Error analysis for binary classification machine learning models: a comparison of methods

BIOX7008

Oliver Pienaar – 4700 8216

Dr. Moji Ghadimi - m.ghadimi@uq.edu.au

Table of Contents

1| Abstract

Binary classification machine learning algorithms are often used in biomedical scenarios. The ability to evaluate a classification algorithm's performance is critical, so performance metrics such as the confusion matrix are commonly used. The confusion matrix provides insight into the performance of a classifier by categorising the error types present in the dataset. However, the confusion matrix provides no insight into how the classifier's performance varies with differing subtypes of the data. Therefore, we would like to identify methods that can analyse error categories like the confusion matrix, but in subsets of the data. However, we were unable to locate a review of any "subset error analysis" methods. As such, we aimed to identify, implement, test and compare methods currently available for subset error analysis in a biomedical context.

We identified three error analysis methods – error heatmap (HM), error decision tree (EDT) and Slice Finder (SF) – and created a fourth which we termed "one-feature error distribution" (OFED). We trained a random forest classifier on a heart disease dataset, then demonstrated each error analysis method on the classifier's test dataset output. We also compared their ability to identify artificially created error subsets in the classifier's test dataset output.

The OFED was found to be effective at identifying single-feature problematic subsets but was ineffective at locating multi-feature subsets and relatively arduous to use. The heatmap was effective at identifying one- and two-feature problematic subsets but even more arduous than OFED. The EDT method gave less context than the OFED and HM methods but was somewhat effective at identifying multi-feature problematic subsets and was not arduous. Slice Finder gave even less context than the EDT method but was slightly more effective at identifying problematic subsets while also not being arduous. We concluded that the use of multiple of these methods in conjunction would likely be more effective for subset error analysis than any of the individual methods alone.

2| Introduction

Binary classification machine learning algorithms are a type of statistical modelling in which an algorithm learns to "classify" a data point as one of two classes based on its characteristics (1). These algorithms are commonly used in biomedical scenarios (2-4). The ability to evaluate a classification algorithm's performance is critical (5). As such, performance metrics are commonly used to evaluate the effectiveness of binary classification algorithms (3-5). Analysis of errors is commonly integrated with these metrics, the most popular format being confusion matrices, which identify how many data points from each class are correctly or incorrectly classified to be from those classes (Figure 1)(5). These methods/metrics tend to be utilised on the entire population of the testing data to assess how the model as a whole performs (3-5). However, performance and error analysis of subsets of the data can provide valuable insights into how the model's effectiveness varies with differing subtypes of the data (6, 7). Take, for example, a classification model trained to classify which individuals in a population were more likely to develop melanoma. If the whole-population error rate was found to be 35%, the developer may conclude that the model is underperforming on the data. However, if this same analysis was to be examined at the sub-population level, it may be identified that the model performs excellently on individuals with lighter skin tone, but very poorly on those with darker skin tone. As such, knowledge of subset error analysis techniques can be a valuable tool for machine learning developers.

The four error categories prevalently utilised in classification literature are "true positive" (TP), "true negative" (TN), "false positive" (FP) and "false negative" (FN). These categories are commonly displayed using a 2x2 confusion matrix (Figure 1)(5). It consists of an "actual class" column and "predicted class" row (or vice versa). The function is to indicate the quantity of data points that are either 1) correctly classified as "True" (true positive), 2) correctly classified as "False" (true negative), 3) incorrectly classified as "True" (false positive) or 4) incorrectly classified as "False" (false negative). These error categories are useful because they provide more information about the classifier's performance than a simple "correct" or "incorrect" (5). For example, the melanoma classifier may be found to have acceptable

performance on the positive individuals (few FNs) but very poor performance on the negative individuals (many FPs).

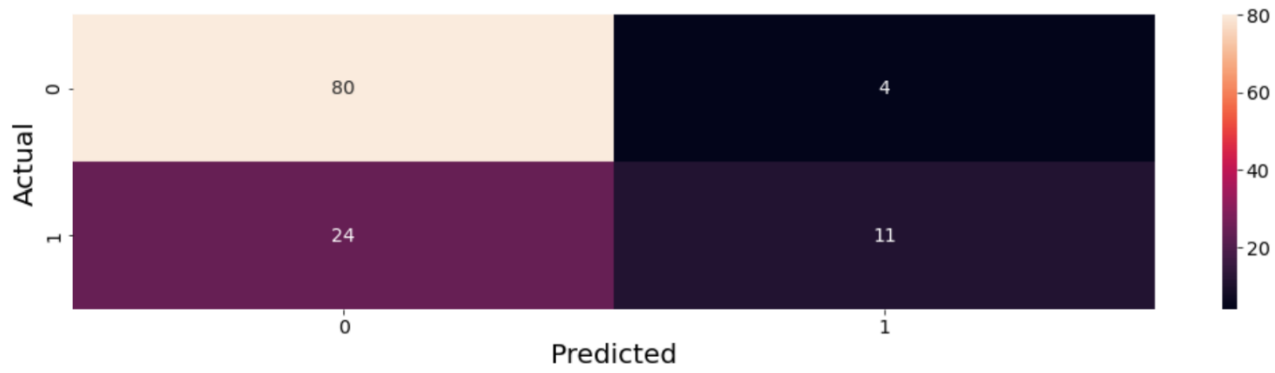


Figure 1 – Confusion matrix plotting "Actual" label (0 = False, 1 = True)(Y-Axis) against "Predicted" label (X-Axis). Darker colour indicates lower quantity.

So, combining these two concepts to gain the benefits of both seems logical. However, we were unable to locate any reviews or summaries of these subset error analysis tools and techniques available. As such, the primary goal of this paper was to identify, implement and test techniques that assess error types in subsets of the data for binary classification machine learning models.

In order to identify and collate the subset error analysis tools available we searched for both scholarly articles and general web posts (forums, blogs, guides etc.). Exhaustive permutations/variations of the search terms "error", "analysis", "binary", "classification" "machine learning", "true positive" and so on were used to search the Scopus, IEEE, ACM and Google Scholar databases and general google search. Sources from all of recent history were included in the search as classifiers have been around for 50+ years (8).

Three primary subset error analysis methods were identified – error heatmap (HM) (9), error decision tree (EDT) (9) and Slice Finder (SF) (6). We came up with a fourth, termed one-feature error distribution (OFED).

Each of these methods are ordinarily used to analyse non-specific error rate of a model (no discrimination of error categories) by using them on the whole dataset (6, 9). However, we can gain the additional insight of error categories by using them on true negative (TN/FP) and true positive (TP/FN) values in the dataset individually.

2.1| One-Feature Error Distribution

The one-feature error distribution with Jensen-Shannon distance method is used to statistically and visually compare the distribution of two error categories within a feature. A bar graph of relative error distribution at differing values of the feature is created to allow for visual inspection and the distance between the distributions is quantified utilising the Jensen-Shannon distance (Figure 2). The Jensen-Shannon distance is a measure of similarity between two distributions (10).

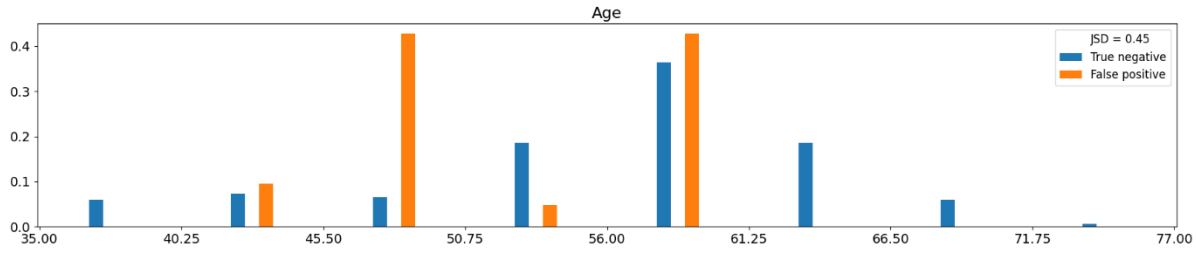


Figure 2 – Column graph of fraction (Y-axis) of “True negative” (blue) and “False positive” (orange) occurrences at varying values of “Age” feature (X-axis)(years), and Jensen-Shannon distance between the distributions.

2.2| Two-Dimensional Error Heatmap

The two-dimensional error heatmap (as seen in Figure 3) is a descriptive tool that visually displays error rates at varying values of two features. The user selects two features and can

view error rates at all value combinations of the two features. For example, if, in a diabetes classifier, the features “BMI” and “Age” were selected, the user could examine how error rates vary with BMI in young people and old people. No statistical analysis occurs, the tool is solely for visual inspection of error rates across varying values of two features.

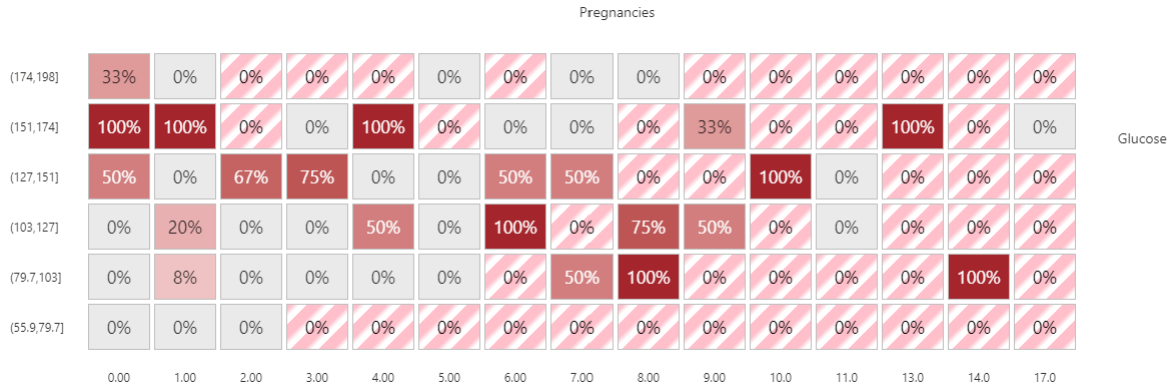


Figure 3 – Microsoft two-dimensional error analysis heatmap displaying type II error rate at varying value combinations of “Pregnancies” (X axis) and “Glucose” (Y axis) features. Striped “0%” values indicate no data points (misclassified or otherwise) at that combination of values.

