IT258

Advanced Processing

ASSIGNMENT 3

211AI016, 211AI018

TEAM 18 | Dhiraj Choudhary D, Venkata Ravi Ram G

We need to perform advanced analysis processes like :

- *Correlation analysis*
- *Covariance analysis*
- *Dimensionality Reduction*
- *Feature engineering.*

Specifications of dataset –

➢ **train_clinical_data.csv**

- visit_id - ID code for the visit.
- visit month - The month of the visit, relative to the first visit by the patient.
- patient_id - An ID code for the patient.
- updrs_[1-4] - The patient's score for part N of the Unified Parkinson's Disease Rating Scale. Higher numbers indicate more severe symptoms. Each sub-section covers a distinct category of symptoms, such as mood and behavior for Part 1 and motor functions for Part 3.
- upd23b_clinical_state_on_medication - Whether or not the patient was taking medication such as Levodopa during the UPDRS assessment. Expected to mainly affect the scores for Part 3 (motor function). These medications wear off fairly quickly (on the order of one day) so it's common for patients to take the motor function exam twice in a single month, both with and without medication.

➢ **supplemental_clinical_data.csv** Clinical records without any associated CSF samples. This data is intended to provide additional context about the typical progression of Parkinsons. Uses the same columns as **train_clinical_data.csv**.

This dataset cannot be used in making our prediction but just to get additional insights on the trends of the clinical data hence we are not cleaning this dataset as it doesn't contain the patients' peptide and protein value obtained from their CSF tests.

The clinical data and supplemental clinical data have been merged in order to observe the trends of updrs values. Viz; correlation among target values.

```
df_clinic = []
tmp = pd.read_csv("/kaggle/input/amp-parkinsons-disease-progression-prediction/train_clinical_data.csv")
tmp["CSF"] = 1
df_clinic.append(tmp)
tmp = pd.read_csv("/kaggle/input/amp-parkinsons-disease-progression-prediction/supplemental_clinical_data.c
tmp["CSF"] = 0
df_clinic.append(tmp)
df_clinic = pd.concat(df_clinic, axis=0).reset_index(drop=True)
df_clinic = df_clinic.rename(columns={"upd23b_clinical_state_on_medication": "medication"})
```

```
df_clinic
```

	visit_id	patient_id	visit_month	updrs_1	updrs_2	updrs_3	updrs_4	medication	CSF
0	55_0	55	0	10.0	6.0	15.0	NaN	NaN	1
1	55_3	55	3	10.0	7.0	25.0	NaN	NaN	1
2	55_6	55	6	8.0	10.0	34.0	NaN	NaN	1
3	55_9	55	9	8.0	9.0	30.0	0.0	On	1
4	55_12	55	12	10.0	10.0	41.0	0.0	On	1
...	...	...	...	...	...	...	...	...	...
4833	65382_0	65382	0	NaN	NaN	0.0	NaN	NaN	0
4834	65405_0	65405	0	5.0	16.0	31.0	0.0	NaN	0
4835	65405_5	65405	5	NaN	NaN	57.0	NaN	NaN	0
4836	65530_0	65530	0	10.0	6.0	24.0	0.0	NaN	0
4837	65530_36	65530	36	8.0	4.0	15.0	4.0	On	0

4838 rows × 9 columns

- **train_peptides.csv** Mass spectrometry data at the peptide level. Peptides are the component subunits of proteins.
  - visit_id - ID code for the visit.
  - visit_month - The month of the visit, relative to the first visit by the patient.
  - patient_id - An ID code for the patient.
  - UniProt - The UniProt ID code for the associated protein. There are often several peptides per protein.
  - Peptide - The sequence of amino acids included in the peptide. See this table for the relevant codes. Some rare annotations may not be included in the table. The test set may include peptides not found in the train set.
  - PeptideAbundance - The frequency of the amino acid in the sample.

```
train_peptides = pd.read_csv("/kaggle/input/amp-parkinsons-disease-progression-prediction/train_pe
train_peptides
```

	visit_id	visit_month	patient_id	UniProt	Peptide	PeptideAbundance
0	55_0	0	55	O00391	NEQEQPLGQWHLS	11254.30
1	55_0	0	55	O00533	GNPEPTFSWTK	102060.00
2	55_0	0	55	O00533	IEIPSSVQQVPTIIK	174185.00
3	55_0	0	55	O00533	KPQSAVYSTGSNGILLC(UniMod_4)EAEGEPQPTIK	27278.90
4	55_0	0	55	O00533	SMEQNGPGLEYR	30838.70
...	...	...	...	...	...	...
981829	58648_108	108	58648	Q9UHG2	ILAGSADSEGVAAPR	202820.00
981830	58648_108	108	58648	Q9UKV8	SGNIPAGTTVDTK	105830.00
981831	58648_108	108	58648	Q9Y646	LALLVDTVGPR	21257.60
981832	58648_108	108	58648	Q9Y6R7	AGC(UniMod_4)VAESTAVC(UniMod_4)R	5127.26
981833	58648_108	108	58648	Q9Y6R7	GATTSPGVYELSSR	12825.90

981834 rows × 6 columns

```
all(train_proteins[['visit_id', 'UniProt']].value_counts() == 1)
```

```
True
```

➢ **train_proteins.csv** Protein expression frequencies aggregated from the peptide level data.

