Appendix A. Extended preprocessing description

The methods for handling the data for each of the datasets were mainly based on literature on machine learning pipelines (**???**). The step by step procedures id described at Figure 3. This same pipeline was adapted to the different questions set by the hypotheses.

Step by step method

Clinical phenotypes imbalance

As mentioned above, the datasets are longitudinal measurements, and in the diagnostic models they are treated as cross-sectional. When doing the balancing of the dataset so that the class imbalance will not skew the results, the main balancing process is done for the classes. Still, to the clinician the prediction of a person who shows consistently healthy state for 11 visits is not as important as the prediction of a person who jumps between stages of impairment or diagnosis. The distribution of the diagnosis spectrum of subjects is described below for each dataset, and then a method for handling this process is described.

ADNI The diagnosis per person was checked and the results are described below. As shown, the patients with a single diagnosis together make up for around 60% of the visits. When dealing with imbalance, reducing these groups, especially subjects with extreme missingness, can help with the balancing process and the imputation at the same time. The ADNIMERGE columns, missingness is low in comparison to the general missingness, and that is reflected in the extensive use in the literature of this subset.

NACC dataset The distribution of missingness of the NACC dataset is more uniform between the types of predictors, and it includes more neuropathologies. Additionally, the diagnostic process tends to be independent of the previous diagnoses. Still, through the grouping suggested in Section 2.3. it is possible to go through the same analysis as done for ADNI. As shown in the table below, the subjects that show only a single diagnosis (AD, HC) are still the most represented with more than 60% of the visits. The balancing process should aim at these categories.

Different from ADNI, the missingness is not as extreme, and the class distribution is more complex. The class MCI is not as well-represented as the other classes, and the major groups (FTLD, DwMD) are well-represented, while VaD is also following, where the subjects diagnosed with both VaD and AD are more present. The previously discussed diagnostic process of MCI vs HC, AD vs MCI, and HC vs MCI vs AD are still significant. Still, new potential diagnostic questions are observed to be relevant with subjects being in the VaD vs AD vs MCI vs HC, and DwMD vs FTLD vs AD spectrums.

It is unrealistic to build a model that can predict all the different groups correctly, and it is an expensive task computationally to find the best parameters for the process of imputation, feature selection, and imbalance handling. In this case, building a model for each of the questions mentioned above is more ecologically valid, and reduces some of the overhead of building big models. The relevant columns included in the relevant missingness % are demographics, npa, impair, and nps.

Procedure The categories with most representation were reduced for both datasets. It is worthy to remind that these procedures do not aim to balance the classes of the visits, but the distribution of clinical phenotypes: stable profiles and changing profiles. At the same time they try to improve the problem of missingness through picking with priority the cases with least missing values. The removal was done based on the procedure:

1. get the number of visits equal to the next *single* category based on least missingness,
2. get the subjects that are part of the most representative categories,
3. prioritize the visits with the least missing data from each subject.

In the ADNI dataset the categories affected were `CN` and `MCI`, both reduced to a closer number to the number of `Dementia` visits (~ 700 visits). For the NACC dataset the procedure was applied twice. For the Model 3 and 4 the `AD` and `HC` categories were reduced to the number of `MCI` visits (~ 1140 visits). For the Model 5 the categories affected were `AD`, `HC`, and `HC,AD` reduced to the number of `FTLD` visits (4082). The code producing this reduction is added at Appendix C.

Table 1: Distribution of diagnosis, ADNI and NACC (>50 subj).

Diagnosis spectrum	Subjects	Visits (%)	Avg. visit no.	Miss. %	Relevant Miss. %
ADNI					
CN	604	2007 (31.5 %)	3.32	63.72	20.61
MCI	490	1804 (28.31 %)	3.68	65.01	17.73
Dementia	276	700 (10.99 %)	2.54	64.44	19.47
Dementia, MCI	172	909 (14.27 %)	5.28	61.89	14.38
CN, MCI	140	812 (12.74 %)	5.8	68.14	15.53
CN, Dementia, MCI	19	122 (1.91 %)	6.42	77.7	20.25
CN, Dementia	3	18 (0.28 %)	6	82.64	24.17
NACC					
AD	15833	38266 (27.38 %)	2.42	58.7	48.12
HC	14825	56766 (40.62 %)	3.83	60.19	40.42
AD, HC	2361	14271 (10.21 %)	6.04	59.9	44.42
FTLD	2018	4082 (2.92 %)	2.02	57.38	51.35
DwMD	1888	3750 (2.68 %)	1.99	57.36	48.35
MCI	821	1142 (0.82 %)	1.39	58.47	42.4
HC, MCI	670	3512 (2.51 %)	5.24	59.94	41.15
VaD	655	1153 (0.83 %)	1.76	58.91	45.94
AD, VaD	459	2016 (1.44 %)	4.39	58.28	44.53
AD, MCI	408	1731 (1.24 %)	4.24	58.59	41.81
AD, HC, MCI	384	2956 (2.12 %)	7.7	59.93	43.59
HC, VaD	325	1855 (1.33 %)	5.71	59.7	42.97
AD, DwMD	306	1204 (0.86 %)	3.93	57.37	48.5
AD, HC, VaD	232	1762 (1.26 %)	7.59	59.09	45.48
AD, FTLD	231	844 (0.6 %)	3.65	58.03	50.86
DwMD, HC	181	924 (0.66 %)	5.1	58.66	43.55
DwMD, FTLD	79	250 (0.18 %)	3.16	55.82	48.79
AD, MCI, VaD	65	410 (0.29 %)	6.31	58.16	39.97
FTLD, HC	62	308 (0.22 %)	4.97	59.07	37.01
AD, HC, MCI, VaD	61	569 (0.41 %)	9.33	59.12	43.36
AD, DwMD, HC	52	375 (0.27 %)	7.21	58.99	44.88

Missing data

The reasons why data can be missing in a clinical dataset for Dementia are various. Some of them are described below.

