Compulsory Exercise 2: Prediction of Alzheimers Disease from Longitudinal MRI Data

Chester Henry Charlton Kasper Eikeland Tinus Garshol

20 April, 2023

Abstract

While Alzheimer's disease is traditionally diagnosed by medical professionals, in this report we detail the use of statistical learning models to classify Alzheimer's disease. Longitudinal MRI data was used from a mix of nondemented and demented patients. This data also included several non-MRI features, such as a short objective mental exam, education level, and the final diagnosis of dementia, which was used as a target variable for the models. The models examined were random forest, support vector machines, and logistic regression. Support vector machines and decision trees were chosen based on applicability, and logistic regression was chosen due to its relevance and consistent use in the medical field. The selected models were trained on a training subset of the data independently and then compared using cross-validation on the training data. From this, the optimal model was selected for the minimal misclassification error. Finally, this model was evaluated on the test data as a means to assess real world performance of the model. While we determined that the logistic model provided the best classification given a simple misclassification error loss function, our confidence in this became muddled by evidence that this result was unstable.

Introduction: Scope and Purpose

The goal of this project is to develop a model that can classify the diagnosis of dementia, specifically Alzheimer's. To do this we use quantified data from MRI scans and an objective psychological test to form the basis of prediction. Data was found from Kaggle, however the original data was collated by the Open Access Series of Imaging Studies. The dataset includes MRI visits of 150 individuals between the ages of 60 and 96, providing data for several different visits of each patient to an MRI center, at which time a short psychological exam was also taken. In addition to this, the data includes a clinical diagnosis of dementia (either demented or nondemented) for the patient which we use as a target variable. While a simple and interpretable model is desirable, our primary goal was to develop a model that can rule out some patients from further treatment, as this would reduce the load on medical professionals, particularly doctors. As such, our main focus is prediction rather than inference since we are mainly concerned with our ability to emulate the assessment of doctors on these patients. Withstanding this, we chose a small but diverse set of models: Random forest decision trees, support vector machines, and logistic regression. Within these, logistic regression is by far the most interpretable model, and also is relevant if only for its ongoing use in the medical field. SVM's and random forests are less interpretable but provide diversity in the model set. Additionally, these provide an interesting contrast to the medical field's conventional model.

Data Availability

The data can be found at the following Kaggle page: https://www.kaggle.com/datasets/jboysen/mri-and-alzheimers?select=oasis_longitudinal.csv

We used the oasis_longitudinal.csv dataset.

Descriptive data analysis/statistics

Description of Features:

MRI.ID: The unique identifier of an MRI scan. This feature was not used.

Group: The group that the patient was classified in the across the various scans. Possible values are "Undemented," denoting a patient without symptoms of dementia throughout all scans, "Demented" denoting a patient with symptoms of dementia across all scans, or "Converted," denoting someone who began without symptoms of dementia but developed them at a further scan. We chose to throw away "Converted," since our primary goal is to develop a classification model for patients who either have dementia or not, and the Converted class complicates this since patients in this class would fall into different categories at different times.

Visit: The order of the visit in which the scan was done, indexed from 1. This data was not used for our model, but was used for processing the data.

MR Delay: The number of days since the previous scan. Marked as 0 for the initial scan. Similarly, this data was not used for the model directly, but helped us for data processing as will be described later.

M.F: Categorical variable for male or female patients at birth. M denotes male, F denotes female. This binary variable was included in the models, however feature selection points to it not being a decisive feature for classification.

Hand: The patient's dominant hand. All of the patients were right-handed. This feature was thrown out.

EDUC: The education level of the patient, measured in years (including elementary school).

SES: The socioeconomic status of the patients as classified by the Hollingshead Index. Values range from 1 (most elite status) to 5 (lowest status). This feature was thrown out since it as not consistently recorded accross the patients.

MMSE: The score of the Mini Mental State Examination. Ranges from worst (0) to best (30), and represents a measure of cognitive impairment. An example of this test is attached as an appendix. We determined from this example the that score is objective, and as such is relevant to use for an analysis where the goal is to reduce pressure on medical professionals. Due to the objective nature of the exam, it could theoretically be performed by an automated system.

CDR: The patient's clinical dementia rating, as assessed by a medical professional. Ranges from 0 to 3, with any score above 0 representing a form of dementia. This is our target feature for classification, which we transformed into a binary variable.

ASF: The atlas scaling factor of the patient. A measure standardizing the intra-cranial volume.

eTIV: Estimated total intracranial volume.

nWBV: Normalized whole-brain volume. This feature is measured as the percent of the possible total volume that is detected as grey or white matter.

Data Processing

The source data that we accessed had a number of peculiarities that needed to be resolved before we could use it. First and foremost, we noted that there were some features that were not feasible to use. These included Subject.ID, MRI.ID, Hand, and SES. Subject.ID is not useful to us as it is simply a unique identifier of the patient. This might have been useful if we were integrating this data into a larger context of data, but here it was not applicable. MRI.ID is similarly not useful since it is a unique identifier of the particular MRI scan. Following guidelines for tidy data, we are keeping each row as a unique observation (wide format) and so this is not necessary. Hand is a feature stating the dominant hand of the patient, and since all of the patients in this study were right-handed, it is not a valid feature to use as a predictor. SES denotes the socioeconomic status of the patient. This would have been an interesting feature to examine for predictive capacity, but it was not exhaustively collected like the other features and so many of the observations had this feature missing. Our two options to treat this were either to (1) throw out SES as a feature, or (2) remove observations that had this feature missing. Since such a large number of the observations were missing SES, we determined that it would be better to sacrifice it as a feature and preserve the size of our observation set.

Beyond removing features, the next peculiarity is the longitudinal nature of the dataset, meaning that it includes multiple observations of the same individual at different points in time. The methods that we have learned in this course do not pertain to time-series data, so our solution was to add features, where relevant, that capture the basic properties of the time series information. MMSE, eTIV, nWBV, and ASF were use to create delta features (denoted as dMMSE, deTIV, etc) that represent the difference between the first and last observations of a particular feature for a particular patient, standardized with respect for time. For example, if a particular patient has their first MMSE recording stated as 28 and last as 22 with 300 days in between the measurements, their dMMSE is $\frac{28-22}{300} = 0.02$. This allows us to use the information conveyed by the longitudinal data while still using the methods from this course.

Finally, the dataset includes three classes of diagnosis for a patient which are constant across their MRI visits:

> Nondemented: This patient does not currently have a dementia diagnosis and will not develop one over the course of their visits.

> Demented: This patient currently has a dementia diagnosis and will maintain that diagnosis over the course of their visits.

> Converted: This patient began their visits without dementia, and developed dementia over the course of their visits.

As stated before, our goal is to predict based on the features in the dataset whether a patient has dementia or not, and we included the longitudinal data by means of calculating the deltas. We saw this as incompatible with using the Converted class of patients, since across all of their visits we cannot say that they were either demented or nondemented in entirety. Therefore we removed the observations that were marked as Converted. This was not a significant loss, since there were only 14 such patients, leaving 136 observations to work with.

