Machine Learning methods to predict the Unified Parkinson's Disease Rating Scale

Comp 598: Applied Machine Learning Final project

Andres Felipe Rincón 260584148

Ryan Razani 260631628

Nicolas Angelard-Gontier 260532513

***Abstract* - In this project, we implemented machine learning models to provide an assessment of the Unified Parkinson's Disease Rating Scale (UPDRS) from clinical tests on patients affected by the Parkinson disease. The Unified Parkinson's Disease Rating Scale (UPDRS) is often used to track Parkinson disease symptoms progression in patients. To provide an estimation for the UPDRS based on machine learning models, we implemented linear and polynomial regression, Support Vector Regression (SVR) and deep neural networks models. In particular, we used the Parkinson's Telemonitoring dataset [2], and applied data preprocessing and feature selection algorithms. Subsequently, we applied linear and polynomial regression as a baseline algorithm, followed by SVR and deep neural networks models to obtain better predictions for the motor UPDRS test variable (T1), and total UPDRS score test variable (T2). Using the Neural Network model, applying Ridge regularization, and 6-best feature selection from polynomial features of degree 2 added to 16 original features of dataset, we obtained a Mean-Squared-Error (MSE) of 0.004 UPDRS units for the motor UPDRS variable and 0.187 UPDRS units for the total UPDRS score.**

Github: https://github.com/NicolasAG/MachineLearning-project4

***Index Terms* - Machine Learning, unified Parkinson's disease rating scale (UPDRS). Mean-Squared-Error (MSE), Linear Regression, Polynomial Regression, Support Vector Regression (SVR), Neural Networks.**

1. Introduction

Clinical tests are essential for the treatment of patients susceptible of developing the Parkinson's disease or affected by the Parkinson's disease. Currently, multiple clinical tests exist for patients with the Parkinson's disease. Nevertheless, as addressed in [1], clinical examinations require multiple clinical visits for the patients and these examination are time consuming for the medical staff. Therefore, a remote and autonomous clinical test represents a good alternative for improving clinical tests time efficiency and to provide Parkinson's disease clinical tests at a larger scale. Some research has addressed this issue in a less explicit way. Indeed, using voice recordings from healthy patients and patients affected with the Parkinson's disease, machine learning classifiers were applied in [3] to classify patients with Dysphonia, a symptom of the Parkinson's disease. In addition, it was shown in [4] that a bootstrapped classifier could be applied for detecting voice disorder on voice recordings from healthy patients and patients affected by the Parkinson's disease.

In this paper, we apply linear and polynomial regression, support vector regression, and deep neural networks models to the Parkinson's Telemonitoring dataset [2]. The dataset has 22 attributes and consists of 5875 voice recordings from 42 individuals with early-stage Parkinson's disease [2]. In more detail, as described in [2], the features of this dataset consist on the subject age (Integer value); subject gender: '0' for male and '1' for female; test time: Time since recruitment into the trial; motor_UPDRS: Clinician's motor UPDRS score; total_UPDRS: Clinician's total UPDRS score; Jitter(%), Jitter(Abs), Jitter: RAP, Jitter: PPQ5, Jitter: DDP: measures of variation in fundamental frequency; Shimmer, Shimmer(dB), Shimmer: APQ3, Shimmer: APQ5, Shimmer:APQ11, Shimmer: DDA: measures of variation in amplitude; NHR and HNR: measures of ratio of noise to tonal components in the voice; RPDE: a nonlinear dynamical complexity measure; DFA: signal fractal scaling exponent; PPE: a nonlinear measure of fundamental frequency variation. In this project, we predicted the total_UPDRS score value and the motor UPDRS score using linear and polynomial regression, SVR and deep neural networks models. Note that, the UPDRS serves to map Parkinson's disease symptoms severity in patients [1]. In theory, the total UPDRS score range is 0-176 and the motor UPDRS score range is 0-108 [1], however in our data set, we found the following statistical measures:

Table 1: Dataset statistics

Data set	MOTOR UPDRS	TOTAL UPDRS
Min	5.0377	7
Max	39.511	54.992
Range	34.4733	47.992
Mean	20.871	27.576
Std.	8.12858964	10.6993726
Variance	66.0739696	114.476573

2. Methodology

2.1 Pre-processing, Feature design and Selection.

The only pre-processing that was applied was the standardization of the data in the case of the neural networks algorithm. Indeed, for each feature in the dataset, we removed the mean and scaled to unit variance as following:

$$x_{ij}^{New} = \frac{x_{ij}^{Old} - \mu_j}{\delta_j} \qquad (2.0)$$

Where, μ_j and δ_j are the average and standard deviation value respectively for feature j. Centering and scaling processing on each feature is done by Scikit-learn [12].

In terms of feature design and feature selection, we created new features using polynomial feature creation from the sklearn library [8], and selected a few features using two methods: univariate feature selection, and Recursive Feature Elimination (RFE).

