Assignment 1

Exploration (Classification)

VU Machine Learning WS 2022

Group 37:

Branimir Raguž	|	12123474
Stefan Miljević	|	11721427
Rastko Gajanin	|	11930500

Table of Contents

Introduction

This report provides a detailed description of the experimentation process related to the first assignment of this course. The report contains not only the relevant information about the process, but also our considerations and justifications of multiple decisions that had to be made including the choice of the dataset, evaluation measures, preprocessing steps and further parameters. The implementation language is *Python* and we will predominantly rely on the *scikit-learn* and *pandas* libraries.

The structure of the report was inspired by the *CRISP Data Mining model*:

For each dataset we firstly describe it examine the data and attempt to understand it as best as possible (Subsection Exploration). The datasets and the respective URLs are listed in the Subsection Dataset sources. All facts presented in these subsections are the result of the *Exploratory Data Analysis* (See respective Source files.)

Based on this exploration, the data are cleaned and prepared for the modeling phase (Subsection Preprocessing). Furthermore, three model types that will be employed in the modeling step are **K-Nearest-Neighbors (KNN), Multilayer Perceptron (MLP)** and **Random Forests (RF).**

Subsection Modeling contains the details about the modeling process, including hyperparameter optimization techniques and implications of model types on the chosen datasets. After that, the performance results of the models are presented in the Subsection Performance Evaluation. The performance is evaluated with the following evaluation metrics: **Accuracy, Precision, Recall, F1 Score** and **Runtime (Efficiency)** Lastly, we offer a discussion of the obtained results along with further considerations and overall evaluation of the experiment process.

Purchase

Exploration

The purchase dataset contains data about the shopping behavior of customers. The columns denote the products available and are binary. I.e., if a customer bought the product the cell value is set to 1, otherwise to 0. This means that all features are of *Nominal Type*. The target variable contains 100 classes which define 100 different buying types. For the simplicity of the models, we assume these buyer types are independent and mutually exclusive, even though that statement is false. The goal of our models will be to predict the buyer type based on the products a customer bought so far. Dataset dimensions are (10k x 602) **with** the *ID* and *class* columns.

We, further, found out that the dataset contains *no missing values* and *no duplicate rows.* After plotting the correlation matrix (Figure 1) we found out that the features are not at all strongly correlated. Even though the feature variables are categorical, the simple Pearson correlation matrix can give us reliable enough insight due to the binary nature of these variables.

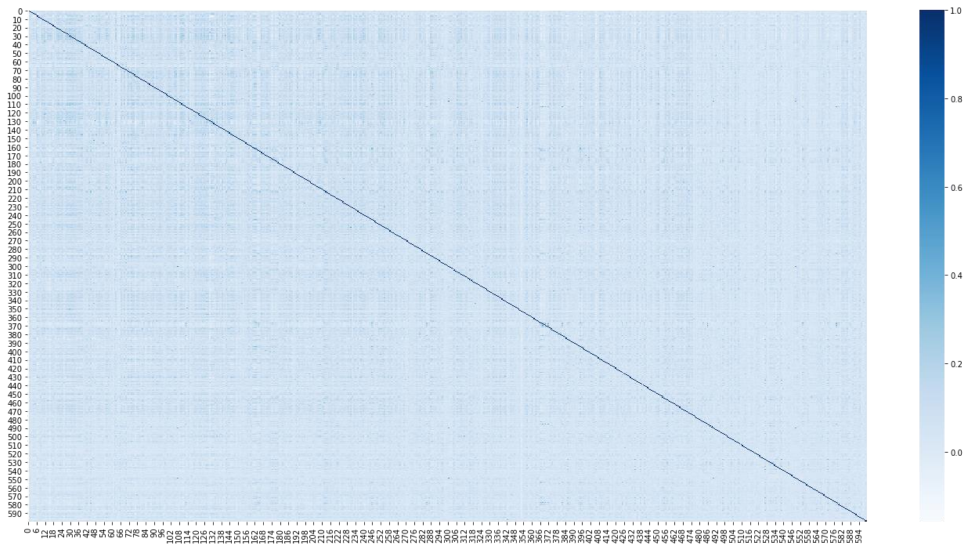

FIGURE 1: CORRELATION MATRIX OF THE PURCHASE DATASET FEATURES

The distribution of the target variable is presented in Figure 2. We observe that the target attribute is moderately imbalanced. This, along with the high cardinality of C_y (C_y – set of target classes) could pose certain challenges in the modeling process and could potentially have a negative impact on the accuracy of the trained models. To address this issue, the classes could be merged (binned) (e.g. 0-20, 20-50 etc.) to reduce the number of classes. Based on our assumption that the classes are independent this would however not make sense. If there was further information about the similarity of the classes, this would be a valid option.

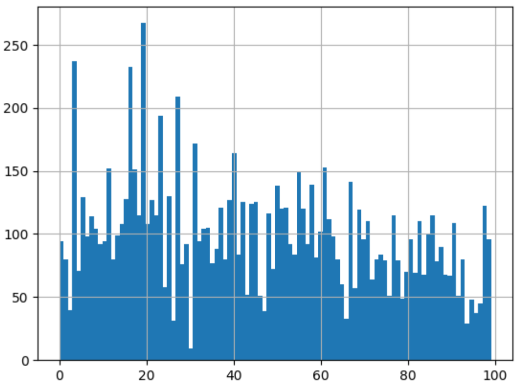

FIGURE 2 : PURCHASE DATASET TARGET DISTRIBUTION

Preprocessing

The **lrn** dataset was initially split into train and test datasets with the **80-20 split** strategy and the observation sampling was performed randomly with the **seed value set to 42.**

Since there are no missing values and no duplicates, no imputation nor row dropping was performed. All features are in {0, 1}, hence no min-max scaling was needed and since there are two feature classes, the feature variables typically follow a binomial (unimodal or bimodal) distribution, which implies that there is no need for standard scaling.

Principal Component Analysis (PCA) was carried out (only on the training set) with the goal of reducing dimensionality and removing noise from the dataset. Based on this analysis the scree plot was produced which shows how much variance is explained by the first 15 principal components (Figure 3).

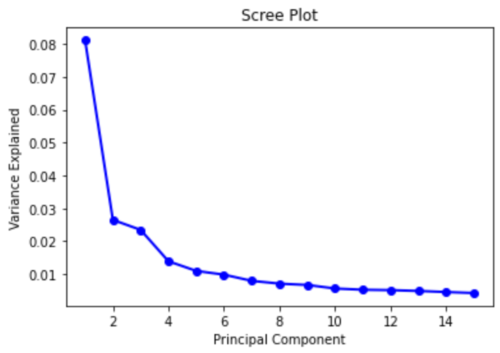

FIGURE 3: RESULTING SCREE PLOT OF PCA PERFORMED ON PURCHASE DATA

Since the features are mostly decorrelated, we see that the principal components themselves **explain very little variance** (~19.3% is explained by the first 10 PCs). Due to the lack of explanation for what each feature represents, we cannot really interpret the produced principal components. It could however be interesting to see what items influence them.

