City, University of London
Computer Science BSc

Academic Year 2017/2018
Introduction to Data Mining (IN3011)
Coursework Report

Analysis of a Human Resources Analytics Dataset

by: Benjamin Fockter

Lecturer: Peter Smith

Document v. 1.0
13th December 2017

Contents

1. Description of the Problem

1.1 What is the Data Set?

I am examining a data set regarding human resource analytics at a fictional company; the data is simulated. It contains 14,999 entries and 10 variables. Of these, 3 are continuous, 3 are categorical, and 3 are discrete with the last variable being the classifier, namely '`left`' – indicating whether the employee has left the company or otherwise. Furthermore, 8 of these variables are numerical with the last 2 being `String`. Of the these: '`Sales`' contains 10 unique categories and '`Salary`' contains 3. Naturally these variables will go through "data wrangling" throughout the project, including exhaustive conversion to numerical values in preparation for machine learning models.

1.2 What do you propose to do with it?

I will use the data to answer the general question: Why are the firm's employees leaving? Upon gaining more information and gaining a better understanding of the data I will narrow down this broad question to specific sub-questions or hypotheses. The final purpose of this data mining analysis will be to not only present my statistical results but also try and identify which values contribute to workers leaving the most. Lastly, I'll use machine learning to try and predict which employee will leave next.

1.3 Background reading

Because the data is simulated there wasn't any background information provided about the company or any peripheral facts (e.g. recruiting strategies) that may have helped give context to the problem. Despite this I will continually tap into the problem domain and also further use my own intuition and past work experience to guide me through the project. The only reading I used to supplement me were online articles about typical human resource (HR) issues and standard cases where a company suffered losses due to bad management.

1.4 Reference

The data set comes from Kaggle. It is a data science website that provides data-sets on a broad range of topics. The dataset is released under the CC BA-4.0 (creative commons share-alike) license. Full reference to the data including reference to some articles used for backgrounding reading can be found under the main heading, '*References*'.

2. Analysis of the Data

2.1 What do you propose to do with the data?

Primarily, to familiarise myself with the dataset I will perform a full exploratory data analysis (EDA). This will allow me to understand the noise, structure, and relationships between variables within the data. I will follow this up with a series of normality tests, testing either the entire variable or just samples, to check for normal distributions. This will aid me further on when I need to account for techniques which assume normality. Additionally, I will rely on other statistical tests such as T-tests, and Chi-square for hypothesis testing, then Kruskal-Wallis to see if samples originate from the same distribution (Wallis, 1952). Lastly, I will prepare the data for machine learning (ML) models. This concluding step is crucial in being able to experiment with multiple algorithms which will ultimately help me predict who will leave the company next.

2.2 What techniques do you propose to use and why?

I will use an amalgamation of techniques for both EDA and machine learning. In addition to the techniques mentioned above for exploration, I'll also use standard procedure and clean the data. This will include renaming variables, dropping outliers, and converting `Strings` into numerical form (for categorical variables) – while this may be done early as a temporary measure to create certain plots or carry out certain calculations, it will also constitute during preprocessing of data for ML. In addition, I'll quickly check for any missing data-points, then briefly examine mean, median, mode, variance, skew and kurtosis (excess kurtosis) for each variable if there is anything outstanding. Similarly, I will cross plot each variable with the classifier '`left`' (by color), to scan for any notable trends or patterns. Moreover, I will construct a correlation matrix to pick out correlation coefficients of the different pairs of variables. I aim to do this in order to save time doing an exhaustive search between every combination of variables, yet try to minimize the probability of missing an important statistical finding that may be invaluable later on. With those findings I do find important, I plan to display them using bar, KDE (kernel density estimation), scatter, box, and violin plots. I will adhere to: Jarque-Bera test for normality over larger sets (e.g. n > 2000), and Kolmogorov-Smirnov to measure the distance between the function of my samples and the cumulative distribution function (CDF) of a second sample or the distribution in question. If I want to test for normality in smaller samples, specifically with unspecified mean and variance, I will use Shapiro-Wilk. To compare 2 probability distributions I'll also look at Q-Q plots. The above, coupled with the statistical tests already mentioned for significance will conclude my EDA. To scrutinize data consistency I'll use K-means and DBSCAN for cluster analysis. During pre-processing, I will also employ linear and non-linear dimensionality reduction, particularly principle component analysis (PCA) and Isomap. Scaling, feature selection and feature engineering will also be included. For modeling

and prediction I will experiment with Logistic Regression, Decision Trees, and some ensemble methods like Random Forest. To visualise and check the accuracy of my algorithms I will use a Confusion Matrix.

2.3 How do you propose carrying out your analysis?

I will carry out all computations in Jupyter using Python 2 code. The dynamic-typed nature of python plus its versatile libraries for calculations and visualisations make it ideal for data science projects. Mainly I will use packages: SciPy, NumPy, Matplotlib, Pandas and Seaborn, with the addition of some other toolkits for 3D representations. I'll also use Scikit-learn for preprocessing and regression, for example.

3. Interpretation of the Results

3.1 Exploratory Data Analysis (EDA)

```
data.rename(columns={'number_project':'projects'}, inplace = True)
data.rename(columns={'average_montly_hours':'avg_monthly_hours'}, inplace = True)
data.rename(columns={'time_spend_company':'time_spent_at_company'}, inplace = True)
data.rename(columns={'Work_accident':'work_accident'}, inplace = True)
data.rename(columns={'sales':'department'}, inplace = True)
```

Figure 1

I start by importing my data, examining it and cleaning it accordingly. For simplicity's sake and ease-of-use I rename certain variables as shown in *Figure-1*.

	satisfaction_level	last_evaluation	projects	avg_monthly_hours	time_spent_at_company	work_accident	promotion_last_5years	department	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
3	0.72	0.87	5	223	5	0	0	sales	low	1
4	0.37	0.52	2	159	3	0	0	sales	low	1
5	0.41	0.50	2	153	3	0	0	sales	low	1
6	0.10	0.77	6	247	4	0	0	sales	low	1
7	0.92	0.85	5	259	5	0	0	sales	low	1
8	0.89	1.00	5	224	5	0	0	sales	low	1
9	0.42	0.53	2	142	3	0	0	sales	low	1
10	0.45	0.54	2	135	3	0	0	sales	low	1
11	0.11	0.81	6	305	4	0	0	sales	low	1
12	0.84	0.92	4	234	5	0	0	sales	low	1
13	0.41	0.55	2	148	3	0	0	sales	low	1
14	0.36	0.56	2	137	3	0	0	sales	low	1

Figure 2

Figure-2 shows the head of the dataset. Note that: '`satisfaction_level`' is the rated level of happiness by the employee at the company, '`last_evaluation`' is the score that the employee received at the company for their work, '`projects`' is the number of projects an employee completes while at work, '`avg_monthly_hours`' is self-explanatory,

'`time_spent_at_company`' is the number of years a worker has been employed by the firm, '`work_accident`' indicates whether the employee had a workplace accident, '`promotion_last_5years`' indicates whether the employee has been promoted in the last 5 years, '`department`' is self-explanatory, '`salary`' as well, and '`left`' tells us whether the employee chose to leave the company or not. Also note that '`satisfaction_level`', '`last_evaluation`', and '`avg_monthly_hours`' are continuous; '`work_accident`', '`department`', '`salary`', are categorical; '`projects`', '`time_spent_at_company`', and '`promotion_last_5years`' are discrete.

	satisfaction_level	last_evaluation	projects	avg_monthly_hours	time_spent_at_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

Figure 3

Upon looking at *Figure-3* I can see that none of the columns are missing any entrees (there are 14,999 rows in the data). A brief look at monthly hours I see that it is quite high – a regular 9 to 5 job would yield 160 hours over a month; '`std`' is also high. Most people have been at the company for 3 years which can be deduced by looking at the 50th percentile value. Dividing the people who stayed (`0`s) by 14,999 and people who left (`1`s) by 14,999 equals 76 and 23.8, respectively. This indicates that about 24% of employees left the company.

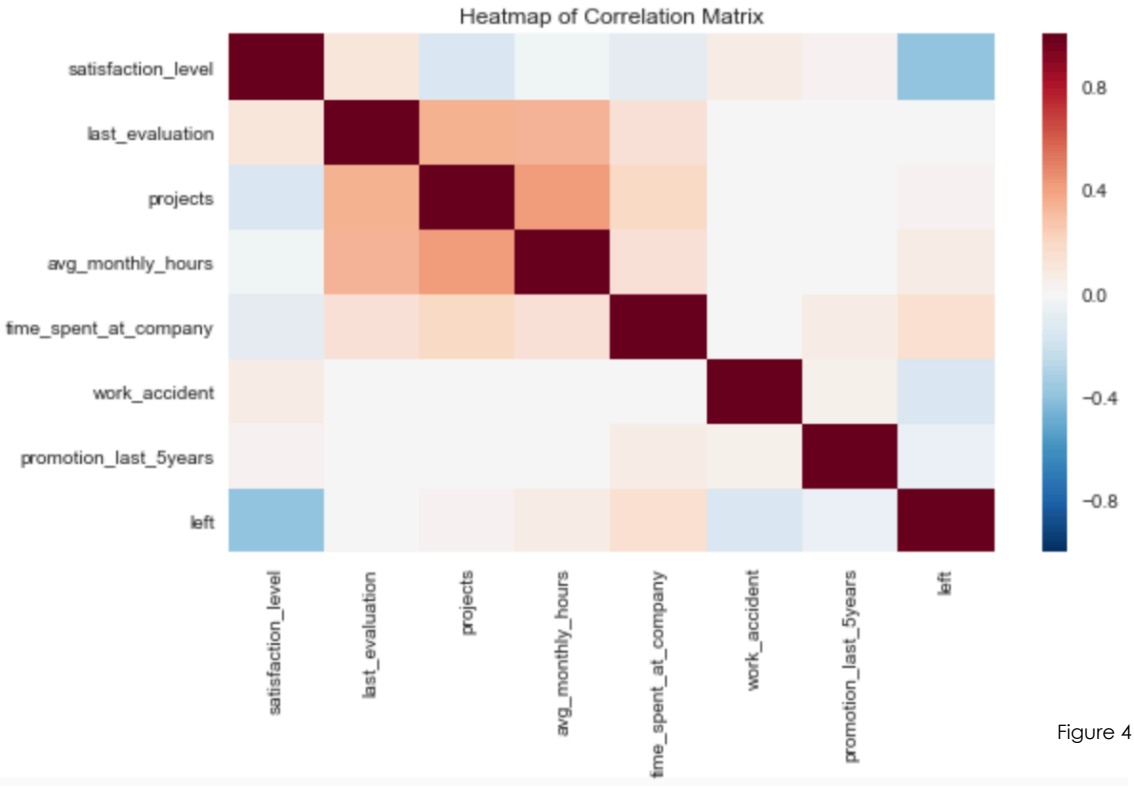


Figure 4

From *Figure-4* I can see that there is some correlation between: '`last_evaluation`' and '`projects`'; '`last_evaluation`' and '`avg_monthly_hours`'; '`projects`' and '`avg_monthly_hours`'. It also seems there is a slight negative correlation between the classifier '`left`' and '`satisfaction_level`'. Note that this matrix shows linear correlations between features and thus is not able to reveal any non-linear ones. A further cross-plot (not shown) reveals the same sort of phenomena.

