Author Name : SRAVAN KUMAR VASAM
R & D Project : Machine Learning Regression models
Technologies : R version 4.0.2, Rstudio, Linux
Year of submission : 2020

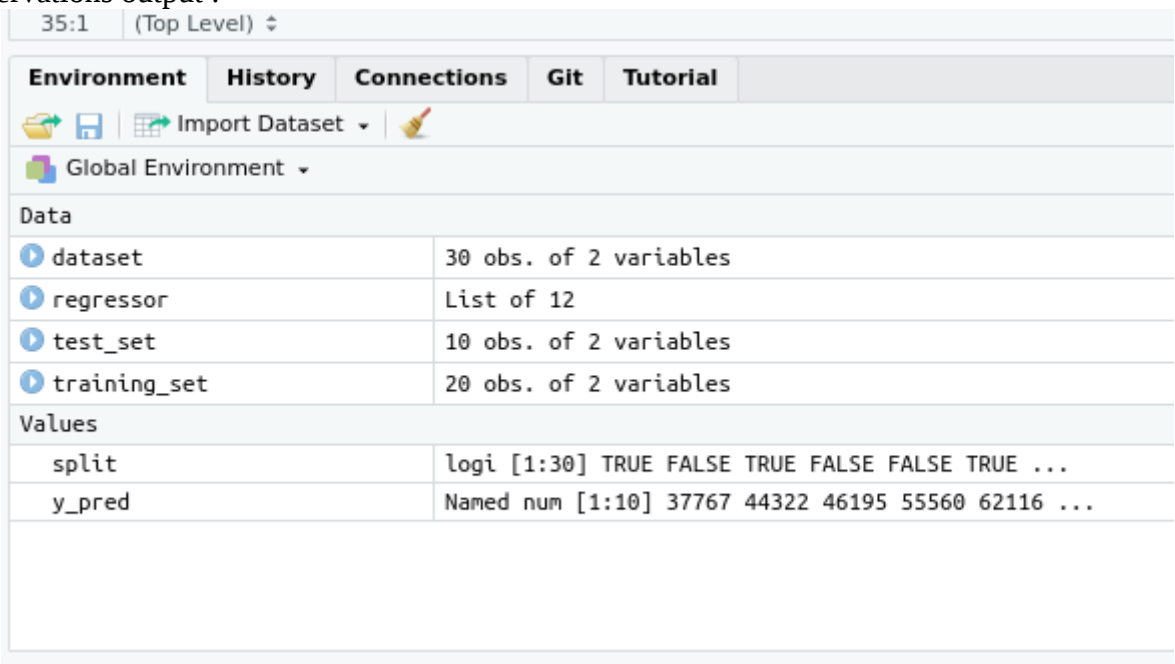
1--> Regression model--> there are 6 different types of regression models-->Simple Linear, multiple Linear, polynomial, support vector, decision tree, random forest.

1. SIMPLE LINEAR REGRESSION MODEL (CODE, OUTPUT, GRAPH)

```
# Simple Linear Regression

# Importing the dataset
dataset = read.csv('Salary_Data.csv')

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Salary, SplitRatio = 2/3)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)

# Fitting Simple Linear Regression to the Training set
regressor = lm(formula = Salary ~ YearsExperience,
               data = training_set)

# Predicting the Test set results
y_pred = predict(regressor, newdata = test_set)

# Visualising the Training set results
library(ggplot2)
ggplot() +
  geom_point(aes(x = training_set$YearsExperience, y = training_set$Salary),
             colour = 'red') +
  geom_line(aes(x = training_set$YearsExperience, y = predict(regressor, newdata = training_set)),
            colour = 'blue') +
  ggtitle('Salary vs Experience (Training set)') +
  xlab('Years of experience') +
  ylab('Salary')

# Visualising the Test set results
library(ggplot2)
ggplot() +
  geom_point(aes(x = test_set$YearsExperience, y = test_set$Salary),
             colour = 'red') +
  geom_line(aes(x = training_set$YearsExperience, y = predict(regressor, newdata = training_set)),
```

```
        colour = 'blue') +
  ggtitle('Salary vs Experience (Test set)') +
  xlab('Years of experience') +
  ylab('Salary')
```

Observations output :

