Homework 9 worksheet

Madhu Jagdale

Prepared for ITMD/ITMS/STAT 514, Spring 2021

Packages

```
library(ggplot2)
library(knitr)
library(ISLR)
```

```
## Warning: package 'ISLR' was built under R version 4.0.5
```

```
library(reticulate)
```

```
## Warning: package 'reticulate' was built under R version 4.0.4
```

```
options(scipen = 4)
#py_install("pandas")
#py_install("statsmodels")
#py_install("matplotlib")
```

Simple linear regression in R

> Using the lm() function to perform a simple linear regression with mpg as the response and horsepower as the predictor using the following code .

```
require(ISLR)
data("Auto")
fit.lm <- lm(mpg ~ horsepower, data = Auto)
summary(fit.lm)
```

```
##
## Call:
## lm(formula = mpg ~ horsepower, data = Auto)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5710  -3.2592  -0.3435   2.7630  16.9240
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 39.935861   0.717499   55.66   <2e-16 ***
## horsepower  -0.157845   0.006446  -24.49   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.906 on 390 degrees of freedom
## Multiple R-squared:  0.6059, Adjusted R-squared:  0.6049
## F-statistic: 599.7 on 1 and 390 DF,  p-value: < 2.2e-16
```

What is the predicted mpg associated with a horsepower of 98? What are the associated 95% confidence and prediction intervals?

```
new <- data.frame(horsepower = 98)
predict(fit.lm, new)
```

```
##        1
## 24.46708
```

```
predict(fit.lm, new, interval = "confidence")
```

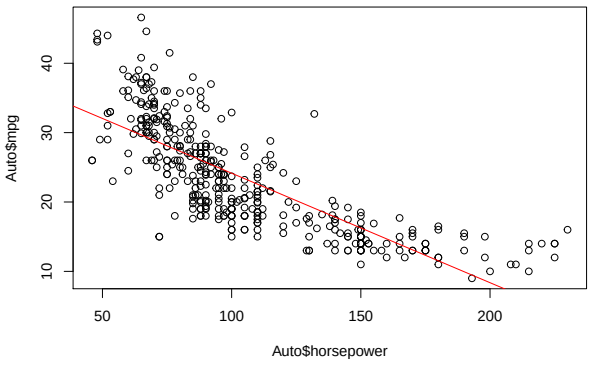
```
##        fit      lwr      upr
## 1 24.46708 23.97308 24.96108
```

```
predict(fit.lm, new, interval = "prediction")
```

```
##        fit     lwr      upr
## 1 24.46708 14.8094 34.12476
```

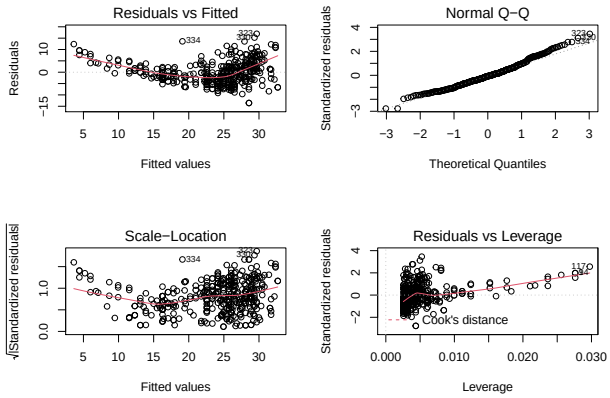
Ploting the response and the predictor.

```
plot(Auto$horsepower, Auto$mpg)
abline(fit.lm, col="red")
```



Using the plot() function to produce diagnostic plots of the least squares regression fit.

```
par(mfrow=c(2,2))
plot(fit.lm)
```



residuals vs fitted plot shows that the relationship is non-linear

Simple linear regression in Python

```
import pandas as pd
import numpy as np
from pandas import DataFrame as df
import statsmodels.api as sm
import matplotlib.pyplot as plt
import os
```

```
df = pd.read_csv (r'C:\Users\madhu\OneDrive\Desktop\Auto.csv')

df.head()
```

```
##     mpg  cylinders  displacement  ... year  origin                       name
## 0  18.0          8         307.0  ...   70       1  chevrolet chevelle malibu
## 1  15.0          8         350.0  ...   70       1          buick skylark 320
## 2  18.0          8         318.0  ...   70       1         plymouth satellite
## 3  16.0          8         304.0  ...   70       1              amc rebel sst
## 4  17.0          8         302.0  ...   70       1                ford torino
##
## [5 rows x 9 columns]
```

```
X,y= df["horsepower"] , df["mpg"]
```

```
X = sm.add_constant(X)
np.asanyarray(df)
```

```
## array([[18.0, 8, 307.0, ..., 70, 1, 'chevrolet chevelle malibu'],
##        [15.0, 8, 350.0, ..., 70, 1, 'buick skylark 320'],
```

```
##         [18.0, 8, 318.0, ..., 70, 1, 'plymouth satellite'],
##         ...,
##         [32.0, 4, 135.0, ..., 82, 1, 'dodge rampage'],
##         [28.0, 4, 120.0, ..., 82, 1, 'ford ranger'],
##         [31.0, 4, 119.0, ..., 82, 1, 'chevy s-10']], dtype=object)
```

```
#model = sm.OLS(y,X).fit()
```

