ADNI Neuroimaging Dataset Alzheimer's Prediction Part 1

This dataset originates from the ADNI is the Alzheimers Disease Neuroimaging Initative. The goal in this notebook is to develop a model to predict Alzheimer's disease (and the probability) based on 579 lipidomics (from mass spectrometry) and 116 MRI neuroimaging features.

CN = Cognitively Normal

LMCI = Late Mild Cognitive Impairment

AD = Alzheimer's Disease

It is a cross sectional dataset consists of 760 total patients (213 CN, 375 LMCI, and 172 AD). For now, the LMCI patients will be removed, and models will be developed to predict the probability of Alzheimer's disease at baseline. Later on, in Part 2, the best model (as assessed by cross validation in this notebook) will be used to predict P(AD) for the LMCI patients.

The patients in this lipidomic/MRI baseline cross-sectional study were also followed for 8 years (in a separate longitudinal dataset), and we will determine whether a greater P(AD) at baseline in the LMCI patients is correlated with a worse cognitive outcome/trajectory over 8 years.

```
PyObject <module 'sklearn.preprocessing' from '/Users/rokapre/opt/anaconda3/envs/pynomkl/l
```

```
begin

using CSV, DataFrames, DataFramesMeta
using CategoricalArrays
using Statistics, StatsBase
using PyCall
using MLJ
using MLJLinearModels
using MLJDecisionTreeInterface

skl = pyimport("sklearn")
pp = pyimport("sklearn.preprocessing")

end
```

RID	ACYLCARNITINE_12_0_	ACYLCARNITINE_13_0_	ACYLCARNITINE_14_0_	ACYLCA

	RID	ACYLCARNITINE_12_0_	ACYLCARNITINE_13_0_	ACYLCARNITINE_14_0_	ACYLCA
1	3	2.75143	0.861787	1.39619	2.06113
2	5	2.78549	0.687368	1.11531	2.44162
3	6	7.26652	1.19243	2.73534	8.69197
4	8	2.90049	2.04487	1.67404	2.3017
5	10	11.7055	1.51607	3.32561	16.5753
6	14	4.47849	2.75814	2.2299	5.11648
7	15	3.85705	16.0293	1.58329	4.44374

```
rawdata = CSV.read("ADNI_cross_sect.csv",DataFrame); dropmissing!(rawdata)
```

Preprocessing

The first thing we do is split off the lipid features from the MRI features. The MRI features need to be normalized by the Estimated Total Intracranial Volume (ETIV) column. Following this we split off the LMCI patients.

For the classifier models in MLJ, the labels (DX_bl) must be converted to ordered categorical factors (with CN as the base level).

```
CategoricalArrays.CategoricalVector{String, UInt32, String, CategoricalArrays.Categorical
```

```
begin

#Preprocess MRI features beforehand by dividing by ETIV (Est. Total Intracranial
volume)

lipid_data = rawdata[:,2:580]
MRI_data_pre = rawdata[:,581:696]

names_lipid = names(lipid_data)
names_MRI = names(MRI_data_pre)

ETIV = rawdata.ETIV
DX_bl = rawdata[:,[:DX_bl]]

MRI_data = MRI_data_pre ./ ETIV

newdata = hcat(lipid_data,MRI_data,DX_bl)

#Split off non MCI (CN and AD) from MCI for now

MCIdata = @linq newdata |> where(:DX_bl .== "LMCI")
NoMCIdata = @linq newdata |> where(:DX_bl .!= "LMCI")


```

```
MCIdata.DX_bl = categorical(MCIdata.DX_bl)
NoMCIdata.DX_bl = categorical(NoMCIdata.DX_bl,levels=["CN","AD"],ordered=true)


end
```

For the cross validation loop, we need to have a train-test split function. Additionally, we will be testing 5 models: 4 L2-regularized Logistic Regressions with no features log transformed, lipids log transformed, MRI volumes log transformed, and both log transformed as well as a Random Forest model (on untransformed features). Additionally, a 6th model which is the average of the 4 logistic model predictions will be tested as well

