FERMI Data Analysis Report

Cindy Xiong (36-290)

Fall 2021

Contents

Introduction

The Fermi Gamma-Ray Space Telescope is used to detect high-energy photons produced by astronomical objects, among which are gamma-ray-emitting BL Lacertae objects or BL Lacs. BL Lacs have beamed jets of matter pointed directly towards the Earth, and this is worthy of noting because jets usually stream from black holes. However, BL Lacs are known for being difficult to identify due to their spectra's lack of diagnostic features, since a spectrum shows the relative amount of light emitted by an object at different energies.

Thus, the goal of this analysis is to effectively classify BL Lacs using a dataset derived from Fermi LAT 10-Year Point Source Catalog (4FGL) (Ajello et al., 2020)**, as well as determine the predictive variables most useful for said classification.**

Data

Univariate Exploration

Below, univariate data analysis is performed on the data by summarizing and creating histograms for the quantitative variables.

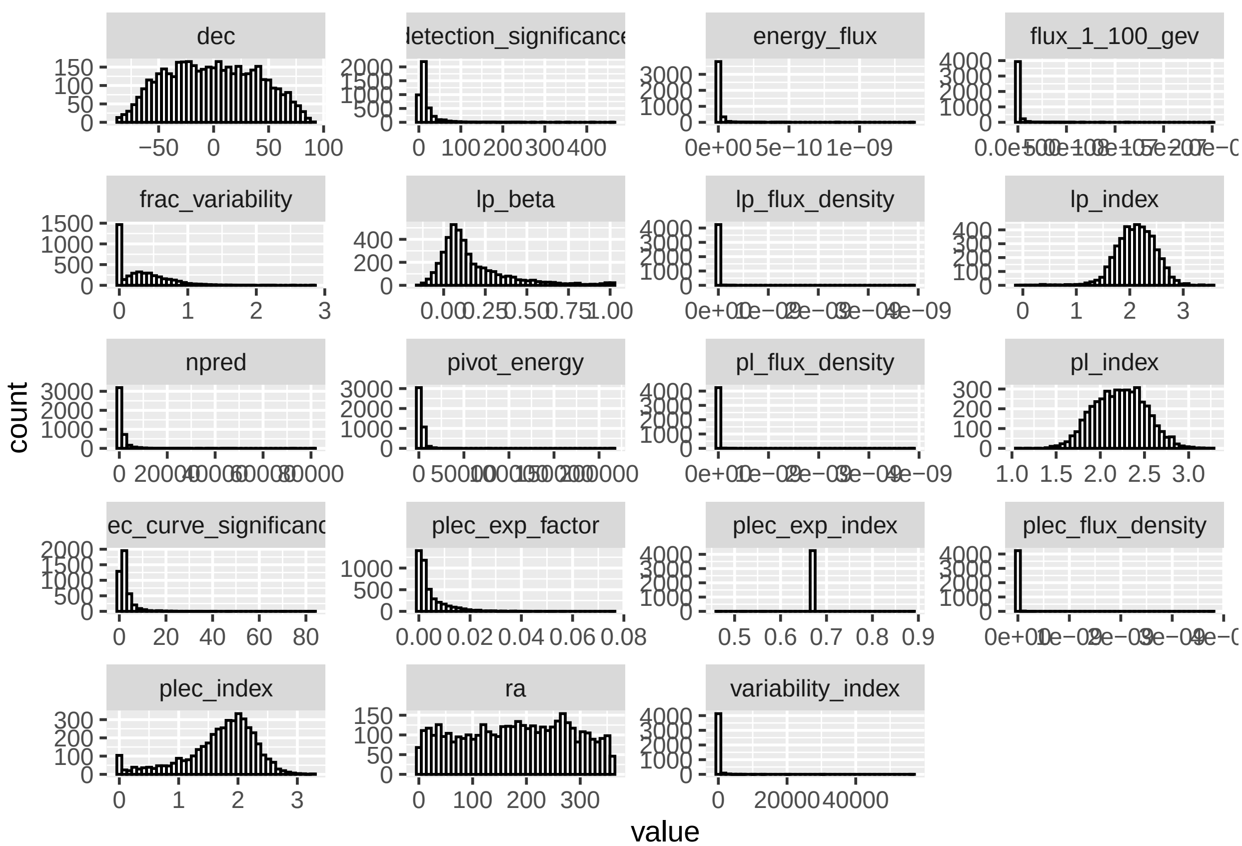
```
df_quant_pred <- dplyr::select(df, -spectrum_type, -source_type)
summary(df_quant_pred)
```

```
##       ra                dec           flux_1_100_gev
##  Min.   :  0.0983   Min.   :-87.2847   Min.   :3.209e-11
##  1st Qu.: 95.0698   1st Qu.:-32.5792   1st Qu.:1.669e-10
##  Median :183.9696   Median : -1.5308   Median :2.984e-10
##  Mean   :180.0451   Mean   : -0.7731   Mean   :1.152e-09
##  3rd Qu.:265.4746   3rd Qu.: 32.1567   3rd Qu.:6.969e-10
##  Max.   :359.9817   Max.   : 88.7375   Max.   :1.993e-07
##  detection_significance  pivot_energy       energy_flux
##  Min.   :  4.058        Min.   :   226.6   Min.   :5.707e-13
##  1st Qu.:  6.091        1st Qu.:  1026.7   1st Qu.:2.067e-12
##  Median :  9.633        Median :  1725.3   Median :3.587e-12
##  Mean   : 17.733        Mean   :  2586.0   Mean   :1.097e-11
##  3rd Qu.: 18.044        3rd Qu.:  3014.2   3rd Qu.:7.446e-12
##  Max.   :465.154        Max.   :215093.7   Max.   :1.372e-09
##  pl_flux_density        pl_index     lp_flux_density        lp_index
##  Min.   :0.000e+00   Min.   :1.050   Min.   :0.000e+00   Min.   :-0.0838
##  1st Qu.:2.800e-14   1st Qu.:2.005   1st Qu.:3.200e-14   1st Qu.: 1.8856
##  Median :1.430e-13   Median :2.224   Median :1.690e-13   Median : 2.1224
##  Mean   :2.662e-12   Mean   :2.220   Mean   :2.933e-12   Mean   : 2.1115
##  3rd Qu.:7.000e-13   3rd Qu.:2.426   3rd Qu.:8.670e-13   3rd Qu.: 2.3644
##  Max.   :3.859e-09   Max.   :3.241   Max.   :3.875e-09   Max.   : 3.5371
##     lp_beta        plec_flux_density     plec_index    plec_exp_factor
##  Min.   :-0.1631   Min.   :0.000e+00   Min.   :0.000   Min.   :-0.000910
##  1st Qu.: 0.0438   1st Qu.:3.100e-14   1st Qu.:1.419   1st Qu.: 0.000620
##  Median : 0.1074   Median :1.650e-13   Median :1.822   Median : 0.001920
##  Mean   : 0.1733   Mean   :2.850e-12   Mean   :1.698   Mean   : 0.004253
##  3rd Qu.: 0.2435   3rd Qu.:8.380e-13   3rd Qu.:2.106   3rd Qu.: 0.005525
##  Max.   : 1.0000   Max.   :3.763e-09   Max.   :3.263   Max.   : 0.074630
##  plec_exp_index   plec_curve_significance     npred          variability_index
##  Min.   :0.4602   Min.   : 0.000          Min.   :   20.23   Min.   :    0.44
```

```
##  1st Qu.:0.6667   1st Qu.: 0.890         1st Qu.:  276.20   1st Qu.:    8.52
##  Median :0.6667   Median : 1.780         Median :  521.31   Median :   13.53
##  Mean   :0.6666   Mean   : 2.755         Mean   : 1273.81   Mean   :  127.47
##  3rd Qu.:0.6667   3rd Qu.: 3.100         3rd Qu.: 1039.39   3rd Qu.:   29.53
##  Max.   :0.8816   Max.   :83.000         Max.   :80821.95   Max.   :56365.37
##  frac_variability
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.2557
##  Mean   :0.3247
##  3rd Qu.:0.5211
##  Max.   :2.8268
```

