Parkinson's Disease Progression Prediction

Prithvijit Dasgupta (prithvid) Nowrin Mohamed (nowrin) Sachin Salim (sachinks)
University of Michigan, Ann Arbor

Abstract:

Parkinson's disease is a chronic and progressive neurodegenerative disorder that affects the brain's dopamine-generating neurons, causing difficulties in everyday tasks such as speaking, writing, and walking. The severity of these symptoms worsens over time, posing a challenge for patients. This disease is typically diagnosed through clinical observations of motor symptoms, which can be subjective and prone to errors. Early non-motor symptoms may also be overlooked, making early diagnosis challenging. To address these issues, researchers have turned to machine learning to improve the accuracy and objectivity of PD diagnosis. In this study, we propose an approach for predicting Parkinson's disease progression with the dataset from Kaggle using several regression models like Linear Regression, Decision Tree Regression, Random Forest Regression and Extra Trees Regression. Comparison of these models are also done using SMAPE as the comparison parameter.

Motivation:

The motivation for the proposed project is to develop a model for predicting the progression of Parkinson's Disease using protein and peptide data measurements. Parkinson's Disease is a chronic and progressive neurological disorder that affects a significant portion of the population, causing symptoms such as tremors, stiffness, and impaired balance and coordination (Postuma et al., 2019). This project is chosen because early and accurate prediction of disease progression can help healthcare providers plan and adjust treatment regimens accordingly, which can significantly improve patients' quality of life.

The project aims to address the specific question of whether protein and peptide data can be used to predict the progression of Parkinson's Disease. Heemels et al., has conducted a study with proteins and peptide data measurement and have identified specific molecules that cause the disease so that researchers can gain insights into the underlying biological processes involved in Parkinson's Disease progression, which can help to identify potential new targets for drug development and improve our understanding of the disease.

This project is particularly important because current methods for assessing Parkinson's Disease progression rely on clinical observation, which may not be sensitive enough to detect subtle changes in disease progression or provide insights into the underlying biological processes involved (Postuma et al., 2019). Using protein and peptide data to predict disease progression could provide a more objective and comprehensive measure of disease progression and help to identify novel therapeutic targets. The study "Accuracy Improvement for Predicting Parkinson’s Disease Progression" in Scientific Reports developed a machine learning model that achieved an accuracy of 76% in predicting PD progression using data from the Parkinson’s Progression Markers Initiative (Maitra et al., 2019).

Overall, this project has the potential to significantly advance our understanding of Parkinson's Disease and improve patient care. By developing a model for predicting disease progression using protein and peptide data, we can identify new biomarkers for monitoring disease progression, improve our understanding of the biological mechanisms involved in Parkinson's Disease progression, and develop new therapeutic strategies for treating the disease (Athauda & Foltynie, 2015).

Data Sources:

The AMP Parkinson's Disease Progression Prediction Dataset, provided by the Accelerating Medicines Partnership for Parkinson's Disease, is accessible on Kaggle and contains clinical and molecular data from approximately 1000 participants. The dataset is presented in CSV format and has been downloaded and accessed through Pandas.

The primary dataset, "train_clinical_data.csv," includes demographic data, such as visit_id, visit_month, and patient_id, as well as disease severity scores and motor and non-motor symptom scores represented by Unified Parkinson's Disease Rating Scale (UPDRS) scores, such as updrs_1, updrs_2, updrs_3, updrs_4, and upd23b_clinical_state_on_medication (the information of if a patient is prescribed medication or not on a particular visit). The dataset comprises 2615 records, with a file size of 72 KB, and data types include string, integer, and boolean. The suuplemetal_clinical_data.csv contains 2223 patients information. This represents clinical records without any associated CSF samples. This data is intended to provide additional context about the typical progression of Parkinsons and shares the same columns as the clinical data.

The molecular data is split into protein and peptide data, available in "train_proteins.csv" and "train_peptides.csv," respectively. The protein dataset includes demographic data such as visit_id, visit_month, and patient_id, as well as molecular data like NPX and UniProt. NPX represents the normalized protein expression, indicating the protein's frequency of occurrence in the sample. However, this does not necessarily have a one-to-one relationship with component peptides as some proteins contain repeated copies of a given peptide. UnitProt refers to the UniProt ID code of the associated protein. The dataset consists of 233K records and has a file size of 7.3 MB, with data types including string, integer, and float.

The peptide dataset contains demographic data like visit_id, visit_month, and patient_id, similar to the protein and clinical data, as well as peptide and peptide abundance. Peptide represents the sequence of amino acids included in the peptide. The dataset consists of 982K records and has a file size of 48.9 MB, with data types including string, integer, and float. It is worth noting that the test set may contain peptides not found in the train set, and peptide abundance represents the frequency of the amino acid in the sample.

