19MIS1018_SWE4012 - MACHINE LEARNING_LAB-3_SIMPLE LINEAR REGRESSION

August 9, 2022

NAME : B DEVI PRASAD

REG NO : 19MIS1018

SLOT: L13+L14

FACULTY: Prof. BHARADWAJA KUMAR

1 TOPIC : SIMPLE LINEAR REGRESSION

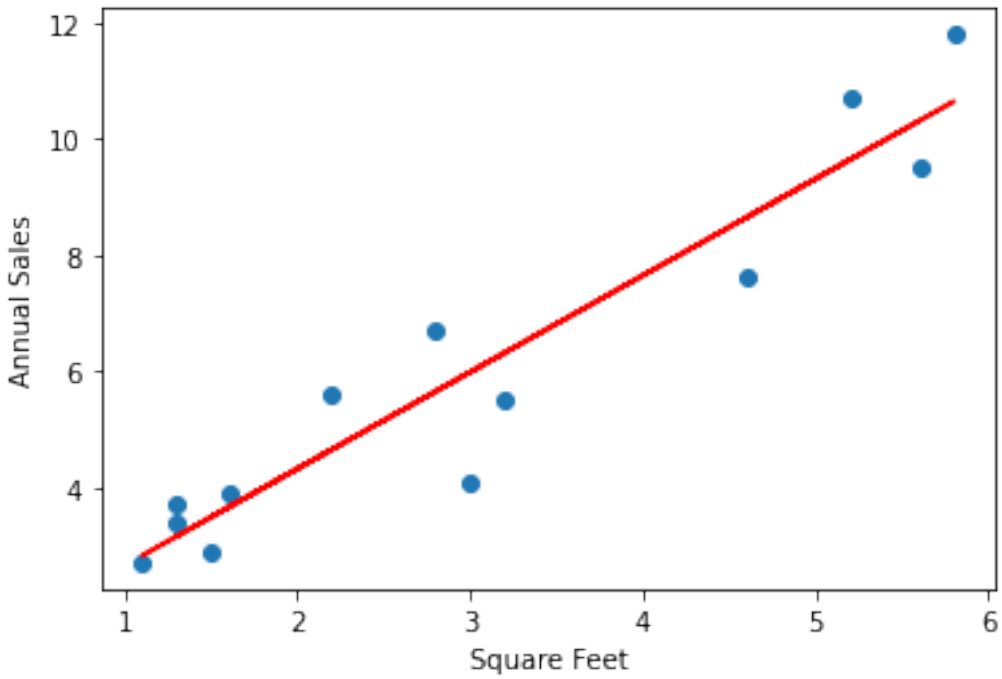
[1]:
```
import math
import csv
import pylab

# Finding Regression parameters using the formulas

def mean(Xs):
    return sum(Xs) / len(Xs)

def std(Xs, m):
    normalizer = len(Xs) - 1
    return math.sqrt(sum((pow(x - m, 2) for x in Xs)) / normalizer)

def fit(X,Y):
    m_X = mean(X)
    m_Y = mean(Y)

    def cov(Xs, Ys):
        sum_xy = 0
        for (x, y) in zip(Xs, Ys):
            meandiff_x = x - m_X
            meandiff_y = y - m_Y
            sum_xy +=  meandiff_x *  meandiff_y
        return sum_xy

    def var(Xs):
        sumv = 0
```

```
        for x in Xs:
            meandiff_x = x - m_X
            sumv += pow( meandiff_x,2)
        return sumv

    covr=cov(X,Y)
    vari=var(X)
    b1=cov(X,Y)/var(X)
    b0 = m_Y - b1 * m_X

    return b0,b1

def read_data(filename):

    with open(filename, 'r') as csvfile:
        datareader = csv.reader(csvfile, delimiter=',')
        headers = next(datareader)
        metadata = []
        traindata = []
        for name in headers:
            metadata.append(name)

        for row in datareader:
            traindata.append([float(x) for x in row])

        return (metadata, traindata)


metadata, traindata = read_data("sales.csv")

X = []
Y= []
for i in traindata:
    X.append(i[0])
    Y.append(i[1])


b0,b1=fit(X,Y)

print (" Estimated parameters of Regression:\t  b0 = %2.4f  b1 = %2.4f"%␣
 ↪(b0,b1))


# Predicting the value for x=4


def predict(x):
```

```
    val=1.6699*x+0.9645
    return val

test=4
final=predict(test)

print("Predicted y =", final ,"for x = ", test )


# Finding Correlation between X and Y

def pearson_r(Xs, Ys):
    sum_xy = 0
    sum_sq_v_x = 0
    sum_sq_v_y = 0
    m_X = mean(X)
    m_Y = mean(Y)
    for (x, y) in zip(Xs, Ys):
        var_x = x - m_X
        var_y = y - m_Y
        sum_xy += var_x * var_y
        sum_sq_v_x += pow(var_x, 2)
        sum_sq_v_y += pow(var_y, 2)
    return sum_xy / math.sqrt(sum_sq_v_x * sum_sq_v_y)


r = pearson_r(X, Y)

print("Correlation between Square Feet  and Annual Sales", r)

print("Coefficient of determination", pow(r,2))


# Finding T-test value for 12 degrees of freedom

def tvalue(r,length):
    ttestval = 0.0
    ttestval = r*(math.sqrt((length-2)/(1-pow(r,2))))
    return ttestval

length=len(Y)
ttestval=tvalue(r,length)

print("T Test Value = ", ttestval )


# Plotting the regression Line
```

```
intercept=b0
slope=b1
y_p=[]
e_p=[]

for x in X:
    yp = intercept + slope * x
    y_p.append(yp)

pylab.xlabel("Square Feet")
pylab.ylabel("Annual Sales")
pylab.plot(X, Y, 'o')
pylab.plot(X, y_p, 'r-')
pylab.show()


# Finding Residuals and plotting Residuals


for i in range(0,14):
    error = Y[i] - y_p[i]
    e_p.append(error)

pylab.xlabel("Predicted Annual Sales")
pylab.ylabel("Residuals")
pylab.plot(y_p,e_p, 'o')
pylab.show()
```

```
 Estimated parameters of Regression:      b0 = 0.9837  b1 = 1.6661
Predicted y = 7.6441 for x =  4
Correlation between Square Feet  and Annual Sales 0.9487126260383095
Coefficient of determination 0.9000556468045053
T Test Value =  9.952951600714048
```




```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Input In [1], in <module>
    139 # Finding Residuals and plotting Residuals
    142 for i in range(0,14):
--> 143     error = Y[i] - y_p[i]
    144     e_p.append(error)
    146 pylab.xlabel("Predicted Annual Sales")

IndexError: list index out of range
```

