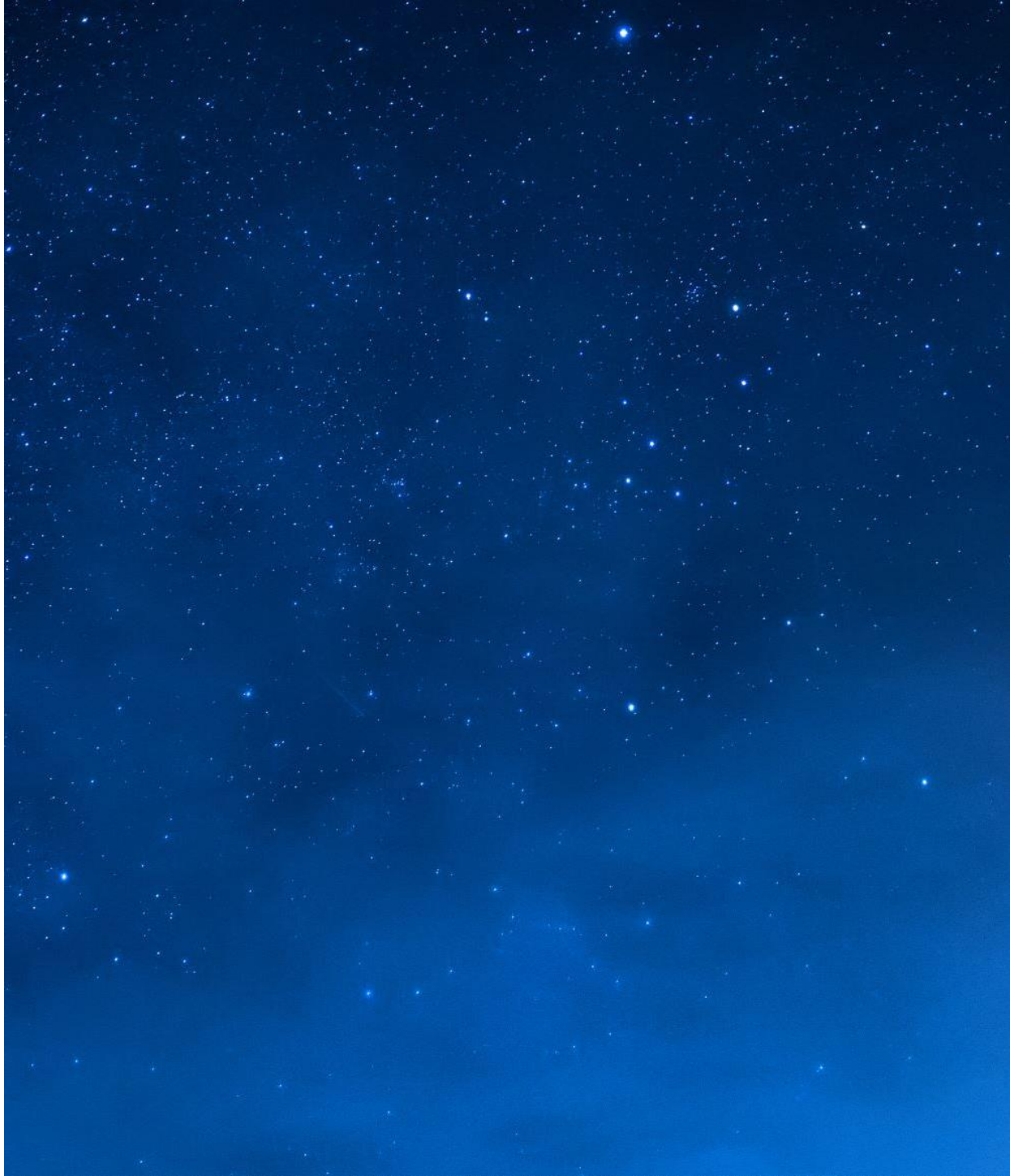
CLASSIFYING OBJECTS IN THE NIGHT SKY

Navid Mohseni, Sylvester Mensah,
Ethan Corr



AGENDA

In this presentation, we will

- Describe our research question
- Describe our dataset
- Initial data analysis
- Fit two different classification models
  - Multinomial Logistic classifier
  - Multi-Class SVM classifier
- Compare Performance
- Conclusions

