Machine Learning

Exercise 1 - Group 18

Clara Pichler[1]
Hannah Knapp[2]
Sibel Toprakkiran[3]

This report presents our work for Assignment 1 of the lecture Machine Learning at TU Vienna. After providing a brief overview of the datasets used and the preprocessing steps applied, we explain the workflow of applying three classifiers to these datasets, along with the results obtained. These steps are documented in several Jupyter Notebooks included in the submission, using Python 3.12.5.

1 Data Description

In this section we will give a short overview of the different data sets we used for this task. It will give information on the structure of the data and what our target values for each data set will be.

1.1 Gym Members Exercise Tracking

The first data set can be found on Kaggle and gives us a detailed overview of gym members' exercise routines, physical attributes, and fitness metrics. We chose this data set because we think it could be an interesting task to classify the type of training a gym member does.

The data set has a total of 973 entries and 15 attributes and includes information on the gym member's physical form like age, gender, height and weight, but also about their exercise frequency (in a week) or the water intake as well as the type of workout they perform. From the histograms below (see Figure 1) we can observe the distribution of all attributes that are ratio data (or numerical). The only exceptions are the features `Gender`, `Workout_Type` and `Workout_Frequency`, which the latter can be categorized as interval data. `Gender` with the unique values male and female and `Workout Type` are nominal data, as one cannot order these attributes. Our goal is to classify the data into the workout types, which are `Yoga`, `HIIT` (High Intensity Interval Training), `Cardio` and `Strength`. The number of members are almost evenly divided between the workout types, with the most members choosing strength training (258 members) and the least member choosing `HIIT` training (221 members).

A very important note to add is that this data set is generated to reflect realistic exercise tracking scenarios with diverse participants. It is not recommended to use it for research purposes, however, for this project it will be enough to get a general understanding of the topic.

[1] 11917694 - e11917694@student.tuwien.ac.at
[2] 11901857 - e11901857@student.tuwien.ac.at
[3] 09426341 - e9426341@student.tuwien.ac.at

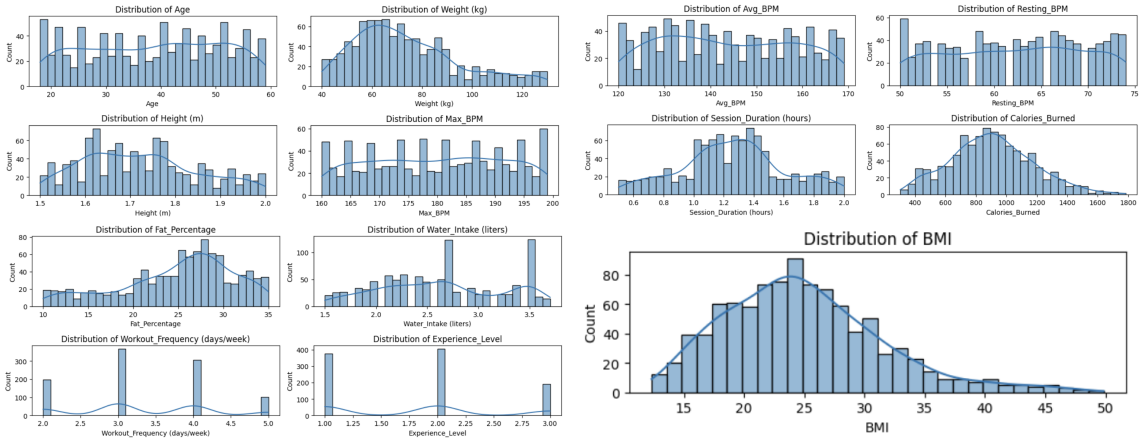

Figure 1: Histogram of values

1.2 Abalones

The second data set can be found on the website of UCI and gives information about different charateristics of abalones. We selected the Abalone data set because marine biology is an interesting field for all of us and it is a quite good data set for such a classification task. The goal is to predict the age of an abalone using the physical measurements which are the other features. In general, the age is determined by cutting the shell through the cone, staining after that, and counting the number of rings present through the microscope.

