Machine Learning Engineer Nanodegree Capstone Project

Predict Employee Retention

Mamidi Karthik Kumar
February 18, 2019

1. Definition

Project Overview

Every year a lot of companies hire several employees. These companies invest time and money for training those employees, not only this but there are training programs within for their existing employees too. The main aim of these programs is to increase the effectiveness of their employees [1]

HR Analytics:

Human resource analytics (HR analytics) is an area in the field of analytics that refers to applying analytic processes to the human resource department of an organization in the hope of improving employee performance and therefore getting a better return on investment. HR analytics does not just deal with gathering data on employee efficiency. Instead, it aims to provide insight into each process by gathering data and then using it to make relevant decisions about how to improve these processes.

Retention in HR:

Employee Retention refers to the techniques employed by the management to help the employees stay with the organization for a longer period. Employee retention strategies go a long way in motivating the employees so that they stick to the organization for the maximum time and contribute effectively. Sincere efforts must be taken to ensure growth and learning for the employees in their current assignments and for them to enjoy their work.

Employee retention has become a major concern for corporates in the current scenario. Individuals once being trained tend to move to other organizations for better prospects. Lucrative salary, comfortable timings, better ambience, growth prospects are some of the factors which prompt an employee to look for a change. Whenever a talented employee expresses his willingness to move on, it is the responsibility of the management and the human resource team to intervene immediately and find out the exact reasons leading to the decision.

As an organization invests time and money in grooming an individual and make him ready for work and to understand the corporate culture and when an individual resigns from his present organization, it is most likely that he would join the competitors. So, the management must understand the difference between a valuable employee and an employee who doesn't contribute much to an organization. Hence efforts must be made to encourage the employees so that they stay happy in the current organization and do not look for change.

For this project, I will be using the HR Employee Retention [2] dataset from Kaggle. This was uploaded for examining employee retention and will be suited for this analysis.

It must be noted that this is a simulated dataset. Unfortunately, real company data suitable for this problem is hard to available, the reason may be due to a combination of sensitive corporate value and personal privacy the dataset would contain. The only other similar dataset I found was also simulated, which is also from Kaggle IBM HR Analytics Employee Attrition & Performance [3].

As a result, this will limit the direct application of our results. Although we should be able to apply the same principles & model to a real-world dataset.

Problem Statement

The main problem I want to address here is: How can a company better act to minimise the negative of employee turnover?

Every Organization is trying to gain maximum results and especially employees looking for better opportunities to fulfil their demands. So, the retention of an individual in an organization is not for long and without the employees the organization cannot function well. So, It is important to understand the various problems an organization faces in order to maintain the employees and use methods to overcome these problems and retain employees.

It is a classification problem and it uses the data of previous employees which have worked for the company and by finding pattern it predicts the retention in the form of yes or no. It uses various parameters of employees such as salary, number of years spent in the company, promotions, number of hours, work accident etc.

To address this problem, we will ultimately be looking to be able to accurately predict employees who leave. Towards this goal, we will use the following outline:

1. Data Pre-processing: Look over the data to understand the features. Perform any appropriate feature removal or scaling. In this step we will perform operations like replacing some of the feature labels like 'sales' to 'department' and 'Work_accident' to 'work_accident' etc. We can visually inspect all our numeric features to make sure we don't have any outliers. We use a boxplot to show the variability of the data, including outliers. For ease of visualisation, we can apply simple rescaling ((xv-min) / (max - min)) so that they are displayed on the same axis scale in our visualisation. For Boolean features as these are provided as integer values of 0 or 1. We'll first that ALL values are 0 or 1, and then convert to explicit Boolean data types

and the we will inspect the categorical data to ensure labelling is consistent, and then encode the categories appropriately.

2. Feature Investigation: Measures central tendency, variation within features. In this step we will find central tendency by using mean, standard deviation. For feature correlations we plot all numerical features against each other, then just continuous numerical features against each other.
3. Principle component analysis: Are there any underlying features? and what are the most and least significant features?
4. Cluster Analysis: Are there any significantly different groupings of employees, with regards to retention rates? I would like to consider using Gaussian Clustering or K-Means Clustering for this dataset.
5. Modelling: Identify and evaluate multiple appropriate algorithms to model the data to make classification predictions. Choice of model based on predictive power and suitability for use. I plan on using supervised learning techniques such as Logistic Regression, Decision Trees, Random Forest Classifier and Support Vector Machines.

We can use an analysis from the publisher of the data [4] as a benchmark model for our predictive model. An accuracy of 97% was achieved here with a Random Forest Classifier model. So, I expect to at least achieve a similar level of accuracy in our final model.

Metrics

This dataset is an example of a class imbalance problem because of the skewed distribution of employees who did and did not leave. More skewed the class means that accuracy breaks down. As a result, evaluating our model based on accuracy is wrong thing to do.