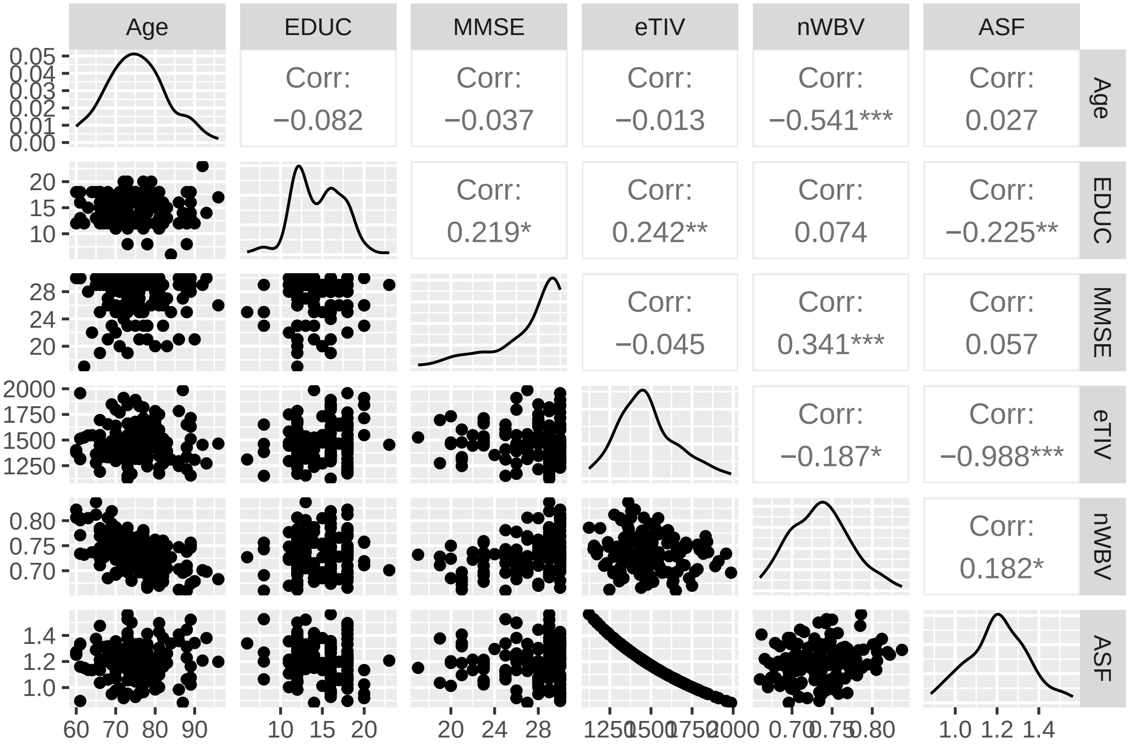
```
# Patient is demented if CDR is > 0
longitudinal.df <- read.csv("oasis_longitudinal.csv")
longitudinal_deltas <- longitudinal.df %>%
  group_by(Subject.ID) %>%
  summarise(dMMSE = (last(MMSE) - first(MMSE)) / last(Visit),
            deTIV = (last(eTIV) - first(eTIV)) / last(Visit),
            dnWBV = (last(nWBV) - first(nWBV)) / last(Visit),
            dASF = (last(ASF) - first(ASF)) / last(Visit),
            dAge = (last(Age) - first(Age)) / last(Visit)) %>%
  na.omit()

baseline_with_deltas <- longitudinal.df %>%
  filter(Visit == 1) %>%
  left_join(longitudinal_deltas, by = c("Subject.ID")) %>%
  filter(Group != "Converted") %>% # filter out converted cases
  mutate(Demented = as.factor(Group != "Nondemented"),
         M.F = as.factor(M.F)) %>%
  dplyr::select(-c(Subject.ID, MRI.ID, Group, Visit, MR.Delay, Hand, CDR, SES))
```

```
set.seed(1337)
# Longitudinal data
l.train.indices <- sample(1:nrow(baseline_with_deltas), floor(.75 * nrow(baseline_with_deltas)))
longitudinal.train <- na.omit(baseline_with_deltas[l.train.indices,])
longitudinal.test <- na.omit(baseline_with_deltas[-l.train.indices,])
```

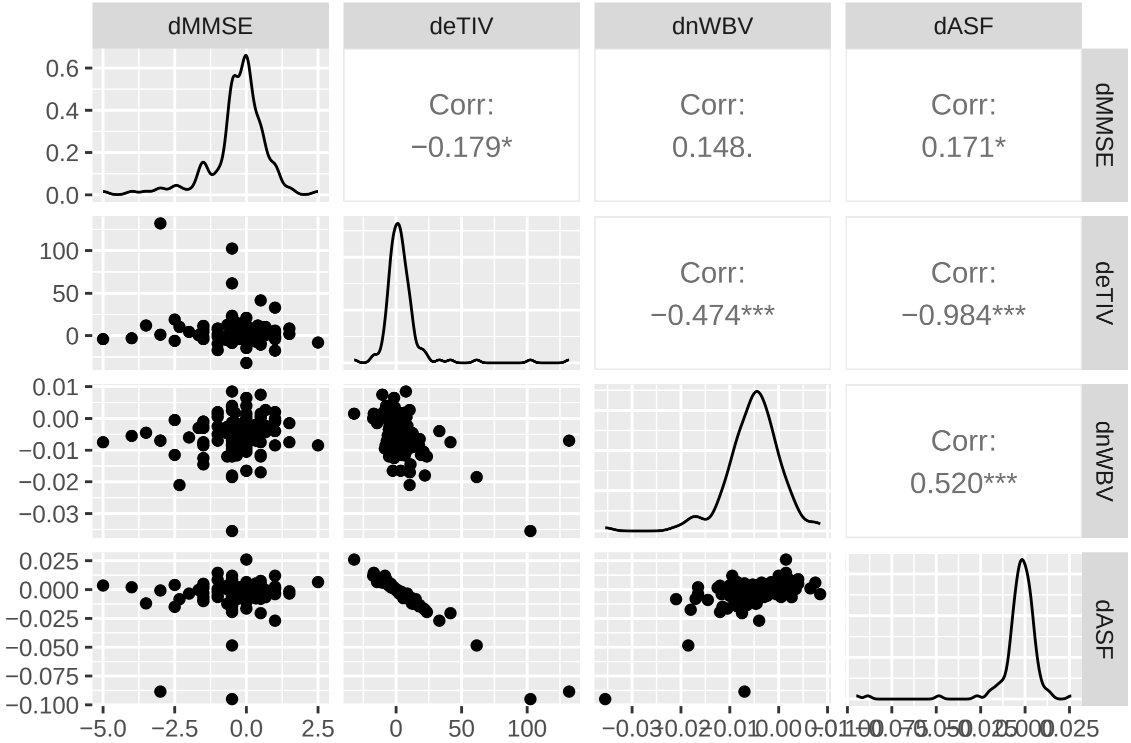
EDA:

```
# select some quantitative data and plot the pairs of it
pairwise_scatter_data <- baseline_with_deltas[,c("Age", "EDUC", "MMSE",
                                   "eTIV", "nWBV", "ASF" )]
pairwise_scatter_data.deltas <-  baseline_with_deltas[,c("dMMSE",
                                   "deTIV", "dnWBV", "dASF")]
# make pair plot for non-delta features
pair_plot <- ggpairs(data=pairwise_scatter_data)
# make pair plot for delta features
pair_plot.deltas <- ggpairs(data=pairwise_scatter_data.deltas)
```

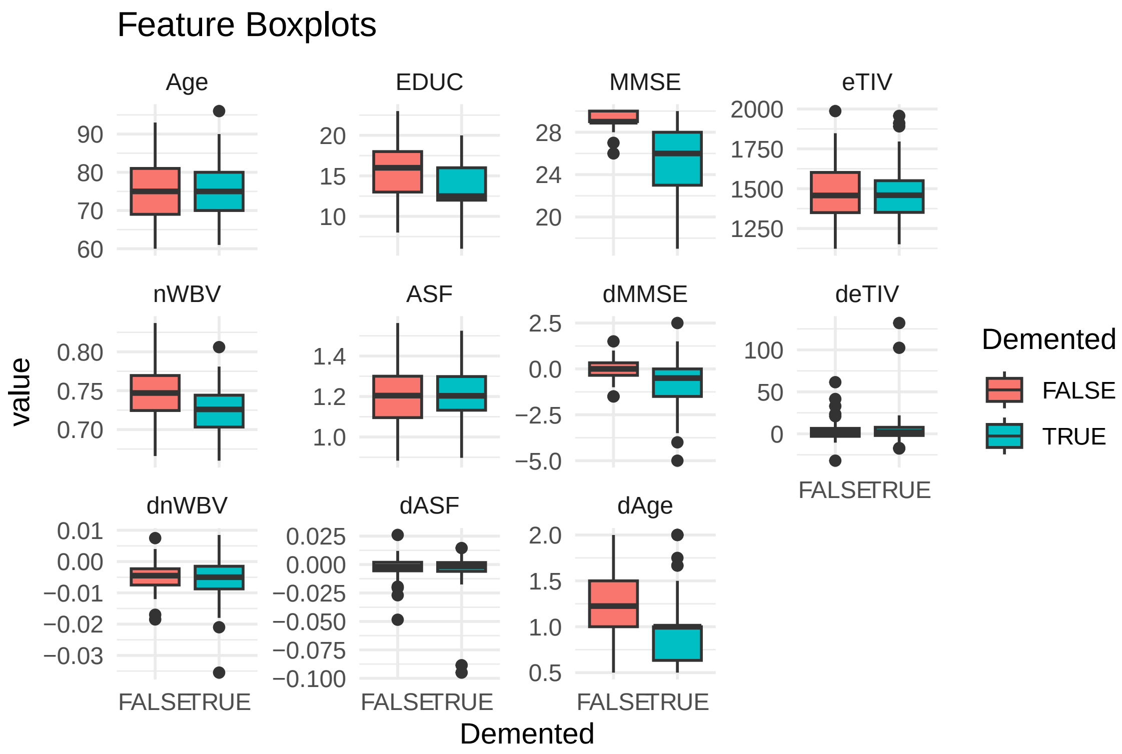
```
# boxplots
require(reshape2)
# melt data for ggplot boxplot of all relevant features
baseline_with_deltas.m <- melt(baseline_with_deltas[-1], id.var = "Demented")
# make boxplot of all relevant features features
boxplots <-
  ggplot(data = baseline_with_deltas.m, aes(x=Demented, y=value)) +
  ggtitle("Feature Boxplots") +
  geom_boxplot(aes(fill=Demented)) +
  theme_minimal()
boxplots.composite <- boxplots + facet_wrap( ~ variable, ncol=4, scales = 'free_y')




# show pair plot for non-delta features
pair_plot
```



```
# show pair plot for delta features
pair_plot.deltas
```



