Week 3 Code Examples, CSC 400, Spring 2024

1. Decision Trees
2. Random Forest
3. Visualization

Decision Trees

First, we'll load the packages we'll need for the data and some cleaning. You may need to install some of these packages.

```
library(MASS)
data(Boston)
library(tidymodels)
library(tidyr)
```

Next, we'll take a look at the data.

```
boston_data_long <- Boston %>%
  pivot_longer(cols = everything(),
               names_to = "variable",
               values_to = "value")
boston_histograms <- ggplot(boston_data_long, aes(x = value)) +
  geom_histogram(bins = 30, color = "black", fill = "lightblue") +
  facet_wrap(~variable, scales = "free", ncol = 4) +
  labs(title = "Histograms of Numeric Variables in the Boston Housing Dataset",
       x = "Value",
       y = "Frequency") +
  theme_minimal()
print(boston_histograms)
```

We set a seed for reproducibility, and then we split the data into training and testing sets.

```
set.seed(123)
data_split <- initial_split(Boston, prop = 0.75)
train_data <- training(data_split)
test_data <- testing(data_split)
```

Because the data in this dataset is all numerical, we'll use regression mode in this example, but the code for classification is similar.

```
tree_spec <- decision_tree() %>%
  set_engine("rpart") %>%
  set_mode("regression")

tree_fit <- tree_spec %>%
  fit(medv ~ ., data = train_data)
```

Now we make predictions for the test data and look at how well the model did.

```r
predictions <- tree_fit %>%
  predict(test_data) %>%
  pull(.pred)

metrics <- metric_set(rmse, rsq)
model_performance <- test_data %>%
  mutate(predictions = predictions) %>%
  metrics(truth = medv, estimate = predictions)

print(model_performance)
```

If we had new data we wanted to make a prediction for, we make a tribble (a type of dataframe used in the tidyverse) and test it.

```r
new_data <- tribble(
  ~crim, ~zn, ~indus, ~chas, ~nox, ~rm, ~age, ~dis, ~rad, ~tax, ~ptratio, ~black, ~lstat,
  0.03237, 0, 2.18, 0, 0.458, 6.998, 45.8, 6.0622, 3, 222, 18.7, 394.63, 2.94
)
predictions <- predict(tree_fit, new_data)
print(predictions)
```

We can begin to visualize the results by looking at which variables were the most important in making the decision.

```r
library(vip)
var_importance <- vip::vip(tree_fit, num_features = 10)
print(var_importance)
```

We can also plot the graph of the tree and list the rules that created the tree.

```r
rpart.plot(tree_fit$fit, type = 4, extra = 101, under = TRUE, cex = 0.8, box.palette = "auto")

rules <- rpart.rules(tree_fit$fit)
print(rules)
```

We can look at an example using the rpart package.

```r
library(ISLR)
library(rpart)
library(rpart.plot)
tree <- rpart(Salary ~ Years + HmRun, data=Hitters, control=rpart.control(cp=.0001))
best <- tree$cptable[which.min(tree$cptable[,"xerror"]),"CP"]
pruned_tree <- prune(tree, cp=best)
prp(pruned_tree)
```

See reference [4] for examples with classification, and more examples with pruning trees, and plotting elbow plots and ROC curves.

Random Forest

We'll start by loading packages and the iris dataset.

```
library(randomForest)
library(datasets)
library(caret)

data<-iris
str(data)
data$Species <- as.factor(data$Species)
table(data$Species)
```

We split into test and training data.

```
set.seed(222)
ind <- sample(2, nrow(data), replace = TRUE, prob = c(0.7, 0.3))
train <- data[ind==1,]
test <- data[ind==2,]
```

We can then create our random forest model. The output here includes the confusion matrix for our classification predictions. In this case, we are using the default settings of 500 trees and two variables tried at each split. These parameters can be adjusted and that may be needed for large datasets with many more variables.

```
rf <- randomForest(Species~., data=train, proximity=TRUE)
print(rf)
```

Examine the confusion matrices for both the test and training data.

```
p1 <- predict(rf, train)
confusionMatrix(p1, train$ Species)

p2 <- predict(rf, test)
confusionMatrix(p2, test$ Species)
```

We can plot a number of graphs to help us assess the model.

```
plot(rf)
partialPlot(rf, train, Petal.Width, "setosa")
MDSplot(rf, train$Species)

hist(treesize(rf),
     main = "No. of Nodes for the Trees",
     col = "green")

varImpPlot(rf,
           sort = T,
           n.var = 10,
           main = "Top 10 - Variable Importance")
importance(rf)
```

See the documentation for the rpart and randomForest packages to see how to adjust the parameters.

Resources:

1. https://www.datacamp.com/tutorial/decision-trees-R
2. https://www.guru99.com/r-decision-trees.html
3. https://www.statology.org/plot-decision-tree-in-r/
4. https://www.r-bloggers.com/2021/04/decision-trees-in-r/
5. https://www.r-bloggers.com/2021/04/random-forest-in-r/