Homework #7

Eric Pettengill

```r
library(ISLR)
library(tidyverse)
library(caret)
library(tree)
library(randomForest)
```

8.4.8 (a), (b), (d), (e)

Using the `Carseats` data predict `Sales` using regression trees.

(a) Train/Test split

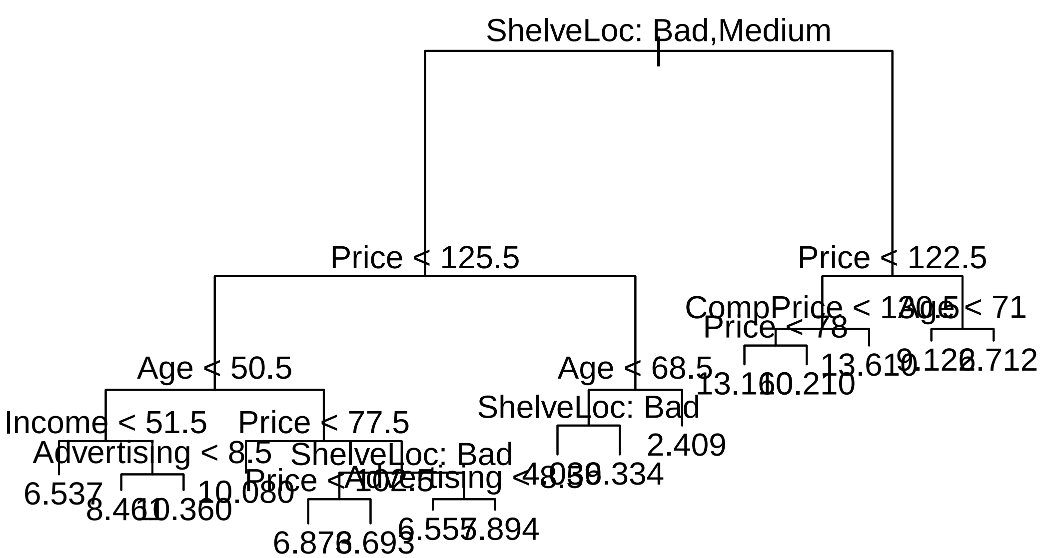
```r
set.seed(1126)

train <- sample(1:nrow(Carseats), trunc(nrow(Carseats)/2))
```

(b) Fit a regression tree to the training set. Plot the tree and calculate the test MSE.

```r
sales.tree <- tree(Sales ~ ., subset = train, data = Carseats)
```

```r
plot(sales.tree)
text(sales.tree, pretty = 0)
```



The plot is difficult to interpret since the variable names are overlapping but price is most important in determining sales.

```r
sales.preds <- predict(sales.tree, Carseats[-train, ])

(sales.mse <- mean((sales.preds - Carseats[-train, 1])^2))
```

