Machine learning techniques towards working condition analysis

Joan Samper, Neil de la Fuente, Daniel Vidal

***Abstract*— During these pages, you will be guided through a path that reports our experience and learning outcomes during the final machine learning subject project. For this project, we decided to make an analysis and exposure of machine learning techniques applied to the analysis of job conditions. We will also show some models that can help us in the tough task of maintaining employees happy.**

I. INTRODUCTION

For this project, we have worked with a database of different employees, in various fields and working conditions. The data is distributed in such a way that each row corresponds to a different worker and each column to a characteristic of him/her. This datum has been chosen because there are many observations, that differ a lot from each other as well as for its variable quality and usefulness. Features such as attrition, salary hike, and so on are not usually seen in this kind of dataset, yet are so enriching.

For starting to work on the database, some questions and objectives have been established, with their possible and corresponding machine-learning techniques. These are the main methods we will use and the questions we will try to answer:

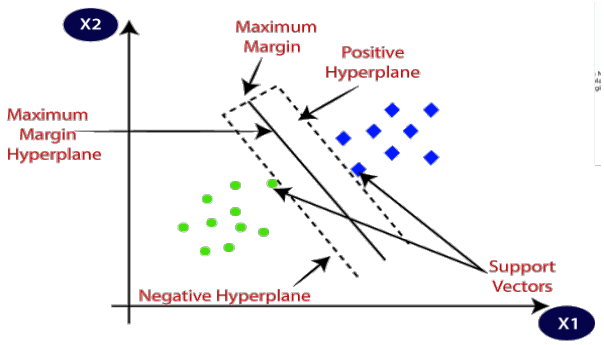
1. SVM: Can we classify if a worker will be suffering attrition or not?
2. DECISION TREES: What work level should I expect in the future? Are there some critical features for this prediction?
3. RANDOM FOREST REGRESSION: Is it possible to predict the Monthly Income of an employee?

Before working on the database, those values that are not useful or easy to work with have been cleaned. For instance, all the variables that did not contribute anything to any of the objectives planned for the project have been deleted from the original data frame, some examples of these unuseful variables are Employee number, which was just the employees' ID or employee count, which had always value 1. Once we realized that there were not any missing values (NaN) in the updated data, we got our data clean and ready for modeling.

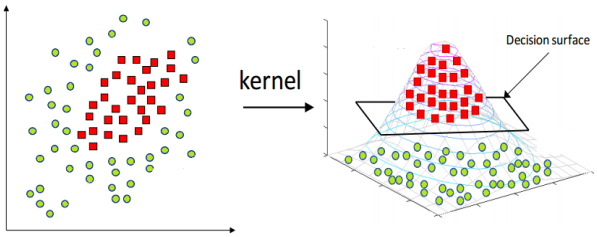
II. SVM: CAN WE CLASSIFY IF A WORKER WILL BE SUFFERING ATTRITION OR NOT?

Support Vector Machines (SVM) are a powerful tool for classification. The idea of SVMs is quite simple: The algorithm creates a hyperplane that separates the data into classes (In our case, two classes: 'YES' (suffers attrition) and 'NO' (Is happy with his/her job). The good thing about SVM is that gets the best dividing hyperplane, as it is capable to find the points closest to the line from both classes. These points are called support vectors. The distance between the support vectors and the hyperplane is computed. This distance is the maximum margin, and our goal is to maximize it.

Here you have a graphical representation of how an SVM linear classification would look like:



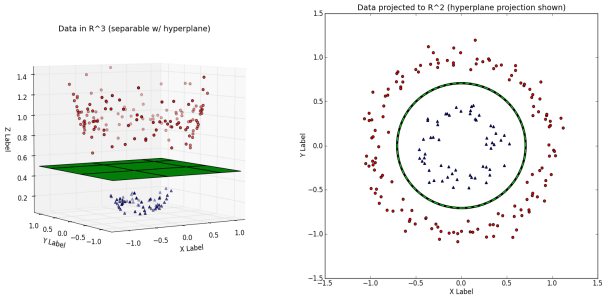
It is similar for not linearly separable data; We obviously can not draw a straight line to separate the classes, but what we can do is to convert it to a higher dimension so it can be separated by a hyperplane:



Now data is clearly linearly separable on Z-axis.

