Data Analysis Report – Final Draft: 36-290 Fall 2021

christina choi

11/27/2021

Classification of Active Galaxies Observed by SDSS

Introduction

The Sloan Digital Sky Survey has a catalog containing data on over 200 million galaxies. Galaxy data typically include images along with measures of brightness from five different bandpasses (denoted u, g, r, i, and z) spanning the optical regime of the electromagnetic spectrum. These measures of brightness, or magnitudes, can reveal interesting information about galaxies. When a galaxy is *active*–meaning it forms stars at a relatively greater rate or has a supermassive black hole in its center that consumes stars/gas/dust at an enhanced rate–its spectrum will reveal "spikes" called emission lines. These emission lines along with other features from a spectra can be used to make inferences about whether the galaxy is star-forming or whether it has active nucleus.

We will attempt to classify galaxies as either starform or having an active nucleus using features from their spectra with the dataset created by Zhang et al. (2019).

There are a total of five predictor variables used in this investigation, with one response variable:

Variables	**Description of Variables**
u_g, g_r, r_i, i_z	The four colors of the galaxy. The colors are differences in logarithmic measures of brightness, or magnitudes. Magnitudes are highly correlated with each other as well as galaxy distance.
z	Galaxy redshift. Redshift refers to the ratio of the observed wavelength of a photon from an object to its wavelength when it was emitted, minus 1.
label	The factor response variable in this dataset, that indicates whether a given galaxy is observed to be star-forming (denoted `STARFORM`) or with active nuclei (denoted `AGN`).

Data

The variable `df` is a data frame with 28,820 rows and ten columns that contains information on 28,820 different galaxies. The last column of the data frame named `label` contains the response variable, which labels each galaxy as either star-forming (denoted `STARFORM`) or with active nuclei (denoted `AGN`).

The response classes are not perfectly balanced, as there are 15,521 STARFORM and 13299 AGN observations. STARFORM galaxies make up roughly 53.9% of the dataset, while AGN galaxies make up roughly 46.1% of the data.

The following histograms visualize the distribution of the predictor variables mentioned above.

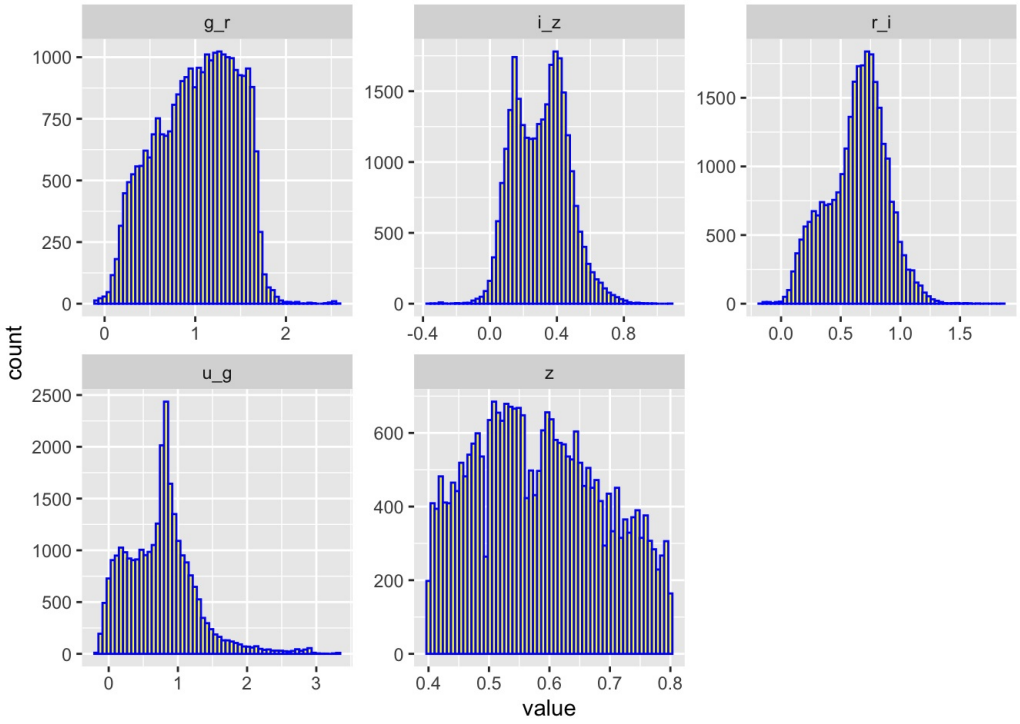
There are variables we will not be using that are present in the dataframe–these particular variables have been removed as seen below.

Additionally, there are several outliers present in the data. In order to get rid of the irregularities in the data, the rows for which g-r is exactly 0 have been filtered out.

```
df<- df %>% filter(., g_r!= 0)
```

```
df <- df %>% select(-O3_Hb, -O2_Hb,-sigma_o3, -sigma_star)
```

```
df.gathered <- df %>% select(.,u_g,g_r,r_i,i_z,z) %>% gather(.)
ggplot(data=df.gathered,mapping=aes(x=value)) + geom_histogram(color="blue",fill="yellow",bins=60) +
  facet_wrap(~key, scales='free')
```



After being filtered, `df` now has 28,151 rows and ten columns. There are now 15,306 STARFORM galaxies (which make up roughly 54.37% of the total data), and 12,845 AGN galaxies (which make up roughly 45.63% of the total data).

Below is a summary of the dataset that we will be working with, and a list of the names of the predictor variables. Outlier data has been filtered out, as mentioned above.

