Class Project 702

Tibui Ivan Ade and Jeffrey Amankwah

2023-03-08

We are using ALL data Goal - Use supervised machine learning to predict if a patient would be MDR based on the features in the ALL dataset - Determine which features account for MDR

Exploratory Data Analysis - Frequency of MDR - Analysis of MDR vs Other features (Age,sex, BT (Stages), remission, CR ) - Using these features create a model which can predict MDR. - Determing feature importance from teh model (Determine the risk factor associted with MDR)

Title : Using Supervised ML methods to predict MDR in ALL patients

Chapter Wise 1. Literature of ALL and the DATA (ALL) 2. Methods used 3. EDA - Distribution mdr in our Data ( Bar Chart) - Other phenotycal distribution (Mdr - Sex, mdr -Age,mdr-BT, mdr -transplant) - Genes 4. Modelling Anova, Genefilter, 5. Explanation of results

library(ALL)

## Loading required package: Biobase

## Loading required package: BiocGenerics

##   
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:stats':  
##   
## IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':  
##   
## anyDuplicated, aperm, append, as.data.frame, basename, cbind,  
## colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,  
## get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,  
## match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,  
## Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,  
## table, tapply, union, unique, unsplit, which.max, which.min

## Welcome to Bioconductor  
##   
## Vignettes contain introductory material; view with  
## 'browseVignettes()'. To cite Bioconductor, see  
## 'citation("Biobase")', and for packages 'citation("pkgname")'.

data(ALL)  
ALL

## ExpressionSet (storageMode: lockedEnvironment)  
## assayData: 12625 features, 128 samples   
## element names: exprs   
## protocolData: none  
## phenoData  
## sampleNames: 01005 01010 ... LAL4 (128 total)  
## varLabels: cod diagnosis ... date last seen (21 total)  
## varMetadata: labelDescription  
## featureData: none  
## experimentData: use 'experimentData(object)'  
## pubMedIds: 14684422 16243790   
## Annotation: hgu95av2

?ExpressionSet  
?ALL

Preposing Set - Set MDR (multi-drug resitance) as target \*\* Determine the distributrion of MDR (how pos and neg) \*\*

test - using ANOVA to select the best genes to predict with ## ANOVA

ALLmdr <- ALL[,ALL$mdr %in% c("NEG", "POS")]  
anova.pValue <- apply(exprs(ALLmdr), 1, function(x) anova(lm(x ~ ALLmdr$mdr))$Pr[1])  
names <- featureNames(ALL)[anova.pValue<0.001] # getting features - Using ANOVA for dimensionality reduction with a cutoff p value - 0.001 - We list this genes   
names

## [1] "33655\_f\_at" "36693\_at" "38430\_at" "38433\_at" "38828\_s\_at"  
## [6] "38907\_at" "41082\_at"

##Generating ANOVA DATA Set

ALLmdrnames <- ALLmdr[names, ]  
mdr\_data=as.data.frame(t(exprs(ALLmdrnames)))  
#mdr <- factor(ALLmdrnames$mdr)  
mdr\_data$mdr=ALLmdr$mdr  
mdr\_data