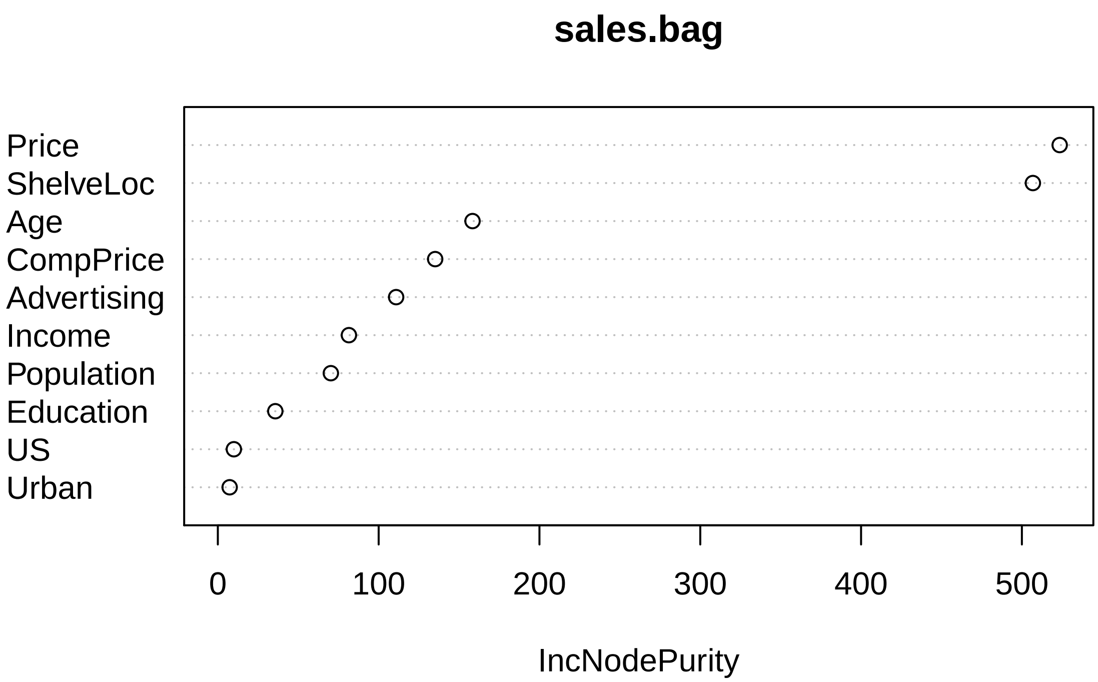
```
## [1] 4.689165
```

(d) Use the bagging approach. Calculate the test MSE. Use the `importance()` function to determine which variables are most important.

```
sales.bag <- randomForest(Sales ~ ., subset = train, data = Carseats, mtry = ncol(Carseats)-1)
sales.bag.pred <- predict(sales.bag, Carseats[-train, ])
(sales.bag.mse <- mean((sales.bag.pred - Carseats[-train, 1])^2))
```

```
## [1] 2.129456
```

```
varImpPlot(sales.bag)
```



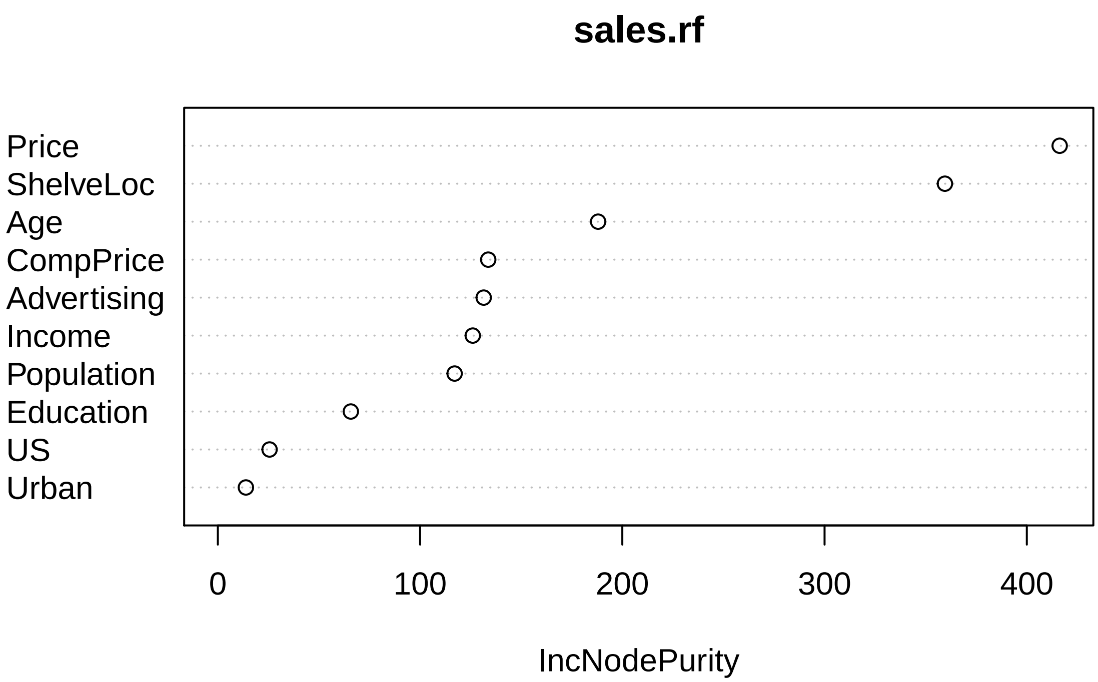
(e) Use random forests. Calculate the test MSE. Use the `importance()` function to determine which variables are most important. Describe the effect of m, the number of variables considered at each split, on the error rate obtained.

```
sales.rf <- randomForest(Sales ~ .,
                         subset = train,
                         data = Carseats,
                         mtry = round(sqrt(ncol(Carseats) - 1)))

sales.rf.pred <- predict(sales.rf, Carseats[-train, ])
(sales.rf.mse <- mean((sales.rf.pred - Carseats[-train, 1])^2))
```

```
## [1] 2.628759
```

```
varImpPlot(sales.rf)
```



MSE table

	m	MSE
1	1.00	4.69
2	10.00	2.13
3	3.00	2.63

As we can see as the number of splits, m, increase the test MSE decreases.

