Olaru Bogdan-Ioan

Project 1

The dataset used for this project can be found on *UC Irvine Machine Learning Repository* at this link. The information of dataset is gathered from 188 patients with Parkinson's Disease (107 men and 81 women) with ages ranging from 33 to 87 at the Department of Neurology in Cerrahpaşa Faculty of Medicine, Istanbul University. The control group consists of 64 healthy individuals (23 men and 41 women) with ages varying between 41 and 82.

Just from the above information we can observe that dataset isn't well balanced, but it should be enough. The dataset contains a lot of features: id, gender, Baseline features (Jitter variants, Shimmer Variants, Fundamental frequency parameters, Harmonicity parameters, Recurrence, Period Density Entropy, Detrended Fluctuation Analysis, Pitch Period Entropy), Time frequency features (Intensity Parameters, Format Frequencies, Bandwidth), Mel Frequency Cepstral Coefficients (MFCCs), Wavelet Transform based Features, Vocal fold features and Tunable Q-factor Wavelet Transform (TQWT) features. So this dataset already has some feature engineering which resulted in a lot of features, 755. We will start by scaling the dataset and removing the id column, then will have to keep only the most relevant features regarding to the labels.

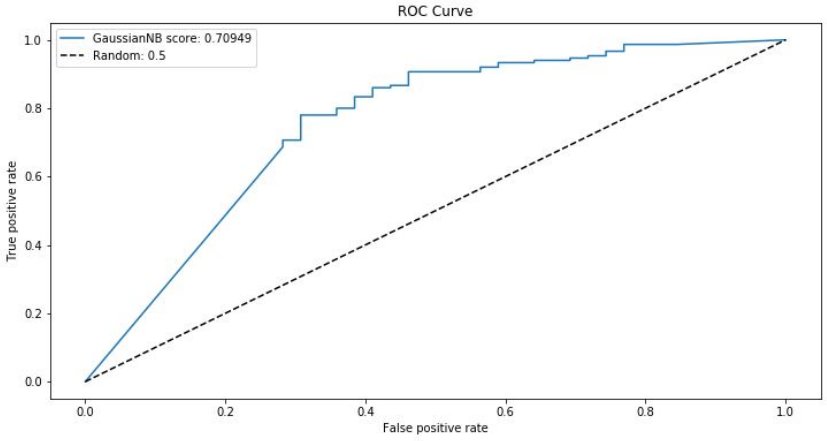
For this project I chose to apply 2 methods in order to predict a classification of data. Fist method I will use is Naive Bayes, and more exactly the Gaussian Naive Bayes because most of the data are continuous. Naive Bayes is a classification algorithm for binary and multi-class classification problems. It is called naive Bayes because the calculation of the probabilities for each hypothesis are simplified to make their calculation tractable. Rather than attempting to calculate the values of each attribute value P(A, B, C|X), they are assumed to be conditionally independent given the target value. This is a very strong assumption that is most unlikely in real data. Nevertheless, the approach performs surprisingly well on data where this assumption does not hold. Naive Bayes can be extended to continuous attributes, most commonly by assuming a Gaussian distribution. And the second method is Support Vector Machine (SVM). SVM is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: the algorithm creates a line or a hyperplane which separates the data into

classes. According to the SVM algorithm we find the points closest to the line from both the classes. These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane. In this case I will use an SVM with an 'rbf' kernel for a higher accuracy, with a regularization parameter of 150, because we want to avoid misclassification as much as possible, and a gamma of 0.1.