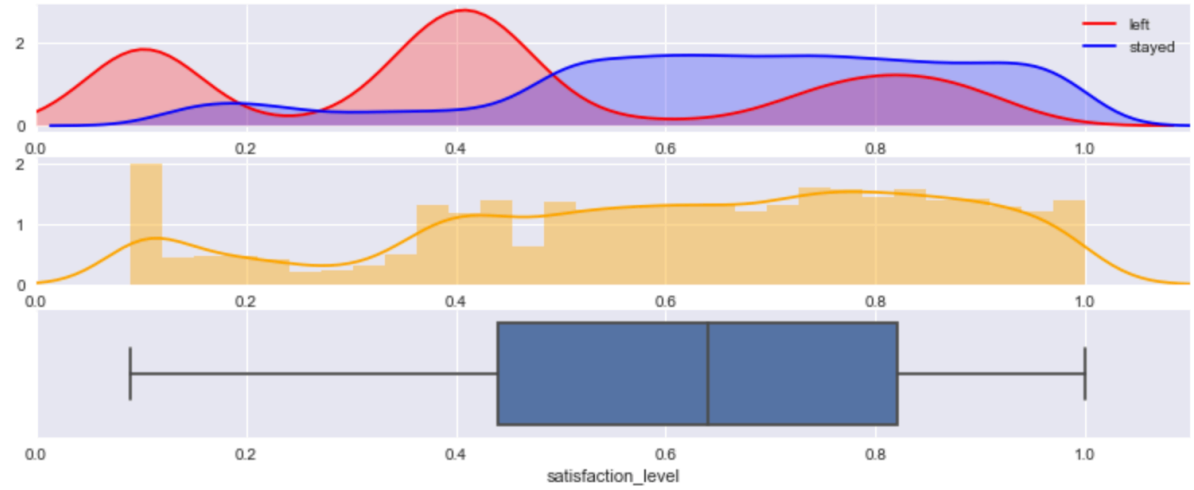


Figure 5

Thoroughly looking at satisfaction I can see that the top 40% is generally pleased. Histogram shows that a very small number of employees are very unhappy. It also seems like a bimodal distribution with one mode below 0.2 and another around 0.4. Surely upon checking I see the following:

```
satis = data.satisfaction_level
satis.median()
```

```
0.64
```

```
satis.mode()
```

```
0    0.1
dtype: float64
```

```
st.describe(satis)
```

```
DescribeResult(nobs=14999, minmax=(0.08999999999999997, 1.0), mean=0.6128335222348155 8, variance=0.06181720064708762 9, skewness=-0.47631270075042725, kurtosis=-0.6710350084807528)
```

Mode is 0.1 representing the extremely displeased workers. Mean and median are in line with each other. There is a slight negative skew, i.e. data is being pulled to the left, and a slight negative kurtosis. Box plot is providing visuals of min and max, 1st and 3rd quartile, and the median inside. There doesn't seem to be any outliers. My first intuition tells me that there should be some correspondence between low satisfaction and work accidents, so I check and get the following:

```
low_satis = data[satis < 0.3]
high_satis = data[satis >= 0.5]
```

```
print ((np.sum(low_satis.work_accident) / float(len(low_satis)))*100)
print (np.sum(high_satis.work_accident) / float(len(high_satis))*100)
```

```
11.7245005258
16.1866359447
```

I can see that 11% of the employees who have low satisfaction had an accident. But 16% with high satisfaction also had an accident, which goes against common sense (one would expect that high satisfaction would be accompanied by fewer accidents; drawing from the problem domain I can guess that maybe those workers who have had an accident may have been compensated in some way, thus the high satisfaction?). Trying to see if there is anything between satisfaction and last evaluation I get:

```
print ((np.mean(low_satis.last_evaluation))*100)
print ((np.mean(high_satis.last_evaluation))*100)
```

```
78.1650893796
74.0814132104
```

This tells us that those people who have low satisfaction are also getting high evaluations (possibly for their hard work?).

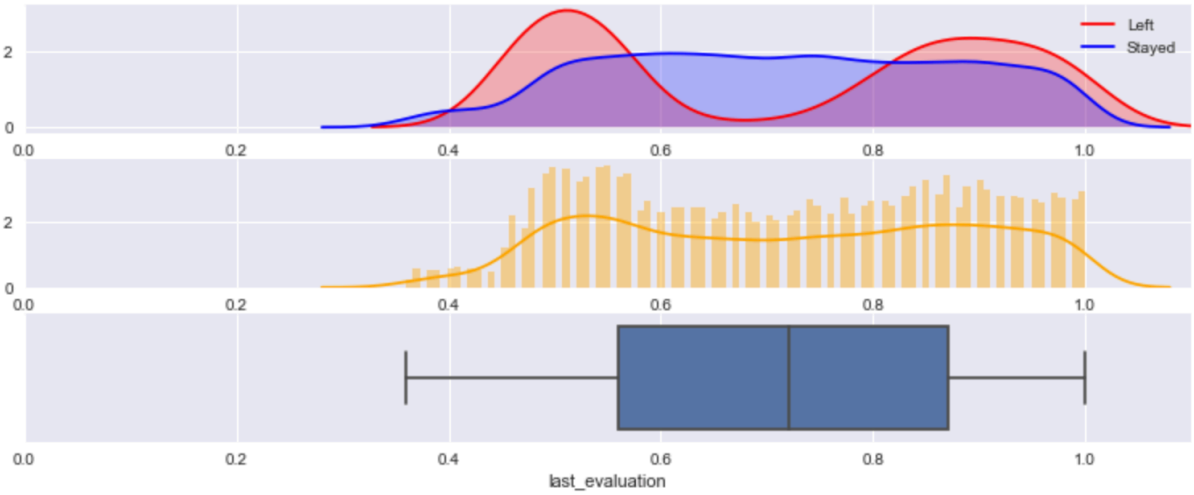


Figure 6

Thoroughly looking at '`last_evaluation`' I can see that it is also bimodal. Adjusting for '`bins`' in the histogram to equal 100 I can see an inverse bump, as well as a high left shoulder visually seeing the mode at around 50-55%, but also observing that the second highest rating is around 87-89%. Basically, there is a cluster of people getting bad scores and another getting good scores. Bringing up other attributes I see:

```
l_eval = data.last_evaluation
```

```
print (l_eval.median())
print (l_eval.mode())
```

```
0.72
0    0.55
dtype: float64
```

```
st.describe(l_eval)
```

```
DescribeResult(nobs=14999, minmax=(0.35999999999999999, 1.0), mean=0.7161017401160078, variance=0.029298864431563071,
skewness=-0.0266190874373217, kurtosis=-1.2390272795182624)
```

Mean and median are quite high and also similar at 71.6% and 72%, respectively. It can be guessed that a large number of employees are working well. Mode is 55% which goes against the assumption that most of the workers got a good evaluation, now I can assume that a minority of very highly evaluated workers are pulling up the average. The distribution has a slight negative skew; kurtosis is around -1.2, which can be seen on the histogram.

Next variable to inspect is '`projects`'. Based upon the attributes of the variable (shown below), I can tell that most employees work on 4 projects; skew is slightly positive, i.e. being pulled to the right, kurtosis is slightly negative and variance is 1.5.

```
num_p = data.projects
```

```
print (num_p.median())
print (num_p.mode())
```

```
4.0
0    4
dtype: int64
```

```
st.describe(num_p)
```

```
DescribeResult(nobs=14999, minmax=(2, 7), mean=3.8030535369024601, variance=1.5192839143892429, skewness=0.3376718386
088254, kurtosis=-0.49571279698843007)
```

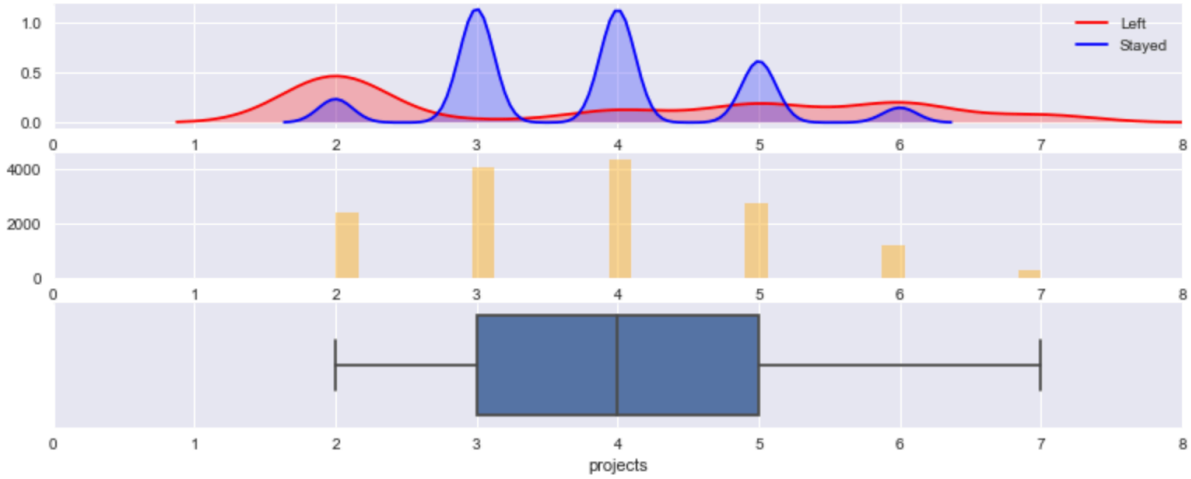


Figure 7

From *Figure-7* I see most employees who stayed worked on either 3,4 or 5 projects; the KDE plot also makes it seem like the distribution might be bimodal. Most employees who left worked on around 2 projects. The histogram shows the mean quite clearly and also tells us there is a small but significant group working on a lot more projects (i.e. 7). The box plot shows us the max, min and the quartiles, and also tells me there are no outliers; clearly some employees work a lot harder than others. I'm going to quickly check '`projects`' with average monthly hours as I saw some correlation in the matrix earlier:

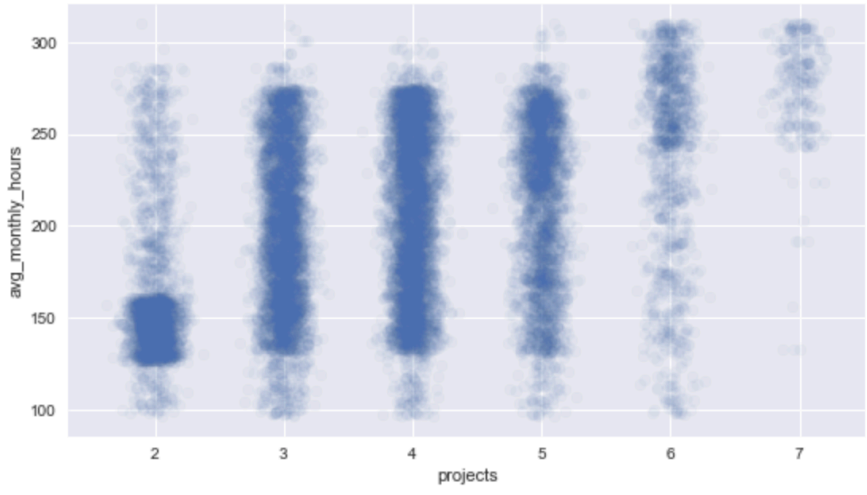


Figure 8

Figure-8 confirms my observations so far and further gives a visual emphasis that there seems to be 2, maybe 3 main groups of employees.

