Employee Future Prediction

Authors: João Reis, 98474 (50%) and Ricardo Rodriguez, 98388 (50%)
Mestrado em Engenharia Informática - DETI
Fundamentos de Aprendizagem Automática
Professor: Pétia Georgieva

***Abstract*—The purpose of this project is to implement machine learning models capable of helping a company's human resources department to predict if an employee will leave the company in the next 2 years based on some data. All the implemented machine learning algorithms are compared with each other and with algorithms from similar works by their accuracy.**

***Index Terms*—Machine Learning, Cross-validation, Hyperparameter Tuning, Logistic Regression, Support Vector Machines, Neural Network, Voting**

I. Introduction

In this paper, we present a study on developing machine learning models for predicting employee turnover within a company. The ability to predict which employees are likely to leave the company in the near future can aid organizations in retaining valuable talent and reducing the costs associated with high turnover rates. Thus, an accurate prediction of employee turnover is very useful and allows organizations to address issues related to employee turnover.

We propose different machine learning approaches that utilize a combination of demographic and individual data to train and evaluate several models. The implemented models are then compared in terms of their predictive accuracy according to the base, hypertuned and k-fold cross validated models.

The dataset used for this project is publicly available on *Kaggle* [1]. The notebook containing all the implemented code is available on our GitHub repository [2]. The reason behind the choosing of this topic is due to the fact that it's different than the health field, which we interacted with in the last project, and because it can very useful for a company's human resources department to predict whether or not an employee is going to leave.

II. State of the art

As stated before, the dataset is publicly available on *Kaggle* and, at the time we're writing this, there is a total of 48 notebooks with different purposes and approaches to the same context.

Unlike what was done in the last project, this time we decided to visualize, interpret, preprocess and train the data with some machine learning models before analysing other notebooks from *Kaggle*, in order to test the knowledge we acquired in class and in the previous project [3] without being influenced by other studies.

After implementing our approach on this problem, we started analysing the most popular notebooks, as well as the ones with which had the best results for the models, in order to better comprehend the authors reasoning and methodologies. Our highest priority was to verify if our procedure matched with the procedure of the notebooks with the best accuracy, with a special focus on how data was preprocessed and how model hyperparameters were chosen, and try out some observed methodologies.

Initially, we analysed Sonali Singh's notebook [4], the most popular one on *Kaggle*, which is very well structured and informative. Sonali's preprocessing of the data consisted of dropping duplicates, encoding categorical features, balancing data and feature scaling. The dataset split ratio was 70/30. Table I presents the best cross-validated models, being AdaBoost and ExtraTrees the most accurate ones.

Table I
Sonali Singh's notebook models and accuracy

Machine Learning Model	Test Accuracy Score
Ada Boost Classifier	0.817553
Extra Trees Classifier	0.815846
Decision Tree Classifier	0.780894
Logistic Regression	0.769828
Random Forest Classifier	0.722098

Gagan Shandra's notebook [5] preprocessing approach is the same as the one from Sonali Singh, for the exception of Gagan not balancing the data. The dataset was also splitted with a 70/30 ratio and Gradient Boosting, Stochastic Boosting and Cat Boosting classifiers were the models with the best results, with accuracies of 84%, as observed in Table II (hypertuned models).

Table II
Gagan Shandra's notebook models and accuracy

Machine Learning Model	Test Accuracy Score
ADA Boost	0.855995
Gradient Boost	0.852619
Cat Boost	0.848853
Stochastic Gradient Boosting	0.843123
Hypertuned Decision Tree Classifier	0.841572

Manthan Shettigar's notebook [6] presented a 80/20 data split, featuring the removal of duplicates and a bigger feature engineering transformation with the use of OneHotEncoder [7] and OrdinalEncoder [8] to encode the features. Decision Tree Classifier Light Gradient Boosting Machine (LGBM) and XGB Classifier were models with the best results with accuracies of 83%, as depicted in Table III.

Table III
MANTHAN SHETTIGAR'S NOTEBOOK MODELS AND ACCURACY

Machine Learning Model	Test Accuracy Score
Light GBM Classifier	0.830018
Decision Tree Classifier	0.835443
Ada Boost Classifier	0.757685
XGB Classifier	0.830018
Gradient Boost Classifier	0.759494

Adytia's notebook [9] was the last one to be analysed and we observed that his procedure on data preprocessing only consisted of encoding features with OrdinalEncoder. All the implemented models were evaluated by cross-validation and the results are presented in the table below:

Table IV
ADYTIA'S NOTEBOOK MODELS AND ACCURACY

Machine Learning Model	Test Accuracy Score
Support Vector Machines	0.8787
Light GBM Classifier	0.8769
Hist. Gradient Boost Classifier	0.8750
Cat Boost Classifier	0.8746
Gradient Boost Classifier	0.8638

In this project, our main goal is to present similar, or even better, accuracy scores than other notebooks on the *Kaggle* dataset, alongside with complementary information for this classification problem such as the confusion matrix (accuracy, precision, recall, F1 score, etc.).

We will also explain our initial perspective on how to approach this problem and the changes made after research and the analysis of the previously mentioned notebooks.

III. DATA ANALYSIS

In this chapter, we will explore and analyse the chosen dataset. Understanding the data is a crucial step to comprehend not only the problem but also to outline preprocessing course of action.

This dataset is comprised of 4653 rows and 9 columns, which have the following variables:

- **Education** - Education level [Bachelors, Masters, PhD].
- **JoiningYear** - Joining year of the employee [2012, 2013, 2014, 2015, 2016, 2017, 2018]
- **City** - Working location (India) [Bangalore, Pune, New Delhi]
- **PaymentTier** - Payment tier of the employee [1 (highest), 2 (mid-level), 3 (lowest)]
- **Age** - Age of the employee [20-45]
- **Gender** - Gender of the employee [Male, Female]
- **EverBenched** - Ever kept out of projects for 1 month or more ['Yes', 'No']
- **ExperienceInCurrentDomain** - Experience in the current domain [0-7]
- **LeaveOrNot** - Which employee will leave the organizaion? [0 (no), 1 (yes)]

Our main goal is to implement a machine learning model capable of accurately predicting wether the target variable *LeaveOrNot* is 0 (will not leave the company) or 1 (will leave the company).

