# Final Project phase 2

## Evan Hildreth

ames <- read\_csv("ames\_student.csv")

## Rows: 2053 Columns: 81

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## chr (47): MS\_SubClass, MS\_Zoning, Street, Alley, Lot\_Shape, Land\_Contour, Ut...  
## dbl (34): Lot\_Frontage, Lot\_Area, Year\_Built, Year\_Remod\_Add, Mas\_Vnr\_Area, ...

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

In phase 1 of this project we conducted an exploratory data analysis on the Ames dataset to examine what might be strong variables to predict which homes in Ames, Iowa were / are to be sold for ‘Above\_Median’ selling prices. Out of the 80 variables in this Ames dataset 20 were found to be potentially good predictors of “Above\_Median.” So our models will be built off of these 20 variables we identified and we will not consider the other 59 unless deemed necessary to expand.

The one “created field” from our phase 1 analysis, ‘total baths’ will also be included.

ames <- mutate(ames, Total\_Baths = Half\_Bath + Bsmt\_Full\_Bath + Bsmt\_Half\_Bath + Full\_Bath)

ames\_select <- ames %>% dplyr::select(Alley, Utilities, Overall\_Qual,Roof\_Matl,Exter\_Qual,Bsmt\_Qual,Kitchen\_Qual,Fireplace\_Qu,Paved\_Drive,Pool\_QC,Misc\_Feature,Garage\_Type, Neighborhood,Year\_Built,Year\_Remod\_Add,Bsmt\_Unf\_SF,Total\_Bsmt\_SF,TotRms\_AbvGrd,Total\_Baths,Garage\_Area, Above\_Median)

We are also going to do the same data transformations where we turn each of the character values into factors as this will make it easier to build our various models with, especially as we compare to Above\_Median.

ames\_select <- ames\_select %>% mutate\_if(sapply(ames\_select, is.character), as.factor)

#summary(ames\_select)

#str(ames\_select)

After we have selected the values we wanted and made the necessary transformations I wanted to take a brief moment to check of the variables in the dataset. Since in phase 1 we did not find any missing values and we are using a subset of the same dataset we can assume that there will not be any missing values. For the markdown they will be commented out to conserve space.

### Data training/testing split

For the sake of simplicity and uniformity between our different models we will the same 60/40 training/testing split for our models.

set.seed(1234)  
ames\_select\_split <- initial\_split(ames\_select, prop = 0.6, strata = Above\_Median)  
ames\_select\_train <- training(ames\_select\_split)  
ames\_select\_test <- testing(ames\_select\_split)

From these splits we will look at 3 types of models: Logistic Regression, classification trees, and neural networks. The main indicator of a successful model will be looking at accuracy, specificity, and sensitivity and the main metrics of the strength of a model.

## Logistic Regression

The first model type we will look at is logistic regression, which is arguably the most simple model type as well. For the sake experimentation we will first use a model that will contain all of our variables and then narrow down which ones prove to be best.

### Logistic Regression prep and training

ames\_model\_lg <-   
 logistic\_reg(mode = "classification") %>%  
 set\_engine("glm")   
  
ames\_recipe\_lg <- recipe(Above\_Median ~., ames\_select\_train)  
  
logreg\_wf <- workflow() %>%  
 add\_recipe(ames\_recipe\_lg) %>%   
 add\_model(ames\_model\_lg)  
  
ames\_fit\_lg <- fit(logreg\_wf, ames\_select\_train)

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

options(scipen = 999)  
#summary(ames\_fit\_lg$fit$fit$fit)  
#for sake of consciousness I am going to comment out this summary as it is quite long

