COLEGIO DE CIENCIAS E INGENIERÍA

INGENIERÍA INDUSTRIAL

IIN-3007E - BUSINESS ANALYTICS

NRC: 3469

Final Project

SEMESTER: 202310 – Primer Semestre 2023/2024

Sofia Vargas 00217204

Said Berrezueta 00321711

Joseph Galarza 00324551

María Baldeón Calisto

DATE: 12/13/2023

Índice

Introduction

In recent years, there has been a growing concern about cardiovascular diseases, as they can impact individuals across various age groups, genders, and social backgrounds, resulting in significant health consequences. Therefore, the main aim of this project is to develop a model capable of predicting the probability of an individual developing a cardiovascular disease. This prediction will be based on an analysis of their personal characteristics and daily habits.

Due to the large amount of data, preprocessing must be performed, taking into account the nature of the variables and their outliers, as well as balancing the data to ensure a well-represented sample. Once the preprocessing is complete, models are trained using different algorithms to compare them and select the best among them.

Preprocessing

During the Exploratory Data Analysis, it was determined that the original database contained 308,854 rows and 19 columns, encompassing 5,868,226 data points of types object, int64, and float64. The pandas library treats null and NaN values similarly, and 3,189 instances of such data were identified. These data were removed during the preprocessing, as null values can impact the performance of the model, as indicated by Zhang, J. and Zhang, L. (2021). Furthermore, no negative values were detected.

To identify outliers in the categorical variables, unique values were obtained, and subsequently, those inconsistent with the variables and their classes were removed. Following this process, the variable types were changed to categorical.

To identify outliers in the numeric variables, the appropriate data types were assigned, errors were removed, and boxplots were plotted. It was observed that the variables “Height” and “BMI” had extremely high and low values, which were deemed inconsistent. Consequently, these values were removed following the interquartile range principle. According to Panda and Dash (2015), values greater than 1.5 times the interquartile range are removed. The remaining variables were kept as they provided coherent information.

In order to analyze the correlation between variables, categorical variables were transformed into dummy variables. The transformation facilitated the generation of a correlation matrix, revealing a strong correlation between the predictive variables "BMI" and

"Weight", followed by the correlation between "Weight" and "Height". Additionally, there was correlation between the "Male" class of the variable "Sex" and the variables "Weight" and "Height".

By examining a correlation plot between the predictive variables and the response variable, it was determined that the development of heart disease exhibits a stronger correlation with the presence of diabetes and arthritis. Additionally, there is a positive correlation with the age range exceeding 80 years. In contrast, the response variable shows the lowest correlation with fruit consumption, height, non-binary gender, and age ranges between 55 and 64.

To visualize and analyze the predictive variables, histograms were utilized. Exhibit 1 shows the histograms for categorical variables that represent the frequency of each class.

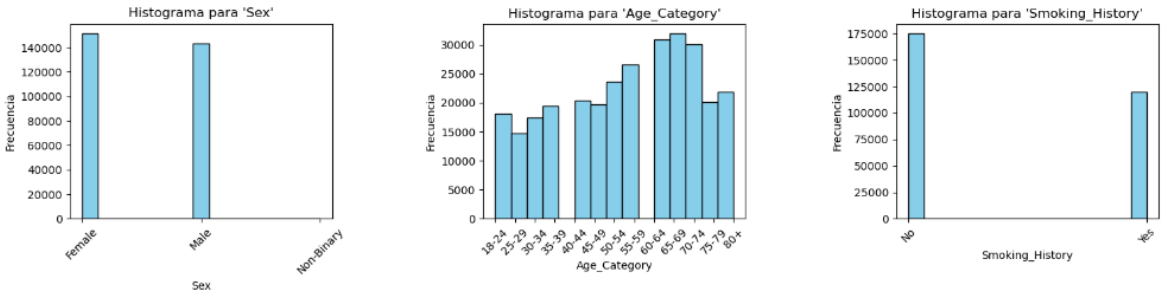


Exhibit 1: Histograms for categorical values

In contrast, histograms for numeric variables create bins and display the frequency within those ranges. Exhibit 2 shows that the variables 'Weight' and 'BMI' seem to follow a normal distribution given their bell-shaped curves; however, the other numeric variables exhibit an empirical distribution.

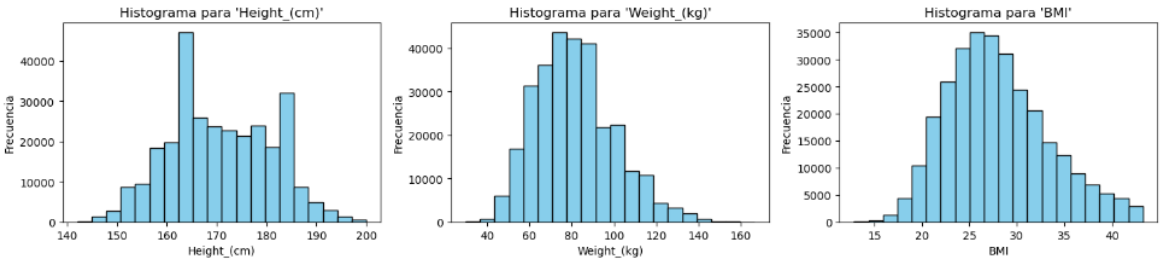


Exhibit 2: Histograms for numerical values

Unbalanced data

According to Chawla et al. (2004), an unbalanced dataset is characterized by a non-uniform distribution of the target variable, meaning there is a significant difference in the number of examples for each class of the target variable. This imbalance can pose challenges for machine learning algorithms, potentially leading to bias towards the majority class. Unbalanced datasets may result in several issues for machine learning algorithms, including reduced accuracy, increased false negatives, and heightened overfitting. There are a number of techniques that can be applied during training to reduce the effect of unbalanced datasets, including undersampling and oversampling.

As López et al. (2013) mention, undersampling involves removing examples of the majority class from the training set. This can help to reduce the bias of the machine learning algorithm towards the majority class.

On the other hand, He and Bai (2009) state that oversampling involves creating new examples of the minority class. This can help to increase the exposure of the machine learning algorithm to the minority class.

Data partition

According to Shmueli et al. (2016), in order to train a model, it is necessary to divide the data into three different datasets, this are: training set, validation set, and test set. The same author explains that the training set is typically the largest one, and commonly has between 40% to 70% of the data; furthermore, the validation set and the test set usually have the same size, meaning that the rest of the data is divided equally for both. For this project, the data was divided like this: 70% for the training set, 15% for the validation set, and 15% for the test set, which is better represented in Exhibit 3. Of course, the proportion of observations of the original data set is kept for each of the new three sets, meaning that none of them have a different proportion of positive nor negative observations.

