SVM Regression

I decided to use the bmw used car dataset again for regression as it has provided good results in the past. https://www.kaggle.com/datasets/adityadesai13/used-car-dataset-ford-and-mercedes

---

Lets go ahead and load in the data, divide it into train and test data, and factorize it.

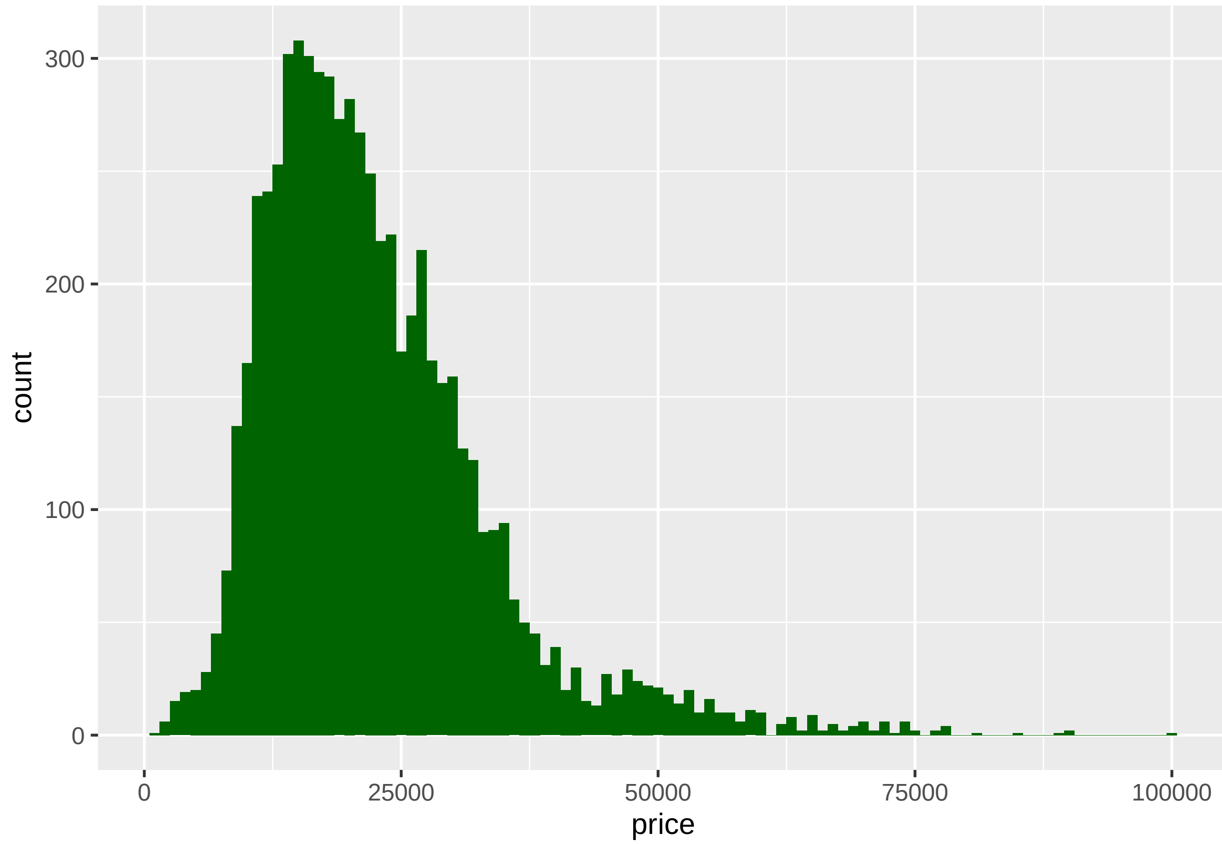
```
library(readr)
df <- read.csv("bmw.csv")

df$model <- as.factor(df$model)
df$transmission <- as.factor(df$transmission)
df$fuelType <- as.factor(df$fuelType)

set.seed(1)
groups <- c(train=.6, test=.2, validate=.2)
i <- sample(cut(1:nrow(df),nrow(df)*cumsum(c(0,groups)), labels=names(groups)))
train <- df[i=="train",]
test <- df[i=="test",]
vald <- df[i=="validate",]
```

Lets take a look at some of the data.

