Predicting why employees leave prematurely

Udacity Machine Learning Nanodegree

-Siddharth Pratap Singh

October 7 2017

Background and Definition

Domain Background:

It is very important for an establishment to understand the mindset of their employees. To know the reasons behind the premature leaving of the employees is a major objective of the companies. If understood well, this can help in increasing the employee productivity and overall growth of the company.

Problem Statement:

The employees have been leaving from the company and the problem is to understand and predict if an employee will probably leave or not.

The reasons behind leaving any company can be many. Some of these can be less number of projects that a person was involved in, if the people have promotions in a very long time, if they are frequently asked to work overtime in a company where they are not satisfied with their work quality, a number of bad evaluations from the managers etc. These factors along with many others are involved in contributing towards what kind of a relationship an employee has with his/her company.

If we have information about such factors like mentioned above, we can get a hint as to what might have been an underlying relationship between different factors, or which ones of them contribute the most. This dataset presents us with some of such factors. We have values ranging from time spent at the company, to average monthly hours that an employee spends at the company, whether they had a work accident that turned them off from the company or what kind of an evaluation they had the last time. The dataset also presents us salary and the department of the employee too.

The problem statement is simple. We want to understand and predict the reasons to know why the employees of any establishment leave. My goal is to understand which among these factors contribute towards this the most and predict if the employee will leave or not given this data about him.

Datasets and Inputs:

The data is taken from a Kaggle competition. A brief description of the features in the dataset :

- Satisfaction Level (ranges from .01 to 1 with 1 being the highest).
- Last evaluation (ranges from .36 to 1, presents us with a measurable quality of the employee as rated by the managers, 1 represents the best evaluation).
- Number of projects

- Average monthly hours
- Time spent at the company (number of years that the employee has been at the company).
- Whether they have had a work accident (1 if yes 0 if not)
- Whether they have had a promotion in the last 5 years (1 if yes 0 if not)
- Departments (column sales)
- Salary
- Whether the employee has left (1 for yes/0 for no)

The dataset contains these features' information about 14999 employees of which nearly 24% have left. We have information about the employees who have left (nearly ¼ of the dataset contains information about such employees) and we can use this information to make a prediction on other employees, i.e if they will leave or not by comparing these features against each other, comparing which features are independent and which are dependent etc.

Evaluation metrics:

False Positives: People we predicted as leaving but did not leave.

False Negatives: People we predicted as not leaving and left.

Since both the metrics are useful I compared the models on precision_score, recall_score, accuracy_score. Now accuracy score doesn't work well in imbalanced classes but I will be implementing resampling to even this accuracy paradox out.

Analysis

Data Exploration:

A sample of the dataset that I have is:

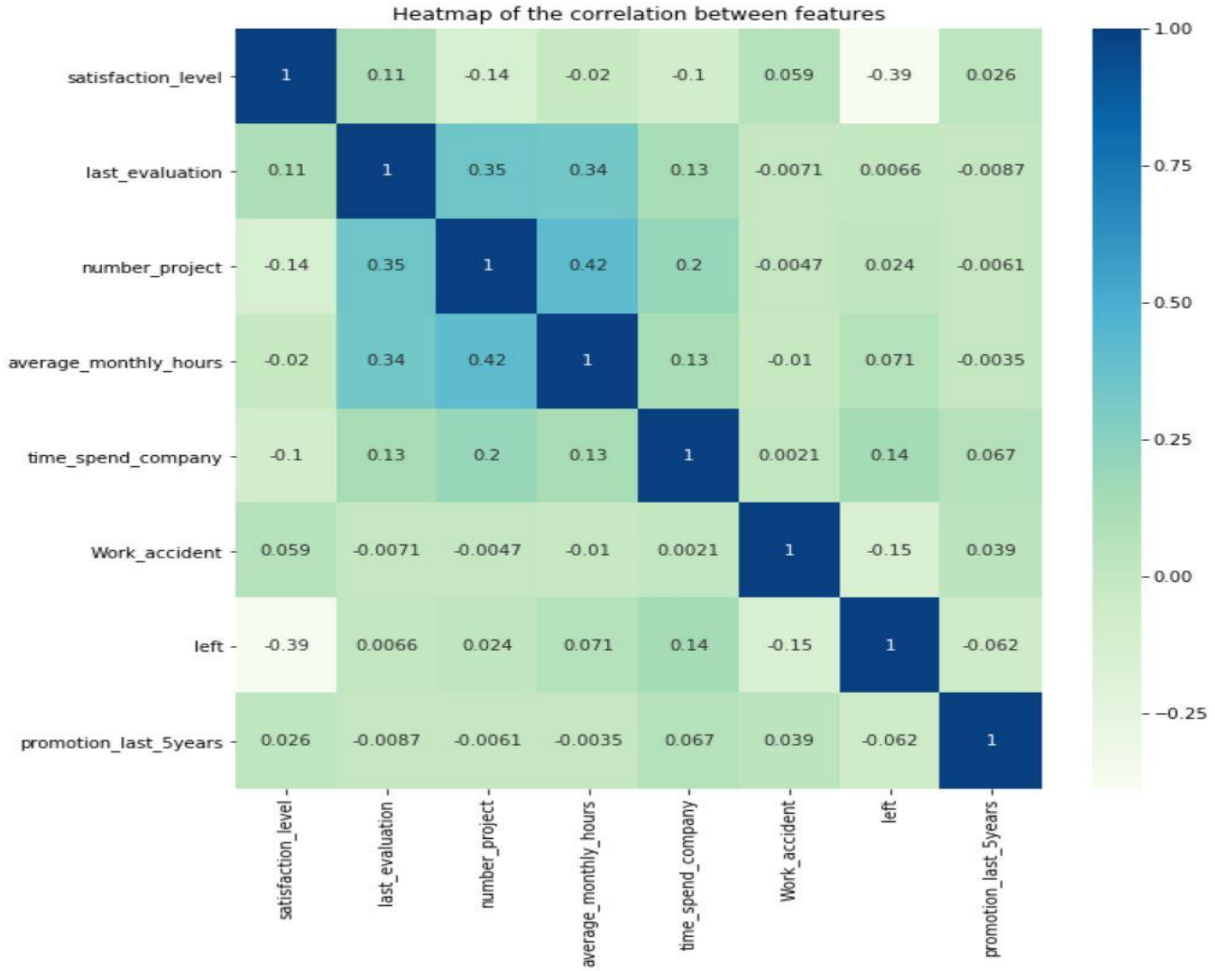
satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	left	promotion_last_5years	department	salary
0.38	0.53	2	157	3	0	1	0	sales	low
0.80	0.86	5	262	6	0	1	0	sales	medium
0.11	0.88	7	272	4	0	1	0	sales	medium
0.72	0.87	5	223	5	0	1	0	sales	low
0.37	0.52	2	159	3	0	1	0	sales	low

The basic description of these features and their values is:

	satisfaction_level	last_evaluation	number_project	average_monthly_hours	time_spend_company	Work_accident	left	promotion_last_5years
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	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
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

The total number of datapoints is 14999. The classification is binary (left=True -> 1; left=False -> 0). Salary, department and work_accident are categorical features with salary and department bring non-numerical.

Heatmap of the Correlation:

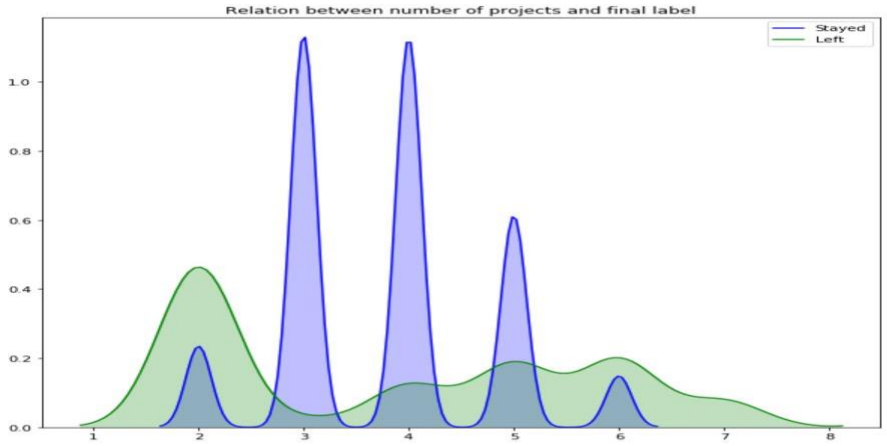


- The satisfaction level has an inverse relation with people leaving.
- The average_monthly_hours also has a direct relationship with number of projects which seems obvious as more projects would require more hours.

