Classification Report

Our study is based on a dataset originating from a Portuguese bank, spanning from 2008 to 2013, containing a random sample of 4100 observations, including an output target, indicating if a customer has bought a long-term deposit account, and 19 input features, containing several indicators, from telemarketing specific and product details to client demographics and socio-economic indicators, providing a comprehensive set of variables for our analysis. We aim to enhance telemarketing efforts by integrating machine learning models into the decision-making process and accurately predicting a phone call's outcome, therefore enabling a more targeted customer selection and prioritisation during campaigns (Essex, 2022). The ultimate goal is to increase the effectiveness of these campaigns while reducing associated costs and time. In order to achieve our goal, our project contains two closely interlinked parts, one of Exploratory Data Analysis (EDA) and one of statistical modelling.

EDA can be characterised as a well-established "statistical tradition" inherited by the machine learning and data science disciplines during the 4th industrial revolution (Behrens, 1997). EDA primarily aims to discover underlying patterns, outliers, correlations and trends. It assists in forming hypotheses by also utilising data visualisation tools for better comprehension of the outputs. Several steps are part of the EDA, such as data cleaning (handling missing data, removing outliers, etc.), data visualisation (box plots, histograms, scatter plots, etc.), and statistical measures (mean, median, mode, standard deviation, etc.) (Chatfield, 1985). The model-developing component of machine learning follows the completion of the EDA. In our project, we had to utilise Machine Learning Classification, a type of Supervised Learning that categorises data into classes (Alpaydin, 2010), to project the outcome variable into yes/no categories by first training the model and then predicting the outcomes. The main techniques that we used were Logistic Regression, Linear Support Vector Classification, K-Nearest Neighbors (KNN), Decision Trees, and Random Forest (Kotsiantis et al., 2007; Ortner, 2020).

Our EDA process started by investigating high-level metrics of our dataset, such as columns, data types, and null values, and looking at the dataset's top five and bottom five rows. The dataset was mainly clean without any null values. We had to rename some column names for ease of use. We also created a summary of basic metrics (such as minimum, maximum, mean and median) of numeric columns and ensured that our dataset had no duplicate tuples (rows). Moreover, we produce the 5-figure summary and the box plots of numeric columns. The "duration" column was highly fluctuating, with significant outliers being present (Figures 1 and 2). The rest of the columns presented no significant findings at this stage.

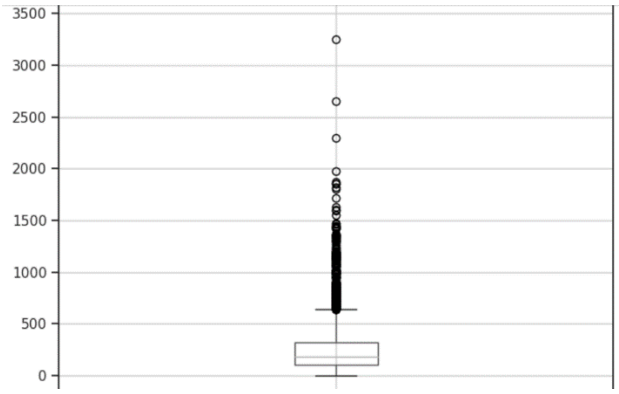

Figure 1: "Duration" column boxplot

```
count    4100.000000
mean      256.754634
std       254.399637
min         0.000000
25%       103.000000
50%       181.000000
75%       317.000000
max      3643.000000
Name: duration, dtype: float64
```

Figure 2: "Duration" column summary of metrics

Additionally, we visualised the column distributions finding significant asymmetry and kurtosis metrics in the "duration", "campaign", "pdays", and "previous" columns (Figures 3 , 4, 5 and 6). Other metrics, such as "age" and "consumer_price_index", appeared to be normally distributed, which is expected due to their nature (Figures 7 and 8).

```
age                        0.714790
duration                   3.300644
campaign                   4.008526
pdays                     -4.762708
previous                   4.019693
empoloyment_rate          -0.727862
consumer_price_index      -0.217439
consumer_confidence_index  0.283387
rate                      -0.715005
employees                 -1.075823
dtype: float64
```

Figure 3: Skew (asymmetry) metrics

```
age                        0.440224
duration                  20.891411
campaign                  25.348525
pdays                     20.693901
previous                  22.070335
empoloyment_rate          -1.042253
consumer_price_index      -0.823263
consumer_confidence_index -0.321729
rate                      -1.396713
employees                  0.062451
dtype: float64
```

