DNB Data Scientist for Enterprise Nanodegree Program

Capstone Project

John Gunnar Blostrupsen
December 5th , 2018

I. Definition

Project Overview

Nowadays, in all the hype on neural networks, artificial intelligence, machine learning and data science, it is important not to forget that there still is need for us humans to play along with the new technologies. For example, in the domain in which I have worked for the last 16 years (banking), the turnover of employees, especially the youngest ones, is accelerating. If a company really wants to stay competitive, it is crucial to know how to prevent key employees from leaving the company. Therefore, it is important to have insights into what are the main causes for employees to choose to quit. Such insight will help companies to keep more of their valuable employees. The conventional way to keep talents is increasing salaries and/or offer bonuses, but today's working environment demand other approaches. There are many opinions on this – one example can be found here at forbes.com.

There are quite a few articles to be found on the subject, and I would like to make a reference to this one at www.inc.com, where it is said that about 75 % of the reasons for costly voluntary turnover actually come down to things that managers can influence.

I would like to get further insight on this by using techniques I have learnt the last few months, attending Udacity's Data Scientist for Enterprise Nanodegree Program. (Suitable enough, attending this course (and acknowledging the fact that my employer is willing to prioritize investments in such re-skill programs) has in itself made me more eager to stay with my current employer ☺).

Problem Statement

In an increasingly competitive job market, with continuous competition for attracting valuable competence and resources, it is important to not only attract the right people, but also keeping the employees you got. It is therefore important to acquire insight into why some employees choose to leave.

The reasons for employees to decide to quit may be many and complicated to comprehend, but there should be possible to extract some insight on this from data that contains information related to a company's employees, and I found a dataset on kaggle.com that I want to dig into, in order to try to understand reasons for employees to leave a company – focusing on finding eventual main reasons and try to make predictions on whether an employee will quit or not.

I consider this to be a classic classification problem, as I want to predict a label (whether an employee leaves or not), and I will also use the data given for each employee to find the main reasons for turnover, by investigating eventual correlations in the data.

The strategy for solving this problem will be to choose appropriate models to compare against each other, and then pre-process the data in order for it to fit the different models, by applying techniques related to the imbalance of the dataset, such as RobustScaler and Over-sampling techniques as SMOTE (Syntethic Minority Over-Sampling Technique). I also will apply OneHot encoding on the categorical features, to prepare for Machine Learning. Furthermore, tuning of some parameters in the models will be carried out, and finally I will compare the outputs/results in order to find the most appropriate model to solve this problem.

Metrics

As the dataset is quite imbalanced (3571 out of 14999 (23.8 %) employees have left the company), it is appropriate to look at Precision, Recall and F1-score metrics when evaluating the results, rather than the Accuracy. Simply labelling all employees as "will not leave" would give us an accuracy of 76.2 % - but it would not make any sense.

I will also use Cross-Validation, due to the relatively small datasets I get after splitting into train- and test sets, in addition to addressing the data imbalance by using the oversampling the SMOTE algorithm (Synthetic Minority Over-Sampling Technique). Such oversampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample.

Thus, looking at false positives (employees I predicted left, but didn't leave) and false negatives (employees I predicted as didn't leave, but left), and comparing models on precision score, recall score and accuracy score (adjusted by resampling) will represent my metrics foundation.

II. Analysis

Data Exploration

The dataset for this project was downloaded from Kaggle.com, and it is a pretty clean dataset, containing 14999 rows by 10 columns. There are no missing data, and the majority is numerical data. Only two columns have string format ('object'): 'sales' (department) and 'salary'. The features 'sales, 'salary' and 'work_accident' are categorical. Here is a sample of the dataset, before I did any adjustments:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	sales	salary
0	0.38	0.53	2	157	3	0	1	0	sales	low
1	0.80	0.86	5	262	6	0	1	0	sales	medium
2	0.11	0.88	7	272	4	0	1	0	sales	medium

Here is a sample of the dataset, after I renamed a few columns and changed column order, for my own visual convenience in jupyter notebook:

	satisfact_level	last_eval	number_projects	average_monthly_hours	time_spent_company	work_accident	promotion_last_5years	dep	salary	left
0	0.38	0.53	2	157	3	0	0	sales	low	1
1	0.80	0.86	5	262	6	0	0	sales	medium	1
2	0.11	0.88	7	272	4	0	0	sales	medium	1