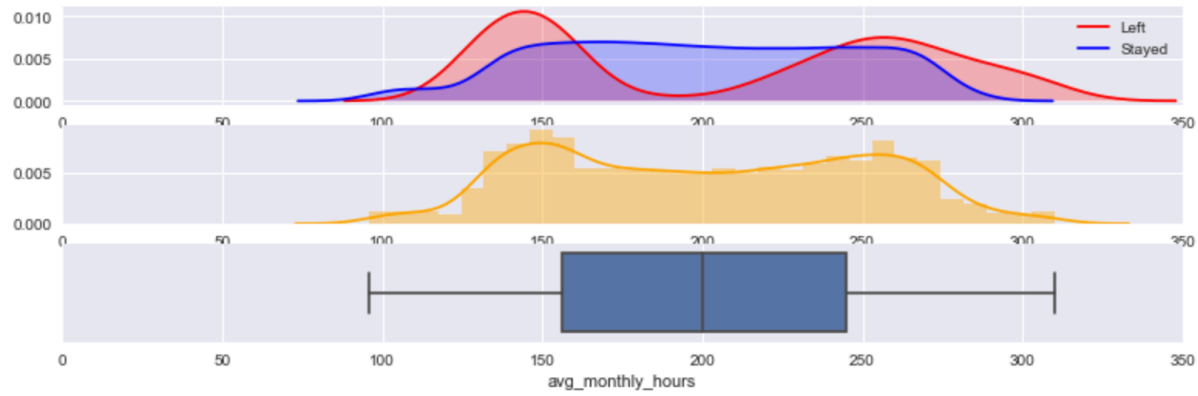


Figure 9

Naturally, I examine average monthly hours next, *Figure*-9. Upon first look at the histogram it might be the case that this distribution harbours two normal distributions, this is because of the two bumps close to each other; again, samples will be examined with Jarque-Bera or even Shapiro-Wilk normality tests depending on the number of data points I get once I check count. Boxplot shows what I've noticed at the start – that the mean is too high. I expect the mean to be around 200, and also see that there might even be two modes (the latter one around 250 hours). This could be a defining feature of why some employees are leaving – simply being over-tasked. The box-plot's 1st and 3rd quartiles show 25% of employees have worked 160 hours and 75% have worked 240 hours, respectively.

```
avgmh = data.avg_monthly_hours
```

```
print (avgmh.median())
print (avgmh.mode())
print (avgmh.value_counts().head())
```

```
200.0
0    135
1    156
dtype: int64
156    153
135    153
149    148
151    147
160    136
Name: avg_monthly_hours, dtype: int64
```

```
st.describe(avgmh)
```

```
DescribeResult(nobs=14999, minmax=(96, 310), mean=201.05033668911261, variance=2494.3131748099559, skewness=0.0528367
0471826943, kurtosis=-1.1350032510931285)
```

Closer inspection shows that there are in fact 2 modes, at 135 and 156, but not where I initially predicted. Regardless, there are 153 employees working both those hour amounts. Note also that the program is treating the data as discrete here, whereas I need to look at it from a continuous point of view. Skew is 0.05, kurtosis is -1.3 (hence the inverse bump). Variance is very high at 2494.31. Box plot indicates some unsettlingly high values, max being more than 300 hours! To put it in perspective, a 10-hour day with only 1 day off comes to only 240. A sum of 310 requires a 13-hour work day with only 1 day weekends over a month; this seems unhealthy and an obvious reason why an employee would leave. To capture this from a different angle I am going to create a new feature named '`work efficiency`' to show how many hours an employee spent on a single project:

work efficiency	left
(192.334, 749.333]	0.191503
(749.333, 1304.667]	0.348383
(1304.667, 1860.0]	0.101083

```
data['avg_hour_project'] = (data['avg_monthly_hours'] * 12) /data['projects']
data['work efficieny'] = pd.cut(data['avg_hour_project'], 3)
data[['work efficiency', 'left']].groupby(['work efficiency']).mean()
```

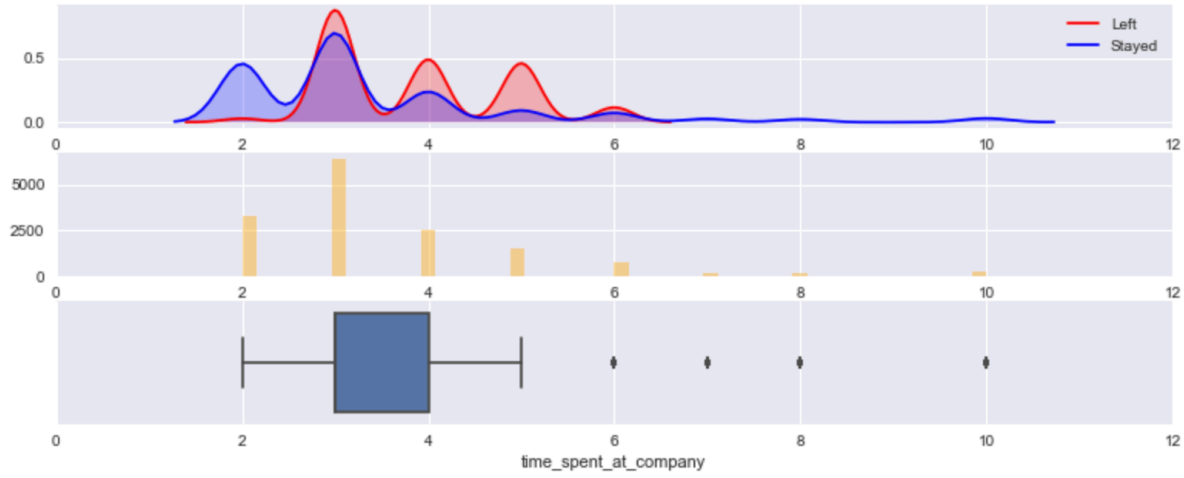


Figure 10

Figure-10 depicts information regarding '`time_spent_at_company`'. From the KDE plot I can see that at around 3,4 and 5 years employees leave often. Drawing from the problem domain, one reason could be lack of promotion; it could also be that 5 years is the maximum time that some of the hard-workers I've seen earlier endure that amount of workload. The box plot tells us there are outliers. These will be dropped during pre-processing for ML. Also interesting to see that none of the workers have been employed for less than 2 years and most have been around for 3 years. Taking a closer look at the distribution's attributes:

```
tpc = data.time_spent_at_company
```

```
print (tpc.median())
print (tpc.mode())
```

```
3.0
0    3
dtype: int64
```

```
st.describe(tpc)
```

```
DescribeResult(nobs=14999, minmax=(2, 10), mean=3.4982332155477032, variance=2.1319978117223641, skewness=1.8531336980238529, kurtosis=4.771219697202597)
```

Mean is 3.49, median 3, and mode 3. Kurtosis is high at 4.77, as well as variance at 2.13. Skew is 1.85, i.e. it's being pulled to the right – there must be some employees who've been around for quite a long time. This raises some questions: What is the reason that some workers leave early at around 2 years, yet there are employees serving 10 years. What is the main difference between these two clusters? Is it possibly the way they are treated with promotions? To try and answer some questions I quickly check the most obvious plausible reason – salary.

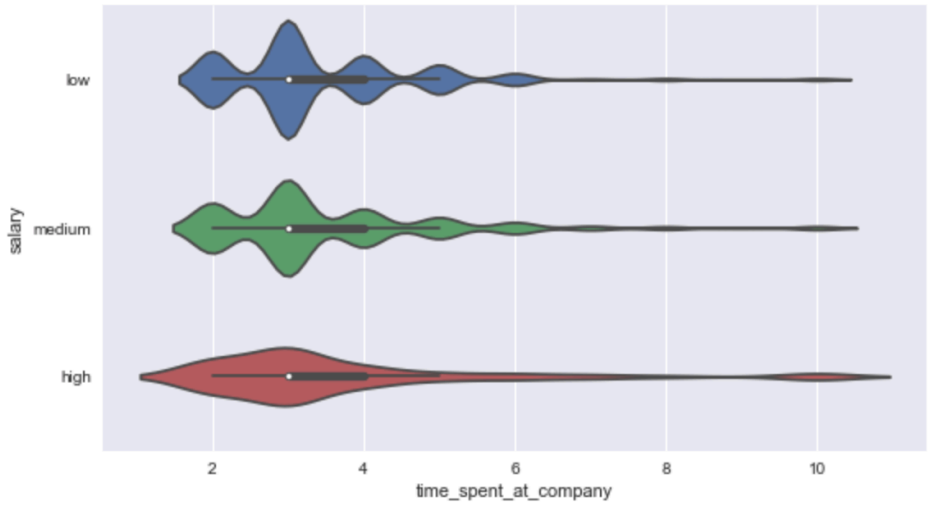


Figure 11

Figure-11 tells us there is no compensation by the firm for loyalty (i.e. workers staying with the firm for a longer period of time). This could be why people are leaving.

Next, I look at work accidents:

```
wa = data.work_accident
wa.value_counts()
```

```
0    12830
1     2169
Name: work_accident, dtype: int64
```

Around 14.5 % of employees had accidents. Other information given from the description about the distribution is not useful. From multi-variable analyses and looking back at the correlation matrix I see that surprisingly there isn't any particular linear relationships between `work_accident` and other variables.

Similarly, promotion in the last 5 years doesn't seem to play any crucial role amongst other variables. Correlation matrix confirms this. As said earlier, the lack of interaction between '`promotion_last_5years`' and other variables actually may indicate that this is one reason why employees are leaving.

```
pp = data.promotion_last_5years
pp.value_counts()
```

```
0    14680
1      319
Name: promotion_last_5years, dtype: int64
```

About 2.12% of workers have gotten a promotion; an extremely small amount of nearly 15,000 workers.

Figure-12 shows the different departments within the firm. Sales by far houses the greatest number of employees, followed by Technical, Support, and IT.

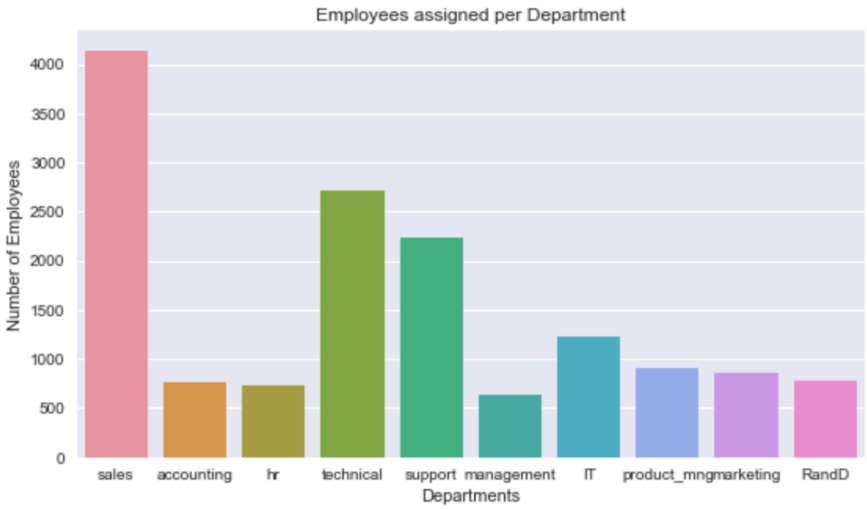


Figure 12

Looking at the '`salary`' variable I see most employees are in the '`low`' category, however the number between '`low`' and '`medium`' is not too different. It could be that those loyal customers I've seen in earlier models is the management who are the only ones making serious money and thus staying, whereas the general workforce is lacking motivation due to insufficient money and promotions. This is also why more years doesn't mean more pay.

```
1 data.salary.value_counts()
```

```
low       7316
medium    6446
high      1237
Name: salary, dtype: int64
```

I proceed by doing multidimensional analysis. *Figure-13* shows the most interesting information discovered so far so I will investigate this further. This scatter plot seems to indicate that there is a concentration of employees who work average hours (8-hour work day, as can be seen since the square spans across the 130-160 hour mark), however receive a poor evaluation score. This graph also shows that there are mainly 3 types of employees, (indicated by the 3 rectangles).