The data (after transformation, if any) will be robustly scaled (by median and IQR) to keep features on the same units. Robust scaling is used so that potential outliers in the 579+116 lipid+MRI features do not affect the scaling as much. We use scikit-learn from Python through Julia to perform the scaling. The sklearn RobustScaler fitted object is stored in a Julia struct that contains a bool indicating whether or not log transform was applied to lipids or MRI volumes.

```
begin

function train_test_split(data::DataFrame,response::Symbol;prop_train=0.8)
    ntot = nrow(data)
    n_train = Int(floor(prop_train*ntot))
    train_idx = sample(1:ntot,n_train,replace=false)

    train_X = data[train_idx,Not(response)]
    train_Y = data[train_idx,response]

    test_X = data[Not(train_idx),Not(response)]
    test_Y = data[Not(train_idx),response]

    return train_X,train_Y,test_X,test_Y
end

struct PreprocessParams
    lipidnames :: Vector{String}
    MRInames :: Vector{String}
    log_lipid::Bool
    log_MRI::Bool
    RS:: PyObject
end

"""Preprocess(data;log_lipid,log_MRI)

This function does further preprocessing including log transforming the
lipid or MRI features separately. And then using RobustScaler() to scale
the data for use in ML modeling.

Note: MRI features should be normalized by ETIV beforehand

"""
function preprocess(data,lipidnames,MRInames;log_lipid=false,log_MRI=false)
    lipids = Matrix(data[:,lipidnames])
    MRI = Matrix(data[:,MRInames])
    if log_lipid == true
        lipids = log.(lipids)
    end
    if log_MRI == true
```

```
            MRI = log.(MRI)
        end
        X = hcat(lipids,MRI)
        RS = pp.RobustScaler()
        RS.fit(X)
        return PreprocessParams(lipidnames,MRInames,log_lipid,log_MRI,RS)
    end


    function MLJ.transform(preprocessor::PreprocessParams,data::DataFrame)
        lipids = Matrix(data[:,preprocessor.lipidnames])
        MRI = Matrix(data[:,preprocessor.MRInames])
        if preprocessor.log_lipid==true
            lipids = log.(lipids)
        end
        if preprocessor.log_MRI==true
            MRI = log.(MRI)
        end
        X = hcat(lipids,MRI)
        X_scaled = preprocessor.RS.transform(X)
        return(X_scaled)
    end

    end
```

Evaluation Metrics

In addition to using the typical classification evaluation metrics (Accuracy, Sensitivity, Specificity) we would also like to assess the probability predictions of CN/AD in a continuous sense. To do so, we can use the **cross entropy loss**. In order to more easily interpret this loss, we normalize it by the cross entropy (logistic) loss corresponding to probability predictions that are simply the mean values of CN/AD in the test set. This is known as a test set **Pseudo-R2**.

$$R^2_{ps} = 1 - \frac{CE(model)}{CE(mean)}$$

```
PsR2 (generic function with 1 method)
```

```
    begin

    function PsR2(phat,Ytrue)
        @assert(classes.(phat) == classes.(Ytrue))
        nclasses = length(classes(Ytrue))
        pnull = UnivariateFinite(classes(Ytrue),repeat([1/nclasses],nclasses))
        pnull = repeat([pnull],length(Ytrue))
        nulldev = mean(cross_entropy(pnull,Ytrue))
        modeldev = mean(cross_entropy(phat,Ytrue))
        PsR2 = 1 - modeldev/nulldev
        return(PsR2)
    end

    end
```

Model Averaging

As mentioned earlier, we will be using 4 different logistic regression models corresponding to the possible combos of transforming none/either/both of the lipid and MRI features. Below is the function used to combine the results of these models. Note that we will be using weights of 1/4 for simplicity. Ideally, the model weights for model averaging itself will also be computed through cross validation, but in this case the data is very limited and a cross validation loop is already being used for model selection (and within that, the Logistic Regression L2 regularizer λ will also be selected by CV).

```
PredAverage (generic function with 1 method)
```

```
begin

function PredAverage(preds,wts)
    @assert(length(unique(classes.(preds))) == 1)
    cats = classes(preds[1])
    probs = [pdf(p,cats) for p in preds]
    wt_probs = wts .* probs
    avg_prob = sum(wt_probs)
    avg_prob = avg_prob ./ sum(avg_prob,dims=2) #normalize in case
    return(UnivariateFinite(classes(preds[1]),avg_prob))
end

end
```

