Machine Learning Assignment

880083-M-6 / Master DSS

Group: 25

Mathilde Hellinga - 2103865

Romy den Hartog - 2098947

Yasemin Ozturk - 2104098

Wafa Mohamed - 2097778

Date: 20-06-2023

Chapter 1: Data

The data representation consists of 22 features and 12,205 instances, with no missing or duplicate data. The target variable is 'Transaction' (binary classification) with binary, multiclass and continuous variables. For all continuous variables we checked the outliers, and we concluded all features are right-skewed (Appendix A). The EDA analysis shows that this data is highly imbalanced: the number of negative instances (84.37%) outnumbers the positive numbers (15.63% positive transactions). Two features were non-numeric: Customer_Type and Weekday, so we had to encode them. Dummy coding was used to encode Customer_Type because it lacks an inherent ordering relationship, and Weekday is a Boolean variable, so we encoded it to 0 and 1. The data now has 24 features. After EDA, the data was split: 80% of the data was reserved for training and validation and 20% was reserved as test data. Because machine learning models can be sensitive to outliers and variables that have different scales, featuring scaling was applied after the train-test split by using the sklearn function StandardScaler. This data preprocessing helps to avoid some features with large values dominating the model and ensures that the used model is effective.

After the training data was scaled, the Correlation matrix (Appendix B) was used for identifying feature correlations. Highly correlated features, which are linearly dependent and have a similar impact on the target variable, were removed if their correlation coefficient exceeded 0.5. After features that satisfied this threshold were dropped (4 continuous features in total), a Decision Tree was used to further investigate the importance of features.

The features were selected based on their feature importance obtained from the decision tree, all features with a Gini threshold smaller than 0.01 were dropped. This means that only 11 of 24 features are going to be used for training and testing the chosen model (Appendix C).

To split the training data into training and validation data, stratified k-fold with 10 folds is used. A stratified k-fold is chosen because the data is highly imbalanced. A stratified k-fold is an extension of a regular k-fold and is recommended when data is imbalanced and the majority class belongs to the negative class (Nnamoko & Korkontzelos, 2020). This method ensures that class distribution in each fold is the same as the initial data, 15.63% of this data is true transactions. When stratified k-fold is applied each created fold will have the same percentage of transactions. For the hyperparameter tuning stratified k-fold is combined with RandomizedSearchCV. Class weight was used to improve model performance (Neural Network and Logistic Regression). This method reduces biased predictions towards the majority class without changing the class ratio, as oversampling and under sampling strategies do. Class weight penalizes misclassification of the minority class by giving it more weight while giving the majority class less weight. (Singh, 2020).

Chapter 2: Models

In this study, three models— K-Nearest Neighbors (KNN), Logistic Regression and Neural Network (NN)—have been selected to tackle the binary classification task. In Appendix D all the models and their assumptions are described. Since we are uncertain whether our prediction task is a linear or non-linear problem, multiple models with different approaches were explored.

Due to its simplicity in handling classification problems, KNN serves as a baseline model. KNN is suitable for situations where there may not be a clear pattern in the data because it does not make any assumptions about how the data is distributed or the connections between the features. Due to its non-linearity, it can handle complex decision boundaries by capturing complex relationships between features and the target variable. Because of its interpretability, Logistic Regression was chosen as one of the models. It makes it simpler to comprehend the underlying relationships by revealing how each attribute affects the target variable. Additionally, Logistic Regression assumes that there is a linear relationship between the features and the target variable's log-odds, which may be suitable in some circumstances. Deep learning NN models were chosen because they recognize complex patterns and

non-linear data relationships. Compared to other models, NN can learn hierarchical representations and identify intricate structures. They are ideal for handling complex and large datasets because of their adaptability and suitability for challenging classification issues.