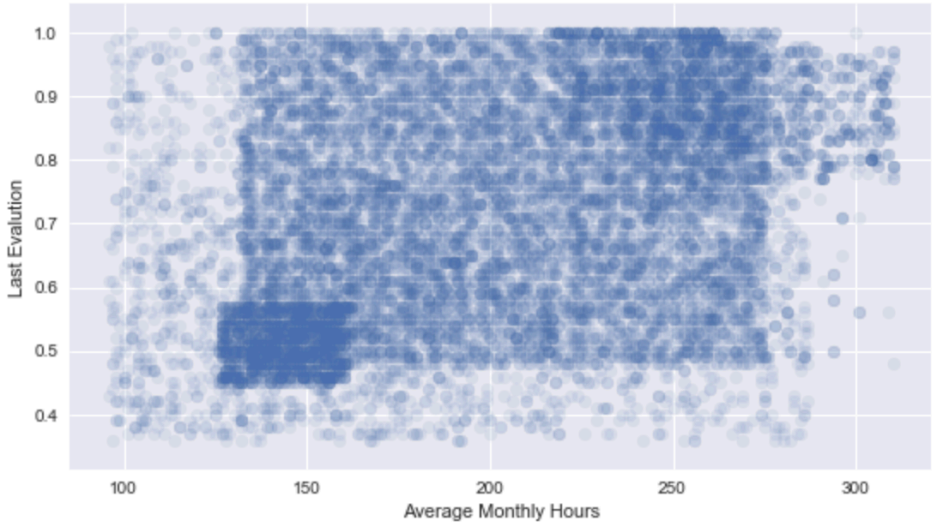


Figure 13

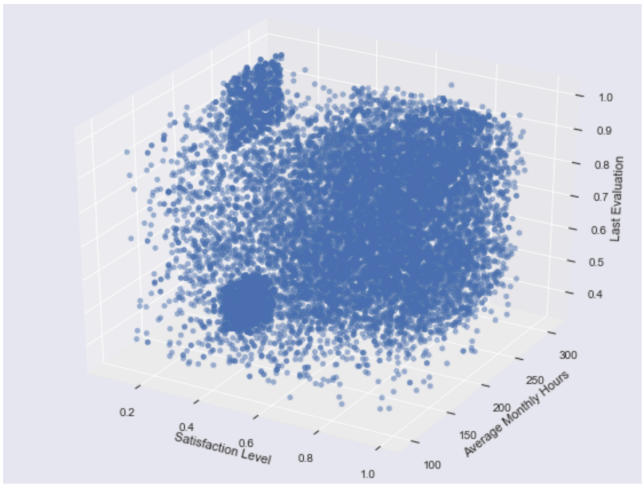


Figure 14a

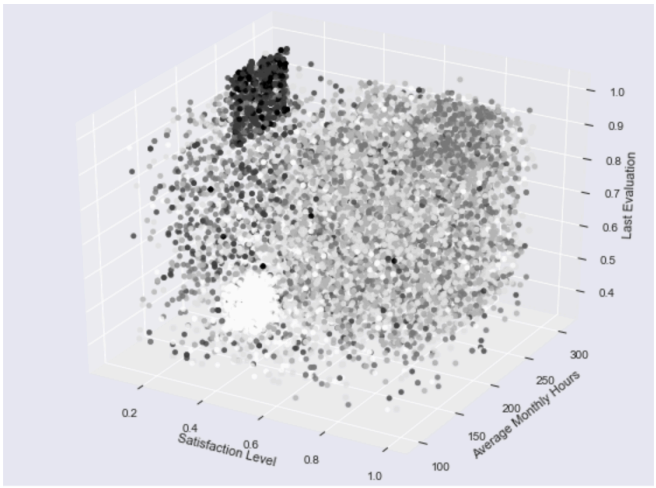


Figure 14b

Figure-14a displays a multivariable visualization with number of projects by color. Adjusting the default color parameter I see a clearer picture, *Figure-14b*. The black square at the back represents employees who put in long hours at the workplace (high monthly hours) working on numerous projects, but are currently not satisfied despite given a high rating for their hard work. In comparison, the white cluster is what the dark blue square represents on the 2D scatter plot. This plot shows some additional information indicating that this group is not really satisfied even though they receive a considerable amount of fewer projects than their colleagues. An important note to consider is both groups are at the far end of the '`satisfaction_level`' spectrum, thus I can conclude that few or a considerably high number of projects lead to low satisfaction (average working hours are also vastly different). I will keep these axes and map '`time_spent_at_company`' (term in office) to color and further identify possible trends, *Figure-15a*. Again, adjusting for color, I get a better picture, *Figure-15b*. I can infer that there is a cluster of employees who ***are*** satisfied, despite long hours (and according to the graph above numerous projects), while also being evaluated high for their work.

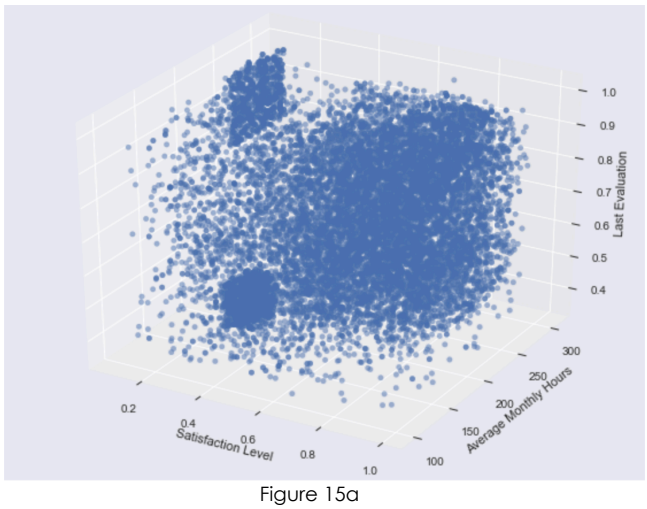


Figure 15a

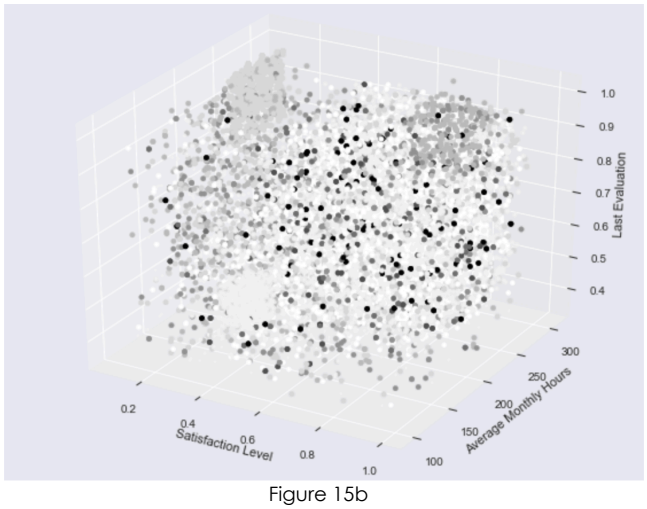


Figure 15b

Another crucial point is that the apathetic group (low to medium work hours, low evaluation, low satisfaction, few projects) are also quite new to the firm. Whereas the hard-working cluster (high hours, high evaluation, exceptionally low satisfaction) have been at the company for a little longer. Hypothesis: drawing from the problem domain, one possible reason for workers leaving (or general issue at the firm) could be the common problem that the firm is not giving its new comers adequate work (being the cause of apathy and detachment from the firm), and overworking the employees that already acquired the skills to be successful (leading to exhaustion and thus low satisfaction). This could be due to bad management.

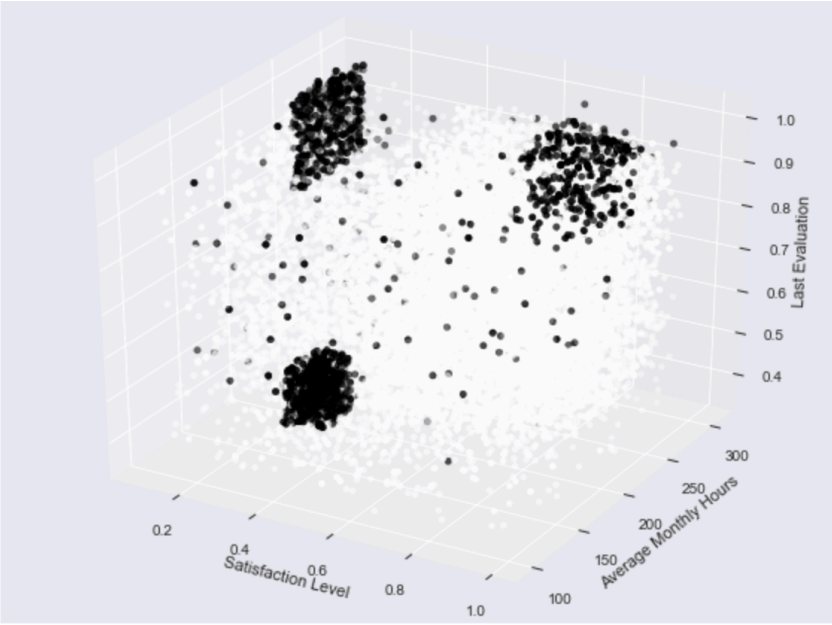


Figure 16

As a last step, I apply the classifier variable '`left`' to the plot, shown in *Figure-16*. Since '`left`' is a binary variable the black dots mean the employee left the firm and the white indicates they didn't. I can see that the 3 main groups are ***all*** leaving the company. I didn't expect highly evaluated employees to leave. I hypothesize the following as explanation:

1. Group # 1, the hard-working crew are leaving because they are over-worked by the company

2. Group # 2, the workers who've lost interest are leaving because they are bored of redundant work

3. Group # 3, the loyal employees who have been with the firm for 7+ years are leaving for 2 possible reasons: either they retire (ones being at the company for 10 years), or they simply move on because they are overqualified and/or receieve a higher paying position elsewhere.

3.2 Checking for Normality and Significance

Although most of the plots didn't show normal distributions regarding the data, I will take a brief look using QQ-plots. If a set has an approximate normal distribution, a QQ-plot of the sample observations will result in an approximate straight line. This will also tell me whether ANOVA, Chi-square or other tests will be applicable. Note that for the QQ-plots I will pick the standard normal distribution to compare my values against; i.e. I will plot the i^{th} ordered value against the $(i/n+1)^{th}$ quantile of the standard normal distribution.

