# Use of multiple machine learning techniques

For a given problem there are number of conditions and perquisites e.g. data sample size, type of data, correlation, number of classes, type of problem hence experimentation is required to decide the right machine learning model. The studies suggest building multiple models and use with the best metric. This project involves experimentation for different features are selected different models, machine learning techniques and metrics to provide a comprehensive detail to select the right model for developing the web application. The models applied to distinguish between AD and MCI patients and healthy control include Logistic Regression, linear discriminant analysis (LDA), k-nearest neighbours algorithm (kNN), decision trees(CART), random forest(RF), naïve Bayes (NB), Neural Network, support vector machine (SVM) and XGBoost. Ensemble learning is also used. The process involves combining the result of multiple models to produce a single output. It improves the performance of the models. The process requires data of lots of features which are not correlated.

The performance of the models if evaluated through confusion matrix and AUC-roc curve. Confusion matrix is a table to describe the performance of a classification model. The figure.. <https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62>

includes the

true positives (TP): Cases which are predicted as true and are actually true

true negatives (TN): Cases which are predicted as false and are actually false

false positives (FP): Cases which are predicted as true but are actually false

false negatives (FN): Cases which are predicted as false but are actually true

classification task with more than two classes presents different challenges than binary classifcations. It assumes that each sample is assigned to one and only one class e.g. a diagnosis can either be mild cognitive impairment or dementia but not both at the same time. Imbalanced dataset is when classes are not represented equally results in skewed distribution and hence certain machine learning techniques are difficult to implement. To handle class imbalance smote technique is being used. This object is an implementation of SMOTE - Synthetic Minority Over-sampling Technique. SMOTE uses a nearest neighbors algorithm to generate new and synthetic data we can use for training our model.

Again, it’s important to generate the new samples only in the training set to ensure our model generalizes well to unseen data.

<https://towardsdatascience.com/methods-for-dealing-with-imbalanced-data-5b761be45a18>

With imbalanced datasets accuracy is not a good evaluation metric and confusion matrix is a better technique to summarize the performance measure how model is performing on classifying various classes.

<https://towardsdatascience.com/machine-learning-multiclass-classification-with-imbalanced-data-set-29f6a177c1a>

“based on confusion matrix other metrics can be calculated. F1-score, area under the ROC curve are available for binary classification problems. To generalize this to multi-class, assuming we have a One-vs-All (OvA) classifier, we can either go with the “micro” average or the “macro” average. In “micro averaging,” we’d calculate the performance, e.g., precision, from the individual true positives, true negatives, false positives, and false negatives of the the k-class model:”

<https://sebastianraschka.com/faq/docs/multiclass-metric.html>

The confusion matrix is good for calculating the AUC-ROC curve. AUC-ROC curve. It is a 2D graph where the x-axis is the measure of the true positive rate or recall

TPR = TP / TP + FN

while the y-axis is the measure of false positive rate or miss rate

FPR = FP / FP + TN

<https://en.wikipedia.org/wiki/Sensitivity_and_specificity>

The model with high performance has more Area under the curve (AUC) with top left corner of the plot as an ideal point. IT is where FPR is 0 and TPR is 1. It is also ideal to maximize TPR while minimizing FPR. It is commonly used to visualize the performance of the binary classifier. This project is for multi-classification and hence it is necessary to binarize the output. One ROC curve is drawn per class and after using macro-averaging, which gives equal weight to the classification of each class.

<https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html#multiclass-settings>

# Interpreting Model Prediction

The modelling is built upon the principle that small differences can be used to find patterns to determine the various classes. However, models in ideal situation are going to be true when patients are being classified in the real-life (Zhang, Simon and Yu, 2017)

The reason why the model makes a prediction is cruial to explain the models to the end user. It helps in early adoption and understanding, confidence in the model. However, good performace models are often complex and difficult to interpret. Such as ensemble or deep learning. This project uses a framework for interpreting predcitions from the model called SHAP (SHapley Additive exPlanations). SHAP assigns each feature an importance value for a particular prediction based on cooperative game theory. “The Shapley value is the average marginal contribution of a feature value across all possible coalitions.”