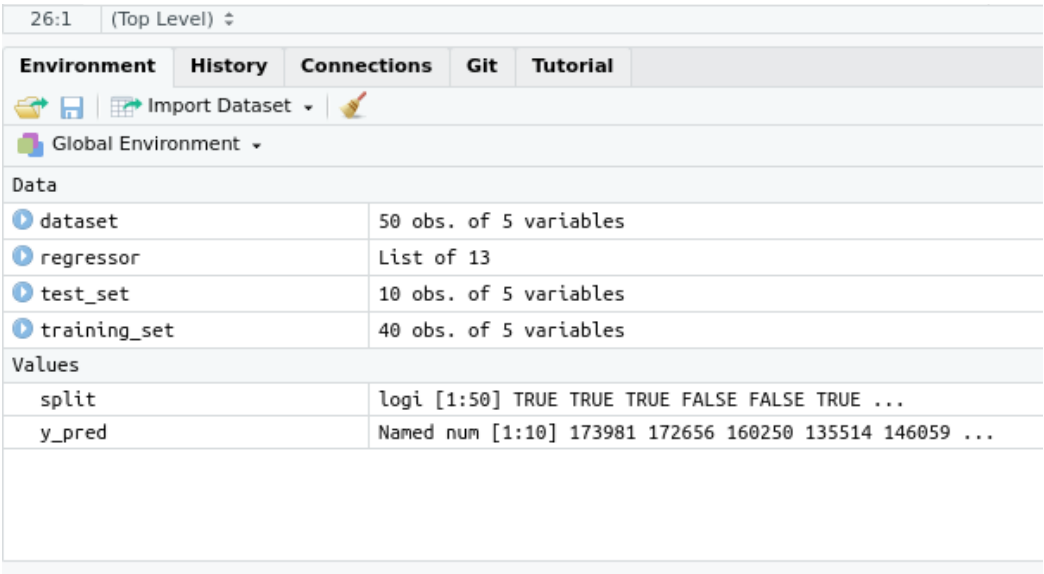


Training set Results :

Test set Results :

2. MULTIPLE LINEAR REGRESSION MODEL (CODE, OUTPUT, GRAPH) :

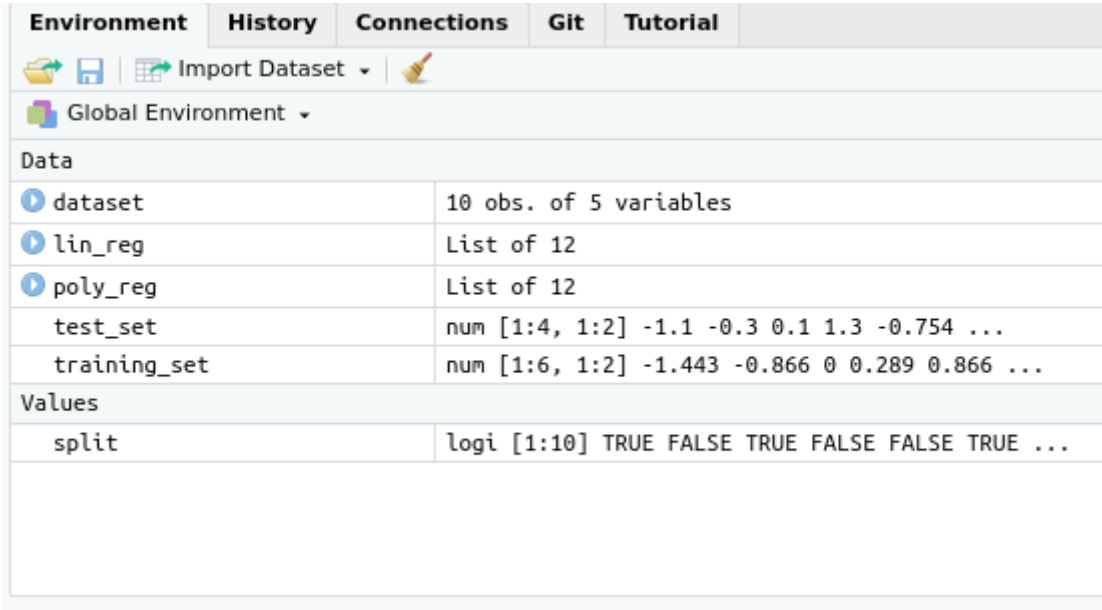
```
# Multiple Linear Regression

# Importing the dataset
dataset = read.csv('50_Startups.csv')

# Encoding categorical data
dataset$State = factor(dataset$State,
                       levels = c('New York', 'California', 'Florida'),
                       labels = c(1, 2, 3))

# Splitting the dataset into the Training set and Test set
# install.packages('caTools')
library(caTools)
set.seed(123)
split = sample.split(dataset$Profit, SplitRatio = 0.8)
training_set = subset(dataset, split == TRUE)
test_set = subset(dataset, split == FALSE)

# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)

# Fitting Multiple Linear Regression to the Training set
regressor = lm(formula = Profit ~ .,
               data = training_set)

# Predicting the Test set results
y_pred = predict(regressor, newdata = test_set)
```