2.3| Error Decision Tree

The error decision tree is a decision tree classifier used to identify the combinations of features – and therefore subsets – that lead to the highest error rates of a classifier (9, 11). A decision tree model is trained using the same features as the classifier but with the classifier’s test dataset, and the tree’s target variable is the classifier’s error category. As such, the decision tree learns which combinations of features lead to high/low error rates in the classifier. As decision trees are easily interpretable, the tree provides a white-box model of the error. For example, in Figure 4, 100% of participants with a “BloodPressure” of <= 79.0 and “Glucose” of <= 134.0 were misclassified as false negatives (red box).

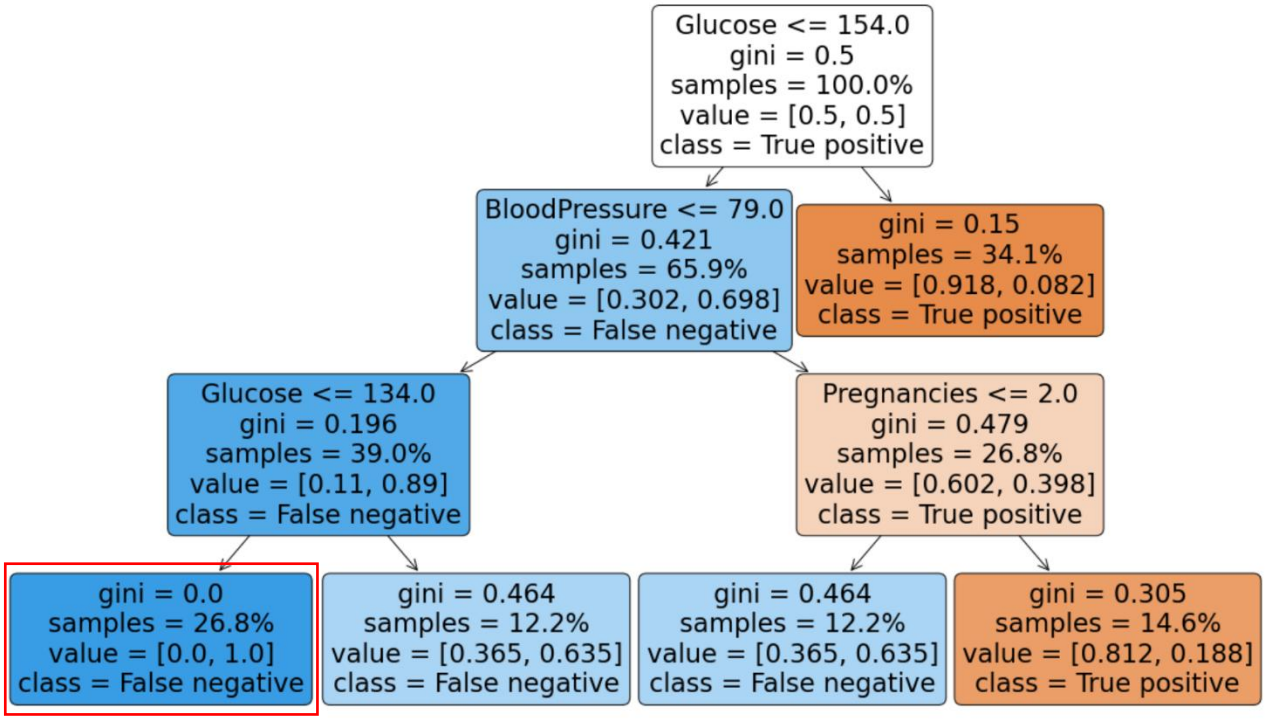


Figure 4 – Scikit-learn decision tree model trained on random forest diabetes classifier test dataset output to classify actual positive data points as either “True positive” or “False negative”. Red box indicates high-error subset.

2.4| Slice Finder

In error analysis, slice finding methods identify “slices” (subsets) of the data with the highest error rates. There are currently multiple slice-finding tools available (7, 12, 13), but we will be focusing on Slice Finder.

Slice finder identifies slices with a clustering algorithm (e.g. lattice search) then identifies the most “problematic” slices based on a combination of error effect size, slice size and divergence from the remainder of the data. The goal of Slice Finder is to provide insight into which subsets of the data have the highest error rate in an interpretable manner. As such, “simpler” slices – those with fewer criteria – will be favoured over more complicated slices.

For example, if the slice “Sex == M, Age > 70, Height > 180cm, Smoker == Yes” has the same size and error effect size as the slice “Sex == M, Age > 75”, the second option will be ranked higher.

As such, the aim of this study was *to implement, test and compare the one-feature error distribution, error heatmap, error decision tree and Slice Finder methods*. This was done on the output of a random forest binary classifier trained on a biomedical dataset.

3| Materials and Methods

3.1| Error Analysis Methods

This report focuses on binary classifier subset error analysis methods. We have defined these as techniques that can be used to investigate the level of true negatives, false positives, true positives and false negatives within subsets of a binary classifier’s output. The four methods outlined in this section – one-feature error distribution, heatmap, error decision tree and Slice Finder – conform to this definition.

3.1.1| One-Feature Error Distribution

The feature and two error categories to be displayed are chosen. Feature values are then binned – categorical features with eight or fewer values are binned by value, while continuous features are divided into eight bins of equal value range. The frequency of each error category in each bin is recorded (e.g. 12 TN and 5 FP errors at “age == 25-30”). These frequencies are then calculated as a fraction of the error type’s total and plotted against feature value as columns on the same graph (Figure 2). This was then displayed using the python Matplotlib library.

The Jensen-Shannon distance was then calculated:

Equation 1 – Jensen-Shannon distance (14)

$$D_{JS}(\mathrm{P} \parallel \mathrm{Q}) = \sqrt{\frac{D_{KL}(\mathrm{P} \parallel \mathrm{M}) + D_{KL}(\mathrm{Q} \parallel \mathrm{M})}{2}}$$

Where P and Q are the two error distributions, M = 0.5 * (P + Q) and D_{KL} is the Kullback-Leibler divergence:

Equation 2 - Kullback-Liebler divergence (15)

$$D_{KL}(\mathrm{P} \parallel \mathrm{Q}) = \sum_{x \in X} P(x) \log\left(\frac{\mathrm{P(x)}}{\mathrm{Q(x)}}\right)$$

While the OFED graph in the form shown in Figure 2 is effective when few are examined, it becomes arduous when attempting to search through many features. In order to increase efficiency, we created an “n_features x 2” condensed format (Figure 5) using the python Matplotlib library.

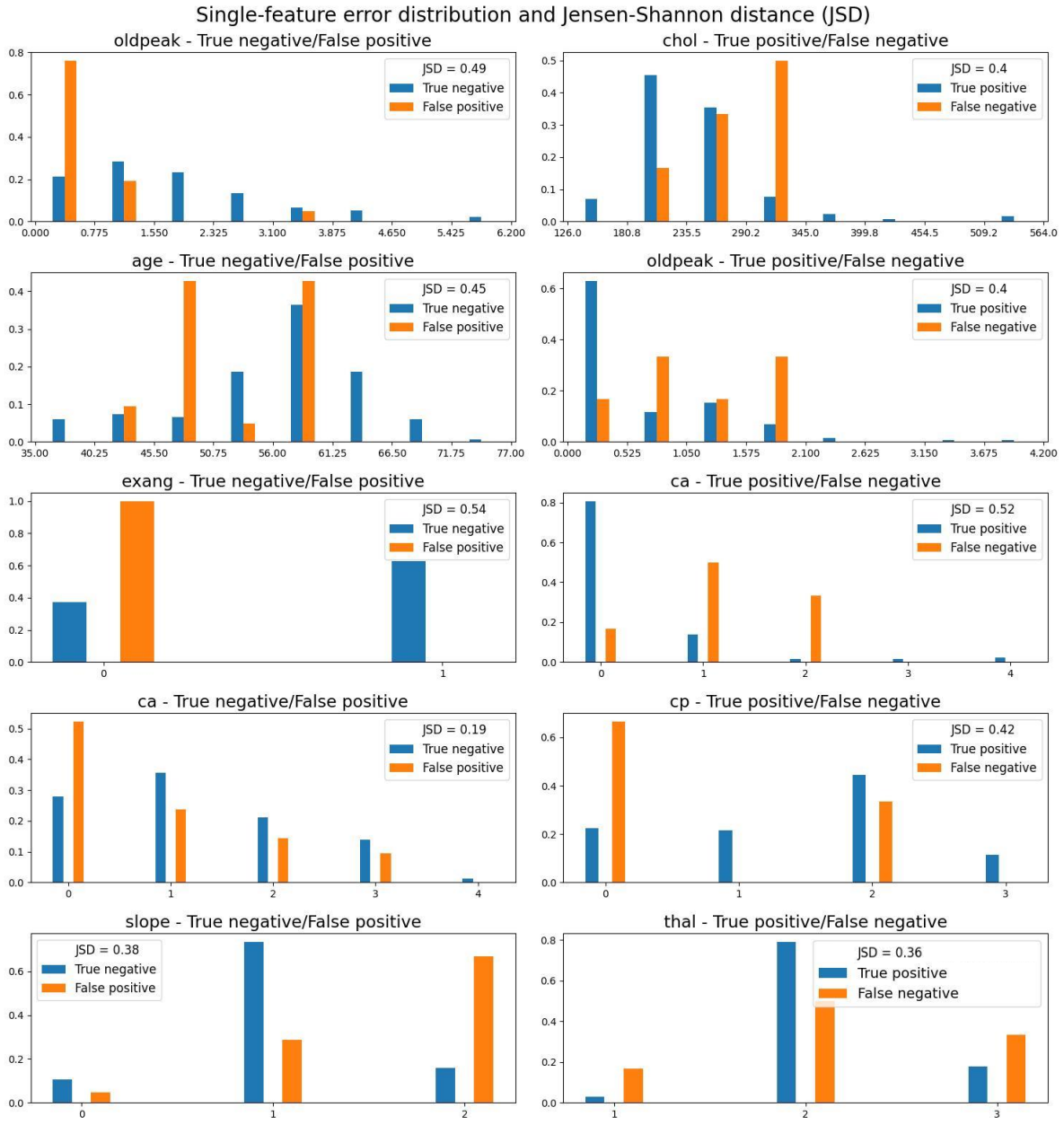


Figure 5 – 5x2 grid of column graphs showing fraction (Y-axis) of true negative (blue) and false positive (orange) (left column) and true positive (blue) and false negative (orange) (right column) occurrences at varying values of the features listed above the graphs (X-axis), and Jensen-Shannon distance between the distributions.