Figure 4: Kurtosis metrics

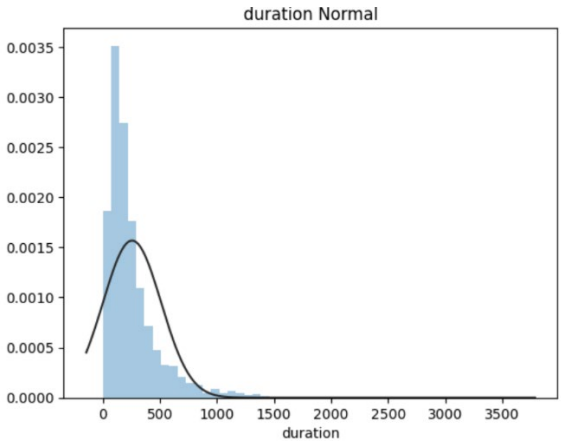

Figure 5: "Duration" distribution

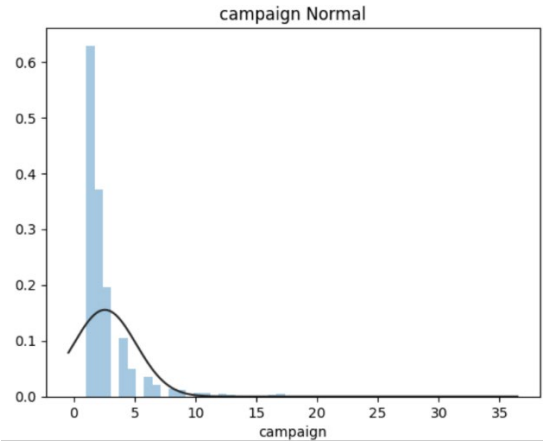

Figure 6: "Campaign" distribution

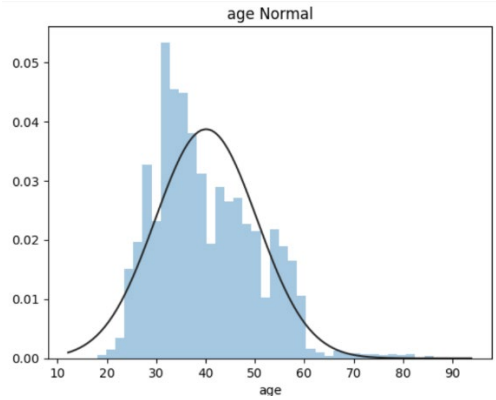

Figure 7: "Age" distribution

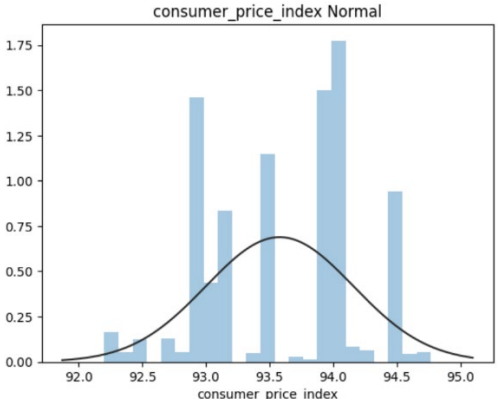

Figure 8: "Consumer_price_index" distribution

We also visualised the pair plots (based on the target column) and correlation metrics (by creating a corresponding heat map to visualise outcomes better), finding significant correlations only between metrics of similar nature, such as "employees" and "employment_rate" (Figures 9, 10 and 11).

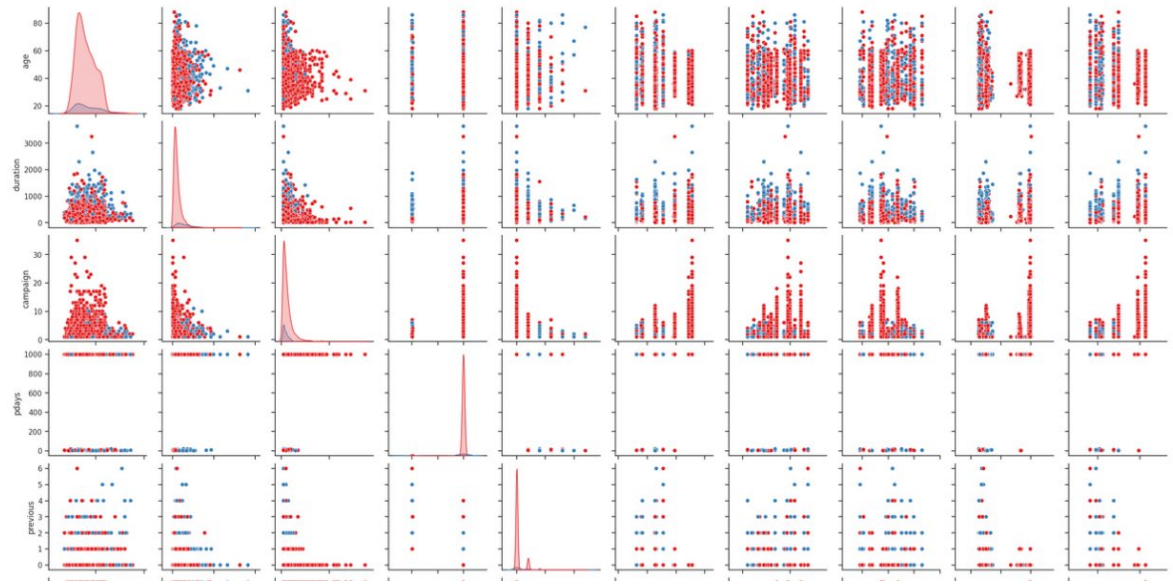

Figure 9: Pairplot based on the target variable

```
age                          -0.019036
duration                     -0.026404
campaign                      0.176226
pdays                         0.271350
previous                     -0.415502
empoloyment_rate              1.000000
consumer_price_index          0.755238
consumer_confidence_index     0.195653
rate                          0.970312
employees                     0.897050
Name: empoloyment_rate, dtype: float64
```