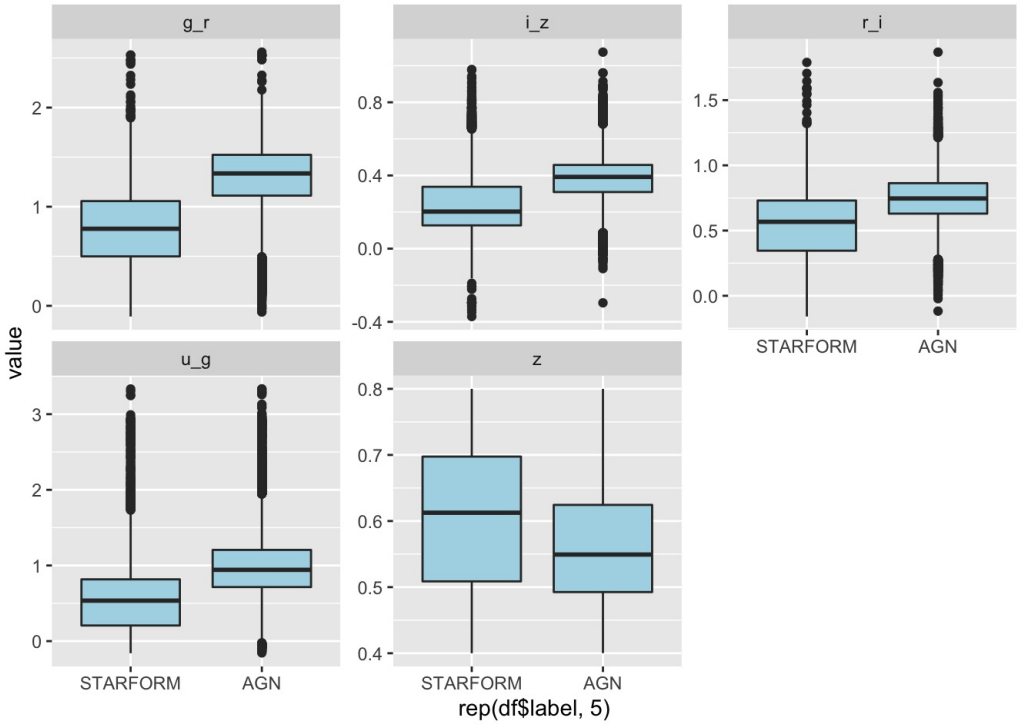
```
summary(df)
```

```
##        z                u_g                g_r                r_i         
##  Min.   :0.4000   Min.   :-0.1599   Min.   :-0.1078   Min.   :-0.1589  
##  1st Qu.:0.5017   1st Qu.: 0.3795   1st Qu.: 0.6809   1st Qu.: 0.4780  
##  Median :0.5812   Median : 0.7688   Median : 1.0534   Median : 0.6658  
##  Mean   :0.5850   Mean   : 0.7497   Mean   : 1.0164   Mean   : 0.6382  
##  3rd Qu.:0.6647   3rd Qu.: 0.9969   3rd Qu.: 1.3712   3rd Qu.: 0.8038  
##  Max.   :0.8000   Max.   : 3.3347   Max.   : 2.5594   Max.   : 1.8681  
##       i_z               label      
##  Min.   :-0.3709   STARFORM:15306  
##  1st Qu.: 0.1701   AGN     :12845  
##  Median : 0.3109                   
##  Mean   : 0.3049                   
##  3rd Qu.: 0.4199                   
##  Max.   : 1.0742
```

Looking at the boxplots that display the distribution of the different variables as well as their respective labels, there does seem to be a visual association between the predictors and the classes “AGN” and “STARFORM”.

For example, taking a look at the color predictor variables, the galaxies labeled as AGN tend to have greater median magnitudes compared to those labeled as STARFORM.

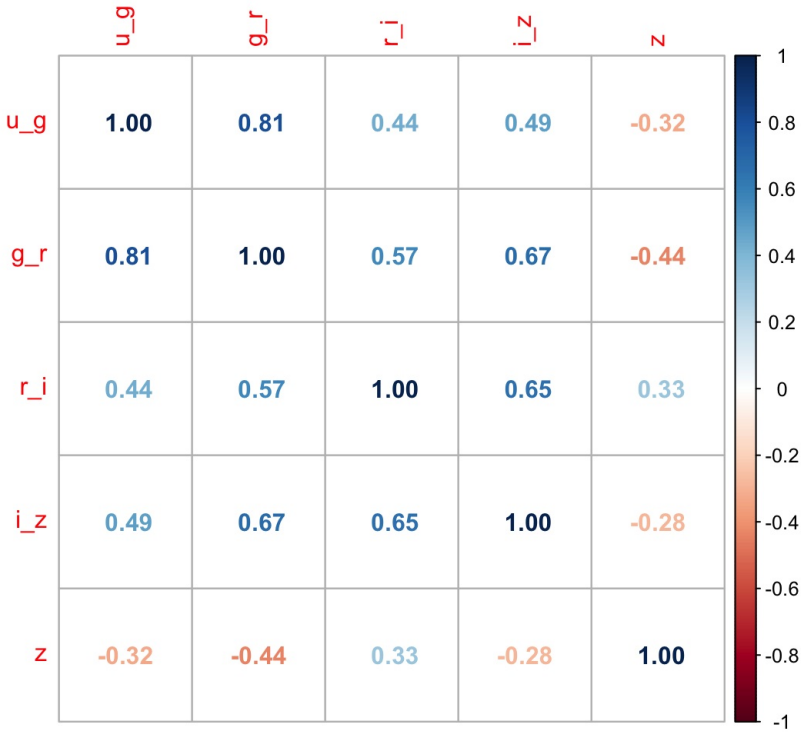
```
df.facet = df %>% dplyr::select(.,-label) %>% gather(.)
ggplot(data=df.facet,mapping=aes(y=value,x=rep(df$label,5))) + geom_boxplot(fill="lightblue") + facet_wrap(~key,s
cales='free_y')
```



Due to the fact that there are clear visual associations between some of the predictor variables and the response, statistical learning will allow us to determine a useful model.

Next, we want to look at the correlation between some of the variables.

```
df %>% dplyr::select(.,u_g, g_r,r_i, i_z,z) %>% cor(.) %>% corrplot(.,method="number")
```



It seems like most of the color variables are moderately if not strongly correlated to each other. Generally, the correlation coefficients seem to be at least above 0.3, indicating some level of correlation between most if not all the variables.