- `visit_id` - ID code for the visit.
- `visit_month` - The month of the visit, relative to the first visit by the patient.
- `patient_id` - An ID code for the patient.
- `UniProt` - The UniProt ID code for the associated protein. There are often several peptides per protein. The test set may include proteins not found in the train set.
- `NPX` - Normalized protein expression. The frequency of the protein's occurrence in the sample. May not have a 1:1 relationship with the component peptides as some proteins contain repeated copies of a given peptide.

```
[23]: train_proteins = pd.read_csv("/kaggle/input/amp-parkinsons-disease-progression-prediction/train_pr
      train_proteins
```

[23]:

	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
...	...	...	...	...	...
232736	58648_108	108	58648	Q9UBX5	27387.8
232737	58648_108	108	58648	Q9UHG2	369437.0
232738	58648_108	108	58648	Q9UKV8	105830.0
232739	58648_108	108	58648	Q9Y646	21257.6
232740	58648_108	108	58648	Q9Y6R7	17953.1

232741 rows × 5 columns

+ Code + Markdown

```
all(train_proteins[['visit_id', 'UniProt']].value_counts() == 1)
```

[25]: True

Correlation Analysis

Correlation analysis is a statistical technique used to examine the relationship between two or more variables. It measures the strength and direction of the association between variables, indicating whether they are positively or negatively correlated. By calculating a correlation coefficient, such as the Pearson's correlation coefficient, it quantifies the degree of linear dependence between variables.

Correlation analysis helps to understand the pattern of relationship between variables and provides insights into how changes in one variable might affect another. It is widely applied in various fields, including economics, social sciences, and data analysis, to explore connections, make predictions, and uncover meaningful patterns in data.

Initially, the correlation between updrs values was calculated In order to observe how strongly change in one variable alters change in the other.

```
x = df_clinic['updrs_1']
y = df_clinic['updrs_2']
z = df_clinic['updrs_3']
```

[52] ✓ 0.1s

```
print("the pearson correlation between updrs_1 and updrs_2 values:")
st.pearsonr(x,y)[0]
```

[53] ✓ 0.0s

```
the pearson correlation between updrs_1 and updrs_2 values:

0.6123822155904514
```

```
print("the pearson correlation between updrs_2 and updrs_3 values:")
st.pearsonr(y,z)[0]
```

[34] ✓ 0.1s

```
the pearson correlation between updrs_2 and updrs_3 values:

0.5302471946187702
```

```
print("the pearson correlation between updrs_1 and updrs_3 values:")
st.pearsonr(x,z)[0]
```

[35] ✓ 0.0s

```
the pearson correlation between updrs_1 and updrs_3 values:

0.2546524247892682
```

Clearly, as we observed in the EDA phase, updrs_2 is co-related to updrs 1 and updrs 3 wheras, updrs 1 and updrs 3 are too weakly correlated.

Later on we calculated the correlation between npx and UniProt values which we fount out to be so weakly negative.

```
x = train_proteins['UniProt']
y = train_proteins['NPX']
print("the spearman correlation between UniProt and NPX values:")
st.spearmanr(x,y).correlation
```

[38] ✓ 0.6s

```
the spearman correlation between UniProt and NPX values:

-0.28635903235123744
```

The same was observed between peptide abundance and UniProt values. Hence uniport is too weakly correlated to any of npx or abundancies.