Figure 10: Correlation metrics

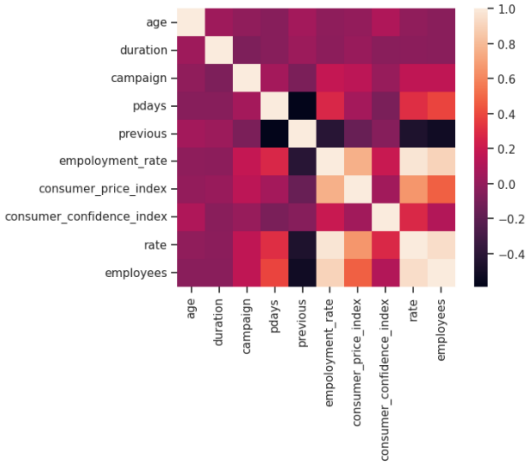

Figure 11: Heat map

Moreover, we looked through the different categories based on the target variable, finding that the better-educated population opted for the program. Most opt-ins were in the middle of the year. The findings regarding opt-ins were similar for people that opted out of the program (Figures 12, 13, 14 and 15).

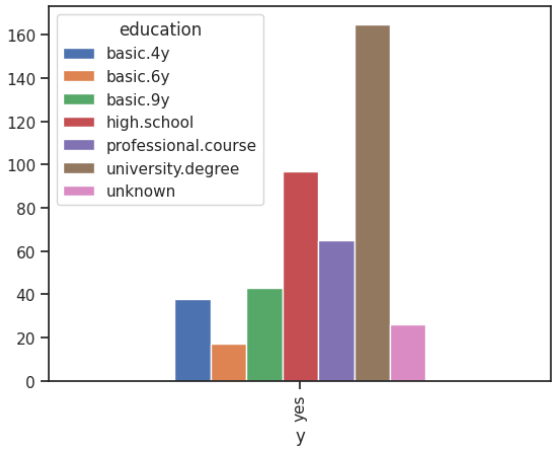

Figure 12: Opt-ins based on "education"

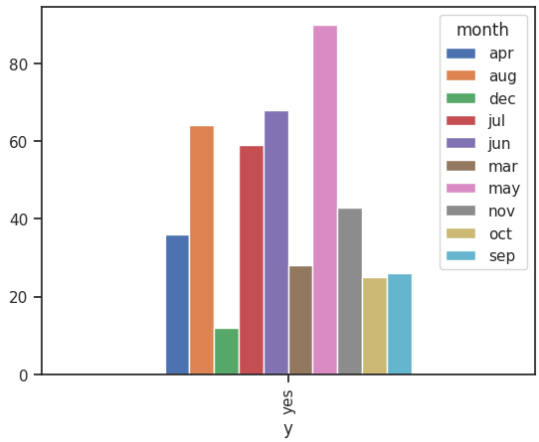

Figure 13: Opt-ins based on the "month"

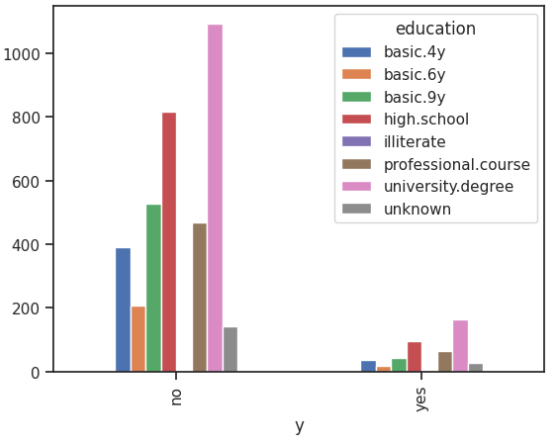

Figure 14: Opt in/out based on "education"

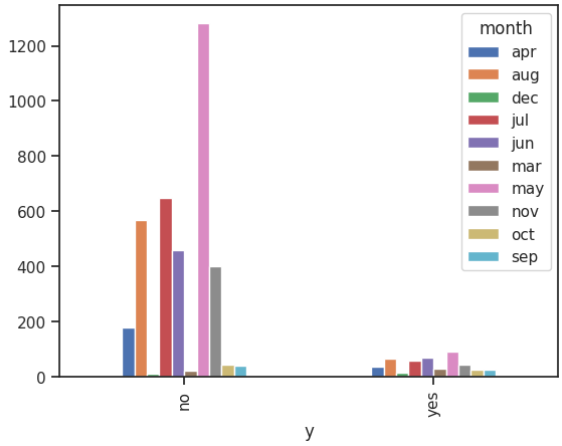

Figure 15: Opt in/out based on the "month"

Regarding numerical values, people of 25-55 age opted-in for the program, being between 93 to 94 in the "consumer_price_index" variable, though following the same norm as the categorical variables, the same categories opted out from the programme. These were also the most significant findings of the EDA (Figures 16, 17, 18 and 19).