The links for datasets are as below,

1. Clinical data - Link
2. Proteins data - Link
3. Peptides data - Link
4. Supplemental clinical data - Link

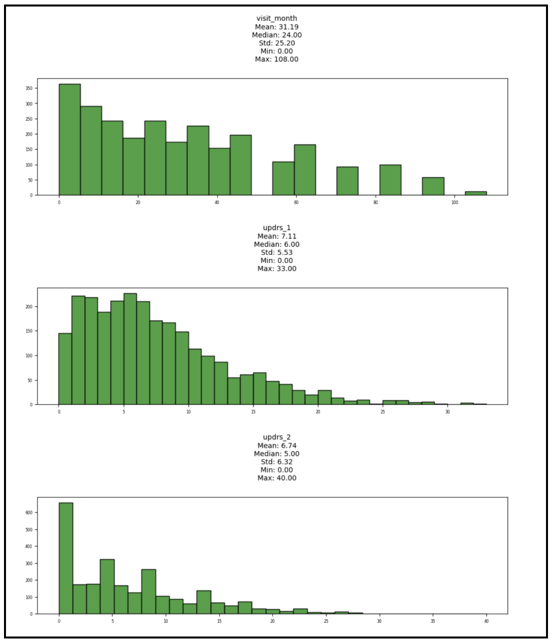
Exploratory Data Analysis and Data Manipulation:

EDA is an important first step in data analysis, as it helps to identify potential issues or inconsistencies in the data, and also helps to generate insights that can guide subsequent analyses. We performed EDA on the three different datasets we had collected which is explained in detail below,

EDA and Manipulation on Clinical data:

The initial step in analyzing a dataset is to determine its dimensions. Upon reading the data, it was determined that the dataset consists of 8 columns and 2615 rows. The subsequent step involves assessing the central tendency, dispersion, and shape of the dataset's distribution. In this regard, missing values were identified in columns updrs_1, updrs_2, updrs_4, and upd23b_clinical_state_on_medication, with values of 1, 2, 1038, and 1327, respectively. The average, minimum, and maximum values of each column were also computed, revealing that not all tests are conducted during every patient visit, and medication usage may differ from one visit to another. Furthermore, the data comprises 248 unique patient_id values and 17 unique visit_month values.

Histograms are a graphical representation of the distribution of a dataset, displaying the frequency of observations that fall within specific value ranges or bins. A histogram plot was constructed for the visit_month and updrs[1-4] columns, as depicted in Figures 1 and 2.



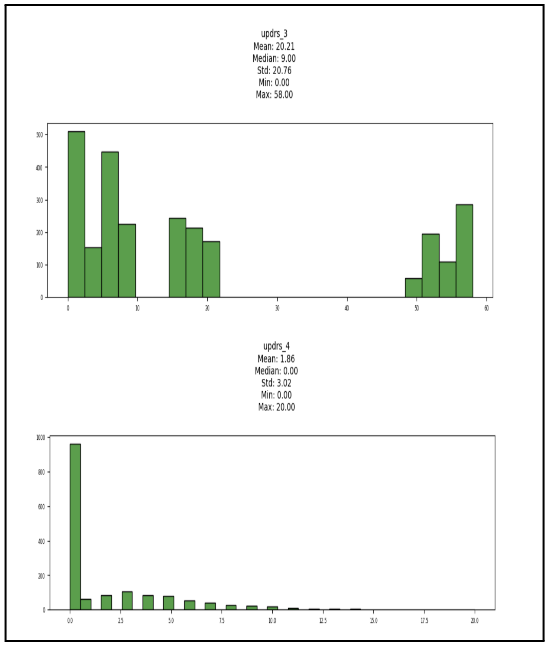


Fig.1. & Fig.2 Histogram plot on visit_month, updrs_[1-4]

From the plots, it can be seen that the frequency of people visiting over the course of the disease has gradually reduced and this could be because they might have passed away or might have stopped visiting because the medication might not be working on that or for any personal reasons. There are lots of 0 values in updrs_2, updrs_3 and updrs_4 but updrs_1 distribution is more balanced. After performing exploratory data analysis (EDA), it was discovered that medication was prescribed to patients 775 times, while in 513 instances, it was not prescribed. It should be noted that patients may not receive medication during every visit, even if they were prescribed it previously. This is because the data collection methodology is designed in such a way that if medication was given during one visit, the next visit is preferred to be without medication.

To further understand the correlation between the UPDRS scores, a heatmap was created and analyzed (as shown in Figure 3). The heatmap revealed that all the values are positively correlated, indicating that an increase in Non-Motor Experiences of Daily Living (NEDL) is accompanied by an increase in Motor Section symptoms. It was also observed that UPDRS_1 and UPDRS_2 are highly correlated.