Here is a basic description of the features:

	satisfact_level	last_eval	number_projects	average_monthly_hours	time_spent_company	work_accident	promotion_last_5years	left
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.021268	0.238083
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.144281	0.425924
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000

satisfact_level: from 0.1 (lowest) to 1.0 (highest).

last_eval: managers' rating of employee, where 1.0 is the highest

number_projects: how many projects the employee is involved in

average_monthly_hours: time spent on monthly average

time_spent_company: number of years employed

work_accident: 0=no, 1=yes

promotion_last_5years: indicates whether an employee had a promotion last 5 years (0=no, 1=yes)

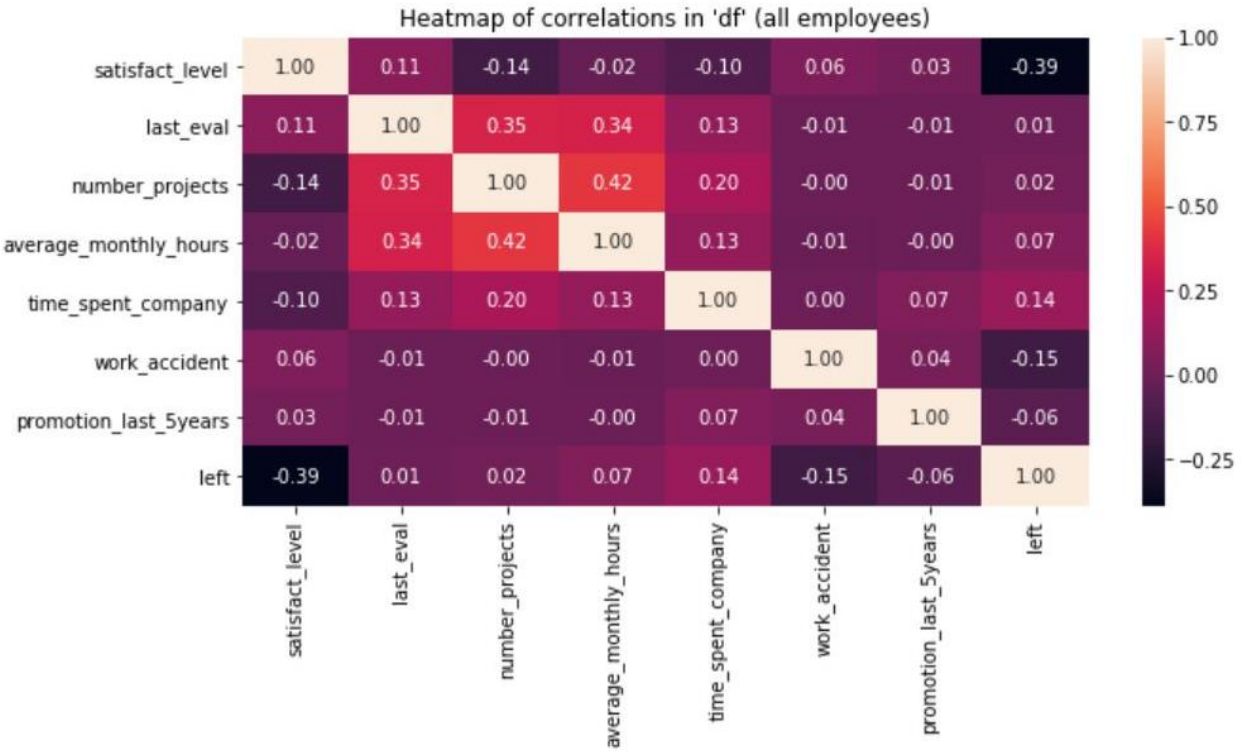
dep: departments in the company

salary: low/medium/high

left: The classification / label which is binary (left=True->1, left=False->0).

