Introduction to Machine Learning in 2023

https://www.v7labs.com/blog/supervised-vs-unsupervised-learning

- Supervised Learning
  - Classification
    - Identity Fraud Detection
    - Image Classification
    - Customer Retention
    - Diagnostics
  - Regression
    - Population Growth Prediction
    - Estimating life expectancy
    - Market Forecasting

Supervised and unsupervised machine learning are two different approaches to training machine learning models.

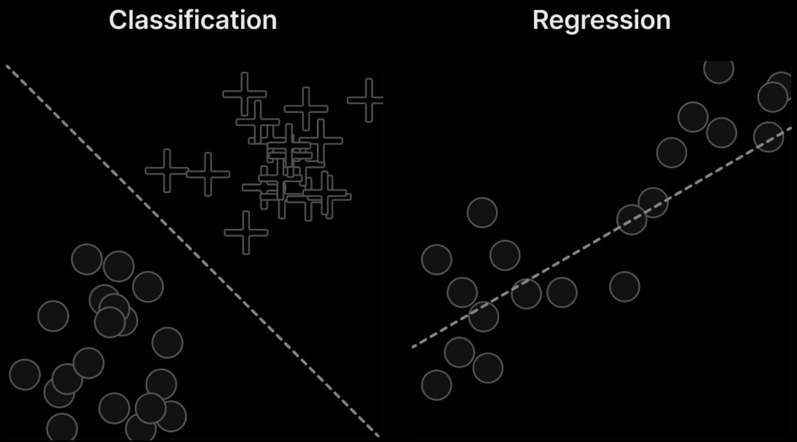
Supervised learning involves training a model on a labeled dataset, where each data point is associated with a target label or output. The goal of supervised learning is to learn a mapping from input features to output labels, such that the model can predict the correct label for new, unseen data. We have been working on supervised machine learning with techniques like linear regression, decision trees, support vector machines, and neural networks.

In contrast, unsupervised learning involves training a model on an unlabeled dataset, where there are no target outputs to guide the learning process. The goal of unsupervised learning is to discover patterns, structure, and relationships within the data. Examples of unsupervised learning algorithms include face recognition.

Introduction to
Classification and Regression Techniques

Classification and regression techniques are statistical methods used to analyze data and make predictions. These techniques are widely used in machine learning, data mining, and other applications where prediction is important.

Classification techniques are used to predict a categorical variable, while regression techniques are used to predict a continuous variable. Some common classification and regression techniques include Lasso, Ridge, and K-Nearest Neighbors (KNN).



https://www.v7labs.com/blog/supervised-vs-unsupervised-learning V7 Labs

Pragati Baheti
Microsoft

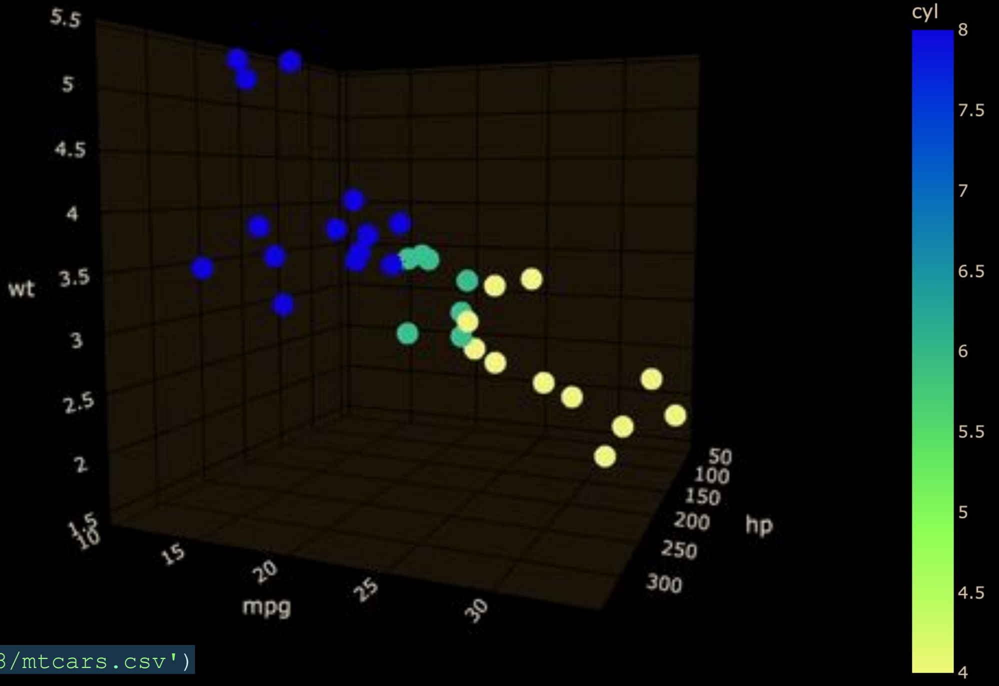
Pragati is a software developer at Microsoft, and a deep learning enthusiast. She writes about the fundamental mathematics behind deep neural networks.

Grace Dwinnell

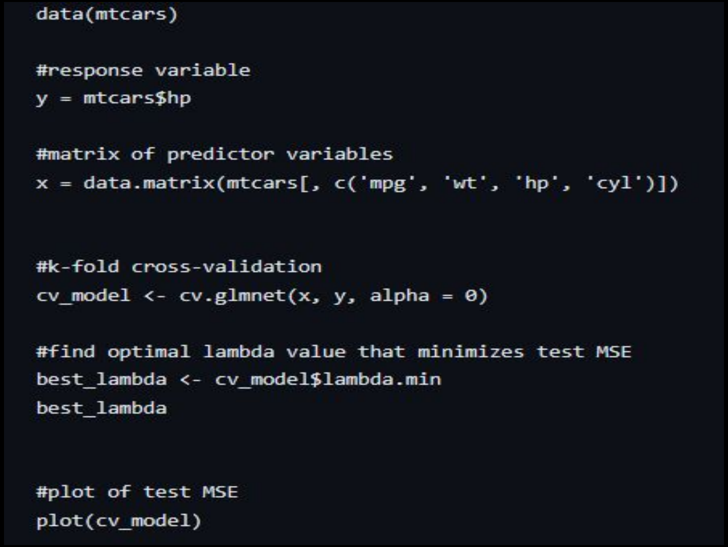
MTCARS 🚘 DATASET

MtCars is a built-in dataset in R Studio with 32 observations and 11 attributes for those car types shown.

```
import plotly.express as px
import pandas as pd
df = pd.read_csv(r'/Users/grace/Documents/math358/mtcars.csv')
fig = px.scatter_3d(df, x='hp', y='mpg', z='wt',color='cyl')
fig.show()
```



Grace Dwinnell



```
data(mtcars)

#response variable
y = mtcars$hp

#matrix of predictor variables
x = data.matrix(mtcars[, c('mpg', 'wt', 'hp', 'cyl')])


#k-fold cross-validation
cv_model <- cv.glmnet(x, y, alpha = 0)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda


#plot of test MSE
plot(cv_model)
```

Dataset	***MSE***
Mtcars, cyl	9.268
Spam, type	0.6746652
Heart, DEATH_EVENT	141.4084
Wine, Wine	0.8784257
Fast Food, calories	9783.067

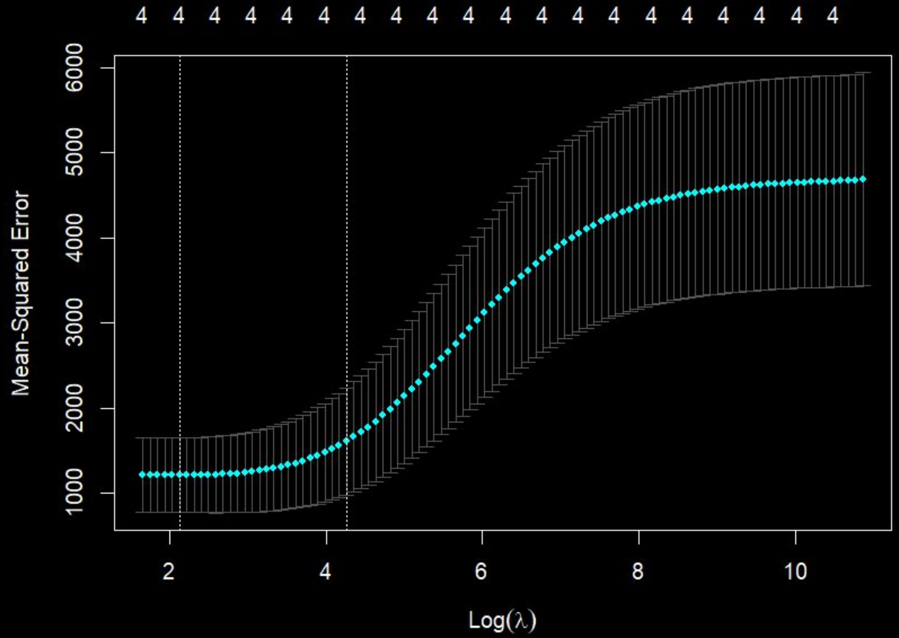
Lasso Regression

Lasso regression is a type of linear regression that uses regularization to prevent overfitting. It works by adding a penalty term to the cost function that shrinks the coefficients towards zero.