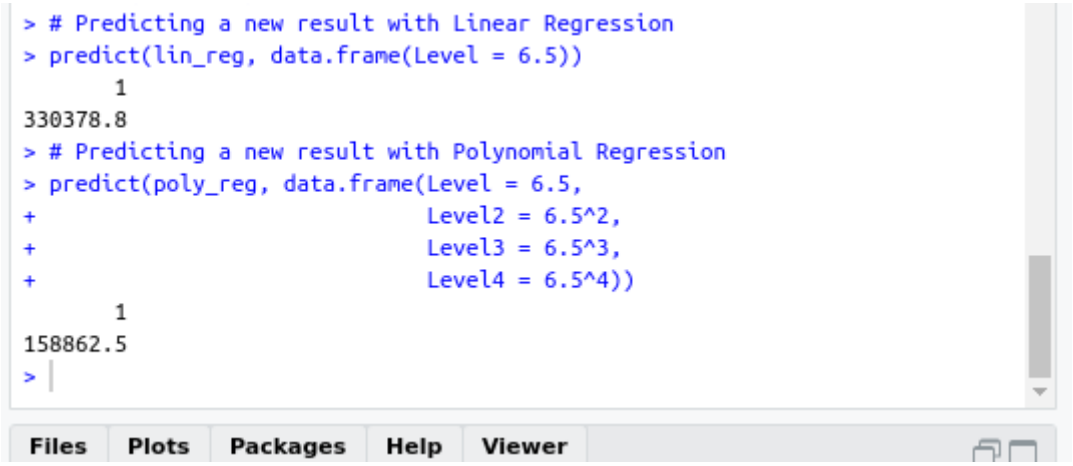


3. POLYNOMIAL REGRESSION (CODE, OUTPUT, GRAPH) :

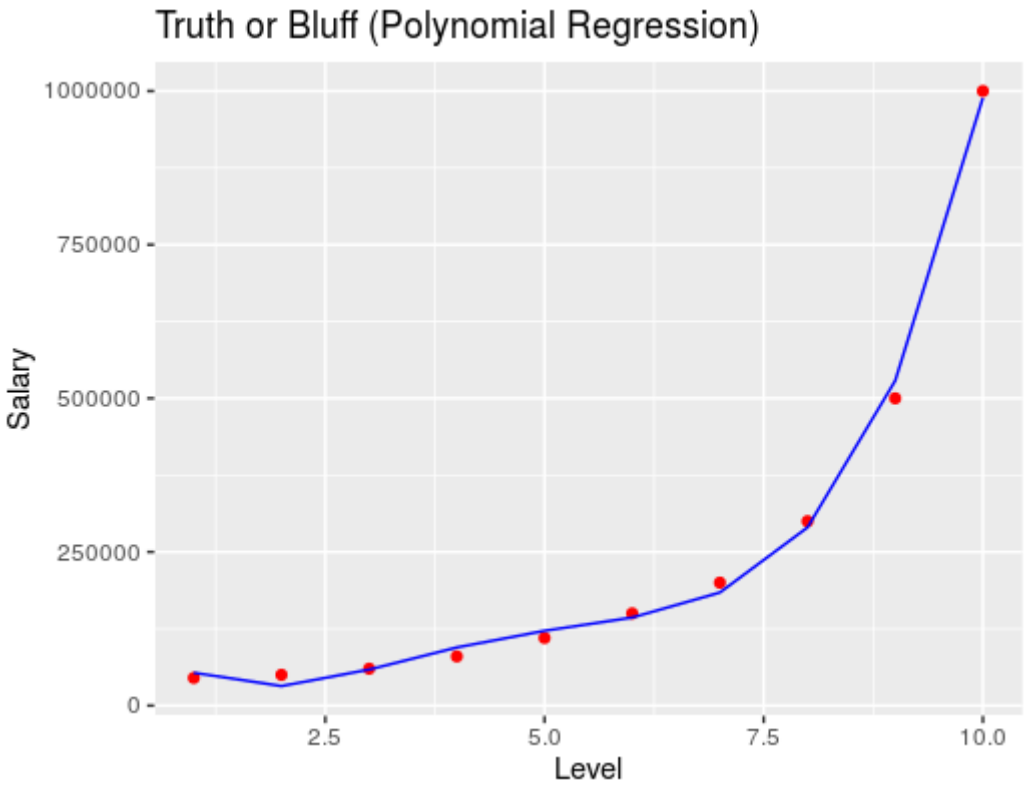
```
# Importing the dataset
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]

# Splitting the dataset into the Training set and Test set
# # install.packages('caTools')
# library(caTools)
# set.seed(123)
# split = sample.split(dataset$Salary, SplitRatio = 2/3)
# training_set = subset(dataset, split == TRUE)
# test_set = subset(dataset, split == FALSE)

# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)

# Fitting Linear Regression to the dataset
lin_reg = lm(formula = Salary ~ .,
             data = dataset)

# Fitting Polynomial Regression to the dataset
dataset$Level2 = dataset$Level^2
dataset$Level3 = dataset$Level^3
dataset$Level4 = dataset$Level^4
poly_reg = lm(formula = Salary ~ .,
              data = dataset)

# Visualising the Polynomial Regression results
# install.packages('ggplot2')
library(ggplot2)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = dataset$Level, y = predict(poly_reg, newdata = dataset)),
            colour = 'blue') +
  ggtitle('Truth or Bluff (Polynomial Regression)') +
  xlab('Level') +
  ylab('Salary')

# Visualising the Regression Model results (for higher resolution and smoother curve)
# install.packages('ggplot2')
library(ggplot2)
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.1)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = x_grid, y = predict(poly_reg,
                                        newdata = data.frame(Level = x_grid,
                                                             Level2 = x_grid^2,
```

```
                                        Level3 = x_grid^3,
                                        Level4 = x_grid^4))),
            colour = 'blue') +
  ggtitle('Truth or Bluff (Polynomial Regression)') +
  xlab('Level') +
  ylab('Salary')

# Predicting a new result with Linear Regression
predict(lin_reg, data.frame(Level = 6.5))

# Predicting a new result with Polynomial Regression
predict(poly_reg, data.frame(Level = 6.5,
                             Level2 = 6.5^2,
                             Level3 = 6.5^3,
                             Level4 = 6.5^4))
```