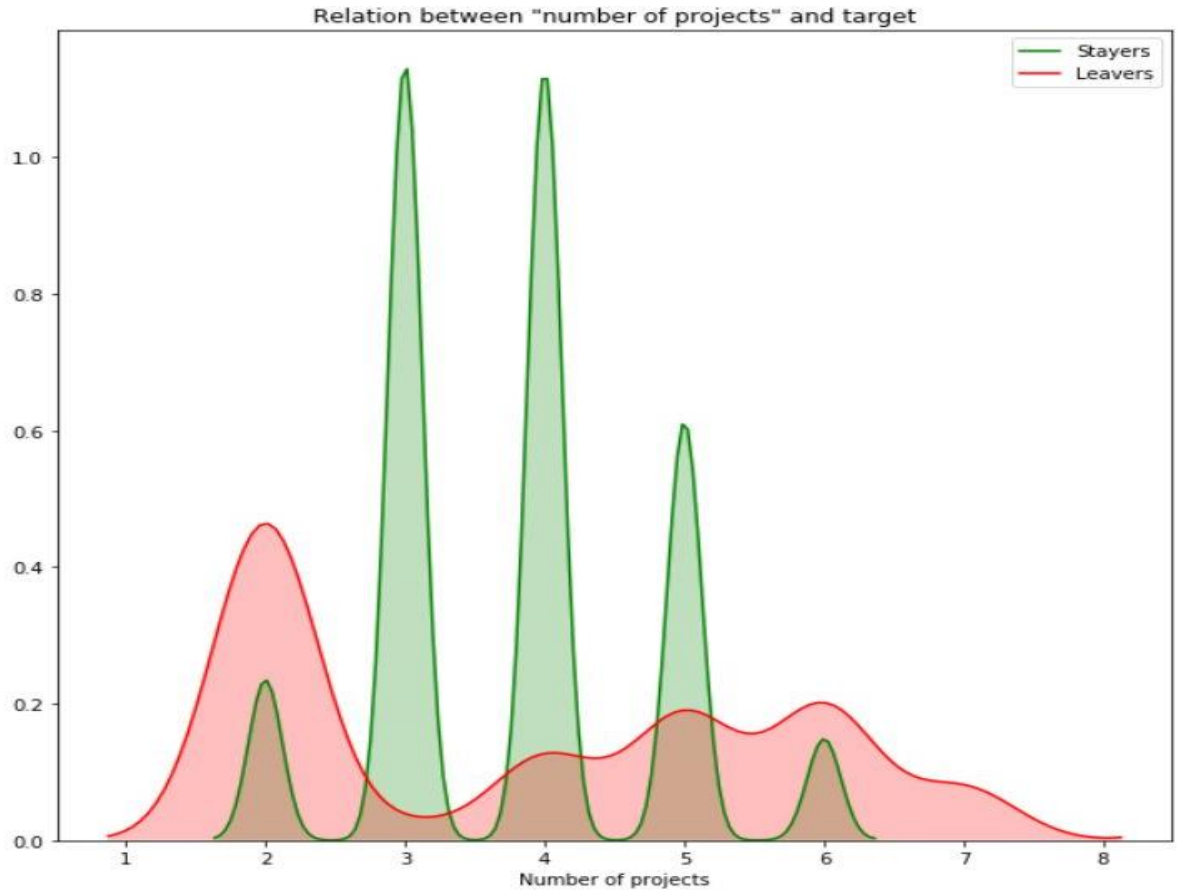
Exploratory Visualization

I created a heatmap to visualize any correlations between the features in the dataset.