The data contains 4177 samples (individual abalones) and nine attributes including their height, weight, sex and of course their rings. For the target attribute `Rings` we would choose in a pre processing step to change it from integer to bins of age (ordinal) so we can classify the age of the abalones. `young` (1-5 rings), `adult` (6-13 rings) and `senior` (13+ rings). The distribution of the `Rings` attribute shows a concentration at lower values, indicating the dataset has many younger abalones and only a few older ones. The most abalones have between 8 to 11 rings. It is important to understand the scale to identify trends or outliers. Also different machine learning algorithms might perform better according to the distribution of the target variable. In Figure (2) we can see the distribution of the target and the feature attributes.

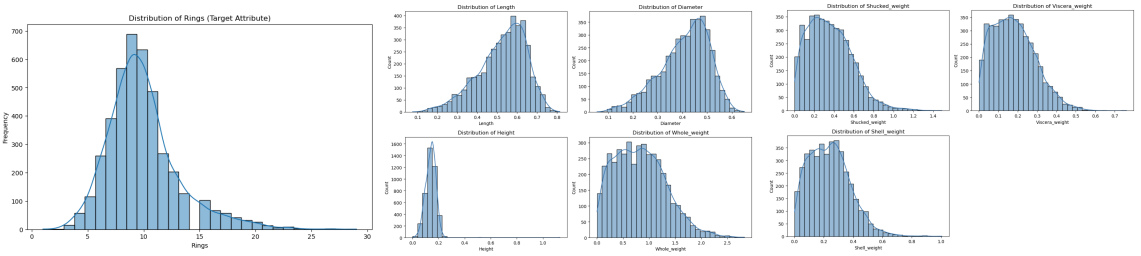

Figure 2: Distribution of target and feature attributes for abalone dataset

1.3 Amazon Reviews

Two of the data sets were part of a competition on Kaggle. This data set contains different amazon reviews by customers. The goal is to classify which customer gave the review.

The data set has 750 entries and 10.002 atttributes, where the attribute `Class` is the only column of the type object, all others are integers. It has 50 unique values which represent the customers. We were given no data description of the attributes or meanings behind the values, therefore, it is quite hard to get a feel for this data set or show graphics for the overwhelming size of the attributes. The different attributes could be a term matrix where in each column the number of a specific word in the review is documented.

1.4 Congressional Voting

The last data set, also from a competition from Kaggle gives us an overview of congressional voting based on questions answered by voters. The goal is to classify for which system of goverment a voter voted for, democrat or republican.

This data set has 218 entries and 18 attributes. The target value of this data set is `Class`, which party the people voted for. About 130 voted for democrats whereas the rest voted for republican. We predict in this classification task based on the questions voters answered with either yes `y` or no `n`. All attributes are nominal data. This data set is the only one of the four which has some missing values `unknown`. Figure (3) visualizes the missing values. How we will deal with it is explained in Subsection (2.4). There are around 40% `republicans` and 60% `democrats`.

