Swithun Michaelangelo Chiziko 17487140

SUMMARY

We investigated the data types of all the variables in the given dataset. Several redundancies found with variables being perfectly correlated. These were found to have been derived from other variables in the data set. Imputation was performed using the multivariate framework known as MICE by specifying a decision tree regressor for imputation. It was unclear whether the learned model was used to impute the split datasets as this was done separately to reduce data leakage and overfitting. Our feature selection process eliminated the dimensions to 18 from 37. Using hyperparameter fine tuning, we were able to increase the accuracy of our models with K-NN which yielded the highest accuracy at k = 40. Decision Tree Classifier showed the highest consistency on both the validation set and test set of 79.35%. We expect this to perform the best for this reason on new data. The distribution across the dataset remained unchanged with new data.

INTRODUCTION

Sensitive information is often masked through data obfuscation. A massive part of the Analytical problem is known to be the cleaning and preparation process. This report details the preparation and classification of an obfuscated dataset consisting 1200 records and 32 variables. We detail each phase from exploring data types, explanatory analysis, imputation, variable selection and fine tuning and classification.

DATA PREPARATION & EXPLANATORY ANALYSIS

DATA TYPES

After importing the data using the sqlite3 package, we begin the explanatory process by checking the data types for each variable as automatically assigned by Python. TABLE 1 shows the data types;

Att.	dtype (Python)	Variable type (interpretation/treatment)	Justification
Att00	float64	continuous/float64	Variable is continuous
Att01	object	categorical/int64	For easy computation in Python
Att02	int64	discrete/int64	Variable is discrete
Att03	float64	continuous/float64	Variable is continuous
Att04	float64	continuous/float64	Variable is continuous
Att05	float64	continuous/float64	Variable is continuous
Att06	int64	discrete/int64	Variable is discrete
Att07	float64	continuous/float64	Variable is continuous
Att08	object	categorical/int64	For easy computation in Python
Att09	float64	continuous/float64	Variable is continuous
Att10	float64	continuous/float64	Variable is continuous
Att11	float64	continuous/float64	Variable is continuous
Att12	float64	continuous/float64	Variable is continuous
Att13	float64	continuous/float64	Variable is continuous
Att14	float64	continuous/float64	Variable is continuous
Att15	float64	continuous/float64	Variable is continuous
Att16	float64	continuous/float64	Variable is continuous
Att17	float64	continuous/float64	Variable is continuous
Att18	float64	continuous/float64	Variable is continuous
Att19	float64	continuous/float64	Variable is continuous
Att20	float64	continuous/float64	Variable is continuous
Att21	int64	categorical/int64	For easy computation in Python
Att22	int64	discrete/int64	Variable is discrete
Att23	int64	categorical/int64	For easy computation in Python
Att24	float64	continuous/float64	Variable is continuous
Att25	float64	continuous/float64	Variable is continuous
Att26	float64	continuous/float64	Variable is continuous
Att27	float64	continuous/float64	Variable is continuous
Att28	float64	continuous/float64	Variable is continuous
Att29	object	categorical/int64	For easy computation in Python
Class	float64	categorical/int64	For easy computation in Python

TABLE 1: SHOWING DATA TYPES

For better eye visualising, we converted the data frame to a csv format in order to navigate through the data and investigate and/or compare any inferences visually. From this we can see that attributes assigned float64 are numeric continuous variables whereas those with int64 are either discrete variables or categorical. Three attributes namely Att01, Att08 and Att29 were imported as having an object data type. We note that these variables are string variables. As this is a classification problem in Python, a lot of the modelling and packages to be used will require them to be in some sort of numerical format for the processes to work. Thus, the strings would have to be converted into some nominal number using one-hot encoding. We cannot convert them into some arbitrary number as Python can read them in as ordinal data which can cause bias when creating our model as the greater the number assigned, the greater the impact the model places on that observation when predicting the independent variable. This poses a challenge as we having no way of knowing whether these specific variables should be interpreted as having a rank because no domain knowledge has been provided and the data is obfuscated.

VARIABLE DISTRIBUTIONS AND EXPLANATORY ANALYSIS

In order to understand the data, the distribution across each variable is important to justify any treatment of missing values, feature transformation and selection for our classification model. Standard deviation and variance a crucial. Although all the classifiers that will be used in this project can be considered non-parametric (i.e., they do not make any assumptions about the underlying data) with the exception of Naive Bayes which can be both, distributions are useful to avoid the use of spurious data that may seem to reduce bias but increase variance which may in turn lead to overfitting (Atiku and Obagbuwa 2021). Overfitting can result from having redundant variables and as such, studying the distributions as well as correlation and covariance that explain the relationships among the independent variables can greatly impact the viability of the final predictive model (Demšar and Zupan. 2021).

In this section we will outline the analysis of some descriptive statistics including skewness found in the dataset. This will also be used to justify any feature scaling techniques that are adopted in our classification process.

