COMP9417 Project: Parkinson’s Disease Progression Prediction

z5433256	z5411809	z5097965	z5440755	z5415903
Xiaotian Wei	Ruoao Zhang	Hongyu Chen	Junjie Yu	Chenxu Wu

1. Introduction

Parkinson’s disease (PD) is a disabling brain disorder that affects movements, cognition, sleep, and other normal functions and unfortunately, there is no current cure for it. However, rapid developments in AI and machine learning offer hope in predicting the disease's progression, which may help enabling better personalized treatment plans for patients.
An opportunity to apply machine learning in this context can be found in the featured and completed Kaggle competition which is:
https://www.kaggle.com/competitions/amp-parkinsons-disease-progression-prediction
The aim of this competition is to predict the severity and progression of Parkinson's disease. Specifically, the goal is to train a model that can predict The Unified Parkinson’s Disease Rating Scale (UPDRS)scores which includes 'updrs_1', 'updrs_2', 'updrs_3', and 'updrs_4'.
This competition provides a pinned example of data processing guidance[1], where the Random forest model is applied to complete the task. However, our group hopes to apply different types of machine learning methods learned in COMP9417 course to this competition, compare their performance and then we select the best model among them. We finally choose to implement Logistic Regression model, SVM, XGBoost, MLP and deep learning model to complete to task.
The competition official uses sMAPE as the evaluation metric. However, in order to better compare the performance of the models, we also use MSE as evaluation metric.

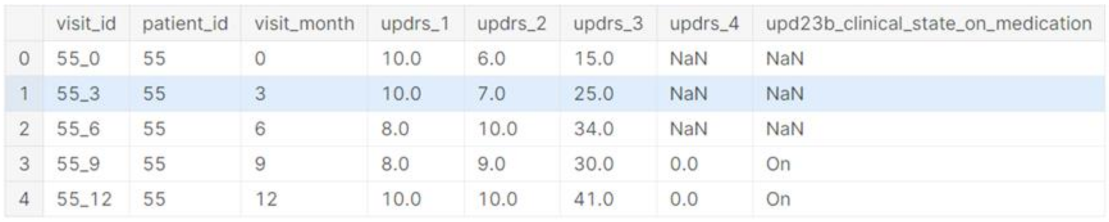
Table 1-1 evaluation metric

sMAPE	MSE
$\text{SMAPE} = \frac{1}{n}\sum_{t=1}^{n}\frac{\lvert F_t - A_t\rvert}{(A_t + F_t)/2}$	$\text{MSE}= \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2$

2. Data processing

2.1. An overview of dataset

The dataset consists of three parts: the protein dataset which records the NPX (Normalized Protein Expression) values of all Parkinson's related proteins for each visit, the peptide dataset which records the Peptide Abundance of each peptide in every Parkinson's related protein and the clinical dataset which records the Unified Parkinson’s Disease Rating Scale (UPDRS). Each dataset includes different biological and clinical information about patients.
We first use the Pandas library.to load these datasets and we print out the first 5 entries to see all dimensions of the three datasets.

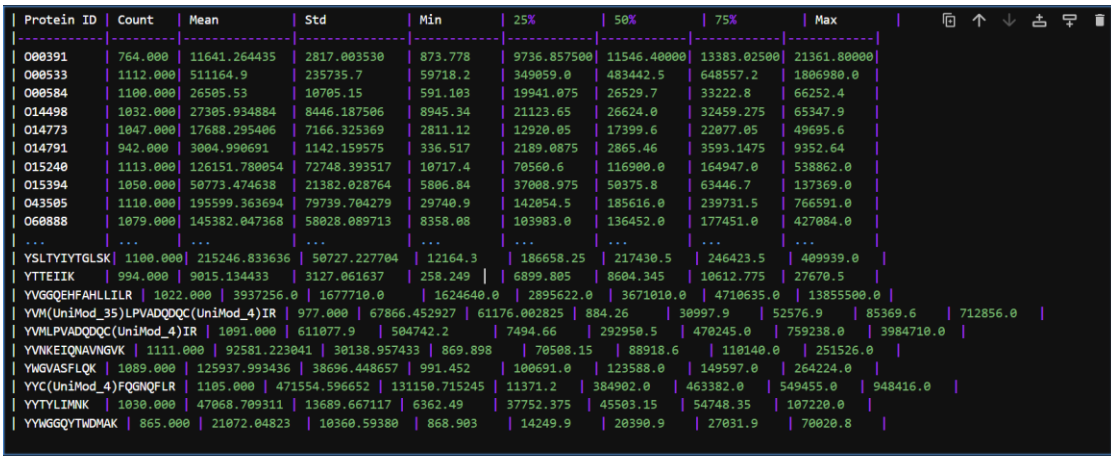


	visit_id	patient_id	visit_month	updrs_1	updrs_2	updrs_3	updrs_4	upd23b_clinical_state_on_medication
0	55_0	55	0	10.0	6.0	15.0	NaN	NaN
1	55_3	55	3	10.0	7.0	25.0	NaN	NaN
2	55_6	55	6	8.0	10.0	34.0	NaN	NaN
3	55_9	55	9	8.0	9.0	30.0	0.0	On
4	55_12	55	12	10.0	10.0	41.0	0.0	On

