IT258

Data Pre-processing and Modelling

ASSIGNMENT 2

211AI016, 211AI018

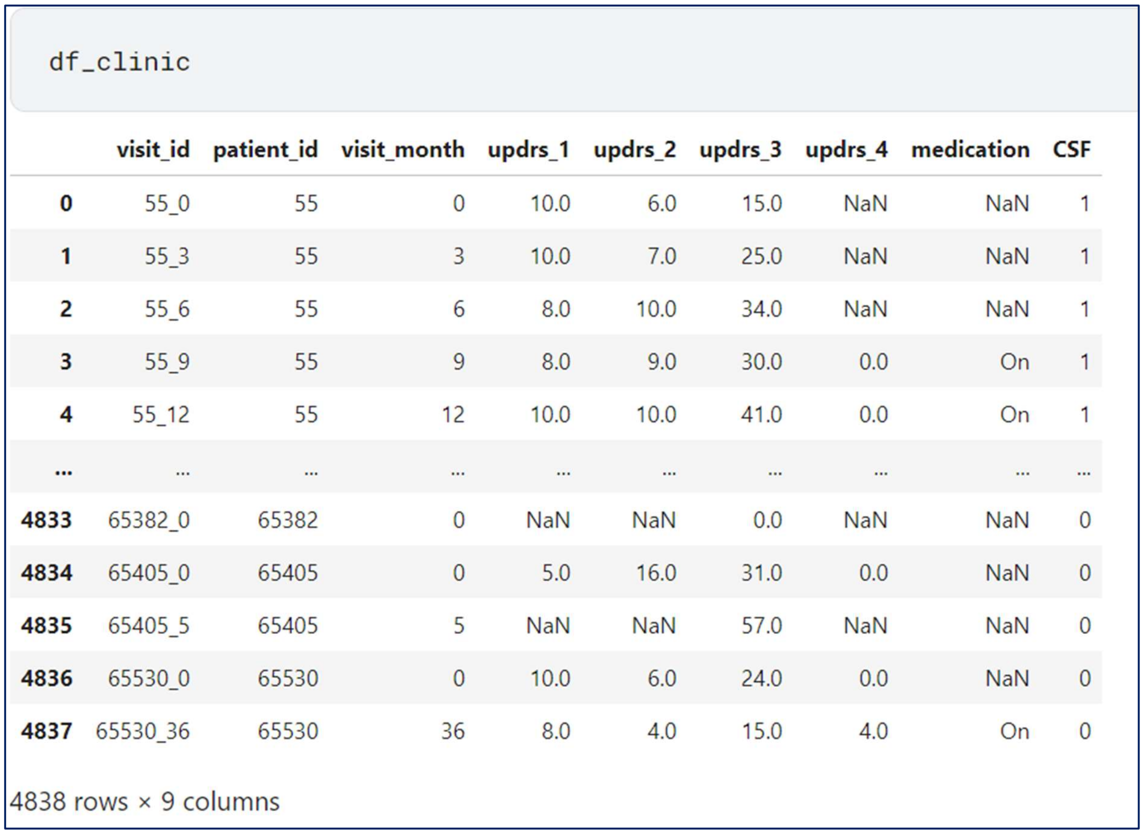
TEAM 18 | Dhiraj Choudhary D, Venkata Ravi Ram G

*We have 4 datasets **train_protiens** , **train_peptides** , **train_clinical** and **supplemental_clinical** data This report provides description of each dataset and how we have cleaned, preprocessed each 4 of the datasets and finally brought out single usable dataframe (modelling) which is ready for training the models.*

Cleaning and preprocessing of train_clinical_data.csv, supplemental_clinical_data.csv

Specifications of dataset -

- 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.



```
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

Preprocessing -

