Statistical Learning Home Exercise

Exercise 2 – Parkinson's Disease Classification Using Speech Signal Features

Saumya Gupta

h20saugu@du.se

May 16, 2021

Introduction

This report aims to describe the author's solutions to the second home exercise of the Statistical Learning course at Dalarna University during the second period of the Masters in the Data Science programme. The report will explain the two tasks solved as part of the exercise and the dataset provided for the same. After this, the document will state the data manipulations (if any) and describe the feature selection methods used for task 1. Then, it will walk the reader through the algorithms used as per the tasks, with and without PCA, for dimensionality reduction, establish the performance results, and finally discuss and compare the findings for each of the two solutions.

Background: Parkinson's disease (PD) causes motor system disorders, for example – speech disorder. According to Okan et al. (2019), 90% of the times, vocal impairments are exhibited in the early stages of the disease. Hence, several studies process speech signals and extract features valuable for PD assessments and build various reliable decision support systems. This analysis is related to performing PD classification for humans using extracted features from speech signals of diseased and non-diseased humans. In this analysis, we do not focus on the feature extraction methods. Out task is classification using these features and comparative analysis of algorithm performance for the two cases mentioned below. Hence, this report does not explain the details or the concepts related to feature extraction or the features themselves. As directed in the home exercise 2, the author will use the dataset developed in the analysis from Okan et al. (2019) to accomplish two tasks:

1. Perform classification using five algorithms – Decision Tree, Support-Vector Machine, Bagging approach, Random Forest, and Boosting method.

2. Use Principal Component Analysis (PCA) and perform the same classification task. Discuss the difference in the performance of algorithms between using the original features versus using the principal components.

Data and Related Methods

Dataset: As mentioned previously, the data comes from a previous study that wanted to perform a comparative analysis of different feature extraction methods for the speech signal PD assessments. For this, they collected data from 252 humans – 188 PD patients (107 men and 81 women) from the Department of Neurology in Istanbul University and 64 healthy individuals (23 men and 41 women). The study did three repetitions of the experiment (speech signal processing from sustained phonation of vowel 'a') with each of these individuals. Hence, the dataset has three rows of 'extracted features' for each of them. It results in 756 observation and 755 variables, which are features extracted using different methods. There are no missing data points. The author believes that the data is biased because we have data points belonging to each of the two classes in the ratio of 1:3. Still, this analysis takes no actions in these regards.

Variable Usage: In the dataset, an 'id' column identifies observations for the same individual. We do not use this integer column anywhere in the training of the models. Nevertheless, we do not drop them altogether since the analysis uses it later during majority voting in predictions to handle multiple observations for an individual. Apart from the extracted features, we also have a 'gender' variable, numerically coded into integer type (1 – male, 0 – female). Looking at the gender distribution for the two classes, we cannot comment on the importance of gender in detecting the disease. Hence gender is not discarded and is used in the same form. The rest of the variables are of numeric type. We convert the 'class' variable to factor type because many classification functions in R differentiate between regression and classification tasks using the class of response variable.

Standardization: The dataset consists of variables with different ranges. Also, some of the algorithms used are scale-sensitive, for example – PCA. Hence normalization is done for all variables except already encoded variables – 'gender' and 'class'. Of course, the 'id' variable is excluded from this too. The author believes 'app_entropy_shannon_10_coef', which is of type integer64, pose some problem in the scaling process (transforming to null values). Hence, we scale it separately.

High Correlations Among Many Variables: After normalization, we check the dataset for highly correlated variables (threshold set to > ± 0.8). We find that of 755 variables, 537 variables have a high relation with some of the other variables in the dataset. The report will talk about handling high correlations separately for both tasks.

Train-Test Split: We use 80% of the individuals for training, randomly picked from the pool of 252 diseased and non-diseased individuals and the rest 20% for testing our models. In this way, we ensure that all three measurements belonging to an individual go to only one set, not partially to training and testing. In other words, we can say we use the information of randomly picked 201 individuals for training and the remaining 51 for testing. Since we pick them randomly, the distribution of diseased and non-diseased individuals remains the same as that of the complete data.