1. *Unknown (no information).* For example, the score of a standard test like MMSE was skipped, but no note of it was taken. This type of test overlaps with other tests, so it can be predicted if the other tests are available. Otherwise, in case that the test is specific and might be of importance to the diagnostic process, it can be noted that the lack of this test is important through a missing indicator.
2. *Unknown (differently noted).* For example, a test was taken, but the results for a specific subpart was not recorded. The explanation can be written in the textual report of the patient, but in the numerical data might be missing. If the report is still reachable, then a checking for the report should complete the data, and a percentile-based imputation can be provided based on the other scores, in combination with the reason. So the results can be inferred from other patients having similar test scores. If it is not inferrable the missing data is marked with an indicator.
3. *Non-presence.* For example, the drugs taken from the patient might be encoded through up to 40 possible spots (like in the case of NACC). For a patient that takes 1 drug, the other 39 spots will be encoded as Nan. This data should not be replaced, but correctly recognized as not missing. For statistical purposes (testing with Chi-square for selection purposes requires non-missing data), a transformation of these predictors might be necessary into new columns. In case that the information of this predictors is shared with other predictors, then they need to be dropped. Another case can be when the presence of the test shows a potential hypothesis testing of the clinician. For example, the DTI scan was ordered for several patients, and for some of them biomarkers had positive diagnostic value. Still, non-presence of the others can be part of default procedure. In this case marking the missing data needs to be done (**???**).
4. *Not assessed (unrelated to disease).* For example, a verbal refusal from the patient to perform on a test. This can happen if the testing procedure is very long, or the patient is tired. The procedure should follow similar path to reason (2).
5. *Not assessed (because of disease).* For example, patients with a severe AD might not be able to perform in some cognitive tests. Imputing based on percentile scores can be set, for grouping the patient with similarly scoring patients, with similar demographics and same level of impairment.
6. *Version change.* In datasets like ADNI or NACC there are different versions, so some of the variables might be from previous versions, so they are missing in the latest versions. Another case is when new data comes from research and the patient might be contacted. For example, family history is concentrated mostly on impairment, but new data related bipolar disorder with FTLD - then the patient might be asked for this type of information.
7. *Extreme missingness.* The patient's data is missing to a large extent for the main clinical tests, and only sparse information is available. This patient's data might be dropped because replacing the missing data will only confirm the assumptions the imputers have been built on. Exceptional can be the cases where the patient represents an edge-case, and the course can be marking the missing data with indicators of missingness.

Possible strategies As seen above, the missing data might have some diagnostic value - might tell us the degree of the disease, or it might have no diagnostic value - missing completely at random. Knowing why the data is missing allows for a strategy for using the missingness to add value to the model. Also, having a category for the missing data allows imputation that is interdependent. The missingness is classified into three categories: missing at random (MAR), missing completely at random (MCAR), or missing not at random (MNAR). MAR can be observed in case (5) where the missingess is correlated with the diagnosis. MCAR can be observed where the missingess is not correlated with any feature like in case (4) where the patients refusal is unrelated to the score. MNAR can be observed in case (1) where the patient not completing the depression questionnaire might be related to the fact that they are depressed, but it is not shown in the data (Pedersen et al. 2017).

The different ways to handle the missing data have been mentioned above: drop the row with incomplete data (as in case 7), transform the data (as in case 3), impute the data (most of the cases where infering is possible), add missing indicators together with a standard imputer (as in case 2). Imputation is the standard

process for simpler models than ensemble (like SVM or Logistic Regression), as they require full dataset. There are univariate (mean, median, mode), multivariate (iterative, k-nearest neighbors), or deep-learning (neural networks) imputers that can handle the missing data. Still, imputation can have an impact because it will skew the data based on the assumptions, so it is preferrable to trust models that do not need processing like the imputation. The missingness can be allowed for some ensemble models mainly based on decision trees (like XGBoost and LightGBM) that handle the problem natively through tree-classifiers that use the split method (see section 2.3.2.). In this case, not handling the missing data can be of impact to the feature importance.

In these cases a general strategy with an iterative imputer like `ExtraTreesRegressor` (Bogdanovic, Eftimov, and Simjanoska 2022) can be applied, as it is able to regress between correlations not observable to the statistician. Some of the imputing strategies have been tested on a subsample of ADNI dataset on Dementia, and the results have been similar between imputers, with improvements only on run-time (McCombe et al. 2022). These applications are limited because the process goes through a synthetic process of creating missingness.

Table 2: Sample patient characteristics for ADNI.

Variable name	Short descriptor	Missing %	Count	Type
demographics				
AGE	AGE	0.05	72.5 (7.0)	Numerical
PTEDUCAT	PTEDUCAT	0.0	16.3 (2.6)	Numerical
PTGENDER	PTGENDER, count(Male)	0.0	5018 (53.09%)	Categorical
genetics				
APOE4	APOE4, count(0.0)	2.13	5410 (58.48%)	Categorical
nimg				
AV45_bl	AV45_bl	27.35	1.2 (0.2)	Numerical
Entorhinal	Entorhinal	63.72	3557.0 (771.5)	Numerical
FDG_bl	FDG_bl	19.13	1.3 (0.1)	Numerical
Fusiform	Fusiform	63.72	17945.6 (2698.2)	Numerical
Hippocampus	Hippocampus	59.44	6891.7 (1221.9)	Numerical
MidTemp	MidTemp	63.72	19835.6 (2924.9)	Numerical
TAU_bl	TAU_bl	28.81	272.6 (123.6)	Numerical
Ventricles	Ventricles	60.42	38999.1 (21183.7)	Numerical
WholeBrain	WholeBrain	58.2	1033359.0 (107495.4)	Numerical
npa				
ADAS11	ADAS11	32.43	11.0 (8.1)	Numerical
ADAS13	ADAS13	32.86	16.6 (11.2)	Numerical
ADASQ4	ADASQ4	32.18	4.6 (3.0)	Numerical
EcogSPDivatt	EcogSPDivatt	32.66	2.0 (1.0)	Numerical
EcogSPLang	EcogSPLang	30.79	1.7 (0.8)	Numerical
EcogSPMem	EcogSPMem	30.84	2.1 (1.0)	Numerical
EcogSPTotal	EcogSPTotal	30.85	1.8 (0.9)	Numerical
EcogSPVisspat	EcogSPVisspat	32.12	1.6 (0.9)	Numerical
MMSE	MMSE	32.1	27.1 (3.8)	Numerical
MOCA	MOCA	33.61	23.3 (4.7)	Numerical
RAVLT_forgetting	RAVLT_forgetting	33.04	4.2 (2.8)	Numerical
RAVLT_immediate	RAVLT_immediate	32.89	37.9 (13.9)	Numerical
RAVLT_learning	RAVLT_learning	32.88	4.6 (2.8)	Numerical
TRABSCOR	TRABSCOR	34.91	108.2 (69.9)	Numerical

Table 3: Sample patient characteristics in NACC.