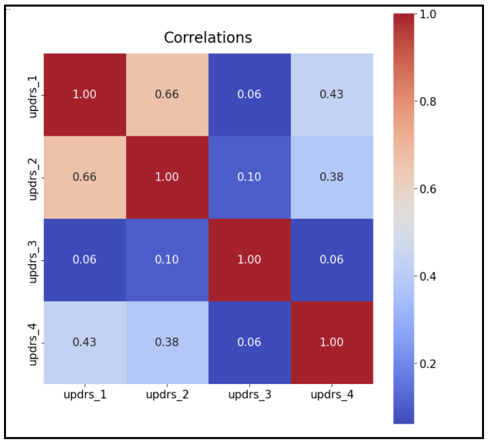


Fig.3. Heat map showing the correlation between updrs values

To examine the progression of UPDRS scores for a specific patient, a line plot was generated, which is displayed in Figure 4. This plot shows how the scores change over several months for the selected patient. It is worth noting that patients who are prescribed medication may exhibit improvement, but to verify this, another plot was created (Figure 5). This plot aimed to investigate whether medication has a substantial impact on a patient's condition. Surprisingly, it was observed that the disease continued to progress in both cases, even with heavier or lighter medication regimes.

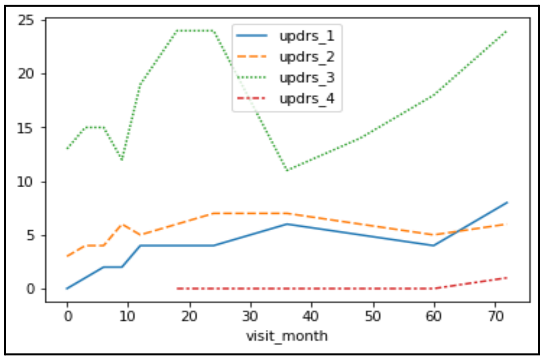


Fig. 4 Disease progression of Patient 23391

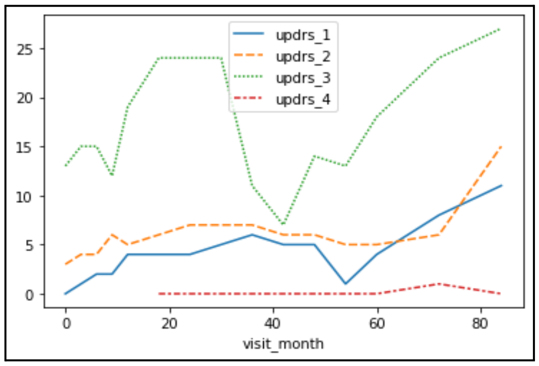


Fig.5. Progression while on light medication

Data Transformation and Filling NaN:

Upon analyzing the dataset, it was discovered that there are no null values in the CSF observations with the peptide and protein data. However, null values were present in the clinical and supplemental clinical data. To address this issue, the median of the entire UPDRS_[1-4] column was initially used to fill the NaN values as a baseline analysis. Additionally, for the 'upd23b_clinical_state_on_medication' column, NaN values were filled with the string 'Missing'.

However, using the median of the entire column to fill in missing values may not always provide accurate results since the data belongs to different patients. Therefore, a more precise method was used to improve the accuracy. Specifically, the missing values of UPDRS scores for each patient were replaced with the mean of their respective UPDRS scores. This approach helped fill around 70% of the missing values. For the remaining missing values, the data was grouped by visit month and filled with the mean data for each month. Finally, any remaining missing values were filled with the mean data for each UPDRS score across the entire dataset. Also, there are no duplicate entries found in the data. The code for handling missing values is below,

```
def fill_na(data):
  data_1 = data.groupby('patient_id').transform(lambda x: x.fillna(x.mean()))
  data_2 = data_1.groupby('visit_month').transform(lambda x: x.fillna(x.mean()))
  data_3 = data_2.fillna(data_2.mean())
  return data_3
```

EDA and Manipulation on Proteins and Peptides data:

The protein data comprises protein expression frequencies derived from the peptide-level data and is associated with UniProt ID codes. The number of patients in the protein data matches that in the clinical data, and there are 227 distinct proteins. On average, each patient has 938.5 proteins, and each patient-visit has 209.1 proteins. The UniProt ID codes are unique identifiers assigned to protein sequence entries in the comprehensive UniProt database, which contains functional and sequence information for proteins. These IDs typically start with a prefix that identifies the type of entry and are composed of letters and numbers. A histogram plot of the NPX values resulted in a heavily skewed distribution. Thus, a log scale was used which provided a more or less normal distribution as shown in Figure 6.