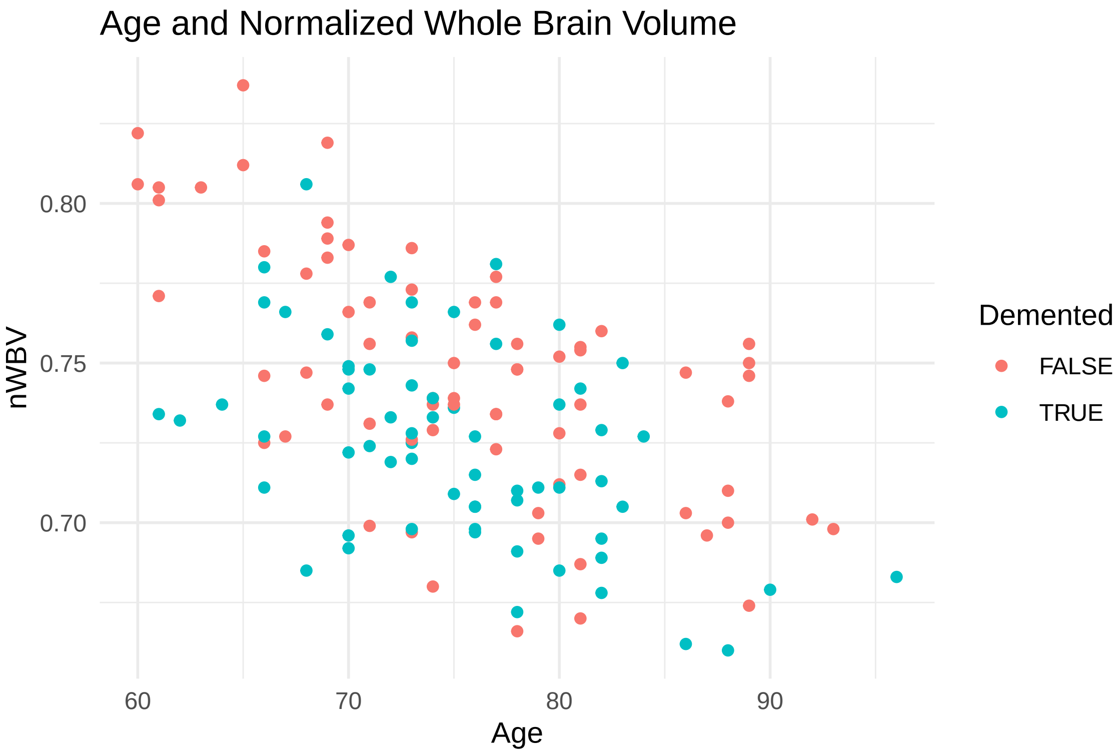
```
boxplots.composite
```



```
summary(baseline_with_deltas)
```

```
##   M.F          Age             EDUC             MMSE             eTIV
##   F:78   Min.   :60.00   Min.   : 6.00   Min.   :17.00   Min.   :1123
##   M:58   1st Qu.:70.00   1st Qu.:12.00   1st Qu.:26.00   1st Qu.:1350
##          Median :75.00   Median :14.00   Median :29.00   Median :1459
##          Mean   :75.28   Mean   :14.47   Mean   :27.38   Mean   :1478
##          3rd Qu.:80.00   3rd Qu.:16.00   3rd Qu.:30.00   3rd Qu.:1572
##          Max.   :96.00   Max.   :23.00   Max.   :30.00   Max.   :1987
##
##       nWBV             ASF             dMMSE             deTIV
##   Min.   :0.6600   Min.   :0.883   Min.   :-5.0000   Min.   :-32.000
##   1st Qu.:0.7085   1st Qu.:1.116   1st Qu.:-0.5000   1st Qu.: -2.750
##   Median :0.7365   Median :1.204   Median : 0.0000   Median :  2.000
##   Mean   :0.7359   Mean   :1.204   Mean   :-0.3101   Mean   :  4.679
##   3rd Qu.:0.7582   3rd Qu.:1.300   3rd Qu.: 0.2500   3rd Qu.:  7.500
##   Max.   :0.8370   Max.   :1.563   Max.   : 2.5000   Max.   :132.000
##                                    NA's   :1         NA's   :1
##      dnWBV                dASF                 dAge          Demented
##   Min.   :-0.035500   Min.   :-0.095000   Min.   :0.500   FALSE:72
##   1st Qu.:-0.008000   1st Qu.:-0.005583   1st Qu.:1.000   TRUE :64
##   Median :-0.004600   Median :-0.001750   Median :1.000
##   Mean   :-0.005193   Mean   :-0.003520   Mean   :1.111
##   3rd Qu.:-0.002000   3rd Qu.: 0.002000   3rd Qu.:1.333
##   Max.   : 0.008500   Max.   : 0.026000   Max.   :2.000
##   NA's   :1           NA's   :1           NA's   :1
```

```r
ggplot(data=baseline_with_deltas, aes(Age, nWBV, colour = Demented)) +
  ggtitle("Age and Normalized Whole Brain Volume") +
  geom_point() +
  theme_minimal()
```



Methods

We consider three different classifiers, logistic regression, random forests and support vector machines. We first introduce and describe the classifiers, then describe the methods used for performance evaluation and model selection.

Model Descriptions

Logistic regression This method was chosen as a candidate due to its continued use in the medical field and value as a method to give classification probabilities. This is especially useful in our case since we have a binary target variable (demented or nondemented) and a combination of binary and nonbinary predictors. Logistic regression fits a function for n variables that outputs a probability between 0 and 1 for each observation. It is in this sense that logistic regression is useful for classification, since with that set of probabilities for each observation, we can implement a decision threshold in order to make the prediction bivalent. For instance, we could choose a threshold of 0.6, and the process would go like this: Our logistic model outputs a probability of 0.4 that a particular observation belongs to the target class, and then that observation gets a classification of 0, denoting that it does not belong to the target class. The logistic model outputs a probability of 0.62 that a particular observation belongs to the target class, and then that observation gets a classification of 1, denoting that it belongs to the target class. By tuning this threshold we can achieve different characteristics for our model, such as different sensitivities and specificities for the classification. As we lower the threshold for the classification, the sensitivity increases and specificity decreased, and similarly when we raise the threshold, the sensitivity decreases and the specificity increases. As an example, imagine that we decreased the threshold entirely to its lowest possible value, 0. This would mean that every observation would be classified as within the target class, and it is trivially obvious that $P(A|B = 1.0)$, where A stands for "the observation is classified into the target class," and B stands for "the observation truly belongs to the target class.

Below is an outline of the overall strategy for how we fitted the logistic model:

1. Select 0.5 as classification threshold.
2. Perform AIC feature selection by training on entire training set. Record best feature set.
3. Perform 5-fold CV with best features on training set to assess variability of sensitivity, specificity, and misclassification error.
4. Perform 5-fold CV in parallel with selected SVM and Random Forest models to select final model for testing.

Classification threshold: We chose a probability threshold of 0.5 for this logistic model (see above for explanation of threshold importance). This is somewhat arbitrary, but we did not have the time to implement a calibration for this. In the future, we could divide our data into a training, validation, and test set, and then use the validation set to tune the probability threshold for a desired sensitivity and specificity balance.