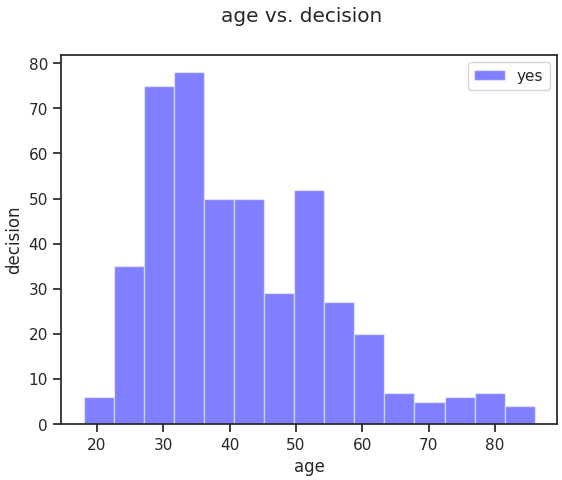

Figure 16: Opt-ins based on "age"

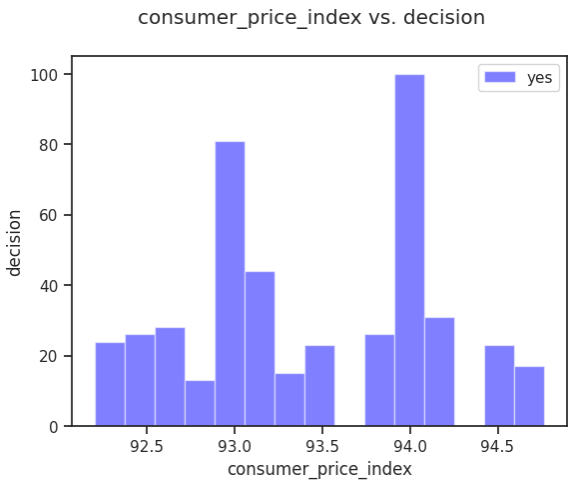

Figure 17: Opt-ins based on "consumer_price_index"

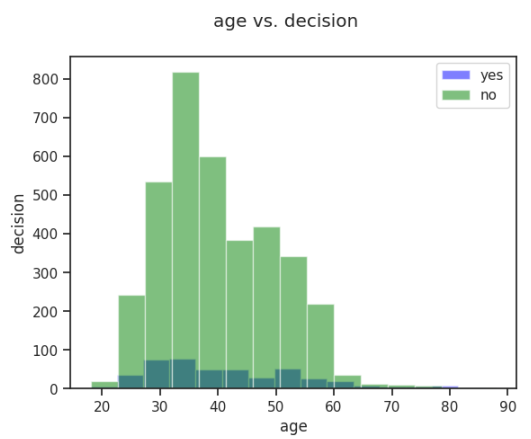

Figure 18: Opt in/out based on "age"

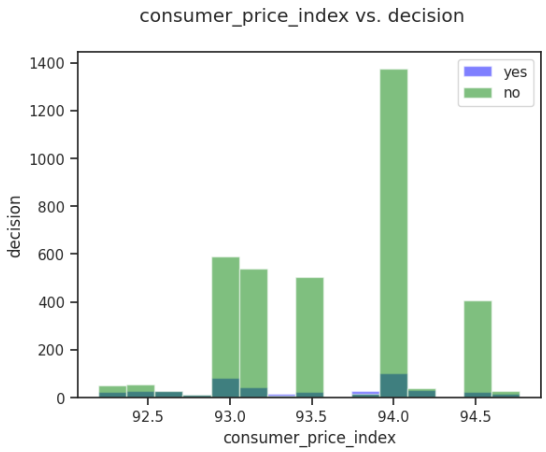

Figure 19: Opt in/out based on "consumer_price_index"

Continuing with our model selection, we started by copying the dataset to a new dataset and changing the categorical target variable to 0 and 1 values (for no and yes, respectively). We also change the columns "loan", "housing", and "default" for yes, no and unknown values to 0 and 1. Moreover, we assigned numerical values to days and months and performed one-hot-encoding to categorical variables with more than two categories. Finally, we split the dataset into two, one having the dataset without the target variable (X) and one having only the target variable (y). All features were selected as we noticed that the models performed better despite some having high asymmetry, as discussed in previous paragraphs.

