Building and Evaluation:

Out of the training set provided, we hold out 5% for testing purposes. We have decided to apply three models namely Gradient Boosting (scikit-learn), CatBoost and LightGBM. All of these are gradient boosting models that fit boosted trees by minimizing the error gradient. Each model has their pros and cons but these three are generally being used currently in the industry as base learners (along with XGBoost, AdaBoost, etc.) before combination to perform ensemble learning.

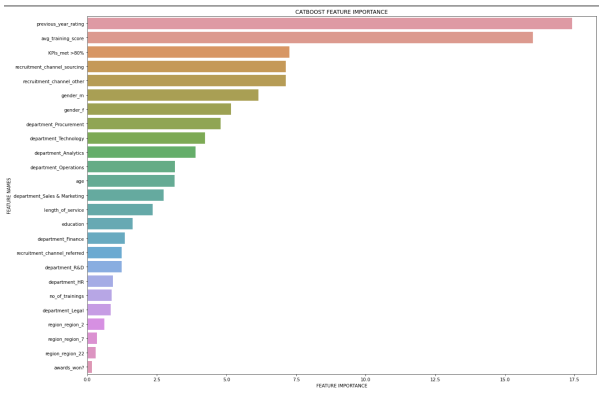
We have performed Random Search Cross Validation for all three models to generate the best set of parameters using hyperparameter optimization. Random Search was the best option considering it is computationally less expensive than Grid Search. However, there is quite a bit of chance that we might have missed better parameters due to the random nature.

We have output the best parameters into a csv and saved model objects by serializing them using pickle so we don’t have to fit them again and again. Following are the results and feature importances of all three models:

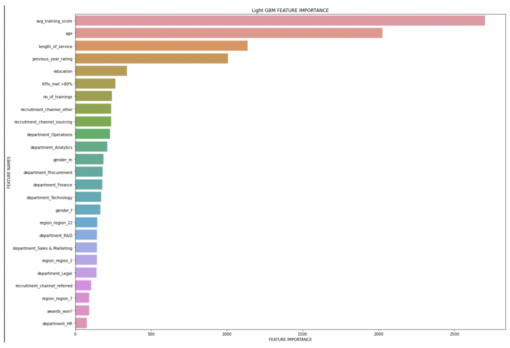
Gradient Boosting:
F1 score calculated by competition portal: 0.465
Feature importances ->



CatBoost:
F1 score calculated by competition portal: 0.4523
Feature importances ->



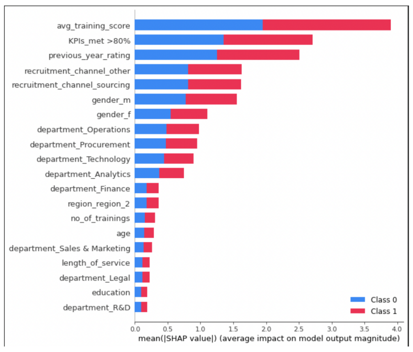
LightGBM:
F1 score calculated by competition portal: 0.4528
Feature importances ->



We are getting F1 scores in the range of 0.45-0.46. Feature importance plots vary a bit but highest feature importance columns remain more or less the same in all models.

SHAPLEY values for interpretability:

We have computed global shapley values for the LightGBM model..



According to the global interpretation, we see that ‘avg_training_score’, ‘KPIs_met>80%’, ‘previous_year_rating’, ‘recruitment_channel_other’, ‘recruitment_channel_sourcing’ were top 5 variables contributing most to the model outcome and the extent to which they influence the decision of the model is huge. We should perform more feature engineering on the top columns in this plot.

Ensemble Learning:

In order to improve F1 scores even further, we will use three different variations of ensembles namely Hard Voting, Soft Voting and Stacking. An ensemble can make better predictions and achieve better performance than any single contributing model. Ensembles are heavily used during competitions in hope that they will improve the metric.

Hard Voting:
Prediction by the majority of all models is used as the final prediction.
F1 score calculated by competition portal: 0.4639

Soft Voting:
Average of predictions by all models is used as the final prediction.
F1 score calculated by competition portal: 0.4690

Stacking:
Predictions of all models are input to a simple Decision Tree Classifier and final prediction output by the model.
F1 score calculated by competition portal: 0.4602

Conclusion:

Ensemble learning showed improvements. Soft Voting has given us the best F1 score (~0.47) of all models. I believe that the use of Hyperparameter optimization and both basic and advanced Ensemble techniques have helped in getting a better F1 score as compared to a single model with no tuning. However, I think there still is a huge room for improvement. This score is still ~0.05 less than the best score of the competition. At a high level, I think even the best score for this competition is not good enough in order to get a model deployed in a real life setting. WNS Analytics should consider adding in more data both in terms of features (for example punctuality, difficulty of projects worked on) as well as more points in both the classes.

Additional strategies we can try/tried but failed:

1) More feature engineering
2) XGBoost (tried but dependencies issues when xgb package interacts with sklearn)
3) Blending Ensemble
4) Extend Model interpretability using SHAPLEY values (did global for LightGBM but should do local also and on CatBoost and GradientBoosting as well)
5) Bayesian Hyperparameter Tuning (tried but again dependencies issues for hyperopt library)
6) Deep Neural Network (tried but too heavy on the computation)