Thus, we can classify the data by increasing its dimensionality until it is linearly separable and then project the decision boundary back to the 2D plane using mathematical

transformation. We would get something like this:



But finding the optimal transformation for a given dataset can be difficult as well as computationally inefficient when we have many features. For instance, in our dataset, we have +30 features, and it would be so impractical to do these transformations as loads of polynomial combinations would be required.

Luckily the SVM proposes that is not necessary to do this transformation explicitly. Just defining a similarity function that computes the dot product in the new space should be enough, this is known as a kernel.

Kernel Trick

With the kernel trick, instead of doing the dot product on the original vectors, we map the data points to a new space and we do the dot product in this new space, to do this we could use a mapping function that raises our features to different powers. These dot products are a very easy calculation and are computationally cheap. This would be just the representation of how dot products would be calculated in that space, we don't have to actually do it, and that is the reason why it is called a "trick".

In conclusion, we could say that we can find kernels, which are functions of calculated products in high-dimensional spaces, without explicitly going to these high-dimensional spaces. That high dimensional dot product will correspond to taking our original vectors, mapping them into a new space of dimensionality n, and doing a dot product between those huge vectors. Both results would be actually the same, but the way we do it with the kernel trick is much simpler, with a very simple formula:

$$K(x, z) = (x^T z)^n$$

Furthermore, we can get results of a very high dimension, even mapping our data points to infinite dimensions if we choose our kernel right: Mercers Theorem.

This is super effective and useful in our project, where we have many features and a decision boundary could only be drawn in a high dimensional space.

There are many different kernels that we can use, we tried several, such as rbf (Gaussian), polynomial or linear kernels. Finally, we decided to go for a linear kernel as it gave the best results, is the fastest to compute, and is also the simplest to understand:

$$K(x, z) = (x^T z)$$

With this linear kernel and a couple of tuning hyperparameters: C and γ, were defined by grid search.

C tuning parameter: C controls the trade-off between a smooth decision boundary and the correct classification of training points. A large value of c means you will get more training points correctly but overfitting would end up appearing if C is too high. After applying grid search we got that the best C value for us would be: 0.1

γ ***tuning parameter:*** Gamma defines how far the influence of a single training example reaches. If it is low: every point has a far reach. If it is high: every point has a close reach. If γ is too high: the decision boundary will be dependent upon the points that are very close to the line which effectively, ignoring the points further from the decision boundary because the closer points get heavier weight. On the other hand, if the gamma value is low even the far-away points get considerable weight and we get a more linear curve. After applying grid-search, we decided to use γ: 0.001

Finally, we fitted the SVM model with the according hyperparameters to our data and predicted the 'Attrition' value, which was a binary classification between 'Yes' and 'No'.

Thoughts on results of SVM

The result we got was outstanding and valuable: 89 % of accuracy using 294 support vectors. We have to take into account that we had many more 'No attrition' data points than attrition data points, thus there were approximately 5 'No' support vectors for each 'Yes' support vector. This explains why values for precision, recall, and f-score are higher for the 'No' classification. The whole sum of metrics makes this model quite reliable and useful for our purpose

Practical demonstration: Dimos and Ana

We wanted to do a practical demonstration of how our model classifies so readers can understand the results, as well as why this attrition classification is useful and valuable for workers and companies.

To make this possible the following two new instances were generated. These instances were very polarized in every feature for an easier understanding of the outcome.

Dimos:

- 30 years.
- High education.
- $19206 monthly income (much higher than the mean).
- 30 % of salary hike.
- Research and Development field of work.
- No overtime.
- ...

Ana:

- 57 years.
- Low education.
- $2700 monthly income (much lower than the mean).
- -10 % of salary hike.
- Human Resources field of work.
- High overtime.
- ...

Our classifier aimed to discover if Dimos and Ana will be attrited or not. Our classification model told us that Dimos would not be suffering attrition and that Ana would be willing to quit her job as she is actually attrited.