Whereas, Peptide and peptide abundancies showed somewhat better correlation than all these.

```
x = train_peptides['UniProt']
y = train_peptides['Peptide']
z = train_peptides['PeptideAbundance']
print("the spearman correlation between UniProt and PeptideAbundance values:")
st.spearmanr(x,z).correlation
```

[41] ✓ 3.7s

```
the spearman correlation between UniProt and PeptideAbundance values:

-0.1598701337706347
```

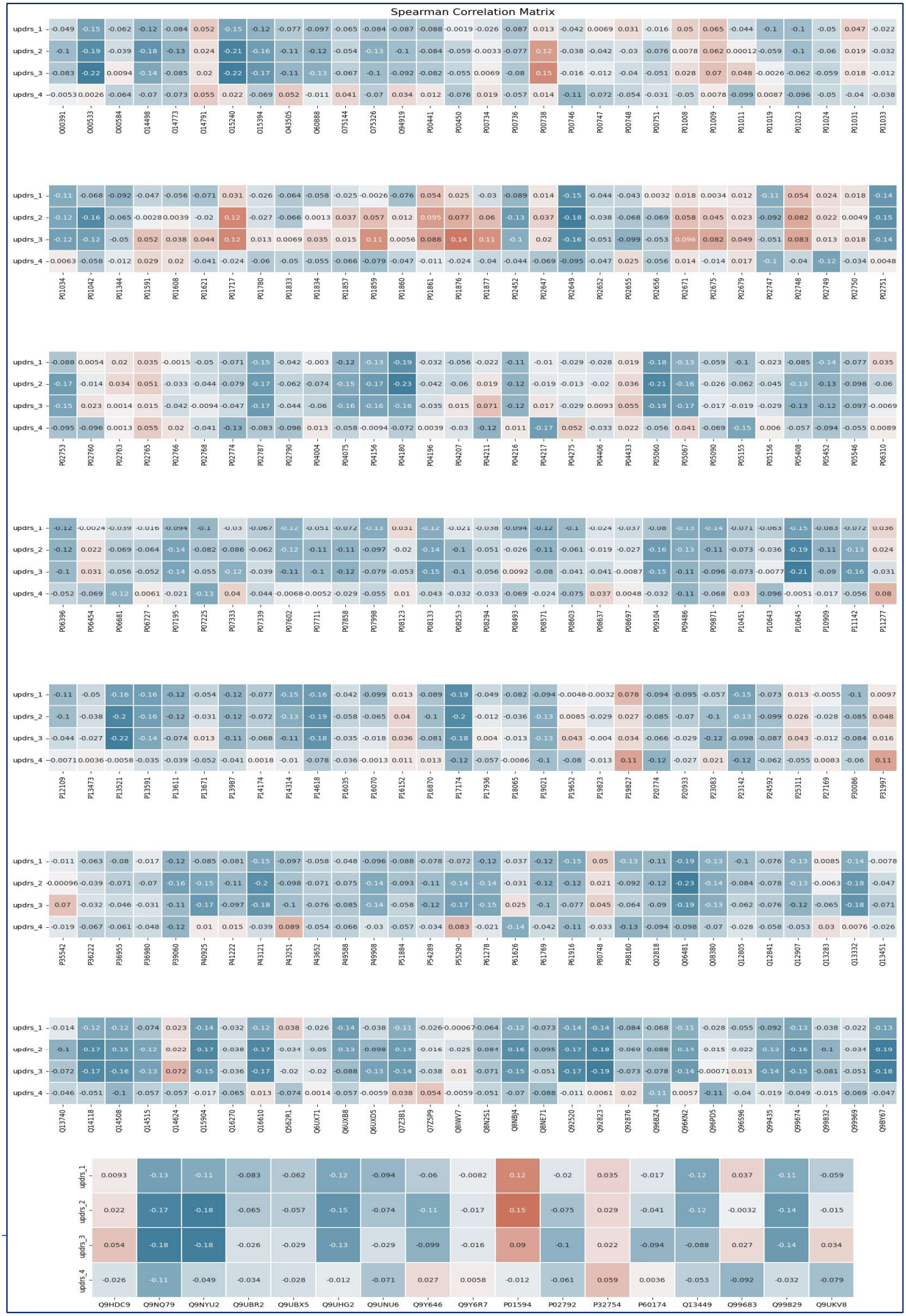
```
print("the spearman correlation between Peptide and PeptideAbundance values:")
st.spearmanr(y,z).correlation
```

[42] ✓ 4.0s

```
the spearman correlation between Peptide and PeptideAbundance values:

0.031790835174999116
```

Later we considered co-relation between all protein abundancies and updrs values and constructed matrix in order to visualize which is as shown below:


Spearman Correlation Matrix
updrs_1
updrs_2
updrs_3
updrs_4
O00391 O00533 O00584 O14498 O14773 O14791 O15240 O15394 O43505 O60888 O75144 O75326 O94919 P00441 P00450 P00734 P00736 P00738 P00746 P00747 P00748 P00751 P01008 P01009 P01011 P01019 P01023 P01024 P01031 P01033
P01034 P01042 P01344 P01591 P01608 P01621 P01717 P01780 P01833 P01834 P01857 P01859 P01860 P01861 P01876 P01877 P02452 P02647 P02649 P02652 P02655 P02656 P02671 P02675 P02679 P02747 P02748 P02749 P02750 P02751
P02753 P02760 P02763 P02765 P02766 P02768 P02774 P02787 P02790 P04004 P04075 P04156 P04180 P04196 P04207 P04211 P04216 P04217 P04275 P04406 P04433 P05060 P05067 P05090 P05155 P05156 P05408 P05452 P05546 P06310
P06396 P06454 P06681 P06727 P07195 P07225 P07333 P07339 P07602 P07711 P07858 P07998 P08123 P08133 P08253 P08294 P08493 P08571 P08603 P08637 P08697 P09104 P09486 P09871 P10451 P10643 P10645 P10909 P11142 P11277
P12109 P13473 P13521 P13591 P13611 P13671 P13987 P14174 P14314 P14618 P16035 P16070 P16152 P16870 P17174 P17936 P18065 P19021 P19652 P19823 P19827 P20774 P20933 P23083 P23142 P24592 P25311 P27169 P30086 P31997
P35542 P36222 P36955 P36980 P39060 P40925 P41222 P43121 P43251 P43652 P49588 P49908 P51884 P54289 P55290 P61278 P61626 P61769 P61916 P80748 P98160 Q02818 Q06481 Q08380 Q12805 Q12841 Q12907 Q13283 Q13332 Q13451
Q13740 Q14118 Q14508 Q14515 Q14624 Q15904 Q16270 Q16610 Q562R1 Q6UX71 Q6UX88 Q6UXD5 Q7Z3B1 Q7Z5P9 Q8IWV7 Q8N251 Q8NBJ4 Q8NE71 Q92520 Q92823 Q92876 Q96BZ4 Q96KN2 Q96PD5 Q96S96 Q99435 Q99674 Q99832 Q99969 Q9BY67
Q9HDC9 Q9NQ79 Q9NYU2 Q9UBR2 Q9UBX5 Q9UHG2 Q9UNU6 Q9Y646 Q9Y6R7 P01594 P02792 P32754 P60174 Q13449 Q99683 Q99829 Q9UKV8

Inferences drawn:

- There are a lot of proteins to examine with the correlation matrix. Let's start by defining what we would consider to be a somewhat significant correlation (positive or negative). Values that are 0.1 or below are likely to have little correlation to the UPDRS target scores, and are likely just noise.
- A quick scan reveals that there are several candidates that may not be useful in our regression: O00533,O14498,O15240,O15394,O43505,O60888,P00738,P01034,P01042,P01717,P02452,P02649,P02751,P02753,P02787,P04075,P04156…….
- There are some proteins that are weak correlates only to updrs_4. These are: P00746,P02749,,P02774,P04211,P04217,P05155,P06681,P19827,P20774,P31997,P61626,Q96BZ4,Q96PD5

Covariance Analysis

Covariance analysis, also known as covariance matrix analysis or covariance structure analysis, is a statistical method used to examine the relationships and dependencies among multiple variables. It focuses on estimating and analyzing the covariance matrix, which measures the co-variability between pairs of variables in a dataset. Covariance analysis provides insights into the strength and direction of the linear relationship between variables, allowing researchers to understand the interconnections and patterns within the data.

It is commonly used in fields such as finance, social sciences, and psychology to investigate complex relationships and determine the extent to which variables vary together. Additionally, covariance analysis is a fundamental tool in multivariate analysis, where it plays a crucial role in assessing the fit of statistical models and testing hypotheses about the relationships between variables.

Covariance between updrs values were observed(except updrs4)

	updrs_1	updrs_2	updrs_3
updrs_1	31.134410	20.215184	17.969788
updrs_2	20.215184	35.000127	39.672358
updrs_3	17.969788	39.672358	159.937257

Base to compare results after Dimensionality reduction and feature engineering

Linear Regression Model without Features Selection

- At first, we try to create, train, and evaluate linear regression models with all the peptides as independent variables.
- This time, we will predict updrs_1 (y_1), updrs_2 (y_2), updrs_3 (y_3), and updrs_4 (y_4) separately from the other columns as independent variables.

Finally these were the results we got.

The Results without Features Selection

Target	MSE	MAE	R2	SMAPE
UPDRS 1	181.345190	8.976894	-7.242988	123.677741
UPDRS 2	141.033834	7.714924	-2.961842	119.039553
UPDRS 3	886.743183	21.082013	-2.568938	115.794226
UPDRS 4	11.084991	2.387561	-0.501428	122.636215

- To evaluate the results of the linear regression model, we can look at the mean squared error (MSE), mean absolute error (MAE), R-squared (R2), and symmetric mean absolute percentage error (SMAPE) for each of the four UPDRS scores (UPDRS 1-4).
- Generally, a MSE, MAE, or SMAPE value of 0 indicates a perfect performance of the model, while higher values indicate a worse fit. A R2 value of 1 indicates a perfect fit, while lower values indicate a worse fit.
- It seems that the metrics, such as the SMAPE values are considerably high. This could indicate that there are large differences between the predicted values and the true values.