Figure 2-1 the first 5 entries of the clinical dataset

2.2. Data cleaning and integration

Data cleaning is the first step in our data preprocessing. We first checked each dataset for missing and anomalous values. After analyzing, we found that some columns have a certain number of missing values. To tackle this, we deleting rows with excessive missing values and we fill the missing values of numeric features with the median.
After cleaning the data, we merge the three datasets into a complete dataset for modeling. The strategy we used is to group and pivot the protein and peptide datasets and then merge them with the clinical dataset.



```
| Protein ID | Count   | Mean          | Std           | Min         | 25%         | 50%         | 75%         | Max         |
|------------|---------|---------------|---------------|-------------|-------------|-------------|-------------|-------------|
| O00391     | 764.000 | 11641.264435  | 2817.003530   | 873.778     | 9736.857500| 11546.40000| 13383.02500| 21361.80000|
| O00533     | 1112.000| 511164.9      | 235735.7      | 59718.2     | 349059.0    | 483442.5    | 648557.2    | 1806980.0   |
| O00584     | 1100.000| 26505.53      | 10705.15      | 591.103     | 19941.075   | 26529.7     | 33222.8     | 66252.4     |
| O14498     | 1032.000| 27305.934884  | 8446.187506   | 8945.34     | 21123.65    | 26624.0     | 32459.275   | 65347.9     |
| O14773     | 1047.000| 17688.295406  | 7166.325369   | 2811.12     | 12920.05    | 17399.6     | 22077.05    | 49695.6     |
| O14791     | 942.000 | 3004.990691   | 1142.159575   | 336.517     | 2189.0875   | 2865.46     | 3593.1475   | 9352.64     |
| O15240     | 1113.000| 126151.780054 | 72748.393517  | 10717.4     | 70560.6     | 116900.0    | 164947.0    | 538862.0    |
| O15394     | 1050.000| 50773.474638  | 21382.028764  | 5806.84     | 37008.975   | 50375.8     | 63446.7     | 137369.0    |
| O43505     | 1110.000| 195599.363694 | 79739.704279  | 29740.9     | 142054.5    | 185616.0    | 239731.5    | 766591.0    |
| O60888     | 1079.000| 145382.047368 | 58028.089713  | 8358.08     | 103983.0    | 136452.0    | 177451.0    | 427084.0    |
| ...        | ...     | ...           | ...           | ...         | ...         | ...         | ...         | ...         |
| YSLTYIYTGLSK| 1100.000| 215246.833636 | 50727.227704  | 12164.3     | 186658.25   | 217430.5    | 246423.5    | 409939.0    |
| YTTEIIK    | 994.000 | 9015.134433   | 3127.061637   | 258.249 |   | 6899.805    | 8604.345    | 10612.775   | 27670.5     |
| YVGGQEHFAHLLILR | 1022.000 | 3937256.0 | 1677710.0     | 1624640.0   | 2895622.0   | 3671010.0   | 4710635.0   | 13855500.0 |
| YVM(UniMod_35)LPVADQDQC(UniMod_4)IR | 977.000 | 67866.452927 | 61176.002825 | 884.26      | 30997.9     | 52576.9     | 85369.6     | 712856.0    |
| YVMLPVADQDQC(UniMod_4)IR | 1091.000 | 611077.9    | 504742.2      | 7494.66     | 292950.5    | 470245.0    | 759238.0    | 3984710.0   |
| YVNKEIQNAVNGVK | 1111.000 | 92581.223041 | 30138.957433 | 869.898     | 70508.15    | 88918.6     | 110140.0    | 251526.0    |
| YWGVASFLQK | 1089.000 | 125937.993436 | 38696.448657 | 991.452     | 100691.0    | 123588.0    | 149597.0    | 264224.0    |
| YYC(UniMod_4)FQGNQFLR | 1105.000 | 471554.596652 | 131150.715245 | 11371.2     | 384902.0    | 463382.0    | 549455.0    | 948416.0    |
| YYTYLIMNK  | 1030.000 | 47068.709311 | 13689.667117 | 6362.49     | 37752.375   | 45503.15    | 54748.35    | 107220.0    |
| YYWGGQYTWDMAK | 865.000 | 21072.04823  | 10360.59380   | 868.903     | 14249.9     | 20390.9     | 27031.9     | 70020.8     |
```

Figure 2-2 a print-out of processed dataset

Considering that the pinned solution divides the validation set from the training set, in order to ensure data consistency during the process of model comparison, we basically follow the recommended data processing method while we set a fixed random seed in the dataset partition process.

3. Implementation of Linear Regression model

3.1. Model selection and model building

Since we decide to build a linear model, logistic regression is the most basic model to analyze.

In statistics, the logistic model (also known as the logit model) is a statistical model designed to predict the probability of a certain event occurring. It does this by modeling the log-odds of the event as a linear combination of one or more independent variables or responses. Essentially, the log-odds refers to the logarithm of the odds of the event. This is a valuable way to analyze probability, because it re-scales the probabilities into a spectrum.

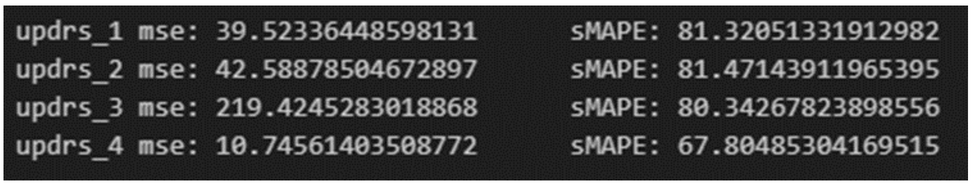
Logistic regression, which is used in regression analysis, predicts the parameters of a logistic model. This means it determines the coefficients in the linear combination, or put another way, it finds the weights for the independent variables in the model that best predict the log-odds of the event[2].

We choose LogisticRegression from linear_model of sklearn as our model. The hyperparameters are choosing as below:

Table 3-1: parameters chosen for Logistic Regression Model[3]

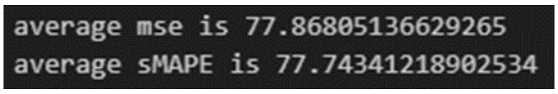
Max_iter	Solver	Multi_class	C	Tol
5000	Sag	Multinomial	3	1e-4

The reason for choosing those parameters is to improve the model's performance which is visualized by MSE (Mean Squared Error) and sMAPE (Symmetric Mean Absolute Percentage Error). The main idea is to minimize both two metrics. The final metrics are as follows:



```
updrs_1 mse: 39.52336448598131      sMAPE: 81.32051331912982
updrs_2 mse: 42.58878504672897      sMAPE: 81.47143911965395
updrs_3 mse: 219.4245283018868      sMAPE: 80.34267823898556
updrs_4 mse: 10.74561403508772      sMAPE: 67.80485304169515
```

Figure 3-1: metrics MSE and sMAPE



```
average mse is 77.86805136629265
average sMAPE is 77.74341218902534
```

Figure 3-2: overall performance

3.2. Model training

As other models, preprocessing data is very important. By analyzing our dataset, it is obvious to see the null value in the dataset. We tried two methods to minimize the impact. First is to replace the null value with mean value of the feature column, the other is to replace with 0 for instance. The second method is unexpected to perform better. It may because it is better to minimize the weight of the null value, as 0 is much smaller than other values in the dataset.
We also tried to reduce the features of the original dataset, since there are too many attributes. Methods such as descriptive statistics, inferential statistics and correlation analysis are considered. However, it seems that each method gives very different results based on the model of logistic regression. Therefore, we decide to use all features of the original dataset, since it is comprehensive and complete.
Model begins in default parameters, which gives an unreliable result. Model's convergence during training is mainly influenced by iterations, so max iteration needs to be increased. By testing, an iteration below 1000 may cause not possible to converge for specific label of the dataset. The solver of this model is 'sag', which is suitable for large-scale linear problems and is apt for large dataset with a smaller memory requirement. It uses the Stochastic Average Gradient descent algorithm. It is better compared to 'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', and 'saga'. C and tol referred to inverse of regularization strength and tolerance for stopping. Unsuitable values for these two may cause overfitting or underfitting.

3.3. Brief summary

From Figure 3-2, we can glean insights into the performance of a logistic regression model based on two metrics: the average Mean Squared Error (MSE) and the Symmetric Mean Absolute Percentage Error (sMAPE). As in the description of this competition, sMAPE is prioritized as the principal evaluation metric. Across all labels, the logistic regression model exhibits reasonable performance in terms of sMAPE, with 'updrs_4' displaying the lowest at 67.80 and 'updrs_2' the highest at 81.47.
However, when we shift our attention to the secondary judgement criterion, the MSE, there are clear difference in model performance across different labels. The 'updrs_3' label showcases a remarkably high MSE value of 219.42, which significantly contrasts with the other labels, particularly 'updrs_4' that has an MSE of just 10.74. Given that the MSE evaluates the average squared differences between predicted and actual values, it is theoretically sensitive to larger errors, emphasizing them in its computation. This indicates that the logistic regression model, while broadly effective, is not good for 'updrs_3'.
Logistic regression, as a model, assumes a linear relationship between the independent variables and the logarithmic odds of the dependent variable. While it is often suitable for classification problems, it might not capture the complexities or non-linear patterns associated with certain labels, such as 'updrs_3'. The differences in MSE values across the labels might be indicative of this limitation. It seems that 'updrs_3' data does not fit well with the capabilities of logistic regression, leading to larger prediction errors.