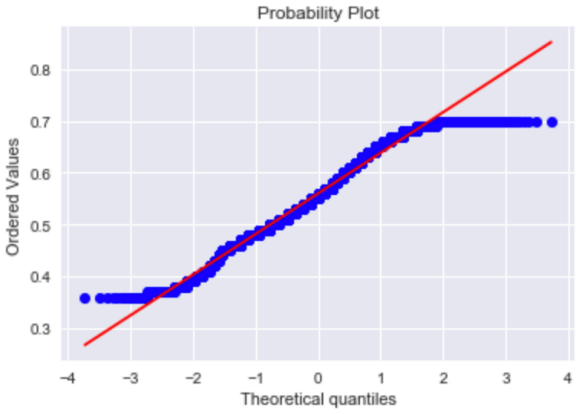


Figure 17a

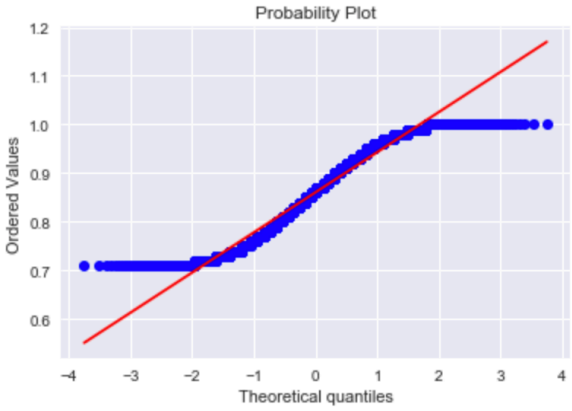


Figure 17b

Earlier in the EDA I made an observation that '`last_evaluation`' had an inverse bump (due to negative kurtosis) accompanied by two "shoulders" on either side. While the distribution itself is likely not normal, I investigate further whether it could have an underlying cluster or group that resembles normality. *Figure-17a* and *Figure-17b* show the low-half and high-half of the distribution, respectively, (code used for partition shown later). From the pictures it might seem that these distributions are normal. However, it is also possible that they are uniform distributions – the tails are truncated. The top part below the red line signifies the large values that are not as extreme as would be expected if the sample was from a normally distributed population. Similarly, the bottom part signifies the smallest values that are not as extreme as would be expected. Hence we get a curvature through the line.

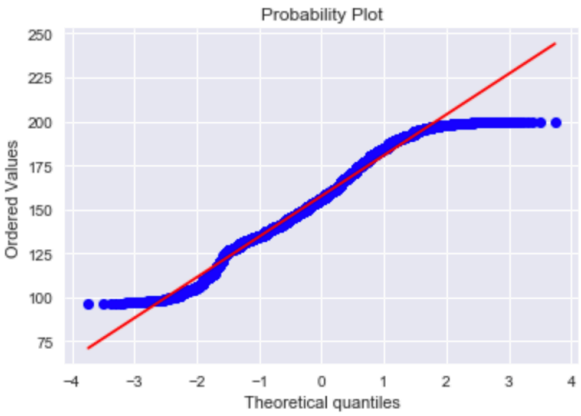


Figure 18a

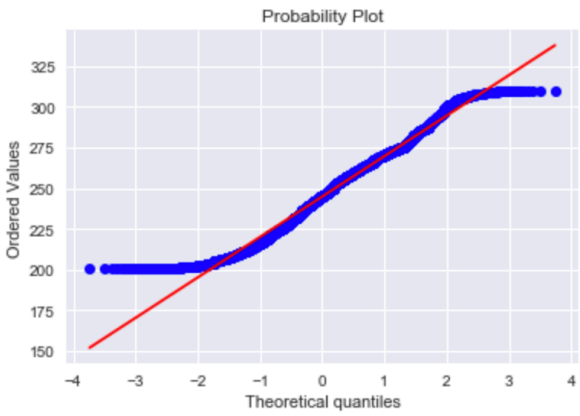
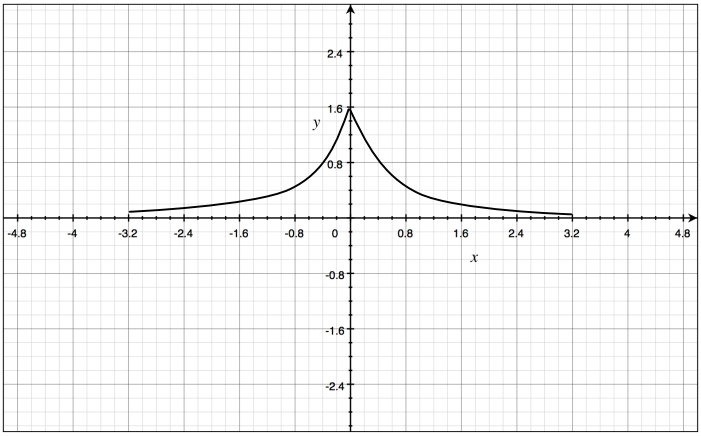


Figure 18b

Following the same line of reasoning as mentioned for evaluation, I noticed the same pattern in '`avg_monthly_hours`', and so accordingly I carried out the same test in the same manner. *Figure-18a* and *Figure-18b* show samples taken from the low and high end of the distribution. As can be seen, the results are very similar due the reasons already mentioned. Additionally this sort of pattern rules out a right-skewed sample (where both tail and head would be above the red line). Note also that the above samples used for the QQ-plot are just 1 out of 20 iterations. This was to make sure natural variability doesn't bias the evidence.

Next, I'll apply a series of normality tests for completeness and confirmation. For the following I will assume a significance level of α = 0.05. If the resulting p-value is less than this value, I reject the null hypothesis, and if it's greater than, I don't reject. Note that "not rejecting" is ***not*** the same as accepting. Why? Because the result is just one piece of evidence.

Example: here Jarque-Bera may give p-value > 0.05 and Kolmogorov-Smirnov *p-value < 0.05.*



Normality tests will be done on continuous variables alone. I will use Kolmogorov-Smirnov (KS) and Jarque-Bera for larger test samples (>2000). Because these normality tests pick up on different features of the distribution (KS measures the distance between the function and the cumulative distribution function (CDF); Jarque-Bera checks if skew and kurtosis are normal) I justify using both. Specifically, KS compares the largest absolute difference between the CDFs. For the following tests the hypotheses are the same:

H_0: normally distributed

H_A: not normally distributed

I will start with '`satisfaction_level`' and run Kolmogorov-Smirnov:

```
1 st.kstest(data.satisfaction_level, 'norm')
```

```
KstestResult(statistic=0.53585639258517215, pvalue=0.0)
```

As expected I get a value below the significance level (0.05) and so I reject the null hypothesis – this distribution is not normal. *Note: the actual p-value is* ***not*** *0, python doesn't display values smaller than 3.128173…e-63 on my system. This applies for other values below.*

Next variable is '`last_evaluation`':

```
st.jarque_bera(data.last_evaluation)
```

```
(961.20022987139532, 0.0)
```

Again, value is below the significance level (0.05) and so I reject the null hypothesis – this distribution is also not normal. However, I can partition the `Series` and confirm my observations gathered during the QQ-plots. I will quickly split above and below the 0.7 mark:

```
low_half_eval = data[data.last_evaluation<=0.7]
high_half_eval = data[data.last_evaluation>0.7]
```

```
print (st.kstest(low_half_eval.avg_monthly_hours, 'norm'))
print (st.kstest(high_half_eval.avg_monthly_hours, 'norm'))
```

```
KstestResult(statistic=1.0, pvalue=0.0)
KstestResult(statistic=1.0, pvalue=0.0)
```

The resulting p-values are still well below the significance level so I reject the null hypothesis. Thus, these are also not normally distributed. Next variable is '`avg_monthly_hours`':

```
st.jarque_bera(data.avg_monthly_hours)
```

```
(812.07038927330223, 0.0)
```

P-value is below 0.05 so I reject the null hypothesis. Since the distribution looked as if it may have had 2 normal distributions inside I will split at the 200 hour mark and check (samples are smaller so I'll use Shapiro–Wilk for better accuracy):

```
low_half = data[data.avg_monthly_hours<=200]
high_half = data[data.avg_monthly_hours>200]
```

```
print (st.shapiro(low_half.avg_monthly_hours))
print (st.shapiro(high_half.avg_monthly_hours))
```

```
(0.980006992816925, 3.198931551610232e-31)
(0.9802083969116211, 5.350604371524861e-31)
```

The resulting p-values are very close to zero and obviously below my significance level, thus I reject the null hypothesis. These samples are not normally distributed.

Having tested all the continuous variables and found that none conform to a normal distribution, I proceed without relying too much on statistical tests that require it. For this reason I cannot rely on T-test, or standard ANOVA. Consequently, I will adhere to the Pearson's chi-square test (goodness-fit), and Kruskal-Wallis (one-way ANOVA on ranks).
I will start with the Chi-square test: assume a significance value of $\alpha = 0.05$

H_0: The proportion of low, medium, and high salary is 49%, 43%, and 8%, respectively.

H_A: At least one of the proportions in the null hypothesis is false

My null hypothesis specifies the proportion of observations at each level of the categorical variable, and the alternate hypothesis is that at least one of the specified proportions is not true (StatTreck, 2017).

```
global_counts = pd.DataFrame(data.salary.value_counts())
global_counts
```

	salary
low	7316
medium	6446

```
global_ratios = global_counts / len(data)
global_ratios
```

	salary
low	0.487766
medium	0.429762

Departments	Degree of Freedom *(number of variable categories - 1)*	Exptected Frequency Counts	Test statistics	Critical-value (critical value associated with 2 degrees of freedom)	P-value *(resulting)*	Decision
Sales	2	exs = global_ratios * len(sales) salary low 2019.350623 medium 1779.214614 high 341.434762	18.537	0.103	9.43E-05	Reject null
Technical	2	exs = global_ratios * len(technical) salary low 1326.723115 medium 1168.952597 high 224.324288	4.382	0.103	0.111	Don't reject null
Support	2	exs = global_ratios * len(support) salary low 1087.230082 medium 957.939463 high 183.830455	13.421	0.103	0.001	Reject null
IT	2	exs = global_ratios * len(IT) salary low 598.488699 medium 527.317955 high 101.193346	3.567	0.103	0.168	Don't reject null
Product Management	2	exs = global_ratios * len(product_mng) salary low 439.964798 medium 387.645310 high 74.389893	0.881	0.103	0.643	Don't reject null
Marketing	2	exs = global_ratios * len(marketing) salary low 418.503100 medium 368.735782 high 70.761117	2.000	0.103	0.367	Don't reject null
R&D	2	exs = global_ratios * len(RandD) salary low 383.871725 medium 338.222682 high 64.905594	5.310	0.103	0.070	Don't reject null
Accounting	2	exs = global_ratios * len(accounting) salary low 374.116408 medium 329.627442 high 63.256150	2.606	0.103	0.271	Don't reject null
HR	2	exs = global_ratios * len(hr) salary low 360.458964 medium 317.594106 high 60.946930	5.132	0.103	0.0768269	Don't reject null
Management	2	exs = global_ratios * len(management) salary low 307.292486 medium 270.750050 high 51.957464	345.312	0.103	1.03E-75	Reject null

Figure 19

We reject the null-hypotheses under two equivalent rules: p-value ≤ 0.05; chi-square statistic is greater than the critical value. This defines the rejection region. The critical-value associated with the statistic is the probability that having 2 degrees of freedom is more

extreme than 0.05, which was derived from the 'Lower-tail critical values distribution table' included in the appendix. In this approach, the sampling method was simple random sampling, the variables being studied were categorical and each level of variable had an expected count of at least 50. From *Figure-19* I can conclude that the Sales, Support, and Management department differs from the theoretical global distribution for salary. Drawing from the problem domain it makes sense that managers are outliers and given earlier findings, Sales is probably out of line due to its number of workers. Support however surprisingly does not follow the global trend.

