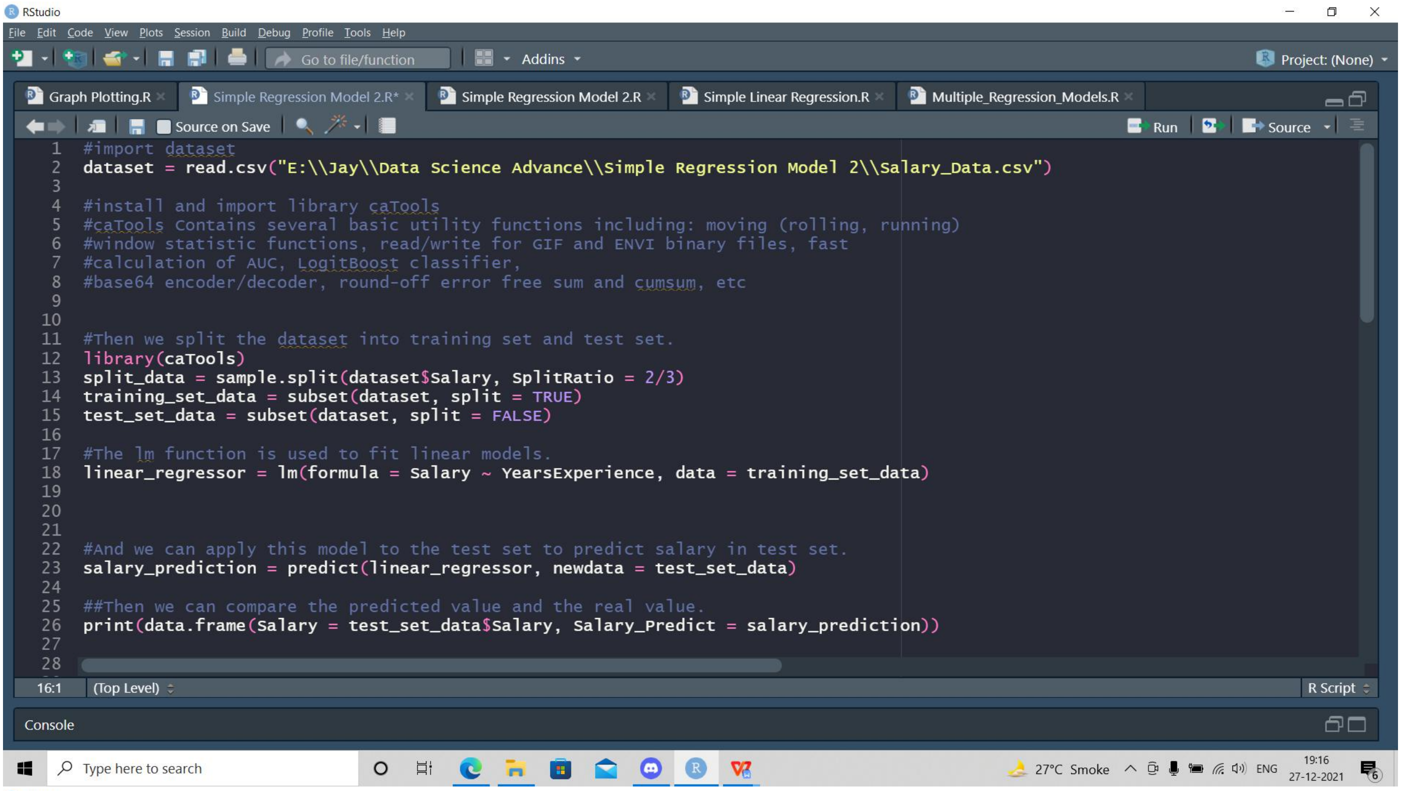
Simple Regression Model 2

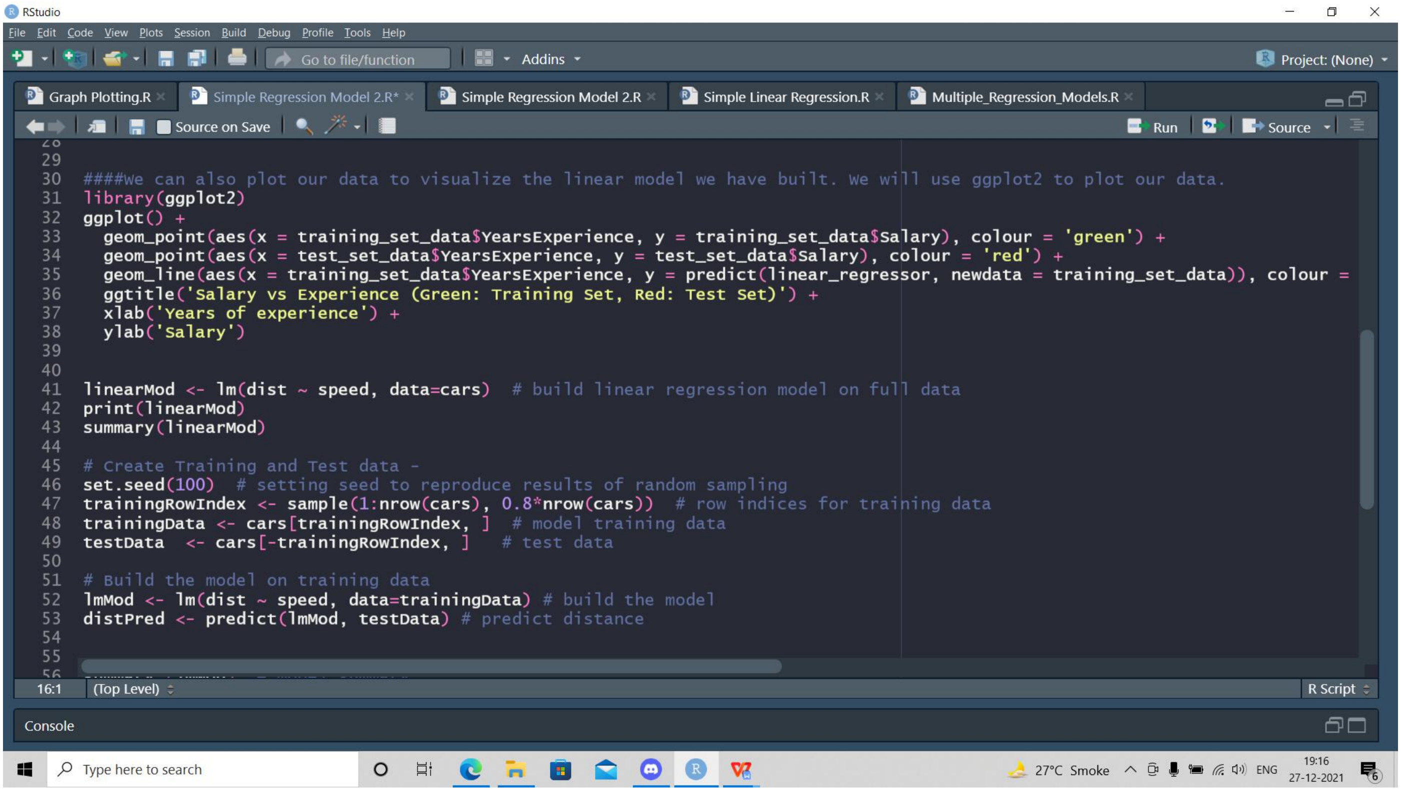
--Jay Rathod

Code:



