FACULDADE DE ECONOMIA UNIVERSIDADE DO PORTO

EMPLOYEE SATISFACTION

A CLASSIFICATION PROBLEM

BUSINESS ANALYTICS

Beatriz Almeida I 202202757
Carolina Resende I 201905137
Joana Abreu I 202202709
Patrícia Silva I 202202895

JUNE 2023

Table of Contents

Introduction

The present report aims to cover the talking point regarding employees' satisfaction through a Business Analytics approach, using Python as a working tool.

The worldwide business environment is quickly changing, and organizations that are able to adapt will thrive. As a result, firms must devise strategies for facing the fierce competition. One of the most difficult difficulties that organizations face today is understanding how to manage workforce churn. This could be attributed to a lack of enthusiasm and commitment to the organization, emphasizing the importance of job satisfaction (Singh & Tiwari, 2011).
Job satisfaction has a wide range of effects on various aspects of organizational life. Some of them are related to the influence of job satisfaction on employee productivity, loyalty and absenteeism. Furthermore, study evidence suggests that while satisfaction does not always lead to improved individual performance, it does lead to improved departmental and organizational success (Aziri, 2011).
In addition, several factors influence job happiness, including the nature of the work, compensation, advancement opportunities, workgroups, and working conditions. Therefore, a manager's concern for people impacts the job design (scope, depth, interest, perceived value), compensation (external and internal consistency), working environment, social interactions, perceived long-term opportunities, and levels of ambition (Aziri, 2011), regarding employees.

The Problem at Hand

Job satisfaction is one of the most challenging concerns that today's managers face when it comes to managing their personnel (Aziri, 2011). Hence, it was decided to cover the mentioned issue, using a set of data regarding employee satisfaction related to the diverse variables that influence it. Therefore, it is relevant to critically evaluate and understand the obtained dataset, as well as apply several classification problem models, using Python in order to analyse employee satisfaction management problem and, thus their predictive performance.

The Dataset

The dataset used is purely fictional thus, it does not represent any organization. Moreover, it was created in order to help the data analysts play around with the trends and insights on the employee job satisfaction problem. Therefore, the mentioned dataset has the following attributes:

Table 1 – Attributes of the dataset, their description, type, and classes (categorical)

Predictor	Description	Type	Class
emp_id (Excluded)	Unique ID of the employee		
age	Age of the employee	Numerical	
Dept	Department to which the employee belongs	Categorical	HR Marketing Purchasing Sales Technology
location	Employee location	Categorical	City and Suburb
education	Employee's education status	Categorical	PG and UG
recruitment_type	Mode of recruitment to which the employee was subjected	Categorical	On-Campus Recruitment Agency Referral Walk-in
job_level	The job level of the employee: 1 being the least and 5 being the highest position	Categorical	1 to 5
rating	The previous year's rating of the employee: 1 being the least and 5 being the highest score	Categorical	1 to 5
onsite	Has the employee ever gone to an onsite location?	Categorical	Binary (0 and 1)
awards	Number of awards received by the employee	Numerical	
certifications	Is the employee certified?	Categorical	Binary (0 and 1)
salary	Net Salary of the employee	Numerical	
satisfied **(Outcome Variable)**	Is the employee satisfied with his job?	Categorical	Binary (0 and 1)

Model Utilization

Graphical Data Analysis

In order to explore the dataset in a visual and comprehensive way and to extract relevant information, plots were employed so that the variables could be linked not only to each other but also to the outcome variable, the employees' satisfaction.

Hence, it may be concluded that there are more satisfied employees (these being more than 250 individuals) than dissatisfied ones through the analysis of the histogram of the outcome variable, as it is shown in Exhibit 1.

Via pair plot, as can be observed in Exhibit 2, regarding the *job level*, the majority of satisfied employees find themselves in the higher categories, while the greater part of the unfulfilled ones belongs to the 1st level. Concerning the *recruitment type*, among the pleased employees, the prevailing number was recruited "On Campus".

Through the correlation table and matrix, as Exhibit 3 and Exhibit 4 illustrate, it is possible to see that, of all the variables studied, *rating* is the one that is most correlated with employees' satisfaction. Also, *job level* and *salary* are highly correlated in an almost perfect positive correlation, suggesting that one of them should be removed in order to avoid the multicollinearity problem.