Next, I will use Kruskal-Wallis (one-way ANOVA on ranks). The 3 ordinal groups in question will be '`rookies`', '`experienced`', and '`veterans`'. These 3 features are created from '`time_spent_at_company`', roughly correlating with the 3 clusters of employees we have discovered so far. Rookies, experienced, and veteran workers have been at the company 2-3 years, 4-6 years, and 7-10 years, respectively.

```
rookies = data[data.time_spent_at_company<=3]
experienced = data[ (4 <= data.time_spent_at_company) & ( data.time_spent_at_company <= 6)]
veterans = data[data.time_spent_at_company>=7]
```

My assumed alpha level is α = 0.05. Hypotheses are:

H_0: There is no difference between the medians of the groups

H_A: There is at least one group who's median is different

Degrees of freedom is k – 1, where k is the number of groups. Thus 3 – 1 = 2. For the critical value I will adhere to the Chi-square table (upper tail), which gives 5.991. If the chi-square result (the test statistic) is greater than this value I reject the null hypothesis indicating that there is some difference between the 3 groups.

```
st.kruskal(rookies, experienced, veterans)
```

```
KruskalResult(statistic=38.160707389222807, pvalue=5.1702039802457141e-09)
```

Note that the result (above) is calculated after the data is already ranked, i.e. the values are organized from smallest to largest. Because the statistical value is greater than the critical level, it follows that there is at least one group with a significantly different median among the three groups. This confirms the findings so far.

3.3 Dimensionality Reduction

I will perform dimensionality reduction and feature extraction to avoid excessive computational costs for learning and limit overfitting. If my data does overfit, it will accurately identify on the training set but perform inaccurately on the test set. I want to preserve the relevant information in the data needed to learn accurate models while also addressing these two issues. I will experiment with two different types of dimensionality reduction: principle component analysis (PCA), and Isomap. PCA performs linear dimensionality reduction to emphasize variation and strong patterns within the data whereas Isomap uses a non-linear reduction method. As preparation, I will normalise my dataset. I will also reduce the size of the dataset to 8,000 (simple random sample) points in order to be memory and time efficient.

```
N = StandardScaler()
N.fit(data)
data_norm = N.transform(data)
```

```
index = np.random.randint(0,data_norm.shape[0],size=8000)
```

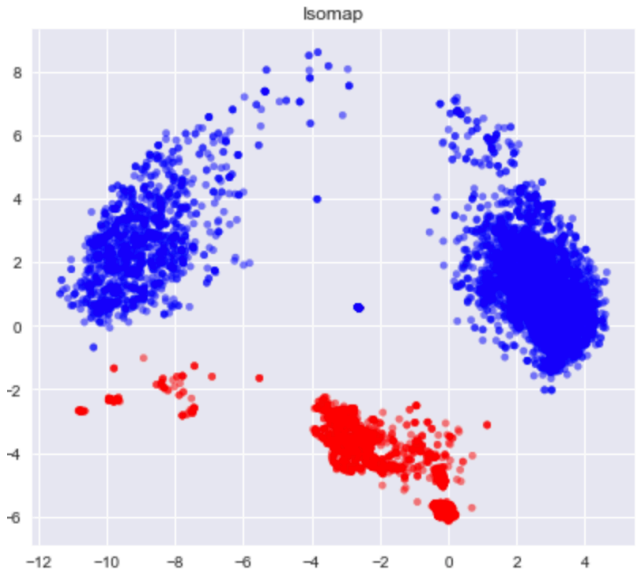


Figure 20a

Figure 20b

Figure-20a and *Figure-20b* show dimensionality reduction with Isomap and PCA, respectively. The blue dots signify employees who've stayed at the firm and the red dots indicate employees who left. Because there aren't any strong linear relationships between variables (as seen from the correlation matrix already), PCA performs poorly. It tries to find directions of maximum variance of the data. Isomap however correctly separates those workers who left and those who stayed. The mapping method in Isomap is done using classical metric multidimensional scaling based on pairwise distance between data points that is calculated by Euclidian distance (J. B. Tenenbaum, 2000) (Vin de Silva, 2003).

3.4 Cluster Analysis

Next, I will follow up with clustering in reduced-dimensional space. To reiterate, the methods so far indicated a high likelihood of 3 types of employees: discouraged, over-worked, and veterans. Earlier in the EDA *Figure-16* also indicated that **all** three types of workers account for employees leaving. The following cluster analyses will be used to solidify these patterns. I will start by using k-means from the insight gained from *Figure-13* to *15*. The algorithm for choosing the initial values for cluster centers will be '*k-means++*'.

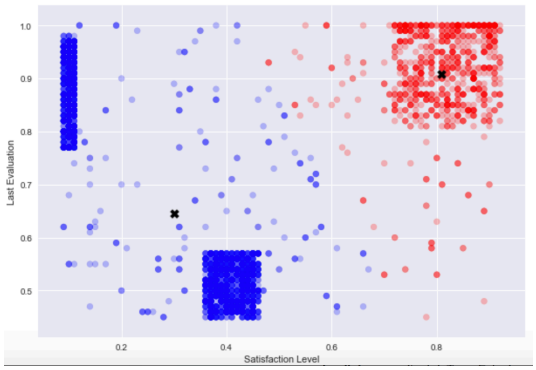


Figure 21a; `n_clusters=2`

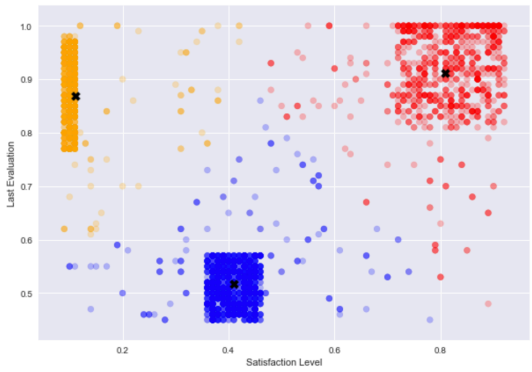


Figure 21b; `n_clusters=3`

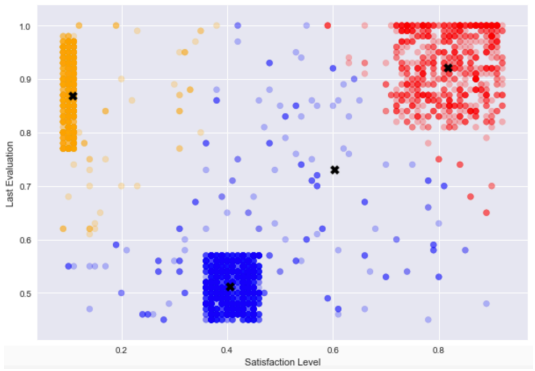


Figure 21c; `n_clusters=4`

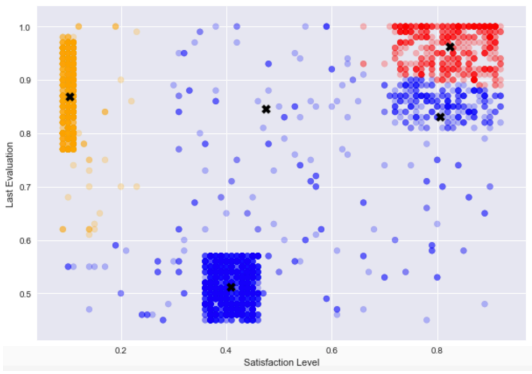


Figure 21d; `n_clusters=5`

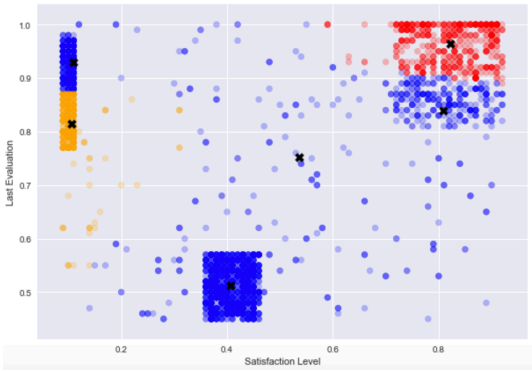


Figure 21e; `n_clusters=6`

Figure-21a to '*e*' show the different iterations, differentiated by the value that the '`n_clusters`' parameter takes – which determines the number of centroids to generate. Evidently,

Figure-21b yields the best result correctly identifying the 3 clusters. The orange cluster represents the over-worked employees, the blue cluster the discouraged group and the red cluster the veterans. *Note: the different clusters are marked with the 'x'.*

The second form of cluster analysis will be 'Density-based spatial clustering of application with noise' (DBSCAN). Due to the nature of the data, I wanted to experiment with a method that finds core samples of high-density and deduces clusters from it. The algorithm groups points that are closely packed and marks outliers in low-density regions (Martin Ester, 1996). *Figure-22a* to *'f'* shows the DBSCAN classifying employees who have left, with noise being superimposed onto the data (only shown on the final figure). *Note: the color of cluster changes randomly each iteration.*

