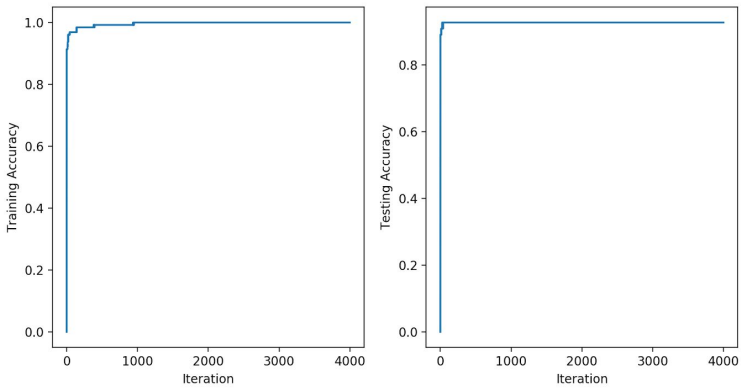
# Comaprison between Neural Networks and AdaBoost /SVM Results

**Data used: digits.csv ( Training: 70%;  Testing: 30%)**
**Programming Language: Python**

1. **AdaBoost Results** ( Training and Testing Accuracy):

```
# training 255
# testing 110
After iteration 100, Train Accuracy = 0.968627,Test Accuracy = 0.927273
After iteration 200, Train Accuracy = 0.984314,Test Accuracy = 0.927273
After iteration 300, Train Accuracy = 0.984314,Test Accuracy = 0.927273
After iteration 400, Train Accuracy = 0.992157,Test Accuracy = 0.927273
After iteration 500, Train Accuracy = 0.992157,Test Accuracy = 0.927273
After iteration 600, Train Accuracy = 0.992157,Test Accuracy = 0.927273
After iteration 700, Train Accuracy = 0.992157,Test Accuracy = 0.927273
After iteration 800, Train Accuracy = 0.992157,Test Accuracy = 0.927273
After iteration 900, Train Accuracy = 0.992157,Test Accuracy = 0.927273
After iteration 1000, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1100, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1200, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1300, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1400, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1500, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1600, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1700, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1800, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 1900, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2000, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2100, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2200, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2300, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2400, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2500, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2600, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2700, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2800, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 2900, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3000, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3100, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3200, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3300, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3400, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3500, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3600, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3700, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3800, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 3900, Train Accuracy = 1.000000,Test Accuracy = 0.927273
After iteration 4000, Train Accuracy = 1.000000,Test Accuracy = 0.927273
```

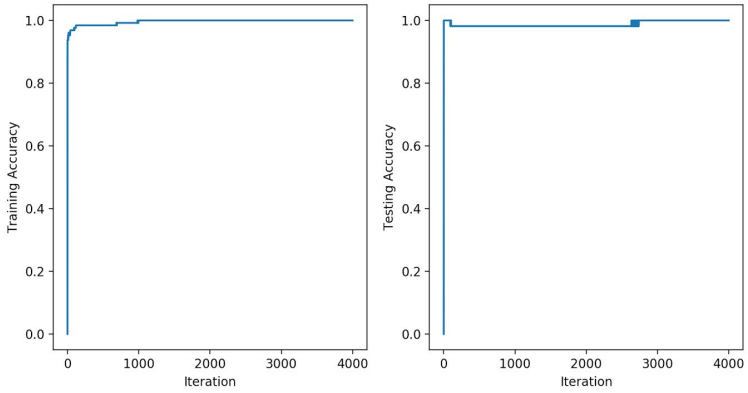
2. **AdaBoost Plots** (Training Accuracy and Testing Accuracy V/s. Iteration):



