CA4010 - Data Mining Report

Anna Parisi - Ploumpi & Rahma Ahmed

(15104869 & 14305416)

Date: 10/12/2017

Predicting employee job involvement

Table Of Contents

Introduction

For this assignment, we decided to focus on how different factors contribute to a person's motivation and commitment to what they are doing, and more specifically, how things like environment satisfaction, distance from home or years since last promotion influence how involved a person is in their job.

Dataset Description

The dataset we used is called "IBM HR Analytics Employee Attrition & Performance" and we found it on Kaggle (here). The original dataset contained thirty-five (35) attributes, containing information about each employee (e.g. marital status, education, age etc) and their job (e.g. department, salary, hours per week etc.), and a thousand four hundred and seventy (1470) rows, but for this assignment we made some adjustments (more information on this is included in the Dataset Preparation section).

We decided to use this dataset because, most importantly, the number of rows was satisfactory; the majority of the datasets we found on this topic contained about five to eight hundred rows (500 - 800), which would mean our training and testing sets would not have enough information for us to make a solid prediction, since the sample size would be too small. In addition, most datasets had so many missing values that we would either have to completely discard entire columns or we would risk introducing a lot of bias to our data when filling them in.

Goal

Our goal for this assignment is to predict how involved an employee is to their job (job involvement) by taking into account their age, how far they have to travel to get to work (distance from home), how much they like the space they are working in (environment satisfaction), their salary (monthly income) and how many years it has been since they were last promoted (years since last promotion).

This prediction could potentially help employers increase job involvement up to a point, at least as far as these factors are concerned. Using common sense, it is expected that the higher the salary and the environment satisfaction and the lower the distance from home and the amount of years since the employee's last promotion, the higher the job involvement would be. However, relationships like how a higher salary could counteract a high distance from home value, or how a very high environment satisfaction could potentially make up for a slightly lower salary are not immediately noticeable, and that is part of what we are trying to achieve with this project.

Dataset Preparation

Dataset Trimming

As mentioned in the introduction, our dataset contains thirty-five (35) attributes and a thousand four hundred and seventy (1470) rows. While the number of rows was within the acceptable range, we had to vastly decrease the number of attributes in order to be able to make a prediction.

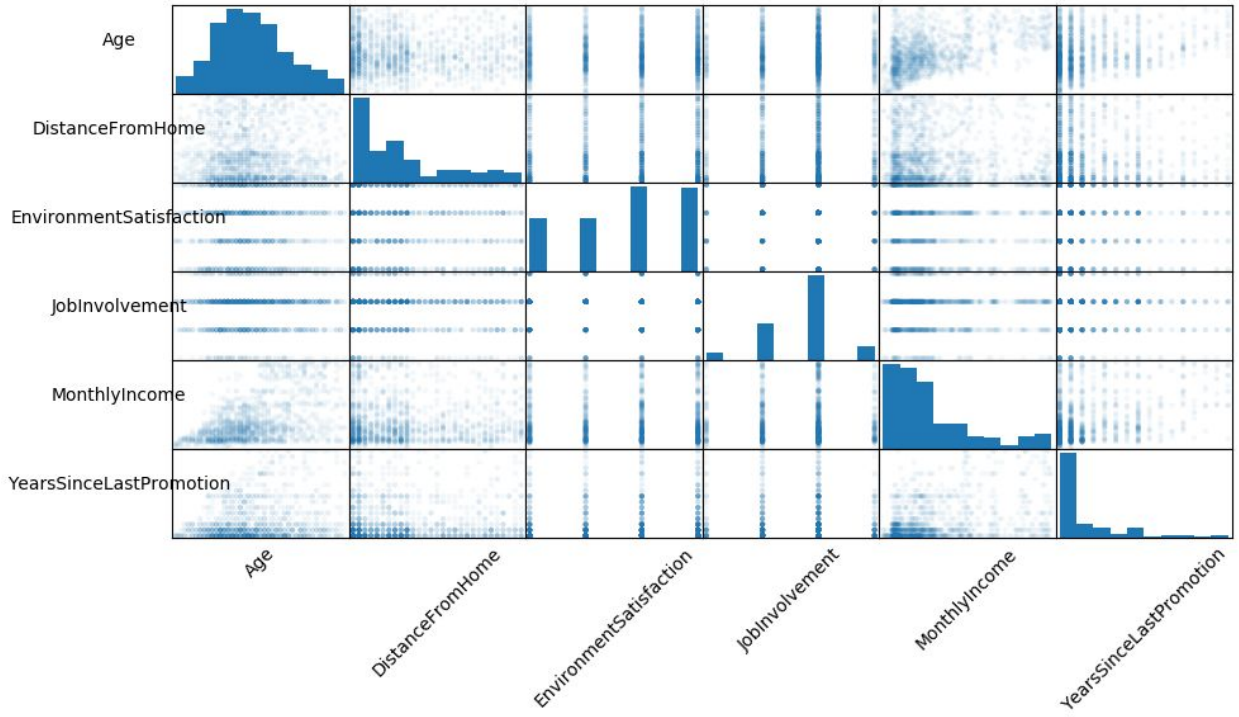
The first step was getting rid of redundant columns. For example, the original dataset contains both the employee's age and a column stating whether they are above eighteen (18) years old or not; which makes the later column redundant. Then we removed irrelevant attributes, like the employee number, the employee count and standard hours per week, since they offer no insight into what we are trying to predict (e.g. all employees had to work 80 hours a week and the employee number is a randomly generated number).