Cross-Validation for Model Assessment

Below is the cross-validation setup for this analysis. Note that Logistic Regression hyperameter λ for L2 regularization and the min*purity*increase for a Random Forest (200 trees) will itself be selected in an inner cross validation setup. The λ will range from 0.001 to 1000 with 10 points spaced on the log scale, and the min*purity*increase will range from 1e-8 to 1 with 10 points also spaced on the log scale.

Every iteration the accuracy, sensitivity, specificity, and pseudo R2 will be computed for each of the 6 models. Also note that scaling is performed on the training only and this is used to scale the test set in each iteration, so as to avoid data leakage.

```
RandomForestClassifier
```

```
begin

@load LogisticClassifier pkg=MLJLinearModels
@load RandomForestClassifier pkg = DecisionTree

end
```

```
begin

@time begin

runs = 20
AccResults=DataFrame(BGLM_nolog=fill(-9.9,runs),
                     BGLM_liplog = fill(-9.9,runs),
                    BGLM_MRIlog = fill(-9.9,runs),
                    BGLM_bothlog = fill(-9.9,runs),
                    BGLM_avg = fill(-9.9,runs),
                    RF = fill(-9.9,runs))
SensResults = deepcopy(AccResults)
SpecResults = deepcopy(AccResults)
PsR2Results = deepcopy(AccResults)

for i=1:runs

    Xtrain,Ytrain,Xtest,Ytest = train_test_split(NoMCIdata,:DX_bl)

    prep_nolog = preprocess(Xtrain,names_lipid,names_MRI)
    Xtrain_nolog_sc = MLJ.transform(prep_nolog,Xtrain)
    Xtest_nolog_sc = coerce(MLJ.transform(prep_nolog,Xtest),Continuous)

    prep_liplog = preprocess(Xtrain,names_lipid,names_MRI,log_lipid=true)
    Xtrain_liplog_sc = MLJ.transform(prep_liplog,Xtrain)
    Xtest_liplog_sc = MLJ.transform(prep_liplog,Xtest)

    prep_MRIlog = preprocess(Xtrain,names_lipid,names_MRI,log_MRI=true)
    Xtrain_MRIlog_sc = MLJ.transform(prep_MRIlog,Xtrain)
    Xtest_MRIlog_sc = MLJ.transform(prep_liplog,Xtest)

    prep_bothlog =
preprocess(Xtrain,names_lipid,names_MRI,log_lipid=true,log_MRI=true)
    Xtrain_bothlog_sc = MLJ.transform(prep_bothlog,Xtrain)
    Xtest_bothlog_sc = MLJ.transform(prep_bothlog,Xtest)

    BGLM = MLJLinearModels.LogisticClassifier(penalty=:l2)
    λ_range = range(BGLM,:lambda,lower=0.001,
                    upper=1000,scale=:log);

    TunedBGLM = TunedModel(model=BGLM,resampling=CV(nfolds=5),
                            tuning=Grid(resolution=10),range=λ_range,
                            measure=cross_entropy);

    RF = MLJDecisionTreeInterface.RandomForestClassifier(n_trees=200)
    cp_range = range(RF,:min_purity_increase,lower=1e-8,upper=1,scale=:log)

    TunedRF = TunedModel(model=RF,resampling=CV(nfolds=5),
                        tuning=Grid(resolution=10),range=cp_range,
                        measure=cross_entropy)


    TunedBGLM_nolog = machine(TunedBGLM,Xtrain_nolog_sc,Ytrain)