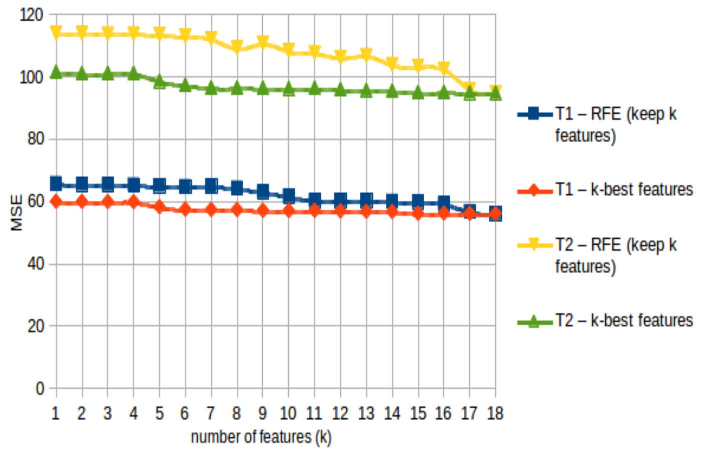
a) Feature Construction:

The polynomial feature creation allow us not only to see the effect of individual features, but also to see the effect of combinations of different original features on the target. For our baseline algorithm we constructed polynomials of degree 2, 3 and 4. In particular we build 210 features from the original 19 features for polynomial regression of degree 2, 1540 features for polynomial regression of degree 3, and 8855 features for polynomial regression of degree 4. These features were built by using combinations of the original features in the dataset. In addition, for the SVR algorithms we only build features of polynomial 2 and 3 as the MSE improvement was not worth the running time increase. Eventually, in Neural Networks (NN), the first 4 columns of the original data set, namely: "subject", "age", "sex", and "test_time" were not considered in the process as the remaining 16 features resulted in better prediction. After extending the train set to polynomial features of degree 2, 3, 4, 5 and 6; K best features were selected.

b) Feature Selection:

In terms of feature selection, the first method consisted on selecting the k highest scoring features, using univariate linear regression tests taken from the machine learning library sklearn [5]. These univariate linear regression tests consisted on computing F-scores from correlation of the input features with respect to the predicted variables [6]: motor UPDRS score and total UPDRS score. In addition, recursive feature elimination (RFE) was tested with k values from 2 up to 19. This algorithm performs a greedy search in the feature space: starting with all features, it removes one feature at a time until it reaches k features. At each step, the algorithm removes the feature that increased most the MSE. A comparison of these two methods has been made for both motor UPDRS (T1) and total UPDRS (T2) as we can see in Figure 1 below.

Figure 1: Linear regression with different feature selection algorithms



We noticed that for any k value, the univariate k-best method produces a lower MSE so we continued our project with this selected method.

2.2 Algorithms

2.2.1 Linear and Polynomial regression

In this project, we implemented linear and polynomial regression as baseline algorithms. The linear regression algorithm implemented is represented by the following equation: $Y = wX$ where w corresponds to the weight vector, X corresponds to the feature matrix, and Y corresponds to the target vector. For polynomial regression of degrees 2, 3 and 4 we used a different X matrix with the same amount of examples, but with more features. In particular, X corresponds to the combination of the 19 original features, and is of the following form:

$[1, x_1, \dots, x_{19}, x_1{}^2, x_1x_2, \dots, x_1x_{19}, x_2{}^2, x_2x_3, \dots, x_2x_{19}, \dots, x_{19}{}^d]$

In addition, in order to avoid overfitting, we had to use some kind of regularization for the polynomial features. We recorded the resulting MSE for no regularization at all, Lasso regularization, and Ridge regularization. The different cost functions are the following [10]:

No regularization:

$$min_w \, ||X_w - y||_2{}^2 \qquad (2.1)$$

Lasso regularization:

$$min_w \, \frac{1}{2n_{samples}} ||X_w - y||_2{}^2 + \alpha||w||_1 \qquad (2.2)$$

Ridge regularization:

$$min_w \, ||X_w - y||_2{}^2 \, + \, \alpha||w||_2{}^2 \qquad (2.3)$$

The results of these different cases are presented in the section 3.1.1.

In order to choose the hyper-parameter α that measures how much weight to put on the weight size rather than the actual squared error loss, we used 5-fold cross-validation. This consisted on partitioning 80% of the dataset randomly into 5 subsets followed by cross validation iterations. At each iteration, we performed validation on one retained subset while training on the remaining 4 subsets.

2.2.2 Support Vector Regression (SVR)

Moreover, since the variables motor UPDRS (T1) and total UPDRS (T2) are continuous, we used a regression version of SVM, implemented in the sklearn machine learning library, called SVR. Like in regular SVM, the SVR algorithm tries to minimize the weight vector $||w||$. Furthermore, this method tries to learn a function $f(x)$ that is at most ϵ units away from the actual output y. Thus, we don't care for points that do not exactly lie on the learned function, as long as they are at a distance less than or equal to ϵ [9]. This allow us to be slightly more tolerant in our predictions. Eventually, for points that are further than ϵ we introduce new variables (ξ_i and $\xi_i{}^*$) that measure the distance from ε that the point is [9]. Thus, these kind of points are $\epsilon + \xi_i$ away from the actual output y_i and we try to get as few as possible of them. The amount of these 'incorrect' points is controlled by a constant C. This is like performing a regularization task as previously described in polynomial regression. As a whole, the SVR algorithm for linear regression consists on solving the following optimization problem [9]:

$$min\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{k}(\xi_i + \xi_i{}^*) \qquad (2.4)$$

Subject to:
$$< w, x_i > +b - y_i \leq \epsilon + \xi_i{}^*$$
$$-< w, x_i > -b + y_i \leq \epsilon + \xi_i$$
$$\xi_i \xi_i{}^* \geq 0$$

Where w is the weight vector, C determines the flatness of the prediction curve, ξ_i is the distance that point i is from the tolerated margin of error: ϵ, and $< w, x_i > +b$ is the prediction made for point x_i based on w and b.