ATT00

All numerical variables were tested to identify distribution and skewness to justify their treatment for any future modelling phase i.e., imputation. A normally distributed variable whose datapoint falls around the mean could be justification to impute missing values using the mean for instance. The tests included histogram plots, Q-Q plots, a Kolmogorov-Smirnov test and boxplots for outlier analysis. The Kolmogorov-Smirnov tests the identifiable variable distribution function against the cumulative distribution function of normal distribution. A sample output for the variable Att00 is depicted below:

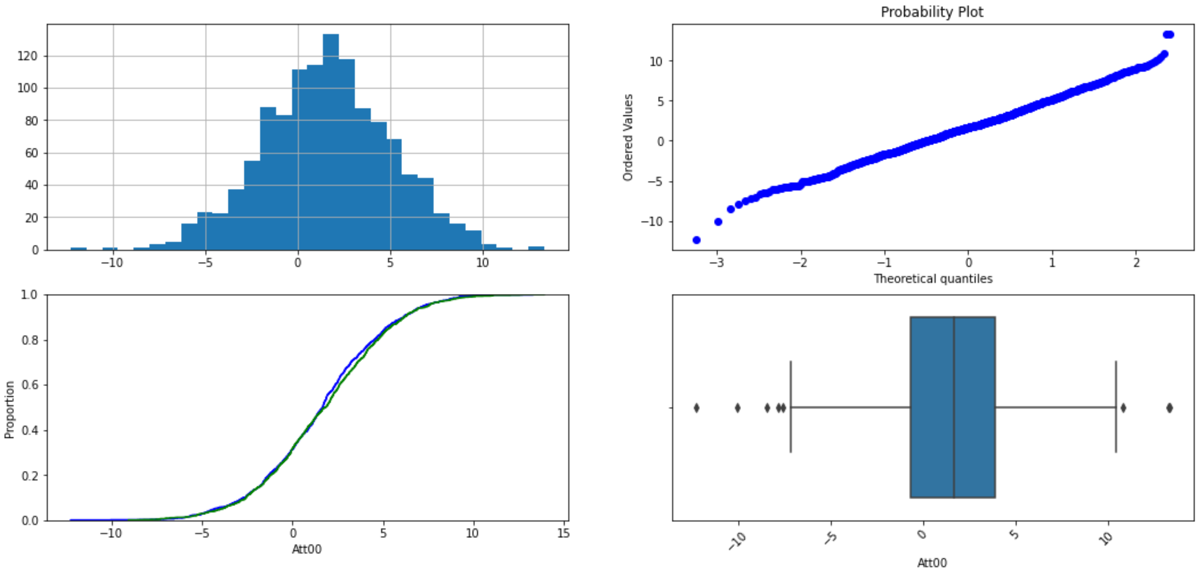


FIGURE 2: DIAGNOSTIC PLOTS (ATT00)

First three plots show partial normal distribution. Median of ~ 1.663 > than mean of 1.606569 implies negative skew with most outliers on the left tail as shown in the boxplot.

ATT02

First three plots show mostly normal distribution. Median of ~ -11304.5 < mean -12175.998333 implies positive skew with most outliers on the right tail. Boxplot confirms this.

ATT03

First three plots show mostly normal distribution. Median of ~ 0.558202 mean < 0.639387 implies positive skew with most outliers on the right tail. Boxplot shows all outliers for Att03 are on the right.

ATT04

First three plots show normal distribution. Median of ~ 2.463898 > mean 1.772553 implies negative skew with most outliers on the left tail. Boxplot confirms this.

ATT05

First three plots show normal distribution. Median of ~ -87.374565 < mean -79.439103 implies positive skew with most outliers on the right tail. Boxplot confirms this.

ATT06

First three plots show normal distribution. Median of ~ -243.500000 < mean -122.075000 implies positive skew with most outliers on the right tail. Boxplot confirms this.

ATT07

First three plots show normal distribution. Median of ~ -46.293813 < mean -42.089354 implies positive skew with most outliers on the right tail. Boxplot confirms this.

ATT01, ATT08, ATT21, ATT23 & ATT29

Att01, Att08 and Att29 have 10, 3, and 7 unique string variables respectively. By grouping our data by Att01, Att08 and Att29, we can begin to note the distribution across categories. Att01 has 3 categories with less than 6 observations namely "ACKH (5), TRRP (1) and UJJW (5). Att08 has 1 category (VEVT) represented in exactly one observation. Finally, Att29 has only 1 category (PJIY) represented in only 1 observation. These observations will later be considered for removal so as not to influence the dataset as they are categorical outliers within their overall variable distribution. At this point we can assume that the inconsistencies within the groupings may be attributed to missing data as every number should be the same across all row groups which is not the case for class, Att00 and Att08 attributes after performing a simple groupby function in Python.

Att21 as well as Att23 have an almost equal representation of 1's and 0's. Att21 has 614 0's and 586 1's whereas Att23 has 619 1's and 581 0's found using print((df["Att23"].values == 0).sum()) as per adjustment to variable.

ATT09

First three plots show deviations from normal distribution. Median of ~ 0.104200 mean > 0.057382 implies negative skew with most outliers on the left tail. Boxplot confirms this.