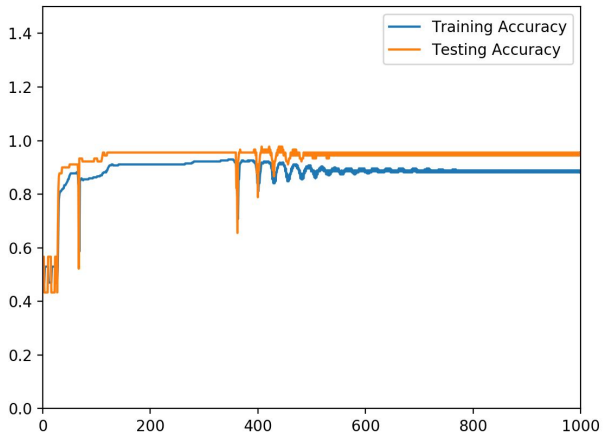
3. **SVM results**(Training and Testing Accuracy)**:**

```
# training 255
# testing 110
After iteration 100, Train Accuracy = 0.976471,Test Accuracy = 0.981818
After iteration 200, Train Accuracy = 0.984314,Test Accuracy = 0.981818
After iteration 300, Train Accuracy = 0.984314,Test Accuracy = 0.981818
After iteration 400, Train Accuracy = 0.984314,Test Accuracy = 0.981818
After iteration 500, Train Accuracy = 0.984314,Test Accuracy = 0.981818
After iteration 600, Train Accuracy = 0.984314,Test Accuracy = 0.981818
After iteration 700, Train Accuracy = 0.992157,Test Accuracy = 0.981818
After iteration 800, Train Accuracy = 0.992157,Test Accuracy = 0.981818
After iteration 900, Train Accuracy = 0.992157,Test Accuracy = 0.981818
After iteration 1000, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1100, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1200, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1300, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1400, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1500, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1600, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1700, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1800, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 1900, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 2000, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 2100, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 2200, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 2300, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 2400, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 2500, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 2600, Train Accuracy = 1.000000,Test Accuracy = 0.981818
After iteration 2700, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 2800, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 2900, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3000, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3100, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3200, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3300, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3400, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3500, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3600, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3700, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3800, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 3900, Train Accuracy = 1.000000,Test Accuracy = 1.000000
After iteration 4000, Train Accuracy = 1.000000,Test Accuracy = 1.000000
```

4. **SVM Plots** (Training and Testing Accuracy V/s. Iteration):



5. Neural Networks:




```
('Iteration ', 0, 'Training Accuracy: ', 0.51481481481481484, 'Testing Accuracy: ', 0.56666666666666665)
2layer_nn.py:114: RuntimeWarning: overflow encountered in exp
  prtest = 1 / (1 + np.exp((-1) * Ztest1.dot(beta)))
2layer_nn.py:97: RuntimeWarning: overflow encountered in exp
  pr = 1 / (1 + np.exp((-1) * Z1.dot(beta)))
('Iteration ', 20, 'Training Accuracy: ', 0.52962962962962967, 'Testing Accuracy: ', 0.43333333333333335)
('Iteration ', 40, 'Training Accuracy: ', 0.83333333333333337, 'Testing Accuracy: ', 0.90000000000000002)
('Iteration ', 60, 'Training Accuracy: ', 0.87777777777777777, 'Testing Accuracy: ', 0.91111111111111109)
('Iteration ', 80, 'Training Accuracy: ', 0.85555555555555551, 'Testing Accuracy: ', 0.92222222222222228)
('Iteration ', 100, 'Training Accuracy: ', 0.86296296296296293, 'Testing Accuracy: ', 0.93333333333333335)
