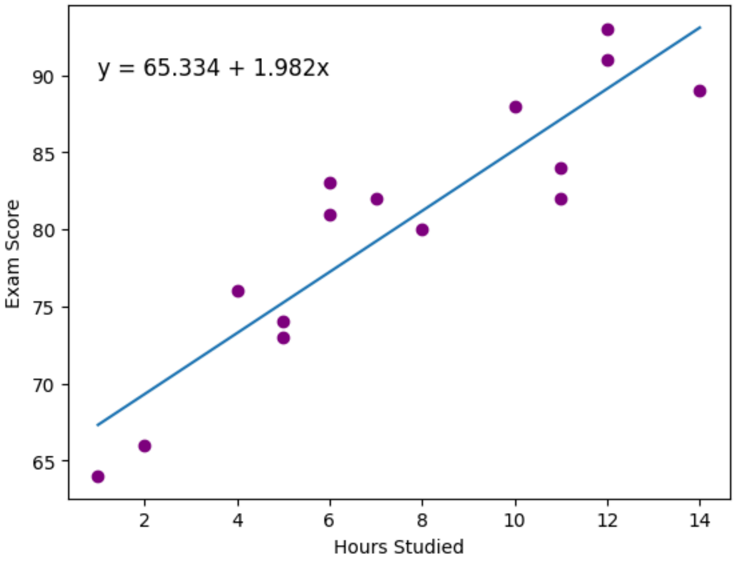
```
import pandas as pd

#create DataFrame
df = pd.DataFrame({'hours': [1, 2, 4, 5, 5, 6, 6, 7, 8, 10, 11, 11, 12, 12, 14],
                   'score': [64, 66, 76, 73, 74, 81, 83, 82, 80, 88, 84, 82, 91, 93, 89]})

#view DataFrame
print(df)
```

```
    hours  score
0       1     64
1       2     66
2       4     76
3       5     73
4       5     74
5       6     81
6       6     83
7       7     82
8       8     80
9      10     88
10     11     84
11     11     82
12     12     91
13     12     93
14     14     89
```

```
import statsmodels.api as sm

#define predictor and response variables
y = df['score']
x = df['hours']

#add constant to predictor variables
x = sm.add_constant(x)

#fit linear regression model
model = sm.OLS(y, x).fit()

#view model summary
print(model.summary())
```

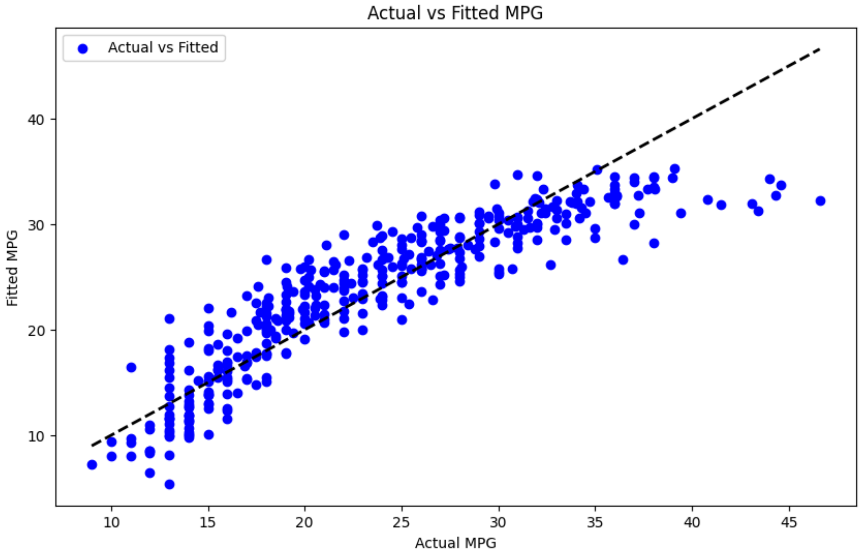
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  score   R-squared:                       0.831
Model:                            OLS   Adj. R-squared:                  0.818
Method:                 Least Squares   F-statistic:                     63.91
Date:                Mon, 22 Apr 2024   Prob (F-statistic):           2.25e-06
Time:                        08:52:01   Log-Likelihood:                -39.594
No. Observations:                  15   AIC:                             83.19
Df Residuals:                      13   BIC:                             84.60
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         65.3340      2.106     31.023      0.000      60.784      69.884
hours          1.9824      0.248      7.995      0.000       1.447       2.518
==============================================================================
Omnibus:                        4.351   Durbin-Watson:                   1.677
Prob(Omnibus):                  0.114   Jarque-Bera (JB):                1.329
Skew:                           0.092   Prob(JB):                        0.515
Kurtosis:                       1.554   Cond. No.                         19.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
/usr/local/lib/python3.10/dist-packages/scipy/stats/_stats_py.py:1806: UserWarning: kurtosistest only valid for n>=20 ... continuing
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
import matplotlib.pyplot as plt
import numpy as np