## 33655\_f\_at 36693\_at 38430\_at 38433\_at 38828\_s\_at 38907\_at 41082\_at mdr  
## 01005 3.620319 3.411613 3.097963 3.741312 7.970141 3.326365 4.130910 NEG  
## 01010 3.855915 3.684611 3.126004 3.802945 7.063812 3.402101 4.466888 POS  
## 03002 3.817345 3.206579 2.804531 3.751225 7.138603 3.138100 4.260769 NEG  
## 04006 3.799043 3.431849 3.258467 3.672545 7.432463 3.214402 4.190008 NEG  
## 04007 3.458871 3.020218 3.423281 3.401272 8.359594 2.983769 4.008179 NEG  
## 04008 3.780309 3.290995 2.967507 3.552835 7.546242 3.195774 4.018990 NEG  
## 04010 3.718474 3.256720 3.082838 3.517697 7.625587 3.153587 4.295994 POS  
## 04016 3.513268 3.215534 2.806101 3.747399 8.191605 2.967385 4.049841 NEG  
## 06002 3.834030 3.287146 2.893262 3.807259 8.266034 3.278762 4.211621 NEG  
## 08001 3.737383 3.333413 3.086902 3.842660 7.782654 3.248337 4.064822 NEG  
## 08011 3.578650 3.420585 3.187481 3.920922 7.918819 3.551011 4.087648 NEG  
## 08012 3.644503 3.230227 2.883819 3.423560 8.936500 2.932402 3.963747 NEG  
## 08018 3.793815 3.318976 2.980283 3.749804 8.388201 3.352595 4.253114 NEG  
## 08024 3.717714 3.611318 2.909742 3.963104 7.838427 3.230736 4.235956 POS  
## 09008 3.690663 3.118150 2.739041 3.574331 7.030079 3.082421 4.248389 NEG  
## 09017 3.861665 3.384327 2.920743 3.696046 7.029666 2.927837 4.263216 NEG  
## 11005 3.632832 3.450839 3.073491 4.153358 7.539500 3.452627 4.303951 NEG  
## 12006 3.876335 3.336441 2.833062 3.901536 7.437986 3.305658 4.253899 NEG  
## 12007 3.670048 3.354043 3.066289 3.972881 7.684615 3.292940 4.049164 NEG  
## 12012 3.749581 3.211202 3.050550 3.982717 7.523282 3.209548 4.227102 NEG  
## 12019 3.754241 3.439510 2.854015 3.865441 7.746460 3.356029 4.255196 POS  
## 14016 3.858016 3.565115 3.090231 4.214185 7.144022 3.406932 4.178603 NEG  
## 15001 3.821197 3.618298 2.970212 3.826603 8.259570 3.150540 4.323413 NEG  
## 15004 3.744012 3.969854 2.740389 4.055263 7.728388 3.217198 4.509927 NEG  
## 15005 3.701766 3.504099 3.221344 3.730657 7.835973 3.480227 4.149258 POS  
## 16004 4.003810 3.474962 3.166498 3.835629 7.128244 3.566091 4.567071 NEG  
## 16009 3.954700 3.715796 3.032165 4.044840 7.279273 3.156199 4.418199 POS  
## 19005 3.578149 3.414325 3.052829 3.837446 8.231125 3.205952 4.441229 NEG  
## 20002 3.700387 3.422002 3.073136 3.578566 7.966102 3.145674 4.339881 NEG  
## 22009 3.869230 3.499658 3.018244 3.906524 7.461119 3.283860 4.090568 NEG  
## 22010 3.830073 3.487092 3.041463 3.745064 7.479274 3.428400 3.991930 NEG  
## 22011 3.790334 3.256421 2.895134 3.656190 7.744947 3.269068 4.221802 NEG  
## 22013 3.777872 3.452300 3.100019 3.848310 7.315348 3.499732 4.130585 NEG  
## 24001 3.534716 3.463109 3.116356 3.815705 7.795513 3.036268 4.039612 NEG  
## 24005 3.605971 3.074917 2.809736 3.557238 8.414925 3.326162 4.076144 NEG  
## 24008 3.712528 3.568002 3.194373 3.897636 8.010986 3.116482 3.961353 NEG  
## 24010 3.955701 3.801642 3.093114 4.064683 7.542333 3.386747 4.189828 NEG  
## 24011 3.961744 3.365947 3.134960 4.379207 7.487291 3.235082 5.075849 POS  
## 24017 3.760633 3.350834 2.982844 3.812108 7.790117 3.363856 4.372903 NEG  
## 24018 3.966461 3.773146 3.308716 3.957366 7.458638 3.443705 4.395984 POS  
## 24019 3.781507 3.402599 2.966851 3.664039 7.667427 3.347041 4.155823 NEG  
## 24022 3.964015 3.383295 3.251873 4.522929 6.863005 3.251695 4.199564 POS  
## 25003 3.987258 3.421636 3.229524 4.030142 7.730503 3.495194 4.078337 POS  
## 25006 3.830159 3.458223 3.233894 3.994612 7.058110 3.223380 4.498843 NEG  
## 26001 3.953376 3.626458 3.058185 3.951758 7.884542 3.501843 4.260372 POS  
## 26003 3.706254 3.244741 2.762245 3.801473 7.487131 3.083737 4.106102 NEG  
## 26005 3.927358 3.522280 3.004128 3.803756 7.852623 3.203465 4.271786 NEG  
## 26008 3.930826 3.626018 3.203324 3.622939 7.404470 3.206483 4.311358 NEG  
## 27003 3.713701 3.800514 2.863978 4.330532 7.468833 3.506254 4.288046 POS  
## 27004 3.628088 3.188708 2.710728 3.798500 8.135568 3.188329 4.031315 NEG  
## 28001 4.372349 3.595362 3.365154 4.443635 6.888066 3.488382 4.623949 POS  
## 28003 3.965198 3.647583 3.426390 3.964769 7.793668 3.838518 4.403198 NEG  
## 28005 3.592397 3.208980 2.987175 3.724451 7.688876 3.351883 4.201683 POS  
## 28006 3.879697 3.606303 2.902672 3.965353 7.879246 3.311279 4.355825 NEG  
## 28007 3.630185 3.166387 2.853102 3.647508 8.112087 3.074481 4.093057 NEG  
## 28019 3.892079 3.376063 3.020953 3.944822 7.490927 3.188101 4.329099 NEG  
## 28021 3.698354 3.143721 2.873870 3.492341 7.422219 3.092320 4.202989 NEG  
## 28023 3.712792 3.247064 2.726071 3.634211 8.715577 2.922662 4.070501 NEG  
## 28024 3.691960 3.215505 3.024606 3.592064 7.444096 3.149752 4.025248 NEG  
## 28028 3.596951 3.246220 3.135766 3.842506 8.408105 3.089698 4.233279 NEG  
## 28031 3.744003 3.538473 2.855334 3.774511 8.134160 3.086430 4.280329 NEG  
## 28032 4.146309 3.120206 3.340110 3.814277 7.817723 3.338060 4.066394 NEG  
## 28035 3.719911 3.193610 3.187179 3.956230 7.884421 3.355883 4.176022 POS  
## 28036 3.849615 3.375257 3.140422 3.958001 7.673960 2.918108 4.471630 NEG  
## 28037 3.697182 3.398349 2.935754 3.734116 7.310471 3.409230 3.973286 POS  
## 28042 3.798207 3.305762 3.002293 3.774772 7.765525 3.237789 4.259985 NEG  
## 28043 3.782547 3.428449 3.070420 3.907899 7.865482 3.142111 4.200605 NEG  
## 28044 3.747582 3.421555 3.135776 3.716149 8.040247 3.283038 4.192893 NEG  
## 28047 3.834881 3.286755 2.790445 3.789987 8.167927 3.433004 4.054579 NEG  
## 30001 3.753389 3.396732 3.037759 3.596965 7.341104 3.164748 4.490154 NEG  
## 31007 3.956380 3.760465 3.125336 4.363987 7.759440 3.416183 4.340844 NEG  
## 31011 3.671596 3.393391 3.007147 3.827752 7.490177 3.398645 4.457988 POS  
## 33005 3.524609 3.210904 2.967481 3.818184 7.942288 3.101853 4.221620 NEG  
## 36001 3.785059 3.240863 2.919467 3.640368 8.200440 3.272161 4.181533 NEG  
## 36002 3.681940 3.289554 2.967583 3.699100 8.204628 3.098804 4.234036 NEG  
## 37013 3.828735 3.236902 3.292804 3.895210 7.596735 3.212299 4.152041 NEG  
## 43001 3.944849 3.619396 2.979728 4.247848 7.679805 3.310744 4.187413 POS  
## 43004 3.652408 3.370886 3.345162 3.838951 7.827539 3.455381 4.315682 NEG  
## 43007 3.803279 3.245977 2.915080 3.747391 8.792058 3.129570 4.233724 NEG  
## 43012 3.591596 3.386536 2.934657 3.773976 8.312312 3.321095 4.096838 NEG  
## 48001 3.723649 3.449407 3.114025 3.968049 7.574992 3.264674 4.212756 NEG  
## 49006 3.586389 3.604193 2.992771 3.816595 8.108770 3.211187 4.180059 NEG  
## 57001 3.466277 3.312906 2.853504 3.605181 8.101223 3.010078 4.081745 NEG  
## 62001 3.756991 3.279828 2.916028 3.704940 8.150339 3.196861 4.257715 NEG  
## 62002 3.583264 3.228700 2.891868 3.769155 7.426211 3.089218 3.988135 NEG  
## 62003 3.770456 3.397959 2.937926 3.886674 7.826372 3.306684 4.197030 NEG  
## 63001 3.769466 3.536006 2.931857 4.021004 8.460284 3.078377 4.079879 NEG  
## 64001 3.612040 3.159557 2.843461 3.712106 7.616303 3.349956 4.186390 NEG  
## 64002 3.737526 3.430320 2.911884 3.599609 7.968682 3.302960 4.301417 NEG  
## 65005 3.545069 3.237337 2.681179 3.412523 8.011825 3.159170 4.034122 NEG  
## 68001 3.768855 3.690654 3.127798 3.770301 7.395527 3.416550 4.689826 NEG  
## 68003 3.952255 3.082327 3.074194 3.745485 7.882182 3.065493 3.963959 NEG  
## 84004 3.754080 3.210164 2.785540 3.883789 8.335838 3.200252 4.067355 NEG  
## 01003 3.884864 3.225940 3.039834 3.837738 7.360750 3.340946 4.117812 NEG  
## 01007 3.874775 3.594719 2.995971 3.770501 8.511152 2.996518 4.320971 NEG  
## 02020 3.852093 3.371399 3.037056 3.933682 7.627064 3.382380 4.052852 NEG  
## 04018 3.831715 3.382257 3.111258 3.968627 8.119574 3.099611 4.268828 NEG  
## 09002 3.729932 3.460592 3.129598 3.907216 8.022574 3.501165 4.355096 NEG  
## 10005 3.791710 3.291081 3.088862 4.012088 8.083680 3.168106 4.131992 NEG  
## 11002 3.776512 3.376430 3.133082 4.156081 7.646978 3.146670 4.109477 NEG  
## 12008 3.839231 3.406565 3.006043 3.967696 7.287461 3.521776 4.049424 NEG  
## 15006 3.793211 3.575924 2.944631 3.491781 7.902353 3.411966 4.479426 NEG  
## 16002 3.561386 3.310981 2.879377 3.727971 7.771413 2.971037 4.026914 NEG  
## 16007 3.750922 3.308193 2.857163 3.721785 8.181794 3.232954 4.182935 NEG  
## 17003 3.901036 3.544285 3.275124 4.336927 7.179878 3.568819 4.236822 POS  
## 18001 3.737115 3.378427 3.246202 3.674767 8.147868 3.030334 4.407717 NEG  
## 19002 3.621329 3.496664 2.762538 3.843947 8.125779 3.263578 4.145527 NEG  
## 19008 4.537803 3.775499 4.090239 4.171892 7.545750 3.264716 5.169473 POS  
## 19014 3.690186 3.280877 2.913219 3.815795 7.772050 3.264448 4.161439 NEG  
## 19017 4.229879 3.751877 3.278537 3.680436 7.346574 3.448104 5.075954 POS  
## 20005 3.617413 3.294290 3.179639 3.929757 8.363296 3.249719 4.105627 NEG  
## 24006 3.994142 3.654828 2.926685 3.894845 7.756590 3.463517 4.143230 NEG  
## 26009 4.220419 3.277942 3.013266 3.501635 7.330038 3.320889 4.482563 NEG  
## 28008 3.679515 3.327776 2.793980 3.858081 8.219200 2.947242 4.124823 NEG  
## 28009 3.706964 3.432073 3.026657 3.972925 8.716820 3.236265 4.049223 NEG  
## 31015 3.655340 3.429285 3.070072 3.986572 7.958904 3.077103 4.091259 POS  
## 37001 3.839785 3.196011 2.903731 4.082103 8.229812 3.287839 4.291477 NEG  
## 43006 3.742395 3.248729 3.881502 3.811511 8.008063 3.260336 3.961246 POS  
## 43015 3.889773 3.140359 2.773804 3.618637 7.602782 3.145916 3.998643 NEG  
## 44001 3.619704 3.378858 3.073962 3.947810 8.143087 3.025980 4.118808 NEG  
## 49004 3.923862 3.462667 2.915762 3.667892 7.690940 3.430201 4.301597 POS  
## 56007 3.673898 3.322574 2.929108 3.866592 7.477301 3.288400 4.396975 NEG  
## 64005 3.766783 3.481573 3.150663 4.027980 7.166359 3.312031 4.329836 NEG  
## 65003 3.848170 3.746488 2.869679 3.957641 7.790916 3.399312 4.067544 NEG  
## 83001 3.743351 3.347158 3.113346 3.818819 8.869521 3.228511 4.242456 NEG