('Iteration ', 120, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 140, 'Training Accuracy: ', 0.90740740740740744, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 160, 'Training Accuracy: ', 0.91111111111111109, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 180, 'Training Accuracy: ', 0.91111111111111109, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 200, 'Training Accuracy: ', 0.91111111111111109, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 220, 'Training Accuracy: ', 0.91111111111111109, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 240, 'Training Accuracy: ', 0.91111111111111109, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 260, 'Training Accuracy: ', 0.91111111111111109, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 280, 'Training Accuracy: ', 0.91851851851851851, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 300, 'Training Accuracy: ', 0.92222222222222228, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 320, 'Training Accuracy: ', 0.92222222222222228, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 340, 'Training Accuracy: ', 0.92592592592592593, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 360, 'Training Accuracy: ', 0.90000000000000002, 'Testing Accuracy: ', 0.82222222222222219)
('Iteration ', 380, 'Training Accuracy: ', 0.92222222222222228, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 400, 'Training Accuracy: ', 0.85925925925925928, 'Testing Accuracy: ', 0.78888888888888886)
('Iteration ', 420, 'Training Accuracy: ', 0.92222222222222228, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 440, 'Training Accuracy: ', 0.91111111111111109, 'Testing Accuracy: ', 0.9555555555555556)
('Iteration ', 460, 'Training Accuracy: ', 0.87777777777777777, 'Testing Accuracy: ', 0.93333333333333335)
('Iteration ', 480, 'Training Accuracy: ', 0.87777777777777777, 'Testing Accuracy: ', 0.93333333333333335)
('Iteration ', 500, 'Training Accuracy: ', 0.89629629629629626, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 520, 'Training Accuracy: ', 0.90370370370370368, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 540, 'Training Accuracy: ', 0.89629629629629626, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 560, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 580, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 600, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 620, 'Training Accuracy: ', 0.89629629629629626, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 640, 'Training Accuracy: ', 0.89629629629629626, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 660, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 680, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 700, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 720, 'Training Accuracy: ', 0.8925925925925926, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 740, 'Training Accuracy: ', 0.8925925925925926, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 760, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 780, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 800, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 820, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 840, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 860, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 880, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 900, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 920, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 940, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 960, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
('Iteration ', 980, 'Training Accuracy: ', 0.88888888888888884, 'Testing Accuracy: ', 0.94444444444444442)
```