4. Implementation of SVM model

4.1. The selection of model

Support vector machine (SVM) is a supervised machine learning algorithm used for solving classification or regression problems. It is widely used because of high speed and good performance with limited samples[4]. We think it is a good choice for detecting four target values with this method.
SVM can divided into support vector classification (SVC) and support vector regression (SVR). In this project, we use SVM to predict the Parkinson’s disease progression. The picture below is used in order to explain SVM better.

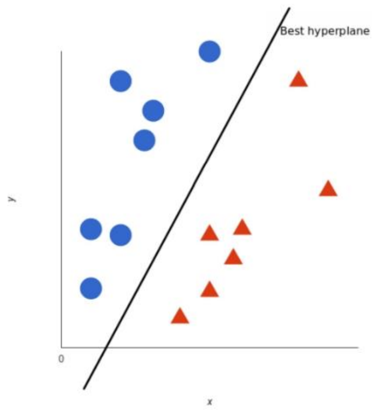


Figure 4-1 choosing the best hyperplane

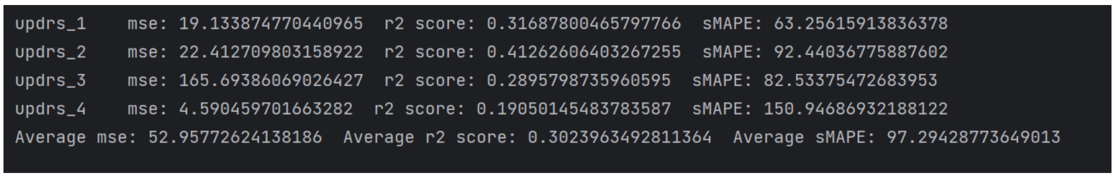
The dots in the picture stand for the training data and the line is the best hyperplane to classify all data into two different types. The condition for choosing the best hyperplane is let the distance between hyperplane and nearest dots from each type to become largest. And SVR is changed based on this. The dots also stand for training data but they are not classified. The condition to choose the best hyperplane is to let the distance between hyperplane and farthest dots to become smallest. SVR is different with normal linear regression. If the prediction is not equal to true value, the normal regression algorithm should calculate the loss but in SVR model, it should be calculated when the absolute value of difference is larger than margin we set. And we can improve SVR model by maximize the width of margin and minimize the total loss of SVR model.

4.2. Model training

We preprocess the dataset in previous part and we decide to divide the total dataset into training set which is 80% of dataset and testing set which is 20% of dataset. And when we print out the dataset after processing, it can be found that there are many null values in the table. It cannot be processed when training the model. So we decide to use 0 to replace these table grids. When we try to build a model, the choice of parameters is very important and parameters of model can great influence the result of detection. We want to find the better parameters of model and use the GridSearchCV[5] from sklearn package to solve the problem here. It can compare the result of different parameter combinations and gives us the best parameter. We enter some values by our experience and find the best combination with GridSearchCV. After that we train the model with training set. It can teach the model to recognize patterns and relationships between features and their corresponding target labels and after training, the model can give the predicted values of updrs_1, updrs_2, updrs_3 and updrs_4 which can help to determine the Parkinson progression of patients.

4.3.Result analysis

In order to evaluate the model better, we use mean squared error (MSE), r2 score and symmetric mean absolute percentage error(sMAPE). They all can evaluate the result of progression tasks. And we think sMAPE can provide a more balanced assessment when there is a large deviation in the dataset. So we mainly use sMAPE to evaluate the model. The results of SVM model are shown below:



```
updrs_1    mse: 19.133874770440965  r2 score: 0.31687800465797766  sMAPE: 63.25615913836378
updrs_2    mse: 22.412709803158922  r2 score: 0.41262606403267255  sMAPE: 92.44036775887602
updrs_3    mse: 165.69386069026427  r2 score: 0.2895798735960595  sMAPE: 82.53375472683953
updrs_4    mse: 4.590459701663282  r2 score: 0.19050145483783587  sMAPE: 150.94686932188122
Average mse: 52.95772624138186  Average r2 score: 0.3023963492811364  Average sMAPE: 97.29428773649013
```

Figure 4-2 The results of SVM model

And in order to show the result more clearly, we plot the result:

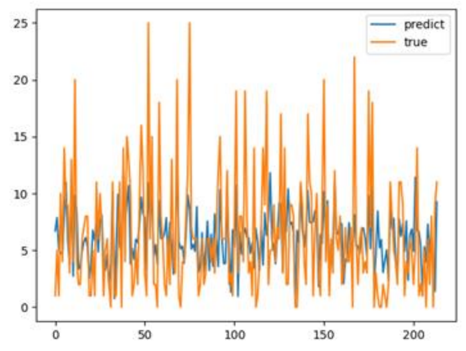
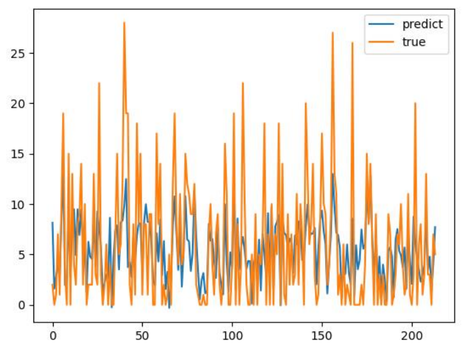
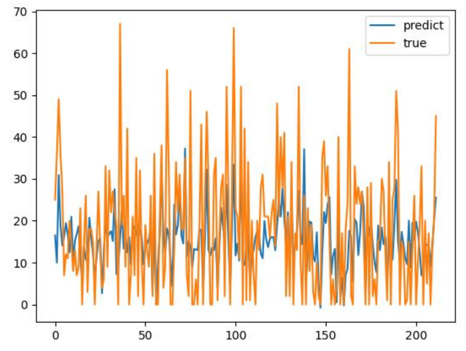
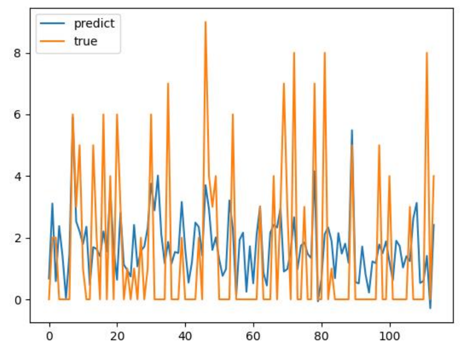
	
(a). updrs_1	(b). updrs_2
	
(c) updrs_3	(d) updrs_4

Figure 4-3 Visualization of the results of SVM model

It can be seen that the result of three different evaluation standards. And the total evaluation of this model. The values of mse and sMAPE are better if they are smaller and r2 score is better if it is closer to 1. The result of this model is acceptable. It shows that this model can effectively detect the value of updrs_1, updrs_2, updrs_3 and updrs4. However, the detection of updrs_4 is the worst because compared with other targets, updrs4 has a very large sMAPE . We think it is because the values of updrs4 in dataset is totally different with three other targets. And we can see that in plot the image of updrs_4 is more sparses than others. We think it proves that the values of updrs_4 changes more abrupt. And the detection of updrs_4 is harder.