Hubble Deep Field Image

RESEARCH QUESTION

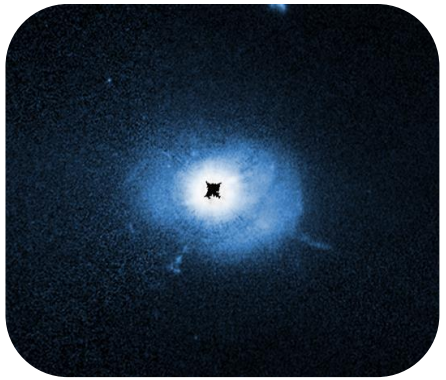
Our research question is “Can we classify night sky objects by using electromagnetic (EM) spectrum data?”

In our dataset from the Sloan Digital Sky Survey, astrophysicists categorized night-sky objects into 3 distinct classes:

1. **Star**
2. **Galaxy**
3. **Quasar (quasi-stellar object)**

Sun (star)

Milky Way (galaxy)

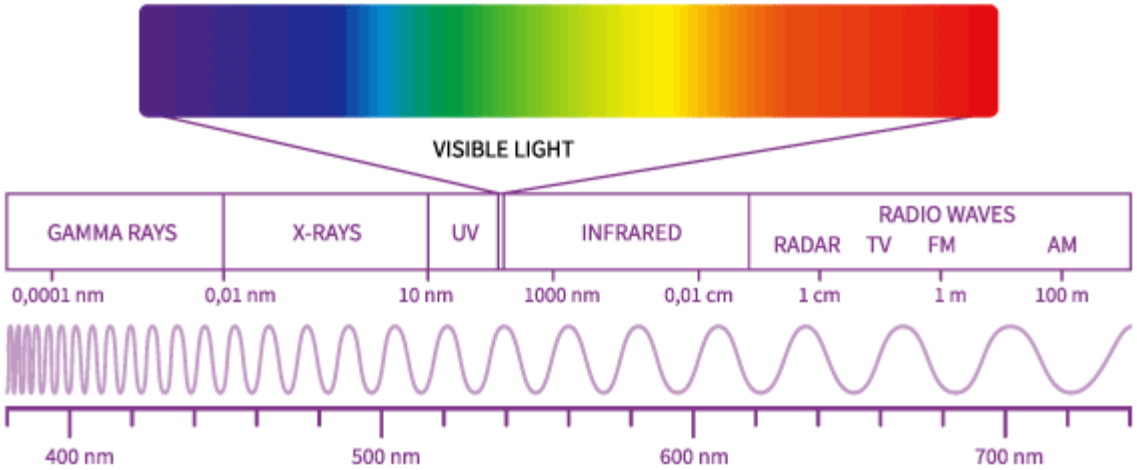


3C 273 (quasar)

