Data Science Project Report

Matyáš Skalický*
Czech Technical University in Prague
skalimat@fit.cvut.cz

Joona Hietala*
LUT University
joona.hietala@student.lut.fi

Rosemarie Wijngaarden*
University of Twente
r.wijngaarden@student.utwente.nl

1 PARKINSON'S DISEASE DATASET

The following section describes the team progress on the Parkinson's disease classification dataset. The dataset is available at [1].

1.1 Description of the dataset

The dataset was gathered from 188 patients with Parkinson's disease (PD) (107 male, 81 female) and 64 healthy individuals (23 male, 41 female) with ages ranging from 33 to 87. The data was collected with a microphone set to 44.1 KHz. The recordings were of a sustained vowel /a/ with three repetitions from each subject. Several speech features were extracted from these recordings including which have been known to be affected by PD.

The study proposed the tunable Q-factor wavelet transform (TQWT) as a novel feature extractor to detect PD in speech recordings. This was investigated by comparing extracted features to conventional extracted speech features. For this purpose the dataset is composed of a high number of TQWT features and some baseline features.

1.2 Statistical description

The dataset consists of 755 columns representing different measured attributes and of 756 records each representing one measurement of a patient. There are 252 unique participant IDs in the dataset. All of the patients present in the dataset were either classified with PD, or not.

None of the patients was reclassified between the triplo measurements. The dataset does not contain missing or NA values. One of the measurements for patient ID 37 is duplicated.

Since all of the target values are either 0 or 1, we are dealing with a binary classification task. The target variable is highly unbalanced, which needs to be taken in consideration when evaluating the performance of classification models.

One of the attributes describes the gender of the patient. We can see that the gender is correlated with the target classification variable, as more male patient have PD. What is also interesting, is that the genders are not balanced in the dataset.

Histograms representing each attribute were generated to get an overview of the different attributes (Fig 1). This visualization was useful to see that the variables have different data ranges as well as distributions. However, since our knowledge of the domain is very limited, it is hard to draw any conclusions about each attribute.

Correlation matrix of each attribute in the dataset (Fig 2) shows that some of the variables are highly correlated. This indicates, that when using algorithms which presume independence of variables like Naïve Bayes, we should take this into consideration.

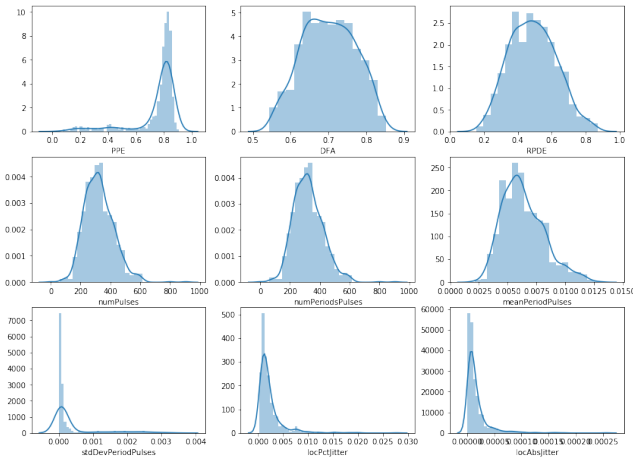

Figure 1: Analysis of the distributions of the variables.

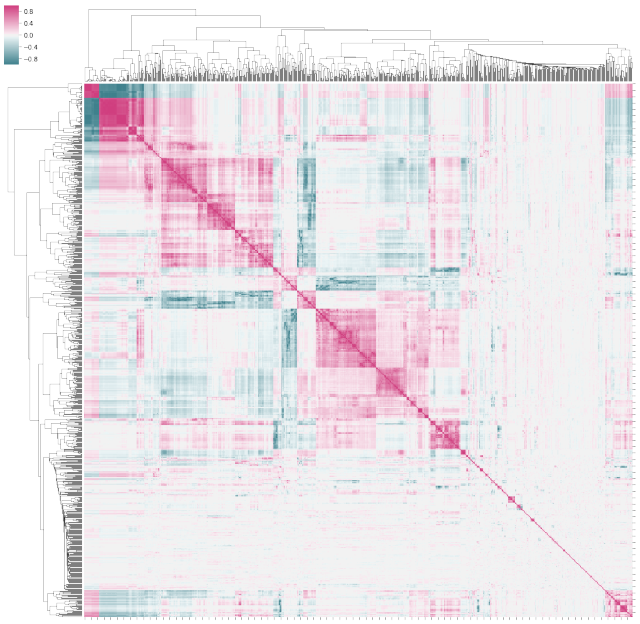

Figure 2: Correlation matrix of the Parkinson dataset.

1.3 Dataset preprocessing

A duplicate measurement of the patient ID 37 was removed from the dataset. Along with the main dataset, a standardized and a normalized datasets were prepared for use in algorithms which are sensitive to the scale and shift of the individual variables.