I will use Classification report, AUC Score, and ROC Curve to evaluate the models. For an initial evaluation I used all the features, and the results are:

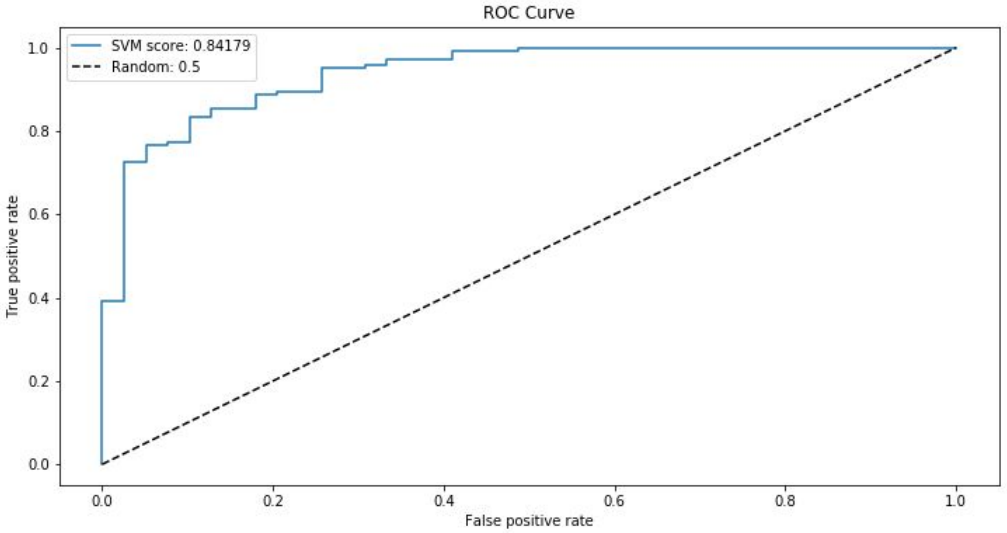
- for GaussianNB:

	precision	recall	f1-score	support
0.0	0.397059	0.692308	0.504673	39.000000
1.0	0.900826	0.726667	0.804428	150.000000
accuracy	0.719577	0.719577	0.719577	0.719577
macro avg	0.648943	0.709487	0.654550	189.000000
weighted avg	0.796874	0.719577	0.742574	189.000000



- for SVM:

	precision	recall	f1-score	support
0.0	0.763158	0.743590	0.753247	39.000000
1.0	0.933775	0.940000	0.936877	150.000000
accuracy	0.899471	0.899471	0.899471	0.899471
macro avg	0.848466	0.841795	0.845062	189.000000
weighted avg	0.898568	0.899471	0.898985	189.000000

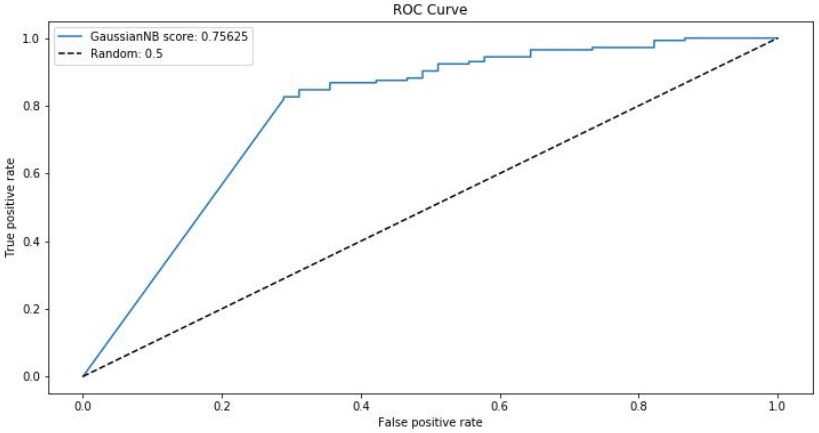


So, we can observe that SVM performs better, and the accuracy isn't so bad for both algorithms, but I think that the AUC Score can be better, so I will try to keep only some of the features that are more relevant for labels. In order to do that I'm using SelectKBeast from sklearn with f_regression as score function.