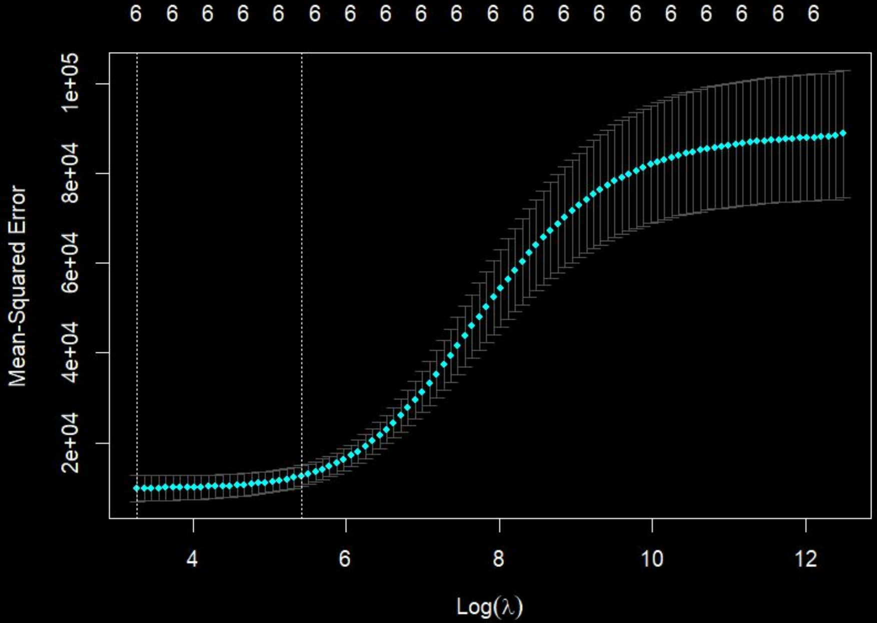
This technique is particularly useful when dealing with high-dimensional data, where there are many variables but only a few are relevant. Lasso regression can help identify the most important variables and remove the irrelevant ones.

Grace Dwinnell

Lasso Regression



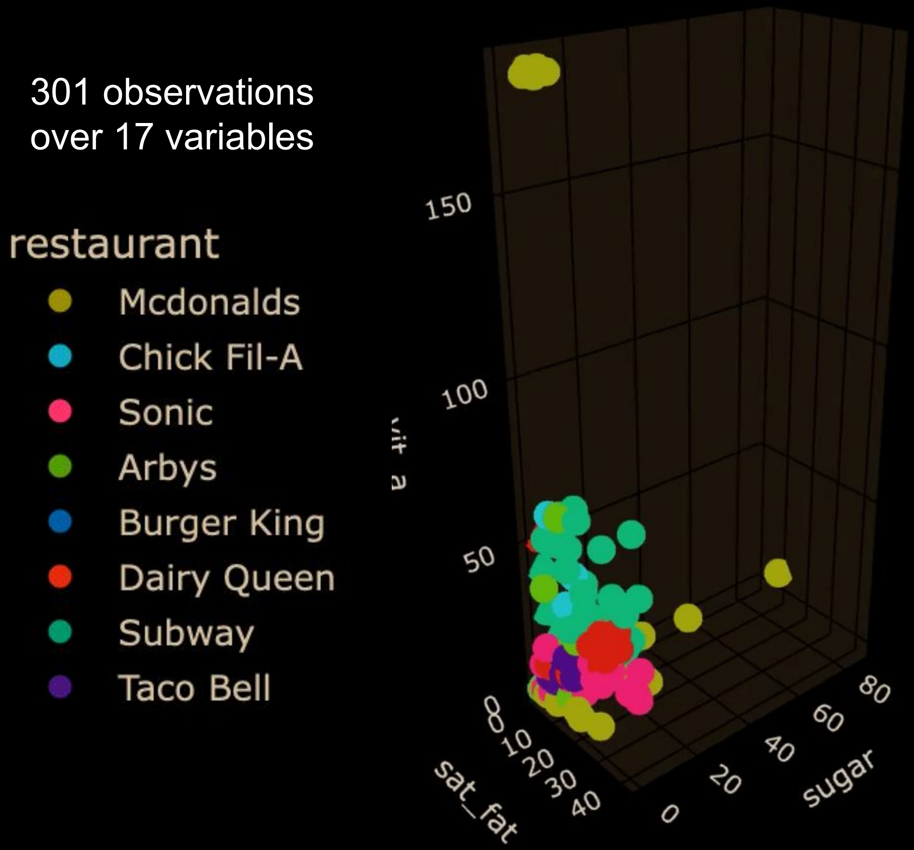
MtCars



FastFood

Dana Sutfin

301 observations
over 17 variables



Nutrition 🍔

DATASET

This Fast Food dataset provides a comprehensive breakdown of the nutritional content of various fast food products from popular fast food chains

With information on calories, fat, carbohydrates, protein, and other key nutrients, this dataset provides a valuable resource for nutritionists, researchers, and health-conscious individuals.

Dana Sutfin

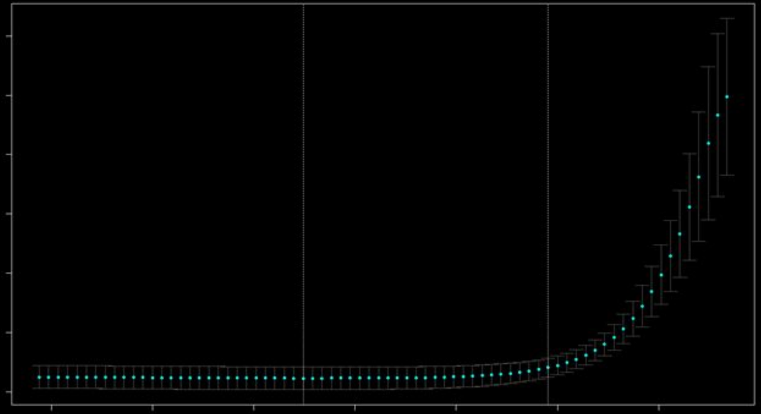
Ridge Regression

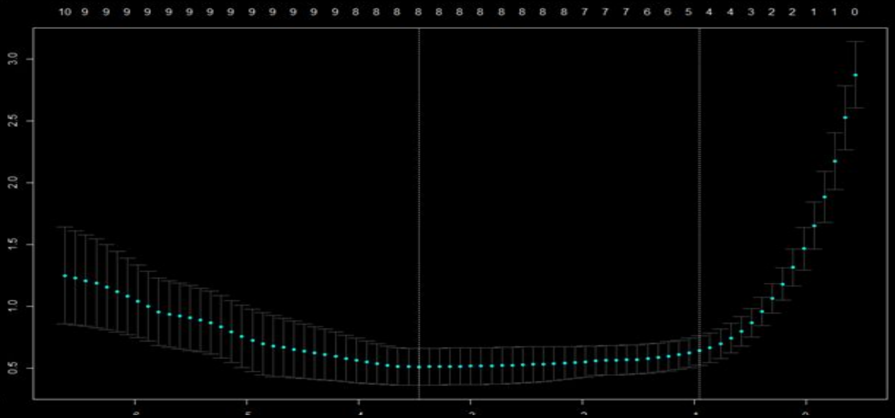
Ridge regression is another type of linear regression that uses regularization to prevent overfitting. It works by adding a penalty term to the cost function that shrinks the coefficients towards zero, but unlike lasso regression, it doesn't force any coefficients to be exactly zero.

This technique is particularly useful when dealing with multicollinearity, where there are strong correlations between predictor variables. Ridge regression can help reduce the impact of these correlations and improve the stability of the model.

Dataset	***MSE***
Mtcars, cyl	10.09911
Spam, type	0.3667885
Heart, DEATH_EVENT	0.3750558
Wine, Wine	0.6631258
Fast Food, calories	112401.4