In relation to this, we decided to, first, analyze the models with all the variables and then, analyze all models again in two different scenarios: one excluding the *job level* variable and the other excluding the *salary*. This will allow us to conclude which models have the best predictive performance and, consequently, if the variables analyzed are appropriate to address the problem of employee satisfaction.

Preprocessing of the data

Firstly, we started by analyzing whether our dataset had missing values (NA) because if it did, we would have to implement a strategy in order to solve this problem before estimating the models. Fortunately, our dataset doesn't have any missing values. After that, we also checked the distribution of observations by classes, that is, we checked whether our dataset was unbalanced. In this case, we see that it is slightly unbalanced with 53% of instances belonging to class 1 (satisfied employees) and only 47% belonging to the negative class (dissatisfied employees). The fact that the majority class is the one of interest almost eliminates this problem, but even so, when estimating the various models, we tried to answer this question in order to obtain the best possible performances.

Regarding the *preprocessor*, we already specified the type of each variable analyzed. So, for the numerical ones we applied the *StandardScaler* transformation, and, in the case of the categorical variables, we enforce the *OneHotEncoder* transformation dropping the first category for each feature when encoding in order to avoid multicollinearity.

Models - Classification Problem

Since our outcome variable is categorical, assuming the value 1 for satisfied employees and 0 otherwise we should the models that work with classification problems such as K-Nearest

Neighbors (KNN), Naive Bayes, Trees, Random Forests, Logistic Regression, Neural Networks, Support Vector Machine (SVM) and XGBoost.

First, we divided the dataset into two different data frames: one with only the outcome variable and the other with only the predictors. Then, we split again into training and test sets, respectively, 80 e 20%. The training set was used to fit the model and the test set was used to see how each model works with unseen or new data.

After creating the model, we analyze the predictive performance of each model in training and test sets. For simplification purposes, we will only analyze the *accuracy* and *recall* scores of each model. The *accuracy* is important because it represents the overall predictive performance of the model, i.e., the capacity of the model to predict well the instances. The *recall* score has special relevance because what interests us is the positive class, that is, employees who are satisfied and this measure gives us the percentage of positive instances correctly predicted by the out-of-the-total instances that are actually positive. Obviously, in the Python file, all the predictive measures were calculated so it is possible to see the results of all of them there (precision, recall, accuracy, and confusion matrix). Still regarding *recall*, as it was our priority in the steps that we apply *GridSearchCV* we change its optimization goal to improve the *recall* by setting *scoring='balanced'*.

Regarding the **KNN** model, we start by applying the *KNeighborsClassifier* with a k=3 (k is the number of neighbors). Because the results between both sets (train and test) were so different (accuracy = 0.72 and 0.51, respectively) we suspected of overfitting. So, to prevent this, we tried several values for *n_neighbors*, using *GridSearchCV* and *Cross-Validation* to find its optimal value. Unfortunately, the results became even worse, with the training set having a perfect performance (accuracy and recall equal to 1) but not generalizing well to new data (accuracy = 0.58 and recall = 0.56). As the results did not improve, we tried a regularization technique by setting the weights='distance' and also attempted to change the distance metric used to "Manhattan" instead of using the default option (Euclidean distance). Still, with all of these changes, the results did not improve. Lastly, due to the little imbalance in the dataset, we also applied the *RandomOverSampling* technique, but nothing has been improved.

Concerning the **Naive Bayes**, we start by applying the *GaussianNB* and the results were better than the previous models with an accuracy and recall of 0.53 and 0.48 both in training and test sets. Even though the results were not so different we address the question of overfitting and imbalance using a *GridSearchCV* to find the best value for the var_*smoothing* and applying the *RandomOverSampling* technique, respectively. Despite this, the *recall* results always remained below 0.5 indicating that less than half of the true instances are correctly predicted by the model with unseen data.