Based on each model's advantages and the features of the dataset, these three models were selected. KNN provides a simple and non-linear perspective on handling classification problems, Logistic Regression offers interpretability and NN capture complicated interactions. With the use of several models, we intend to examine different approaches, take advantage of each one's strengths, and gain a thorough grasp of the data while also enhancing the performance of the overall prediction model. KNN was selected as the baseline model due to the model's simplicity.

Chapter 3: Model training and architecture experimentation

Model 1: K-Nearest Neighbors

For the KNN model we chose the number of splits to be 10, to keep it generalizable. Hyperparameters are tuned for a range between 0 and 30 neighbors with 2 steps in between. Also, three different distances to measure between the neighbors are tested (Euclidean, Minkowski and Manhattan). For the hyperparameter tuning of the weights of the model, we distinguish between uniform (equal) and distance (inverse). After training this model, we chose the best combination given the highest resulting f1 score. The selection of the metrics and K values impacts the KNN model performance. When using ‘distance’ weights in ‘Euclidean’ and ‘Minkowsk’ metrics, the F1 score initially rose as K increased until it peaked at K = 10. The obtained f1 scores slightly decreased after that. The highest f1 score obtained in both situations was 0.517. The f1 score initially rose until K = 6, reaching its peak at 0.495 with the 'Manhattan' metric and 'distance' weights.

Model 2: Logistic Regression (using SGDClassifier)

The SGDClassifier was selected for its suitability with large datasets. 'log' as the chosen loss function corresponds to Logistic Regression, which is ideal for our data. The regularization strength in the model was controlled using 'alpha', and values of 0.0001, 0.001, 0.01, and 0.1 were tested. For learning rate, we tested constant', 'optimal' and 'invscaling' and for eta (stepsize) we tried 0.01, 0.1 and 1.0. Grid search was utilized to select the optimal hyperparameters for our logistic regression model. The chosen hyperparameters included an eta of 1.0 and an alpha of 0.01 using 'optimal' (f1: 0.630). The performance of the model is sensitive to the variations of hyperparameters. Reducing regularization improved predictive abilities, as evidenced by higher f1 scores and lower alpha values. Stronger regularization with higher alpha values resulted in lower f1 scores indicating poorer model performance. Increasing the eta value generally led to better overall performance; however, for eta = 1.0 with an alpha of 0.1 (f1: 0.371), a decline in f1-score was seen. The 'optimal' learning rate had an inconsistent impact on the performance of the model across all hyperparameter variations.

Model 3: Neural Network

For the Neural Network, we tuned in three different stages: number of neurons in each layer (2 layers), learning rate, dropout, batch size and epochs. The optimizer Adam (tends to converge faster) and the kernel initializer normal were used for all stages. For the first stage, we only tuned the number of neurons: [5, 10, 15, 20, 25, 30], this gave us an f1_score of 0.61 and std_test_score of 0.032 and the best parameters were {'hidden_1': 30} and {'hidden_2': 10}. For the second stage, we tuned the number of neurons: [5, 10, 15, 20, 25, 30], dropout: [np. linspace (0, 0.1)], and the learning rate: [np. linspace (0, 0.1)]. Result was an f1_score of 0.62 and std_test_score of 0.033. The best parameters were {'learning_rate': 0.43061224489795924}, {'hidden_1': 30}, {'hidden_1': 20} and {'dropout': 0.07551020408163266}. For the third stage, we decided to tune the number of neurons to be: [5,10, 15, 20,25,30], dropout: [np. linspace (0, 0.1)], and the learning rate: [np. linspace (0, 0.1)], batch_size: list

[(range (180, 400, 1))], and epochs: [40, 60,80,100, 120]. Here we got an f1_score of 0.66 and a std_test_score of 0.025, the best parameters were: {'learning_rate': 0.43061224489795924, 'hidden_2': 30, 'hidden_1': 5, 'epochs': 100, 'dropout': 0.06326530612244899, 'batch_size': 347}. For the test data, we selected the model that we got in stage 3. This model has the highest f1 score, and the lowest standard deviation. Low standard deviation means the model's performance is more consistent across different k-folds.