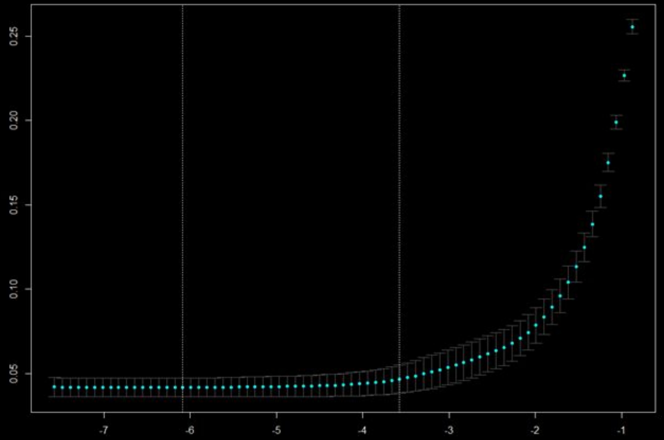
Dana Sutfin





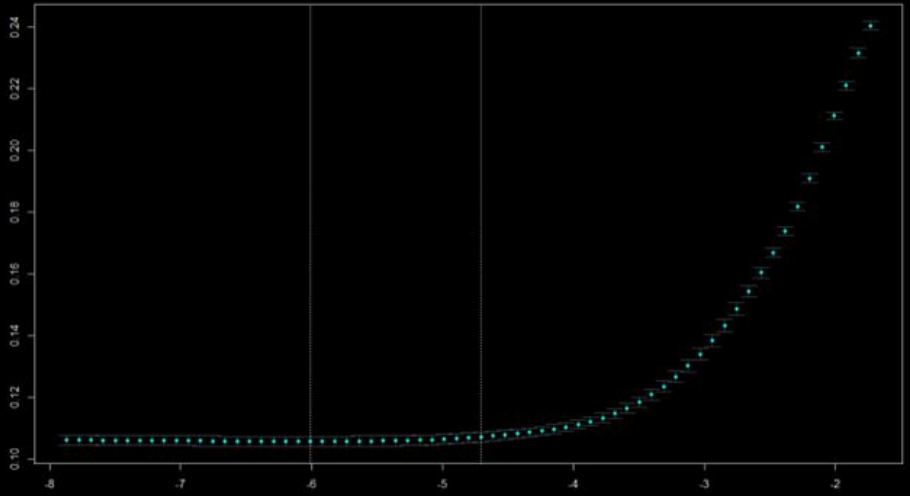
Mtcars

Fast food

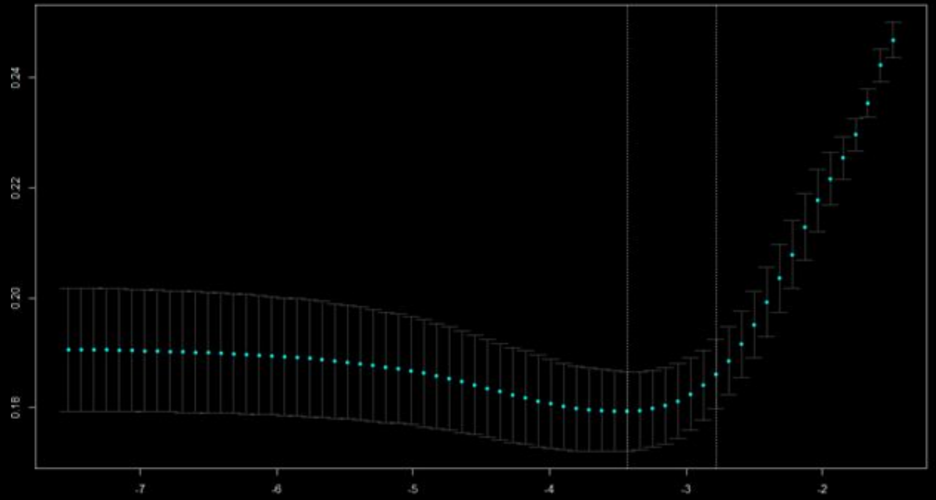


Wine

Dana Sutfin



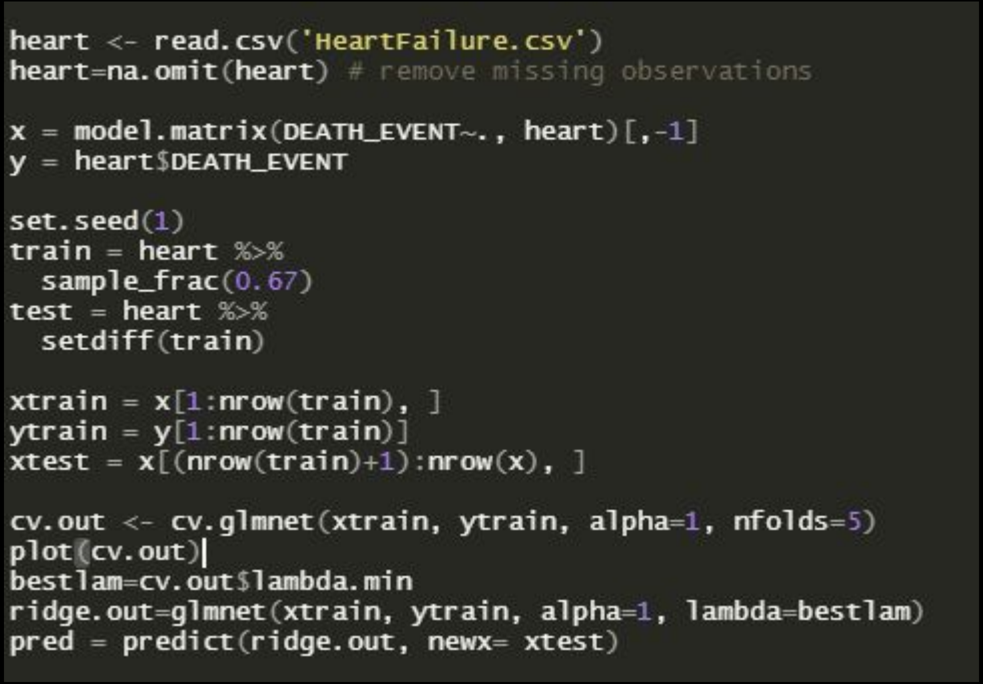
Spam



Heart Failure

Dana Sutfin

Code:



```
heart <- read.csv('HeartFailure.csv')
heart=na.omit(heart) # remove missing observations

x = model.matrix(DEATH_EVENT~., heart)[,-1]
y = heart$DEATH_EVENT

set.seed(1)
train = heart %>%
  sample_frac(0.67)
test = heart %>%
  setdiff(train)

xtrain = x[1:nrow(train), ]
ytrain = y[1:nrow(train)]
xtest = x[(nrow(train)+1):nrow(x), ]

cv.out <- cv.glmnet(xtrain, ytrain, alpha=1, nfolds=5)
plot(cv.out)
bestlam=cv.out$lambda.min
ridge.out=glmnet(xtrain, ytrain, alpha=1, lambda=bestlam)
pred = predict(ridge.out, newx= xtest)
```

Hannah Phommachanthone

Heart Failure ❤️

DATASET

13 columns, 299 rows

age - patient age in years
anaemia- whether patients had a decrease in RBC or hemoglobin
0=”No”
1=”Yes”
creatine_phosphokinase- level of CPK in the patient’s blood
DEATH_EVENT- whether the patient died during follow-up period
0=”No”
1=”Yes”
diabetes- if the patient is diabetic
0=”No”
1=”Yes
ejection_fraction- % of blood leaving the heart after a contraction
high_blood_pressure- whether patients are hypertensive
0=”No”
1=”Yes”
platelets- amount of platelets in blood
serum_creatinine- level of serum creatinine in blood
serum_sodium- level of serum sodium in blood
smoking- whether the patient is a smoker
0=”No”
1=”Yes”
time- how many days after follow-up

The Heart Failure data set was obtained from UC Irvine’s Machine Learning Repository. The data set contains information collected from 299 patients during their follow-up after having heart failure.

The data set consists of 13 variables that each patient had a value for. For KNN classification, the response variable was DEATH_EVENT. For KNN regression, the response variable was platelets.

About KNN

KNN Explained

Hannah Phommachanthone

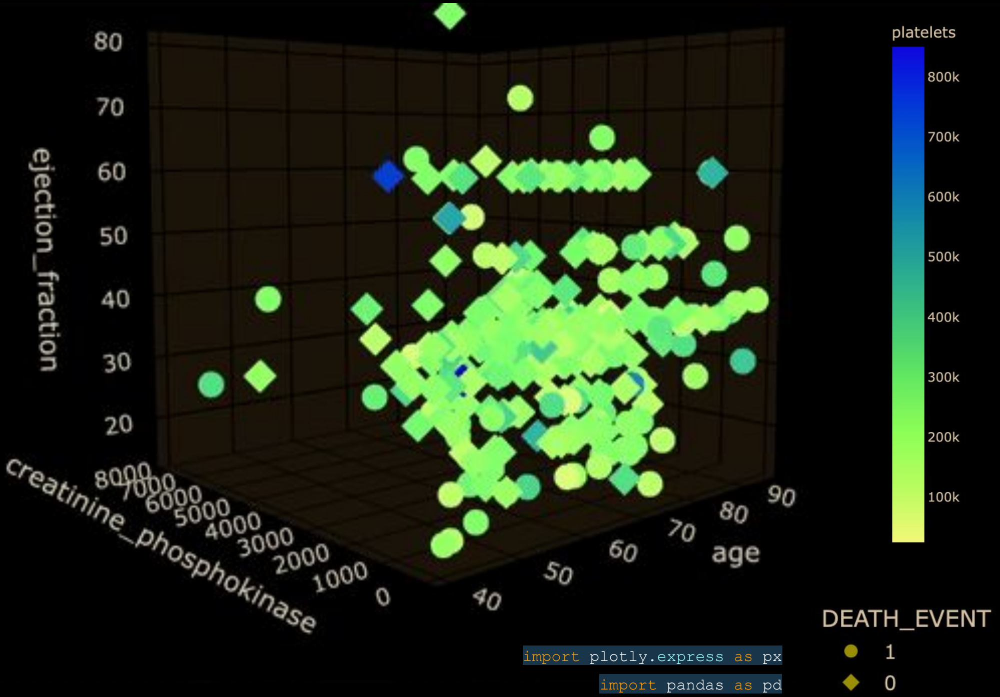
K-Nearest Neighbors

KNN is a non-parametric classification technique that works by finding the k nearest neighbors to a given data point and assigning it the majority class of those neighbors.

This technique is particularly useful when dealing with nonlinear relationships between predictor variables and the target variable. However, it can be computationally expensive and may not work well with high-dimensional data.