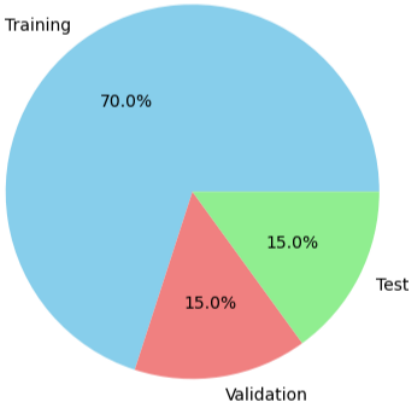


Exhibit 3: Percentages of the data partition

Balancing the data

The two balancing techniques discussed earlier are applied to the training sets. The undersampling method reduces the quantity of observations from the majority class to equal the quantity of the minority. In this case, this method is applied using the "imbalanced-learn" library, and the function "RandomUnderSampler". After applying this function, the resulting data set ends up with 16612 observations from class 0, and 16612 observations from class 1.

The oversampling method is applied using the same library, and the function 'RandomOverSampler'. With this method, the quantity of observations from the minority class equals the quantity of the majority one. After this, the resulting data set has 189904 observations from class 0, and 189904 observations from class 1.

Selecting the best method

After applying the two balancing methods, it is necessary to determine which one works better. To do this, we trained two logistic regression models, each one using the resulting data set of each method. With the two models trained, the following hypothesis test was defined:

Ho = The data fit of model A is better or equal than model B

Ha = The data fit of model B is better than model A

Model A being the one trained using the resulting set from the oversampling method, and model B using the set from the undersampling method. To compare these models, the likelihood of each model is calculated, as well the degrees of freedom. The p-value is

obtained by calculating the likelihood ratio of the models and using a goodness of fit test. With this information, the p-value obtained from the test is ‘0.0’, meaning that selecting any typical value of alpha will lead to the same conclusion. In this case, the null hypothesis is rejected, meaning that model B has better results, and that the better technique for balancing the data is the undersampling method.

Machine learning algorithms

With balanced data, four machine learning algorithms were implemented to assess their performance. Initially, RandomForest was employed due to its balanced nature, constructing decision trees using a subset of the data. In Python, the number of trees to be created is specified. According to Oshiro, Paula & Veloso (2017), having a too high number of trees helps minimize the quadratic error in splits. However, an excessively high number can be disadvantageous, so a recommended number of 129 trees is suggested. To find an optimal number, testing was conducted with 500 trees, and after 150 trees, the same conclusion was reached, as the difference did not prove significant

Another implemented model was Logistic Regression, chosen for being a widely used classification method in the field of machine learning. For its implementation, we chose to use the statsmodels library and followed the recommendation of Xueqiang, Ying, & Michael. (2017). The application of L1 or L2 penalties can lead to a loss of efficiency or overfitting due to the simplification of models, especially when data is scarce. For this reason, and given that some variables like 'Non-Binary' might not be considered for the model, we preferred not to use a penalty.

The next two models utilized hyperparameter optimization using the Grid Search technique. This method, as described by Pedregosa et al. (2007), exhaustively explores possible combinations of model parameters and selects those that yield the most optimal results. The first of these models was KNN, chosen for its simplicity and effectiveness in classification based on the nearest distance and the majority of compared or so-called neighbors (Rivest, 2000). Through hyperparameter optimization, it was determined that the optimal value for the k parameter was 2, as it provided the best result in terms of accuracy during evaluation

The final model, SVC (Support Vector Classifier), classifies data using a hyperplane with parameters C and gamma. C controls regularization (from 1 to 1000), while gamma

adjusts the kernel (from 1 to 0.0001). C values show minimal variation over broad ranges, suggesting the use of intermediate points for optimization (Cristianini & Shawe, 2000). In the project, C ([1, 100, 1000]) and gamma ([1, 0.01, 0.001]) ranges were optimized by dividing the dataset into 5 parts and evaluating each. After assessing combinations, the best model was identified with C=100 and gamma=0.001 and has a score near of 0.746.

The Support Vector Classifier (SVC) aims to find a hyperplane in an 18-dimensional space (one for each predictive variable) that optimizes the separation between individuals with heart disease and those without it. According to Fan, S. (2018), it seeks to maximize the width of the margin or the distance between the hyperplane and the closest points of each class. This model can be represented with the function:

$$h(x_i) = \begin{cases} +1 & if \quad \vec{w} \cdot \vec{x} + b \geq 0 \\ -1 & if \quad \vec{w} \cdot \vec{x} + b < 0 \end{cases}$$

This function assigns a class (-1 or 1, another binary notation) to the input data x_i following the condition and these parameters: the dot product between the weight vector $\vec{w}$ and the predictive variable vector $\vec{x}$, and the intercept term b. It is important to note that the vector $\vec{w}$ is adjusted during the model training to find the hyperplane that best separates the classes in the feature space.

Evaluation metrics

Model	Set	Accuracy	Precision	Sensitivity	Specificity
Logistic Regression	Training	0,74	0,72	0,78	0,70
	Validation	0,70	0,19	0,79	0,70
	Test	0,71	0,18	0,78	0,70
KNN	Training	0,77	1	0,54	1
	Validation	0,73	0,10	0,31	0,76
	Test	0,73	0,10	0,29	0,76
Random Forest	Training	1	1	1	1
	Validation	0,71	0,19	0,79	0,70
	Test	0,71	0,19	0,79	0,71
SVC	Training	0,79	0,76	0,84	0,73
	Validation	0,68	0,17	0,79	0,67
	Test	0,68	0,17	0,78	0,67

Exhibit 4: Metrics summary

In this case, we determined that the most costly error would be to say that someone won't develop a cardiovascular disease when he actually will (a false negative). In a medical context, this could lead to a failure to provide necessary care or interventions to an individual who is at risk. This, in turn, could result in more severe health problems, higher treatment costs, or, in extreme cases, loss of lives.

For this reason, the main metric here is sensitivity, which limits the quantity of false negatives. But also, it is required to have a good level of accuracy since it represents the percentage of correct assignations. With these considerations, and the information provided in Exhibit 4, the random forest algorithm is the best of the four. The following are the ROC curves for the selected model, and the rest of the curves and the complete matrices are shown in the annexes.