Regarding our models, we performed investigatory loops for all used models. For some of them, we noticed high inconsistencies between outputs, indicating that the models were overfitting, and thus we used the base variables. More specifically, we have used LogisticRegression, LinearSVC, KNeighbors, DecisionTree, Bagging, AdaBoost and RandomForest classifiers (Ortner, 2020).

Logistic regression (LR) is one of the most common methods used in machine learning in problems where we need to predict categorical or binary variables. The advantage of linear regression is that it applies a sigmoid function fitted for the given features (Swaminathan, 2018), thus performing better in yes/no problems as opposed to using a linear function used for continuous value prediction

(IBM, N.D.). In our specific example, LR was unstable after multiple runs of a fine-tuned model, possibly signifying overfitting. Considering this and the limited data, we proceeded with the base LR model, getting an average precision of 0.62 (62%) by comparing 1000 runs (Figure 20).

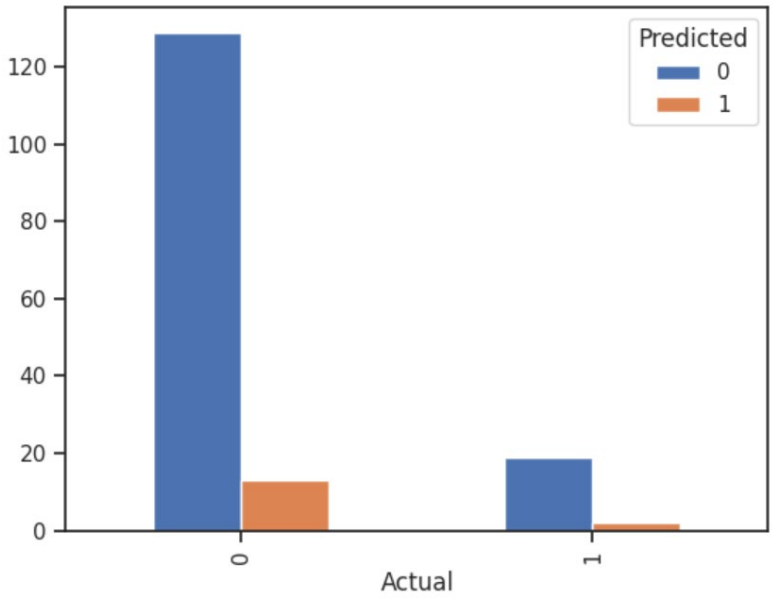

Figure 20: Comparison of actual vs predicted values for Logistic Regression

Linear Support Vector Machine (Linear SVC) was the second model we attempted to fine-tune. Linear SVC is regularly used in problems with high dimensionality. It defines a decision space, finds the closest vectors (data points) and calculates the decision lines to decide on the yes/no output (tutorialspoint, N.D.). In our specific example, Linear SVC was unstable after multiple runs of a fine-tuned model, possibly signifying overfitting. Considering this and the limited data, we proceeded with the base Linear SVC model, getting an average precision of 0.59 (59%) by comparing 1000 runs (Figure 21).

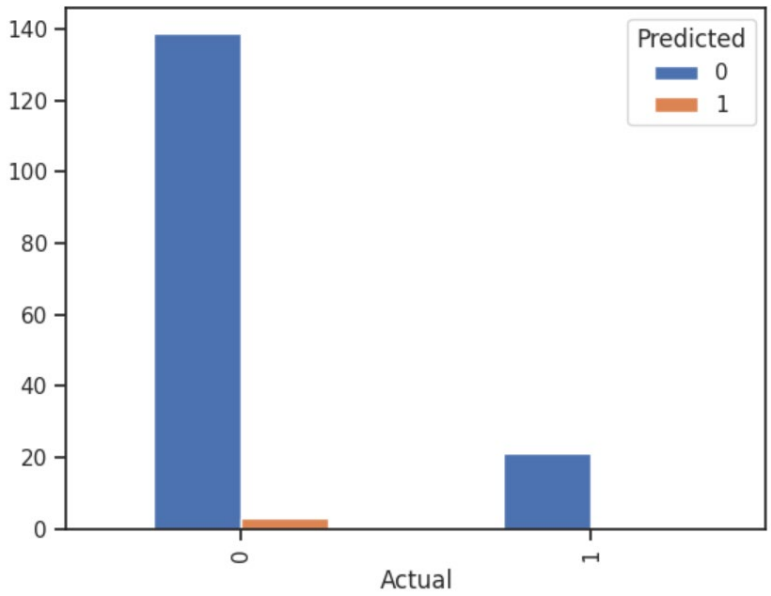

Figure 21: Comparison of actual vs predicted values for Linear Support Vector Machine

N-Nearest Neighbours (KNN) was the third model that we explored. KNN is a proximity-based majority-voting system, grouping points that have close proximity to each other, thus assuming that "similar points can be found near one another" (IBM, N.D.). In this classifier, no over-fitting was observed while exploring parameter assignment, therefore, a more complicated model was used, with eight nearest neighbours and uniform weighting being some of the parameters we used. The KNN model achieved an average precision of 0.63 (63%) by comparing 1000 runs (Figure 22).