#find line of best fit
a, b = np.polyfit(df['hours'], df['score'], 1)

#add points to plot
plt.scatter(df['hours'], df['score'], color='purple')

#add line of best fit to plot
plt.plot(df['hours'], a*df['hours']+b)

#add fitted regression equation to plot
plt.text(1, 90, 'y = ' + '{:.3f}'.format(b) + ' + {:.3f}'.format(a) + 'x', size=12)

#add axis labels
plt.xlabel('Hours Studied')
plt.ylabel('Exam Score')
```

```
Text(0, 0.5, 'Exam Score')
```



```
import seaborn as sns
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt

# Load the dataset
mpg_df = sns.load_dataset('mpg')

# Check for missing values
print(mpg_df.isnull().sum())

# Drop rows with missing values
mpg_df = mpg_df.dropna()

# Convert data types to ensure compatibility
mpg_df['horsepower'] = pd.to_numeric(mpg_df['horsepower'], errors='coerce')

# Perform regression analysis
# Define independent variables (features)
X = mpg_df[['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model_year']]
# Add constant for intercept
X = sm.add_constant(X)

# Define dependent variable (target)
y = mpg_df['mpg']

# Fit the regression model
model = sm.OLS(y, X).fit()

# Print regression results
print(model.summary())

# Plot the fitting line
fig, ax = plt.subplots(figsize=(10, 6))

# Scatter plot of actual data points
ax.scatter(y, model.fittedvalues, label='Actual vs Fitted', color='blue')

# Plot the diagonal line
# Plot the diagonal line
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)

ax.set_xlabel('Actual MPG')
ax.set_ylabel('Fitted MPG')
ax.set_title('Actual vs Fitted MPG')
ax.legend()

plt.show()
```

```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model_year      0
origin          0
name            0
dtype: int64
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.809
Model:                            OLS   Adj. R-squared:                  0.806
Method:                 Least Squares   F-statistic:                     272.2
Date:                Mon, 22 Apr 2024   Prob (F-statistic):          3.79e-135
Time:                        09:45:52   Log-Likelihood:                -1036.5
No. Observations:                 392   AIC:                             2087.
Df Residuals:                     385   BIC:                             2115.
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const          -14.5353      4.764     -3.051      0.002     -23.902      -5.169
cylinders       -0.3299      0.332     -0.993      0.321      -0.983       0.323
displacement     0.0077      0.007      1.044      0.297      -0.007       0.022
horsepower      -0.0004      0.014     -0.028      0.977      -0.028       0.027
weight          -0.0068      0.001    -10.141      0.000      -0.008      -0.005
acceleration     0.0853      0.102      0.836      0.404      -0.115       0.286
model_year       0.7534      0.053     14.318      0.000       0.650       0.857
==============================================================================
Omnibus:                       37.865   Durbin-Watson:                   1.232
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               60.248
Skew:                           0.630   Prob(JB):                     8.26e-14
Kurtosis:                       4.449   Cond. No.                     8.53e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly spec
[2] The condition number is large, 8.53e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```


Actual vs Fitted MPG
● Actual vs Fitted
Fitted MPG
Actual MPG