9.7.7 (a), (b), (c)

Using the `Auto` data set predict whether a given car gets high or low gas mileage.

(a) Create a binary variable (1 = cars above median mileage, 0 = cars below median mileage)

```
auto <- Auto %>%
  mutate(mpg = as.factor(if_else(mpg > median(mpg), 1, 0)),
         origin = as.factor(origin),
         cylinders = as.factor(cylinders)) %>%
  select(-name)
```

(b) Fit support vector classifier cross validating the value of `cost`. Report the cross-validation errors. Comment.

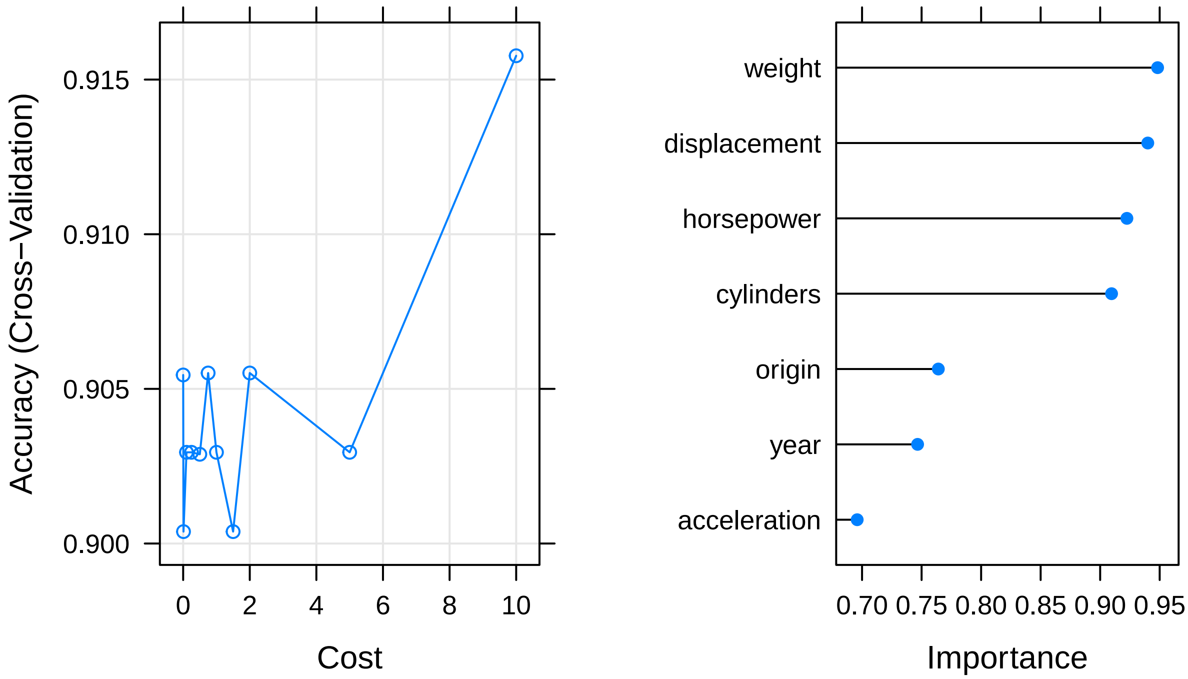
```
trControl <- trainControl(method = "cv",
                          number = 5)

svm.linear <- train(mpg ~ .,
                    method = "svmLinear",
                    trControl = trControl,
                    tuneGrid = expand.grid(C = c(0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5, 10)
                                           ),
```

```
                    tuneLength = 10,
                    data = auto)

importance <- varImp(svm.linear, scale=FALSE)

gridExtra::grid.arrange(plot(svm.linear), plot(importance), ncol = 2)
```