    fit!(TunedBGLM_nolog,verbosity=0)
    Ptest_BGLM_nolog = MLJ.predict(TunedBGLM_nolog,Xtest_nolog_sc)
    Ytest_BGLM_nolog = MLJ.predict_mode(TunedBGLM_nolog,Xtest_nolog_sc)

    AccResults.BGLM_nolog[i] = accuracy(Ytest_BGLM_nolog,Ytest)
    SpecResults.BGLM_nolog[i] = specificity(Ytest_BGLM_nolog,Ytest)
    SensResults.BGLM_nolog[i] = sensitivity(Ytest_BGLM_nolog,Ytest)
    PsR2Results.BGLM_nolog[i] = PsR2(Ptest_BGLM_nolog,Ytest)


    TunedBGLM_liplog = machine(TunedBGLM,Xtrain_liplog_sc,Ytrain)
    fit!(TunedBGLM_liplog,verbosity=0)
    Ptest_BGLM_liplog = MLJ.predict(TunedBGLM_liplog,Xtest_liplog_sc)
    Ytest_BGLM_liplog = MLJ.predict_mode(TunedBGLM_liplog,Xtest_liplog_sc)

    AccResults.BGLM_liplog[i] = accuracy(Ytest_BGLM_liplog,Ytest)
    SpecResults.BGLM_liplog[i] = specificity(Ytest_BGLM_liplog,Ytest)
    SensResults.BGLM_liplog[i] = sensitivity(Ytest_BGLM_liplog,Ytest)
    PsR2Results.BGLM_liplog[i] = PsR2(Ptest_BGLM_liplog,Ytest)


    TunedBGLM_MRIlog = machine(TunedBGLM,Xtrain_MRIlog_sc,Ytrain)
    fit!(TunedBGLM_MRIlog,verbosity=0)
    Ptest_BGLM_MRIlog = MLJ.predict(TunedBGLM_MRIlog,Xtest_MRIlog_sc)
    Ytest_BGLM_MRIlog = MLJ.predict_mode(TunedBGLM_MRIlog,Xtest_MRIlog_sc)

    AccResults.BGLM_MRIlog[i] = accuracy(Ytest_BGLM_MRIlog,Ytest)
    SpecResults.BGLM_MRIlog[i] = specificity(Ytest_BGLM_MRIlog,Ytest)
    SensResults.BGLM_MRIlog[i] = sensitivity(Ytest_BGLM_MRIlog,Ytest)
    PsR2Results.BGLM_MRIlog[i] = PsR2(Ptest_BGLM_MRIlog,Ytest)

    TunedBGLM_bothlog = machine(TunedBGLM,Xtrain_bothlog_sc,Ytrain)
    fit!(TunedBGLM_bothlog,verbosity=0)
    Ptest_BGLM_bothlog = MLJ.predict(TunedBGLM_bothlog,Xtest_bothlog_sc)
    Ytest_BGLM_bothlog = MLJ.predict_mode(TunedBGLM_bothlog,Xtest_bothlog_sc)

    AccResults.BGLM_bothlog[i] = accuracy(Ytest_BGLM_bothlog,Ytest)
    SpecResults.BGLM_bothlog[i] = specificity(Ytest_BGLM_bothlog,Ytest)
    SensResults.BGLM_bothlog[i] = sensitivity(Ytest_BGLM_bothlog,Ytest)
    PsR2Results.BGLM_bothlog[i] = PsR2(Ptest_BGLM_bothlog,Ytest)

    Ptest_BGLMavg = PredAverage([Ptest_BGLM_nolog,Ptest_BGLM_liplog,Ptest_BGLM_MRIlog,
                        Ptest_BGLM_bothlog],fill(0.25,4))
    Ytest_BGLMavg = mode.(Ptest_BGLMavg)

    AccResults.BGLM_avg[i] = accuracy(Ytest_BGLMavg,Ytest)
    SpecResults.BGLM_avg[i] = specificity(Ytest_BGLMavg,Ytest)
    SensResults.BGLM_avg[i] = sensitivity(Ytest_BGLMavg,Ytest)
    PsR2Results.BGLM_avg[i] = PsR2(Ptest_BGLMavg,Ytest)


    TunedRF_nolog = machine(TunedRF,Xtrain_nolog_sc,Ytrain)
    fit!(TunedRF_nolog,verbosity=0)
    Ptest_RF = MLJ.predict(TunedRF_nolog,Xtest_nolog_sc)
    Ytest_RF = MLJ.predict_mode(TunedRF_nolog,Xtest_nolog_sc)

    AccResults.RF[i] = accuracy(Ytest_RF,Ytest)
    SpecResults.RF[i] = specificity(Ytest_RF,Ytest)
    SensResults.RF[i] = sensitivity(Ytest_RF,Ytest)
    PsR2Results.RF[i] = PsR2(Ptest_RF,Ytest)

end

end

end
```