About KNN

KNN Explained

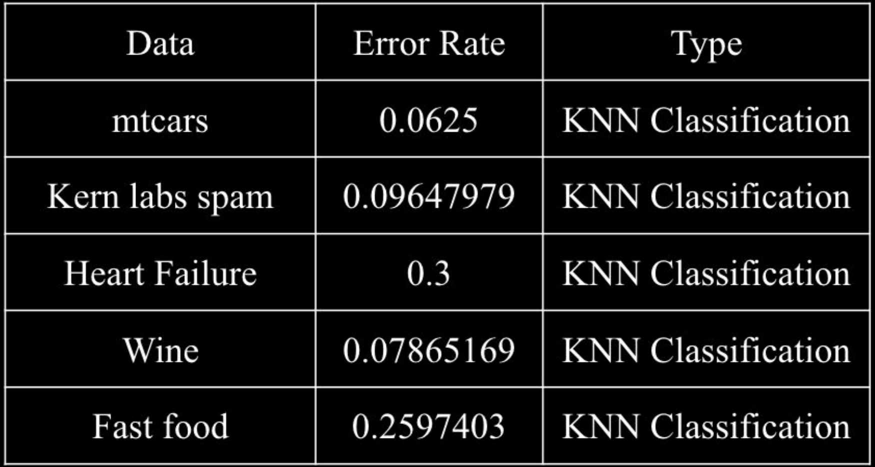


```
import plotly.express as px
import pandas as pd
df = pd.read_csv(r'/Users/hannah/Downloads/HeartFailure.csv')
print(df)
fig = px.scatter_3d(df, x='age', y='creatinine_phosphokinase', z='ejection_fraction',
                    color='platelets', symbol='DEATH_EVENT')
fig.show()
```

Hannah Phommachanthone

KNN

Classification



Data	Error Rate	Type
mtcars	0.0625	KNN Classification
Kern labs spam	0.09647979	KNN Classification
Heart Failure	0.3	KNN Classification
Wine	0.07865169	KNN Classification
Fast food	0.2597403	KNN Classification

The KNN classification technique was run on these five data sets and their error rates compared. Error rates for KNN classification indicate how many classifications are actually correct.

For mtcars, kern labs spam, heart failure, wine, and fast food data sets, the response variables were “cyl” “type,” “DEATH_EVENT,” “Wine,” and “restaurant,” respectively.

About KNN

KNN Explained

Hannah Phommachanthone

KNN Heart Failure classification code

```
set.seed(1)

HeartX=model.matrix(DEATH_EVENT~., data=Heart)[,-1]
HeartX=scale(HeartX)
idx2=sample(1:nrow(Heart))
npart=floor(0.5*nrow(Heart))
idx=idx2[1:npart]

HeartX_train = HeartX[idx,]
HeartY_train = Heart$DEATH_EVENT[idx]

HeartX_test = HeartX[-idx,]
HeartY_test = Heart$DEATH_EVENT[-idx]

#Error rate
predicted = knn(train = HeartX_train, test = HeartX_test, cl= HeartY_train, k=5)
mean(HeartY_test != predicted)
```

Hannah Phommachanthone

KNN Heart Failure code for finding optimal K

```
set.seed(1)

M=20
miserror=rep(0, M)

for (i in 1: M){
  pred=knn(train=HeartX_train,test=HeartX_test,cl=HeartY_train,k=i)
  miserror[i]=mean(pred != HeartY_test)
}
which(miserror==min(miserror))
```

The objective is to minimize the misclassification error.

M sets the maximum number of neighbors (k) to consider while searching for the optimal k value.

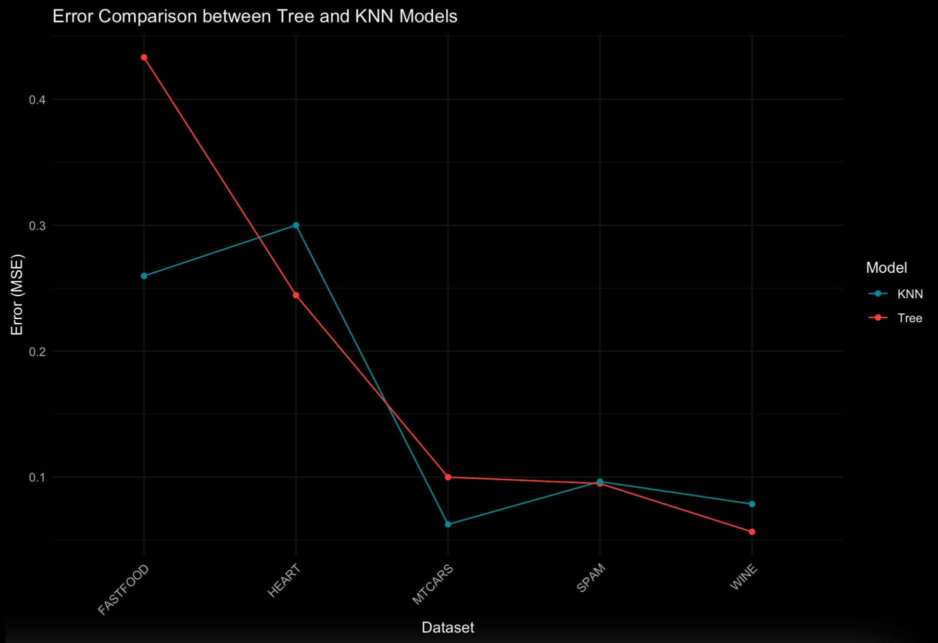
M should be chosen based on the size of the data. Small datasets should use a large M value to improve accuracy. Large datasets should use a smaller M to speed up the algorithm.

Hannah Phommachanthone

K-Nearest Neighbors

While KNN is commonly used as a non-parametric classification technique, it can also be used as a regression technique. KNN regression works by using values associated most closely to the k that is the response variable and combines the values to get a singular output.

For mtcars, kern labs spam, heart failure, wine, and fast food data sets, the response variables were “hp,” “free,” “platelets,” “Malic.acid,” and “calories,” respectively.



About KNN

KNN Explained

Hannah Phommachanthone

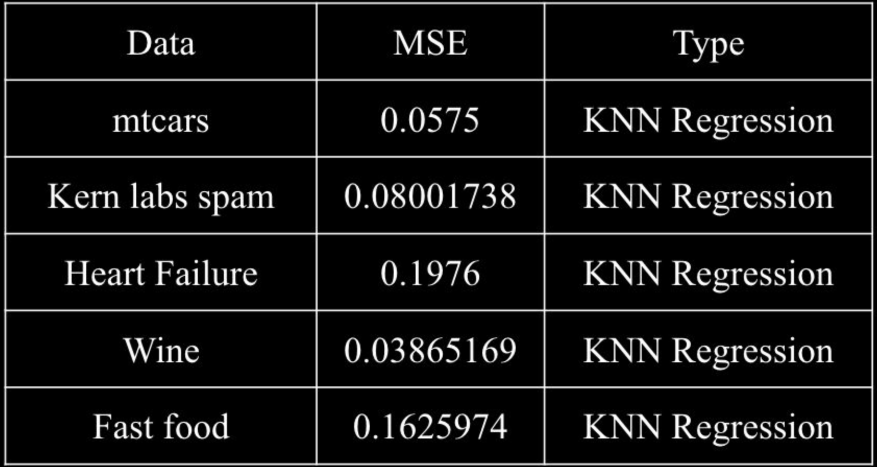
```r
data <- data.frame(
  Dataset = rep(c("MTCARS", "SPAM", "HEART", "WINE", "FASTFOOD"), 2),
  Error = c(0.1, 0.09492754, 0.2444444, 0.05660377, 0.4333, 0.0625, 0.09647979, 0.3, 0.07865169,
0.2597403),
  Model = rep(c("Tree", "KNN"), each = 5)
)

# Create the ggplot line chart
ggplot(data, aes(x = Dataset, y = Error, group = Model, color = Model)) +
  geom_line() +
  geom_point() +
  theme_minimal() +
  labs(title = "Error Comparison between Tree and KNN Models",
       x = "Dataset",
       y = "Error (MSE)",
       color = "Model") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

Hannah Phommachanthone

KNN

Regression



Data	MSE	Type
mtcars	0.0575	KNN Regression
Kern labs spam	0.08001738	KNN Regression
Heart Failure	0.1976	KNN Regression
Wine	0.03865169	KNN Regression
Fast food	0.1625974	KNN Regression

The KNN regression technique was run on these five data sets and their mean squared errors compared. MSE is a measure of how much deviance there is between the predicted values for the response variables and the actual values.

Based on the MSE results, the wine data set had the least amount of deviance between predicted Malic.acid amount and actual Malic.acid amount while the heart failure data set had the most deviance.

About KNN

KNN Explained

Hannah Phommachanthone

KNN Heart Failure regression code

```
set.seed(1)

HeartX=model.matrix(platelets~., data=Heart)[,-1]
HeartX=scale(HeartX)
idx2=sample(1:nrow(Heart))
npart=floor(0.5*nrow(Heart))
idx=idx2[1:npart]

HeartX_train = HeartX[idx,]
HeartY_train = Heart$platelets[idx]

HeartX_test = HeartX[-idx,]
HeartY_test = Heart$platelets[-idx]