5. Implementation of XGBoost model

5.1. Selection and model building

The pinned solution in this competition applied the model of Random Forest, which inspired us that some other kind of decision tree based ensemble learning methods may also be suitable to solve the problem.
We choose XGBoost algorithm which is a gradient boosting algorithm and it can be used for our task. In XGBoost, each decision tree is constructed sequentially and gradually, and in each step of tree building, the results of the previous decision tree would be considered to obtain better prediction results[6].
The choice of hyperparameters depends on their impact on the accuracy of the model. Among all commonly used hyperparameters in XGBoost, we picked n_estimators which is the number of decision trees, learning_rate which controls the change of the weights on each decision tree's prediction, max_depth which limits the depth of the decision trees to control model complexity and prevent overfitting, min_child_weight which represents the minimum sum of instance weights needed in a child for a split to happen and subsample which is used to control the fraction of samples used for training each individual tree for later hyperparameter tuning.

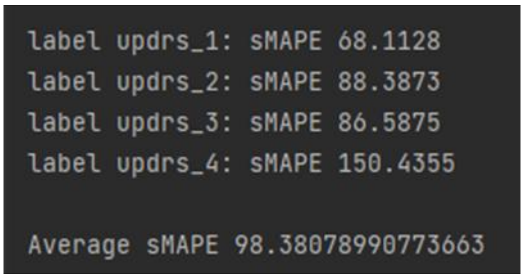
5.2. XGBoost training

We divided the dataset processed in chapter2 into a training set and a validation set in a ratio of four to one with a fixed random seed. For hyperparameter tuning, we at first tried manual parameter tuning method which turned out to be very inefficient. Then we tried to use some automatic parameter tuning methods such as GirdSearchCV and RandomizedSearchCV. However, both methods are very time-consuming in the actual training process due to the excessive combination of parameters.
Finally, we turned to Bayesian search method which makes use of previously tried parameter combinations to find parameter combinations that make the objective function more optimal[7]. In this way it will be less time-consuming than GirdSearchCV and RandomizedSearchCV. So we applied Hyperopt for hyperparameter tuning. The best parameter combination is shown in table 4-1:

Table 5-1 best parameter combination

Model for	n_estimators	learning_rate	max_depth	min_child_weight	subsample
updrs_1	685	0.046	4	4	0.84
updrs_2	692	0.046	3	4	0.44
updrs_3	483	0.056	4	6	0.55
updrs_4	406	0.030	6	4	0.50

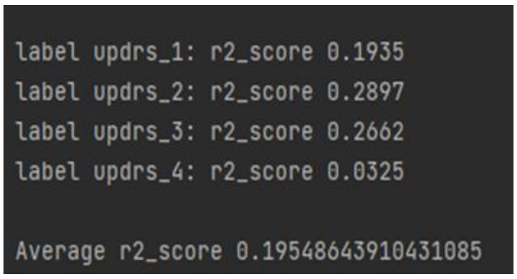
After getting the best parameter combinations for the first 4 segments of the UPDRS, updrs_1, updrs_2, updrs_3 and updrs_4, we print out the model evaluation metrics as follows:



```
label updrs_1: sMAPE 68.1128
label updrs_2: sMAPE 88.3873
label updrs_3: sMAPE 86.5875
label updrs_4: sMAPE 150.4355

Average sMAPE 98.38078990773663
```

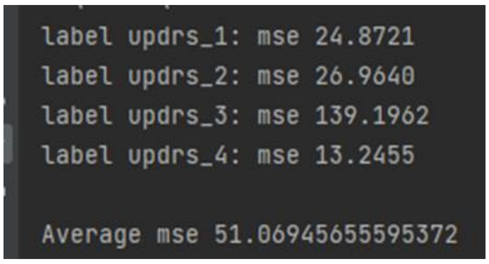
(a) sMAPE of XGBoost



```
label updrs_1: r2_score 0.1935
label updrs_2: r2_score 0.2897
label updrs_3: r2_score 0.2662
label updrs_4: r2_score 0.0325

Average r2_score 0.19548643910431085
```

(b) r2_score of XGBoost



```
label updrs_1: mse 24.8721
label updrs_2: mse 26.9640
label updrs_3: mse 139.1962
label updrs_4: mse 13.2455

Average mse 51.06945655595372
```

(c) mse of XGBoost

Figure 5-1 evaluation metrics of XGBoost

5.3. feature importance analysis

During the model training process, in order to visualize the importance of features to updrs_1, updrs_2, updrs_3 and updrs_4, we printed out the top 60 features in the feature importance ranking. For updrs_1, feature QPLGMISLMK has the highest ranking of 39.0 and the overall trend of importance of features is a smooth decline. For updrs_2, feature P04211 has the highest ranking of 20.0 and the overall trend of importance of features is a smooth decline. For updrs_3, feature DVMYTDWKK and P01861 have the highest ranking of 22.0 and the overall trend of importance of features is a smooth decline. For updrs_4, feature EWSVNSVGK has the highest ranking of 178.0 while the second feature only has a ranking of 49.0 so there is a sudden drop. And the trend of importance of the rest features is a smooth decline. The importance of features to updrs_4 is shown in figure 5-2:

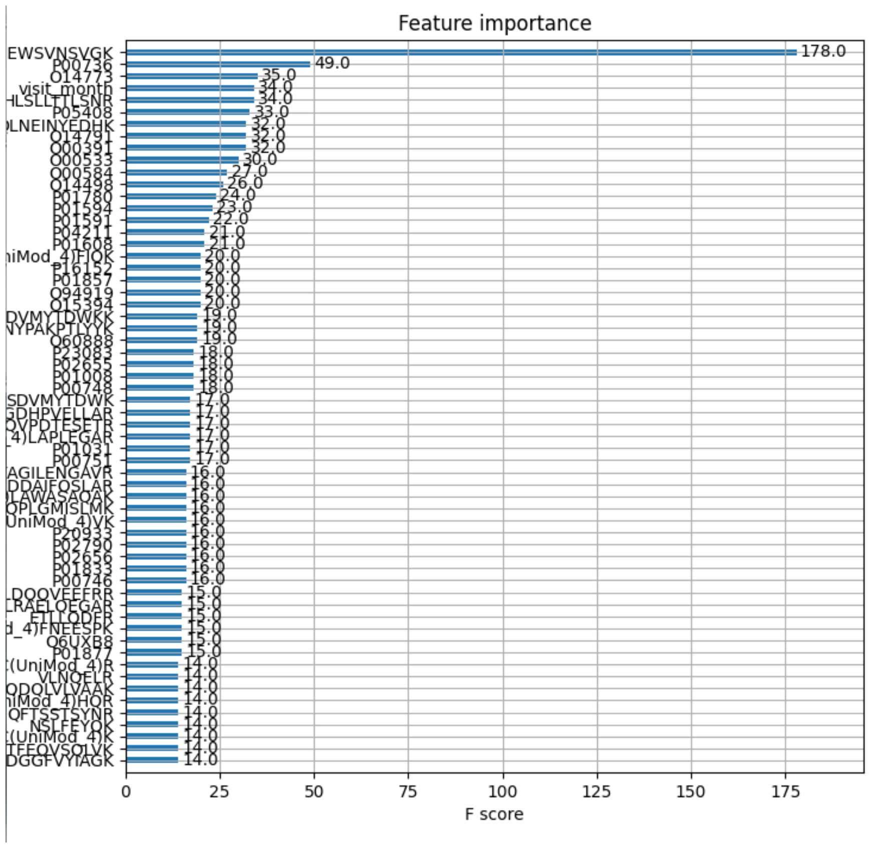
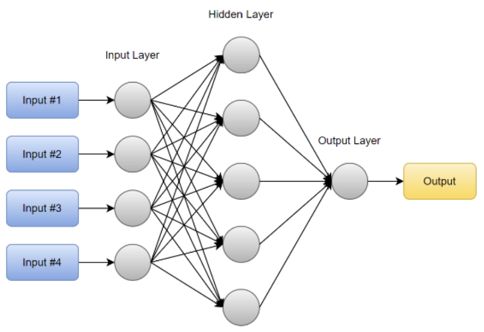