From looking at these results with all of the variables it looks like the most significant ones are Total\_Bsmt\_SF, Totrms\_AbvGrd, Total\_Baths, Garage\_Detatchd, and Overal\_Qual\_Avg with Garage\_Area, Year\_Remod\_Add, Bsmt\_Unf\_SF, and Overall\_Qual\_Excellent being strong but not as strong predictors. We will run this model again with just these variables to see if we can get an lower (better) AIC than 676.85

ames\_recipe\_lg2 = recipe(Above\_Median ~ Overall\_Qual + Garage\_Type + Year\_Remod\_Add + Bsmt\_Unf\_SF + Total\_Bsmt\_SF + TotRms\_AbvGrd + Total\_Baths + Garage\_Area, ames\_select\_train)  
  
logreg\_wf2 = workflow() %>%  
 add\_recipe(ames\_recipe\_lg2) %>%   
 add\_model(ames\_model\_lg)  
  
ames\_fit\_lg2 = fit(logreg\_wf2, ames\_select\_train)

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

options(scipen = 999)  
#summary(ames\_fit\_lg2$fit$fit$fit)  
#Same as the previous chunk

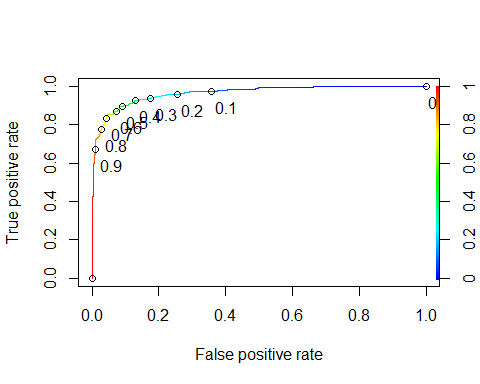
So when we tried to run the ‘tuned’ logistic regression model looking at only the statistically significant variables we actually see a slightly higher AIC of 640.93 compared to the AIC we got for all variables of 597.69

Next, we are going to run predictions on the selected model set as running with every variable seems to be too large to work with and only gives a minor decrease in AIC values.

predictions\_log\_reg\_train <- predict(ames\_fit\_lg2, ames\_select\_train, type="prob")  
  
#head(predictions\_log\_reg\_train)

predictions\_log\_reg\_train <- predict(ames\_fit\_lg2, ames\_select\_train, type="prob")[2]  
  
#head(predictions\_log\_reg\_train)

ROCRpred\_ames\_log\_reg\_train <- prediction(predictions\_log\_reg\_train, ames\_select\_train$Above\_Median)  
  
ROCRperf\_ames\_log\_reg\_train <- performance(ROCRpred\_ames\_log\_reg\_train, "tpr", "fpr")  
plot(ROCRperf\_ames\_log\_reg\_train, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



From looking at the AUC curve we do see that this does look to be a fair strong model as it has a very low false positive rate and the curve is fairly close to a right angle.

as.numeric(performance(ROCRpred\_ames\_log\_reg\_train, "auc")@y.values)

## [1] 0.9638323

opt.cut <- function(perf, pred){  
 cut.ind <- mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],  
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf\_ames\_log\_reg\_train, ROCRpred\_ames\_log\_reg\_train))

## [,1]  
## sensitivity 0.8928000  
## specificity 0.9174917  
## cutoff 0.5141958

Running these numbers we see that we have a very high sensitivity and specificity score using the training data. Next we will try to evaluate accuracy.

t1 = table(ames\_select\_train$Above\_Median,predictions\_log\_reg\_train > 0.4340579)  
t1

##   
## FALSE TRUE  
## No 532 74  
## Yes 55 570

(t1[1,1]+t1[2,2])/nrow(ames\_select\_train)

## [1] 0.8952071

And with this log\_reg model we come up with an accuracy of ~89% which already is very solid but it could be improved so we will do some threshold testing.

t2 <- table(ames\_select\_train$Above\_Median,predictions\_log\_reg\_train > 0.5)  
t2

##   
## FALSE TRUE  
## No 551 55  
## Yes 64 561

(t2[1,1]+t2[2,2])/nrow(ames\_select\_train)