Aggregating Results

```
aggregate_results (generic function with 1 method)
```

```
begin

function aggregate_results(results::DataFrame)
    return(combine(results,nrow,
    :BGLM_nolog=>mean=>:meanBGLM_nolog,
    :BGLM_nolog=>std=>:sdBGLM_nolog,
    :BGLM_liplog=>mean=>:meanBGLM_liplog,
    :BGLM_liplog=>std=>:sdBGLM_liplog,
    :BGLM_MRIlog=>mean=>:meanBGLM_MRIlog,
    :BGLM_MRIlog=>std=>:sdBGLM_MRIlog,
    :BGLM_bothlog=>mean=>:meanBGLM_bothlog,
    :BGLM_bothlog=>std=>:sdBGLM_bothlog,
    :BGLM_avg=>mean => :meanBGLM_avg,
    :BGLM_avg=>std=> :sdBGLM_avg,
    :RF => mean => :meanRF,
    :RF => std => :sdRF))
end

end
```

Accuracy

All of the logistic regression models perform very close, though the models with both lipid/MRI features log transformed perform the best ($81.2\% \pm 5.5\%$) while the model with only lipids log transformed is 2nd best ($80.2\% \pm 5.6\%$). The averaged logistic regression performs has the 3rd best accuracy, but the lowest SD ($80.1\% \pm 4.9$)

Random Forest performs the worst ($74.4\% \pm 5.4\%$), and is likely too complex of a model for the small sample size of this dataset with LMCI excluded.

Given that lipidomic and MRI biomarkers are positively skewed, it is not surprising that log transforming appears to help. A distribution of the features closer to multivariate normal may be contributing to improving linear separability of the classes

MeanAccRes =

	nrow	meanBGLM_nolog	sdBGLM_nolog	meanBGLM_liplog	sdBGLM_liplog	meanBGLM
1	20	0.779221	0.0541239	0.807792	0.0561525	0.777922

```
MeanAccRes = aggregate_results(AccResults)
```

Sensitivity

This follows a similar trend to accuracy. The best sensitivity comes from the model with both lipids/MRI features log transformed at $76.8\% \pm 9.1\%$ followed by the model with only lipids log transformed at $76.4\% \pm 8.9\%$. Averaging logistic models again appears to worsen the mean results but improve the SD with sensitivity at $73.8 \pm 7.7\%$.

RF has the worst sensitivity at $62.0\% \pm 9.4\%$

MeanSensRes =

	nrow	meanBGLM_nolog	sdBGLM_nolog	meanBGLM_liplog	sdBGLM_liplog	meanBGLM
1	20	0.718882	0.0913763	0.763577	0.0888083	0.682256

```
MeanSensRes = aggregate_results(SensResults)
```

Specificity

In the case of specificity, the results are similar across the logistic regression models but the MRI log transformed model has the highest specificity at $86.2\% \pm 5.9\%$ while the averaged model has the 2nd highest specificity at $85.6\% \pm 5.1\%$. The model with both lipid/MRI log transformed had a specificity of $85.6\% \pm 4.9\%$, lipid only log transformed had a specificity of $84.9\% \pm 5.3\%$ and the model with none transformed had a specificity of $83.3\% \pm 6.0\%$

RF actually has a similar specificity as well at $85.4\% \pm 5.9\%$

MeanSpecRes =

	nrow	meanBGLM_nolog	sdBGLM_nolog	meanBGLM_liplog	sdBGLM_liplog	meanBGLM
1	20	0.832705	0.0598668	0.848673	0.0527347	0.862091

```
MeanSpecRes = aggregate_results(SpecResults)
```

Pseudo R^2

While the classification results are decent, the Pseudo R^2 based on binary cross entropy loss is not so stellar. This indicates that obtaining exact probabilities conditional on the MRI and lipidomic features is difficult, and the data is perhaps too noisy.

