Machine Learning (2023W)

Assignment 1: Classification

Ivan Babiy (12142160) , Yat Hin Chan (12331419), Ahmed Šabanović (12330648)

19 November 2023

Contents

1 Choice of data sets

1.1 Nursery

The nursery data set was derived from a hierarchical decision model that ranks nursery-school applications for nursery schools in terms of their acceptance/rejection criteria. It has 12960 rows, 8 features and 1 target class. All features of ordinal type.

Nursery has a tall shape (12960x8) and categorical features, which contrasts with *Spambase*.

The categorical nature of the features suggests that the dataset may require preprocessing, such as one-hot encoding or label encoding, before being used in machine learning algorithms.

The target variable is notably unbalanced. Classes, like `recommend` and `very_recom`, are significantly underrepresented compared to others, such as `not_recom`, `priority`, and `spec-prior`.

This imbalance could result in sparse data, and potentially impact the performance of the machine learning models that are trained on this data set. One possible mitigation for this is to merge the class with few samples into a similar class (e.g., merging `recommend` and `very_recom`).

1.2 Spambase

This data set includes 4601 emails, each with 57 ratio features and 1 target variable. 54 of the features contain the relative frequencies of different words and characters, and 3 contain summary statistics of the length of sentences in all caps. The target attribute, 'Class', denotes whether an email is a spam.

The Spambase data set is an imbalanced data set due to the target variable having a disproportionate representation of non-spam emails (labeled as '0') compared to spam emails (labeled as '1'). This imbalance within the target variable could cause the model to become biased toward the majority class during training.

2 Pre-processing

2.1 Nursery

One-hot encoding

Since the attributes in the nursery data are categorical, we chose to apply one-hot encoding. This step allows using models that require attributes to be of numerical data type.

However, this method may lead to loss of information, since the attributes in nursery are not just of nominal type. When applying one-hot encoding on ordinal attributes, the ordinal information is lost.

Label encoding

Therefore, we also experimented label encoding. This method preserves the order of classes, as opposed to one-hot encoding.

It is important to note that this method makes the assumption that the 'degree' of each value differ linearly from each other. For instance, for the attribute `housing`, `convenient` > `less_conv` by 1, and `less_conv` > `critical` by 1.

Thus, label encoding was applied based on metadata and assuming 1 as the distance between each class.

2.2 Spambase

Log transformation

When examining the distribution of the input variables, we discovered that all input variables are heavily skewed, spanning a few orders of magnitude, as shown in figure 1. We therefore applied log transformation to see how it affects classification performance.

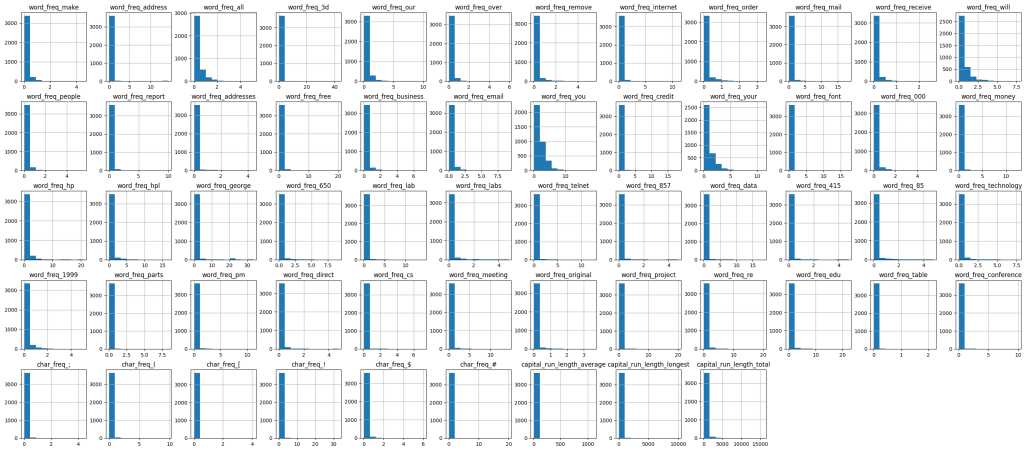


Figure 1: Histograms of input variables in Spambase

Min-max scaling

Since the attributes in Spambase are relative frequencies within each document, i.e. each row, applying min-max scaling along the column may destroy the relationships between frequency values within each row. However, to explore the effects the scaling on different data sets and classifiers, we applied min-max scaling on Spambase as well.