As we can see in Fig. 1, the distribution of *LeaveOrNot* is not perfectly balanced (50/50) but it isn't completely disproportionate. In fact, the dataset has an approximate ratio of 66/34.

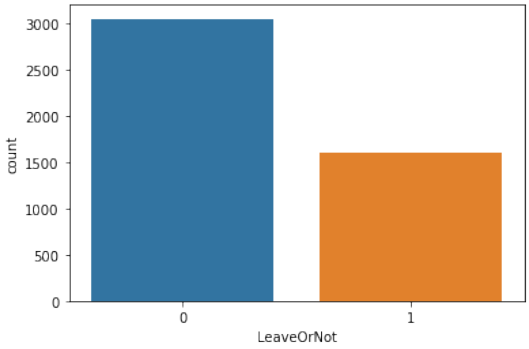

Figure 1. Leave or Not Distribution

A. Categorical Features

The categorical features in this dataset are Education, City, Gender, and EverBenched, whose definitions were presented previously in this chapter.

The following figures present the distribution of each categorical feature when compared with to the target variable *LeaveOrNot*.

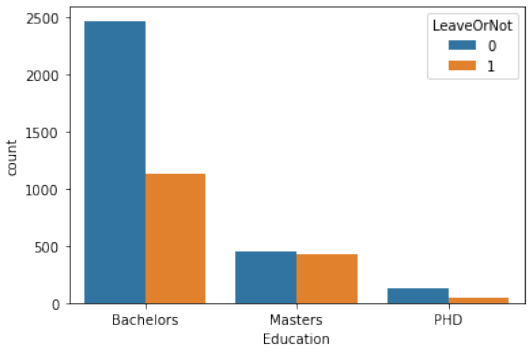

Figure 2. Education Distribution

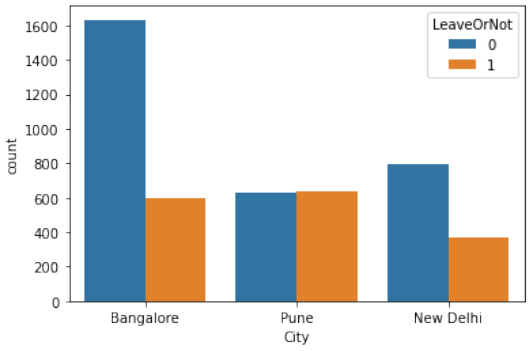

Figure 3. City Distribution

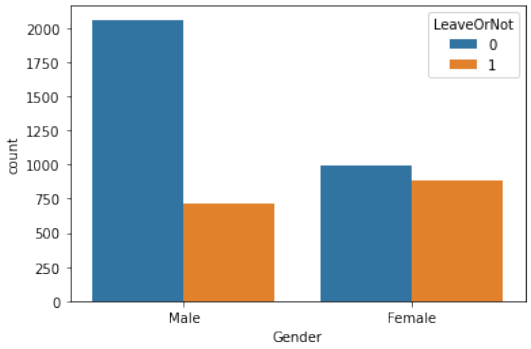

Figure 4. Gender Distribution

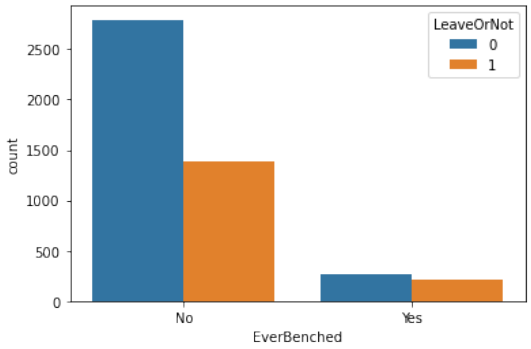

Figure 5. Ever Benched Distribution

B. *Continuous Features*

The continuous features of the dataset are JoiningYear, PaymentTier, Age and ExperienceInCurrentDomain.

The following figures present the distribution of each continuous feature when compared with *LeaveOrNot*, the target variable.

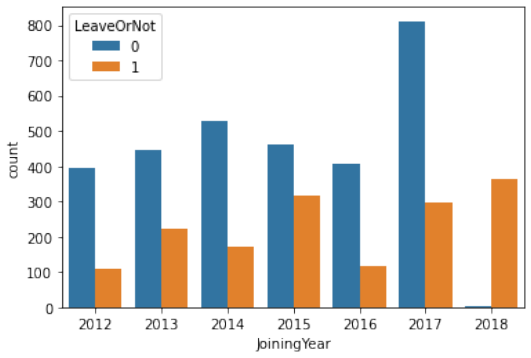

Figure 6. Joining Year Distribution

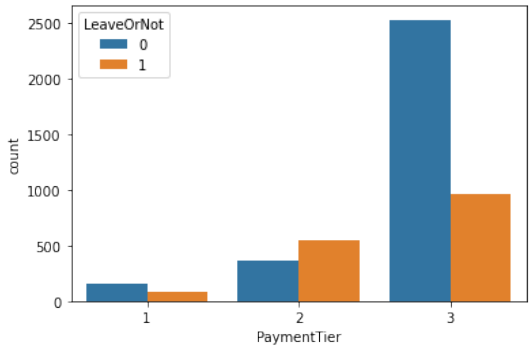

Figure 7. Payment Tier Distribution

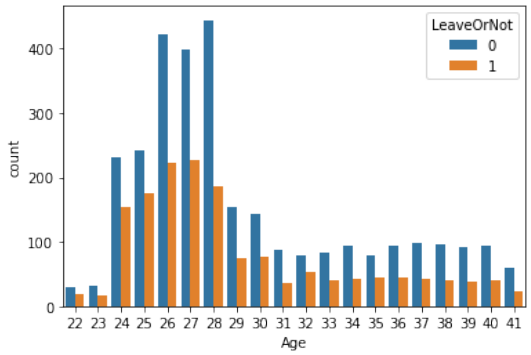

Figure 8. Age Distribution

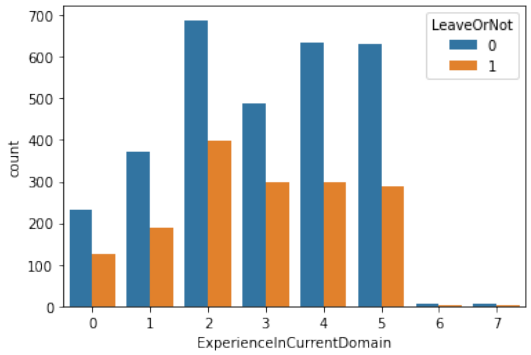

Figure 9. Experience in Current Domain Distribution

In the next figure, we can visualize how experience in the current domain affects on payment tier.