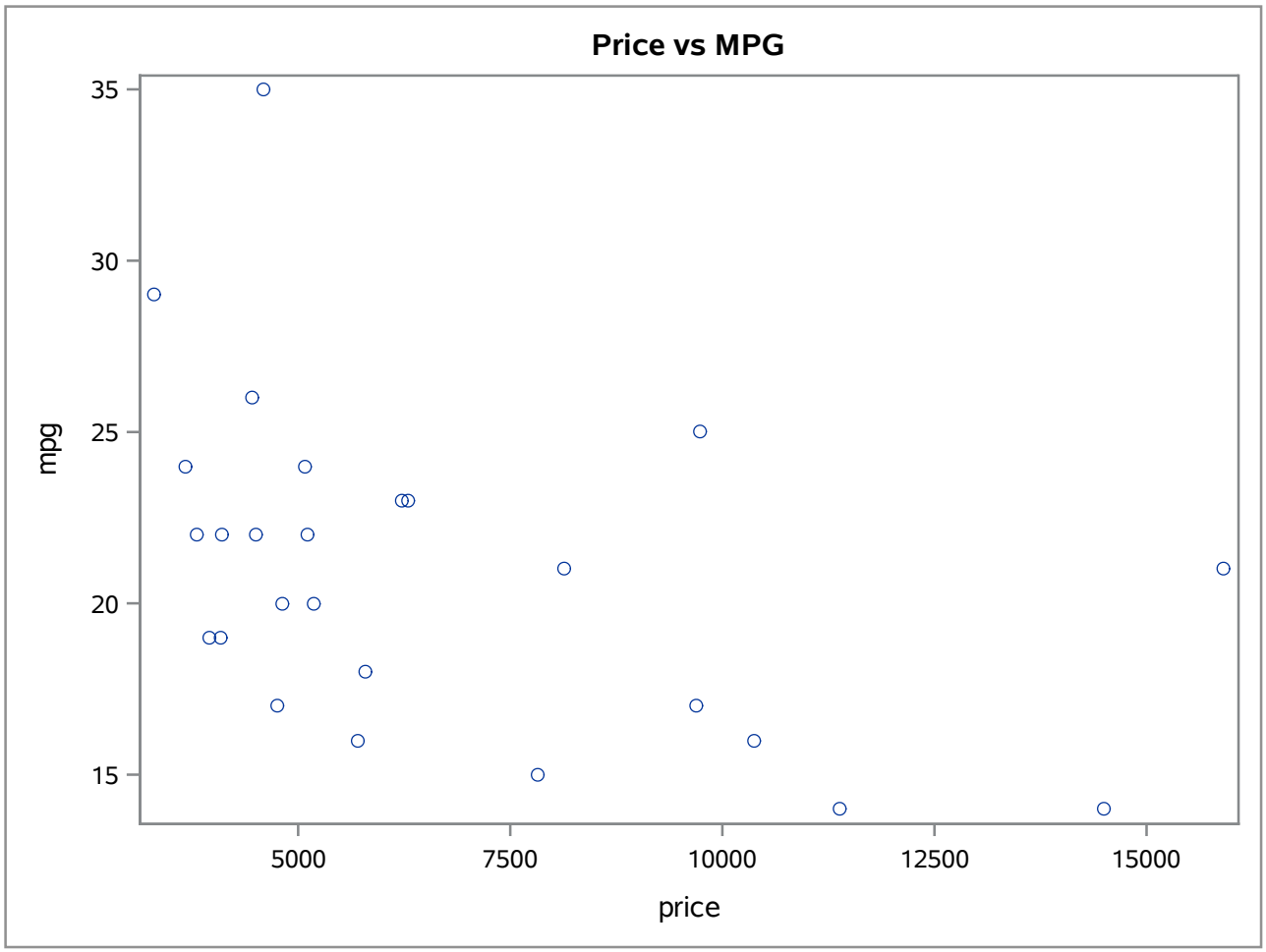
**SVM has the best accuracy while neural networks is the slowest.**