*Author ordering determined by coin flip over a Google Hangout.
[1] https://archive.ics.uci.edu/ml/datasets/Parkinson%27s+Disease+Classification

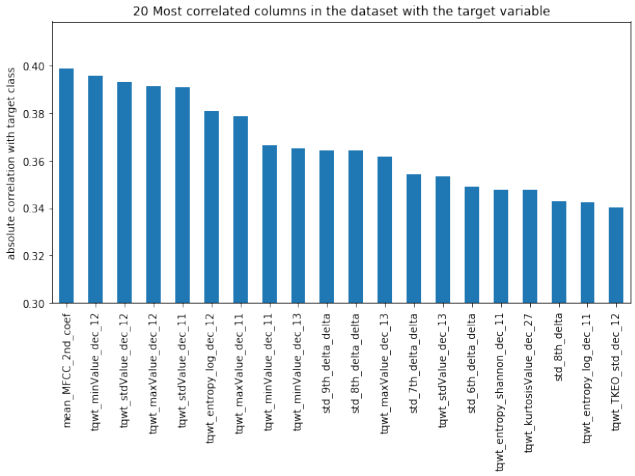

Figure 3: Most correlated variables with the target class.

In order to see the unbiased performance of the used models, 20% of the dataset was first put off as a testing dataset. We made sure that the data in the testing dataset contain the same distribution of the target value as the training dataset (Fig 4). This data is untouched until the model and hyperparameters are selected. Since the dataset contains usually 3 readings per patient, it was necessary to make sure that there are no samples from the patient of a same ID both in training and testing dataset. This would heavily alter the scores of algorithms such as KNN which could find the other measurements of a same person very easily.

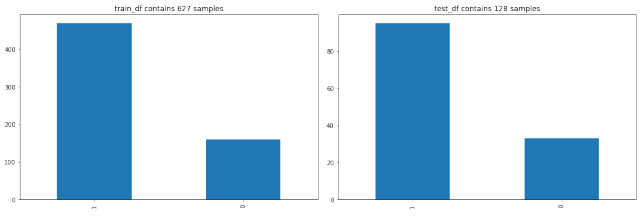

Figure 4: Class variable distribution in train/test datasets.

Instead of splitting the training dataset into a train and validation set, a 5-fold cross validation was used. To ensure, that the IDs of the patients are not mixed, a stratified K-Fold was first used on unique ID's only. These were then replaced with actual samples for ID. Stratified K-Fold guarantees the same distribution of the target class between all of the splits.

1.4 Unsupervised learning

1.4.1 Association rules. To extract the possible association rules, we have utilized the *mlxtend* library. Since the dataset contains a wide range of variables, we assume that the most interesting columns should be correlated with the target class variable. Ideal rule which would help with classification would be one that contains the target class as a consequent with high confidence.

We first selected 20 best features using the SKLearn's *SelectKBest* function. The dataset was then discretized (each feature into 3 possible cuts) and dummyfied.

The top five association rules sorted first by confidence and then subsequently by support can be seen in the Table 1.

Table 1: Extracted association rules with highest confidence.

antecedents	consequents	conf	supp	lever
tqwt_entropy_shannon_dec_12_low tqwt_TKEO_std_dec_13_low	tqwt_TKEO_mean_dec_12_low	1.0	0.973	0.019
tqwt_entropy_shannon_dec_12_low	tqwt_TKEO_mean_dec_12_low	1.0	0.97	0.019
tqwt_entropy_shannon_dec_11_low	tqwt_TKEO_std_dec_11_low	1.0	0.968	0.015
tqwt_TKEO_std_dec_12_low tqwt_entropy_shannon_dec_11_low	tqwt_TKEO_mean_dec_12_low	1.0	0.963	0.018
tqwt_TKEO_mean_dec_12_low	tqwt_TKEO_std_dec_11_low	1.0	0.963	0.015

We have also extracted the rules which might be potentially useful for classification – the rules with the class as the consequent. The results can be seen in the Table 2.

Table 2: Association rules with the class as consequent.

antecedents	consequents	conf	supp	lever
tqwt_entropy_shannon_dec_13_low, tqwt_entropy_shannon_dec_11_low, tqwt_TKEO_std_dec_12_low	class	0.791	0.73	0.041
tqwt_entropy_shannon_dec_13_low, tqwt_entropy_shannon_dec_11_low, tqwt_entropy_shannon_dec_12_low, tqwt_TKEO_std_dec_12_low	class	0.791	0.73	0.041
tqwt_TKEO_mean_dec_12_low, tqwt_entropy_shannon_dec_13_low, tqwt_entropy_shannon_dec_11_low, tqwt_TKEO_std_dec_12_low	class	0.791	0.73	0.041

As we can see, the algorithm has identified the antecedent *tqwt_TKEO_mean_dec_12* to have the highest support and confidence. Unluckily, we have not been able to find a "magic pill" combination of the parameters to find an exact rule to obtain the target class with a high confidence.

1.4.2 Clustering. The goal of unsupervised learning and in particular clustering is to find similar groups of data – clusters. First, we explore hierarchical clustering, where each sample first starts as a cluster. These clusters are then iteratively merged into larger groups.

To visualize the distances between the clusters, we utilize dendrograms. Dendrograms were generated for the raw, normalised and standardised dataset. An example for normalized data can be seen in Fig 5.