ATT10

First three plots show normal distribution. Median of ~ 63.534198 mean > 62.063112 implies negative skew with most outliers on the left tail. Boxplot confirms this.

ATT11

First three plots show mostly normal distribution. Median of ~ 8.777016 mean > 6.314273 implies negative skew with most outliers on the left tail confirmed by boxplot.

ATT12

First three plots show mostly normal distribution. Median of ~ -0.216777 > mean -0.436393 ignoring the signs implies skew with most outliers on the left. Boxplot shows most outliers are on the right.

ATT13

Not normally distributed. Median of ~ 7.430563 mean = 7.430563. The distribution is perfectly symmetrical with a defined centre which was shown in the boxplot. There are no outliers.

ATT14

First three plots show normal distribution. Median of ~ -0.412360 mean < 0.336955 implies positive skew with most outliers on the right confirmed by the boxplot.

ATT15

First three plots show normal distribution. Median of ~ 2.152275 mean > 1.991142 implies negative skew with most outliers on the left confirmed by the boxplot.

ATT16

First three plots show normal distribution. Median of ~ -41.922157 mean < -33.929722 implies positive skew with most outliers on the right confirmed by boxplot.

ATT17

First three plots show normal distribution. Median of ~ -0.075582 mean < 0.475639 implies positive skew with most outliers on the right confirmed by boxplot.

ATT18

First three plots show mostly normal distribution. Median of ~ 2.731787 mean < 3.129096 implies positive skew with all outliers on the right. Boxplot confirms this.

ATT19

First three plots show mostly normal distribution. Median of ~ 0.794882 mean > 0.725818 implies negative skew with most outliers on the left. Boxplot shows most outliers to be on the right.

ATT20

Not normally distributed. The distribution is perfectly symmetrical with a defined centre which was shown in the boxplot. There are no outliers confirmed. by the boxplot. Median of ~ 4.614506 mean = 4.614506.

ATT22

First three plots show mostly normal distribution. Median of ~ -529.500000 mean < -368.534167 implies positive skew with most outliers on the right confirmed by the boxplot.

ATT24

Not normally distributed. The distribution is perfectly symmetrical with a defined centre which was shown in the boxplot. There are no outliers confirmed. by the boxplot. Median of ~ 1.518331 = mean 1.518331.

ATT25

First three plots show mostly normal distribution. Median of ~ 2.555237 mean > 2.382607 implies negative skew with most outliers on the left confirmed by the boxplot.

ATT26

First three plots show mostly normal distribution. Median of ~ 0.114347 mean < 0.131220 implies positive skew with most outliers on the right confirmed by box plot.

ATT27

First three plots show mostly normal distribution. Median of ~ 22.767702 > mean 21.821867 implies negative skew with most outliers on the left. Boxplot shows most outliers to be on the right.

ATT28

First three plots show mostly normal distribution. Median of ~ -0.788438 < mean is -0.548260 implies positive skew with all outliers on the right confirmed by boxplot.

CLASS (DEPENDENT VARIABLE)

Class is imbalanced with 201 0's, 301 1's and 498 2's representing more than double the 0's.