Subset selection: We chose to use the Aikake Information Criterion. The possibilities we could have chosen for this were Cp, BIC, AIC, and Adjusted Rˆ2. Cp and Adjusted Rˆ2 are not appropriate since they rely on RSS, which is not compatible with a logistic function, and after this we arbitrarily chose AIC instead of BIC, which based on our knowledge will not be significant. AIC was then used using the *bestglm* package in R using the entire training set. We chose to use exhaustive feature selection since the time of computation is not problematic for this number of features and observations. The selected features were Age, MMSE, nWBV, and dMMSE. A plot of the AIC as a function of the number of variables is included below titled "Logistic AIC Subset Selection," showing that around 4 or 5 variables is the local minimum, which we prefer.

CV: We used cross validation mainly for getting an idea of how much the misclassification error rate, sensitivity, and specificity varied. While this was not used for model selection, it provided to us a sense of how much these measures could vary.

Random forests We use Random Forests as a classifier as described in [ISLR2]. Random Forests is a tree ensemble method that can be used for both regression and classification. Each tree is trained on a bootstrap sample of the training data. The splits are made using a random subset of $m < p$ predictors. This decorrelates the trees and presents an improvement over bagging thereof [cite ISLR]. For each tree we perform a classification on the Out-Of-Bag (OOB) observations, the training observations not in the bootstrap sample the given tree was trained on. In this sense, each tree votes for a given class for each OOB observation. We then aggregate all votes and the forest predicts the class for each observation in the training data as the one which got the majority of the votes.

This method of growing decision trees and making predictions offers for higher predictive accuracy at the cost of lower interpretability compared to standard classification trees. For our purposes we value the predictive accuracy higher. However, a case can be made for more interpretability as such a decision tree could in theory be used as an easy-to-follow clinical test for deciding whether or not a patient has AD based on similar data. Still, if the predictive accuracy of such a test is not high enough, its value as a clinical test is lowered.

We implement Random Forests through the R package `randomForest`.

Random Forests can be tuned by two parameters, the number of trees B and the amount of predictors to consider for each split m. The number of trees used to fit a random forest does not impact the bias-variance tradeoff as long as its high enough as high B does not lead to overfitting [ISLR]. Thus we may plot the cumulative OOB error estimate for a fitted Random Forest and see where an increased number of trees give no error decrease. The OOB error estimate is the aggregated misclassification rate calculated on the training observations not in the bootstrap sample the tree was trained on.

For this initial fit to decide the amount of trees we use the standard value for m in classification, $m = \sqrt{p}$. After choosing a number of trees, we may compare the OOB error estimate for different values of m and choose the model with lowest OOB error estimate.

Support vector machines Support vector machines are hyperplanes which separate the feature space and is commonly used in classification tasks. The core idea is that the method finds the optimal separating hyperplane given a set of points by solving a complex minimization task using kernel functions. The solution yields a set of support vectors…something The choice of a kernel function determines the shape of the hyperplane. The four most common ones are: linear, polynomial, radial and sigmoid. Usually, as the feature space increases data tends to become more linearly separable. So a linear kernel is a natural choice to test. Polynomial kernels increase rapidly in complexity as dimensions increase. Also, there is the issue of which degree polynomial one wants to choose. Larger degree polynomial hyperplanes quickly result in overfitting of data. Thus, for the sake of simplicity we will avoid testing this kernel. The RBF-kernel, or the radial basis function kernel is widely used in practice and often yields good results, so it would be interesting to compare with this one. The Sigmoid kernel is usually good for binary classification problems, but is avoided in this investigation. In pre-testing, the results did not seem promising for this kernel, so it will be avoided here. Although, it would be interesting to investigate it further in a more detailed SVM analysis.

A SVM with a linear kernel has a tuning parameter C, whilst a radial kernel has tuning parameters C and gamma. C is the "cost-function" and acts as a weight of how hard we penalize data that lies within the margin of the boundary separating the prediction space. Small C's result in large margins, but can lead to underfitting and vice versa. Gamma is closely related to the curvature of the separating boundary, or how stiff we allow the boundary to be. The value of these parameters result in a bias-variance tradeoff. Thus, for the radial kernel both parameters must be tuned simultaneously.

The "e1071" package in R has an svm function. It can also perform k-fold CV on the data which returns an accuracy metric which we will use to tune the model parameters.

The accuracy is a percentage score of how many predictions were correctly classified from CV.

Model Selection

To select a model we need to consider all possible classification methods within our selected scope that includes random forests, logistic, and SVM. Following this we then make a decision based on some measure of goodness-of-fit. For this measure we chose the misclassification rate. We primarily used this to compute a cross-validated error estimate for one model of each classification method once the "best" instance of our three classification methods had been chosen. The selection of the "best" instance of each classifier varies depending on the model type, and is discussed in the methods section of each model type. Rather than compare all model types at each step, we have chosen to only compare model instances within each classifier, only comparing the selected "best" instances of different model classifier types at the final step. This vastly reduces the size of the search space and thus decreases the computation time to a reasonable level. A graphical representation of our model selection algorithm is shown below.

Here we will provide an overview of the process we followed to arrive at a final model.

1. Collect the source data.
2. Process data: select relevant features, add deltas, remove time series data.
3. Separate processed data into training and test partitions.
4. Tune hyperparameters and perform feature selection for each model type.

5. Select best model from SVM, random forest, and logistic with selected hyperparameters and features, as relevant.
6. Evaluate best model on held-out test set.
7. Discuss results.

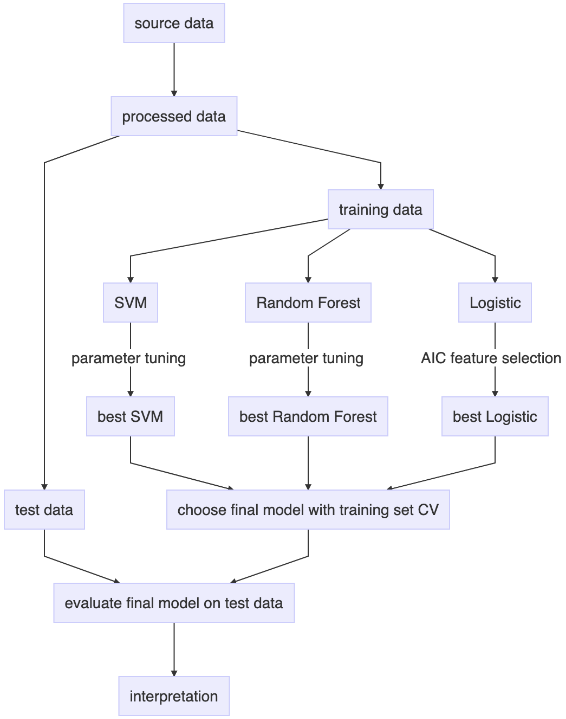


Figure 1: “Master Plan”

Model evaluation

Once we have selected the best models of the random forest, SVM, and logistic classifiers, we used 5-fold cross validation on the training dataset to select between them. For simplicity we chose the misclassification erro rate as the loss function. This means that our final model choice is dictated to be the model with the lowest misclassification error rate. Since we are assessing with 5-fold CV, we chose to use the average misclassification error rate of the 5 folds. Additionally, we use sensitivity and specificity as metrics to assess the performance of our models with respect to how they would be useful for real world applications. Below is a summary of their significance and how they were calculated.

Sensitivity: This represents the probability of classifying an observation as being in the target class, when it truly is in the target class. This is significant for analyzing and predicting how a model will misclassify observations in the future more specifically than just the misclassification error rate. We calculated this by dividing number of observations predicted to be in the target class by the total number of observations genuinely in the target class.

Specificity: This represents the probability of classifying an observation as not being in the target class, when it truly is not in the target class. This is significant for analyzing and predicting how a model will misclassify observations in the future more specifically than just the misclassification error rate. We calculated this by dividing number of observations predicted to not be in the target class by the total number of observations genuinely not in the target class.