## [1] 0.9033306

Running through a few thresholds I found that the one that produced the highest accuracy is 0.5 producing an accuracy of of ~90% which is only a mild improvement over the original accuracy value of ~89.5%. For this application of a 90% accuracy rating with sensitivity and specificity of 89% and 91% receptively.

t3 <- table(ames\_select\_train$Above\_Median,predictions\_log\_reg\_train > 1)  
t3

##   
## FALSE  
## No 606  
## Yes 625

(t3[1])/nrow(ames\_select\_train)

## [1] 0.4922827

For good measure I also wanted to judge this against a naive model that made the impossible assumption that all homes were sold Above\_Medain which resulted in an accuracy of ~49%. So I think it is safe to say that we have a sold model based off the training dataset.

### Logistic Regression Testing Set

However, we want to make sure our data is not over-fitted to the training data and so we are going to re-run the previous steps but with the testing data to see how it compares.

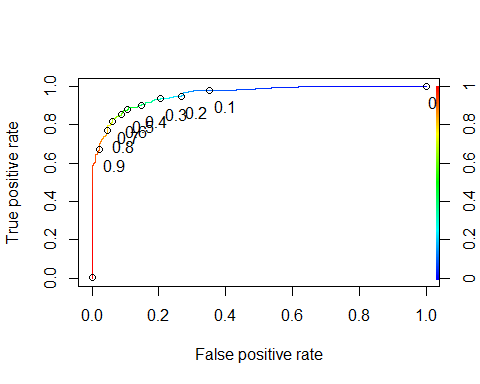
predictions\_log\_reg\_test <- predict(ames\_fit\_lg2, ames\_select\_test, type="prob")  
  
head(predictions\_log\_reg\_test)

## # A tibble: 6 x 2  
## .pred\_No .pred\_Yes  
## <dbl> <dbl>  
## 1 0.504 0.496  
## 2 0.603 0.397  
## 3 0.00230 0.998  
## 4 0.118 0.882  
## 5 0.00362 0.996  
## 6 0.0946 0.905

predictions\_log\_reg\_test <- predict(ames\_fit\_lg2, ames\_select\_test, type="prob")[2]  
  
head(predictions\_log\_reg\_test)

## # A tibble: 6 x 1  
## .pred\_Yes  
## <dbl>  
## 1 0.496  
## 2 0.397  
## 3 0.998  
## 4 0.882  
## 5 0.996  
## 6 0.905

ROCRpred\_ames\_log\_reg\_test <- prediction(predictions\_log\_reg\_test, ames\_select\_test$Above\_Median)  
  
ROCRperf\_ames\_log\_reg\_test <- performance(ROCRpred\_ames\_log\_reg\_test, "tpr", "fpr")  
plot(ROCRperf\_ames\_log\_reg\_test, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7))



as.numeric(performance(ROCRpred\_ames\_log\_reg\_test, "auc")@y.values)

## [1] 0.9568075

opt.cut <- function(perf, pred){  
 cut.ind <- mapply(FUN=function(x, y, p){  
 d = (x - 0)^2 + (y-1)^2  
 ind = which(d == min(d))  
 c(sensitivity = y[[ind]], specificity = 1-x[[ind]],  
 cutoff = p[[ind]])  
 }, perf@x.values, perf@y.values, pred@cutoffs)  
}  
print(opt.cut(ROCRperf\_ames\_log\_reg\_test, ROCRpred\_ames\_log\_reg\_test))

## [,1]  
## sensitivity 0.8875598  
## specificity 0.8910891  
## cutoff 0.4812254

Running the testing dataset split we do see slightly lower values for sensitivity and specificity than what we saw with the training dataset but not significantly so that can’t potentially be chalked up to a smaller sample size (at least in part).

t4 <- table(ames\_select\_test$Above\_Median,predictions\_log\_reg\_test > 0.4812254)  
t4

##   
## FALSE TRUE  
## No 360 44  
## Yes 47 371

(t4[1,1]+t4[2,2])/nrow(ames\_select\_test)

## [1] 0.8892944

Again we see a slightly lower accuracy of 88% compared to the initial accuracy of 89% with the training set which again could be chalked up to a smaller sample size. But I would say at this point, without any threshold testing, we can pretty clear say that this model is very strong and does not suffer form any over-fitting.