I prefer to use recall rate:

Recall = correctly_predicted_leavers / total_leavers

and I also prefer to use precision:

Precision = correctly_predicted_leavers / predicted_total_leavers

and, F1-Score:

F1-score = 2 (precision * recall) / (precision + recall)

This is motivated by wanting to minimise the effect of employees leaving, so it makes sense to place extra importance on minimising misclassifying leavers that is false negatives.

2. Analysis

Data Exploration

The dataset consists of 15,000 samples with 9 features plus 1 label for retention. They contain a mixture of numeric, Boolean, and label values. From the dataset it is possible to see that satisfaction_level could be correlated with other variables such as work_accident. This dataset contains numerical features, Boolean features and categorical features as well. The total number of data points are 14999. As for splitting the dataset, we split our data into features and labels in separate training, validation and test sets. We will use validation set to help measure our model's performance as we develop it and hold out the test set until final evaluation of the model.

- **satisfaction_level:** It is a numerical measure of employee satisfaction from 0 which is low, to 1, which is high.
- **last_evaluation:** It is a numerical measure of how well the employee was performing at their last evaluation, from 0 which is low, to 1, which is high.
- **number_project:** The number of projects the employee was assigned to.
- **average_monthly_hours:** The average numbers of hours the employee worked each month, rounded to an integer value.
- **time_spend_company:** The time the employee has spent at the company, the number denoting the number of years.
- **work_accident:** It shows whether the employee was involved in a work accident, 0 which is no, and 1, which is yes.
- **performance_last_5years:** It shows whether the employee was promoted in the last 5 years, 0 for no and 1 for yes.
- **sales:** It shows which department the employee worked in.
- **salary:** A text label for low, medium or high.

And finally, the label representing the retention is:

- **left:** It shows whether the employee left or not, 0 for no and 1 for yes.

There are some inconsistencies and errors in the feature names. Categorical data will need to be one hot or label encoded for further analysis.

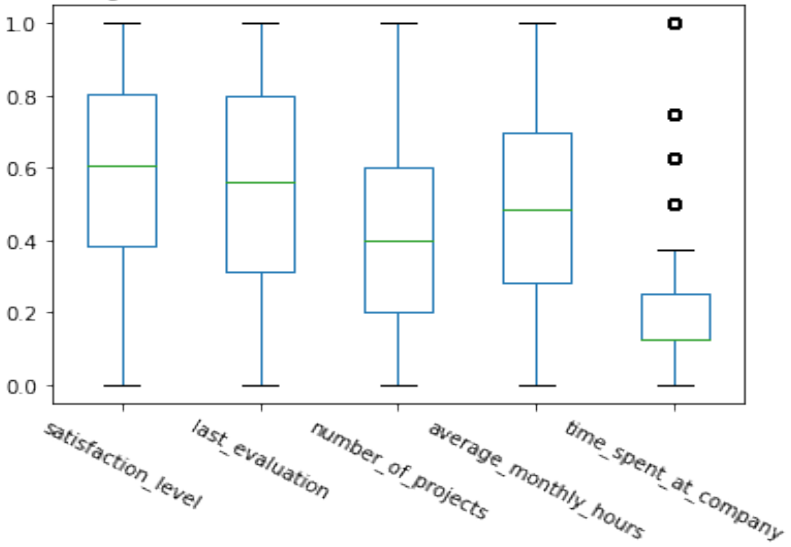
It is important to note that our labels are not evenly balanced. We have ~24% left=True and ~76% left=False. This is compounded by the additional important of positive classifications due to the additional cost of an employee leaving. We should have enough data from both categories to be able to develop a good model, but we may wish to address this imbalance later if we need to improve results.

Exploratory Visualizations

For the numerical features, we can look at the distribution of their values to spot outliers. Outliers could be present due to unusual cases, natural variation, or simply errors in the data.

Figure 1 displays the data distributions in a box plot for these variables. In this chart, all features have been scaled so that data is between 0 and 1 for ease of visualisation and comparison.

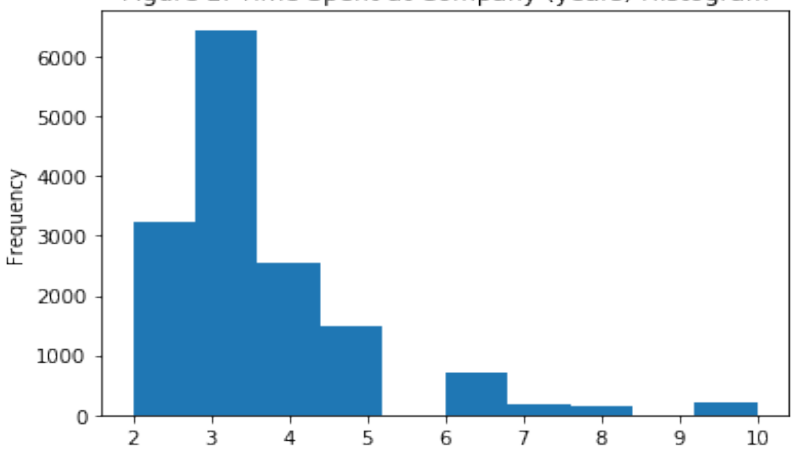
Figure 1. Data Distributions of Numerical Features



We can see from this that for most features, there are no significant outliers.