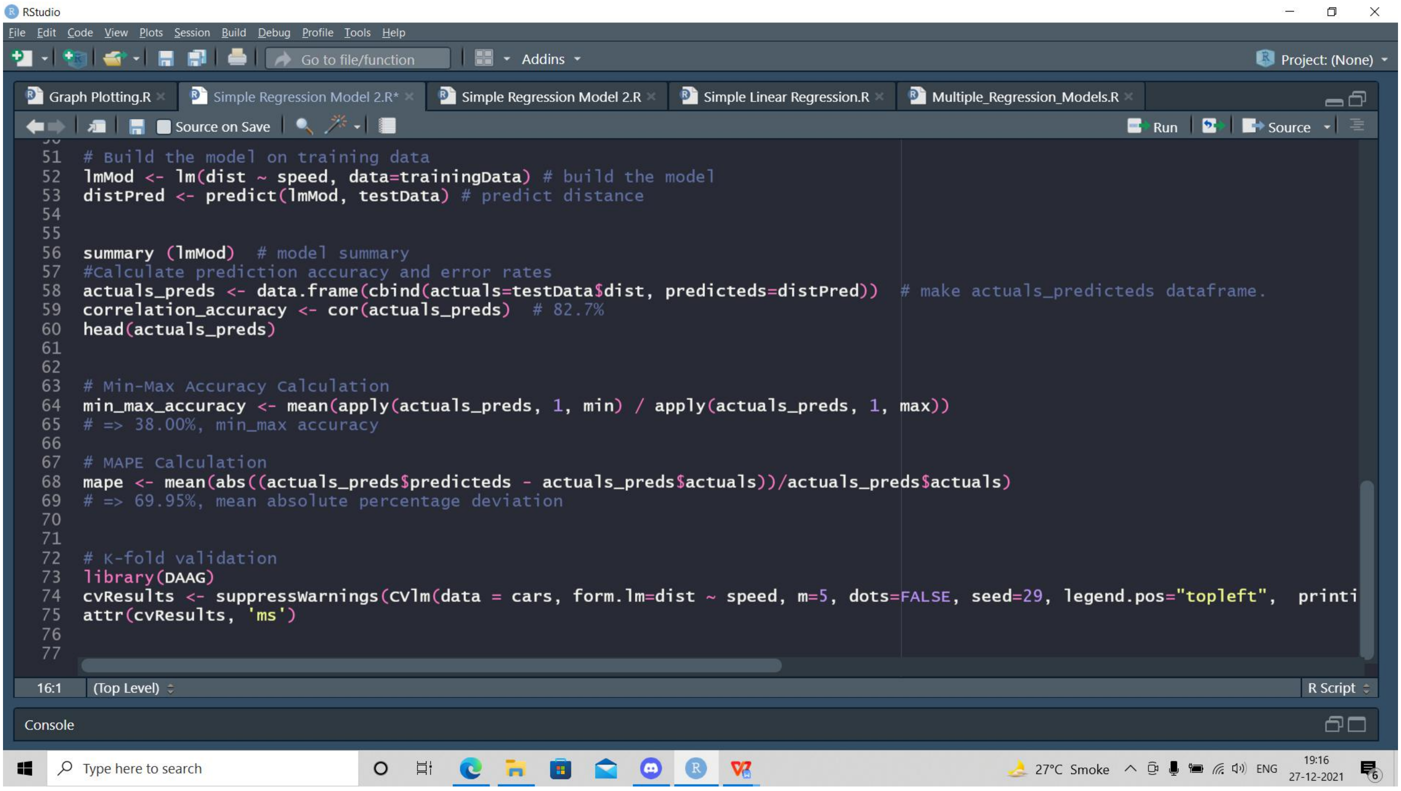
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Graph Plotting.R
Simple Regression Model 2.R*
Simple Regression Model 2.R
Simple Linear Regression.R
Multiple_Regression_Models.R
Source on Save
Run
Source
16:1
(Top Level)
R Script
Console
Type here to search
27°C Smoke
ENG
19:16
27-12-2021

```r
#import dataset
dataset = read.csv("E:\\Jay\\Data Science Advance\\Simple Regression Model 2\\Salary_Data.csv")

#install and import library caTools
#caTools Contains several basic utility functions including: moving (rolling, running)
#window statistic functions, read/write for GIF and ENVI binary files, fast
#calculation of AUC, LogitBoost classifier,
#base64 encoder/decoder, round-off error free sum and cumsum, etc


#Then we split the dataset into training set and test set.
library(caTools)
split_data = sample.split(dataset$Salary, SplitRatio = 2/3)
training_set_data = subset(dataset, split = TRUE)
test_set_data = subset(dataset, split = FALSE)

#The lm function is used to fit linear models.
linear_regressor = lm(formula = Salary ~ YearsExperience, data = training_set_data)



#And we can apply this model to the test set to predict salary in test set.
salary_prediction = predict(linear_regressor, newdata = test_set_data)

##Then we can compare the predicted value and the real value.
print(data.frame(Salary = test_set_data$Salary, Salary_Predict = salary_prediction))