Let's see some results selecting the top 500 features:

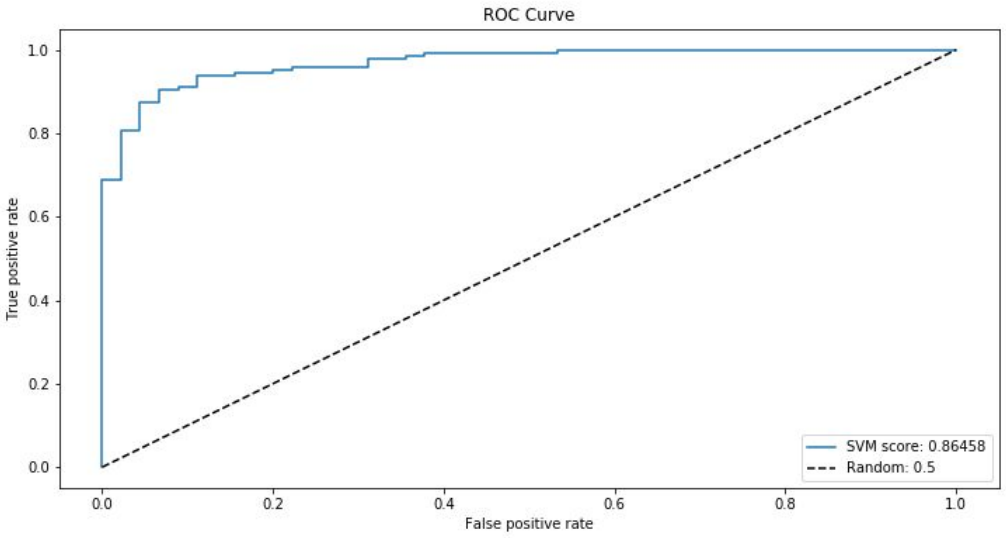
- for GaussianNB:

	precision	recall	f1-score	support
0.0	0.604167	0.644444	0.623656	45.000000
1.0	0.886525	0.868056	0.877193	144.000000
accuracy	0.814815	0.814815	0.814815	0.814815
macro avg	0.745346	0.756250	0.750424	189.000000
weighted avg	0.819297	0.814815	0.816827	189.000000



- for SVM:

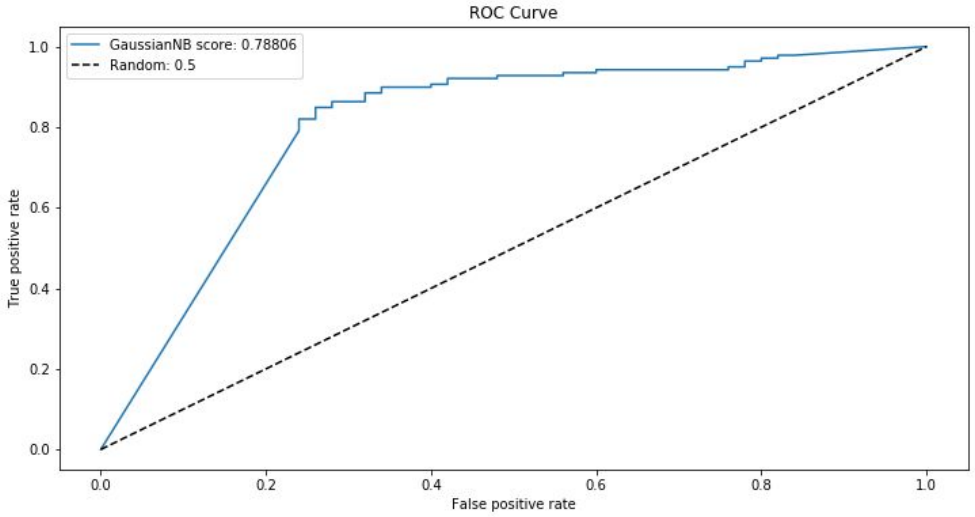
	precision	recall	f1-score	support
0.0	0.833333	0.777778	0.804598	45.000000
1.0	0.931973	0.951389	0.941581	144.000000
accuracy	0.910053	0.910053	0.910053	0.910053
macro avg	0.882653	0.864583	0.873089	189.000000
weighted avg	0.908487	0.910053	0.908966	189.000000