However, for good measure we will do some threshold testing,

t5 <- table(ames\_select\_test$Above\_Median,predictions\_log\_reg\_test > 0.5)  
t5

##   
## FALSE TRUE  
## No 361 43  
## Yes 52 366

(t5[1,1]+t5[2,2])/nrow(ames\_select\_test)

## [1] 0.8844282

Running a few different threshold values we actually were not really able to tease out any higher accuracy score than what we did with the original threshold.

### Logistic Regression consulusions

Overall, this ended up being a very strong model with high scores for all major metrics for the model on both the training and testing dataset. Naturally if we were doing much more critical work that required a higher level of precision and accuracy (like fraud detection or disease prediction) but for predicting a relatively lower-risk item, such as home prices being above or below median; I am happy with these results.

## Classification Trees

The next model type we will explore is classification trees. This is a bit more complex of a model and approach compared to logistic regression to create (sometimes) but it does have the advantage of being easier to understand since it actually can be visualized.

For this we will use the same 60/40 split that we used for the logistic regression models and we will also start off by using all of the 20 variables we have in out ames\_select dataset and will further narrow down if necessary. First, we will set up our initial model using all the variables.

### Building the model

ames\_recipe\_CT <- recipe(Above\_Median ~., ames\_select\_train)  
  
tree\_model <- decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")   
  
ames\_CT\_wflow <-   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(ames\_recipe\_CT)  
  
ames\_CT\_fit <- fit(ames\_CT\_wflow, ames\_select\_train)

The model is now built now we are going to look at the results of this model.

ames\_CT\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")

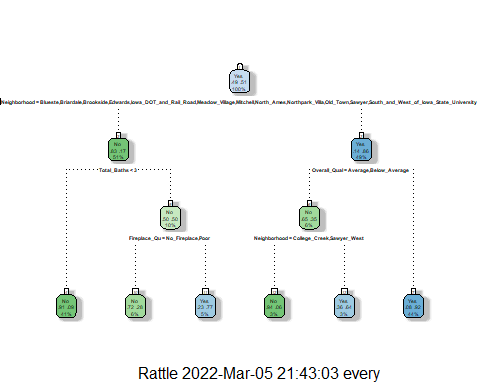
## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## Please use `extract\_fit\_parsnip()` instead.

## n= 1231   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 1231 606 Yes (0.49228270 0.50771730)   
## 2) Neighborhood=Blueste,Briardale,Brookside,Edwards,Iowa\_DOT\_and\_Rail\_Road,Meadow\_Village,Mitchell,North\_Ames,Northpark\_Villa,Old\_Town,Sawyer,South\_and\_West\_of\_Iowa\_State\_University 623 105 No (0.83146067 0.16853933)   
## 4) Total\_Baths< 2.5 499 43 No (0.91382766 0.08617234) \*  
## 5) Total\_Baths>=2.5 124 62 No (0.50000000 0.50000000)   
## 10) Fireplace\_Qu=No\_Fireplace,Poor 68 19 No (0.72058824 0.27941176) \*  
## 11) Fireplace\_Qu=Excellent,Fair,Good,Typical 56 13 Yes (0.23214286 0.76785714) \*  
## 3) Neighborhood=Bloomington\_Heights,Clear\_Creek,College\_Creek,Crawford,Gilbert,Greens,Northridge,Northridge\_Heights,Northwest\_Ames,Sawyer\_West,Somerset,Stone\_Brook,Timberland,Veenker 608 88 Yes (0.14473684 0.85526316)   
## 6) Overall\_Qual=Average,Below\_Average 71 25 No (0.64788732 0.35211268)   
## 12) Neighborhood=College\_Creek,Sawyer\_West 35 2 No (0.94285714 0.05714286) \*  
## 13) Neighborhood=Clear\_Creek,Crawford,Gilbert,Northwest\_Ames,Somerset,Timberland,Veenker 36 13 Yes (0.36111111 0.63888889) \*  
## 7) Overall\_Qual=Above\_Average,Excellent,Good,Very\_Excellent,Very\_Good 537 42 Yes (0.07821229 0.92178771) \*