Z-score standardisation

Similarly to min-max scaling, we expected z-score standardisation to alter the relations between values in a row, and therefore would yield worse performance. We included it for comparison with other data sets and across classifiers.

2.3 Loan

Loan is a multi-class classification data set containing mostly numerical attributes. The rest are categorical attributes - some ordinal, some nominal. The target class `grade` has 7 possible values.

One-hot encoding

There are numerous nominal attributes in Loan which requires one-hot encoding to ensure compatibility with certain classifiers, like kNN and regression classifiers.

Besides nominal attributes, we also applied this encoding to ordinal attributes to compare its effect with label encoding.

Label encoding

Some ordinal features in Loan can be directly replaced with numerical values. For example, `term` has the values `36 months` and `60 months`. We therefore converted the values to `36` and `60`; There are also columns indicating `Y`/`N` values, which we encoded to `0` and `1`; For other ordinal columns, we again assumed a linear relationship between values and encode them as `[0, 1, 2, 3, ...]`.

Log transformation

While exploring the data, we discovered features with heavy skews. We applied log transformation to compare performance with using them as is.

Min-max scaling

Unlike in Spambase, the numerical columns in Loan has no relationship with other values on the same row. They also have their own unit and value ranges . Min-max scaling can prevent any attributes from dominating the learning. We therefore experimented whether it can improve prediction performance.

Z-score standardisation

Based on similar reasoning with min-max scaling, we apply z-score standardisation to Loan's numerical columns.

2.4 Breast cancer

The breast cancer data set contains measurements from breast tumor biopsies, including features like size, texture, and shape. The data set is useful for building models that predict if a tumor exists or not. The target variable has two binary values, specifically "true" and "false."

Log transformation

Due to the data set being quite left skewed and the column `areaWorst`, having a particular bad skew, we opted for preprocessing. We applied a logarithmic transformation to this data set to see if it improves performance and alleviates skewness. As we can see the from figures 2 and 3, the data noticeably shifted toward a more centralized distribution.

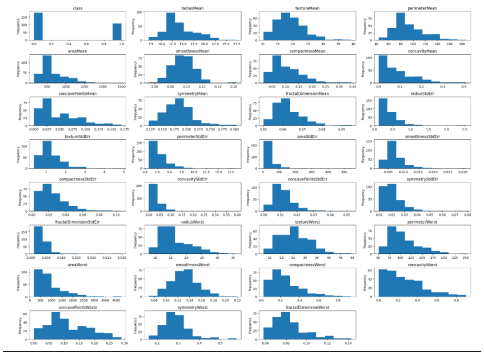


Figure 2: Histograms of the raw breast cancer data

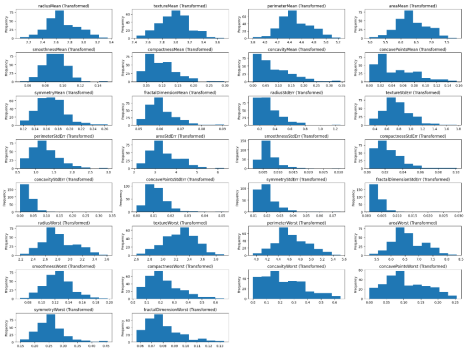


Figure 3: Histograms of the log-transformed breast cancer data

Min-max scaling

There are 3 types of statistic of different measurements in this data set, worst, mean, and standard error. While it makes sense to scale the worst and mean attributes to the same value ranges or scale, it does not for standard errors since these attributes are already comparable.