```
print(f'Unique Clinical Data patient #: {train_cd["patient_id"].nunique()}')
print("--------------------------------------------------------")
print(f'Null Values Found in Clinical Data:')
for col in train_cd.columns:
    print(f'Null values found in {col}: {train_cd[col].isna().sum()}')
print('')
```

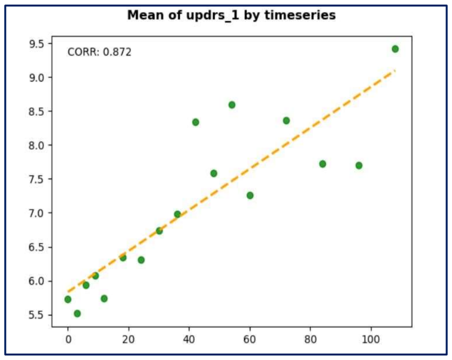
```
Unique Clinical Data patient #: 248
---------------------------------------------------------
Null Values Found in Clinical Data:
Null values found in visit_id: 0
Null values found in patient_id: 0
Null values found in visit_month: 0
Null values found in updrs_1: 1
Null values found in updrs_2: 2
Null values found in updrs_3: 25
Null values found in updrs_4: 1038
Null values found in upd23b_clinical_state_on_medication: 1327
```

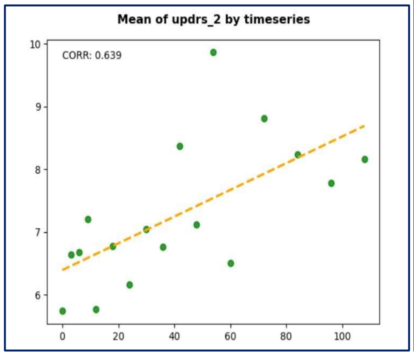
Handling NaN values :

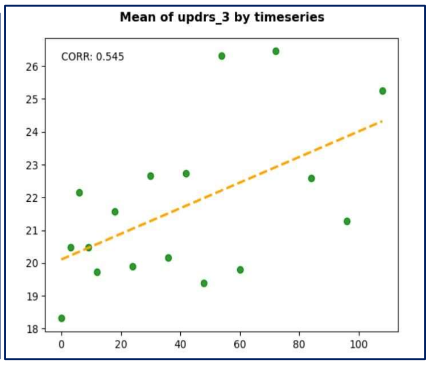
Clearly there are 1, 2 and 25 missing values of updrs_1, updrs_2 and updrs_3 respectively; which is comparatively too low than dataset size. Also in data exploration EDA phase we know that the updrs scores fit in a linear curve (as shown in below figures);

Therefore the most suitable method for updrs_1, updrs_2 and updrs_3 is imputing using linear interpolation

And since updrs_4 has almost half its entries as unknown and are MCAR type due to the patient not taking the test we just replaced the value with zero

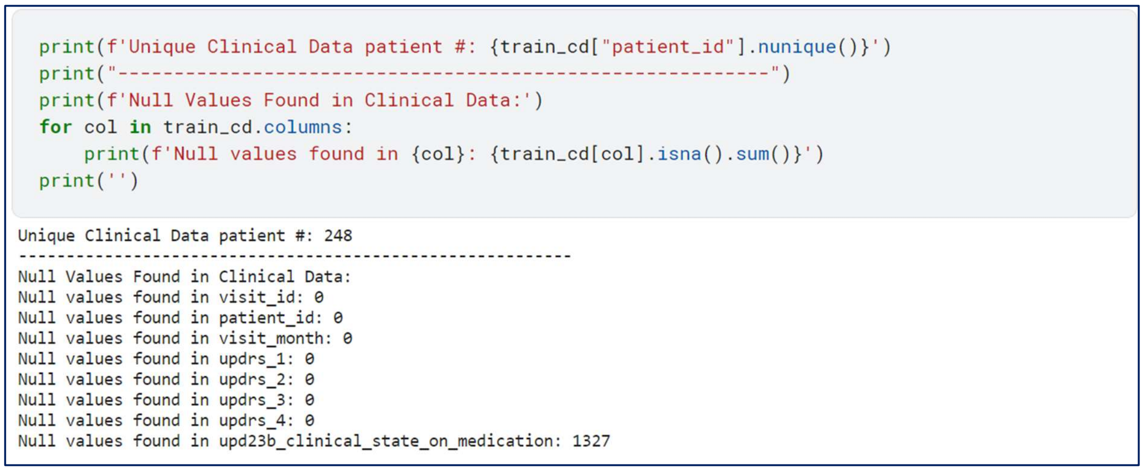






```
train_cd.updrs_1 = train_cd.updrs_3.interpolate(method='linear', axis=0)
train_cd.updrs_2 = train_cd.updrs_3.interpolate(method='linear', axis=0)
train_cd.updrs_3 = train_cd.updrs_3.interpolate(method='linear', axis=0)
train_cd['updrs_4'] = train_cd['updrs_4'].fillna(0)
```

And now we see:



```
print(f'Unique Clinical Data patient #: {train_cd["patient_id"].nunique()}')
print("-----------------------------------------------------------")
print(f'Null Values Found in Clinical Data:')
for col in train_cd.columns:
    print(f'Null values found in {col}: {train_cd[col].isna().sum()}')
print('')
```

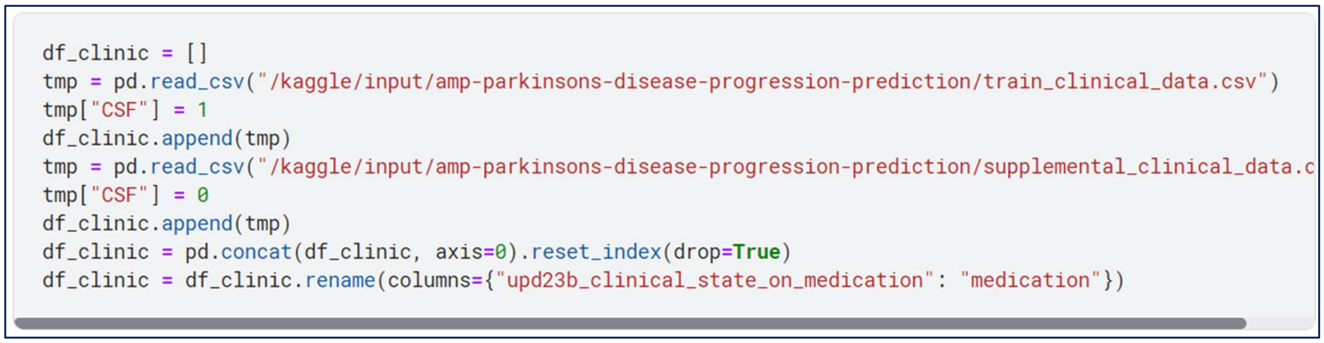
```
Unique Clinical Data patient #: 248
-----------------------------------------------------------
Null Values Found in Clinical Data:
Null values found in visit_id: 0
Null values found in patient_id: 0
Null values found in visit_month: 0
Null values found in updrs_1: 0
Null values found in updrs_2: 0
Null values found in updrs_3: 0
Null values found in updrs_4: 0
Null values found in upd23b_clinical_state_on_medication: 1327
```

NOTE: the null values shown in medication shouldn’t be updated or modified since it displays if a patient is on or off medication so hence we could just replace them by 1 for (on) and 0 for (off) to remove this.