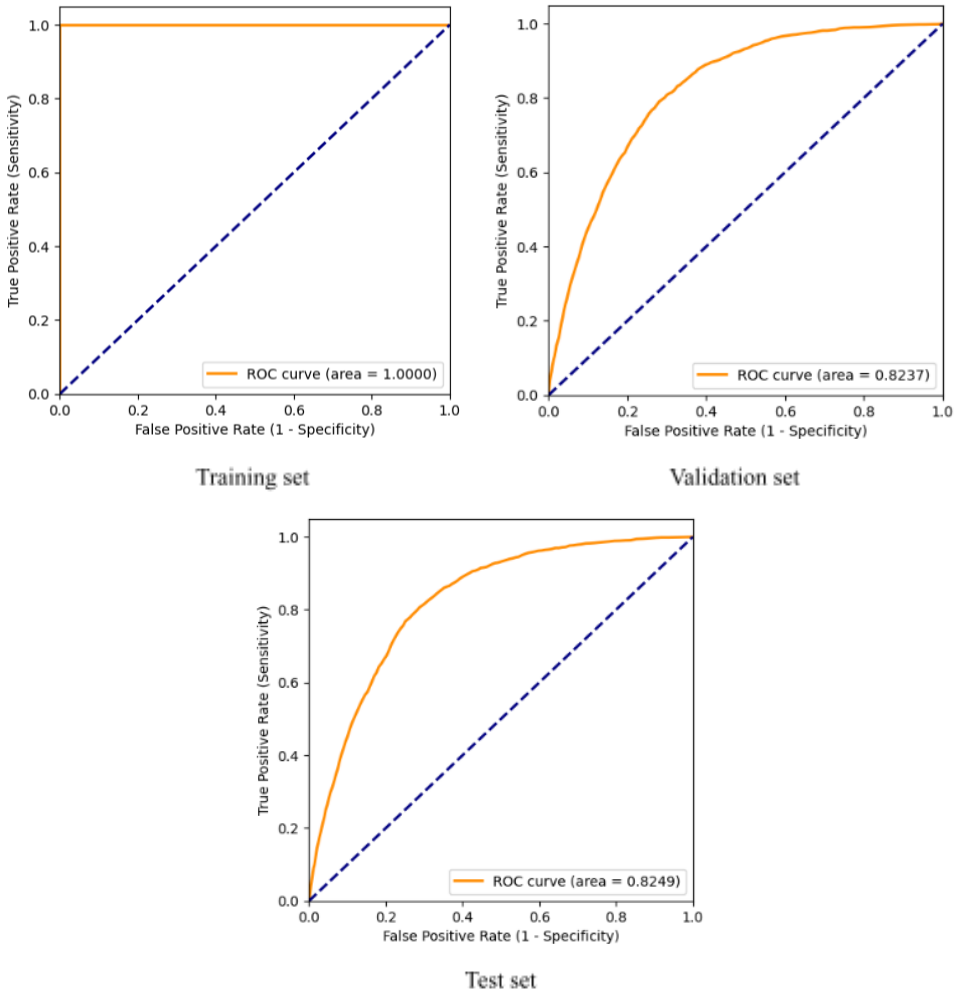


Exhibit 5: ROC curves of the training, validation, and test sets of the random forest ensemble

Just by looking at the first curve in Exhibit 5, it might seem like the model is overfitting the data, but, the curves of the validation and test data have a pretty high value as well, meaning that this model is actually balanced.

The most important predictor variables for the prediction

It is crucial to conduct a thorough analysis of the most important predictor variables for prediction. In this context, a method based on a random forest is employed, renowned for its ability to create sets of individual decision trees, each trained with a randomly sampled subset from the original training data to assess which variable contributes the least impurity. This code provides insights into the importance of each variable in relation to the response variable. Based on these findings, a graph was generated within the project to visualize the correlation of the top eight most important variables, which is presented in Exhibit 6. It was observed that these variables exhibit a more significant correlation compared to using the entire set of variables.

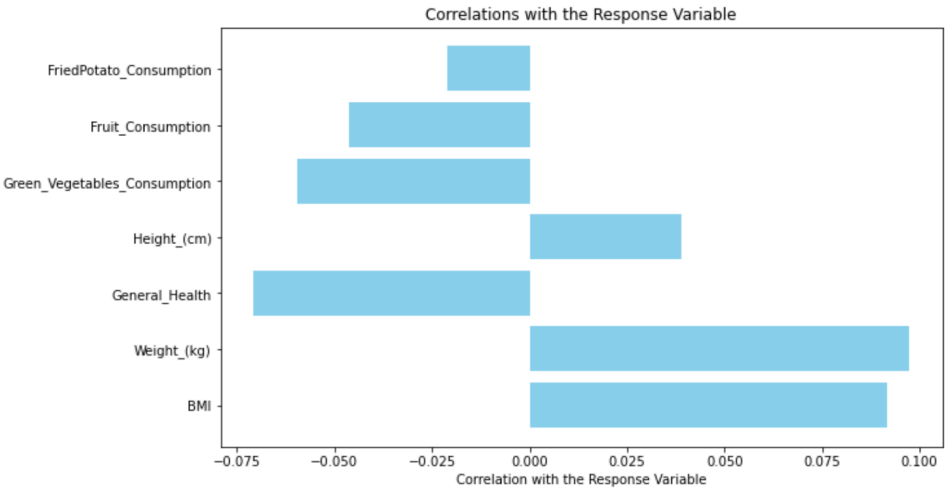


Exhibit 6: Most important variables from the random forest ensemble

Cost of implementing the machine learning model

Calculating the cost of implementing a machine learning model involves an analysis that encompasses various aspects. To determine the expenses associated with deploying the model in a business or hospital environment, it is essential to delve into factors such as hardware requirements, software licensing fees, cloud computing services, personnel salaries, and ongoing software maintenance. Therefore, we have used a Microsoft Azure pricing calculator as a starting point, which indicates the scenario that best aligns with our objective

of having the best tools for machine learning models. This includes data collection, cloud-based storage, analytics, decision-making analytics, and grouping sharing through Azure Cosmos for accessibility through web applications and mobile devices. With all these components, Exhibit 7 indicates the costs of everything mentioned.

Service category	Service type	Estimated monthly cost
Databases	Azure Synapse Analytics	$ 6.419,07
Analysis	Azure Analysis Services	$ 95,04
Analysis	Power BI Embedded	$ 1.445,83
Storage	Storage Accounts	$ 15.818,40
Databases	Azure Cosmos DB	$ 454,39
Support	Software maintenance	$ 29,00
Data analysts	Personal	$ 3.700,00
	Total	**$ 27.961,74**

Exhibit 7: Calculated costs of implementation

The benefits of this are that if you use a 3 year subscription, the cost is reduced. Additionally, there is efficiency in maintenance, comprehensive analysis, consideration of decision-making analysis, and accessibility for multiple platforms. On the other hand, as disadvantages, there is a significant initial cost, dependence on providers, in this case, Microsoft Azure, and expenses for the services.

