Parkinson's Disease Detection from Speech Features

Statistical Pattern Recognition and Analysis
SS22-CSE-802
Hamzeh Alzweri

May, 2022

1 Introduction

Parkinson's is a neurological disease caused by decreased dopamine levels in the brain, it is one of the most common brain diseases in the U.S. second only to Alzheimer, sixty thousands people are diagnosed with it each year. It is caused when the brain cells that make Dopamine stop working or die, Dopamine is the chemical responsible for coordinating movements, motivations and rewards. What makes Parkinson's special is that no specific path of symptoms exist, each patient faces a unique set of symptoms like loss of smell, imbalanced walking, sleep problems and depression. But, there are classical motor symptoms which are stiffness, slowness of movement and resting tremors. Parkinson's is usually diagnosed using neurological history information of the patient along with motor skills tests and observations, it would be useful if there is a way to diagnose Parkinson's without visiting the clinic or conducting any screening. One potential way to do this is using the features of speech signals, because Parkinson's also affects voice signals of the patient. Patients may experience difficulty in spelling words or making sounds, lowered voice tone and a low pitch range to count a few.

In this project, I will be experimenting with 4 different classifiers; Decision Trees, Bayesian Classifier, K-NN, and SVM in order to classify the vocal patterns of the individuals to patient, 1, or healthy, 0. Another target is to understand the relationship between these features and the decision of the

model, and how these features vary among Parkinson's patients and healthy individuals.

I will evaluate the results in terms of Positive Predictive Value, Negative Predictive Value, Sensitivity, Specificity, and F1 score. This is to make sure that the model evaluation is correct as we can have high f1 score but low Specificity(essentially, recall for negative values; $tn/(tn + fp)$) if the model is underperforming on the negative samples only.

2 Dataset

Data was gathered in a study conducted at the Department of Neurology in Cerrahpa A Ya Faculty of Medicine, Istanbul University. The data were gathered from 188 patients with PD (107 men and 81 women) with ages ranging from 33 to 87 and 64 healthy individuals (23 men and 41 women) with ages varying between 41 and 82 to produce 756 features. During the data collection process, the sustained phonation of the vowel was collected from each subject with three repetitions, so each patient has 3 records of features in the dataset.

2.1 Data preparation

To prepare the data for classification, I first removed the 432 TQWT features as suggested in the original paper for this dataset [1]. Then I observed high correlation between some features as in Figure 1. Therefore I decided to remove features with correlation higher than 0.6 and then apply Sequential floating forward selection to further reduce dimensionality and choose significant features, but I suspected that I'm removing some important features in the process, so I tested all possible scenarios as we will see in the experiments below.

I split the dataset into 80% training and 20% testing data, I kept two things in mind in the process which are:

1. **Set sampling**: The data is imbalanced, as we have more infected patterns than healthy patterns, so if I randomly split the data then the test set might include low number of healthy patterns, if any, which will yield wrong evaluation results. Therefore, I am splitting the data such that half of the test data is healthy and the other half is infected,

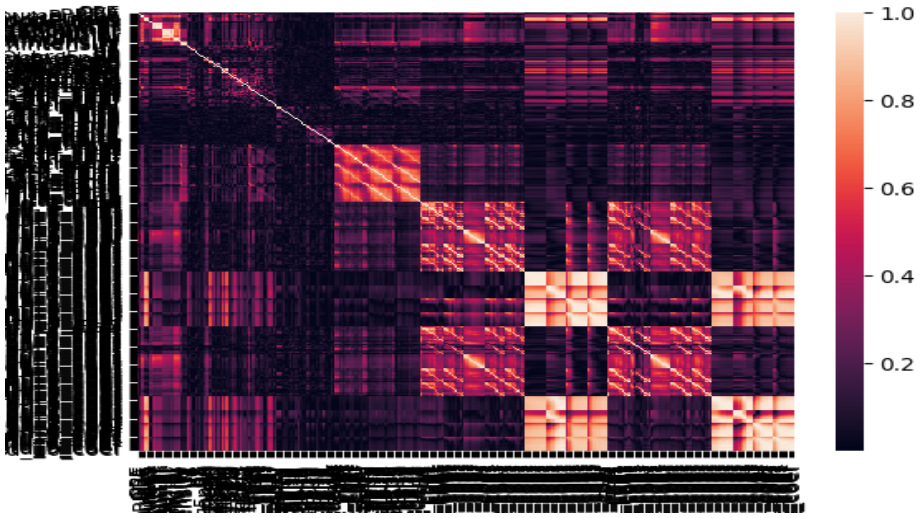

Figure 1: Heatmap of the correlation between features

this leaves the training data with a little bit more imbalance but we will deal with that in the training.