## T\_test

ALLmdr <- ALL[,ALL$mdr %in% c("NEG", "POS")]  
ttest.pValue <- apply(exprs(ALLmdr), 1, function(x) t.test(x ~ ALLmdr$mdr)$p.value)  
names\_t <- featureNames(ALL)[ttest.pValue<0.001] # getting features - Using ANOVA for dimensionality reduction with a cutoff p value - 0.001 - We list this genes   
names\_t

## [1] "33720\_at" "34898\_at" "36779\_at" "37693\_at" "38828\_s\_at"  
## [6] "38907\_at" "40293\_at"

ALLmdrnames\_t <- ALLmdr[names\_t, ]  
mdr\_data\_t=as.data.frame(t(exprs(ALLmdrnames\_t)))  
#mdr <- factor(ALLmdrnames$mdr)  
mdr\_data\_t$mdr=ALLmdr$mdr  
mdr\_data\_t

## 33720\_at 34898\_at 36779\_at 37693\_at 38828\_s\_at 38907\_at 40293\_at mdr  
## 01005 4.748294 3.052503 4.773883 4.816850 7.970141 3.326365 4.550801 NEG  
## 01010 4.562337 3.169355 5.334876 4.143627 7.063812 3.402101 4.799130 POS  
## 03002 4.059297 7.087880 4.872357 4.992990 7.138603 3.138100 4.541469 NEG  
## 04006 4.538930 5.339488 5.207546 4.261345 7.432463 3.214402 4.839809 NEG  
## 04007 4.472144 6.217558 4.665137 4.683168 8.359594 2.983769 4.396813 NEG  
## 04008 4.963730 5.695673 5.226888 5.049855 7.546242 3.195774 4.684994 NEG  
## 04010 4.315872 3.023786 5.229055 4.042616 7.625587 3.153587 4.690597 POS  
## 04016 6.206608 2.890548 4.815500 5.055254 8.191605 2.967385 4.473719 NEG  
## 06002 4.772853 3.259769 5.014375 4.495283 8.266034 3.278762 4.663583 NEG  
## 08001 4.557922 3.601918 4.774295 4.809614 7.782654 3.248337 4.632139 NEG  
## 08011 5.314593 3.344268 4.559217 5.733940 7.918819 3.551011 4.760324 NEG  
## 08012 4.836130 3.146958 4.644156 4.198300 8.936500 2.932402 4.737355 NEG  
## 08018 4.545820 3.283924 4.992371 5.017179 8.388201 3.352595 4.694405 NEG  
## 08024 4.326876 3.487737 5.148086 4.594187 7.838427 3.230736 4.802462 POS  
## 09008 4.387827 3.273433 4.980359 4.792122 7.030079 3.082421 4.601224 NEG  
## 09017 4.377305 3.268799 4.972214 4.479384 7.029666 2.927837 4.564564 NEG  
## 11005 4.572585 8.194423 4.989519 5.181461 7.539500 3.452627 4.841450 NEG  
## 12006 4.238660 3.410007 4.738563 4.415946 7.437986 3.305658 4.725263 NEG  
## 12007 4.679921 4.187375 4.957899 4.386769 7.684615 3.292940 4.653116 NEG  
## 12012 4.473797 3.451247 4.937382 4.485598 7.523282 3.209548 4.521066 NEG  
## 12019 4.621500 3.001450 5.073899 4.468714 7.746460 3.356029 4.767241 POS  
## 14016 4.226411 3.327709 5.236654 4.476877 7.144022 3.406932 4.966371 NEG  
## 15001 4.898204 6.282934 4.881969 5.677635 8.259570 3.150540 4.501990 NEG  
## 15004 5.186401 4.021529 4.834907 4.810653 7.728388 3.217198 4.490102 NEG  
## 15005 4.360449 3.008915 4.987831 4.171806 7.835973 3.480227 4.711369 POS  
## 16004 4.370491 3.238728 5.338434 4.278588 7.128244 3.566091 4.683236 NEG  
## 16009 4.474038 3.082517 5.247739 4.390692 7.279273 3.156199 4.966350 POS  
## 19005 5.069992 6.266313 5.156040 4.560799 8.231125 3.205952 4.712751 NEG  
## 20002 4.482390 2.871408 4.798750 4.638321 7.966102 3.145674 4.873320 NEG  
## 22009 4.516402 3.321925 5.120652 5.266260 7.461119 3.283860 4.708381 NEG  
## 22010 4.706461 3.328507 5.188763 4.545848 7.479274 3.428400 4.761913 NEG  
## 22011 5.364389 3.310844 4.756151 5.301279 7.744947 3.269068 4.489437 NEG  
## 22013 4.251628 3.156014 5.024268 4.973249 7.315348 3.499732 4.762920 NEG  
## 24001 5.640991 5.378275 4.657717 5.159739 7.795513 3.036268 4.186742 NEG  
## 24005 5.319058 3.028861 4.501263 4.980596 8.414925 3.326162 4.193632 NEG  
## 24008 4.954236 5.877239 4.492677 4.807597 8.010986 3.116482 4.376436 NEG  
## 24010 4.896715 4.315347 4.353978 4.575554 7.542333 3.386747 4.559146 NEG  
## 24011 4.074895 3.415070 4.969040 4.432353 7.487291 3.235082 4.643770 POS  
## 24017 3.866935 3.507562 5.137188 4.550711 7.790117 3.363856 4.901515 NEG  
## 24018 4.313035 3.885303 5.472456 4.363018 7.458638 3.443705 5.101197 POS  
## 24019 4.402762 2.995762 5.251313 4.128003 7.667427 3.347041 4.701027 NEG  
## 24022 4.310888 2.985219 4.858156 4.638439 6.863005 3.251695 4.735006 POS  
## 25003 4.463063 3.128072 5.067807 4.470462 7.730503 3.495194 4.661477 POS  
## 25006 3.771074 3.053491 5.342280 4.286926 7.058110 3.223380 5.006115 NEG  
## 26001 4.596462 3.306254 5.087418 4.448906 7.884542 3.501843 4.794579 POS  
## 26003 4.606633 3.037482 4.835391 4.566851 7.487131 3.083737 4.516843 NEG  
## 26005 4.503694 3.175239 4.963382 4.768415 7.852623 3.203465 4.800203 NEG  
## 26008 4.075310 8.277361 5.383820 4.374633 7.404470 3.206483 4.792332 NEG  
## 27003 4.183149 3.232968 5.475587 4.517047 7.468833 3.506254 4.572325 POS  
## 27004 4.466964 3.876721 4.716224 4.841007 8.135568 3.188329 4.292290 NEG  
## 28001 4.761973 3.556117 4.997091 4.103513 6.888066 3.488382 4.870297 POS  
## 28003 4.978510 3.068604 5.291013 4.646750 7.793668 3.838518 5.007726 NEG  
## 28005 4.815279 3.024954 4.778677 4.798025 7.688876 3.351883 4.534125 POS  
## 28006 4.905456 5.615002 4.386769 4.621392 7.879246 3.311279 4.314529 NEG  
## 28007 4.938089 2.713324 4.562011 4.966047 8.112087 3.074481 4.306395 NEG  
## 28019 4.398795 3.096763 5.039494 4.975972 7.490927 3.188101 4.813589 NEG  
## 28021 4.019196 2.941595 4.986918 5.606700 7.422219 3.092320 4.328651 NEG  
## 28023 4.649916 3.647329 4.839754 5.181917 8.715577 2.922662 4.390035 NEG  
## 28024 