Conclusions and limitations

- An advantage of using a random forest is that it works fine with variables that present multicollinearity (such as BMI), meaning that there is less work in the preprocessing of the data. On the other hand, implementing this type of algorithm will require more resources in the future due to the vast amount of data.
- Instead of doing an artificial balancing of the data (oversampling or undersampling), it would be better to gatter real data from the minority class, so all the data analyzed is real. This will lead to a better model.
- The variable ‘Sex Non-Binary’ could lead to false conclusions in the model, and there is not enough data of it, so, it would be better to eliminate it.
- Even though the selected model is the best for this data set, its implementation requires a significant inversion, which could be a problem depending on the situation of the Hospital.

Github

https://github.com/JosephGalarza/BA-Project-Predicting-cardiovascular-disease-FInal

References

Chawla, N. V., Japkowicz, N., & Kotz, S. (2004). Editorial: Special issue on learning from imbalanced data sets. ACM SIGKDD Explorations Newsletter, 6(1), 1-6.

Cómo Crear un Mapa de Calor de Correlaciones en Python. (s/f). CodeToDevs | Python, Django, CSS, JavaScript. Recuperado el 20 de noviembre de 2023, de https://www.codetodevs.com/como-crear-mapa-calor-correlaciones-python-corrcoef-heatmap/

Correlation and Scatterplots — Basic Analytics in Python. (s/f). Sfu.ca. Recuperado el 20 de noviembre de 2023, de https://www.sfu.ca/~mjbrydon/tutorials/BAinPy/08_correlation.html

Cristianini, N., & Shawe-Taylor, J. (2000). *Support Vector Machines*. Cambridge University Press. https://doi.org/10.1017/CBO9780511801389

Fan, S. (2018, mayo 7). Understanding the mathematics behind Support Vector Machines. *Shuzhan Fan*. https://shuzhanfan.github.io/2018/05/understanding-mathematics-behind-support-vector-machines/

Follow, I. I. (2021, noviembre 25). How to split a Dataset into Train and Test Sets using Python. GeeksforGeeks. https://www.geeksforgeeks.org/how-to-split-a-dataset-into-train-and-test-sets-using-python/

He, H., & Bai, Y. (2009). SMOTE: Synthetic minority over-sampling technique. Advances in artificial intelligence, 163-172.

Hernandez, R. D. (2022, enero 26). Bucle for en Python: Ejemplo de for i en rango. freecodecamp.org.

https://www.freecodecamp.org/espanol/news/bucle-for-en-python-ejemplo-de-for-i-en-range/

López, V., Fernández, A., García, S., Pazzani, M., & Herrera, F. (2013). An overview of classification techniques for imbalanced data: State-of-the-art and open challenges. Knowledge and information systems, 33(1), 1-40.

Oshiro, C., de Paula, R., & Veloso, P. A. S. (2017). To Tune or Not to Tune the Number of Trees in Random Forest. Journal of Machine Learning Research, 18(1), 1-40. https://www.jmlr.org/papers/volume18/17-269/17-269.pdf

Over-sampling — version 0.11.0. (s/f). Imbalanced-learn.org. Recuperado el 20 de noviembre de 2023, de https://imbalanced-learn.org/stable/over_sampling.html

Panda, D. K., y P. K. Dash. (2015). "Outlier detection for data cleaning: A survey." Computers and Informatics 42, no. 1 (2015): 1-33. https://www.sciencedirect.com/science/article/pii/S0957417414002000

Pandas: Get dummies. (s/f). Stack Overflow. Recuperado el 20 de noviembre de 2023, de https://stackoverflow.com/questions/36285155/pandas-get-dummies

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Vanderplas, J. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Rivest, R.A. (2000). The optimal value of k for k-nearest neighbors. Machine Learning, 39(1), 5-22.

Shmueli, G., Patel, N., & Bruce, P. (2016). Data mining for business analytics?: concepts, techniques, and applications in Microsoft Office Excel with XLMiner (3rd ed.). Wiley.

Termehch, A., Doskenov, B., Young Lee, G., & Alzamil, L. (2021). Data cleaning for machine learning: A survey. arXiv preprint arXiv:2109.07127. https://arxiv.org/abs/2109.07127

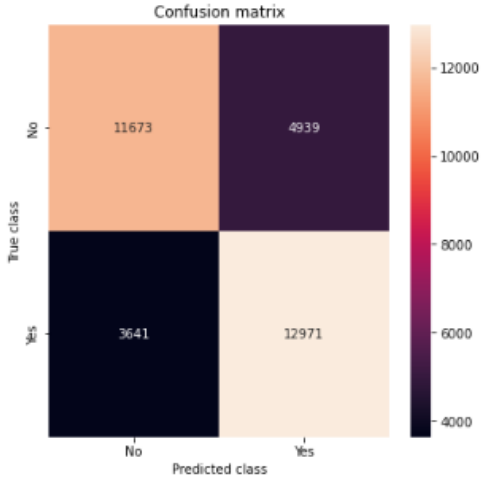
Tibshirani, R., Hastie, T., & Friedman, J. (2001). *Regularization and variable selection via the elastic net*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(2), 227-242.

Undersampling and oversampling imbalanced data. (2018, abril 9). Kaggle.com; Kaggle. https://www.kaggle.com/code/residentmario/undersampling-and-oversampling-imbalanced-data

Zhang, J., & Zhang, L. (2021). Efficient Outlier Detection for High-Dimensional Data. vol. 48, no. 12, pp. 2451-2461, Dec. 2018, doi: https://doi.org/10.1109/TSMC.2017.2718220

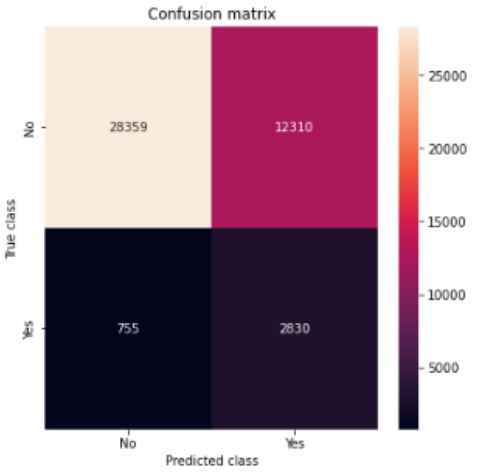
Annexes

Performance measures



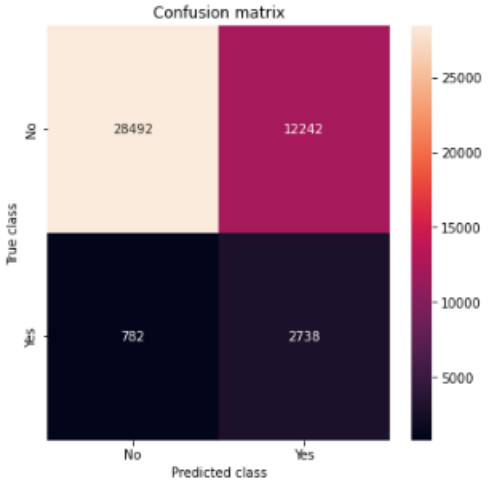
	precision	recall	f1-score	support
0.0	0.76	0.70	0.73	16612
1.0	0.72	0.78	0.75	16612
accuracy			0.74	33224
macro avg	0.74	0.74	0.74	33224
weighted avg	0.74	0.74	0.74	33224