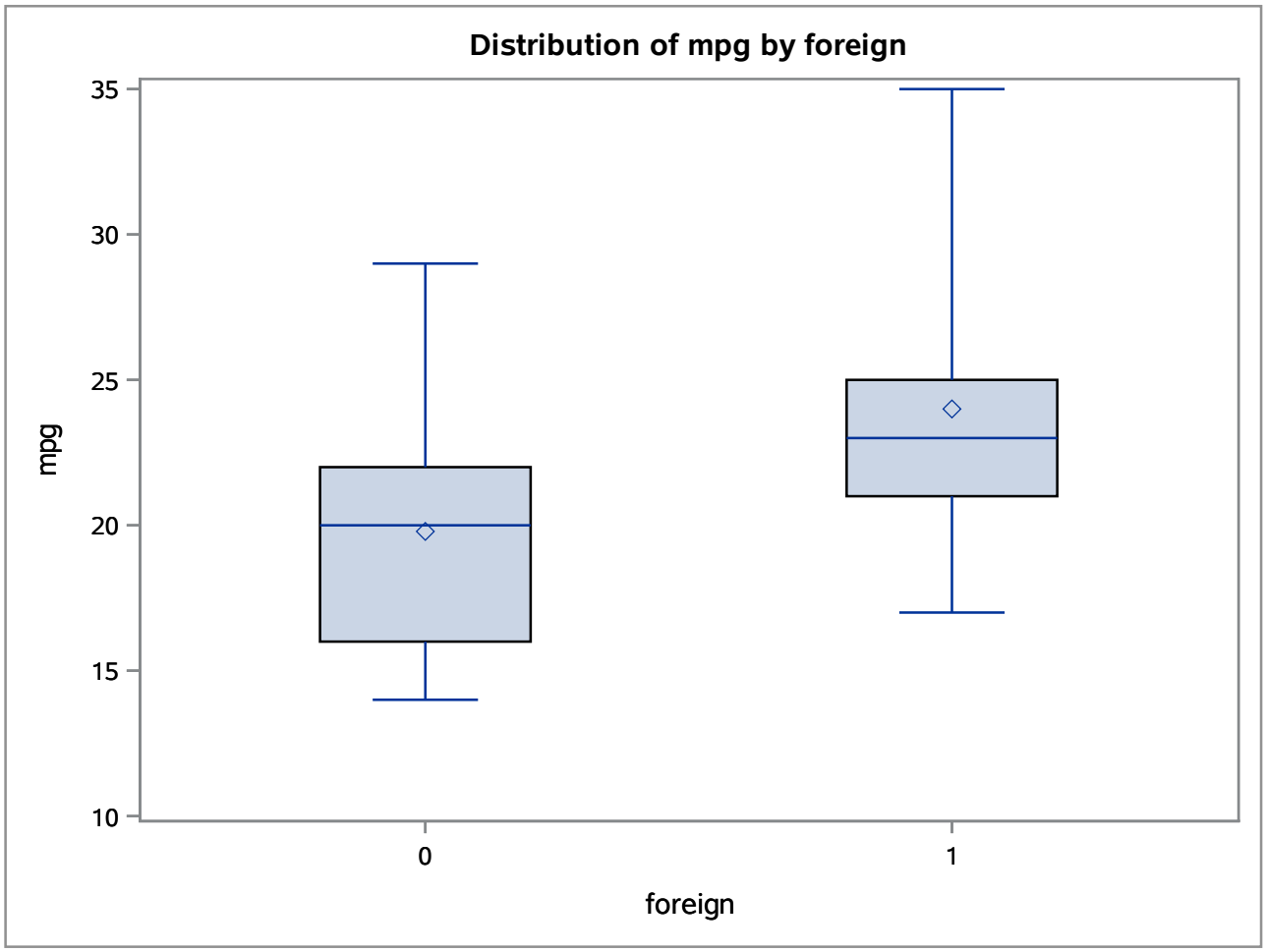
## The CORR Procedure

| 2 Variables: | length mpg |
|---|---|

| Simple Statistics | | | | | | |
|---|---|---|---|---|---|---|
| Variable | N | Mean | Std Dev | Sum | Minimum | Maximum |
| length | 26 | 190.07692 | 18.17014 | 4942 | 163.00000 | 222.00000 |
| mpg | 26 | 20.92308 | 4.75750 | 544.00000 | 14.00000 | 35.00000 |

| Pearson Correlation Coefficients, N = 26<br>Prob > \|r\| under H0: Rho=0 | | |
|---|---|---|
| | length | mpg |