From the scree plot we can also see that the principal components beyond 10th carry very little information and could represent noise.

Therefore, based on the PCA results, we decided to experiment with training the model with the full training dataset as is and compare these results to the model trained with **only the first 10 principal components** extracted from the training set.

Classifiers

KNN

Since the goal is to predict the customer behavior based on the similarity of feature instances, we expect a decent result from this classifier since it is indeed based on the similarity measure. We will be measuring the accuracy achieved with two distance metrics: 'minkowski' which is the default and 'hamming' distance, because we believe it could bring improvement because of the binary (categorical) nature of the features.

For hyperparameter tuning we have decided to use Randomized Search with 5-fold cross validation with *accuracy* as the objective function and 7 samples for each parameter setting (*n_iter*). The runtime behavior of this method was the primary reason we have made this decision. The hyperparameter tuned was the *number of neighbors* and the potential values were:

[3, 5, 7, 9, 11, 13, 15]

After tuning the best accuracy (**60.35%**) was achieved with **15 neighbors**. Increasing the number of neighbors further only slightly increased the accuracy, therefore we chose 15 as the smaller number for performance reasons.

Random Forest

The random forest classifier might be useful for this dataset because it is robust against noise in the data. This is because we previously identified that the majority of PCs are noise.

The hyperparameter that was optimized is the *number of estimators* and the following values were considered:

[100, 125, 150, 175, 200, 250, 300]

RandomSearchCV was used once more with the same parameters as the one for KNN and after the optimization the model with **250** estimators achieved the highest accuracy (**67.45%**).

MLP

The MLP has the most promising properties for this dataset. This is predominantly attributed to the binary nature of the features and the capabilities of MLP to extract complex patterns from a high dimensional dataset such as this one. We expect this model to be the best performing one out of all three.

Hyperparameter tuning was carried out using RandomSearchCV again, but this time more samples were considered (`n_iter = 15`) and the following hyperparameters were tuned (with the respective values):

Size of the hidden layers: [(400, 200), (300, 150), (200, 100), (400), (300), (200), (100)]

L2 penalty parameter (α) : [0.0001, 0.0002, 0.0003]

Learning rate: [0.001, 0.002, 0.003]

Activation Function: ['relu', 'tanh', 'identity', 'logistic']

Weight optimizer: ['adam', 'lbfgs']

The best accuracy (**80.64%**) was achieved with the model that has one hidden layer with **300 neurons,** with alpha set to **0.0001**, **identity** activation function, learning rate **0.001** and **lbfgs** weight optimizer. We further extended the default maximum number of iterations to **500**, for the sake of solution quality.

Model comparison

All three classifiers have been trained with the best respective parameters found through hyperparameter optimization. Additionally, each classifier was trained with only the first 10 PCs extracted from the training set. The holdout method was also compared to cross validation for each combination. Lastly for the sake of validating our statement about scaling (i.e. that it is not necessary), we trained each classifier with a standard scaler applied beforehand. This would either confirm or falsify our claim.

The models were evaluated based on the evaluation metrics named in the Introduction. Precision, Recall and F1 score were **micro and macro averaged**, since there is so many target classes and their distribution is unbalanced. The summary of the results is presented in the Table 1. The value of the best achieved metric for each of the classifiers is marked in bold and the cell with best overall value across all classifiers is marked with green shading.

Algorithm	*Preprocessing*	*Validation*	*Accuracy*	*Precision*	*Recall*	*F1*	*Training Time*
MLP	PCA(n_components=10)	holdout	0.8005	**0.790092**	**0.787065**	**0.781938**	47.102698
MLP	PCA(n_components=10)	cv	**0.8150**	None	None	None	230.785442
MLP	StandardScaler()	holdout	0.6730	0.656155	0.645295	0.641652	9.740894
MLP	StandardScaler()	cv	0.6889	None	None	None	51.931343
MLP	None	holdout	0.7410	0.739019	0.727395	0.723729	25.889454
MLP	None	cv	0.7357	None	None	None	127.415929
RF	PCA(n_components=10)	holdout	0.6695	**0.676946**	**0.632915**	**0.634654**	14.132092
RF	PCA(n_components=10)	cv	**0.6846**	None	None	None	73.968327
RF	StandardScaler()	holdout	0.3195	0.298994	0.260758	0.23996	12.832227
RF	StandardScaler()	cv	0.3248	None	None	None	67.094736
RF	None	holdout	0.3170	0.315945	0.266936	0.251332	12.127202
RF	None	cv	0.3234	None	None	None	65.176207
KNN	PCA(n_components=10)	holdout	0.6160	**0.634503**	**0.581094**	**0.587542**	0.277100
KNN	PCA(n_components=10)	cv	**0.6188**	None	None	None	2.407242
KNN	StandardScaler()	holdout	0.2540	0.261608	0.225149	0.202575	0.128731
KNN	StandardScaler()	cv	0.2406	None	None	None	2.136305
KNN	None	holdout	0.3175	0.337902	0.289291	0.267579	0.021011
KNN	None	cv	0.2950	None	None	None	2.923312

TABLE 1: PERFORMANCE COMPARISON (MACRO AVG) (PURCHASE)

Discussion

We observe that PCA significantly impacts the models performance, in some cases it even “doubles” the accuracy. The MLP model has the best performance metrics both for micro and macro averaged trials, as it was expected. This however comes with the cost of significantly longer training time.

We further notice that the StandardScaler does not bring much when it comes to model performance, which supports the claim made in EDA.

Cross validation offers a slight improvement in the accuracy of the models, this however can be justified by the fact that the models “saw” more of the test data during their training when employing CV. On the other hand, when the holdout method is used, the test data has never been seen by the models. This results in slightly poorer accuracy of the models evaluated with the holdout method.

We further examined what is the class with the largest number of misclassifications. *Is it the class with most or least observations?....*

Breast Cancer

Exploration

This dataset contains various measurement data of patients that are suspected to have recurring breast cancer. The goal is to predict whether the patient has “recurrence-events” or not, which is expressed through the target variable. The features seem to represent three different statistics (*Mean, StdDev* and *Worst*) for the same sample of measurements (hence they all have numerical – continuous – type). This might be one of the reasons we see such high correlation between the certain features (presented in the Figure 5).

Furthermore, there are *no missing values* and *no duplicates* in the dataset.

The majority of features has a bell-shaped distribution (Figure 4) but with different value ranges, which indicates that a normalization would be reasonable. Some StdErr attributes are very right-skewed so a log transform might also be useful. This skew is typically caused by outliers, of which there are plenty in this dataset. These outliers will need to be addressed, as we will see in the Preprocessing step.