Feature Engineering

Feature engineering is a critical process in machine learning that involves transforming and selecting relevant features from raw data to improve the performance and efficiency of models. Feature selection, a key component of feature engineering, aims to identify the most informative and discriminative features that contribute the most to the target variable while discarding irrelevant or redundant ones. By reducing the dimensionality of the feature space, feature selection not only enhances computational efficiency but also helps to alleviate the curse of dimensionality and mitigate overfitting.

Various feature selection techniques exist, including filter methods that evaluate features based on statistical measures, wrapper methods that utilize the performance of a specific model, and embedded methods that incorporate feature selection within the model training process. Effective feature selection not only simplifies the model but also enhances its interpretability, generalization capability, and predictive accuracy, enabling better decision-making and insights from the data.

Feature Selection

There are several techniques we can use to select features from a large number of independent variables:

1. **Univariate Feature Selection**: This method selects the features with the highest correlation with the target variable using statistical tests like chi-squared test, ANOVA F-test, mutual information, etc.

2. **Recursive Feature Elimination**: This method recursively removes features from the dataset and selects the features that contribute the most to the model's accuracy.

3. **Principal Component Analysis (PCA)**: PCA is a dimensionality reduction technique that transforms the original features into a new set of uncorrelated features called principal components. We can select the top principal components that explain the majority of the variance in the data.

4. **Regularization Methods**: Lasso and Ridge regression are two popular regularization methods that shrink the coefficients of the less important features to zero, leaving only the most important features in the model.

5. **Tree-Based Methods**: Tree-based models like Random Forest and XGBoost can be used to rank the importance of the features based on their contribution to the model's accuracy.

We can also combine multiple feature selection techniques to get a more accurate and robust feature set.

Implementation of few feature selection methods :

- **Linear Regression Model with Univariate Feature Selection**
  - To perform Univariate Feature Selection, we can use the SelectKBest class from the scikit-learn library.
  - We are using the F-test score (f_regression) as the scoring function to rank the features. We selected the top 10 features based on this score (k=10). Once we fit the selector on the independent variables and target variable, we can get the indices and names of the selected features using the get_support and columns methods, respectively.

```
#·Add·a·title·to·the·DataFrame.
print("The·Results·with·Univariate·Feature·Selection")

#·Create·a·dictionary·with·the·metrics·for·each·target.
metrics_dict_KBest·=·{
····'Target':·['UPDRS·1',·'UPDRS·2',·'UPDRS·3',·'UPDRS·4'],
····'MSE':·[mse_updrs1,·mse_updrs2,·mse_updrs3,·mse_updrs4],
····'MAE':·[mae_updrs1,·mae_updrs2,·mae_updrs3,·mae_updrs4],
····'R2':·[r2_updrs1,·r2_updrs2,·r2_updrs3,·r2_updrs4],
····'SMAPE':·[smape(y_test_updrs1,·y_pred_updrs1),·smape(y_test_updrs2,·y_pred_updrs2),·
·····smape(y_test_updrs3,·y_pred_updrs3),·smape(y_test_updrs4,·y_pred_updrs4)]
}

#·Create·a·Pandas·DataFrame·from·the·dictionary.
metrics_df_KBest·=·pd.DataFrame(metrics_dict_KBest)

#·Set·the·'Target'·column·as·the·index.
metrics_df_KBest.set_index('Target',·inplace·=·True)

#·Display·the·DataFrame.
metrics_df_KBest
```

```
[ ]
```

```
The Results with Univariate Feature Selection
```

Target	MSE	MAE	R2	SMAPE
UPDRS 1	22.051638	3.902138	-0.002350	74.352861
UPDRS 2	34.647653	4.738632	0.026698	101.986585
UPDRS 3	237.379871	12.838956	0.044601	96.618764
UPDRS 4	7.907121	2.161096	-0.070995	148.004627

We could see better results except for SMAPE for updrs_4 (y_4) by Univariate Feature Selection than those without features selection.