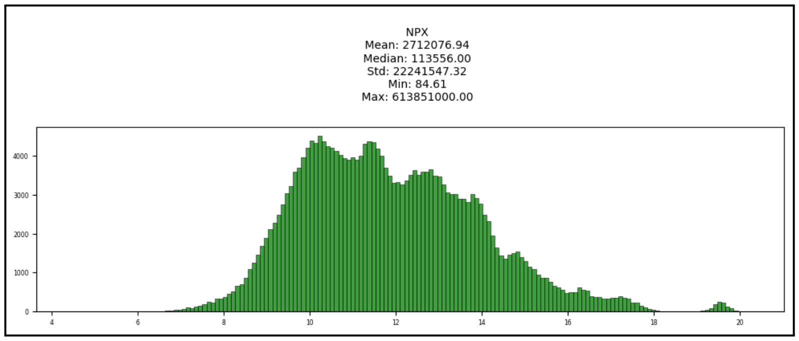


Fig.6. Histogram of NPX values on log scale

The UniProt database was examined, and it was determined that UniProt 220 appeared the most frequently, occurring 36 times. The distribution of the data is approximately normal, with a mean of 209.11, a minimum value of 37, and a maximum value of 224. The 25th, 50th, and 75th percentiles were 206.000000, 212.000000, and 216.000000, respectively. Overall, the data appears to be normally distributed.

To detect outliers and examine the distribution of protein expression levels, the NPX data for several proteins were plotted. Figure 7 illustrates that the NPX values are not significantly distant from the 75th percentile and could potentially enhance the model's predictive ability. Therefore, removing these values may not be necessary.

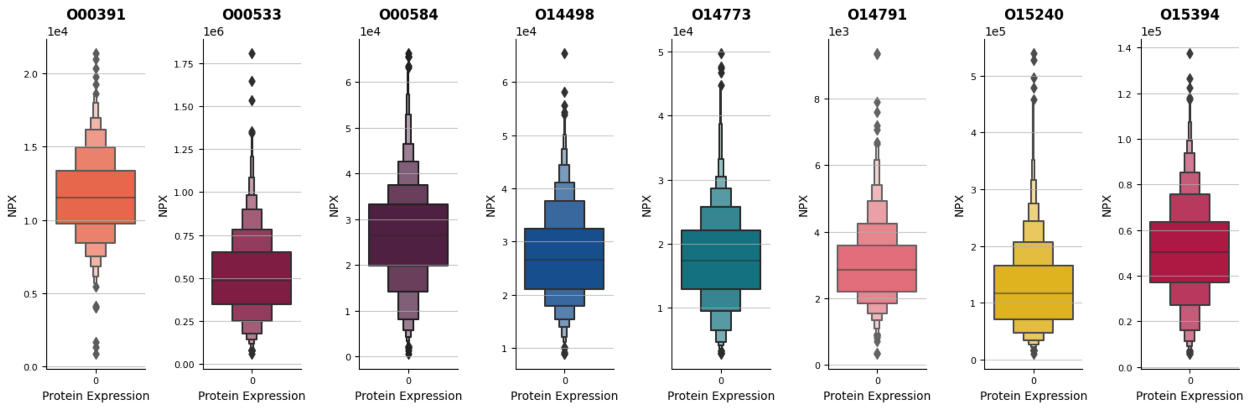


Fig.7. Box plot of NPX distribution of certain Protein Expression

Similar to the NPX plot, the plot for Peptide Abundance was heavily skewed and was plotted on log scale as shown in Figure 8. The data seems to be more or less normally distributed with less outliers.

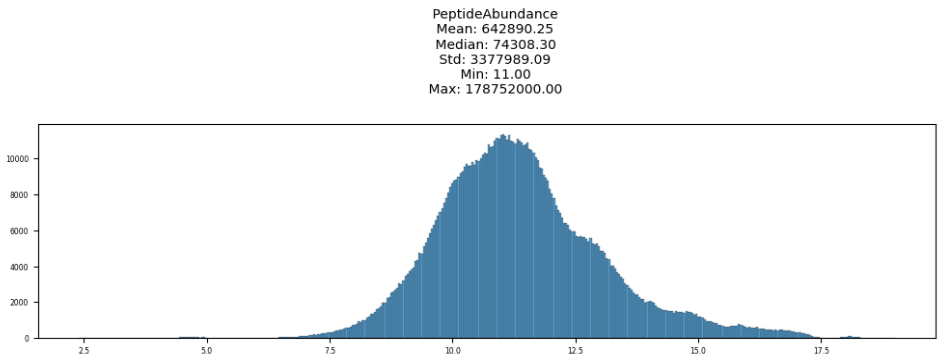


Fig.8. Histogram of Peptide Abundance values on log scale

Data Transformation and Filling NaN:

The initial conclusion that protein and peptide data had no null values was incorrect. After pivoting the data, 7.88% and 8.87% of missing values were found, respectively. To fill in the missing data, a patient- and visit-month grouping approach was used, followed by filling with mean data for each protein or peptide. Only 1068 visits had common data across protein, peptide, and clinical data, indicating that some visits were not significant. And there are no duplicates present in the data. The data transformation and finding common values code is as below,

```
def pivot_proteins():
  return proteins_data.pivot(
      index=['visit_id', 'patient_id', 'visit_month'], columns='UniProt', values='NPX')

def pivot_peptides():
  return peptides_data.pivot(
      index=['visit_id', 'patient_id', 'visit_month'], columns='Peptide', values='PeptideAbundance')
```

```
common_indices = np.intersect1d(peptide_features.index.values,
    clinical_features.index.values)
print(f"{len(common_indices)} visits are common between input\
and output features")
```

The distribution of top 5 NPX and Peptide Abundance data across visit months was analyzed. The protein and peptide data were merged with clinical data based on visit_id and the coefficient of variation was calculated for the top 5 proteins and peptide abundance values. The data was plotted separately for patients not on medication, those on medication, and missing data in Figure 9 and 8. The analysis suggests that medication may not significantly affect NPX values or the progression of the disease. The below code is written only for proteins and similary done for peptides as well. The code for this is as below,

```
train_proteins_df_agg = proteins_data[['patient_id', 'UniProt', 'NPX']]
train_proteins_df_agg = train_proteins_df_agg.groupby(['patient_id', 'UniProt'])['NPX']\
                      .aggregate(['mean', 'std'])
train_proteins_df_agg['CV_NPX[%]'] = train_proteins_df_agg['std'] / train_proteins_df_agg['mean'] * 100
# Mean CV value for UniProt
NPX_cv_mean = train_proteins_df_agg.groupby('UniProt')['CV_NPX[%]'].mean().reset_index()
NPX_cv_mean = NPX_cv_mean.sort_values(by='CV_NPX[%]', ascending=False).reset_index(drop=True)
protein_cv_top5 = NPX_cv_mean[:5]['UniProt']
train_proteins_df_agg_top5 = train_proteins_df_agg.query('UniProt in @protein_cv_top5').reset_index()
# Sort by protein_cv_top5 rank
train_proteins_df_agg_top5['order'] = 0
for i, protein in enumerate(protein_cv_top5):
  index = train_proteins_df_agg_top5.query(f'UniProt=="{protein}"').index
  train_proteins_df_agg_top5.loc[index, 'order'] = i
train_proteins_df_agg_top5.sort_values(by='order', inplace=True)
```

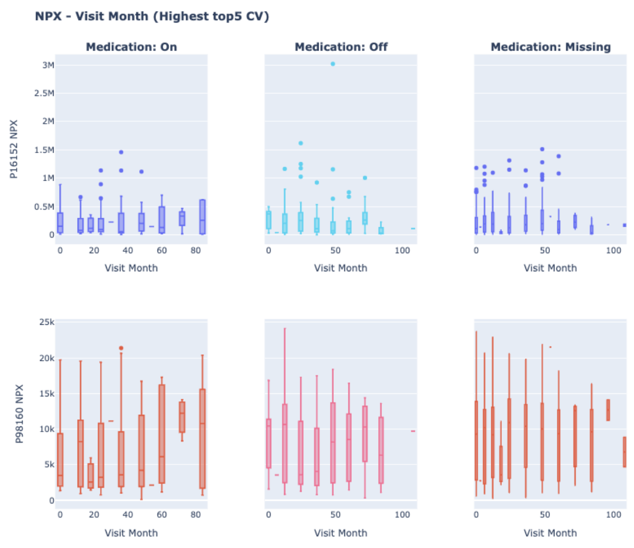


Fig.9. Distribution of NPX

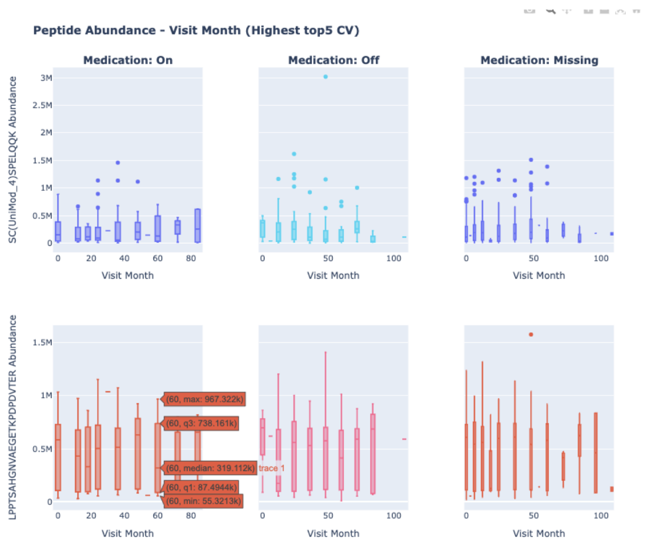


Fig.10. Distribution of Peptide abundance

EDA and Manipulation on Supplemental Clinical data:

The Supplemental data includes 771 unique patient_id values and 8 unique visit_month values. Concatenation of clinical data with supplemental clinical data was performed as they have identical shape (columns). Comparative study results are displayed in Figures 11 and 12.