We find that the logistic model with both lipid/MRI features transformed provides the highest value at 0.371 ± 0.125 with the model that has lipid only coming a very close 2nd at 0.370 ± 0.124

Thus, instead of using the exact probability predictions on the LMCI patients in the next section (after the best model is fitted to the full data), we will bin the probability predictions into 5 categories (0-20, 20-40, etc). This accounts for a greater degree of noise in the probability predictions and is justified by the fact that the classification into 2 categories performed decently well. Thus we anticipate (or rather, make an educated assumption) that having 5 catefories/bins of LMCI based on the probabilities will also be representative. Then we can examine the correlation between the binned predicted probability and longitudinal outcome of the LMCI patients.

```
MeanPsR2 =
```

	nrow	meanBGLM_nolog	sdBGLM_nolog	meanBGLM_liplog	sdBGLM_liplog	meanBGLM
1	20	0.282378	0.128392	0.369953	0.124022	0.313151

```
MeanPsR2 = aggregate_results(PsR2Results)
```

Overall Assessment and Model Refitting

From the above, it appears that while the logistic models performed very close, log transforming both the lipidomic and MRI volumetric features is the way to go, considering it scores highest in each of the metrics (except specificity, though it comes close). We will now refit this model to the full dataset and examine the coefficient values

```
[34mMachine{ProbabilisticTunedModel{Grid,…},…} @389 [39m trained 1 time; caches data
  args:
    1:   [34mSource @859 [39m ⏎ `AbstractMatrix{ScientificTypes.Continuous}`
    2:   [34mSource @161 [39m ⏎ `AbstractVector{ScientificTypes.OrderedFactor{2}}`
```

```
begin

YNoMCI = NoMCIdata.DX_bl

prepNoMCI_bothlog =
preprocess(NoMCIdata,names_lipid,names_MRI,log_lipid=true,log_MRI=true)

XNoMCI_bothlog_sc = MLJ.transform(prepNoMCI_bothlog,NoMCIdata)

BGLM = MLJLinearModels.LogisticClassifier(penalty=:l2)
λ_range = range(BGLM,:lambda,lower=0.001,
                upper=1000,scale=:log)

TunedBGLM = TunedModel(model=BGLM,resampling=CV(nfolds=5),
                       tuning=Grid(resolution=10),range=λ_range,
                       measure=cross_entropy);

TunedBGLM_BothLogFinal = machine(TunedBGLM,XNoMCI_bothlog_sc,YNoMCI)
fit!(TunedBGLM_BothLogFinal,verbosity=0)

end
```

```
bestmod = LogisticClassifier(
              lambda = 9.999999999999998,
              gamma = 0.0,
              penalty = :l2,
              fit_intercept = true,
              penalize_intercept = false,
              solver = nothing) [34m @587 [39m
```

```
bestmod=fitted_params(TunedBGLM_BothLogFinal).best_model
```

Interpretation

```
coefs =
 Float64[0.015503,  -0.0357977,  0.0113196,  0.110941,  0.127988,  -0.0999003,  -0.0568761,
```

```
coefs = fitted_params(TunedBGLM_BothLogFinal).best_fitted_params.coefs
```