| length | 1.00000 | -0.76805<br><.0001 |
| mpg | -0.76805<br><.0001 | 1.00000 |


Price vs MPG
35
30
25
20
15
mpg
5000
7500
10000
12500
15000
price


Distribution of mpg by foreign
35
30
25
20
15
10
mpg
0
1
foreign

## The REG Procedure
## Model: MODEL1
## Dependent Variable: mpg

| | |
|---|---|
| **Number of Observations Read** | 26 |
| **Number of Observations Used** | 26 |

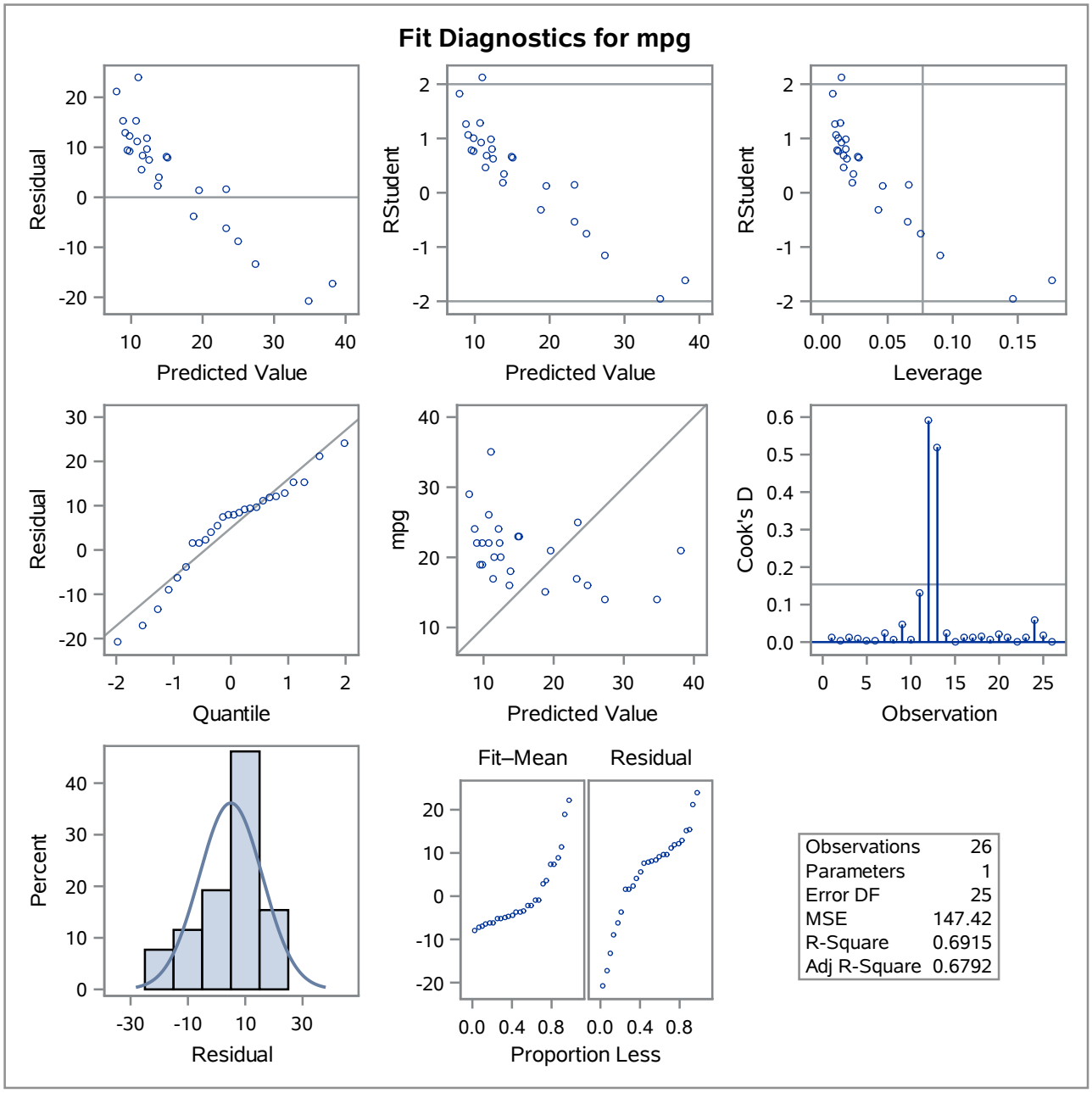
**Note:** No intercept in model. R-Square is redefined.