Handling Samples from the Same Individual: Following Okan et al. (2019), we use majority voting on the predictions of the three samples to make a final prediction on the subject. We use the mode aggregation technique for it. For evaluation, we compare these predictions with the mode aggregated values of the actual labels.

Evaluation Metrics: We use f1-score and overall accuracy to measure the models' performance in the analysis. Below are the formulas:

- **Accuracy:** The accuracy is the average number of correct predictions. The higher, the better.

$$Accuracy = \frac{Total\ correct\ predictions}{Total\ instances}$$

- **F1-score:** It represents a balance between precision and recall. Its value lies between 0 and 1, 1 for perfect classification.

$$\frac{2 * Precision * Recall}{Precion + Recall}$$

Task 1

The first task is straightforward. We have labelled train data, and all the variables are scaled and numerically coded to move forward. However, we have many variables in our data, almost equal to the number of observations. Since task 1 does not mention anything about feature selection, we perform feature selection before training each of our algorithms.

Feature Selection Technique: Feature selection reduces the computational costs, which could mean less training time, facilitated by many folds in some cases, like trees. It could also improve the performance of models, not to mention the lesser chances of overfitting. One could make feature selections to cure the curse of dimensionality, handle high correlations between variables and discard the variables not significantly related to the response.

There are many implicit feature selection models. We use many of these in this analysis. The advantage of implicit feature selection methods is that they are relatively fast since these embed the selection process within the model fitting process; no external feature selection tool is required (Kuhn & Johnson,

2019). We use all 753 (removing 'id' and 'class') features as the starting feature set for each of our models. Below described are the feature selection and modelling details for each model:

Decision Tree:

- We first fit the data on all 753 predictors. The fitted tree uses 21 variables in its construction with 24 terminal nodes and gives a misclassification error rate of 0.02653.
- The tree, during its construction itself, performs some feature selection.
- We use 10-fold cross-validation to find the deviance as a function of cost-complexity parameter k and use the number of terminal nodes of the tree with the smallest deviance to prune the tree further. The new tree uses only three variables for its construction, but the training misclassification error increases to 0.1642 (6 times the error by the initially fitted tree).

Support-Vector Machine:

- SVM is a black-box model that does not have any implicit feature selection. Hence, we use lasso regression to make feature selection on all features.
- We perform 10-fold cross-validation for 'glmnet' (Simon et al., 2011) with the alpha value equal to 0.5 and use the cross-validated 'glmnet' model and the optimal value chosen for lambda to make predictions.
- Then, we chose the predicted coefficient estimates that are not 0 in the training of the SVM model.
- We use both linear and radial kernel while fitting the support vector classifier to check which kernel better explains the data.
- We tune the cost and gamma hyperparameters (gamma only for radial kernel) using grid search separately. For linear, we get the cost adjusted to 0.1, and for radial, cost and gamma parameters get tuned to 5 and 0.01, respectively.
- Linear and radial kernels produce 165 and 236 support vectors, respectively.

Bagging:

- We use feature importance scores calculated by random forest to perform feature selection.
- For threshold, we take the top 25 in terms of the average drop in accuracy value.
- For picking ntree - the number of trees to grow, we use the out-of-bag (OOB) estimate of error. 50 proves to be a good enough number of trees to grow since the OOB error stabilizes, increasing the number of trees further.

Random Forest:

- We use the same feature set selected using random forests during the bagging process.
- We find the optimal value of mtry – the number of variables to randomly pick at each split, using the OOB errors estimate again, selecting the mtry with minimum OOB error. We use 500 trees for this tuning step.
- We then fit the model with the tuned mtry value. Here too, we use the adjusted ntree value from bagging.

Boosting:

- We use feature relative influence scores calculated by stochastic gradient boosting to perform feature selection.
- For threshold, we take the top 25 in terms of the relative influence score.

- We need to tune the learning rate, the number of trees, and the tree complexity for minimizing the prediction error for boosting. However, we choose 100 trees and shrinkage of 0.01 to fit the model.