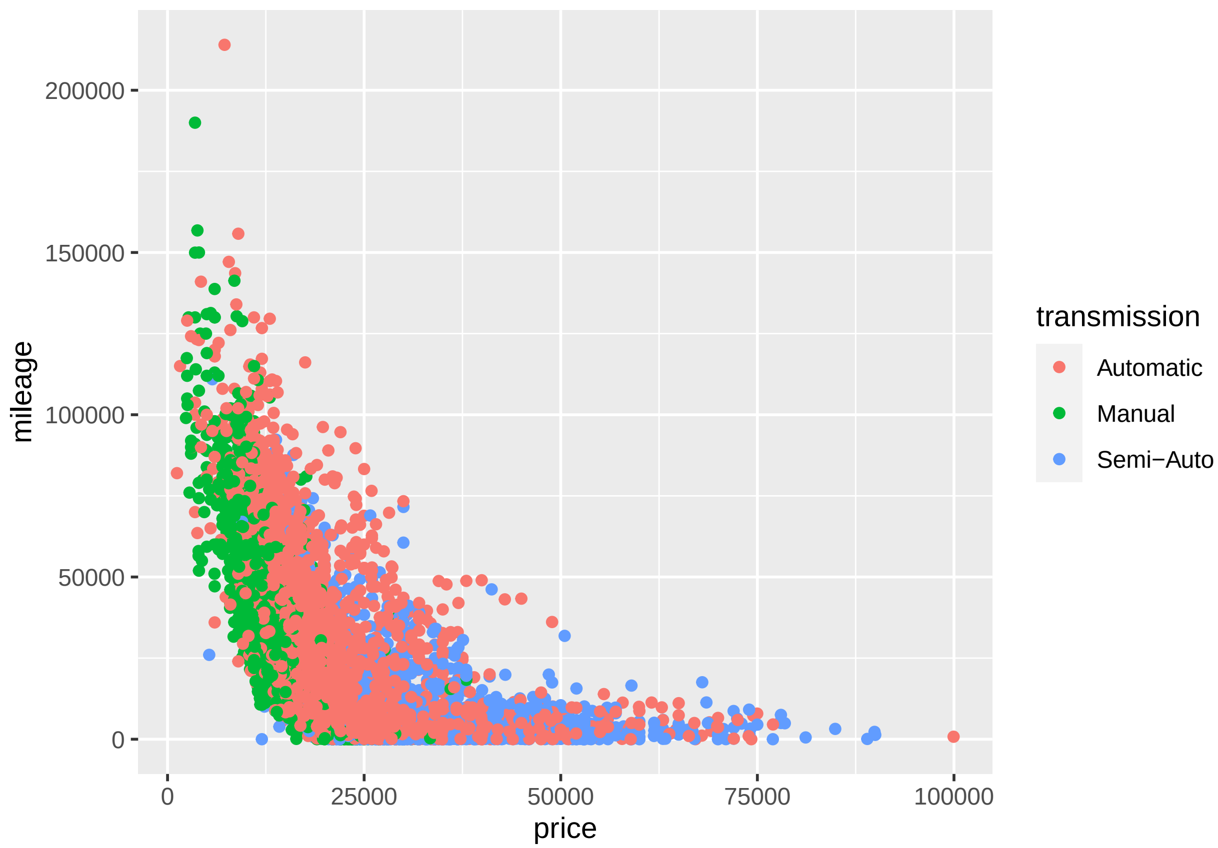
```
library(ggplot2)
head(train)
```

```
##        model year price transmission mileage fuelType tax  mpg engineSize
## 1   5 Series 2014 11200    Automatic   67068   Diesel 125 57.6        2.0
## 3   5 Series 2016 16000    Automatic   62794   Diesel 160 51.4        3.0
## 8   2 Series 2018 16250       Manual   10401   Petrol 145 52.3        1.5
## 10  5 Series 2016 14250    Automatic   36099   Diesel  20 68.9        2.0
## 12  1 Series 2017 11800       Manual   29840   Diesel  20 68.9        2.0
## 15        X3 2017 22000    Automatic   19057   Diesel 145 54.3        2.0
```

```
ggplot(train, aes(x = price)) + geom_histogram(fill="darkgreen", binwidth = 1000)
```




```
ggplot(train, aes(x = price, y = mileage)) + geom_point(aes(color = transmission))
```



Interestingly it seems that there might be some correlation between the type of transmission and the price. It seems that the manual cars tend to have lower prices despite maybe having lower mileage as well, where low mileage usually means higher price. This is followed by the automatic transmissions having the next higher price and the semi-auto with the best price.