- The average_monthly_hours also has a direct relationship wth last evaluation. More the hours a person puts in, the better is the evaluation of the employee.
- The heatmap also indicates a direct relationship between number of projects and last_evaluation.

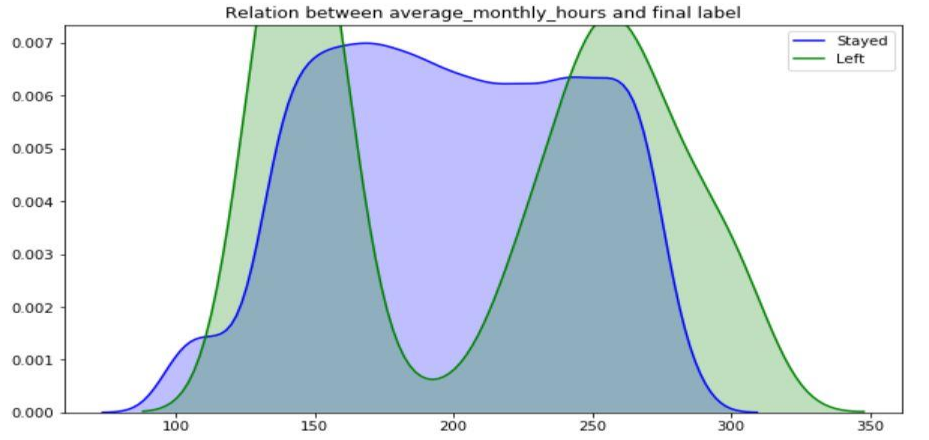
Density plots:

Number of projects Vs Left:



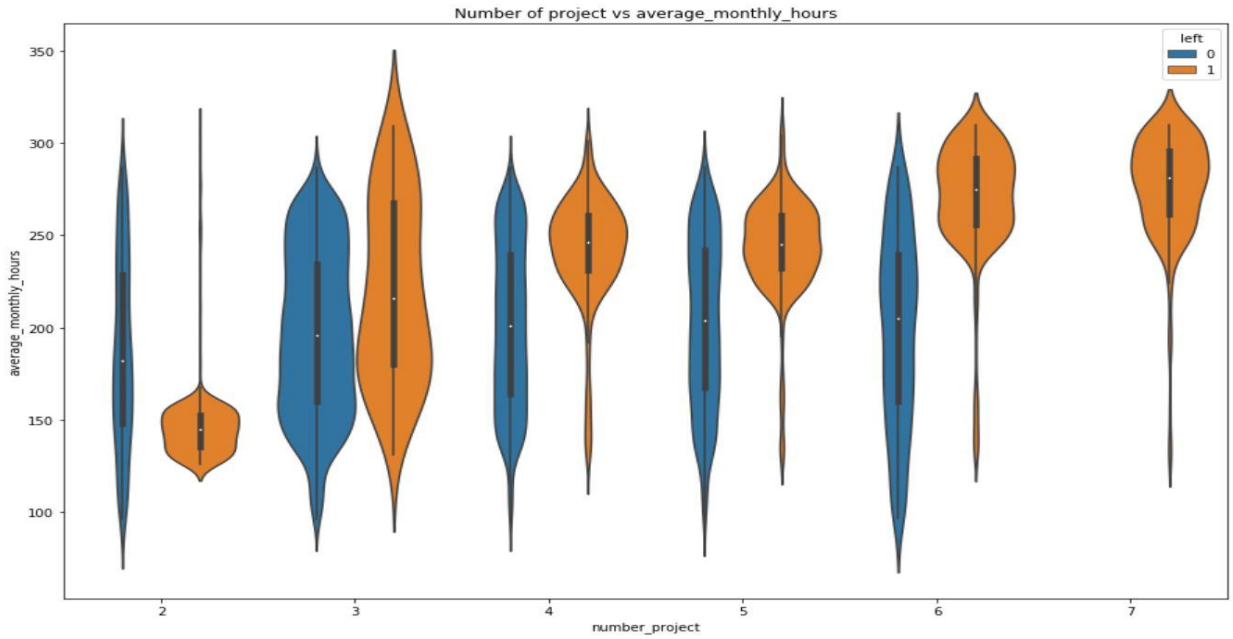
- People who left have either had very low projects or very high number of projects. Majorly people with only 2 projects tend to leave.
- People with 2 projects, and 6 projects tend to leave more.
- People with more than 7 projects are definitely seeming to leave.