```



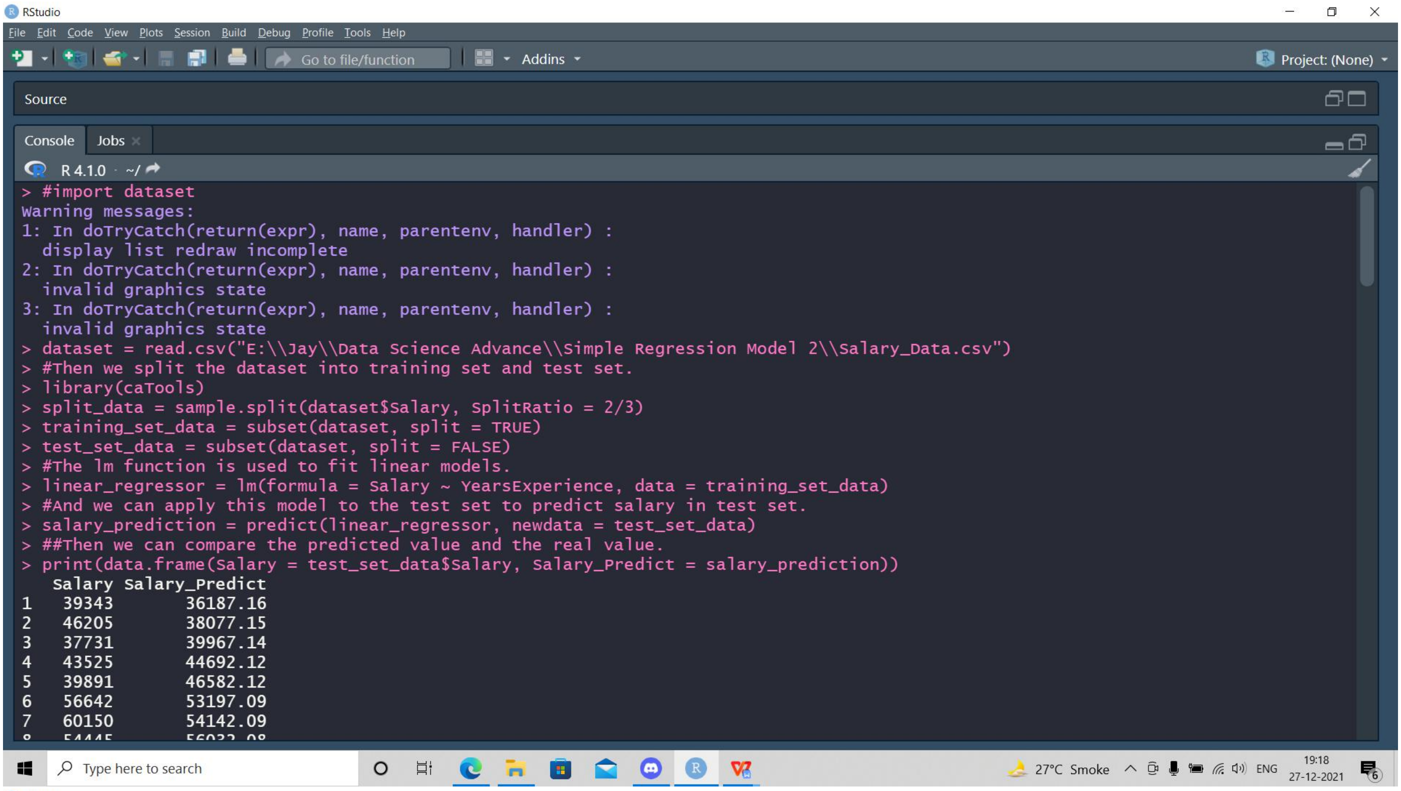
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Graph Plotting.R
Simple Regression Model 2.R*
Simple Regression Model 2.R
Simple Linear Regression.R
Multiple_Regression_Models.R
Source on Save
Run
Source
16:1
(Top Level)
R Script
Console
Type here to search
27°C Smoke
ENG
19:16
27-12-2021

```r

####We can also plot our data to visualize the linear model we have built. We will use ggplot2 to plot our data.
library(ggplot2)
ggplot() +
  geom_point(aes(x = training_set_data$YearsExperience, y = training_set_data$Salary), colour = 'green') +
  geom_point(aes(x = test_set_data$YearsExperience, y = test_set_data$Salary), colour = 'red') +
  geom_line(aes(x = training_set_data$YearsExperience, y = predict(linear_regressor, newdata = training_set_data)), colour =
  ggtitle('Salary vs Experience (Green: Training Set, Red: Test Set)') +
  xlab('Years of experience') +
  ylab('Salary')


linearMod <- lm(dist ~ speed, data=cars)  # build linear regression model on full data
print(linearMod)
summary(linearMod)

# Create Training and Test data -
set.seed(100)  # setting seed to reproduce results of random sampling
trainingRowIndex <- sample(1:nrow(cars), 0.8*nrow(cars))  # row indices for training data
trainingData <- cars[trainingRowIndex, ]  # model training data
testData  <- cars[-trainingRowIndex, ]   # test data

# Build the model on training data
lmMod <- lm(dist ~ speed, data=trainingData) # build the model
distPred <- predict(lmMod, testData) # predict distance