Results

Table 1. shows the overall accuracy and f1-score for the five models constructed using the same data. We have an evaluation for the decision tree before and after pruning, and for the support-vector machine, we have an assessment for the linear and radial kernel.

Table 1. Overall Accuracy and F1-Score for Model Trained on Original Selected Features

Evaluation Metric	Decision Tree	Support-Vector Machine	Bagging	Random Forest	Boosting
Overall Accuracy	Before 0.82 After Pruning 0.86	Linear Kernel 0.88 Radial Kernel **0.90**	0.86	0.80	0.84
F1-Score	Before 0.61 After Pruning 0.63	Linear Kernel 0.70 Radial Kernel **0.71**	0.63	0.50	0.56

- Support-vector machine with radial kernel performs the best both in terms of accuracy and f1-score. The scores are slightly better than those for the linear kernel, which means linear classifier could very well classify this data.
- Among the bagging, random forests and boosting models, bagging performs the best. We expect this, given it includes the cleverest averaging of trees.
- There is not much difference in these models' accuracy (0.80 - 0.90) because every model performs well. Nevertheless, we can say that the support-vector machine performs the best while the random forest performs the least in the test.
- Interestingly, in the case of the decision tree, the tree using 21 variables had a significantly less misclassification rate with training and performed less compared to the pruned tree with three variables. The former tree may be overfitting, and pruning helped create a better model.

Task 2

The second task asks to use the Principal Component Analysis (PCA) technique on the numerical features within the dataset. It uses an appropriate number of PCs as predictors for performing the same classification task.

What PCA Does: The dataset at hand is high dimensional and has many highly correlated variables. Performing PCA on such a data set can do two things. First, it can help us reduce the dimensionality where we can pick only those components that explain a significant variance of the input collectively, say 90%. Second, it can handle the high correlation problem by producing parts that would be uncorrelated. It is needed for our analysis in task 2, considering we are not using any other feature selection techniques.

There is much discussion regarding PCA usage before prediction algorithms, where the relationship between predictors and response is significant. PCA works without peeking at the outcome. Also, since we ignore some components that explain lower variances in PCA, we are losing some information for sure. We also lose on the interpretability of the predictors. We cannot determine which ones are

important and which ones are not from the perspective of prediction. While the PCs can improve the discriminative power of the classifier in some cases, it might not be a beneficial technique in a lot of other instances in which we already have feature selection embedded in the model fitting, for example – random forests.

As far as task 2 is concerned, since all the tree-based approaches discussed in the analysis have implicit feature selection, we will not apply PCA before them. It will work for sure, but the outputs, for example – the importance of PCs produced by random forests, will not make sense. The only model left for analysis then is the SVM model. We will apply PCA before classification using SVM and determine the difference in both – first where SVM worked with selected original features (discussed as part of task 1) and second where SVM works with PCs to perform the same classification.

We use the same starting 753 variables (removing 'id' and 'class'), all of which are numeric. Before PCA, we have the same scaled, train test data. After prediction, we use the same evaluation metrics on majority voted predictions.

Number of Components: If the number of dimensions, p > the number of the observation, n, n-1 PCs get constructed. However, here, the analysis produces n (= number of training observations) PCs instead of n-1. It is because we use the 'prcomp' function. Here, the calculation is done by a singular value decomposition (SVD) of the (centred and possibly scaled) data matrix, not by using Eigen on the covariance matrix (R Core Team, R Foundation for Statistical Computing, 2020). The SVD returns $min(row, column)$ singular values. We then use 90% as the threshold for cumulative input variance to choose the number of components. We could have gone for a higher number, say 98%, but this results in a sufficiently large number of support vectors on model fitting and overfitting even with enough tuned hyperparameter for the radial kernel. Hence, we move forward with 90% input variance.

Results

The PCA produces 603 (= number of training observations) components in total. Of these, we select the initial 105 components that explain 90% of the input variance collectively. The PCA object is used on the test to return the results, from which desired number of components are extracted and prepared for testing. We then use these selected PCs for training the algorithm. Fig. 1 shows the comparison of linear and radial kernel SVMs performance results for task 1 with lasso feature selection (original features) and task 2 with PCs.