We had decided not to focus on the employees' personal lives, but instead focus on factors that the company could influence (e.g. if the employee lives too far away, offer support for relocation), so attributes such as marital status, or how happy the employee is in their relationship also had to go. In addition, we wanted to look at the employees as a whole, not according to what department they were in, or their gender, or what their field of study, so these attributes were removed as well.

This process helped us narrow down the attributes from thirty-five (35) to six (6): age, business travel, distance from home, environment satisfaction, monthly income and years since last promotion. Out of those, we decided to discard business travel, since the vast majority of the employees have stated that they either don't travel at all, or they do so rarely, which once again, does not offer us a lot of insight.
Our dataset now consists of 6 attributes and 1470 rows. The attributes are:

1. Age: age of the employee (in years)
2. Distance from Home: distance of the work from the employee's home (in kilometers)
3. Environment Satisfaction: how much the employee is satisfied working in that environment (range: 1 - 4) The 4 values represent Low, Medium, High, and Very High respectively.
4. Job Involvement: motivation, commitment, and involvement of the employee to their work (range: 1- 4 representing low, medium, high and very high respectively).
5. Monthly Income: the salary an employee receives every month from their work (in euros)
6. Years since last promotion: number of years the employee was last promoted

We also had to see if the attribute correlations were strong enough for us to use them; however the results were slightly worrying, since the correlations fell well below 0.1, which means that the attributes are not really dependent on one another. Therefore, we calculated the attribute correlations for the entire dataset, but the result was the same: all correlations are very weak (more information on this in the Insights section):