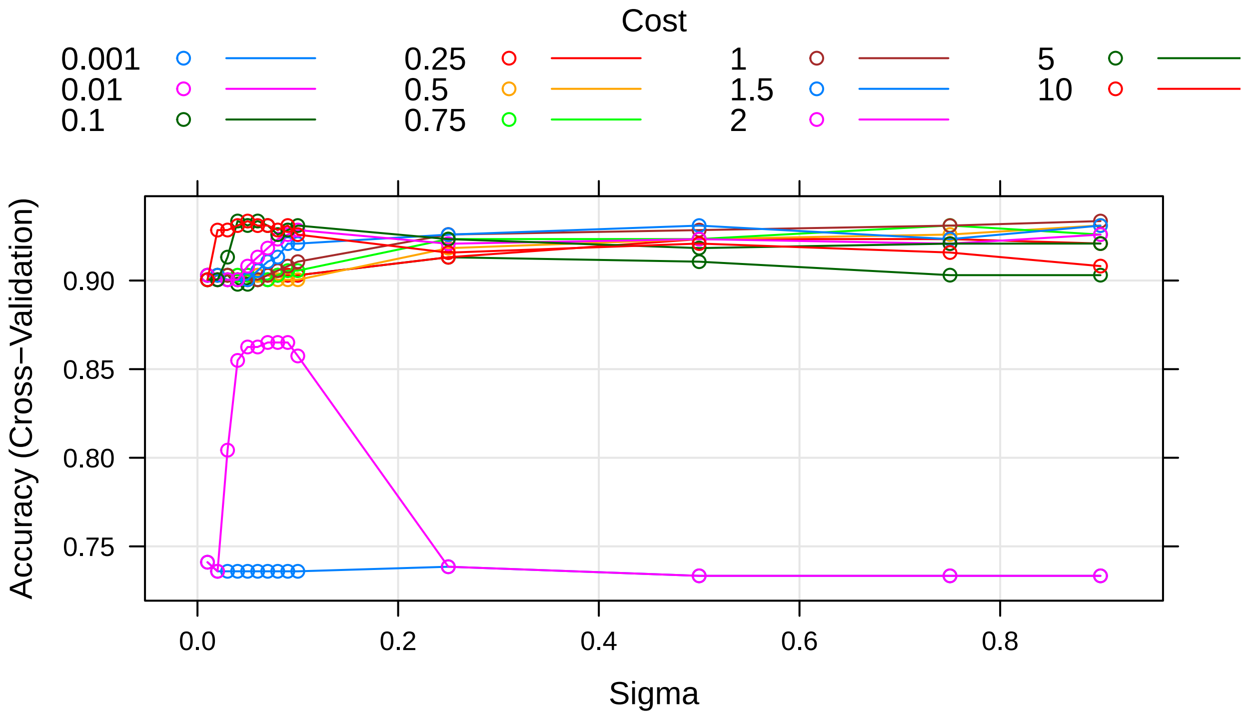
As we can see above, as C increases the error rate decreases, or the prediction accuracy increases. Also, weight, displacement, horsepower, and cylinders are the most important variables, which intuitively make sense. As those variables increase, the more likely a car is to have poor gas mileage.

(c) Repeat (b) with radial(`gamma`) and polynomial(`degree`) basis kernels.