There are 4283 data in total, 1226 (28.625%) of which are `BLL` and 3057 (71.375%) of which are `NOT_BLL`. Many variables seem to have outliers at the higher end, because the differences between their maximums and medians are much higher than the differences between their minimums and medians. Most of `plec_exp_index` have the same values from the six-number summary, so it may not be an useful predictor variable.

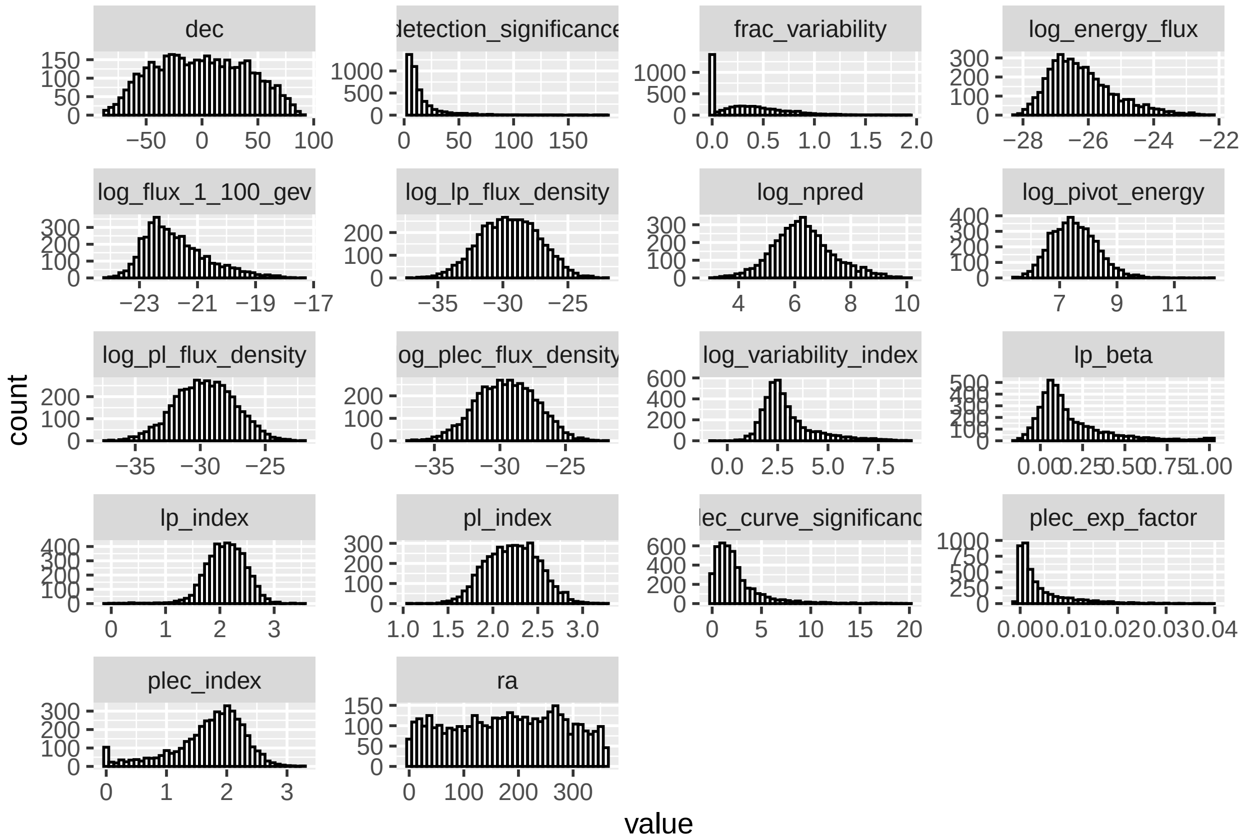
```r
ggplot(data=gather(df_quant_pred),mapping=aes(x=value)) +
  geom_histogram(color='black',
                 fill='white',
                 bins=40) +
  facet_wrap(~key,scales='free',ncol=4)
```