Z-score standardisation

We applied z-score standardisation as an alternative scaling/standardisation method for comparison purposes. Its primary aim was to contrast the results against those obtained through min-max standardization. Ultimately, the results between these two preprocessing techniques were quite similar, as we can see from figures 4 and 5.

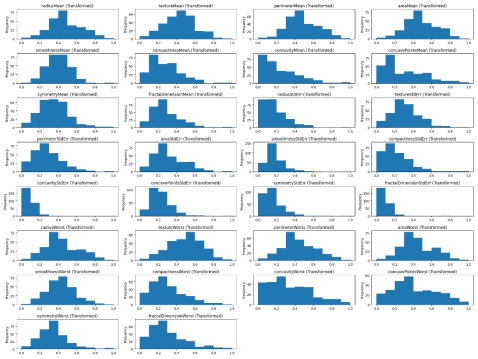


Figure 4: Histograms of the minmax transformed data

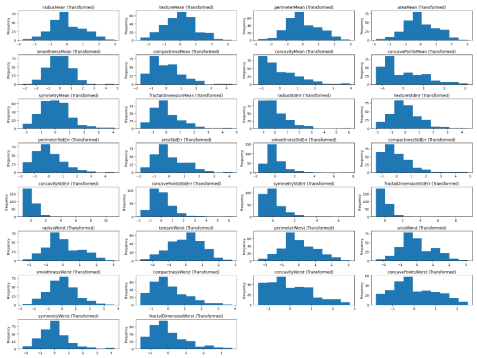


Figure 5: Histograms of the z-score transformed breast cancer data

3 Classifiers

3.1 Decision tree

Decision trees are rule-based classifiers that build a tree-like structure to make decisions. Decision trees are versatile and efficient classifiers suitable for various data sets. Due to their interpretability, decision trees provide a clear and intuitive representation of decision-making, which was quite useful. The `DecisionTreeClassifier` in scikit-learn allows for fine-tuning via various parameters such as:

1. Max depth - limits the maximum depth of the decision tree.
2. Min Samples Split - specifies the minimum number of samples required to split an internal node.
3. Min Samples Leaf - sets the minimum number of samples needed to form a leaf node

The table below shows how some Although decision tree theoretically supports categorical data, scikit-learn's implementation does not. Therefore, for data sets with categorical attributes, we could only apply decision tree after encoding.

3.2 k-nearest neighbors

This is a distance-based classifier. We tuned

1. Number of neighbours `n_neighbors` - 1 to 15.
2. Distance function `p` - 1 and 2. I.e. Euclidean and Manhattan distances

and compared the performances.

Since this is distance-based, categorical values in the data sets need to be encoded into numerical values. As mentioned in section 2, we used one-hot encoding and label encoding, and compared the performance.

3.3 Logistic regression

3.3.1 Binary classification

Logistic Regression is a statistical method used for binary classification, which means it is used to predict the probability that an instance belongs to one of two classes. Thus, it is particularly useful when the target variable is categorical and has two possible outcomes (e.g., 0 or 1, TRUE or FALSE).

It uses the logistic function (i.e., sigmoid function) to model the probability that a given input belongs to the positive class. The logistic function is an S-shaped curve that maps any real-valued number to a value between 0 and 1.

Using *scikit-learn*'s LogisticRegression(), we tuned:

1. C (Inverse of Regularization Strength), it controls the regularization strength.
2. Penalty (e.g., L1, L2), it is the regularization type and can prevent overfitting.
3. Solver (e.g., Liblinear (default), Limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS), Newton-CG, Stochastic Average Gradient (Sag), Saga), it is an optimization algorithm used to find the coefficients that minimize the cost function.
4. Number of iterations (e.g., 100; 1,000; 10,000), it represents the maximum number of iterations taken for the solver to converge (i.e., reach a solution).

3.3.2 Multi-class classification

The Multimodal Logistic Regression (i.e., Softmax Regression) is an extension that allows the logistic regression to handle more than two classes. Instead of having a single logistic function, it uses multiple logistic functions, each associated with a particular class. The softmax function is then applied to convert the raw output scores into probabilities, ensuring that the sum of probabilities across all classes equals 1. The model predicts the class with the highest probability.