```

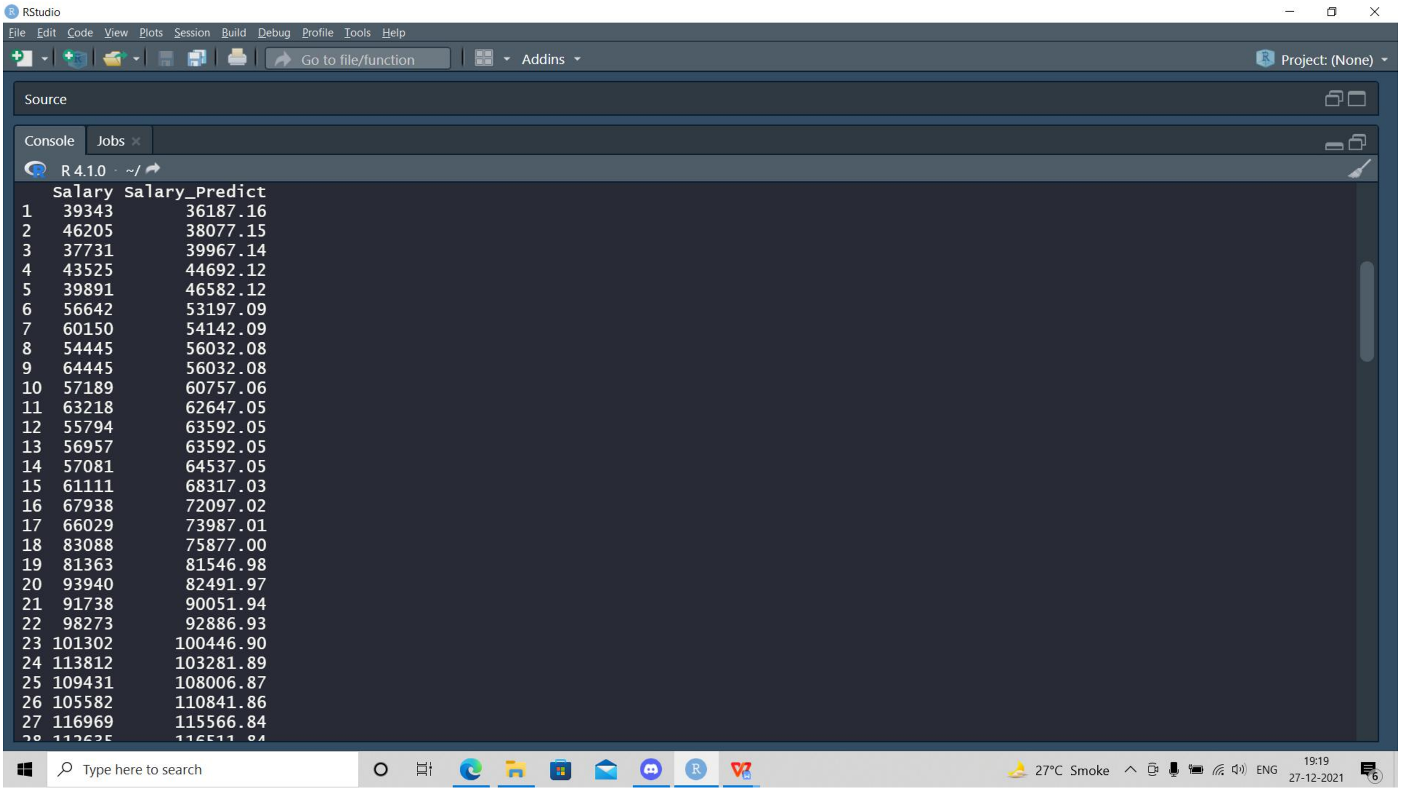

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Graph Plotting.R
Simple Regression Model 2.R*
Simple Regression Model 2.R
Simple Linear Regression.R
Multiple_Regression_Models.R
Source on Save
Run
Source
16:1
(Top Level)
R Script
Console
Type here to search
27°C Smoke
ENG
19:16
27-12-2021

```r
# Build the model on training data
lmMod <- lm(dist ~ speed, data=trainingData) # build the model
distPred <- predict(lmMod, testData) # predict distance


summary (lmMod)  # model summary
#Calculate prediction accuracy and error rates
actuals_preds <- data.frame(cbind(actuals=testData$dist, predicteds=distPred))  # make actuals_predicteds dataframe.
correlation_accuracy <- cor(actuals_preds)  # 82.7%
head(actuals_preds)


# Min-Max Accuracy Calculation
min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))
# => 38.00%, min_max accuracy

# MAPE Calculation
mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actuals))/actuals_preds$actuals)
# => 69.95%, mean absolute percentage deviation


# K-fold validation
library(DAAG)
cvResults <- suppressWarnings(CVlm(data = cars, form.lm=dist ~ speed, m=5, dots=FALSE, seed=29, legend.pos="topleft",  printi
attr(cvResults, 'ms')


```

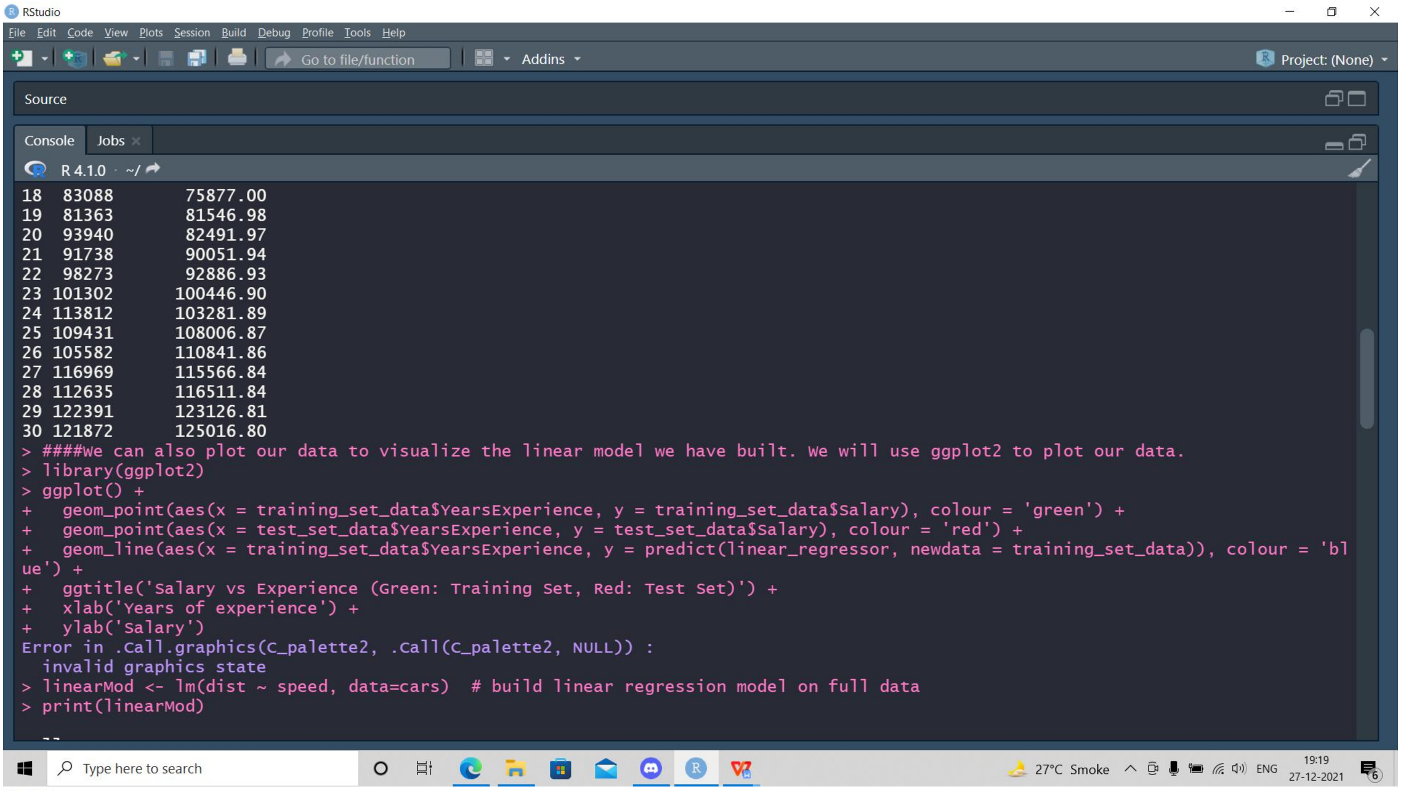
Output:



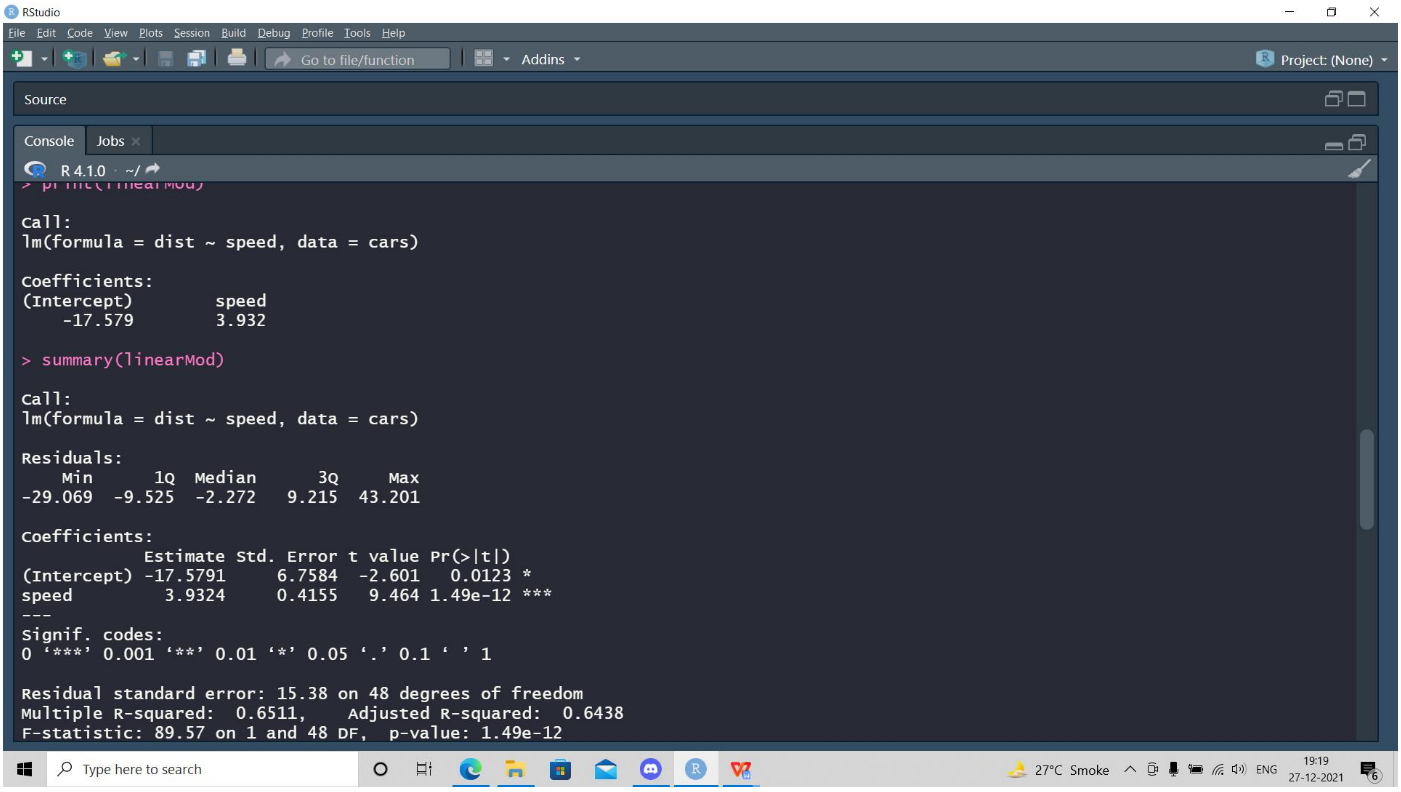
```
> #import dataset
Warning messages:
1: In doTryCatch(return(expr), name, parentenv, handler) :
  display list redraw incomplete
2: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
3: In doTryCatch(return(expr), name, parentenv, handler) :
  invalid graphics state
> dataset = read.csv("E:\\Jay\\Data Science Advance\\Simple Regression Model 2\\Salary_Data.csv")
> #Then we split the dataset into training set and test set.
> library(caTools)
> split_data = sample.split(dataset$Salary, SplitRatio = 2/3)
> training_set_data = subset(dataset, split = TRUE)
> test_set_data = subset(dataset, split = FALSE)
> #The lm function is used to fit linear models.
> linear_regressor = lm(formula = Salary ~ YearsExperience, data = training_set_data)
> #And we can apply this model to the test set to predict salary in test set.
> salary_prediction = predict(linear_regressor, newdata = test_set_data)
> ##Then we can compare the predicted value and the real value.
> print(data.frame(Salary = test_set_data$Salary, Salary_Predict = salary_prediction))
   Salary Salary_Predict
1   39343       36187.16
2   46205       38077.15
3   37731       39967.14
4   43525       44692.12
5   39891       46582.12
6   56642       53197.09
7   60150       54142.09
8   54445       56032.08
```