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**.

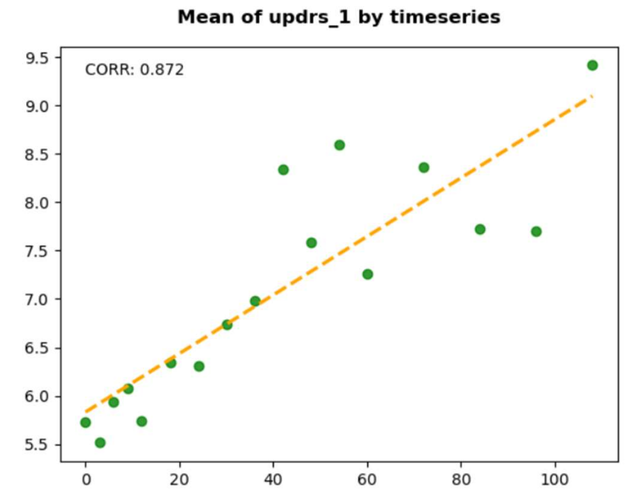
this dataset cannot be used in making our prediction but just to get additional insights on the trends of the clinical data hence we are not cleaning this dataset as it doesn’t contain the patients’ peptide and protein value obtained from their CSF tests.

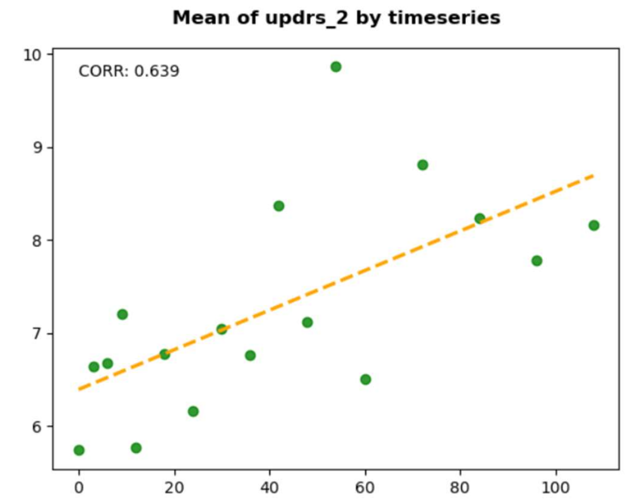
The clinical data and supplemental clinical data have been merged in order to observe the trends of updrs values. Viz;

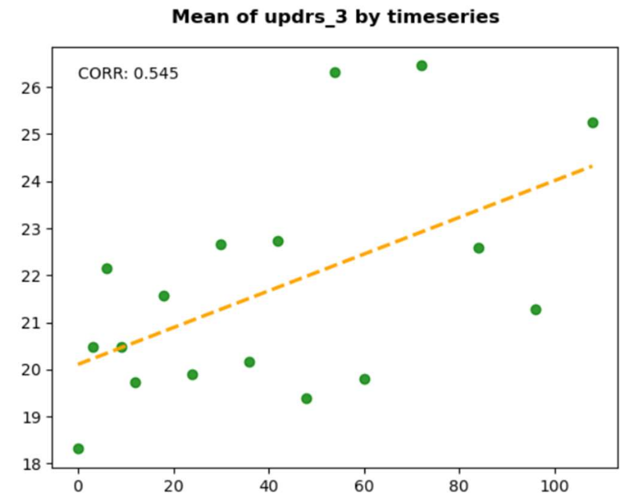


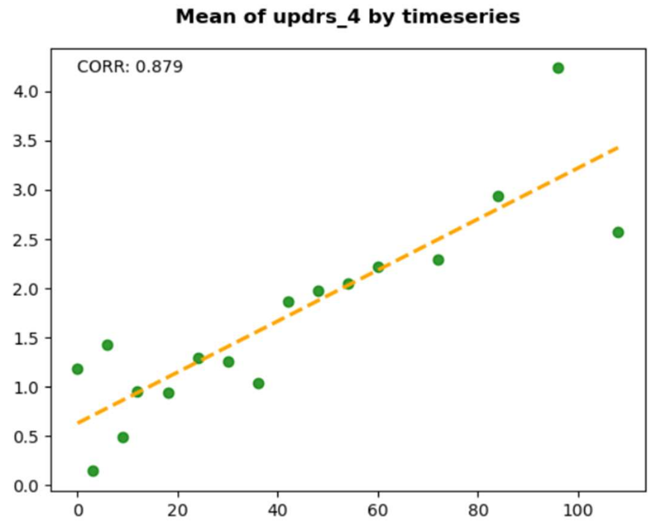
```
df_clinic = []
tmp = pd.read_csv("/kaggle/input/amp-parkinsons-disease-progression-prediction/train_clinical_data.csv")
tmp["CSF"] = 1
df_clinic.append(tmp)
tmp = pd.read_csv("/kaggle/input/amp-parkinsons-disease-progression-prediction/supplemental_clinical_data.c
tmp["CSF"] = 0
df_clinic.append(tmp)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