Figure 10. How does experience affects payment tier? (1 - highest, 2 - medium, 3 - lowest)

Analysing the Figure 10, we can verify that the highest-paid tier (1) has the same ExperienceInCurrentDomain median

between males and females but females in this tier have, in general, less experience than males. In the Payment Tier 2 (medium), the average ExperienceInCurrentDomain is the same between males and females but the most experienced females surpass the value in years of most experienced males by one increment. In the Payment Tier 3 (lowest), both male and female have the same distribution.

Figure 11 presents the pairplot visualization between continuous features of the dataset, allowing to distinguish male from female values.

Figure 11. Bi-variate distribution of continuous features

C. Data Conclusions

After exploring the data visualization of categorical and continuous features, we can jump to some conclusions:

- In terms of **Education**, employees with **Masters Degree** are the most likely group to leave the company.
- Employees who joined in **recent years** are most likely to leave the company.
- Employees from **Pune City** are most likely to leave.
- In terms of **Salary**, employees with **Payment Tier 3** (lowest payment tier) are most likely to leave.
- **Female** employees leave more often than male counterparts.
- Most of the employees who have been **benched** left the company.
- Employees with **less experience** leave the company more often than more experienced ones.
- **Young** employees are the most likely to leave.

IV. Data Preprocessing

Preparing the data is very important, since we need to ensure that the data is in a format that can be effectively used by the machine learning models and produce more accurate results.

For this project, we started verifying if the dataset had null values in it and we found this was not the case. Then, we checked if there were duplicated rows and we found out there were 1889 duplicated records. Although there were 1889 duplicated records in the dataset, we didn't remove them since duplicates are what provide the weight of the evidence and we want to train the machine learning model by accumulating real-life data, considering frequency of rows. The same thought process happened for data balancing.

For this project, we chose an 80/20 split ratio, with 80% of the dataset being used for the training of machine learning models and the remaining 20% for model testing.

A. Feature Encoding

As we observed in the Data Visualization chapter, we have four categorical features in the dataset. Therefore, we need to convert those categorical variables into numerical values, so that they can be used as input features for our machine learning models.

There are different ways to encode categorical features, such as one-hot encoding and ordinal encoding, both of them mentioned in the state of art chapter.

Ordinal encoding assigns an integer value to each category based on its position in an ordinal scale. While this method is useful when working with ordinal variables (which present an ordinal relationship between them), this is only the case for the Education variable.

Thus, we decided to use one-hot encoding that creates a binary column for each unique category in the feature and assigns a 1 or 0 value to indicate the presence or absence of that category in a given sample. In contrary to ordinal encoding, this method is useful when working with categorical features with non-ordinal relationship.

B. Correlation

Correlation matrices are fundamental to understand how our data variables are correlated with each other. It's a powerful tool to summarize the dataset, to identify and visualize patterns and what features are the most significant in the given data. Figure 12 presents the correlation matrix between all encoded features (categorical and continuous).

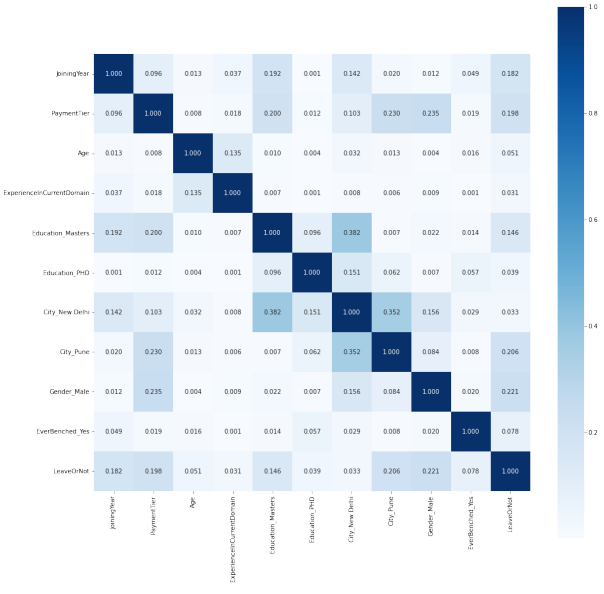

Figure 12. Correlation matrix between features

Analysing *Fig. 12*, we can identify a decent correlation between *Education_Masters* and *City_New Delhi* (0.382) and *City_New Delhi* and *City_Pune* features.

However, the most important thing we can induce when interpreting the correlation matrix in *Fig. 12* is the correlation between the features and the target variable *LeaveOrNot*, in order to understand which features have the most impact over the prediction of an employee's future.

C. Feature Scaling

The last step of data preprocessing was feature scaling, which ensures that all input features are on the same scale, avoiding poor results from some machine learning algorithms which are sensitive to scale.

For feature scaling of the dataset, we used *scikit-learn*'s StandardScaler. StandardScaler uses the standardization method which involves subtracting the mean of the feature and dividing by the standard deviation, as depicted in Figure 13.

Standardization:

$$z = \frac{x - \mu}{\sigma}$$

with mean:

$$\mu = \frac{1}{N}\sum_{i=1}^{N}(x_i)$$

and standard deviation

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Figure 13. Standardization formula [11]

This results in a standard normal distribution where the mean is 0 and the standard deviation is 1, while not changing the shape of the distribution.

V. Machine Learning Models

A. Introduction

To solve this classification problem, we implemented some Machine Learning models, such as Logistic Regression, Support Vector Machine, Multi-Layer Perceptron, Ensemble Voting Classifier, Gradient Boosting Classifier, and Light Gradient Boost Machine.

Ensemble learning is a way of generating various base classifiers from which a new classifier is derived which performs better than any constituent classifier. The model Ensemble Voting Classifier and the model Gradient Boosting Classifier are types of ensemble learning. We chose to implement Logistic Regression, Support Vector Machine and Multi-Layer Perceptron to compare the results of this "simple and basic" models with the results of ensemble learning models.