Chapter 4: Results + Discussion

The evaluation metrics used were f1 score, precision, and recall assessing the model's performance on highly imbalanced data. Accuracy was deemed inappropriate due to its potential for false assumptions in such cases. However, the f1 score provided a more reliable measure by balancing precision and recall. In Table 1 of Appendix E, KNN (the baseline model) achieved a precision of 0.73, recall of 0.44, and an f1 score of 0.55. Although it had higher precision than other models tested, its recall and overall f1 score remained relatively low indicating insufficient prediction of positive instances and a notable number of false negatives (Appendix F). This might be because we have not changed the decision threshold to account for class imbalance hence the majority class dominating during the majority voting. The slightly higher f1 test score (0.55) compared to the validation score (0.51) implies that the model is somewhat overfitting. In contrast, both Logistic Regression and NN show good generalization. The f1 test score of 0.62 for Logistic Regression is the same as the f1 validation score of 0.62, indicating robust performance and good generalization. Similarly, the NN achieved a f1 score of 0.65 on the test data, about the same as the validation data (0.66), suggesting consistent and reliable performance on unseen data. You can also see that both Logistic Regression and NN have a high recall score, indicating a high prediction of positive instances. This can be attributed to the class weight that was applied to address the imbalanced data distribution, prioritizing the correct prediction of the minority/positive class. However, both models have lower precision, suggesting a higher rate of false positives. Further tuning the other hyperparameters could potentially capture more complex patterns to improve the precision scores of the NN and Logistic Regression.

The results suggest that NN and Logistic Regression perform well in terms of f1, recall, and generalization. However, precision can be improved by tuning parameters further and exploring different techniques for imbalanced data. A stratified k-fold with a value of 10 was used to train the models which helped maintain class distribution and enhance the model's ability to generalize on unseen data through the assessment of multiple subsets.

Transparency is ensured as we documented hyperparameters (e.g., epochs, neurons), feature selection process, threshold for selecting features, and all project packages mentioned in our py. document. These practices enhance transparency and make it easier for others to reproduce our work.

Workload planning

We had real-life weekly meetings that we all attended. For the first phase, we all investigated the data through Exploratory Data Analysis. As a group, we made decisions on various aspects such as scaling methods and feature selection. Yasemin and Wafa focused on binary classification models, selecting the most suitable ones. Romy and Mathilde took the first steps of EDA. Wafa made the correlation matrix, and we all chose and deleted the highly correlated combinations. Romy wrote the first two chapters of the report. Mathilde kept track of the planning and division of roles. Yasemin made the code and the report for KNN, Mathilde and Romy for Logistic Regression and Wafa for the NN. Wafa compared the models and wrote the results and discussion. Yasemin made the flowchart of the methodology (Appendix G: Flowchart) and made the layout for the report.