Average monthly hours vs left:



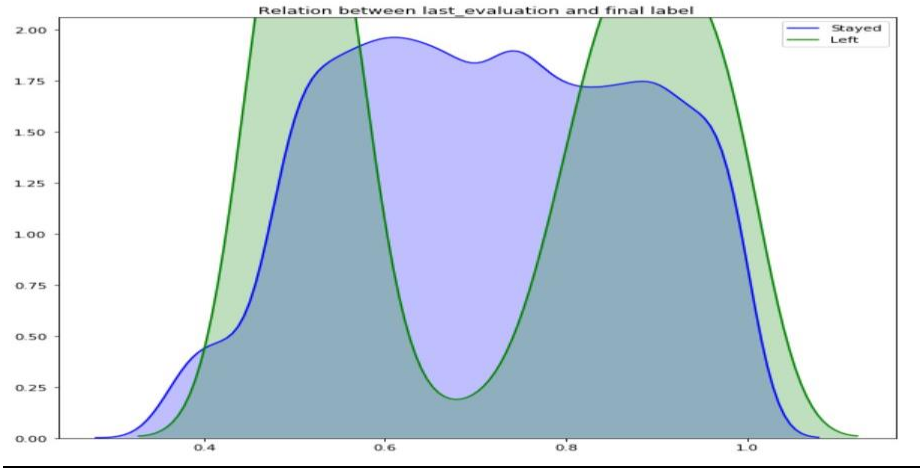
- People who left either had to put in least average monthly hours or very high monthly hours.
- The distribution suggests that a spike in the number of hours tends to make people leave.
- Could this be related to the number of projects? I will plot a number of projects vs average_monthly_hours plot with respect to the class label.

Average monthly hours vs number of projects:



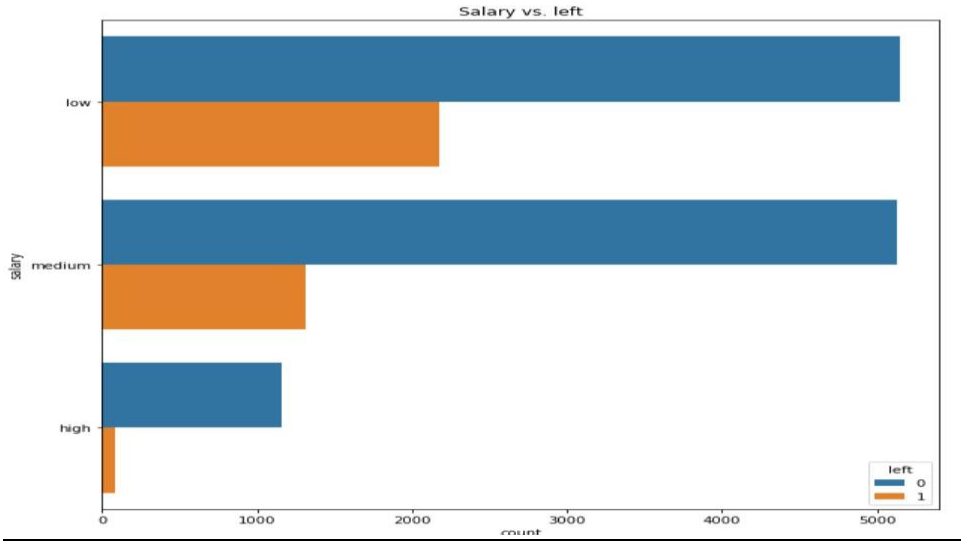
- This affirms my hypothesis that with the increase in the number of projects, the average monthly hours also get a spike which results in people leaving.
- Majority of the people who leave with projects more than 4 tend to have a high number of average monthly hours.
- We can see that there is a group with little average monthly hours and few projects who leave.
- There is also a chunk of people who don't leave and don't have work to do. (Enjoying eh?)