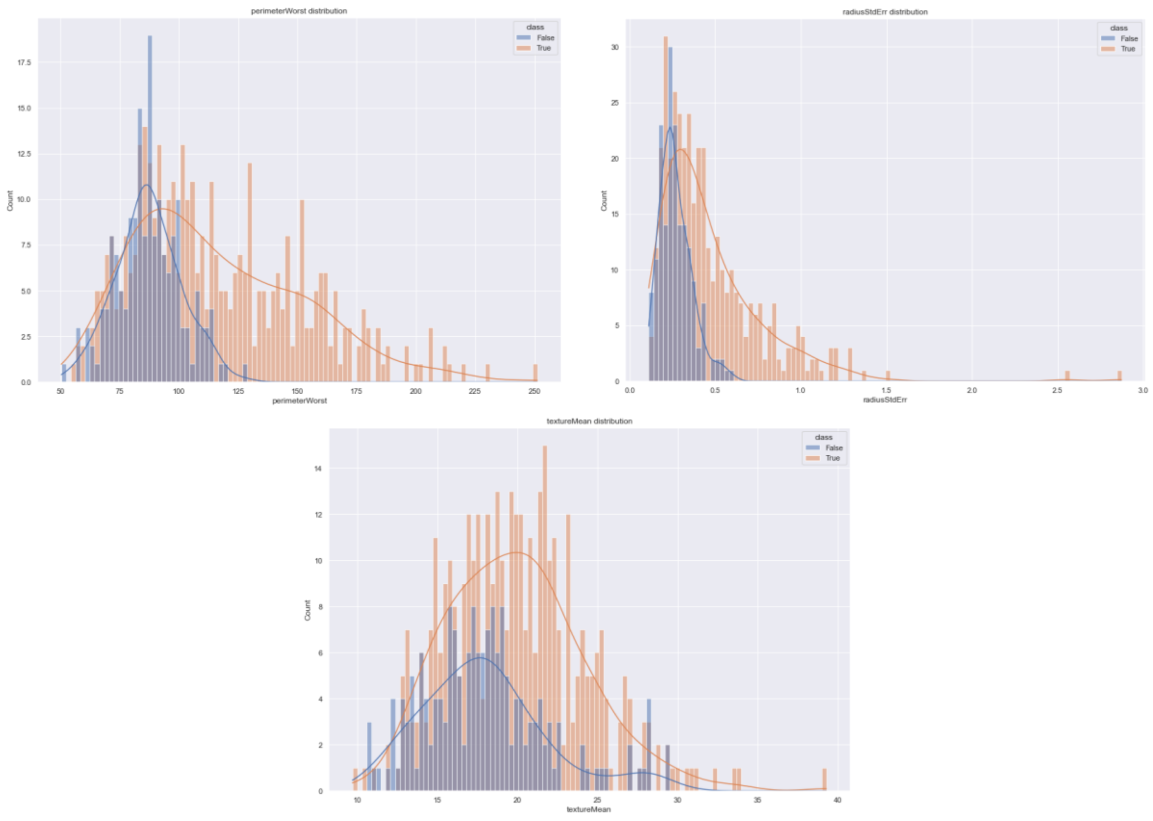

Figure 4: Distributions of perimeterWorst (top left), radiusStdErr (top right) and textureMean (bottom) Attributes grouped by class

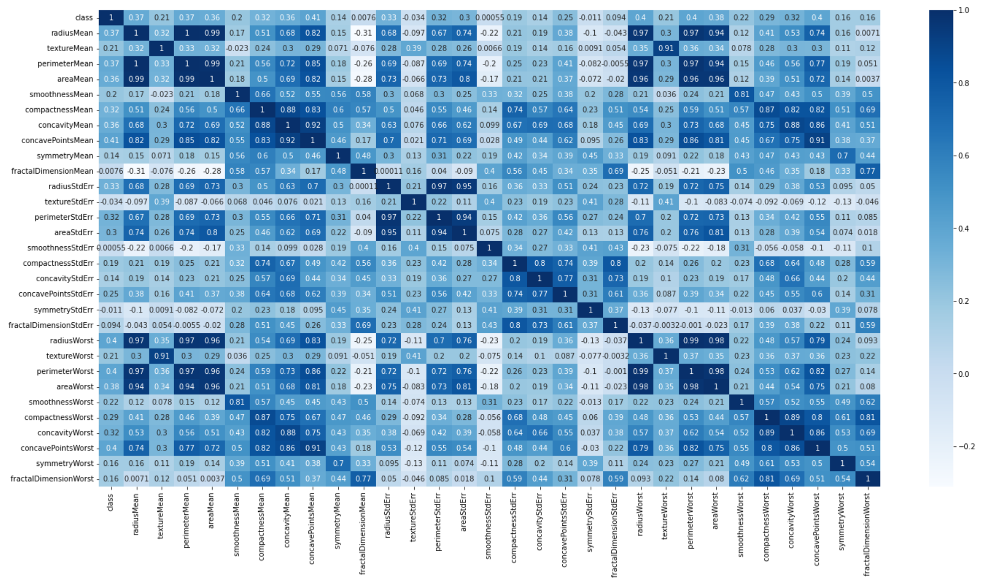

FIGURE 5: CORRELATION MATRIX OF THE BREAST CANCER DATASET

Preprocessing

Firstly, the dataset is split onto the train and test subsets using the 70-30 split.

No missing values should be cleaned, and no duplicates should be addressed.

Since the data have very different ranges, scaling is a necessity. Three different scalers will be employed and their influence on the performance will be observed. The scalers in question are: **Standard Scaler, Min-Max Scaler** and **Robust Scaler.** The latter is of particular value because of the outliers present in the dataset.

Since the features are numerical and the target variable is already encoded with 0/1 encoding, there is no need for additional encoding in this step.

PCA was also carried out since the correlation matrix indicates multiple features are correlated with each other. Removing these redundant (noisy) features should improve performance of the models. The PCA results produced the scree plot visible in Figure 6. We see that it has the potential of significantly reducing the dimension of the given dataset. This however is not as necessary because there are not that many features in the original dataset.

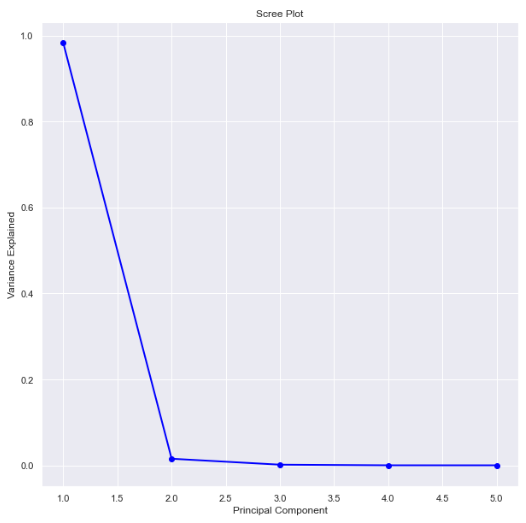

FIGURE 6: SCREE PLOT (BREAST CANCER TRAINING DATA)

Classifiers

KNN

The KNN classifier could be a good fit for this dataset because of the potentially present noise. Since there are not that many observations, the training time should not be long. Since this is a similarity-based classifier, it makes sense to employ it because similar cancer measurements should very well imply its future behavior.