3.1.2| Two-dimensional Error Heatmap

The error heatmap as shown in Figure 2 is available from Microsoft (9). However, it has some limitations. Firstly, it displays the errors in each two-feature subset exclusively as a percentage of the total values in that bucket, without displaying the error count (9). This excludes important information about the relative size of each subset, an important factor in subset analysis as large subsets with high error rates are more important than small subsets with similarly large error rates. For example, the Microsoft heatmap would display a subset containing only one false positive (and no other values) in the same way as a subset containing exclusively 50 false positives (and no other values).

Secondly, Microsoft's heatmap does not natively discriminate between error categories – it merely classifies points as error/not error.

In order to address these issues we created our own heatmap using the Seaborn and Matplotlib python libraries. As shown in Figure 6, our heatmap displays the subset error count (top number) and total value count of the error and its partner category (TN/FP and TP/FN) in that subset (bottom number). The error category is customisable such that any error category can be displayed, and as either count or percentage. The heatmap colour intensity indicates the error count relative to all other subsets in the heatmap. As such, the subset with the highest error count, irrespective of total values (error and not) in that subset, will appear white while the subsets with lowest error count (usually 0) will appear dark.

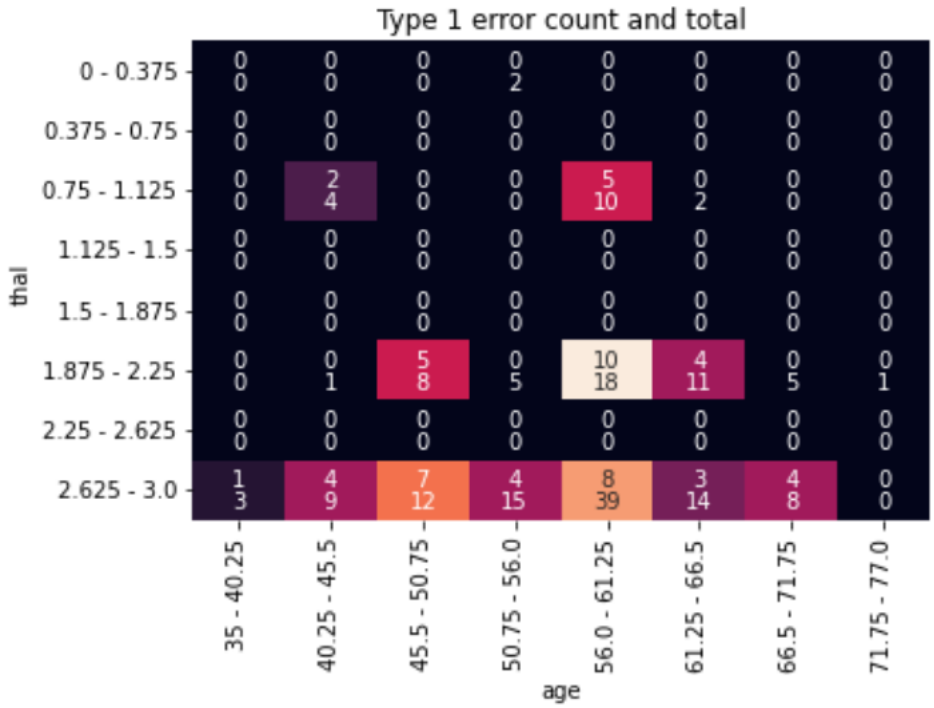


Figure 6 – Single heatmap showing type 1 (false positive) error count (top number) and total true negative and false positive count (bottom number) in each two-feature subset combination of “thal” and “age” features.

When generating heatmaps as used in this study, the primary steps in the program were to firstly divide the two chosen features into bins in the same manner as the OFED method where categorical feature values become bins and continuous feature values are divided into eight bins with equal value ranges. Secondly, count the number of the chosen error type and its pair (TN/FP or TP/FN) in each two-feature value combination subsets. Thirdly, generate the heatmap based off the chosen error type count, then display that count on top and the pair count on the bottom.

However, similarly to the OFED method, viewing heatmaps individually can be arduous/inefficient when analysing large feature sets as the number of heatmaps scales exponentially with features ($n_{heatmaps} \cong 2n_{features}^2$). To reduce the visual space occupied by the heatmaps, we created an automated “n_features x n_features” heatmap grid (Figure

6). One grid fits two customisable error types – one in the top-right half and one in the bottom-left half (Figure 7). The colour intensity/whiteness is localised to individual heatmaps such that the subset with the highest error count in each individual heatmap will appear white.

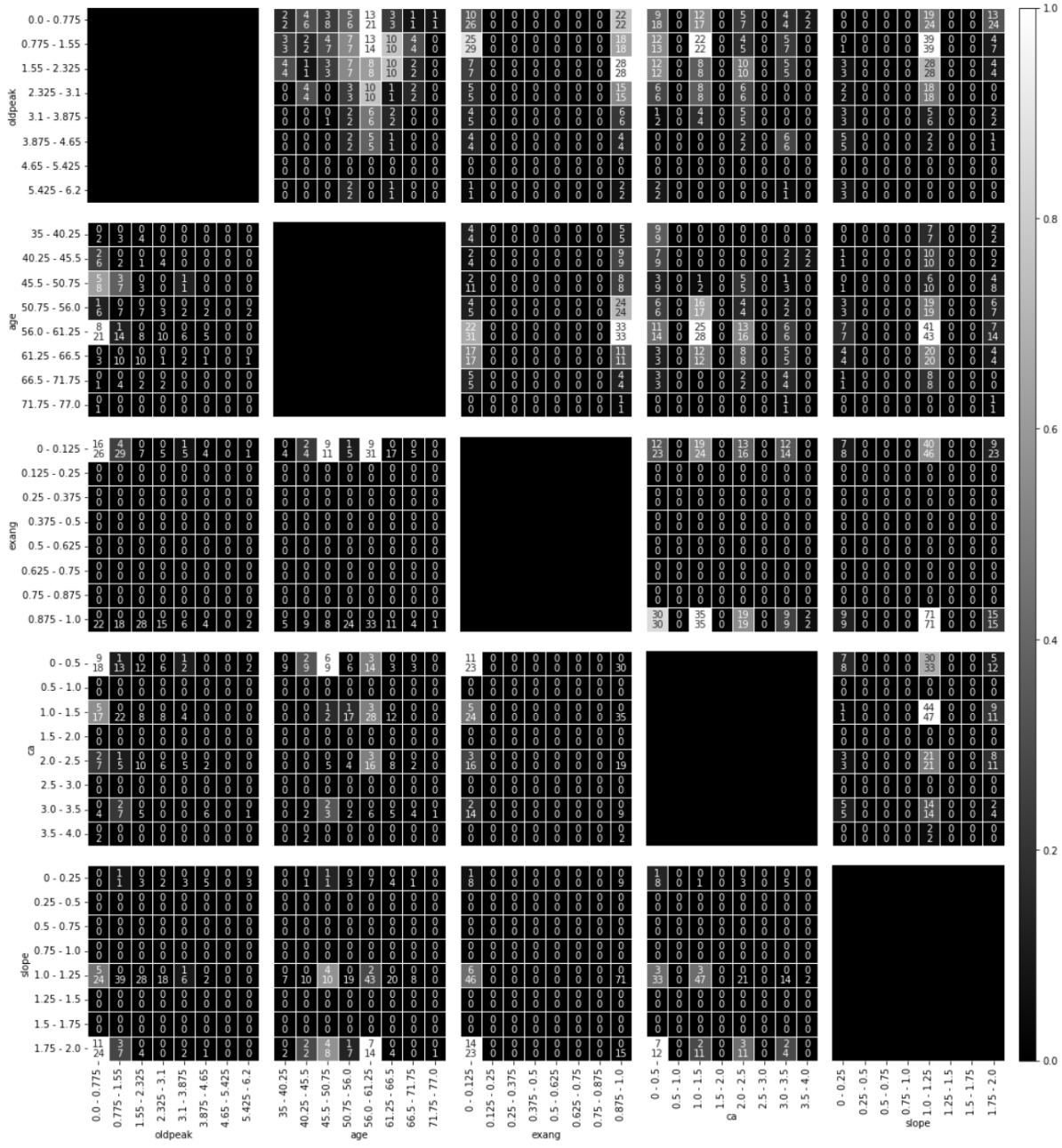


Figure 7 – 5x5 grid of two-feature error heatmaps displaying true negative (top right) and false positive (bottom left) error count (top number) and total value count (bottom number) at varying value combinations of the features on each axis. Lighter colour indicates higher objective error count within the individual heatmap.

3.1.3| Error Decision Tree

Any decision tree classifier can be utilised to create the error decision tree and multiple companies including Dataiku and Microsoft have their own native versions (9, 11).

The error decision tree was implemented using scikit-learn with a max depth of four, class weights balanced, minimum impurity decrease of 0.01 and minimum samples per leaf of 0.1. The tree was trained on the same input features as the classifier it was analysing but using the classifier's test dataset, with the target variable as the classifier's error category.

3.1.4| Slice Finder

Problematic slices are found utilising a lattice search, a breadth-first searching algorithm which ensures that simpler/more interpretable slices are examined first (6). The algorithm first traverses the shallowest row of slices which contains all independent features, as shown in Figure 8. If these features are categorical (e.g. sex = male/female) each category was treated as a slice. Continuous features (e.g. age = 1,2,3…) were binned into value ranges (e.g. age = [1 – 5].

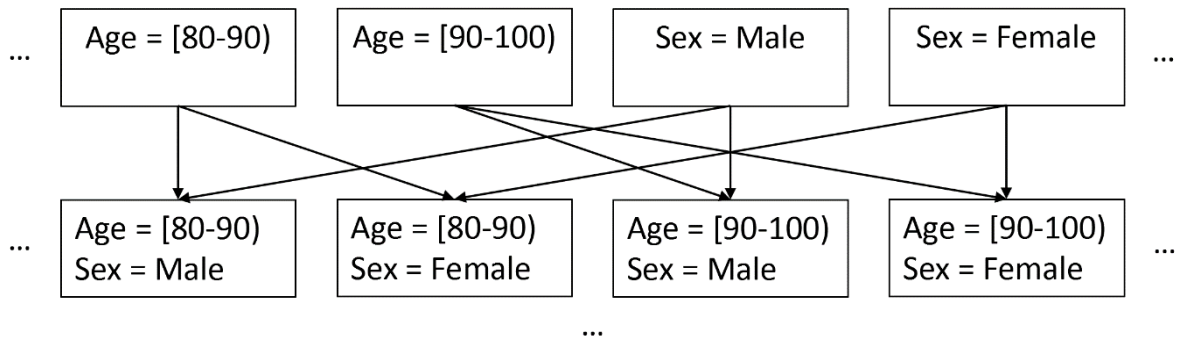


Figure 8 – Lattice structure search space involving categorical feature "Sex" and part of continuous feature "Age". Top row exclusively contains independent feature subsets while second row contains two-feature subsets (adapted from Slice Finder example).