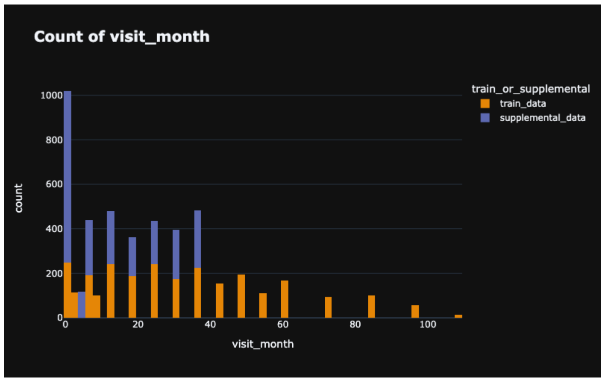


Fig.11. Visit_month comparison

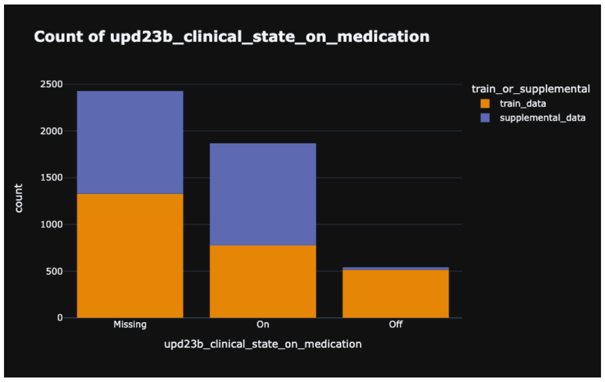


Fig. 12. Medication comparison

There are several differences between supplemental and clinical data. One major difference is that supplemental data completely lacks information on specific proteins and peptides, while clinical data typically includes this information. Additionally, the sets of patients included in each dataset are mutually exclusive.

Both supplemental and clinical data underwent various types of data manipulation and analysis, such as filling in missing values (NaN), merging and concatenating data frames, and pivoting tables. These techniques are used to conduct descriptive and inferential analyses of the data and to gain insights from the information contained in the datasets.

Challenges Faced in Data Manipulation:

In order to determine the most effective method of filling in missing values, we conducted a thorough analysis. It became clear early on that filling each protein's missing values with its mean or median across the entire dataset may not be the optimal approach. This is due to the significant correlation between the missing protein values and the specific patient being measured, as well as the month in which the measurement was taken. Therefore, we examined the variation in average protein values across patients and months, ultimately determining that the patient being measured was the best predictor for the missing protein values, followed by the month of measurement. Based on this analysis, we developed a strategy for filling in the missing protein values. However, when it came to the missing medication values, it was difficult to determine an appropriate replacement strategy. Since these values were binary and there was limited information available about why they were missing, we ultimately decided to simply label them as "Missing".

Training, Evaluation and Prediction

This study utilized the PyCaret library to train and compare multiple models, tune hyperparameters, perform cross-validation, and optimize the model for better accuracy. Since there were no sources available on how to combine the four different UPDRS scores into a single label, a separate model was created for each score.

The pre-processed dataset was split into three parts - a training set, a validation set, and a holdout set. The training set was used to train the machine learning model, while the validation set was used to evaluate the model's performance and tune its hyperparameters. The holdout set was used to test the model's performance after training it on both the training and validation datasets. Some patients were randomly selected and their information was held-out during the model selection/optimization process. The remaining dataset was passed to PyCaret for further tasks.

The PyCaret setup() function was used to split the data into training and validation sets, with the default train-test split of 70-30 being used in this project.

```
s = pycaret.regression.setup(data=final_X, target=Y)
```

The PyCaret framework was utilized to evaluate different models by utilizing the compare_model() function. This function enables users to compare various machine learning models supported by PyCaret. Most of the models are implemented in scikit-learn, but there are also a few additional models such as LightGBM, XGBoost, and Elastic Net. During the model comparison, the R2 score was used as a metric for evaluating model performance. A higher R2 score indicates a better fit of the model with the training data. Hence, the model with the highest R2 score was preferred in the comparison process.

```
pycaret.regression.compare_models(sort='R2')
```

This gives the best possible model with the highest R2 score as devised in Eq. 1. Except for UPDRS 4, the Extra Trees Regressor model was observed to have the highest R2 scores (for UPDRS 4 alone, Elastic Net Regression model had the higher value).

$$R^2 = \Sigma(\widehat{y}_i - \bar{y})^2 / \Sigma(y - \bar{y})^2 \qquad \text{Eq. 1}$$

where, $\widehat{y}_i$ represents the prediction or a point on the regression line, $\bar{y}$ represents the mean of all the values and y_i represents the actual values or the points.