Appendix A: Histogram continuous variables

Figure 1: Histogram Account Page Time

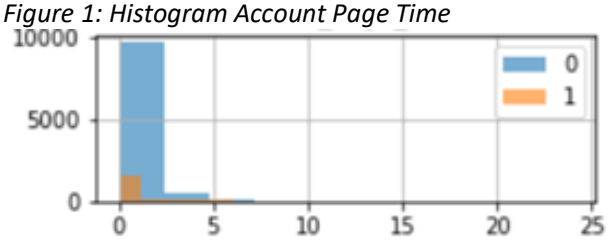


Figure 2: Histogram Account Page

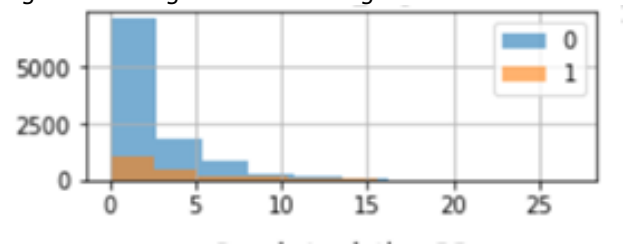


Figure 3: Histogram Info Page

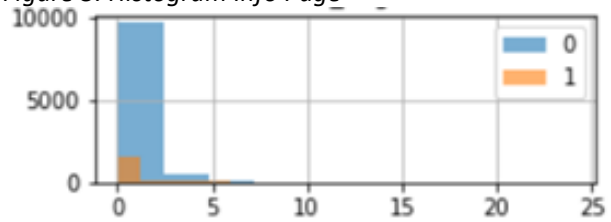


Figure 4: Histogram Info Page Time

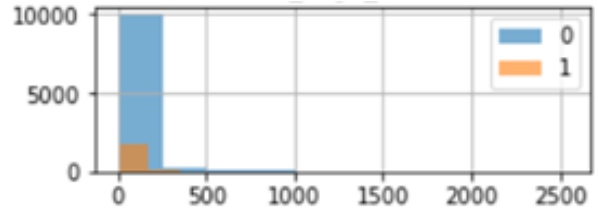


Figure 5: Histogram Google Analytics BR

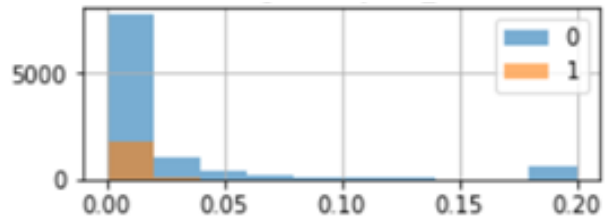


Figure 6: Histogram Google Analytics ER

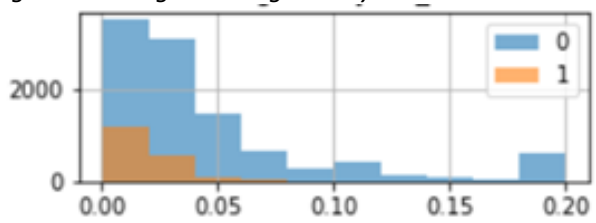
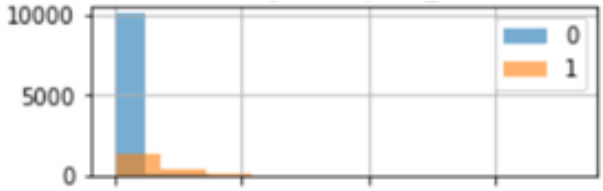
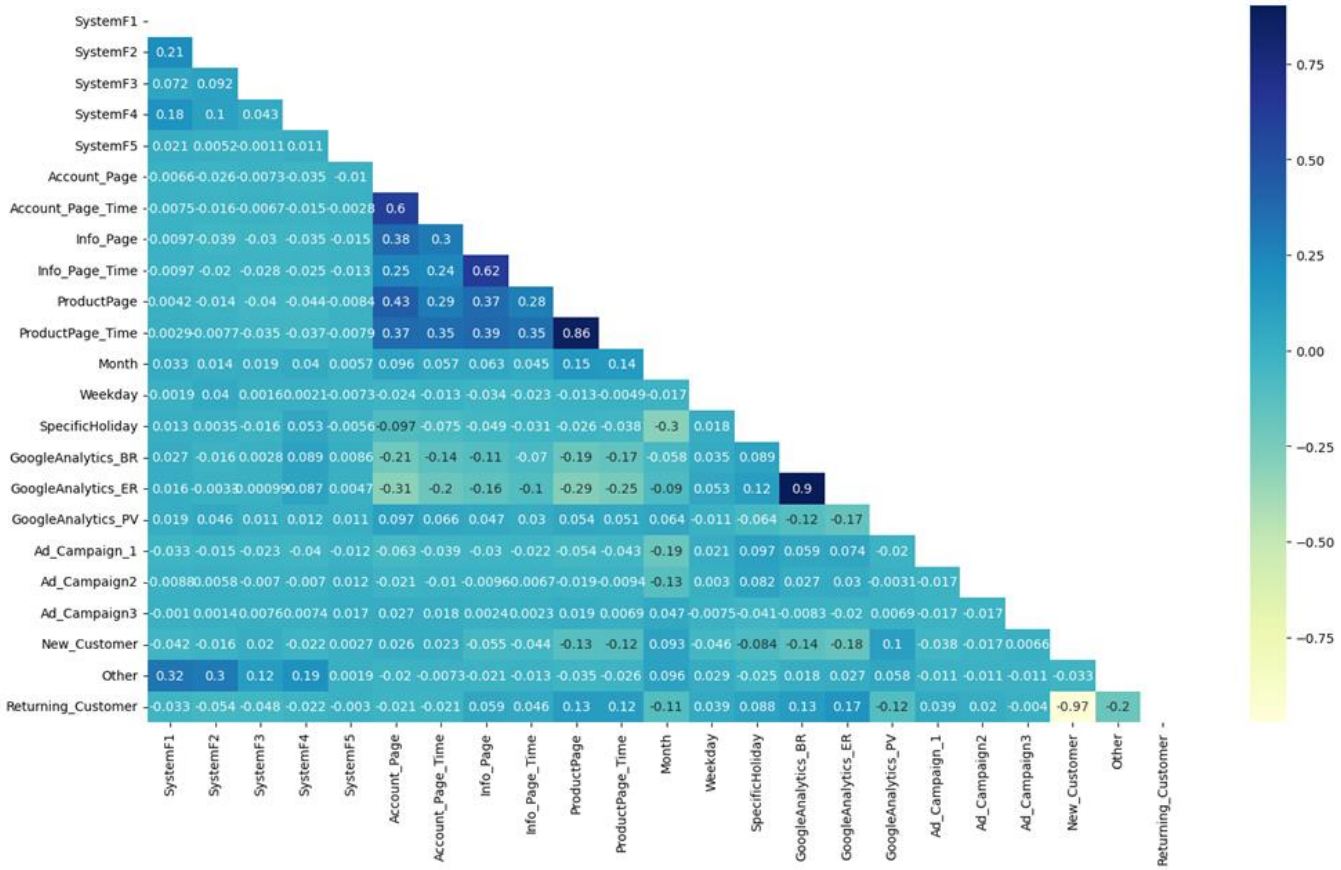


Figure 7: Histogram Google Analytics PV



Appendix B: Correlation Matrix

Figure 2: Correlation Matrix between variables



Appendix C: Gini-scores Decision Tree

Ad_Campaign_1	0.0004612122620023806 4
Other	0.0005047450520383205
Ad_Campaign2	0.001098407152973331
SpecificHoliday	0.004062380180929495
Returning_Customer	0.0046445175319311015
New_Customer	0.004699857546935573
Ad_Campaign3	0.005454009191164911
Weekday	0.008601106851498246
SystemF1	0.0191669217725956339
SystemF2	0.020597443320762594
SystemF3	0.03327366879524307
SystemF5	0.035194831124789216
SystemF4	0.038124744499669634
Month	0.043648608072812005
Info_Page_Time	0.04663635844171157
Account_Page	0.05155890221761881
ProductPage	0.10877696856971816
GoogleAnalytics_ER	0.11315760781514352
GoogleAnalytics_PV	0.4603377096004626

Table 3: Gini-scores per variable

Appendix D: Binary classification models

Logistic Regression:

In Logistic regression is a linear model for binary classification problems and the value of the independent variable are linearly combined. The linear combination of the independent variables is transformed by sigmoid function to obtain the predicted probability of binary outcome. This model can be prone to overfitting, low bias and high variance (Zhang, Geisler, Ray, & Xie, 2021). Assumptions: large sample size, no outliers, binary dependent variable, distribution has logistic function, and no multicollinearity between independent variables.