With regard to the **Trees**, we start by applying the *DecisionTreeClassifier* and the results were very bad with the training set perfectly fitting the data (including the statistical noise) but generalizing badly for new data (accuracy = 0.49 and recall = 0.43). Seeing this, we suspect again of overfitting and in order to prevent this we apply a parameter to control the growth and complexity of the tree, *ccp_alpha* through a *GridSearchCV*. Surprisingly, the results didn't change significantly, which forced us to investigate the discrepancy between the results in other ways. We try to correct the imbalance with class_weights='balanced' and we obtain a perfect *recall* score for both training and test sets and a medium accuracy score of 0.51 and 0.61, respectively. This has to be with the *Receiver Operating Characteristic* that we applied in all *GridSearchCV* as mentioned above. This parameter forced the model to improve the recall and that is what happened for a *ccp_alpha* equal to 0.01. Finally, we try several other common hyperparameters used in trees, but the results were mixed and not as stable as with only the previous parameters.

With respect to the **Random Forest** model, the procedure was very similar to the one applied in Trees. First, we applied the *RandomForestClassifier*, where the results again showed overfitting. Then we reestablish the *class_weights='balanced'* parameter to balance the dataset and we impose limits on the growth of the various trees that make up the forest through *ccp_alpha*. However, in all these steps, the discrepancy in the results remained. In our last attempt to improve the model, we applied a GridSearchCV on the most common parameters applied to the model under analysis (it took a long time to run despite the small size of the dataset), and only then did the values change. Performance in the training set went down, representing a reduction in overfitting perhaps, but a large difference between all measures of performance between training and testing remained.

When estimating the model through ***LogisticRegression***, we impose right at the beginning the *class_weight='balanced'* and we add a new parameter, max_iter to limit the maximum number of iterations the optimization algorithm underlying this model (gradient descend typically) would perform to converge to the optimal solution. The results showed relatively close values between both sets, but in order to bring them even closer and correct this small overfitting we used the *Lasso Regularization* adding the *penalty* and the *solver* underlying the technique and a value of 0.5 in the parameter *C* that measures the inverse of the regularization strength. The result was contrary to expectations, with the values deviating slightly. Finally, instead of imposing a value on *C*, we created a hyper by assigning it several values, and through a

GridSearchCV we looked for the one that supposedly would allow us to obtain a better result, but that was not what happened.

In what concerns to the **Neural Network** model, we first apply *MLPClassifier* with 3 hidden layers with respectively 3, 4 and 5 neurons. The results were better compared to the majority of previous models but the difference between both sets kept up again. As we suspect that overfitting would be to blame, we apply the *L2 regularization* to the model through a *GridSearchCV* for the best *alpha* and the results remain bad. Our last attempt to improve the predictive performance of the model was to apply a RandomOverSampling and the respective results changed. The recall score in both sets became perfect but the accuracy in the training was about 0.5 and in the test 0.61.

Despite the previous models having been taught in class, we decided to go further and analyze two different models, but with a lot of potentials to improve our results due to their characteristics of efficiency.

Regarding the **Support Vector Machine (SVM)**, it aims to find an optimal decision boundary between different classes by maximizing the margin, which is the distance between the decision boundary and the nearest data points. First, we used the model in “normal” terms giving rise to results similar to the models we saw earlier, with a very significant difference between the training and test set (recall = 0.85 and 0.46, respectively). The SVM has several common hyperparameters that can be optimized to the specific dataset like the majority of the other models. In this sense, we did a *GridSearchCV* on the parameters *C*, *kernel* which defines the type of kernel function to be used in SVM, and gamma which is specific for one of the kernel options. Although the results in the test set increased a little bit, in the training set they became almost perfect, leading to a higher difference between the sets.

Concerning the **XGBoost** model, is an ensemble model that combines the predictions of multiple individual decision trees to make accurate predictions. We can think that it is equal to the trees model, but in fact, it uses a different learning algorithm and additional optimization techniques to improve the robustness of the model, i.e., its ability to generalize. Despite being more difficult to use, the XGBoost is more efficient. To estimate the model, first, we simply apply the “normal” conditions using *XBGClassifier*. Then, because the results were very poor (recall = 1 and 0.5, respectively), we suspected overfitting, and we applied a regularization technique that prevents that phenomenon by doing a *GridSearchCV* in two hyperparameters: *alpha* and *lambda*. To our dismay, this model was also not able to deal with the problem of worker satisfaction.