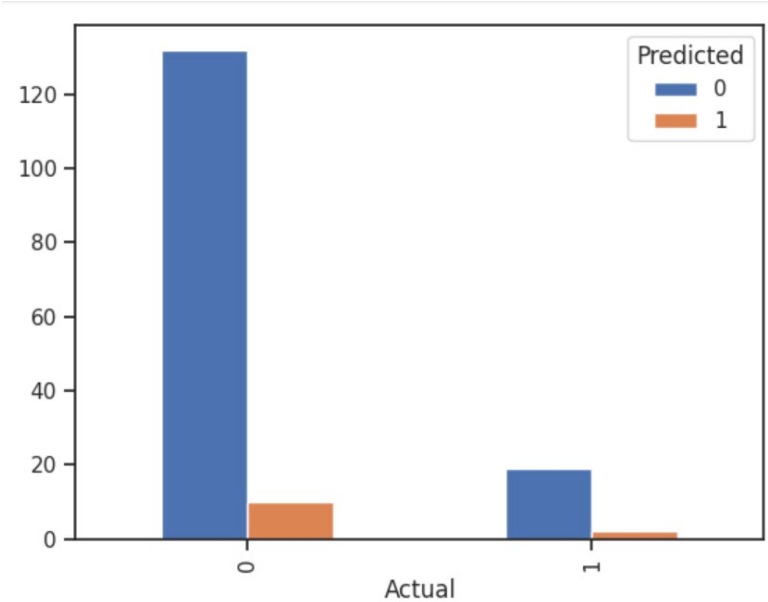

Figure 22: Comparison of actual vs predicted values for N-Nearest Neighbours

Another model we used was the Decision Tree algorithm, which uses a classic tree structure (root, branches, leaves) and utilises a greedy search algorithm to find the tree's split points per the given features (IBM, N.D.). Similarly to the KNN algorithm, no overfitting was observed; therefore, we could specify the maximum nodes and the search depth, as well as our splitting method. The model achieved

an average precision of 0.61 (61%) by comparing 1000 runs (Figure 23). Along with the Decision Tree algorithm, we also investigated the use of a Bagging Classifier (which is considered the Decision Tree's extension) to reduce variance, mainly due to the high variance of categories as explained in the EDA part (outliers, high standard deviation and significant difference between minimum and maximum values), and re-train the same model in many different samples. The results of the algorithm (by using the same Desicition Tree parameters) were marginally better than the plain Decision Tree classifier, performing with an average of 0.67 (67%) by comparing 1000 runs (Figure 24).

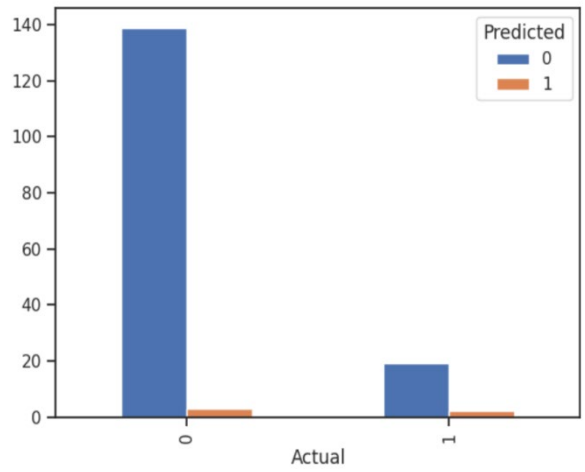

Figure 23: Comparison of actual vs predicted values for Decision Trees

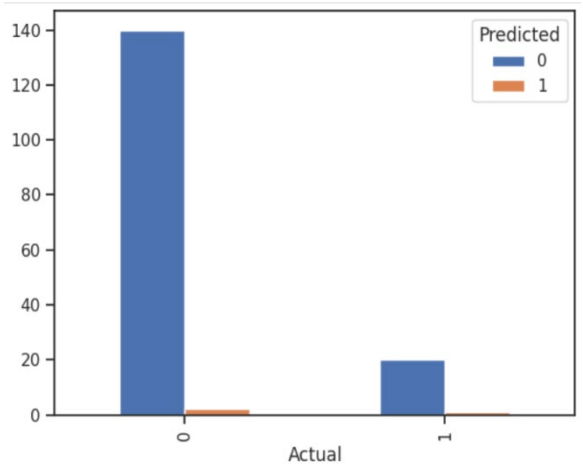

Figure 24: Comparison of actual vs predicted values for Bagging Classifier

The AdaBoost (Adaptive Boosting) classifier works similarly to the Bagging Classifier, meaning a classifier using another base classifier (weak learner), but is sensitive to outliers, which are present in some instances of our dataset, as we showed during the EDA part. Despite its disadvantage can be proven a better-performing classifier in specific problems (De, 2020), and, in our case, the model achieved an average precision of 0.8 (80%) by comparing 1000 runs (Figure 25).