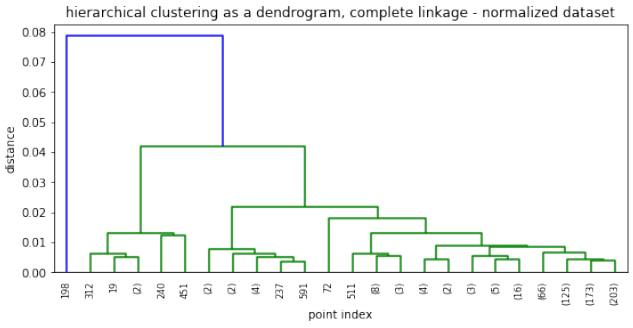

Figure 5: Dendrogram for normalized dataset.

The dendrogram can be used to determine the distance where we "cut" the graph and select the number of clusters. After trying

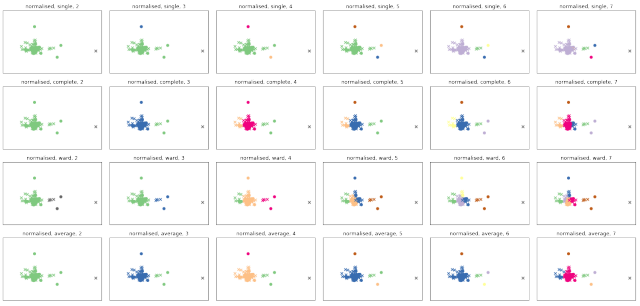

Figure 6: Hierarchical clustering on normalized dataset with different linkages and numbers of clusters.

out different cuts, we have visualizes different numbers of clusters also with different linkages. The result for normalized dataset can be seen in Fig 6. The different clusters are visualized with different colors and the target class as either 'o' or 'x'. The coordinates of the points were obtained by reducing the input dimensionality with PCA.

After the analysis of Fig 6, we were not able to identify clusters which would split the dataset with respect to the target class. Neither we have found any highly separated clusters in the dataset which could be easily explained.

We have then tried the agglomerative approach and evaluated the results. In order to determine the best number of the clusters, an "elbow" graph (Fig 7) is utilized. In this case, we have found that *k=5* clusters works the best and also that the clustering appears to be the most promising on a normalized dataset.

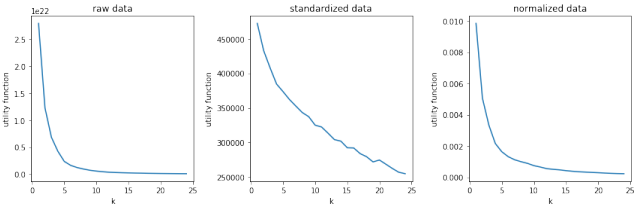

Figure 7: Elbow graph for different datasets.

Results of the K–Means clustering can be seen in Fig 8. We have not been able to obtain any definitive results regarding the data and especially the target class variable. An interesting anomaly which we have noticed that the patient ID 202 is usually in a different cluster and also plotted away from the other samples. By analyzing the results, we can see that this measurement differs a lot from the other measurements of the same patient. We can only guess what led to this measurement. Hearth rate is extremely high in two of the measurements, whereas the the patient was calm in the third measurement and his hearth rate was way lower.

1.5 Classification

We are dealing with binary classification task. We must take in mind, that the target class is heavily disbalanced, so that the classic accuracy score might not be the best metric. In order to compare the

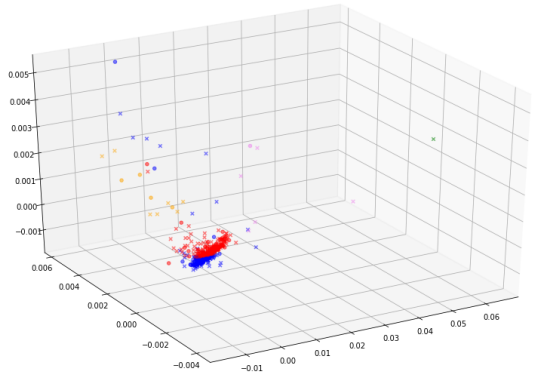

Figure 8: K–Means with k=5 on normalized data.

classifiers, we utilized the ROC AUC score averaged among all of the validation folds. Classifiers which are known to be performing badly on data with features of different data types and scales were also trained with standardized and normalized data and the results were compared.

To illustrate the effect of unbalanced target variable, a sklearn DummyClassifier was trained. This classifier predicts based on the the training set's class distribution with the result accuracy of 62.5%.

1.5.1 Naïve Bayes. Naïve Bayes (NB) expects the input variables to be independent. Since is is not parametrizable, no hyperparameter tuning was needed. First, NB classifier was trained on the original dataset with the accuracy of of 71.85%. In order to extract only the most correlated variables with the target, we utilized the SelectKBest class provided in the SKlearn library. It was observed that NB works best with top 13 variables and we were able to achieve the accuracy of 81.19% (Fig 9). As an experiment, we have tested reducing the input dimensionality by running PCA on the original training dataset. This has, as expected, not produced very usable results with the best accuracy of 76.54% with 1 dimensional output. This performance is however still better than with the original dataset.

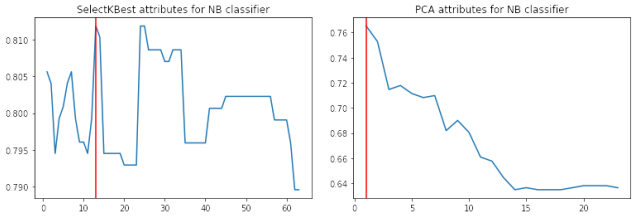

Figure 9: Selecting lower-dimensional inputs for NB.