The effect size (see Equation 3 (below)) of these slices are then all calculated, and those with an effect size above a user-determined threshold are recorded and considered “candidate” slices. A Welch’s t-test (Equation 4) is then used to identify whether the mean loss of each individual candidate slice is significantly higher than its complement (the whole dataset excluding datapoints in the candidate slice). Candidate slices with significantly higher mean loss are stored as “problematic” slices. All feature values found in any of the problematic slices are then removed from the lattice (Figure 8). The algorithm then repeats, traversing and analysing the second row and so on, until the entire lattice is searched or there are no buckets remaining.

Equation 3 - Effect size

$$\emptyset = \sqrt{2} * \frac{\varphi(S,h) - \varphi(S',h)}{\sqrt{\sigma_S^2 + \sigma_{S'}^2}}$$

Where S is the set of points in the slice, S’ is the complement to S, $\varphi(S,h)$ is a classification loss (error) function (we used log loss) and σ_S^2 is the variance of individual point losses in S.

Equation 4 - Welch’s t-test

$$t = \frac{\mu_S - \mu_{S'}}{\sqrt{\sigma_S^2/|S| + \sigma_{S'}^2/|S'|}}$$

Where μ_S is the mean loss in S, σ_S^2 is the variance of individual point losses in S, and |S| is the number of points in S.

The "slice_finder.py" python program from Yeounoh Chung's Slice Finder Github repository (16) was implemented. The default loss metric (log loss) was utilised and the effect size and subset size cut-offs were 0.5 and 5 respectively.

3.2| Dataset

A 1988 heart disease dataset was used in this study (17). It contains 308 data points and 13 features excluding the (binary) heart disease label. The features and their abbreviation and units respectively (if appropriate) are age ("age", years), sex ("sex", M/F), chest pain category ("cp"), resting blood pressure ("trestbps", mmHg), serum cholesterol ("sc", mg/dl), fasting blood sugar ("fbs", mg/dl), resting electrocardiograph results ("restecg"), maximum heart rate ("thalach", bpm), exercise-induced angina ("exang", Y/N), ST depression induced by exercise relative to rest ("oldpeak"), peak exercise ST segment slope ("slope"), number of major vessels coloured by fluoroscopy ("ca") and thalassaemia ("thal").

3.3| Classifier Model

We utilised a random forest model as the binary classifier to be analysed. The random forest was created using scikit-learn. GridSearchCV was utilised for hyperparameter tuning, with number of trees as 10, 20, 30 so on to 80, max features using "auto" and "sqrt", max depth of two and four, minimum samples to split of 2 and 5, minimum samples at each leaf node of 1 and 2, and bootstrap using "True" and "False". Train/test split was 70%/30%.

3.4| Experimental Design

The experiment is comprised of two components – “demonstration” and “test of efficiency with artificial test set”. The demonstration component involved using each error analysis method on the RF heart test dataset output. The aim of the demonstration component was to demonstrate the utility of each error analysis method. The efficiency test component involved creating copies of the RF heart disease test dataset that contained artificially misclassified 100% error subsets of differing sizes. The aim of the efficiency test was to test and compare how capable each method is of locating each of the differently sized 100% error subsets.

3.4.1| Preparation

Firstly, the random forest was trained on the heart disease dataset. Input features were “age”, “sex”, “cp”, “trestbps”, “sc”, “fbs”, “restecg”, “thalach”, “exang”, “oldpeak”, “slope”, “ca” and “thal” and target variable was the heart disease label. All 308 data points were utilised. The random forest was then used to classify each data point in the test dataset and the error category for each data point was recorded.

3.4.2| “Demonstration” Procedure

The Slice Finder and error decision tree methods were then run on the actual negative (TN/FP) and actual positive (TP/FN) data points individually and problematic subsets for each were noted. For the sake of brevity, only the top five most problematic features – three from SF and two from EDT – from the actual negative and actual positive groups were then used to create a 5x5 heatmap and 5x2 OFED for each group.

3.4.3| “Test of Efficiency with Artificial Test Set” Procedure

Subsets of the actual negative (TN/FP) data points were artificially misclassified such that three datasets were created. The original dataset, an “artificial 1” dataset in which all data points in the subset “1.55 < oldpeak < 2.625” were set as false positives and an “artificial 2”

dataset in which all data points in the subset “1.55 < oldpeak < 2.625” that were also found in the subset “56 <= age < 61.25” were set as false positives. Six “oldpeak” and “age” heatmaps were created – a TN and FP heatmap for each dataset, six OFEDs were created – an “oldpeak” and “age” OFED for each dataset – and SF and EDT were run on each dataset.

4| Results

The results section is comprised of two main subsections – “demonstration” and “test of efficiency with artificial test set”. The “demonstration” subsection contains the results from utilising the OFED, HM, EDT and SF methods on all four error categories (TN/FP/TP/FN) from the random forest binary classifier’s original heart disease test dataset output. The “test of efficiency with artificial test set” subsection contains the results from using the same methods on only the actual negative values (TN/FP) in same classifier’s original, “artificial 1” and “artificial 2” test dataset outputs.

4.1| Demonstration

This section outlines the results from the OFED, HM, EDT and SF methods being utilised on both actual negative (TN/FP) and actual positive (TP/FN) data points in the random forest binary classifier’s original dataset test output. For the sake of brevity only OFEDs and HMs of the top five most problematic features (as identified by EDT and SF) are displayed.

4.1.1| One-Feature Error Distribution

The output of using the OFED method on both actual negative (TN/FP) and actual positive (TP/FN) data points is shown in Figure 9. For the true negative/false positive distributions, “slope” had a JSD of 0.2 and max difference of 0.28 at “1”, “age” had a JSD of 0.36 and max difference of 0.20 at “45.5 – 50.75”, “cp” had a JSD of 0.5 and max difference of 0.63 at “0”,

“restecg” had a JSD of 0.09 and max difference of 0.02 at “2” and “oldpeak” had JSD of 0.38 and max difference of 0.26 at “0”.

For the true positive/false negative distributions, “slope” had a JSD of 0.17 and max difference of 0.21 at “2”, “thal” had JSD of 0.58 and max difference of 0.77 at “3”, “thalach” had JSD of 0.46 and max difference of 0.35 at “135.8 - 149”, “oldpeak” had JSD of 0.39 and max difference of 0.15 at “1.050 – 1.575” and “age” had JSD of 0.45 and max difference of 0.32 at “58.38 – 64.25”.

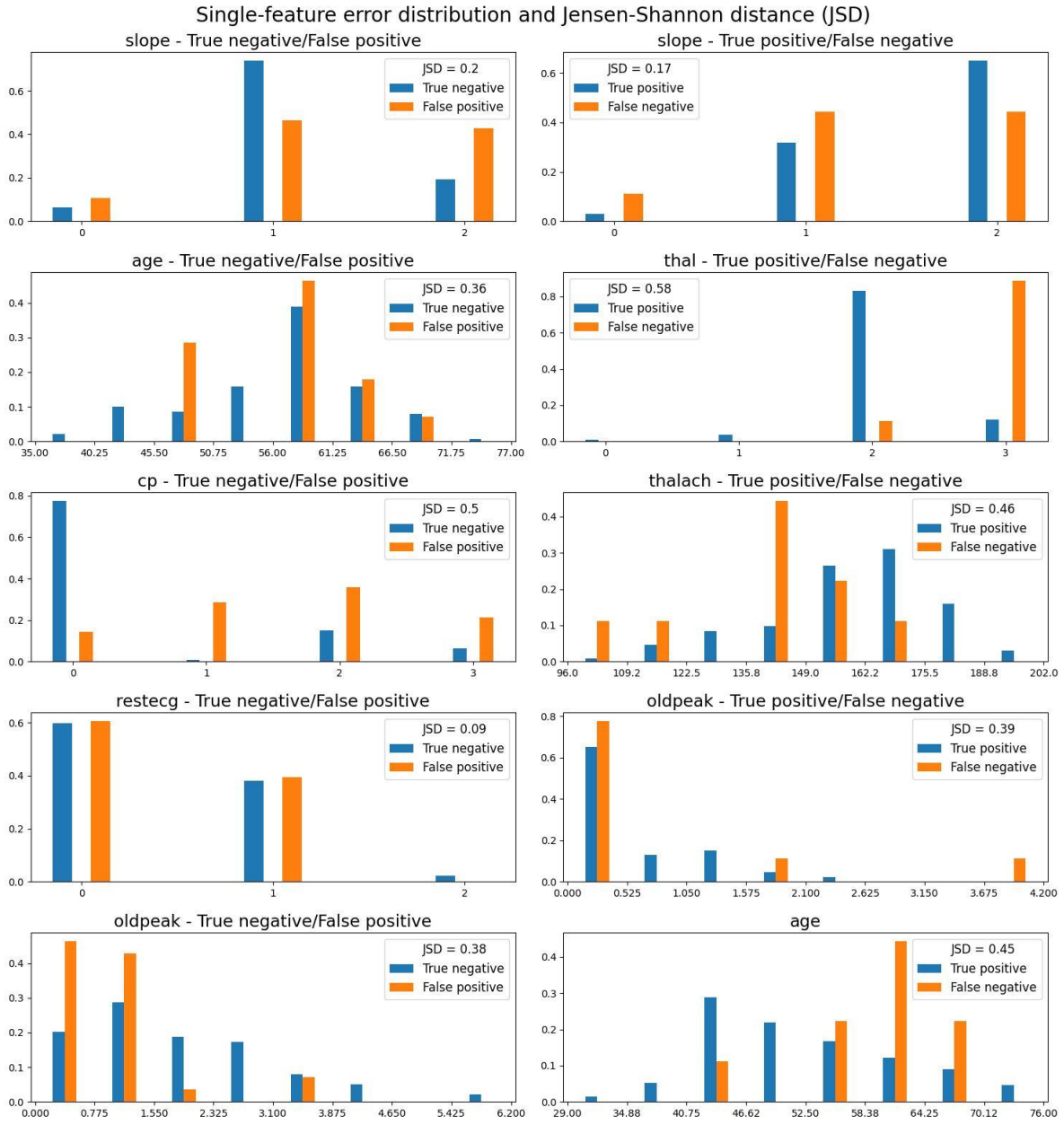


Figure 9 – 5x2 grid of column graphs showing fraction (Y-axis) of “True negative” (blue) and “False positive” (orange) (left column) and “True positive” (blue) and “False negative” (orange) (right column) occurrences at varying values of the features listed above the graphs (X-axis), and Jensen-Shannon distance between the distributions.