From the histograms, we can see that 12 out of the 19 quantitative variables are heavily skewed to the right, with just a single bar extending from the leftmost part of the plot. Additionally, `lp_beta` is also right skewed, but not as much. This implies that transformations will later have to be performed on these variables to achieve a more symmetric distribution. All of the `flux` variables also have single outliers that cause their distributions to become significantly more skewed, which will be removed in further analysis. Other than that, most of the distributions are unimodal, and `plec_index`, `pl_index`, `lp_index`, and `dec` are the quantitative variables with the most normal distributions.

The necessary right-skewed variables are log-transformed and then their histograms are outputted below. The log transformations replaced the originals in the changed dataframe and `plec_exp_index` was also removed, since it is not a helpful predictor. Additionally, `plec_curve_significance`, `plec_exp_factor`, `frac_variability`, and `detection_significance` were filtered in order to reduce outliers on the higher end.

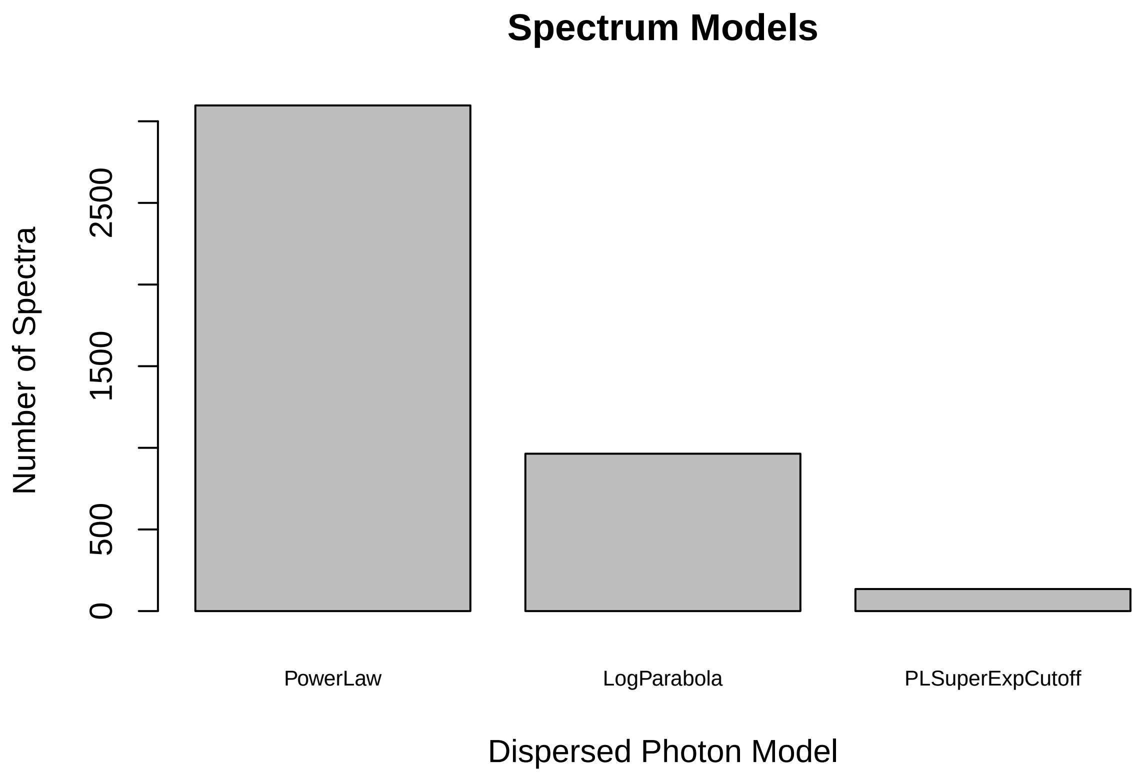
```r
df = filter(df, plec_curve_significance < 20,
            plec_exp_factor < 0.04,
            frac_variability < 2,
            detection_significance < 200)
df_changed = df

df_changed$log_flux_1_100_gev = log(df$flux_1_100_gev)
df_changed$log_energy_flux = log(df$energy_flux)
df_changed$log_lp_flux_density = log(df$lp_flux_density)
df_changed$log_npred = log(df$npred)
df_changed$log_pivot_energy = log(df$pivot_energy)
df_changed$log_pl_flux_density = log(df$pl_flux_density)
df_changed$log_plec_flux_density = log(df$plec_flux_density)
df_changed$log_variability_index = log(df$variability_index)

df_changed %>% dplyr::select(., -flux_1_100_gev, -energy_flux,
                             -lp_flux_density, -npred, -pivot_energy,
                             -pl_flux_density, -plec_flux_density,
                             -variability_index,
                             -plec_exp_index) -> df_changed

df_changed %>% dplyr::select(., -spectrum_type, -source_type) -> df_quant_pred

ggplot(data=gather(df_quant_pred),mapping=aes(x=value)) +
  geom_histogram(color='black',
                 fill='white',
                 bins=40) +
  facet_wrap(~key,scales='free',ncol=4)
```