#Mean squared error
predout=knn.reg(train=HeartX_train, test=HeartX_test, y=HeartY_train, k=5)
pred=predout$pred
mean((HeartY_test - pred)^2)
```

Justin Matthew Newman

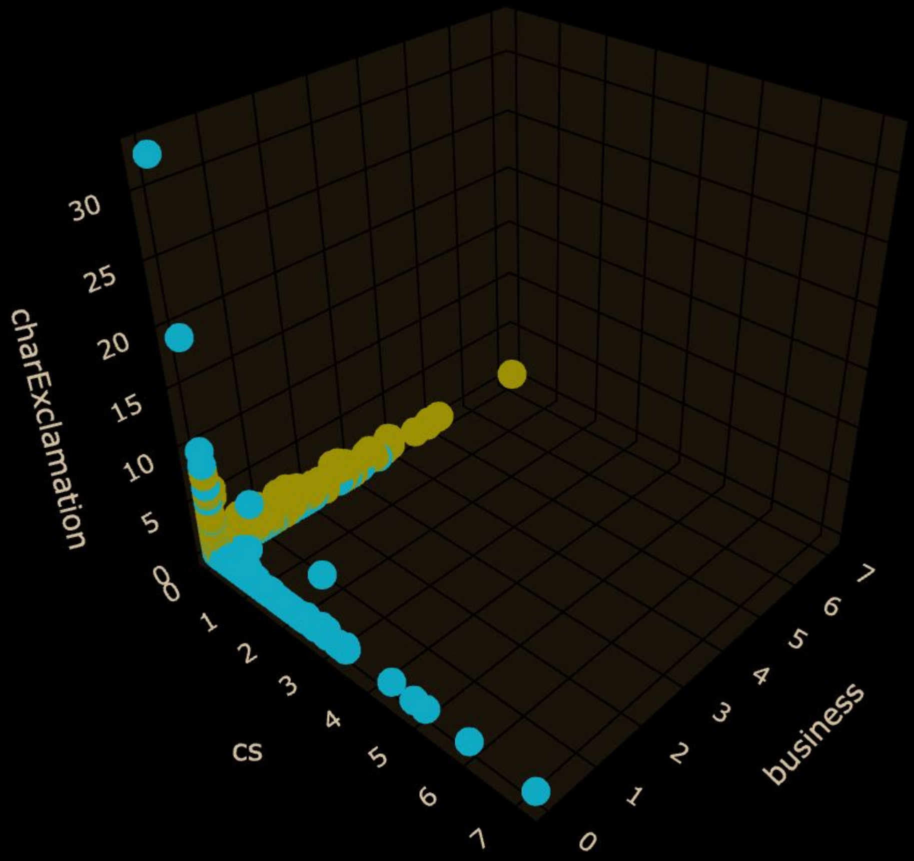
SPAM EMAIL 📧 DATASET

The Kern Labs spam email dataset has been widely used in research on spam filtering algorithms, machine learning, and natural language processing. It is a valuable resource for anyone interested in developing and testing email classification models, and it provides a standardized way to compare the performance of different algorithms. There are 57 variables indicating the frequency of certain words and characters in the email.

type

● spam

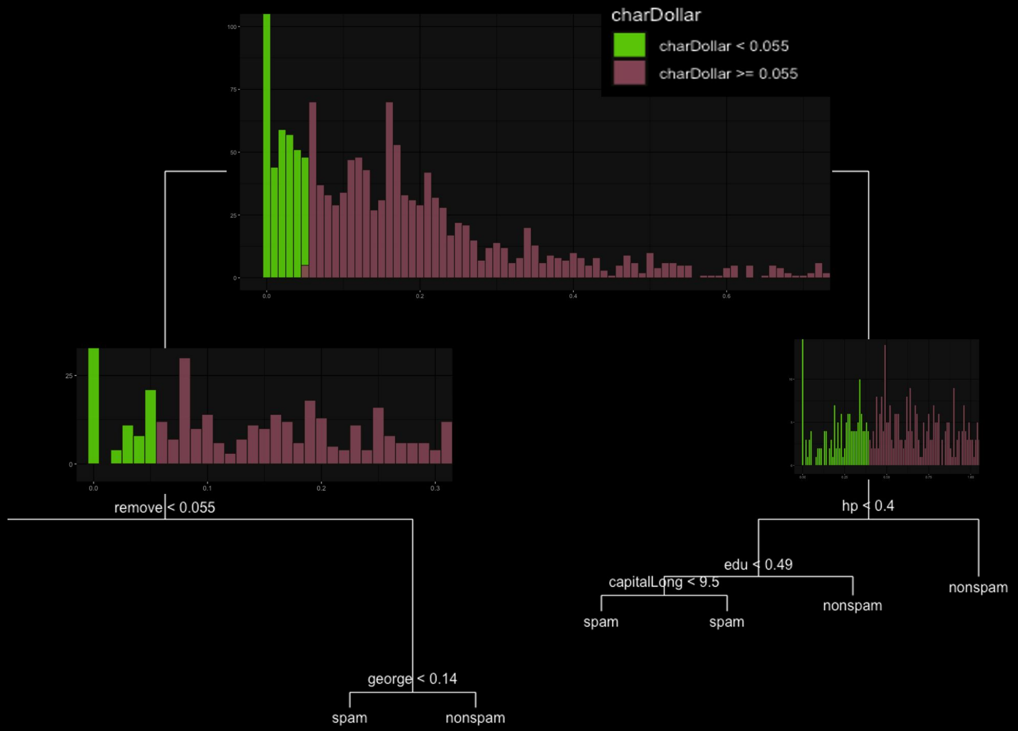
● nonspam



Justin Matthew Newman

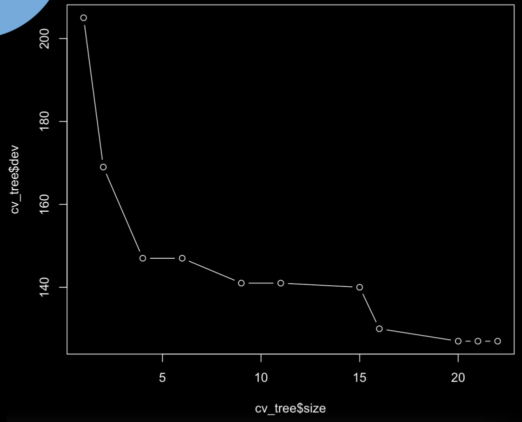
Decision Trees

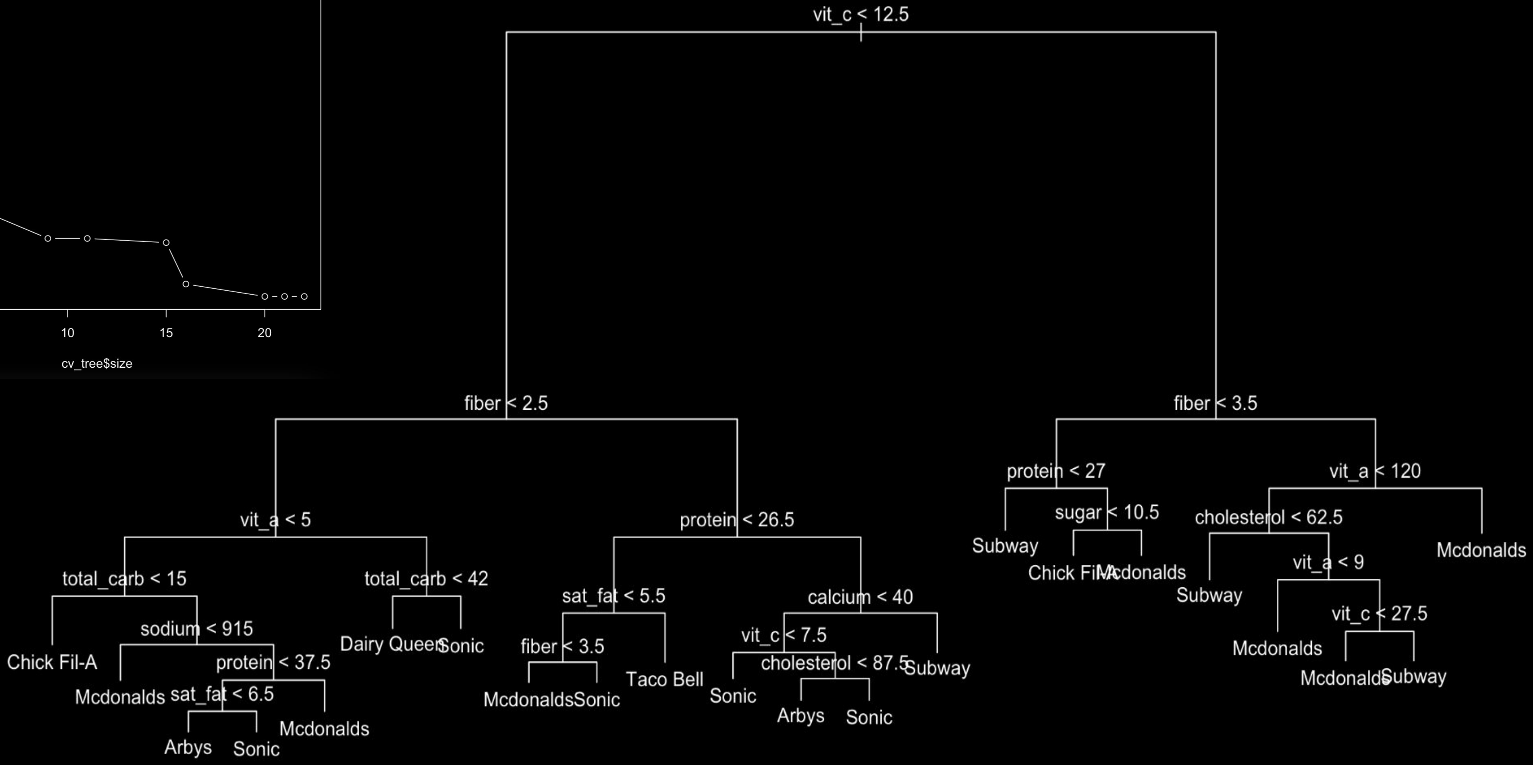
Classification



Decision trees are a type of classification technique that work by recursively splitting the data based on the values of predictor variables.

This technique is particularly useful when dealing with complex relationships between predictor variables and the target variable. However, it can be prone to overfitting and may not work well with noisy data.

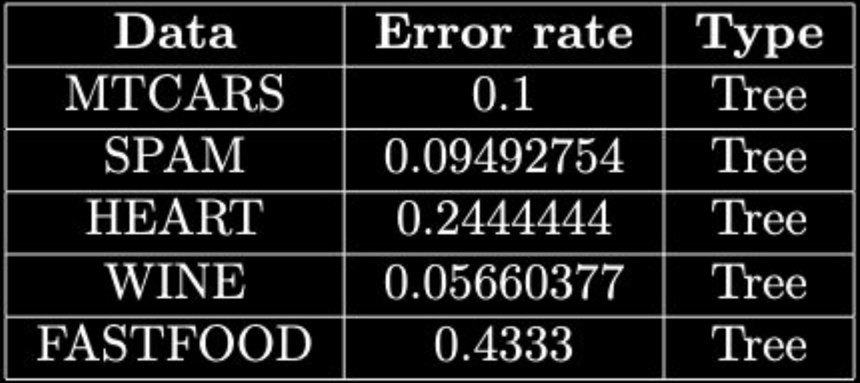

cv_tree$dev
140
160
180
200
5
10
15
20
cv_tree$size


vit_c < 12.5
fiber < 2.5
fiber < 3.5
vit_a < 5
protein < 26.5
protein < 27
vit_a < 120
total_carb < 15
total_carb < 42
sat_fat < 5.5
calcium < 40
sugar < 10.5
cholesterol < 62.5
Mcdonalds
Subway
Chick Fil-A
Mcdonalds
vit_a < 9
Subway
sodium < 915
Dairy Queen
Sonic
fiber < 3.5
Taco Bell
vit_c < 7.5
Subway
vit_c < 27.5
Mcdonalds
Chick Fil-A
protein < 37.5
Mcdonalds
sat_fat < 6.5
Mcdonalds
Sonic
Sonic