Variable name	Short descriptor	Missing %	Count	Type
demographics				
EDUC	Years of education	0.53	15.5 (3.3)	Numerical
INDEPEND	Level of independence, count(1.0)	0.3	96749 (58.43%)	Categorical
NACCAGE	Subject's age at visit	0.0	74.6 (10.2)	Numerical
SEX	Subject's sex, count(2.0)	0.0	81360 (48.99%)	Binary
family-history				
NACCFAM	Indicator of first-degree f...	8.94	79419 (52.51%)	Binary
genetics				
NACCAM	In this family, is there ev...	79.71	28080 (83.34%)	Categorical
health-history				
ANXIETY	Anxiety, count(0.0)	92.67	8904 (73.16%)	Categorical
CBSTROKE	Stroke, count(0.0)	30.5	93671 (81.15%)	Categorical
DEP2YRS	Active depression in the la...	30.86	70143 (61.09%)	Binary
DIABETES	Diabetes, count(0.0)	30.48	85780 (74.29%)	Categorical
RBD	REM sleep behavior disorder...	92.81	10550 (88.33%)	Categorical
SMOKYRS	Total years smoked cigarett...	32.44	10.1 (15.2)	Numerical
impair				
CDRGLOB	Global CDR®, count(0.0)	0.0	70531 (42.47%)	Categorical
CDRLANG	Language, count(0.0)	14.31	92592 (65.06%)	Categorical
COMPORT	Behavior, comportment, and ...	14.31	100659 (70.73%)	Categorical
MEMORY	Memory, count(0.0)	0.0	71799 (43.23%)	Categorical
ORIENT	Orientation, count(0.0)	0.0	92949 (55.97%)	Categorical
PERSCARE	Personal care, count(0.0)	0.0	117379 (70.68%)	Categorical
npa				
ANIMALS	Animals — Total number of a...	14.07	16.8 (7.1)	Numerical
BOSTON	Boston Naming Test (30) — T...	43.24	24.3 (6.5)	Numerical
NACCMMSE	Total MMSE score (using D-L...	40.52	25.5 (6.1)	Numerical
REYDREC	Rey Auditory Verbal Learnin...	99.46	8.6 (4.7)	Numerical
TRAILA	Trail Making Test Part A — ...	20.87	45.3 (30.9)	Numerical
WAIS	WAIS-R Digit Symbol	41.42	46.7 (24.9)	Numerical
nps				
AGIT	Agitation or aggression in ...	5.86	110230 (70.5%)	Binary
ANX	Anxiety in the last month, count(0.0)	5.96	104890 (67.16%)	Binary
DEL	Delusions in the last month, count(0.0)	6.0	124962 (80.05%)	Binary
DISN	Disinhibition in the last m...	5.91	118631 (75.92%)	Binary

ADNI dataset The missing data in the ADNI dataset were presented with `null` mostly with no reasoning provided, except in specific cases where codes were used like `-4`, `99`. For clinical datasets the information on the nature of the missingness of the data is in between case 1 and case 3, where the missingness is probably MNAR, but the source is not reachable, or the data present is of possible clinical importance, but the data is too sparse to apply consistent methods of imputation. The missing data distribution can be found at Figure 6.

NACC dataset The missing data in the NACC dataset was stored in different codes, depending on the predictor. Each code that represented the missing data according to the guidebook was turned into a null value, later to be imputed. The edge cases where these codes are actually values, not to be transformed into null, are taken care using the script at Appendix C. The codes are described below.

- -4 - Not available
- 8, 88, 888, 8888 - Not applicable
- 95, 995 - Physical problem
- 96, 996 - Cognitive/behavior problem
- 97, 997 - Other problem
- 98, 998 - Verbal refusal
- 9, 99, 999, 9999 - Unknown

The distribution of missingness in NACC has been shown in Figure 7. There were no obvious patterns, except the `additional` type of predictors that were more missing. Still, in the models these predictors were not used, so further analysis was not performed.

Procedure The process of imputation was different for each model, as to not introduce bias from the data that the model is not seeing. A vanilla pipeline was defined which separates the columns into categorical and numerical, and handles them separately (see section 3.4.3. for the pipeline description and usage).

- Categorical predictors were imputed through the `most_frequent` and `constant` imputation strategies of `SimpleImputer`.
- Numerical predictors were imputed through different algorithms suggested by the literature: `SimpleImputer`, `KNNImputer`, or `ExtraTreesRegressor` (Pedregosa et al. 2011).

Post imputation, the features went through scaling (for numerical) and encoding (for categorical), and then the datasets created were tested through an `RandomForestClassifier` on their performance. The imputing method with the best `f1_score` average was used for the further development of the models for each dataset.

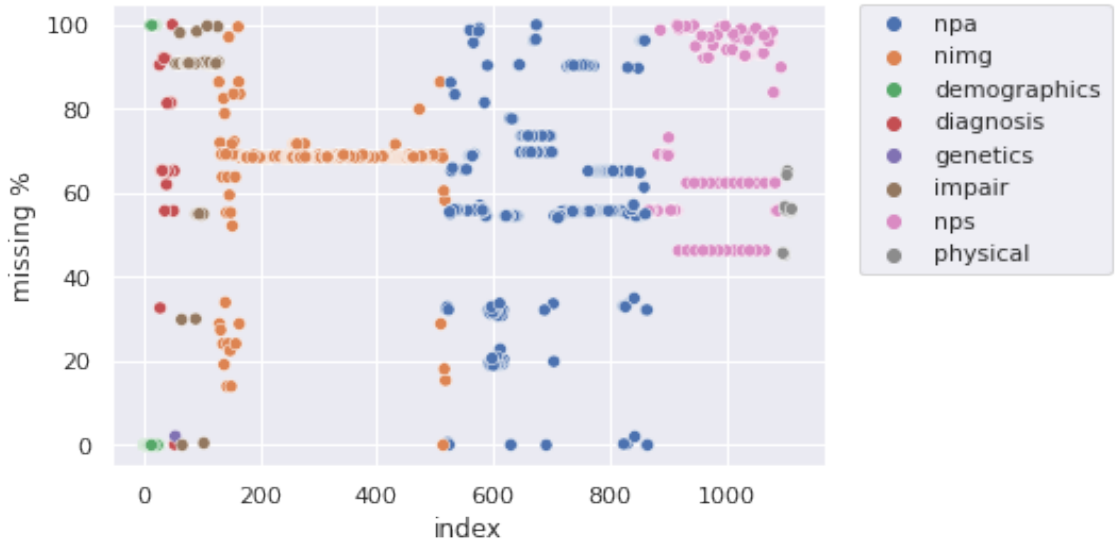

Figure 1: Missingness in ADNI by predictor category.

Figure 2: Missingness in NACC by predictor category.

Predictor selection

The feature selection process is ongoing filtering of the predictors for increasing the generalizability of the models. The selection can happen in several stages of the model building, starting as mentioned above on the clinical data considered (in section 3.1. for ADNI the files were selected), and up to post-hoc analysis of the model. The selection before the modeling process is not based on the relation between predictor-outcome (diagnosis in our case). It can include filtering because of (I) clinical reasoning, (II) literature review, (III) too many missing values, or (IV) redundancy or overlap of features. Similar reasoning was used for filtering the files previously, but it can extend to the features included in the model too. This process makes the models more computationally feasible too.

For models that work with clinical data, it is common to select the predictors based on previous research and the suggested literature. For example, in the latest explainability analysis on the ADNI dataset, the feature selection was performed manually (Bogdanovic, Eftimov, and Simjanoska 2022). This allows for a concentration on a subset of features and improves the timing and performance of the model. Still, when dealing with extensive datasets such as NACC and ADNI and the outcome variables are various, selecting the best predictors from the rule of thumb can be misleading. In these cases, it is not clear which combination of predictors is better suited for certain tasks.