The results are improved for both algorithms. The most considerable improvement is for GaussianNB, but the SVM still performs better. I also tried with the best 600 features:

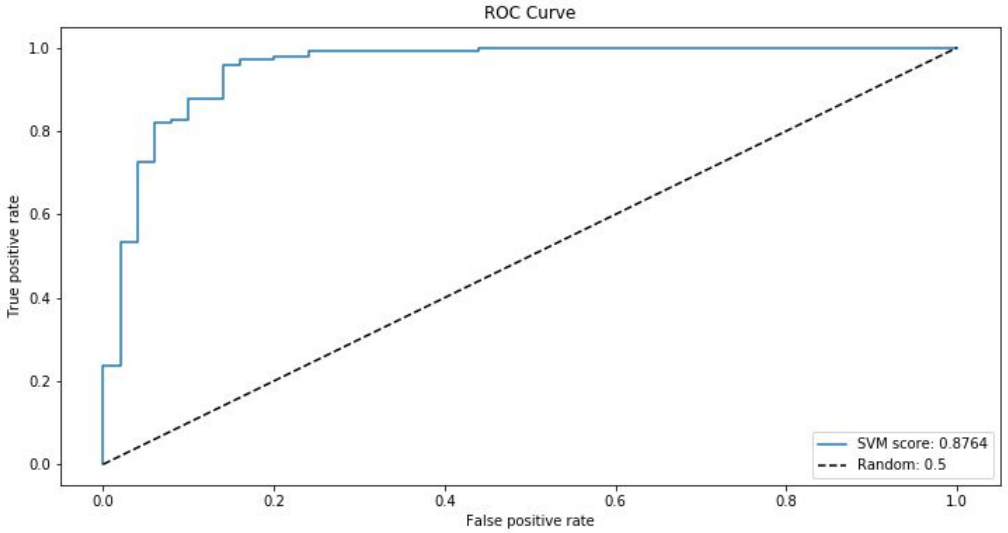
- for GaussianNB

	precision	recall	f1-score	support
0.0	0.642857	0.720000	0.679245	50.000000
1.0	0.894737	0.856115	0.875000	139.000000
accuracy	0.820106	0.820106	0.820106	0.820106
macro avg	0.768797	0.788058	0.777123	189.000000
weighted avg	0.828102	0.820106	0.823213	189.000000



- for SVM

	precision	recall	f1-score	support
0.0	0.974359	0.760000	0.853933	50.000000
1.0	0.920000	0.992806	0.955017	139.000000
accuracy	0.931217	0.931217	0.931217	0.931217
macro avg	0.947179	0.876403	0.904475	189.000000
weighted avg	0.934381	0.931217	0.928275	189.000000