2 TOPIC : SIMPLE LINEAR REGRESSION USING SKLEARN

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
```

[3]:
```
data = pd.read_csv("FuelEconomy.csv")
data.head()
```

[3]:

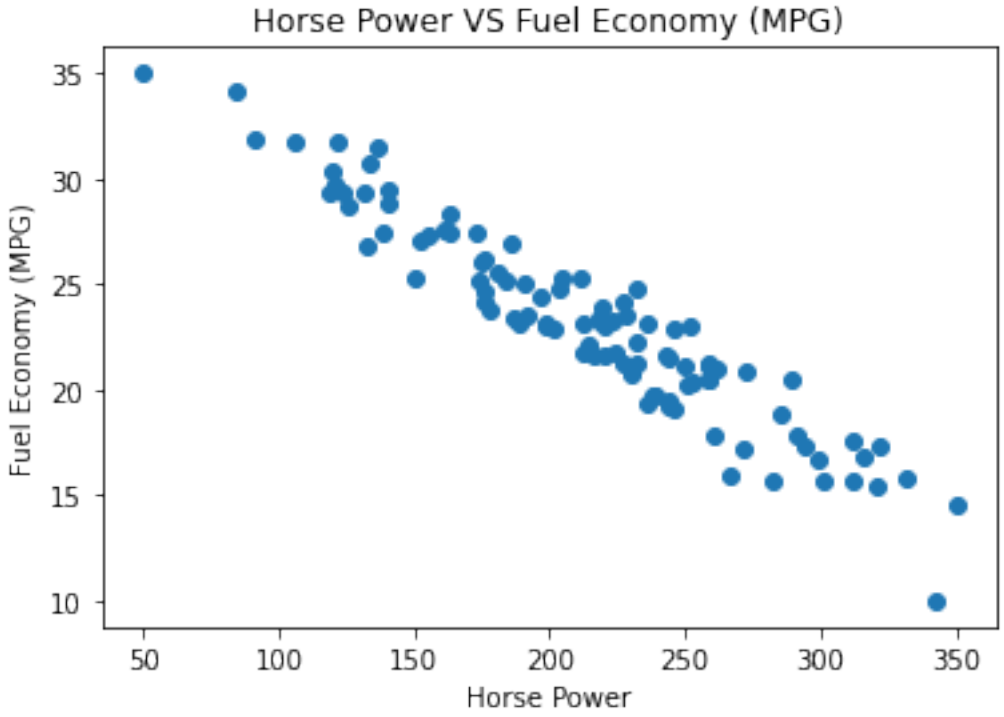
	Horse Power	Fuel Economy (MPG)
0	118.770799	29.344195