1.5.2 Instance-based Learning. KNeighborsClassifier is very sensitive to different feature types and scales. This can be solved either by normalizing (and/or) standardizing the data. Primary hyperparameter of the KNN model is the number of neighbors (*k*). In order to obtain the best performing model, we have optimized the hyperparameters of the KNN for original, normalized and standardized

dataset (Fig 10). While the original dataset performed best for the n_neighbors: 5, the model trained on normalised and standardised data performed best for 13 and respectively 14 neighbors. While the performance of the model on the normalised and standardised datasets was roughly similar (74.77% and 75.09%), performance on the standardised dataset was superior with accuracy reaching 82.29%.

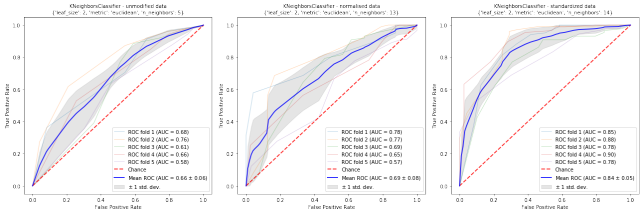

Figure 10: Comparison of KNN algorithm with raw, normalised and standardised data.

We have tuned the hyperparameters *leaf_size*, *metric*, *n_neighbors* and *weights*. The best KNN model was on standardized data with *leaf_size: 2, metric: euclidean, n_neighbors: 14, weights: uniform.*

1.5.3 Decision Trees. In order to obtain the best performance of a DecisionTreeClassifier, we have tuned the following parameters: *criterion*, *max_depth*, *min_samples_leaf* and *min_samples_split*.

A DecisionTreeClassifier was observed to achieve accuracy of 77.83% with the hyperparameters: *criterion: gini, max_depth: 5, min_samples_leaf: 3, min_samples_split: 5.*

1.5.4 Random Forests. RandomForestClassifier is an algorithm which is well known for performing well even without extensive data preparation. Unlike KNN, it is robust against different data scales and in this case outperforms a DecisionTreeClassifier by with an accuracy of 84.34%.

The best hyperparameters for RandomForestClassifier were tuned to values of *max_depth: 11, max_features: 7, min_samples_leaf: 2, min_samples_split: 0.001, n_estimators: 60.*

1.5.5 XGBoost. XGBoost classifier outperforms even the random forrest with the accuracy of 84.2%. We have achieved these results by tuning the hyperparameters to values of *n_estimators: 90, min_child_weight: 2.5, max_depth: 8, learning_rate: 0.15, gamma: 0.55, colsample_bytree: 0.8.*

We have also trained the SKLearn's AdaboostClassifier and compared its performance to the XGBoost. However with accuracy of 82.89% we found it's performance inferior.

1.6 Evaluation and critical analysis

We have evaluated all of the classification algorithms on a training dataset containing 80% of the data samples. We have utilized a 5–fold cross validation on the training dataset.

The only concern we have is that in case of KNN, the data standardization and normalization was done before the actual separation into each cross validation set. This way, we should expect getting better results since the data is normalized over a larger portion of samples. However this should be a) negligible and b), we are still only in the "classifier selection" phase, so the slightly skewed result should not be an issue.

Table 3: CV performance of evaluated classifiers.

	accuracy	ROC AUC
DummyClassifier	62.52%	0.50
GaussianNB	81.19%	0.80
KNeighborsClassifier	82.26%	0.84
DecisionTreeClassifier	77.65%	0.63
RandomForestClassifier	84.34%	0.83
XGBClassifier	84.20%	0.87
AdaBoostClassifier	82.89%	0.86

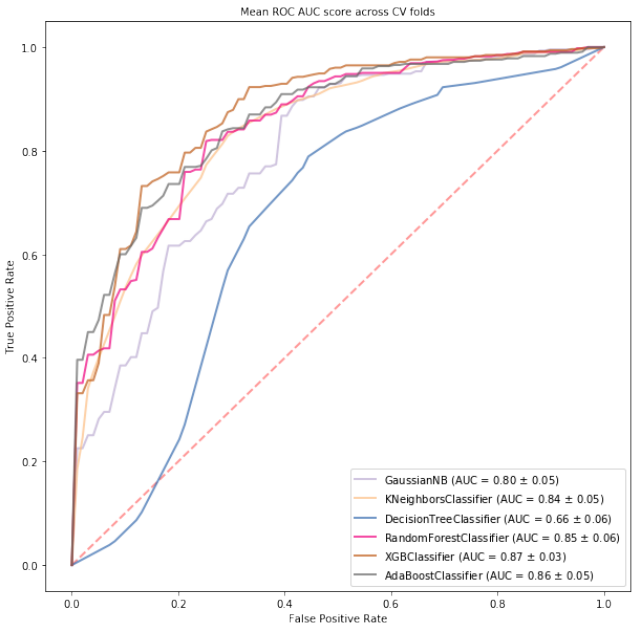

Figure 11: Comparison of the trained classifiers.

Based on the results in the Table 3 and from Fig 11 we have selected the XGBClassifier as the best performing classifier. Even though the RandomForestClassifier has achieved a slightly better accuracy, based on the data analysis, we know that the dataset is highly imbalanced and since that, we have chosen the ROC AUC score as our primary metric.