Training set



	precision	recall	f1-score	support
0.0	0.97	0.70	0.81	40669
1.0	0.19	0.79	0.30	3585
accuracy			0.70	44254
macro avg	0.58	0.74	0.56	44254
weighted avg	0.91	0.70	0.77	44254

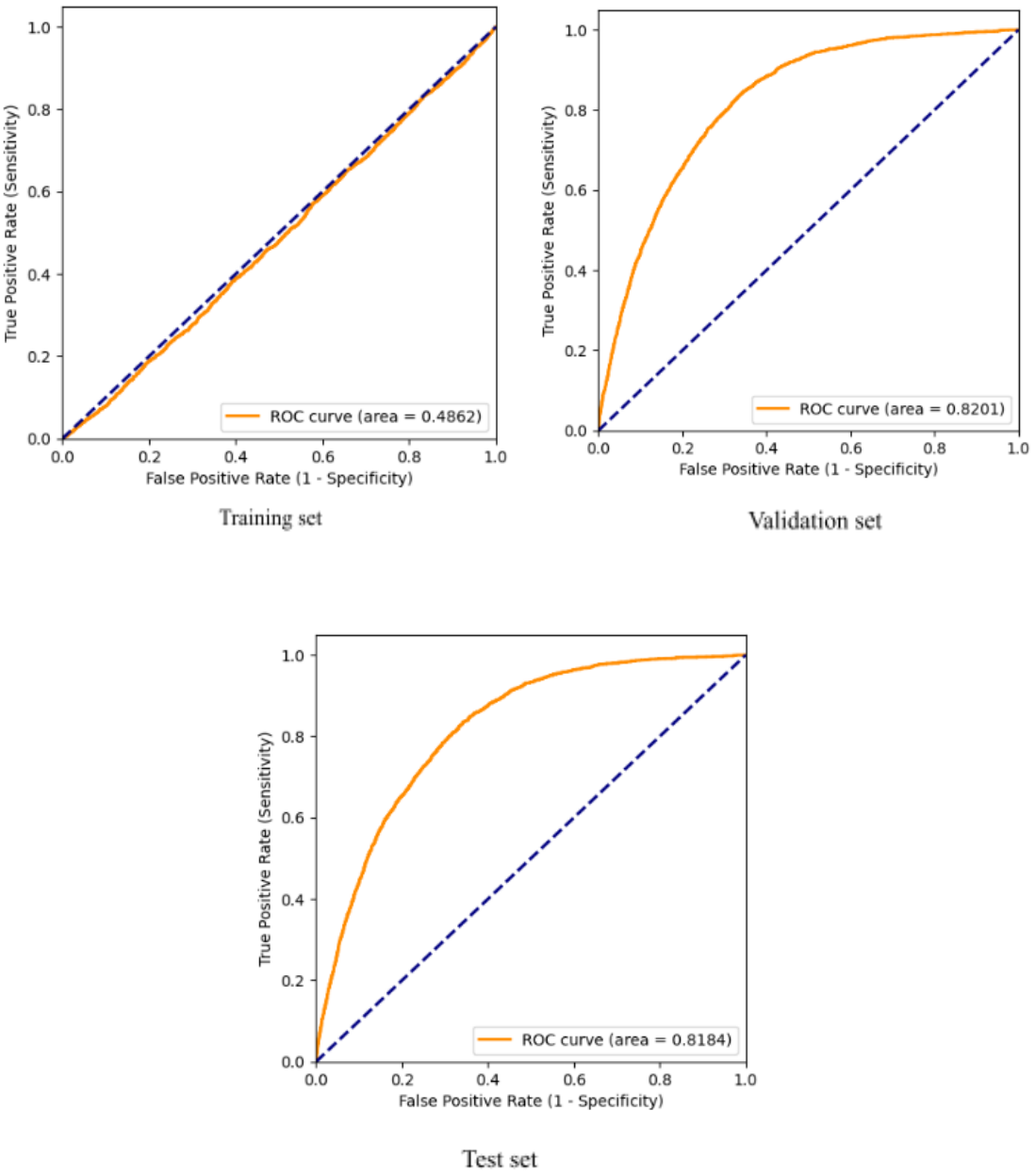
Validation set



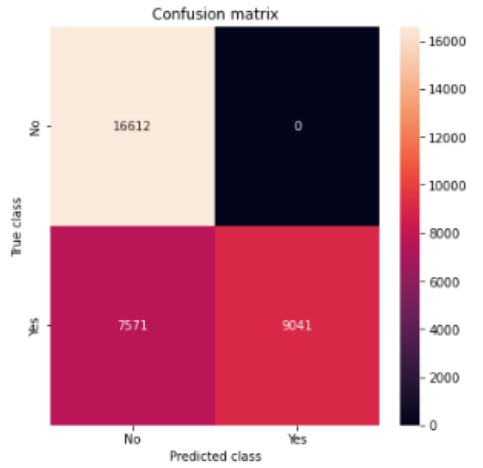
	precision	recall	f1-score	support
0.0	0.97	0.70	0.81	40734
1.0	0.18	0.78	0.30	3520
accuracy			0.71	44254
macro avg	0.58	0.74	0.55	44254
weighted avg	0.91	0.71	0.77	44254