cholesterol < 87.5
Mcdonalds
Subway
Mcdonalds
Arbys
Sonic
Arbys
Sonic

Justin Newman

Decision Trees

Classification



Data	Error rate	Type
MTCARS	0.1	Tree
SPAM	0.09492754	Tree
HEART	0.2444444	Tree
WINE	0.05660377	Tree
FASTFOOD	0.4333	Tree

Decision trees are a type of classification technique that work by recursively splitting the data based on the values of predictor variables.

This technique is particularly useful when dealing with complex relationships between predictor variables and the target variable. However, it can be prone to overfitting and may not work well with noisy data.

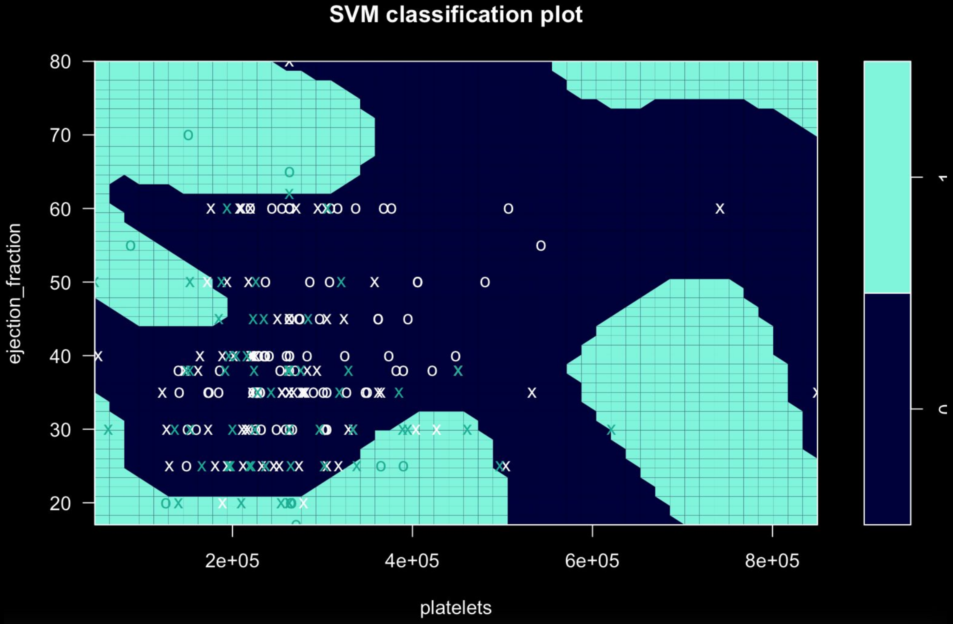
Justin Matthew Newman

SVM
Classification

Support Vector Machine (SVM) is a popular machine learning algorithm used for classification tasks. It is based on the idea of finding a hyperplane that separates two classes of data points with the maximum margin.

We leveraged a tuning process using cross-validation to find the optimal cost and gamma parameters.

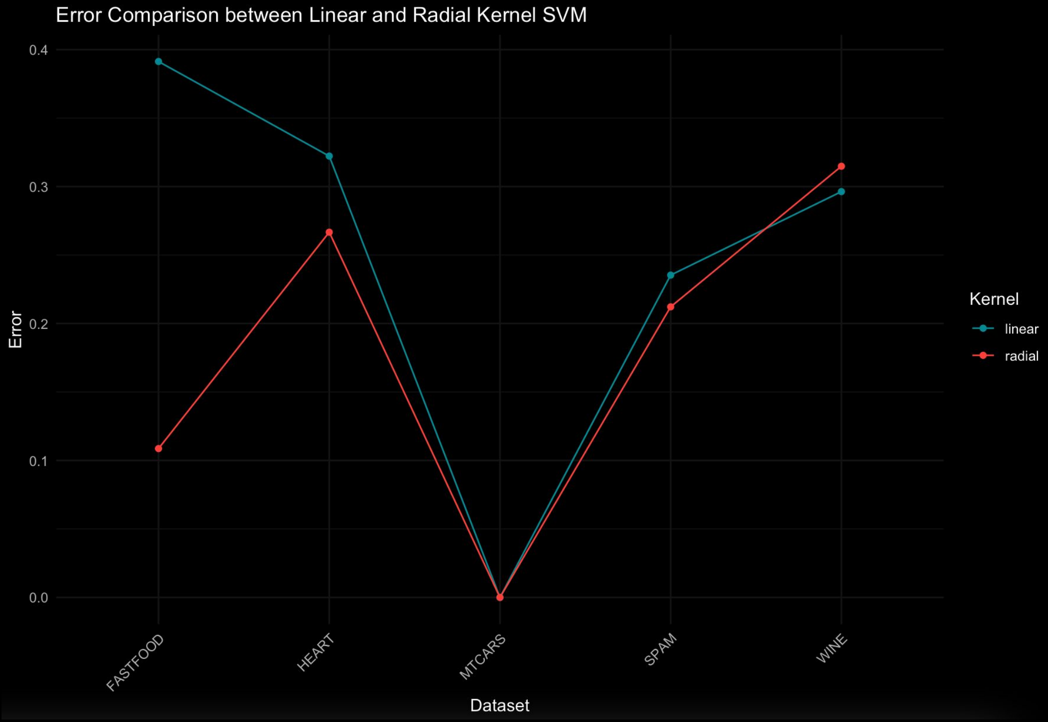
Increasing the cross-validation folds, expanding the cost and gamma values, and considering different kernels can also help improve the accuracy.



Justin Matthew Newman

SVM

Classification



Data	Error/MSE	Type	Type2
MTCARS	0	SVM	linear
SPAM	0.2353367	SVM	linear
HEART	0.3222222	SVM	linear
WINE	0.2962963	SVM	linear
FASTFOOD	0.3913043	SVM	linear
MTCARS	0	SVM	radial
SPAM	0.2121651	SVM	radial
HEART	0.2666667	SVM	radial
WINE	0.3148148	SVM	radial
FASTFOOD	0.1086957	SVM	radial

Table 2: Results of SVM with linear and radial kernel on various datasets

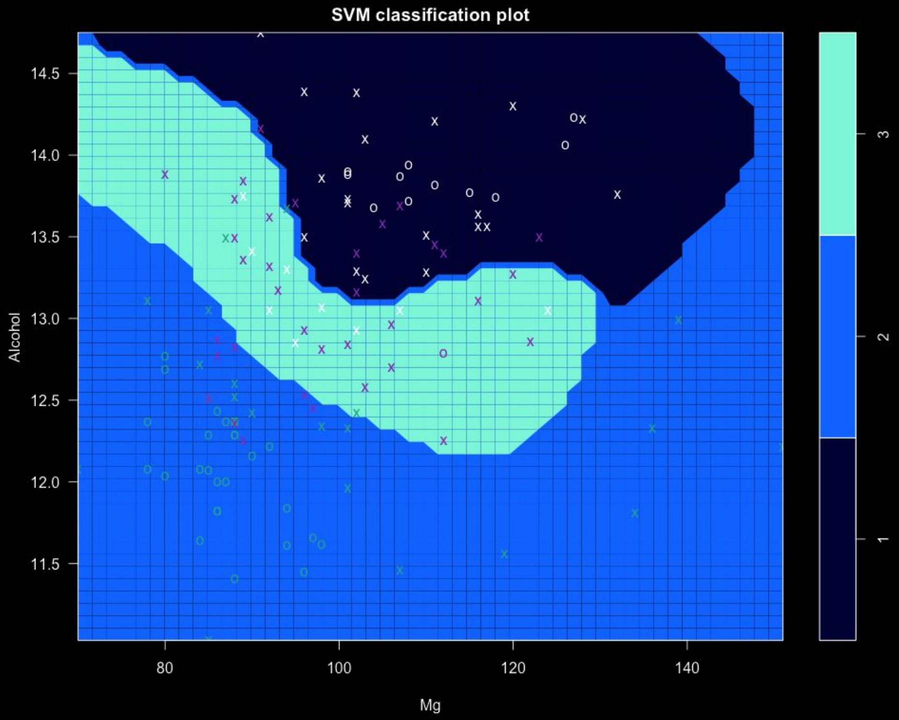
More

Justin Matthew Newman

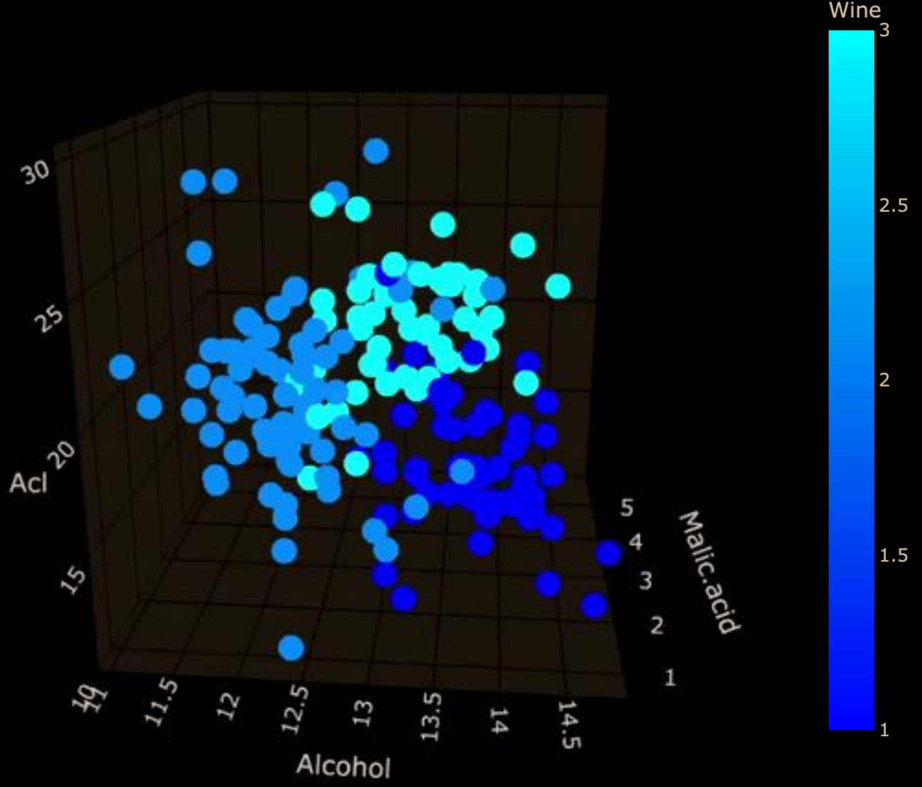
SVM

Classification