Initially, we tested each one of the previously mentioned models with the base/default model. Then, we did the Hyper-parameter tuning of the base model and, finally, K-Fold Cross-Validation of the hyper-tuned model was done, in order to maximize the potential of these models.

This chapter demonstrates the implementation and results of the machine learning base models, with no hyper-parameter tuning and k-fold cross validation.

For each outcome of the models we implemented, a confusion matrix is going to be presented. The confusion matrix is used in machine learning classification problems in since it presents the number of true positives, false positives, false negatives and true negatives for the variable we're predicting. Thus, a "better" machine model is the one who has the most number of true negatives and true positives while having the smallest number of false negatives and false positives possible.

B. Logistic Regression

One of the most popular Machine Learning algorithms is Logistic Regression. This model is used for solving classification problems, like this one, and predicts the probability of a binary (1/0 or yes/no) event occurring. In this case, one (1) if the employee leaves the company and zero (0) if he stays.

a) ***Base Model****:* The base model confusion matrix, the classification report and the respective results are described in the following figures.

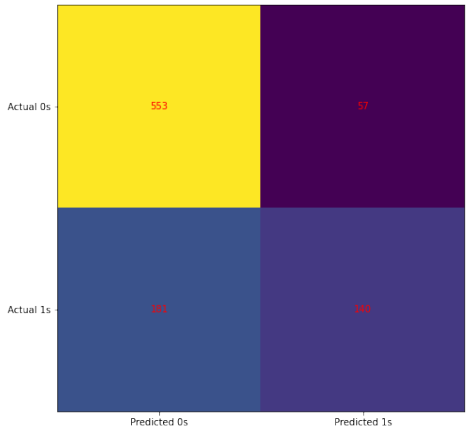

Figure 14. Logistic Regression Confusion Matrix

We are focused in getting the smallest number of False Negatives possible, *ie*, cases when an employee leaved the company but the model predicted the contrary (he stays). So, by analysing the last figure, we can confirm there is a low number of false negatives and we can consider our model as being relatively accurate.

The classification report of Logistic Regression is presented below:

Table V
Logistic Regression Classification Report

	Precision	Recall	F1-Score	Support
0	0.75	0.91	0.82	610
1	0.71	0.44	0.54	321
accuracy			0.74	931
macro avg	0.73	0.67	0.68	931
weighted avg	0.74	0.74	0.73	931

The **accuracy** of the Logistic Regression model is **74.44%** while the **F1-Score** is **54.05**%.

*b) **Hyper-Parameter Tuning**:* The parameters used to optimize the base model are depicted in the figure below:

Table VI
ALL LOGISTIC REGRESSION PARAMETERS

Parameter	Values
solver	['liblinear','sag']
max_iter	[100, 200]
C	[0.01, 0.1, 0.5, 1, 10, 100]
class_weight	['none', 'balanced']
penalty	['none', 'l1', 'l2']

The best parameters selected for the Logistic Regression model are:

Table VII
BEST LOGISTIC REGRESSION PARAMETERS

Parameter	Values
solver	'sag'
max_iter	100
C	0.1
class_weight	'none'
penalty	'l2'

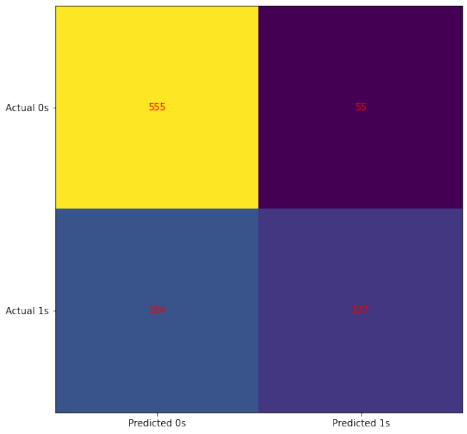

Figure 15. Hyper-tuned Logistic Regression Confusion Matrix

Table VIII
HYPER-TUNED LOGISTIC REGRESSION CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.75	0.91	0.82	610
1	0.71	0.43	0.53	321
accuracy			0.74	931
macro avg	0.73	0.67	0.68	931
weighted avg	0.74	0.74	0.72	931

The **accuracy** of the Logistic Regression model is **74.33%** while the **F1-Score** is **53.41%**.

*c) **K-Fold Cross Validation**:* For **K-Fold Cross Validation**, there was **no improvement** when applying cross validation to the hyper-tuned model. Thus, the results are exactly the same as the previous step.

C. Support Vector Machine

Support Vector Machine (SVM) is one of the most robust prediction method. This model consists on create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. The extreme vectors or support vectors define the gap between categories.

*a) **Base Model**:* In the base model, we use the model as explained above without any optimization. The confusion matrix, as well as the classification report, is presented below:

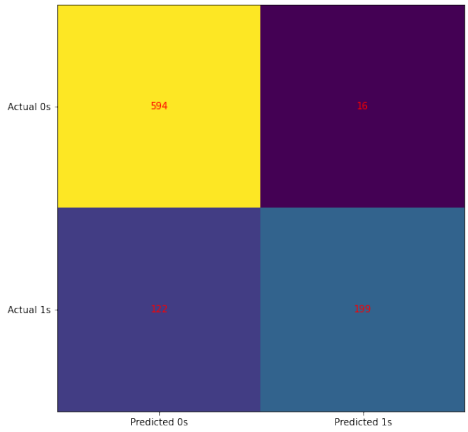

Figure 16. SVM Confusion Matrix

Table IX
SVM CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.83	0.97	0.90	610
1	0.93	0.62	0.74	321
accuracy			0.85	931
macro avg	0.88	0.80	0.82	931
weighted avg	0.86	0.85	0.84	931

The **accuracy** obtained with this model was **85.18%** but the **F1 Score** was a little bit higher with **74.25%**.