Principal Component Analysis

After we have filtered out the missing values, looked at the correlation between the predictors, and visualized the different distributions of the variables, we will be using Principal Component Analysis, or PCA, using a dataset with only the predictor variables.

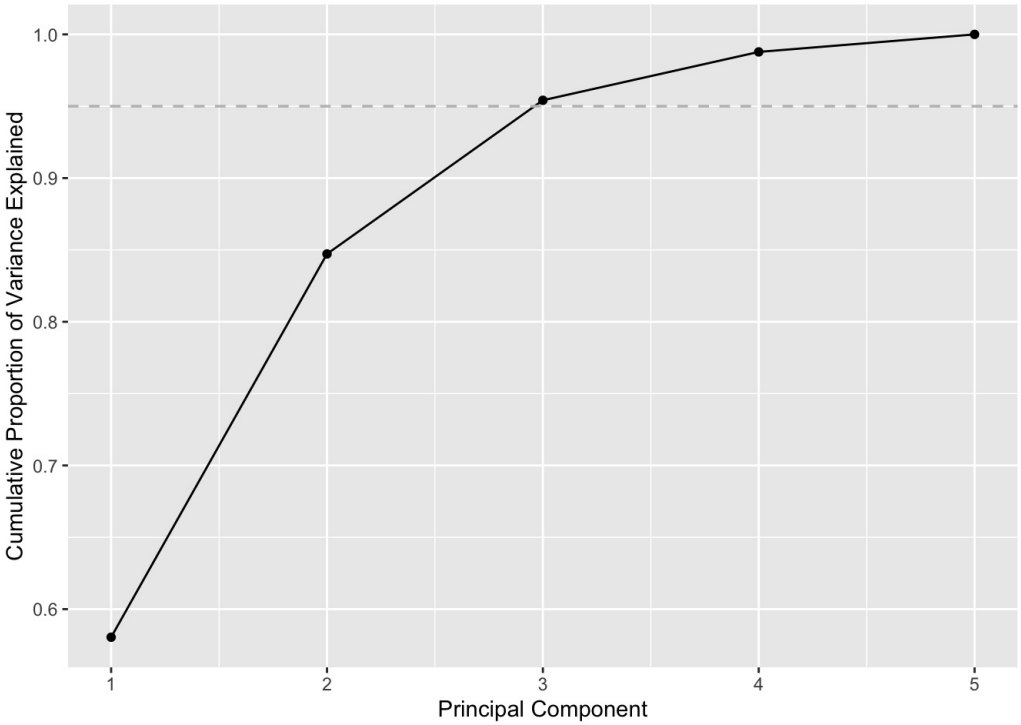
```
df.no.lab = subset(df, select = -label)
pca.out = prcomp(df.no.lab, scale=TRUE, retx = TRUE, center = TRUE, tol = NULL)

v = pca.out$sdev^2
pve = v/sum(v)
cpve = round(cumsum(v/sum(v)),3)
cpve
```

```
## [1] 0.580 0.847 0.954 0.988 1.000
```

```
pr_var = data.frame(varExplained = pca.out$sdev^2)
pr_var = pr_var %>% mutate(pve = varExplained / sum(varExplained))
```

We can see that the first principal component explains 58% of the variance in the data, the next PC explains 84.7% of the variance, and so on and so forth. We plot the cumulative PVE as follows:

```
ggplot(pr_var, aes(as.numeric(row.names(pr_var)), cumsum(pve))) +
  geom_point() + geom_line()+ geom_hline(yintercept=.95, linetype="dashed", color="grey", size=.6) +
  xlab("Principal Component") +
  ylab("Cumulative Proportion of Variance Explained")
```



Looking at the plot of the cumulative proportion of variance explained,it seems that around 3 PCs should be retained. If we were to adopt 3 PCs, we would be able to drop the last two principal components, which we can assume represents the random variation in the data, and still be able to reconstruct the input data with 95.4% of the overall variance “explained”. It appears overall, PCA would provide some level of reduced dimensionality.

Logistic Regression

Now we move onto logistic regression, using a 70-30 data split for the training and testing sets.