2. **Patient overlap**: Since we have 3 patterns from each patient, if we let our patterns that belong to the same patient to be sampled in both training and testing data, then the model will be able to classify them easily during testing since it already saw them during training, as patterns be- longing to the same patient will be similar, so we will be having an overly optimistic evaluation of the model due to this form of data leakage. Therefore, I split the data using groups of the *ID* feature such that the same ID of the patient can only belong to either training or testing data.

3 Experiments, Results and Analysis

For this section, I will present all the experiments along with their results and explanations. I found it more intuitive to present a walk through of the process of obtaining the final results than 3 separate sections. I will test multiple classifiers and report the experiments and results for each of them

below.

3.1 Experiments with Decision Trees

First, we need to select features as the data is highly dimensional, first thing to do is remove the TQWT features as suggest by the original paper for the dataset [1]. Now I am going to use feature selection (SFFS) to reduce the dimensionality, it's because if I use features transformation I will lose the interpretability of the features which is one of the main goals of this project. I will also try to remove correlated features to get a low number of features in order to avoid curse of dimensionality and unstable calculations while making training run faster. I hesitated whether to remove correlated features first or apply SFFS as some important features might be lost, I will test both these cases as well as using each individually and go with what gives better overall results, the code for this subsection is at the *DecisionTrees.py* script.

1. **Applying SFFS then removing correlated features:** results for applying feature engineering to obtain 30 features, then removing correlated features to end up with 12 features only (removed any corr ¿ 0.60 using the Pearson matrix). I also used the **"balanced"** weighting for the classes in the tree classifier which weights every class by its distribution's inverse in order to solve the class imbalance problem. For example, if we have in a group 5 patterns of label 1, 2 patterns of label 0, and the ratio is 3:1 then the tree will multiply the number of 0 labeled patterns by 3 and so counts them as 6 and chooses the label 0 for any pattern that falls into that group or bin.

   I also used a minimum sample split of 5 for each node to prevent overfitting, the below are the final features and results obtained:
   Final 14 selected features are:

   ['gender', 'rapJitter', 'GQ_prc5_95', 'mean_1st_delta', 'mean_7th_delta_delta' 'mean_9th_delta_delta', 'std_1st_delta', 'std_delta_delta_log_energy', 'std_12th_delta_delta', 'det_entropy_log_10_coef', 'det_TKEO_mean_2_coef', 'det_TKEO_std_9_coef', 'app_entropy_shannon_5_coef', 'det_LT_entropy_shannon_1_coef']

The following results were obtained when fitting the Decision Tree model mentioned above to these features:

PPV	NPV	Sensitivity	Specificity	F1
0.647	0.6	0.558	0.346	0.578

2. **Removing correlated features first then applying SFFS:** The 30 selected features are:

['gender', 'PPE', 'DFA', 'minIntensity', 'f1', 'f2', 'f3', 'f4', 'b2', 'b3', 'GQ_prc5_95', 'GQ_std_cycle_open', 'GNE_SNR_TKEO', 'VFER_NSR_TKEO', 'IMF_NSR_SEO', 'mean_MFCC_4th_coef', 'mean_MFCC_6th_coef', 'mean_MFCC_7th_coef', 'mean_MFCC_12th_coef', 'mean_1st_delta', 'mean_4th_delta', 'mean_6th_delta', 'mean_9th_delta', 'mean_11th_delta', 'mean_12th_delta', 'mean_delta_delta_log_energy', 'mean_7th_delta_delta', 'mean_9th_delta_delta', 'Ea' 'det_entropy_log_1_coef']

Results obtained when fitting the Decision Tree mode to these features:

PPV	NPV	Sensitivity	Specificity	F1
0.586	0.5208	0.512	0.320	0.531

The performance decreased for all metrics when I removed correlated features before applying SFFS, this suggests that there exists some specific combinations of features which are better at separating the classes and removing some features broke these combinations. So I will test the model using each of the uncorrelated features and engineered features directly in order to decide which feature set to go with.