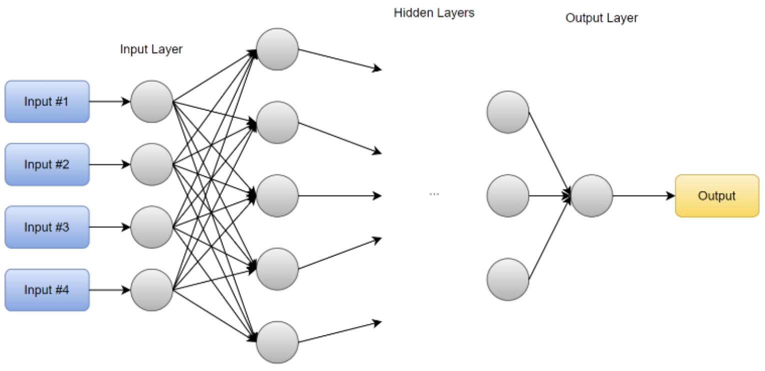
Figure 5-2 Feature importance ranking of updrs_4

Considering that the number of features is huge, a possible optimization solution of XGBoost in future is to delete feature columns with low importance for each target value.

6. Implementation of Multi-Layer Perceptron and Deep Learning

In this section, multi-layer perceptron (MLP) and deep learning models[8] are applied to give predictions to four labels, namely *updrs_1*, *updrs_2*, *updrs_3* and *updrs_4*. MLP consists of three types of layers, namely input layer, hidden layer and output layer, while inside each layer multiple neurons are formed based on the weighted sum of their inputs. Deep learning is based on the architecture of MLP and the only critical difference is that deep learning model introduces multiple hidden layers. Brief structures of MLP and deep learning are provided below.





(a). MLP (b). Deep Learning Model

Figure 6-1 Brief structures of MLP and deep learning

The data frame (after processed using the original data set) that fed into the neural network has more than 1100 features, which is considered to be a relatively large value compared with total amount of data. Therefore, in order to improve model efficiency and prevent the potential over-fitting problem, principal component analysis (PCA) is applied to decrease the total number of features before training. The number of new features after PCA becomes 200, captures more than 90% variance of the original data frame. The Scree plot for PCA (explained variance by each principal component) and the plot for cumulative explained variance are provided below:

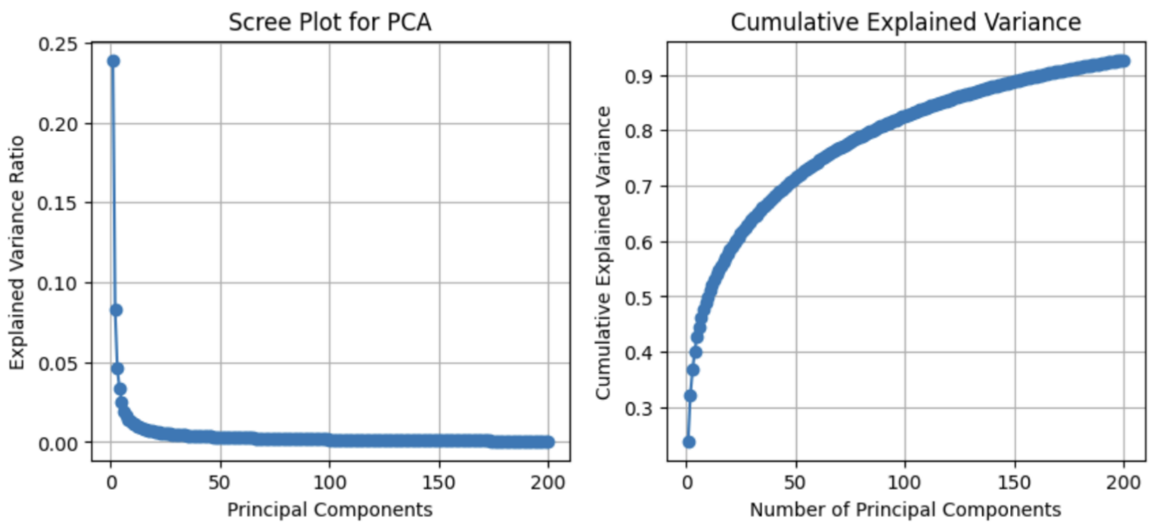


Figure 6-2: Scree Plot and Cumulative Explained Variance Plot

During MLP training phase, the values of two parameters (optimizer and learning rate) are pending and the best combinations of them will be selected according to model that results in the smallest SMAPE. The basic structures of MLP model used in the present project are given below:

Table 6-1 Structure of MLP Used

Activation Function	Rectified Linear Unit	**Number of Epochs**	1000

within Each Layer	(ReLU)		
Number of Hidden Layers	1	**Early Stopping**	Yes, patience = 10
Loss	Mean Absolute Error (MAE)	**Batch Size**	32

The best parameters for four labels and the related three evaluation metrices (MSE, SMAPE and R2) are summarized as follows:

Table 6-2 Optimal Parameters for MLP.