Trends in overall updrs vaues have shown that it has been increasing and thence can be used for linear interpolation and also it is expected nature.



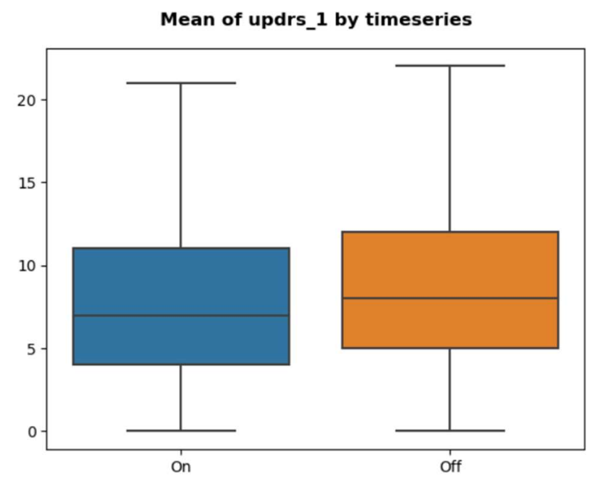


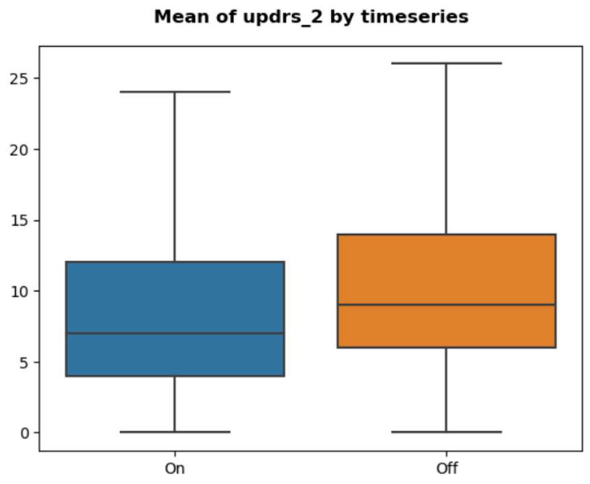


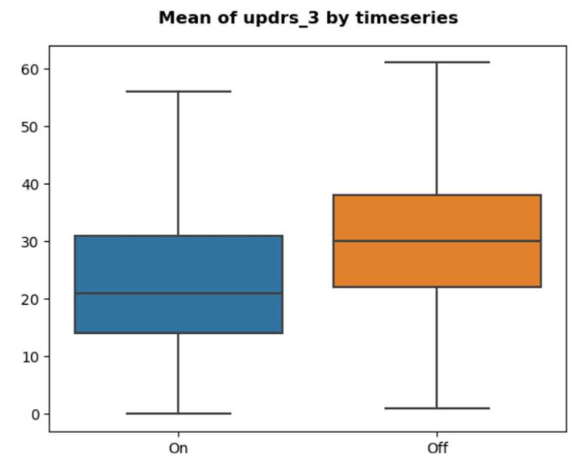


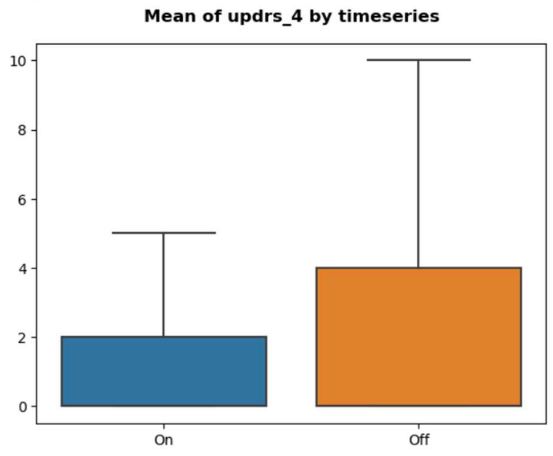
Now we are interested in knowing whether medication has been effective in order to know whether to consider it as a parameter for modelling or not.

Clearly, median of medication 'ON' being lower than that of 'OFF' signifies that medication has been affective and hence **Medication is to be considered as parameter while modelling.**