Test set

Annex 1: Confusion matrices of the logistic regression model

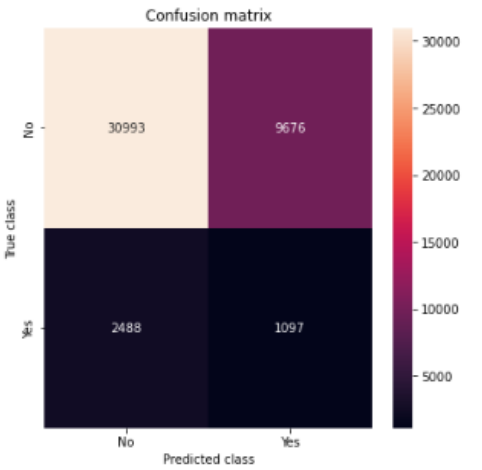


Annex 2: ROC curves of the logistic regression model



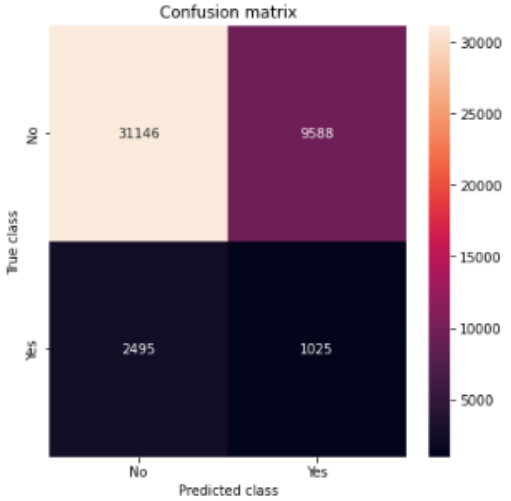
	precision	recall	f1-score	support
0.0	0.69	1.00	0.81	16612
1.0	1.00	0.54	0.70	16612
accuracy			0.77	33224
macro avg	0.84	0.77	0.76	33224
weighted avg	0.84	0.77	0.76	33224

Training set



	precision	recall	f1-score	support
0.0	0.93	0.76	0.84	40669
1.0	0.10	0.31	0.15	3585
accuracy			0.73	44254
macro avg	0.51	0.53	0.49	44254
weighted avg	0.86	0.73	0.78	44254

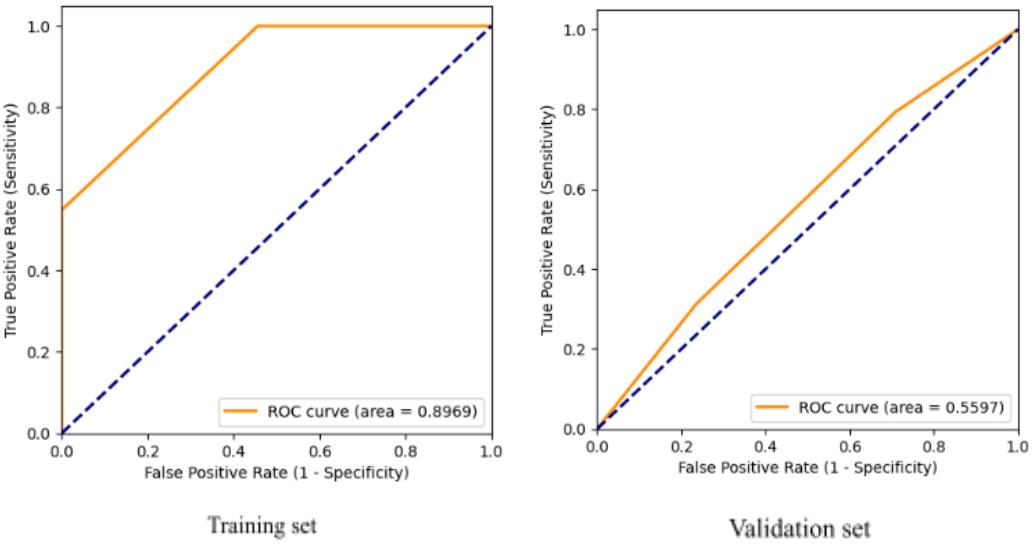
Validation set

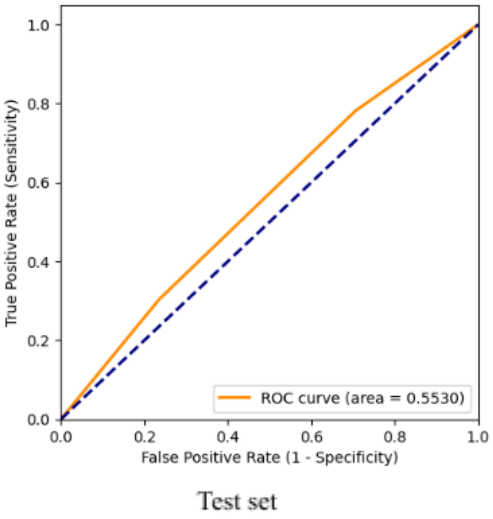


	precision	recall	f1-score	support
0.0	0.93	0.76	0.84	40734
1.0	0.10	0.29	0.15	3520
accuracy			0.73	44254
macro avg	0.51	0.53	0.49	44254
weighted avg	0.86	0.73	0.78	44254

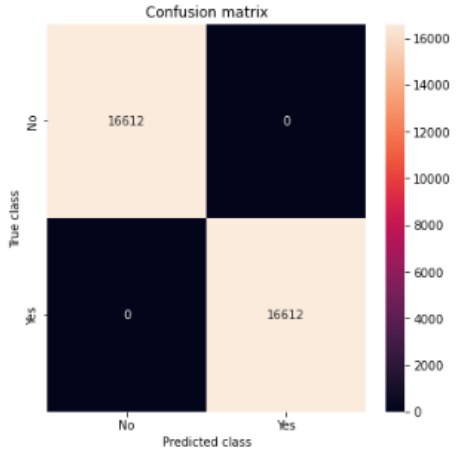
Test set

Annex 3: Confusion matrices of the KNN model



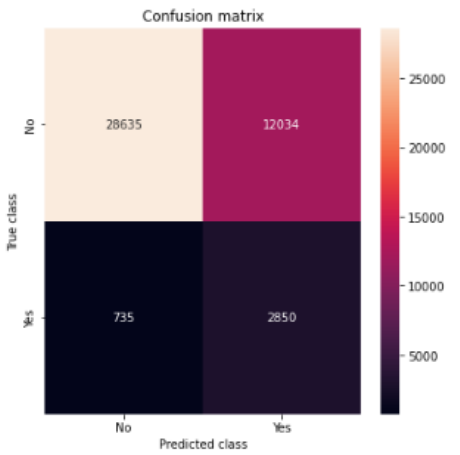


Annex 4: ROC curves of the KNN model



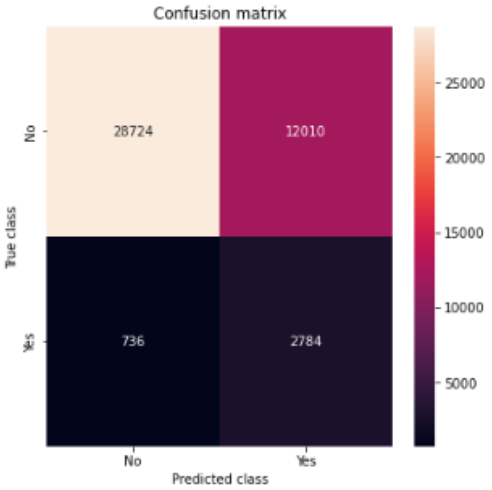
	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	16612
1.0	1.00	1.00	1.00	16612
accuracy			1.00	33224
macro avg	1.00	1.00	1.00	33224
weighted avg	1.00	1.00	1.00	33224