```
train_index <- sample(1:nrow(mtcars_subset), 0.7 * nrow(mtcars_subset))
train_data <- mtcars_subset[train_index, ]
test_data <- mtcars_subset[-train_index, ]
# Train the SVM classifier using a linear kernel
svmfit <- svm(am ~ ., data = train_data,  kernel = "linear", cost = 10,  scale = FALSE)
# Perform cross-validation to choose the right cost
tune_out <- tune(svm,  am ~ ., data = train_data, kernel = "linear", ranges = list(cost = c(0.001, 0.01, 0.1,
1, 5, 10, 100)))
# Get the best model
bestmod <- tune_out$best.model
# Make predictions on the test data
test_pred <- predict(bestmod, test_data)
# Display the confusion matrix
cm_linear <- table(true = test_data$am, predicted = test_pred)
print(cm_linear)
# Calculate the error rate for the linear kernel
error_rate_linear <- 1 - sum(diag(cm_linear)) / sum(cm_linear)
cat("Error rate (linear kernel):", error_rate_linear, "\n")
# Plot the decision boundary of the best model
plot(bestmod, train_data)
# Train the SVM classifier using a radial kernel
svmfit <- svm(am ~ ., data = train_data, kernel = "radial", gamma = 1, cost = 1)
# Perform cross-validation to choose the right cost and gamma
tune_out <- tune(svm,  am ~ ., data = train_data,  kernel = "radial",  ranges = list(cost = c(0.1, 1, 10, 100,
1000), gamma = c(0.5, 1, 2, 3, 4)))
# Get the best model
bestmod <- tune_out$best.model
# Make predictions on the test data
test_pred <- predict(bestmod, test_data)
# Display the confusion matrix
cm_radial <- table(true = test_data$am, predicted = test_pred)
print(cm_radial)
# Calculate the error rate for the radial kernel
error_rate_radial <- 1 - sum(diag(cm_radial)) / sum(cm_radial)
cat("Error rate (radial kernel):", error_rate_radial, "\n")
# Plot the decision boundary of the best model
plot(bestmod, train_data)
```



Machine Learning using R studio



WINE DATA

The UCI Wine dataset is a well-known dataset in machine learning and data analysis. It contains the results of a chemical analysis of wines from three different cultivars in the same region in Italy. The data was collected in 1984 and contains 178 instances, each representing a different wine. Each instance in the dataset has 13 features or attributes that describe the chemical composition of the wine, including alcohol content, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavonoids, non-flavonoid phenols, proanthocyanidins, color intensity, and hue of diluted wines.

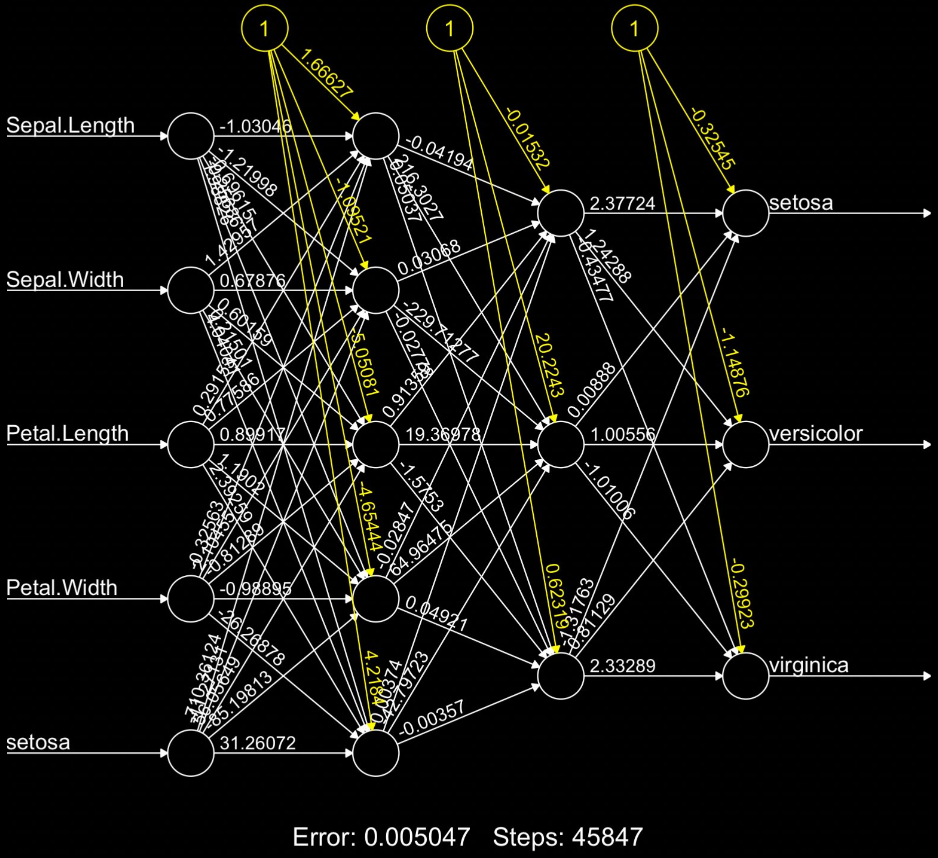
DATA

More

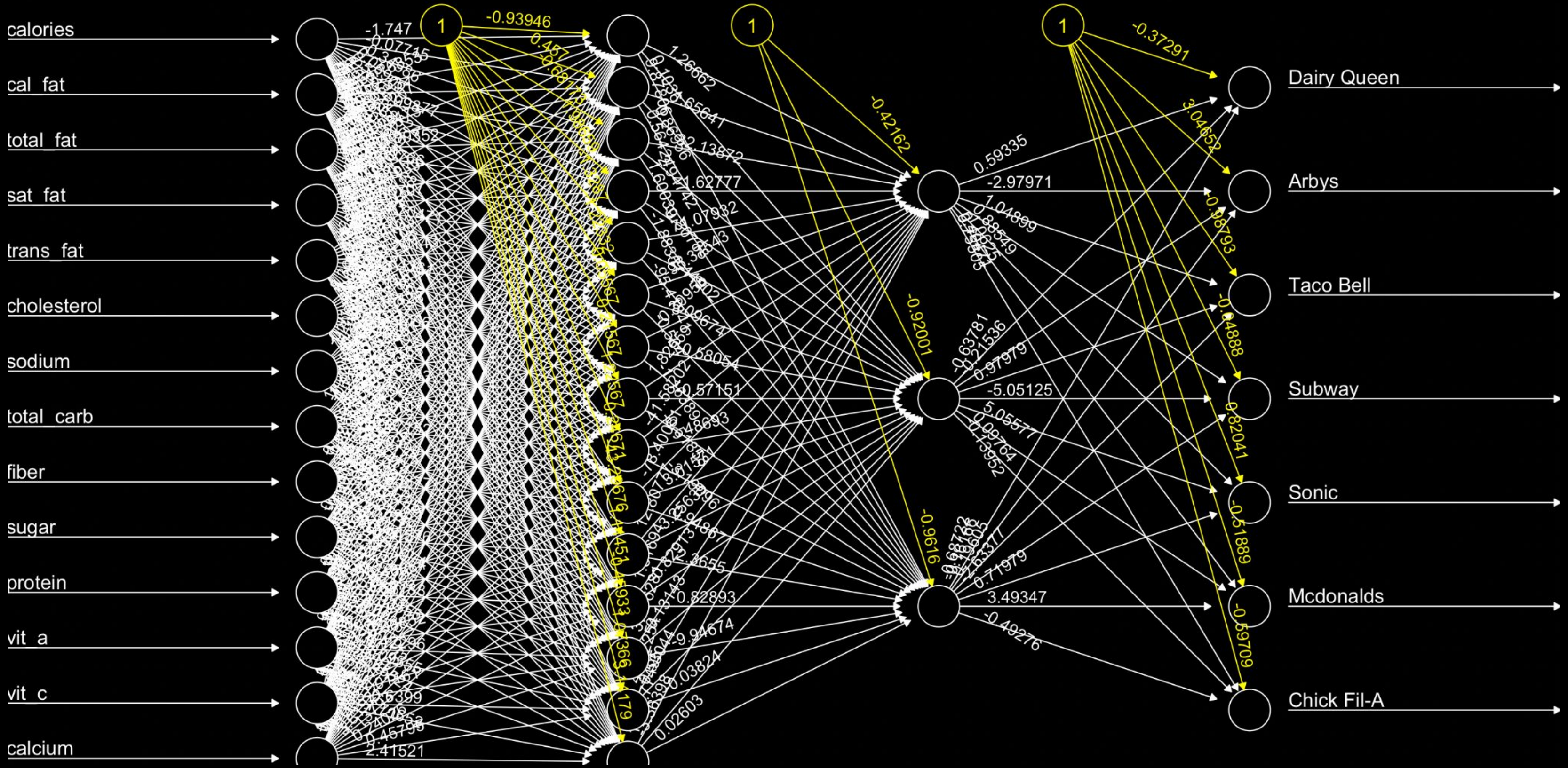
Justin Matthew
Newman

Neural Networks

I defined our neural network model. We used a simple three-hidden-layer architecture with four neurons in the first hidden layer and then three neurons in the remaining hidden layers:



Machine Learning using R studio



BOOSTING

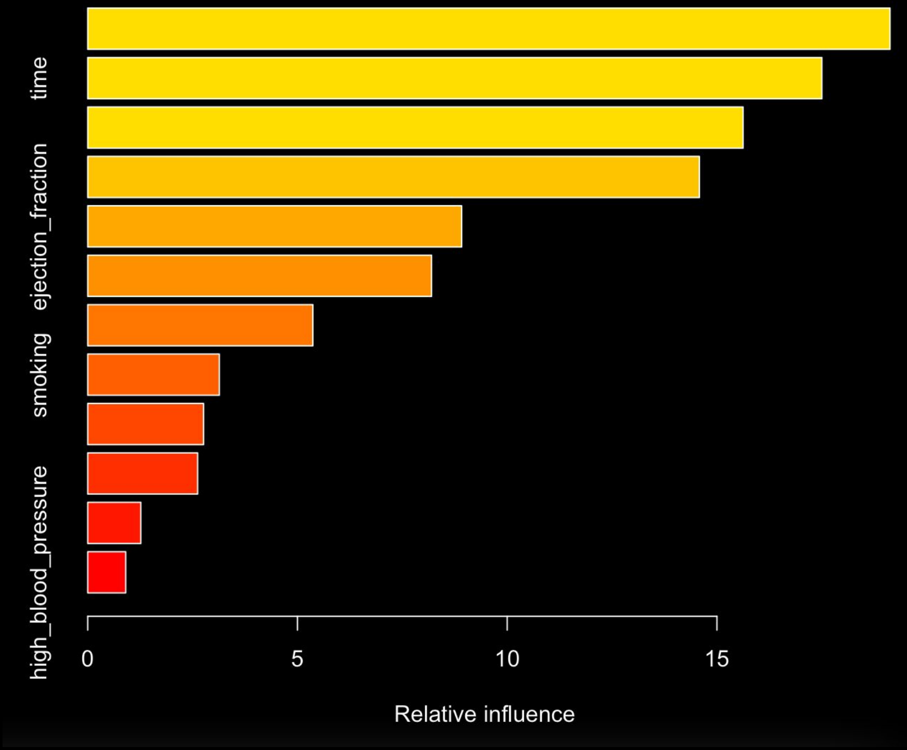
Regression

Gradient (boosting) regression is a machine learning technique that builds multiple models sequentially from the original model in an effort to make each subsequent model better. For the regression models that were originally ran, a boosting regression was ran on the model with the exception of the mtcars model.

For kern labs spam, heart failure, wine, and fast food data sets, the response variables were “free,” “platelets,” “Malic.acid,” and “calories,” respectively.

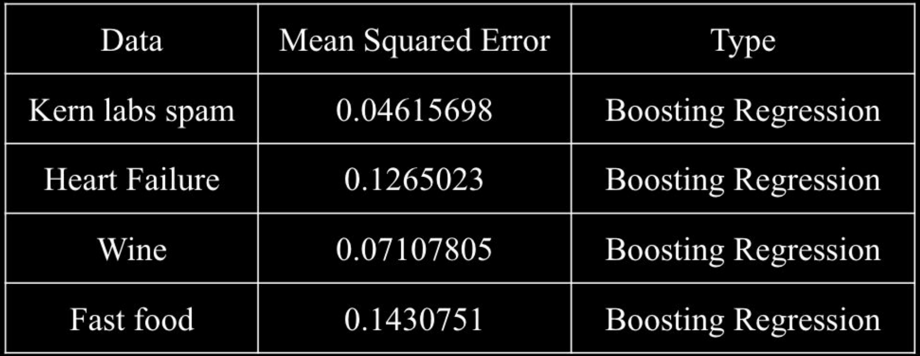
About Boosting

Boosting Explained



Boosting

Regression



Data	Mean Squared Error	Type
Kern labs spam	0.04615698	Boosting Regression
Heart Failure	0.1265023	Boosting Regression
Wine	0.07107805	Boosting Regression
Fast food	0.1430751	Boosting Regression

The boosting regression technique was run on these four data sets and their mean squared errors compared. The mtcars data set was not ran through boosting regression because the sample size was too small. MSE is a measure of how much deviance there is between the predicted values for the response variables and the actual values.

Based on the MSE results, the kern labs spam data set had the least amount of deviance and the fast food data set had the most amount of deviance.

About Boosting

Boosting Explained

Boosting code for regression model

<u>Kern labs spam</u>