3. **Fitting the model using uncorrelated features without using SFFS:** The resulting 69 feature set and results are below :

['gender', 'PPE', 'DFA', 'RPDE', 'numPulses', 'stdDevPeriodPulses', 'minIntensity', 'f1', 'f2', 'f3', 'f4', 'b1', 'b2',

'b3', 'b4', 'GQ_prc5_95', 'GQ_std_cycle_open', 'GNE_SNR_TKEO' 'GNE_SNR_SEO', 'GNE_NSR_SEO', 'VFER_mean', 'VFER_SNR_TKEO' 'VFER_SNR_SEO', 'VFER_NSR_TKEO', 'IMF_SNR_SEO', 'IMF_SNR_TKEO', 'IMF_NSR_SEO' 'IMF_NSR_TKEO', 'mean_MFCC_0th_coef', 'mean_MFCC_1st_coef', 'mean_MFCC_3rd_coef', 'mean_MFCC_4th_coef', 'mean_MFCC_5th_coef', 'mean_MFCC_6th_coef', 'mean_MFCC_7th_coef', 'mean_MFCC_8th_coef', 'mean_MFCC_9th_coef', 'mean_MFCC_10th_coef', 'mean_MFCC_11th_coef', 'mean_MFCC_12th_coef', 'mean_delta_log_energy', 'mean_1st_delta', 'mean_2nd_delta', 'mean_3rd_delta', 'mean_4th_delta', 'mean_5th_delta', 'mean_6th_delta', 'mean_7th_delta', 'mean_8th_delta', 'mean_9th_delta', 'mean_10th_delta', 'mean_11th_delta', 'mean_12th_delta', 'mean_delta_delta_log_energy', 'mean_1st_delta_delta', 'mean_2nd_delta_delta', 'mean_3rd_delta_delta', 'mean_4th_delta_delta', 'mean_5th_delta_delta', 'mean_6th_delta_delta', 'mean_7th_delta_delta', 'mean_8th_delta_delta', 'mean_9th_delta_delta', 'mean_10th_delta_delta', 'mean_11th_delta_delta', 'mean_12th_delta_delta', 'std_MFCC_2nd_coef', 'Ea', 'det_entropy_log_1_coef']

PPV	NPV	Sensitivity	Specificity	F1
0.612	0.562	0.538	0.346	0.556

4. **Fitting the model using features obtained from SFFS directly:**
   Features and results are below:

['gender', 'rapJitter', 'meanAutoCorrHarmonicity', 'meanNoiseTo-HarmHarmonicity', 'GQ_prc5_95', 'mean_1st_delta', 'mean_7th_delta_delta', 'mean_9th_delta_delta', 'std_1st_delta', 'std_delta_delta_log_energy', 'std_12th_delta_delta', 'det_entropy_log_10_coef', 'det_TKEO_mean_2_coef', 'det_TKEO_std_9_coef', 'app_entropy_shannon_5_coef', 'app_entropy_log_3_coef', 'app_det_TKEO_mean_6_coef', 'app_det_TKEO_mean_7_coef', 'app_TKEO_std_5_coef', 'app_TKEO_std_6_coef', 'app_TKEO_std_7_coef', 'app_TKEO_std_9_coef', 'det_LT_entropy_shannon_1_coef', 'det_LT_entropy_shannon_10_coef', 'det_LT_entropy_log_5_coef','app_LT_entropy_shannon_3_coef', 'app_LT_entropy_log_3_coef', 'app_LT_entropy_log_7_coef', 'app_LT_TKEO_mean_8_coef', 'app_LT_TKEO_std_8_coef']

PPV	NPV	Sensitivity	Specificity	F1
0.589	0.587	0.570	0.474	0.574

Based on these results, the best approach was to apply SFFS to all features and use them directly (its' F1 score is very close to F1 score of all other approaches, but the Specificity increases when we apply SFFS only, therefore I will choose this approach), next I will plot the Probability Density Functions of the uncorrelated features to check if there are any significant features left out since we are using SFFS only, plot is in Figures **2** and **3**.