Putting all this information together, we can conclude that the **KNN**, **Random Forests**, **SVM**, and **XGBoost** models are not good at explaining employee satisfaction, which means that they are not able to adequately capture the factors that influence this management problem nor to provide accurate predictions about it.

Looking at the recall, in relation to the other models, **GaussianNB** and **Logistic Regression** are the only ones that, despite presenting values that are not very high (between 44 and 51%), in the context of the problem of employee satisfaction (relatively simple), we can consider reasonably good. This is because it means that these models can predict, on average, almost half of the positive instances correctly, and above all, they are able to apply the knowledge they learned during training to make accurate predictions on never-before-seen data (mainly the GNB on which the difference between sets is smaller). So, these are the most stable and robust among all we have estimated.

Table 2 – Predictive performance of GaussianNB model (best) with hyperparameter ‘var_smoothing’

GaussianNB (‘var_smoothing’)	**Measure**	**Train**	**Test**
	Accuracy	0.53	0.49
	Recall	0.53	0.49

Table 3 – Predictive performance of Logistic Regression model (best) with parameters ‘class_weight’ and ‘max_iter’

Logistic Regression (Class_weight= ‘balanced’ + max_iter)	**Measure**	**Train**	**Test**
	Accuracy	0.57	0.51
	Recall	0.57	0.51

When it comes to the other two models, **Trees**, and **Neural Networks**, these with certain changes and optimizations manage to present reasonable results, but better results in the test set than with the training data, which can mean underfit of the data, which suggests that other values for the hyperparameters should have been considered.

Conclusion

As mentioned before, we decided to analyse the models again in two different scenarios: excluding the *job level* variable (1) and the other excluding the *salary* (2). In fact, it was possible to conclude that most of the models have better predictive performance once we eliminate one of the correlated variables (correcting the problem of multicollinearity).

We choose to go further analysing the Naive Baynes Model and the Logistic Regression considering the two possible new scenarios, once these were the models that presented better predictive performance.

Regarding the **Naive Baynes** model, we can conclude that the results were practically the same as before, having a slight increase in scenario (1), where the recall score was 0.55 and 0.51 in the train and test set. However, the application of the *var_smothing* and the *RandomOverSampling* decreased both scenarios' predictive performance in the test set.

Considering the **Logistic Regression** Model, likewise, before, the results were very similar to the ones predicted in scenarios (1) and (2). In fact, in scenario (1) the results after all the adjustments were exactly the same, whereas in (2) the accuracy and the recall score decreased a lot in the training test.

It is possible to conclude that the best variable to eliminate would be j*ob level,* however, the predictive performance of the model did not increase significantly, so there would be no need for that.

The dataset utilized is purely fictional, thereby, does not represent any real company, leading to a possible justification for the low obtained values. Furthermore, another explanation relates to the fact that there are few instances in the dataset and also some variables are not the most appropriate to predict the present problem. Therefore, a possible solution passes by changing some of the variables and/or introducing more variables, such as ones related to the working environment, social interactions and advancement opportunities, according to the literature.

With effect, it is not entirely possible to assist the fictional organization in handling the current challenge. However, it was possible to provide it with some valuable insights regarding certain variables that may influence employees' satisfaction and of which the company must be aware.

References

Aziri, B. (2011). Job satisfaction: A literature review. *Management research & practice*, *3*(4).

GeeksforGeeks. (2023a). XGBoost. *GeeksforGeeks*. https://www.geeksforgeeks.org/xgboost/

GeeksforGeeks. (2023b). Support Vector Machine Algorithm. *GeeksforGeeks*. https://www.geeksforgeeks.org/support-vector-machine-algorithm/

Office, S. P. P. a. D. S. M. R. (2020). A guide to controlling your XGBoost model. *Capital One*. https://www.capitalone.com/tech/machine-learning/how-to-control-your-xgboost-model/

Singh, S. K., & Tiwari, V. (2011). Relationship between motivation and job satisfaction of the white-collar employees: A case study. *Management insight*, *7*(2), 31-39.