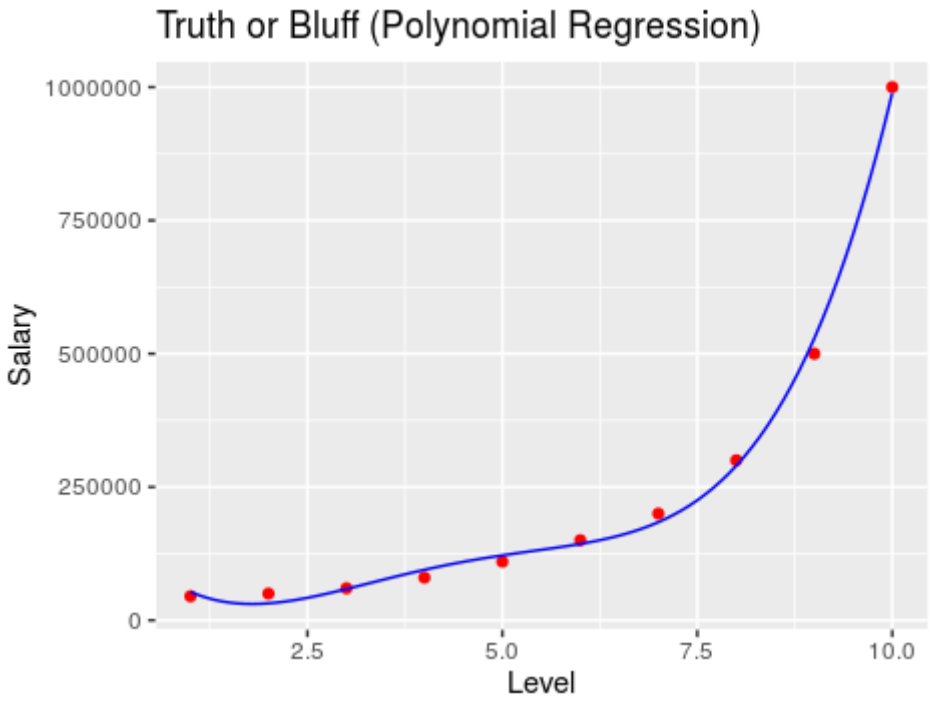




Normal Rplot :



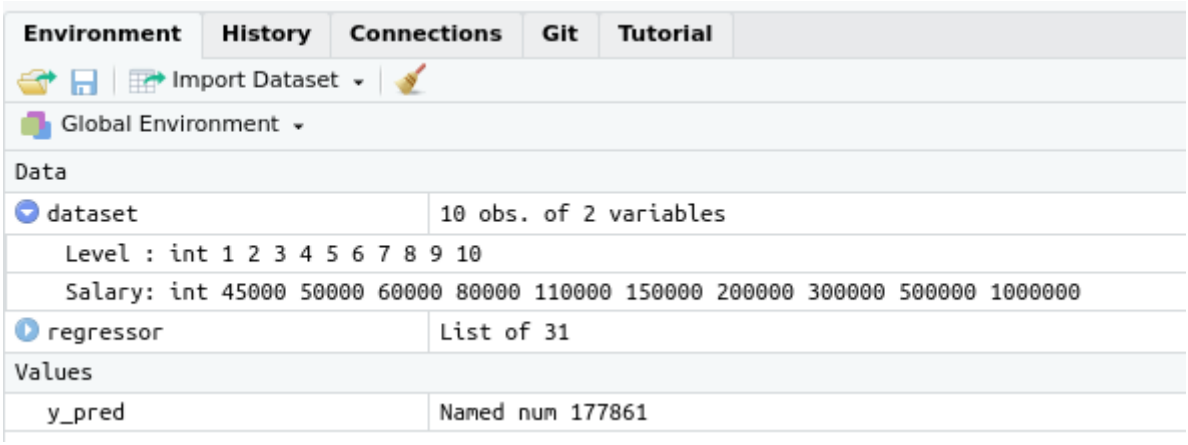
Smooth Rplot :