ames\_select\_tree = ames\_CT\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## Please use `extract\_fit\_parsnip()` instead.

fancyRpartPlot(ames\_select\_tree)



ames\_CT\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.68151815 0 1.0000000 1.0000000 0.02894507  
## 2 0.03465347 1 0.3184818 0.3217822 0.02113954  
## 3 0.02475248 2 0.2838284 0.3102310 0.02082665  
## 4 0.01650165 4 0.2343234 0.2722772 0.01972509  
## 5 0.01000000 5 0.2178218 0.2656766 0.01952105

Looking at this classification tree we can see that the model ‘automatically’ choose a number of variables that it though was most significant to determine if a home was sold for above median pricing or not. From this it seems that the category that has the most ‘yes’ would be for the home to be in one of a few neighborhoods and having the home overall quality to be average or bellow average. This seems to go again some of our intuitive ideas about the data but this is what we found. However, it seems that the highest likelihood to be below\_median is for homes to be not in one of those select neighborhoods and to have less than 3 baths. This seems to be a bit more intuitive than the “yes” category.

Next we will gather summary statistics on the performance of this model using the training data and then compare with the testing data.

### Model peformance

ames\_ct\_pred\_train <- predict(ames\_CT\_fit, ames\_select\_train, type = "class")  
#head(ames\_ct\_pred\_train)  
#The above head was done to test that only .pred\_class shows up.

confusionMatrix(ames\_ct\_pred\_train$.pred\_class,ames\_select\_train$Above\_Median,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 538 64  
## Yes 68 561  
##   
## Accuracy : 0.8928   
## 95% CI : (0.8741, 0.9095)   
## No Information Rate : 0.5077   
## P-Value [Acc > NIR] : <0.0000000000000002  
##   
## Kappa : 0.7855   
##   
## Mcnemar's Test P-Value : 0.794   
##   
## Sensitivity : 0.8976   
## Specificity : 0.8878   
## Pos Pred Value : 0.8919   
## Neg Pred Value : 0.8937   
## Prevalence : 0.5077   
## Detection Rate : 0.4557   
## Detection Prevalence : 0.5110   
## Balanced Accuracy : 0.8927   
##   
## 'Positive' Class : Yes   
##

From running this model on the training set we find that we get an accuracy of about ~89% which when rounded is the same as accuracy as with the logistic regression model we created earlier using the same training data, which again is fairly strong for this use case. It is also encouraging that we have a very low P-Value indicating that this model is statistically significant. We might have some concern that it could be lower, but that’s getting picky.

Next, lets check for performance using the testing data.

ames\_ct\_pred\_test <- predict(ames\_CT\_fit, ames\_select\_test, type = "class")  
#head(ames\_ct\_pred\_train)

confusionMatrix(ames\_ct\_pred\_test$.pred\_class,ames\_select\_test$Above\_Median,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 345 61  
## Yes 59 357  
##   
## Accuracy : 0.854   
## 95% CI : (0.828, 0.8775)   
## No Information Rate : 0.5085   
## P-Value [Acc > NIR] : <0.0000000000000002  
##   
## Kappa : 0.708   
##   
## Mcnemar's Test P-Value : 0.9273   
##   
## Sensitivity : 0.8541   
## Specificity : 0.8540   
## Pos Pred Value : 0.8582   
## Neg Pred Value : 0.8498   
## Prevalence : 0.5085   
## Detection Rate : 0.4343   
## Detection Prevalence : 0.5061   
## Balanced Accuracy : 0.8540   
##   
## 'Positive' Class : Yes   
##

Similar to when we tested using the testing data for the logistic regression model we find that there is a slight dip in accuracy compared to the training. Again this could be accounted to being a smaller dataset. However this is a bit more of a drop compared to its logistic regression counterpart with ~85% accuracy for this classification tree and 88% for the logistic regression. Overall I would say this is a very strong model but maybe less so than its logistic regression counterpart.