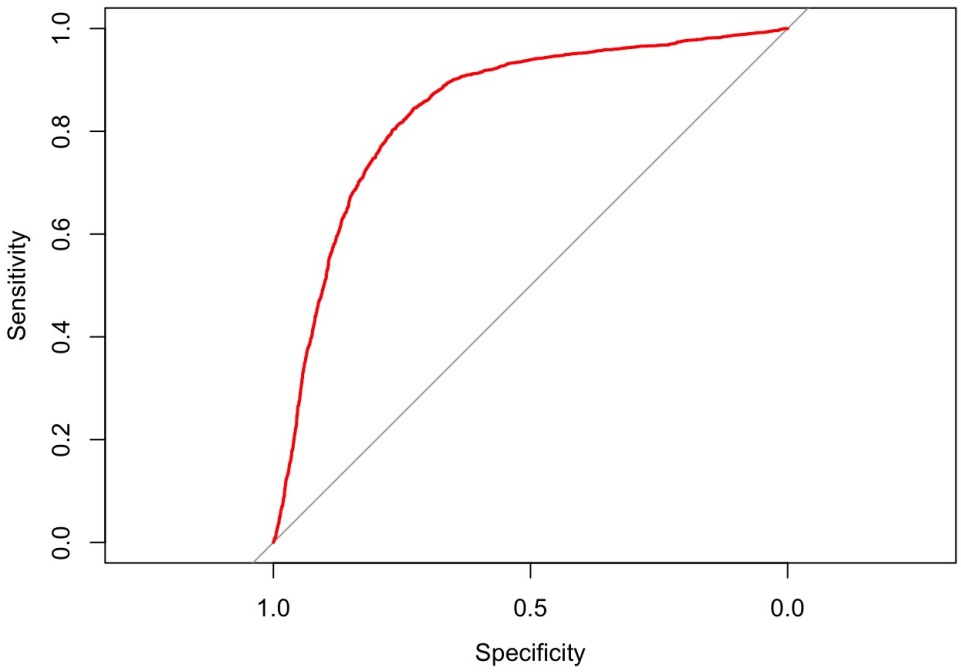
```
predictors = df.no.lab
response   = df$label

#split data, logistic regression model learned on split data with test-set assessment
set.seed(100)
fraction=.7
sp = sample(nrow(predictors), round(fraction*nrow(predictors)))
pred.train = predictors[sp ,]
pred.test = predictors[-sp ,]
resp.train = response[sp]
resp.test = response[-sp]


contrasts(df$label)
```

```
##          AGN
## STARFORM   0
## AGN        1
```

```
#logistic regression model using training data
glm.fit = glm(resp.train~.,data=pred.train,family="binomial")
summary(glm.fit)
```

```
##
## Call:
## glm(formula = resp.train ~ ., family = "binomial", data = pred.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.4233  -0.7586  -0.2770   0.7953   3.2818
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.47969    0.24393 -18.365  < 2e-16 ***
## z           -0.20364    0.41034  -0.496   0.6197
## u_g          0.13718    0.06552   2.094   0.0363 *
## g_r          2.75285    0.11068  24.873  < 2e-16 ***
## r_i          1.40049    0.20462   6.844 7.69e-12 ***
## i_z          1.57033    0.21557   7.284 3.23e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 27186  on 19705  degrees of freedom
## Residual deviance: 19406  on 19700  degrees of freedom
## AIC: 19418
##
## Number of Fisher Scoring iterations: 5
```

```
glm.prob=predict(glm.fit,newdata=pred.test,type="response")

glm.roc = suppressMessages(roc(resp.test,glm.prob))

plot(glm.roc,col="red",xlim=c(1,0),ylim=c(0,1))
```




```
cat("AUC for logistic regression: ",glm.roc$auc,"\n")
```

```
## AUC for logistic regression:  0.8405296
```

Below we calculate the VIF values of each variable in the model to assess whether or not there is any multicollinearity. It appears that z and r_i in particular have VIF values greater than 5, and it seems that there is some presence of multicollinearity but not too much. This is consistent with the results from PCA, where some level of dimensionality could be dropped but not by a lot. Since our objective is prediction, we are not removing variables with high VIF.

```
car::vif(glm.fit)
```

```
##        z      u_g      g_r      r_i      i_z
## 5.409062 2.203268 4.259079 5.780827 2.801098
```

Best Subset Selection

```
df.train = cbind(pred.train,resp.train)
df.test = cbind(pred.test,resp.test)

names(df.train)[names(df.train) == "resp.train"] <- "y"
names(df.test)[names(df.test) == "resp.test"] <- "y"


bg.outAIC= suppressMessages(bestglm(df.train,family=binomial, IC="AIC"))
```

```
bg.outAIC$BestModel
```

```
##
## Call:  glm(formula = y ~ ., family = family, data = Xi, weights = weights)
##
## Coefficients:
## (Intercept)          u_g          g_r          r_i          i_z
##     -4.5950       0.1335       2.7871       1.3115       1.6409
##
## Degrees of Freedom: 19705 Total (i.e. Null);  19701 Residual
## Null Deviance:       27190
## Residual Deviance: 19410     AIC: 19420
```

For the AIC model, we retain four of the five predictor variables, dropping z.

```
AIC.resp.prob = predict(bg.outAIC$BestModel, newdata=df.test, type="response")

AIC.resp.pred = ifelse(AIC.resp.prob>0.5,"AGN", "STARFORM")


mcr.AIC = mean(AIC.resp.pred!=resp.test)
mcr.AIC
```

```
## [1] 0.2178804
```

```
mean(AIC.resp.pred==resp.test)
```

```
## [1] 0.7821196
```

```
table(AIC.resp.pred, resp.test)
```

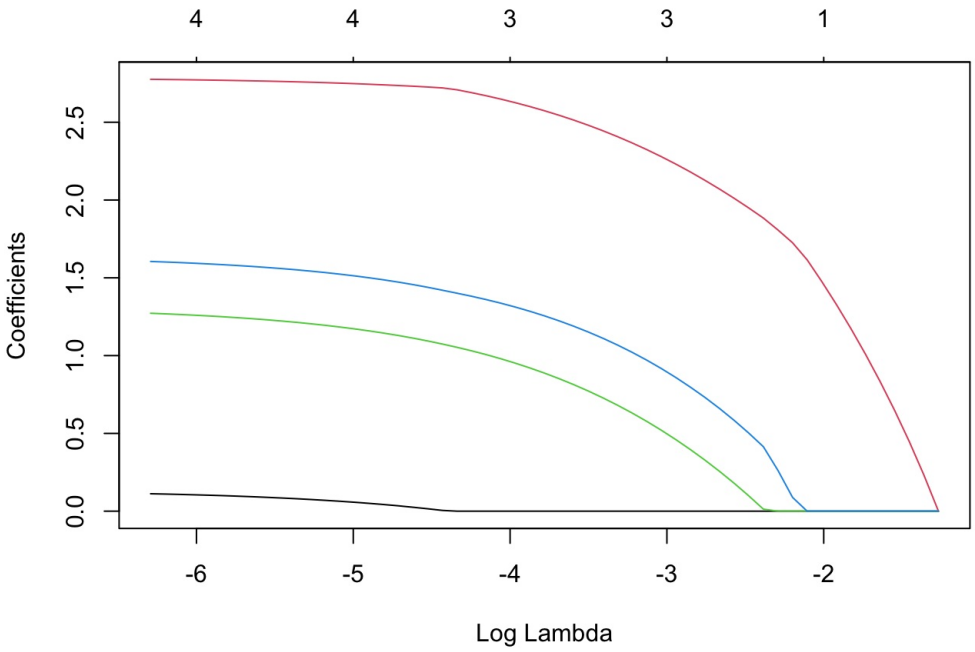
```
##              resp.test
## AIC.resp.pred STARFORM  AGN
##      AGN          1029 2988
##      STARFORM     3617  811
```

AIC misclassification rate is 0.2178804.