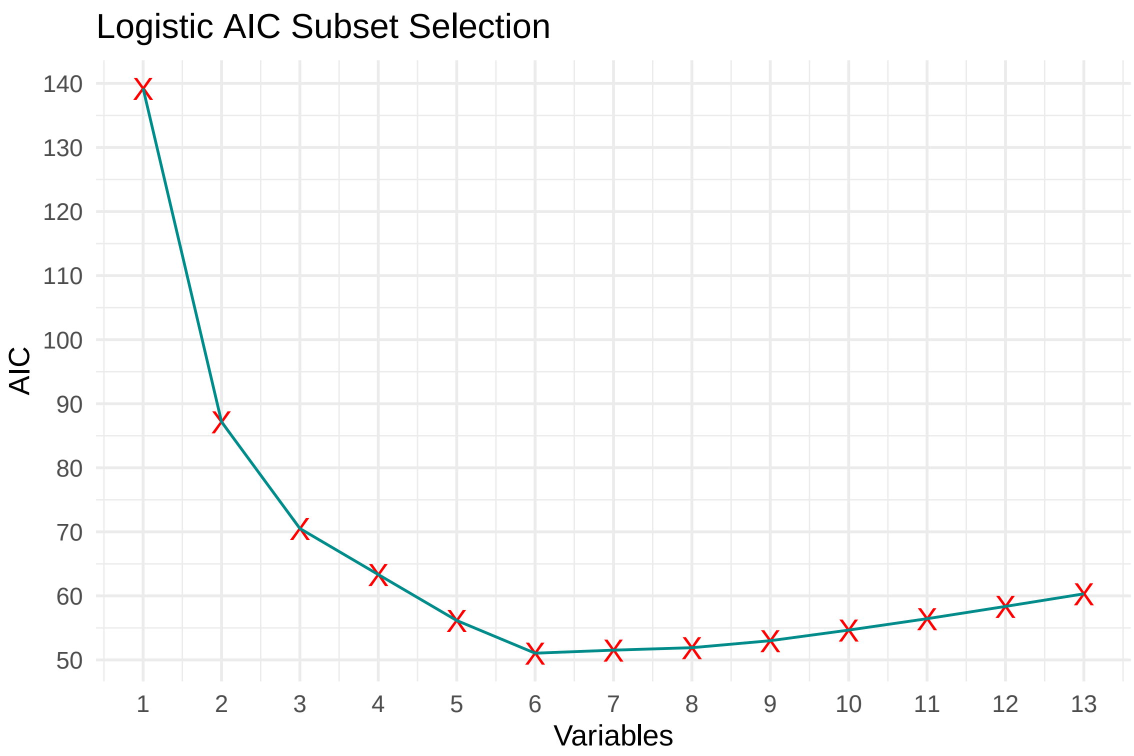
Misclassification error rate: This represents the probability of misclassifying an observation, in general. This is the single metric that we use to objectively judge our models' performances. This is calculated simply by dividing the total number of observations that were predicted to be in the wrong class by the total number of observations.

Results and interpretation

Logistic regression

Select optimal features Here we see that the ideal AIC subset selection includes features Age, MMSE, nWB, and dMMSE. By performing this features selection we are increasing the bias and decreasing the variance of the model, hopefully leading to a better test set performance.

```
# use AIC to select features
longitudinal.logistic.best.aic <- bestglm(longitudinal.train, family = binomial, IC = "AIC", method ="fo
logistic.best <- longitudinal.logistic.best.aic$BestModel
# get subset table as a dataframe for future use
longitudinal.logistic.aic.subsets <-
  longitudinal.logistic.best.aic$Subsets %>%
  data.frame() %>%
  cbind(variables = 1:nrow(longitudinal.logistic.best.aic$Subsets))
# make feature-AIC plot
longitudinal.logistic.aic.subsets.plot <-
  ggplot(longitudinal.logistic.aic.subsets, aes(x = variables, y = AIC)) +
  geom_point(color = "red", shape = "X", size = 4) +
  geom_line(color ='darkcyan') +
  ggtitle("Logistic AIC Subset Selection") +
  labs(x = "Variables", y = "AIC") +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
  theme_minimal()

longitudinal.logistic.aic.subsets.plot
```



```
set.seed(1337)
#randomize data
longitudinal.train.rand <- longitudinal.train[sample(nrow(longitudinal.train)),]

#make n folds
n <- 5
folds <- cut(seq(1,nrow(longitudinal.train.rand)),breaks=n,labels=FALSE)

#vector to store sensititivities
sens.vec <- c()
#specificities
spec.vec <- c()
#misclassivication errors
misc.vec <- c()
# cumulative confusion matrix
logistic.confusion.cumulative <- matrix(data = c(0,0,0,0), nrow=2,ncol=2)


for(i in 1:n){
    #get test index
    test.index <- which(folds==i,arr.ind=TRUE)
    #separate test data and train data
    test.data <- longitudinal.train.rand[test.index, ]
    train.data <- longitudinal.train.rand[-test.index, ]
    #train best logistic model for training partition
    logistic.i <- glm(Demented ~ Age + MMSE + nWBV + dMMSE, data = train.data, family = binomial)
    #generated predicted probabilities on test partition
    logistic.i.probs <- predict(logistic.i, newdata = test.data, type = "response")
    #convert probabilities to classification based on a P = 0.5 decision threshold
    logistic.i.pred <- ifelse(logistic.i.probs > 0.5, 1, 0)
    # generate confusion table for predictions
    logistic.i.confusion <- table(logistic.i.pred, test.data$Demented)
    rownames(logistic.i.confusion) <- c('pred 0','pred 1')
    colnames(logistic.i.confusion) <- c('true 0','true 1')
    # calculate sensitivity and specificity
    logistic.sens <- logistic.i.confusion[2,2] / sum(logistic.i.confusion[,2])
    logistic.spec <- logistic.i.confusion[1,1] / sum(logistic.i.confusion[,1])
    logistic.misc <- (logistic.i.confusion[2,1] + logistic.i.confusion[1,2]) / sum(logistic.i.confusion)
    #append accuracy metrics to persistent vectors
    sens.vec <- c(sens.vec,logistic.sens)
    spec.vec <- c(spec.vec,logistic.spec)
    misc.vec <- c(misc.vec, logistic.misc)
    #add confusion matrix to persistent object
    logistic.confusion.cumulative <- logistic.i.confusion + logistic.confusion.cumulative
}
# compute cumulative sensitivity, specificity and misclassification error accross the folds
sens.cumulative <- logistic.confusion.cumulative[2,2] / sum(logistic.confusion.cumulative[,2])
spec.cumulative <- logistic.confusion.cumulative[1,1] / sum(logistic.confusion.cumulative[,1])
misc.cumulative <- (logistic.confusion.cumulative[2,1] + logistic.confusion.cumulative[1,2]) /
  sum(logistic.confusion.cumulative)


sens_spec_misc <- c(sens.cumulative, spec.cumulative, misc.cumulative)
sens.vec
```

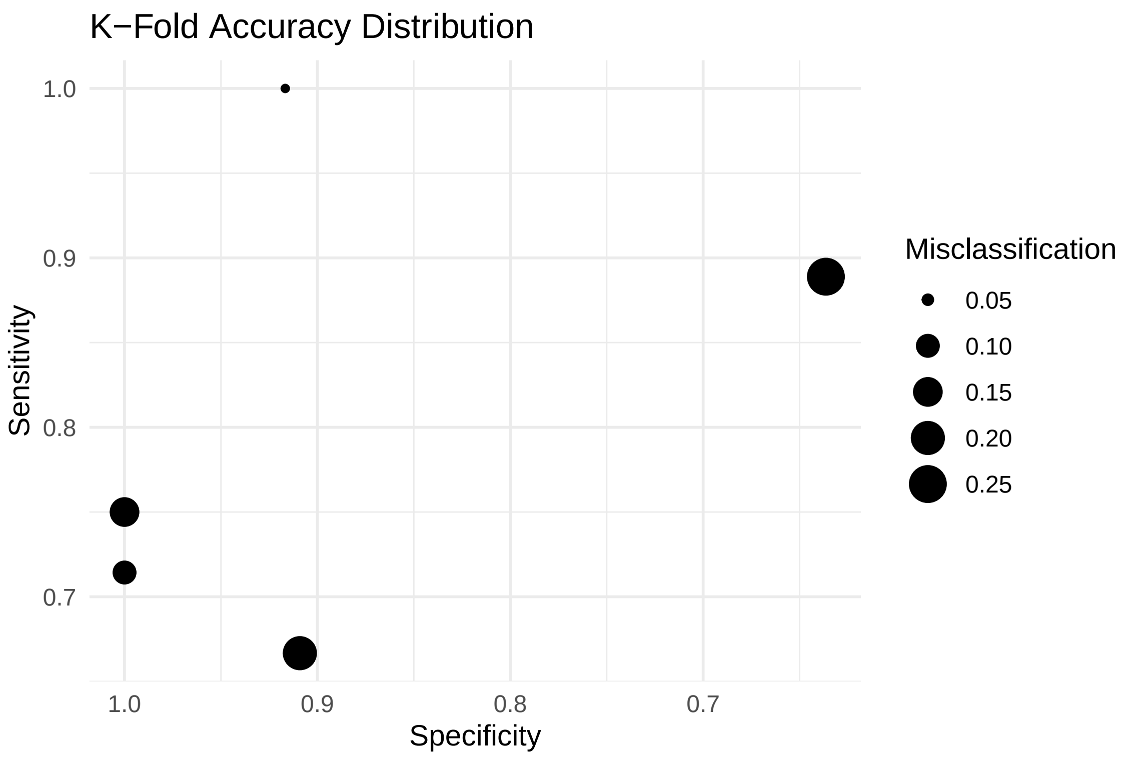
CV for evaluating the performance of fitting the logistic model, using previously calculated AIC feature set

```
## [1] 1.0000000 0.8888889 0.6666667 0.7142857 0.7500000
```

```
spec.vec
```

```
## [1] 0.9166667 0.6363636 0.9090909 1.0000000 1.0000000
```

```
# visualize some things
sens_spec_misc.df <- data.frame(sens = sens.vec, spec = spec.vec, misc = misc.vec)

logistic.accuracy.plot <-
  ggplot(sens_spec_misc.df, aes(x = spec, y = sens))+
  geom_point(aes(size = misc)) +
  labs(size = "Misclassification") +
  scale_x_reverse() +
  ggtitle("K-Fold Accuracy Distribution")+
  labs(x = "Specificity", y = "Sensitivity") +
  theme_minimal()


logistic.accuracy.plot
```



```
logistic.confusion.cumulative
```

```
##
## logistic.i.pred true 0 true 1
##          pred 0     49      9
##          pred 1      6     37
```

```
sens_spec_misc
```

```
## [1] 0.8043478 0.8909091 0.1485149
```