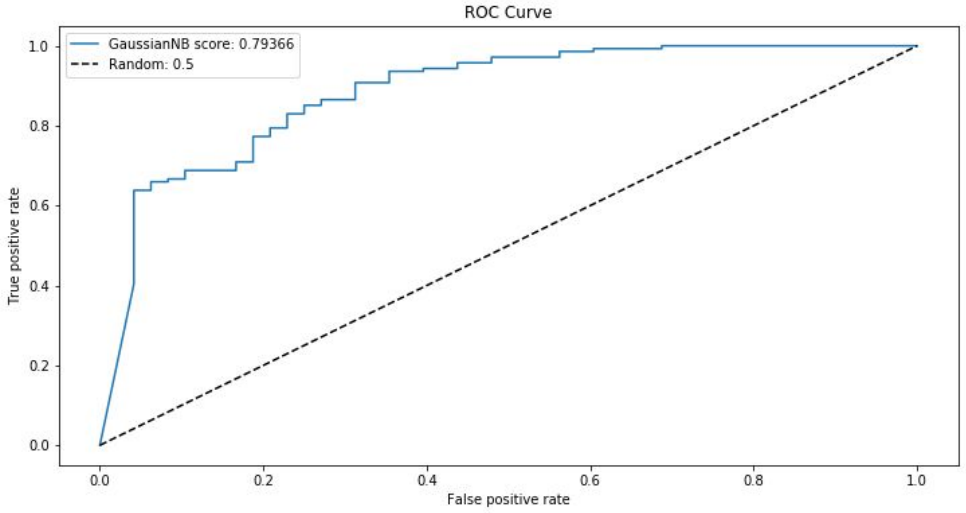


This one has a slight better accuracy for GaussianNB and a more considerable improvement for SVM. But both of them still misclassifies a lot of healthy people.

After checking with different numbers of features, I observed that the result obtained with 600 features was the best. Although, reducing too much the number of features doesn't affect the predictions as much as I expected. Here is an example for 50 features:

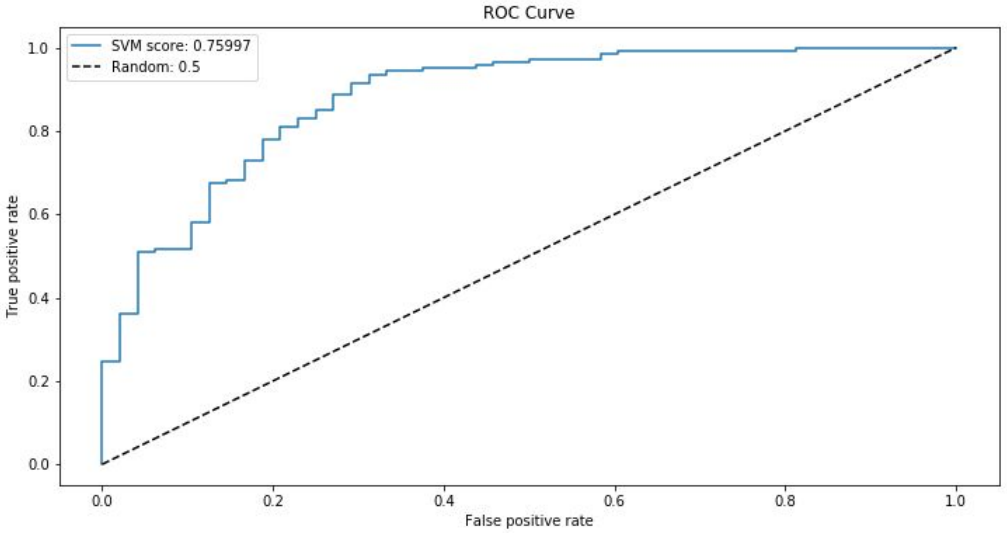
- for GaussianNB

	precision	recall	f1-score	support
0.0	0.636364	0.729167	0.679612	48.000000
1.0	0.902985	0.858156	0.880000	141.000000
accuracy	0.825397	0.825397	0.825397	0.825397
macro avg	0.769674	0.793661	0.779806	189.000000
weighted avg	0.835272	0.825397	0.829108	189.000000



- for SVM:

	precision	recall	f1-score	support
0.0	0.818182	0.562500	0.666667	48.000000
1.0	0.865385	0.957447	0.909091	141.000000
accuracy	0.857143	0.857143	0.857143	0.857143
macro avg	0.841783	0.759973	0.787879	189.000000
weighted avg	0.853397	0.857143	0.847523	189.000000



In conclusion, we observe that most of the features are important in classification of Parkinson's Disease, those features being obtained regarding medical importance. We can find most of the persons that have PD, we have a high rating of misclassifying healthy persons. Maybe that's because of the imbalanced data. I also think that this dataset needs more medical knowledge in order to have a better feature selection.