After the parameters of the target model were determined, we have used the untouched *testing* dataset which represents 20% of samples of the whole input dataset. The final performance is shown in the Table 4 below.

Table 4: Performance of the final classifier.

	accuracy	ROC AUC
XGBoost	77.34%	0.82

The final unbiased score is worse than the validation score, as expected due to overfitting. However the decrease in performance is very likely to be influenced by the small amount of the data in the testing set.

2 COVERTYPE DATASET

The following section describes the team progress on the Covertype dataset. The dataset is available at [2].

2.1 Description of the dataset

The dataset consists of observations about forest cover type from four wilderness areas located in the Roosevelt National Forest of northern Colorado. The actual forest cover type for a given observation (30 x 30-meter cell) was determined from the US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables were derived from data originally obtained from the US Geological Survey (USGS) and USFS data. Data is in raw form (not scaled) and contains binary (0 or 1) columns of data for qualitative independent variables (wilderness areas and soil types).

2.2 Statistical description

The dataset consists of 581012 observations (rows) and 54 features (columns), with one multi-class target variable. There are 10 numerical and 44 binary variables (4 area types and 40 soil types). The target variable holds seven different cover types (classes). Dataset does not contain any missing or NA values.

The original dataset is highly unbalanced in terms of the target variable (Fig 12), which need to be taken into account when preprocessing data and evaluating the classifier performance.

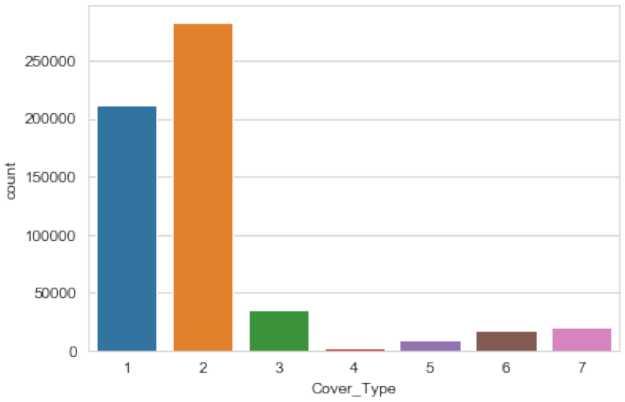

Figure 12: Target variable distribution before sampling.

By generating histograms of the ten continuous numerical variables (Fig 13) we can study the distribution of these features over the dataset. This was useful for understanding data at hand and to explain model performance further in the project. By presenting binary variable counts (Fig 14) we can see that some features represent the majority of the dataset. Yet again, we can't make any significant conclusions from these facts as our knowledge of the domain is limited.

[2]https://archive.ics.uci.edu/ml/datasets/covertype

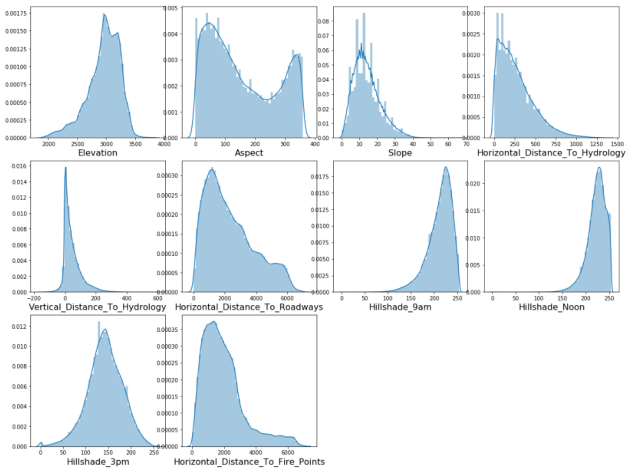

Figure 13: Distributions of continuous variables.

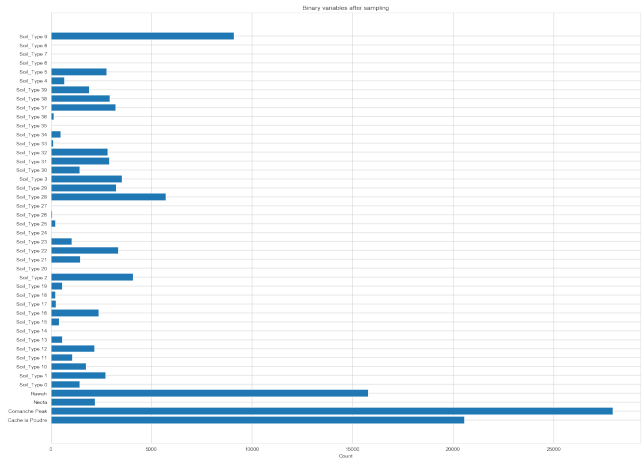

Figure 14: Distributions of binary variables.

From the correlation matrix containing the continuous numerical variables (Fig 15), we can conclude that these features are not strongly correlated with each other, except for a few individual variables that seem to have low correlation. This is generally a good thing for the classifiers, as all variables carry some importance in building the model.

2.3 Dataset preprocessing

As mentioned before and can be seen from target variable distribution (Fig 12), the dataset is highly unbalanced. Before training the models, we need to undersample the majority classes in order not to hinder the performance of our classifiers. We undersampled all classes to include under 10k samples by using RandomUnderSampler from imblearn library.