4.1.2.1| Heatmap – true negative/false positive

The output of using the HM method on actual negative (TN/FP) data points is shown in Figure 10. For actual negative data points, the highest observed subset FP count was 13 in the “0 <= oldpeak < 0.775”∩“0 <= restecg < 0.25” subset.

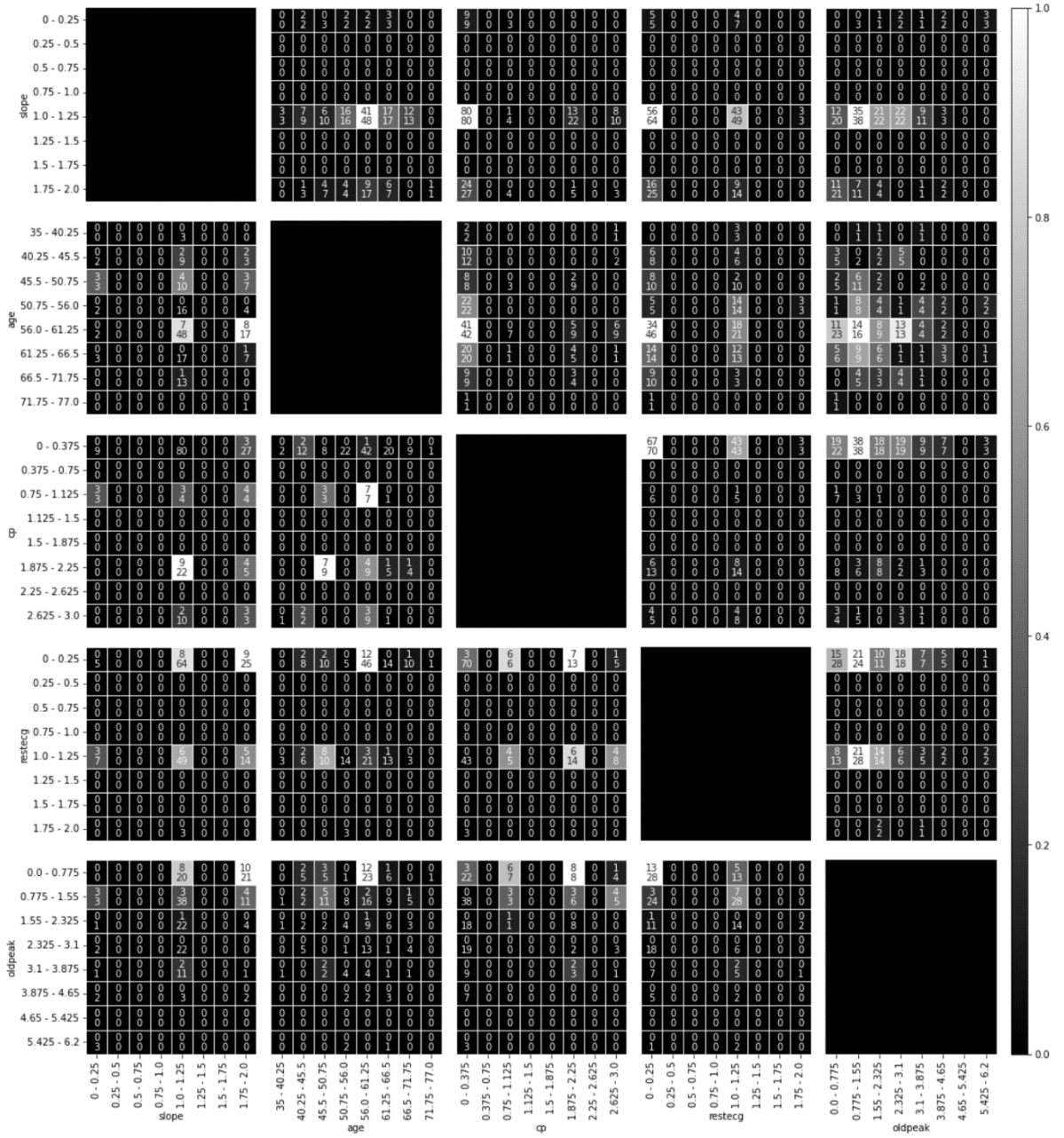


Figure 10 – 5x5 grid of two-feature error heatmaps displaying true negative (top right) and false positive (bottom left) error count (top number) and total point count (bottom number) at varying value combinations of the features on each axis. Lighter colour indicates higher objective error count within the individual heatmap.

4.1.2.2| Heatmap – true positive/false negative

The output of using the HM method on actual positive (TP/FN) data points is shown in Figure 11. For actual positive data points, the highest observed subset FN count was 7 in the "0 <= oldpeak < 0.525"∩"2.625 <= thal < 3.0" subset.

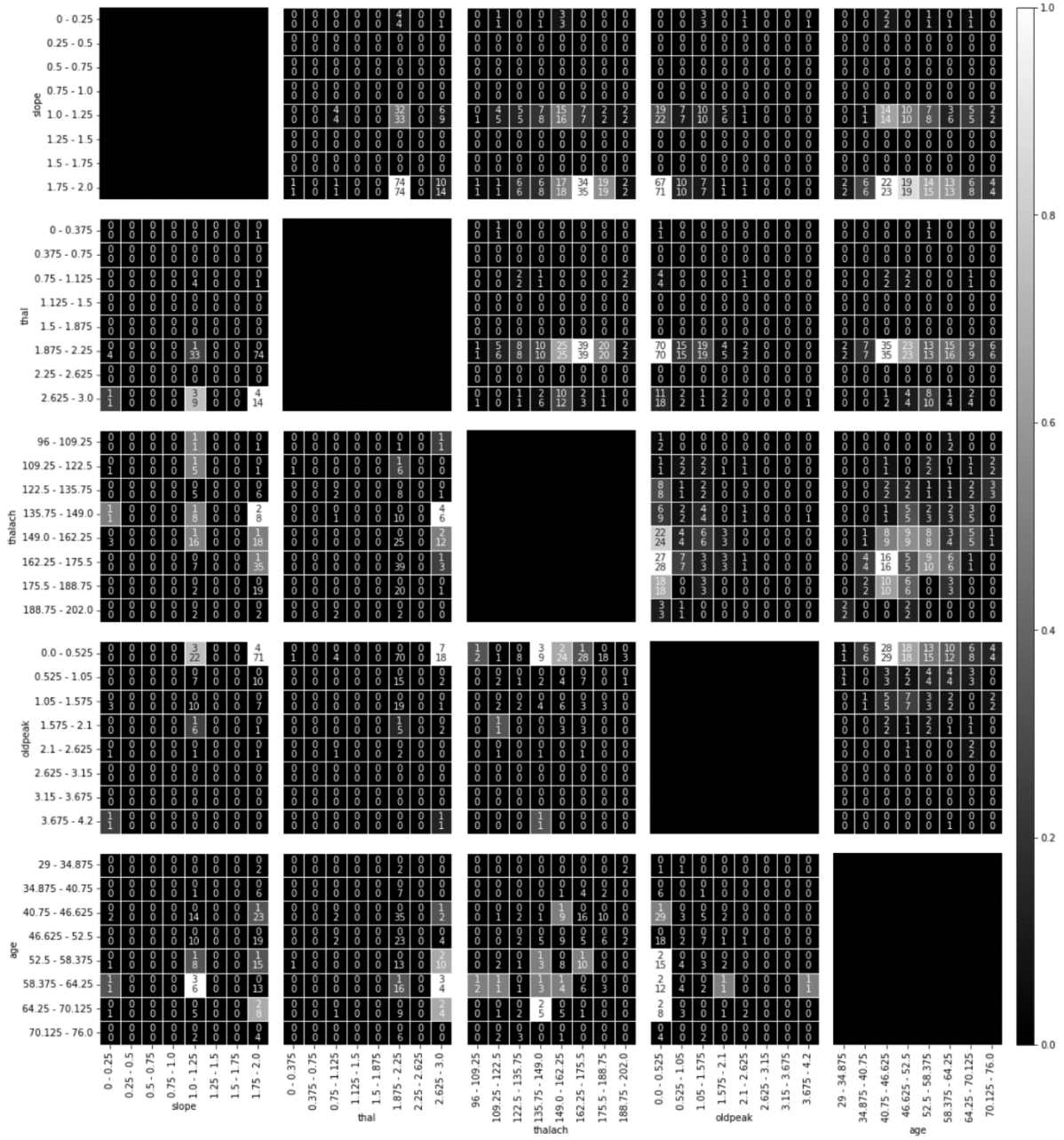


Figure 11 – 5x5 grid of two-feature error heatmaps displaying true positive (top right) and false negative (bottom left) error count (top number) and total point count (bottom number) at varying value combinations of the features on each axis. Lighter colour indicates higher objective error count within the individual heatmap.

4.1.3.1| Error decision tree – true negative/false positive

As shown in Figure 12, when trained on the true negative/false positive RF test output, the EDT identified three problematic subsets. The subset of “cp > 0.5” combined with “oldpeak <= 1.3” contained 19.2% of samples and was mostly false positives, the subset “cp” > 0.5 and “oldpeak” > 1.3 contained 11.4% of samples and was roughly half false positives and the subset “cp <= 0.5” and “oldpeak <= 0.3” contained 10.8% of samples and was also roughly half false positives.