RF

The identical argumentation can be provided for the random forest classifier; however, the RF would select the most informative features. *The results of RFs could further be interpreted to find out what is the most prominent indicator of recurrence.*

MLP

The MLP classifier should again be the best performing one, due to its ability to model complex relationships between the features and, when trained properly, generalize very well. However, for the majority of the questions related to this dataset, the MLP would not be the best fit because it is **not interpretable** enough. *In contrast to RFs we cannot definitively say which features play the most crucial role when deciding whether the cancer is recurring or not, which is a big problem.*

Modeling

Hyperparameter tuning was performed identically as for the Purchase dataset models with the same parameter possibilities, tuning algorithm (RandomSearchCV) and tuning settings. Tuning was performed on the training dataset without PCA applied.

KNN

The model with the highest accuracy (**92.4%**) was the one where the number of neighbors was set to **3.**

RF

The RF model that achieved highest accuracy of **94.4%** is the one with **150** estimators.

MLP

The best performing MLP model achieved **94.4%** accuracy with the following hyperparameters: **'solver': 'adam', 'max_iter': 500, 'learning_rate_init': 0.001, 'hidden_layer_sizes': 200, 'alpha': 0.0001, 'activation': 'logistic'**

We notice that the best MLP model also has one layer, which is the same as for the purchase dataset. Here however, the number of nodes in the layer is smaller.

Model comparison

The three models were trained with following varying settings:

1. There were three *"Scalers"*: StandardScaler, RobustScaler and PCA transformation with **3** PCs.
2. Two evaluation methods: **holdout** and **cv**

Each models hyperparameters were set to those found by hyperparameter tuning performed apriori.

The evaluation metrics used are defined in the Introduction section.

The results of the model comparison are shown in Table 2 (Below). Best value of each metric per classifier is **bolded** and the best value across all classifiers in highlighted in green.

Algorithm	*Preprocessing*	*Validation*	*Accuracy*	*Precision*	*Recall*	*F1*	*Training Time*
MLP	None	holdout	0.953	0.949	0.956	0.952	0.624
MLP	None	cv	0.919	0.926	0.917	0.910	1.162
MLP	PCA(n_components=3)	holdout	0.942	0.938	0.942	0.940	0.287
MLP	PCA(n_components=3)	cv	0.933	0.926	0.912	0.921	1.640
MLP	RobustScaler()	holdout	**0.965**	**0.972**	**0.957**	0.963	0.470
MLP	RobustScaler()	cv	**0.965**	0.969	0.964	0.959	2.986
MLP	StandardScaler()	holdout	**0.965**	**0.972**	**0.957**	0.963	0.360
MLP	StandardScaler()	cv	0.961	0.967	0.955	**0.966**	3.431
RF	None	holdout	0.930	0.932	0.923	0.927	0.209
RF	None	cv	0.933	0.937	**0.941**	0.937	1.165
RF	PCA(n_components=3)	holdout	0.930	0.926	0.932	0.928	0.228
RF	PCA(n_components=3)	cv	0.926	0.930	0.922	0.914	1.352
RF	RobustScaler()	holdout	0.930	0.932	0.923	0.927	0.214
RF	RobustScaler()	cv	**0.944**	**0.945**	0.936	**0.944**	1.181
RF	StandardScaler()	holdout	0.919	0.922	0.909	0.914	0.209

RF	StandardScaler()	cv	0.940	0.936	0.929	0.937	1.228
KNN	None	holdout	0.942	0.942	0.938	0.939	0.003
KNN	None	cv	0.912	0.918	0.903	0.905	0.044
KNN	PCA(n_components=3)	holdout	0.942	0.942	0.938	0.939	0.004
KNN	PCA(n_components=3)	cv	0.909	0.913	0.901	0.902	0.105
KNN	RobustScaler()	holdout	**0.953**	**0.957**	**0.947**	**0.951**	0.006
KNN	RobustScaler()	cv	0.930	0.936	0.919	0.924	0.071
KNN	StandardScaler()	holdout	**0.953**	**0.957**	**0.947**	**0.951**	0.003
KNN	StandardScaler()	cv	0.937	0.941	0.928	0.932	0.060

TABLE 2: MODEL COMPARISON (BREAST CANCER)

Discussion

We observe that the MLP classifiers take the leading places in the table, not only for F1 statistic, but also for the remaining ones. It should be noted as well that the metric values are **all above 0.9** for all models, meaning the differences between their performance are relatively small.

It can be seen that the StandardScaler produces slightly better results than the RobustScaler, PCA and None scaler. This is of course expected for the distance based algorithms such as KNN and even MLP to a certain extent.

The cross validated MLP models seem to have the longest runtime which is around 3 seconds.

For this dataset, the Precision statistic plays an important role, because it defines to which extent did the model correctly classify recurrent cancer patterns.

Another interesting insight would be to look at feature importance extracted from the best performing RF model. Such information is contained in Figure 7 where we see that the variables *areaWorst, concavePointsMean* and *permeterWorst* play the most important role when deciding the outcome for the observation. This would allow analysts and medical professionals to further research into the topic and why are these factors so important.

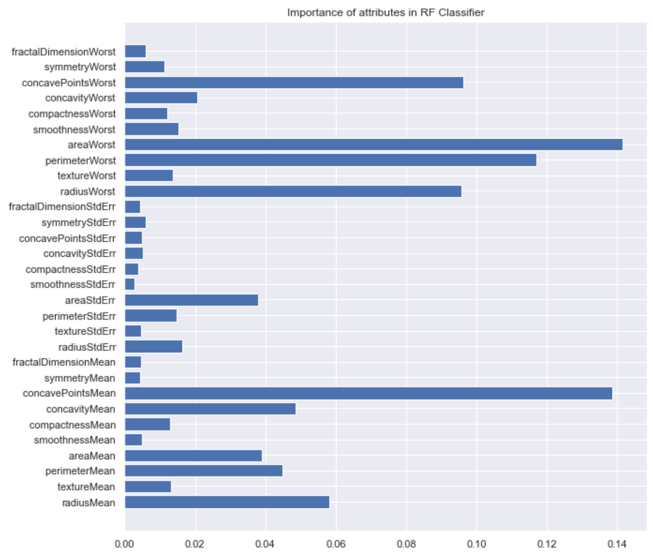

FIGURE 7: FEATURE IMPORTANCE (BREAST CANCER)

Taiwanese Bankruptcy

Exploration

The dataset represents various financial/bank data collected from the Taiwanese citizens. It contains 95 feature variables and 6819 observations and the goal is to predict whether the given citizen is bankrupt or not.

As one may assume, the bankruptcy is not such a common phenomenon, therefore, the target variable is heavily imbalanced, with only **220 (3.2%)** observations where the citizen is actually bankrupt.

Further exploration shows that there are *no missing values*.