DESCRIBE DATASET

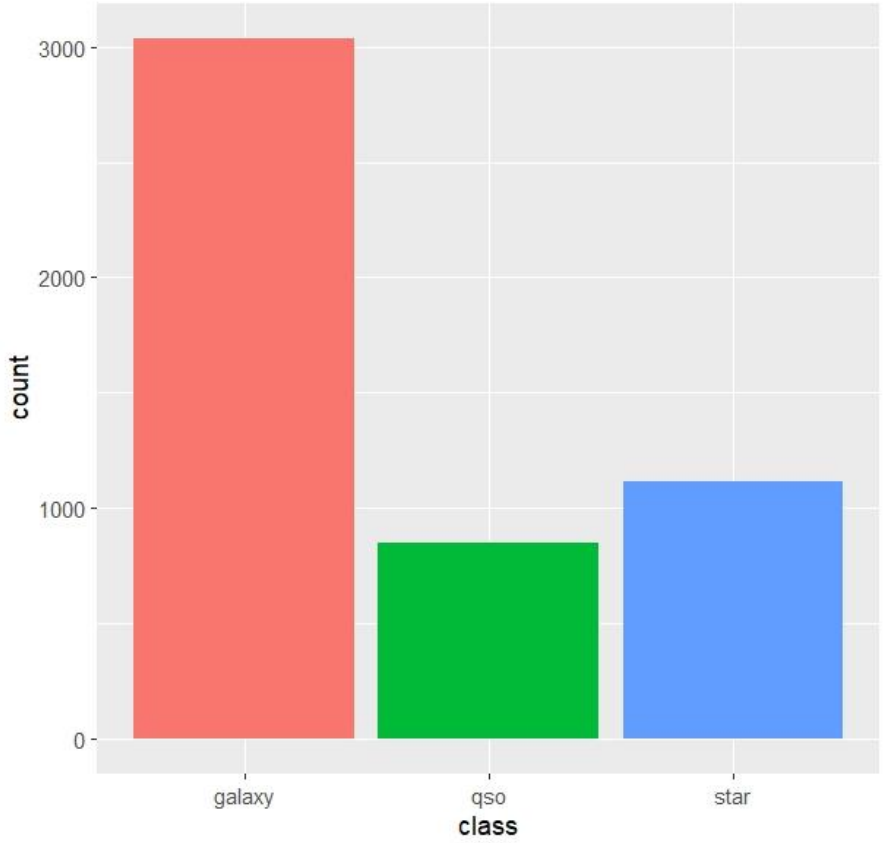
Our dataset consists of 100k observations of night-sky objects and selected 4 continuous features after cleaning data of irrelevant features. All features describe some quality of the light coming from the object. They are

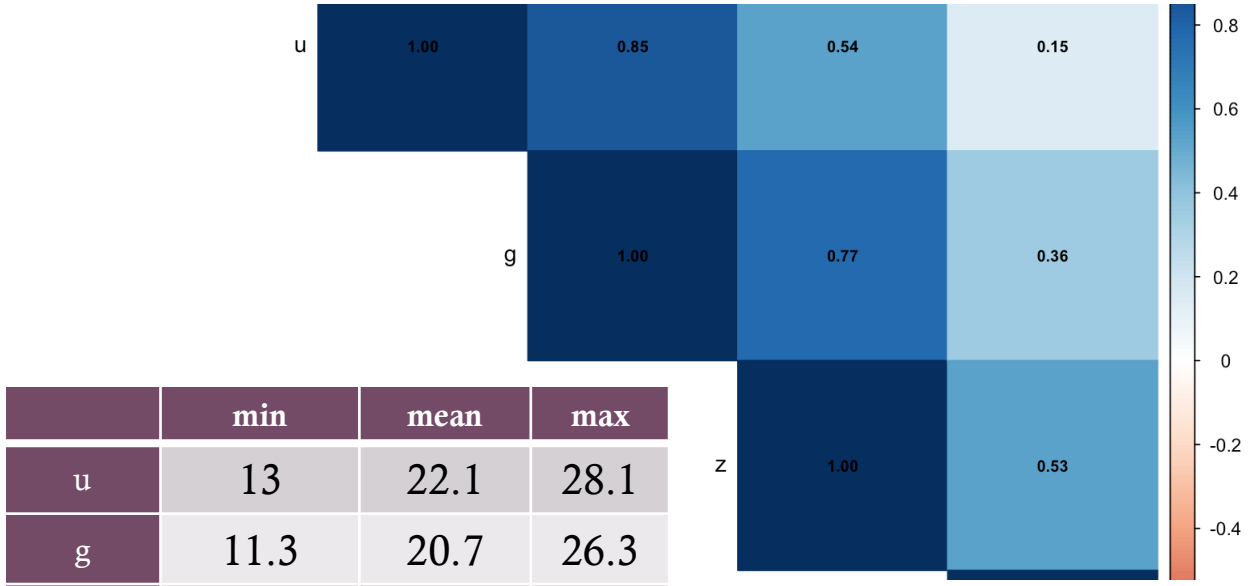
1. **u (ultraviolet band)**
2. **g (green band)**
3. **z (infrared band)**
4. **Redshift (change in wavelength)**