```
1    176.326567           24.695934
2    219.262465           23.952010
3    187.310009           23.384546
4    218.594340           23.426739
```

[4]:
```
data = data[["Horse Power","Fuel Economy (MPG)"]]
print(data.head())
```

```
   Horse Power  Fuel Economy (MPG)
0   118.770799           29.344195
1   176.326567           24.695934
2   219.262465           23.952010
3   187.310009           23.384546
4   218.594340           23.426739
```

[5]:
```
#Scatter Plot:
#ENGINESIZE VS CO2EMISSIONS :

plt.scatter(data["Horse Power"],data["Fuel Economy (MPG)"])
plt.title("Horse Power VS Fuel Economy (MPG)")
plt.xlabel("Horse Power")
plt.ylabel("Fuel Economy (MPG)")
plt.show()
```



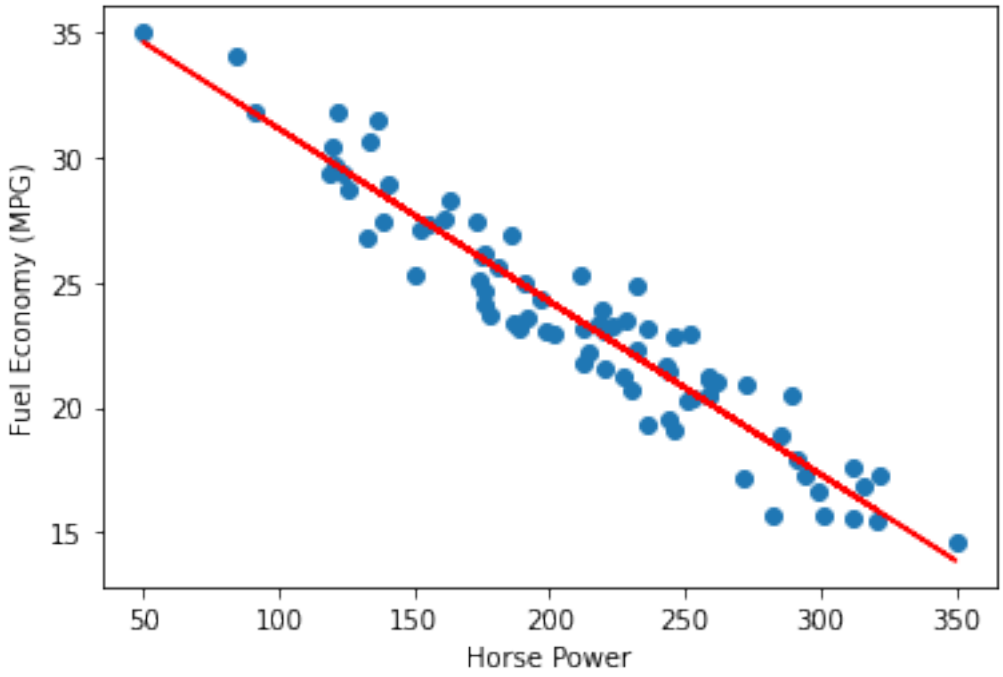
[6]:
```
#Taking 80% of the data for training and 20% for testing:
num = int(len(data)*0.8)

#Training data:
train = data[:num]

#Testing data:
test = data[num:]

print ("Data: ",len(data))
print ("Train: ",len(train))
print ("Test: ",len(test))
```

```
Data:  100
Train:  80
Test:  20
```

[7]:
```
#Training the model:

regr = linear_model.LinearRegression()

train_x = np.array(train[["Horse Power"]])
train_y = np.array(train[["Fuel Economy (MPG)"]])

regr.fit(train_x,train_y)

coefficients = regr.coef_
intercept = regr.intercept_

#Slope:
print ("Slope: ",coefficients[0])