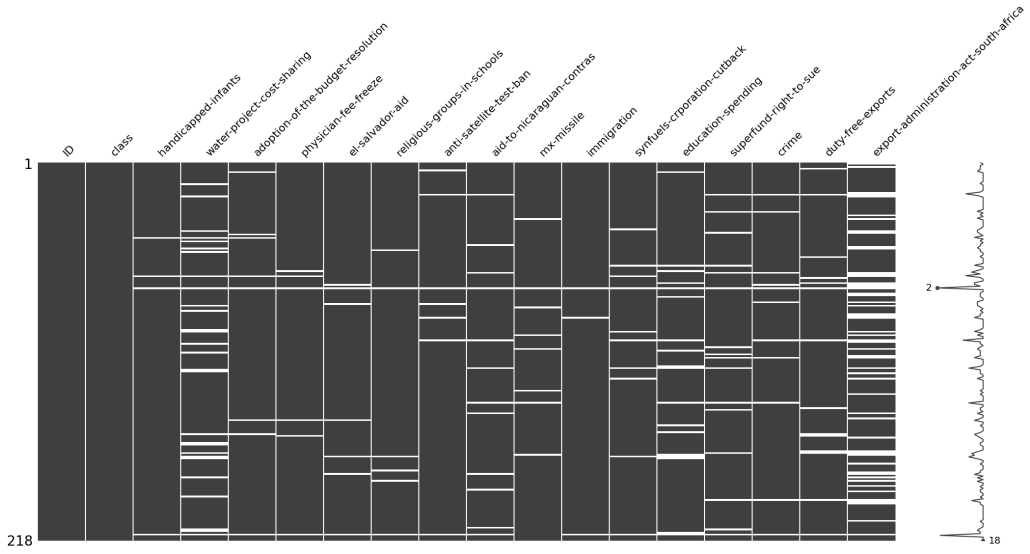

Figure 3: Missing Values

2 Pre-Processing

For pre-processing we looked at the types of the attributes, created dummy variables, detected outliers, handeled skewed data and missing values if existed. We first saved each change as a new data frame to compare the influence of different pre-processing afterwards. In the final code, however, one can see that we settled on creating a pipeline which included some of the pre-processing

steps like scaling or outlier handeling. The reason for this change was that we realized that we were pre-processing our data before splitting it into train and test set. This can create data leakage, which means the preprocessing process might include insights or statistics derived from the entire dataset, making the test data no longer independent (using the mean for imputing missing values, scaling etc.). Models trained with test data influence may fail to predict completely new data, as they have seen parts of the test set.

2.1 Gym Members Exercise Tracking

This data set has fortunately no missing values and also has already fitting types of the attributes. We focussed in outlier detection and scaling. First we created dummy variables for `Gender` and also our target `Workout Type`. `Female` will be 0 and `Male` will be 1, as well as `Yoga` = 0, `HIIT` = 1, `Cardio` = 2 and `Strength` = 3. This is the only step of the pre-processing which is not included in the pipeline.

For the outlier detection we tried out two different methods, through the interquartile range IQR and the Z-score. The boxplots[1] of all of the attributes can be seen in Figure (4). The IQR is the difference between the first and third quartile. Any number greater than the sum of 1.5× IQR and the third quartile is a suspected outlier, as well as, any number less than 1.5× IQR subtracted by the first quartile. With this method we found 41 outliers. For the Z-score method we labled every entry with a Z-score greater or equal to 3 as an outlier and through that had 13 outliers and with 2.5 as the threshold 44 outliers, however, we used 3 so there is a bigger difference to the data frame using IQR outlier detection. For the pipeline we settled on IQR method.