This example is quite obvious but easy to understand. The worst the job conditions of workers are, the more attrition the company will suffer, as the workers will end up quitting. On the other hand, good job conditions will make workers happier in their jobs and push them towards better performance for their company.

III. DECISION TREES: WHAT WORK LEVEL SHOULD I EXPECT IN THE FUTURE?

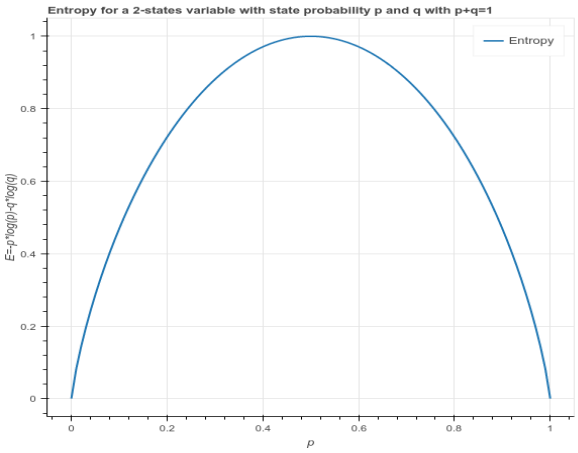
Decision trees are a very powerful tool of machine learning used for classification, specifically a supervised classifier. It takes a collection of features as input and returns a single output that is used as a prediction. One of its main advantages is the lightness of the model in relation to performance. The learning of this type of model is achieved by finding a function that can be seen as a hypothesis that is consistent with the training data. These hypotheses are combinations of the training features, but there are so many possible combinations, specifically 2^{2^n} combinations, so what decision trees give us is a smart algorithm to choose the best combination quickly. This algorithm is mainly implemented in a recursive way, expanding a binary tree and searching through specific branches based on some criterion.

***splitter parameter**:* The first criterion we have to define is which branch should the algorithm choose while expanding. For this purpose, we can choose different options, but the ones implemented in sklearn and used in this paper are the greedy algorithm and the random search. In the greedy search method, we always expand the most promising branch, so that one with the best heuristic or the one that gives more information. This greedy search is implemented in skleran setting the parameter "splitter.ºf the decision tree creation method to best. The other option is to set the "splitter"parameter to random", where the branch is randomly selected but pondering with more probability the branches with better heuristics or more information, so the best branches are not mandatory to be chosen, but they are more likely.

***Criterion parameter**:* These described decision methods are a very important component of the classifier, but now the question is how to determine which branches give more information, in other words, to determine the heuristic values of every node.

To do this in sklearn the parameter criterion is used. This parameter can be set with the values "ginni", 'entropy' or "log loss". The 'entropy' parameter determines the heuristics by seeing which features give more information to the classifier, obtaining as much information as possible in the first branch splittings. This information amount is calculated with the following formula:

$$\sum_{i=0}^{n} -p_i log_2(p_i)$$



In the plot, we can see that the information gain will be maximum when we have a 50 % of a chance of every class we want to split, and a 0 information gain if we have a 100 % chance of 1 class, or what is the same, all the samples to split are from the same class.

The 'gini' criterion uses almost the same principle but with a different formula that is the following:

$$\sum_{i=0}^{n} 1 - p_i^n$$

If the gini function is plotted with the entropy function can be seen that gini is almost the same as the entropy function, but it can only achieve a maximum value of 0.5.

The advantage of using this function is that it has faster training than the entropy method, but it changes the performance of the classifier.

The last parameter that we can use for the criterion is the log loss, which is the following function:

$$L(y,p) = -(ylog(p) + (1-y)log(1-p)$$

Being $y = 1, 0$ the label of the predicted class and being $p = P(y = 1)$ probability of y being from class 1

This method returns a 0 or a 1 as output, so it can not be used with the random 'splitter', because it will always perform the best search. This behavior obviously affects the performance of the classifier.