#Intercept:
print("Intercept: ",intercept)
```

```
Slope:  [-0.06915471]
Intercept:  [38.07730221]
```

[10]:
```
#Plotting the regression line with train data:

plt.scatter(train["Horse Power"], train["Fuel Economy (MPG)"])
plt.plot(train_x, coefficients[0]*train_x + intercept,color="red")
plt.xlabel("Horse Power")
plt.ylabel("Fuel Economy (MPG)")
plt.show()
```



[11]:
```
regr.predict(data[["Horse Power"]])
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:443: UserWarning: X has feature names, but
LinearRegression was fitted without feature names
  warnings.warn(
```

[11]:
```
array([[29.8637424 ],
       [25.88349008],
       [22.91427067],
       [25.1239334 ],
       [22.96047468],
       [25.91726947],
       [19.30583732],
       [17.71636423],
       [29.34921691],
       [26.78085766],
       [15.82049927],
       [29.74525016],
       [27.32959798],
       [24.81932044],
       [23.43523767],
       [20.15356451],
       [21.71725299],
       [24.86190938],
```

```
[29.51002303],
[28.65106936],
[23.36540199],
[22.00229414],
[29.6376506 ],
[28.36574673],
[21.24865408],
[25.73800778],
[20.20605073],
[22.04344439],
[31.75377753],
[29.81077963],
[13.87315472],
[25.90751084],
[16.46699498],
[18.53388832],
[28.51564513],
[17.25798844],
[22.31877683],
[34.61956685],
[27.54173077],
[20.61211594],
[22.9941709 ],
[26.13818037],
[21.22172462],
[25.2117574 ],
[22.33281786],
[28.90257699],
[24.46353828],
[21.17877258],
[26.91496685],
[28.82280764],
[27.66908676],
[17.96108311],
[21.077664  ],
[20.73957478],
[19.20458546],
[24.30530562],
[20.69011415],
[18.37314871],
[24.13388342],
[26.04857973],
[22.13972325],
[18.05770582],
[22.81340983],
[23.41076177],
[25.04127847],
```

```
       [25.55560228],
       [21.09569646],
       [32.22875085],
       [16.51780829],
       [16.23703561],
       [22.64337893],
       [17.36336113],
       [25.95109949],
       [19.96899624],
       [20.21406423],
       [22.85644896],
       [23.24468991],
       [15.88200507],
       [20.18975411],
       [21.72467249],
       [24.31839821],
       [20.79664762],
       [23.12043877],
       [22.01843076],
       [22.56299149],
       [21.56063472],
       [15.13878305],
       [30.72909194],
       [23.99936407],
       [14.36372168],
       [21.64108961],
       [23.89735103],
       [22.35289631],
       [28.9805457 ],
       [20.05043736],
       [26.81818689],
       [19.62201044],
       [21.2152262 ],
       [28.32926834],
       [25.34432642]])
```

[12]:
```
predicted_data = regr.predict(data[["Horse Power"]])
predicted_data[0:5]
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:443: UserWarning: X has feature names, but
LinearRegression was fitted without feature names
  warnings.warn(
```

[12]:
```
array([[29.8637424 ],
       [25.88349008],
       [22.91427067],
       [25.1239334 ],
```

```
       [22.96047468]])
```

[13]:
```
predicted_train = regr.predict(train[["Horse Power"]])
predicted_train[0:5]
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:443: UserWarning: X has feature names, but
LinearRegression was fitted without feature names
  warnings.warn(
```

[13]:
```
array([[29.8637424 ],
       [25.88349008],
       [22.91427067],
       [25.1239334 ],
       [22.96047468]])
```

[14]:
```
predicted_test = regr.predict(test[["Horse Power"]])
predicted_test[0:5]
```