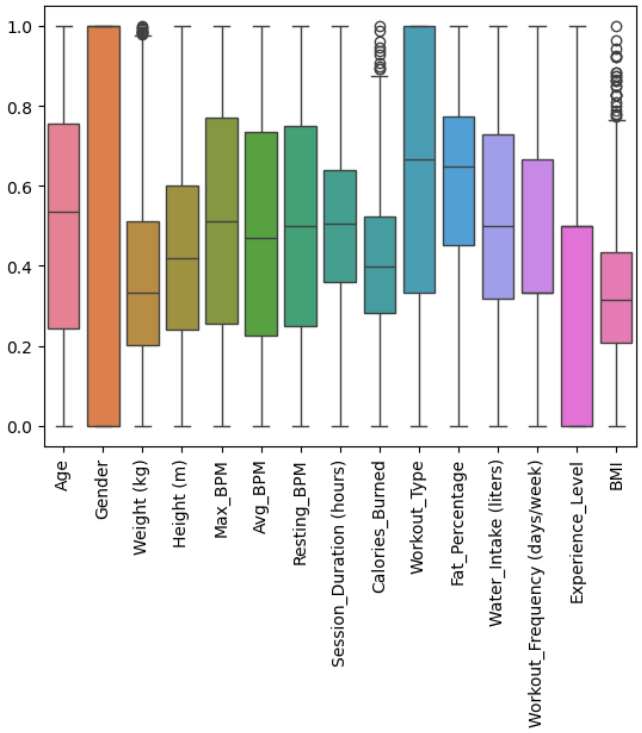

Figure 4: Outliers for Gym Data Set

[1]Note that this is from the scaled data since without scaling the plot would be not very informative.

Depending on the type of categorization algorithm we perform, scaling the data can influence the outcome of the categorization. From Figure (1) one was able to see that especially the attributes `Weight`, `Calories_Burned`, `BMI` and `Fat_Percentage` are not at all normally distributed. This can become a problem especially for using Support Vector Machines as the classifier. First we used an appropriate transformation, either logarithm (bmi, weight), square root (calories burned) or square transformation (fat percentage), however, in the final state of our code we decided on using the `StandardScaler()` which is provided by the Sklearn library. Figure (1) shows the effect of scaling.

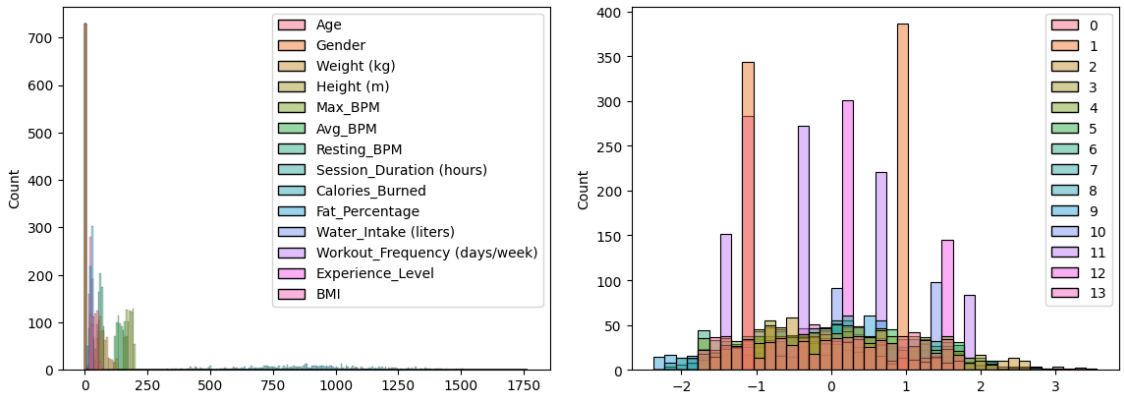

Figure 5: Histograms of not scaled and scaled Gym Data Set

We also looked at the correlation matrix. Every feature has a correlation less than 0.07 with the target `Workout_Type`. Therefore we already did not expect our models to do very well on this data set.

2.2 Abalone

First, we confirmed that there were no null values in the dataset. However, two rows with a `Height` value of 0 were identified as likely false observations and were removed. The categorical attribute Sex was encoded numerically, with Female (`F`) replaced by 1, Male (`M`) by 2, and Infant (`I`) by 0. For the target attribute `Rings`, which indicates the age of abalones, we created three age classes: young, adult, and senior. To address outliers present in each numerical attribute, we compared and evaluated two methods—Interquartile Range (IQR) and Z-score scaling—for outlier detection and removal.

Standardization and scaling were also considered for the numerical variables. Examining the variances revealed that `Whole Weight` had a significantly higher variance compared to other attributes, making it a strong candidate for standardization. Additionally, the correlation matrix was analyzed, showing that `Rings` has the highest correlation with `Shell Weight`. The other attributes exhibited similar correlations, ranging between 0.42 and 0.57. Regarding the distribution of the `Sex` attribute, the dataset contains the highest number of males, with infants slightly outnumbering females.