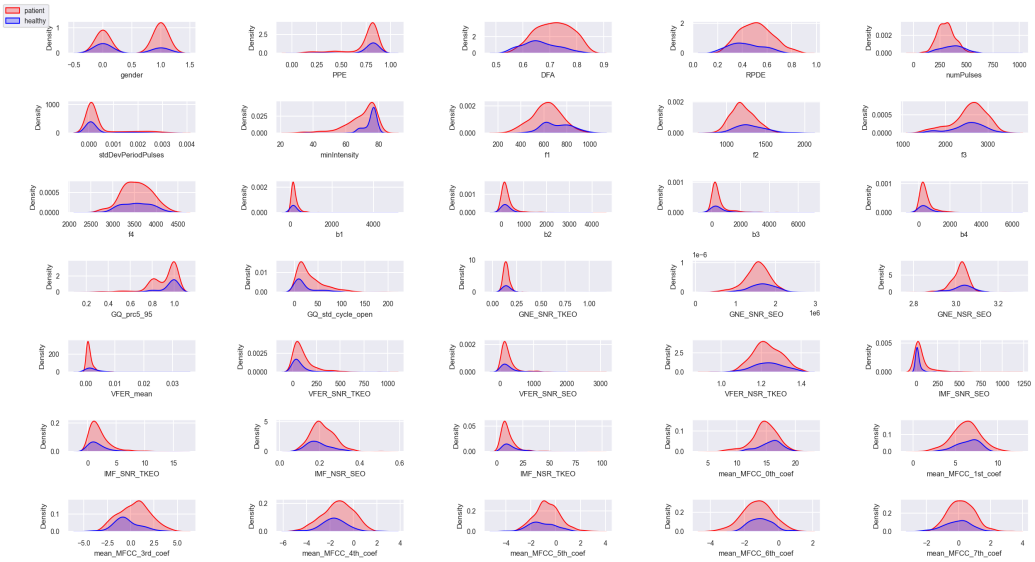

Figure 2: Densities of the uncorrelated features pt1

According to the distributions, it seems like SFFS is removing some features that have slightly different distributions for each class like RPDE, DFA, numPulses, and PPE, therefore I re-added these features and tested again to check whether they improve the model's performance and the following results obtained:

PPV	NPV	Sensitivity	Specificity	F1
0.687	0.680	0.608	0.410	0.625

All the metrics improved except for Specificity but its still higher than any other set, so I kept these features. I noticed that Shimmer and Harmonicity features are not included either, and based on domain knowledge they

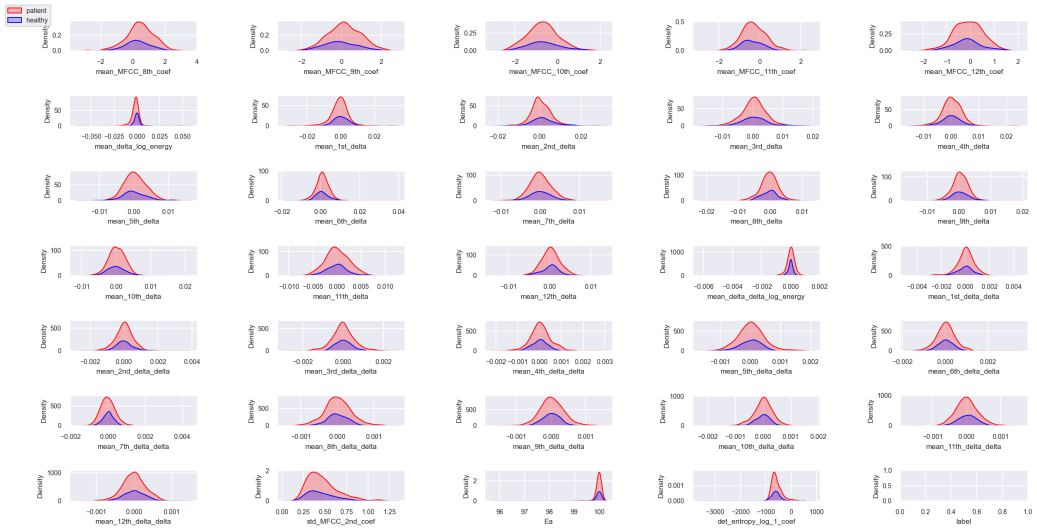

Figure 3: Densities of the uncorrelated features pt2

could be important features as Parkinsosn affects them, so I plotted their distributions in Figure 4 to double check.

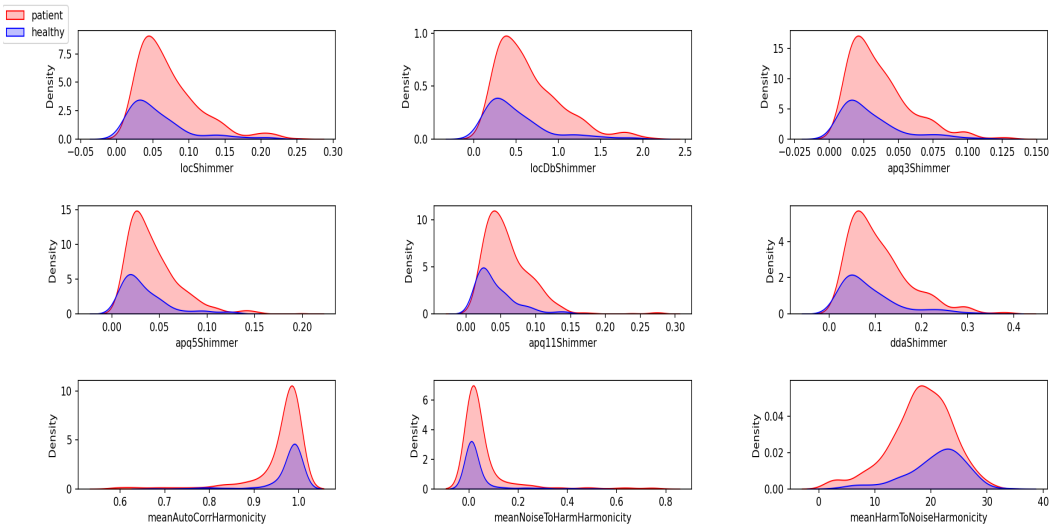

Figure 4: Densities for Shimmering and Harmony features.