- **Linear Regression Model with Recursive Feature Elimination (RFE)**
  - Recursive Feature Elimination (RFE) is a method to select the best features by recursively considering smaller and smaller subsets of features. In each iteration, the model is trained on the remaining features and the feature with the lowest importance is removed.
  - To add RFE to the linear regression model, we can use the RFE class from scikit-learn.
  - Here, n_features_to_select is the number of features to select and step is the number of features to remove at each iteration. The selector.transform method selects only the selected features from the training and testing data, and the linear regression model is fit on the selected features. Finally, the performance of the model is evaluated on the selected features.

```
# Add a title to the DataFrame.
print("The Results with Recursive Feature Elimination")

# Create a dictionary with the metrics for each target.
metrics_dict_RFE = {
    'Target': ['UPDRS 1', 'UPDRS 2', 'UPDRS 3', 'UPDRS 4'],
    'MSE': [mse_updrs1, mse_updrs2, mse_updrs3, mse_updrs4],
    'MAE': [mae_updrs1, mae_updrs2, mae_updrs3, mae_updrs4],
    'R2': [r2_updrs1, r2_updrs2, r2_updrs3, r2_updrs4],
    'SMAPE': [smape(y_test_updrs1, y_pred_updrs1), smape(y_test_updrs2, y_pred_updrs2),
              smape(y_test_updrs3, y_pred_updrs3), smape(y_test_updrs4, y_pred_updrs4)]
}

# Create a Pandas DataFrame from the dictionary.
metrics_df_RFE = pd.DataFrame(metrics_dict_RFE)

# Set the 'Target' column as the index.
metrics_df_RFE.set_index('Target', inplace = True)

# Display the DataFrame.
metrics_df_RFE
```

```
[ ]
```

```
The Results with Recursive Feature Elimination
```

Target	MSE	MAE	R2	SMAPE
UPDRS 1	21.977045	3.894909	0.001040	74.143696
UPDRS 2	35.088510	4.768768	0.014314	102.209420
UPDRS 3	235.830139	12.762255	0.050838	96.059475
UPDRS 4	7.948161	2.158215	-0.076554	148.734648

- We could see better results except for SMAPE for updrs_4 (y_4) by Univariate Feature Selection than those without features selection. In addition, the results are similar to those of Univariate Feature selection.
- Contrary to Univariate Feature Selection, this RFE method does not guarantee to keep a specific variable, such as visit_month column. If this variable is eliminated, the prediction will be the same regardless of visit_month.

Dimensionality Reduction

Dimensionality reduction is a technique used to reduce the number of features or variables in a dataset while preserving the most relevant information. With the increasing complexity and size of data, dimensionality reduction methods offer valuable solutions for data analysis and machine learning tasks. By reducing the dimensionality, these methods simplify the dataset, alleviate computational burdens, and mitigate the risk of overfitting.

Popular approaches for dimensionality reduction include Principal Component Analysis (PCA), which transforms the data into a new set of uncorrelated variables called principal components, and t-SNE (t-distributed Stochastic Neighbor Embedding), which maps high-dimensional data into a lower-dimensional space while preserving local structure. Dimensionality reduction enables researchers and practitioners to gain insights, visualize data, and improve the performance and interpretability of models by focusing on the most important features that contribute to the variability in the data.

Now since we have too many proteins and peptides of which several have no impact or corelation with updrs values(target), we can create few meaningful features using PCA to reduce computational cost.

Linear Regression Model with Principal Component Analysis (PCA)

- To add Principal Component Analysis (PCA), we can use the PCA class from the sklearn.decomposition module. Here, we add PCA to Univariate Feature Selection.
- We first apply PCA to reduce the dimensionality of the data to 50 components, and then select the top 10 features with the highest F-values from the PCA-transformed data.The rest of the code remains the same. Note that we may need to experiment with different values of n_components to find the optimal number of components to use.
- Applying PCA on select-features.

```
# Add a title to the DataFrame.
print("The Results with Univariate Feature Selection and PCA")

# Display the DataFrame.
metrics_df_PCA
```

[168] ✓ 0.0s

```
The Results with Univariate Feature Selection and PCA
```

Target	MSE	MAE	R2	SMAPE
UPDRS 1	22.676545	3.974427	-0.030755	75.104070
UPDRS 2	34.699919	4.727209	0.025230	103.524073
UPDRS 3	244.659934	13.018261	0.015300	96.215940
UPDRS 4	6.716807	2.092563	0.090229	148.476482

```
# Add a title to the DataFrame.
print("The Results without Features Selection")

# comparison with the results without features selection
metrics_df_all
```