4.726726 2.876775 4.600329 4.922235 7.444096 3.149752 4.464788 NEG  
## 28028 5.148611 3.141952 4.763129 4.860666 8.408105 3.089698 4.748630 NEG  
## 28031 4.583001 3.327148 4.795012 4.680781 8.134160 3.086430 4.478770 NEG  
## 28032 4.688187 4.759578 5.240980 4.767021 7.817723 3.338060 4.698513 NEG  
## 28035 4.284429 2.709885 5.226335 4.603171 7.884421 3.355883 4.770597 POS  
## 28036 4.284650 3.140610 5.106030 4.183192 7.673960 2.918108 4.624786 NEG  
## 28037 4.181525 2.868764 4.962360 4.461980 7.310471 3.409230 4.767361 POS  
## 28042 5.176036 3.296905 4.852741 4.595050 7.765525 3.237789 4.722553 NEG  
## 28043 4.648841 3.298164 4.731436 4.624279 7.865482 3.142111 4.359033 NEG  
## 28044 4.324340 3.077589 5.067464 4.579176 8.040247 3.283038 4.502535 NEG  
## 28047 4.978036 3.436208 4.926609 4.824117 8.167927 3.433004 4.405038 NEG  
## 30001 4.407536 4.093534 5.080251 4.654825 7.341104 3.164748 4.732737 NEG  
## 31007 5.227229 3.505413 4.918875 4.398491 7.759440 3.416183 4.692399 NEG  
## 31011 4.906211 2.910137 5.178249 4.555186 7.490177 3.398645 4.597926 POS  
## 33005 4.684653 2.881488 4.998448 4.490848 7.942288 3.101853 4.686328 NEG  
## 36001 4.917802 3.093698 4.792524 4.278123 8.200440 3.272161 4.615857 NEG  
## 36002 4.699290 6.139695 4.809879 4.617560 8.204628 3.098804 4.302353 NEG  
## 37013 4.395439 2.882007 4.922160 4.704556 7.596735 3.212299 4.758249 NEG  
## 43001 4.433018 3.435542 5.104667 4.452488 7.679805 3.310744 4.726160 POS  
## 43004 4.754487 3.020615 5.378054 4.400323 7.827539 3.455381 4.730033 NEG  
## 43007 4.857070 4.469427 4.940483 4.376435 8.792058 3.129570 4.489171 NEG  
## 43012 4.873913 3.310834 4.840680 4.424542 8.312312 3.321095 4.568992 NEG  
## 48001 4.652643 3.256782 5.051152 4.771275 7.574992 3.264674 4.881042 NEG  
## 49006 4.596012 3.206671 4.789999 5.158078 8.108770 3.211187 4.474999 NEG  
## 57001 5.359862 2.947528 4.757224 3.995928 8.101223 3.010078 4.661881 NEG  
## 62001 4.807784 3.057827 4.832104 4.603317 8.150339 3.196861 4.463000 NEG  
## 62002 4.552365 3.280484 4.736745 4.564096 7.426211 3.089218 4.389532 NEG  
## 62003 4.877852 4.155647 5.213724 4.406712 7.826372 3.306684 4.874918 NEG  
## 63001 5.980415 6.111802 4.283993 6.695237 8.460284 3.078377 4.286344 NEG  
## 64001 4.747521 3.487918 4.800589 5.223079 7.616303 3.349956 4.364380 NEG  
## 64002 4.749058 3.188857 5.227282 3.722451 7.968682 3.302960 4.606834 NEG  
## 65005 4.503896 3.877372 4.635530 4.996165 8.011825 3.159170 4.278186 NEG  
## 68001 3.897863 3.073320 6.017166 3.908738 7.395527 3.416550 4.971960 NEG  
## 68003 4.439342 3.023062 4.978318 4.654642 7.882182 3.065493 5.168444 NEG  
## 84004 4.905556 5.591820 4.471884 5.671363 8.335838 3.200252 4.096175 NEG  
## 01003 4.630903 4.452096 5.099889 4.268155 7.360750 3.340946 4.718316 NEG  
## 01007 4.629421 4.809535 5.010314 4.296010 8.511152 2.996518 4.553023 NEG  
## 02020 4.560447 3.248930 5.469171 4.080328 7.627064 3.382380 4.844402 NEG  
## 04018 4.473039 3.216890 5.190512 4.044996 8.119574 3.099611 4.583370 NEG  
## 09002 5.012847 3.812883 4.988599 4.533378 8.022574 3.501165 4.673472 NEG  
## 10005 5.894138 3.321691 4.335060 4.881533 8.083680 3.168106 4.162211 NEG  
## 11002 5.057909 8.233269 4.535491 4.523267 7.646978 3.146670 4.511707 NEG  
## 12008 4.660431 5.149957 5.194343 4.476162 7.287461 3.521776 4.983201 NEG  
## 15006 4.331840 3.233572 5.250727 4.056277 7.902353 3.411966 5.120179 NEG  
## 16002 4.891928 3.512977 4.604960 4.712526 7.771413 2.971037 4.509602 NEG  
## 16007 4.570661 3.182628 4.595600 4.378601 8.181794 3.232954 4.286365 NEG  
## 17003 4.676755 3.384226 5.063275 4.246814 7.179878 3.568819 4.734707 POS  
## 18001 4.233792 2.892015 5.026329 4.660690 8.147868 3.030334 4.409395 NEG  
## 19002 5.282458 3.021111 5.225623 4.522331 8.125779 3.263578 4.792093 NEG  
## 19008 4.309404 3.463364 5.395471 4.200684 7.545750 3.264716 5.075402 POS  
## 19014 4.695341 3.039366 5.070852 4.027424 7.772050 3.264448 4.738144 NEG  
## 19017 4.680671 3.807391 5.273082 4.488206 7.346574 3.448104 4.738384 POS  
## 20005 5.192635 2.853105 5.043859 5.101967 8.363296 3.249719 4.657860 NEG  
## 24006 4.791576 3.259339 5.217033 3.845722 7.756590 3.463517 4.631493 NEG  
## 26009 4.027161 3.583581 5.530832 4.213349 7.330038 3.320889 4.573308 NEG  
## 28008 5.020789 3.403072 4.235506 4.584324 8.219200 2.947242 4.431732 NEG  
## 28009 4.533427 3.142638 4.926172 4.734120 8.716820 3.236265 4.799335 NEG  
## 31015 4.334581 3.265567 5.000622 4.313334 7.958904 3.077103 4.532241 POS  
## 37001 5.119073 3.048675 5.030321 4.298595 8.229812 3.287839 4.663241 NEG  
## 43006 4.601317 3.109099 5.054181 4.550243 8.008063 3.260336 4.590295 POS  
## 43015 4.607357 4.749075 4.954865 3.872435 7.602782 3.145916 4.529790 NEG  
## 44001 4.580531 3.316191 5.100521 4.285088 8.143087 3.025980 4.279082 NEG  
## 49004 4.817288 3.140514 5.031338 4.478476 7.690940 3.430201 4.724438 POS  
## 56007 4.187951 3.440238 4.903668 4.364435 7.477301 3.288400 4.636464 NEG  
## 64005 4.421143 3.499438 5.114051 4.617037 7.166359 3.312031 4.726162 NEG  
## 65003 4.879017 3.128274 4.850476 4.767073 7.790916 3.399312 4.767976 NEG  
## 83001 4.562119 3.107670 5.014658 3.948453 8.869521 3.228511 4.768500 NEG