Employee Satisfaction Index Dataset. (2020, June 27). Kaggle. https://www.kaggle.com/datasets/mohamedharris/employee-satisfaction-index-dataset

Exhibits

Exhibit 1- Histplot of the Outcome Variable

```
[5] sns.histplot(df['satisfied'])

<Axes: xlabel='satisfied', ylabel='Count'>
```

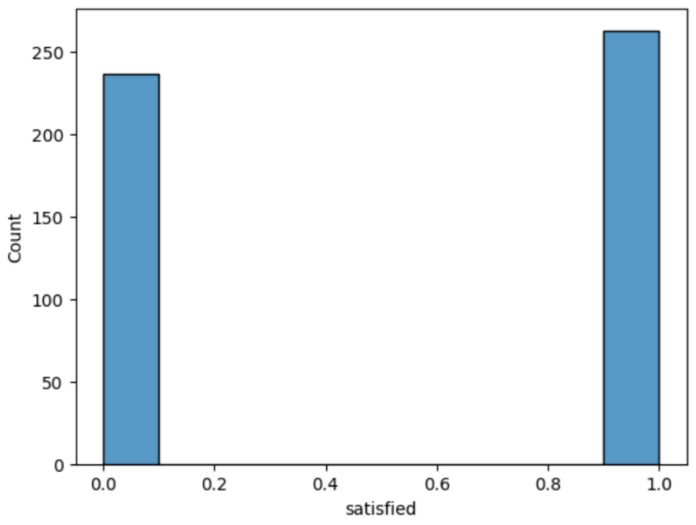


Exhibit 2-General Pairplot

```
sns.pairplot(df)

<seaborn.axisgrid.PairGrid at 0x7f6ef306bfa0>
```

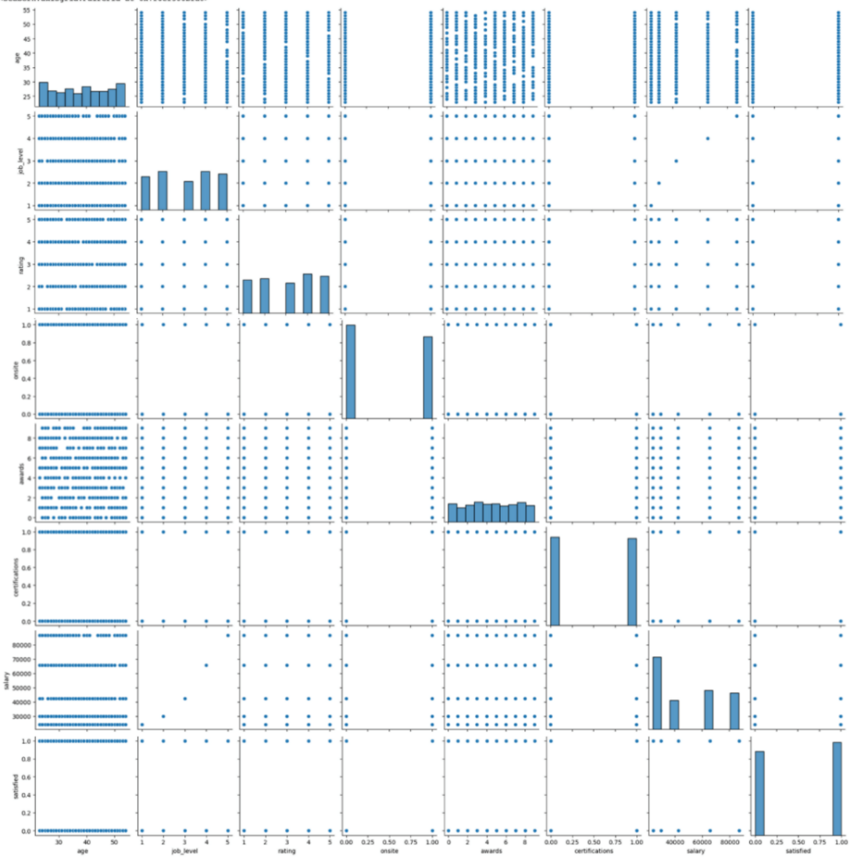
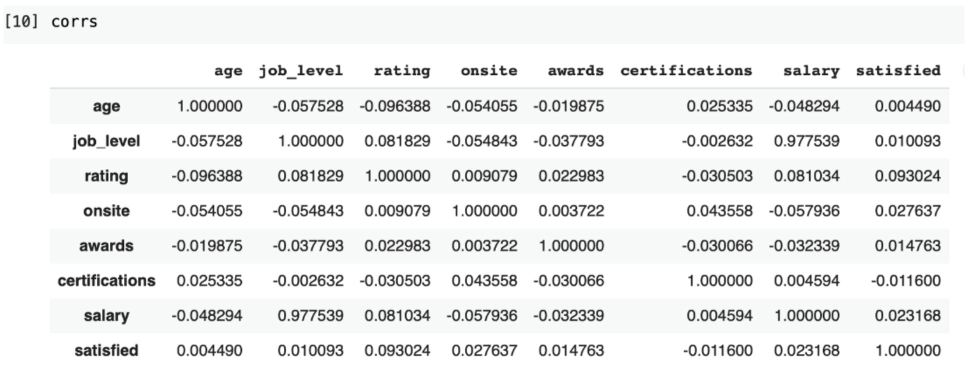