Eventually, like in regular SVM, to allow for non-linear learning functions we put this optimization problem in its dual form by using the Lagrangian multipliers, and perform the "Kernel trick", that maps points to another dimension. The kernel used in this study is the Radial Basis Function (RBF): $e^{(-\gamma|x-x'|^2)}$ from the sklearn library [7]. Therefore, two parameters were estimated to apply SVR: γ and C. As mentioned above, the SVM parameter C represents the trade off between, bias and variance: a high C means that we try to learn correctly most of the training points (so lot of support vectors, high variance and low bias) while a low C aims at building a smoother learning function (few support vectors, low variance and high bias) [7]. In addition, the kernel parameter gamma represents the influence of a single training point on others: a large γ means that the influence is small, so the distance $|x - x'|$ has to be small for x' to be affected by x, while a small γ means that the influence on a single training point is large [7]. The estimation of these two parameters are discussed in section 3.1.2.

2.2.3 Deep Neural Network

We also implemented deep neural networks (DNN) trained by backpropagation with mean square error cost function. It is expected from DNN to have better performance than baseline algorithms since it allows to learn complex/nonlinear function of input data.

For the task of regression, each node in the output layer is considered as a neuron with a linear activation function to predict continuous values of motor UPDRS (T1) and total UPDRS (T2). The activation function of hidden layers considered to be linear and nonlinear function which will be discussed in more details in section 3.1.3. Among several transfer functions, sigmoid function; (x), and linear function turned out to be more effective in our prediction problem. The equation of these activation functions are as following:

$$\text{Sigmoid function: } \sigma(x) = \frac{1}{1+\exp(-x)} \qquad (2.5)$$

$$\text{Linear function: } Y(WX + b) = WX + b \qquad (2.6)$$

In our feedforward neural network structure each weight is learnt by stochastic gradient descent on the error. Mean Square Error (MSE) of the network output is calculated as:

$$E = \frac{1}{2}\sum_{k\in K}(O_k - O_t)^2 = \frac{1}{2}\left\|\mathrm{O_k} - \mathrm{O_t}\right\|^2 \qquad (2.7)$$

Lasso regularization:

$$min_w\ \frac{1}{2}||O_k - O_t||_2^2 + \alpha||w||_1 \qquad (2.8)$$

Ridge regularization:

$$min_w\ \frac{1}{2}\ ||O_k - O_t||_2{}^2\ +\ \alpha||w||_2{}^2 \qquad (2.9)$$

Where, K is the number of nodes at the output layer. Ok is the prediction value at node k from output layer and Ot is the actual UPDRS target value corresponding to the same node in the output layer.

After predicting the target values the error will be calculated and propagates back through the network to adjust the weights in the direction that reduces it.

$$\Delta w_{ij}^{(\iota)} = -\eta\frac{\partial E}{\partial w_{ji}} \qquad (2.10)$$

$$w_{ij}^{(\iota)}(t+1) = w_{ij}^{(\iota)}(t) + \Delta w_{ij}^{(\iota)}(t) \qquad (2.11)$$

This procedure known as Backpropagation (BP) is most widely used algorithm for supervised learning of neural networks [11]. Its idea is based on repeated application of the chain rule to compute the influence of each weight in the network with respect to an error function.

3. Testing and Validation

In order to perform the testing and validation of the algorithms, we partitioned the dataset into two sets: a training and validation set containing 80% of the examples of the dataset, and a test set containing 20% of the examples of the dataset. For each algorithm, we used the training and validation set to estimate the parameters and perform feature selection.

3.1 Parameter Estimation and feature reduction

This section shows the parameter estimation for the algorithms implemented.

3.1.1 Linear and Polynomial regression

To pick the best α regularization hyper parameter and the optimal number of features of the linear and polynomial regression algorithms, we performed 5-fold cross validation on 80% of the dataset (randomly selected). We tested the following α rates for polynomial regression with both ridge and lasso regularization: 1e-10, 1e-9, 1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1.0, 1e+1. After picking the best α rate for each cases we measured the MSE (without any feature selection yet) as described in *Table 2* below:

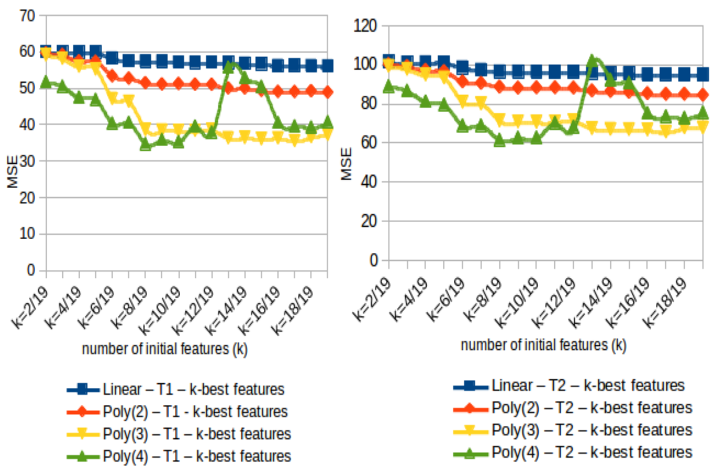
Table 2: MSE values for different polynomial regression and regularization

5-FOLD CROSS VALID.	No regularization	Lasso regularization	Ridge regularization
Linear regression 19 features	56.227, 94.992	56.100, 94.679 $\alpha = 10^{-6}$	56.452, 95.054 $\alpha = 10^{-7}$
Polynomial regression degree 2 ; 210 features	62.773, 94.738	50.042, 86.797 $\alpha = 10^{-4}$	48.955, 84.656 $\alpha = 10^{-3}$
Polynomial regression degree 3 ; 1540 features	19521.097, 46560.070	42.087, 74.963 $\alpha = 10^{-4}$	37.326, 68.000 $\alpha = 10^{-1}$
Polynomial regression degree 4 ; 8855 features	N/A	N/A	40.865, 75.738 $\alpha = 10^{-1}$

The polynomial of degree 4 was only tested on the ridge regularization because without any regularization we suspected the MSE to be ridiculously high (because of overfitting, like in the degree 3 case) and with the lasso regularization, the running time to pick the best α rate was too long. Moreover, we can see that even with regularization, degree 4 features start to overfit because of the huge number of features created. Overall we noticed that Ridge regularization was giving us slightly better results so we decided to continue our work with this technique.

In terms of feature selection, we tested k best features with k from 2 up to 19, among 19 attributes from the original dataset. In the case of polynomial regression, we did the same thing and constructed our polynomial features based on the top k original features. It is true that this method may have missed a few 'ideal' feature combinations, but we decided to do that for our baseline algorithm in order to better compare the MSE for our predictions on the motor UPDRS (T1) and the total UPDRS (T2) in one graph as we can see in *Figure 2.*

Figure 2: MSE of linear and polynomial regression using k best feature selection and Ridge regularization



The above plot shows that we reach the best MSE with polynomial features of degree 4, with 8-best features, giving MSE (motor UPDRS) ~ 35 and MSE (total UPDRS) ~ 60.

3.1.2 SVR

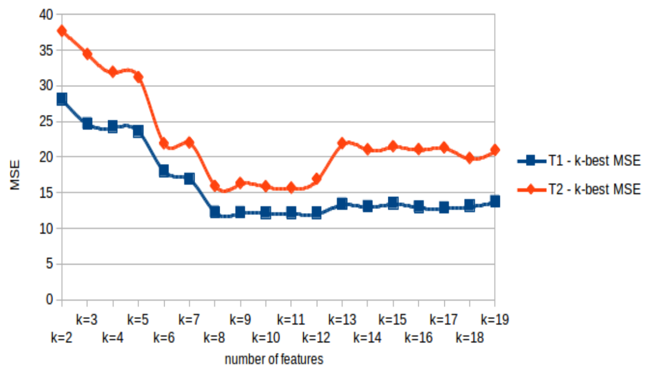
For the parameter estimation of SVR with the Radial Basis Function (RBF) kernel, we performed 5-fold cross validation on 80% of the dataset (randomly selected) and we tested the following values for the C parameter: 0.01; 0.1; 1; 10; 100; 1,000; 10,000; 100,000; and 1,000,000. For the γ parameter, we tested the following values: 0.01, 0.1, 1, 10 and 100.

Table 3: Estimation of the C and γparameters of SVR (with linear features) using the MSE from the motor UPDRS (T1) and total UPDRS (T2)

MSE(T1), MSE(T2)	C=0.01	C=0.1	C=1.0	C=10	C=100	C=1,000	C=1e4	C=1e5	C=1e6
ɣ=0.01	64.870, 114.902	58.859, 106.068	53.895, 93.120	51.042, 86.425	47.237, 75.896	39.681, 57.321	35.655, 80.294	N/A	N/A
ɣ=0.01	65.965, 116.184	63.137, 112.524	49.023, 85.305	27.715, 34.634	16.465, 25.266	13.936, 21.103	14.092, 21.450	13.907, 21.954	15.324, 22.266
ɣ=1.0	66.166, 116.385	64.901, 114.831	53.933, 102.367	23.013, 76.483	21.512, 72.489	20.931, 72.084	21.036, 74.004	20.746, 71.917	20.974, 72.1711
ɣ=10	66.403, 116.718	65.862, 116.063	62.511, 114.163	46.391, 112.642	45.341, 112.318	45.024, 112.297	44.566, 112.279	44.768, 112.000	45.497, 112.282
ɣ=100	66.278, 116.549	66.339, 116.600	65.768, 116.350	62.584 114.708	62.289, 114.518	62.394, 114.495	62.106, 114.525	62.411, 114.503	62.111, 114.477

We first tried the algorithm on the original 19 features, by performing multiple trials with combinations of the parameters mentioned above. Such results can be found in the *Table 3*. We found that the best value for the C parameter was 1000, and 0.1 for the parameter. After fixing those parameters for our linear feature case, we performed the k-best feature algorithm with different k from 2 to 19, and decided which k will give the lowest MSE based on 5 fold cross validation as we can see in Figure 3.