Label	Optimizer	Learning Rate	MSE	SMAPE	R2
updrs_1	RMSprop	0.01	20.8074	60.7825	0.3000
updrs_2	RMSprop	0.05	17.1805	88.7718	0.4249
updrs_3	Adam	0.005	104.3269	80.7892	0.5107
updrs_4	Adam	0.0001	9.2088	155.2564	0.1997

The change of training MAE and validation MAE according to epochs are visualized as follows:

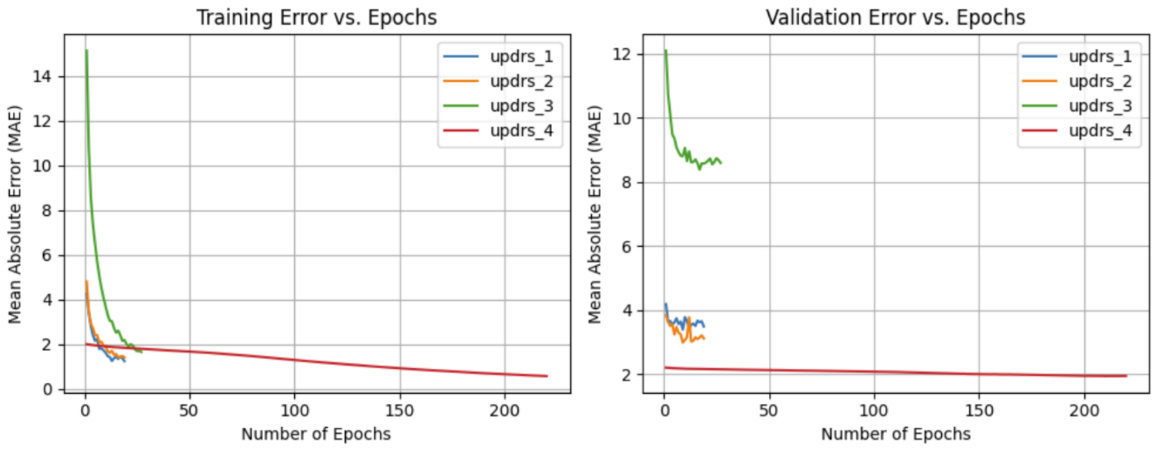


Figure 6-3: MAE vs Epochs

With respect to training errors, all four curves representing four labels show obvious convergence trend with the *updrs_4* curve being much more moderate, nearly horizontal to the x-axis (because the initial MAE is the smallest, being approximately 2 at epoch 0). On the other hand, the validation losses of *updrs_1*, *updrs_2* and *updrs_3* gradually converge, despite the fluctuation observed. The convergence of *updrs_4* is not evident, being almost horizontal which is analogous to the training loss.
Likewise, during deep learning model training phase, the values of two parameters (optimizer and learning rate) are pending and the best combinations of them will be selected according to model that results in the smallest SMAPE. The basic structures of deep learning model used in the present project are given below (slightly different compared to MLP):

Table 6-3 Structure of Deep Learning Model Used

Activation Function within Each Layer	Rectified Linear Unit (ReLU)	**Number of Epochs**	1000
Number of Hidden Layers	5	**Early Stopping**	Yes, patience = 10
Loss	Mean Absolute Error (MAE)	**Batch Size**	32

The best parameters for four labels and the related three evaluation metrices (MSE, SMAPE and R2) are summarized as follows:

Table 6-4 Optimal Parameters for Deep Learning Model.

Label	Optimizer	Learning Rate	MSE	SMAPE	R2
updrs_1	SGD	0.05	20.8764	59.5199	0.3394
updrs_2	SGD	0.05	17.2270	87.2999	0.4233
updrs_3	SGD	0.01	116.2129	78.9238	0.4550
updrs_4	RMSprop	0.005	8.6791	162.6440	0.2457

The change of training MAE and validation MAE according to epochs are visualized as follows:

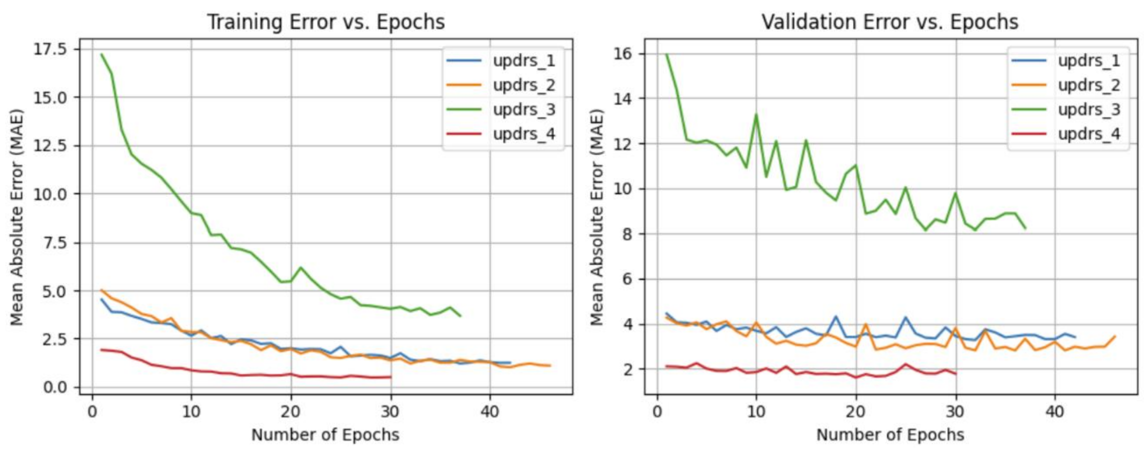


Figure 6-4 MAE vs Epochs

It can be seen obvious convergence trend for all four training loss curves and the validation loss curve for label *updrs_3* has the most evident convergence trend for this deep learning model. Other three validation curves fluctuate and are not evidently converging, with the validation MAE for label *updrs_4* only changes a very little amount during the whole training.

7. Model comparison and conclusion

In this section, results of different models are compared based on our chosen evaluation metrices including SMAPE as well as MSE with SMAPE being the primary metric. These models include: logistic regression, XGBoost, MLP, deep learning and SVM. The SMAPE values of five models are summarized as follows:

Table 7-1: The SMAPE values of five models

Model	updr_1	updr_2	updr_3	updr_4	Average
Logistic Regression	81.3205	81.4714	80.3427	67.8049	77.8680
XGBoost	68.1128	88.3873	86.5875	150.4355	98.3808
MLP	60.7825	88.7718	80.7892	155.2564	96.4000
Deep Learning	59.5199	87.2999	78.9238	162.6440	97.0969
SVM	63.2562	92.4404	82.5338	150.9469	97.2943

The Random forest model used in the example has a SMAPE value of 98.7369, hence all of our models perform better than it.