Mean of updrs_1 by timeseries
0
5
10
15
20
On
Off


Mean of updrs_2 by timeseries
0
5
10
15
20
25
On
Off


Mean of updrs_3 by timeseries
0
10
20
30
40
50
60
On
Off

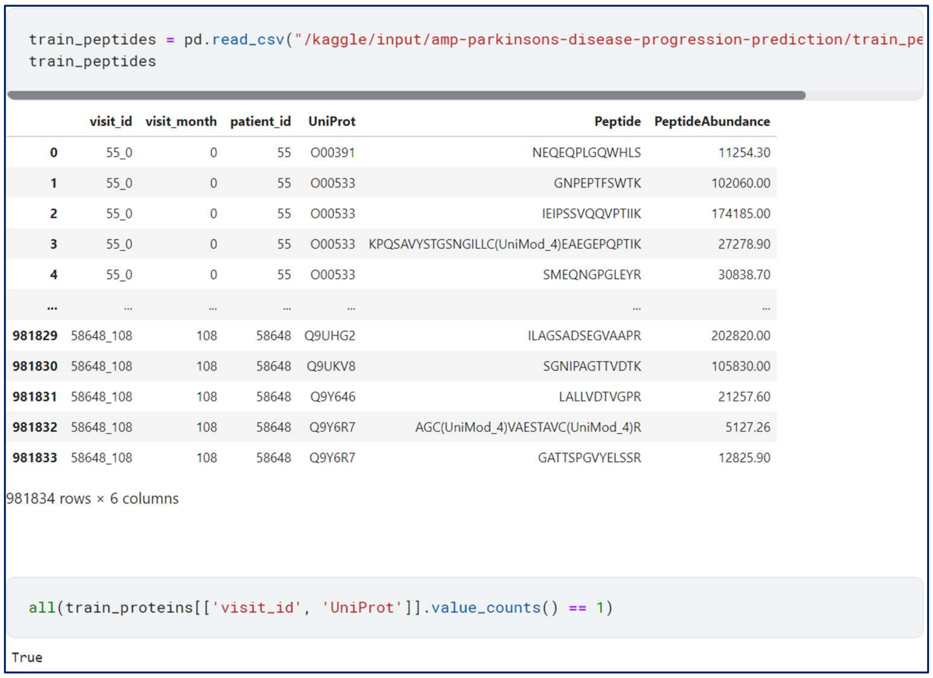

Mean of updrs_4 by timeseries
0
2
4
6
8
10
On
Off

Cleaning and preprocessing of train_peptides.csv, train_proteins.csv

Specifications of Dataset -

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)
```

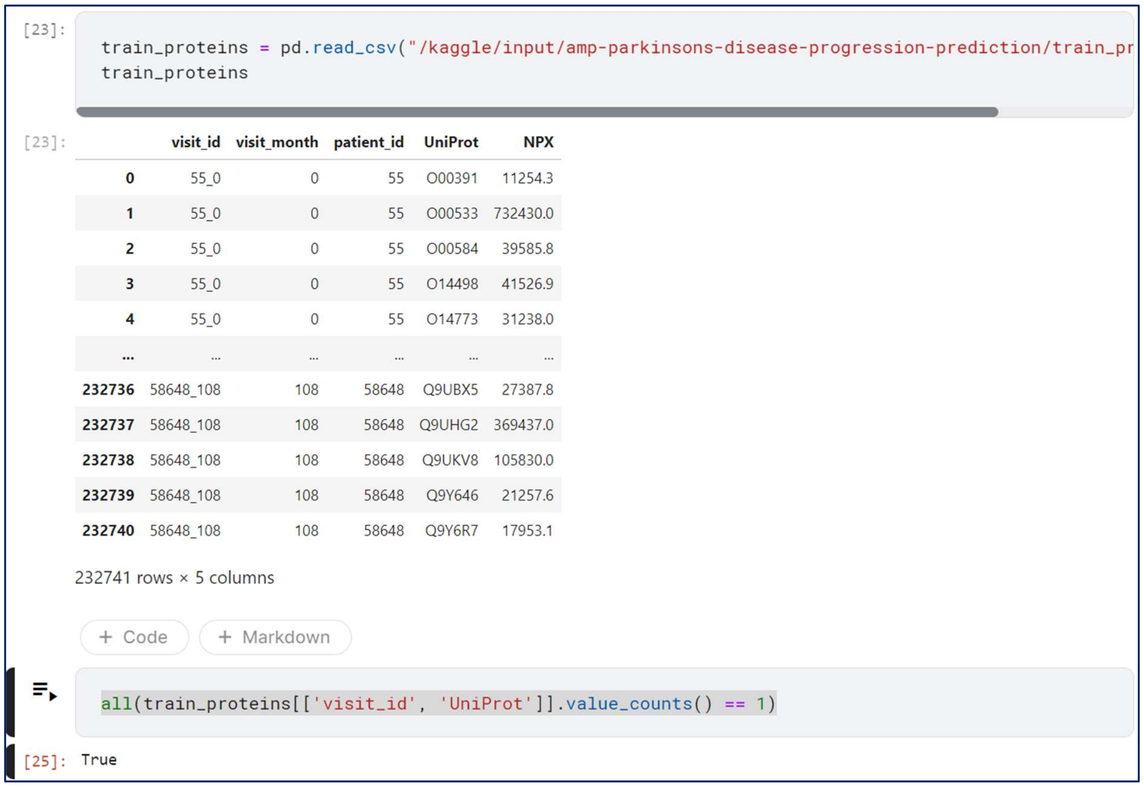
```
True
```

This shows that there are no missing values in the data and there is no scope of inconsistency since this data is from a authentic source and all the variables have matching value types as they should and relevant datasets of this sort have the same datatypes hence
We can say this data is already good to go

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

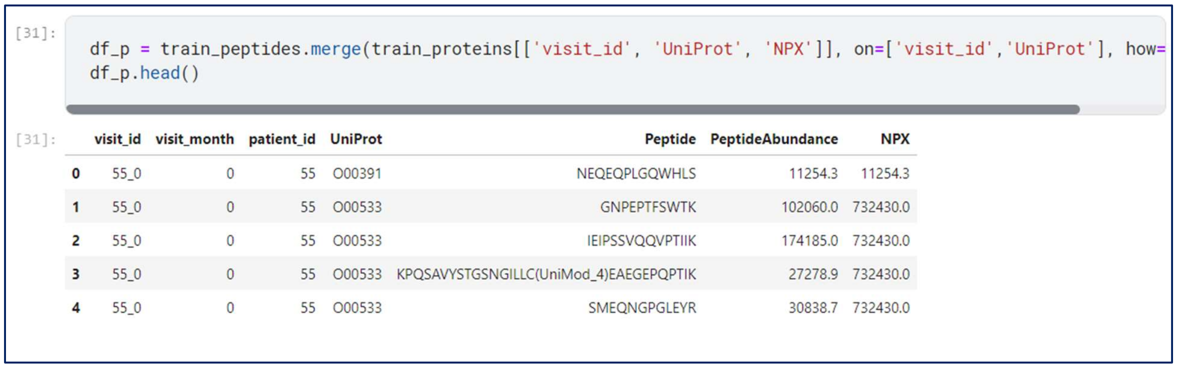
+ Code + Markdown

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