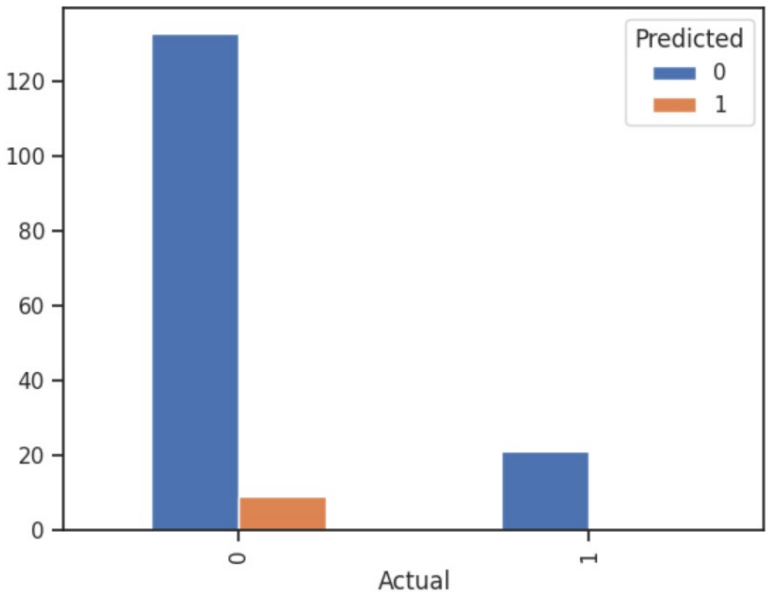

Figure 25: Comparison of actual vs predicted values for AdaBoost

Additionally, we explored using a Random Forest classifier (considered an extension of the Bagging algorithm), an algorithm that combines multiple decision trees by selecting uncorrelated features in every tree and then combining all the decision trees to reach a single result (IBM, N.D.). In our use case, the classifier achieved an average accuracy of 0.53 (53%) by comparing 1000 runs (Figure 26).

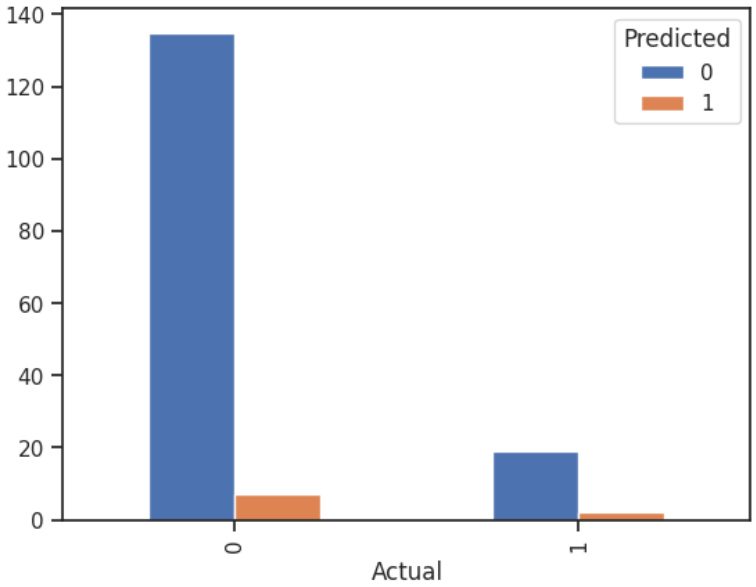

Figure 26: Comparison of actual vs predicted values for Random Forest

Finally, we compared all the results by using a Linear Regression model (as explained before) and indicating an arbitrary threshold above which every value would be classed as positive (1); otherwise, it would be classed as negative (0). Surprisingly, the model achieved an accuracy of 0.8 (80%). However, it is of concern if it will perform with the same precision in a more extensive, diverse dataset (Figure 27).