### Clasification treees model conclusions

Another very strong model but not as strong compared to logistic regression. We could potentially spend some time to ‘prune’ the dataset to squeeze out some better performance from the classification trees model like we did with logistic regression but the model did that for us ‘automatically’ and the variables it choose to work with are very similar to the ones we did with the logistic regression model ‘manually.’ Good model, just could be better.

## Neural Networks

The last model type we will run will be neural networks. In many ways neural networks work similar to classification trees but on a much more complex scale. The complexity can be beneficial in that it can find some more novel groupings and connections with the data that can sometimes lead to some increased accuracy but the complexity comes with the cost of processing time, but in computation and sometimes the coding.

First, we will use the same training/ testing splits that we have used to set up and then test the neural networks model.

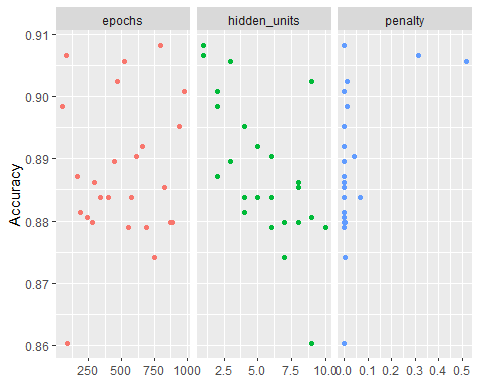
### Creating the neural network model

set.seed(1234)  
fold <- vfold\_cv(ames\_select\_train, v=5)  
  
ames\_select\_NN\_recipe\_train <- recipe(Above\_Median ~., ames\_select\_train) %>%  
 step\_normalize(all\_predictors(), -all\_nominal()) %>% #normalize the numeric predictors, not needed for categorical  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
ames\_select\_NN\_model <-   
 mlp(hidden\_units = tune(), penalty = tune(),   
 epochs = tune()) %>%  
 set\_mode("classification") %>%   
 set\_engine("nnet", verbose = 0)   
  
ames\_select\_NN\_workflow <-   
 workflow() %>%   
 add\_recipe(ames\_select\_NN\_recipe\_train) %>%   
 add\_model(ames\_select\_NN\_model)  
  
set.seed(1234)  
ames\_select\_NN\_tune <-  
 tune\_grid(ames\_select\_NN\_workflow, resamples = fold, grid = 25)

Just to comment on the processing power needed for this model it took about 6 minutes for this model to render and a decently powerful laptop.

Now that we have created the model we are going to graph it out and then we are going to optimize the model using the best neural connections from it.

ames\_select\_NN\_tune %>%  
 collect\_metrics() %>%  
 filter(.metric == "accuracy") %>%  
 select(mean, hidden\_units, penalty, epochs) %>%  
 pivot\_longer(hidden\_units:epochs,  
 values\_to = "value",  
 names\_to = "parameter"  
 ) %>%  
 ggplot(aes(value, mean, color = parameter)) +  
 geom\_point(show.legend = FALSE) +  
 facet\_wrap(~parameter, scales = "free\_x") +  
 labs(x = NULL, y = "Accuracy")



From what we see here there is a range of values in terms of accuracy but it is more or less about what we saw from the other 2 models so far (a range of ~86% to ~91%). Hidden units and penalty look to be in line with what we would expect for a neural network.