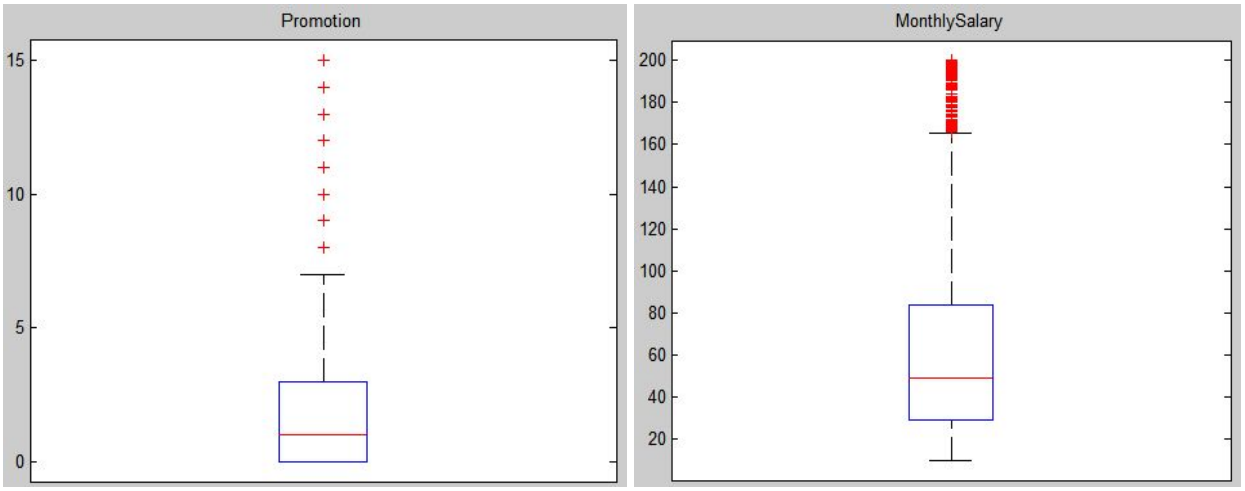
Data Cleaning

Once we finished trimming our dataset, it was time to start cleaning our data. This coincided well with our data cleaning workshop, since it helped us understand our dataset a lot better.

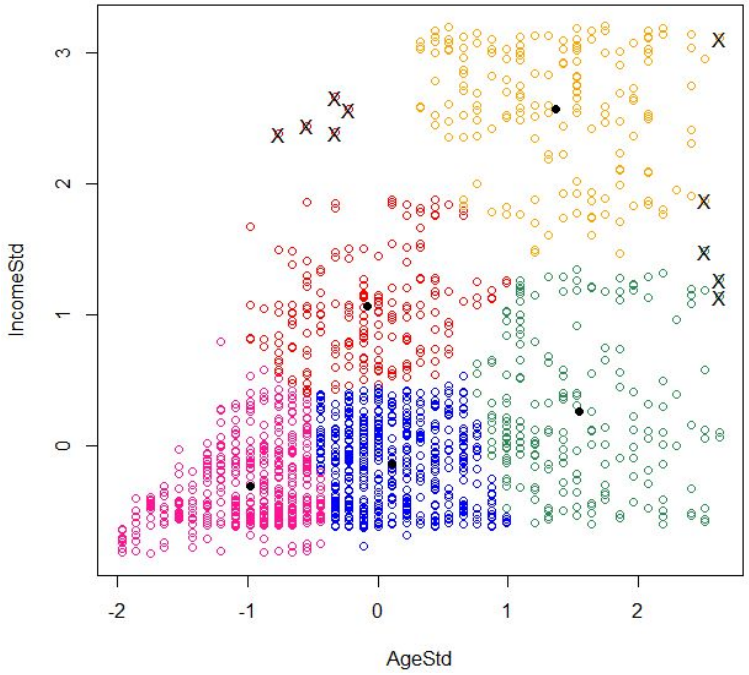
The first thing we tried to do is deal with any incomplete values, or values that needed to be adjusted (e.g. convert alphanumeric values to numbers etc.). This is the first time we realised how pre-processed our dataset was; it was mentioned during the lectures that most kaggle datasets have been pre-cleaned, but we weren't expecting that, not only there were no missing values, but no values needed adjustment or transformation either; for example, in the dataset description on Kaggle, it is stated that environment satisfaction can be low, medium, high or very high, but in the dataset that was already represented as 1, 2, 3 and 4 accordingly.

The next step was to deal with any noisy data, so to spot any random errors or variance in the measured variables and to apply smoothing techniques so it does not skew our data. The downside of this is that smoothing techniques are highly invasive, since you effectively change your data and we were wary of introducing bias to our dataset or losing important unique values. This was the reason why we decided to not apply binning to our data, so we opted for clustering instead, since this technique is very useful for highlighting potential outliers which can skew the data.

When we first plotted the data, we saw that some attributes contained extreme values:



Therefore, we were expecting quite a few outliers. The following graphs showcase an example of K-means clustering using the attributes Age and Monthly Income.