According to the distributions, I decided to test after adding *mean-*

HarmToNoiseHarmonicity and *locShimmer* features as their means are a bit different for each class:

PPV	NPV	Sensitivity	Specificity	F1
0.683	0.692	0.628	0.461	0.638

Yet again, results improved, especially the Specificity, now I want to understand which features is the model using to make decisions, figure 5 shows the features importance according to the model.

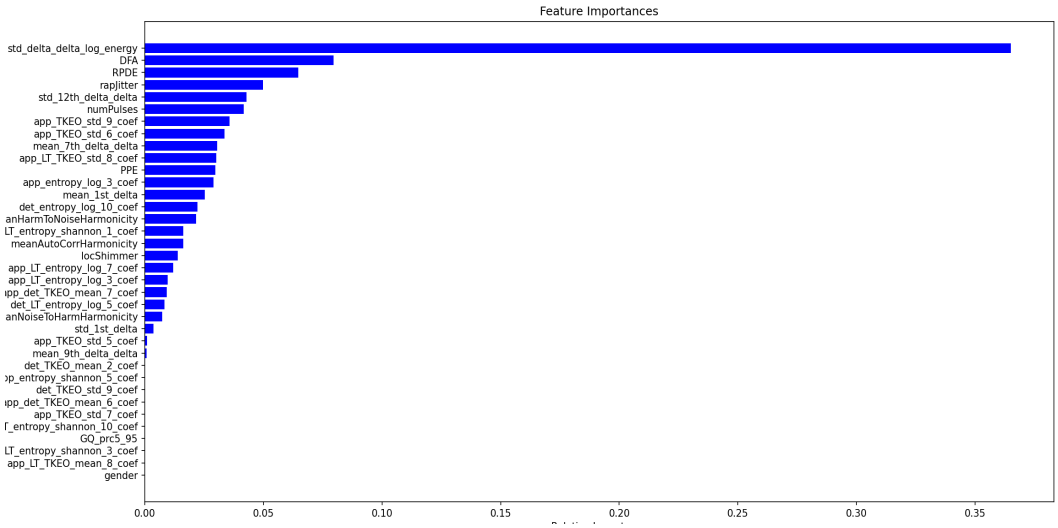

Figure 5: Features importance, here it is the normalized reduction of the Gini index brought by the feature.

As we can see, std_delta_delta_log_energy (energy here is energy of signals) feature is the most significant one in terms of distinguishability between classes, then comes the *DFA, RPDE, rapJitter, std_12th_delta_deleta* and numPulses. I am going to use box-plot of these features to show relationship of their values with the class label in Figure 6. This figure provides us with very useful insights about the vocal symptoms of Parkinson's, infected individuals tend to have higher values of jitter and energy than the healthy individuals while healthy individuals have higher number of pulses. I will now test these features with other classifiers.

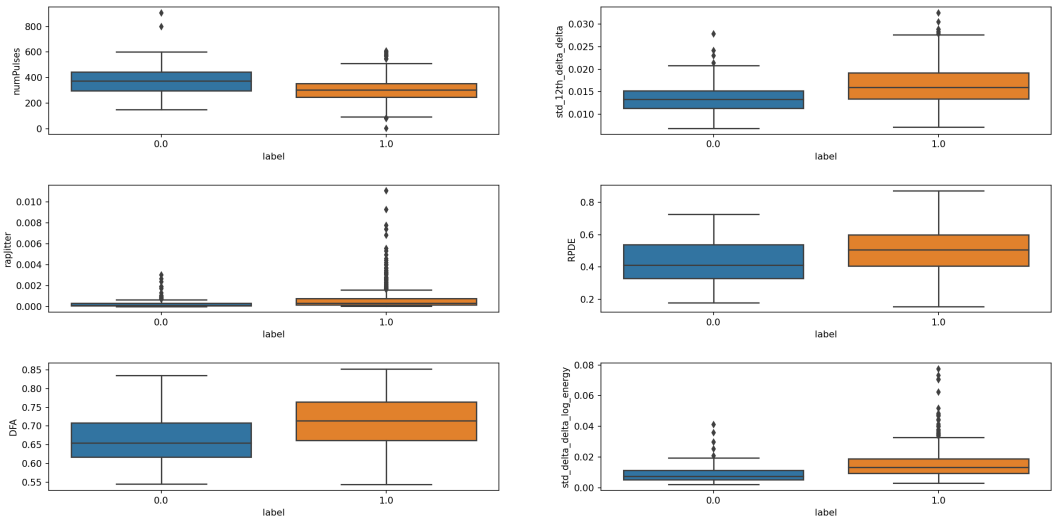

Figure 6: Relation of feature values to the classes.