<http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions>

# Data Description

The data has 12741 rows and 1907 columns. There are 1737 unique patient values of which 957 are males and 780 are females. Patients with APOE4 with values of 0, 1 and 2 are 522, 298, 62 respectively. Most of the patients have 204, 171, 128 of 16 , 18 and 20 years of education. Surprisingly some of the subjects had only few years of education e.g., 1 for 6 years, 5 for 8 years. Divorced :, married 653, never married 33 and widowed 99. There are different races of the subject with 819 as white and 35 as black.

There are 3837 rows with where the diagnosis is unknown and the rest are as follows:

MCI 3932

NL 2668

Dementia 1732

MCI to Dementia 372

NL to MCI 108

MCI to NL 77

Dementia to MCI 12

NL to Dementia 3

Find how many unique patients had the condition for both training and test set.

The experiments have been designed to process the data

**Experiment 1**

The unknown diagnosis has been not selected and there are only three diagnosis with the values replaced NL to MCI': 'MCI', 'MCI to Dementia': 'Dementia', 'MCI to NL' : 'NL', 'NL to Dementia': 'Dementia', 'Dementia to MCI': 'MCI. Therefore there were only three diagnosis codes with the counts as MCI 4052

NL 2745

Dementia 2107

In the dataset the actual age of the subject as they visit is not known. Few features are selected based on the literature review from different categories . A few feature is created which tells the age of the subject at their visit. Also the monthly change in their condition for various features 'AGE\_UPDATED', 'FDG\_CHANGE', 'AV45\_CHANGE', 'CDRSB\_CHANGE', 'ADAS11\_CHANGE', 'MMSE\_CHANGE',

'RAVLT\_immediate\_CHANGE', 'Hippocampus\_CHANGE', 'WholeBrain\_CHANGE', 'Entorhinal\_CHANGE', 'MidTemp\_CHANGE', 'ABETA\_UPENNBIOMK9\_04\_19\_17\_CHANGE',

'TAU\_UPENNBIOMK9\_04\_19\_17\_CHANGE', 'PTAU\_UPENNBIOMK9\_04\_19\_17\_CHANGE'

' is created to find the change in their condition. Also other factors such as 'PTGENDER', 'PTEDUCAT', 'PTETHCAT', 'PTRACCAT', 'PTMARRY', 'APOE4', are converted into categorical features

The missing values are filled with 0.0 to create a table which has no visit and recorded by the expert hance there was no condition.

The dataset is divided into training set with 8841, 36 columns and test set with 5177 and 36 columns to predict the three clinical diagnosis for each visit for the patient. Different models such as Logistic Regression, linear discriminant analysis (LDA), k-nearest neighbours algorithm (kNN), decision trees(CART), random forest(RF), naïve Bayes (NB), Neural Network, support vector machine (SVM) . SVM and KNN are the best performing models with after using one vs the rest classifier

One-vs-the-rest (OvR) multiclass/multilabel strategy

Also known as one-vs-all, this strategy consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes. In addition to its computational efficiency (only n\_classes classifiers are needed), one advantage of this approach is its interpretability. Since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier. This is the most commonly used strategy for multiclass classification and is a fair default choice. <https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html>

**Confusion Matrix**

|  |  |
| --- | --- |
| **Model** | **Averaged Multiclass ROC AUC Score** |
| Logistic Regression | 0.595 |
| Linear Discriminative Analysis | 0.593 |
| K-nearest Neighbours | 0.612 |
| Decision Tree | 0.651 |
| Random Forest | 0.581 |
| Naïve Bayes | 0.524 |
| Neural Network | 0.60 |
| Support Vector Machine | 0.619 |