##Deviding Data to Test and Train

# set the seed for reproducibility  
set.seed(123)  
#install.packages("caret")  
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

# create a 70% train and 30% test split  
#trainIndex <- createDataPartition(mdr\_data$mdr, p = 0.7, list = FALSE)  
#train <- mdr\_data[trainIndex, ]  
#test <- mdr\_data[-trainIndex, ]

Run a collection of classification models and determine which model is best perfomant on r. Models should include random forest, svm, gradient boosted tree and more

# Load necessary libraries  
library(caret)  
library(randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':  
##   
## margin

## The following object is masked from 'package:Biobase':  
##   
## combine

## The following object is masked from 'package:BiocGenerics':  
##   
## combine

library(e1071)  
library(xgboost)  
library(lattice)  
library(rpart)  
library(nnet)  
  
# Function to train models and compare performance  
train\_and\_compare\_models <- function(data, target) {  
 # Split the data into training and testing sets  
 set.seed(45)  
 index <- createDataPartition(data[[target]], p = 0.7, list = FALSE)  
 train\_set <- data[index,]  
 test\_set <- data[-index,]  
   
 # Train the random forest model  
 set.seed(42)  
 rf\_model <- train(train\_set[, -which(colnames(train\_set) == target)], train\_set[[target]], method = "rf")  
   
 # Train the support vector machine model  
 set.seed(42)  
 svm\_model <- train(train\_set[, -which(colnames(train\_set) == target)], train\_set[[target]], method = "svmRadial")  
  
 # Define the tuning grid for the gradient-boosted tree model  
 gbm\_tune\_grid <- expand.grid(  
 nrounds = 100,  
 max\_depth = 6,  
 eta = 0.3,  
 gamma = 0,  
 colsample\_bytree = 1,  
 min\_child\_weight = 1,  
 subsample = 1  
 )  
  
 # Train the gradient-boosted tree model  
 set.seed(42)  
 gbm\_model <- train(  
 train\_set[, -which(colnames(train\_set) == target)],  
 train\_set[[target]],  
 method = "xgbTree",  
 tuneGrid = gbm\_tune\_grid  
 )  
  
 # Train the recursive partitioning model  
 set.seed(42)  
 rp\_model <- train(train\_set[, -which(colnames(train\_set) == target)], train\_set[[target]], method = "rpart")  
  
 # Train the logistic regression model  
 set.seed(42)  
 glm\_model <- train(train\_set[, -which(colnames(train\_set) == target)], train\_set[[target]], method = "glm", family = "binomial")  
  
 # Train the neural network model  
 set.seed(42)  
 nn\_model <- train(train\_set[, -which(colnames(train\_set) == target)], train\_set[[target]], method = "nnet", trace = FALSE)  
   