Note the above accuracy plot, we can see that over the course of 5 folds the accuracy measures that we chose (sensitivity and specificity) were all above 0.5, so the model is working even on an iterated basis. Additionally, we get a sense of the misclassification error rate.

```
# train logistic model on entire training set
longitudinal.logistic <- glm(Demented ~ Age + MMSE + nWBV + dMMSE, data = longitudinal.train, family = 
# extract training probabilities
longitudinal.logistic.train.probs <- longitudinal.logistic$fitted.values
# extract training classifications
longitudinal.logistic.train.preds <- ifelse(longitudinal.logistic.train.probs > 0.5, 1, 0)
# generate confusion table for model on training data
longitudinal.logistic.confusion <- table(longitudinal.logistic.train.preds, longitudinal.train$Demented
rownames(longitudinal.logistic.confusion) <- c('pred 0','pred 1')
```

```
colnames(longitudinal.logistic.confusion) <- c('true 0','true 1')
# sensitivity and specificity
longitudinal.logistic.sens <- longitudinal.logistic.confusion[2,2] / sum(longitudinal.logistic.confusio
longitudinal.logistic.spec <- longitudinal.logistic.confusion[1,1] / sum(longitudinal.logistic.confusio
# misclassification error
longitudinal.logistic.misc <- (longitudinal.logistic.confusion[2,1] + longitudinal.logistic.confusion[1
  sum(longitudinal.logistic.confusion)
#accuracy vector
sens_spec_misc <- c(longitudinal.logistic.sens, longitudinal.logistic.spec, longitudinal.logistic.misc)
sens_spec_misc
```

Train final model

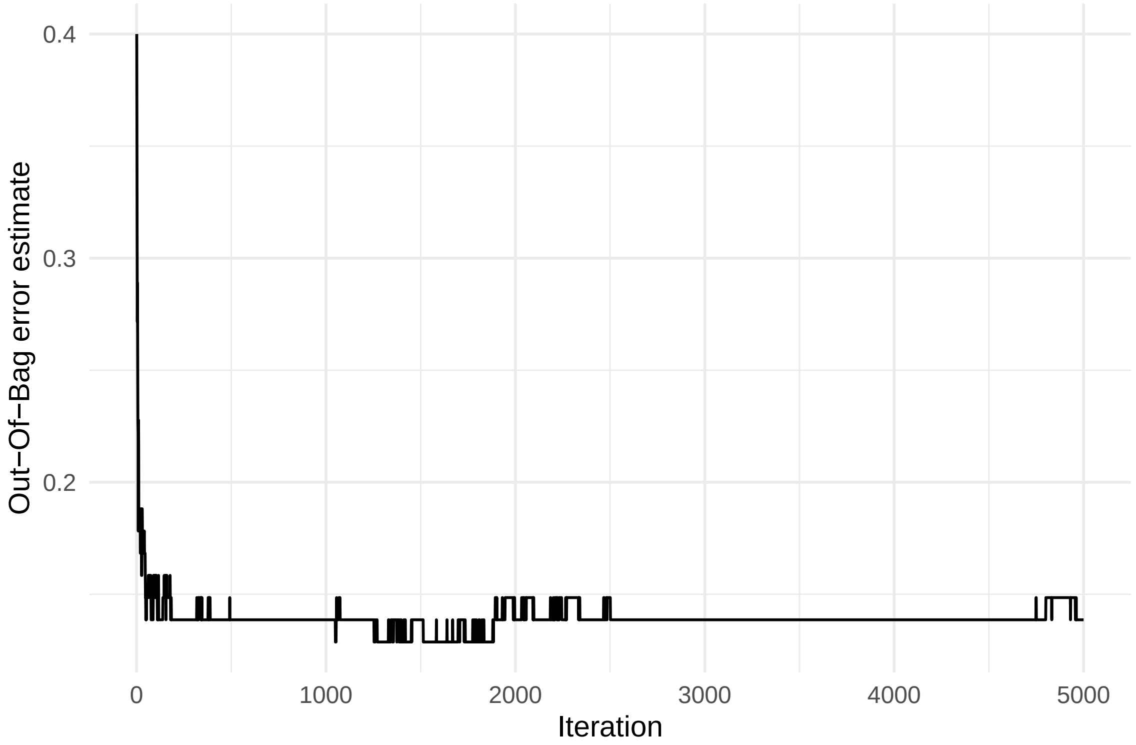
```
## [1] 0.8260870 0.9454545 0.1089109
```

Note that the final logistic model trained on our entire training set yields quite a nice sensitivity, specificity, and misclassification rate. Obviously we cannot conclude too much from this since it is not iterated, but it means that our model fitting is functioning as it should.

Random forest

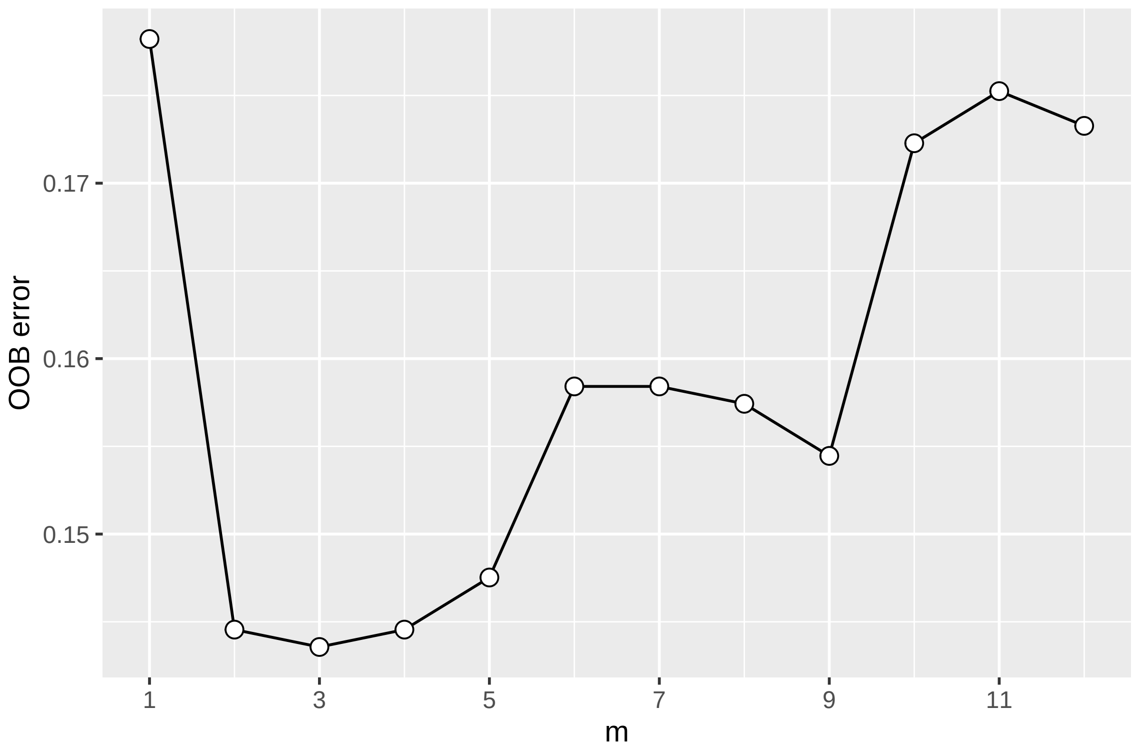
We proceed by fitting a Random Forest to the training set and tune the parameters to find the "best" instance of a Random Forest classifier. First we consider the number of trees in the forest, B. We fit a random forest and plot the cumulative OOB error.

```
set.seed(1337)

p <- ncol(longitudinal.train) - 1 # number of predictors
N <- nrow(longitudinal.train) # number of training observations


trainx <- select(longitudinal.train, -Demented)
trainy <- longitudinal.train$Demented

rf <- randomForest(Demented ~., data = longitudinal.train, ntree = 5000, mtry = sqrt(p), importance = T
err.df <- data.frame(x=1:5000, err = rf$err.rate[,1])

ggplot(err.df, aes(x=x)) +
  geom_line(aes(y = err)) +
  labs(x = "Iteration", y = "Out-Of-Bag error estimate") +
  theme_minimal()
```