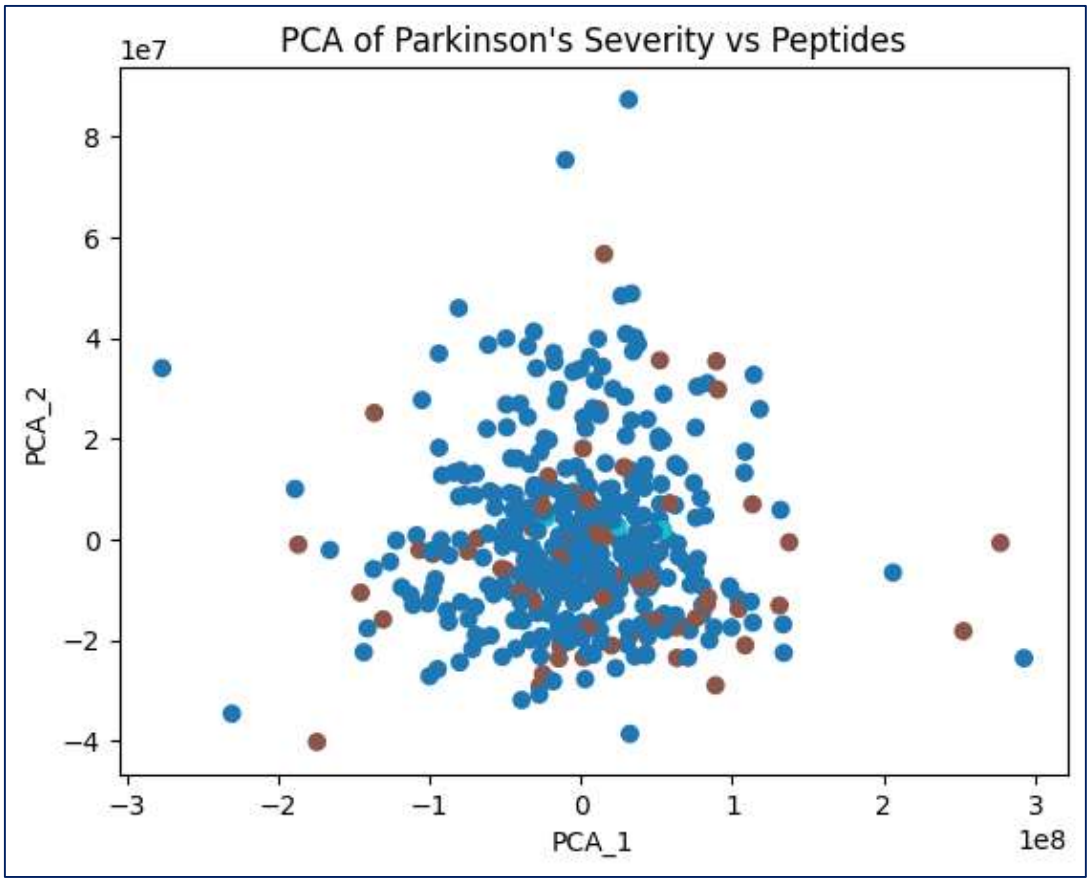
[164] ✓ 0.0s

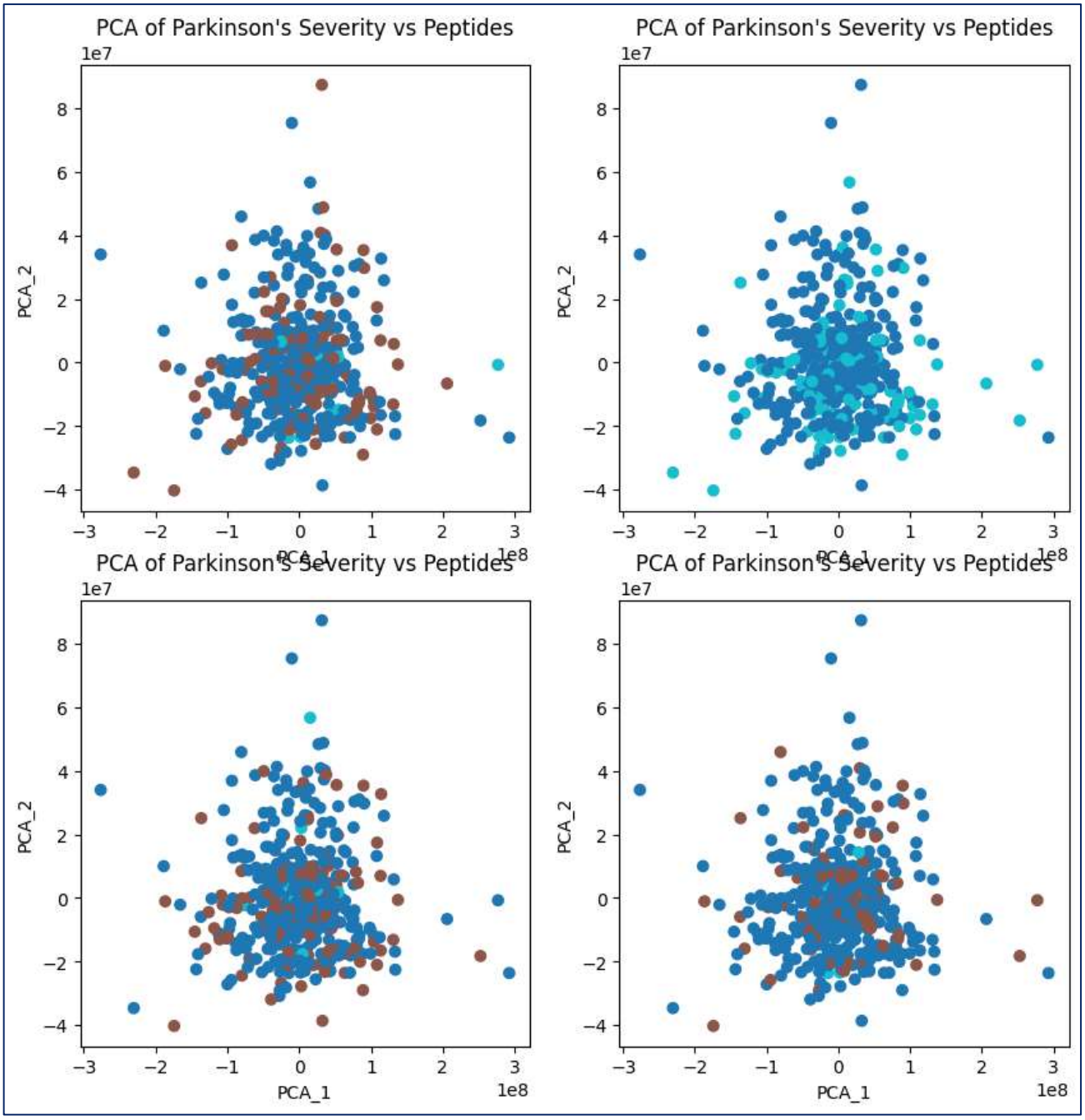
```
The Results without Features Selection
```

Target	MSE	MAE	R2	SMAPE
UPDRS 1	181.345190	8.976894	-7.242988	123.677741
UPDRS 2	141.033834	7.714924	-2.961842	119.039553
UPDRS 3	886.743183	21.082013	-2.568938	115.794226
UPDRS 4	11.084991	2.387561	-0.501428	122.636215

- We could see better results except for SMAPE for updrs_4 (y_4) by PCA and Univariate Feature Selection than those without features selection.
- In addition, the results are similar to those of Univariate Feature selection.

VISUALIZATION OF PCA:




PCA of Parkinson's Severity vs Peptides
1e7
PCA_2
PCA_1
1e8
PCA of Parkinson's Severity vs Peptides
1e7
PCA_2
PCA_1
1e8
PCA of Parkinson's Severity vs Peptides
1e7
PCA_2
PCA_1
1e8
PCA of Parkinson's Severity vs Peptides
1e7
PCA_2
PCA_1
1e8

Comparision of results / Conclusion

The Results without Features Selection

Target	MSE	MAE	R2	SMAPE