 # Test the models  
 rf\_preds <- predict(rf\_model, test\_set[, -which(colnames(test\_set) == target)])  
 svm\_preds <- predict(svm\_model, test\_set[, -which(colnames(test\_set) == target)])  
 gbm\_preds <- predict(gbm\_model, test\_set[, -which(colnames(test\_set) == target)])  
 rp\_preds <- predict(rp\_model, test\_set[, -which(colnames(test\_set) == target)])  
 glm\_preds <- predict(glm\_model, test\_set[, -which(colnames(test\_set) == target)])  
 nn\_preds <- predict(nn\_model, test\_set[, -which(colnames(test\_set) == target)])  
   
 # Calculate confusion matrices  
 rf\_cm <- confusionMatrix(rf\_preds, test\_set[[target]])  
 svm\_cm <- confusionMatrix(svm\_preds, test\_set[[target]])  
 gbm\_cm <- confusionMatrix(gbm\_preds, test\_set[[target]])  
 rp\_cm <- confusionMatrix(rp\_preds, test\_set[[target]])  
 glm\_cm <- confusionMatrix(glm\_preds, test\_set[[target]])  
 nn\_cm <- confusionMatrix(nn\_preds, test\_set[[target]])  
   
   
# Calculate model accuracies  
 model\_accuracies <- data.frame(  
 Model = c("Random Forest", "SVM", "Gradient Boosting", "Recursive Partitioning", "Logistic Regression", "Neural Network"),  
 Accuracy = c(rf\_cm$overall["Accuracy"], svm\_cm$overall["Accuracy"], gbm\_cm$overall["Accuracy"],  
 rp\_cm$overall["Accuracy"], glm\_cm$overall["Accuracy"], nn\_cm$overall["Accuracy"])  
 )  
  
comparison\_plot <- bwplot(Accuracy ~ Model, data = model\_accuracies,  
 scales = list(x = list(rot = 45, cex = 0.8, tck = 1)))  
  
  
  
  
# Add the title using the update function from lattice package  
comparison\_plot <- update(comparison\_plot, main = "Model Performance Comparison")  
 return(list(rf\_cm = rf\_cm, svm\_cm = svm\_cm, gbm\_cm = gbm\_cm, rp\_cm = rp\_cm, glm\_cm = glm\_cm, nn\_cm = nn\_cm, comparison\_plot = comparison\_plot))  
}  
  
# Function to compute average variable importance across models  
average\_var\_importance <- function(models) {  
 var\_importances <- lapply(models, varImp)  
 variable\_names <- rownames(var\_importances[[1]])  
 avg\_importance <- Reduce('+', lapply(var\_importances, as.data.frame)) / length(var\_importances)  
 rownames(avg\_importance) <- variable\_names  
 return(avg\_importance)  
}

## Testing ANOVA data with model to predict MDR

# Run the function with your data  
# Replace "mdr\_data" with your dataframe and "mdr" with your target variable  
results <- train\_and\_compare\_models(mdr\_data, "mdr")  
  
# Display the confusion matrices  
cat("Random Forest Confusion Matrix:\n")

## Random Forest Confusion Matrix:

print(results$rf\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 28 6  
## POS 2 1  
##   
## Accuracy : 0.7838   
## 95% CI : (0.6179, 0.9017)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.7440   
##   
## Kappa : 0.0976   
##   
## Mcnemar's Test P-Value : 0.2888   
##   
## Sensitivity : 0.9333   
## Specificity : 0.1429   
## Pos Pred Value : 0.8235   
## Neg Pred Value : 0.3333   
## Prevalence : 0.8108   
## Detection Rate : 0.7568   
## Detection Prevalence : 0.9189   
## Balanced Accuracy : 0.5381   
##   
## 'Positive' Class : NEG   
##

cat("\nSupport Vector Machine Confusion Matrix:\n")

##   
## Support Vector Machine Confusion Matrix:

print(results$svm\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 30 7  
## POS 0 0  
##   
## Accuracy : 0.8108   
## 95% CI : (0.6484, 0.9204)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.59926   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 0.02334   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8108   
## Neg Pred Value : NaN   
## Prevalence : 0.8108   
## Detection Rate : 0.8108   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : NEG   
##

cat("\nGradient Boosted Tree Confusion Matrix:\n")

##   
## Gradient Boosted Tree Confusion Matrix:

print(results$gbm\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 26 6  
## POS 4 1  
##   
## Accuracy : 0.7297   
## 95% CI : (0.5588, 0.8621)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.9239   
##   
## Kappa : 0.0107   
##   
## Mcnemar's Test P-Value : 0.7518   
##   
## Sensitivity : 0.8667   
## Specificity : 0.1429   
## Pos Pred Value : 0.8125   
## Neg Pred Value : 0.2000   
## Prevalence : 0.8108   
## Detection Rate : 0.7027   
## Detection Prevalence : 0.8649   
## Balanced Accuracy : 0.5048   
##   
## 'Positive' Class : NEG   
##

cat("\nRecursive Partitioning Confusion Matrix:\n")

##   
## Recursive Partitioning Confusion Matrix:

print(results$rp\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 30 7  
## POS 0 0  
##   
## Accuracy : 0.8108   
## 95% CI : (0.6484, 0.9204)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.59926   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 0.02334   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8108   
## Neg Pred Value : NaN   
## Prevalence : 0.8108   
## Detection Rate : 0.8108   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : NEG   
##

cat("\nLogistic Regression Confusion Matrix:\n")

##   
## Logistic Regression Confusion Matrix:

print(results$glm\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 27 6  
## POS 3 1  
##   
## Accuracy : 0.7568   
## 95% CI : (0.588, 0.8823)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.8528   
##   
## Kappa : 0.0513   
##   
## Mcnemar's Test P-Value : 0.5050   
##   
## Sensitivity : 0.9000   
## Specificity : 0.1429   
## Pos Pred Value : 0.8182   
## Neg Pred Value : 0.2500   
## Prevalence : 0.8108   
## Detection Rate : 0.7297   
## Detection Prevalence : 0.8919   
## Balanced Accuracy : 0.5214   
##   
## 'Positive' Class : NEG   
##

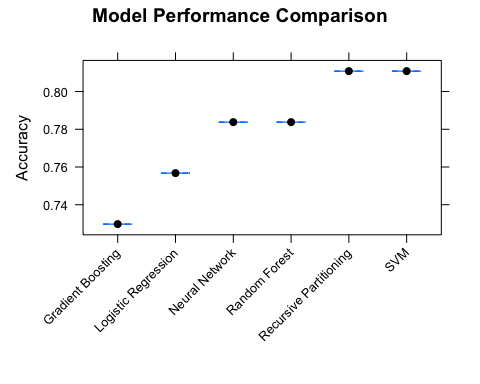
cat("\nNeural Network Confusion Matrix:\n")

##   
## Neural Network Confusion Matrix:

print(results$nn\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 28 6  
## POS 2 1  
##   
## Accuracy : 0.7838   
## 95% CI : (0.6179, 0.9017)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.7440   
##   
## Kappa : 0.0976   
##   
## Mcnemar's Test P-Value : 0.2888   
##   
## Sensitivity : 0.9333   
## Specificity : 0.1429   
## Pos Pred Value : 0.8235   
## Neg Pred Value : 0.3333   
## Prevalence : 0.8108   
## Detection Rate : 0.7568   
## Detection Prevalence : 0.9189   
## Balanced Accuracy : 0.5381   
##   
## 'Positive' Class : NEG   
##