We see that 5000 trees is enough as the OOB error estimate stabilizes and it is not too slow to fit the model. Now we choose a value for m. We fit 10 models for each $m \in \{1, ..., 12\}$ and compare the average OOB error for all values of m. Seeing as the amount of predictors is small enough so it is computationally feasible.

```
set.seed(1338)
B = 5000
oob.df <- data.frame(m = 1:p, oob.err = rep(0,p))

for (m in 1:p){
  oob_err_m <- matrix(nrow = 10, ncol = B)
  for (i in 1:10) {
    model <- randomForest(Demented ~., data = longitudinal.train,
                          mtry = m, ntree = B)
    oob_err_m[i] <- model$err.rate[B,1]
  }
  oob.df$oob.err[m] <- colMeans(oob_err_m) # save oob error
}

m <- oob.df$m[which.min(oob.df$oob.err)] # save best m value

ggplot(data = oob.df, aes(x = m)) +
  geom_line(aes(y = oob.err)) +
  geom_point(aes(y = oob.err), shape = 21, colour = "black", fill = "white", size = 2.8 ) +
  scale_x_continuous(breaks = seq(1,p,2)) +
  labs(x = "m", y = "OOB error")
```



Here we see that the best value for m is $m = 3$. Thus our best fit of a random forest uses $B = 5000$ trees and $m = 3$ randomly chosen predictors.

SVM

We now fit a Support Vector Machine to the training data. To start, we stardardize the training observations.

```
st.long.train <- longitudinal.train
# Standardize/normalize all the numeric features
st.long.train[,2:12] <- scale(st.long.train[,2:12])
```

We run an initial test using the linear and radial kernels under default parameters and test their accuracy on the training set with 10-fold CV. The accuracy is printed for the linear kernel then the radial kernel.

```
set.seed(58008)
# Linear kernel
lin.svm <- svm(Demented~.,data=st.long.train,kernel="linear",cross=10)
print(lin.svm$tot.accuracy)
```

```
## [1] 85.14851
```

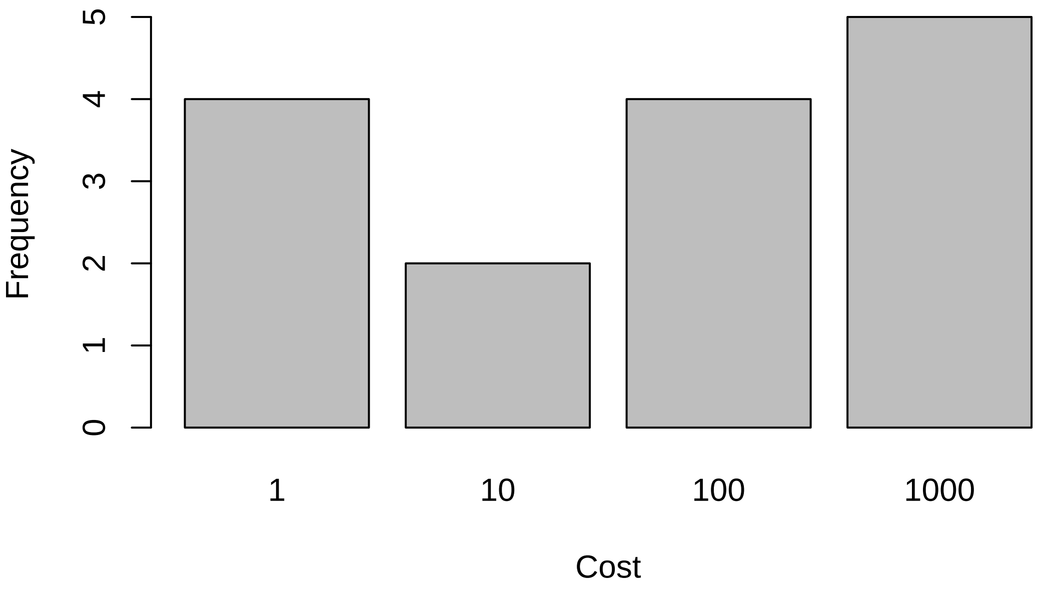
```
# Radial kernel
rad.svm <-svm(Demented~.,data=st.long.train,kernel="radial",cross=10)
print(rad.svm$tot.accuracy)
```

```
## [1] 81.18812
```

Both kernels perform decent, with accuracy $> 75\%$. The linear kernel performs slightly better under default choice of parameters.

C is the only tuning parameter for a linear kernel, so we start there. Here we will perform 15 iterations testing values of C in $[10^{-4}, 10^{3}]$ (by orders of magnitude). For each run, we will choose the best C.

```r
set.seed(58008)
# Testing accuracy for different costs "C"
# We'll do multiple passes with k-fold CV using the training data.
# Range of costs, by orders of magnitude (C=10^4 results in max iteration problems)
# Linear kernel
C = 10^(-4:3)
best_cost_lin = c()
for (j in 1:15) {
  accuracy = c()
  for (i in 1:length(C)) {
    lin.svm <- svm(Demented~.,data=st.long.train,kernel="linear",cost=C[i],cross=10,scale=FALSE)
    accuracy <- c(accuracy,lin.svm[29])
  }
  index <- which.max(accuracy)
  best_cost_lin <- c(best_cost_lin,C[index])
}
tab <- table(best_cost_lin)
y <- names(tab)
x <- as.numeric(tab)
barplot(x, names.arg = y, ylab = "Frequency", xlab = "Cost")
```



A cost of 1000 seems to be most prevalent after 15 runs.

For the radial kernel, we tune gamma and cost simultaneously as these parameters are correlated. The e1071 library has a built-in tune function which performs this analysis. We'll do 10 iterations, with gamma in $[10^{-4}, 10^3]$ and cost in $[10^{-4}, 10^3]$. For each iteration, we choose the best pair.

```
set.seed(58018)
# 10 runs with 10-fold CV. Selecting the best combination of gamma and C
# in each iteration.
gammas <- c()
costs <- c()
for (i in 1:10) {
  obj <- tune(svm, Demented~., data = st.long.train,ranges = list(gamma = 10^(-4:3), cost = 10^(-4:3)),
  GC <- summary(obj)[1]
  best_cost <- GC[[1]]$cost
  best_gamma <- GC[[1]]$gamma
  gammas <- c(gammas,best_gamma)
  costs <- c(costs,best_cost)
}
tab <- table(gammas,costs)
print(tab)
```

```
##        costs
## gammas  10 1000
##   0.001  0    2
##   0.01   1    0
##   0.1    7    0
```

From 10 runs, a value of gamma = 0.1 and cost = 10 produced the best results 7 times.