	feature_type	feature_name	coefvalue	abscoefvalue
1	"MRI"	"HIPPL"	-0.814549	0.814549
2	"MRI"	"HIPPR"	-0.625209	0.625209
3	"MRI"	"TEMPMIDL"	-0.472815	0.472815
4	"MRI"	"TEMPINFL"	-0.402697	0.402697
5	"Lipid"	"GM3_D18_1_20_0_"	0.390729	0.390729
6	"Lipid"	"DE_18_1_"	0.384181	0.384181
7	"MRI"	"PARAHIPPL"	-0.372029	0.372029
8	"MRI"	"CINGMIDL"	0.356974	0.356974
9	"Lipid"	"TG_O_52_2_NL_16_0_"	-0.329902	0.329902
10	"MRI"	"PARAHIPPR"	-0.30234	0.30234

```
begin

p_lipid = length(names_lipid)
p_MRI = length(names_MRI)

coef_data = DataFrame(feature_type=vcat(fill("Lipid",p_lipid),fill("MRI",p_MRI)),
    feature_name = vcat(names_lipid,names_MRI),
    coefvalue = coefs,
    abscoefvalue = abs.(coefs))

sort!(coef_data,:abscoefvalue,rev=true)
first(coef_data,10)

end
```

According to the above, the best $\lambda = 10$ found by cross validation of fitting the logistic regression on log transformed (and scaled) lipidomic/MRI features. Additonally, in the coefficients we see that the ones corresponding to the left/right hippocampus (HIPPL/HIPPR) volume from MRI are most predictive of Alzheimer's disease. This is not surprising considering that the hippocampus is the area of the brain most involved in memory. The next most important features are those corresponding to volumes of the medial temporal lobe (TEMPMIDL/TEMPMIDR), which is also involved in cognitive/emotional processing. An important lipidomic feature which shows up in the top 10 is related to GM3, a ganglioside. Gangliosides have been shown to enhance the formation of $A\beta$ plaque: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3025365/.

Despite there being fewer MRI features in the dataset, the top 10 most important features seem to correspond to MRI rather than lipidomic. This may be due to the fact that MRI measurements are measurements directly of the brain, whereas lipidomic measurements may only indirectly correlate once they have already impacted these brain regions. More studies would be needed to determine the mechanisms involved.

Obtain Predictions

We now obtain the predictions on the LMCI patients using the fitted logistic model on the log scale

```
XMCI_bothlog_sc =
375×695 Matrix{Float64}:
  0.025849   -0.428074    0.246245   …   0.930599    0.793746     0.405814
 -0.349362   -0.378741    0.0692862     -2.5366     -2.96478     -0.109929
 -0.172676    1.79379    -0.422          0.0760295   3.46238e-5  -0.260041
  0.81113     0.167741    0.532641      -0.360423    0.133748     1.04856
  0.561025    0.258841    0.585372      -0.992381   -0.680317     0.508574
 -0.724301   -1.06685    -0.529042   …  -1.37464    -0.0427204    0.735425
 -1.00076    -0.531046   -0.989011       0.300239    0.89791      0.87306
  ⋮                                  ⋱
 -0.0345166  -0.674284   -0.200868      -0.121521   -0.550786    -0.538838
 -0.0377841   0.0224368  -0.194798   …  -0.0130335  -0.633432    -1.05486
 -0.0805152  -0.193277    0.577118       0.176329   -0.1234       0.0778093
 -0.101169   -0.775568   -0.479362       0.621319    0.547162     0.441616
 -0.877494   -0.944106   -0.797604       0.473576    0.0240073   -0.587052
 -0.961659   -1.70823    -0.562148       1.0274      0.554907    -0.317845
```

```
XMCI_bothlog_sc = MLJ.transform(prepNoMCI_bothlog,MCIdata)
```

```
ProbAD_MCI =
  Float64[0.051884,  0.411303,  0.147606,  0.0673143,  0.429472,  0.672642,  0.530993,  0.640
```

```
ProbAD_MCI = pdf(MLJ.predict(TunedBGLM_BothLogFinal,XMCI_bothlog_sc),classes(YNoMCI))
[:,2]
```

```
  String["[0.0, 0.2)",  "[0.4, 0.6)",  "[0.6, 0.8)",  "[0.8, 1.0)",  "[0.2, 0.4)"]
```