The exception is for time_spent_at_company. But if we look at a histogram of the data without scaling (figure 2), the data looks reasonable given the nature of this feature: We would expect to see more people staying at the company at the lower end (in this case, 2-4 years) and then for the amounts to tail off as there is a kind of decay of employees. The maximum value here is 10, which is only about twice the value of common data. I consider this reasonable enough to leave in.

Figure 2. Time Spent at Company (years) Histogram

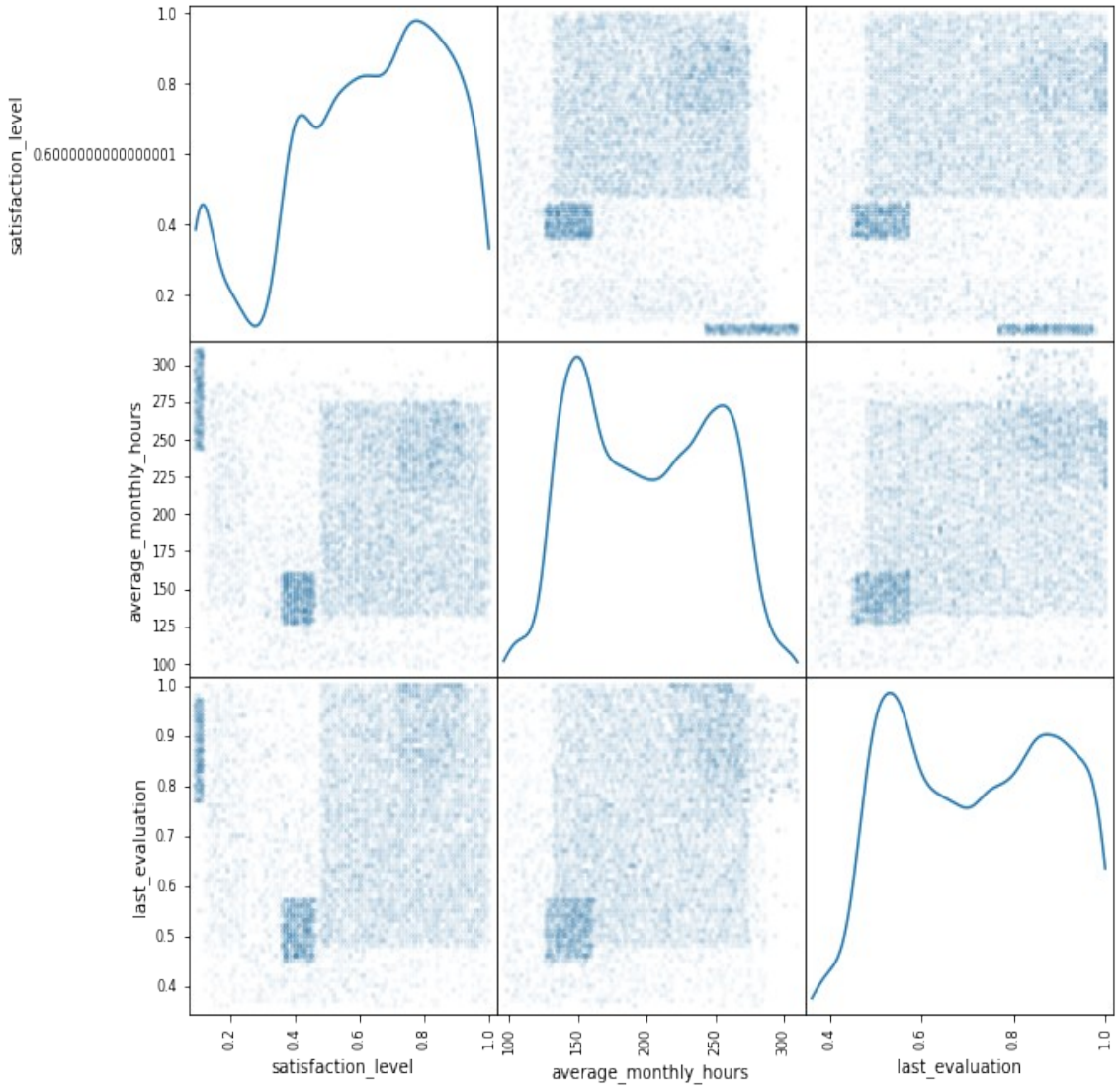


Additional Exploration

As part of the data exploration, I also explored possible correlations between continuous numerical features. This highlight some interesting results, as can be seen below in Figure 3. First, there is some clear clustering that can be seen here. Further, within each visible cluster there appears to be sharp truncated edges on the distribution of the data. These are unusual and I suspect related to how the data was simulated.