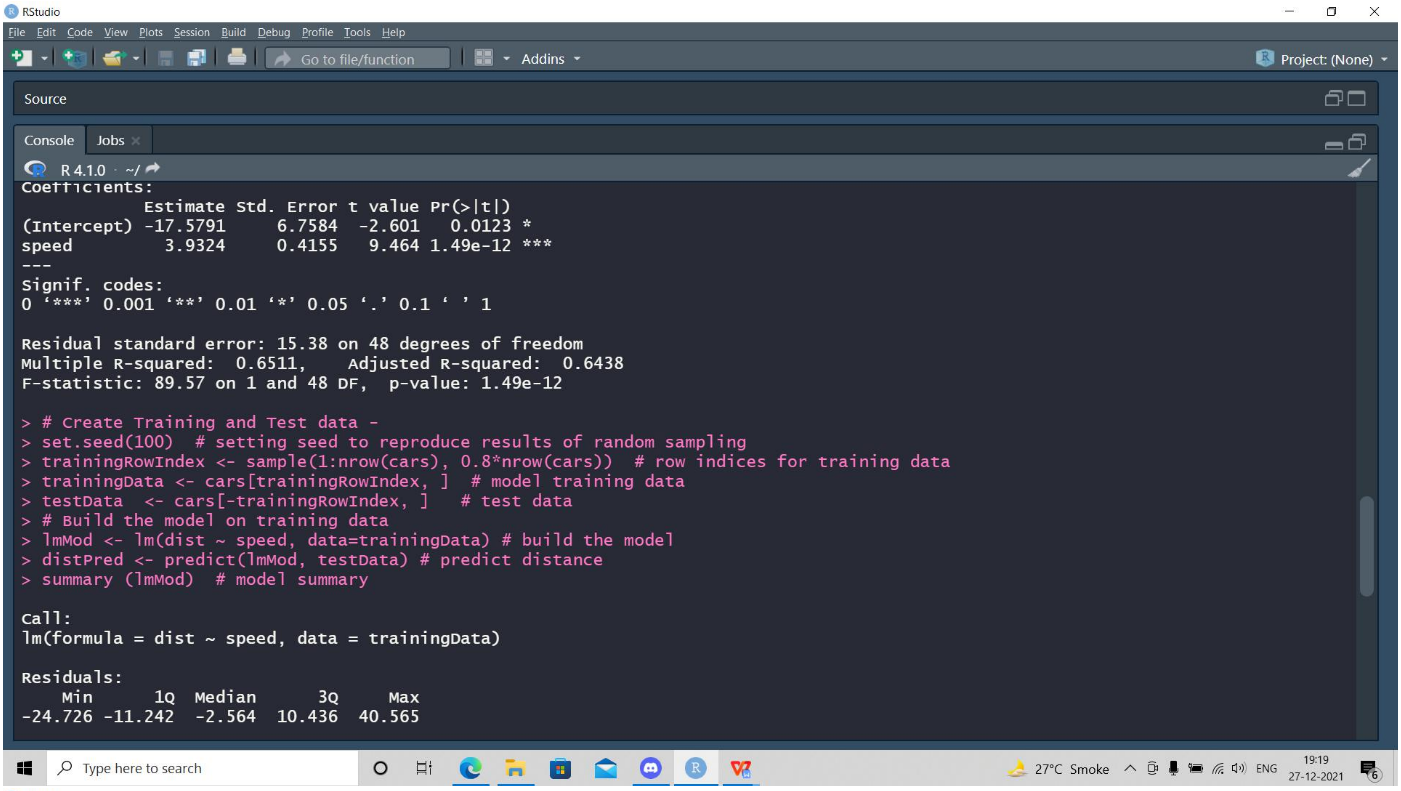
```
   Salary Salary_Predict
1   39343       36187.16
2   46205       38077.15
3   37731       39967.14
4   43525       44692.12
5   39891       46582.12
6   56642       53197.09
7   60150       54142.09
8   54445       56032.08
9   64445       56032.08
10  57189       60757.06
11  63218       62647.05
12  55794       63592.05
13  56957       63592.05
14  57081       64537.05
15  61111       68317.03
16  67938       72097.02
17  66029       73987.01
18  83088       75877.00
19  81363       81546.98
20  93940       82491.97
21  91738       90051.94
22  98273       92886.93
23 101302      100446.90
24 113812      103281.89
25 109431      108006.87
26 105582      110841.86
27 116969      115566.84
28 112635      116511.84
```



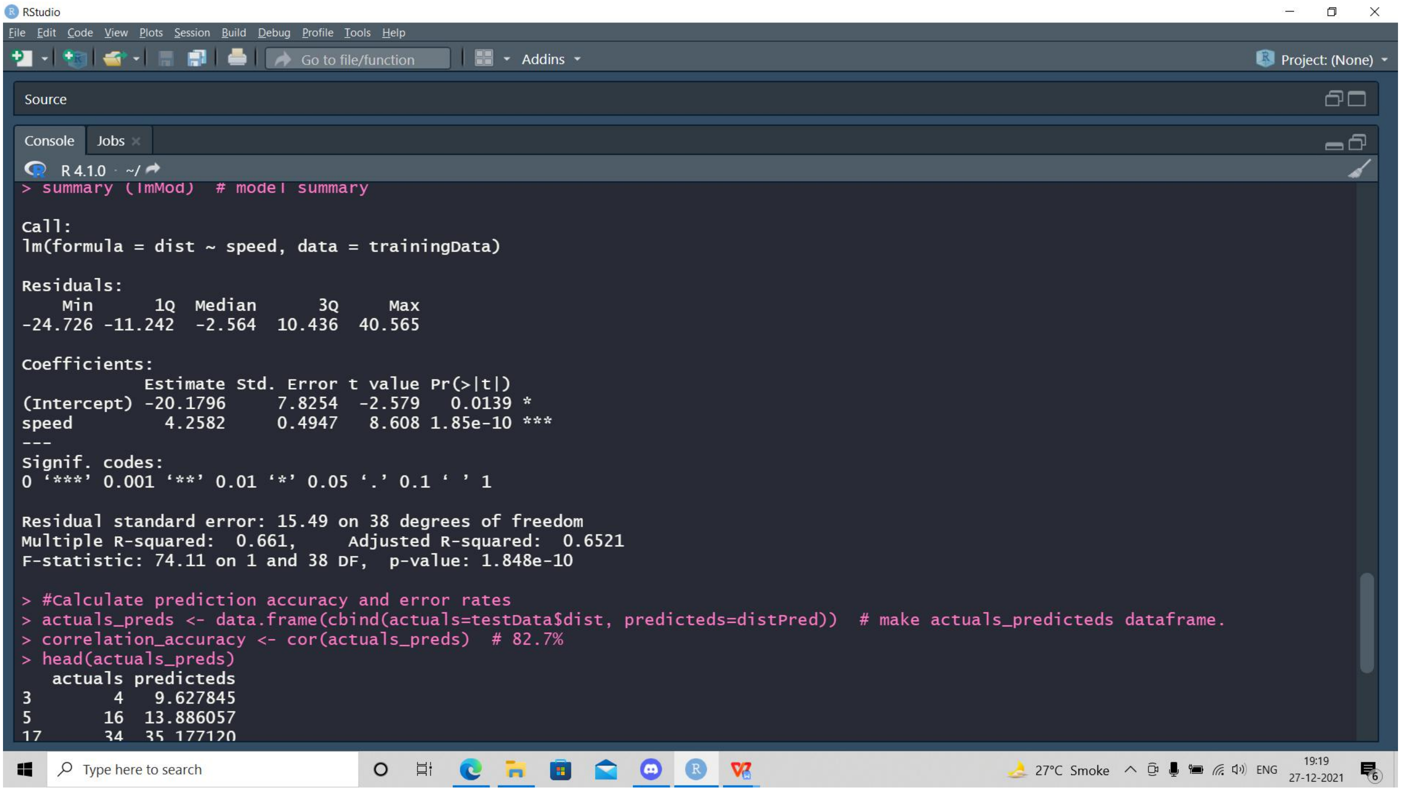
```
18  83088        75877.00
19  81363        81546.98
20  93940        82491.97
21  91738        90051.94
22  98273        92886.93
23 101302       100446.90
24 113812       103281.89
25 109431       108006.87
26 105582       110841.86
27 116969       115566.84
28 112635       116511.84
29 122391       123126.81
30 121872       125016.80
> ####We can also plot our data to visualize the linear model we have built. We will use ggplot2 to plot our data.
> library(ggplot2)
> ggplot() +
+   geom_point(aes(x = training_set_data$YearsExperience, y = training_set_data$Salary), colour = 'green') +
+   geom_point(aes(x = test_set_data$YearsExperience, y = test_set_data$Salary), colour = 'red') +
+   geom_line(aes(x = training_set_data$YearsExperience, y = predict(linear_regressor, newdata = training_set_data)), colour = 'bl
ue') +
+   ggtitle('Salary vs Experience (Green: Training Set, Red: Test Set)') +
+   xlab('Years of experience') +
+   ylab('Salary')
Error in .Call.graphics(C_palette2, .Call(C_palette2, NULL)) : 
  invalid graphics state
> linearMod <- lm(dist ~ speed, data=cars)  # build linear regression model on full data
> print(linearMod)
```