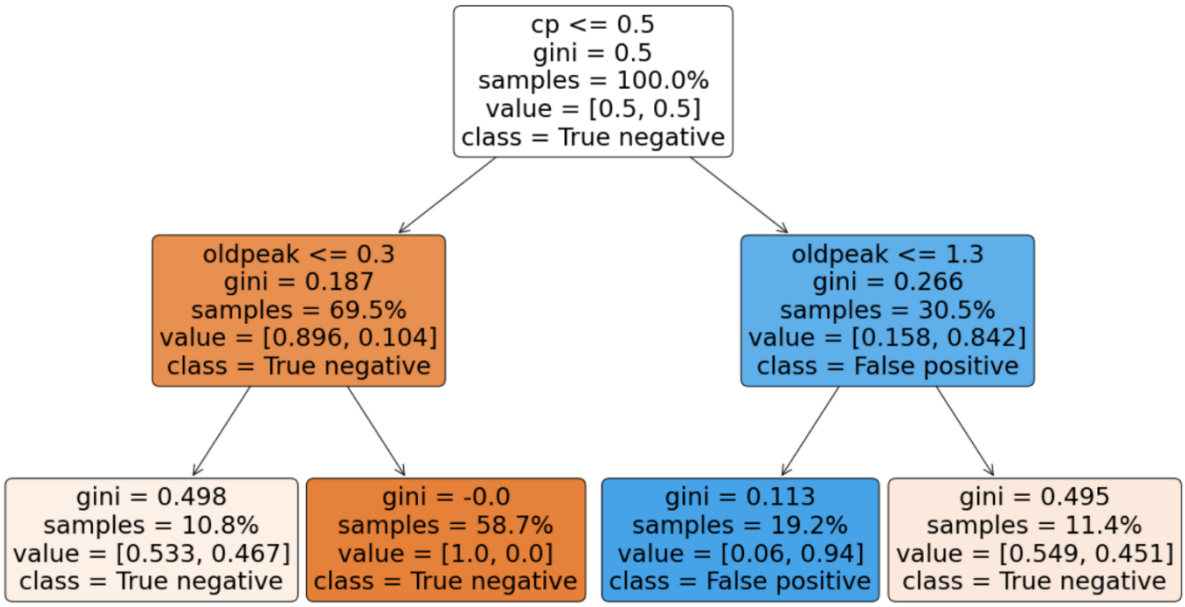


Figure 12 – Scikit-learn decision tree model trained on RF test output to classify actual negative data points as either “True negative” or “False positive”.

4.1.3.2| Error decision tree – true positive/false negative

As shown in Figure 13, when trained on the true positive/false negative RF test output, the EDT identified two problematic subsets. The subset of “thal > 2.5” contained 17% of samples and mostly false positives while the subset “thal <= 2.5” and “thalach <= 131.5” contained 11.3% of samples and was roughly half false positives.

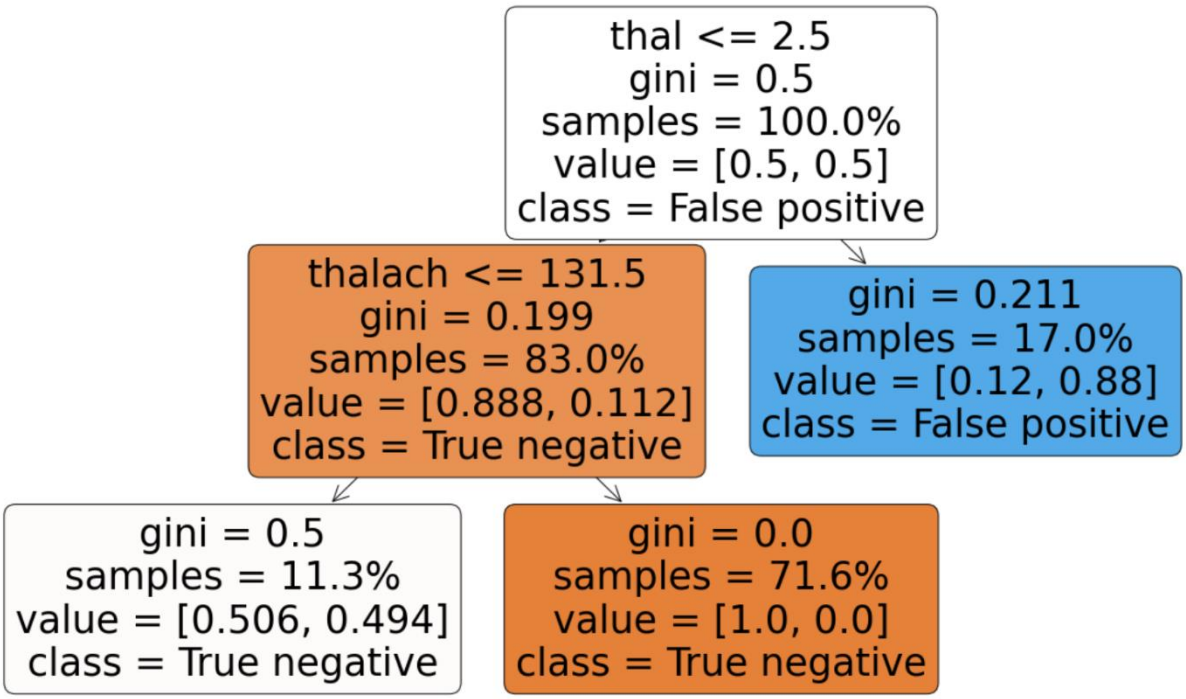


Figure 13 – Scikit-learn decision tree model trained on RF test output to classify actual positive data points as either “True positive” or “False negative”.

4.1.4| Slice Finder – true negative/false positive

The output of using the SF method on actual negative (TN/FP) and actual positive (TP/FN) data points is shown in Table 1. TN/FP problematic slices were “slope = 0”, “age = 48 – 50” and “cp = 1 ∩ restecg = 1” and the only TP/FN problematic slice was “slope = 2”.

Table 1 – “Problematic” slices in RF test output. Slices in actual negative (TN/FP) and actual positive (TP/FN) groups identified by Slice Finder to have size > 5 and effect size >= 0.5.

	Feature	Value	Size	Effect size
TN/FP	slope	0	12	0.97
	age	48 – 50	11	0.66
	cp restecg	1 1	5	0.5
TP/FN	Slope	2	90	0.70

4.2| Test of Efficiency with Artificial Test Set

This section shows the results from utilising the OFED, HM, EDT and SF methods on actual negative (TN/FP) data points in the random forest binary classifier’s original, “artificial 1” and “artificial 2” test datasets’ outputs. Only OFEDs and HMs of the “oldpeak” and “age” features are displayed.

4.2.1| One-Feature Error Distribution

“Oldpeak” and “Age” OFED results from the efficiency test are shown in Figure 14. For “oldpeak”, JSD is 0.38 (max difference 0.28 at “0”) in the original dataset, increases to 0.55 (max difference 0.47 at “2”) in the “artificial 1” dataset and decreases to 0.37 (max difference 0.28 at “0”) in the “artificial 2” dataset.

For “age”, JSD is 0.36 (max difference 0.32 at “45.5 – 50.75”) in the original dataset, decreases to 0.19 (max difference 0.14 at “45.5 – 50.75”) in the “artificial 1” dataset and increases to 0.40 (max difference 0.24 at “56.0 – 61.25”) in the “artificial 2” dataset.

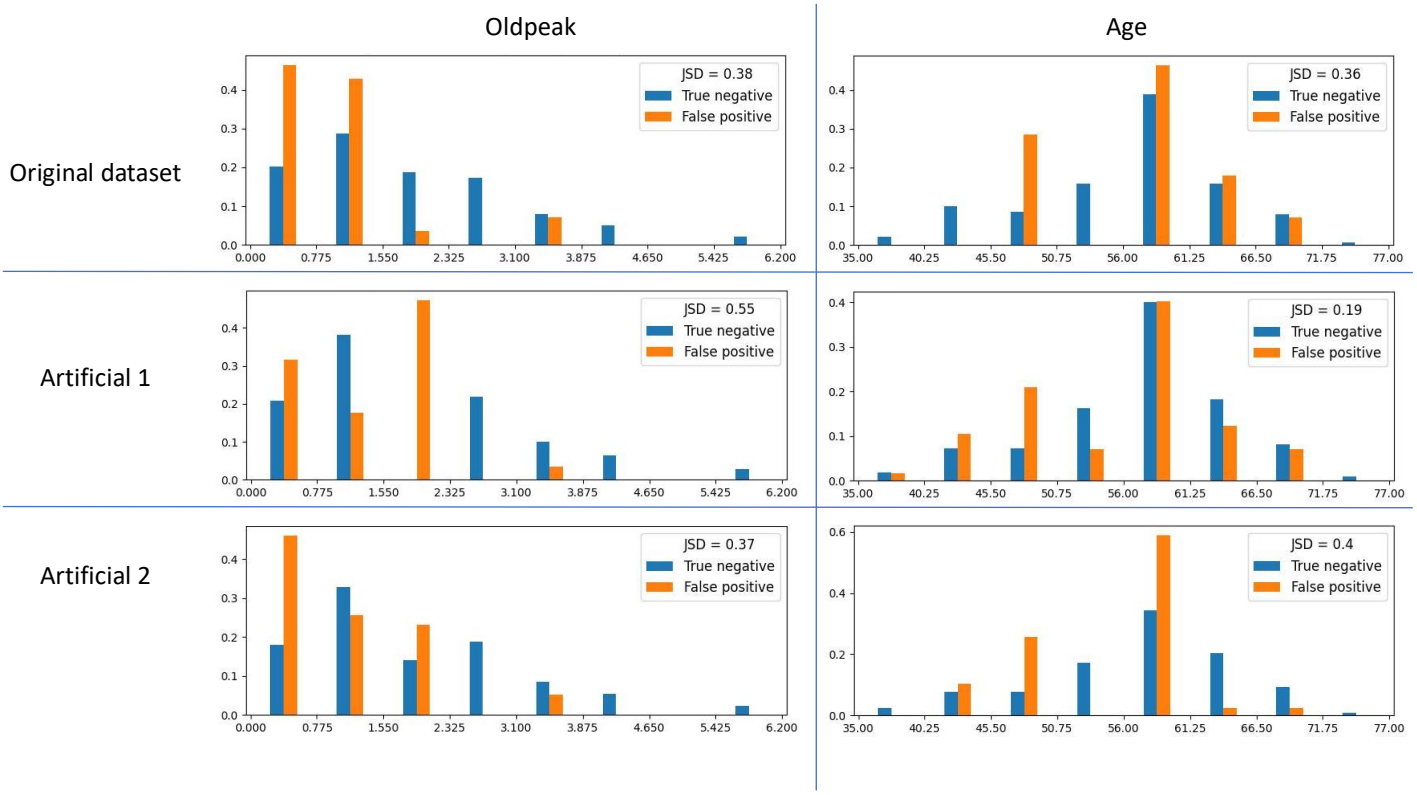


Figure 14 – 3x2 grid of column graphs showing fraction (Y-axis) of “True negative” (blue) and “False positive” (orange) occurrences in the original dataset (top row), “artificial 1” condition (middle row) and “artificial 2” condition (bottom row) at varying values of “Oldpeak” (left column) and “Age” (years)(right column) features (X-axis), and Jensen-Shannon distance (JSD) between the distributions.