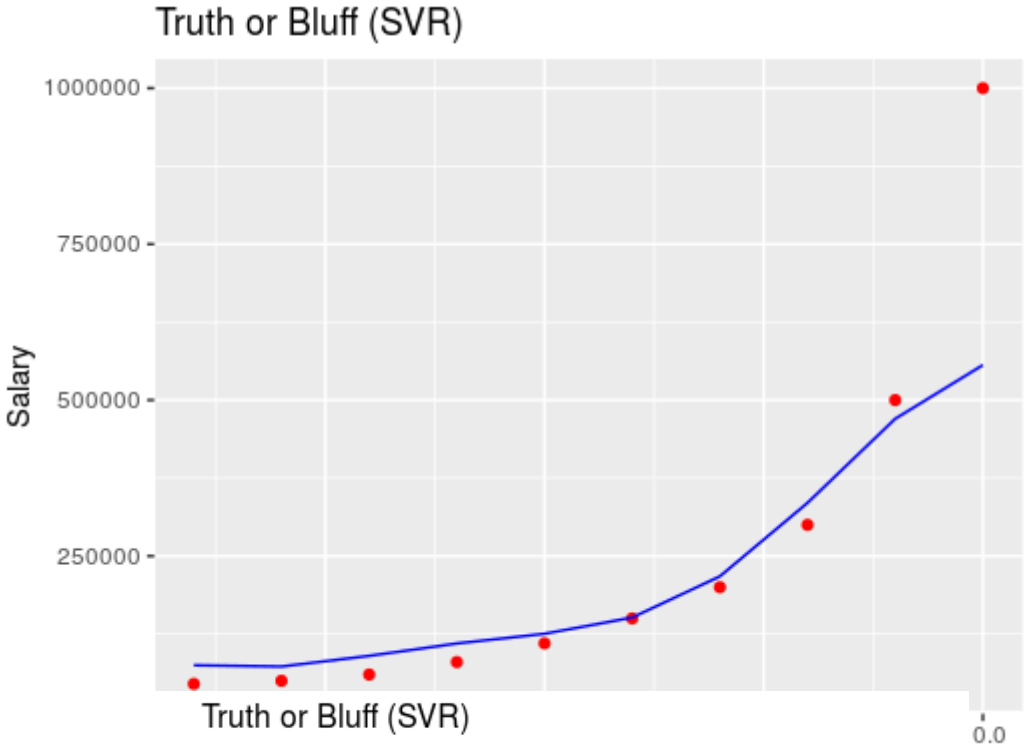
4. SUPPORT VECTOR REGRESSION (SVM) :

```
# SVR

# Importing the dataset
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]

# Splitting the dataset into the Training set and Test set
# # install.packages('caTools')
# library(caTools)
# set.seed(123)
# split = sample.split(dataset$Salary, SplitRatio = 2/3)
# training_set = subset(dataset, split == TRUE)
# test_set = subset(dataset, split == FALSE)

# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)

# Fitting SVR to the dataset
# install.packages('e1071')
library(e1071)
regressor = svm(formula = Salary ~ .,
                data = dataset,
                type = 'eps-regression',
                kernel = 'radial')

# Predicting a new result
y_pred = predict(regressor, data.frame(Level = 6.5))

# Visualising the SVR results
# install.packages('ggplot2')
library(ggplot2)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = dataset$Level, y = predict(regressor, newdata = dataset)),
            colour = 'blue') +
  ggtitle('Truth or Bluff (SVR)') +
  xlab('Level') +
  ylab('Salary')

# Visualising the SVR results (for higher resolution and smoother curve)
# install.packages('ggplot2')
library(ggplot2)
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.1)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = x_grid, y = predict(regressor, newdata = data.frame(Level = x_grid))),
```