Figure 3: k-best features from linear features with SVR (RBF kernel, C=1000, γ= 0.1)



By doing so, we found that the lowest MSE was performed with the best 8 features, giving an overall MSE of 12.326 for motor UPDRS and 15.975 for total UPDRS.

Furthermore, we performed the same parameter selection technique for polynomial features of degree 2 and 3. The MSE results for degree 2 features can be found in *Table 4* below.

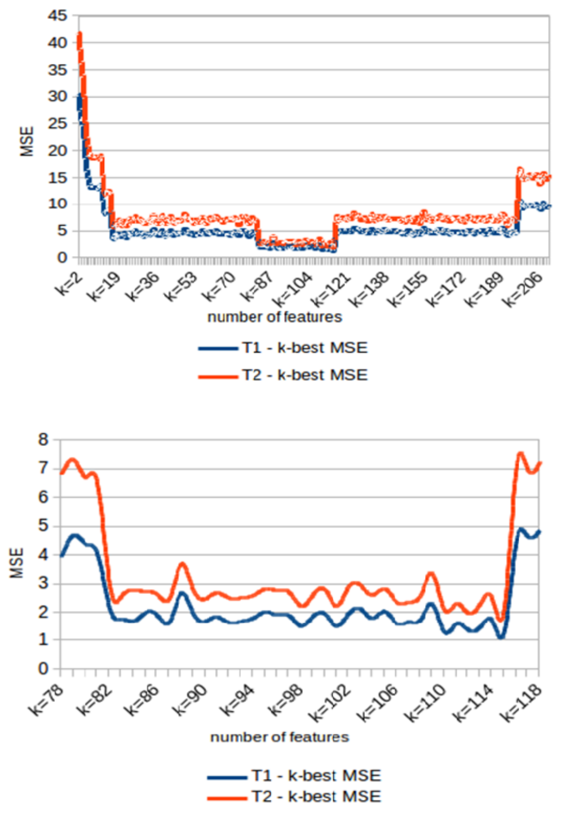
Table 4: Estimation of the C and γparameters of SVR with polynomial features of degree 2

MSE(T1), MSE(T2)	C=100	C=1,000	C=10,000	C=100,000
ɤ=0.000001	32.349, 57.850	21.450, 39.209	>1h	too long
ɤ=0.00001	10.023, 16.084	8.560, 13.259	9.708, 15.322	9.296 14.278
ɤ=0.0001		24.914, 41.533	24.772, 41.214	24.986, 42.065
ɤ=0.001		48.827, 85.073	48.897, 85.003	47.871, 83.424
ɤ=0.01			62.908, 110.732	63.068, 110.657

From this table we can see that in this case the best values for C and γ are 1000 and 1e-5 respectively. However, we noticed that the best MSE (without any feature selection) for polynomial features of degree 3 was not better than for polynomial features of degree 2. This is shown in Table 9 of the appendix. In addition, the running time was greatly increased with degree 3 features, so we decided not to perform feature selection for this case. On the other hand, for polynomial features of degree 2, we greatly reduced our MSE by performing 5-fold cross validation with k-best feature selection: k from 2 to 210 as we can see in *Figure 4*.

From this figure, we can see that the best MSE is found with 81 < k < 116, this is why we zoomed on this region to better see that the best MSE is obtained by choosing the best 115 features from the 210 polynomial features. The resulting mean squared errors are 1.329 for motor UPDRS and 2.156 for total UPDRS.

Figure 4: k-best features from polynomial features of degree 2 with SVR (RBF kernel, C=1000, γ= 1e-5)



3.1.3 Neural Networks

When using neural networks, we excluded 3 attributes from the dataset: the age, the sex of the individuals and the duration of the clinical tests (test_time). Moreover, we selected the best features of data set for the neural networks algorithm based on k highest score of univariate linear regression test.

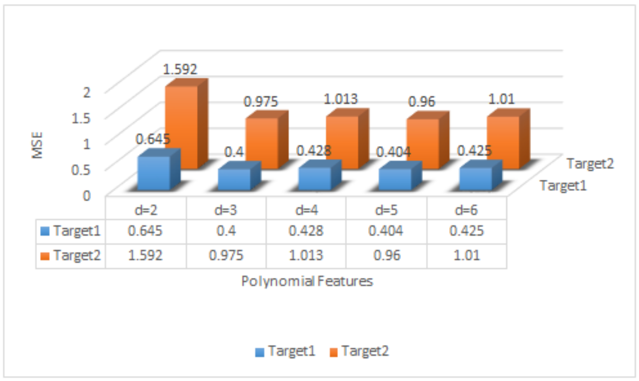
To identify the best hyperparameters for feed forward neural network, 5-fold cross validation is applied to the training set which enabled us to save time and achieve very low MSE values. The MSE results of validation set of Shallow Neural Network (SNN) and DNN using Ridge or Lasso regularization were compared in Table 8 presented in Appendix. It is shown that DNN performs much better due to its advantage of learning nonlinear functions. The sigmoid transfer function and linear transfer function are referred to 'Sig' and 'Lin', respectively. Target 1, and 2 were predicted at the same time with same NN architecture, hence the output layer consists of 2 nodes.