Instead of modifying the logistic regression algorithm itself, the One-vs-All (i.e., One-vs-Rest) is a strategy where separate binary logistic regression models are trained for each class (e.g., in a problem with `k` classes, `k` binary classifiers are trained).

Therefore, in the cases of nursery and loan, the multimodal logistic regression was implemented instead, by setting the *multi class* argument in the LogisticRegression() function to `multinomial` or `ovr`.

4 Experiment design

4.1 Independent variables

We change below variables as part of our experiment:

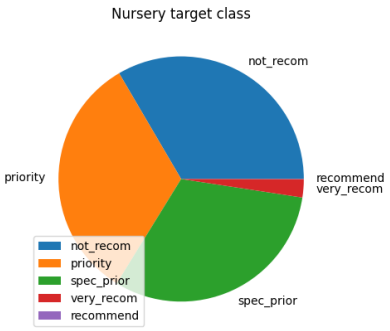
1. Classifier - Decision tree, k-nearest neighbours, logistic regression
2. Classifier parameters - as detailed above
3. Combination of pre-processing techniques - One-hot encoding, label encoding, log transformation, min-max scaling, z-score standardisation

The effect of each variable will be examined by changing each one while holding the others constant.

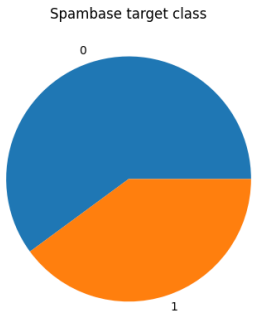
4.2 Performance metrics and evaluation

In both binary classification data sets, Spambase and Breast Cancer, the target classes are imbalanced (figure 6), with one class being more important than the other. There are fewer emails classified as spams in Spambase, and fewer samples classified as positive diagnoses in Breast Cancer. However, both of these positive classes are more important than the negative class.

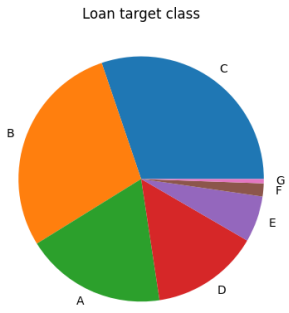
Therefore, in addition to accuracy, we also measure performance with F1 score for these data sets.



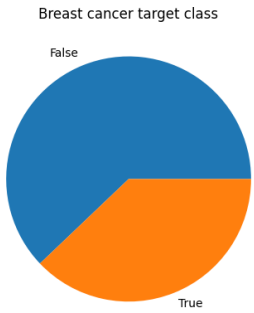
(a) Distribution of Nursery's target class



(b) Distribution of Spambase's target classs



(c) Distribution of Loan's target class



(d) Distribution of Breast Cancer's target class

Figure 6: Target class distributions

However, both accuracy and F1 only tell us the whether the predictions are correct, but not how wrong the predictions are. This is relevant to both of our multi-class classification data sets, where the target class is ordinal. In Nursery, it would be slightly wrong to classify `very_recom` as `recommend`, but very wrong to classify `very_recom` as `not_recom`; similarly in Loan, it would be slightly wrong to classify `A` as `B`, but very wrong to classify `A` as `G`. A better way to evaluate the prediction performance for these two data sets might be to first encode the class as numeric values, then calculate the RMSE, etc.

4.3 Ensuring comparability between classifiers

Theoretically, our classifiers require different input types - kNN and logistic regression require numeric inputs and decision tree supports numeric and categorical ones. However, since sklearn's implementation of decision tree does not support categorical input, we transformed all attirubutes in all data sets into numerical values. As a result, the three classifiers run on the same exact data sets.

5 Results analysis

Having implemented the 3 classifiers across 4 different datasets, each with different combinations of data pre-processing methods, target type (e.g., binary, multi class), and having experimented and fine-tuned different hyperparameters, the following insights were extracted.