PyCaret also provides an easy to use tune_model() function which enables developers to quickly tune their models. The tune_model() function was used to optimize the model selected in the previous step to have the best hyperparameters. The hyperparameters of the model are the parameters that are not learned from the data but are set before training, such as the regularization parameter in a logistic regression model. Using this function, cross-validation to search for the best hyperparameters within the default search space is performed.

```
pycaret.regression.tune_model(best_model, choose_better=True, optimize='SMAPE')
```

In the context of feature engineering, a time offset approach was adopted to predict disease progression. This approach follows a Markov chain pattern, whereby the current UPDRS score is calculated based on the previous month's available UPDRS score. To select the most relevant features for training, a correlation analysis and pairplot visualization were conducted. Based on the results, only UPDRS_1 and UPDRS_2 were deemed suitable for training, as they exhibit a steady progression and are not scaled. On the other hand, the protein data were scaled using a standard scalar since the distribution was wide.Moreover, feature selection was performed by selecting only the protein features and omitting the peptide features. This decision was made because each protein is composed of peptides, and the protein data is already encoded.

The optimization was performed on the SMAPE as devised in Eq. 2 metric as this was the metric that the Kaggle competition is being evaluated on. As a final step, the ensemble of the model was done using a Bagging Regressor with 20 estimators using the ensemble_model() function. For most of the cases, the model performed better with the ensemble model.

```
pycaret.regression.ensemble_models(tuned_model, method='Boosting', choose_better= True, n_estimators=20)
```

$$SMAPE = 1/n * \sum |forecast\ value - actual\ value| / (|actual\ value| + |forecast\ value|) / 2 \quad \text{Eq. 2}$$

finalize_model() was used to train the model on the entire dataset and finally, predict_model() was used on the holdout patient data to predict their UPDRS score. The model's workflow is as described in Figure 13.

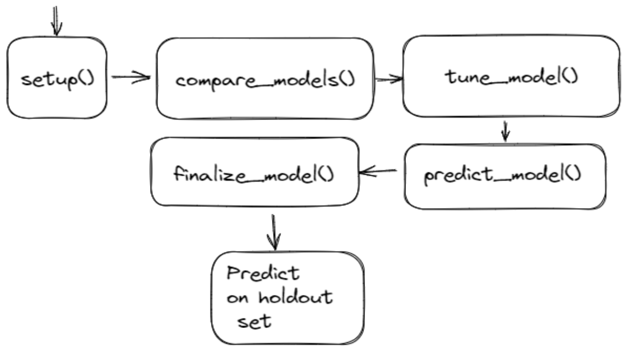


Fig. 13.. The model's workflow

Discussion

A qualitative discussion is necessary on our predicted progressions for the holdout patients. Figure 14 and 15 are two of the observed predictions, where the model does somewhat well and the other where the model is not able to predict very well.

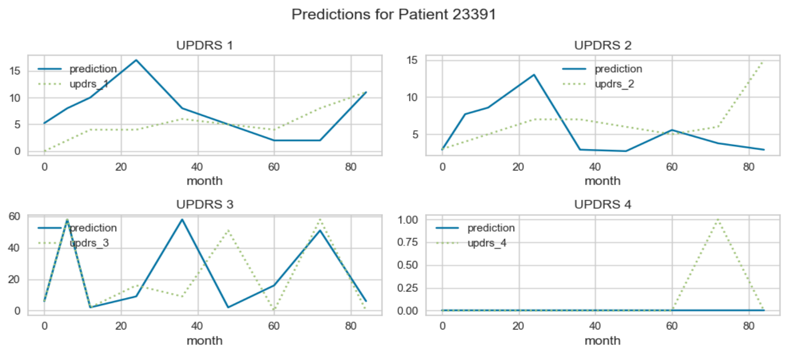


Fig. 14. The predicted time series for Patient 23391

For patient 23391, UPDRS 1 and UPDRS3 showed a tendency to overestimate the predicted value, however it can be argued that the predicted value trended in a generally correct direction. UPDRS 4 was the column in the dataset that had the most empty values, which were filled in with zeroes. The model seemed to learn that predicting zero was the best way to handle predicting this feature. Owing to the sparsity of data, the model could not be generalized for UPDRS 4. UPDRS 2 was the worst performing label for this patient where the values were overestimated and the trend did not finally match.