Electromagnetic Spectrum

EXPLORATORY DATA ANALYSIS (EDA)

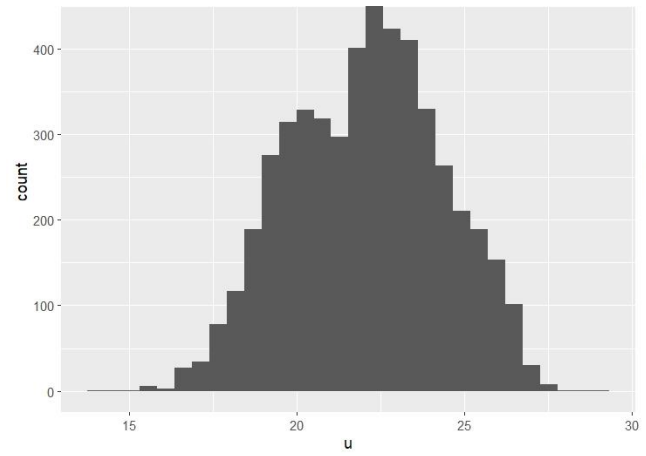


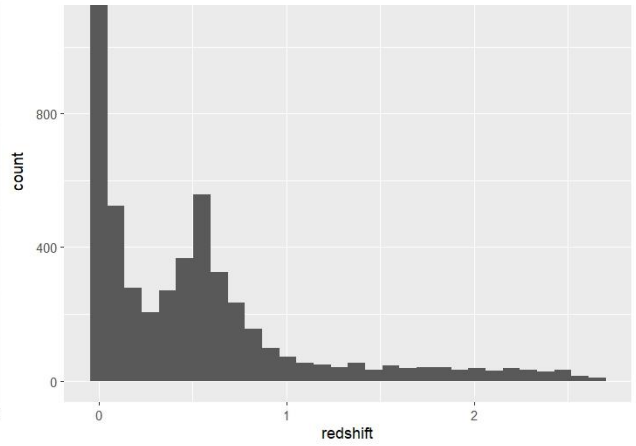


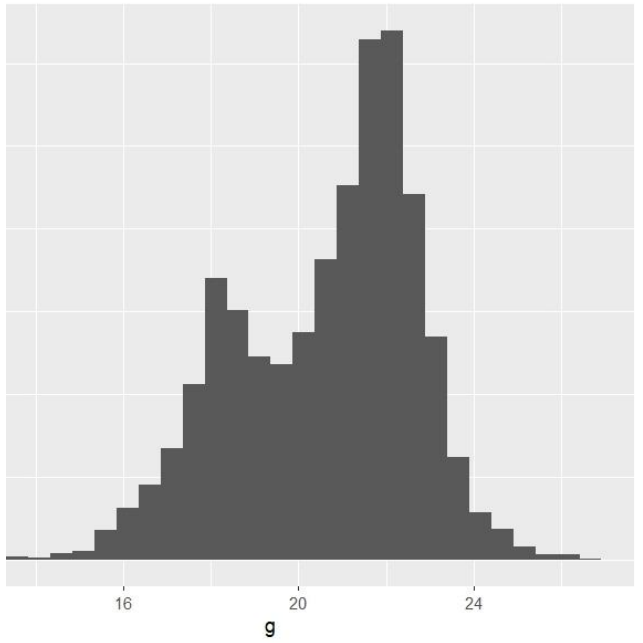
	min	mean	max
u	13	22.1	28.1
g	11.3	20.7	26.3
z	11.2	18.7	26.3
Redshift	-0.00414	0.516	2.65