Another direction is to use all the features. Ensemble models like XGBoost are mostly immune to the multicollinearity of numerical variables or the similarity of categorical variables in a classification task. The underlying algorithm picks the best separator among the predictors (see section 2.4.2.). If two features are highly collinear-picking, either one has the same result. Still, these models suffer in terms of maintaining the rank for feature importance. Suppose two features are highly collinear, trees will distribute the importance to both features, thus reducing the suggested clinical role.

These shortcomings of the manual and holistic methods add importance to the process of selecting the best representative predictors early. This can be done either in an unsupervised or supervised process (Pedregosa et al. 2011). In the unsupervised process, the features are compared with each other, and the ones with the most variance are picked. Examples are the Threshold method, the Pearson correlation coefficient for numerical features, and the Chi-square statistics for categorical predictors. In the supervised process, how much the feature variance affects the outcome impacts which features are to be selected. Examples are the ANOVA f-statistic, L1 regularization (Lasso), and ExtraTreesRegressor (Bogdanovic, Eftimov, and Simjanoska 2022).

Sample NPA correlations of ADNI dataset

Sample Impairment correlations of NACC dataset

One automated variable selection method standard is Sequential Feature Selection (SFS). It is a greedy procedure that selects a feature to be added or removed to an initial set of features based on a baseline estimator (Ferri et al. 1994). SFS-based selection interpretative power because one can start from a known group of clinical features and then extend the list by observing whether an extension can increase the explanatory power. SFS is used for genetic data using specific metrics for the training.

Initial selection As there were over 1000 variables to consider for not a significantly bigger sample (<10000 for ADNI, <150000 for NACC), an initial selection of the variables was done for reducing the feature space. The main process was manually selecting which features were excludable for the following reasons:

- Detail: the information is specific, and goes beyond the aim of this work.
- Diagnosis: the information is a very strong predictor of diagnosis by design (ex. medications for AD).
- Not described: the column is not described.
- Redundant: the same information is found in the dataset in another form.
- Missingness: the feature has more than 90% missingness.

Table 4: Reduction of parameter space.

Reason	NACC	ADNI
Detail	72	448
Diagnosis	68	0
Missingness	310	260
Not described	141	0
Redundant	169	6

Procedure Several methods were tested for reducing the feature space after the initial manual reduction.

- *Unsupervised*: Pearson correlation coefficient (`PCC`) was applied to the numerical features and the Chi-square (`Chi2`) statistic to the categorical variables.
- *Supervised*: `ExtraTreesRegressor` (`ETR`) was applied to both types of data.
- *SFS*: Backward elimination was applied with a `RidgeCV` estimator to all data. This estimate is based on `RidgeRegressor` which is a type of Ordinary Least Squares method that imposes a penalty on the estimators. The estimator is accompanied by cross-validation that allows for stable results.

Besides the features selected, a set of clinically relevant features was set. If the methods did not select these features, they were added to the selected features. While forcing these features back in the method can add some correlations that decrease feature importance later, a post-prediction analysis was added to the correlation between the features supported by the models.

Predictor transformations

Transforming the predictors is a procedure that considers the assumptions of the models, and changes the data shape accordingly without losing information. Standard transformations are scaling numerical features and encoding categorical features.

Scaling: Scaling is necessary for models that use distance measures or weighted sums (like linear regression). Firstly a Kolmogorov-Smirnov test (`KS-test`) was applied to the numerical data to test for possible skewness that can happen because some tests might have ceiling or floor effects (**???**). If the data were normally distributed `RobustScaler` (removes the mean and scales to unit variance, and is robust to outliers) is applied, otherwise, a log-transform `FunctionTransformer` was performed. The scaling is not applied to tree models, as the scale does not impact them.

Encoding: Encoding is necessary for models that accept only numerical features, while the dataset contains categorical features too. In this case, new features are created for each category, showing presence or absence. `OneHotEncoder` is applied. Encoding was not applied to the LightGBM model.

Imbalanced classes

Dementia datasets are imbalanced because the prevalence of the diseases differs in the population. Additionally, the clinical relevance of predictive models is smaller for well-defined groups (take for example an advanced form of Alzheimer's Disease). The clinician can see the transition phases of these subjects, and does not need support. The main value that can be added to the models (based on the present state of technology) is to create more differential models for the under-represented classes. The selection of the technique of sampling should be based on clinical knowledge about the diseases that can increase the separability of the classes (ex. for classes like AD vs others, the AD phenotype can be more distinguishable) or not affect it (ex. for FTLD and DwMD that are comorbid groups increasing the separability means adding bias to the model). The library `imbalanced-learn` provides support for the different methods to handle imbalance with a certain degree of adaptability (**???**).

- Over-sampling: With the process of over-sampling, new data points are included in the dataset to balance the importance of the classes. A simple method is `RandomOverSampler` which increases the instances by resampling the existing classes. Other over-sampling methods are based on synthetic instances (like `SMOTE` or `ADASYN`), and cannot be used for clinical data because they require numerical data only. `SMOTENC`, an extension of `SMOTE`, accepts categorical and numerical predictors and uses the nearest neighbors method to generate numerical data and the most frequent category for the categorical data.
- Under-sampling: Reduces the number of instances for over-represented classes t. The method of reduction can be either through a controlled process, or by selecting for more representative samples. For the first version, `RandomUnderSampler` reduces the over-represented class randomly, while the `NearMiss` can use the nearest neighbors method to find appropriate metrics to separate the classes better. The cleaning technique selects better representatives of classes. `Tomek's links` detect the samples that are the nearest neighbors of each other, and are from different classes, and remove them. `EditedNearestNeighbors` reduces the samples based on neighborhood agreement to increase homogeneity. `InstanceHardnessThreshold` reduces the samples with lower probabilities when classified through an algorithm. These methods apply to the clinical data that assume a separability between the classes.
- Combination: As over-sampling can produce noisy data, under-sampling can be used to reduce the introduced noise. This can create new synthetic data that help with the separability of the problems. Examples are `SMOTEENN` and `SMOTETOMEK`.
- Ensemble of samplers: The classification algorithms that are based on some kind of bagging or boosting are trained through small subsets of the dataset. When doing the bagging of the subsets, it is possible to create balanced subsets - and the classifiers will tend to be more sensitive to the under-sampled classes. Example usages are based on `BalancedBaggingClassifier` and `RUSBoostClassifier`.