2.3 Amazon Reviews

For this data set we had the least information on. Fortunately it also has no missing values, because it would be quite difficult imputing missing values we do not know anything about. When using either IQR or z-scores both methods lable every entry as an outlier. This can happen when working with such a data set which has so many features (10.000). In high-dimensional spaces,

distances between points can be less meaningful, *curse of dimensionality*, which can cause distance-based metrics like Z-scores to produce extreme values. Additionally, if our data is highly skewed in any column, this can make typical IQR and z-score thresholds capture too many points as outliers. When working with such many features, feature selection is almost a must. We dried many feature selection methods like univariate selection or PCA but settled on *Recursive Feature Elimination* (RFE) and selecting the k best by using Chi-squared. For the scaling we also considered `StandardScaler()` again.

2.4 Congressional Voting

We will have to decide how to handle the missing value 'unknown' because two of our classifiers will not be able to handle missing values. We tried several different ways to handle the missing values in this data set. First we tried to keep the missing values as a category. Next we tried to impute the missing values with the median and lastly we used the MICE algorithm to impute the missing values. But at first we deleted the rows where we had more then 10 missing values. In a next step we use dummy variables for *Class* and the other attributes. 'Democrats' will be 0, 'Republican' will be 1, 'n' will be 0, 'y'. For the data set where we use an extra category for the missing values 'unknown' the value is '2'. In the our pipeline we did got rid of the outliers and scaled our data. We have only one column with outliers *export-administration-act-south-africa*. We used flooring and capping for the outlieres. We looked at how good each of these data sets compared in the training stage. There wasn't really a huge difference in the different methods for handling the missing values. Imputation with the median gave overall the best results.

3 Classification

In this section we will shortly describe the classifiers we chose. We will give an overview how we trained the models for the different data sets and what to keep in mind about when training and comparing the different models for each data set.
For the classification task we have chosen three different classifiers:

- **Random Forest Classifier**

  Random Forest leverages multiple decision trees to process large, complex datasets, handle high-dimensional feature spaces, and provide insights into feature importance. Random Forests are especially effective at achieving high predictive accuracy while reducing the risk of overfitting.

- **Support Vector Machines**

  Support Vector Machine (SVM) excels by identifying the maximum separating hyperplane that divides classes in the feature space, making it robust for both binary and multiclass classification tasks and still effective in cases where number of dimensions is greater than the number of samples. It uses a subset of training points (support vectors) in the decision function, so it is also memory efficient.

- **Multilayer Perceptron**

  A Multi-Layer Perceptron (MLP) is a type of artificial neural network composed of multiple layers of interconnected neurons. Each neuron uses nonlinear activation functions, enabling the network to model and learn complex patterns in data. MLPs can capture nonlinear relationships, making them highly effective for classification.

To be concise and make the different classifiers comparable we used the same training test splits where we used the function `train_test_split` with the split being 3:1.

For Support Vector Machines we always considered three kernels, linear, polynomial and the Radial Basis Function (RBF) which the latter is used to perform transformation when there is no prior knowledge about data with a radial basis function for improvement. The other values like the degree of the polynomial kernel or the γ for the RBF kernel are all given arrays.

For Random Forest we tried with different hyperparameters including number of trees in the foreset, max number of features considered for splitting a node, max number of levels in each decision tree, min number of data points placed in a node before the node is split, min number of data points allowed in a leaf node `min_samples_leaf` and method for sampling data points e.g. `bootstrap`.

For Multilayer Perceptron we tried different sizes and numbers of hidden layers. We tried also different activation functions like `tanh`, `relu` or a logistic function. We used different solvers, values for the regulation parameter alpha, and so on.