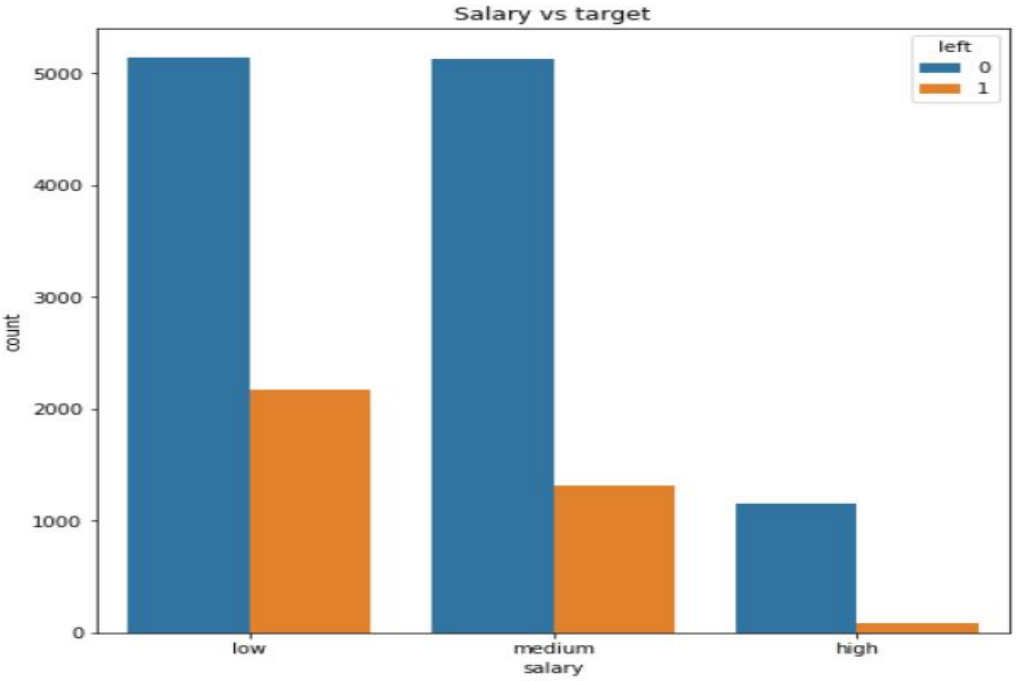
The heatmap indicates a negative correlation between ‘satisfact_level’ and ‘left’, implying that those who stays tend to be more satisfied (not surprisingly). We can also see the obvious connection between ‘number_projects’ and ‘average_monthly_hours’. Interestingly, it seems that both these two features correlates with ‘last_eval’, indicating that the employees with the highest score on the last evaluation are those who spent more hours and took part in more projects.

By looking more closely on the features that seems to correlate the most with ‘left’ in density plots, one may gain more insight:

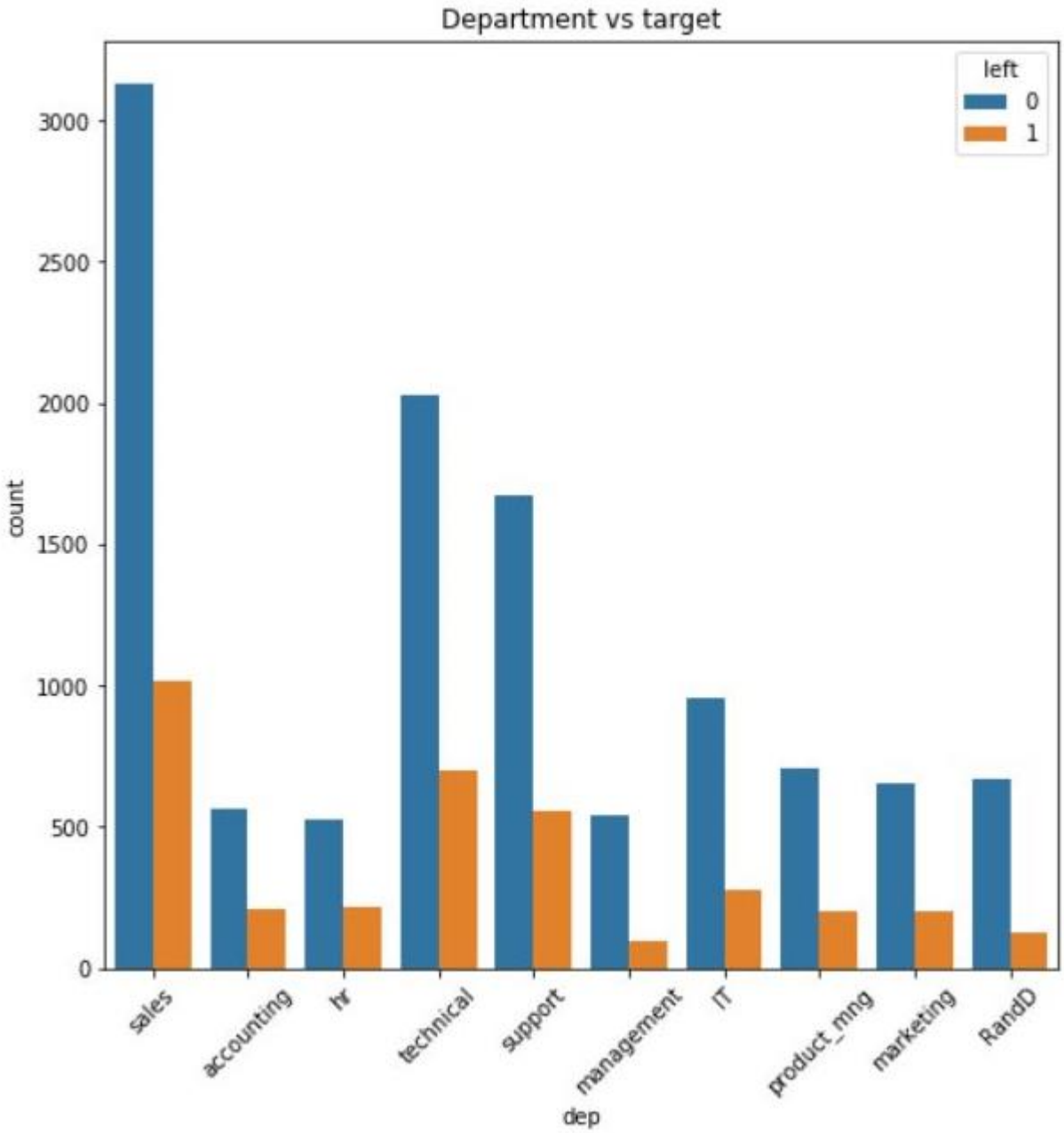


The density plot above shows that among those who have left, most were involved in either 2 projects or more than 5-6 projects.

As the features ‘salary’ and ‘dep’ are not shown in the heatmap (because they aren’t numerical), I want to look at how these are distributed, using countplots:



There are fewer “leavers” among those who are in the “high” salary category.



Most of the employees that leave come from the sales, technical or support departments.

Algorithms and Techniques

As I am here looking at a classification problem, and want to predict a label (whether an employee leaves or not), I will use supervised classification algorithms. I have chosen three different models for performance evaluation: Logistic Regression, Decision Tree Classifier, and Random Forest Classifier. Here follows a brief description and an explanation of the reasons for choosing these models:

Logistic Regression use the logistic function (or the sigmoid function), which creates an S-shaped graph that maps values between 0 and 1, converting the log-odds to probability. I

want to use this model as a supplement to the base rate/best naive solution as a basis for comparison to the other models, as Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables, and suits situations with a dichotomous dependent variable, and no high correlations among the predictors.

Decision Tree Classifier is a classification technique, where the tree branches are represented by conjunctions of features, leading to the tree leafs that represent the class labels (here: whether an employee left or not). I want to use this model as it should suit both this problem domain and dataset pretty well (i.e. there are no missing values in the dataset), and decision trees handles both categorical data and numerical data. It also offers the possibility of validation through statistical testing.

Random Forest Classifier is an ensemble method that uses decision trees, by creating several decision trees and outputs the mean prediction and the classification of each tree – correcting for the decision trees eventual overfitting on the training set. I want to use this model because as it acts as a collection of decision trees, whose results are aggregated into one result. Thus, it is a more robust model than a single decision tree, as it may provide a more accurate and stable prediction.

It is also advisable to do a cross-validation when carrying out such analysis on smaller data sets. When splitting up the dataset into training- and test-sets, we have fewer datapoints to train the algorithm on. K-fold Cross-validation is a way to handle that issue, as it divides the data into K folds, then use K-1 sets for training while the remaining set is used for testing. This is trained and tested several times, and resulting in the average result from the sets.

Benchmark

As a reference, I will use base rate/best naive solution, as 76.2 % of the employees in the dataset have not quit, such an approach would give 76.2 % accuracy, guessing the majority class since I have an imbalanced dataset, that is - if I presumed that no one would leave. (Of course, I aim to get better prediction than that.)

III. Methodology

Data Preprocessing

As mentioned, the dataset here is quite small and imbalanced (just 23.8 % of the rows are related to the employees that left the company). This calls for some preprocessing before the models can handle the analysis in an efficient way. One technique related to this is to resample the data. Such resampling may be carried out by Random Under-Sampling or Random Over-Sampling. Random Undersampling balances class distribution by randomly eliminating majority class examples. This is done until the majority and minority class instances are evened out. Random Over-Sampling increases the number of instances in the minority class by randomly replicating them in order to present a higher representation of the minority class in the sample. The lattest is the most appropriate approach, due to the relatively small dataset.