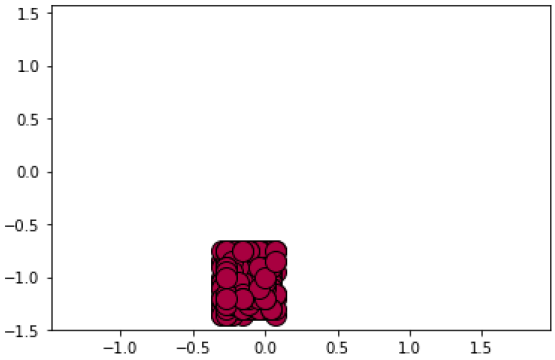


Figure 22a

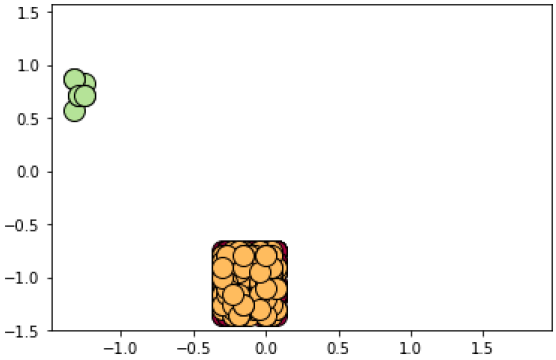


Figure 22b

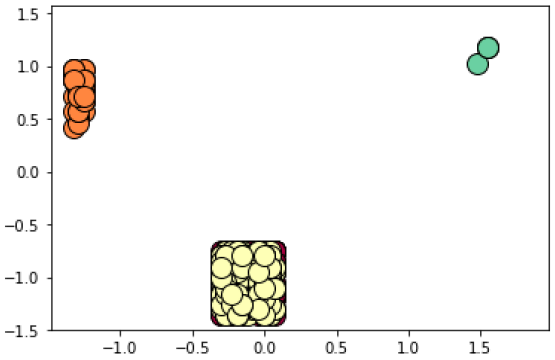


Figure 22c

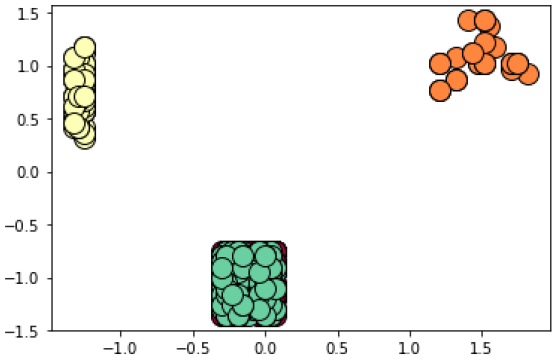


Figure 22d

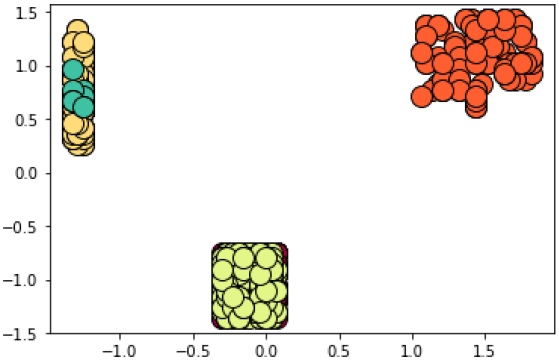


Figure 22e

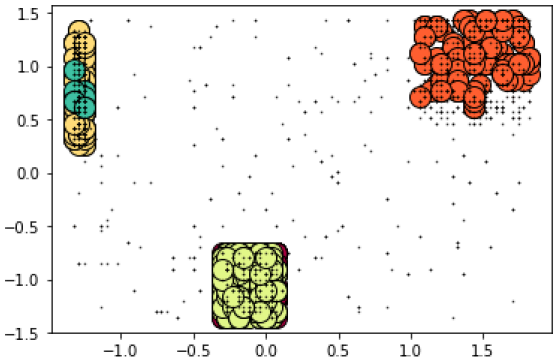


Figure 22f

As can be seen, DBSCAN is more accurate than k-means but it's weakness is noticeable later on. The algorithm does well where clusters are similar in density. Adjusting for '`eps`' (the maximum distance between two samples for them to be considered to be in the same neighbourhood) and '`min_samples`' (the number of samples, or total weight, in a neighbourhood for a point to be considered as a core point) one can see that the algorithm starts to harness a greater number of clusters (Pedregosa, 2011). The algorithm gathers points to be in the same cluster when it can "hop" from one to the other no more than epsilon (the threshold distance). I start by setting epsilon to 0.9 and specifying a high number of minimum neighbours (ex. 300-400). Consequently, the algorithm finds the densest region, shown in *Figure-22a*. I then gradually lower '`min_samples`' thereby increasing the likelihood that a point can be considered part of the sample. At `min_samples=150` we can see the 3 tightest regions being picked up by the algorithm (*Figure-22d*). However, by the time a point only requires 125 samples to be considered a core point, the algorithm starts to pick up a 4th cluster. This is likely due to the fact that the data still contains too many dimensions. Doing some deeper EDA on our findings from earlier we can see confirmation that the data varies in density. This is clearer once *Figure-16* is adjusted for transparency – sparser points become less visible but the tighter regions are still noticeable (shown in *Figure-23*).

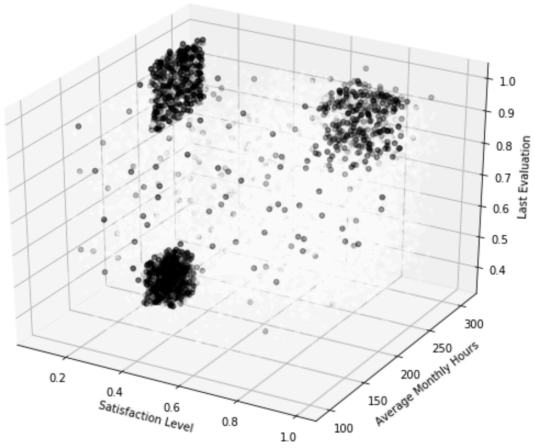


Figure 23

3.5 Decision Tree Classifier

I will test using a decision tree classifier to correctly determine whether an employee falls into the category of '`stayed`' or '`left`'. This will also highlight my top variables and further explore their impact. Additionally, this will assist me in understanding the process in other models when I use less variables. The tree in *Figure-24* (below) has a `max_depth=3` and an accuracy score of 95%. The most important variable is located at the root of the tree, in this case `satisfaction_level`. The 3 distinct groups identified earlier can also be picked out: the "veterans" correspond to path [`satisfaction_level`<=0.465{False} →

`time_spent_at_company`<=4.5{False} → `last_evaluation`<=0.805{False}]; the "over-worked" crowd corresponds to path [`satisfaction_level`<=0.465{True} → `projects`<=2.5{False} → `last_evaluation`<=0.115{True}]; and the "discouraged" crowd corresponds to path [`satisfaction_level`<=0.465{True} → `projects`<=2.5{True} → `last_evaluation`<=0.57{True}]. Lastly, the path [`satisfaction_level`<=0.465{False} → `time_spent_at_company`<=4.5{True} → `avg_monthly_hours`<=290.5{True}] highlights those workers who chose to stay at the firm – 76% of the workforce.

```
DTC = DecisionTreeClassifier(max_depth=3)
cv_results = cross_val_score(DTC,X_train, Y_train, cv=kfold, scoring="accuracy")
cv_results.mean()
```

```
0.95239079072183286
```

```
<IPython.core.display.Image object>
```

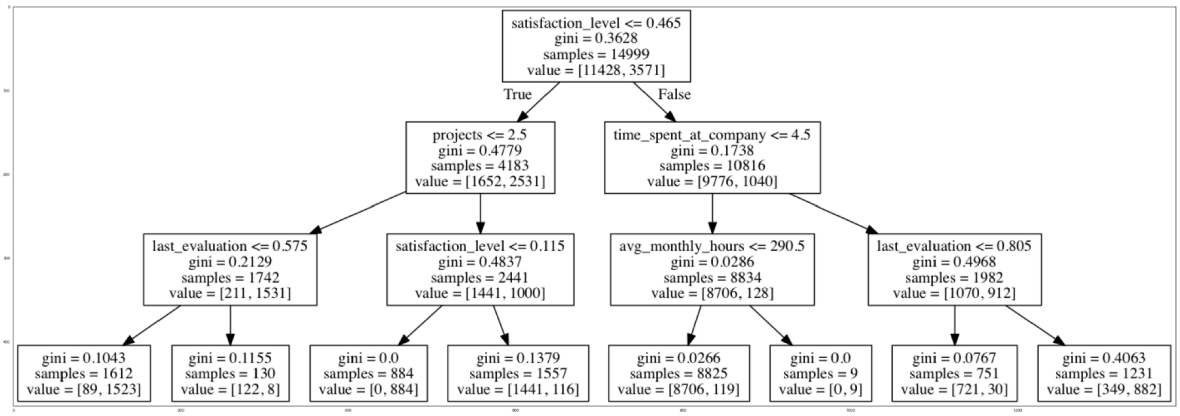


Figure 24

It's also evident that `satisfaction_level` is probably the most important feature, followed by `projects` and `time_spent_at_company`. *Figure-25a* to *'b'* show accuracy for the decision tree with `max_depth=5` and `max_depth=8`, respectively. At greater depths the tree's classifying capabilities gradually start to decline, thus depth 8 seems the most optimal.

```
DTC = DecisionTreeClassifier(max_depth=5)
cv_results = cross_val_score(DTC,X_train, Y_train, cv=kfold, scoring="accuracy")
cv_results.mean()
```

```
0.9725303962653612
```

Figure 25a

```
DTC = DecisionTreeClassifier(max_depth=8)
cv_results = cross_val_score(DTC,X_train, Y_train, cv=kfold, scoring="accuracy")
cv_results.mean()
```

```
0.98093213134761403
```

Figure 25b

3.6 Random Forest Classifier

Next, I will use a random decision forest for classification which essentially builds from my earlier attempt to classify a class variable by constructing a multitude of decision trees. This method works by averaging multiple decisions trees and outputting the class that is the mode of the classes from the individual trees being "grown". This is also why the label ensemble method applies, because its decision comes from a collection of results. I will fit this algorithm with various sub-samples of the dataset to improve accuracy. The python implementation will also always pick the same sample size, however these samples will be drawn with replacement.

```
rfc = RandomForestClassifier()
rfc.fit(X_train, Y_train)
rfc_score_train = rfc.score(X_train, Y_train)
print("Train set accyracy score: " + str(rfc_score_train))
rfc_score_test = rfc.score(X_test, Y_test)
print("Test set accuracy score: " + str(rfc_score_test))
```

```
Train set accyracy score: 0.998666555546
Test set accuracy score: 0.988333333333
```

Figure 26

This model seems to be very accurate, correctly classifying employees leaving or staying 98.8% of the time.

```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test, preds)
```

```
array([[4538,    7],
       [  61, 1394]])
```

```
---Random Forest Model---
```

	precision	recall
0	0.99	1.00
1	0.99	0.96
avg / total	0.99	0.99

The confusion matrix allows us to visualize the performance of the machine learning model. In the implementation above, cell [0,0] represents the *true negatives*, [1,0] *false negatives*, [0,1] *false positives* and [1,1] *true positives*. Thus, the result contained 4538 true negatives and *7 false negatives* – essentially what this is saying is 4538 employees predicted not to leave didn't, however 7 predicted to stay actually left the company. Similarly, there were 61 false positives and 1394 true positives – essentially 61 employees predicted to leave ended up staying whereas 1394 predicted to leave actually did leave. An important point here is that the main concern for the company are the false negatives (It's in the firm's interest to keep employees who might leave later than to send away ones that actually want to be loyal). Moreover, `precision` describes how useful the classification was whereas `recall` describes the completeness of the procedure. I can conclude that a large amount of relevant points was selected (recall), and a large amount of points were relevant (precision). The numbers are rounded.

3.7 Logistic Regression

My choice for this technique is justified because my dependent variable is binary – employee either left (`1`) or stayed (`0`). In this section I will employ logistic regression to predict whether an employee belongs to one of these classes based on a selection of independent variables. Logistic regression models the probability of success based on the formula:

$$\text{logit}[\theta(x)] = \log\left[\frac{\theta(x)}{1-\theta(x)}\right] = \alpha + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_i x_i$$ (Rupesh Khare, 2011)

The equation above shows the relationship between, the dependent variable "success", denoted (θ) and the independent variables which will be the predictors, denoted as x_i. Alpha (α) is the constant in the equation and β are the coefficients of the predictor variables (Rupesh Khare, 2011).

```
import statsmodels.api as sm
iv = ['satisfaction_level','projects','time_spent_at_company', 'int']
logReg = sm.Logit(y_test, X_test[iv])
answer = logReg.fit()

answer.summary
answer.params
```

```
Optimization terminated successfully.
         Current function value: 0.469234
         Iterations 6

satisfaction_level      -3.758404
projects                -0.097457
time_spent_at_company    0.181405
int                      0.647410
dtype: float64
```

I proceeded to use the 3 top variables identified by the decision tree classifier earlier. The values above are the calculated coefficient values for each variable and the constant '`int`' signifies the effect of any uncontrollable variable that may affect the result. From the above I can model the equation as:

Predictive score for employee leaving = `satisfaction_level`(-3.758404) + `projects`(-0.097454) + `time_spent_at_company`(0.181405) + 0.647410

Substituting values 0.7 for satisfaction, 3 for projects, and 3 for time at company would result in 0.15 (rounded):

```
coef = answer.params
def y (coef, satisfaction_level, projects, time_spent_at_company) :
    return coef[3] + coef[0]*satisfaction_level + coef[1]*projects + coef[2]*time_spent_at_company
y1 = y(coef, 0.7, 3, 3)
p = np.exp(y1) / (1+np.exp(y1))
p
```

```
0.15037967159488888
```