Most of these histograms look much more symmetric than the old ones for the variables before the log-transformations.

Now, we can move on to the categorical predictor variable, `spectrum_type`.

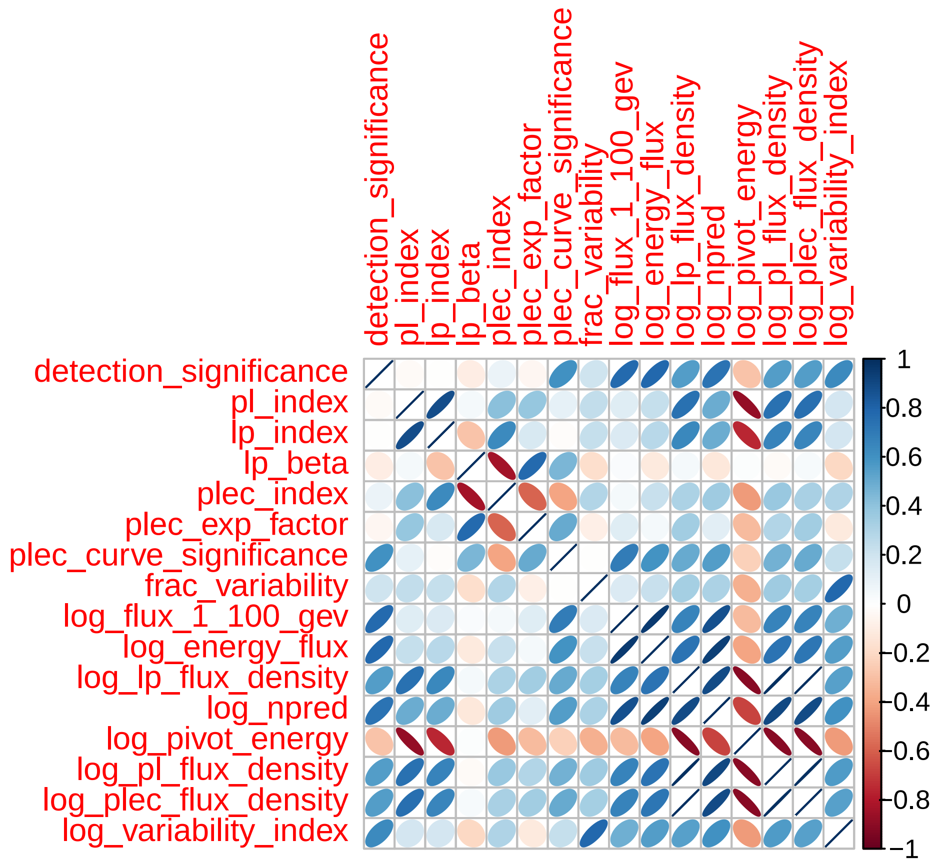
```
unique_spectrums <- unique(df$spectrum_type)
df$spectrum_type = (factor(df$spectrum_type, levels=unique_spectrums))
barplot(table(factor(df$spectrum_type, levels=unique_spectrums)),
        main = "Spectrum Models",
        xlab = "Dispersed Photon Model",
        ylab = "Number of Spectra",
        cex.names = 0.7)
```



From the bar plot, we can see that for the sole categorical predictor variable, `spectrum_type`, the PowerLaw model was the spectrum model used the most often, with LogParabola being the second most common, and PLSuperExpCutoff being used the least.