4.2.2| Heatmap

“Oldpeak” and “age” heatmap results from the efficiency test are shown in Figure 15. As expected, the false positive error count in the “1.55 < oldpeak < 2.325” subset was 1 out of 27 in the original dataset, 27 out of 27 in the “artificial 1” dataset and 9 out of 27 in the “artificial 2” dataset, all 9 located within the “56.0 <= age < 61.25” subset (Figure 15). The inverse occurred for true negative count in the same subsets.

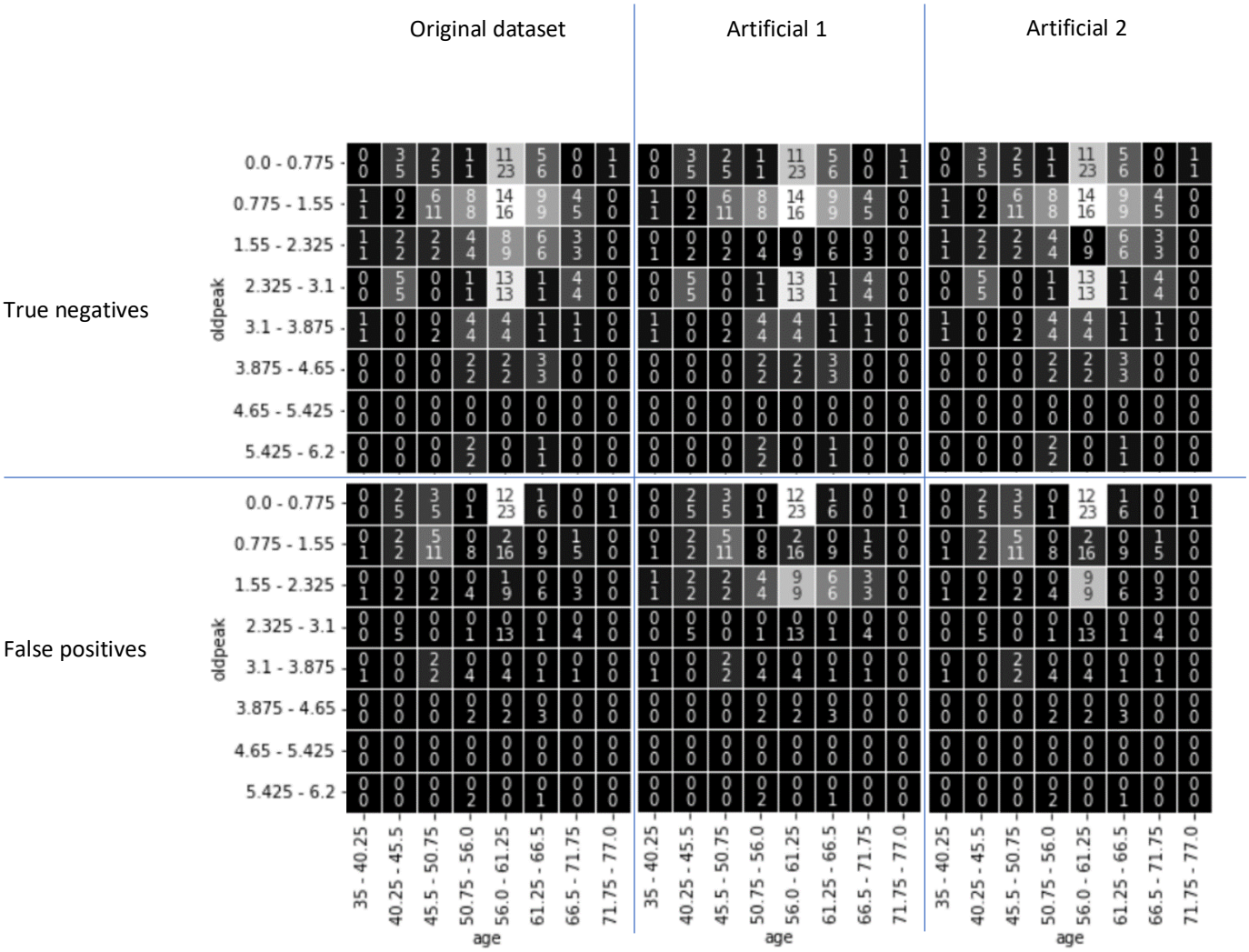


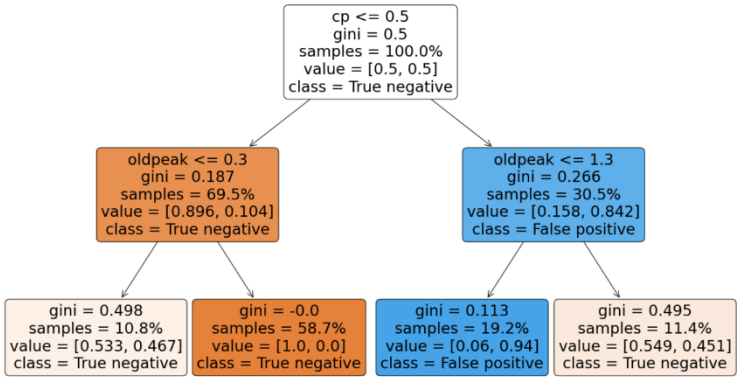
Figure 15 – 2x3 grid of two-feature error heatmaps displaying true negative (top row) and false positive (bottom row) count (top number) and total point count (bottom number) at varying value combinations of the features “oldpeak” (y-axis) and “age” (x-axis) in the original dataset (left), “artificial 1” condition (middle) and “artificial 2” condition (right). Lighter colour indicates higher objective error count within the individual heatmap.

4.2.3| Error decision tree

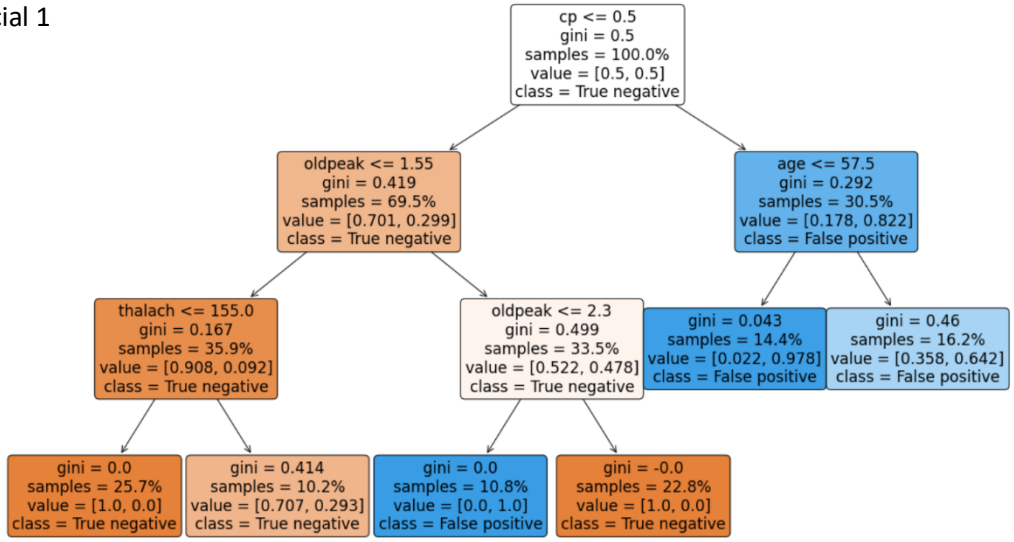
EDT results from the efficiency test are shown in Figure 16. The “artificial 1” EDT somewhat effectively identifies the “artificial 1” (“oldpeak”) subset by utilising the bounds “1.55 < oldpeak <= 2.3” within the “cp <= 0.5” subset. The “artificial 2” EDT does not effectively identify the “artificial 2” (“oldpeak” and “age”) subset, making no reference to the “age”

feature and only utilising the “oldpeak <= 1.1” bound in a manner similar to the original dataset EDT.

Original dataset



Artificial 1



Artificial 2

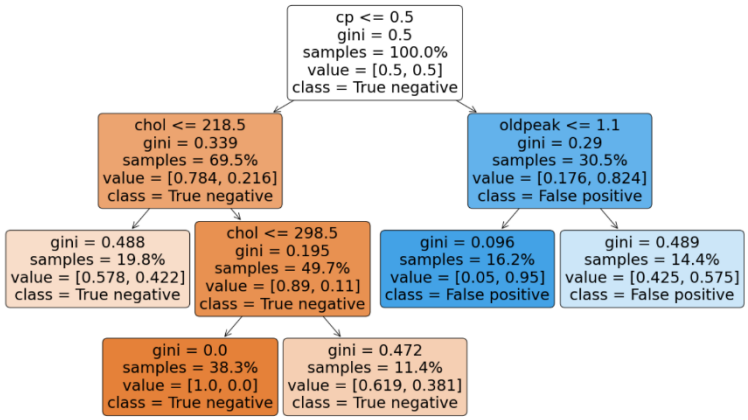


Figure 16 – Scikit-learn decision tree models trained on RF test output to classify actual negative data points as either “True negative” or “False positive” in the original dataset (top), “artificial 1” condition (middle) and “artificial 2” condition (bottom).

4.2.4| Slice Finder

Slice Finder results from the efficiency test experiment are shown in Table 2. Slice Finder fairly effectively identifies the “artificial 1” (“oldpeak”) subset by identifying two slices that encapsulate the subset – “1.4 <= oldpeak <= 1.8” and “1.9 <= oldpeak <= 2.4”, slightly overshooting the actual bounds (“1.55 < oldpeak < 2.325”). Slice Finder also somewhat effectively half identifies the “artificial 2” (“oldpeak” and “age”) subset by identifying the “55 <= age <= 56” subset but making no reference to the “oldpeak” feature.

Table 2 – “Problematic” slices in RF test output actual negative (TN/FP) data points. Slices identified by Slice Finder to have size > 5 and effect size >= 0.5. in “original”, “artificial 1” and “artificial 2” datasets. Slices containing values within artificially misclassified subsets are bolded.