Procedure The two datasets had an imbalance of the classes post-inclusion as shown in the figures of participant flow and the table below. The respective names or codes in the datasets are included. As mentioned above, the sampling process can impact the process of classification. Two types of tasks will be handled: separation of well-defined groups and separation of comorbid groups.. The heterogeneity of the classification types imposes the need to try the different options of sampling, and check the results. The results can suggest some inherent information in the datasets. The sampling techniques that were tested are the following, and the results can be found in Chapter 4.

- Imb1 - No sampling: balance through logodds that increase the significance of the less represented class,
- Imb2 - `RandomOverSampler`
- Imb3 - `RandomUnderSampler`
- Imb4 - `SMOTENC` and `NearMiss`: Case synthesization that aims at hyperspace boundaries.

Clinical data issues

While the direction for proper preprocessing was described above, several issues were expected because of the clinical data properties.

- *Cross-sectional vs longitudinal data*: While considered cross-sectional in previous research, the datasets are made of several visits for each patient. This has two major implications: potential data leakage and over-representation. Data leakage can happen when the test includes patients that were in the training dataset. In this case, the model can make the prediction directly from what it saw in the training, without actually learning. The over-representation happens when certain characteristic skews the prevalence of the diseases. For example, while it is easier to get data from healthy subjects and participants with MCI, it is more difficult to do so for patients with more advanced stages, like AD or LBD. So in the dataset, there are subjects with more than 10 visits, and all of these visits are normal and similar, and patients with progressed stage with less representation, but more in number.
- *Size of the datasets*: Previous research has concentrated on a small number of variables included infor the models, and that has helped with concentrating on the features to be important clinically (Bogdanovic, Eftimov, and Simjanoska 2022). Still, in an explorative work where the features are being examined, including as many potentially relevant features as possible is of relevance. This imposes a computational burden on the process. Based on preliminary runs of the models, it was observed that the imputation of more than 100000 visits of the NACC dataset (with some level of extreme missingness) through iterative imputing of the whole features set is not feasible, nor possible for a reproducible pipeline with the ability to explore each feature to the detail.
- *Imbalance-imputation trade-off*: There is a trade-off between the imputation and the balancing process, as both synthesize data. The imputation fills missing information in the existing visits through a simple or iterative procedure, while the balancing process either re-samples the existing visits (through `RandomOverSampler`) or synthesizes completely new samples (through `SMOTENC`). While both practices are good practices to introduce new data to improve the model's metrics, their impact on the clinical data is of significant importance. In works where feature importance and single case analysis are of value, these two processes need to be minimized and the real data prioritized.

The problems are related to each other. For example, the over-representation problem is important in both the size of the problem and the imbalance. Also, with the increase of the number of features, the imputation's synthesized data impact is higher. There are differences in the datasets too that are because of the artifacts of data collection. The type of missingness in ADNI dataset is affected by the merging process, where the missingness in the `ADNIMERGE.csv` data is more distributed than in the other sources, as shown in Figure 10. This can show the path to reducing the missingness without introducing bias.

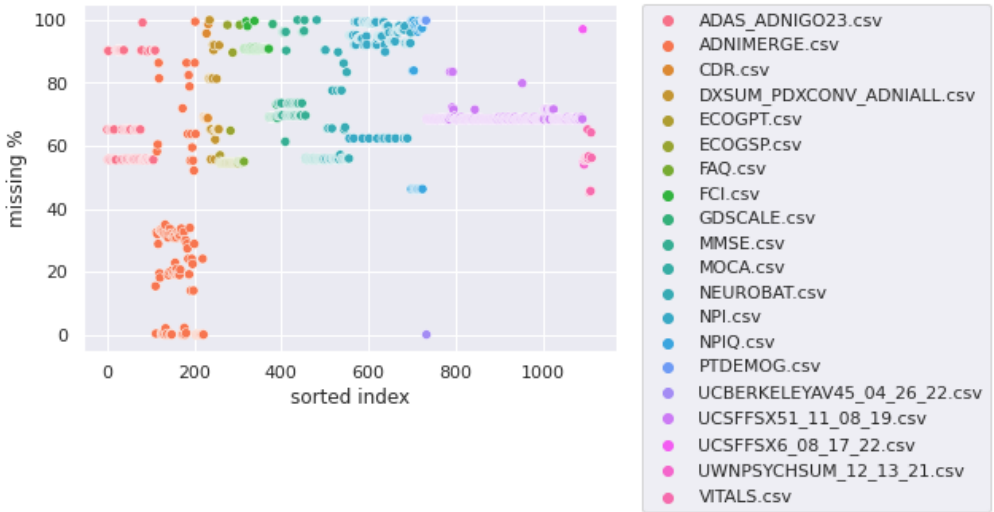

Figure 3: Missingness in ADNI by predictor source.

Diagnosis vs allowed predictor missingness in cases, ADNI.

To handle the potentially introduced bias, an explorative process was followed to investigate the dependency of missingness on features like `ADNI-Protocol`, `EXAMDATE`, and `Category` of the predictors. It was observed

that neither feature had an impact. The next step was to see whether the missingness was related to the diagnosis. Cases were dropped based on the percentage of predictors they had missing. It was observed that the relative frequency of the diseases did not significantly change. This suggests independence of missingness from diagnostic value, as shown in Figure 11.

Appendix C: Useful Scripts

Creating ADNI dataset

Merging ADNI files based on columns (Mentioned in 3.1.1.)

```
import pandas as pd
# get all the files related to ADNI
adni_files_df = pd.read_csv(ADNI_FILES_SELECT, index_col=0)
# read ADNIMERGE file
adni_base_df = pd.read_csv(ADNIMERGE_LOCATION)
# filter based on protocol
adni_base_df = adni_base_df[adni_base_df["COLPROT"].isin(["ADNI2", "ADNI3"])]
# fix viscode-2 problem
adni_base_df["VISCODE2"] = adni_base_df["VISCODE"]
cols = ["PHASE", "Phase", "ProtocolID", "COLPROT"]  # usefule later for merging
# expand the base dataframe with data from the other files
expanded_adni_df = adni_base_df
for i, file_row in adni_files_df[adni_files_df['Select']==1].iterrows():
    # set the file name
    file = file_row["block"]
    if file == "ADNIMERGE.csv":
        continue
    # extract the data in this
    file_i_df = pd.read_csv(ADNI_BASE_PATH + file)
    # set which cols are shared between ADNIMERGE and this file
    col_to_merge = ["RID", "VISCODE2"] + [i for i in cols if i in file_i_df.columns]
    # expand adni dataframe
    # ATTENTION: it creates replicated columns that were processed again
    expanded_adni_df = pd.merge(
        expanded_adni_df,
        file_i_df,
        how="left",
        left_on=["RID", "VISCODE2", "ORIGPROT"],
```

```
        right_on=col_to_merge,
        suffixes=(None, "_y"),
    )
# save the file for future role
expanded_adni_df.to_csv(ADNI_BASE_PATH+"expanded_adni_df.csv")
```