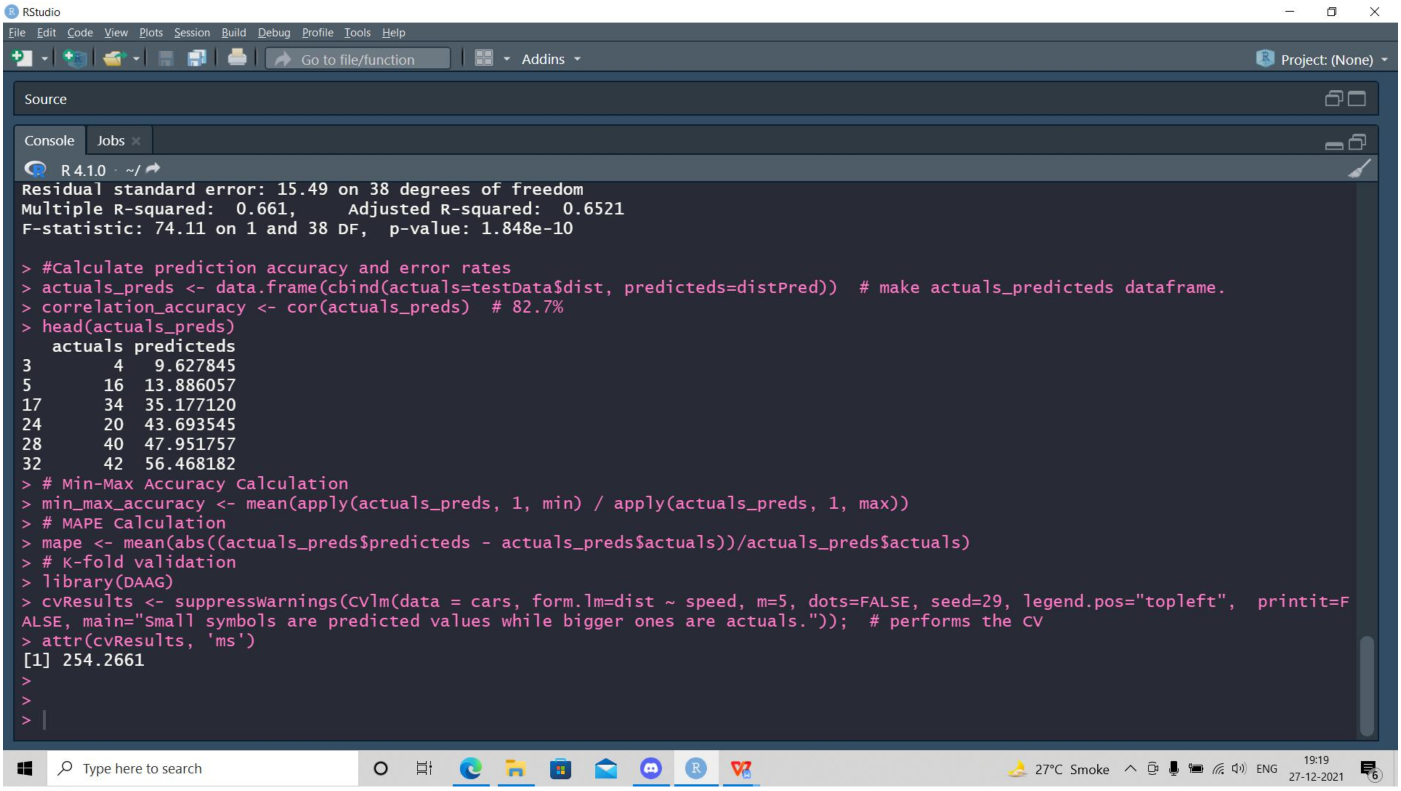
```
> print(linearMod)

Call:
lm(formula = dist ~ speed, data = cars)

Coefficients:
(Intercept)        speed  
    -17.579        3.932  

> summary(linearMod)

Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min      1Q  Median      3Q     Max 
-29.069  -9.525  -2.272   9.215  43.201 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -17.5791     6.7584  -2.601   0.0123 *  
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:  
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,	Adjusted R-squared:  0.6438 
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12
```