CORRELATION

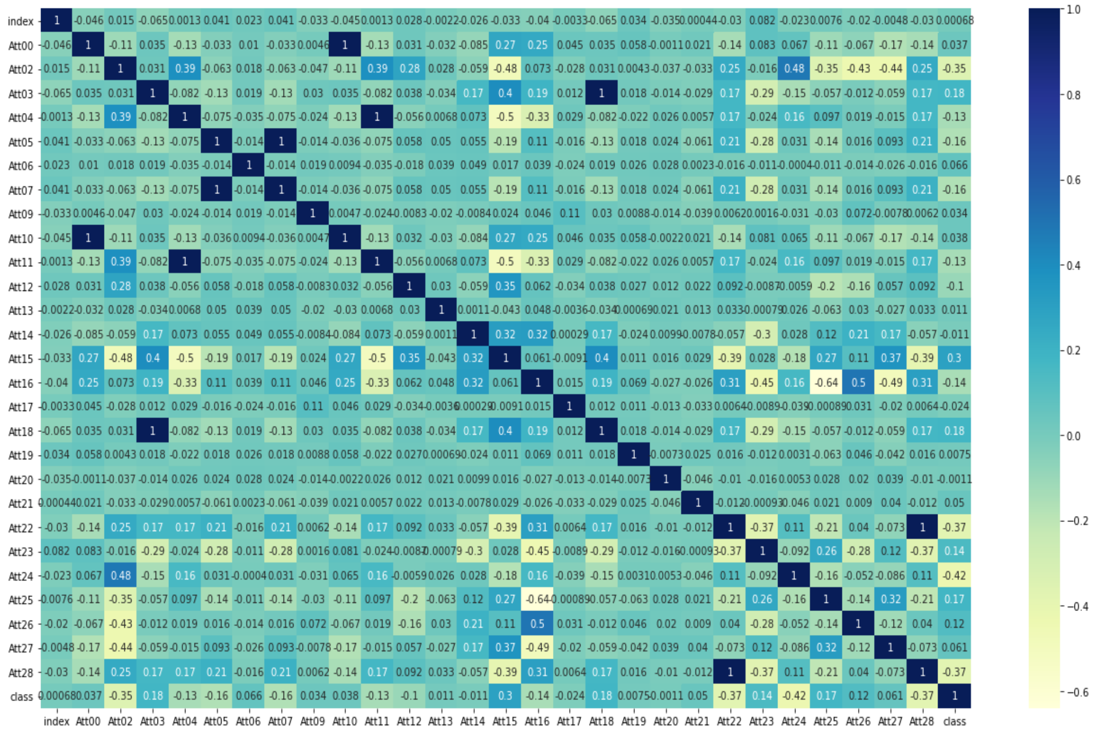


FIGURE 2: CORRELATION

Figure 2 above shows the correlation amongst the variables. The deep blue shade shows very high correlation with the diagonal centre depicting the correlation of the attribute to themselves. What we notice is that outside the diagonal line, other variables also have a perfect positive correlation of 1. These include Att10 and Att00, Att18 and Att03, Att11 and Att04, Att07 and Att05 and finally Att28 and Att22.

One reason for this could that these attributes could be derived from each other. To explore this, we checked the quotients of the variables. Table 2 shows our findings;

Attributes	Quotient
Att07/Att05	~ 0.52983
Att10/Att00	~ 38.5662
Att11/Att04	~ 3.56225
Att18/Att03	~ 4.8939
Att28/Att22	~ 0.00149

TABLE 2: CORRELATED ATTRIBUTES

The findings suggest that the numerator attribute is equal to the quotient multiplied by the denominator attribute. This will be useful in deriving the missing values if present in any one of the attributes present in the table. Manually, this can also be considered proof of high multicollinearity with the dataset given the existence of high intercorrelations among 5 pairs of the variables. In linear models, this can cause estimated coefficients to be insensitive to the impact of the other independent variables and can give high confidence levels that are in reality, grossly inaccurate. As such, we shall explore removing on item from each pair in the feature selecting part of this report.

OUTLIER ANALYSIS

For due to lack of context, outliers were left in the model aside the few categorical variables in Att01, 08, and 29 to avoid leaving any observations out.

SPLITTING DATA

In order to prevent overfitting, we need to try and ensure that the test and training sets have the same distribution. Thus, categorical variables that appear in the training set also need to appear in the test set. We split the data during pre-processing 70% training and 30% test after which the data was further split into a validation set. The fourth set was the new data with no class entries (Demšar and Zupan. 2021).

MISSING VALUES

Missing values were found in two variables namely Att00 and Att09 with 0.75% and 48.42% missing from these attributes respectively. As both attributes are numerical, we checked their distributions by using diagnostics plots. We found both attributes to have a normal distribution by testing for normality. Under these circumstances, it would be reasonable to replace the missing entries with the mean as the tests show that most values lie around the mean. However, given the difference in the percentage of missing values for both, the test for normality can only be confidently applied to Att00 and not Att09 because the latter is missing nearly half its data.

In the section “CORRELATION”, we explored the data and found that Att00 was correlated to Att10 with a correlation of 1. The quotient of Att10/Att00 was found to be ~ 38.5662 across all observations. We concluded that this meant the missing values of Att00 could be found by multiplying the corresponding value of Att10 by 38.5662 and to that effect, we can replace the missing values of Att00 by deriving them from Att10. As for Att09, we opted to employ the MICE framework.

MICE (Multiple Imputation by Chained Equations) was used from the missForest package. Using the MICE framework, the modelling of each variable is performed by considering the condition of the other variables in the dataset. It uses the other variables as predictors to fill the values of an incomplete independent attribute. It trains a model using the other variables in order to make a prediction that will replace the missing entry. It performs multiple imputation cycles until it chooses stable parameters with which to work with in building the predictive model. The cycles are done in order to reduce the bias as incomplete variables are included in the prediction of other incomplete variables. As stated earlier, because both variables are continuous, we can use the iterative imputer offered in the missForest package using decision trees to model the predictions (Mera-Gaona et al 2021). This was done separating for each split dataset to avoid data leakage that could lead to an overestimated model of the test set for instance inappropriately influences the training set. The challenge here was ensuring that it was using the same model coefficients learned in the trainset to input the values in test and validation sets and that the relationship between the variable remained the same after imputation. The Decision Tree regressor was used for imputation (Demšar and Zupan. 2021).

DUPLICATES

No duplicate instances or attributes were detected in the dataset.

IRRELEVANT ATTRIBUTES

When data has many dimensions, there can often be redundant variables that affect the accuracy of a model. If there are not a sufficient number of samples for each class of the dependent variable, it can often cause unnecessary high-dimensionality and this can in turn obscure the relevant attributes. As such, irrelevant attributes are of great interest when it comes to developing an accurate predictive model.