Last Evaluation vs left:



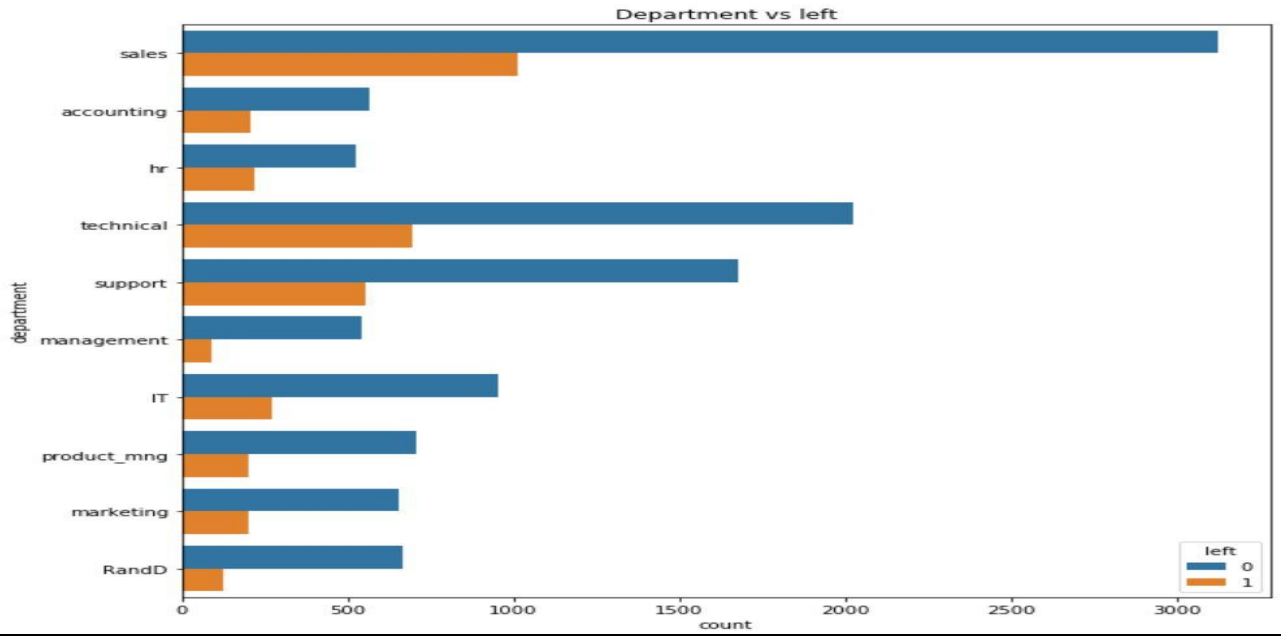
- There seem to be two kind of people, people who had a bad evaluation and left and people who had a good evaluation and left.

Salary vs Left:



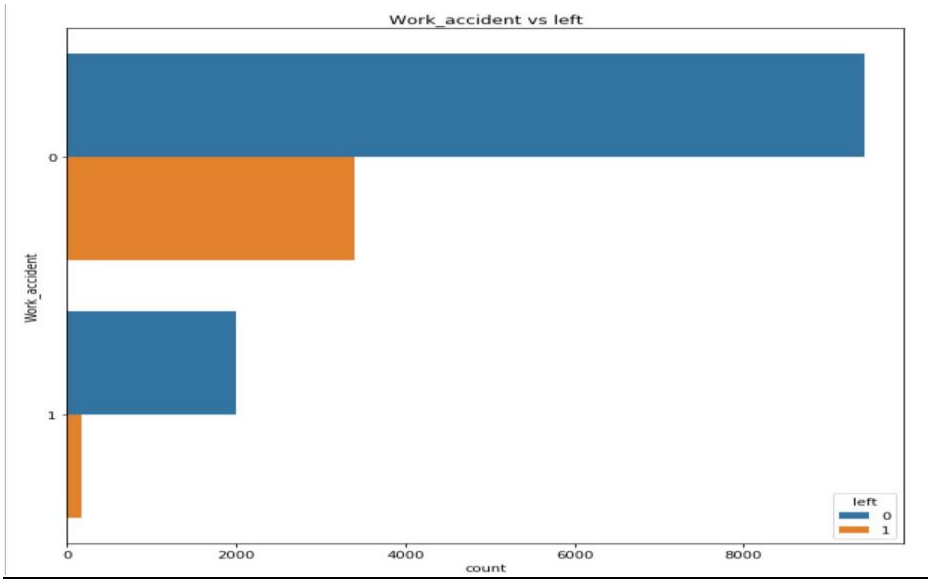
- People with low and medium salaries are the ones that tend to leave.
- The higher the salary, the number of people leaving the company decreases.