Since there are too many features to plot, only the distribution of arbitrary four feature variables is presented in Figure 8. From the rest of the features and these four plots we see that their distributions are usually bell shaped, some are skewed, and some have very low variance. All of this should be taken into account when preprocessing the data. e.g., Log transformation for the skewed variables and normalization for all variables.

We further noticed that *Net income flag* feature has all equal values (i.e. variance=0). Since this feature does not carry any information content, it has been dropped.

After plotting the Correlation matrix (Figure 9), we see that there are multiple variable pairs that are moderately correlated. Therefore, PCA might be able to reduce the information redundancy contained in the correlated features.

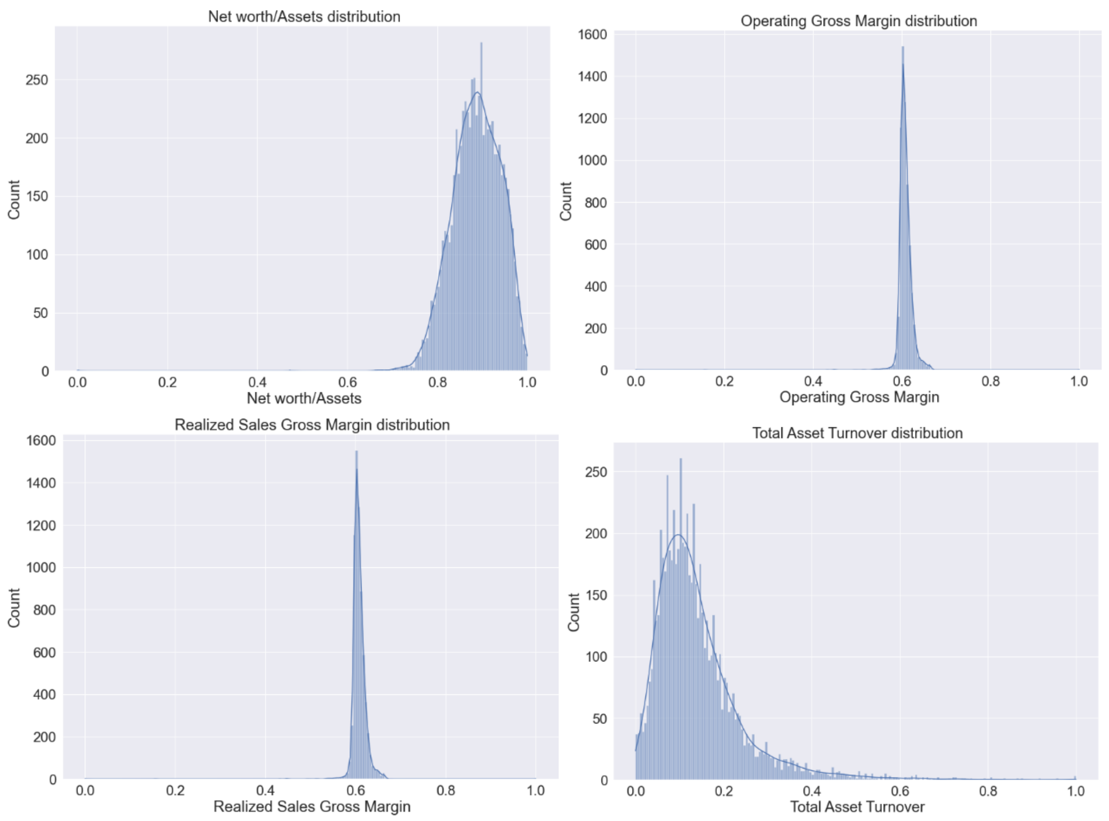

FIGURE 8: FEATURE DISTRIBUTIONS (TAIWANESE BANKRUPTCY)

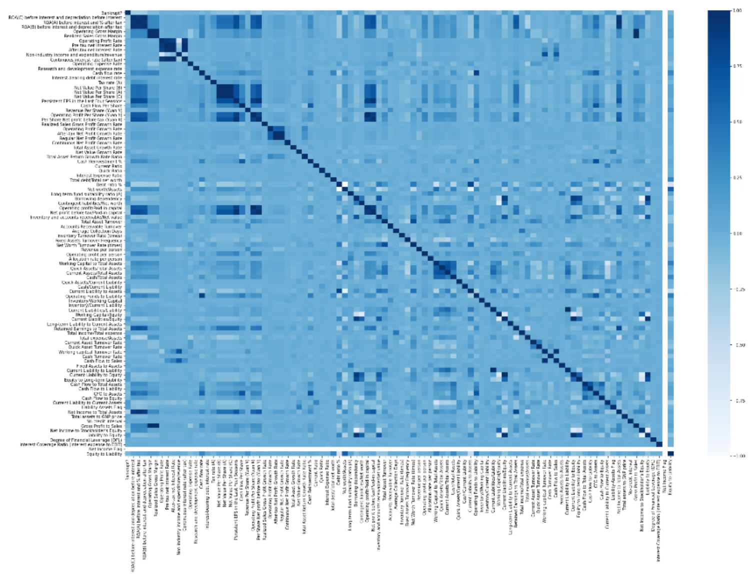

FIGURE 9 : CORRELATION MATRIX (TAIWANESE BANKRUPTCY)

Preprocessing

We firstly split the data into train and test sets using the 70-30 split. The stratified shuffle was used to ensure the equal distribution of the target in train and test datasets. This ensures the **equal representation** of both target classes, which is important because the dataset is heavily imbalanced. Another method of addressing the imbalance would be to either subsample or oversample the data. Both options would be viable since the number of observations is neither too small nor too big. This however was not performed in our experiments due to temporal constraints.

Performing the PCA with 10 components shows that it is a sensible tool to apply before the modeling because by selecting only the first **8 PCs**, **98%** of the train dataset variance is explained and the dimensionality is significantly reduced, compared to the initial 95 features. This could potentially allow the model to generalize better. The scree plot showing how much of the variance is explained by the first 10 PCs is shown in Figure 10.

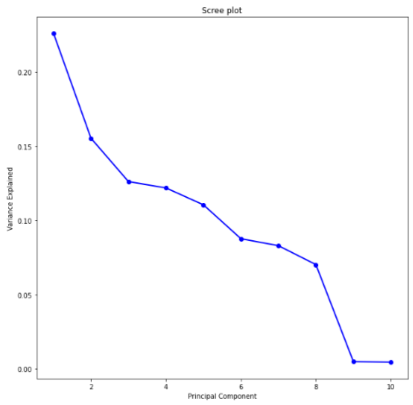

FIGURE 10: SCREE PLOT OF THE FIRST 10 PCS (TAIWANESE BANKRUPTCY)

Classifiers

KNN

The methodology of the KNN classifier can also be well applied on this dataset. However, due to its high dimensionality it might not be the most optimal setting to simply feed all variables into the model. A dimensionality reduction could address this problem.