**AUC Score per class**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Normal** | **Mild Cognitive Impairment** | **Dementia** |
| Logistic Regression | 0.601 | 0.529 | 0.655 |
| Linear Discriminative Analysis | 0.599 | 0.526 | 0.655 |
| K-nearest Neighbours | 0.636 | 0.594 | 0.608 |
| Decision Tree | 0.676 | 0.606 | 0.672 |
| Random Forest | 0.594 | 0.510 | 0.661 |
| Naïve Bayes | 0.536 | 0.508 | 0.529 |
| Neural Network | 0.622 | 0.540 | 0.658 |
| Support Vector Machine | 0.605 | 0.563 | 0.573 |

**AUC\_ROC SVM**

# Experiment 2

Includes few features

Here is a list of biomarkers we suggest participants unfamiliar with ADNI data to start with:

* The main measures to be predicted: DX, ADAS13, Ventricles
* Cognitive tests: CDRSB, ADAS11, MMSE, RAVLT\_immediate
* MRI measures: Hippocampus, WholeBrain, Entorhinal, MidTemp
* PET measures: FDG, AV45
* CSF measures: ABETA\_UPENNBIOMK9\_04\_19\_17  (amyloid-beta level in CSF), TAU\_UPENNBIOMK9\_04\_19\_17 (tau level), PTAU\_UPENNBIOMK9\_04\_19\_17 (phosphorylated tau level)
* Risk factors: APOE4, AGE

The age is calculated according to the visit. Also education and gender is taken into consideration. Gender is made into dummy variables . The missing values are imputed and scaled to predict the three clinical status. Models include decision tree with multiple leaf nodes ranging from 5, 50, 500, 5000 , random forest and xgboost with one vs rest classifier . deciioon tree with 50 nodes performs the best

**AUC Score per class**

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Normal** | **Mild Cognitive Impairment** | **Dementia** |
| Decision Tree | 0.903 | 0.748 | 0.898 |
| Random Forest | 0.724 | 0.600 | 0.877 |
| XGBoost | 0.908 | 0.659 | 0.853 |
| XGBoost with Parameter Tuning | 0.876 | 0.540 | 0.815 |

Experiment sub-section

'CDRSB', 'ADAS11', 'MMSE', 'RAVLT\_immediate', 'WholeBrain', 'AGE\_UPDATED'

XGBoost Results with Parameter Tuning

{1: 0.6201996211192528, 2: 0.5423566284773178, 3: 0.9018423363923679}

Experiment 3

Same as above with all the clinical diagnosis

**AUC Score per class**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Normal** | **Normal to Mild Cognitive Impairment** | **Normal to Dementia** | **Mild Cognitive Impairment** | **Mild Cognitive Impairment to Normal** | **Mild Cognitive Impairment to Dementia** | **Dementia to Mild Cognitive Impairment** | **Dementia** | **Unknown** |
| Decision Tree | 0.62 | 0.5 | 0.5 | 0.75 | 0.5 | 0.5 | 0.5 | 0.93 | 0.98 |
| Random Forest | 0.56 | 0.49 | 0.5 | 0.74 | 0.5 | 0.53 | 0.5 | 0.94 | 0.98 |
| XGBoost | 0.55 | 0.5 | 0.5 | 0.74 | 0.5 | 0.51 | 0.5 | 0.94 | 0.98 |
| XGBoost with Parameter Tuning | 0.63 | 0.50 | 0.5 | 0.77 | 0.5 | 0.53 | 0.5 | 0.95 | 0.98 |

Few features:

XGBoost Results with Parameter Tuning

{1: 0.5649078420739002, 2: 0.5, 3: 0.5, 4: 0.74721778783109, 5: 0.5, 6: 0.505801502605132, 7: 0.5, 8: 0.9390951656605728, 9: 0.9888937648328511}

### Imputation of lost values

For various reasons, many real-world data sets contain missing values, often encoded as blanks, NaNs, or other placeholders. However, these data sets are incompatible with scikit-learn estimators that assume that all values ​​in a matrix are numeric, and that they all have and have meaning. A basic strategy for using incomplete datasets is to discard rows and / or complete columns that contain missing values. However, this has the price of losing data that can be valuable (though incomplete). A better strategy is to impute the lost values, that is, to deduce them from the known part of the data.