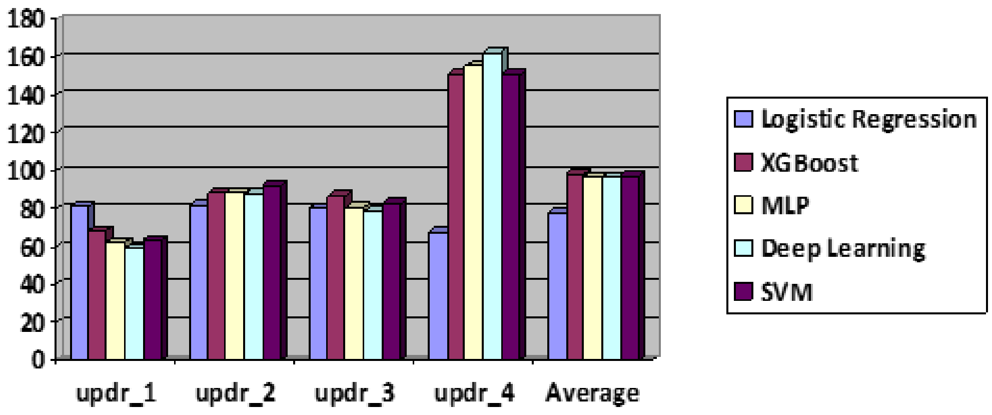


Figure 7-1: Bar chart of sMAPE values for five models

From this graph, it can be seen that SMAPE value of all four labels of model logistic regression model are relatively low. When it comes to *updr_4*, this model evidently outperforms other three models. Additionally, when it comes to average SMAPE, logistic regression model also has the lowest value. Therefore, logistic regression is considered to be the model with the best performance.
The MSE values of five models are summarized as follows:

Table 7-2: The MSE values of five models

Model	updr_1	updr_2	updr_3	updr_4	Average
Logistic Regression	39.5234	42.5888	219.4245	10.7456	78.0706

XGBoost	24.8721	26.9640	139.1962	13.2455	51.0695
MLP	20.8074	17.1805	104.3269	9.2088	37.8809
Deep Learning	20.8764	17.2270	116.2129	8.6791	40.7489
SVM	19.1339	22.4127	165.6939	4.5905	52.9578

The Random forest model used in the example has a MSE value of 60.0534, all of our models has a lower MSE value than it except Logistic Regression model.

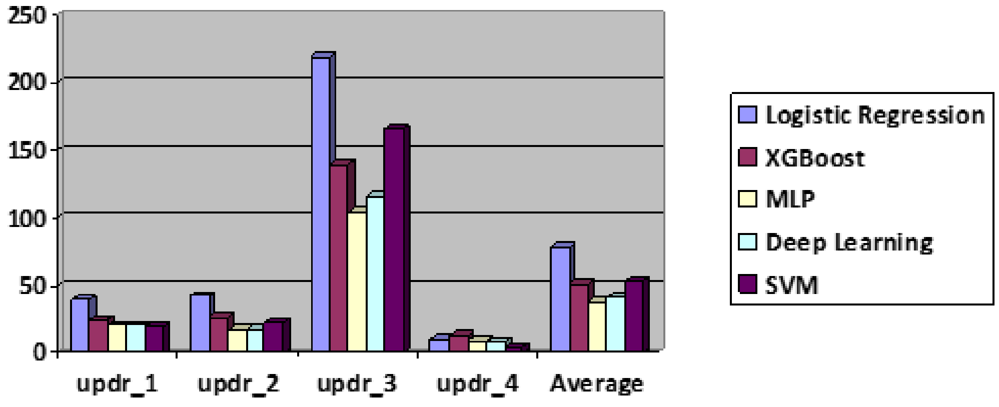


Figure 7-2 Bar chart of MSE values for five models

Interestingly, it can be observed that although logistic regression has the best performance regarding the SMAPE value, when it comes to MSE, it has the highest value among all models. The model with the lowest MSE is the MLP model. However, since the present competition requires that SMAPE should be as small as possible, therefore, we still choose the logistic regression model as the desired model.

8. Reference

[1] https://www.kaggle.com/code/gusthema/parkinson-s-disease-progression-prediction-w-tfdf

[2] Hilbe, Joseph M. *Logistic regression models*. CRC press, 2009.

[3] sklearn. linear_model. LogisticRegression—. "scikit-learn 0.24. 2 documentation."

[4] Cristianini, N., Ricci, E. (2008). Support Vector Machines. In: Kao, MY. (eds) Encyclopedia of Algorithms. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30162-4_415

[5] https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

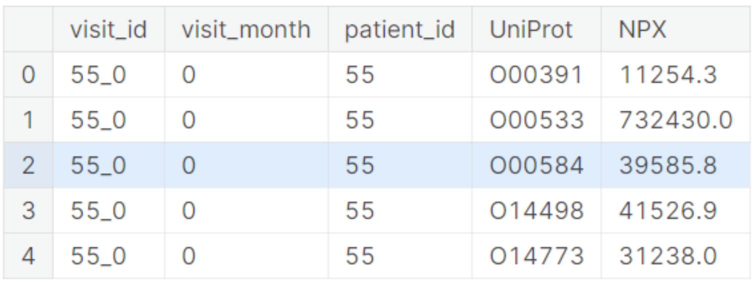
[6] Chen, Tianqi, and Carlos Guestrin. "Xgboost: A scalable tree boosting system." *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016.

[7] Bergstra, James, Dan Yamins, and David D. Cox. "Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms." *Proceedings of the 12th Python in science conference*. Vol. 13. 2013.

[8] aykin, Simon. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1998.

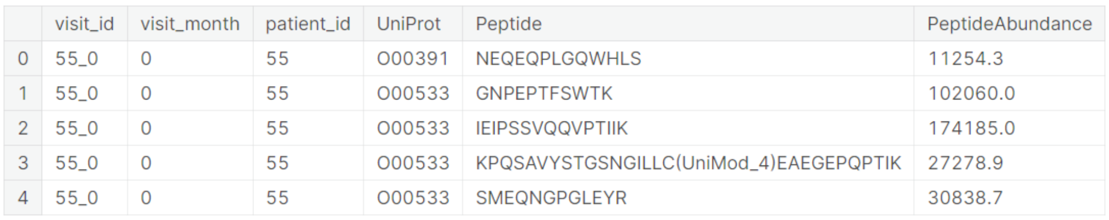
9. Appendix

the first 5 entries of the protein dataset and the peptide dataset:



	visit_id	visit_month	patient_id	UniProt	NPX
0	55_0	0	55	O00391	11254.3
1	55_0	0	55	O00533	732430.0
2	55_0	0	55	O00584	39585.8
3	55_0	0	55	O14498	41526.9
4	55_0	0	55	O14773	31238.0

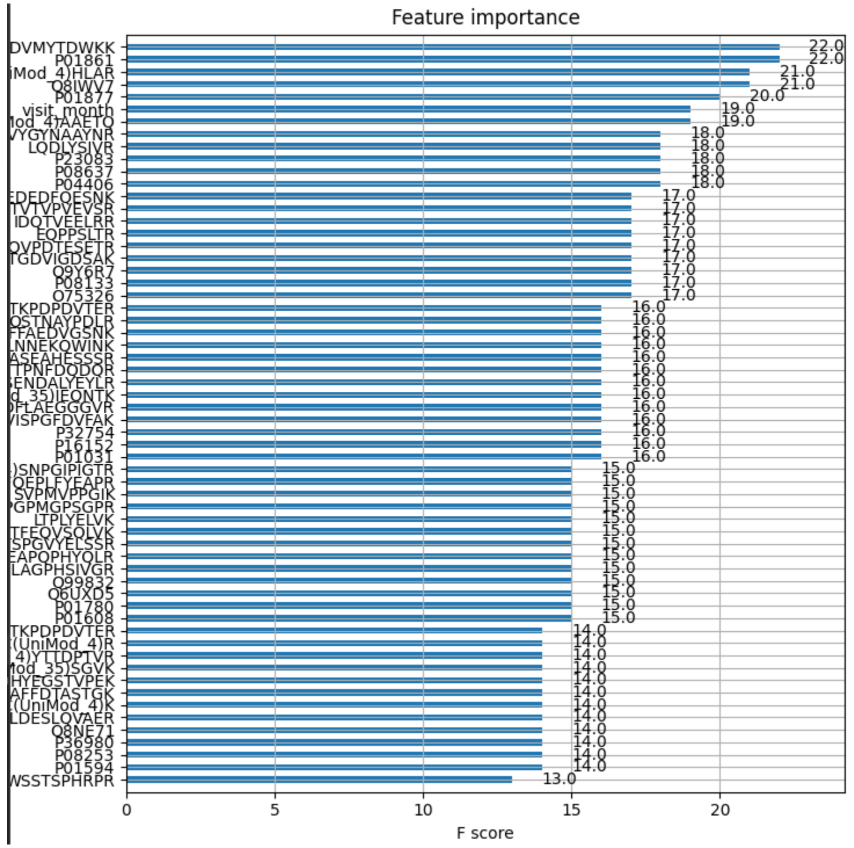
Figure 9-1: the first 5 entries of the protein dataset