Meaning that a specific employee corresponding with these values would have a 15% chance of leaving the company. I found that fewer independent variables tend to be more

accurate, so I maintain this amount. However, perhaps expectedly, using different important variables also indicated by the decision tree gives us different results. For example:

```
import statsmodels.api as sm
iv = ['satisfaction_level','last_evaluation','time_spent_at_company', 'int']
logReg = sm.Logit(y_train, X_train[iv])
answer = logReg.fit()

answer.summary
answer.params
```

```
Optimization terminated successfully.
         Current function value: 0.467233
         Iterations 6

satisfaction_level      -3.769022
last_evaluation          0.207596
time_spent_at_company    0.170145
int                      0.181896
dtype: float64
```

Here I replaced projects with '`last_evaluation`' and got the values shown above. This would modify the equation to:

> ***Predictive score for employee leaving*** = `satisfaction_level`(-3.769022) + last_evaluation(0.207596) + `time_spent_at_company`(0.170145) + 0.181896

```
coef = answer.params
def y (coef, satisfaction_level, last_evaluation, time_spent_at_company) :
    return coef[3] + coef[0]*satisfaction_level + coef[1]*last_evaluation + coef[2]*time_spent_at_company
y1 = y(coef, 0.7, 0.8, 3)
p = np.exp(y1) / (1+np.exp(y1))
p
```

```
0.14431462559738251
```

With the same chosen values as before, here we get answer of 14% (rounded). Using this variation the model deviates only by 1% giving a slightly lower probability that an employee with these attributes would leave. I argue that due to our decision tree giving higher priority to the '`projects`' variable, the first variation of this logistic regress model is more accurate. Nevertheless '`satisfaction_level`' persists to be our most important attribute and thereby the most indicative of whether an employee will either leave or stay at the firm.

```
model = LogisticRegression()
model = model.fit(X_train,y_train)
print ("Test set accuracy score: " + str(model.score(X_test,y_test)))
print ("Train set accuracy score: " + str(model.score(X_train,y_train)))
```

```
Test set accuracy score: 0.761333333333
Train set accuracy score: 0.764397033086
```

Lastly, I want to point out the accuracy of the model (above). In comparison to the random forest classifier, logistic regression scores much lower – achieving 76% (rounded) on the test set. Implying that the former model is better suited for identifying employees who are prone to leave.

3.8 Summary

In conclusion, the exploration done on this dataset has: allowed us to understand the structure of the company in question, revealed certain underlying characterises that employees possess, and highlighted which features are the most important indicators of said employees staying or leaving their positions. Specifically, the findings are:

- There are 3 types of employees that are leaving the company. These workers account for 23.8% of the staff.
- The first group, dubbed as "veterans", are highly-evaluated, receive a lot of work, are very satisfied, but despite being with the company for many years, are leaving because they are not getting compensated (promoted and payed comfortably) for their dedication.
- The second group, dubbed as the "over-worked", are experienced and recognised for their skills (high-evaluation), but for precisely this reason they are given an unreasonable amount of work which tanks their happiness levels. Because Managers fail to recognise that these employees are slowly losing the will to live, they are seeing them leave.
- The third group are the "new-recruits" and in a sense they have it the worst. They are undervalued, and despite showing potential, the company is not moving them up or around the company. Consequently, they are unsatisfied because there is no motivation to perform – no good pay, no interesting projects, just consecutive bad ratings.
- Despite trying to create new features like `work efficiency` by tapping into the problem domain, the machine learning models continually relied on mainly 5 features to correctly classify whether an employee would leave or stay: `satisfaction_level`, workload (`projects`), and tenure (`time_spent_at_company`), in that order.

Possible advice to increase employee retention could be: be wary of employees who are getting very poor or extremely high-performance scores during company evaluation. Consider setting up a program to compensate your most valued staff while developing a scheme to train and expand the knowledge of your new recruits. Moreover, if the bosses want to predict who is the most liable to leave their position, they should strongly consider using a model that implements a random forest classifier.

References

Rupesh Khare, D. K. C. K. C. G. G., 2011. *Employee Attrition Risk Assessment using Logistic Regression Analysis.* Ahmedabad, Indian Institute of Management.

StatTreck., 2017. *Stat Treck.* [Online]
Available at: http://stattrek.com/chi-square-test/goodness-of-fit.aspx?Tutorial=AP
[Accessed 10 11 2017].

J. B. Tenenbaum, V. d. S. a. J. C. L., 2000. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science,* 290(5500), pp. 2319-2323.

Martin Ester, H.-P. K. J. S. X. X., 1996. *ADensity-BasedAlgorithmfor DiscoveringClusters in LargeSpatial Databaseswith Noise,* München: University of Munich.

Pedregosa, 2011. Scikit-learn: Machine Learning in Python. *JMLR,* Volume 12, pp. 2825-2830.

Vin de Silva, J. B. T., 2003. *Global versus local methods in nonlinear dimensionality reduction,* Stanford: MIT press.

Wallis, K., 1952. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association,* 47(260), pp. 583-621.

Database: https://www.kaggle.com/ludobenistant/hr-analytics/data

Articles used for background reading:

- https://hbr.org/2016/11/hr-cant-change-company-culture-by-itself
- https://www.wsj.com/articles/why-the-best-leaders-want-their-superstar-employees-to-leave-1475460841
- https://www.linkedin.com/pulse/human-resources-career-nice-people-brian-walker

Appendix

Tables used for critical values of the chi-squared distribution: http://www.itl.nist.gov/div898/handbook/eda/section3/eda3674.htm

Upper-tail critical values of chi-square distribution with ν degrees of freedom

	Probability less than the critical value				
ν	0.90	0.95	0.975	0.99	0.999
1	2.706	3.841	5.024	6.635	10.828
2	4.605	5.991	7.378	9.210	13.816
3	6.251	7.815	9.348	11.345	16.266
4	7.779	9.488	11.143	13.277	18.467
5	9.236	11.070	12.833	15.086	20.515

Lower-tail critical values of chi-square distribution with ν degrees of freedom

	Probability less than the critical value				
ν	0.10	0.05	0.025	0.01	0.001
1.	.016	.004	.001	.000	.000
2.	.211	.103	.051	.020	.002
3.	.584	.352	.216	.115	.024
4.	1.064	.711	.484	.297	.091
5.	1.610	1.145	.831	.554	.210

Some larger junks of code for major diagrams (or groups of diagrams):

Code for Figure-4

```
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 9
fig_size[1] = 5
plt.rcParams["figure.figsize"] = fig_size
plt.gcf().set_size_inches(*fig_size)
corr = data.corr()
corr = (corr)
sb.heatmap(corr,
           xticklabels=corr.columns.values,
           yticklabels=corr.columns.values)
sb.plt.title('Heatmap of Correlation Matrix')
```

Code for Figure-5 (similar for 6,7,9,10)

```
fig, axs = plt.subplots(nrows= 3, figsize=(13, 5))
```

```
sb.kdeplot(employees_left.satisfaction_level, ax=axs[0], shade=True, color="r")
kde_plot = sb.kdeplot(employees_stayed.satisfaction_level, ax=axs[0], shade=True,
color="b")
kde_plot.legend(labels=['left', 'stayed'])
hist_plot = sb.distplot(data.satisfaction_level, ax=axs[1], bins = 30, color =
'orange')
box_plot = sb.boxplot(data.satisfaction_level, ax=axs[2])
kde_plot.set(xlim=(0,1.1))
hist_plot.set(xlim=(0,1.1))
box_plot.set(xlim=(0,1.1));
```

Code for Figure-16 (similar for 14,15,23)

```
width = 10
height = 8
fig = plt.figure(figsize=(width, height))
ax = fig.add_subplot(111, projection='3d')
x = data['satisfaction_level']
y = data['avg_monthly_hours']
z = data['last_evaluation']
cc = data['left']
ax.scatter(xs=x, ys=y, zs=z, c=cc, alpha = 0.25, cmap='gray_r')
ax.set_xlabel('Satisfaction Level')
ax.set_ylabel('Average Monthly Hours')
ax.set_zlabel('Last Evaluation')
plt.show()
```

Code for Figure-17 (similar for 18)

```
st.probplot(low_half_eval.last_evaluation, plot = plt);
```

Code for Figure-20

```
left_colors = data["left"].map(lambda s : "b" if s==0 else "r")
fig, axes = plt.subplots(1,2,figsize=(15,6))
axes[1].scatter(iso_representation[:,0],iso_representation[:,1],
            c = left_colors[index],alpha=0.5,s=20)
axes[1].set_title("Isomap")
axes[0].scatter(pca_representation[:,0],pca_representation[:,1],
            c = left_colors[index],alpha=0.5,s=20)
axes[0].set_title("PCA")
```

Component Variable output from PCA

```
print pca.components_
```

```
[[-0.15466669  0.47628145  0.55906693  0.54095109  0.32233294 -0.06821813
   0.18754567 -0.00584545 -0.01704403]
 [ 0.58355942  0.27750059  0.09279646  0.13519911 -0.0916641   0.23960084
  -0.61423581  0.18697783  0.27839654]]
```

Code for Figure-21

```
kmeans_colors = ['red' if c == 0 else 'orange' if c == 2 else 'blue' for c in
kmeans.labels_]
fig = plt.figure(figsize=(10, 7))
plt.scatter(x="satisfaction_level",y="last_evaluation", data=data_left,
alpha=0.25,color = kmeans_colors)
plt.xlabel("Satisfaction Level")
plt.ylabel("Last Evaluation")
plt.scatter(x=kmeans.cluster_centers_[:,0],y=kmeans.cluster_centers_[:,1],color="bl
ack",marker="X",s=100)
plt.show()
```

Code for Figure-22

```
data = pd.read_csv("/Users/benjaminfocktern/Desktop/DataMining
(IN3011)/Project/HR_data.csv", index_col=False)
```

```
data['sales'].replace(['sales', 'accounting', 'hr', 'technical', 'support',
'management',
        'IT', 'product_mng', 'marketing', 'RandD'], [0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
inplace = True)
data['salary'].replace(['low', 'medium', 'high'], [0, 1, 2], inplace = True)
factor_scanned_data = pd.DataFrame()
factor_scanned_data = data


data_left = factor_scanned_data.copy()[factor_scanned_data['left']==1]
len(data_left['left'])
dat = data_left.copy().drop(['left'], axis=1)


dat = StandardScaler().fit_transform(dat)

db = DBSCAN(eps=0.9, min_samples=250).fit(dat)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

unique_labels = set(labels)
colors = plt.cm.Spectral(np.linspace(0, 1, len(unique_labels)))
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = 'k'

    class_member_mask = (labels == k)

    xy = dat[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
             markeredgecolor='k', markersize=14)

    xy = dat[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
             markeredgecolor='k', markersize=0, alpha = 0.9)

plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```

Code for Figure-24

```
dot_data = StringIO()
tree.export_graphviz(DTC, out_file=dot_data,feature_names=train_features)

graph = pydot.graph_from_dot_data(dot_data.getvalue())[0]
graph.set_lwidth(200)
graph.set_lheight(100)

display(Image(graph))
png_str = graph.create_png(prog='dot')
sio = StringIO()
sio.write(png_str)
sio.seek(0)
img = mpimg.imread(sio)
imgplot = plt.imshow(img, aspect='equal')
fig_size = plt.rcParams["figure.figsize"]
fig_size[0] = 50
fig_size[1] = 50
plt.rcParams["figure.figsize"] = fig_size
plt.gcf().set_size_inches(*fig_size)
plt.show(block=False)
```