Maximum depth:: Finally, we have the maximum depth parameter. This parameter basically describes how much depth the decision tree will take, limiting the number of expanded nodes, and stopping the model before over-fitting. This is a very important parameter that will affect a lot in the performance of our model and choosing it right is crucial.

Model training and analysis

Now that the basic concepts of decision trees have been explained some data prediction, questions, and data insights can be done with these classifiers.

With this section, a very interesting question is pretended to be answered: Which job level should be expected in the future?

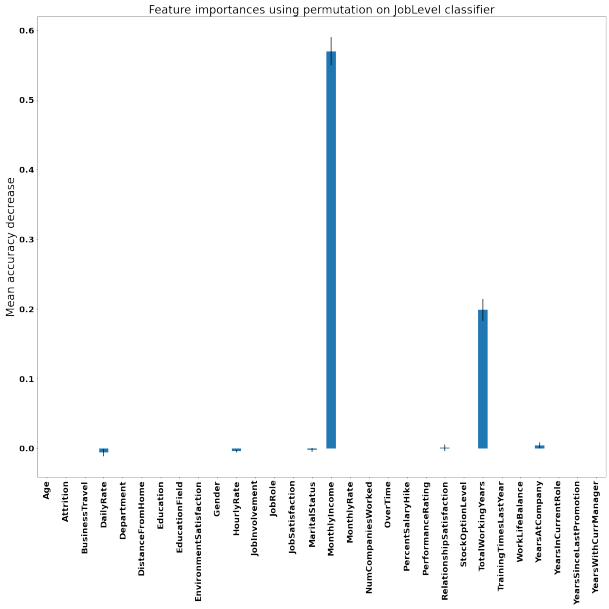
In order to answer this question a decision tree will be used, using the feature of the job level as output. This feature have levels from 1 to 5, being 1 the worst job level and 5 the best job level, finally we will conclude which features are the most decisive at the time of determining the job level of a worker. For the creation and training of decision trees purpose we have used the sklearn module. The first step is to create and train a tree. For the first predictions, all the features of our initial dataset except for the job level will be used. In order to determine the best hyperparameters for the model a grid search is used with the following parameters to try.

1. criteria: Gini, entropy, log loss
2. splitter: best, random
3. max depth(depth of the tree): 2,5,10,15,20,25

The best result of the job-level decision tree grid search with all the initial features:

1. criterion: entropy
2. splitter: best
3. max depth: 20
4. accuracy: 0.92

Now it's interesting to analyze the most decisive features in this tree. As the tree's direct representation is difficult to visualize because we have so many branches, the weight of the parameters in the model is visualized in the next graphic:



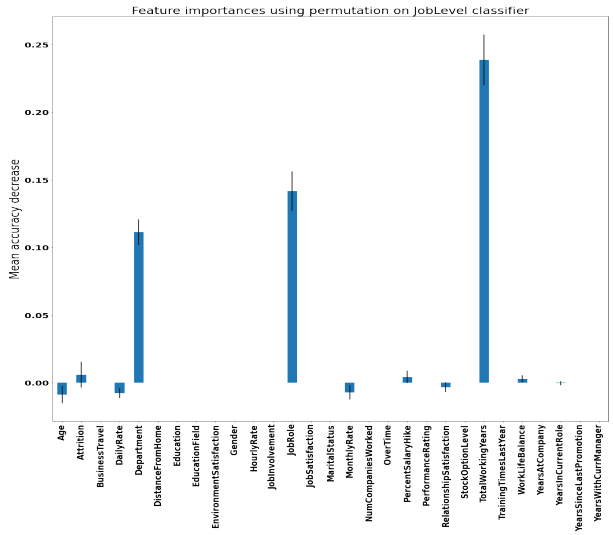
In the last barplot can clearly be seen that the 2 most decisive features to determine the job level of an employee are his monthly income in the first place and the total working years of that individual. These are 2 very reasonable features that clearly determine the job level because as higher the income higher is your job level and as much years you spend working on a job most experience you have and the higher the job level you acquire.