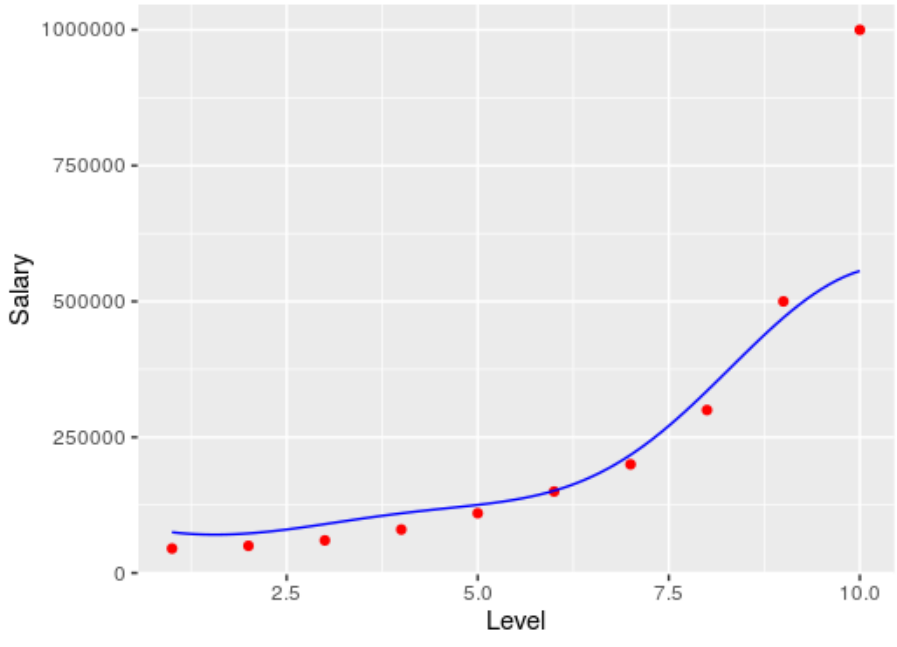
```
        colour = 'blue') +
ggtitle('Truth or Bluff (SVR)') +
xlab('Level') +
ylab('Salary')
```



SVM normal Rplot :