Training set



	precision	recall	f1-score	support
0.0	0.97	0.70	0.82	40669
1.0	0.19	0.79	0.31	3585
accuracy			0.71	44254
macro avg	0.58	0.75	0.56	44254
weighted avg	0.91	0.71	0.78	44254

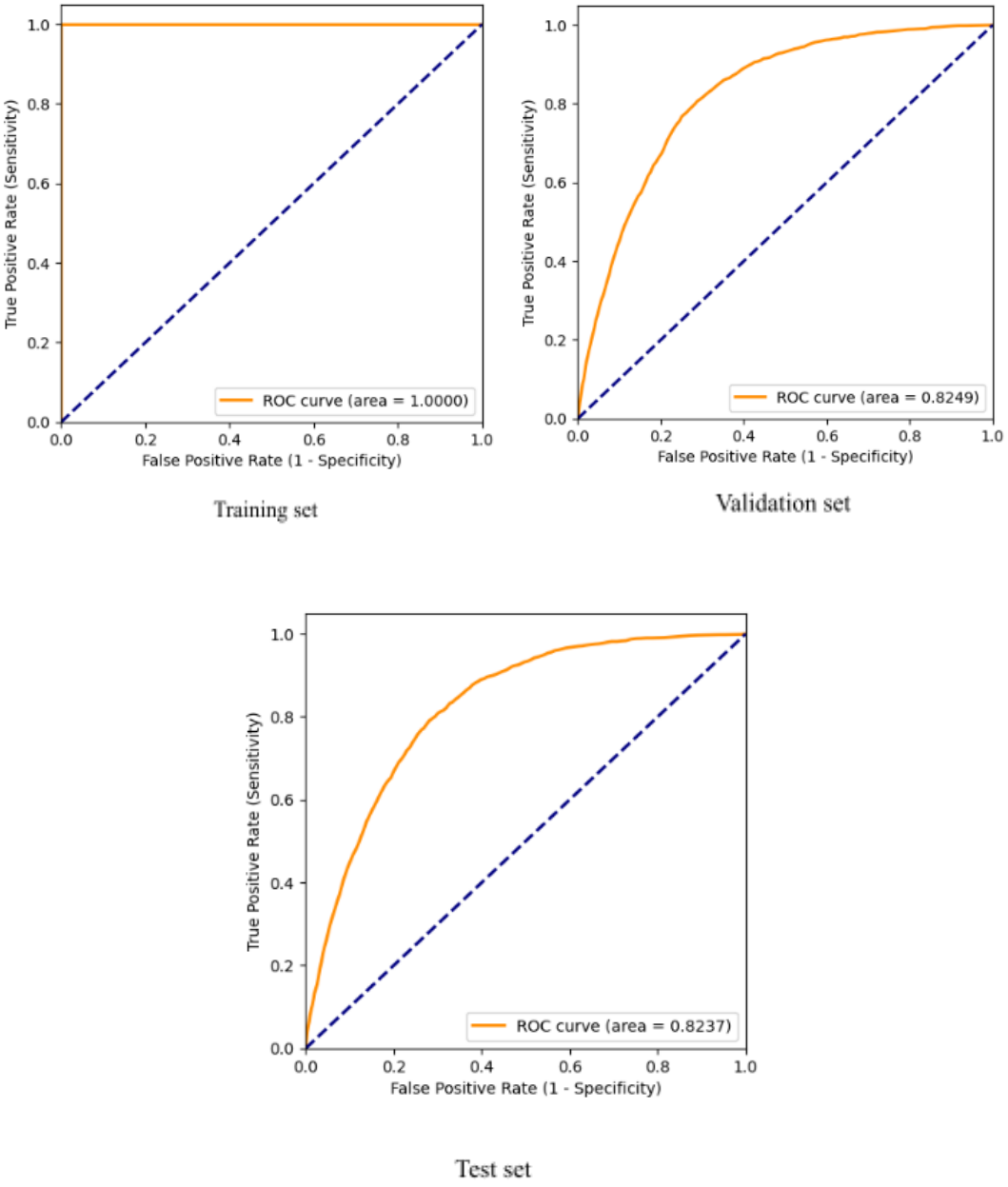
Validation set



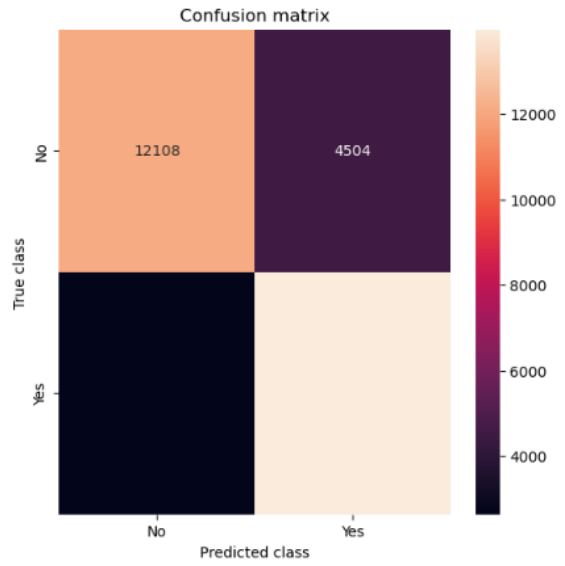
	precision	recall	f1-score	support
0.0	0.98	0.71	0.82	40734
1.0	0.19	0.79	0.30	3520
accuracy			0.71	44254
macro avg	0.58	0.75	0.56	44254
weighted avg	0.91	0.71	0.78	44254