It is observed that the number of hidden units is an important hyper-parameter to consider since high number units in hidden layers can cause overfitting. In contrast, just few neurons in the hidden layer does not allow the network to train well and capture nonlinear/complex functions of inputs and the system will be less robust to noise.

In the case of NN, we primarily started the preprocessing by keeping k best features first and creating the polynomial features of degree d on the selected features as shown in Table 5 of Appendix. However, we noticed that by changing the order we could improve the performance as shown in Table 6, 8 and figure 5. Hence, we generated a new feature matrix consisting of all polynomial combinations of the features with degree d [8], and chose the K best features as a train set. For most cases, it is shown the MSE values with L2 (Ridge) regularization is slightly better than L1 (Lasso).

In Figure 5, the best NN structure from Table 5 was considered. Several possible subsets of polynomial features were created, and the performance of the best k features was measured on the validation set.

Figure 5: Effect of feature selection and polynomial features on MSE



Finally, the best ten MSE results of validation set of DNN using 5-fold cross validation is shown in Table 6. This table shows different choice of hyper parameters such as hidden layer/hidden nodes, learning rate and ridge regularization constant. These results were obtained when 6 best features were selected after polynomial features of degree 2 were extended to raw data. It implies that the best DNN architecture for our regression problem is a network with 2 hidden layers of 400 sigmoid units followed by 5 linear units.

The optimization technique in implementing neural network is based on stochastic gradient descent, which with appropriate choice of hyper parameters the running time can be reduced. Mini-batch size and number of training iteration play key role as hyperparameter in training time. To achieve the above results, we used 30 mini batch and 10 iteration. Larger mini batch size yield in faster computation but consumes more examples to reach to the same error than with smaller mini batch size. Optimum number of training can be obtained by observing the training and validation behaviour and set the number of iteration to the point that validation error starts to increase.

Table 6: preliminary MSE results on validation set for Target1 and Target 2 predictions

Sorted Based on T1&T2 MSE				
Layer	Learning rate	Alpha	MSE: T1	MSE: T2
[400, 5]	0.0009	0.9	0.148	0.543
[400, 5]	0.0009	0.8	0.15	0.557
[400, 6]	0.0009	0.01	0.15	0.554
[400, 6]	0.0009	0.1	0.151	0.564
[400, 5]	0.0009	0.7	0.153	0.575
[400, 6]	0.0009	0.7	0.153	0.572
[400, 6]	0.0009	0.8	0.154	0.58
[400, 5]	0.0007	0.9	0.154	0.546
[400, 6]	0.0009	0.9	0.155	0.587
[400, 6]	0.0007	0.01	0.156	0.559

3.2 Experimental results

In this section, we show the prediction results in the test set for linear and polynomial regression, SVR and neural networks. These results were obtained by training our algorithms on 80% of the data (what we used for parameter selection in previous sections), and predicting on 20% of the data (the test set). The following *Table 7* summarizes these results for each of the algorithms that were implemented.

Table 7: Test set results

Algorithms	MSE motor UPDRS variable	MSE total UPDRS variable
Linear Regression, 19-best features	54.954	93.257
Polynomial regression of degree 2, ridge α=1e-3, 19-best features	47.052	82.856
Polynomial regression, degree 3, ridge α=1e-1, 17-best features	34.604	64.52
Polynomial regression, degree 4, ridge α=1e-1, 8-best features	33.314	60.624
SVR, C = 1000, ɣ=1e-1, Linear features, 8-best features	11.266	13.475
SVR, C=1000, ɣ=1e-5, polynomial features of degree 2, 115-best features	0.829	1.563
NN, learning rate=0.0009, L2	0.004	0.187

We can conclude this project by saying that the best algorithm found is Neural Networks (NN), which has an MSE of 0.004 for the motor UPDRS variable and an MSE of 0.187 for the total UPDRS variable. The result is obtained where the polynomial features of degree 2 are added to raw dataset and 6 best features are elected. The structure of NN was shown to be optimized by having 2 hidden layers of 400 and 5 neurons respectively.

4. Discussion

In this paper, we presented three different regression based algorithms to predict Unified Parkinson's Disease Rating Scale along with their mean square error performance and choice of hyperparameters. We were able to achieve a mean squared error (MSE) of 0.004 for predicting motor UPDRS and 0.187 for predicting total UPDRS. In comparison with previous and accurate studies, our approach improved significantly the prediction of the motor and total UPDRs variables. Indeed, in [1] the predicted results were about 7.5 UPDRS units from the clinician's tests. An idea that should be further studied would be to see if learning the motor UPDRS score first, making a prediction about it, and then learning the total UPDRS score with our predictions of motor UPDRS would increase our prediction accuracy for the total UPDRS. We tried to implement such idea in our baseline algorithm by adding a new column to our X matrix: "*predicted motor UPDRS"*, but we didn't notice much difference on the MSE of total UPDRS. Thus, we believe that a good future work for this dataset would be to study in more details how those two features are related.