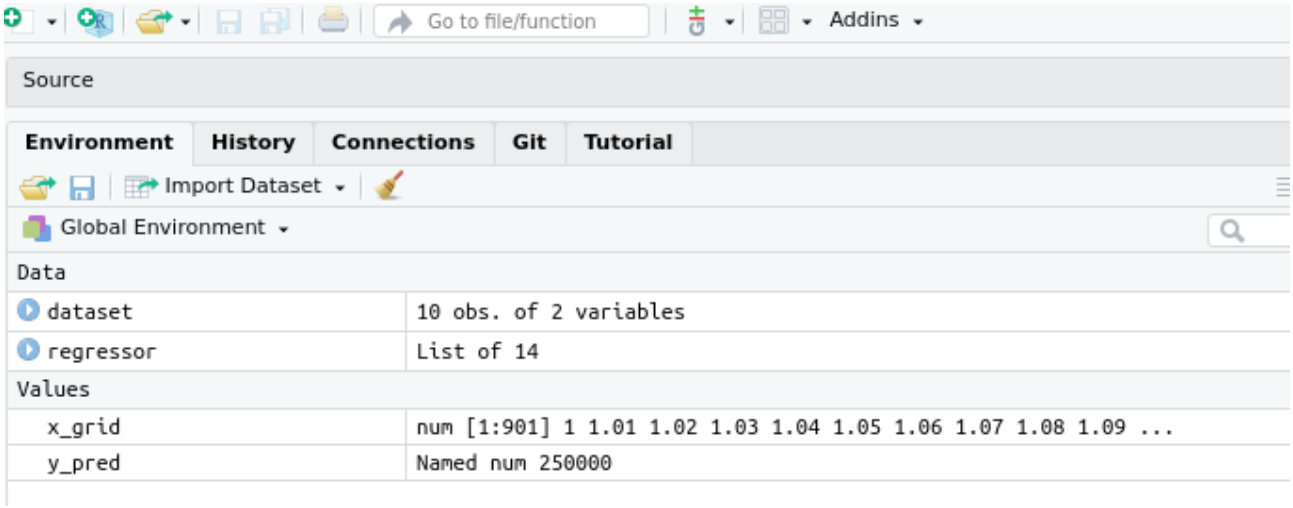


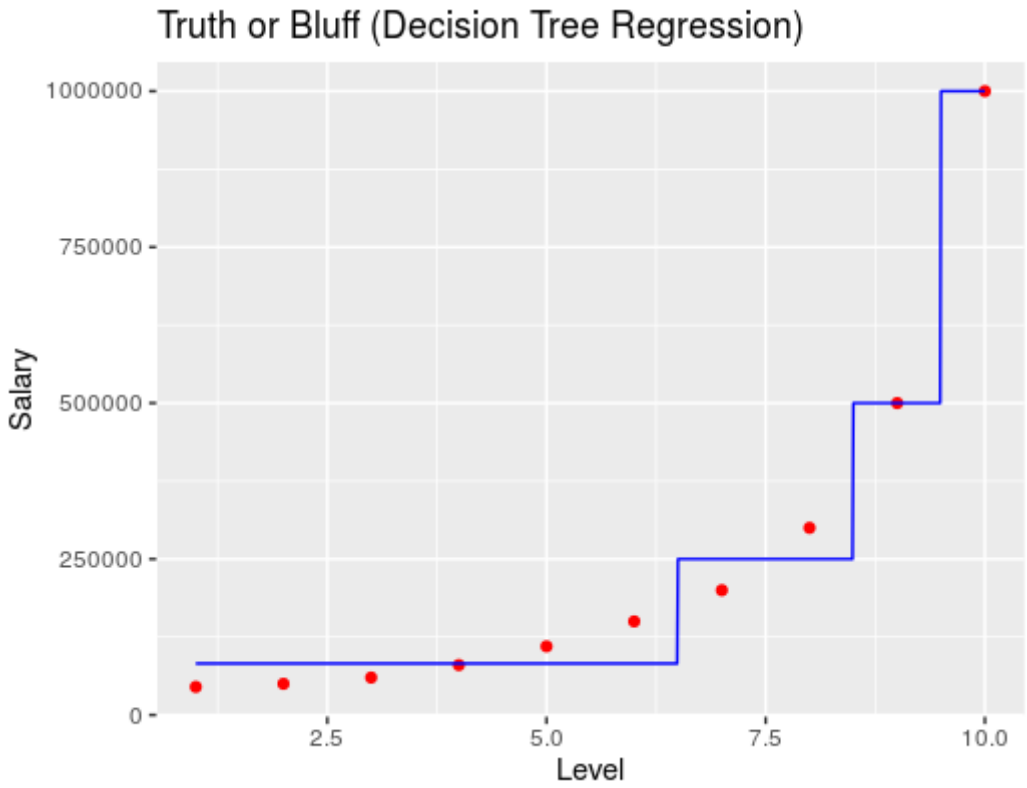
SVM smoother Rplot :

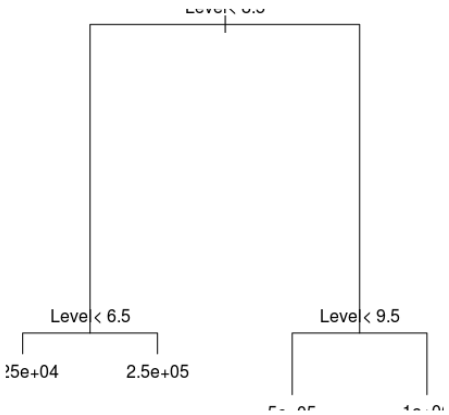


5. DECISION TREE REGRESSION MODEL (CODE, OUTPUT, GRAPH) :

```
# Decision Tree Regression

# Importing the dataset
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]

# Splitting the dataset into the Training set and Test set
# # install.packages('caTools')
# library(caTools)
# set.seed(123)
# split = sample.split(dataset$Salary, SplitRatio = 2/3)
# training_set = subset(dataset, split == TRUE)
# test_set = subset(dataset, split == FALSE)

# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)

# Fitting Decision Tree Regression to the dataset
# install.packages('rpart')
library(rpart)
regressor = rpart(formula = Salary ~ .,
                  data = dataset,
                  control = rpart.control(minsplit = 1))

# Predicting a new result with Decision Tree Regression
y_pred = predict(regressor, data.frame(Level = 6.5))

# Visualising the Decision Tree Regression results (higher resolution)
# install.packages('ggplot2')
library(ggplot2)
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.01)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = x_grid, y = predict(regressor, newdata = data.frame(Level = x_grid))),
            colour = 'blue') +
  ggtitle('Truth or Bluff (Decision Tree Regression)') +
  xlab('Level') +
  ylab('Salary')

# Plotting the tree
plot(regressor)
text(regressor)
```

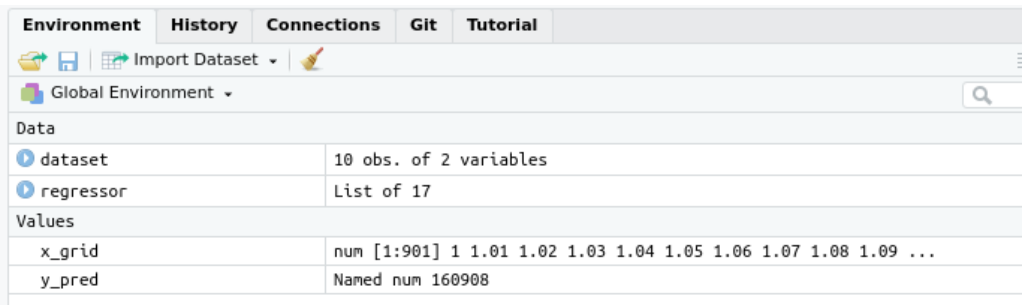

Go to file/function
Addins
Source
Environment
History
Connections
Git
Tutorial
Import Dataset
Global Environment
Data
dataset
10 obs. of 2 variables
regressor
List of 14
Values
x_grid
num [1:901] 1 1.01 1.02 1.03 1.04 1.05 1.06 1.07 1.08 1.09 ...
y_pred
Named num 250000