Merging duplicated columns from joining ADNI files

```
# split into columns that are duplicate and non-duplicate
# ['a', 'b', 'b_y'] -> non-duplicate: ['a','b'], duplicate: ['b_y']
non_duplicate_cols = [col for col in expanded_adni_df.columns if "_y" not in col]
duplicate_cols = sorted(
    list(set([col.split("_y")[0] for col in expanded_adni_df.columns if "_y" in col]))
)
# ['a', 'b', 'b_bl', 'b_y', 'c', 'c_y', 'c_y.1'] -> dict_vals: {'b':['b_y'], 'c': ['c_y', 'c_y.1']}
# bl: baseline, _y: appended from the merging
dict_vals = {
    col: [i for i in expanded_adni_df.columns if i.startswith(col) and not i.endswith("_bl")]
    for col in duplicate_cols
}
# merge all the cols in the dict_vals (for ex. b, b_y, b_y.1) into a single column (b)
merger_df = expanded_adni_df
for i, vals in dict_vals.items():
    merger_df[i] = merger_df[vals].apply(
        lambda x: ([i for i in x if not pd.isnull(i)] + [np.nan])[0], axis=1
    )
```

NACC variables

NACC variables filter (Mentioned in 3.2. and 3.2.1.)

```
# initial setup
column_verifier = {
    "column": None,
    'Binary': False,
```

```
    'Categorical': True,
}  # useful for keeping same structure
cv_list = []
# set replacable values with NaN
replaceable_values = [-4] + [ 8, 9] + [88, 888, 8888,  999, 9999] + [*range(95, 100)] + [*range(995, 100
replaceable_values += [str(i) for i in replaceable_values]
# reasons for missing
defaults = {
    "-4": "Not available",
    "8": "Not applicable",
    "9": "Unknown",
    "88": "Not applicable",
    "888": "Not applicable",
    "8888": "Not applicable",
    "95": "Physical problem",
    "995": "Physical problem",
    "96": "Cognitive/behavior problem",
    "996": "Cognitive/behavior problem",
    "97": "Other problem",
    "997": "Other problem",
    "98": "Verbal refusal",
    "998": "Verbal refusal",
    "99": "Unknown",
    "999": "Unknown",
    "9999": "Unknown",
}


for col in nacc_df:
    cv = column_verifier.copy()
    cv['column'] = col
    row_miss = {}   # which values not missing in row
    # if the type of data is an object, categorical
    if nacc_df[col].dtype == "O":
```

```
    cv_list.append(cv)
    continue
# if not int or float, then print it to check its value
if not nacc_df[col].dtype == "int64" and not nacc_df[col].dtype == "float64":
    cv_list.append(cv)
    print(col)
    continue
# Otherwise, its a column containing numbers
# -4 is extensively used, take it out of the system now
if -4 in nacc_df[col]:
    row_miss[-4] = defaults[str(-4)]
n_unique = nacc_df[col].nunique()
if n_unique >= 15:          # Set a baseline for uniquenes - (15 picked as an upper bound)
    cv['Categorical'] = False
# count all the values included in the dataset
vals = nacc_df[col].value_counts().to_dict()
# separate them into potentially noting missing or otherwise
non_missingable = set(vals.keys()) - set(replaceable_values)
missingable = set(vals.keys()) & set(replaceable_values)
# the decision is made based on the non-missingables
# ex: data[col] = {-4, 0, 1, 9} --> binary
if len(non_missingable) == 2:
    cv['Binary'] = True
if len(missingable) and len(non_missingable):
    # ex: data[col] = {-4, 0, 1, 2, 3, 4, 9} --> categorical
    if (int(max(missingable)) == 8 or int(max(missingable)) == 9) and int(max(non_missingable)) <= 6
        for miss in missingable:
            row_miss[miss] = defaults[str(int(miss))]
    # ex: data[col] = {-4, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 88, 99} --> categorical
    elif int(max(missingable)) - int(max(non_missingable)) > 10:
        for miss in set(missingable)-set([8,9, 8.0, 9.0]):
            row_miss[miss] = defaults[str(int(miss))]
# store
```

```
    cv['missing'] = '\n'.join(str(i)+':'+j for i,j in row_miss.items())
    cv_list.append(cv)
```

Balancing phenotypes

Getting spectrum for each

```
def stats_spectrum(df, PTID="RID", DX="DX", relevant_cols=[]):
    # set levels of missingness
    df["missingness"] = df.isna().sum(axis=1)
    df["relevant_missingness"] = df[relevant_cols].isna().sum(axis=1)
    groups = {}
    groups_ids = {}
    groups_missingness = {}
    groups_relevant = {}
    # group by clinical phenotype
    for i, gr in df.groupby(PTID):
        tx = tuple(sorted(set(gr[DX]) - set([np.nan])))
        if tx not in groups:
            groups[tx] = []
            groups_ids[tx] = []
            groups_missingness[tx] = []
            groups_relevant[tx] = []

        groups[tx].append(len(gr))  # visits for each participant
        groups_ids[tx].append(i)  # participant ids
        groups_missingness[tx].append(
            np.mean(gr["missingness"]) / len(df.columns) * 100
        )  # missingness of all rows in df
        groups_relevant[tx].append(
            np.mean(gr["relevant_missingness"]) / len(relevant_cols) * 100
        )  # missingness of relevant columns
    # put everything in a dataframe for descriptions
    stats = []
    total_visit_count = 0
    for i, vals in groups.items():
        name = ", ".join(sorted([str(k) for k in i]))
        stats.append(
```

```
            [
                name,
                len(vals),
                np.sum(vals),
                round(np.mean(vals), 2),
                round(np.mean(groups_missingness[i]), 2),
                round(np.mean(groups_relevant[i]), 2),
            ],
        )
        total_visit_count += np.sum(vals)
    stats_df = pd.DataFrame(
        stats,
        columns=[
            "Diagnosis spectrum",
            "Subjects",
            "Visits (%)",
            "Avg. visit no.",
            "Miss. %",
            "Relevant Miss. %",
        ],
    )
    stats_df["Visits (%)"] = stats_df.apply(
        lambda row: f"{row['Visits (%)']} ({round(row['Visits (%)'] / total_visit_count *100, 2)} %)",
        axis=1,
    )
    print()
    print(stats_df.sort_values("Subjects", ascending=False).to_markdown(index=False))
    print()
    # return groups of ids
    return groups_ids, groups
```