# Display the comparison plot  
print(results$comparison\_plot)



Testing T-test Data with Models to predict MDR

# Run the function with your data  
# Replace "mdr\_data" with your dataframe and "mdr" with your target variable  
results <- train\_and\_compare\_models(mdr\_data\_t, "mdr")

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# Display the confusion matrices  
cat("Random Forest Confusion Matrix:\n")

## Random Forest Confusion Matrix:

print(results$rf\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 24 6  
## POS 6 1  
##   
## Accuracy : 0.6757   
## 95% CI : (0.5021, 0.8199)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.9852   
##   
## Kappa : -0.0571   
##   
## Mcnemar's Test P-Value : 1.0000   
##   
## Sensitivity : 0.8000   
## Specificity : 0.1429   
## Pos Pred Value : 0.8000   
## Neg Pred Value : 0.1429   
## Prevalence : 0.8108   
## Detection Rate : 0.6486   
## Detection Prevalence : 0.8108   
## Balanced Accuracy : 0.4714   
##   
## 'Positive' Class : NEG   
##

cat("\nSupport Vector Machine Confusion Matrix:\n")

##   
## Support Vector Machine Confusion Matrix:

print(results$svm\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 30 7  
## POS 0 0  
##   
## Accuracy : 0.8108   
## 95% CI : (0.6484, 0.9204)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.59926   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 0.02334   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8108   
## Neg Pred Value : NaN   
## Prevalence : 0.8108   
## Detection Rate : 0.8108   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : NEG   
##

cat("\nGradient Boosted Tree Confusion Matrix:\n")

##   
## Gradient Boosted Tree Confusion Matrix:

print(results$gbm\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 27 4  
## POS 3 3  
##   
## Accuracy : 0.8108   
## 95% CI : (0.6484, 0.9204)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.5993   
##   
## Kappa : 0.3476   
##   
## Mcnemar's Test P-Value : 1.0000   
##   
## Sensitivity : 0.9000   
## Specificity : 0.4286   
## Pos Pred Value : 0.8710   
## Neg Pred Value : 0.5000   
## Prevalence : 0.8108   
## Detection Rate : 0.7297   
## Detection Prevalence : 0.8378   
## Balanced Accuracy : 0.6643   
##   
## 'Positive' Class : NEG   
##

cat("\nRecursive Partitioning Confusion Matrix:\n")

##   
## Recursive Partitioning Confusion Matrix:

print(results$rp\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 30 7  
## POS 0 0  
##   
## Accuracy : 0.8108   
## 95% CI : (0.6484, 0.9204)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.59926   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 0.02334   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8108   
## Neg Pred Value : NaN   
## Prevalence : 0.8108   
## Detection Rate : 0.8108   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : NEG   
##

cat("\nLogistic Regression Confusion Matrix:\n")

##   
## Logistic Regression Confusion Matrix:

print(results$glm\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 26 5  
## POS 4 2  
##   
## Accuracy : 0.7568   
## 95% CI : (0.588, 0.8823)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.8528   
##   
## Kappa : 0.1612   
##   
## Mcnemar's Test P-Value : 1.0000   
##   
## Sensitivity : 0.8667   
## Specificity : 0.2857   
## Pos Pred Value : 0.8387   
## Neg Pred Value : 0.3333   
## Prevalence : 0.8108   
## Detection Rate : 0.7027   
## Detection Prevalence : 0.8378   
## Balanced Accuracy : 0.5762   
##   
## 'Positive' Class : NEG   
##

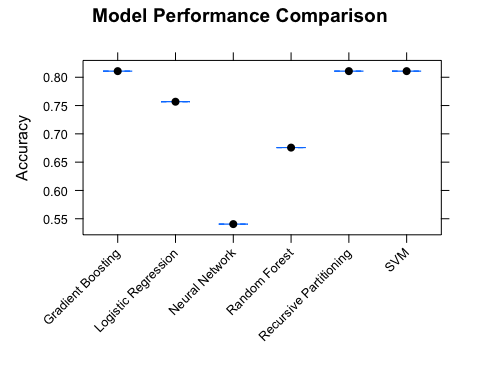
cat("\nNeural Network Confusion Matrix:\n")

##   
## Neural Network Confusion Matrix:

print(results$nn\_cm)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NEG POS  
## NEG 14 1  
## POS 16 6  
##   
## Accuracy : 0.5405   
## 95% CI : (0.3692, 0.7051)  
## No Information Rate : 0.8108   
## P-Value [Acc > NIR] : 0.999959   
##   
## Kappa : 0.1778   
##   
## Mcnemar's Test P-Value : 0.000685   
##   
## Sensitivity : 0.4667   
## Specificity : 0.8571   
## Pos Pred Value : 0.9333   
## Neg Pred Value : 0.2727   
## Prevalence : 0.8108   
## Detection Rate : 0.3784   
## Detection Prevalence : 0.4054   
## Balanced Accuracy : 0.6619   
##   
## 'Positive' Class : NEG   
##

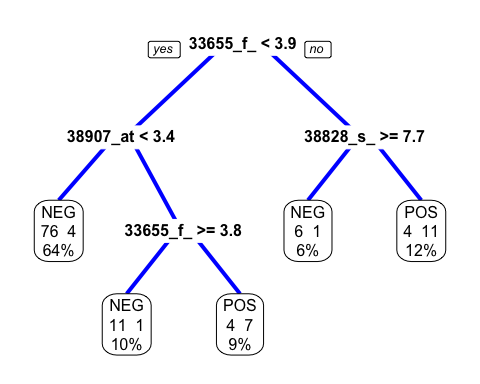
# Display the comparison plot  
print(results$comparison\_plot)



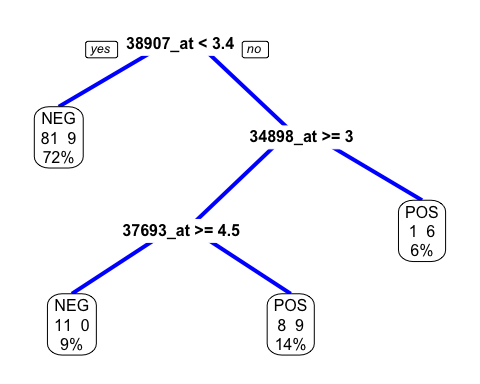
##### 

r\_part visualization after ANOVA

library(rpart)  
library(rpart.plot) # load rpart.plot package  
  
rpartFit <- rpart( mdr~. , data =mdr\_data)  
prp(rpartFit,  
 branch.lwd=4, # wide, thick branches  
 branch.col="blue",  
 extra=101)