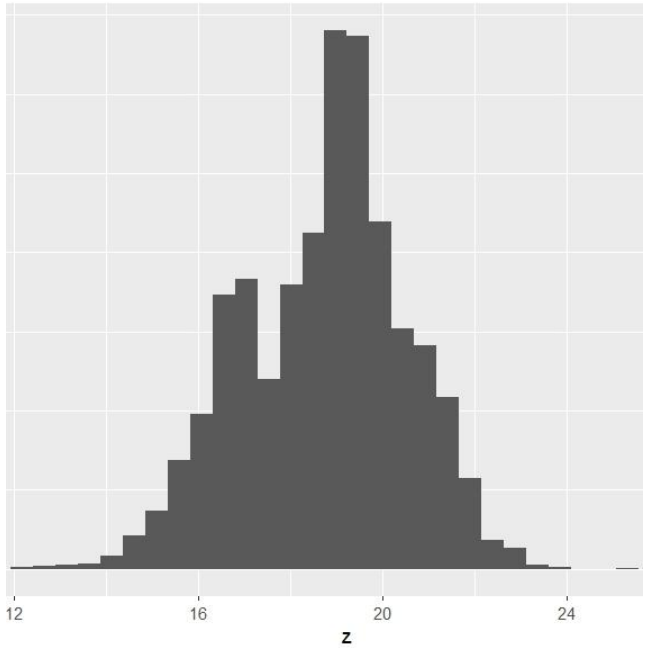
EXPLORATORY DATA ANALYSIS

- By visual inspection none of these variables are normally distributed.