#“‘{python} fix, ax = plt.subplots()

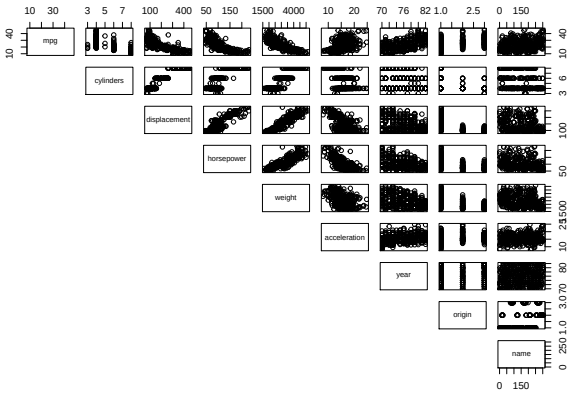
ax.scatter(df[“horsepower”], model.predict(), label=‘fitted’) ax.scatter(df[“horsepower”], df[“mpg”], label=‘original’)

ax.legend() ax.set_title(“Linear model fitted values vs the original dataset”) ax.set_xlabel(“horsepower”) ax.set_ylabel(“mpg”) plt.show()

Multiple linear regression in R

> Produce a scatterplot matrix which includes all of the variables in the data set.

‘‘‘r
pairs(Auto,lower.panel = NULL)



Computing the matrix of correlations between the variables using the function cor() by exclude the name variable, which is qualitative.

```
cor(subset(Auto, select=-name))
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
```

```
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
##              acceleration       year     origin
## mpg             0.4233285  0.5805410  0.5652088
## cylinders      -0.5046834 -0.3456474 -0.5689316
## displacement   -0.5438005 -0.3698552 -0.6145351
## horsepower     -0.6891955 -0.4163615 -0.4551715
## weight         -0.4168392 -0.3091199 -0.5850054
## acceleration    1.0000000  0.2903161  0.2127458
## year            0.2903161  1.0000000  0.1815277
## origin          0.2127458  0.1815277  1.0000000
```

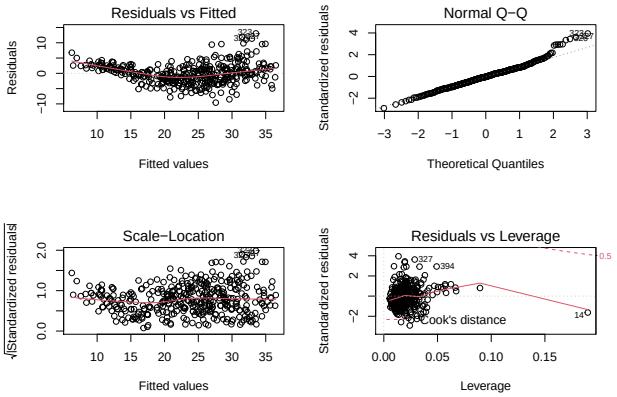
Using the lm() function to perform a multiple linear regression with mpg as the response and all other variables except name as the predictors.

```
fit.lm <- lm(mpg~.-name, data=Auto)
summary(fit.lm)
```

```
##
## Call:
## lm(formula = mpg ~ . - name, data = Auto)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.5903 -2.1565 -0.1169  1.8690 13.0604
##
## Coefficients:
##                Estimate Std. Error t value    Pr(>|t|)
## (Intercept)  -17.218435   4.644294  -3.707     0.00024 ***
## cylinders     -0.493376   0.323282  -1.526     0.12780
## displacement   0.019896   0.007515   2.647     0.00844 **
## horsepower    -0.016951   0.013787  -1.230     0.21963
## weight        -0.006474   0.000652  -9.929     < 2e-16 ***
## acceleration   0.080576   0.098845   0.815     0.41548
## year           0.750773   0.050973  14.729     < 2e-16 ***
## origin         1.426141   0.278136   5.127 0.000000467 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.328 on 384 degrees of freedom
## Multiple R-squared:  0.8215, Adjusted R-squared:  0.8182
## F-statistic: 252.4 on 7 and 384 DF,  p-value: < 2.2e-16
```

Use the 'plot() function to produce diagnostic plots of the linear regression fit.

```
par(mfrow=c(2,2))
plot(fit.lm)
```



Multiple linear regression in Python.

```
X = df.drop(labels=['name','mpg'],axis=1)
y=df['mpg']
```

#“‘{python}

X = sm.add_constant(X) #adding constant for intercept model =sm.OLS(y,X).fit() model.summary() “‘