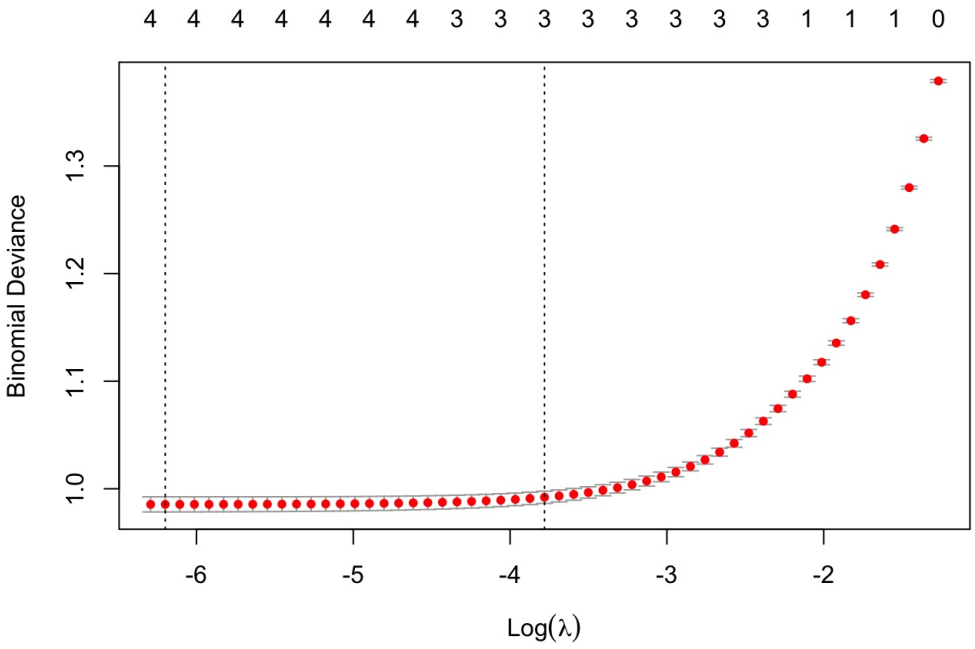
Lasso and Ridge Regression

We attempted to see if lasso and ridge regression was able to be used in this case.

```
#lasso
x = model.matrix(resp.train~.,pred.train)[,-1] ; y = resp.train
out.lasso = glmnet(x,y,alpha=1,family = "binomial")
plot(out.lasso,xvar="lambda")
```




```
set.seed(100)
cv.lasso = cv.glmnet(x,y,alpha=1,family = "binomial")
plot(cv.lasso)
```




```
cv.lasso$lambda.min ; log(cv.lasso$lambda.min)
```

```
## [1] 0.002031058
```

```
## [1] -6.199198
```

```
coef(out.lasso, cv.lasso$lambda.min)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) -4.5210540
## z            .
## u_g          0.1099564
## g_r          2.7749885
## r_i          1.2684566
## i_z          1.6017140
```

```
x.test    = model.matrix(resp.test~.,pred.test)[,-1]
resp.prob = predict(out.lasso,s=cv.lasso$lambda.min,newx=x.test,type="response")
resp.pred = ifelse(resp.prob>0.5,"AGN","STARFORM")
mean(resp.pred!=resp.test) # basically the same as for logistic regression
```

```
## [1] 0.2171699
```

For lasso regression, given optimal value of lambda as 0.002031058, four variables were retained, with no z variable. The missclassification rate is 0.2171699.

```
#ridge regresssion
out.ridge = glmnet(x,y,alpha=0,family = "binomial")

set.seed(100)
cv.ridge = cv.glmnet(x,y,alpha=0,family = "binomial")

cv.ridge$lambda.min ; log(cv.ridge$lambda.min)
```

```
## [1] 0.02812785
```

```
## [1] -3.570995
```

```
coef(out.ridge, cv.ridge$lambda.min)
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) -3.2856800
## z           -1.2121559
## u_g          0.5181058
## g_r          1.7346167
## r_i          1.6313366
## i_z          1.7146662
```

For ridge regression, given optimal value of lambda as 0.02812785, all five variables were retained, suggesting that ridge regression did not lead to improved results.

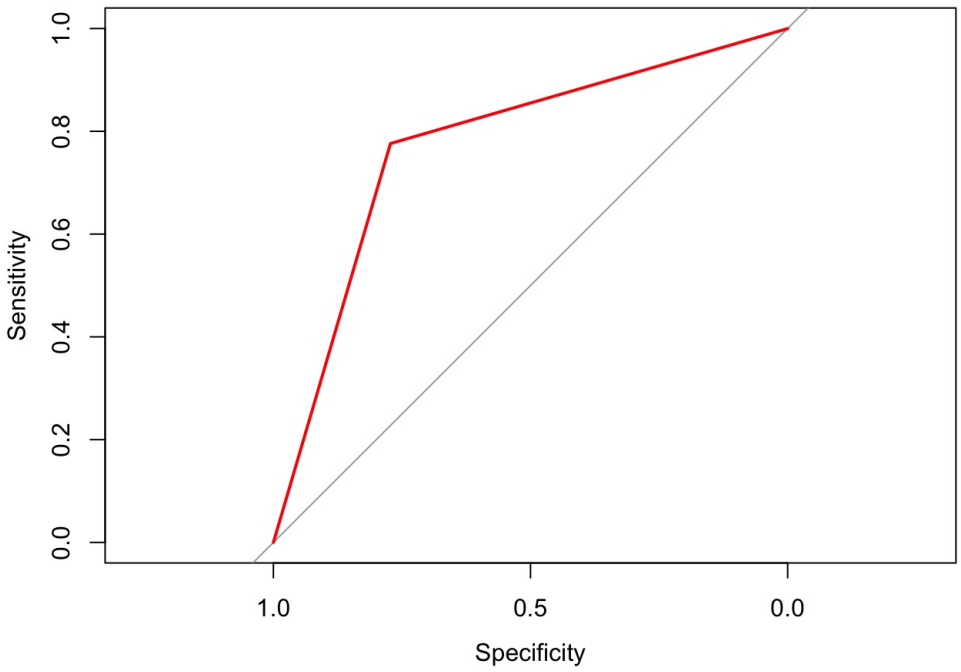
This may have been due to the sample sizes being relatively small compared to the number of predictor variables in our data.