Naïve Bayes:

A linear classifier model that is based on Bayes Theorem. Probabilities of events happening given evidence of another. This model has high bias and low variance. Zero Probability Problem, meaning attribute value doesn't occur in every class gets assigned a 0. Performs better than Logistic Regression, if features are correlated (Varghese, 2018). Assumptions: independence of features, equal importance of features, categorical input variable, small dataset performs better, and the model is generative.

Naïve Bayes vs Logistic Regression:

Naïve bayes is generative model, while Logistic regression is discriminative model. Both perform well on small datasets; however Logistic regression performs better than Naïve Bayes when there is collinearity among the features because Naïve Bayes assumes feature independence (Varghese, 2018).

K Nearest Neighbors:

Classify new instances based on neighbors of records. Highly similar ones are places together, similarity based on distance functions such as Euclidean and Manhattan. For every prediction the distance to entire training set is computed, cannot predict large number of records simultaneously. Sensitive to scale of dataset (Bzdok, Krzywinski, & Altman, 2018). Assumptions: classes are clearly separated, number of records in the training set need to be large, non-parametric, and each sample in training data has the same number of attributes/dimensions. Recognizing that KNN can become biased toward the majority instances of the training space in datasets with imbalances is important. This bias arises from the fact that KNN bases predictions on the class labels of the k nearest neighbors, and the algorithm is prone to favor the dominant class in the neighborhood (Varghese, 2018). This bias arises from the fact that KNN bases predictions on the class labels of the k nearest neighbors, and the algorithm is prone to favor the dominant class in the neighborhood (Varghese, 2018).

Logistic regression vs KNN

Logistic regression is parametric; assumes specific relationship between the features and the target variable, KNN is non-parametric; does not make any assumptions about the underlying data distribution. Further Logistic Regression is faster than KNN; KNN involves computing distances between instances can be computationally if the dataset is large, in contrast Logistics regression training involves solving an optimization problem which is generally faster. Logistics regression assumes a linear decision boundary and KNN non-linear boundary (Varghese, 2018).

Decision Trees and Random Forest:

Tree where each node is instance matching with rule up to that point, one feature selected to split data as much as possible. Biased towards splits on features with large number of levels. Prone to overfitting. A form of decision where multiple decision trees is combined Random Forest. This variant of decision is robust and can better handle overfitting than the traditional decision tree (Prem., n.d.).

Assumptions: records are distributed recursively, feature values are preferred to be categorical, don't need to encode categorical variables, not too large tree, feature scaling is not necessary, and can handle overfitting (Random Forest).

Neural Networks:
Neural networks are a type of machine learning algorithm it tries to mimic the human brain. Neural networks have an input layer, one or more hidden layers and output layers. The input layer receives raw the data directly and the data goes from the input layer to the next layer, the hidden layer. The input layers apply different non-linear functions to process the data. Output layers are the same as hidden layers, the only difference is that the output layer generated the result/prediction. To train the neural network algorithm the so-called backpropagation is applied which is based on gradient descent (Prem., n.d.). Assumptions: doesn't assume linear relationship, can be used for unstructured data, not affected by outliers a lot, large data set without missing data, and suitable for binary and multiclass.

Support Vector Machines:
It is a supervised machine learning model that can be used for different fields such as text classification, image classification and binary classifications. It applies a hyperplane that cares that the problem is linearly separable. This model can handle outliers because it uses soft margin value C, which tells the model the optimal value to avoid misclassification, high C value implies high penalty for outliers (Prem., n.d.). Assumptions: doesn't assume linear relationship, space must be linearly separable, small dataset with a lot of features, suitable for both regression and classification, and training might take too long if the data is large.

KNN vs SVM:
SVM is computationally better than KNN, it can also be better interpreted, because KNN accepts all kinds of boundaries it is result might be less interpretable. However, SVM can identify only limited patterns compared to KNN. Another advantage of SVM compared to KNN is memory, because SVM only needs small subset of data, and KNN requires higher computation because it uses all the available datasets (Bzdok, Krzywinski, & Altman, 2018).