As the monthly income of a person is very difficult to decide and to choose because normally is an imposed amount for a certain job, can be interesting to see which parameters are the ones that influence the job level of a person if the

monthly income of it is not taken into account. In order to do what has been explained, another decision tree is trained but now without the feature of the monthly income. The best hyperparameters of the model are determined with a grid search, obtaining:

1. criterion: gini
2. splitter: best
3. max depth: 5
4. accuracy: 0.7

Clearly, as the monthly income feature is not being used the accuracy of the model falls significantly. Despite that, now the new more decisive features can be analyzed through the next graphic:



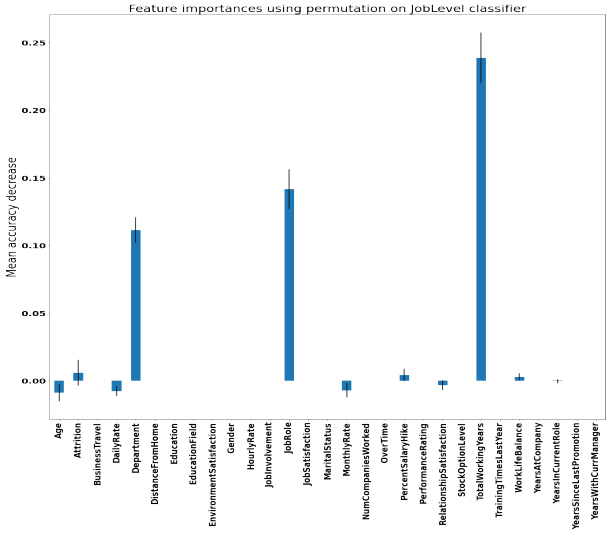
From this graphic is observed that the features that mainly determine the job level that a person achieves are the department where this person is working, the job role that is assumed, and obviously the total working years of that person working, a feature that has already been determined by the first decision tree model. All these features make a lot of sense because is logical to assume that some departments depending on their importance in a company, pay more than others. For example, the informatics department of a company is expected to have a higher job level and a higher salary than the cleaning department of that company. With the job, the role is obviously related to the job level, and the same already exposed example of the informatics and cleaning departments can be done but with the personal role of every individual of the company.

Another important feature of a company is the attrition level of the employees. Determining the factors that cause attrition can be very useful for a boss in order to have a happier staff, increase productivity or determine job incentives without monetary compensation. In this case the output will be the attrition level, that is a binary feature, being 0 a person that does not have work attrition and 1 a person that suffer it.

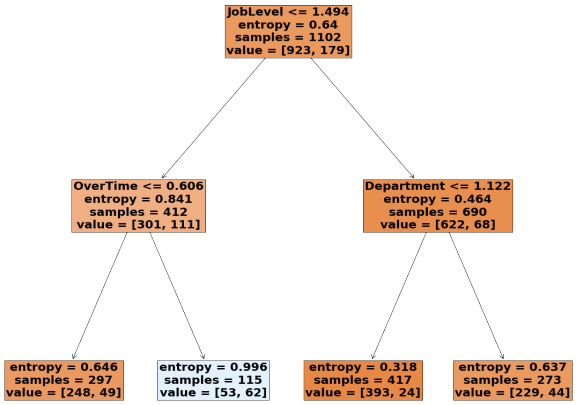
In order to predict the tree again a tree will be trained with all the parameters performing a grid search to find the right hyperparameters to achieve the best performance, obtaining:

1. criterion: entropy
2. splitter: random
3. max depth: 2
4. accuracy: 0.84

Again the determining factors of the attrition are analyzed with a bar plot.



As this tree only needs a tree depth of 2, it can be easily visualized in the next plot.



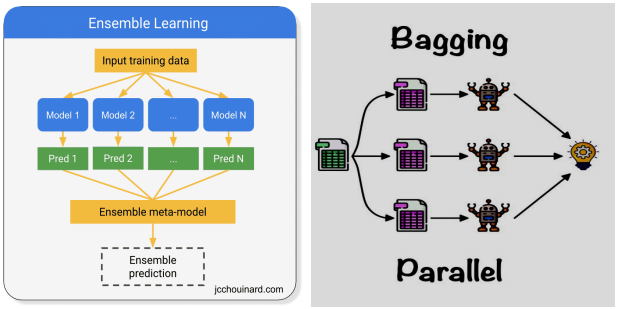
From these plots can be inferred that the main factors that explain attrition are over time and job level, and in the tree, the department label is observed, but as it has been already seen is a redundant feature because of the appearance of the job level. The obtained features make sense because a high responsibility job level presumably will lead to more attrition, and overtime hours in your job that reduce your

free time also will lead to attrition.