In the section “DATA TYPES”, it was noted that in the string attributes, some of these strings which we decided to treat as classes had been only represented in 1 or less than 6 observations. This represented less than 0.6% of the observations across all three string attributes. Because these would affect the distribution between the test and training set upon split, these particular observations were found to be irrelevant for our predictions. Following one hot encoding in the next section, a lot of data was replicated which increase the dimensions of the dataset in order to have these values to be represented as numerical in preparation for the modelling task. Feature extraction using a decision tree regressor were employed in order to address this issue and chose the most significant attributes to perfect out model.

From the section “CORRELATION”, we removed one of the two pair of highly correlated variables. Because they had the same distribution, we could either one in the pair. Original dataset had dimensions of 32 variables. After removing "Att05", "Att00", "Att11", "Att18", and "Att22" from all the split data, the dimensions were reduced to 25 variables.

CATEGORICAL ENCODING

Using Hot One Encoding, the dimensions increased to 40 variables for each of the unique variable in Att01, 08 and 29. The following attributes were removed "Att01_ACKH", "Att01_TRRP", "Att01_UJJW", "Att08_VEVT", "Att29_PJIY"which are derived from the three attributes were removed. These were discussed in section “VARIABLE DISTRIBUTIONS AND EXPLANATORY ANALYSIS” as having less than 6 representations in their respective attribute column observation. The rationale was such that there wasn’t a variable present in the train set that was not there in the other split datasets. Thus, this model would not be well suited to predict observations that had these attributes present in the future datasets.

FEATURE EXTRACTION AND FEATURE SELECTION

Feature extraction is the process of figuring out which of the variables are the most significant with regards to impacting the model. We used the DecisionTreeClassifier to determine feature importance and the following output was produced.

```
Importance score for Att02 is: 0.046622421528110934
Importance score for Att03 is: 0.049705010862990624
Importance score for Att04 is: 0.04108486857057547
Importance score for Att06 is: 0.05175343980964431
Importance score for Att07 is: 0.03951812100908871
Importance score for Att09 is: 0.0
Importance score for Att10 is: 0.0057184180247189294
Importance score for Att12 is: 0.05548791592190612
Importance score for Att13 is: 0.019521246022782646
Importance score for Att14 is: 0.07368747685689299
Importance score for Att15 is: 0.10887871750286135
Importance score for Att16 is: 0.06046536652231321
Importance score for Att17 is: 0.0
Importance score for Att19 is: 0.014674564258077935
Importance score for Att20 is: 0.0
Importance score for Att21 is: 0.0
Importance score for Att23 is: 0.0
Importance score for Att24 is: 0.1284169852040991
Importance score for Att25 is: 0.043582992127501906
Importance score for Att26 is: 0.09902248217082396
Importance score for Att27 is: 0.05276874184226814
Importance score for Att28 is: 0.09322961128273288
Importance score for Att01_BYUB is: 0.0
Importance score for Att01_GHKA is: 0.0
Importance score for Att01_LLTF is: 0.0
Importance score for Att01_LWYW is: 0.0
Importance score for Att01_OSUG is: 0.0
Importance score for Att01_SCIJ is: 0.0
Importance score for Att01_UKEV is: 0.0
Importance score for Att08_HFTX is: 0.013838108233354913
Importance score for Att08_YIFL is: 0.0
Importance score for Att29_FLJD is: 0.0
Importance score for Att29_HUUV is: 0.0
Importance score for Att29_OELG is: 0.0
Importance score for Att29_OQDJ is: 0.0
Importance score for Att29_TOYT is: 0.0
Importance score for Att29_YLWZ is: 0.002023512249061646
```

TABLE 3: FEATURE IMPORTANCE SCORES

All the variables with an importance score of 0 were removed. Dimensions became 18.

SCALING

The decision as to whether or not to feature scaling came as a consequence of the distribution of the numerical variables. Standardisation and Normalisation are the most commonly used feature scaling techniques within the field of data analysis. Where there are underlying assumptions about the data distribution, normalisation is often used. The accuracy of a predictive model can often suffer where the attributes are being measured with different scales (i.e., km and ml). This is because the model can assign higher weights to numbers that seem larger and thus increasing model bias. Through Normalisation, we address the problem of invariance by scaling the data down between 0 – 1. Standardisation scales data by spreading it across some standard deviations away from the mean. Data falling below the mean becomes negative and data falling above it is made positive. The two techniques ensure that the modelling software does not place weight on variables that are large numerically, reducing bias.

For this project we chose to perform standardisation. This is because we will use K-NN as part of our task once we begin to make predictions and also as mentioned in "IRRELEVANT ATTRIBUTES", a decision tree regressor will be used. K-NN utilises the Euclidean distance of sample data points to find what is known as the nearest neighbour. As a result, if 2 data points are being compared through distance properties then they should be scaled similarly in order to avoid one being considered of greater weight due to the fact that the value is larger at a surface level. It assumes that the data has a normal distribution. We use two other classifiers namely Naïve Bayes and decision trees which often do not require scaling as they do not make the same assumptions about the data.