Department vs left:



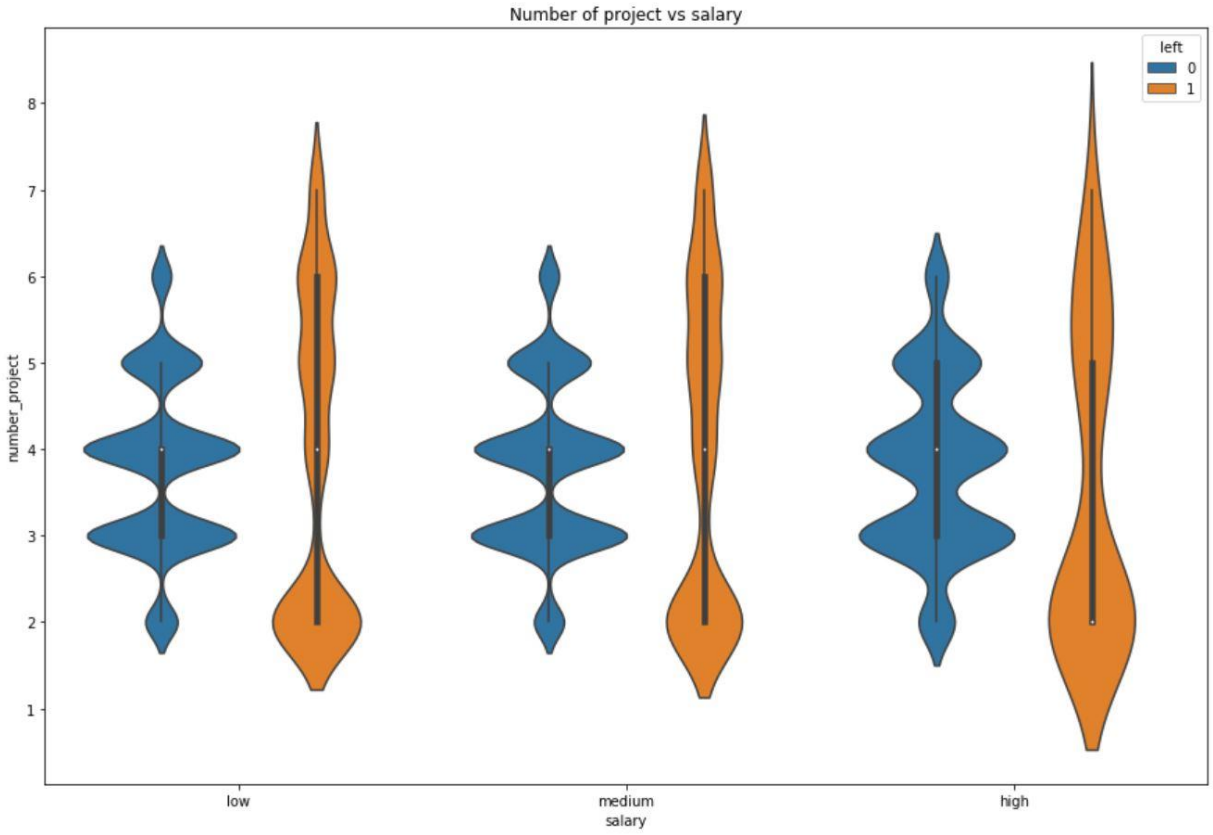
- The sales, technical and support are the top three departments whose people leave.