5.1 Effect of pre-processing

Using Loan, where the most pre-processing techniques were applied, we can compare the effect of each technique. Table 1 shows the accuracies of each classifier under different combinations of techniques.

The accuracies are obtained first by using grid search to select the best model, then calculating predictions on an unseen test set.

Table 1 shows us each classifier is differently affected by different pre-processing techniques. kNN and logistic regression both perform best with z-score standardisation applied to the numerical attributes. Both also benefit from scaling in general, as the scores under min-max scaling and z-score standardisation are higher than no scaling or standardisation applied. In contrast, decision tree performed best without scaling or standardisation.

A possible explanation is that while kNN uses distances and logistic regression uses weights, decision tree just learns boundaries. In kNN and logistic regression, scaling and standardisation help prevent attributes with large values from dominating

onehot	label	log	minmax	z	kNN accuracy	DT accuracy	LogReg accuracy
True	False	False	False	False	0.324500	0.987000	0.535000
True	True	False	False	False	0.324500	0.986000	0.530500
True	False	True	False	False	0.308500	0.987000	0.444500
True	True	True	False	False	0.308500	0.986000	0.449000
True	False	False	True	False	0.440500	0.983000	0.770500
True	True	False	True	False	0.462000	0.969500	0.774000
True	False	True	True	False	0.446000	0.983500	0.766000
True	True	True	True	False	0.475000	0.967000	0.771000
True	False	False	False	True	0.507000	0.919000	0.857500
True	True	False	False	True	0.518500	0.917000	0.852500
True	False	True	False	True	0.489000	0.920000	0.858500
True	True	True	False	True	0.494500	0.916500	0.853000

Table 1: Summary of classification accuracies under different pre-processing techniques on Loan

the distances or weights; But with decision trees, retaining the original distributions and ranges of each attribute may mean that the boundaries would be more accurate.

Another example where scaling and standardisation negatively impact performance is seen in Spambase when using kNN. As mentioned in section 2.2, these techniques destroy the relativity within rows. This could be why they perform worse than just applying log transformation, as shown in figure 7.

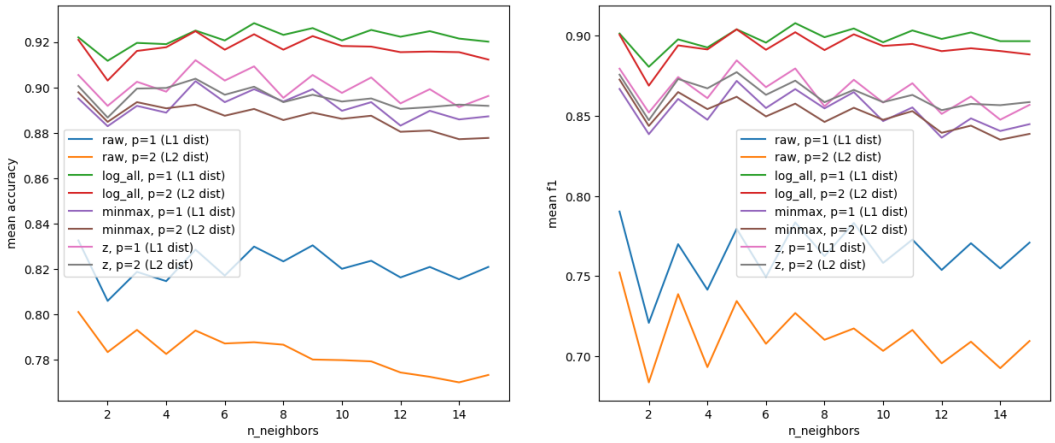


Figure 7: Grid search results of kNN on Spambase

Another type of scaling, log transformation, yielded mixed results. As shown in table 1, on Loan, log often slightly worsens the prediction performance.