DATA CLASSIFICATION

SMOTE

SMOTE is an oversampling technique that aims to tackle data imbalance through interpolation. Interpolation is a type of estimation whereby synthetic data points are introduced into the dataset that are within the known range of the pre-existing data. This creation new data points are not extracted at random and SMOTE prevents duplication when oversampling a minority class so that the new set of data are not identical to the old ones (Muslim et al 2021). In the case of our dataset, the under sampled classes were 0 and 1 which originally had 201 and 301 observations each respectively. Class 2 had 498 observations. This distribution largely maintained across our trainset, test set and validation sets as can be seen in the diagram below:

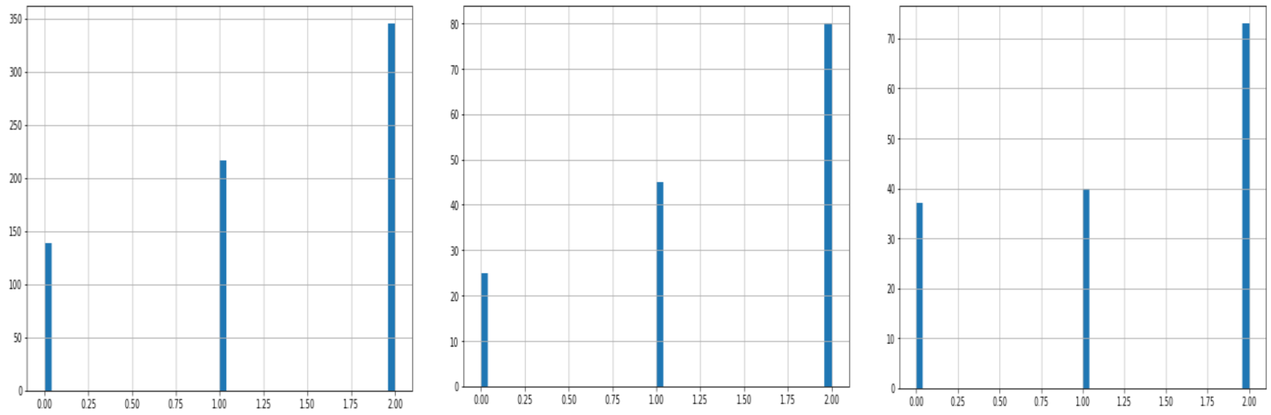


FIGURE 3: SAMPLE SET CLASS DISTRIBUTION

The histogram below shows the distribution post-SMOTE resampling and as is depicted, all classes now have equal representation. To reduce data leakage, we thought it best to only resample the train set so as to maintain the notion that the new data is unseen and should we get new data it may or may not necessarily have the same imbalance or balance of the original dataset (Demšar and Zupan. 2021).

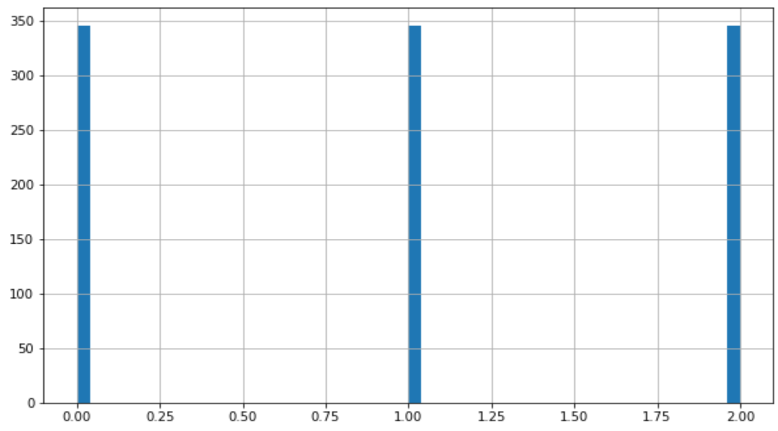


FIGURE 4: RESAMPLED CLASS DISTRIBUTION (TRAINSET)

KNN PREDICTION

The prediction process used three Classifiers namely KNN, Naïve Bays and Decision Trees. Of all the three models trained, KNN yielded the highest results in terms of accuracy. KNN yielded the highest level initially giving an accuracy of 95%. We then performed cross validation, testing at k = 3, 5, 10, 20, 30, 40, 50, 60, 70, 80. K = 40 resulted an accuracy score of ~ 85%, recording the highest of all the k parameters that were chosen. Below is the table showing the accuracy scores that resulted from the model:

	Neighbors	Accuracy
0	3	0.800000
1	5	0.786667
2	10	0.766667
3	20	0.786667
4	30	0.820000
5	40	0.846667
6	50	0.833333
7	60	0.833333
8	70	0.813333
9	80	0.820000

TABLE 4: SHOWING DATA TYPES

From the table above, there seems to be a boundary between k = 3 and k <= 20 above which the k produces the highest accuracy. Of course, below and above 40 seems to be where the large boundary is whereas between 3 seems to be where the small boundary is.