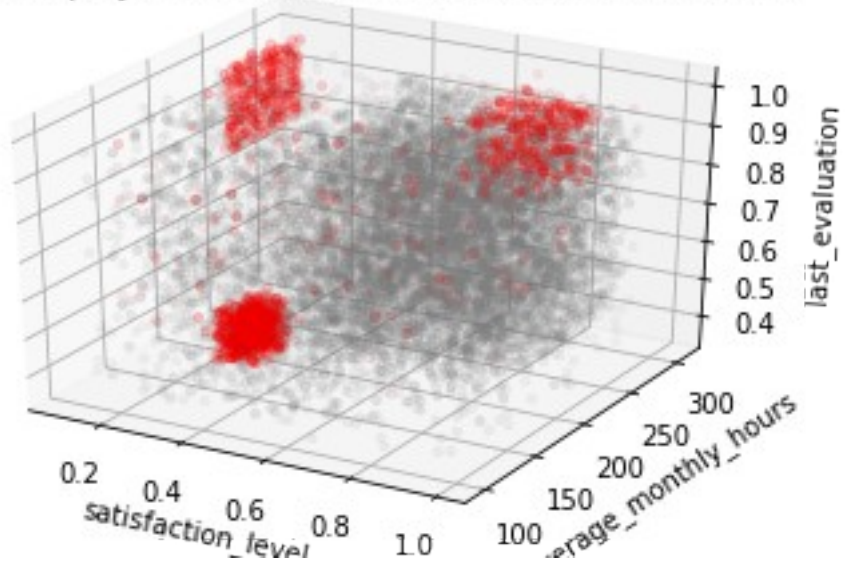
If we are to assume these are artificial artefacts, while this is not an ideal reproduction of real data, I'll proceed from here by assuming that the data distributions were at least generated based on real world insights. We were aware the data was simulated and therefore wouldn't be completely representative of the real world.

Figure 3. Continuous Numerical Features Scatter Matrix



These clusters, split by leavers and displayed in colour, can be further seen in the 3D plot in Figure 4, suggesting clustering may be a useful way of understanding or predicting employees who leave.

Figure 4. Employees who have left (red) and remained (blue)



Given we can see overlapping cluster in our visualisations, I decided it was a good idea to use a soft clustering algorithm to explore clustering within the data. This would allow points to be associated with multiple clusters with different probabilities. I chose a Gaussian Mixture as a clustering algorithm.

Using number of clusters = 5 (determined from visual inspection of the data), we were able to predict distribution centres and, by applying the model to our data, able to see how many employees from each cluster left.

Cluster	Centre	Total Size	Number of Leavers	Percentage of Leavers
0	[0.77, 245.08,0.78]	4656	940	20.2
1	[0.66, 157.5, 0.7]	4811	113	2.3
2	[0.1, 277.27, 0.87]	886	881	99.4
3	[0.58, 204.06, 0.71]	3007	119	4.0
4	[0.41, 143.64, 0.51]	1639	1518	92.6

Taking probability of the best guess per cluster, weighted by cluster sizes, this gives us an estimated accuracy score of about 91%.

For an employee within cluster 2 or 4, we already know they are at a very high risk of leaving. These correspond with high density clusters in the earlier charts. Similarly, an employee in cluster 1 or 3 is likely to remain.

Finally, some Principle Component Analysis (PCA) was performed, but did not raise any interesting insights, and suggested a somewhat even distribution of explained variance across our numerical values.

Algorithms and Techniques

For the final stage of classification modelling, a range of suitable supervised learning model will be tried. The models we'll use are all suitable for classification:

Logistic Regression

Logistic Regression is used to predict one of two outcomes, calculating a logistic probability from a linear function of the input features and optimising the coefficients to maximise the accuracy of the probability. A boosted version was used in the benchmark model, which was chosen for it's good predictive results and simplicity.

Support Vector Machines

Support Vector Machines apply a mapping to the data (via a technique called the Kernel Trick) to transform it into a higher dimensional space. They then find a decision boundary between classes that maximises the decision boundary between classes. They are a proven and successful classification algorithm that should be well suited to this task.

Decision Tree Classifier

Decision Trees are a tree-based graph that repeatedly apply criteria (eg. feature X is True, feature Y > 1.0) to the dataset to split it into smaller datasets. Criteria are chosen by what 'best' splits the data towards good classification (such as maximising Information Gain or similar metrics). They are proven and successful at classification tasks, plus they provide good interpretability of results.