After creating our parameter grid we use `GridSearchCV` or `RandomizedSearchCV` with 3 to 5 folds which helps to loop through our predefined hyperparameters and fit our model on the training set. This and the above explained preprocessing steps were all inculded in a pipeline which was then adjusted to fit each data set. We created this pipeline so that we can do the pre-processing for each fold when using cross validation.

3.1 Gym Members Exercise Tracking

As mentioned in Section (2) when we went over the pre-processing steps, we saved a data frame for each change in the data set. For the gym members exercise tracking data set we have the original one - of course encoding has been done -, two with outliers removed through the IQR and Z-score method and finally also all three of them scaled. So all together we have six data sets. The following table, Table (1), gives a general overview of the accuracy scores of the models for each of the different gym data sets. All of the values are in percent.

Dataframe	SVM	RF	MLP
df_gym	24.03	26.64	22.32
df_gym_scaled	24.03	26.18	24.89
df_gym_IQR	27.47	28.33	28.33
df_gym_IQR_scaled	23.61	26.61	24.46
df_gym_Z	21.67	23.33	27.08
df_gym_Z_scaled	22.08	23.36	23.75

Table 1: Accuracy Scores of the Predictions of the three Classifiers

The scores from the above table were created when we haven't worked with a pipeline and were pre-processing (except of scaling) before splitting the data. We also did only do holdout for these results. Table (8) shows the results of using the pipeline and also compares holdout and cross validation. The parameters 1 and 2 were used for all data sets just to get an idea on how they perform with different data sets:

Classifier	Parameter#	
SVM	1	`C=1000, gamma=0.001, kernel='rbf'`
SVM	2	`C=1000, kernel='linear', probability=True, random_state=42`
RF	1	`n_estimators = 1000, max_features = 'log2', max_depth = 500, min_samples_split = 200, min_samples_leaf = 200, bootstrap = False`
RF	2	`n_estimators = 800, max_features = 'sqrt', max_depth = 65, min_samples_leaf = 30, bootstrap = True, criterion='log_loss'`
MLP	1	`activation='tanh', alpha=0.05, hidden_layer_sizes=(20, 25, 50, 50), max_iter=3000`
MLP	2	`activation='tanh', alpha=0.5, hidden_layer_sizes=(10, 30, 30, 50), learning_rate='constant', max_iter=2000, solver='adam'`

Table 2: Random Parameters

	Holdout or CV	SVM	RF	MLP
best parameters	H	24.03	26.64	22.32
best parameters	CV - KFold	23.61	26.61	24.46
parameters 1	H	24.03	26.18	24.89
parameters 2	H	27.47	28.33	28.33

Table 3: Accuracy Scores of the Predictions of the three Classifiers with the Pipeline

As one can already see all of the models performed poorly on each of the data sets. This was already expected since none of the features have a high correlation to our target value. With the data set that uses IQR outlier detection and is not scaled we had the best results for all of the models. Which we did not expect, since we thought scaling would make it better.

The best results were made through the Random Forest Classifier. This classifier used a maximal depth of 70, 1000 trees 2 as a minimal number of data points allowed in a leaf node and used bootstrap. It has an accuracy of 28.33%.

	Precision	Recall	F1-score	Support
Yoga	0.32	0.34	0.33	58
HIIT	0.23	0.13	0.17	52
Cardio	0.27	0.28	0.27	61
Strength	0.29	0.35	0.32	62
accuracy			0.28	233
macro avg	0.28	0.28	0.27	233
weighted avg	0.28	0.28	0.28	233

Table 4: Classification Report for Random Forest for the Gym Data Set

Already the classification report and confusion matrix give us better results than the SVM, however, they are still bad. Yoga is classified the best and again HIIT the worst.

$$\begin{bmatrix} 20 & 7 & 16 & 15 \\ 17 & 7 & 14 & 14 \\ 13 & 6 & 17 & 25 \\ 13 & 10 & 17 & 22 \end{bmatrix}$$

3.2 Abalone

After analyzing the dataset in the data analysis part (2) we used the insights in preparing the train and test splits for training the model. In the pipeline we tried different combinations of `IQRFilter` vs `ZScoreFilter`, `StandardScaler` vs scaling invidual columns with `np.sqrt` vs no scaler.
Using the StandardScaler on every column didn't bring good results. The outlier detection introduced better results with IQR then with Zscore. Here are the first exploration results on the dataset regarding sclaing, outlier removal etc.