SVM vs Neural Network:
SVM requires a small dataset compared to Neural Networks. The performance or accuracy of Neural Network is higher than SVM because it has different hyperparameters that can be tuned such as epoch, training and loss function (Prem., n.d.).

Appendix E: Classification report for models

	KNN (BASELINE)	***LOGISTIC REGRESSION***	***NEURAL NETWORK***
PRECISION	**0.73**	0.54	0.54
RECALL	0.44	0.73	**0.81**
F1	0.55	0.62	**0.65**

Table 4: Performance of the models on the overall test set. Bold value: highest score on test data

Appendix F: Confusion Matrix

Figure 5: Confusion matrix K-NN model

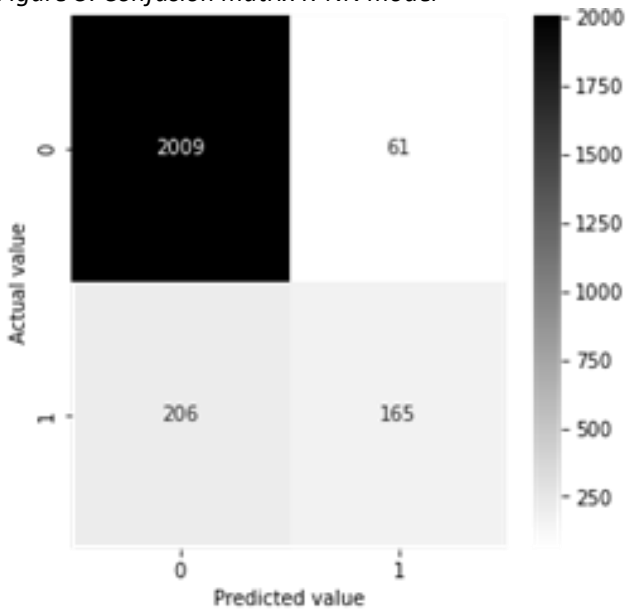


Figure 2: Confusion matrix Logistic Regression

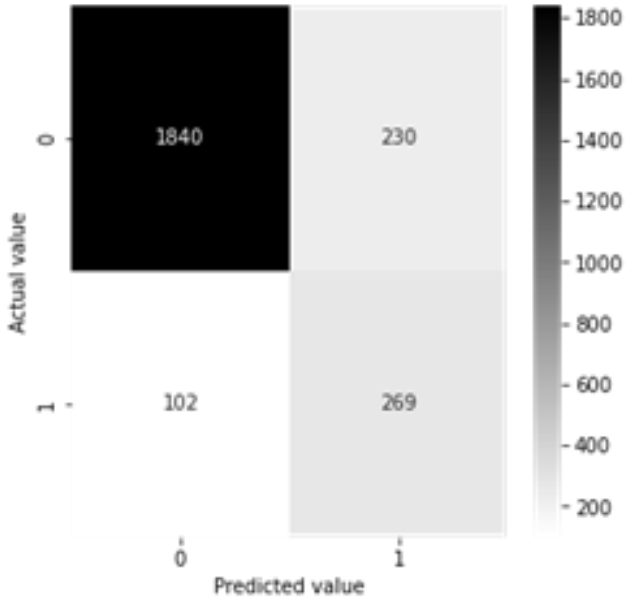
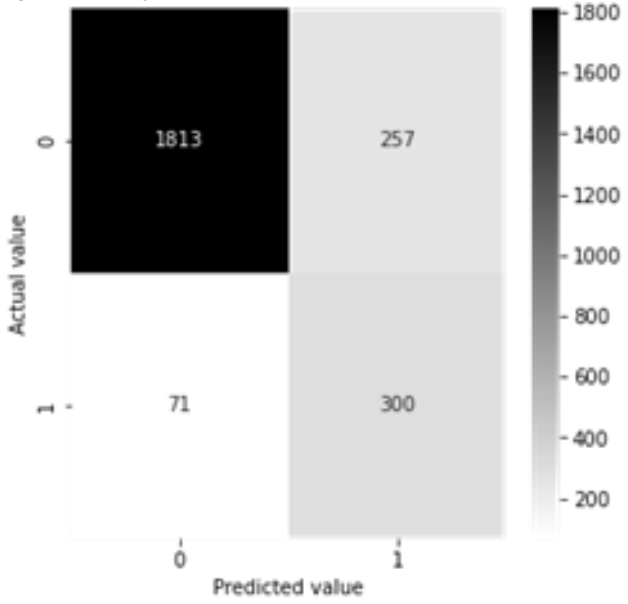
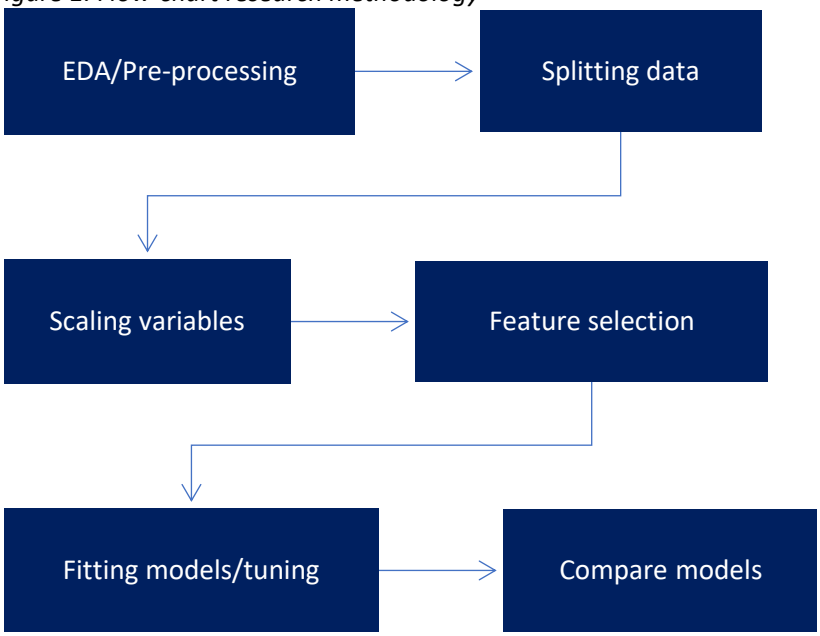


Figure 3: Confusion matrix Neural Network



Appendix G: Research Methodology

Figure 1: Flow-chart research methodology



References

Bzdok, D., Krzywinski, M., & Altman, N. (2018). *Machine Learning: Supervised Methods, SVM and KNN.* Retrieved from HAL: https://hal.science/hal-01657491/document

Nnamoko, N., & Korkontzelos, I. (2020). *Efficient treatment of outliers and class imbalance for diabetes prediction.* Retrieved from Journal of Computational Science, 45, 101151.: https://doi.org/10.1016/j.artmed.2020.101815

Prem. (n.d.). *Random Forest vs. Support Vector Machine vs. Neural Network.* Retrieved from Iunera: https://www.iunera.com/kraken/fabric/random-forest-vs-support-vector-machine-vs-neural-network/#1-support-vector-machines-in-classification

Singh, K. (2020, october 6). *How to Improve Class Imbalance using Class Weights in Machine Learning.* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/10/improve-class-imbalance-class-weights/

Varghese, D. (2018). *Comparative Study on Classic Machine Learning Algorithms.* Retrieved from Towards Data Science: https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222

Zhang, L., Geisler, T., Ray, H., & Xie, Y. (2021). *Improving logistic regression on the imbalanced data by a novel penalized log-likelihood function.* Retrieved from National Library of Medicine: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9542776/