Machine Learning Models and KNN

Next, we will be working with various machine learning models (trees, RF, XGB, Naive Bayes) as well as KNN. Given the size of the dataset and the time necessary to run SVM, Support Vector Machine is not included in this section.

Trees

```
rpart.out =rpart(resp.train~.,data=pred.train)
tree.class.prob= predict(rpart.out,newdata=pred.test, type="prob")[,2]
roc.tree = suppressMessages(roc(resp.test,tree.class.prob))
plot(roc.tree,col="red",xlim=c(1,0),ylim=c(0,1))
```

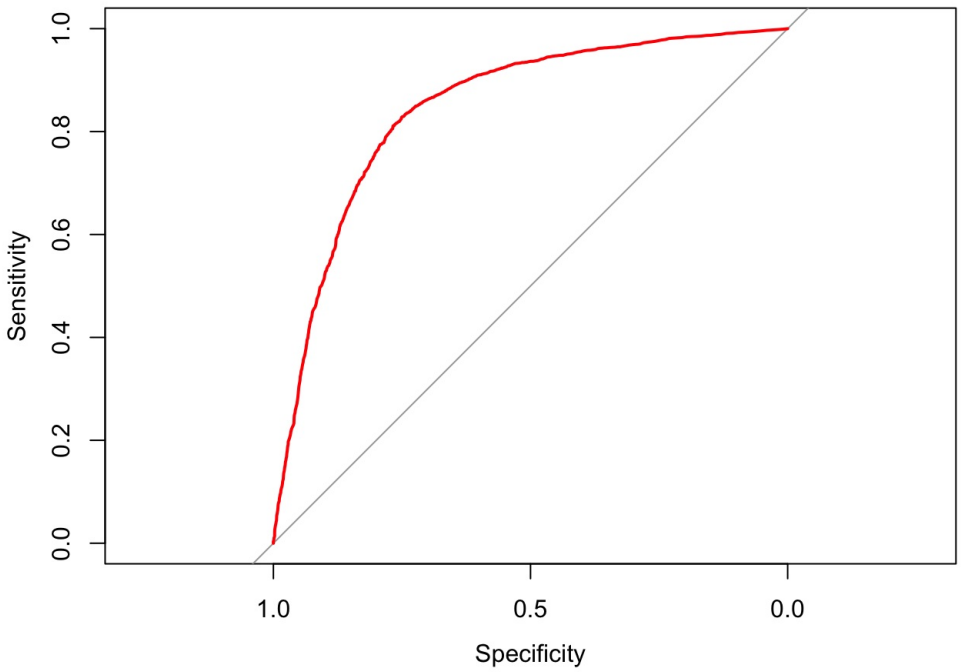



```
cat("AUC for tree: ",roc.tree$auc,"\n")
```

```
## AUC for tree:  0.7742671
```

Random Forest

```
set.seed(100)
rf.out = randomForest(resp.train~.,data=pred.train,importance=TRUE)
resp.prob.rf = predict(rf.out,newdata=pred.test,type="prob")[,2]
roc.rf = suppressMessages(roc(resp.test,resp.prob.rf))
plot(roc.rf,col="red",xlim=c(1,0),ylim=c(0,1))
```




```
cat("AUC for random forest: ",roc.rf$auc,"\n")
```

```
## AUC for random forest:  0.8451971
```