Test set

Annex 5: Confusion matrices of the random forest ensemble

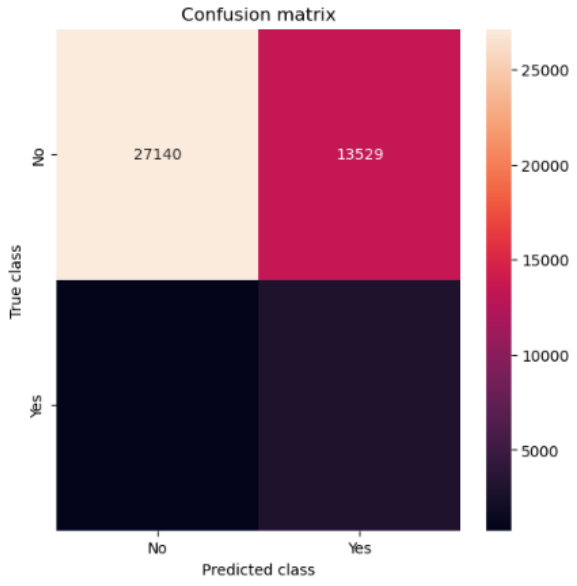


Annex 6: ROC curves of the random forest ensemble



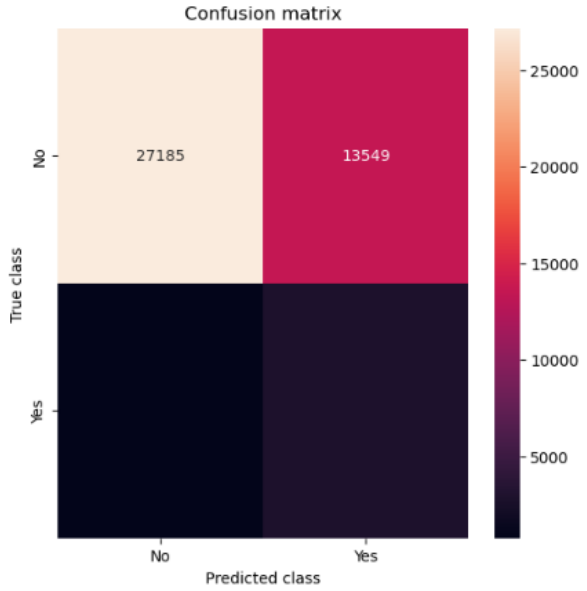
	precision	recall	f1-score	support
0.0	0.82	0.73	0.77	16612
1.0	0.76	0.84	0.80	16612
accuracy			0.79	33224
macro avg	0.79	0.79	0.78	33224
weighted avg	0.79	0.79	0.78	33224

Training set



	precision	recall	f1-score	support
0.0	0.97	0.67	0.79	40669
1.0	0.17	0.79	0.28	3585
accuracy			0.68	44254
macro avg	0.57	0.73	0.54	44254
weighted avg	0.91	0.68	0.75	44254

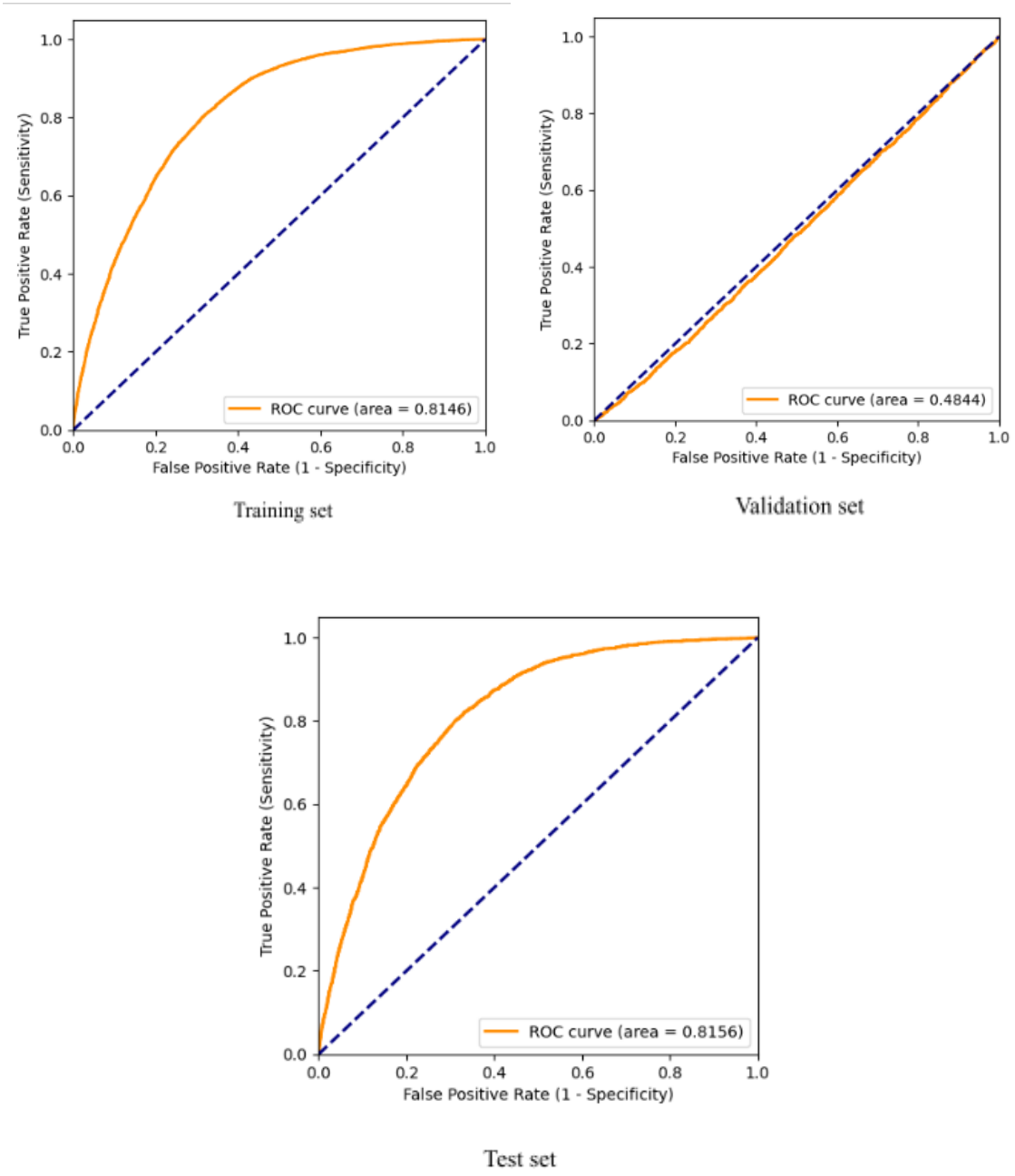
Validation set



	precision	recall	f1-score	support
0.0	0.97	0.67	0.79	40734
1.0	0.17	0.78	0.28	3520
accuracy			0.68	44254
macro avg	0.57	0.73	0.53	44254
weighted avg	0.91	0.68	0.75	44254

Test set

Annex 7: Confusion matrices of the SVC model



Annex 8: ROC curves of the SVC model