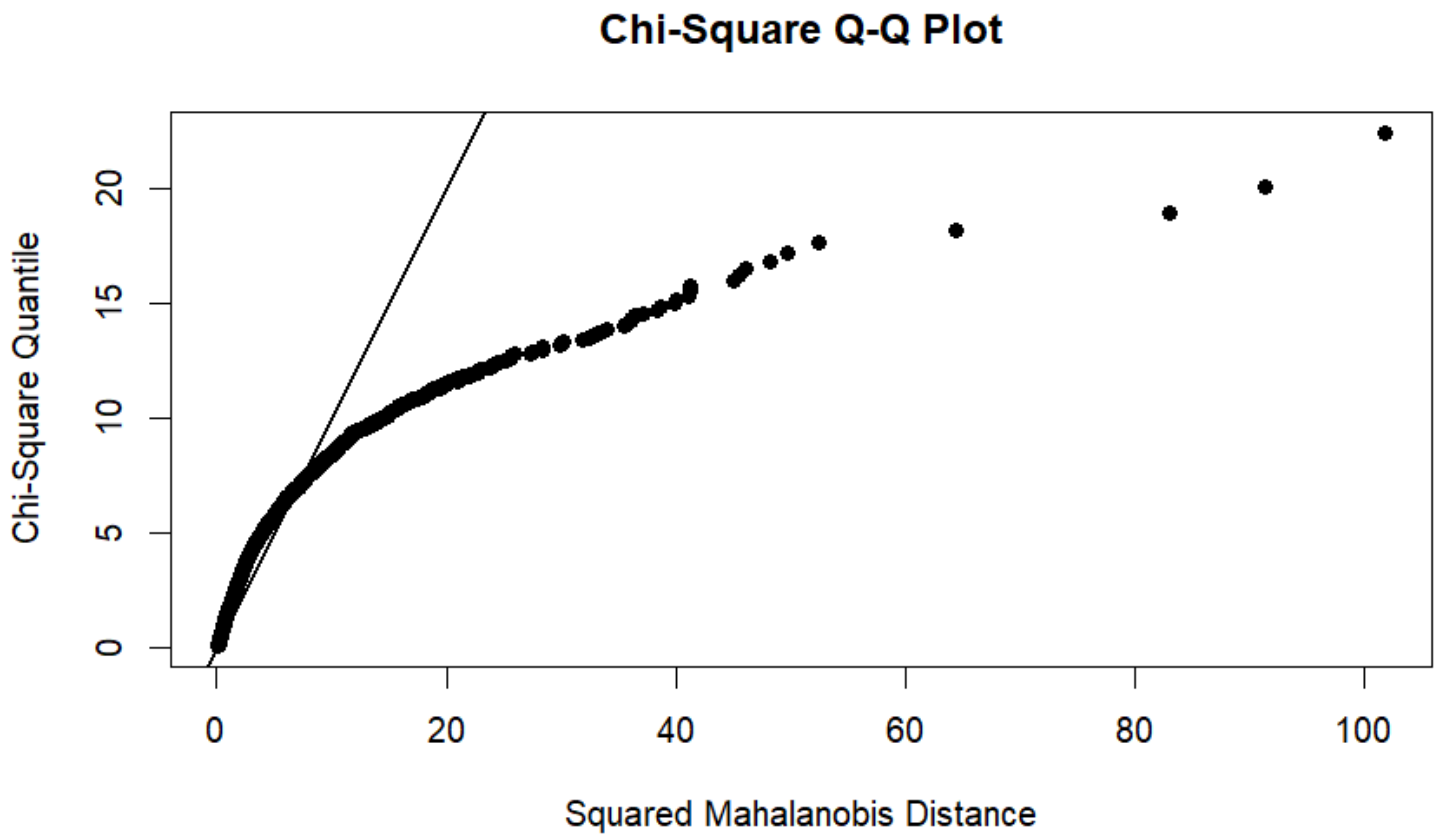




NORMALITY TEST

Before choosing our model, we want to examine the data. We use a subset of 5k observations for analysis and for training & testing.

We first check for normality using Anderson Darling test, but we see the data is not normally distributed within classes, so we do not use LDA or QDA.



Sample normality test plot for galaxy class (P-value < 0.001). Same case for the other classes.

LOGISTIC CLASSIFIER

$$\log\left(\frac{Pr(Y=k \mid \mathbf{x})}{Pr(Y=K \mid \mathbf{x})}\right) = \beta_{k0} + \beta_{k1}x_1 + \cdots + \beta_{kp}x_p.$$

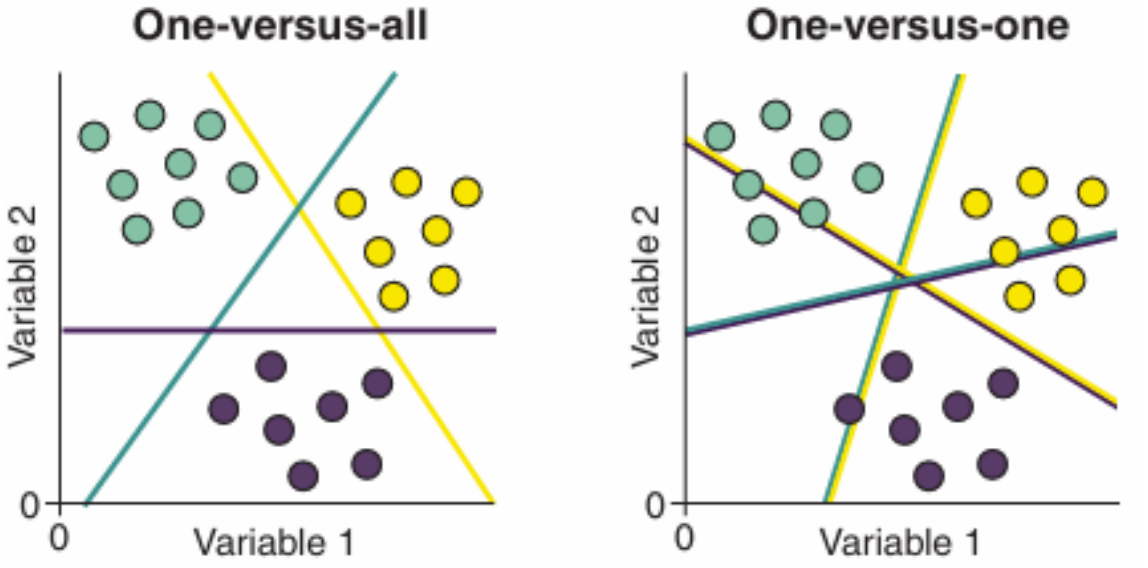
We split our data into training and testing sets, 75% and 25%, respectively.