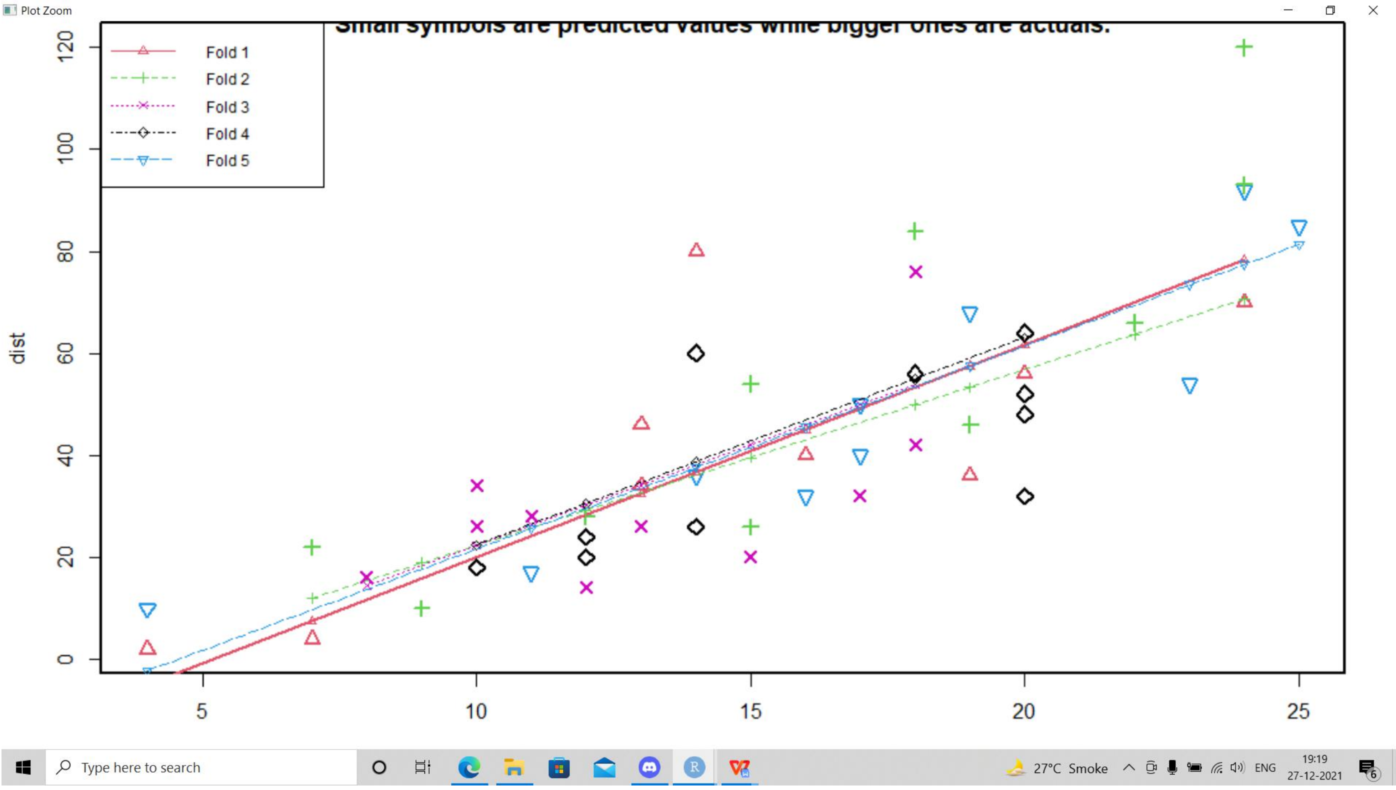
```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *
speed         3.9324     0.4155   9.464 1.49e-12 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.38 on 48 degrees of freedom
Multiple R-squared:  0.6511,	Adjusted R-squared:  0.6438
F-statistic: 89.57 on 1 and 48 DF,  p-value: 1.49e-12

> # Create Training and Test data -
> set.seed(100)  # setting seed to reproduce results of random sampling
> trainingRowIndex <- sample(1:nrow(cars), 0.8*nrow(cars))  # row indices for training data
> trainingData <- cars[trainingRowIndex, ]  # model training data
> testData  <- cars[-trainingRowIndex, ]   # test data
> # Build the model on training data
> lmMod <- lm(dist ~ speed, data=trainingData) # build the model
> distPred <- predict(lmMod, testData) # predict distance
> summary (lmMod)  # model summary

Call:
lm(formula = dist ~ speed, data = trainingData)

Residuals:
    Min      1Q  Median      3Q     Max 
-24.726 -11.242  -2.564  10.436  40.565 
```



```
> summary (lmMod)  # model summary

Call:
lm(formula = dist ~ speed, data = trainingData)

Residuals:
    Min      1Q  Median      3Q     Max 
-24.726 -11.242  -2.564  10.436  40.565 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -20.1796     7.8254  -2.579   0.0139 *
speed         4.2582     0.4947   8.608 1.85e-10 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15.49 on 38 degrees of freedom
Multiple R-squared:  0.661,	Adjusted R-squared:  0.6521
F-statistic: 74.11 on 1 and 38 DF,  p-value: 1.848e-10

> #Calculate prediction accuracy and error rates
> actuals_preds <- data.frame(cbind(actuals=testData$dist, predicteds=distPred))  # make actuals_predicteds dataframe.
> correlation_accuracy <- cor(actuals_preds)  # 82.7%
> head(actuals_preds)
   actuals predicteds
3        4   9.627845
5       16  13.886057
17      34  35.177120
```



```
Residual standard error: 15.49 on 38 degrees of freedom
Multiple R-squared:  0.661,	Adjusted R-squared:  0.6521 
F-statistic: 74.11 on 1 and 38 DF,  p-value: 1.848e-10

> #Calculate prediction accuracy and error rates
> actuals_preds <- data.frame(cbind(actuals=testData$dist, predicteds=distPred))  # make actuals_predicteds dataframe.
> correlation_accuracy <- cor(actuals_preds)  # 82.7%
> head(actuals_preds)
   actuals predicteds
3        4   9.627845
5       16  13.886057
17      34  35.177120
24      20  43.693545
28      40  47.951757
32      42  56.468182
> # Min-Max Accuracy Calculation
> min_max_accuracy <- mean(apply(actuals_preds, 1, min) / apply(actuals_preds, 1, max))
> # MAPE Calculation
> mape <- mean(abs((actuals_preds$predicteds - actuals_preds$actuals))/actuals_preds$actuals)
> # K-fold validation
> library(DAAG)
> cvResults <- suppressWarnings(CVlm(data = cars, form.lm=dist ~ speed, m=5, dots=FALSE, seed=29, legend.pos="topleft",  printit=F
ALSE, main="Small symbols are predicted values while bigger ones are actuals."));  # performs the CV
> attr(cvResults, 'ms')
[1] 254.2661
> 
> 
> |
```


Plot Zoom
Small symbols are predicted values while bigger ones are actuals.
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
dist
Type here to search
27°C Smoke
ENG
19:19