Random Forest Classifier

Random Forests are an ensemble of multiple decision trees, with each tree casting a vote for its prediction and the most voted prediction being the overall returned result. Each decision tree is grown through sampling with replacement from the original data set, using subsets of the input features, and with no pruning. They are usually more powerful than a standard decision tree, but harder to interpret.

We'll use these models with default parameters from the sklearn python library. Based on predictive performance and suitability at addressing our problem statement, we'll decide a single classifier and proceed to hyperparameter optimisation for fine tuning. We'll do this using Grid Search (a systematic and exhaustive search over a defined parameter space) with k-fold cross validation. Parameters to search over will be chosen as appropriate for the model and problem statement.

Benchmark

we can use an analysis from the dataset authored by [4] as a benchmark for our predictive model.

This analysis includes investigation into the variables and their correlations and looks for variables contributing to whether employees leave, using statistics and visualisations to demonstrate findings along the way. As this analysis already shows a significant correlation

between satisfaction_level and employee retention. It also highlights groups of employees who are favourable to retain, based on their evaluation score.

Then the analysis proceeds to the main modelling stage, using cross validation on the dataset with three learners: decision tree, logistic regression, Random Forest and Support Vector Machine.

Of them the final chosen model was Random Forest Classifier and it could predict retention with about 97% accuracy on the test set.

3. Methodology

Data Pre-processing

Our data columns had some mistakes in, to correct these, I changed:

- **Sales** to **department**
- **average_montly_hours** to **average_monthly_hours**

Additionally, I modified some feature names for style consistency and clarity:

- **Work_accident** to **work_accident**
- **number_project** to **number_of_projects**
- **time_spend_company** to **time_spent_at_company**

Boolean features (work_accident, left, promotion_last_5years) were converted from numerical (0, 1) to Boolean data types (False, True).

Categorical features (department, salary) were one-hot encoded. For salary, I considered creating an ordered labelling (low,medium,high to 0,1,2) to reflect their ordered nature. However, I eventually decided against this as there was no way of knowing the difference between each category.

Our dataset of 14,999 was split into training, validation and test sets of sizes 10499, 2250, and 2250 respectively.

Implementation

The four chosen classifiers were implemented using the standard libraries and classes available in the python sklearn library. Default arguments were used, except for using a seed value for random_state to guarantee reproducibility on repeat runs of the code. For metrics, we also used sklearn for its precision_score, recall_score and f1_score functions.
The classifiers were then applied to the training data set and scored using the precision, recall and f1_score measures on both the training and validation data sets. The results were as follows:

Classifier	Training Data			Validation Data		
	Precision	**Recall**	**F1_score**	**Precision**	**Recall**	**F1_score**
Logistic Regression	0.610	0.360	0.452	0.660	0.371	0.475
SVC	0.894	0.919	0.906	0.894	0.892	0.893
Decision Tree Classifier	1.000	1.000	1.000	0.947	0.960	0.953
Random Tree Classifier	1.000	0.992	0.996	0.989	0.946	0.967

Our Tree based classifiers that is Decision Tree and Random Forest Classifier are clearly the best performing model here.

They are overfitting to the training data a little (showing higher scores on training compared to validation) but are still performing highly for the validation set.

The two tree models perform similarly on the validation set, but the Decision Tree has slightly better recall rate, and the Random Forest has slightly better overall accuracy.

Given the greater importance of accurately predicting employees who leave (ie. recall) over employees who remain, plus the interpretability of a Decision Tree, the Decision Tree seemed to be the most successful model and was chosen for further optimisation.

Refinement

To further refine the model, Grid Search was performed to find an optimal model over a range of parameter values. These parameters were chosen based on our understanding of the data and the decision tree model so far. This included relevant feature selection & transformation.

To measure the performance of the model, we would combine both recall and f1_score with equal weight.

To implement Grid Search, we used GridSearchCV from the sklearn library with the cross validation parameter cv set to 3-folds. A custom scorer function was created for the GridSearchCV class to use. As Grid Search could take some time to run, the verbosity argument was used to provide feedback on its progress and the n_jobs argument was used to process models in parallel to speed up the training time.

One problem encountered was in searching over feature selection and transformation options. Initially, these feature options were looped over - each running GridSearch once on the classifier only. However, this didn’t provide a consistent and rigorous means of evaluation & comparison. I researched around this and discovered the concept of Pipelines in sklearn [5]. Pipelines allow chaining together multiple estimators of data transformers. Custom FunctionTransformers were then created to apply feature selection and clustering, so they could be chained together into a pipeline with the main classifier. This allowed the entire process to be evaluated within a single GridSearchCV process.