*b) **Hyper-Parameter Tuning**:* The parameters used to optimize the base model are depicted in the figure below:

Table X
ALL SVM PARAMETERS

Parameter	Values
C	[100, 1000, 2000]
gamma	[0.001, 0.01]
kernel	['linear', 'rbf']

Table XI
BEST SVM PARAMETERS

Parameter	Values
C	1000
gamma	0.01
kernel	'rbf'

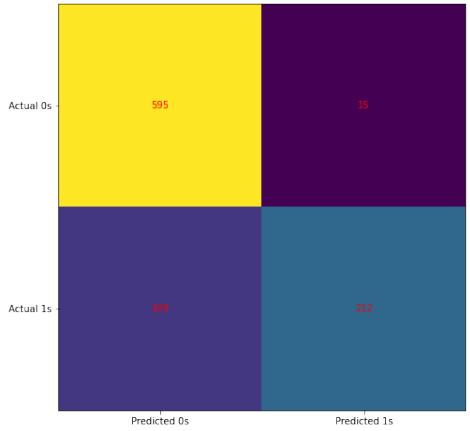

Figure 17. Hyper-tuned SVM Confusion Matrix

Table XII
HYPER-TUNED SVM CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.85	0.98	0.91	610
1	0.93	0.66	0.77	321
accuracy			0.87	931
macro avg	0.89	0.82	0.84	931
weighted avg	0.88	0.87	0.86	931

The accuracy obtained with this the hyper-parameters tuning was better than the accuracy on base model. The percentage value of **accuracy** is **86.68%** but the **F1 Score** is **77.37%**.

*c) **K-Fold Cross Validation**:* All the results using K-Fold Cross Validation, ie, accuracy and F1 Score percentage, confusion matrix and classification report, are equal to the results obtained using **Hyper-Parameter Tuning**.

D. Multi-Layer Perceptron Classifier

Multi-Layer Perceptron, or MLP, is an artificial neural network consisting of three types of layers, whose nodes are called *perceptrons*. The input layer receives all the input values for the features and they are distributed to the first hidden layer of *perceptrons*. Next, the first hidden layer distributes its output to the second hidden layer of *perceptrons*, and so on. In the end, the output layer is responsible for receiving the outcome of the last hidden layer *perceptrons*. [12]

*a) **Base Model**:* For the MLP base model, we only applied one setting (*max_iter = 10000*) and the confusion matrix, as well as the classification report, is presented below:

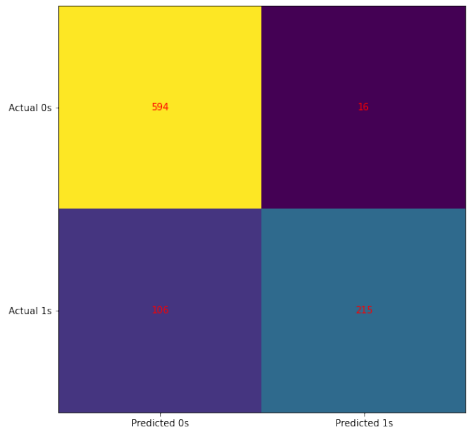

Figure 18. MLP Confusion Matrix

Table XIII
MLP CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.85	0.97	0.91	610
1	0.93	0.67	0.78	321
accuracy			0.87	931
macro avg	0.89	0.82	0.84	931
weighted avg	0.88	0.87	0.86	931

For the base model, the **Accuracy** was **86.89%** and the **F1-Score** was **77.89%**.

*b) **Hyper-Parameter Tuning**:* The parameters used to optimize the base model are depicted in the figure below:

Table XIV
ALL MLP PARAMETERS

Parameter	Values
hidden_layer_sizes	[(15,),(20,),(25,)]
activation	['tanh','relu']
solver	['adam']
max_iter	[10000],
alpha	[0.0001,0.01]

Table XV
BEST MULTI-LAYER PERCEPTRON PARAMETERS

Parameter	Values
hidden_layer_sizes	(25,)
activation	'relu'
solver	'adam'
max_iter	10000
alpha	0.0001

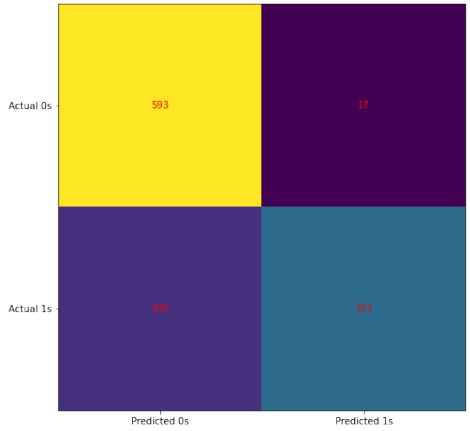

Figure 19. Hyper-tuned MLP Confusion Matrix

Table XVI
HYPER-TUNED MLP CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.86	0.97	0.91	610
1	0.93	0.69	0.79	321
accuracy			0.87	931
macro avg	0.89	0.83	0.85	931
weighted avg	0.88	0.87	0.87	931

For the hyper-tuned model, the **Accuracy** was **87.43%** and the **F1-Score** was **79.06%**, a small improvement in comparison to the base model results.

*c) **K-Fold Cross Validation**:* The Accuracy and F1-Scores for the MLP hyper-tuned model after K-Fold Cross Validation had a small decrease on both accuracy and F1 scores, with percentage values of **86.78%** and **78.07%**, respectively.

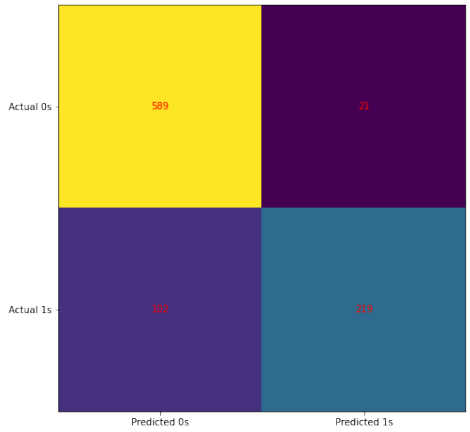

Figure 20. K-Fold Cross Validated MLP Confusion Matrix

Table XVII
K-FOLD CROSS VALIDATED MLP CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.85	0.97	0.91	610
1	0.91	0.68	0.78	321
accuracy			0.87	931
macro avg	0.88	0.82	0.84	931
weighted avg	0.87	0.87	0.86	931

E. Ensemble Voting Classifier

A Voting Classifier trains different models using the chosen algorithms, returning the majority's vote as the classification result. We used a combination of the last models explained: Logistic Regression, Support Vector Machine and Multi-Layer Perceptron Classifier.