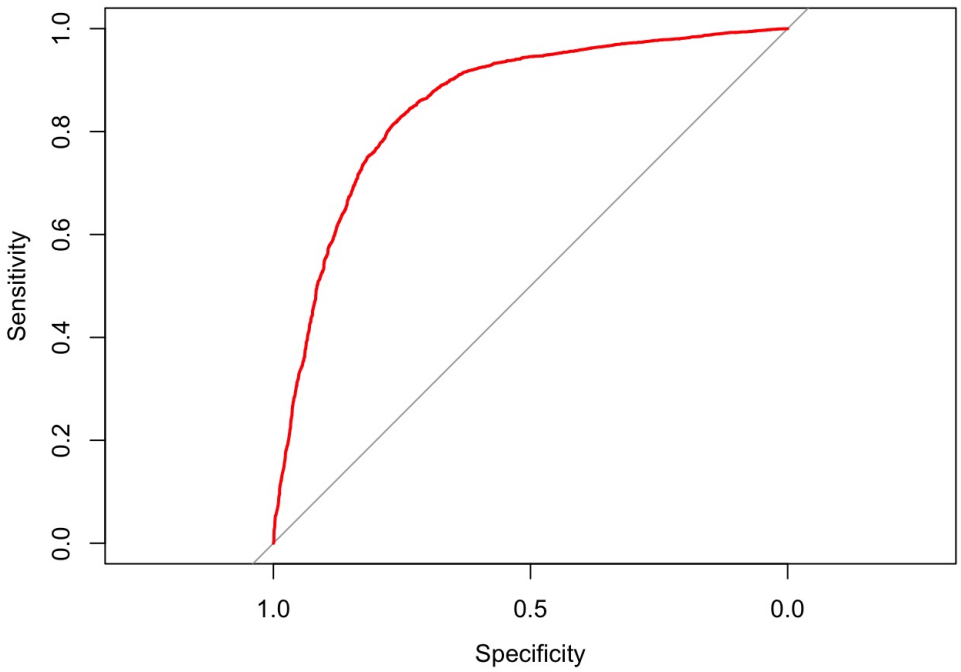
XGB

Since `xgboost` wants integer class labels, we map “STARFORM” to 0 and “AGN” to 1.

```
w = which(response!="STARFORM")
response.new = rep(0,length(response))
response.new[w] = 1
response.int = response.new

set.seed(100)
fraction=.7
sp = sample(nrow(predictors), round(fraction*nrow(predictors)))
pred.train = predictors[sp ,]
pred.test = predictors[-sp ,]
resp.train.int = response.int[sp]
resp.test.int = response.int[-sp]


train = xgb.DMatrix(data=as.matrix(pred.train),label=resp.train.int)
test = xgb.DMatrix(data=as.matrix(pred.test),label=resp.test.int)

set.seed(101)
xgb.cv.out = xgb.cv(params=list(objective="binary:logistic"),train,nrounds=30,nfold=5,verbose=0,eval_metric="erro
r")
cat("The optimal number of trees is", which.min(xgb.cv.out$evaluation_log$test_error_mean))
```

```
## The optimal number of trees is 15
```

```
xgb.out = xgboost(train,nrounds=which.min(xgb.cv.out$evaluation_log$test_error_mean),params=list(objective="binar
y:logistic"),verbose=0,eval_metric="error")

resp.predxg = predict(xgb.out,newdata=test,probability=TRUE)

roc.xgb = suppressMessages(roc(resp.test,resp.predxg))


plot(roc.xgb,col="red",xlim=c(1,0),ylim=c(0,1))
```

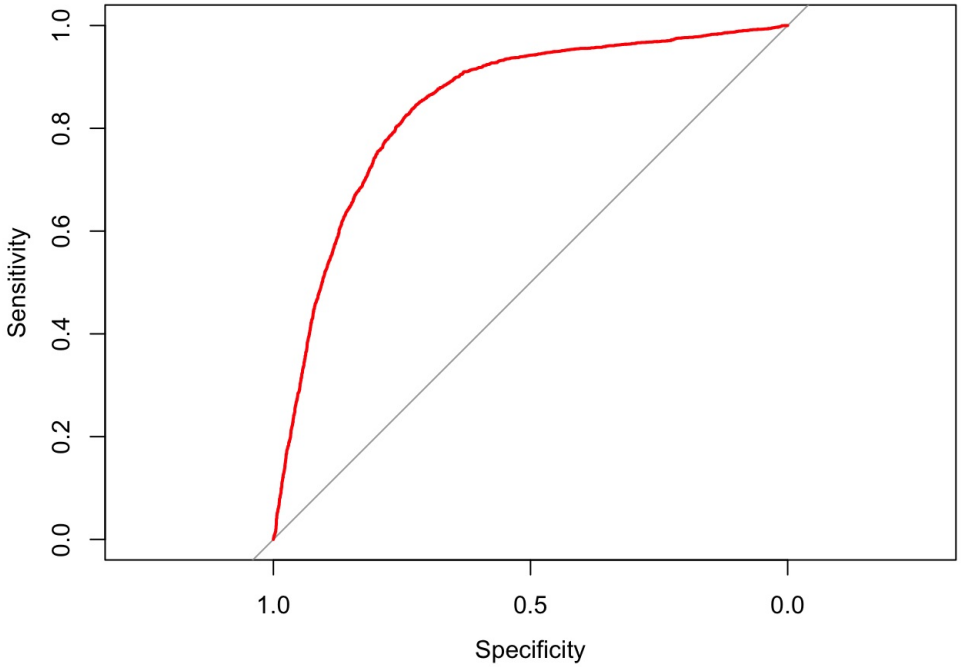



```
cat("AUC for xgb: ",roc.xgb$auc,"\n")
```

```
## AUC for xgb:  0.8524555
```

Naive Bayes

```
nb.out = naiveBayes(resp.train~.,data=pred.train)
nb.prob = predict(nb.out,newdata=pred.test,type="raw")[,2]
roc.nb = suppressMessages(roc(resp.test,nb.prob))
plot(roc.nb,col="red",xlim=c(1,0),ylim=c(0,1))
```




```
cat("AUC for naive bayes: ",roc.nb$auc,"\n")
```

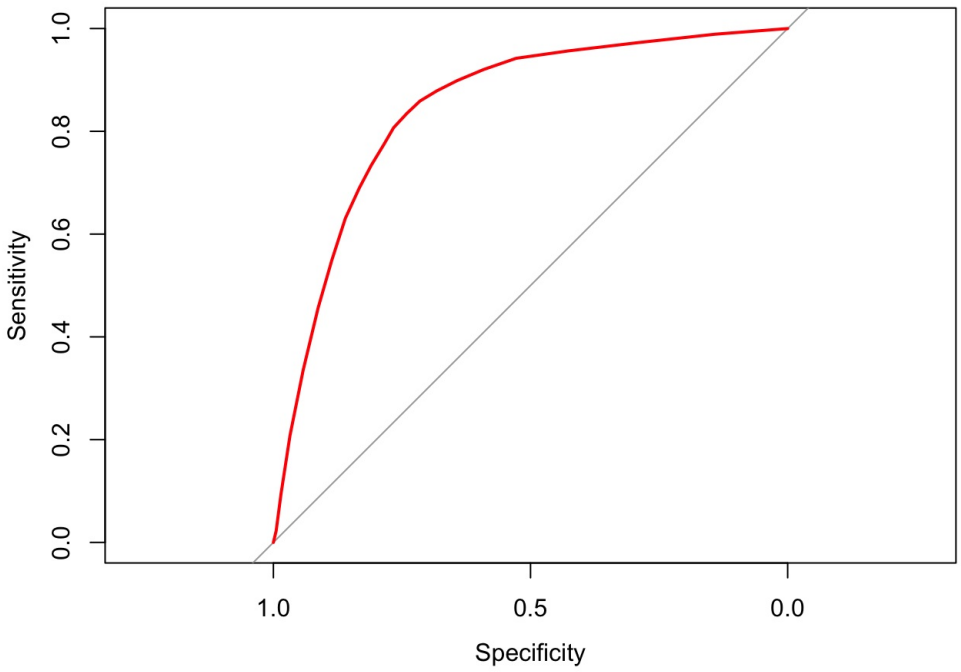
```
## AUC for naive bayes:  0.841971
```