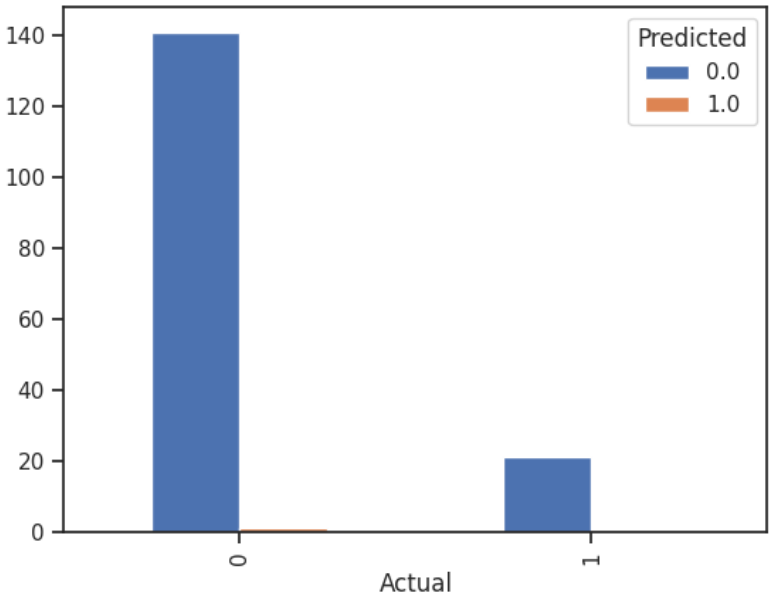

Figure 27: Comparison of actual vs predicted values for Linear Regression

To conclude, our project contained two key components commonly found in machine learning assignments. EDA was instrumental in exploring data and revealing underlying patterns and metrics, providing a robust basis for the machine learning models to follow. The second critical component was the model definition, parameter selection (fine-tunning) and model comparison. Our models contained Logistic Regression, Linear Support Vector Classification, K-Nearest Neighbors (KNN), Decision Trees, and Random Forests, among others. While some models displayed inconsistencies, suggesting overfitting, others, like the AdaBoost classifier and, surprisingly, the Linear Regression with arbitrary thresholds, performed exceptionally well with a precision of 80%. As a future enhancement, neural networks, feature selection technics, and more thorough model performance exploration could be performed, which, coupled with a more extensive dataset, in terms of rows, could provide better results in accuracy and prove production readiness (Figure 28).

Model	Mean	High/Low
LogisticRegression	0.621212121212124	0.621212121212121 / 0.621212121212121
LinearSVC	0.588706747180091	0.833333333333333 / 0.112195121951219
KNeighborsClassifier	0.633333333333328	0.633333333333333 / 0.633333333333333
DecisionTreeClassifier	0.610398680307512	0.9 / 0.421052631578947
BaggingClassifier	0.668777947	0.84 / 0.5
AdaBoostClassifier	0.798023722121466	0.833333333333333 / 0.631578947368421
RandomForestClassifier	0.535015017324944	0.67391304347826 / 0.397058823529411

References:

Hassan, M. (2023) *Programming Exercise* [Submission Component]. MSC in Data Science 2023 Visualising Data January 2023. University of Essex Online.

Behrens, T. (1997) Principles and procedures of exploratory data analysis. *Psychological Methods* 2(2): 131-160. DOI: https://doi.org/10.1037/1082-989X.2.2.131

Chartfield, C. (1985) The Initial Examination of Data. *Journal of the Royal Statistical Society* 148(3): 214-253. DOI: https://doi.org/10.2307/2981969

Alpaydin, E. (2010). *Introduction to Machine Learning.* 2nd ed. Cambridge: The MIT Press. Available from: https://kkpatel7.files.wordpress.com/2015/04/alppaydin_machinelearning_2010.pdf [Accessed 31 July 2023]

Kotsiantis, S. (2007) *Supervised Machine Learning: A Review of Classification Techniques*. Available from: https://datajobs.com/data-science-repo/Supervised-Learning-[SB-Kotsiantis].pdf [Accessed 31 July 2023]

Ortner, A. (2020) Top 10 Binary Classification Algorithms [a Beginner's Guide]. Available from: https://towardsdatascience.com/top-10-binary-classification-algorithms-a-beginners-guide-feeacbd7a3e2 [Accessed 31 July 2023]

Swaminathan, S. (2018) Logistic Regression — Detailed Overview. Available from: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc [Accessed 31 July 2023]

IBM. (N.D.) What is linear regression?. Available from: https://www.ibm.com/topics/linear-regression [Accessed 31 July 2023]

Tutorialspoint. (N.D.) Scikit Learn - Support Vector Machines. Available from: https://www.tutorialspoint.com/scikit_learn/scikit_learn_support_vector_machines.htm [Accessed 31 July 2023]

IBM. (N.D.) What is the k-nearest neighbors algorithm?. Available from: https://www.ibm.com/topics/knn [Accessed 31 July 2023]

IBM. (N.D.) What is a Decision Tree?. Available from: https://www.ibm.com/topics/decision-trees?mhsrc=ibmsearch_a&mhq=decision%20trees [Accessed 31 July 2023]

IBM. (N.D.) What is Bagging?. Available from: https://www.ibm.com/topics/bagging?mhsrc=ibmsearch_a&mhq=bagging [Accessed 31 July 2023]

De, M. (2020) Classification with AdaBoost. Available from: https://community.ibm.com/community/user/ai-datascience/blogs/moloy-de1/2020/11/19/points-to-ponder?mhsrc=ibmsearch_a&mhq=adaboost&_gl=1*12tv6py*_ga*MTE5NTg4OTMxNS4xNjg4MzY0MzE5*_ga_FYECCCS21D*MTY5MDgwNzMzMi44LjEuMTY5MDgxMDYzOC4wLjAuMA.. [Accessed 31 July 2023]

IBM. (N.D.) What is random forest?. Available from: https://www.ibm.com/topics/random-forest?mhsrc=ibmsearch_a&mhq=random%20forest [Accessed 31 July 2023]