*a) **Base Model**:* For the EVC base model, we only applied one setting (*voting = 'hard'*), and it means predicting the class labels with the majority rule voting. The confusion matrix, as well as the classification report, is presented below:

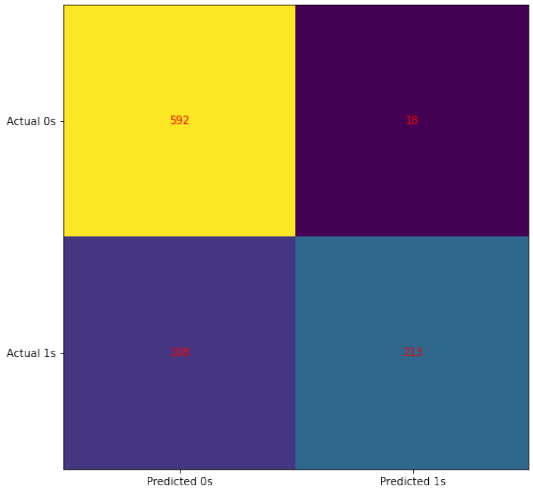

Figure 21. EVC Confusion Matrix

Table XVIII
EVC CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.85	0.97	0.90	610
1	0.92	0.66	0.77	321
accuracy			0.86	931
macro avg	0.88	0.82	0.84	931
weighted avg	0.87	0.86	0.86	931

For the base model, the **Accuracy** was **86.47%** and the **F1-Score** was **77.17%**.

*b) **Hyper-Parameter Tuning**:* The parameters used to optimize the base model are depicted in the figure below:

Table XIX
ALL EVC PARAMETERS

Parameter	Values
voting	['hard', 'soft']
weights	[(1,1,1), (2,1,1), (1,2,1), (1,1,2), (2,2,1), (2,1,2), (1,2,2)]

Table XX
BEST ENSEMBLE VOTING CLASSIFIER PARAMETERS

Parameter	Values
voting	'hard'
weights	(1, 2, 1)

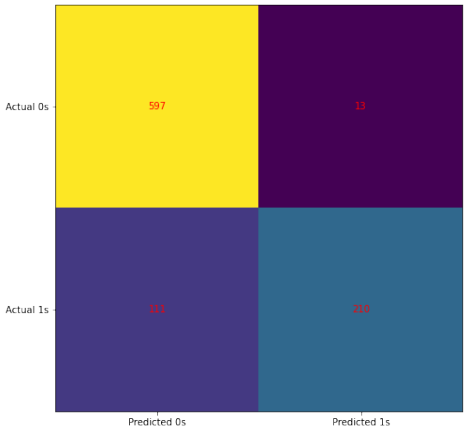

Figure 22. Hyper-tuned EVC Confusion Matrix

Table XXI
HYPER-TUNED EVC CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.84	0.98	0.91	610
1	0.94	0.65	0.77	321
accuracy			0.87	931
macro avg	0.89	0.82	0.84	931
weighted avg	0.88	0.87	0.86	931

For the hyper-tuned model, the **Accuracy** was **86.68%** and the **F1-Score** was **77.21%**, a little better in comparison to the base model results.

*c) **K-Fold Cross Validation**:* Applying K-Fold Cross Validation, the confusion matrix, as well as the classification report, is presented below:

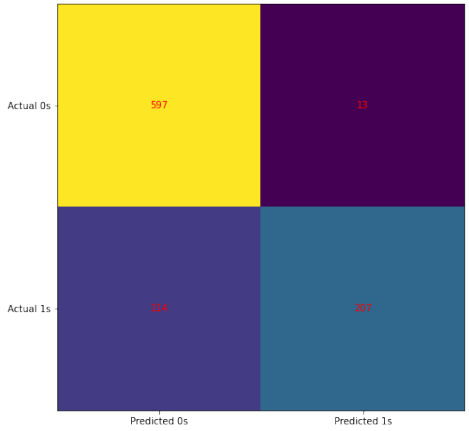

Figure 23. K-Fold Cross Validated EVC Confusion Matrix

Table XXII
K-FOLD CROSS VALIDATED EVC CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.84	0.98	0.90	610
1	0.94	0.64	0.77	321
accuracy			0.86	931
macro avg	0.89	0.81	0.83	931
weighted avg	0.87	0.86	0.86	931

After K-Fold Cross Validation, the **Accuracy** was **86.36%** and the **F1-Score** was **76.53%**.

F. Gradient Boosting Classifier

Gradient Boosting Classifier (GBC) is an ensemble machine learning technique, just like the previous one, that combines multiple weak models to create a stronger model. In each iteration, a new weak model is trained on the errors made by the previous model and these weak models are combined to form a decision tree ensemble. The final model is a weighted sum of all the weak models, where the weights are determined by the gradient descent algorithm. [13]

*a) **Base Model**:* In the base model, we use the model as explained above without any optimization. The confusion matrix, as well as the classification report, is presented below:

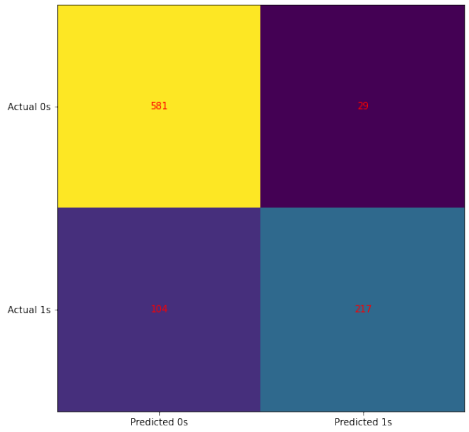

Figure 24. GBC Confusion Matrix

Table XXIII
GBC CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.85	0.95	0.90	610
1	0.88	0.68	0.77	321
accuracy			0.86	931
macro avg	0.87	0.81	0.83	931
weighted avg	0.86	0.86	0.85	931

The **accuracy** obtained with this model was **85.71%** but the **F1 Score** was a little bit higher with **76.54%**.