Bivariate Exploration

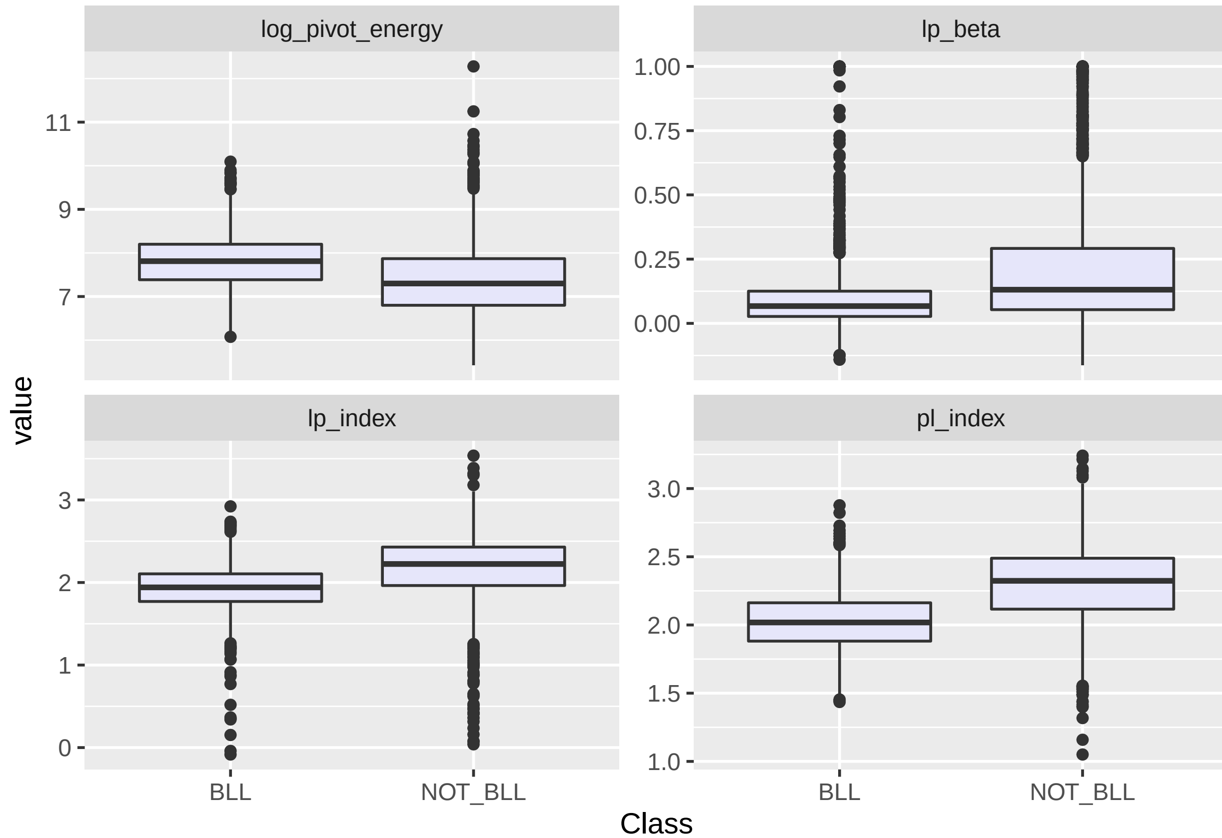
A correlation plot can be created for the quantitative predictor variables, which may later indicate signs of multicollinearity for those who are highly correlated with each other.



From this plot, the strongest correlations between predictor variables can be picked out. It seems that `log_energy_flux`, `log_flux_1_100_gev`, `detection_significance`, and `log_npred` all have strong positive correlations with each other, while `plec_index` and `lp_index`, as well as `log_pivot_energy` and `log_lp_flux density`, have strong negative correlations. Thus, the data exhibits multicollinearity.

Next, the predictor variables can be plotted with the response variable, `source_type`. Since `source_type` is categorical, this will be done with side-by-side boxplots. Below, a few of the boxplots that had more of a significant difference between `NOT_BLL` and `BLL` were plotted.