We hereby state that all the work presented in this report is that of the authors.

Statement of Contribution:
Andres Felipe Rincon Gamboa: participated on making the report. Nicolas Angelard-Gontier: worked on the implementation of the Linear and Polynomial Regression, SVR and the corresponding sections in the report. Ryan Razani: implemented Neural Network and wrote the corresponding sections of the report.

5. References

[1] A Tsanas, MA Little, PE McSharry, LO Ramig (2009)
'Accurate telemonitoring of Parkinson's disease progression by non-invasive speech tests',
IEEE Transactions on Biomedical Engineering (to appear).

[2] Archive.ics.uci.edu, 'UCI Machine Learning Repository: Parkinsons Telemonitoring Data Set', 2015. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring. [Accessed: 01- Dec- 2015].

[3] Little MA, McSharry PE, Hunter EJ, Ramig LO (2009), "*Suitability of dysphonia measurements for telemonitoring of Parkinson's disease*", IEEE Transactions on Biomedical Engineering, 56(4):1015-1022.

[4] Little MA, McSharry PE, Roberts SJ, Costello DAE, Moroz IM. "*Exploiting Nonlinear Recurrence and Fractal Scaling Properties for Voice Disorder Detection*", BioMedical Engineering OnLine 2007, 6:23 (26 June 2007).

[5] Scikit-learn.org, '1.13. Feature-selection-scikit-learn-0.17-documentation', 2015. [Online]. Available: http://scikit-learn.org/stable/modules/feature_selection.html#univariate-feature-selection. [Accessed: 02- Dec- 2015].

[6] Scikit-learn.org, 'sklearn.feature_selection.f_regression-scikit-learn-0.17-documentation', 2015. [Online].Available:http://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.f_regression.html#sklearn.feature_selection.f_regression. [Accessed: 04- Dec- 2015].

[7] Scikit-learn.org, '1.4. Support Vector Machines — scikit-learn 0.17 documentation', 2015. [Online]. Available: http://scikit-learn.org/stable/modules/svm.html#svr. [Accessed: 05- Dec- 2015].

[8] Scikit-learn.org, 'sklearn.preprocessing.PolynomialFeatures-scikit-learn-0.17-documentation', 2015.[Online].Available:http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html. [Accessed: 12- Dec- 2015].

[9] Alex J Smola, Bernhard Scholkopf (1998), "A Tutorial on Support Vector Regression", NeuroCOLT2 Technical Report Series, NC2-TR

[10] Scikit-learn.org, "1.1. Generalized Linear Models — scikit-learn 0.17 documentation", 2015. [Online]. Available: http://scikit-learn.org/stable/modules/linear_model.html#ridge-regression. [Accessed: 12- Dec- 2015].

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error-propagation. In Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1, volume 1, pages 318-362. MIT Press, Cambridge, MA, 1986.

[12] Scikitlearn.org,. 'Scikitlearn.Org'. N.p., 2015. Web. 10 Nov. 2015.
http://scikitlearn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler.transform

Appendix:

Table 5: preliminary MSE results on validation set for Target1 and Target 2 predictions

n-fo1d cross validation	Polynomial Feature(d)	Hidden Structure	Ridge (L2)	MSE(L2) [target1, target2]	Lasso(L1)	MSE(L1) [target1, target2]	Learning rate	Batch size	Iterations	Features all=16
5	1	[‘sig’:450]	0.7	4.84, 12.16	0.7	4.87, 11.89	0.01	30	10	all
5	1	[‘sig’:500]	0.7	6.14, 15.14	0.7	6.34, 15.69	0.01	30	10	all
5	1	[‘sig’:500]	0.7	17.12, 26.56	0.7	17.52, 27.03	0.001	30	10	all
5	1	[‘sig’:450,’Lin’:5]	0.6	4.35, 8.74	0.6	4.42, 9.02	0.0009	30	10	all
5	1	[‘sig’:450,’Lin’:6]	0.6	3.18, 9	0.6	3.18, 9	0.01	30	10	all
5	2	[‘sig’:20]	0.7	20.15, 37.15	0.7	20.84, 38.10	0.01	30	10	all
5	2	[‘sig’:20]	0.7	17.98, 35.21	0.7	18.08, 35.69	0.2	30	10	all
10	1	[‘sig’:450]	0.01	13.59, 25.97	0.01	13.9, 26.03	0.0009	30	10	all
10	1	[‘sig’:450, ’Lin’:6]	0.01	2.99, 8.02	0.01	2.99, 8.02	0.0009	30	10	all
15	1	[‘sig’:450, ’Lin’:6]	0.01	2.09, 5.86	0.01	2.19, 6.06	0.0009	30	10	all
11	1	[‘sig’:485, ’Lin’:6]	0.01	0.96, 2.55	0.01	0.96, 2.55	0.0009	30	10	10
10	1	[‘sig’:485, ’Lin’:6]	0.01	1.31,2.85	0.01	1.31,2.85	0.0009	30	10	9
10	1	[‘sig’:400, ’Lin’:6]	0.01	0.42, 1.5	0.01	0.42, 1.5	0.0009	30	10	9

The best validation MSE score of the same NN structure based on selecting the best k features from polynomial feature of degree d added to 16 features of dataset.