```
set.seed(1)

kern_train = kern %>%
  sample_frac(0.5)
kern_test = kern %>%
  setdiff(kern_train)

set.seed(1)
boost_kern = gbm(free~.,
                 data=kern_train,
                 distribution = "gaussian",
                 n.trees=5000,
                 interaction.depth = 4)
summary(boost_kern)

boost_estimate = predict(boost_kern,
                      newdata= kern_test,
                      n.trees = 5000)

mean((boost_estimate - kern_test$free)^2)

boost_kern2 = gbm(free~., data = kern_train,
                  distribution = "gaussian",
                  n.trees = 5000,
                  interaction.depth = 4,
                  shrinkage = 0.01,
                  verbose = F)

boost_estimate2 = predict(boost_kern2, newdata = kern_test, n.trees = 5000)
mean((boost_estimate2 - kern_test$free)^2)
```

This code is using the Gradient Boosting Machine (GBM) algorithm for a regression problem on the kern dataset from the Kern Labs spam dataset. The objective is to predict the variable called free in the dataset.

distribution="gaussian": Specifies that the target variable is continuous, and the Gaussian distribution is used for the boosting algorithm.

n.trees=5000: The number of trees (boosting iterations) in the model.

interaction.depth=4: The maximum depth of each tree.

CODE:

https://github.com/JustinMatthewNewman/machine_learning_R

Questions?

Comments?