The parameters chosen to explore in Grid Search, and reasons behind their selection, are listed below. A common consideration amongst most parameters was to limit overfitting. This is something decision trees can sometimes suffer from, and we already saw signs of this in our initial model.

Clustering

Our previous clustering analysis showed there was useful information in our clustering model for predicting leavers that already could provide decent classification accuracy. Including this as an additional feature in our data may help our model. We had already demonstrated success with number of clusters = 5, so we used this to inform our parameter value choices.

Training was performed with no clustering, and with 5 and 10 clusters.

Feature Selection

Our initial decision tree model provided importance scores for each feature. The majority of features had very low importance scores (less than 0.005). This contributes extra dimensionality and possibly noise to our data set that can make it harder to train our model and can contribute to overfitting.

Training was performed on: 1) all features; 2) features with importance > 0.005 (satisfaction_level, last_evaluation, time_spent_at_company, number_of_projects, average_monthly_hours).

Decision Tree: max_depth

This determines the depth of our decision tree. A larger depth can yield better results, but at the risk of overfitting and cost of interpretability. A smaller depth risks yielding poorer results but helps limits overfitting and makes the final results easier to understand. Training was performed with max depth values of 3, 5 and 7. These were chosen to be similar to know relevant features to provide enough space for the model to fit well, whilst remaining low to aid interpretability and avoid overfitting.

Decision Tree: min_smaples_split

This is the minimum number of samples required to split a node. A higher value means that decisions will only be made when more data is available to base a decision on and is a useful way to avoid overfitting.

Training was performed with min_samples_split values of 2, 5 and 10.

Decision Tree: class_weight

Our data set is unbalanced (roughly 3x remain to 1x left). Additionally, it's more important to detect leavers - the cost of misclassification of someone who leaves is greater than for someone who stays. By applying weighting to the classes, we can try to help the model counteract this numerical & importance imbalance.

Training was performed with no class weights, and with class weights of: {True:3, False:1} and {True:10, False:1}.

4. Results

Model Evaluation and Validation

The final model returned from this used the following parameters:

1. cluster n_components = None
2. filter_features = True
3. max_depth = 7
4. min_samples_split = 2
5. class_weight = {False: 1, True: 3}

Our final decision tree performance compared to the initial tree is shown below. It's notable that although similar, the final model performs slightly worse. This is likely the result of our choice of parameters which favoured simplicity over pure accuracy to avoid overfitting and aid interpretability. For example, our initial decision tree has a max depth 18 compared to our final model with a max depth of just 7.

	Training Data			**Validation Data**		
	Precision	**Recall**	**f1_score**	**Precision**	**Recall**	**f1_score**
Initial Decision Tree	1.000	1.000	1.000	0.947	0.960	0.953
Final Decision Tree	0.964	0.942	0.952	0.966	0.927	0.952

For final evaluation of our model, we apply it to our test dataset, given the following results:

- Precision = 0.953
- Recall = 0.940
- F1_score = 0.947

Feature selection to the most important (5) features, max depth of 7 and min_samples_split of 2 all should help avoid overfitting. So, if applied to unseen data, it should continue to perform well. It's worth noting the general consistency of the model's accuracy and recall_scores when applied across training, validation and test sets - a good sign that we have largely avoid overfitting.

Class weight is used to favour classification of employees leaving, which helps account for imbalances in the data. However, there is still some doubt about how well this model can be trusted to generalise beyond this data set. As discussed earlier, this data is simulated. Real world data on this topic is hard to come by, so it's hard to know exactly how it will differ from real world data. We saw possible artificial artefacts in earlier figures. However, I do believe that even if the model were to have trouble generalising to real world data, at the

very least the process of development and tuning of the model were robust enough to be able to similarly useful model.

Justification

Our final model has recall_rate of 94.7% using Decision Tree Classifier.

Our Tree based classifiers (Decision Tree & Random Forest) are clearly the best performing models here. They are overfitting to the training data a little (showing higher scores on training compared to validation) but are still performing highly for the validation set. The two tree models perform similarly on the validation set, but the Decision Tree has slightly better recall rate, and the Random Forest has slightly better overall. Given the greater importance of accurately predicting employees who leave (ie. recall) over employees who remain, plus the interpretability of a Decision Tree, the Decision Tree seems to be our most successful model so far. We'll select the Decision Tree for further tuning and analysis compared to the Benchmark model which uses Random Forest Classifier.

Overall, I think the final model is a good model to help solve the problem statement. By being able to more accurately predict potential leavers, along with a model that allows interpretation, a company could act early to avoid employees leaving and therefore reduce the negative effects of employee turnover.

