Simple Linear Regression Algorithm

Using Rocket Propellant Data

Author : Rose Ellison

The Intuition

Linear regression is a type of supervised learning algorithm which predicts continuous values of a given data point by generalising on the data that we have in hand. The linear part indicates that we are using a linear approach in generalising over the data. Simple indicates we are describing the relationship between one dependent and one independent variable using a straight line.

$$\hat{y} = \beta_0 + \beta x_1$$

Step One : Split the data into test and training sets.

Step Two: Calculate the sum of squares

Step Three : Estimate coefficients using lest squares.

Step Four : Make Predictions based on the coefficients.

Step Five : Validate the model.

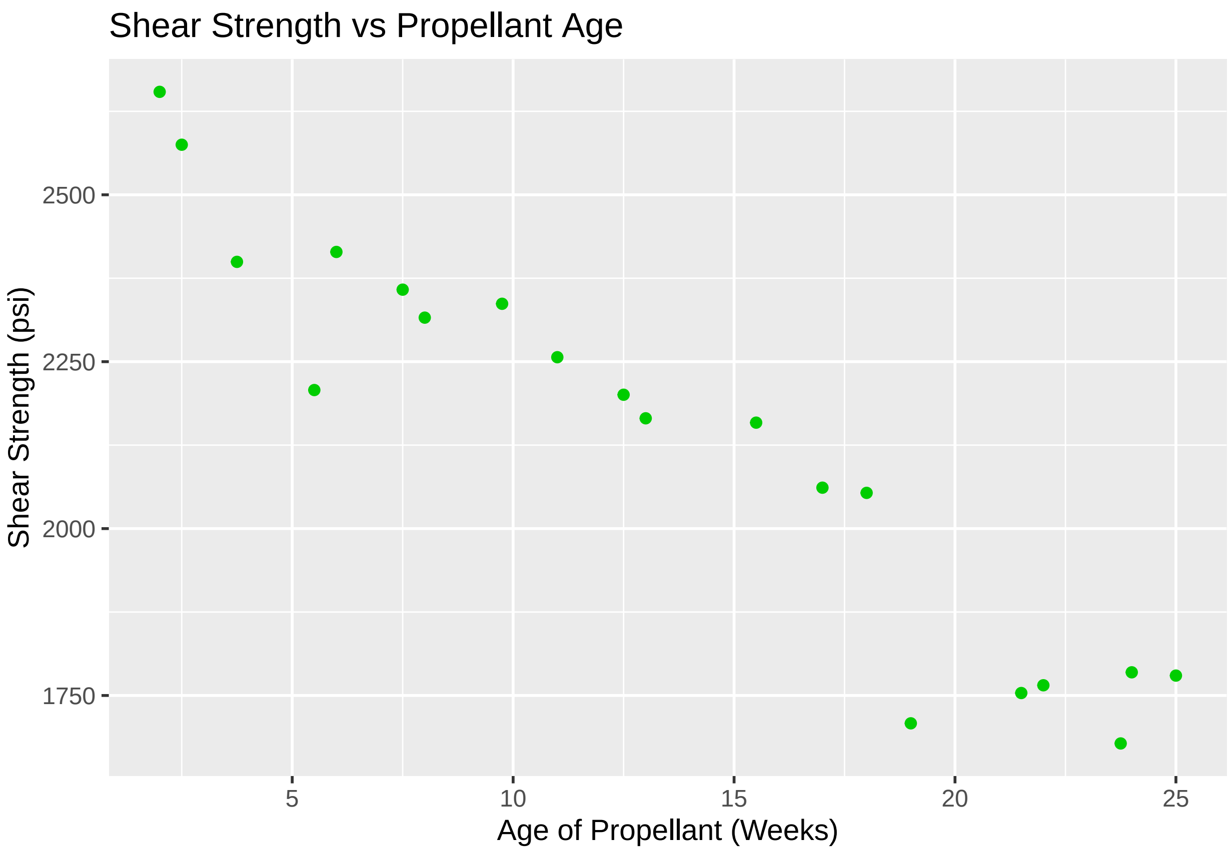
Exploration

Exploration process includes preprocessing and visualizing plots. This algorithm will be implemented on the rocket propellant data. There are two columns; age of rockets propellant(independent variable) and shear strength(continuous dependent variable). We will use simple linear regression to predict if the shear strength of future rocket propellants based on their age.

```r
# Read the data
data = read.csv('../../../_resources/data/Rocket_Prop.csv')
```

Plot

```
ggplot(data,aes(x = data[, 2],
                y = data[, 1])) +
  geom_point(col = 3) +
  labs(x = "Age of Propellant (Weeks)",
       y = "Shear Strength (psi)") +
  ggtitle("Shear Strength vs Propellant Age")
```



From the data, we can see there is a linear relationships between age of propellant and shear strength. For that reason, simple linear regression would be a good model to predict future data points.

Implementation

Step One :

Split the data into test and training sets.

```
# Splitting test and training sets
split <- sample.split(data[ , 1], SplitRatio = 0.6)
training <- subset(data, split == TRUE)
test <- subset(data, split == FALSE)
```

Step Two :

Find the sum of squares

$$sum\ of\ squares\ of\ S_{xx} = \sum(x^2) - \frac{\sum(x)^2}{n}$$

$$sum\ of\ squares\ of\ S_{xy} = \sum(xy) - \frac{\sum(x)\sum(y)}{n}$$

```
x <- training[2]
y <- training[1]
n <- nrow(x)
Sxx <- sum(x^2) - sum(x)^2/n
Sxy <- sum(x*y) - sum(x)*sum(y)/n
```

Step Three :

Estimate coefficients.

$$\beta_1 = \frac{S_{xy}}{S_{xx}}$$

$$\beta_0 = \bar{y} - \beta_1\ \bar{x}$$

```
# Calculate Least Squares Estimates
B1H <- Sxy / Sxx # Slope Estimate
B0H <- (sum(y) / n) - ((sum(x) / n)*B1H) # Intercept Estimate
```

Step Four : Make predictions

Put it all together and make predictions

$$\hat{y} = \beta_0 + \beta_1\ x$$

```
simple_lr <- function(training, test, y_index, x_index)
{
  x <- training[2]
  y <- training[1]
  n <- nrow(x)
  Sxx <- sum(x^2) - sum(x)^2/n
  Sxy <- sum(x*y) - sum(x)*sum(y)/n
```

```r
  B1H <- Sxy / Sxx # Slope Estimate
  B0H <- (sum(y) / n) - ((sum(x) / n)*B1H)

  x_test <- test[2]

  for (i in 1:nrow(test))
  {
    yhat <- B0H + B1H * x_test[i,]
    test$y_pred[i] <- yhat
  }

  return(test)
}

test<- simple_lr(training, test, 1, 2)
```

Results

Compare predicted results(y-pred column) to the actual y values(y column) on our test set.

```r
test[ , c(1, 3)]
```

```
##          y   y_pred
## 1  2158.70 2063.520
## 2  1678.15 1741.746
## 5  2207.50 2453.548
## 10 2256.70 2239.032
## 11 2165.20 2161.027
## 12 2399.55 2521.803
## 14 2336.75 2287.786
## 15 1765.30 1810.001
```