Radial

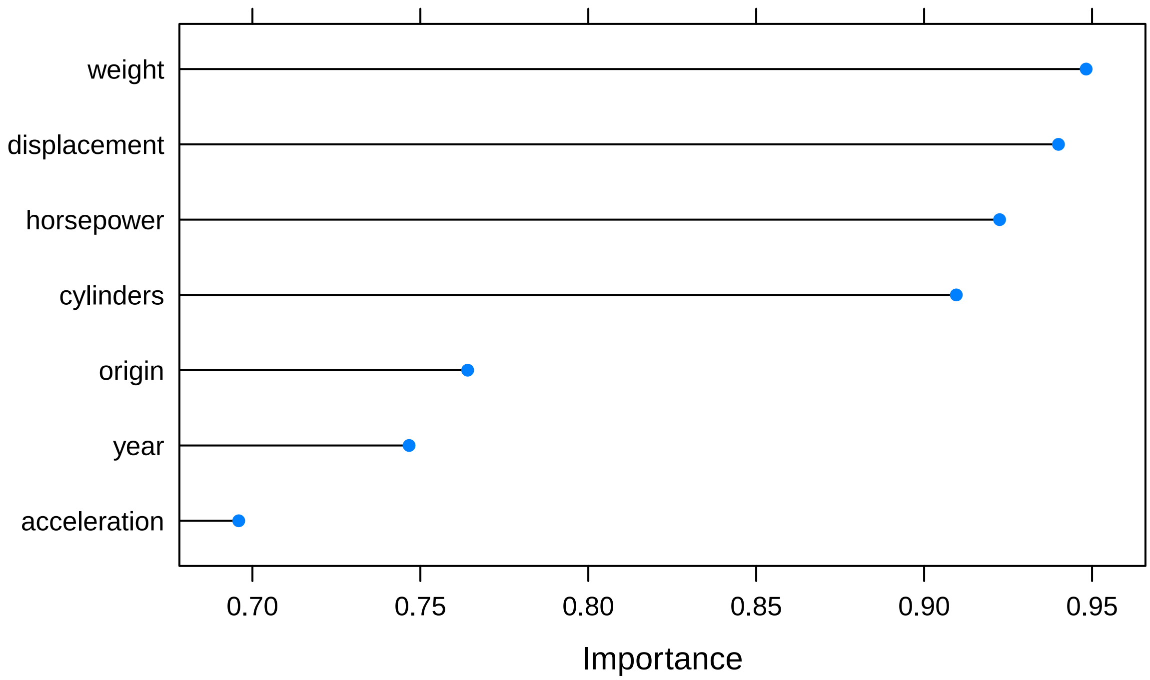
```
svm.radial <- train(mpg ~ .,
                    method = "svmRadial",
                    trControl = trControl,
                    tuneGrid = expand.grid(
                      C = c(0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5, 10),
                      sigma = c(.01, .02, .03, .04, .05, .06, .07, .07, .08, .09, .1, .25, .5, .75, .9)
                      ),
                    tuneLength = 10,
                    data = auto)

plot(svm.radial)
```




```
svm.radial$bestTune
##     sigma C
## 132  0.06 5

importance <- varImp(svm.radial, scale=FALSE)
plot(importance)
```