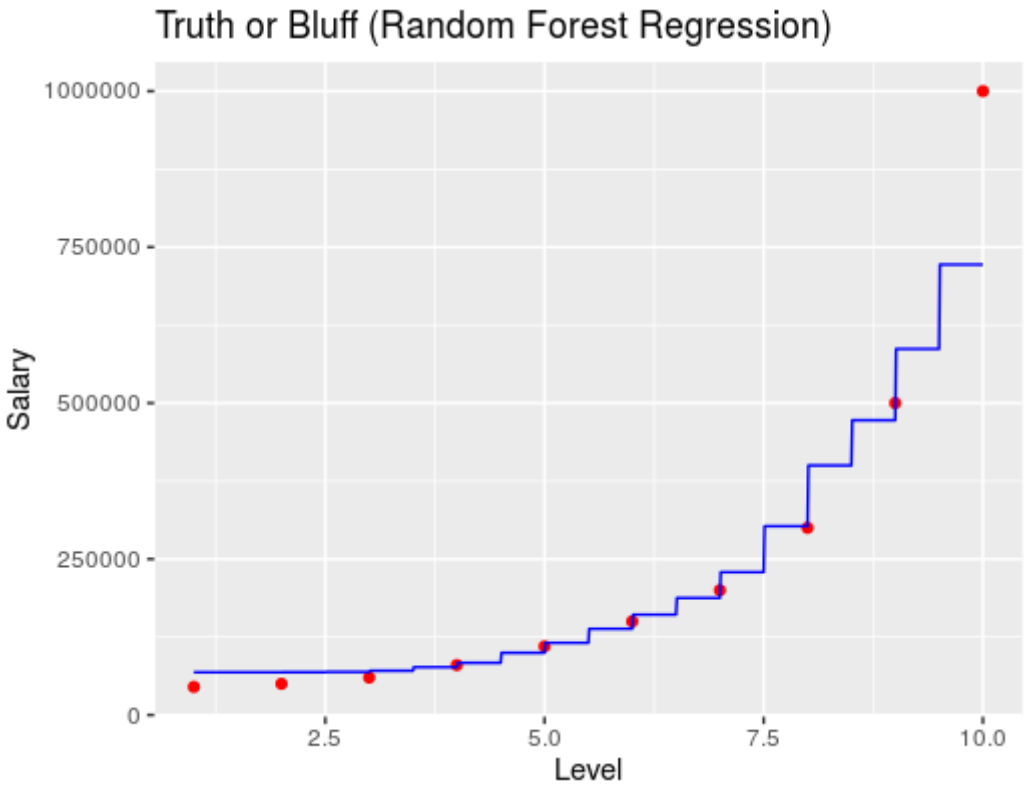

Truth or Bluff (Decision Tree Regression)
1000000
750000
500000
250000
0
Salary
2.5
5.0
7.5
10.0
Level


Level < 6.5
Level < 9.5
2.5e+05

6.RANDOM FOREST REGRESSION MODEL (CODE, OUTPUT, GRAPH)

```
# Random Forest Regression

# Importing the dataset
dataset = read.csv('Position_Salaries.csv')
dataset = dataset[2:3]

# Splitting the dataset into the Training set and Test set
# # install.packages('caTools')
# library(caTools)
# set.seed(123)
# split = sample.split(dataset$Salary, SplitRatio = 2/3)
# training_set = subset(dataset, split == TRUE)
# test_set = subset(dataset, split == FALSE)

# Feature Scaling
# training_set = scale(training_set)
# test_set = scale(test_set)

# Fitting Random Forest Regression to the dataset
install.packages('randomForest')
library(randomForest)
set.seed(1234)
regressor = randomForest(x = dataset[-2],
                         y = dataset$Salary,
                         ntree = 500)

# Predicting a new result with Random Forest Regression
y_pred = predict(regressor, data.frame(Level = 6.5))

# Visualising the Random Forest Regression results (higher resolution)
# install.packages('ggplot2')
library(ggplot2)
x_grid = seq(min(dataset$Level), max(dataset$Level), 0.01)
ggplot() +
  geom_point(aes(x = dataset$Level, y = dataset$Salary),
             colour = 'red') +
  geom_line(aes(x = x_grid, y = predict(regressor, newdata = data.frame(Level = x_grid))),
            colour = 'blue') +
  ggtitle('Truth or Bluff (Random Forest Regression)') +
  xlab('Level') +
  ylab('Salary')
```




Truth or Bluff (Random Forest Regression)
1000000
750000
500000
250000
0
Salary
2.5
5.0
7.5
10.0
Level