With the paramteres tuned for both kernels, we test their performance against one another.

```
set.seed(58008)
# Linear kernel, cost = 1000
lin.svm <- svm(Demented~.,data=st.long.train,kernel="linear",cost=1e+3,cross=10)
lin_accuracy <- lin.svm[29]
print(lin_accuracy)
```

```
## $tot.accuracy
## [1] 83.16832
```

```
# Radial kernel, gamma = 0.1, cost = 10
radial.svm <- svm(Demented~.,data=st.long.train,kernel="radial",gamma=1e-01,cost=1e+1,cross=10)
rad_accuracy <- radial.svm[29]
print(rad_accuracy)
```

```
## $tot.accuracy
## [1] 85.14851
```

Both models have an accuracy of approx. 80%, with the radial kernel outperforming the linear. The accuracy of the SVM with linear kernel has decreased by approx 2%. The SVM with radial kernel has approved in accuracy by approx 5% through tuning.

Cross-validation for final model selection

Now we employ 5-fold CV to get an estimate for the test error using our selected three "best" instances of the different classifiers.

```
set.seed(2222)

#randomize data
longitudinal.train.rand <- longitudinal.train[sample(nrow(longitudinal.train)),]

#make 5 folds
n.fold = 5
folds <- cut(seq(1,nrow(longitudinal.train.rand)),breaks=n.fold,labels=FALSE)

cv_err.df <- data.frame(fold = 1:n.fold, logistic_err = rep(0,n.fold),
                        rf_err = rep(0,n.fold), svm_err = rep(1,n.fold))

for(i in 1:n.fold){
    #get test index
    test.index <- which(folds==i,arr.ind=TRUE)
    #separate test data and train data
    test.data <- longitudinal.train.rand[test.index, ]
    train.data <- longitudinal.train.rand[-test.index, ]

    N <- length(test.data$Demented)
    rf <- randomForest(Demented ~. , data = train.data, mtry = m, ntree = B) # fit random forest
    cv_err.df$rf_err[i] <- sum(predict(rf, newdata = test.data) != test.data$Demented) / N

    logistic <- glm(Demented ~ Age + MMSE + nWBV + dMMSE, data = train.data, family = binomial) # fit l
    cv_err.df$logistic_err[i] <- sum((predict(logistic, newdata = test.data) > .5) != test.data$Demente

    svm.model <- svm(Demented~.,data= train.data ,kernel="radial",gamma=1e-01,cost=1e+01,cross=10) # fi
    cv_err.df$svm_err[i] <- sum(predict(svm.model, newdata = test.data) != test.data$Demented) / N
}

cv_err <- cv_err.df %>%
  select(-c("fold")) %>%
  colMeans()

cv_err
```

```
## logistic_err       rf_err      svm_err
##    0.1290476    0.1380952    0.1885714
```

We see that the logistic regression model scores the lowest CV-error of our three models. However random forest scores very similarly. Additionally through pretesting the code we noticed that the result (i.e. which classifier scores lowest) from the 5-fold CV varies depends on the splits made. This may be an indication of instability in our models as a result of the small amount of observations. Due to time constraints in finalizing this project we do not discuss this in detail, though it is an important remark.

Disregarding the issue of CV uncertainty, the logistic model and random forests seem to outperform SVM. For the logistic model this may be due to the feature selection done prior to fitting the model. Such a feature selection will increase the bias of the model but may decrease the variance if some of the predictors are introducing noise, ultimately leading to a better test-set performance. This seems to be the case for our logistic model. On the other hand, the accuracy of random forests may be due to the increased predictive accuracy achieved with bootstrap aggregation and decorrelation of the grown trees.

Though we cannot confidently conclude that the logistic model outperforms random forests and SVM, we performed model assessment on the held out test set using logistic regression, keeping in mind that our model selection is uncertain. This assessment emulates real-world efficacy as all test observations were excluded from the model selection process.

Model assessment

We now perform model assessment by classifying the observations from the test set using the logistic regression model and comparing the results to the true classifications. We report the misclassification rate, as well as the sensitivity and specificity.

```
pred.probs <- predict(longitudinal.logistic, newdata = longitudinal.test)
predictions <- pred.probs > 0.5 # use 0.5 cutoff

# generate confusion table for model on test data
confusion <- table(predictions, longitudinal.test$Demented)
rownames(confusion) <- c('pred 0','pred 1')
colnames(confusion) <- c('true 0','true 1')
# sensitivity and specificity
sens <- confusion[2,2] / sum(confusion[,2])
spec <- confusion[1,1] / sum(confusion[,1])
# misclassification error
misc <- (confusion[2,1] + confusion[1,2]) /
  sum(confusion)
#accuracy vector
sens_spec_misc <- data.frame(Sensitivity = sens, Specificity = spec, Misclass_Rate = misc)

confusion
```

```
##
## predictions true 0 true 1
##      pred 0     16      4
##      pred 1      1     13
```

```
sens_spec_misc
```

```
##   Sensitivity Specificity Misclass_Rate
## 1   0.7647059   0.9411765     0.1470588
```

The results from the test data show a good predictive accuracy with a misclassification rate of 15%. There were 4 false negatives and 1 false positive, giving a sensitivity of 76% and specificity of 94%. This means we're confident about our positive diagnoses and less confident about our negative diagnoses. For the purpose of predicting AD, being uncertain about a negative diagnosis is problematic, as this makes the model unsuitable for use as a pre-screening test. On the other hand, the high specificity counts in favor of the models usefulness.

Caveats

MMSE as a Feature We chose to include MMSE, which could be contentious. While it is objective in its evaluation, it is typically done by a medical professional. Despite this, we determined that since it is objective and simple, it is reasonable to include it. Some examples of assessment questions that it includes are “what floor are we on?”, “what year is it?”, and “spell the word ‘world’ backwards.” Since these questions are simple and objective enough to be run by a computerized test, it seems reasonable to use them in our project, withstanding the fact that our project aims to reduce the role of human decision making in the medical field (or at least make it easier).

Real-World Use Case Our model selection did not include a loss function that was complicated enough to select for a particular sensitivity or specificity, but this would be desirable for our project. Ideally, our model would be used to screen out patients who definitely do not have Alzheimer’s, which would result in a reduced role for medical professionals. This means that we would prioritize a high sensitivity as opposed to a high specificity. In a future version of this project, we would search for a loss function that punished false negatives more than other errors. In this sense, we are aiming to make our test similar to the method of how HIV testing is run:

> A blood sample from the patient is taken. First this is run through a cheap test (ELISA) that has a tremendously high sensitivity, but a low specificity. If the test is negative, no further testing is necessary and the patient is informed that they do not have HIV. If the test is positive, an expensive but accurate second “gold standard” test (western blot) is performed that has both a high sensitivity and specificity. The second test is done before informing the patient of the results of the first test. After this step, the results of the second test are shared with the patient.

We did not achieve a model that would work for this logic of testing since our sensitivity was too low. However, supposing a future version of this model could achieve a sufficiently high sensitivity, a procedure for diagnosing Alzheimer’s could follow the same logic with a statistical model replacing ELISA, and professional medical diagnosis replacing the western blot test. Thus the logic for a medical diagnosis procedure would go as follows:

> An MRI and MMSE of the patient are taken. First this is run through a statistical model with very high sensitivity to the target. If the test is negative, no further testing is necessary and the patient is informed that they do not have dementia. If the test is positive, a medical professional will act as the second and final “gold standard” test, diagnosing the patient based on the the data and a followup visit if deemed necessary by the medical professional. The second test is done before informing the patient of the results of the first test. After this step, the results of the second test are shared with the patient, and treatment can occur as necessary.

Summary

In this report, we analyzed longitudinal data from the OASIS project to investigate the accurate classification of patients as demented or not based on specific features. The process involved data pre-processing, including the exclusion of redundant features and the introduction of new ones to account for longitudinal data. We employed three models, namely logistic regression, random forests, and support vector machines, to predict a patient’s demented status using a training set. Feature selection with logistic regression identified Age, MMSE, nWBV, and dMMSE as the best subset with AIC and performed best with CV. For random forests, 5000 trees with 3 random predictors had the lowest OOB error, while SVM with a radial kernel, cost-function of 100, and gamma = 0.01 had the best CV. We matched the three models against one another, where logistic regression and random forests outperformed SVM, with logistic regression being the best of the three based on a 5-fold CV. In the end, we tested the logistic regression model on our test data which yielded good prediction accuracy of dementia. The final model scored higher on specificity than sensitivity. This indicates that te model is better suited for screening patients who are demented, as diagnosing patients with dementia comes with a much larger uncertainty.