DECISION TREES

For decision tree, we used 10-Cross Fold validation in order to ensure the best prediction levels. The hyperparameters which we used were done by changing under the criterion, max_depth and min_samples_split parameters were tuned using GridSearch Cross Validation. This yielded an accuracy score of 79.35%. This was done through the fine tuning of the following hyperparameters; criterion: entropy, max_depth: 10, and min_samples_split: 2. Both validation and test sets produced an accuracy score of 79.35%. A reduction in the validation accuracy would have meant that there was overfitting in the model. Steps were taken to reduce the data leakage as discussed in the section “SPLITTING DATA”. Entropy measures the range of probabilities that can hinder optimal levels of SUFFICIENT information gain. A breakeven point were true positives and false negatives can be evenly split by the model acts as an equilibrium for the lowest information gain. This equilibrium point is often regarded as the best-case scenario and with entropy, the case for this scenario is being calculated on a consistent basis at which point if found it is picked for the model (Antal-Vaida 2021).

The maximum depth hyperparameter allows us to control how many features are being considered at a particular split of the node. This was done to reduce overfitting as the deeper the node level, the higher it becomes likely to be overfitting. The model works to exhaust all possible outcomes until the chosen parameter is satisfied. Using our specified criteria, our fitting ran 10 folds for each of 56 candidates, totalling 560 fits. This means that the decision tree created 56 nodes and cross validated them 10 times. Below is a table showing the decision tree performance across multiple max depth hyperparameters:

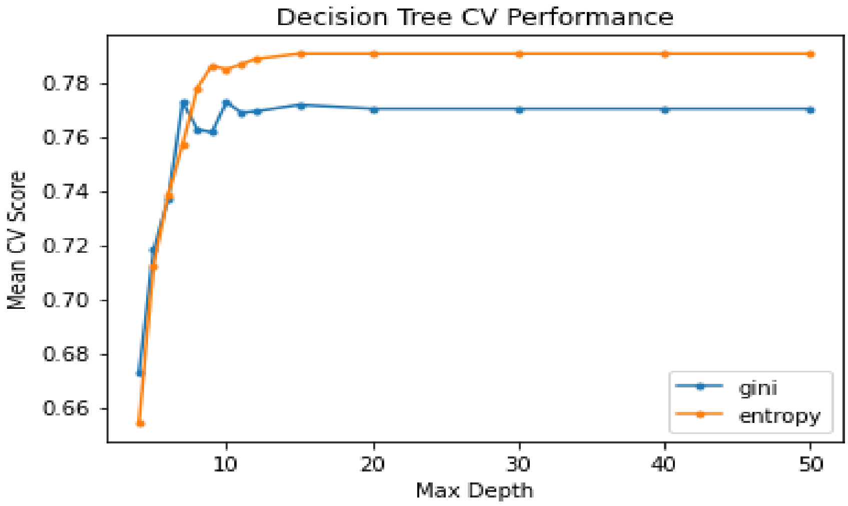


FIGURE 5: GINI VS ENTROPY OPTIMAL MAX DEPTH COMPARISON

NAÏVE BAYES CLASSIFIER

With Naïve Bayes, we registered an accuracy score of 70.86% which we implemented it with SMOTE. The following parameters we used; {'var_smoothing': [1.e+00, 1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06, 1.e-07, 1.e-08, 1.e-09. Variance Smoothing (var_smoothing) here acts as a stabilising factor for attribute values that have a zero-probability occurrence. If the likelihood that a feature with a supposed value belongs a particular classification, then it replaces that probability by estimating the probability of all possible output classes. The values entered for this hyperparameter here are used to figure out the largest variances of each attribute after which the output is added to the stability variance (Atiku and Obagbuwa 2021).

Using our specified criteria, our NB fitting ran 10 folds for each of 10 var smoothing candidates, totalling 100 fits. Figure 6 below shows the comparisons of different

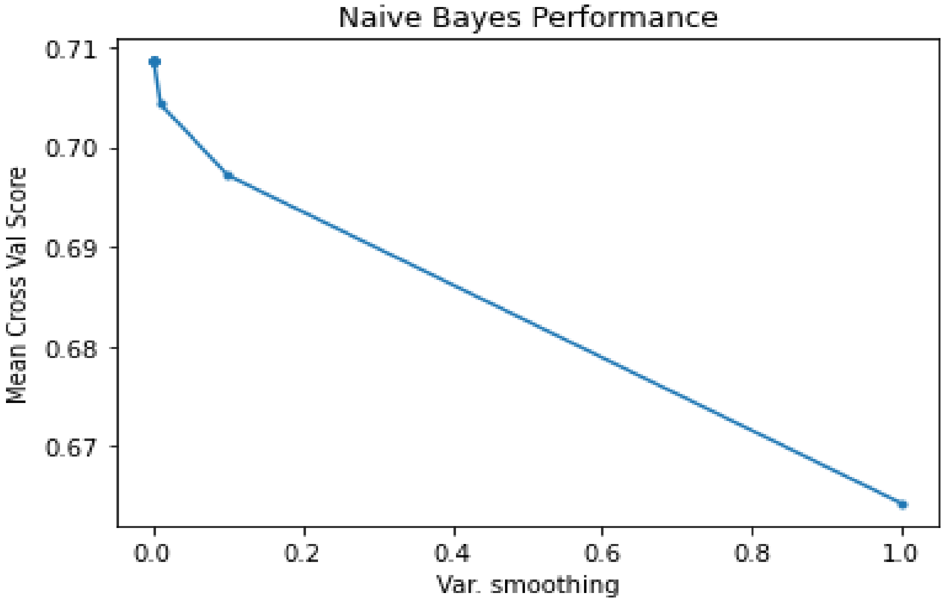


FIGURE 6: VAR SMOOTHING LEVEL MODEL COMPARISONS

Confusion matrices of all three model can be seen in Table 5 Below.