Work Accident vs left:



- There doesn't seem to be a good relationship that we can form out of this comparison.

Salary vs number of projects:



- People who left (either from low, medium or high salary) had either 2 projects or high while mostly ones lying in the 2 projects range.

Algorithms and Techniques:

Classification algorithms:

This problem is a classification problem. It can be solved using supervised classification algorithms. In this project I will be training:

1- Logistic Regression:

For binary classifications, this is a go-to method. Since our dataset if binary too, I will be using this method for classification. Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value).

2- Decision Tree Classifier: Tree models where the target variable can take a discrete set of values are called classification trees; in these tree structures, leaves represent class labels and branches

represent conjunctions of features that lead to those class labels. . Each element of the domain of the classification is called a class(in our case if employee left or not). A decision tree or a classification tree is a tree in which each internal (non-leaf) node is labeled with an input feature.

3- Random Forest Classifier: Ensemble learning refers to using multiple methods or algorithms to obtain better predictive performance. Random Forests is also an ensemble learning technique that uses decision trees. Random Forests operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set. Overfitting is a common problem with decision trees that is handled by random forests.

Cross Validation:

For very small datasets, splitting the data into training and testing results into less number of datapoints for the learning algorithm to train from. This problem is handled by using cross validation. The simple idea behind cross-validation is:

A test is still held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called k-fold CV, the training set is split into k smaller sets (other approaches are described below, but generally follow the same principles). The following procedure is followed for each of the k “folds”:

A model is trained using $k - 1$ of the folds as training data;

the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

Benchmark model:

The benchmark model is the base rate model. A simple heuristic to understand the minimum accuracy in predicting if an employee will leave or not will be predicting the majority label of the dataset. Here 76% of the employees fall in the not left label (0), hence the lowest accuracy would be to say that 76% would not leave. This will be the minimal accuracy on this dataset. I will be using this reference as my benchmark model.

Methodology

Resampling the imbalanced dataset:

Since we see that the data that we have is not much (nearly 15000 records) and also the classes are imbalanced, i.e only 24% of our data belongs to people who left. Now to make our learning more efficient and generalized, one of the ways to deal with imbalanced classes is to resample it.

Resampling is of four types:

1. Under-sampling the majority classes.
2. Over-sampling the minority class.
3. Combining over- and under-sampling.
4. Create ensemble balanced sets.

Under sampling is useful and feasible when the data set is large, nearly hundreds of thousands of datapoints. So I would be using over sampling for this project.

Now over-sampling is done in many ways but I used the SMOTE (*SMOTE: Synthetic Minority Over-Sampling Technique)* algorithm. This works (briefly) by creating synthetic samples from the minor class instead of creating copies. The algorithm selects two or more similar instances (using a distance measure) and perturbing an instance one attribute at a time by a random amount within the difference to the neighboring instances

Before I did resampling I had 14999 total records with only 3571 representing the left=True class. For the training set I had 11999 datapoints. After resampling, we have 9162 datapoints of each class with total of 18324 samples in our training set. I chose to equally weigh both the classes, that is the reason of choosing a ratio of 1.0 in implementing SMOTE:

```
sm=SMOTE(random_state=10,ratio=1.0)
training_features_sm,training_target_sm=sm.fit_sample(training_features,training_target)
```

It is interesting to note the resampling before splitting the dataset into training and testing is a bad method as these synthetic sample also go into the testing set. Then the model will be perfectly able to predict the testing set values and this would increase our recall and accuracy. Thus the better way is to first split the data into training and testing, and then run the resampling algorithms on your training set.

Scaling:

Applying features scaling to the dataset using Robust Scaler. The centering and scaling statistics of this scaler are based on percentiles and are therefore not influenced by a few number of very large marginal outliers. Consequently, the resulting range of the transformed feature values is larger than for the previous scalers and, more importantly, are approximately similar

Implementation:

Since, the data was small, I have trained and implemented all the 3 algorithms using cross-validation and without using cross-validation.