I chose to use the Syntethic Minority Over-Sampling Technique (SMOTE), which briefly works like this: Synthetic samples are made from the minor class. Then, the algorithm selects at least two similar instances (by applying a distance measure). Then, it alters an instance, one attribute at the time, by a random value within the distance to the closest neighbor instances.

Before resampling, the dataset had 14999 rows, including 3571 rows related to the employees that left the company (left=1). This I divided into training set (11999 samples) and test set (3000 samples). The reason for splitting the data before resampling is that I do not want the synthetic samples to go into the testing set. After I resampled the data, I had 18268 rows in the training set. Because I want to weigh the classes equally, the ratio is set to 1.0.

Furthermore, I used sklearn.preprocessing.RobustScaler to scale the features in the dataset, because it uses statistics that are robust to outliers. It removes the median and uses the interquartile range.

Implementation

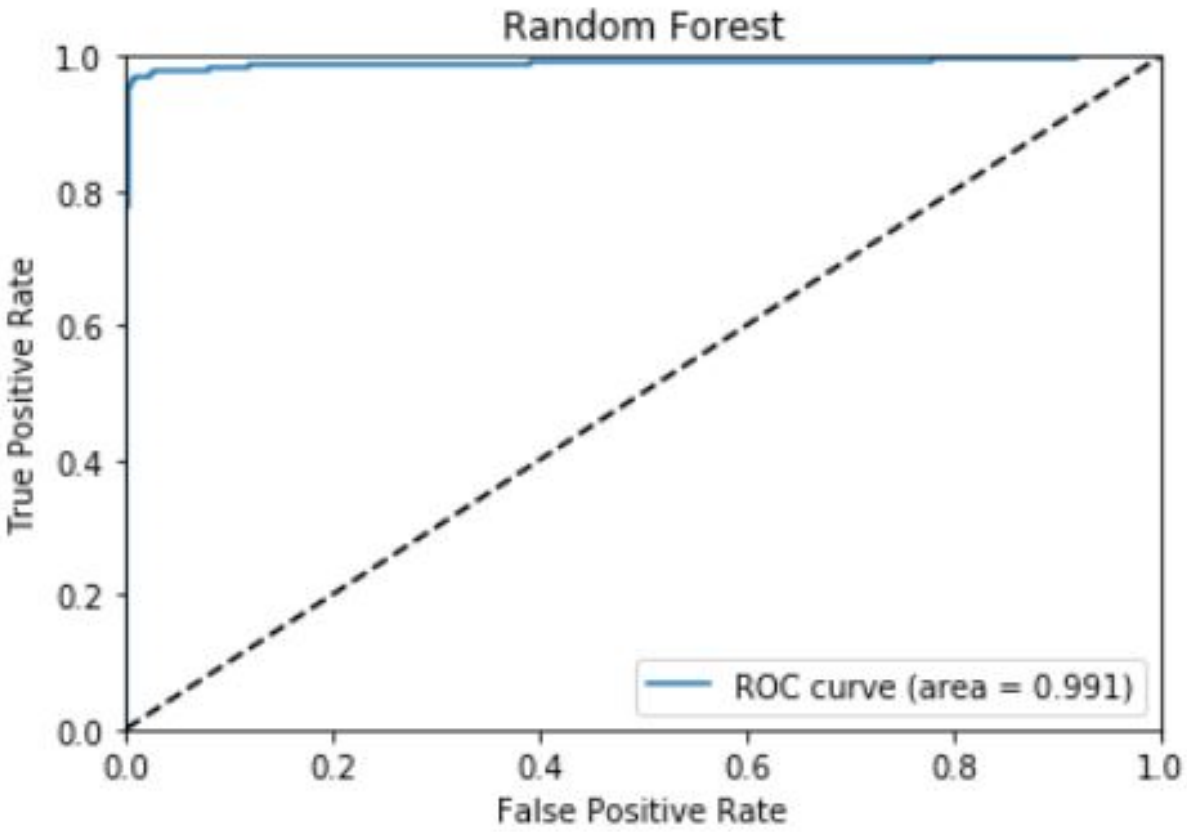
I trained and tested the three chosen models with and without cross-validation, to identify eventual differences, and to find the model that works best regarding prediction, by comparing the chosen metrics to each other. In my Decision Tree Classifier, I set min_samples_split=10, because in sklearn.tree.DecisionTreeClassifier the default value is set to 2, representing the minimum value of the argument. That is, I need minimum two observations to consider a split obviously. As I wanted more samples than 2 to be present in my data for a split to occur, I set 10 as a start, and went from there. For the categorical data, I made binary columns per category by using sklearn.preprocessing.OneHotEncoder. I also applied sklearn.preprocessing.RobustScaler to scale the features in the dataset. I think