*b) **Hyper-Parameter Tuning**:* The parameters used to optimize the base model are depicted in the figure below:

Table XXIV
ALL GBC PARAMETERS

Parameter	Values
n_estimators	[100, 150, 200, 250]
max_depth	[5,7,9]
loss	['log_loss', 'deviance']
learning_rate	[0.001, 0.01, 0.1, 1]

Table XXV
BEST GBC PARAMETERS

Parameter	Values
n_estimators	150
max_depth	7
loss	'deviance'
learning_rate	0.01

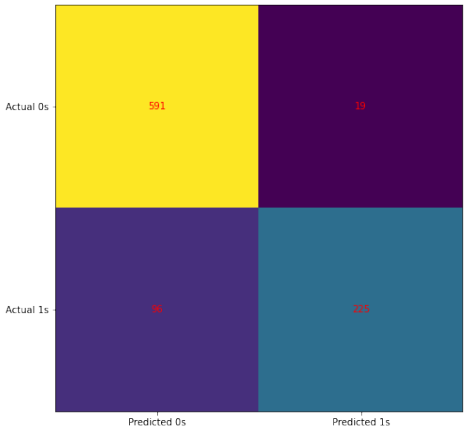

Figure 25. Hyper-tuned GBC Confusion Matrix

Table XXVI
HYPER-TUNED GBC CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.86	0.97	0.91	610
1	0.92	0.70	0.80	321
accuracy			0.88	931
macro avg	0.89	0.83	0.85	931
weighted avg	0.88	0.88	0.87	931

The accuracy obtained with this the hyper-parameters tuning was better than the accuracy on base model. The percentage value of **accuracy** is **87.65%** but the **F1 Score** is **79.64%**.

*c) **K-Fold Cross Validation**:* All the results using K-Fold Cross Validation, ie, accuracy and F1 Score percentage, confusion matrix and classification report, are equal to the results obtained using **Hyper-Parameter Tuning**.

G. Light Gradient Boost Machine

Light Gradient Boosting Machine (LGBM) is a machine learning algorithm that uses tree-based learning algorithms. It has a faster training speed, higher efficiency and higher accuracy than similar models while using less memory.

*a) **Base Model**:* In the base model, we use the model as explained above without any optimization. The confusion matrix, as well as the classification report, is presented below:

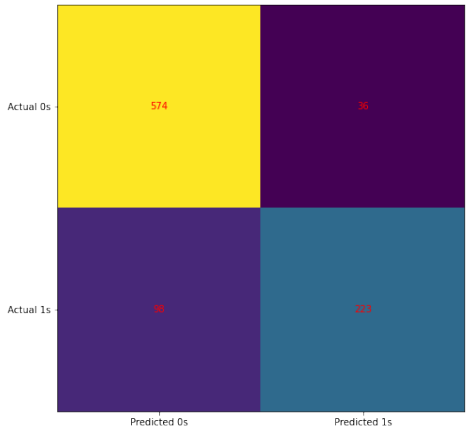

Figure 26. LGBM Confusion Matrix

Table XXVII
LGBM CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.85	0.95	0.90	610
1	0.88	0.68	0.77	321
accuracy			0.86	931
macro avg	0.87	0.81	0.83	931
weighted avg	0.86	0.86	0.85	931

The **accuracy** obtained with this model was **85.60%** but the **F1 Score** was a little bit higher with **76.89%**.

*b) **Hyper-Parameter Tuning**:* The parameters used to optimize the base model are depicted in the figure below:

Table XXVIII
ALL LGBM PARAMETERS

Parameter	Values
boosting_type	["gbdt","dart","rf"]
n_estimators	[5, 10, 15, 25, 50, 100]
max_depth	[3, 5, 7]
learning_rate	[0.01, 0.1, 0.5, 1]
num_leaves	[3,5,7,10,15]

Table XXIX
BEST LGBM PARAMETERS

Parameter	Values
boosting_type	'gbdt'
n_estimators	100
max_depth	5
learning_rate	0.1
num_leaves	5

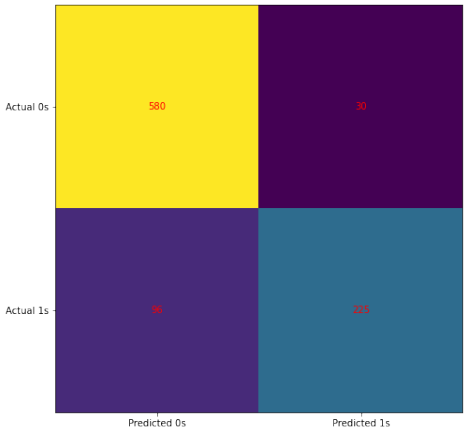

Figure 27. Hyper-tuned LGBM Confusion Matrix

Table XXX
HYPER-TUNED LGBM CLASSIFICATION REPORT

	Precision	Recall	F1-Score	Support
0	0.86	0.95	0.90	610
1	0.88	0.70	0.78	321
accuracy			0.86	931
macro avg	0.87	0.83	0.84	931
weighted avg	0.87	0.86	0.86	931

The accuracy obtained with this the hyper-parameters tuning was better than the accuracy on base model. The percentage value of **accuracy** is **86.46%** but the **F1 Score** is **78.12%**.

*c) **K-Fold Cross Validation**:* All the results using K-Fold Cross Validation, ie, accuracy and F1 Score percentage, confusion matrix and classification report, are equal to the results obtained using **Hyper-Parameter Tuning**.

VI. MODEL COMPARISON

The figure below presents a box plot with information related to the accuracy of all models used in this project.