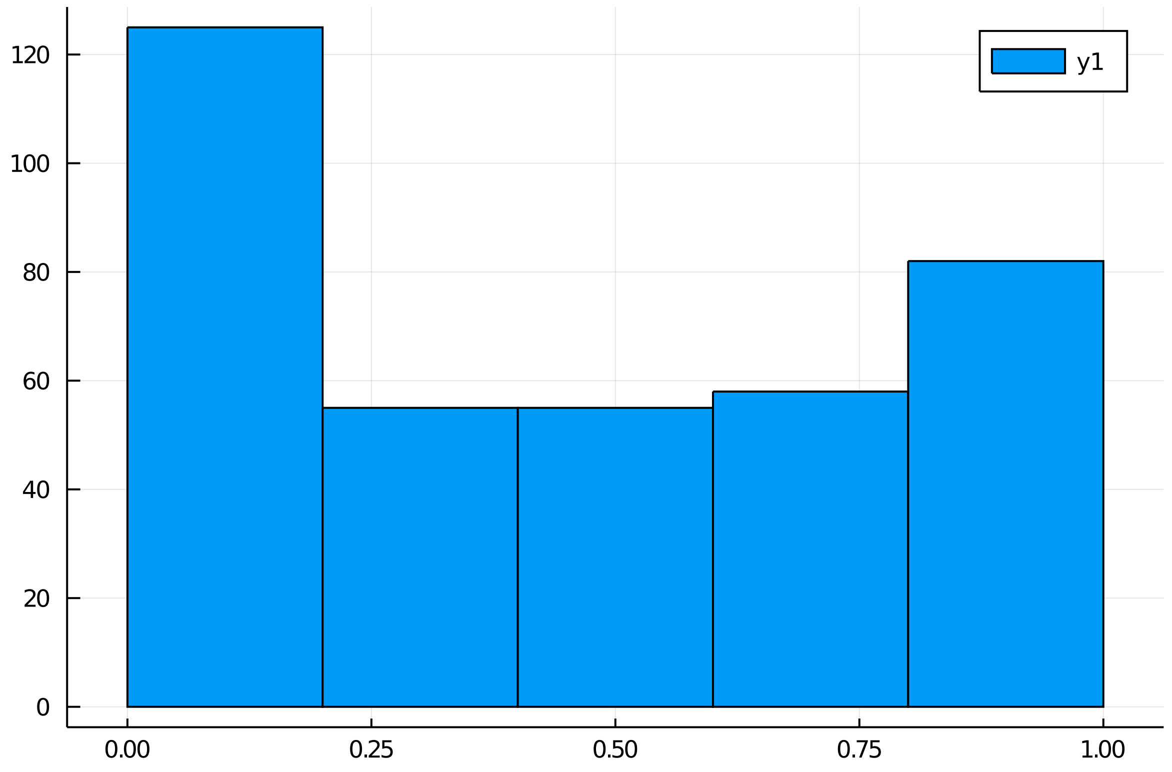
```
BinAD_MCI = cut(ProbAD_MCI,[0,0.2,0.4,0.6,0.8,1])
```

```
  CategoricalArrays.CategoricalVector{String, UInt32, String, CategoricalArrays.Categorical
```

```
begin
MCIdata.ProbAD = ProbAD_MCI
MCIdata.BinAD = BinAD_MCI;
end
```

```
using Plots
```

Below is a histogram of the probability predictions. Most MCI patients (125) are in the 0-0.2 bin.



```
histogram(MCIdata.ProbAD)
```

	BinAD	nrow
1	CategoricalValue{String, UInt32} "[0.0	125
2	CategoricalValue{String, UInt32} "[0.2	55
3	CategoricalValue{String, UInt32} "[0.4	55
4	CategoricalValue{String, UInt32} "[0.6	58
5	CategoricalValue{String, UInt32} "[0.8	82

```
combine(groupby(MCIdata,:BinAD),nrow)
```

Save Predictions on LMCI and Model

```
"MCIdata_withADprobs.csv"
```

```
CSV.write("MCIdata_withADprobs.csv",MCIdata)
```

```
MLJ.save("AD_TunedLogistic_BothLipidMRILog_Scaled.jlso",TunedBGLM_BothLogFinal)
```