Results

Model Evaluation:

Without cross_validation:

1- Logistic Regression:

- Accuracy: 0.7606666666666667
- Precision: 0.5069444444444444
- Recall : 0.7956403269754768

2- Decision Tree:

- Accuracy: 0.9716666666666667
- Precision: 0.9513212795549374
- Recall : 0.9318801089918256

3- Random Forest:

- Accuracy: 0.982
- Precision: 0.9815864022662889
- Recall : 0.944141689373297

After Cross Validation:

Logistic regression: Accuracy: 0.7694433166196277 with (+/- 0.11309087193412021)

Decision tree: Accuracy: 0.9754657765628642 with (+/- 0.01811230284917215)

Random Forest Classifier: Accuracy: 0.9891328885925432 with (+/- 0.01636532892110572)

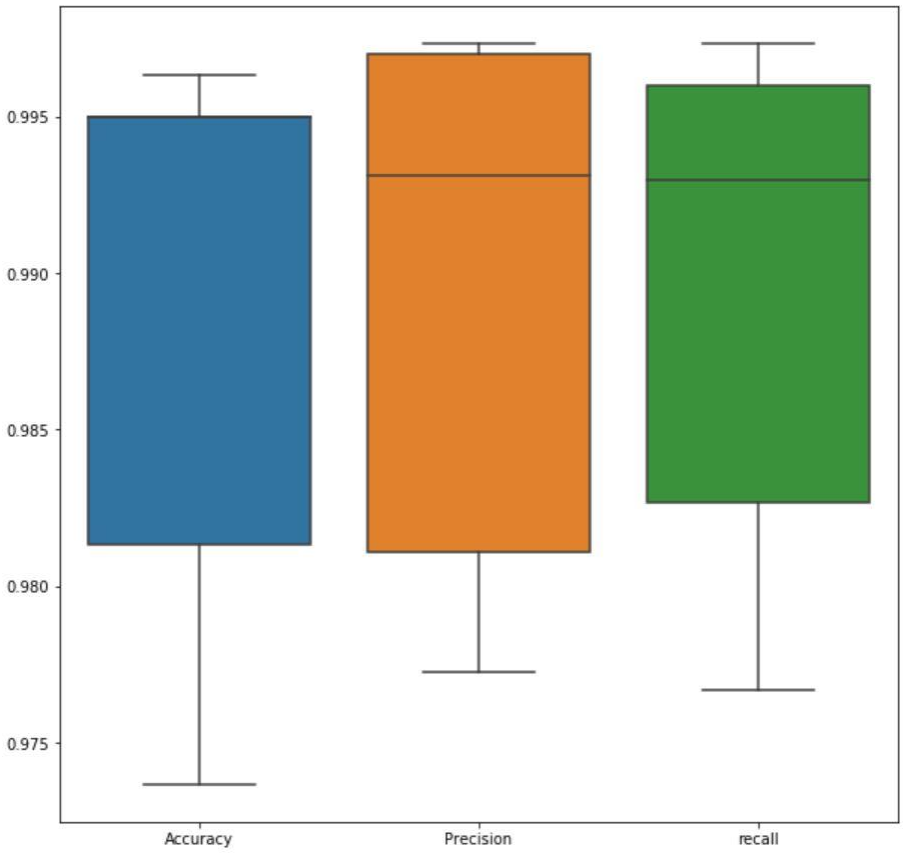
Justification:

The random forest work superbly with the data prediction. The accuracy is highest after applying cross validation with 98.9 % mean accuracy and .02 standard devation.

Conclusion

Free Form Visualization:

All the score depicted in the following figure for the Random Forest Classifier. All the three scores for our model are high with all of them lying above .98.



Reflection:

This project has been fun. It gave me some insight as to how to handle imbalanced dataset as well as using which score to evaluate the model in what condition. I also refined my knowledge of using the cross validation.

Apart from all this, this provides some major insights as to what factors impact the decision of an employee to leave.

Summarization of some of the interesting steps:

- Preprocessing: Resampling using imblearn library. Deciding when to over sample and under sample.
- Scaling: Selecting the methods to scale the dataset
- Cross Validation: Being a small dataset this was an important stuff to go through. This also helped in gaining some efficiency in predicting the target label too.

Improvements:

Trying out different techniques for handling resampling.

Using neural nets and compare the classification prediction.

References:

http://www.jair.org/papers/paper953.html

http://wikipedia.com

http://scikit-learn.org