The Imputer class provides basic strategies for imputation of missing values, using either the mean, the median or the most frequent value of the row or column in which the missing values ​​are found. This class also allows different encodings of missing values.

Experiment 4

It involves the same features as ­­­­­in experiment 2 but it inlvoves grid search and ensembling using voting classifier

#### Selected Parameters

After running RandomizedSearchCV several times, we found the most acceptable parameters for each of our models. We will save these parameters to then make the adjustment of our models.

#### Generating our models

So now let's prepare five learning models as our classification. All these models can be invoked conveniently through the Sklearn library and are listed below:

* random forest sorter
* AdaBoost classifier.
* Gradient Boosting classifer
* Support vector machine
* Extra Trees
* Random Forest : {'n\_estimators': 49, 'min\_samples\_split': 25, 'max\_features': 'auto', 'max\_depth': 22}
* 2° Extra Tree : {'n\_estimators': 60, 'min\_samples\_split': 8, 'max\_features': 'sqrt', 'max\_depth': 37}
* Adaboost : {'n\_estimators': 37, 'learning\_rate': 0.2}
* Gradient Boosting : {'subsample': 0.9, 'n\_estimators': 32, 'min\_samples\_split': 0.01, 'min\_samples\_leaf': 10, 'max\_features': 'sqrt', 'max\_depth': 25, 'loss': 'deviance', 'learning\_rate': 0.025, 'criterion': 'friedman\_mse'}
  + SVM {'C': 25, 'gamma': 0.01, 'kernel': 'linear'}
  + XGBoost {'subsample': 0.6, 'silent': False, 'reg\_lambda': 0.1, 'n\_estimators': 120, 'min\_child\_weight': 1.0, 'max\_depth': 6, 'learning\_rate': 0.01, 'gamma': 0.25, 'colsample\_bytree': 0.9, 'colsample\_bylevel': 0.4}

Soft Voting/Majority Rule classifier for unfitted estimators.

voting='soft', weights=[2,3,3,1,3])

**estimators : *list of (string, estimator) tuples***

Invoking the fit method on the VotingClassifier will fit clones of those original estimators that will be stored in the class attribute self.estimators\_. An estimator can be set to None or 'drop' using set\_params.

**voting : *str, {‘hard’, ‘soft’} (default=’hard’)***

If ‘hard’, uses predicted class labels for majority rule voting. Else if ‘soft’, predicts the class label based on the argmax of the sums of the predicted probabilities, which is recommended for an ensemble of well-calibrated classifiers.

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html>

Ensemble Results with Parameter Tuning

{1: 0.7399317161067308, 2: 0.6133780679753664, 3: 0.8907469968671983}

# Experiment 5

Includes all the variables which are not empty .

Other categorical variables such as 'VISCODE',

'SITE',

'COLPROT',

'ORIGPROT',

'EXAMDATE',

'DX\_bl',

'update\_stamp', were droped

Dummy variable for gender, marital status, race, created three clinical diagnosis

Used variance threshold of over 0.90 and the random forest classier , Univariate feature selection

# Feature selection

Feature selection is an approach followed to increase the accuracy of a model by only having the important predictive feature variables of the given data. It can be seen that most of the approaches have done feature selection in the motive of increasing the accuracy of the model. But it is also necessary to note that the feature variables in the data that has been used by them is vast. Hence, it is safe to conclude that only when there is a huge set of feature variables, feature selection is necessary.

Good for the formulas etc…

(Zhang and Sejdić, 2019)

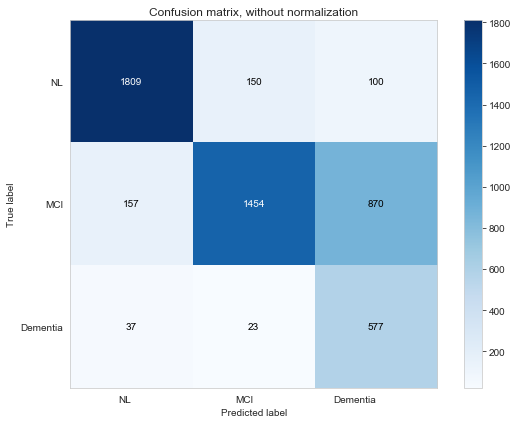
Select top 20 features using chi2chi2 test. Features must be positive before applying test.