5. Conclusion

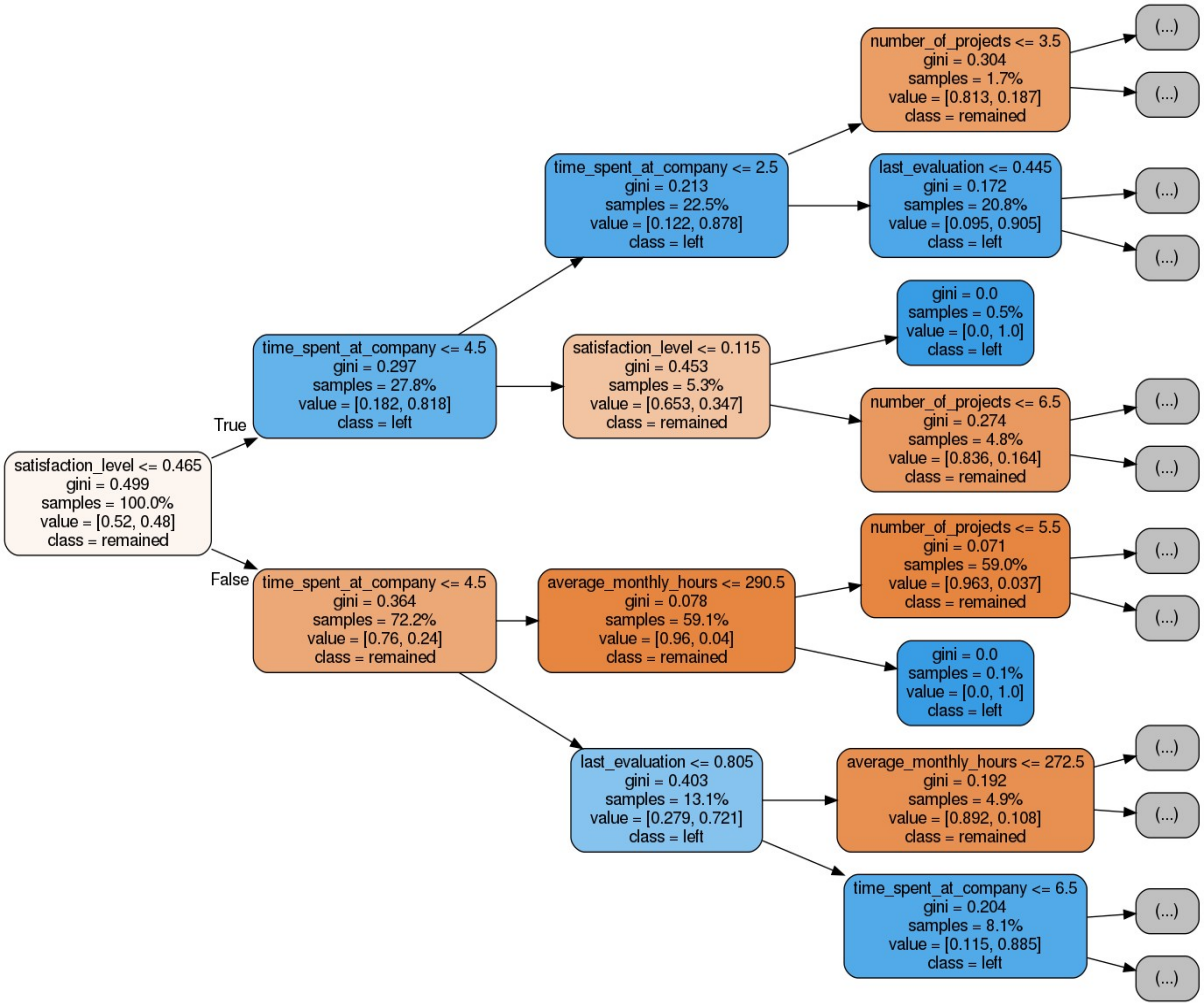
Free-Form visualization

An important factor in development of our final model has been to keep the model simple and understandable, to better aid decision making. Being able to accurately predict that an employee might leave is important. But being able to act in an effective manner is also important. A decision tree model is relevant here, as it’s possible for someone to see what criteria determined the prediction that an employee was going to leave, and there can highlight possible actions. Although we’ve not covered how to extract and understand predictions in this project, a visualisation of the final model helps start to understand how it can be applied.

In Figure 5, each split of the decision tree (for the first 4 layers) is represented. Important information in this diagram is presented, including:

- The criteria for splitting are the first line in each node
- The total number of training samples (as a percentage, across that depth layer) is listed as samples.
- The proportion of samples that are remain or leave are represented both as value and the colour of each node (blue=left, orange=remain), with stronger colours indicating greater proportions of the corresponding class.

Figure 5. Final Decision Tree Model (first 4 layers)



We can take the bottom two nodes of the 4th layer as an example. These represent a subset of employees split on having a high or low score in their last evaluation (<=0.805). Of the employees here who scored lower than 0.805, 88% went on to leave. Since many of these employees are leaving (and assuming most are leaving of their own volition), it's reasonable to think they want to do well in their job. A possible interpretation is that there is lack of clarity between company and employee over what is expected of the employee to perform well.