Dataframe	SVM	RF	MLP
abalone_encoded	76.72	76.15	77.49
abalone_IQR	80.94	80.94	81.03
abalone_Zscore	76.96	76.96	75.76

Table 5: Accuracy Scores of the Predictions of the three Classifiers

	Holdout or CV	SVM	RF	MLP
best parameters	H	77.20	75.86	77.30
best parameters	CV - KFold	73.37	75.29	77.11
parameters 1	H	75.77	72.80	77.49
parameters 2	H	77.20	75.76	76.82

Table 6: Accuracy Scores for abalone dataset of the Predictions of the three Classifiers with the Pipeline

All the models performed quite well on the abalone dataset. This can be explained, as the dataset is clean with not much outliers and small variances on the features and we categorized only 3 classes to predict. Random Forest performed the best concerning to processing time for training the model. The base model for Random Forest performed: The overall accuracy of the best model is **75.86%**.

	Precision	Recall	F1-score	Support
adult	0.78	0.87	0.82	661
senior	0.65	0.45	0.53	173
young	0.76	0.65	0.70	210
accuracy	-	-	0.76	1044
macro avg	0.73	0.66	0.68	1044
weighted avg	0.75	0.76	0.75	1044

Table 7: Classification Report

Model Performance Accuracy = 75.86%.

$$\begin{bmatrix} 578 & 42 & 41 \\ 93 & 78 & 2 \\ 74 & 0 & 136 \end{bmatrix}$$

The best hyperparameters for RandomForest for this dataset are:

$$\text{RandomForestClassifier}(\text{max\_depth} = 80, \text{min\_samples\_leaf} = 4, \text{n\_estimators} = 200)$$

With this parameters the performance showed an improvement of **1.28%** to the base model.
The IQR outlier removal was better then z-score.

We also looked at the confusion matrix and classification report of the other models. As they are very similar, these are not mentioned here.

Processing time	SVM	RF	MLP
base models	7min 46s	704ms	12min 46s

Table 8: Processing times for abalone dataset of the Predictions of the three classifiers

The processing time were much better for the random forest, so it performs very quickly.

3.3 Amazon Reviews

The amazon reviews data set was not only the one with the highest dimension but also had the most classes (50). The number of features were much higer than the number of observations. So feature selection was an important step for this dataset. We experimented among different feature selection algorithms, e.g. `SelectKBest` and `RFE(estimator=RandomForestClassifier)` with 100 features to select.
Our first basic experiment result are shown in the table below.

Dataframe	SVM	RF	MLP
df_reviews	50.53	77.13	62.23
df_rev_rfe	44.15	67.02	54.79
df_rev_k	38.30	63.30	59.57

Table 9: Accuracy Scores of the Predictions of the three Classifiers

The best performing model was RandomForest without performing a feature selection before. We think, that is because RandomForest is a model, which does the feature selection implicitly while running.

	Holdout or CV	SVM	RF	MLP
best parameters	H	57.98	63.30	65.43
best parameters	CV - KFold	55.45	61.04	61.85
parameters 1	H	57.98	12.77	42.02
parameters 2	H	56.38	33.51	45.74

Table 10: Accuracy Scores of the Predictions of the three Classifiers with the Pipeline

	Precision	Recall	F1-score	Support
accuracy			0.65	188
macro avg	0.69	0.64	0.63	188
weighted avg	0.69	0.65	0.64	188

Table 11: Classification Report for MLP for the Reviews Data Set

3.4 Congressional Voting

We saved data frames for the different pre processing steps we did and used them when we did the classification task. For the voting data set we have one where we keep the missing values as a category, one where we impute them with the median value and one where we used the MICE algorithm. So we have three data sets. The following table, Table (1), gives a general overview of

the accuracy scores of the models for each of the different voting data sets. All of the values are in percent.