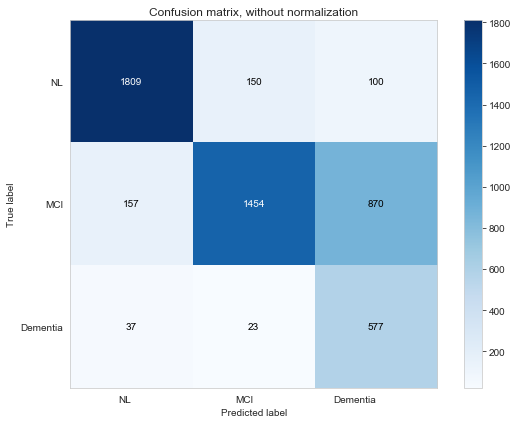
To select 54 features from 1452

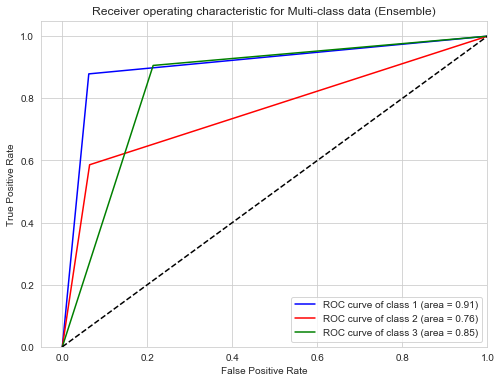
PCA visualization was done to see the variance explained

No imputation was done, however, scaling was done.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Normal** | **Mild Cognitive Impairment** | **Dementia** |
| Decision Tree | 0.88 | 0.74 | 0.82 |
| Random Forest | 0.89 | 0.77 | 0.84 |
| XGBoost | 0.901 | 0.685 | 0.808 |
| XGBoost with Parameter Tuning | 0.901 | 0.685 | 0.808 |
| Ensemble with Parameter Tuning | 0.908 | 0.760 | 0.846 |







Experiment 5 it is ensemble roc curve

# Defining a prediction function

As mentioned earlier, a prediction function will have to be de\_ned here that will do the

predictions in the web application.

A function named predictdisease has been de\_ned here

which will do the predictions with the help of decision tree model built and all the other

input parameters except the dependent variable needs to be fed into it. It is important to

note that without parsing a proper working model and all the input parameters correctly,

the function will not work or at least will not return the appropriate prediction value

which in this case is the status of the person.

After de\_ning the function successfully, it

can be published as a web service which will generate the API key and the location.

# Web application development

The process of development of the web application begins once the predictive models are

built and the one with the maximum accuracy is chosen. Considering the research gap

and the drawbacks of the existing methods, the aim here is to build a simple, easy to

use application which is not too overwhelming or complicated for the people to use.

The development of a web application involves the following steps:

Defining a prediction function that is responsible for performing the predictions.

The relevant input parameters are fed to the model with the highest accuracy

which in turn returns the prediction value which in our case is the result of whether

the person will have the disease or not.

\_ Once the prediction function is successfully defined, the model can be extended

and published as a web service which then helps generate an API location and key

enabling the model to be able to run on any platform.

\_ After publishing the web service, the API location and key is generated which is

then fed to the Request Response Web application. Once the connection between

the web application and the web service is successfully established, the user interface

of the application can be designed. Again, the web application must have a simple

user interface keeping in mind the needs and requirements of the user.

# Publish model as web service

To publish the model as a web service, the Azure ML package in R needs to be installed

and loaded. It also requires an Azure subscription and an Azure ML workspace which

can be created with the use of a Microsoft ID.