Providing clearer job descriptions and responsibilities could be a cheap and simple solution to this. Likewise, more frequent meetings with managers to more continually (perhaps less formally) discuss and reflect on performance and development could help.

Reflection

To summarise this project: I performed some initial data exploration & processing, proceeded to more detailed exploration using more complex unsupervised learning methods (eg. clustering), applied supervised learning techniques, then optimised a chosen model using grid search and cross validation.

On a technical level, I found this a useful project for exploring techniques in python data analysis libraries. In particular, the use of Pipelines in sklearn for more complex modular models. It was also good opportunity to explore visualisation methods such as those in the seaborn library and decision tree visualisation.

I also found it interesting and challenging to discover possible simulation artefacts in the dataset. While I don't feel this discount the analysis, I do feel it introduces a little doubt on the final model's ability to generalise.

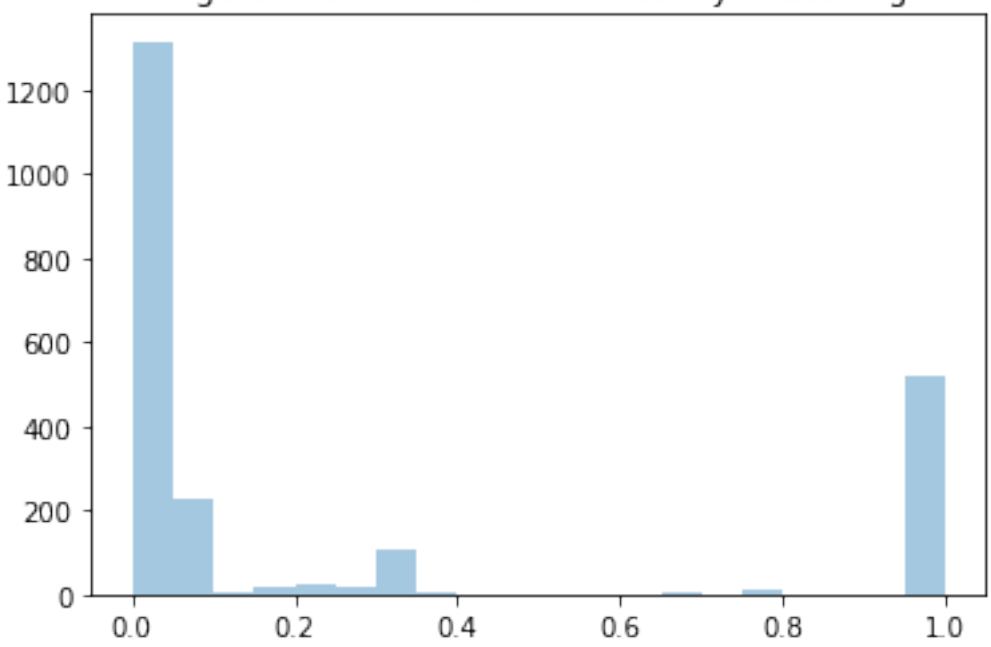
Improvement

I think the biggest improvement that could be made would be to run this process over real world data. However, as mentioned earlier on, this data is sensitive in both a personal and business sense and would be hard to get hold of outside a company.

An interesting possible improvement to the gaussian mixture performed would be to perform a similar clustering technique with different distribution. This is something I didn't have enough knowledge of to implement. In our dataset, our continuous numerical features were largely uniform and truncated at certain ranges - not normally distributed. However, how much of this distribution profile is due to simulation vs real world patterns is debatable.

Our decision tree, as well as making predictions, is also able to provide probabilities of those predictions, as shown below in figure 6. Most employees were predicted with high degree of confidence. However, there is a small grouping with ~30% probability of leaving. That there is a small peak in the distribution here suggest to me that it could be highlight a shortcoming of our model. It would be interesting to explore this subset of the data further to see if there is a reason for it and if we could improve our model further.

Figure 6. Distribution of Probability of Leaving



While there is still scoped to improve this model in f1_score (94.7%) and in particular recall (94.09%), they are already scoring highly so it may be a case of diminishing returns. Perhaps a better way to improve the model's usefulness would be to focus on its application. For example, in developing better means to interpret and act on predictions.

References

[1] Dubey, A. K., Maheshwari, I., & Mishra, A. Predict Employee Retention Using Data Science.

[2] https://www.kaggle.com/gummulasrikanth/hr-employee-retention

[3] https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset

[4] https://www.kaggle.com/randylaosat/predicting-employee-kernelover

[5] http://scikit-learn.org/stable/modules/pipeline.html#pipeline