Reducing subject count

```
def reduce_subject_count(
    df, less_represented, more_represented, groups_ids, PTID="RID", DX="DX"
):
    less_represented_cnt = len(less_represented)
    keep = []
    # make sure the list is sorted before checked
    more_represented = [tuple(sorted(list(i))) for i in more_represented]
    # groups_ids: {('AD'):[ID1, ID2,], ('HC'):[ID3,], ('AD', 'HC'):[ID4, ID5]}
    # iterate through these groups and decide which ones to keep
    for key, vals in groups_ids.items():
        # if this group is over-represented
        if key not in more_represented:
            keep += list(df[df[PTID].isin(vals)].index)
            continue
        if len(key) > 1:
            to_keep = list(
                df[df[PTID].isin(vals)]
                .sort_values(by=[PTID, "missingness", "DX"])
                .drop_duplicates(subset=[PTID, "DX"], keep="first")
                .index
            )
            keep += to_keep
            continue
        else:
            # initially keep at least on visit for each participant
            to_keep = list(
                df[df[PTID].isin(vals)]
                .sort_values(by=[PTID, "missingness"])
                .drop_duplicates(subset=[PTID], keep="first")
                .index
            )
```

```
        # if the number passes the less represented count
        # drop some of the visits
        if len(to_keep) >= less_represented_cnt:
            to_keep = random.sample(to_keep, less_represented_cnt)
        # otherwise, add some random visits to match the same levels
        # of representation
        else:
            difference = less_represented_cnt - len(to_keep)
            all_ind = list(df[(df[PTID].isin(vals)) & (~df.index.isin(keep))].index)
            to_keep = to_keep + random.sample(all_ind, difference)
        keep += to_keep
    return keep
```

C.3.3. Save each subgroup

```
EXPERIMENT_LOWER_REPRESENTATIONS = {
    "M3": "MCI",
    "M4": "MCI",
    "M5": "FTLD",
}
for exp, lower_rep in EXPERIMENT_LOWER_REPRESENTATIONS.items():
    print(f"EXPERIMENT:\t {exp}")
    # extract subset
    nacc_df_model_i = nacc_df[
        nacc_df["DX"].isin(experiments[exp]["diagnostic_vals"].keys())
    ]
    # print spectrum of initial dataset
    groups_ids, groups_visits = stats_spectrum(
        nacc_df_model_i, relevant_cols=nacc_cols_relevant, PTID="NACCID", DX="DX"
    )
    # get the less represented category count of individuals
    less_represented = nacc_df_model_i[
        nacc_df_model_i["NACCID"].isin(groups_ids[(lower_rep,)])
    ]
    # reduce the subject count for the more represented classes
    more_represented = [("HC",), ("AD",), ("AD", "HC")]
    keep = reduce_subject_count(
        nacc_df_model_i, less_represented, more_represented, groups_ids, PTID="NACCID"
    )
    df = nacc_df_model_i[nacc_df_model_i.index.isin(keep)]
    new_groups_ids, new_groups_visits = stats_spectrum(
        df, relevant_cols=nacc_cols_relevant, PTID="NACCID", DX="DX"
    )
    graph_from_phenotypes(new_groups_ids)
    graph_from_phenotypes(new_groups_visits, func=np.sum)
```

```
nacc_df_model_i[nacc_df_model_i.index.isin(keep)].to_csv("../data/NACC_"+exp+".csv")
print("*"*100)
```

Experiment configurations


```
{
  "experiments": {
    "M1": {
      "dataset": "ADNI",
      "diagnostic_vals": {
        "HC": ["CN"],
        "MCI": ["MCI"],
        "AD": ["AD"]
      },
      "predictors": {
        "source": "ADNIMERGE.csv",
        "type": ["demographics", "npa", "nps", "nimg"]
      }
    },
    "M2": {
      "dataset": "ADNI",
      "diagnostic_vals": {
        "HC": ["CN"],
        "MCI": ["MCI"],
        "AD": ["AD"]
      },
      "predictors": {
        "source": "*.csv",
        "type": ["demographics", "npa", "nps"]
      }
    },
    "M3": {
      "dataset": "NACC",
      "diagnostic_vals": {
        "HC": [88],
        "MCI": [26, 27, 28, 29, 30],
```

```
    "AD": [1]
  },
  "predictors": {
    "source": "*.csv",
    "type": ["*"]
  }
},
"M4": {
  "dataset": "NACC",
  "diagnostic_vals": {
    "HC": [88],
    "MCI": [26, 27, 28, 29, 30],
    "AD": [1],
    "VaD": [8]
  },
  "predictors": {
    "source": "*.csv",
    "type": ["*"]
  }
},
"M5": {
  "dataset": "NACC",
  "diagnostic_vals": {
    "HC": [88],
    "AD": [1],
    "FTLD": [6, 7],
    "DwMD": [2, 3, 4, 5]
  },
  "predictors": {
    "source": "*.csv",
    "type": ["*"]
  }
}
```

```
  },
  "datasets": {
    "NACC": {
      "id": "NACCID",
      "date": "VISITYR",
      "gender": "SEX",
      "age": "NACCAGE",
      "education": "EDUC",
      "dx": "NACCETPR",
      "female": 2
    },
    "ADNI": {
      "id": "PTID",
      "date": "EXAMDATE",
      "gender": "PTGENDER",
      "age": "AGE",
      "education": "PTEDUCAT",
      "dx": "DX",
      "female": "Female"
    }
  }
}
```

Grid search parameters

```
unsupervised_selection_params = {
    "transformer__categorical__feature_selection": [
        SelectPercentile(percentile=40, score_func=chi2)
    ],
    "transformer__numerical__feature_selection": [
        SelectPercentile(percentile=10, score_func=chi2),
        SelectPercentile(percentile=40, score_func=chi2),
        SelectPercentile(percentile=10, score_func=mutual_info_classif),
        SelectPercentile(percentile=40, score_func=mutual_info_classif),
    ],
    "transformer__categorical__feature_selection__percentile": [10, 40,],
    # "transformer__numerical__feature_selection__score_func": [mutual_info_classif, chi2, f_classif]
}
supervised_selection_params = {
    "transformer__categorical__feature_selection": [SelectFromModel(DecisionTreeClassifier())],
    "transformer__numerical__feature_selection": [SelectFromModel(DecisionTreeClassifier())],
}



scaling_params = {
    "transformer__numerical__scaling": [
        RobustScaler(),
        StandardScaler(),
    ],
}


simple_sampling_over_params = {"balance": [RandomOverSampler()]}
simple_sampling_under_params = {"balance": [RandomUnderSampler()]}
def get_advanced_sampling_params(categorical_features=[]):
    advanced_sampling_params = {
        "balance": [SMOTENC(categorical_features=categorical_features)],
```

```
        "balance_2": [NearMiss()],
    }
    return advanced_sampling_params


model_params = {
    "classifier": [
        RandomForestClassifier(),
        LGBMClassifier(),
        RandomForestClassifier(),
    ],
}


main_grid_params = {
    "transformer__categorical__imputation": ["passthrough"],
    "transformer__categorical__feature_selection": ["passthrough"],
    "transformer__numerical__imputation": ["passthrough"],
    "transformer__numerical__feature_selection": ["passthrough"],
    "transformer__numerical__scaling": ["passthrough"],
    "classifier": [RandomForestClassifier()],
    "balance": [RandomUnderSampler()],
    "balance_2": ["passthrough"],
}


vanilla_grid_params = {
    key: [val[0]] for key, val in main_grid_params.items()
}  # for control


selection_grid_params = [
    {**main_grid_params, **unsupervised_selection_params},
    {**main_grid_params, **supervised_selection_params},
]
# selection_grid_params.append(vanilla_grid_params)
```

```
scaling_grid_params = [{**main_grid_params, **scaling_params}]
# scaling_grid_params.append(vanilla_grid_params)

sampling_grid_params = [
    {**main_grid_params, **simple_sampling_over_params},
    {**main_grid_params, **simple_sampling_under_params},
]
# scaling_grid_params.append(vanilla_grid_params)

## GRID params
# defined_grid_params = grid_params
PARAMS = {
    "feature_selection": selection_grid_params,
    "imputation": vanilla_grid_params,
    "scaling": scaling_grid_params,
    "sampling": sampling_grid_params,
}
```