| Analysis of Variance | | | | | |
|---|---|---|---|---|---|
| **Source** | **DF** | **Sum of Squares** | **Mean Square** | **F Value** | **Pr > F** |
| **Model** | 1 | 8262.45717 | 8262.45717 | 56.05 | <.0001 |
| **Error** | 25 | 3685.54283 | 147.42171 | | |
| **Uncorrected Total** | 26 | 11948 | | | |

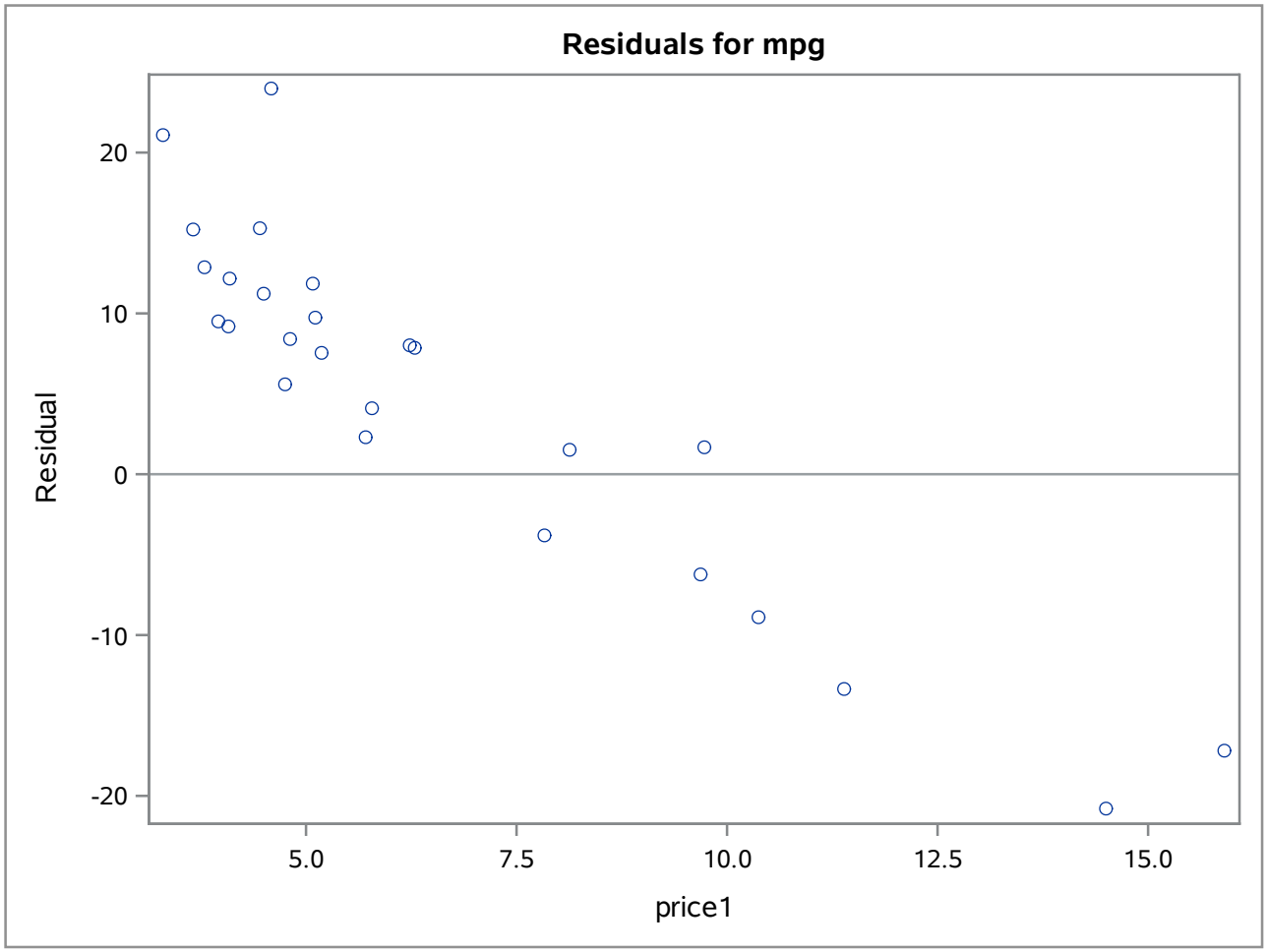
| | | | |
|---|---|---|---|
| **Root MSE** | 12.14173 | **R-Square** | 0.6915 |
| **Dependent Mean** | 20.92308 | **Adj R-Sq** | 0.6792 |
| **Coeff Var** | 58.03035 | | |

| Parameter Estimates | | | | | |
|---|---|---|---|---|---|
| **Variable** | **DF** | **Parameter Estimate** | **Standard Error** | **t Value** | **Pr > \|t\|** |
| **price1** | 1 | 2.39997 | 0.32058 | 7.49 | <.0001 |