my quite simplified approach to this project was the main reason for me not experiencing any severe complications during this coding, and by using default parameters to a great extent.

Refinement

I chose to run GridSearch for my Random Forest Classifier, as this seems to be the most accurate model, and there is not much time to save by implementing an easier model. I adjusted these hyper-parameters: “max_depth”, “n_estimators”, “min_samples_split”, “bootstrap” and “min_samples_leaf”.

Here is a visualization of the ROC:



This is a plot of the true positive rate against the false positive rate for the different possible cutpoints of the dagnostic test, where the area under the curve (0.991) is a measure of accuracy.

Applying feature_importances_ shows that just a few features are found to be of importance - ‘satisfact_level’, ‘time_spent_company’, ‘number_projects’, ‘average_monthly_hours’ and ‘last_eval’, in that descending order. The rest of the features are of small importance.

IV. Results

Model Evaluation and Validation

Comparing the outputs from the models – first, without cross-validation, then with cross-validation, and then trying the model on the initial features – here are my findings:

Without Cross-validation:

Model	Accuracy	Precision	Recall
Logistic Regression	0.765333	0.500883	0.803116
Decision Tree Classifier	0.97	0.927777	0.946176
Random Forest Classifier	0.984	0.978198	0.953258

With Cross-validation:

Model	Accuracy +/- std. deviation
Logistic Regression	0.788899 +/- 0.008584
Decision Tree Classifier	0.973997 +/- 0.004856
Random Forest Classifier	0.984915 +/- 0.004136

Testing Random Forest Classifier on original dataset:

Metric	Score +/- std. deviation
Precision	0.976713 +/- 0.019338
Recall	0.976132 +/- 0.017412
Accuracy	0.975199 +/- 0.017124

Random Forest Classifier is the model among the three that makes most precise predictions whether an employee will leave the company or not.