In this case, the data has been assigned to five different clusters and the values marked with an X symbol are the ones that are the furthest away from their cluster's centroid (this was calculated using the Euclidean Distance). $d(i, j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + \ldots + (x_{in} - x_{jn})^2}$

Although clustering highlighted some potential outliers in the attributes, we soon discovered that they did not actually skew our data; after removing them from the dataset and trying clustering again, the clusters did not change at all, which means that although those tuples' euclidean distance from their cluster's centroid is the highest, there is no reason to treat them as outliers and remove them.

It is also important to note that we had to do some data processing before we were able to successfully do clustering, as you can see from the scale of the graph and the names of the

axes,. The attribute age ranges from eighteen to sixty (18 - 60), so the maximum difference in age is 42. Monthly income on the other hand ranges from around a thousand to twenty thousand (1000 - 20000), so the maximum difference is about nineteen thousand (19000). We didn't take this fact into account when we first tried clustering, so the resulting clusters were effectively dividing the data into zones in a way that did not make sense.

The last thing we needed to identify in our dataset was inconsistent data, field overloading and unique, consecutive or null rules. This did not apply to our dataset after all; for example, all of our data was consistent (all our values are numeric and the ones that represent an alphanumeric attribute such as environment satisfaction and job involvement follow the same pattern: 1 = low, 2 = medium, 3 = high, 4 = very high).

In conclusion, our dataset was very clean and preprocessed. We did not need to remove any tuples, even after viewing the boxplots, performing clustering and identifying suspicious outliers. And our dataset was ready for the prediction algorithms.

Algorithms Used

When trying to make our prediction, we did three things; firstly, we had to decide whether to do classification or clustering. Since we have a known number of labeled classes, we decided to go with classification. Secondly, we split our dataset in 3 different ways: forty percent (40%) of the data went into the training dataset and sixty percent (60%) in the testing dataset, the reverse (60% vs 40%) and finally fifty - fifty (50% vs 50%). Since our dataset is on the smaller side, we wanted to see the effect the size variation of the training set would have on the final prediction. The last thing was to choose the algorithms to use to make our prediction; we narrowed down to the five following algorithms:

1. Naive Bayes Classifiers: the algorithm assumes that the data follow the Gaussian Distribution and it uses the formula $P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\,(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2})$
2. Decision Tree: this algorithm predicts the value of a target variable by learning simple decision rules inferred from the data features (supervised learning method).
3. Random Forest: the algorithm constructs multiple decision trees at training time and outputs the class that is the mode of the classes (ensemble learning method).
4. Extremely Randomized Forest: ExtraTrees follow the Random Forest pattern but additionally the top-down splitting in the tree learner is randomized (ensemble learning method).
5. Support Vector Machines: given labeled training data, the algorithm outputs an optimal hyperplane which categorizes new examples (supervised learning method).

The Naive Bayes Classifiers and the Decision Tree algorithms were introduced in class, so they seemed like a good starting point for us to start experimenting with different prediction algorithms and understanding why they produce the results they did. The Random Forest and the Extra Trees algorithms are an extension of the Decision Tree algorithm; a downside of using Decision Trees is that they tend to overfit to their training set, a downside which the Random Forest takes care of. Building on that, the Extra Trees algorithm reduces the variance of the model a bit more compared to the Random forest, at the expense of a slightly greater increase in bias. We decided to include both of these algorithms to give us a different perspective on the same approach. Finally, the Support Vector Machines was a supervised learning algorithm that we looked at while researching the above algorithms and, since it uses a completely different approach, we decided to try it and compare the results.

The language we used is Python and the algorithms were implemented using the Python Sklearn libraries. We also used pandas library to read the dataset and matplotlib to plot our data. This is a screenshot of part of our program:

```
import pandas
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.svm import SVC

# Read dataset
dataset = pandas.read_csv("DataMining.csv")
dataset = dataset.dropna()
columns = dataset.columns.tolist()
columns = [c for c in columns if c not in ["JobInvolvement"]]
target = "JobInvolvement"

irange = [0.4, 0.5, 0.6]

algorithms = ["Naive Bayes", "Decision Tree",
"Random Forest", "Extremely Randomized Forest", "SVM"]
accuracy = []
ratios = ["4:6", "5:5", "6:4"]

# Split dataset
for i in irange:
    train = dataset.sample(frac=i, random_state=1)
    test = dataset.loc[~dataset.index.isin(train.index)]

    #Algorithms
```