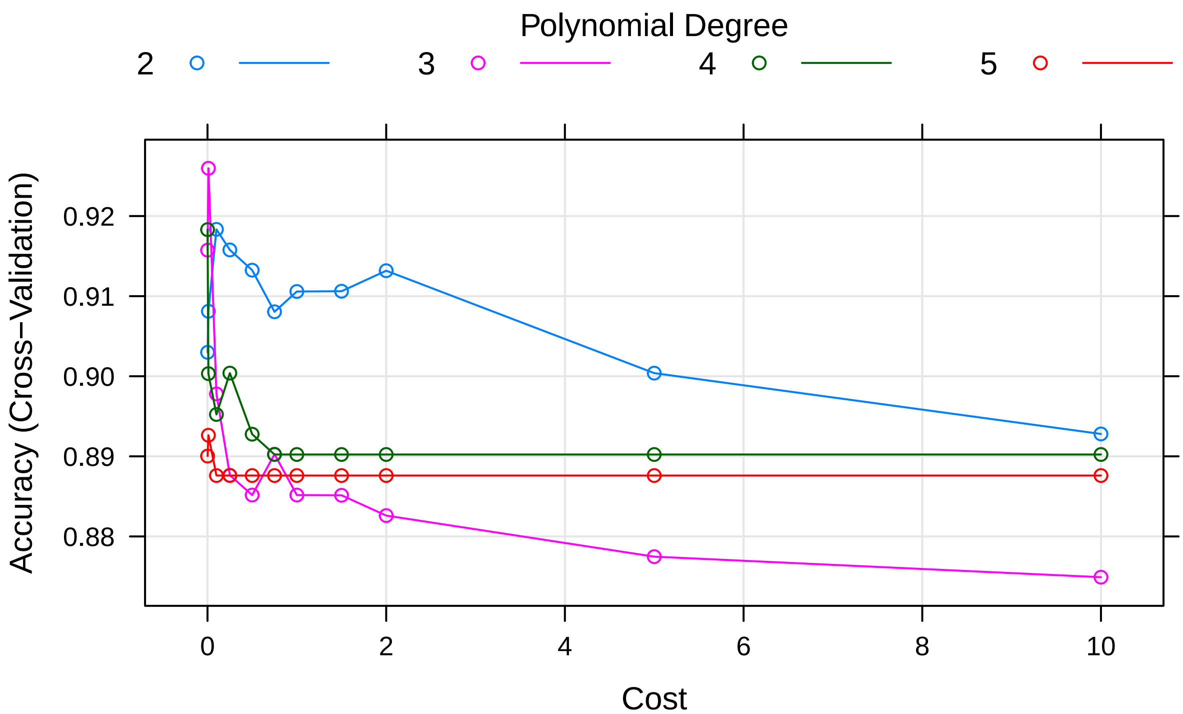


The optimal parameters given by cross-validation are $C = 5$ and $sigma/gamma = 0.06$.

Polynomial

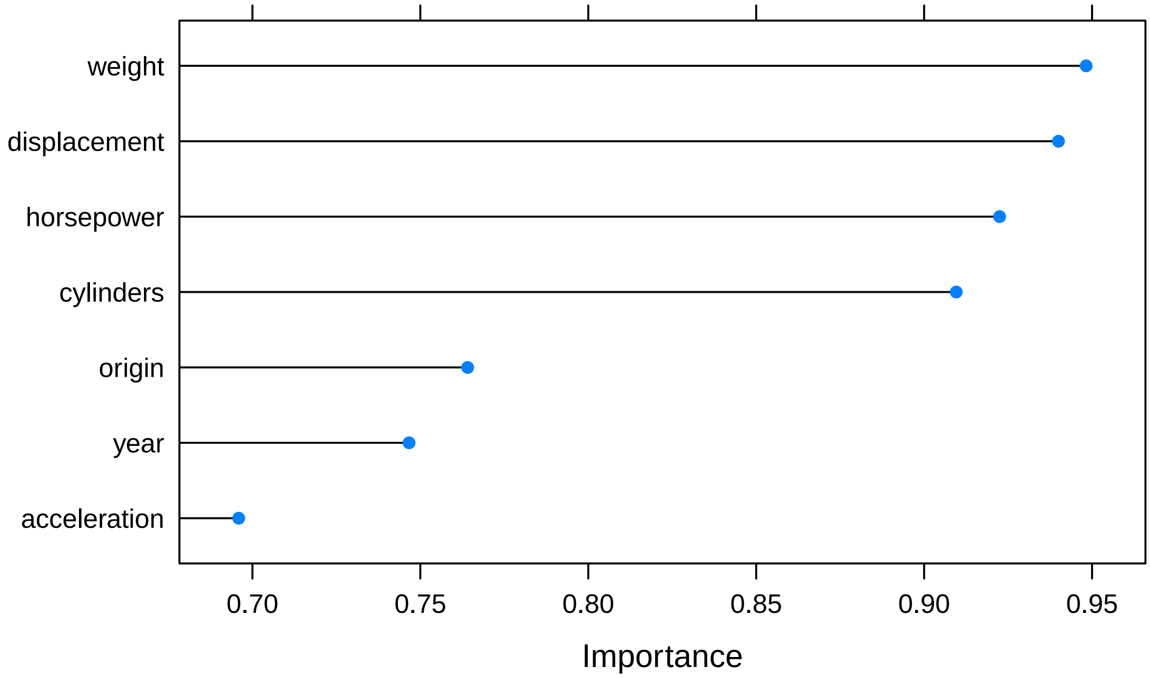
```
svm.poly <- train(mpg ~ .,
                  method = "svmPoly",
                  trControl = trControl,
                  tuneGrid = expand.grid(
                    C = c(0.001, 0.01, 0.1, 0.25, 0.5, 0.75, 1, 1.5, 2, 5, 10),
                    degree = 2:5,
                    scale = 1),
                  tuneLength = 10,
                  data = auto)

plot(svm.poly)
```




```
svm.poly$bestTune
##   degree scale    C
## 6      3     1 0.01

importance <- varImp(svm.poly, scale=FALSE)
plot(importance)
```



The optimal parameters given by cross-validation are $d = 3$ and $C = 0.01$.