```
C:\Users\admin\AppData\Local\Programs\Python\Python39\lib\site-
packages\sklearn\base.py:443: UserWarning: X has feature names, but
LinearRegression was fitted without feature names
  warnings.warn(
```

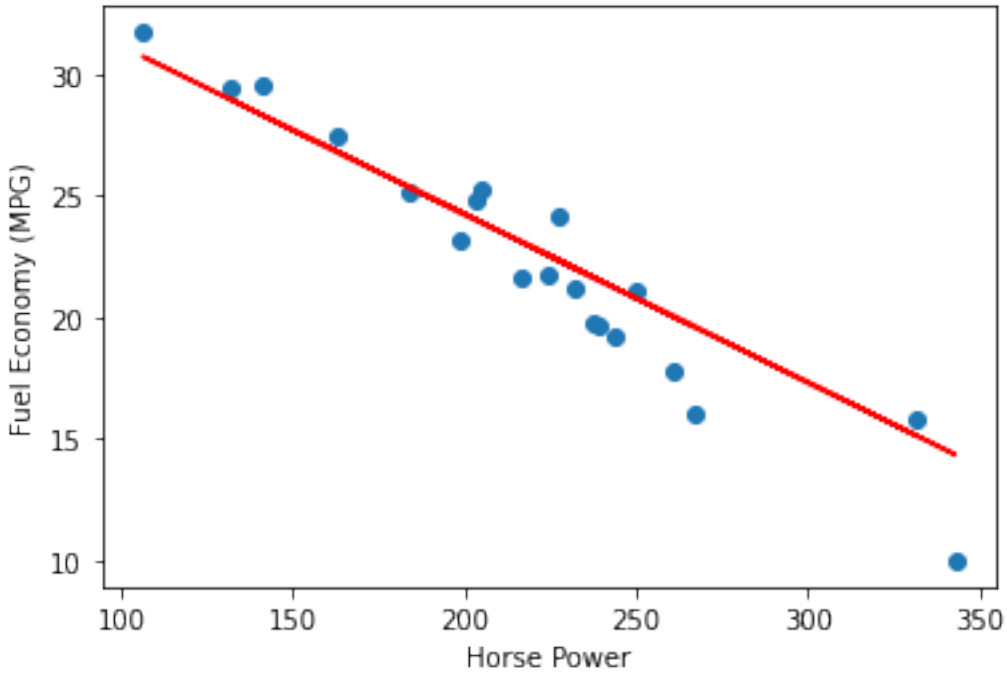
[14]:
```
array([[24.31839821],
       [20.79664762],
       [23.12043877],
       [22.01843076],
       [22.56299149]])
```

[15]:
```
#Plot the regression line for training data:

plt.scatter(train["Horse Power"],train["Fuel Economy (MPG)]")
plt.plot(train["Horse Power"],predicted_train,color="red")
plt.xlabel("Engine_Size")
plt.ylabel("Fuel Economy (MPG)")
plt.show()
```

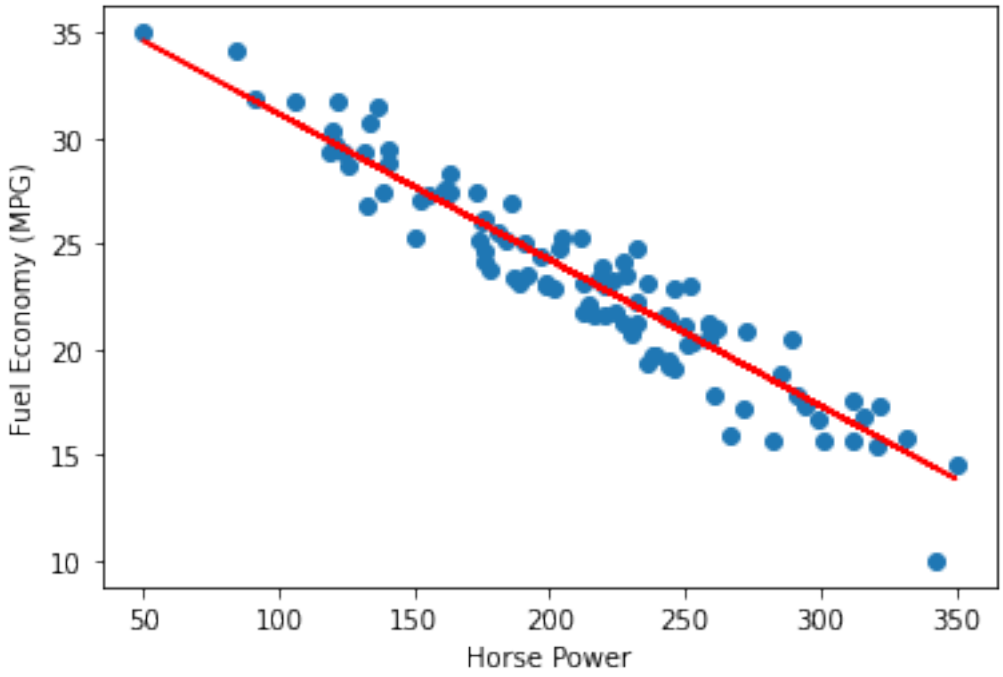
```
  Input In [15]
    plt.scatter(train["Horse Power"],train["Fuel Economy (MPG)]")
                                                               ^
SyntaxError: closing parenthesis ')' does not match opening parenthesis '['
```

[16]:
```
#Plot the regression line for testing data:

plt.scatter(test["Horse Power"],test["Fuel Economy (MPG)"])
```

```
plt.plot(test["Horse Power"],predicted_test,color="red")
plt.xlabel("Horse Power")
plt.ylabel("Fuel Economy (MPG)")
plt.show()
```



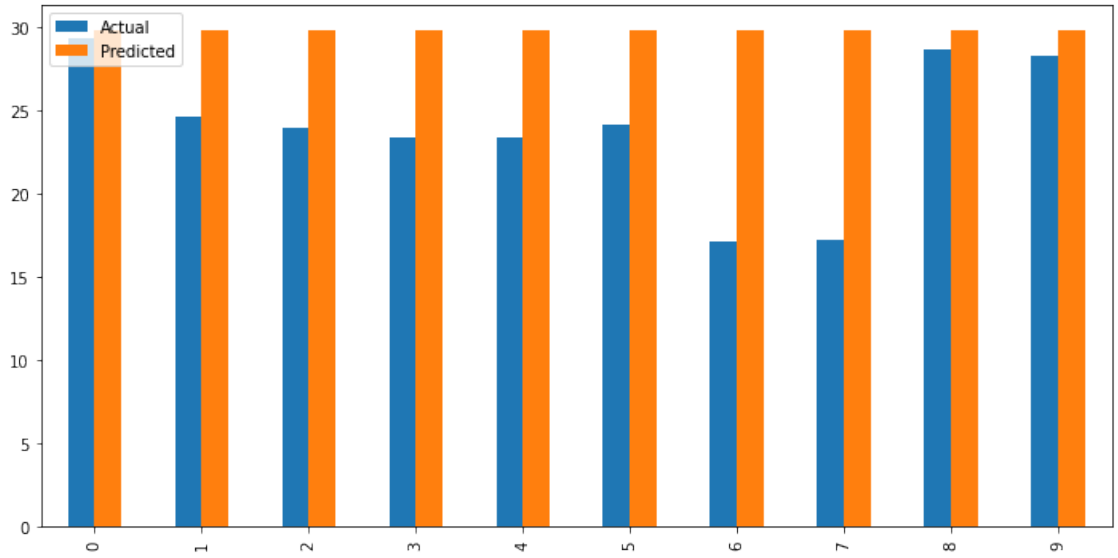
[17]:
```
#Plot the regression line for complete data:

plt.scatter(data["Horse Power"],data["Fuel Economy (MPG)"])
plt.plot(data["Horse Power"],predicted_data,color="red")
plt.xlabel("Horse Power")
plt.ylabel("Fuel Economy (MPG)")
plt.show()
```



[18]:
```
#Create a dataframe for Actual and Predicted values:

A_P_data = pd.DataFrame({"Actual":data["Fuel Economy (MPG)"],"Predicted":
 ↪predicted_data[:][0][0]})
print(A_P_data.head())
```

```
      Actual  Predicted
0  29.344195  29.863742
1  24.695934  29.863742
2  23.952010  29.863742
3  23.384546  29.863742
4  23.426739  29.863742
```

[19]:
```
#Plot the bar graph for actual and predicted values:

A_P_data.head(10).plot(kind='bar',figsize=(12,6))
plt.show()
```



[20]:
```
#Error calculations:

test_x = np.array(test[['Horse Power']])
test_y = np.array(test[['Fuel Economy (MPG)']])

predicted_y= regr.predict(test_x)

res = (predicted_y - test_y)
RSS = (res*res).sum()

print("Residual Sum of Squares: ",RSS)
```

```
Residual Sum of Squares:  62.91040679115408
```