RF

We find the RF model to be the most promising one for this dataset, because the financial data can very easily be split with thresholds. Furthermore, the RFs are robust to outliers and noise, which also might imply the better performance. Lastly, the RF models are relatively interpretable, which allows us to extract feature importance and gain further insights into the data.

MLP

The same argumentation about interpretability as for the Breast Cancer dataset can be offered here. Since the majority of analysts would ask the question of "*What are the most relevant factors (features) when predicting bankruptcy?*" This unfortunately could not be answered clearly enough based on the MLP model, even if it reaches an extremely high accuracy.

Modeling

The hyperparameter optimization for this classifier was performed with the RandomSearchCV algorithm and the identical parameters as for the previous two datasets.

KNN

The KNN models achieved highest accuracy (**96.7%**) with **13 neighbors.**

RF

The RF model with the highest accuracy (**97%**) had **100 estimators**.

MLP

The best performing MLP model achieved accuracy of **96.8%** with the following settings: **'solver': 'adam', 'max_iter': 500, 'learning_rate_init': 0.001, 'hidden_layer_sizes': (200, 100), 'alpha': 0.0002, 'activation': 'tanh'**

Model comparison

The best hyperparameters were chosen for the three models.

They were further trained with following distinct configurations:

1. Scalers: StandardScaler, RobustScaler, MinmaxScaler and PCA with 10 components
2. The two evaluation methods: holdout and cv

Metrics used are named in the Introduction and the results are presented in the Table 3. Best value of each metric per classifier is **bolded** and the best value across all classifiers in highlighted in green.

Algorithm	*Preprocessing*	*Validation*	*Accuracy*	*Precision*	*Recall*	*F1*	*Training Time*
MLP	None	holdout	0.968	0.484	0.500	0.492	3.459
MLP	None	cv	0.967	0.000	0.000	0.000	27.920
MLP	PCA(n_components=10)	holdout	0.965	0.484	0.499	0.491	2.066
MLP	PCA(n_components=10)	cv	0.966	0.221	0.000	0.000	17.999
MLP	RobustScaler()	holdout	0.962	0.642	0.570	0.592	7.787
MLP	RobustScaler()	cv	0.960	0.277	0.082	0.142	63.644
MLP	MinMaxScaler()	holdout	0.968	**0.748**	0.588	0.627	9.163
MLP	MinMaxScaler()	cv	0.956	0.430	0.182	0.228	69.093
MLP	StandardScaler()	holdout	0.957	0.653	**0.655**	**0.654**	18.264
MLP	StandardScaler()	cv	0.941	0.325	0.268	0.266	112.767
RF	None	holdout	**0.970**	**0.796**	**0.596**	**0.642**	1.910
RF	None	cv	0.966	0.527	0.150	0.258	11.450
RF	PCA(n_components=10)	holdout	0.966	0.484	0.499	0.491	1.036
RF	PCA(n_components=10)	cv	0.967	0.300	0.005	0.018	14.302
RF	RobustScaler()	holdout	0.969	0.761	0.581	0.620	2.253

RF	RobustScaler()	cv	0.965	0.560	0.177	0.230	24.602
RF	MinMaxScaler()	holdout	0.967	0.723	0.566	0.598	2.087
RF	MinMaxScaler()	cv	0.966	0.531	0.155	0.242	19.157
RF	StandardScaler()	holdout	**0.970**	0.787	0.589	0.632	1.682
RF	StandardScaler()	cv	0.966	0.575	0.173	0.229	21.266
KNN	None	holdout	**0.968**	0.484	0.500	0.492	0.003
KNN	None	cv	0.967	0.000	0.000	0.000	1.153
KNN	PCA(n_components=10)	holdout	0.968	0.484	0.500	0.492	0.029
KNN	PCA(n_components=10)	cv	0.968	0.000	0.000	0.000	1.318
KNN	RobustScaler()	holdout	0.968	**0.818**	0.515	0.521	0.031
KNN	RobustScaler()	cv	0.967	0.233	0.014	0.025	1.184
KNN	MinMaxScaler()	holdout	0.967	0.484	0.500	0.492	0.013
KNN	MinMaxScaler()	cv	0.967	0.200	0.009	0.017	1.152
KNN	StandardScaler()	holdout	0.967	0.712	**0.536**	**0.557**	0.013
KNN	StandardScaler()	cv	0.968	0.485	0.095	0.155	1.114

TABLE 3: MODEL COMPARISON (TAIWANESE BANKRUPTCY)

Discussion

From the results we observe that the three classifiers had very different performances along the tracked metrics. Firstly, we see that RF models achieve the highest values of accuracy. These were achieved with all three scalers and the holdout evaluation method. The accuracy scores of all models are greater than 0.95, so the differences between them are minor. The KNN model with robust scaler achieved highest Precision, while the MLP model with the standard scaler obtained highest values for the Recall and F1 metrics.

We notice that the longest training time was recorded with MLP models and CV evaluation method, which was expected.

For this dataset we also looked into the most important features identified by the random forest model. The five top ranked features (by feature importance) are:

1. **'Net Income to Stockholder's Equity',**
2. **'Borrowing dependency',**
3. **'Net Value Growth Rate',**
4. **'Persistent EPS in the Last Four Seasons',**
5. **'Net profit before tax/Paid-in capital'**

This insight could be further interpreted by the domain experts.

Eucalyptus

Exploration

Eucalyptus dataset contains various eucalyptus plant measurements, along with the data about where it is planted. The goal is to predict which seedslots are best for soil conservation in seasonally dry hill country, where there are 5 different quality categories (in the target variable).

The dataset has 20 feature variables and 736 observations. There are no duplicates present in the dataset, however there are **missing values**. The column with the most missing values is **Surv**, where there are **94 (12.7%)** values missing. Following are **Ins_res, Vig** and all 3 **_Fm** variables.

In the following the basic statistics of the dataset are provided

- number of classes: 5
- number of missing values: 448
- number of instances with missing values: 95
- number of numeric features: 14
- number of symbolic features: 6

From the correlation matrix of the features (shown in Figure 11) we see that the features are mostly uncorrelated, except for the three **_Fm** features.

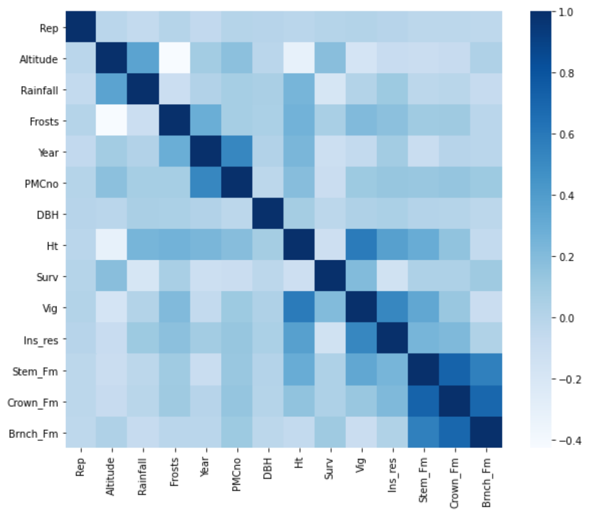

FIGURE 11: CORRELATION MATRIX (EUCALYPTUS)

We further research the target variable and plot its distribution. (Figure 12) The plot implies that the target is only slightly imbalanced. This imbalance should not pose an issue for the trained classifiers, but its effects will be tracked.