KNN

```
k.max = 20
mse.k = rep(NA,k.max)
for ( kk in 1:k.max ) {
  knn.out = knn.cv(train=pred.train,cl=resp.train,k=kk,algorithm="brute")
  mse.k[kk] = mean(knn.out != resp.train)
}
k.min = which.min(mse.k)
cat("The optimal number of nearest neighbors is ",k.min,"\n")
```

```
## The optimal number of nearest neighbors is  20
```

```
knn.out = knn(train=pred.train,test=pred.test,cl=resp.train,k=k.min,prob=TRUE, algorithm="brute")
knn.prob = attributes(knn.out)$prob
w = which(knn.out=="STARFORM")
knn.prob[w] = 1 - knn.prob[w]

roc.knn = suppressMessages(roc(resp.test,knn.prob))
plot(roc.knn,col="red",xlim=c(1,0),ylim=c(0,1))
```




```
cat("AUC for knn: ",roc.knn$auc,"\n")
```

```
## AUC for knn:  0.8449885
```

Conclusion

Model	AUC and MCR values
Decision Tree	AUC: 0.774
Random Forest	AUC: 0.845
XGBoost	AUC: 0.852
Naive Bayes	AUC: 0.842
KNN	AUC: 0.845
logistic regression	AUC: 0.841
AIC	MCR: 0.218

It appears that XGboost has the greatest AUC value of 0.8524555.

Now we use Youden's J to find the optimal threshold, make class predictions based on this threshold, and determine the MCR and confusion matrix for our optimal model using gradient boosting.

```
J = roc.xgb$sensitivities + roc.xgb$specificities - 1

w = which.max(J)
cat("Optimum threshold for xgb: ",roc.xgb$thresholds[w],"\n")
```

```
## Optimum threshold for xgb:  0.4643515
```

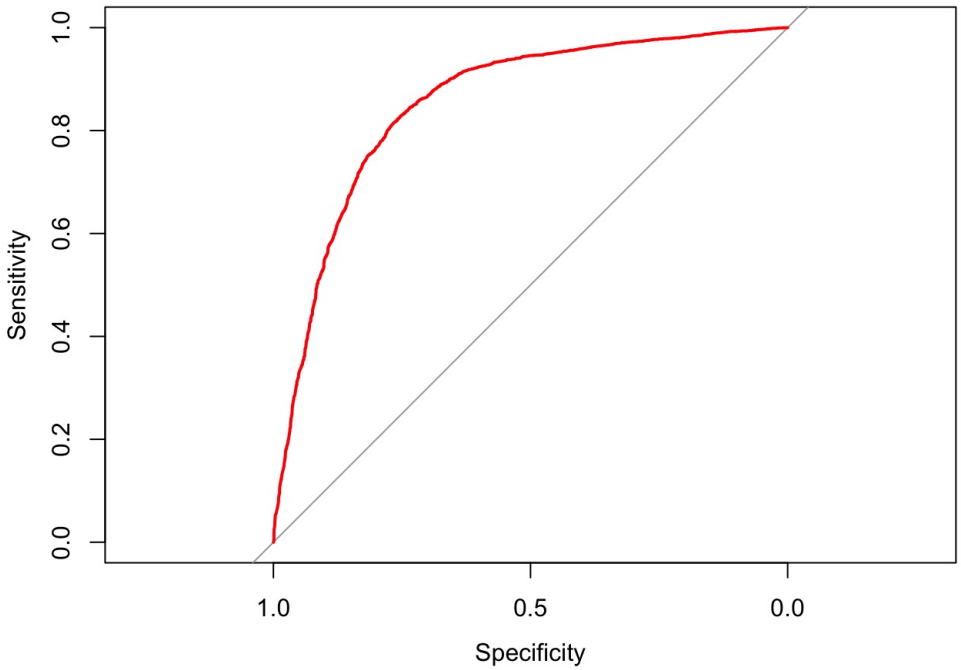
```
resp.predxg = predict(xgb.out,newdata=test,probability=TRUE)
resp.pred.xg = ifelse(resp.predxg>0.4643515, "AGN","STARFORM")

table(resp.pred.xg,resp.test)
```

```
##             resp.test
## resp.pred.xg STARFORM  AGN
##     AGN          1085 3096
##     STARFORM     3561  703
```

```
mean(resp.pred.xg!=resp.test)
```

```
## [1] 0.2117229
```

```
plot(roc.xgb,col="red",xlim=c(1,0),ylim=c(0,1))
```




```
cat("AUC for xgb: ",roc.xgb$auc,"\n")
```

```
## AUC for xgb:  0.8524555
```

The optimum threshold for xgb is 0.4643515 and the MCR is 0.2117229.

Using xgboost which we determined had the highest AUC out of all the other models that were tested, we were able to classify galaxies as either starform or having an active nucleus with a misclassification rate of 21.17%. Looking at the confusion matrix above, it appears that of the 8445 galaxies within our test set, we were able to correctly classify 3,561 galaxies as starform and 3,096 as having active nucleus.

References


```
Freeman, P.E. 2021, online at https://github.com/pefreeman/36-290/blob/master/PROJECT_DATASETS/ACTIVE_CLASS/README.md


James, G., et al. 2013, An Introduction to Statistical Learning, Springer


Zhang, K., et al. 2019, "Machine Learning Classifiers for Intermediate Redshift Emission Line Galaxies", The Astrophysical Journal, online at arxiv.org/pdf/1908.07046.pdf


Grolemund, Hadley Wickham and Garrett. “7 Exploratory Data Analysis | R for Data Science”, online at https://r4ds.had.co.nz/exploratory-data-analysis.html
```