With the decision trees what can be seen is that more important than what the classification model predicts is that very valuable insights into the predicted feature can be extracted from the model its decisions and its performance with certain features. Also, the lightness of the model is a absolutely important feature, letting a very extensive search with lots of parameters in a very short time.

IV. RANDOM FOREST REGRESSION: WHAT MONTHLY INCOME WOULD EARN AN EMPLOYEE?

Random Forest is a power Machine Learning algorithm useful to do Classification models and Regression models. This method is based on Ensemble Learning (fig. a), which consists in executing multiple simple algorithms or models and combining their results with an optimal predictive model. This method is a bagging algorithm (fig. b), which consists in executing all models in a parallel way and taking an average of their outputs as the result. In this case, the Random forest generates 100 decision trees (default value) and we will use it for regression so we can predict continuous values of the Monthly Income.



(a) (b)

Fig. 1: Ensemble learning and Bagging

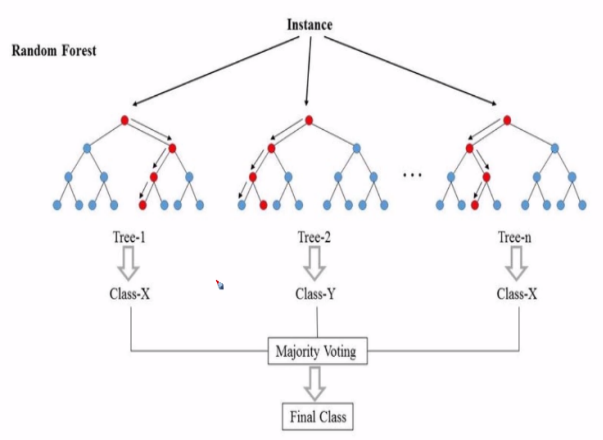
Our objective is to predict the Monthly Income of the employee, discover what are the variables that affect the Monthly Income, analyze the results and realize if this is a good model for our data set according to the prediction error.

But, why did we use Random Forest? Because it is useful to deal with outliers as it handles them automatically, the model is able to fit well with non-scaled data as it uses a rule-based approach instead of distance calculation and it is less impacted by noise.

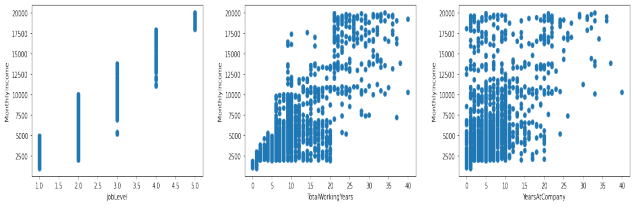
Random forest is very stable, which means that it is not affected much if new data is introduced in the dataset since the new data should affect only one tree.

It handles non-linear parameters efficiently. Non-linear parameters don't affect the performance of a Random Forest, unlike curve-based algorithms. So, if there is high non-linearity between the independent variables, Random Forest may outperform as compared to other curve-based algorithms.

We have also tried to do Linear and Polynomial Regression but we obtained bad results and some problems. The predictions obtained with new data points were absolutely wrong, giving negative and very large values, also we had a problem with the polynomial features which was that the computer was unable to fit the model with degree 5 because it required a huge computation of many many features so the program crashed. The first problem may be because of the outliers and the second problem was because the dataset already contained a lot of features so when we try to convert them to polynomial features we obtain 30^n features, which is computationally expensive.



We did some visualizations to see which variables were more correlated with the Monthly Income variable and the most correlated was the (a)JobLevel, (b)TotalWorkingYears and (c)YerasAtCompany; we also discovered that Manager and Research Director jobs are the best-paid ones.