Lets start with linear regression

First we will tune the hyper parameter C to find the best possible value.

```
library(e1071)
tune.out <- tune(svm, price~., data=vald, kernel="linear",
ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 23630283
##
## - Detailed performance results:
```

```
##    cost    error dispersion
## 1 1e-03 61449437   17919439
## 2 1e-02 41796289   15589739
## 3 1e-01 28223883   15341158
## 4 1e+00 23630283   16805083
## 5 5e+00 23793358   17092360
## 6 1e+01 23810869   17120243
## 7 1e+02 23833289   17179911
```

It seems the best c value is 1, so lets use it for the actual model now.

```
svm_linear <- svm(price~., data = train, kernel = "linear", cost = 1, scale = TRUE)
pred_linear <- predict(svm_linear, newdata = test)
cor(pred_linear, test$price)
```

```
## [1] 0.9338052
```

Looks like linear gets a correlation value of .933 which is quite good.

Now lets try out polynomial

We can tune the C hyper parameter the same way but for polynomial.

```
library(e1071)
tune.out <- tune(svm, price~., data=vald, kernel="polynomial",
ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 95684474
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 144148052   29988639
## 2 1e-02 140100022   29874489
## 3 1e-01 123973024   29270559
## 4 1e+00  95684474   29158914
## 5 5e+00 118598149  180686584
## 6 1e+01 159636339  334379429
## 7 1e+02 139099067  295603586
```

Polynomial had a best value of 100, so lets train the model using 100.

```
svm_poly <- svm(price~., data = train, kernel = "polynomial", cost = 100, scale = TRUE)
pred_poly <- predict(svm_poly, newdata = test)
cor(pred_poly, test$price)
```

```
## [1] 0.9638701
```

Polynomial gets a correlation value of .963 which is even better than linear and an amazing value.

Last we try radial

First we need to tune the C hyper parameter like before, but now we also need to tune gamma.

```
tune.out <- tune(svm, price~., data=vald, kernel="radial",
ranges=list(cost=c(0.001, 0.01, 0.1, 1, 5, 10, 100),
            gamma=c(.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     5   0.5
##
## - best performance: 18419729
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-03   0.5 134365501   31075114
## 2  1e-02   0.5  88442965   29681584
## 3  1e-01   0.5  37556984   23420182
## 4  1e+00   0.5  19392539   18654620
## 5  5e+00   0.5  18419729   17983828
## 6  1e+01   0.5  18797604   17965031
## 7  1e+02   0.5  21448376   17750152
## 8  1e-03   1.0 140818560   31084772
## 9  1e-02   1.0 116106601   30929869
## 10 1e-01   1.0  53945232   26169102
## 11 1e+00   1.0  23781780   18881043
## 12 5e+00   1.0  22395267   18181565
## 13 1e+01   1.0  22647633   18216508
## 14 1e+02   1.0  25367115   17571796
## 15 1e-03   2.0 142954002   31077677
## 16 1e-02   2.0 130538642   30929884
## 17 1e-01   2.0  81543141   29843770
## 18 1e+00   2.0  34482454   19549585
## 19 5e+00   2.0  29964181   18804904
## 20 1e+01   2.0  30358155   18773531
## 21 1e+02   2.0  33704805   18390627
## 22 1e-03   3.0 143330741   31073156
## 23 1e-02   3.0 133624390   30835807
## 24 1e-01   3.0  90942722   30386154
## 25 1e+00   3.0  42296461   20629327
## 26 5e+00   3.0  35302796   19287202
## 27 1e+01   3.0  35767988   19275202
## 28 1e+02   3.0  39836936   19278808
## 29 1e-03   4.0 143486421   31067904
```

```
## 30 1e-02   4.0 134918480   30849986
## 31 1e-01   4.0  95461005   30389511
## 32 1e+00   4.0  47004767   21259124
## 33 5e+00   4.0  39074511   19496033
## 34 1e+01   4.0  39417619   19464308
## 35 1e+02   4.0  43193192   19428237
```

The best value for cost is 5 and gamma is 0.5, so we will use these parameters for the model.

```
svm_rad <- svm(price~., data = train, kernel = "radial", cost = 5, gamma = 0.5, scale = TRUE)
pred_rad <- predict(svm_rad, newdata = test)
cor(pred_rad, test$price)
```

```
## [1] 0.9777572
```

Radial got a correlation value of roughly .978 which is the best so far and overall an amazing predictor.

Analysis

The first thing that should be noted is that all three of the kernel methods got amazing correlation values. However, as we moved to higher dimensions the correlation value increased each time. The first increase from linear to polynomial was the biggest by about 3 percent and then another increase of about 1.5 percent from polynomial to radial. The fact that polynomial did better than linear suggests that the data can be better split when considering more dimensions to the data. Using the polynomial kernel methods we were able to improve, and with radial we are able to control the variance and bias of the fitting. By having the model use a low gamma value it probably had higher bias and lower variance, which also improved the model slightly.