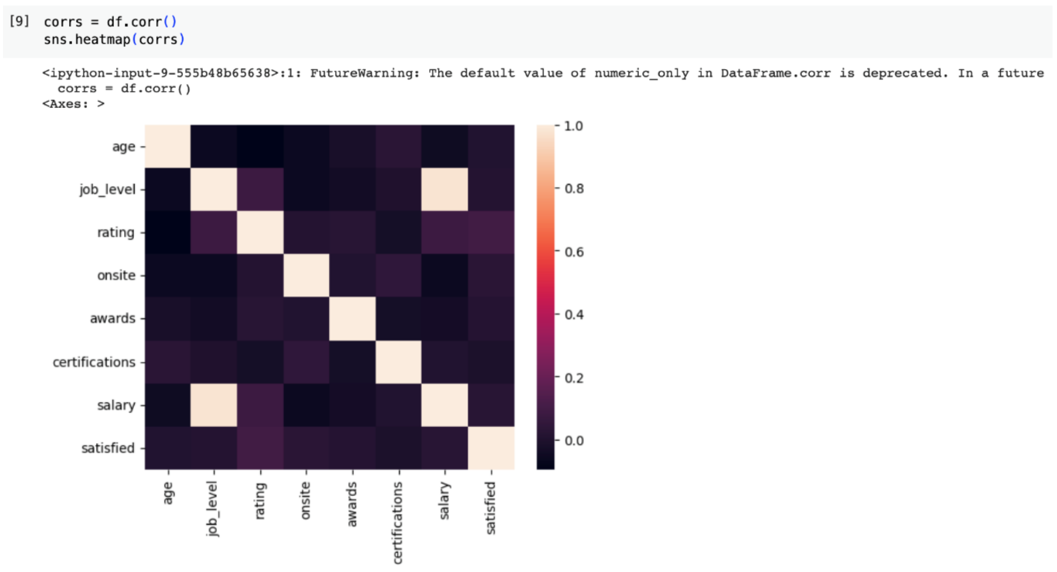
Exhibit 3-Correlation Matrix



```
[10] corrs
```

	age	job_level	rating	onsite	awards	certifications	salary	satisfied
age	1.000000	-0.057528	-0.096388	-0.054055	-0.019875	0.025335	-0.048294	0.004490
job_level	-0.057528	1.000000	0.081829	-0.054843	-0.037793	-0.002632	0.977539	0.010093
rating	-0.096388	0.081829	1.000000	0.009079	0.022983	-0.030503	0.081034	0.093024
onsite	-0.054055	-0.054843	0.009079	1.000000	0.003722	0.043558	-0.057936	0.027637
awards	-0.019875	-0.037793	0.022983	0.003722	1.000000	-0.030066	-0.032339	0.014763
certifications	0.025335	-0.002632	-0.030503	0.043558	-0.030066	1.000000	0.004594	-0.011600
salary	-0.048294	0.977539	0.081034	-0.057936	-0.032339	0.004594	1.000000	0.023168
satisfied	0.004490	0.010093	0.093024	0.027637	0.014763	-0.011600	0.023168	1.000000

Exhibit 4-Correlation Matrix (Heatmap)



```
[9] corrs = df.corr()
    sns.heatmap(corrs)

    <ipython-input-9-555b48b65638>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future
      corrs = df.corr()
    <Axes: >
```