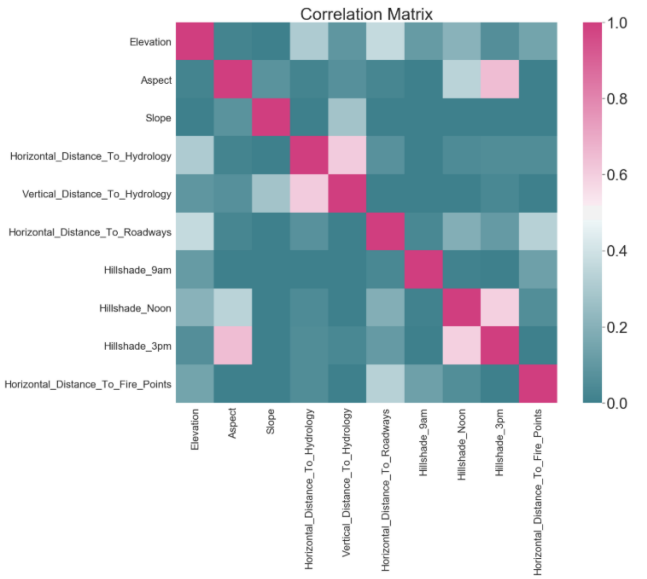

Figure 15: Correlation matrix of the Forestry dataset.

We first used SMOTE for class number 4 before undersampling in order to preserve a decent number of samples. SMOTE (Synthetic Minority Oversampling Technique) works by selecting randomly observations from the minority class and computing the k-nearest neighbors for this chosen observation. The synthetic observations are then added between the chosen observation and its neighbors, oversampling the minority class.

After balancing the dataset we have 66451 observations, equally distributed across all target classes, with 9493 observation in each class. We can use accuracy as a metric for evaluating the performance of the classifiers.

We split the dataset into train and test sets. The train set contains 80% of observations and it is used in model training and hyperparameter tuning with cross-validation. Hyperparameter tuning is done with RandomizedSearchCV with 5 folds, which does a random grid search for given hyperparameter distributions. This approach is less computationally heavy than the brute-force approach of a systematic grid search that tries every combination of given hyperparameters and offers us a tradeoff between trianing time and number of explored hyperparameter combinations.

The test set contains 20% of the data and is used to test the performance of the final model after hyperparameter tuning phase. This way, we can study if the chosen model is overfitted, meaning that the model performs significantly worse than while training.

2.4 Unsupervised learning

2.4.1 Association rules. For the pattern mining we first dummified the continuous numerical features into three features: high, mid and low. The association rules were extracted from dataset without feature selection. No rules containing the cover type as consequent were found. The 5 rules with the highest confidence, support and leverage, respectively, can be seen in Table 5.

Table 5: Extracted association rules with highest confidence, support and leverage.

antecedents	consequents	conf	supp	lever
Aspect_low, Hillshade_Noon_high	Hillshade_9am_high	1.0	0.379	0.041
Aspect_low, Slope_low	Hillshade_9am_high	1.0	0.368	0.04
Aspect_low, Slope_low Hillshade_Noon_high	Hillshade_9am_high	1.0	0.358	0.039
Aspect_low, Hillshade_3pm_mid, Slope_low	Hillshade_9am_high	1.0	0.34	0.037
Aspect_low, Hillshade_3pm_mid, Hillshade_Noon_high	Hillshade_9am_high	1.0	0.333	0.036

2.4.2 Clustering. Since the attributes of the Covertype dataset are combined binary and numerical, we have to take that in mind for clustering. In order to make the attributes comparable, we have first standardized the numerical attributes in the dataset. We have also subsampled the dataset down to 500 samples per each class in order to speed the clustering up.

First, we have attempted the agglomerative clustering approach. We utilized our knowledge gained in KNN and utilized a minkowski metric. In order to determine the number of clusters, we created a dendrogram as seen on Figure 16.

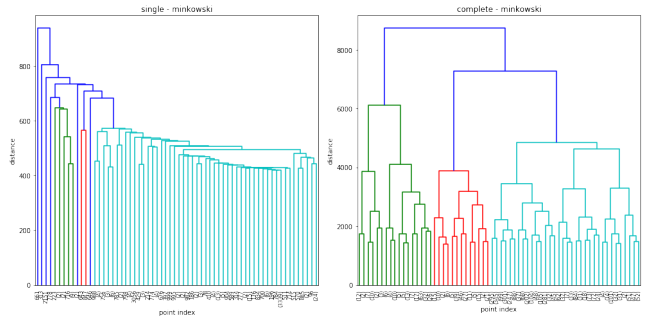

Figure 16: Dendrogram for single and complete linkage.

Based on the analysis of the dendrogram, we can conclude that the ideal number of clusters would be 5 to 7. The complete linkage has shown better results than the single linkage approach. We have decided to continue with 7 clusters.

The second clustering approach was done using the K–Means clustering algorithm. Based on the analysis an elbow graph, 7 clusters were selected. Output is shown in the Figure 17.

We have utilized the silhouette score to asses the clustering performance. We can see, that using the agglomerative clustering, this score is almost the same, only slightly better.