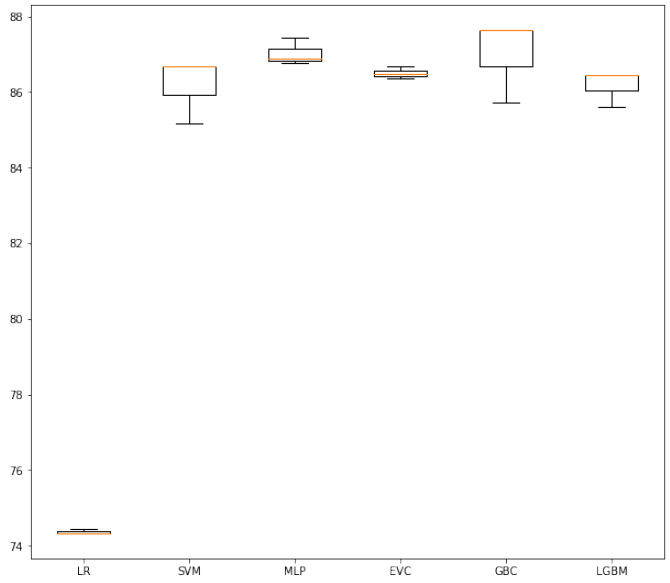

Figure 28. Model Accuracy Comparison

The table below presents the accuracy results of all the implemented machine learning models, regarding to the base, hypertuned and k-fold cross validated models:

Table XXXI
MODEL ACCURACY TABLE

Model	Base	Hypertuned	K-fold CV
Logistic Regression	74.44	74.33	74.33
Support Vector Machine	85.18	86.68	86.68
Multi-Layer Perceptron	86.89	87.43	86.78
Ensemble Voting Classifier	86.47	86.68	86.36
Gradient Boosting Classifier	85.71	87.65	87.65
Light Gradient Boosting Machine	85.60	86.46	86.46

By analysing the previous results, we conclude that the **best model** for this problem is **Gradient Boosting Classifier**, since it got the highest accuracy, whereas the **Logistic Regression** model has the **worst accuracy** of all the other models. However, the remaining models have very similar accuracies to the GBC model, with accuracy percentage values comprehended between 86-87%, which turn out to be very good results.

VII. NOVELTY AND CONTRIBUTIONS

Comparing Sonali Singh's notebook [4] results to ours, there are only two common implemented machine learning models: Logistic Regression and Gradient Boosting Classififer. Our results were **74.33%** for Logistic Regression (with Singh's one being **76.98%**) and **87.65%** for Gradient Boosting Classififer (with Singh's one being **64.53%**). In addition, our remaining models had better results with a 4% additional margin in comparison to other Singh's models.

Gagan Shandra's notebook [5] only shares one model with ours: Gradient Boosting Classifier. Shandra's GBD had an accuracy of **85.26%** while ours had **87.65%**. Shandra's remaining models had a very good accuracy, with percentage values comprehended between 84-85% in comparison to our 86-87% average (not including Logistic Regression).

Shettigar's notebook [6] inspired us to experiment the Light Gradient Boosting Machine model, since it was his best model in terms of accuracy with **83.00%**. Our LGBM had an accuracy of **86.46%**. All of our remaining models had better accuracy results than Shettigar's remaining ones.

Finally, Adytia's notebook [9] was the notebook with the biggest accuracy results when compared to the previous mentioned ones. This time, we shared four models with Adytia's notebook: LR, SVC, LGBM and GBC.

Table XXXII
ADYTIA'S NOTEBOOK MODELS AND ACCURACY

Machine Learning Model	Adytia's Score	Our Score
Logistic Regression	62.45%	74.33%
Support Vector Machines	87.87%	86.68%
Light GBM Classifier	87.69%	86.46%
Gradient Boost Classifier	86.38%	87.65%

While our LR and GBC models were better than Adytia's, his SVC and LGBM models had a slightly higher accuracy than ours. Overall, our models' accuracy results were very similar to Adytia's ones.

VIII. CONCLUSION

This was our second machine learning project and, in retrospective, we consolidated many important concepts learned in the last months. This time, we first explored and analysed the dataset, preprocessed it and implemented some models before considering the state of the art, in order to improve our critical thinking in machine learning.

Our biggest challenge throughout this project was to find out what was the best approach for data preparation in this given context and the extensive research on documentation and articles to find out how to hypertune our machine learning models.

After observing other approaches to this problem, we implemented some new models (GBC and LGBM), and the obtained results were very positive, considering that our results were overall better than the observed notebooks' models accuracies.

IX. ACKNOWLEDGEMENTS

We would like to thank Pétia Georgieva, the teacher of this course, for the clean approach when teaching complex topics of machine learning which were fundamental for the development of this project.

Furthermore, knowledge and experience sharing with colleagues was also important to learn new concepts and try out different approaches to optimize the results of this work.

REFERENCES

[1] Tejashvi (2021) Employee future prediction, Kaggle. Available at: https://www.kaggle.com/datasets/tejashvi14/employee-future-prediction.
[2] João Reis & Ricardo Rodriguez "Employee Future Prediction", GitHub. Available at: https://github.com/joaoreis16/employee_future_prediction
[3] João Reis & Ricardo Rodriguez "Heart Attack Prediction", GitHub. Available at: https://github.com/joaoreis16/heart_attack_prediction/blob/main/FAA_Project_Report.pdf
[4] sonalisingh1411 (2022) "Storytelling & Prediction of Employees Future!", Kaggle. Kaggle. Available at: https://www.kaggle.com/code/sonalisingh1411/storytelling-prediction-of-employees-future.
[5] gaganmaahi224 (2021) "11 classification algos, EDA, queries,visualization", Kaggle. Kaggle. Available at: https://www.kaggle.com/code/gaganmaahi224/11classification-algos-eda-queries-visualization.
[6] Manthanx (2021) "Employee's future ? Eda Precision [0.97]", Kaggle. Kaggle. Available at: https://www.kaggle.com/code/manthanx/employee-s-future-eda-precision-0-97.
[7] Sklearn.preprocessing.onehotencoder scikit. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html.
[8] Sklearn.preprocessing.ordinalencoder scikit. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html.
[9] Adytia M. (2022) "Employee future, EDA, cross-val 99.6% TPR", Kaggle. Kaggle. Available at: https://www.kaggle.com/code/imams2000/employee-future-eda-crossval-99-6-tpr.
[10] Harrell, F. (2017) "Classification vs. prediction", Statistical Thinking. Available at: https://www.fharrell.com/post/classification/.
[11] "Standardization (what is and why to standardize data): Data Science and Machine Learning", Kaggle. Available at: https://www.kaggle.com/general/221956.
[12] Multilayer Perceptron "Multilayer Perceptron - an overview" — ScienceDirect Topics. Available at: https://www.sciencedirect.com/topics/computer-science/multilayer-perceptron .
[13] Aliyev, V. (2020) "Gradient boosting classification explained through python", Medium. Towards Data Science. Available at: https://towardsdatascience.com/gradient-boosting-classification-explained-through-python-60cc980eeb3d.