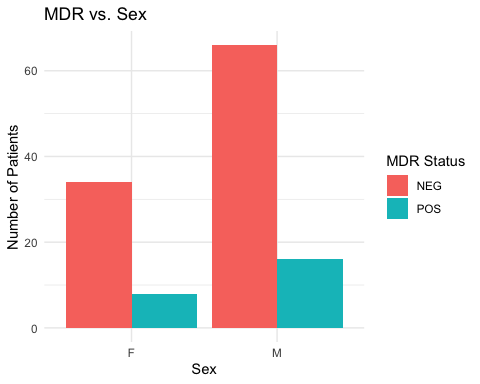
 Rpart - with t\_test

rpartFit <- rpart( mdr~. , data =mdr\_data\_t)  
prp(rpartFit,  
 branch.lwd=4, # wide, thick branches  
 branch.col="blue",  
 extra=101)



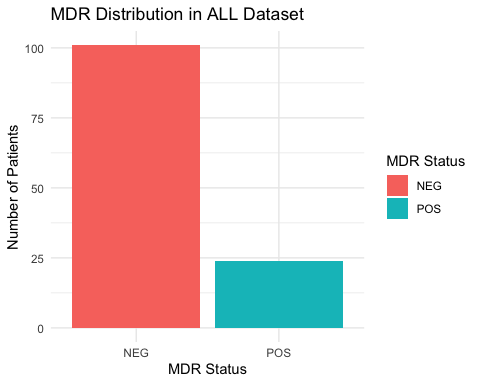
EDA

# Load necessary libraries  
library(ggplot2)  
  
# Create a summary table of MDR counts by sex  
mdr\_sex\_summary <- table(ALL$mdr, ALL$sex)  
  
# Convert the summary table into a data frame  
mdr\_sex\_df <- as.data.frame(mdr\_sex\_summary)  
colnames(mdr\_sex\_df) <- c("MDR", "sex", "count")  
  
# Create the bar plot using ggplot2  
ggplot(data = mdr\_sex\_df, aes(x = sex, y = count, fill = MDR)) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 labs(title = "MDR vs. Sex",  
 x = "Sex",  
 y = "Number of Patients",  
 fill = "MDR Status") +  
 theme\_minimal()



* Phenotypical distribution

# Load necessary libraries  
library(ggplot2)  
  
# Create a summary table of MDR counts  
mdr\_summary <- table(ALL$mdr)  
  
# Convert the summary table into a data frame  
mdr\_df <- as.data.frame(mdr\_summary)  
colnames(mdr\_df) <- c("MDR", "count")  
  
# Create the bar plot using ggplot2  
ggplot(data = mdr\_df, aes(x = MDR, y = count, fill = MDR)) +  
 geom\_bar(stat = "identity") +  
 labs(title = "MDR Distribution in ALL Dataset",  
 x = "MDR Status",  
 y = "Number of Patients",  
 fill = "MDR Status") +  
 theme\_minimal()



Age Distribution

# Load necessary libraries  
library(ggplot2)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following object is masked from 'package:xgboost':  
##   
## slice

## The following object is masked from 'package:randomForest':  
##   
## combine

## The following object is masked from 'package:Biobase':  
##   
## combine

## The following objects are masked from 'package:BiocGenerics':  
##   
## combine, intersect, setdiff, union

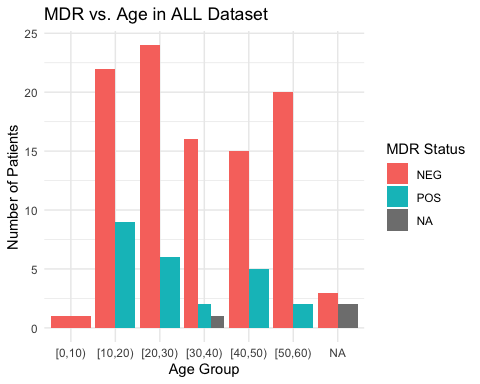
## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(Biobase)  
  
# Extract phenotype data from the ExpressionSet  
all\_pheno\_data <- pData(ALL)  
  
# Define age groups  
age\_bins <- c(0, 10, 20, 30, 40, 50, 60, 70, 120)  
  
# Bin the ages into age groups  
all\_pheno\_data$age\_group <- cut(all\_pheno\_data$age, breaks = age\_bins, right = FALSE)  
  
# Create a summary table of MDR counts by age group  
mdr\_age\_summary <- all\_pheno\_data %>%  
 group\_by(age\_group, mdr) %>%  
 summarise(count = n())

## `summarise()` has grouped output by 'age\_group'. You can override using the  
## `.groups` argument.

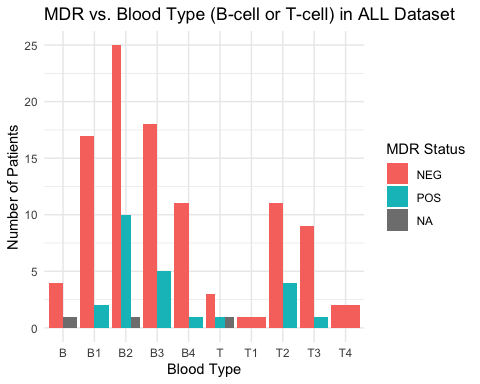
# Create the bar plot using ggplot2  
ggplot(data = mdr\_age\_summary, aes(x = age\_group, y = count, fill = mdr)) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 labs(title = "MDR vs. Age in ALL Dataset",  
 x = "Age Group",  
 y = "Number of Patients",  
 fill = "MDR Status") +  
 theme\_minimal()

 BT

# Load necessary libraries  
library(ggplot2)  
library(dplyr)  
library(Biobase)  
  
# Extract phenotype data from the ExpressionSet  
all\_pheno\_data <- pData(ALL)  
  
# Create a summary table of MDR counts by blood type  
mdr\_bt\_summary <- all\_pheno\_data %>%  
 group\_by(BT, mdr) %>%  
 summarise(count = n())

## `summarise()` has grouped output by 'BT'. You can override using the `.groups`  
## argument.

# Create the bar plot using ggplot2  
ggplot(data = mdr\_bt\_summary, aes(x = BT, y = count, fill = mdr)) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 labs(title = "MDR vs. Blood Type (B-cell or T-cell) in ALL Dataset",  
 x = "Blood Type",  
 y = "Number of Patients",  
 fill = "MDR Status") +  
 theme\_minimal()

 Remission

# Load necessary libraries  
library(ggplot2)  
library(dplyr)  
library(Biobase)  
  
# Extract phenotype data from the ExpressionSet  
all\_pheno\_data <- pData(ALL)  
  
# Create a summary table of MDR counts by remission status  
mdr\_remission\_summary <- all\_pheno\_data %>%  
 group\_by(remission, mdr) %>%  
 summarise(count = n())

## `summarise()` has grouped output by 'remission'. You can override using the  
## `.groups` argument.

# Create the bar plot using ggplot2  
ggplot(data = mdr\_remission\_summary, aes(x = remission, y = count, fill = mdr)) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 labs(title = "MDR vs. Remission Status in ALL Dataset",  
 x = "Remission Status",  
 y = "Number of Patients",  
 fill = "MDR Status") +  
 theme\_minimal()