The output of the agglomerative clustering is more disbalanced than the KMeans. By looking at the Figure 18 we can see, that some of the classes such as *Ponderosa Pine* and *Willow* are usually placed within the same cluster. This could mean that these two classes are similar in terms of the measured attributes. On the other hand, for example *Krummholz* is often spread between many of the clusters.

2.5 Classification

As our target classes are balanced and multi-class, we chose to use the accuracy score, which is the default scoring method of

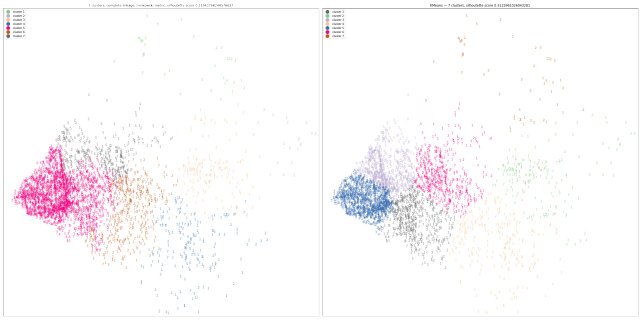

Figure 17: Agglomerative and K–Means clustering.

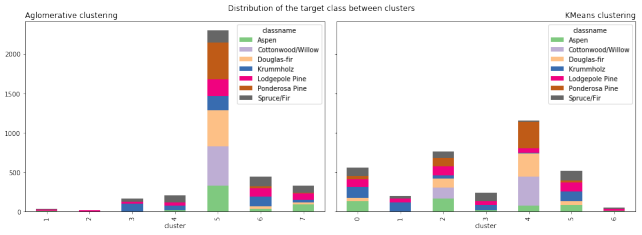

Figure 18: Comparison of target classes and respective assigned clusters

most classifiers. All of the presented accuracy scores are averages between the folds.

2.5.1 Naïve Bayes. We got an accuracy score of 59.88% with Naïve Bayes using the GaussianNB classifier on the training set. It was a good baseline model for benchmarking other models performances.

2.5.2 Instance-based Learning. We got an accuracy score of 88.43% with KNeighborsClassifier by using parameters: *n_neighbors: 5, metric: minkowski* using random grid search with 5-fold cross-validation and 10 iterations. It performed significantly better than NB and tree-based models (except XGBoost), probably due to the binary nature of most variables in the dataset, even without standardization. We trained KNN also with standardized data, but surprisingly, the results did not improve, resulting in an accuracy score of 85.11%.

2.5.3 Decision Trees. We got an accuracy score of 84.08% with DecisionTreeClassifier by using parameters: *min_samples_split: 9, min_samples_leaf: 3, max_depth: 14, criterion: entropy.* It performed fairly well, but not as good as KNN. This might be due to the fact, that the trees are easily overfitted with increasing depth of the model.

2.5.4 Random Forest. We got an accuracy score of 84.21% with RandomForestClassifier by using parameters: *n_estimators: 290, min_samples_split: 2, max_features: auto, max_depth: 12.* When comparing Random forest performance with decision trees, random forests achieve a reduced variance by combining diverse trees, usually at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model.

Table 6: CV performance of evaluated classifiers.

	accuracy
GaussianNB	59.88%
KNeighborsClassifier	88.43%
DecisionTreeClassifier	84.08%
RandomForestClassifier	84.21%
XGBClassifier	91.90%

2.5.5 XGBoost. We got an accuracy score of 91.90% with XGBoost by using parameters *n_estimators: 120, min_child_weight: 2.5, max_depth: 11, learning_rate: 0.25, gamma: 0.6.* It performed significantly better than other tree-based models and KNN.

2.6 Evaluation and critical analysis

The results of all classifiers are summarised in table 6. We can see that XGBoost had the best performance out of every classifier we tried out. XGBoost usually works very well for datasets with high number of features, outperforming other traditional classifiers, which is also the case of the Covertype dataset.

Next we evaluated the model with the test set to see the real performance. We evaluated the final model with a confusion matrix in combination with accuracy score to better understand model performance. These metrics can helps us debug and improve model performance if needed. Performance on the test set is presented in table 7. Our final classifier seems to do very well also on the test set, indicating that the trained models were not significantly overfitted.

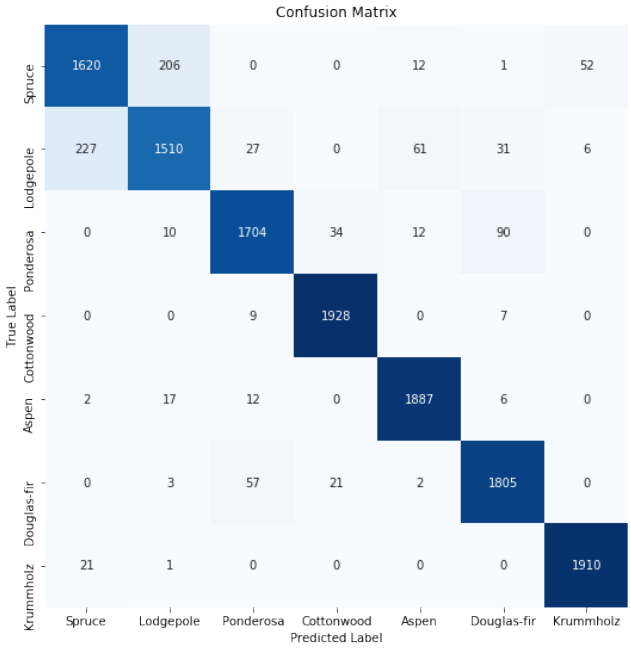

Figure 19: XGBoost confusion matrix on the test set.