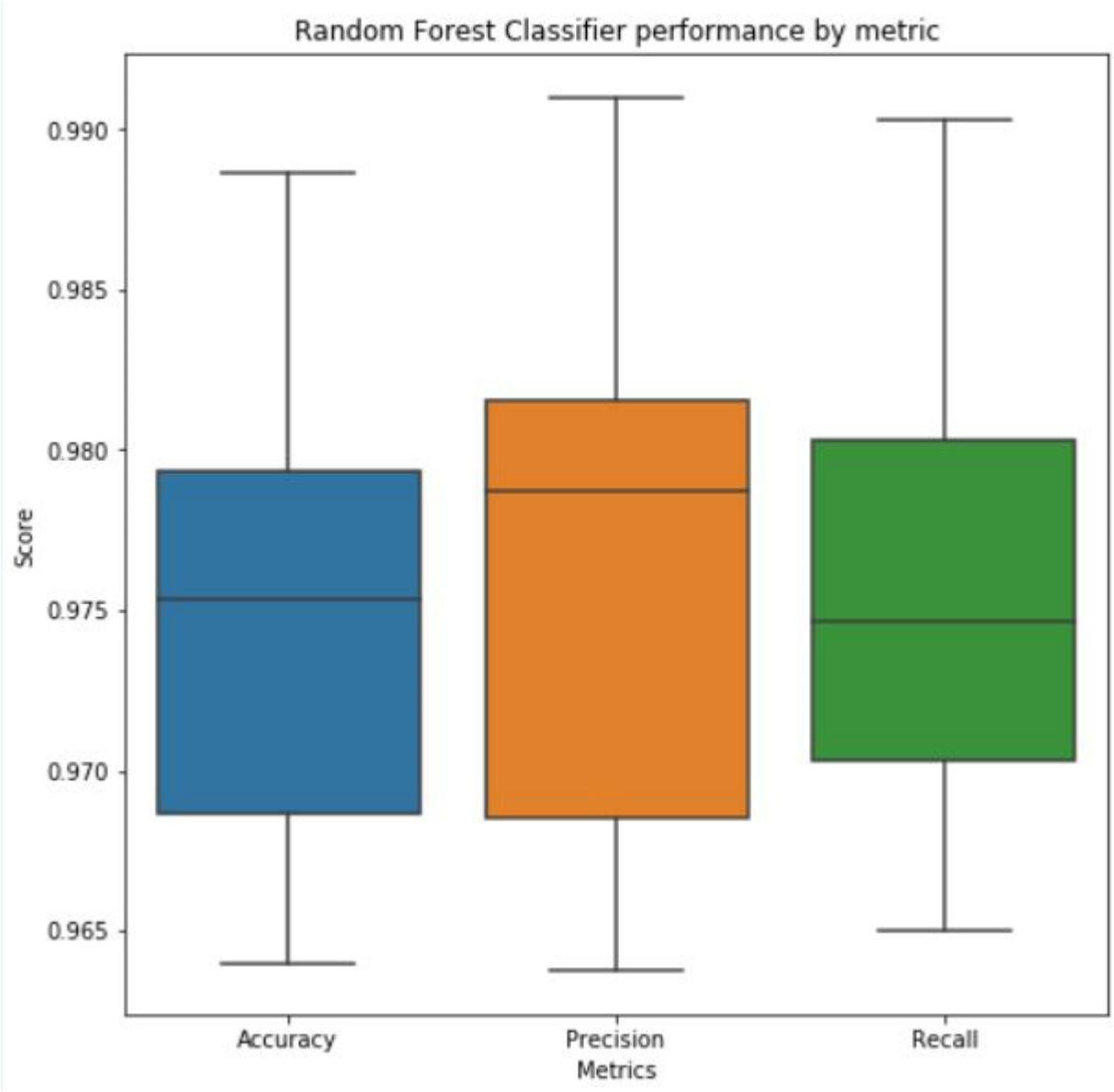
Justification

Random Forest Classifier works pretty well at predicting which employees that will choose to leave the company, with a precision, recall and accuracy of more than 97 %. Compared with the benchmark on 76.2 % accuracy, I can say that the model works quite well.

V. Conclusion

Free-Form Visualization

Here, I would like to visualize the performance of the Random Forest Classifier model, to show how it performs against the benchmark (the best naive solution) on 76.2 %. Here we can see that both precision, recall and accuracy are above 97 %:



Reflection

Initially, I found it hard to find a capstone project that was easy enough for me to grasp, as I still am quite new to Python programming, and yet comprehensive enough in order to meet the capstone assignment requirements. This lead to quite a few iterations on the capstone proposal. When I landed that, I think the project was fun to work with, as I on the way had to read up on how to handle imbalanced datasets, and also what models are best applicable to different situations. I also found it a bit challenging to conduct the cross-validation, as I cannot recall doing that before.

I think my solution, using Random Forest Classifier to predict whether an employee will choose to leave the company or not, is good enough regarding solving this classification problem.

However, I recognize the complexity of such processes (people changing jobs), and I want to emphasize that an employer should not trust such a method fully on its own, but rather use it as input in the continuous process of getting more knowledge on what makes people want to stay in a company. One still have to keep people satisfied and having the feeling of being seen and appreciated.

Improvement

I would possibly try out some more different resampling techniques, to see how it may affect the outcome. Eventually, as I hopefully get more confident in Python programming and machine learning algorithms, as well as basic statistics, I probably would like to try to test other approaches, such as neural network techniques or models like XGBoost and LightGBM. But for now, I had to simplify my Capstone Project as much as I could, in order for me to reach the finish line for this program.

References

https://www.kaggle.com/datasets

https://towardsdatascience.com/beyond-accuracy-precision-and-recall-3da06bea9f6c

https://imbalanced-learn.org/en/stable/generated/imblearn.over_sampling.SMOTE.html

https://beckernick.github.io/oversampling-modeling/

https://www.jair.org/index.php/jair

https://en.wikipedia.org

https://stackabuse.com/cross-validation-and-grid-search-for-model-selection-in-python/

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html