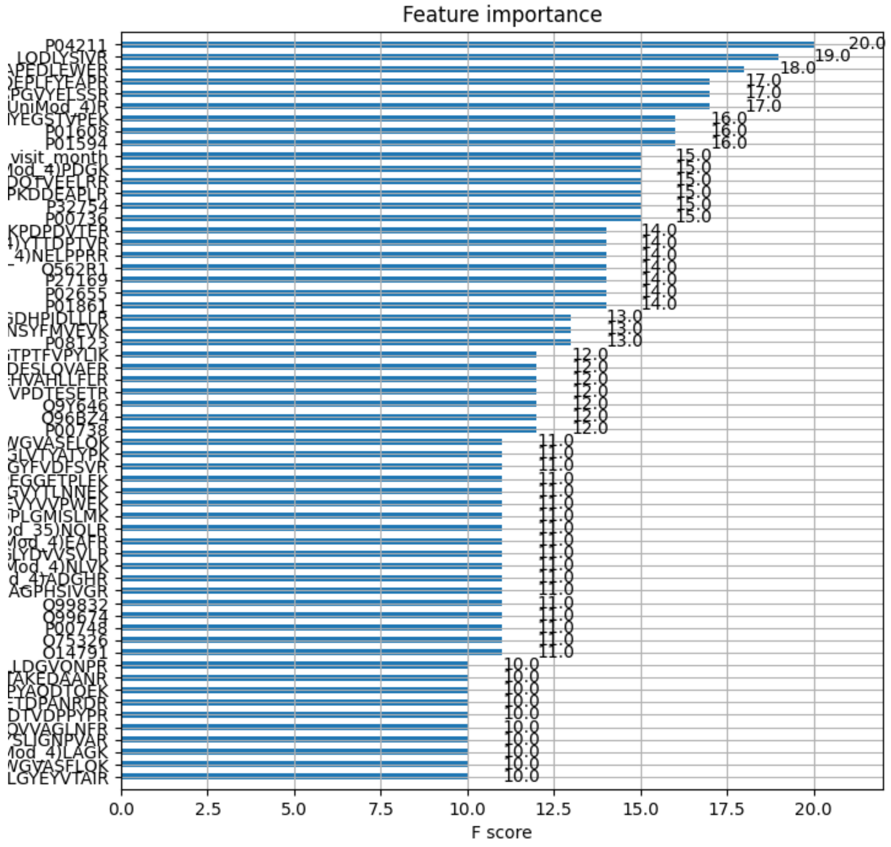
	visit_id	visit_month	patient_id	UniProt	Peptide	PeptideAbundance
0	55_0	0	55	O00391	NEQEQPLGQWHLS	11254.3
1	55_0	0	55	O00533	GNPEPTFSWTK	102060.0
2	55_0	0	55	O00533	IEIPSSVQQVPTIIK	174185.0
3	55_0	0	55	O00533	KPQSAVYSTGSNGILLC(UniMod_4)EAEGEPQPTIK	27278.9
4	55_0	0	55	O00533	SMEQNGPGLEYR	30838.7

Figure 9-2: the first 5 entries of the peptide dataset

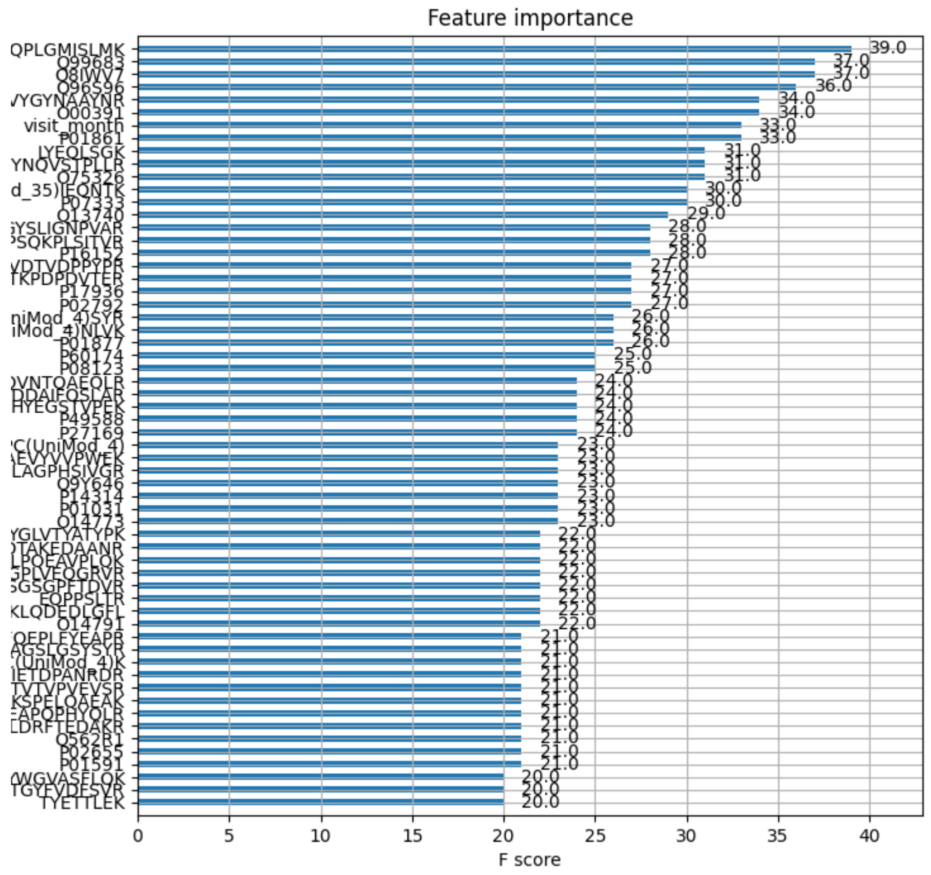
Feature importance ranking of updrs_1,updrs_2 and updrs_3:



(a) Feature importance ranking of updrs_4



(b) Feature importance ranking of updrs_4



(C) Feature importance ranking of updrs_1

Figure 9-3: Feature importance ranking of updrs_1,updrs_2 and updrs_3