We fit a multinomial logistic classifier that performed very well based on the training dataset.

		Prediction		
		Galaxy	**Quasar**	**Star**
True	Galaxy	2248	82	0
	Quasar	29	568	0
	Star	22	1	800

Training data
Misclassification rate:
3.57%

MULTI-CLASS SVM



- Extension of binary SVM to handle multiple classes (3 or more).
- Two common strategies:
  - One-vs-One (OvO): Constructs a binary classifier for every pair of classes.
  - One-vs-All (OvA): Builds a binary classifier for each class versus all remaining classes.
- Decision made by majority vote (OvO).

HYPERPARAMETERS AND TUNING

- Kernel: Determines how data is projected to higher-dimensional space to separate classes better.
- Degree (polynomial kernel): Higher degrees lead to more complex decision boundaries.
- Cost (C): Controls trade-off between maximizing the margin and minimizing classification errors.
- Gamma (γ): Controls influence of a single training example.
  - Large gamma (γ ↑): More complex, granular boundary. Prone to overfitting (high variance).
  - Small gamma (γ ↓): Less complex boundary. Prone to underfitting (high bias).
- Hyperparameter Tuning:
  - Random search or Grid search methods help find optimal values efficiently.

MULTI-CLASS SVM

$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{minimize}} \left\{ \sum_{i=1}^{n} \max\left[0, 1 - y_i f(x_i)\right] + \lambda \sum_{j=1}^{p} \beta_j^2 \right\},$$

- Kernel: Polynomial
- Degree: 3
- Cost (C): 5.92
- Gamma: 3.42

Accuracy: 95.87%

Prediction

True	Galaxy	Quasar	Star
Galaxy	2223	15	21
Quasar	118	521	0
Star	1	0	851

Training data
Misclassification rate:
4.10%

COMPARE PERFORMANCE

We can compare the two models by making predictions with test data:

- Multinomial Logistic classifier
  - 4.20% test data misclassification rate
- Multi-class SVM classifier
  - 5.52% test data misclassification rate

Logistic

Prediction \ Truth	Galaxy	Quasar	Star
Galaxy	769	27	0
Quasar	15	166	0
Star	11	0	262

SVM

Prediction \ Truth	Galaxy	Quasar	Star
Galaxy	766	53	2
Quasar	6	154	0
Star	8	0	261

CONCLUSION

In this presentation,

- We discussed the multi-class SVM classifier method.
- Fit 2 classification models to our night sky objects dataset: a multinomial regression classifier and a multi-class SVM classifier.
- Logistic Regression slightly outperforms Multi-class SVM, having a lower misclassification rate and higher accuracy.

Q&A

Midterm 2

Navid, Ethan, Sylvester

Check LDA/QDA Assumption

We first want to check the distribution of our data. If it comes from approximately MVN, we can use LDA/QDA methods. If not, then we should use alternative methods. Below, we check to see if this is the case.

```
library(tidyverse)
library(MVN)

# load cleaned dataset
df <- data.frame(readRDS("subset_cleaned_data.rds"))
# split into classes
galaxy_df <- df[df$class == "galaxy", ]
star_df <- df[df$class == "star", ]
qso_df <- df[df$class == "qso", ]
# select only numerics
galaxy_test_df <- select(galaxy_df, -"class")
star_test_df <- select(star_df, -"class")
qso_test_df <- select(qso_df, -"class")

# Check each group for MVN
paste('galaxy')
galaxy_result <- mvn(data = galaxy_test_df, mvnTest = "mardia", multivariatePlot = "qq")
print(galaxy_result)
paste('star')
star_result <- mvn(data = star_test_df, mvnTest = "mardia", multivariatePlot = "qq")
print(star_result)
paste('qso')
qso_result <- mvn(data = qso_test_df, mvnTest = "mardia", multivariatePlot = "qq")
print(qso_result)
```

```
# You can also try:
#result <- mvn(data = numeric_df, mvnTest = "hz")      # Henze-Zirkler
#result <- mvn(data = numeric_df, mvnTest = "royston") # Royston's test
```

Logistic

Our results show that for all 3 classes, none of the variables come from a normal distribution. Thus, we will not use LDA/QDA for our analysis. Instead, we will use a multi-class SVM and a logistic (multinomial) classification model.