	Feature	Value	Size	Effect size
Original dataset	slope	0	12	0.97
	age	48 – 50	11	0.66
	cp	1	5	0.50
	restecg	1		
Artificial 1	**oldpeak**	**1.9 – 2.4**	19	**0.76**
	oldpeak	**1.4 – 1.8**	**19**	**0.51**
	slope	0	12	0.62
	age	48 – 50	11	0.54
Artificial 2	**age**	**55 – 56**	**15**	**0.78**
	slope	0	12	0.88
	age	48 – 50	11	0.64

5| Discussion

5.1| Discussion

This study aimed to outline and test subset error analysis methods on a random forest binary classification algorithm. Slice Finder, error decision tree, one-feature error distribution and two-dimensional error heatmap methods were examined. Our key findings were A) The Slice Finder and error decision tree methods are effective at quickly identifying error subsets at macro scale but are imprecise and can give inadequate context at the subset level while the OFED and HM methods provide are inefficient at macro scale but provide superior lower-level context/detail and B) utilising a combination of the macro-scale SF/DT methods and micro-scale OFED/HM methods would be the optimal manner in which to perform comprehensive error analysis of binary classifiers.

Slice Finder uses objective statistical techniques to identify "problematic" slices and priorities simpler slices to promote interpretability. SF has multiple strengths. Due to being automated, after being set up only one "click" is required to identify problematic subsets making it useful for quick debugging/analysis. This is especially advantageous for datasets with large numbers of features in which examining the error counts manually is impractical/arduous. Also, the criteria of subset analysis – loss metric, slice size and effect size – is customisable, allowing the user to identify slices based on their needs. However, Slice Finder has limitations. It identifies specific problematic subsets without providing much context about the surrounding data. This is somewhat problematic because for continuous variables SF's bucketing technique will often combine multiple consecutive data values into one slice, assessing the slice's effect size over this heterogeneous subset. For instance, as seen in Table 2, SF indicates that values in the range "1.4 < oldpeak < 1.8" are problematic despite only values in the range "1.55 < oldpeak < 2.325" being 100% errors. SF's bucketing includes the "oldpeak" value of 1.4 despite the value not lying in the problematic subset. As a result, the user is given imprecise information about the problematic subset which could lead to incorrect conclusions.

Similarly to SF, the error decision tree attempts to identify problematic subsets automatically. However, in the configuration utilised in this report, the methods tend to identify slightly differing subsets. SF favours cherry-picking smaller, single-feature problematic subsets while the EDT tends toward identifying multi-feature subsets, as can be observed in Figure 16 and Table 2. It has similar upsides to SF – it is automatic, requiring only one "click" and is customisable (information gain criteria, depth, class weights etc.) to suit the user's needs. However, the EDT does not seem to suffer from the bucketing issue that SF does. This can be observed in the "artificial 1" subset identification where the EDT effectively outlines the problematic subset (Figure 16, middle) while, as previously mentioned, SF does not (Table 2). However, the EDT does not seem to be as sensitive to smaller 100% error subsets as SF, as is observable in their respective analyses of the "artificial 2" dataset. The EDT completely fails to identify the artificially misclassified subset (Figure 16, bottom) while SF does identify the "age" feature (Table 2). The EDT also provides more context about problematic subsets than the SF method, such as error count within the subset and how problematic the remainder of the data is (other leaves).

Unlike SF and EDT, the two-dimensional error heatmap method does not automatically identify problematic slices. Rather, it displays a graphic showing error count distribution among subsets of two-feature combinations which require the user to manually inspect/search. As a result, the primary downside of the HM is that it is relatively arduous/time-consuming, which scales poorly with the number of features in a dataset. A smaller downside of the HM is that it is only capable of identifying subsets with two features or fewer, unlike SF and EDT which can be configured to identify multi-feature subsets. The primary upside of the HM is the easily interpretable display of two-dimensional data – If the user is aware of which features are of interest to them the HM provides a comprehensive and easily interpretable overview of error layout within the feature(s). This is observable in the false positive heatmaps (bottom row) in Figure 15, where the artificially misclassified subsets are immediately obvious. The HM also allows for exploratory error analysis – the user is free to examine the HM of any features/combination of features, which could be useful if wanting to examine unproblematic features.

Similarly to the HM, the one-feature error distribution with Jensen-Shannon distance method does not automatically identify problematic subsets of the data. Instead, it displays a visual overview of error distributions within individual features and provides a metric (JSD) to quantify the difference. The primary upside of OFED is that it provides similarly effective visual analysis of error subsets within one feature as the HM method does, but has a metric attached to it (JSD) that gives an indication of how problematic a feature likely is without the need for visual inspection. As a result, the OFED method is less susceptible to the time consumption issues that the HM method presents. However, the clear downside is that the JSD metric becomes ineffective when subsets involve more than a single feature. As demonstrated in the "oldpeak" graphs (left column) in Figure 14, the JSD metric increases considerably from the original dataset to the "artificial 1" condition, as the artificially misclassified subset involves only one feature ("1.55 < oldpeak < 2.325"). However, when comparing the original dataset to the "artificial 2" condition, there is no increase in JSD because the artificially misclassified subset involves two features ("oldpeak" and "age") While this could be partially due to the artificially misclassified subset in the "artificial 2" condition being smaller, if the change in subset size was solely responsible for the difference, the JSD would show a smaller but nonetheless present increase from the original to the "artificial 2" dataset, rather than slightly decreasing.

While each of these error analysis methods has utility individually, it appears that combining multiple methods can provide more effective analysis than any individual method. One potential combination with obvious utility is using the SF and/or EDT method in combination with the HM. As outlined, while the SF and EDT methods have some differences, both are able to quickly identify problematic subsets, but do not provide much context. Conversely, the HM method provides a more complete picture about the error distribution breakdown but is arduous and time consuming to use. Therefore, the SF and/or EDT method can be used to identify problematic subsets, then the HM of the features in those subsets can be created. As a result, the benefits of both methods are gained – the comprehensive analysis of HM and the speed of SF and EDT. This was done in the "demonstration" experiment – problematic features from SF (Table 1) and EDT (Figures 12 and 13) being used to create the HMs (Figures

10 and 11) and OFED (Figure 9), from which additional insight about the problematic subsets was gained.

Clearly, each of these error analysis methods provides unique utility. As such, the choice of which method a user should utilise for error analysis comes down to their individual needs. For example, if a user is in a high-paced environment and requires a method that will quickly identify any egregiously problematic error subsets but doesn't need to know the finer details, they might benefit from exclusively using the SF or EDT methods. However, a user attempting to maximally optimise performance who has ample time to do so would likely find a combination of methods more effective.

5.2| Limitations

The experiments in this paper, especially the "efficiency test" component, were only done once (n = 1) and only on one dataset, meaning there is a lack of objective statistical analysis/power and as such the findings should only be extrapolated with caution. Also, the buckets for HM and OFED methods may have been made too large and reduced resolution, meaning some smaller subsets were unable to be discriminated.

5.3| Future Directions

The efficiency test component could be repeated many times on differing datasets to improve statistical significance. The HM and OFED buckets could be reduced in size to improve subset boundary accuracy.

To improve efficiency further investigation could be done into optimal combinations of methods. Perhaps a single tool that combines the methods and has configurations based on the user's needs – for example a "fast" configuration and an "exhaustive" configuration.

The JSD metric could be integrated into the heatmap rather than just the OFED method.

6| References

1. Mendez KM, Reinke SN, Broadhurst DI. A comparative evaluation of the generalised predictive ability of eight machine learning algorithms across ten clinical metabolomics data sets for binary classification. Metabolomics. 2019;15(12):150.
2. Thomsen K, Christensen AL, Iversen L, Lomholt HB, Winther O. Deep Learning for Diagnostic Binary Classification of Multiple-Lesion Skin Diseases. Frontiers in Medicine. 2020;7.
3. Chen P, Pan C. Diabetes classification model based on boosting algorithms. BMC Bioinformatics. 2018;19(1):109.
4. Tufail AB, Ma Y-K, Zhang Q-N. Binary Classification of Alzheimer's Disease Using sMRI Imaging Modality and Deep Learning. Journal of Digital Imaging. 2020;33(5):1073-90.
5. Canbek G, Sagiroglu S, Temizel TT, Baykal N, editors. Binary classification performance measures/metrics: A comprehensive visualized roadmap to gain new insights. 2017 International Conference on Computer Science and Engineering (UBMK); 2017 5-8 Oct. 2017.
6. Chung Y, Kraska T, Polyzotis N, Tae KH, Whang SE, editors. Slice finder: Automated data slicing for model validation. 2019 IEEE 35th International Conference on Data Engineering (ICDE); 2019: IEEE.
7. Pastor E, de Alfaro L, Baralis E. Identifying Biased Subgroups in Ranking and Classification. arXiv preprint arXiv:210807450. 2021.
8. Gray NAB. Constraints on" learning machine" classification methods. Analytical Chemistry. 1976;48(14):2265-8.
9. Nushi B. Responsible Machine Learning with Error Analysis Microsoft Tech Community: Microsoft; 2021 [Available from: https://techcommunity.microsoft.com/t5/ai-machine-learning-blog/responsible-machine-learning-with-error-analysis/ba-p/2141774.
10. Endres DM, Schindelin JE. A new metric for probability distributions. IEEE Transactions on Information Theory. 2003;49(7):1858-60.
11. Dataiku. Model error analysis 2021 [Available from: https://doc.dataiku.com/dss/latest/machine-learning/supervised/model-error-analysis.html.

12. Chung Y, Kraska T, Polyzotis N, Tae KH, Whang SE. Automated Data Slicing for Model Validation: A Big Data - AI Integration Approach. IEEE Transactions on Knowledge and Data Engineering. 2020;32(12):2284-96.

13. Sagadeeva S, Boehm M, editors. Sliceline: Fast, linear-algebra-based slice finding for ml model debugging. Proceedings of the 2021 International Conference on Management of Data; 2021.

14. McCaffrey JD. Jensen-Shannon Distance Example 2021 [Available from: https://jamesmccaffrey.wordpress.com/2021/07/13/jensen-shannon-distance-example/.

15. MacKay DJC, Kay DJCM, MacKay JC, Press CU. Information Theory, Inference and Learning Algorithms: Cambridge University Press; 2003.

16. Chung Y. Slice Finder - GitHub Repository 2018 [cited 2022 12/06]. Available from: https://github.com/yeounoh/slicefinder.

17. Heart Disease Dataset. Kaggle: David Lapp; 2019.