Appendix D: Results Extended

Table 5: ADNI files.

block	Select	Exclusion	Category
ADASSCORES.csv		Redundant	npa
ADAS_ADNI1.csv		Excluded	npa
ADAS_ADNIGO23.csv	True		npa
ADDCOMM.csv		Detail	npa
ADNI2_ECG.csv		Detail	physical
ADNI2_VISITID.csv		Detail	additional
ADNIMERGE.csv	True		demographics
ADNI_CBBRESULTS.csv		Detail	additional
ADNI_EMBICDCB.csv		Detail	additional
ADNI_UCD_WMH_05_02_22.csv		Detail	nimg
ADSP_PHC_COGN_05_06_22.csv		Detail	npa
ADSP_PHC_COGN_DICT_05_06_22.csv		Detail	npa
ADSXLIST.csv		Excluded	health-history
ADVERSE.csv		Detail	health-history
ARM.csv		Detail	physical
AV45FOLLOW.csv		Detail	physical
AV45VITALS.csv		Detail	
BACKMEDS.csv		Redundant	medications
BHR.csv		Detail	additional
BHR_BASELINE_QUESTIONNAIRE.csv		Minimal	additional
BHR_EVERYDAY_COGNITION.csv		Minimal	impair
BHR_LONGITUDINAL_QUESTIONNAIRE.csv		Minimal	additional
BHR_MEMTRAX.csv		Detail	npa
BHR_SP_ADL.csv		Minimal	impair
BHR_SP_CAREGIVER_BURDEN.csv		Minimal	co-participant
BHR_SP_EVERYDAY_COGNITION.csv		Minimal	co-participant
BHR_SP_FAQ.csv		Minimal	co-participant
BHR_SP_INITIAL.csv		Minimal	co-participant
BHR_SP_RELATIONSHIP.csv		Minimal	co-participant
BHR_SP_STUDY_CONFIRMATION.csv		Minimal	co-participant
BLCHANGE.csv		Detail	diagnosis
BLSCHECK.csv		Detail	diagnosis
CBBCOMP.csv		Detail	diagnosis
CCI.csv	True		npa
CDR.csv	True		npa
CLIELG.csv		Minimal	additional

block	Select	Exclusion	Category
CLIELG_ADNI3.csv		Minimal	additional
DXSUM_PDXCONV_ADNIALL.csv	True		diagnosis
ECOGPT.csv	True		npa
ECOGSP.csv	True		npa
EXCLUSIO.csv		Minimal	additional
FAMXHPAR.csv		Redundant	family-history
FAMXHSIB.csv		Redundant	family-history
FAQ.csv	True		impair
FCI.csv	True		impair
FHQ.csv		Excluded	family-history
GDSCALE.csv	True		nps
INCLUSIO.csv		Redundant	additional
INITHEALTH.csv	True		health-history
ITEM.csv		Minimal	npa
MEDHIST.csv	True		medications
MMSE.csv	True		npa
MOCA.csv	True		nps
MODHACH.csv	True		health-history
NEUROBAT.csv	True		npa
NEUROEXM.csv	True		physical
NEUROPATH_05_17_21.csv		Minimal	diagnosis
NEUROPATH_07_06_21.csv		Minimal	diagnosis
NEUROPATH_DICT_05_17_21.csv		Minimal	diagnosis
NPI.csv	True		nps
NPIQ.csv	True		nps
NPSTATUS.csv		Detail	additional
PHYSICAL.csv	True		physical
PTDEMOG.csv	True		demographics
RECADV.csv		Detail	health-history
RECBLLOG.csv		Detail	health-history
RECCMEDS.csv		Detail	medications
RECFHQ.csv		Detail	health-history
RECMHIST.csv		Detail	health-history
REGISTRY.csv		Detail	additional
ROSTER.csv		Detail	additional
STUDYSUM.csv		Detail	additional
TREATDIS.csv		Detail	additional
UCBERKELEYAV45_04_26_22.csv	True		nimg
UCSDVOL.csv		Excluded	nimg

block	Select	Exclusion	Category
UCSFFSX51__11__08__19.csv	True		nimg
UCSFFSX6__08__17__22.csv	True		nimg
UWNPSYCHSUM__12__13__21.csv	True		npa
VISITS.csv		Detail	additional
VITALS.csv	True		physical

Bogdanovic, Bojan, Tome Eftimov, and Monika Simjanoska. 2022. "In-depth insights into Alzheimer's disease by using explainable machine learning approach." *Sci. Rep.* 12 (1).

Ferri, F. J., P. Pudil, M. Hatef, and J. Kittler. 1994. "Comparative Study of Techniques for Large-Scale Feature Selection. This Work Was Suported by a SERC Grant GR/E 97549. The First Author Was Also Supported by a FPI Grant from the Spanish MEC, PF92 73546684." In *Pattern Recognition in Practice IV - Multiple Paradigms, Comparative Studies and Hybrid Systems*, 403–13. Elsevier. https://doi.org/10.1016/b978-0-444-81892-8.50040-7.

McCombe, Niamh, Shuo Liu, Xuemei Ding, Girijesh Prasad, Magda Bucholc, David P Finn, Stephen Todd, Paula L McClean, and Kongfatt Wong-Lin. 2022. "Practical Strategies for Extreme Missing Data Imputation in Dementia Diagnosis." *IEEE Journal of Biomedical and Health Informatics* 26 (2): 818–27. https://doi.org/10.1109/JBHI.2021.3098511.

Pedersen, A B, E M Mikkelsen, D Cronin-Fenton, N R Kristensen, T M Pham, L Pedersen, and I Petersen. 2017. "Missing data and multiple imputation in clinical epidemiological research." *Clin Epidemiol* 2835. https://doi.org/10.2147/CLEP.S129785.

Pedregosa, F, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.