First, we split the data into training and testing, 75% and 25%

```
# Load required packages
library(dplyr)
library(rsample)
# Set seed for reproducibility
set.seed(1234)
# Split data into training and testing (75/25 split)
data_split <- initial_split(df, prop = 3/4)
train_df <- training(data_split)
test_df <- testing(data_split)
```

And now we fit our logistic classification model with the training set & make predictions with our test set

```
# logistic regression model
library(glmnet)
library(Matrix)
library(tidyverse)
# setup data for analysis
X <- as.matrix(select(train_df, -"class"))
Y <- train_df$class
Y <- as.factor(Y)
# fit model
logit_glmnet <- cv.glmnet(X, Y, family = "multinomial", type.measure = "class")
```

We can then make predictions with our training dataset to evaluate training error

```
# make predictions (training data)
X_train <- as.matrix(select(train_df, -"class"))
Y_train <- train_df$class
```

```
Y_train <- as.factor(Y_train)
log_train.pred <- predict(logit_glmnet, X_train, s = "lambda.min", type = "class")
# show training data confusion matrix
paste('training data performance')
confusion_mat_train <- table(log_train.pred,Y_train)
print(confusion_mat_train)
# compute train error
logistic_train_err <- 1-sum(diag(confusion_mat_train))/(5000*.75)
paste0('Logistic classifier train error rate: ',logistic_train_err)
```

and then we can make predictions with our testing dataset to evaluate test error

```
# make predictions for test dataset
X_test <- as.matrix(select(test_df, -"class"))
Y_test <- test_df$class
Y_test <- as.factor(Y_test)
log.pred <- predict(logit_glmnet, X_test, s = "lambda.min", type = "class")
# show table
confusion_mat <- table(log.pred,Y_test)
print(confusion_mat)
# compute test error
logistic_test_err <- 1-sum(diag(confusion_mat))/(5000*.25)
paste0('Logistic classifier test error rate: ',logistic_test_err)
```

SVM

```
trainTask <- makeClassifTask(data = train_data,
                             target = "class")
testTask <- makeClassifTask(data = test_data,
                            target = "class")
#classTask <- makeClassifTask(data = data, target = "class")
svm <- makeLearner("classif.svm")
getParamSet("classif.svm")
getParamSet("classif.svm")$pars$kernel$values


kernels <- c("polynomial", "radial", "sigmoid")
svmParamSpace <- makeParamSet(
  makeDiscreteParam("kernel", values = kernels),
  makeIntegerParam("degree", lower = 1, upper = 3),
```

```
  makeNumericParam("cost", lower = 0.1, upper = 10),
  makeNumericParam("gamma", lower = 0.1, 10))


randSearch <- makeTuneControlRandom(maxit = 50)
cvForTuning <- makeResampleDesc("Holdout", split = 3/4)

parallelStartSocket(cpus = detectCores())
tunedSvmPars <- tuneParams("classif.svm", task = trainTask,
                           resampling = cvForTuning,
                           par.set = svmParamSpace,
                           control = randSearch)
parallelStop()
tunedSvmPars

tunedSvm <- setHyperPars(svm,
                         par.vals = tunedSvmPars$x)
tunedSvmModel <- mlr::train(tunedSvm, trainTask)

#Prediction on Training
train_predictions <- predict(tunedSvmModel,
                             task = trainTask)
train_conf_matrix <- calculateConfusionMatrix(train_predictions)
train_mmce <- performance(train_predictions)
train_mmce

test_predictions <- predict(tunedSvmModel, task = testTask)
test_conf_matrix <- calculateConfusionMatrix(test_predictions)
test_mmce <- performance(test_predictions)

test_mmce
```