From the confusion matrix (Fig 19) we can deduce that the model does not classify all the classes with equal performance. It lacks in classification of for example classes *Spruce* and *Lodgepole*, where

Table 7: Performance of the final classifier.

	accuracy	ROC AUC
XGBoost	93.03%	0.9589

there are relatively many false positives and false negatives. Our internet research revealed [3], that these trees prefer similar growing conditions such as dry soil and full sunlight. We need to keep this flaw in the classifiers performance in mind when possibly using the model.

One possible solution for increasing accuracy of predicting classes 1 and 2 could be to build several models. One model for classifying classes 1 and 2, and one model for classifying the rest.

3 DISCUSSION/COMPARISON

We have analysed two different datasets and build several classification models on top of the presented data. The PD dataset has a high dimensionality and a relatively small amount of records, we're dealing with a binary classification. On the other hand, the Covertype dataset has a high amount of records and a small amount of features. The records of this dataset are classified within 7 discrete classes. In both cases, the datasets are unbalanced.

Because the PD dataset has a relatively small amount of records, we made sure that both the training and test set had the same distribution amongst the target class, as seen in Fig 4. We also had to take into account the triple measurements of the patients and make sure that none of the patients ID were present in both the training and test sets. Because the Covertype dataset has such a high amount of records, the balancing could be done by undersampling the majority classes under 10k samples. Also SMOTE was used on class 4 to get a higher amount of samples.

Both datasets were split into a training set containing 80% of the data and a test set containing 20% of the data. For the validation of the models, a 5-fold cross validation was used in both cases.

Pattern mining was applied on both datasets to obtain the association rules. This technique can be useful to detect relationships between features, which could be used to detect interesting patterns between the features. This can be useful for somebody, who has a good understanding of the domain, however especially in the case of the PD, it was hard for us to reason some of the results. We have also tried to find the rules which have the target class as a consequent which could help us to identify some of the features useful for classification.

What we found in the research paper[4] for which the PD dataset was used is that the research was trying to prove that tqwt extracted speech features could be a novel feature for diagnosing PD. Most of the highest correlated variables in Table 3 are features extracted from tqwt. Also none of the baseline features are amongst the highest correlated variables, suggesting that the tqwt extracted features could be a better method for diagnosing patients with PD. In addition, the rules in Table 2 show rules containing tqwt features with class as consequent, which suggests a high correlation between the tqwt features and diagnosing PD.

We have utilized both the agglomerative approach and the KNN clustering algorithm on both of the datasets. We were not able to detect well separate clusters in context of the PD dataset. However, we were able to identify some of the interesting patients and measurements by analyzing the clustering results.

In terms of the Covertype dataset, the clustering seemed to make more sense. Based on the analysis of the resulting clusters, we were able to identify trees which were often clustered together and therefore we can assume, that these covertypes can be considered somewhat similar in terms of the measured attributes.

When using the NB classifier, the PD dataset scores higher with 81.19% than the Covertype dataset with 59.88%. NB is often used to get the base accuracy for classification models of the dataset as it requires no hyperparameter tuning. It makes a strong assumption that features are independent of each other. The lower score of the Covertype dataset can be partially explained by the features not being independent.

KNN works well on datasets with a low dimensionality and thus the Covertype dataset is expected to have a better performence with this classifier compared to the PD dataset. This is also the case as the Covertype dataset reaches 88.43% accuracy and the PD dataset only 82.26%.

Decision trees are robust against the need of extensive preprocessing as each feature is processed separately. However decision trees are well known to overfit easily. This problem can be resolved by using random forests or boosting to generate several decision trees and combining them into an ensemble classifier with better performance. This was also the case of our classifiers, in Table 3 and 6 we can see, that the DecisionTreeClassifier has, for both datasets, a lower performance in comparison to the RandomForestClassifier and XGBClassifier.

For the PD dataset the ROC AUC scores were used as the primary metric when comparing the classifiers, Fig 3. This was done because the dataset is highly unbalanced and the accuracy score can be misleading, where the ROC AUC shows more information about the overall performance of the classifier. The XGBClassifier was chosen as the best performing classifier, even though the RandomForestClassifier resulted in a higher accuracy.

Because the Covertype dataset is fully balanced after preprocessing, we decided that the accuracy score is good metric to base our final decision on. Thus form the table 6 can be concluded that XGBoost was the best performing classifier among both of the datasets. One of the drawback of this model is that compared to the traditional models such as decision trees, it is very hard to reason why the decision was made.

Nowadays, classifiers based on deep learning are usually the state of the art models. Especially on the datasets with enough data. However, the hyperparameter search space is vast. Our resources did not allow us to test this approach, however based on our experience, we believe, that a model based on neural networks would perform on the second dataset well.

[3] https://treetime.ca/compare.php?pcids=88-99

[4] A comparative analysis of speech signal processing algorithms for Parkinson's disease classification and the use of the tunable Q-factor wavelet transform. Applied Soft Computing, DOI: https://doi.org/10.1016/j.asoc.2018.10.022