After importing the necessary libraries, we imported our dataset into the program and specified which columns to use for prediction and which attribute to predict. Then we specified how to split the dataset into the training and the testing datasets. We used a for loop that starts by splitting our dataset in the specified ratios, which are 4:6, 5:5, and 6:4; then we applied the algorithms.This is one of the algorithms we have used, Naive Bayes Classifier:

```
# Naive Bayes Classifier
model = GaussianNB()
model.fit(train[columns], train[target])
predictions = model.predict(test[columns])
accuracy.append(str(round(accuracy_score(test[target], predictions), 4) * 100))
```

After specifying the type of classification algorithm and the training dataset, we apply the algorithm to the testing dataset. The Naive Bayes classifier fits the training data into a predictive model which takes the rest of the data as a parameter and outputs the predicted values for the predicted column. The accuracy of the algorithm is calculated using the accuracy_score function, which compares the predicted column and the actual column. This function computes subset accuracy, which is the set of labels predicted for a sample that exactly matched the corresponding set of the true actual labels. The output of these function are from 0 to 1, so we rounded them to 4 decimal places and multiplied them by 100 to get percentages that we can compare.The rest of the algorithms were implemented using a similar pattern, and using the same accuracy function so that our conclusions are not biased.

Experiments & Insights

Experiments

Algorithm Application

The first experiment we did was to simply apply the chosen algorithms to the different size variations of the training and testing datasets, while also trying a different number of iterations when applying the five algorithms three times using three different ratios. We expected that the accuracy of the Naive Bayes Classifiers to be the highest and the accuracy of Decision Trees to be the lowest since the attribute correlation is very low.

These are the accuracy results produced by our prediction algorithms:

```
    ['Naive Bayes', 'Decision Tree', 'Random Forest', 'ExtraTrees', 'SVM']
4:6 ['58.05',           '41.27',        '53.29',        '52.38',    '58.05']
5:5 ['58.37',           '40.54',        '53.06',        '51.84',    '58.37']
6:4 ['57.99',           '41.5',         '54.42',        '54.25',    '58.67']
```

The first thing we noticed is that our predicting accuracy is quite low, with the highest one being close to fifty-nine percent (59%); however we were expecting this, since our attribute correlation was so low:

	Age	DistanceFromHome	EnvironmentSatisfaction	JobInvolvement	MonthlyIncome	YearsSinceLastPromotion
Age	1	-0.00168612	0.010146428	0.029819959	0.497854567	0.216513368
DistanceFromHome	-0.00168612	1	-0.016075327	0.00878328	-0.017014445	0.010028836
EnvironmentSatisfaction	0.010146428	-0.016075327	1	-0.008277598	-0.006259088	0.016193606
JobInvolvement	0.029819959	0.00878328	-0.008277598	1	-0.015271491	-0.024184292
MonthlyIncome	0.497854567	-0.017014445	-0.006259088	-0.015271491	1	0.344977638
YearsSinceLastPromotion	0.216513368	0.010028836	0.016193606	-0.024184292	0.344977638	1

As you can see, the Naive Bayes Classifier algorithm had the highest prediction accuracy, along with SVM for the 4:6 and 5:5 dataset division ratios, however for 6:4, SVM scored the highest prediction accuracy we got overall (58.67%). The simple Decision Tree algorithm had the poorest performance, scoring below forty-two percent (42%) accuracy. Looking at the overall results, the predicting accuracy of our algorithms is very low which, even though it is not unexpected because of the very low attribute correlation, means that we are far from our target goal.