### Model Peformance

Next we are going to select the best values from this model and then find the performance metrics for this model.

ames\_select\_best\_nn <- select\_best(ames\_select\_NN\_tune, "accuracy")  
  
ames\_select\_final\_nn = finalize\_workflow(  
 ames\_select\_NN\_workflow,  
 ames\_select\_best\_nn  
)  
  
ames\_select\_final\_nn

## == Workflow ====================================================================  
## Preprocessor: Recipe  
## Model: mlp()  
##   
## -- Preprocessor ----------------------------------------------------------------  
## 2 Recipe Steps  
##   
## \* step\_normalize()  
## \* step\_dummy()  
##   
## -- Model -----------------------------------------------------------------------  
## Single Layer Neural Network Specification (classification)  
##   
## Main Arguments:  
## hidden\_units = 1  
## penalty = 0.00000000428607140962385  
## epochs = 796  
##   
## Engine-Specific Arguments:  
## verbose = 0  
##   
## Computational engine: nnet

ames\_select\_final\_nn\_fit <- fit(ames\_select\_final\_nn, ames\_select\_train)  
  
ames\_select\_predict\_train <- predict(ames\_select\_final\_nn\_fit, ames\_select\_train)  
#head(ames\_select\_predict\_train)

confusionMatrix(ames\_select\_predict\_train$.pred\_class, ames\_select\_train$Above\_Median, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 564 41  
## Yes 42 584  
##   
## Accuracy : 0.9326   
## 95% CI : (0.9171, 0.9459)   
## No Information Rate : 0.5077   
## P-Value [Acc > NIR] : <0.0000000000000002  
##   
## Kappa : 0.8651   
##   
## Mcnemar's Test P-Value : 1   
##   
## Sensitivity : 0.9344   
## Specificity : 0.9307   
## Pos Pred Value : 0.9329   
## Neg Pred Value : 0.9322   
## Prevalence : 0.5077   
## Detection Rate : 0.4744   
## Detection Prevalence : 0.5085   
## Balanced Accuracy : 0.9325   
##   
## 'Positive' Class : Yes   
##

Very interesting results! From the looks of things, this appears to be the strongest model so far in that we have the same P-value compared to the other models using the training data, but the accuracy is much higher at ~93%! It seems that the additional computing time to build the model might actually pay off it it means we can squeeze an extra 3-4% accuracy out.

Naturally, we cannot claim victory for neural networks just yet; we need to test it using the testing data.

ames\_select\_predict\_test <- predict(ames\_select\_final\_nn\_fit, ames\_select\_test)  
#head(ames\_select\_predict\_train)

confusionMatrix(ames\_select\_predict\_test$.pred\_class, ames\_select\_test$Above\_Median, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 361 37  
## Yes 43 381  
##   
## Accuracy : 0.9027   
## 95% CI : (0.8803, 0.9221)   
## No Information Rate : 0.5085   
## P-Value [Acc > NIR] : <0.0000000000000002  
##   
## Kappa : 0.8052   
##   
## Mcnemar's Test P-Value : 0.5762   
##   
## Sensitivity : 0.9115   
## Specificity : 0.8936   
## Pos Pred Value : 0.8986   
## Neg Pred Value : 0.9070   
## Prevalence : 0.5085   
## Detection Rate : 0.4635   
## Detection Prevalence : 0.5158   
## Balanced Accuracy : 0.9025   
##   
## 'Positive' Class : Yes   
##

### Neural Network conclusion

I think at this point it is pretty safe to say that at this point neural networks seems to be the strongest model for this dataset as we have an slightly lower accuracy score on the testing data (like with all the other models) but even the testing model accuracy is a bit higher than some of the other model’s on the training dataset. Beyond that the sensitivity and specificity scores for the testing data here are right up there with the scores for the training data on the other models we created.

In a use-case for the business we will need to weigh the pros and cons of using a neural networks model that provides the best performance but at the cost of much longer processing time, but if it is just output performance neural networks seems to be the best.