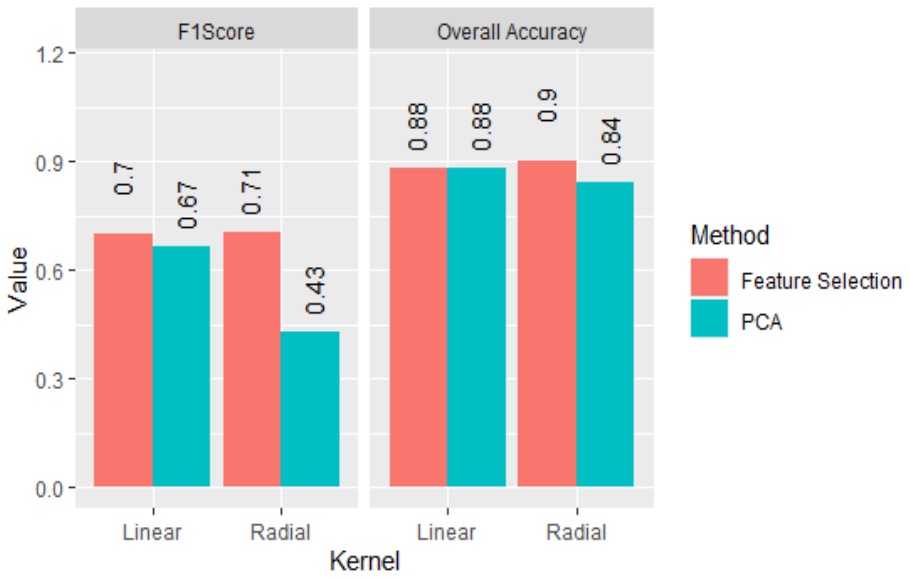

Fig. 1 Comparison between SVM (Linear and Radial) performance with feature selection on original features and that with principal components with the help of f1-score and overall accuracy.

- There is not much difference in the linear models in accuracy, but accuracy for radial drops significantly with PCA.
- In term of the f1-score, we have similar results. The linear and radial kernel SVMs trained on selected original features perform better than those trained on principal components. Radial SVM with PCA has a poor recall and precision.
- Also, models with different kernel perform equally well in the case of original features. However, in the case of PCA, radial performs worse than linear.

Discussion

The analysis tests the performance of implicit feature selection models and black-box models in classifying individuals into diseased and non-diseased for Parkinson's. In the first task, we use explicit feature selection for the SVM algorithm and compare the prediction performance with implicit feature selection tree-based models. SVM performed the best. In the second task, instead of feature selection on original features, PCA was performed to reduce dimensionality, and we used an appropriate number from the resulting PCs for training. Prediction results show that the model with original features performs better than the ones working on principal components.

Limitations

- Rightly pointed by Okan et al. (2019), the cross-validation techniques used in the study can cause biased results since each subject has multiple speech recordings and both training and test sets used in the experiments of these studies include the voice recordings of the same subject.
- Bias in the prediction model can occur due to unbalanced data, which we do not handle in this study.

References

Kuhn, M., & Johnson, K. (2019). *Feature Engineering and Selection: A Practical Approach for Predictive Models.* CRC Press. Retrieved from https://books.google.se/books?id=xy73DwAAQBAJ

Okan, S. C., Serbes, G., Gunduz, A., Tunc, H. C., Nizam, H., Sakar, B. E., . . . Apaydin, H. (2019, January). 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, 74*, 255-263. doi:https://doi.org/10.1016/j.asoc.2018.10.022

R Core Team, R Foundation for Statistical Computing. (2020). *R: A Language and Environment for Statistical Computing.* Vienna, Austria. Retrieved from https://www.R-project.org/

Simon, N., Friedman, J., Hastie, T., & Tibshirani, R. (2011). Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. *Journal of Statistical Software, 39*(5), 1-13. Retrieved from https://www.jstatsoft.org/v39/i05/