UPDRS 1	181.345190	8.976894	-7.242988	123.677741
UPDRS 2	141.033834	7.714924	-2.961842	119.039553
UPDRS 3	886.743183	21.082013	-2.568938	115.794226
UPDRS 4	11.084991	2.387561	-0.501428	122.636215

The Results with Univariate Feature Selection

Target	MSE	MAE	R2	SMAPE
UPDRS 1	22.051638	3.902138	-0.002350	74.352861
UPDRS 2	34.647653	4.738632	0.026698	101.986585
UPDRS 3	237.379871	12.838956	0.044601	96.618764
UPDRS 4	7.907121	2.161096	-0.070995	148.004627

The Results with Recursive Feature Elimination

Target	MSE	MAE	R2	SMAPE
UPDRS 1	21.977045	3.894909	0.001040	74.143696
UPDRS 2	35.088510	4.768768	0.014314	102.209420
UPDRS 3	235.830139	12.762255	0.050838	96.059475
UPDRS 4	7.948161	2.158215	-0.076554	148.734648

The Results with Univariate Feature Selection and PCA

Target	MSE	MAE	R2	SMAPE
UPDRS 1	22.681214	3.974792	-0.030967	75.107179
UPDRS 2	34.699918	4.727209	0.025230	103.524073
UPDRS 3	244.659632	13.018246	0.015301	96.215948
UPDRS 4	6.716946	2.092596	0.090210	148.477033