(a) (b) (c)

Fig. 2: scatter plots between Monthly income and some variables

To start with our Regression model, first, we initialized our 'X' data and our 'y' target value, so the 'X' data will be our data points excluding the Monthly income which will be our 'y' target value. We will split our data into 80 % train data

and 20 % test data so we can test the predictions obtained by our regression model.

Then, we created our forest regressor model from sklearn.ensemble, we fit it into our training data and we ask our model to predict some Monthly incomes with the test data points and compare the results with the test real results.

Most of the predictions that we got were very similar to the original value, but there were some predictions that had a high error, for example, a wrong prediction can have a difference of 3000$ which is a very bad prediction since the mean of the monthly income is 6502$, but there were also predictions with an error lower than 30$ (best error was 8$). The mean error was 1400$ for the test data but when we tried using new data points created by ourselves using values that make sense, that is to say, values that are possible and can be perfect from a real person, we obtained very accurate results, in other words, for values with high performance, high job level, many years working and many things that lead to a good salary we obtain very high Monthly Income and the opposite for values that lead to a bad salary. We thought that The high errors in the predictions in the test may be due to the values in their data points that do not make so much sense or they are persons with special cases and some data that is not taken into account.

We can conclude that the Random Forest Regression is a good method to predict continuous values when there are outliers in the data and your data is not scaled, but it is more computationally demanded. We were able to predict very well the Monthly income with new data and with most of the training data, but if the dataset contains data points that have contradictory values or are special cases (outliers), the predictions can have a very high error.

V. CONCLUSIONS

As a summary of the results of the goals and objectives, we can proudly state that we successfully managed to realize with a good ending all the required techniques and methods.

1. For the first objective, we derive that the SVM is a so powerful tool when it comes to making classifications in datasets with many features, as we can make use of the kernel trick, which uses simple dot products that can make our lives much easier.

   Being capable of understanding the real way that Support Vector Machines and Kernels work is so valuable. In this case, any company could make use of it to see if their employees are willing to quit due to attrition and then search for ways to prevent it.

2. For the second objective, we decided to use decision trees because these are very light and powerful models, that do a good job in the classification task and are very analyzable, so we can take very clear conclusions from the trained model.

   We have seen that the job level and attrition of a person are very predictable outputs with few features. This can be seen thanks to the fact that this type of model gives us a very detailed view of the features that influence more the classifier. In this sense, the decision trees are a white box from which you can get very valuable conclusions and insights about data. This last factor is added to the fact that is a very light model so we can train hundreds of models with different hyperparameters in order to find the ones that perform better with a grid search very fast.

3. For the third objective, we can conclude that we can predict continuous values with regression models. Polynomial regression and linear regression were not so useful for this dataset. But, Random Forest Regression was able to do good predictions using non-scaled data controlling the outliers and the noise. Some of the predictions had a high error but we thought that this is due to data points that are outliers or do not make much sense.

   We obtained very good results predicting new data points and Monthly income so we conclude that this method is very useful for multiple cases of regression due to its multiple advantages of flexibility with the data the only disadvantage it has is that it requires a lot of computations to generate and use many trees.

ACKNOWLEDGEMENTS

We are so grateful to all those who accompanied us in the development of this project, that dedicated a few minutes (or hours) to review the code and help in any way.

Special thanks to Dr. Dimosthenis Karatzas and Dr. Oguz Mulayim, who taught us the techniques used in this project and helped us understand in detail how these models really work. Teaching is sometimes underestimated and we actually want to thank these professors for their big and successful effort.

REFERENCES

[1] SVM: A simple way
[2] Useful Data and Notebooks
[3] hyperparameter tuning using grid-search
[4] Random Forest Algorithm
[5] Bagging and Boosting algorithms
[6] Sklearn decision trees documentation
[7] Deeper explanation about decision trees parameters
[8] Documentation of inspection permutation for decision trees
[9] Kernel Trick in Support Vector Classification
[10] Sklearn.ensemble Random Forest Regressor