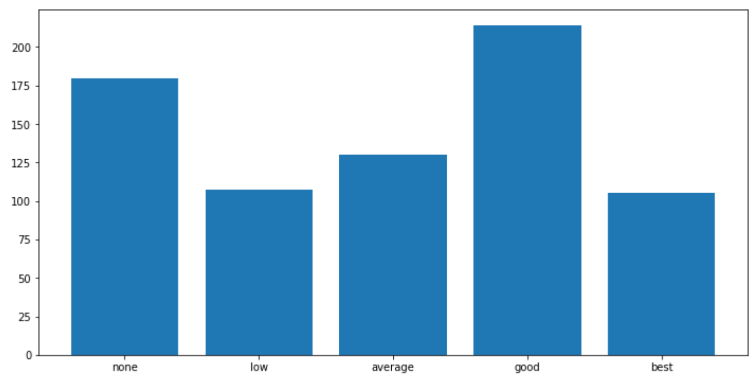

FIGURE 12: TARGET DISTRIBUTION (EUCALYPTUS)

Preprocessing

Since the dataset contains missing values, we considered measuring the performance by dropping instances with missing values and by imputing values.

Simple imputer from *sklearn* has been used to add missing values using the mean of the respective variable for imputation.

The dataset is then split into features and labels.

We further **encode the nominal features** by employing the **one-hot encoding**. After the one hot encoding we end up with 91 features (when imputing) and 78 features (when dropping missing values).

Since the target variable contains multiple classes that have certain ordering, the problem could be considered as ordinal classification. (Note: no distance between the classes is defined). Furthermore, we have a *None* class, which probably represents that the quality could not be determined. There are multiple ways of handling this:

1. Either remove the *None* class and proceed with four classes only. This would be a problem since the number of samples is very limited.
2. Perform an ordinal encoding of the target variable (None:= 0, low := 1, average := 2 ....)
3. Perform a one hot encoding of the target variable

Since this is a classification exercise, we proceeded with the latter option and applied one hot encoding of the target variable.

The dataset was then split into train and test sets using the 70-30 split

Next preprocessing step was scaling, where the numeric features were scaled using Standard-, MinMax- and Robust- Scaler.

In addition, PCA was also used to see if reducing the dimensionality after scaling could eliminate some features, but as seen in the Figure 13, this doesn't bring much value.

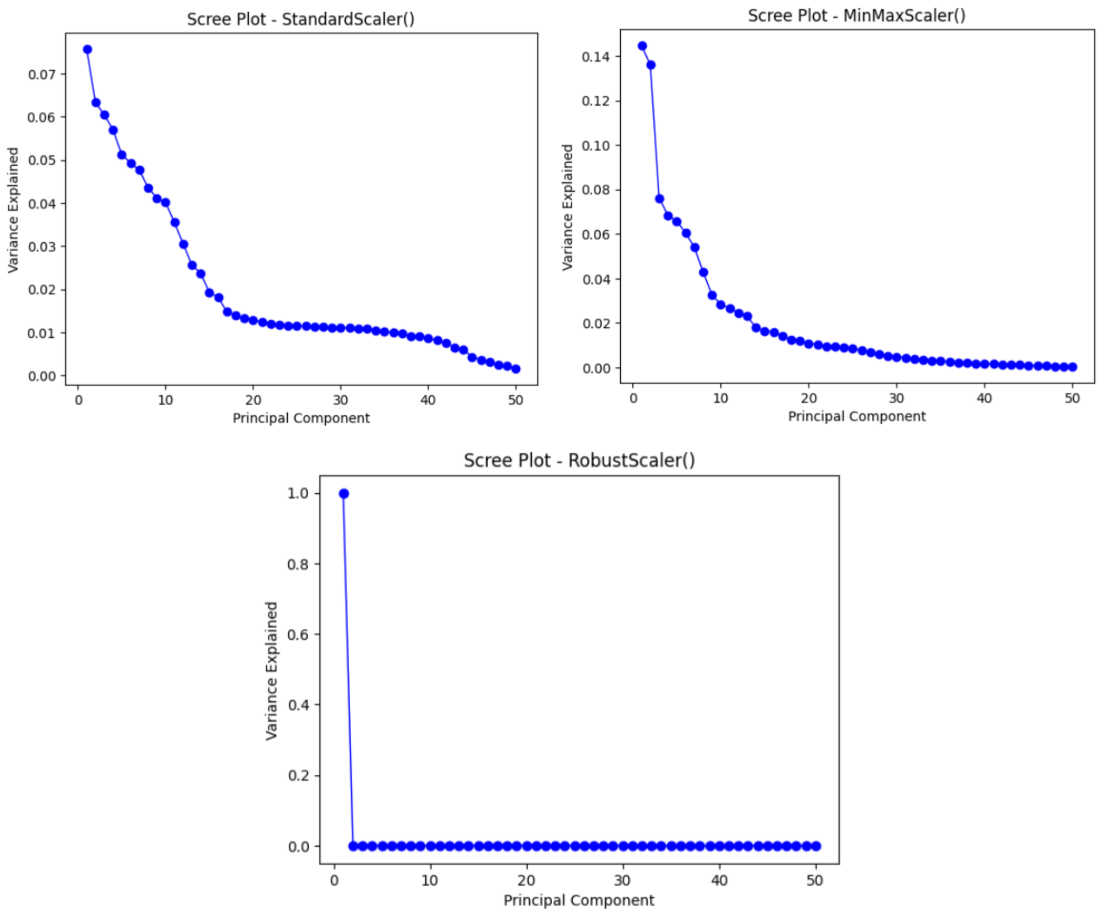

FIGURE 13: SCREE PLOT OF THE PCA USING DIFFERENT SCALERS (EUCALYPTUS)

Classifiers and modeling

KNN

The RandomSearchCV was used to identify what is the optimal number of neighbors for a KNN classifier. From the given k = [3, 5, 7, 9, 11, 13, 15], **k = 3** is the best value for the number of neighbors. It has been tested using imputed and dropped datasets, on different scaling methods, and k=3 was always giving the best results. The highest accuracy was **45%**.

RF

For the Random Forest model was used number of trees (**n_estimators**) as the main parameter for hyperparameter tuning process in RandomSearchCV. From the given estimator values [100, 125, 150, 175, 200, 250, 300, 350], the highest accuracy of **48%** was measured having **125** estimators.