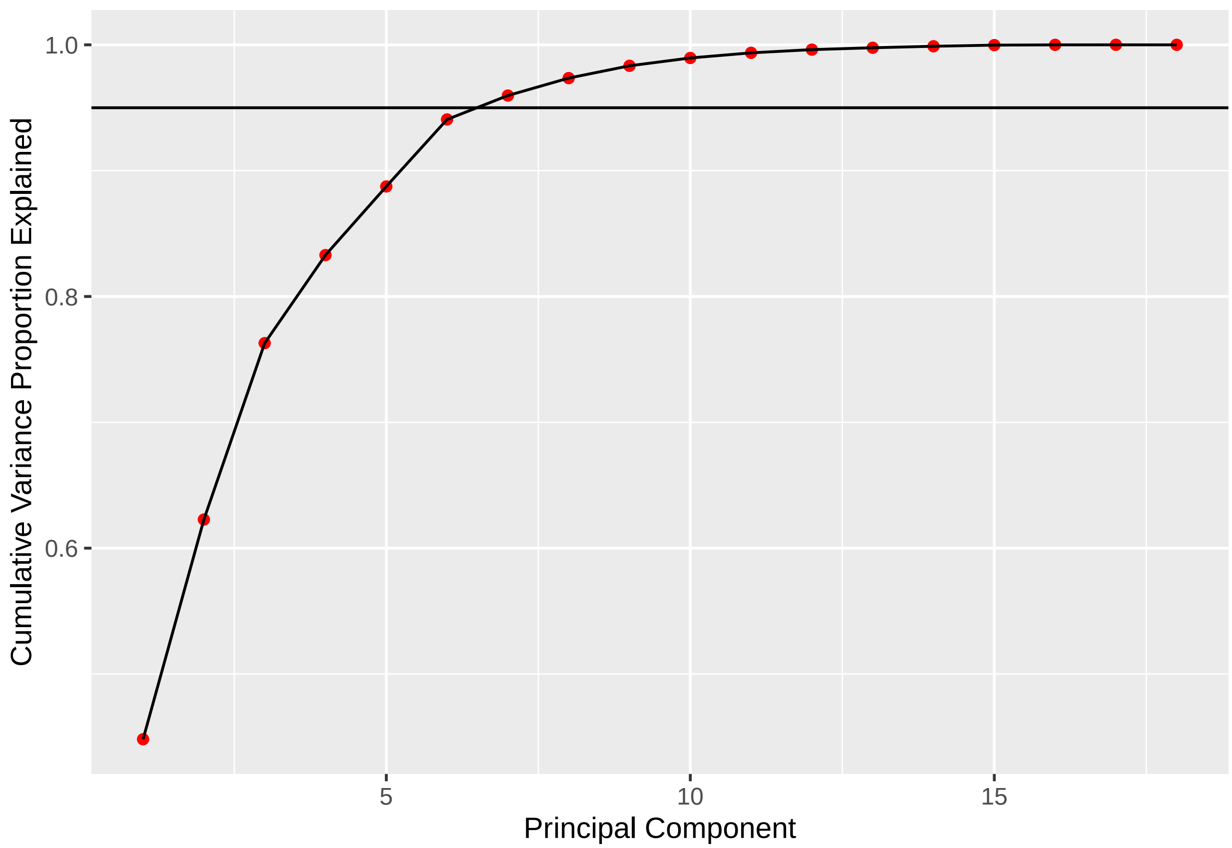
```
df_changed %>% dplyr::select(.,pl_index, lp_index, lp_beta, log_pivot_energy) %>%
  gather(.) -> df.new
ggplot(data=df.new,mapping=aes(x=rep(df$source_type,4),y=value)) +
  xlab("Class") + geom_boxplot(fill="lavender") +
  facet_wrap(~key,scale='free_y')
```



Principal Component Analysis (PCA)

After exploring the data, principal component analysis (PCA) can be performed in order to determine whether the data lies in a lower-dimensional subspace. There are 18 principal components in total, and we can graph the cumulative variance plot below.

```
pca.out = prcomp(x=df_quant_pred,scale=TRUE)
pca.var=pca.out$sdev^2
pve=pca.var/sum(pca.var)
pc = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18)
pov_df=data.frame(pc, cumsum(pve))
ggplot(data=pov_df, mapping=aes(x=pc, y = cumsum.pve.),
       ylim=c(0,1)) + xlab("Principal Component") +
      ylab("Cumulative Variance Proportion Explained") +
      geom_point(color="red") + geom_line() +
      geom_hline(yintercept=0.95)
```



According to the cumulative variance plot, it looks like about seven principal components are worth retaining to mitigate multicollinearity, because the proportion of the variance that each subsequent principal component explains seems to be quite little. However, since prediction is the goal of this analysis, this will not be done.

Best Model Selection

Seven models (Logistic Regression, Best Subset Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, K Nearest Neighbors, Naive Bayes) can be applied to see which one can best classify the data. All of the models are trained with 70% of the data, and tested on the remaining 30%. The ROC curves for each model will be plotted, and the model with the highest AUC will be used to classify the data and calculate the misclassification rate.

```
set.seed(101)
s = sample(nrow(df), nrow(df)*.7)
pred.train = df_quant_pred[s,]
pred.test = df_quant_pred[-s,]
resp.train = df_changed$source_type[s]
resp.test = df_changed$source_type[-s]
```

Logistic Regression

Since the response variable is categorical, a logistic regression model can be fitted to the data. The calculated AUC below is 0.8698.

```
glm.fit = glm(resp.train~., data=pred.train, family=binomial)
glm.probs = predict(glm.fit, newdata=pred.test, type="response")
roc.glm = suppressMessages(roc(resp.test,glm.probs))
```

Best Subset Logistic Regression

Forward-stepwise selection that assumes the use of AIC is applied to the data in order to select the best-subset of predictor variables. The best-subset selection ended up retaining the 12 following variables:

```
bss = log_forward(pred.train, resp.train)
print(bss)
```

```
##  [1] "dec"                       "detection_significance"
##  [3] "log_energy_flux"           "log_lp_flux_density"
##  [5] "log_npred"                 "log_pivot_energy"
##  [7] "log_plec_flux_density"     "log_variability_index"
##  [9] "lp_beta"                   "pl_index"
## [11] "plec_curve_significance" "ra"
```

The calculated AUC below is then 0.8690.

```
pred.bss.train = select(pred.train, all_of(bss))
pred.bss.test = select(pred.test, all_of(bss))
bss.fit = glm(resp.train~., data=pred.bss.train, family=binomial)
bss.probs = predict(bss.fit, newdata=pred.bss.test, type="response")
roc.bss = suppressMessages(roc(resp.test,bss.probs))
cat("AUC for AIC Logistic Regression Model: ",round(roc.bss$auc, 3),"\n")
```

```
## AUC for AIC Logistic Regression Model:  0.869
```