3.2 Experiments with Bayesian classifier

Code for this part can be found in *BayesianClassifier.py*, I normalized the engineered features from the decision tree experiment earlier and removed gender as its categorical and its not important according to 5. I used the minimum risk $R(\alpha_i|x)$ rule where the loss of misclassifying healthy or "0" as patient or "1" is $\lambda_{10} = 5$ and the loss of misclassifying patients as healthy is $\lambda_{01} = 0.5$ to account for the class imbalance that we have, results:

PPV	NPV	Sensitivity	Specificity	F1
0.558	0.481	0.474	0.679	0.497

The classifier has lower performance than the decision tree's F1 score of 0.638, its under-performing negative samples and this could be due to the class imbalance as the density of class 1 or patients is being much greater than the density of healthy individuals in all features. To visualize if this is true, we plot the densities of the features in Figure 7.

As we can see, all the patient features have higher densities than densities of healthy patterns, this is due to the class imbalance problem since we have 486 patterns of patients against 114 healthy patterns in the training set. In an attempt to resolve this, I will under-sample the patients class such that

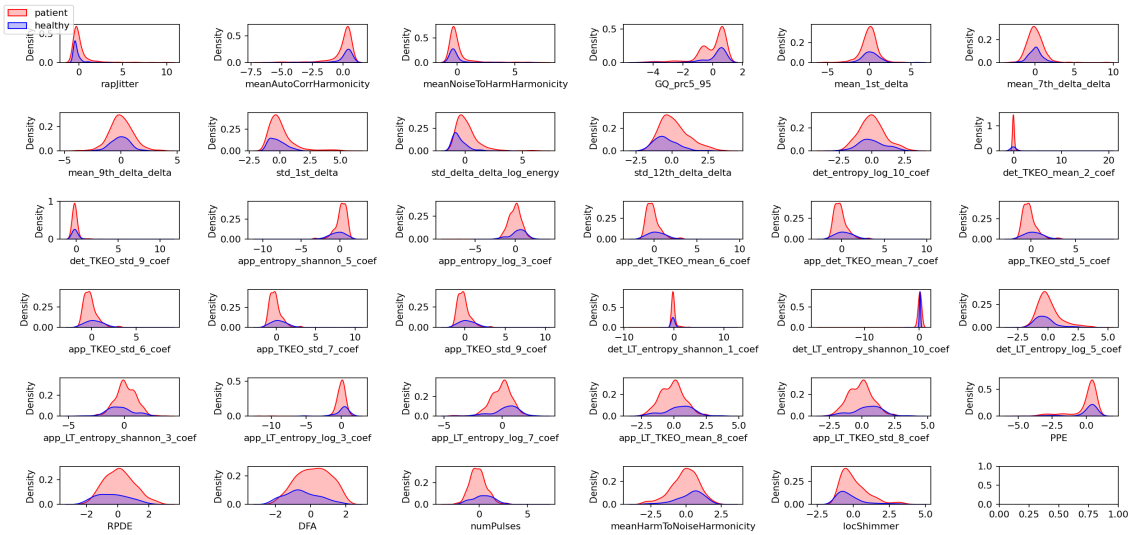

Figure 7: Class conditional density of the features.

the ratio of positive to negative patterns is 2:1 instead of approximately 4:1. Figure 8 shows the new distribution of the under-sampled data.