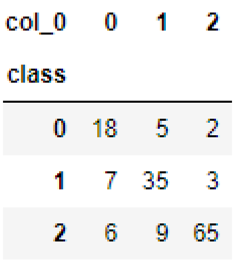
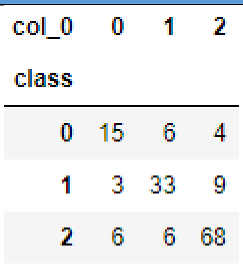
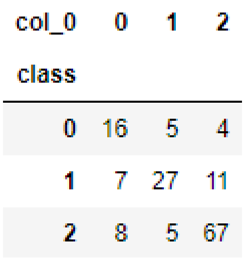
KNN Confusion Matrix	Naïve Bayes Confusion Matrix	Decision Tree Confusion Matrix	
 col_0 0 1 2 class 0 18 5 2 1 7 35 3 2 6 9 65	 col_0 0 1 2 class 0 15 6 4 1 3 33 9 2 6 6 68	 col_0 0 1 2 class 0 16 5 4 1 7 27 11 2 8 5 67	
84.67%	**70.86%**	**79.35%**	Highest Accuracy

Table 5: Comparison of Confusion Matrices

We expect the decision to produce the highest result due to its consistency on the train set and validation set. From the k table, we expect an accuracy level between 70 to 80%. These are our two best models based on our accuracy parameter. Predict01 with KNN had 53 observations for 0, 59 for 1 and 88 for whereas with Decision Trees which is Predict02 we had, 49, 68, and 83 maintain the same distribution as the initial dataset. It is unclear as to whether the new data would have ironically had the same distribution.

CONCLUSION

Our research proves that fine tuning parameter can lead to more accurate modelling results. The true accuracy of the model can only be tested on unseen data. Imputation and using SMOTE may have changed the relationships between the different variables and as such, the impact of this on data that is not as imbalanced could challenge the viability of the model. K-NN proved to have yielded the high accuracy levels with Decision tree being a close second. The fact that Decision tree performed with the same accuracy on the test set and validation set could mean that it would be more consistent with new data. Data leakage was avoided but a large part of the information lost during the feature selection phase did not have any consequential context to consider other modelling techniques such as binning. Without domain knowledge, only data mining techniques are left to produce the best models but without any context to transform the variables for specific model tasks and scope.

REFERENCES

Antal-Vaida, Claudia. 2021. "Basic Hyperparameters Tuning Methods for Classification Algorithms." Informatica Economica 25 (2): 64-74. doi:http://dx.doi.org/10.24818/issn14531305/25.2.2021.06. https://www.proquest.com/scholarly-journals/basic-hyperparameters-tuning-methods/docview/2554700575/se-2.

Atiku, Sulaiman O. and Ibidun C. Obagbuwa. 2021. "Machine Learning Classification Techniques for Detecting the Impact of Human Resources Outcomes on Commercial Banks Performance." *Applied Computational Intelligence and Soft Computing* 2021. doi:http://dx.doi.org/10.1155/2021/7747907. https://www.proquest.com/scholarly-journals/machine-learning-classification-techniques/docview/2578645025/se-2.

Demšar, Janez and Blaž Zupan. 2021. "Hands-on Training about Overfitting." PLoS Computational Biology 17 (3) (03). doi:http://dx.doi.org/10.1371/journal.pcbi.1008671. https://www.proquest.com/scholarly-journals/hands-on-training-about-overfitting/docview/2513684141/se-2.

Mera-Gaona, Maritza, Ursula Neumann, Rubiel Vargas-Canas, and Diego López M. 2021. "Evaluating the Impact of Multivariate Imputation by MICE in Feature Selection." PLoS One 16 (7) (07). doi:http://dx.doi.org/10.1371/journal.pone.0254720. https://www.proquest.com/scholarly-journals/evaluating-impact-multivariate-imputation-mice/docview/2555946583/se-2.

Muslim, M. A., Y. Dasril, A. Alamsyah, and T. Mustaqim. 2021. "Bank Predictions for Prospective Long-Term Deposit Investors using Machine Learning LightGBM and SMOTE." Journal of Physics: Conference Series 1918 (4) (06). doi:http://dx.doi.org/10.1088/1742-6596/1918/4/042143. https://www.proquest.com/scholarly-journals/bank-predictions-prospective-long-term-deposit/docview/2540781225/se-2.