However, on Spambase, log transformation significantly improved prediction results (figure 7). In fact, it even trained the best performing model. On the other hand, utilizing the raw data didn't seem to negatively affect the linear regression model, even though it did affect kNN. Applying the minmax transformation to linear regression yielded worse results than raw, for smaller values of C.

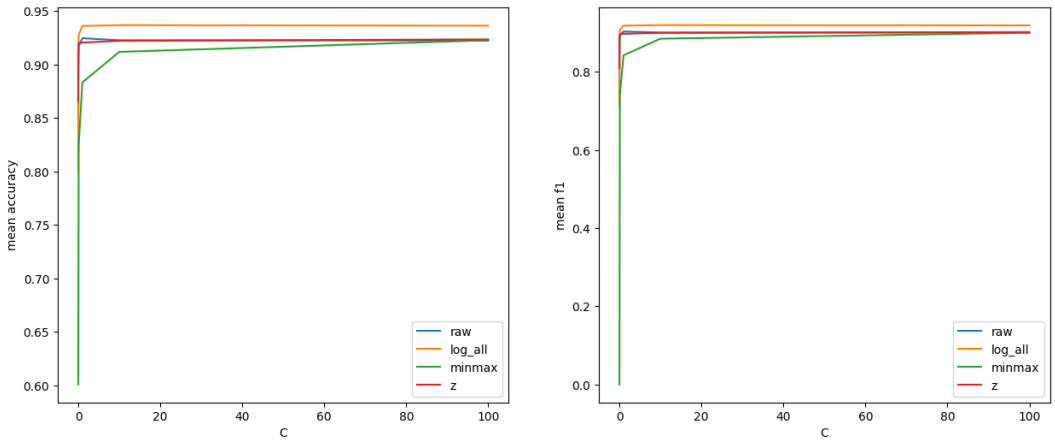


Figure 8: Grid search results of logistic regression on Spambase

Surprisingly, one-hot encoding outperformed label encoding in several cases. This is seen, for example, in the nursery dataset. This may be due to wrongly assuming that classes are equally distanced from each other (as explained previously), which could be introducing error or noise to the data.

Test accuracy	One hot encoding	Label encoding
0.931713	True	False
0.898920	True	True

Table 2: Classification accuracies using one-hot vs. label encoding, on Nursery

The decision tree classifier also had mixed results. Table 3 shows the effects of pre-processing on decision trees. In general, techniques like MinMax scaling and Z-score normalization were not as beneficial. Raw data outperformed these techniques in all cases. Meanwhile, log transformation seems to have a lesser impact on model performance.

Based on these results, the choice of pre-processing technique can significantly influence the decision tree's ability to learn and generalize from the data.

Overall, pre-processing affects differently each dataset, and for each dataset models perform differently with the same pre-processing methods applied.

Dataset	Accuracy (Holdout)	Mean CV Accuracy	Mean CV F1	Grid Search Score
Original	0.931	0.911	0.886	0.895
Log All	0.926	0.910	0.887	0.896
MinMax	0.798	0.910	0.887	0.896
Z	0.848	0.909	0.887	0.895

Table 3: Decision Tree Performance with Various Preprocessing

5.2 Effect of classifier parameters

5.2.1 Decision tree

This is how the decision tree performed for the spambase data set.

1. Max Depth Parameter: Higher accuracy and F1 scores were achieved with deeper trees which captured complex patterns. However, on the 'minmax' and 'z' sets, where accuracy dropped significantly, the tree might have become overly complex, causing overfitting.

2. Grid Search for Max Depth: The best 'max-depth' found was 10 across all datasets.

3. Original Holdout Training/Test Split: Consistently high accuracy and F1 scores on most datasets, except for 'minmax' and 'z' sets, indicating potential overfitting due to higher complexity.