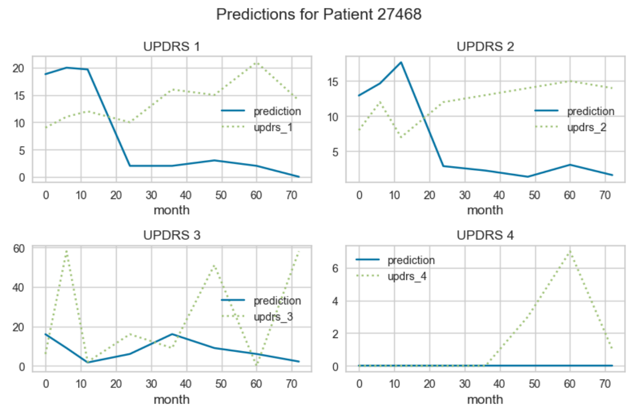


Figure 15. The predicted time series for Patient 27468

The observations for patient 27468 clearly shows that the prediction model clearly cannot be generalized for all patients. The predicted values for the patient did not even trend in the correct direction (a worsening disease should have increasing scores). Only UPDRS 4 had some match in the initial part of the time series (but that is more due to the model predicting 0 for all UPDRS 4 data).

SMAPE might not be the best metric to be selecting the model performance on. We could have kept optimizing a single metric (such as R^2) throughout the entire model creation and test workflow and that might have given us better results for the dataset. Also, since the UPDRS score itself is more akin to a rating that the patient/caregiver provides rather than a measurement made using a tool, it is unclear on how to generalize the label metric across patients. Perhaps one way could be to convert the label itself to its scaled Z-score values in order to predict the label Z-scores.

Considering this model ethically, it should definitely not be something that could be used in production as it clearly does not do a good job of predicting Parkinson's diseases. Deploying such a model would be a hazard and might negatively impact most of the patients (even though it achieves good SMAPE scores during the cross-validated training and evaluation). It is important to do a qualitative check on the predicted values before deploying medical models to the real world.

Future Work

There are several potential avenues for future work in predicting Parkinson's disease. Firstly, to address biases in the dataset, it is recommended to control for confounding variables and use appropriate statistical methods like stratified sampling or propensity score matching. Secondly, an embedding matrix can be used to fill in the NaN values in clinical, protein, and peptide data such as updrs_values and medication_values. This will provide a numeric representation of the categorical data, and the resulting dataset can be utilized for further analysis. Finally, past data can be leveraged to make future predictions about the progression of the disease. Although UPDRS scores and protein/peptide samples may not be available for subsequent visits, the past data can still be used to forecast the progression of Parkinson's disease.

Statement of Work

The team collectively and carefully brainstormed ideas on the problem statement, analysis methods, data manipulation, model selection, and evaluation parameters.

Nowrin Mohamed conducted Exploratory Data Analysis on the Clinical, Proteins, Peptide, and Supplementary clinical data. This included identifying null values, filling NaN values, finding the progression of UPDRS scores across months, and analyzing the correlation between UPDRS scores. Nowrin also worked on the report with input from the rest of the team.

Sachin Salim analyzed the Proteins data and helped develop a baseline model without PyCaret to gain initial insights into the data. Sachin also performed Inferential Analysis and high-level data manipulation, including transforming the protein and peptide data, merging and concatenating different tables, and filling NaN values more intelligently.

Prithvijit Dasgupta handled the training, evaluation, and prediction of the model. This included splitting the dataset into training and evaluation sets, tuning hyperparameters, extracting and selecting features, evaluating and predicting the model. Prithvijit also worked on understanding the clinical data of short-term and long-term patients and how medication affects the progression of UPDRS scores. The training part of the report was also carefully edited by him.

All team members contributed equally to the visualization part, creating graphs as needed throughout the analysis process.

Our collaboration was very fruitful because we all shared knowledge and collectively brainstormed ideas which was an enriching experience. We assessed our strengths and rightly split the work after mutual agreement and we believe because of the comfort we had, we were very productive throughout our project. The only thing we had struggled with was to match our schedules so mostly we met on weekends and assigned our works for the week and reported them. This could have improved in such a way to meet more than twice a week at least virtually so to understand and update progress. But otherwise, this was a dream team.

References:

[1] Athauda, Dilan, and Thomas Foltynie. “The ongoing pursuit of neuroprotective therapies in Parkinson disease.” Nature reviews. Neurology vol. 11,1 (2015): 25-40. doi:10.1038/nrneurol.2014.226
[2] Heemels, Marie-Thérèse. “Neurodegenerative diseases.” Nature vol. 539,7628 (2016): 179. doi:10.1038/539179a
[3] Postuma, Ronald B et al. “MDS clinical diagnostic criteria for Parkinson's disease.” Movement disorders : official journal of the Movement Disorder Society vol. 30,12 (2015): 1591-601. doi:10.1002/mds.26424
[4] Cooper JF, Van Raamsdonk JM. Modeling Parkinson's Disease in C. elegans. J Parkinsons Dis. 2018;8(1):17-32. doi: 10.3233/JPD-171258. Erratum in: J Parkinsons Dis. 2020;10(2):745. PMID: 29480229; PMCID: PMC5836411.
[5] Kaggle
[6] AMP Parkinson's Disease Progression Prediction
[7] PyCaret