```
[25]: True
```

Therefore similar to the peptides data that there are no missing values in the data and there is no scope of inconsistency since this data is from a authentic source and all the variables have matching value types as they should and relevant datasets of this sort have the same datatypes hence
We can say this data is already good to go.

Now-since proteins are made of peptides, its appropriate to combine datasets of both proein and peptide together and hereby reducing the dimensionality.
We pivot both datasets about visit_id and then merge accordingly



```
[31]:
df_p = train_peptides.merge(train_proteins[['visit_id', 'UniProt', 'NPX']], on=['visit_id','UniProt'], how=
df_p.head()
```

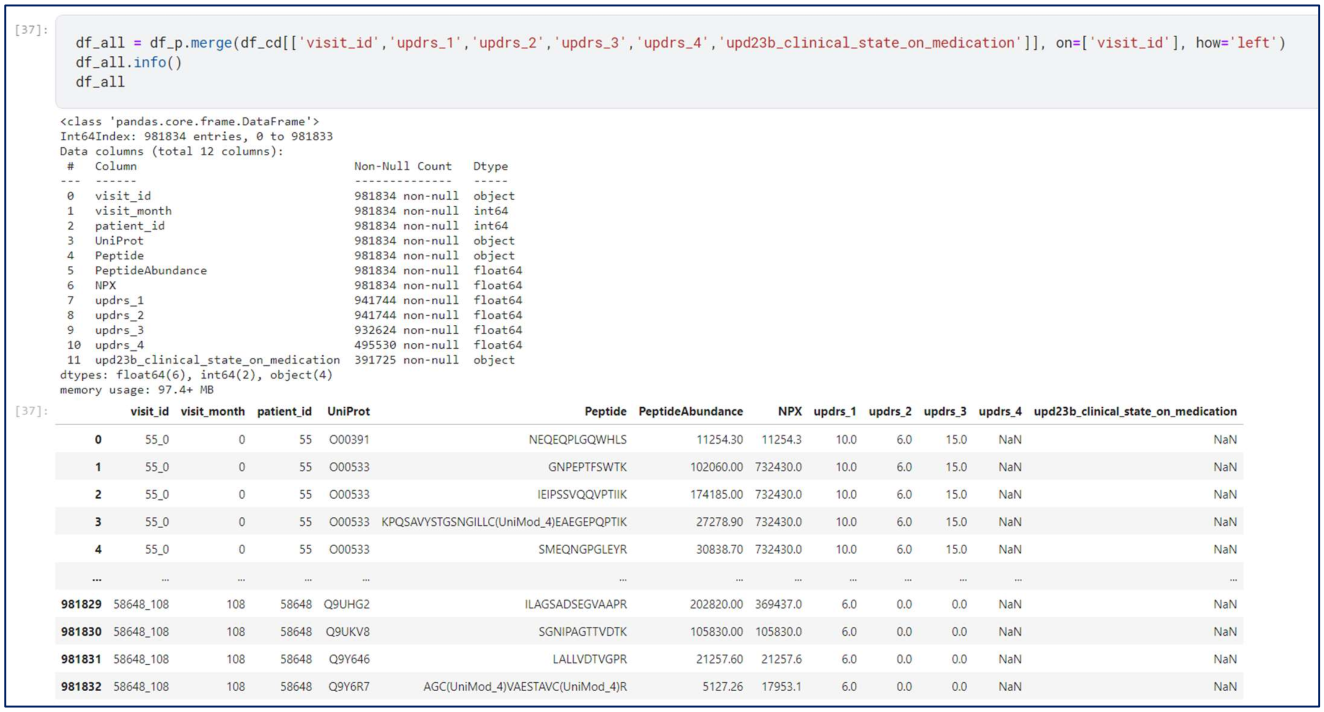
[31]:

	visit_id	visit_month	patient_id	UniProt	Peptide	PeptideAbundance	NPX
0	55_0	0	55	O00391	NEQEQPLGQWHLS	11254.3	11254.3
1	55_0	0	55	O00533	GNPEPTFSWTK	102060.0	732430.0
2	55_0	0	55	O00533	IEIPSSVQQVPTIIK	174185.0	732430.0
3	55_0	0	55	O00533	KPQSAVYSTGSNGILLC(UniMod_4)EAEGEPQPTIK	27278.9	732430.0
4	55_0	0	55	O00533	SMEQNGPGLEYR	30838.7	732430.0

Getting data-frame ready for training

Now we have arrived at a clean, preprocessed datasets of clinical data and protein-peptide data (reduced from 4 datasets to two by handling missing values and merging)

In order to be able to train we have to extract proper features into single dataframe and thence pass it to model which we have achieved by combining the two datasets pivoted at visit ID as shown:



```
[37]: df_all = df_p.merge(df_cd[['visit_id','updrs_1','updrs_2','updrs_3','updrs_4','upd23b_clinical_state_on_medication']], on=['visit_id'], how='left')
      df_all.info()
      df_all
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 981834 entries, 0 to 981833
Data columns (total 12 columns):
 #   Column                               Non-Null Count   Dtype  
---  ------                               --------------   -----  
 0   visit_id                             981834 non-null  object 
 1   visit_month                          981834 non-null  int64  
 2   patient_id                           981834 non-null  int64  
 3   UniProt                              981834 non-null  object 
 4   Peptide                              981834 non-null  object 
 5   PeptideAbundance                     981834 non-null  float64
 6   NPX                                  981834 non-null  float64
 7   updrs_1                              941744 non-null  float64
 8   updrs_2                              941744 non-null  float64
 9   updrs_3                              932624 non-null  float64
 10  updrs_4                              495530 non-null  float64
 11  upd23b_clinical_state_on_medication  391725 non-null  object 
dtypes: float64(6), int64(2), object(4)
memory usage: 97.4+ MB
```

[37]:

	visit_id	visit_month	patient_id	UniProt	Peptide	PeptideAbundance	NPX	updrs_1	updrs_2	updrs_3	updrs_4	upd23b_clinical_state_on_medication
0	55_0	0	55	O00391	NEQEQPLGQWHLS	11254.30	11254.3	10.0	6.0	15.0	NaN	NaN
1	55_0	0	55	O00533	GNPEPTFSWTK	102060.00	732430.0	10.0	6.0	15.0	NaN	NaN
2	55_0	0	55	O00533	IEIPSSVQQVPTIIK	174185.00	732430.0	10.0	6.0	15.0	NaN	NaN
3	55_0	0	55	O00533	KPQSAVYSTGSNGILLC(UniMod_4)EAEGEPQPTIK	27278.90	732430.0	10.0	6.0	15.0	NaN	NaN
4	55_0	0	55	O00533	SMEQNGPGLEYR	30838.70	732430.0	10.0	6.0	15.0	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...
981829	58648_108	108	58648	Q9UHG2	ILAGSADSEGVAAPR	202820.00	369437.0	6.0	0.0	0.0	NaN	NaN
981830	58648_108	108	58648	Q9UKV8	SGNIPAGTTVDTK	105830.00	105830.0	6.0	0.0	0.0	NaN	NaN
981831	58648_108	108	58648	Q9Y646	LALLVDTVGPR	21257.60	21257.6	6.0	0.0	0.0	NaN	NaN
981832	58648_108	108	58648	Q9Y6R7	AGC(UniMod_4)VAESTAVC(UniMod_4)R	5127.26	17953.1	6.0	0.0	0.0	NaN	NaN

Conclusion

Using the inferences from EDA in past weeks, we have efficiently understood the pattern in disease growth and thence applied appropriate methods as forementioned in report to handle missing values and deciding on which parameters to consider for model (for instance whether patient was on medication or not).

Also with some background research of proteins, we understood the connection between proteins and peptides and thence reduced dimensionality of data and finally by combining above two datasets (reduced dataframes) we obtained a final dataframe ready to be trained on various models to obtain targets(updrs values).