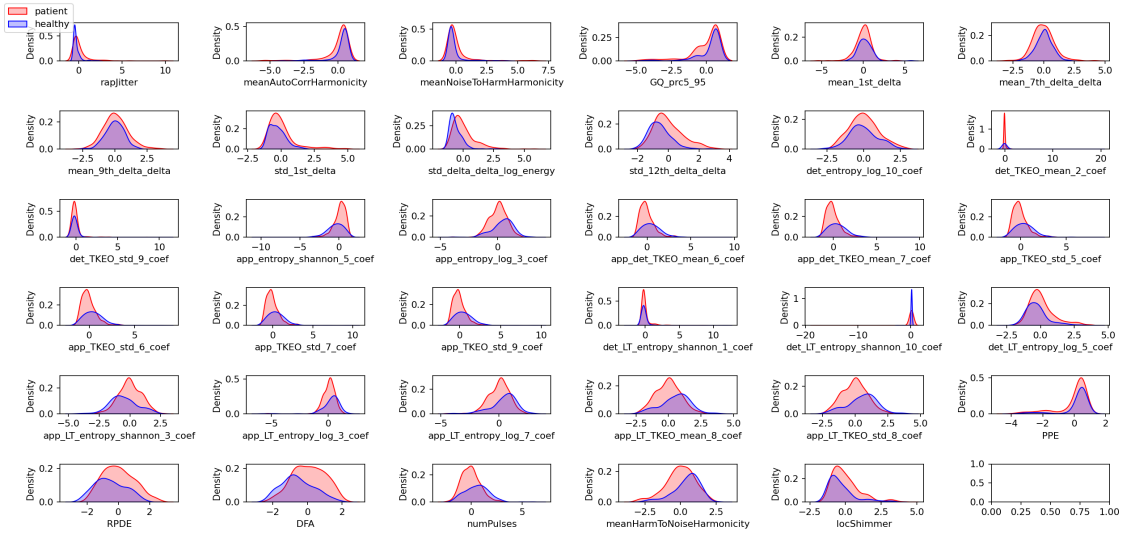

Figure 8: Class conditional density of the under-sampled features.

PPV	NPV	Sensitivity	Specificity	F1
0.545	0.527	0.532	0.615	0.535

In the above table, the F1 score increased indeed, currently I am using a ratio 0.5 of negative to positive patterns, for example 100 negative patterns and 200 positive patterns. I want to plot the F1 score against different ratios to find the best one while holding the loss mentioned above constant, check Figure 9.

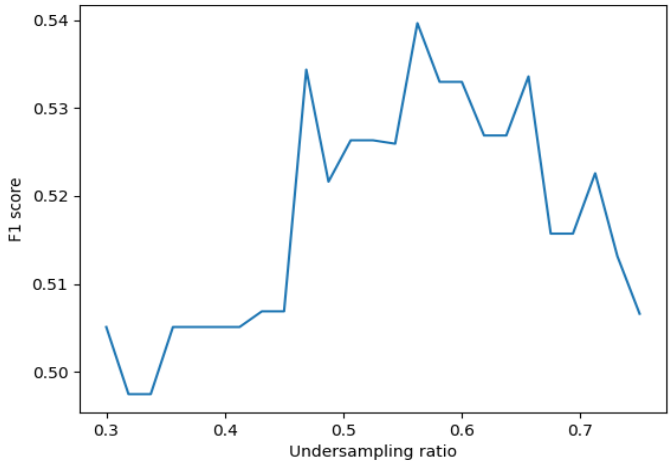

Figure 9: Sampling ration against F1 score

According to the results, ratio of 0.5625 yields the best f1 score of 0.539. This is still somewhat low, in order to understand the nature of the problem, I used PCA to show a scatter plot of the data in Figure 10.
Apparently, the two classes have high overlap using the features that I came up with in the previous experiment, which explains the low f1 scores that we are seeing. In Figure 11 I show the scatter plot after applying PCA using all the features without applying the SFFS, as we can see there is still a high overlap even without picking any features which means most of the error is caused by the nature of the problem. I will now move to the K-NN classifier.