# Configuration and deployment of web application

Once the model is published as a web service, an API key and location gets generated

which can be seen in the R console. These are used to create and deploy the application

which after creation uses this API to run the model in the backend and produce the

prediction results in the web application to the user. Initially, a request response web

application is deployed in Azure. After successfully deploying the application, the API

key and the location is given here after which the user interface of the application can

be con\_gured i.e. the minimum and maximum value of each input is de\_ned here as

required and the web application is deployed. Now, as the deployment is completed, the

input values can be given which will return the prediction values i.e. if the person will

have the disease or not. As all the variables of the models are transformed into numerics

for prediction purposes, entering exact numeric value for each input would be di\_cult for

the user. Thus, a slider button is designed enabling the user to only give the approximate

values as input after which the prediction results will be given. The application has been

designed in a simple, elegant way where the user has to answer very simple questions to

get the results which is the whole point of this research anyway.

# Discussion

We examined how informative cortical models and cortical met-

rics estimates [51] and their combinations are for distinguishing between AD subjects and healthy control subjects. We also eval- uated the performance of several machine learning methods in this task. Due to the limited data size, 10-fold cross-validation was adopted for statistical consistency and robustness of the results. Our results demonstrate that all classifiers considered in our work perform better when using cortical metrics rather than cortical models for capturing the irregularities in of the cortical surfaces by means of fractal dimensions. In addition, we found that the combination of cortical models and corti- cal metrics yields lower performance. This can be attributed to features redundancy and making the classification models more complex. Due to feature redundancy, the cortical models do not add information. However, the classification models become more complex because they need to consider the cortical mod- els.

Our results reveal that the best accuracy of any other classifier

we applied is achieved when it is trained with the combination of cortical metrics and ADAS cognitive test results. This combination of features improves the generalization capacity of the CAD system for detecting AD. In other words, to build up a robust CAD sys- tem for accurately classifying AD patients and control subjects, it is recommended to use both neuroanatomy features and cognitive assessment results To further evaluate the effectiveness of the CAD system incor-

porating the SVM classifier trained with cortical metrics and ADAS cognitive test results, we compare our results with those obtained in recent studies. Specifically, Table 2 compares our results with those obtained by representative state-of-the-art methods of anal- ysis of structural magnetic resonance images to classify AD patients and healthy controls using the same dataset (ADNI). The SVM trained with cortical metrics and ADAS cognitive test results out- performs all previous works. Therefore, our findings validate the combination of cortical metrics, ADAS cognitive test results and SVM for accurate distinction between AD and healthy control sub- jects.

# Conclusion

We investigated the effectiveness of several machine learn-

ing classifiers in classifying AD and healthy control subjects when trained with cortical models, cortical metrics, ADAS cognitive test results or their combinations. We found that all classifiers con- sidered in this work perform better with cortical metrics (cortical thickness and gyrification index) than with cortical models used to capture fractality of the cortical ribbon, pial surface, and gray/white matter boundary surface. Second, combining cortical models and cortical metrics significantly reduces performance of all classifiers. Third, the combination of cortical metrics and ADAS cognitive test results allows achieving the highest performance. Fourth, such CAD system for AD detection outperformed results from previous stud- ies. We conclude that integration of neuroanatomical features and cognitive tests is a promising route to designing effective CAD sys- tems for distinguishing AD and healthy subjects.

(Lahmiri and Shmuel, 2018)

# Future work

Deep learning

Trying different imputing and scalar methods.

There are various techniques involved in improving the performance of imbalanced datasets.

#### Re-sampling Dataset

To make our dataset balanced there are two ways to do so:

1. **Under-sampling:** Remove samples from over-represented classes ; use this if you have huge dataset
2. **Over-sampling:** Add more samples from under-represented classes; use this if you have small dataset

#### SMOTE (Synthetic Minority Over-sampling Technique)

SMOTE is an over-sampling method. It creates synthetic samples of the minority class. We use **[imblearn](https://imbalanced-learn.readthedocs.io/en/stable/generated/imblearn.over_sampling.SMOTE.html" \t "_blank)**python package to over-sample the minority classes .