Binning

When cleaning our dataset, we avoided using Binning, since the number of distinct values per attribute is lowered significantly, but we decided to give it a try and re-apply the algorithms mentioned above in hopes of improving our prediction accuracy. We used Binning on three of our attributes: Age, Distance from Home and Monthly Income:

Age	DistanceFromHome	MonthlyIncome		AgeBinning	DistanceBinning	IncomeBinning
35	18	9069		40	20	10000
59	25	7637		60	30	8000
55	8	14756		60	10	15000
45	7	9724		50	10	10000

We run the algorithms again and the result was

```
    ['Naive Bayes', 'Decision Tree', 'Random Forest', 'ExtraTrees', 'SVM']
4:6 ['58.05',           '40.48',        '52.38',        '52.04',    '58.05']
5:5 ['58.37',           '42.45',        '52.93',        '51.7',     '58.5']
6:4 ['57.99',           '43.03',        '54.59',        '52.55',    '58.84']
```

The prediction accuracy increased by a very small amount; the Naive Bayes Classifiers and the SVM algorithms tie in terms of accuracy at the 4:6 dataset ratio split, but SVM achieved the highest accuracy for the 5:5 and 6:4 splits and the highest accuracy in general with 58.84%. The Decision Tree algorithm has once again the lowest accuracy.

Change Attributes

The last thing we tried was to try and predict a different attribute; while we did not expect different results considering all attribute correlations were low, we decided it was worth a try. We applied the same algorithms to four other attributes, and as expected, our results didn't improve; on the contrary, our predicting accuracy was much lower:

```
    ['Naive Bayes', 'Decision Tree', 'Random Forest', 'ExtraTrees', 'SVM']
4:6 ['28.68',           '27.32',        '28.68',        '27.66',    '30.5']
5:5 ['27.76',           '25.71',        '27.21',        '26.67',    '30.07']
6:4 ['27.21',           '25.0',         '27.55',        '26.19',    '30.44']
```

This is our prediction accuracy for the attribute Environment Satisfaction. It follows a similar pattern as our previous attempts, with the SVM algorithm had the highest accuracy for all dataset ration splits, with the maximum being 30.44% and the Decision Tree algorithm performing the poorest.

Insights

Low Attribute Correlation

We were correct to worry over our very low attribute correlation. When we chose this dataset, we saw it had no missing or N/A values, that there was a significant number of attributes to choose from and that the number of rows was higher than most of the datasets we had look at up until then. On Kaggle, the dataset also came with a small description of each of the attributes, which helped us get a better understanding of the data; therefore we wrongly assumed that it was a good dataset. However, while going through the workshop material and applying it to our dataset, we soon realised how unrelated the attributes were; not only the ones we chose, but all of the in general, as can be seen in the example below:

	DistanceFromHome	Education	EmployeeNumber	EnvironmentSatisfaction	JobInvolvement
DistanceFromHome	1	0.0210418256	0.0329164072	-0.016075327	0.0087832799
Education	0.0210418256	1	0.042070093	-0.0271283133	0.0424376343
EmployeeNumber	0.0329164072	0.042070093	1	0.0176208025	-0.006887923
EnvironmentSatisfaction	-0.016075327	-0.0271283133	0.0176208025	1	-0.0082775982
JobInvolvement	0.0087832799	0.0424376343	-0.006887923	-0.0082775982	1

Low attribute correlation means that there is no real relationship between our data, which means that we can not strongly rely on one or more attributes to predict another. This has a

big impact in our prediction accuracy, which is emphasised in our case by our highest accuracy score being 58.84%. Trying to predict different attributes did not make any difference; if anything our prediction accuracy became much worse, which is to be expected considering that all attribute correlations are so low. Unfortunately there is nothing we can do to change this.

Data Cleaning

Our dataset was highly processed to begin with, so there was very little to be done in terms of cleaning: no missing values, no inconsistent data, no field overloading, even the tuples marked as potential classifiers were well within the accepted range. The most extreme values were found in the Monthly Income attribute, however we could not remove these from our dataset, as we would lose a very significant part of it. Since the dataset was that clean, we did not expect Binning to have much of an effect on the prediction accuracy and we were right; the increase in prediction accuracy was insignificant, since the highest accuracy was 58.84% instead of 58.67%.

To conclude, we believe we did not choose a good dataset. The number of rows is on the lower side, which results in the learning dataset being fairly small. In addition, without a high correlation between attributes, whether it is positive or negative, it is not possible to make a good prediction with high accuracy. We did however learn a lot about our data, therefore we were not surprised by our results and we were able to explain them, which was very rewarding.