4. Cross-Validation: Cross-validation accuracies and F1 scores were generally high and consistent.

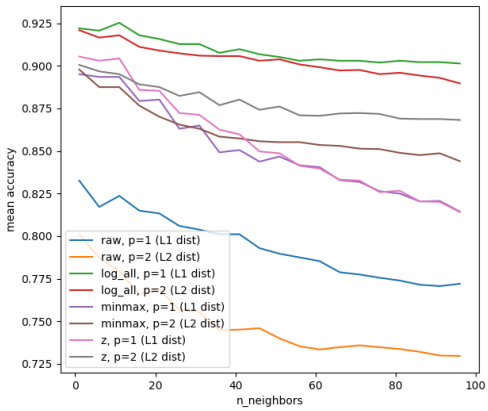
5.Overall: The decision tree classifier performed well on most datasets but showed signs of overfitting on datasets where the tree's depth might have become too deep ('minmax' and 'z' sets).

5.2.2 k-nearest neighbors

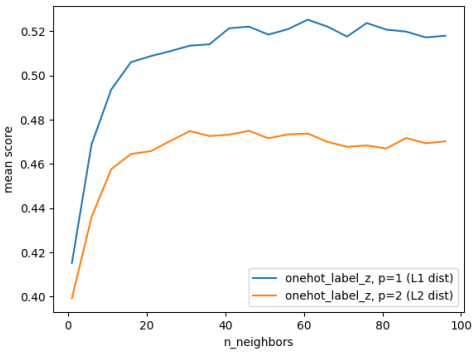
Of the parameters we tuned, the most important parameter affecting the performance is the distance function. L1 (Manhattan) distance performs better than L2 (Euclidean) distance. As shown in figure 9a, L1 distance performs consistently better no matter which pre-processing techniques are applied.

However, we discovered no obvious general trends for how the number of neighbours affect performance, leading us to believe that it is mostly dependent on the data set. From figure 9, we could guess that the optimal number of neighbours is around 20-30% the number of features.

The sensitivity of performance to change in number of neighbours also varies between data sets. However, from 9, we can see that performance curves have higher



(a) kNN grid search results on Spambase



(b) kNN grid search results on Loan

Figure 9: Grid search results of kNN on Spambase and Loan

slopes at smaller number of neighbours, meaning performance may be generally more sensitive to change in number of neighbours when the number is low.

5.2.3 Logistic regression

Fine-tuning parameters for the logistic regression was an exhaustive task, several insights were observed:

- The choice of the regularization parameter C seems to heavily depend on the dataset. For higher values of C, regardless of the combination of pre-preprocessing methods and parameters, the model performance usually converges. Thus, we suspect that larger values of C tend to increase the risk of overfit.

- Regardless of using L1 or L2, the penalty parameter seems to generally avoid overfit. However, L1 does work significantly well on datasets with a lot of features (e.g., spambase, loan), for the narrower datasets (e.g., nursery, breast cancer) there seems to be no significant difference between L1, L2, and not using a regularization type at all.

- As mentioned before, there are at least 5 different possible options for the solver parameter. Generally, the default in Python's sckit-learn (i.e., Liblinear) works fine in most scenarios. Nevertheless, the Newton-CG outperformed the default solver in the nursery dataset. As for the rest of the solvers, no significant differences were observed, with some performing worse.

- The higher the number of iterations, the better the result usually was. Even though computation times increased significantly for values above 1000 (specially

for larger datasets), smaller values weren't usually sufficient for the solver to converge to the best solution (constant warnings notified us of this fact).

6 Conclusion

Based on our experiments, we conclude that among our classifiers, none of them always perform better than the others. The 'best' algorithm always depends on the nature of the data set, its features and target. Pre-processing is essential in increasing the effectiveness of the algorithms. Correctly applying pre-processing techniques techniques requires good understanding of both the data and the techniques themselves.

Furthermore, different algorithms are not affected by pre-processing techniques in the same way. Experimentation in different combinations of processing techniques, algorithms, and their parameters is thus needed to maximise prediction performance.

Through this exercise, we experienced how this experimentation can get complicated as more variables and parameters are taken . This showed us that good coding and engineering practices are vital for the success of data science experiments. It ensures the same logic is always applied, the parameters being tuned are well understood and defined, and the programme is idempotent, etc.