Decision Tree

The calculated AUC of the deicision tree calculated below is 0.7967.

```
set.seed(101)
rpart.out = rpart(resp.train~.,data=pred.train)
tree.pred = predict(rpart.out,newdata=pred.test,type="prob")[,2]  # probability of class 1
roc.tree = suppressMessages(roc(resp.test,tree.pred))
cat("AUC for decision tree: ",round(roc.tree$auc, 3),"\n")
```

```
## AUC for decision tree:  0.797
```

Random Forest

The calculated AUC of the random forest calculated below is 0.8702.

```
set.seed(101)
rf.out = randomForest(resp.train~.,data=pred.train)
rf.probs = predict(rf.out,newdata=pred.test,type="prob")[,2]
roc.rf = suppressMessages(roc(resp.test,rf.probs))
cat("AUC for random forest: ",round(roc.rf$auc, 3),"\n")
```

```
## AUC for random forest:  0.87
```

Gradient Boosting

The calculated AUC of the gradient boosting is 0.8546.

```
set.seed(101)
train = xgb.DMatrix(data=as.matrix(pred.train),label=resp.train)
test  = xgb.DMatrix(data=as.matrix(pred.test),label=resp.test)
set.seed(101)
xgb.cv.out = xgb.cv(params=list(objective="reg:squarederror"),train,
                    nrounds=30,nfold=5,verbose=0)
xgb.out = xgboost(train,nrounds=which.min(xgb.cv.out$evaluation_log$test_rmse_mean),
                  params=list(objective="reg:squarederror"),verbose=0)
xgb.pred    = predict(xgb.out,newdata=test)
roc.xgb = suppressMessages(roc(resp.test,xgb.pred))
cat("AUC for gradient boost: ",round(roc.xgb$auc, 3),"\n")
```

```
## AUC for gradient boost:  0.855
```

K Nearest Neighbors

The calculated AUC of the KNN is 0.6161. The optimal number of nearest neighbors is 7.

```
set.seed(101)
k.max = 50
mse.k = rep(NA,k.max)
for ( kk in 1:k.max ) {
  knn.cv.out = knn.cv(train=pred.train,cl=resp.train,k=kk,prob=TRUE)
  mse.k[kk] = mean(knn.cv.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  7
```

```
knn.out = knn(train=pred.train, test=pred.test, cl=resp.train, prob=TRUE)
knn.prob = attributes(knn.out)$prob
w = which(knn.out=="BLL") # insert name of Class 0 here
knn.prob[w] = 1 - knn.prob[w]   # knn.prob is now the Class 1 probability!
roc.knn = suppressMessages(roc(resp.test,knn.prob))
cat("AUC for KNN: ",round(roc.knn$auc, 3),"\n")
```

```
## AUC for KNN:  0.616
```

Naive Bayes

The calculated AUC of the Naive Bayes is 0.8059.

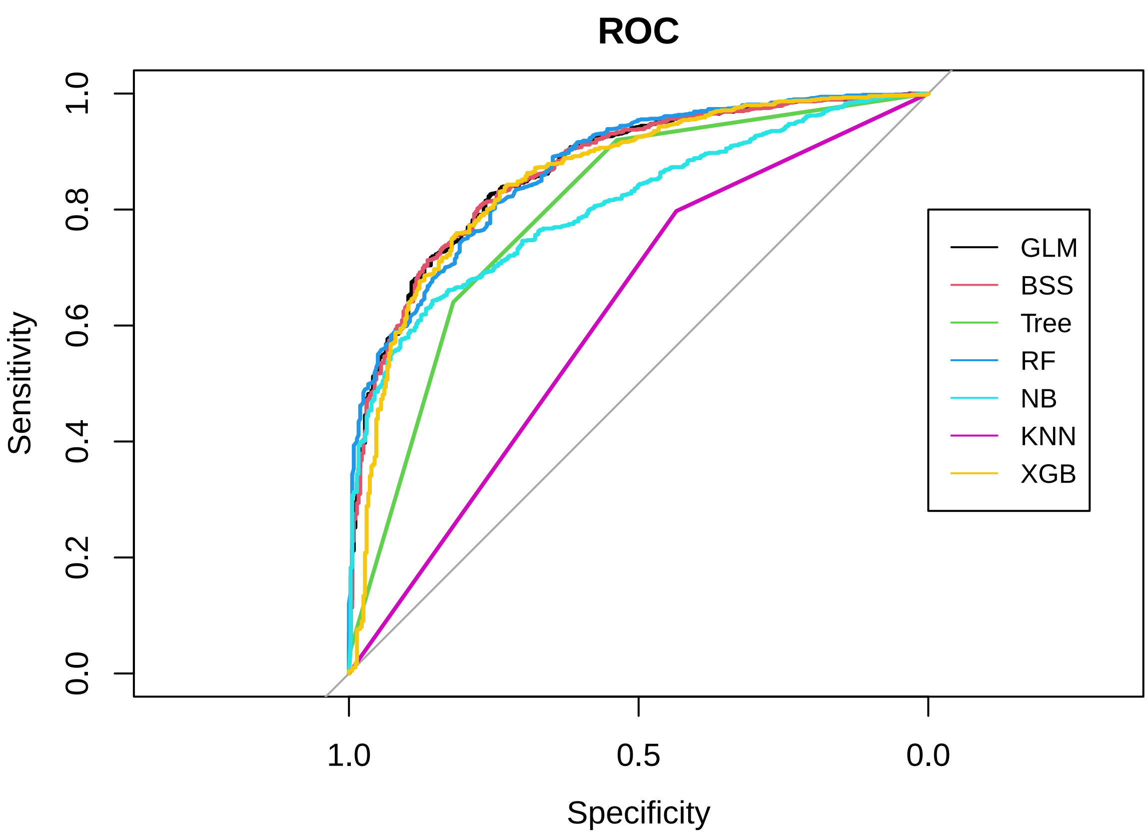
```
set.seed(101)
nb.out = naiveBayes(resp.train~.,data=pred.train)
nb.prob = predict(nb.out,newdata=pred.test,type="raw")[,2]
roc.nb = suppressMessages(roc(resp.test,nb.prob))
cat("AUC for Naive Bayes: ",round(roc.nb$auc, 3),"\n")
```

```
## AUC for Naive Bayes:  0.806
```