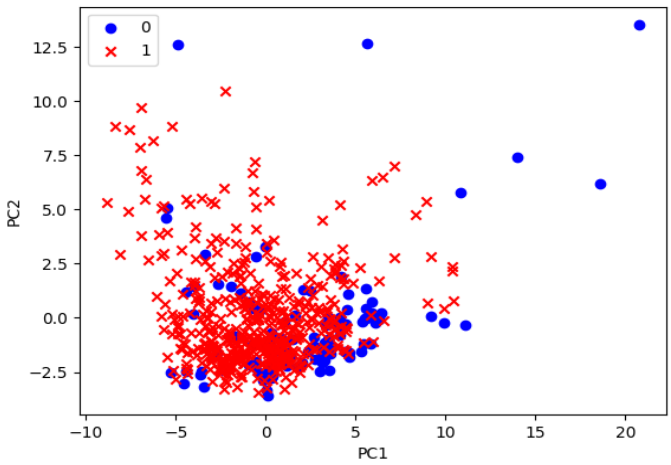

Figure 10: Scatter plot using engineered features of the training data

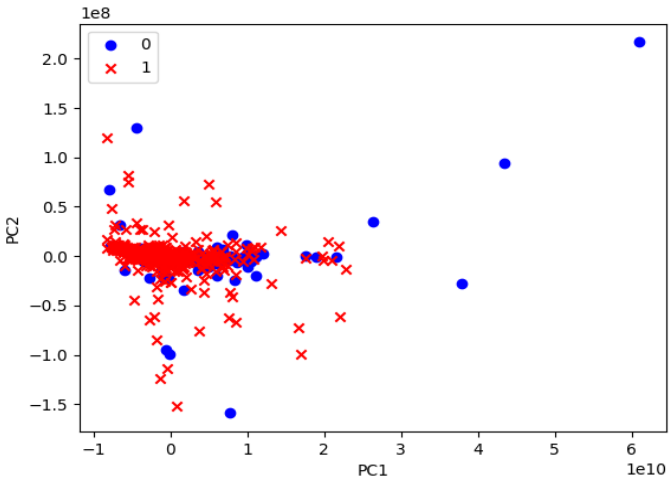

Figure 11: Scatter plot of the training data using all features

3.3 Experiments with K-NN

Code can be found in *KNN.py*, here I apply the KNN classifier to the under-sampled feature set obtained from the previous experiment, I vary the k value and observe the f1 results using Euclidean Distance in Figure 12.

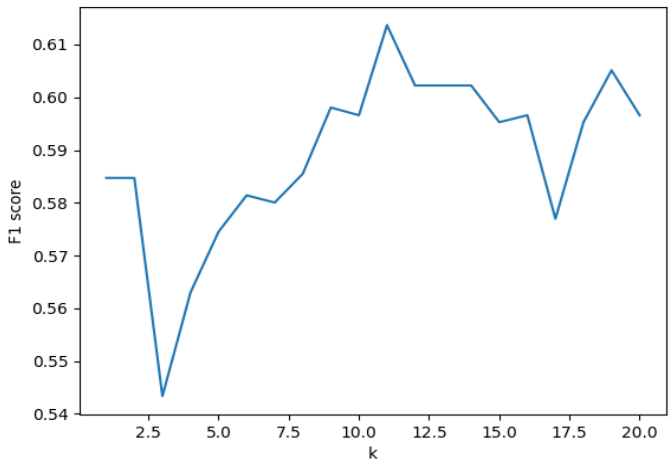

Figure 12: F1 scores against various k values for the K-NN classifier

The best value of k was 11 which yields the following results:

PPV	NPV	Sensitivity	Specificity	F1
0.632	0.639	0.608	0.5	0.613

This is better than the Bayes classifier but still it did not outperform the Decision Tree classifier, next I will try the SVM.

3.4 Experiments with SVM

Code is found in *SVM.py*, for this classifier I will try different kernel functions using the undersampled data. This classifier beats all previous classifiers for this task, results are in Figure 13.

And the following are the full results f or the Linear kernel:

PPV	NPV	Sensitivity	Specificity	F1
0.684	0.704	0.660	0.551	0.664

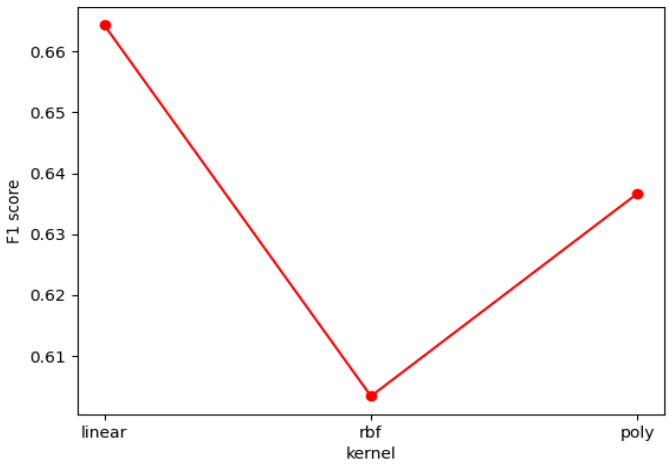

Figure 13: F1 scores against various kernels of SVM

4 Conclusion

From these expirements we find that the *Energy of signals*, Detrended Fluctuation Analysis, Recurrence Period Density Entropy, numPulses, and Jitter features are the most significant features that characterize the sound of a Parkinson's patient.

Patients, on average, tend to have higher Jitter, DFA, RPDE, and energy of signals while having lower number of signal pulses. These features can be used along with another combination of features to obtain an F1 score of 0.664 using an SVM classifier with a linear kernel. The problem, by nature, is complex as classes are naturally overlapping as we saw in figure 11, hence we might need to extract a different set of features in order to improve accuracy and reduce the error using the tested models.

References

[1] Cemal Sakar et al. "A comparative analysis of speech signal processing algorithms for Parkinson's disease classification and the use of the tunable Q-factor wavelet transform". In: *Appl. Soft Comput.* (2019), pp. 255–263.