MLP

RandomSearchCV was also employed for hyperparameter tuning. The search was performed along the following grid of parameters and values:

'hidden_layer_sizes': [(400, 200), (300, 150), (200, 100), (400), (300), (200), (100)]
'alpha': [0.0001, 0.0002, 0.0003],
'learning_rate_init': [0.001, 0.002, 0.003],
'max_iter': [1000],
'activation' : ['relu', 'tanh', 'identity', 'logistic'],
'solver': ['adam', 'lbfgs']

Best performing model was the one with the following values: **{'solver': 'lbfgs', 'max_iter': 1000, 'learning_rate_init': 0.003, 'hidden_layer_sizes': (200, 100), 'alpha': 0.0003, 'activation': 'logistic'}** and it achieved **59%** accuracy.

Results

All three classifiers have been trained with the best respective parameters found through hyperparameter optimization. The comparison is done two times, once for imputed dataset and once for dataset with dropped missing values. As mentioned before, the PCA was not used in the comparison process. Used metrics and validation methods are presented in tables Table 4 and Table 5. Best value of each metric is bolded highlighted in green.

Algorithm	*Preprocessing*	*Validation*	*Accuracy*	*Precision*	*Recall*	*F1*	*Training Time*
MLP	MinMaxScaler()	holdout	**0.554404**	0.579952	**0.546174**	0.557415	3.443190
MLP	StandardScaler()	holdout	0.554404	0.584727	0.545899	**0.562271**	1.949387
MLP	RobustScaler()	holdout	0.497409	0.559334	0.528332	0.541053	16.960771
MLP	RobustScaler()	cv	0.489850	0.516199	0.525264	0.477928	27.814747
KNN	RobustScaler()	holdout	0.487047	0.539653	0.460388	0.493109	0.046977
MLP	StandardScaler()	cv	0.461761	0.472057	0.475244	0.446948	9.938876
MLP	MinMaxScaler()	cv	0.461749	0.523533	0.502935	0.478216	20.295217
KNN	MinMaxScaler()	holdout	0.419689	0.480593	0.376642	0.414410	0.032000
KNN	StandardScaler()	holdout	0.393782	0.473327	0.351508	0.397971	0.035087
KNN	RobustScaler()	cv	0.358818	0.407599	0.330885	0.344816	0.336643
KNN	MinMaxScaler()	cv	0.332364	0.332593	0.296887	0.291540	0.283984
KNN	None	holdout	0.321244	0.404075	0.288527	0.335010	0.033151
MLP	None	holdout	0.305699	0.383779	0.337262	0.356292	14.937997
KNN	StandardScaler()	cv	0.282425	0.394892	0.268547	0.283837	0.279652
RF	StandardScaler()	holdout	0.279793	0.625909	0.231981	0.318896	0.345661
RF	MinMaxScaler()	holdout	0.279793	**0.761463**	0.249076	0.333797	0.320292
RF	RobustScaler()	holdout	0.269430	0.700000	0.228980	0.315607	0.334167
KNN	None	cv	0.268532	0.354845	0.236644	0.260800	0.273537
MLP	None	cv	0.248062	0.401801	0.305558	0.317492	**77.372286**
RF	None	holdout	0.222798	0.568120	0.193029	0.276788	0.396946
RF	StandardScaler()	cv	0.101357	0.425206	0.114018	0.165879	2.096056

RF	MinMaxScaler()	cv	0.096827	0.387584	0.093924	0.144640	1.969922
RF	RobustScaler()	cv	0.065540	0.368889	0.076214	0.122291	1.909135
RF	None	cv	0.059363	0.422879	0.047763	0.084668	2.059346

TABLE 4: MODEL COMPARISON WITHOUT IMPUTING (EUCALYPTUS)

Algorithm	*Preprocessing*	*Validation*	*Accuracy*	*Precision*	*Recall*	*F1*	*Training Time*
MLP	StandardScaler()	holdout	0.597285	0.604818	0.576986	0.590181	2.162776
MLP	RobustScaler()	holdout	0.561086	0.568323	0.558782	0.562833	3.980093
MLP	MinMaxScaler()	holdout	0.556561	0.582382	0.572911	0.577019	2.879056
MLP	MinMaxScaler()	cv	0.517793	0.543419	0.525082	0.513241	21.625387
MLP	RobustScaler()	cv	0.499176	0.530507	0.489413	0.503367	39.266990
MLP	StandardScaler()	cv	0.488239	0.521279	0.482412	0.473255	10.356182
KNN	RobustScaler()	holdout	0.457014	0.485175	0.393294	0.426661	0.053975
KNN	StandardScaler()	holdout	0.420814	0.482975	0.390741	0.428344	0.045994
KNN	MinMaxScaler()	holdout	0.420814	0.490351	0.381717	0.426868	0.043975
KNN	None	holdout	0.416290	0.487959	0.376376	0.422901	0.041970
MLP	None	holdout	0.402715	0.416431	0.399418	0.405505	16.593996
KNN	RobustScaler()	cv	0.358818	0.407599	0.330885	0.344816	0.327894
KNN	MinMaxScaler()	cv	0.332364	0.332593	0.296887	0.291540	0.280001
KNN	StandardScaler()	cv	0.282425	0.394892	0.268547	0.283837	0.277564
RF	None	holdout	0.276018	0.607927	0.228330	0.307584	0.383596
KNN	None	cv	0.274637	0.405656	0.288172	0.296456	76.251797
KNN	None	cv	0.268532	0.354845	0.236644	0.260800	0.262503
RF	StandardScaler()	holdout	0.266968	0.545744	0.226726	0.309185	0.466242
RF	MinMaxScaler()	holdout	0.253394	0.678305	0.214355	0.301407	0.461051
RF	RobustScaler()	holdout	0.244344	0.698659	0.193359	0.265562	0.472104
RF	StandardScaler()	cv	0.116982	0.425857	0.116522	0.138005	2.069188
RF	MinMaxScaler()	cv	0.103077	0.467667	0.097760	0.165352	2.016233
RF	RobustScaler()	cv	0.073389	0.396571	0.069962	0.103817	2.101173
RF	None	cv	0.045300	0.381429	0.059892	0.087567	2.037759

TABLE 5: MODEL COMPARISON WITH IMPUTING (EUCALYPTUS)

Discussion

From the results we observe that there is a slight difference in accuracy when evaluating datasets with and without imputing. The dataset with generated missing values has had better accuracy of 59% than dataset with dropped instances of 55% accuracy. For both datasets, the MLP gives the best, KNN middle and RF worst performance.

Supplementary Material

Dataset sources

- Purchase (https://www.kaggle.com/competitions/184702-tu-ml-ws-22-purchase)
- Breast Cancer (https://www.kaggle.com/competitions/184702-tu-ml-ws-22-breast-cancer)
- Taiwanese Bankruptcy (https://archive.ics.uci.edu/ml/datasets/Taiwanese+Bankruptcy+Prediction)
- Eucalyptus (https://www.openml.org/search?type=data&status=active&id=188&sort=runs)

Source Files

- mani-taiwanese-bankruptcy.ipynb
- Eucalyptus.ipynb
- Kaggle_Purchase.ipynb
- main-breast-cancer.ipynb