Dataframe	SVM	RF	MLP
df_voting	96.30	98.15	94.44
df_voting_keep_unknown	96.30	98.15	92.59
df_voting_impute_median	92.59	98.15	98.15

Table 12: Accuracy Scores of the Predictions of the three Classifiers

	Holdout or CV	SVM	RF	MLP
best parameters	H	94.55	98.15	94.55
best parameters	CV - KFold	95.84	84.69	90.35
parameters 1	H	94.55	49.09	96.36
parameters 2	H	92.73	92.73	94.55

Table 13: Accuracy Scores for voting dataset of the Predictions of the three Classifiers with the Pipeline

We see that the pre processing didn't really have a big influence on the accuracy of the prediction of the model. What we can see is that the random forest model did the best and had the highest accuracy.
Random Forest Classifier has a maximal depth of 10, uses 2000 trees, 10 as a minimal number of data points allowed in a leaf node and doesn't use bootstrap.The best score for the training data is 0.96 and the model has an accuracy of 98.15%.

	Precision	Recall	F1-score	Support
democrat	1.00	0.97	0.98	32
republican	0.96	1.00	0.98	22
accuracy			0.98	54
macro avg	0.98	0.98	0.98	54
weighted avg	0.98	0.98	0.98	54

Table 14: Classification Report for Random Forest on dataset voting

$$\begin{bmatrix} 31 & 1 \\ 0 & 22 \end{bmatrix}$$

4 Conclusion

While comparing our three chosen classifiers Multilayer Perceptron (MLP), Support Vector Machine (SVM), and Random Forest (RF) on the 4 datasets we experimented with different preprocessing steps and parameters to tune our models. Even though choosing and tuning the hyperparameters are of course very important, the Random Forest was the fastest and also overall the most performant one. On the voting, abalone, and gym datasets it performed well without having high preprocessing requirements, and on the Amazon dataset it was helpful for the implicit feature selection. In contrast, the SVM classifier underperformed relative to the other models: We think, that it's because of its sensitivity to noise and class imbalances, particularly in datasets like gym, where

class distributions and attributes created challenges for SVM to choose the right classes and define boundaries between these classes.

In the preprocessing steps we saw that it played a significant role which preprocessing we did, because it had a great influence on the classifier performance. For the datasets like reviews we applied Recursive Feature Elimination (RFE) and scaling, feature selection. This helped manage the high dimensions of the model and aided the models MLP and RF. The MLP, while powerful and flexible, typically requires more data and careful tuning to reach its full potential, making its strong performance on data sets like Rmazon Reviews notable but less surprising given its suitability for high-dimensional problems. However, the high number of attributes may have reduced the computational efficiency of SVM. Even though it was the fastest overall on the smaller datasets such as voting. SVM's weaker performance on larger, noisier datasets and those with class imbalances is consistent with its reliance on well-separated data for optimal (or at least tried to) performance. These outcomes reflect the interplay between the classifiers' strengths, dataset characteristics, and preprocessing steps. For the data sets with outlier removal via IQR and scaling (e.g., gym and voting) we have seen that preprocessing mitigated variability. So RF was better due to its ensemble-based approach that is better in the presence of attribute interactions. The SVM classifier was not good at handling noise or overlapping class boundaries.

Although we tried both Holdout and Cross-Validation for our model evaluation, due to time reasons we could not fully explore it. In our case CV did not give us better results that Holdout but it is better to use it, since it offers a more robust and reliable estimation of model performance, particularly for small datasets like Congressional Voting and Gym. We did however compared the models as fairly as possible through the splitting of the data and maintaining consistent preprocessing acrross all the models and data sets.