Table 8: MSE performance for feature selection and polynomial features of degree 2,3,4,5, and 6

Polynomial feature	MSE	Best Features																
		5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	all
d=2	+ Ridge	1.077, 1.065	0.707, 1.147	0.657, 1.376	0.645, 1.592	0.677, 1.788	0.698, 1.994	0.785, 2.296	0.83, 2.601	0.9, 2.908	0.988, 3.2190	1.085, 3.587	1.208, 3.962	1.413, 4.506	1.671, 5.163	2.013, 5.888	2.221, 6.390	26.609, 50.622
	+ Lasso	1.078, 1.065	0.708, 1.147	0.657, 1.376	0.645, 1.592	0.677, 1.788	0.698, 1.993	0.785, 2.296	0.83, 2.600	0.9, 2.908	0.987, 3.218	1.085, 3.58	1.208, 3.961	1.413, 4.505	1.67, 5.163	2.012, 5.887	2.22, 6.390	26.616, 50.634
d=3	+ Ridge	0.516, 0.804	0.4, 0.975	0.444, 1.245	0.491, 1.513	0.556, 1.706	0.597, 1.925	0.683, 2.2	0.736, 2.521	0.808, 2.827	0.891, 3.154	0.989, 3.529	1.106, 3.868	1.313, 4.406	1.572, 5.045	1.928, 5.784	2.136, 6.278	75.517, 141.797
	+ Lasso	0.516, 0.804	0.4, 0.976	0.444, 1.245	0.492, 1.513	0.557, 1.707	0.597, 1.926	0.683, 2.2	0.736, 2.521	0.808, 2.827	0.892, 3.155	0.989, 3.529	1.106, 3.868	1.313, 4.407	1.573, 5.046	1.928, 5.785	2.137, 6.279	75.495, 141.872
d=4	+ Ridge	0.566, 0.891	0.428, 1.013	0.459, 1.261	0.499, 1.518	0.571, 1.739	0.607, 1.961	0.7, 2.253	0.766, 2.595	0.849, 2.905	0.939, 3.261	1.047, 3.664	1.168, 4.026	1.376, 4.566	1.635, 5.224	1.995, 5.964	2.202, 6.457	96.744, 173.37
	+ Lasso	0.566, 0.891	0.428, 1.013	0.459, 1.261	0.5, 1.519	0.571, 1.739	0.607, 1.962	0.7, 2.253	0.767, 2.595	0.85, 2.906	0.939, 3.262	1.047, 3.665	1.168, 4.027	1.377, 4.566	1.635, 5.224	1.995, 5.965	2.202, 6.458	96.187, 172.056
d=5	+ Ridge	0.528, 0.816	0.404, 0.96	0.441, 1.217	0.487, 1.484	0.563, 1.718	0.591, 1.924	0.687, 2.22	0.751, 2.55	0.829, 2.838	0.912, 3.19	1.02, 3.586	1.143, 3.944	1.352, 4.486	1.61, 5.146	1.966, 5.881	2.174, 6.381	-
	+ Lasso	0.528, 0.816	0.404, 0.96	0.441, 1.218	0.487, 1.485	0.563, 1.719	0.591, 1.925	0.687, 2.221	0.752, 2.551	0.829, 2.839	0.913, 3.191	1.021, 3.587	1.144, 3.945	1.352, 4.487	1.611, 5.146	1.966, 5.882	2.175, 6.382	-
d=6	+ Ridge	0.569, 0.9	0.425, 1.01	0.454, 1.249	0.495, 1.505	0.561, 1.718	0.595, 1.929	0.688, 2.223	0.752, 2.545	0.829, 2.853	0.914, 3.199	1.021, 3.588	1.144, 3.947	1.352, 4.487	1.61, 5.139	1.964, 5.876	2.172, 6.372	-

Table 9: Estimation of the C and γparameters of SVR with polynomial features of degree 3

MSE(T1), MSE(T2)	C=100	C=1,000	C=10,000	C=100,000	C=1,000,000	C=10,000,000
$\gamma = 1e-10$	24.926, 42.136	19.118, 32.273	13.683, 23.051	>2h	too long	too long
$\gamma = 1e-9$	13.085, 20.586	10.522, 16.463	10.349, 16.799	10.776, 17.839	9.992, 15.990	10.317, 16.230
$\gamma = 1e-8$	26.907, 45.181	26.791, 44.930	26.698, 44.641	26.623, 44.550	expect ~26, expect ~45	expect ~26, expect ~45
$\gamma = 1e-7$	48.439, 82.943	48.341, 83.359	48.555, 83.372	47.702, 81.269	expect ~48, expect ~82	expect ~48, expect ~82
$\gamma = 1e-6$	61.653, 106.166	61.542, 106.167	61.604, 106.164	61.618, 106.155	expect ~61, expect ~106	expect ~61, expect ~106