Final Selection

The plot below combines all of the ROC curves of the seven models.

```
plot(roc.glm,col=1,xlim=c(1,0),ylim=c(0,1), main="ROC")
plot(roc.bss,col=2,xlim=c(1,0),ylim=c(0,1), add = TRUE)
plot(roc.tree,col=3,xlim=c(1,0),ylim=c(0,1), add = TRUE)
plot(roc.rf,col=4,xlim=c(1,0),ylim=c(0,1), add = TRUE)
plot(roc.nb,col=5,xlim=c(1,0),ylim=c(0,1), add = TRUE)
plot(roc.knn,col=6,xlim=c(1,0),ylim=c(0,1), add = TRUE)
plot(roc.xgb,col=7,xlim=c(1,0),ylim=c(0,1), add = TRUE)
legend(0, 0.8, legend=c("GLM", "BSS", "Tree", "RF", "NB", "KNN", "XGB"),
       col=c(1, 2, 3, 4, 5, 6, 7), lty=1, cex=0.8)
```



The table below summarizes the AUC values of each of the seven models. From this, it is evident that the random forest model has the highest AUC, although by a small margin in comparison to logistic regression.

```r
data.frame(Model = c('Logistic Regression', 'BSS Logistic Regression',
                     'Decision Tree', 'Random Forest', 'XGB',
                     'KNN', 'Naive Bayes'),
           AUC = c(round(roc.glm$auc, 3), round(roc.bss$auc, 3),
                   round(roc.tree$auc, 3), round(roc.rf$auc, 3),
                   round(roc.xgb$auc, 3), round(roc.knn$auc, 3),
                   round(roc.nb$auc, 3)))
```

```
##                     Model   AUC
## 1     Logistic Regression 0.870
## 2 BSS Logistic Regression 0.869
## 3           Decision Tree 0.797
## 4           Random Forest 0.870
## 5                     XGB 0.855
## 6                     KNN 0.616
## 7             Naive Bayes 0.806
```

Classification

Now that the random forest has been determined to be the most suitable model, it can be used to classify the data. The table below shows how the random forest model classifies the data into "BLL" and "NOT_BLL." The misclassification rate is 20.8%.

```r
J = roc.rf$sensitivities + roc.rf$specificities - 1
w = which.max(J)
threshold = roc.rf$thresholds[w]
cat("Optimum threshold for Random Forest: ",roc.rf$thresholds[w],"\n")
```

```
## Optimum threshold for Random Forest:  0.695
```

```r
rf.pred=rep("BLL",1259)
rf.pred[rf.probs > threshold]="NOT_BLL"
table(rf.pred,resp.test)
```

```
##          resp.test
## rf.pred   BLL NOT_BLL
##   BLL     270     171
##   NOT_BLL  91     727
```

```r
mean(rf.pred != resp.test)
```

```
## [1] 0.2081017
```

Conclusion

In conclusion, this analysis has determined that the random forest model works best to classify BL Lacs from a set of classification models, with a misclassification rate of 20.8%. Additionally, from the best subset selection, this analysis has also determined 12 predictor variables that are the most important for the classification of BL Lacs out of the original 20: `dec`, `detection_significance`, `log_energy_flux`, `log_lp_flux_density`, `log_npred`, `log_pivot_energy`, `log_plec_flux_density`, `log_variability_index`, `lp_beta`, `pl_index`, `plec_curve_significance`, and `ra`.

Bibliography

Ajello, M., Angioni, R., Axelsson, M., Ballet, J., Barbiellini, G., Bastieri, D., Becerra Gonzalez, J., Bellazzini, R., Bissaldi, E., Bloom, E. D., Bonino, R., Bottacini, E., Bruel, P., Buson, S., Cafardo, F., Cameron, R. A., Cavazzuti, E., Chen, S., Cheung, C. C., … Yassine, M. (2020). The fourth catalog of active galactic NUCLEI detected by the Fermi large Area Telescope. The Astrophysical Journal, 892(2), 105. https://doi.org/10.3847/1538-4357/ab791e

Freeman, P. E. 2021, online at https://github.com/pefreeman/36-290/tree/master/PROJECT_DATASETS/FERMI

NASA. (2020, December 17). FERMILPSC - Fermi LAT 10-Year Point Source Catalog (4FGL-DR2). Retrieved September 26, 2021, from https://heasarc.gsfc.nasa.gov/W3Browse/fermi/fermilpsc.html.