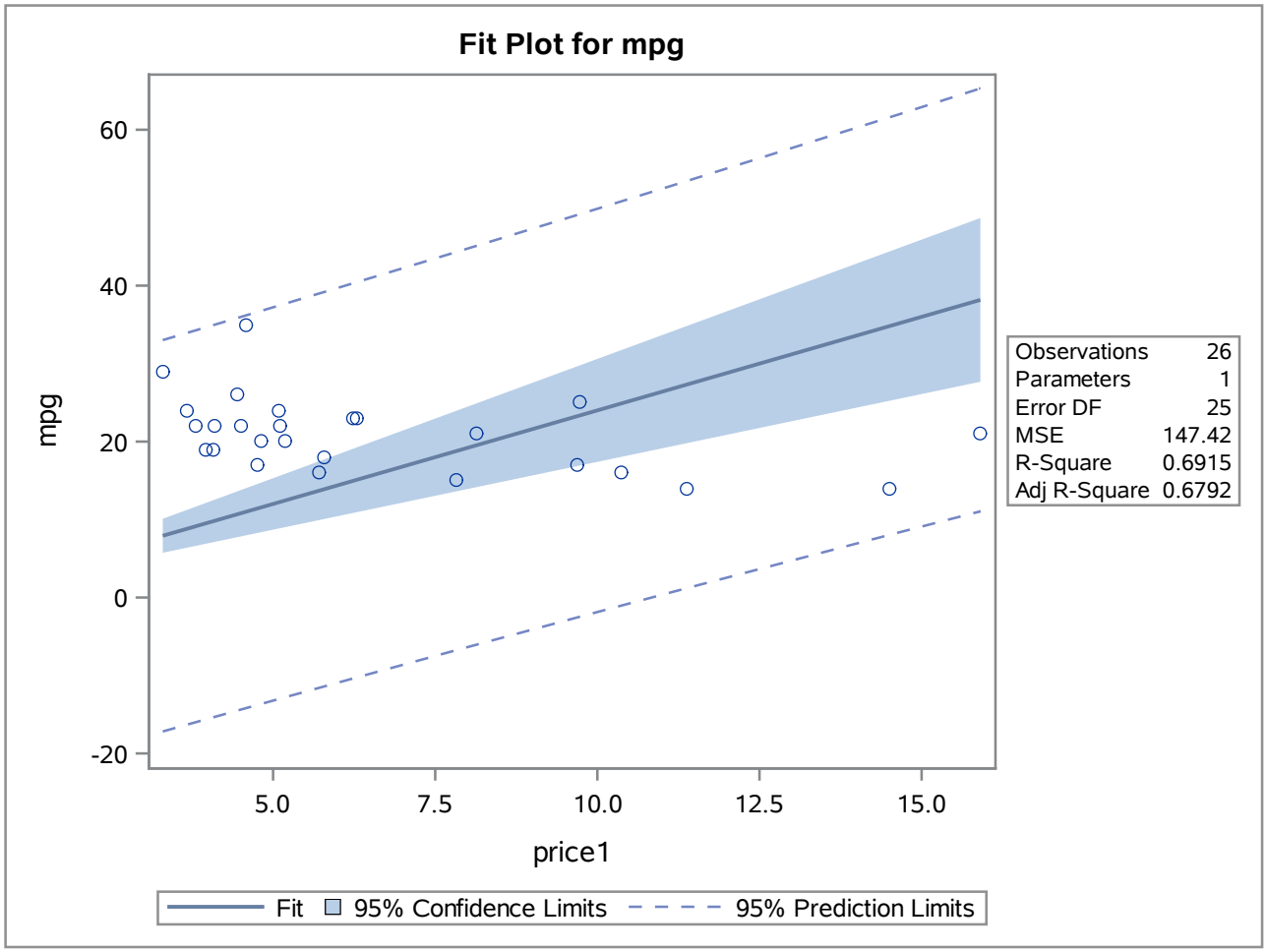
**The REG Procedure**
**Model: MODEL1**
**Dependent Variable: mpg**



**The REG Procedure**
**Model: MODEL1**
**Dependent Variable: mpg**



**The REG Procedure**
**Model: MODEL1**
**Dependent Variable: mpg**



## Linear regression Y= MPG X1 = Length X2 = Length^2

### The GLM Procedure

| | |
|---|---|
| **Number of Observations Read** | 26 |
| **Number of Observations Used** | 26 |

# Linear regression Y= MPG X1 = Length X2 = Length^2

## The GLM Procedure

## Dependent Variable: mpg

| Source | DF | Sum of Squares | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| Model | 2 | 343.2308092 | 171.6154046 | 17.73 | <.0001 |
| Error | 23 | 222.6153446 | 9.6789280 | | |
| Corrected Total | 25 | 565.8461538 | | | |

| R-Square | Coeff Var | Root MSE | mpg Mean |
|---|---|---|---|
| 0.606580 | 14.86922 | 3.111098 | 20.92308 |

| Source | DF | Type I SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| length | 1 | 333.7945975 | 333.7945975 | 34.49 | <.0001 |
| length*length | 1 | 9.4362118 | 9.4362118 | 0.97 | 0.3337 |

| Source | DF | Type III SS | Mean Square | F Value | Pr > F |
|---|---|---|---|---|---|
| length | 1 | 14.72375568 | 14.72375568 | 1.52 | 0.2299 |
| length*length | 1 | 9.43621177 | 9.43621177 | 0.97 | 0.3337 |

| Parameter | Estimate | Standard Error | t Value | Pr > \|t\| |
|---|---|---|---|---|
| Intercept | 135.4470552 | 77.55066223 | 1.75 | 0.0941 |
| length | -1.0047165 | 0.81460651 | -1.23 | 0.2299 |
| length*length | 0.0020976 | 0.00212437 | 0.99 | 0.3337 |

**Linear regression Y= MPG X1 = Length X2 = Length^2**

**The GLM Procedure**

**Dependent Variable: mpg**