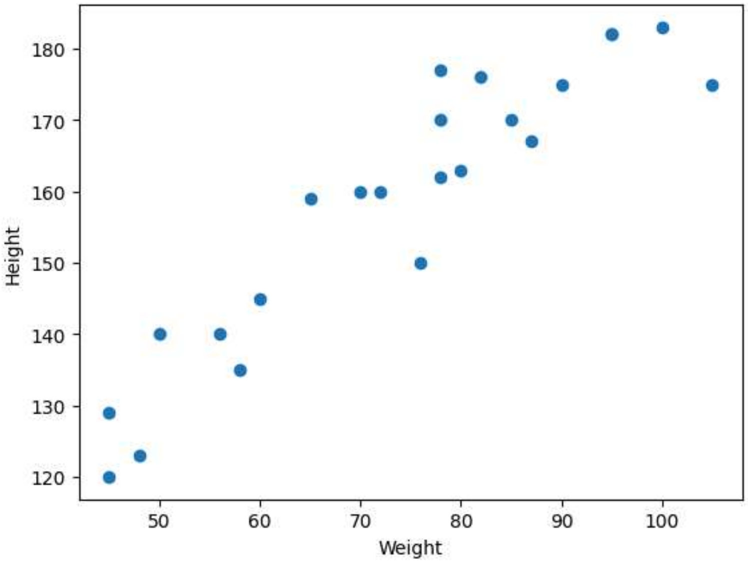
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
```

```
df = pd.read_csv('/content/height-weight.csv')
```

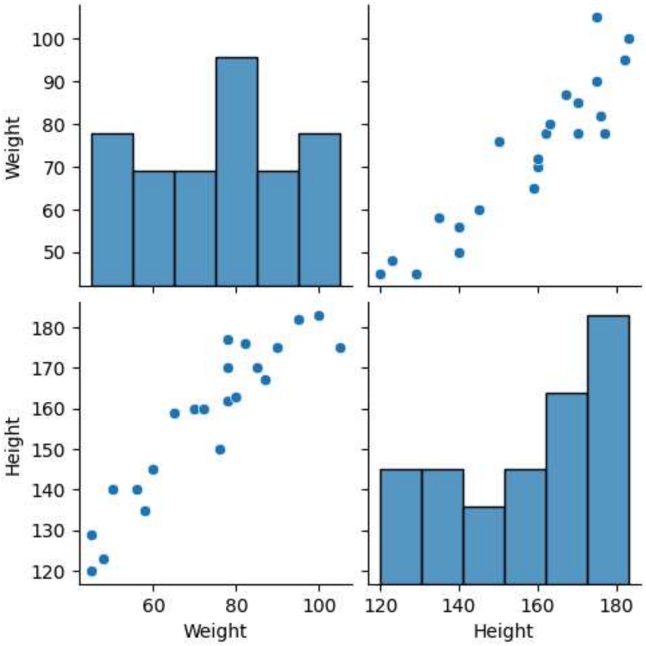
```
#scatter plot
plt.scatter(df['Weight'], df['Height'])
plt.xlabel("Weight")
plt.ylabel("Height")
```

```
Text(0, 0.5, 'Height')
```



```
#seaborn for visualization
import seaborn as sns
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7f08da5d5cc0>
```



```
X = df[['Weight']]
y = df[['Height']]
```

```
X_series=df['Weight']
np.array(X_series).shape
```

```
(23,)
```

```
np.array(y).shape
```

```
(23, 1)
```

```
## Train Test Split
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.25, random_state=42)
```

```
#standardization
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)
```

```
X_test
```

```
array([[ 0.33497168],
       [ 0.33497168],
       [-1.6641678 ],
       [ 1.36483141],
       [-0.45256812],
       [ 1.97063125]])
```

```
from sklearn.linear_model import LinearRegression
```

```
regression=LinearRegression(n_jobs=-1)
```

```
regression.fit(X_train, y_train)
```

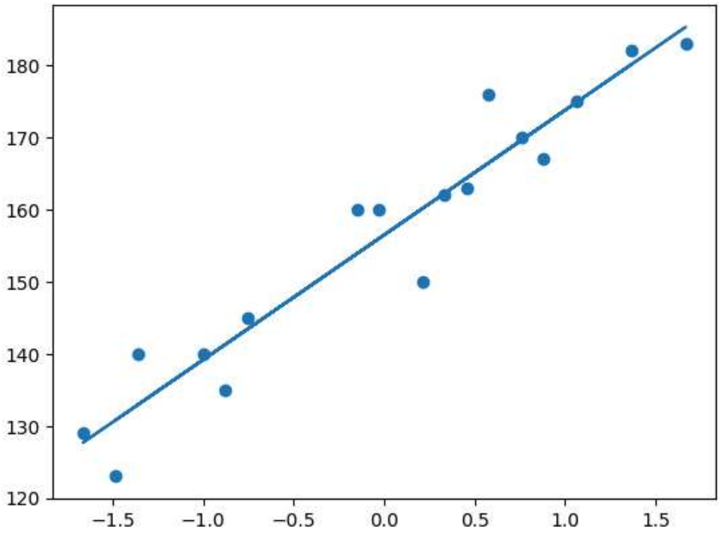
```
▾     LinearRegression
LinearRegression(n_jobs=-1)
```

```
print("Coefficient or slope:", regression.coef_)
print("Intercept:", regression.intercept_)
```

```
Coefficient or slope: [[17.2982057]]
Intercept: [156.47058824]
```

```
plt.scatter(X_train, y_train)
plt.plot(X_train, regression.predict(X_train))
```

```
[<matplotlib.lines.Line2D at 0x7f08cf764bb0>]
```



```
Start coding or generate with AI.
```

prediction of test data predicted height output = interept + coef(Weights) y_pred_test = 156.470 + 17.29(X_test)

```
#prediction of test data
y_pred = regression.predict(X_test)
```

```
#performance Metrics
from sklearn.metrics import mean_absolute_error, mean_squared_error
```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
print(mae)
print(mse)
print(rmse)
```

```
9.66512588679501
114.84069295228699
10.716374991212605
```

R square formula R^2 = 1 SSR/SST R^ 2 = coefficient of determination SSR = sum of squares of residuals SST = total sum of sqares

```
from sklearn.metrics import r2_score
```

```
score = r2_score(y_test, y_pred)
print(score)
```

```
0.7360826717981276
```

```
##Ols Linear Regression
import statsmodels.api as sm
```

```
model = sm.OLS(y_train, X_train).fit()
```

```
prediction = model.predict(X_test)
print(prediction)
```

```
[  5.79440897   5.79440897 -28.78711691  23.60913442  -7.82861638
  34.08838469]
```

```
print(model.summary())
```

```
                                 OLS Regression Results
==============================================================================
Dep. Variable:                 Height   R-squared (uncentered):                   0.012
Model:                            OLS   Adj. R-squared (uncentered):             -0.050
Method:                 Least Squares   F-statistic:                             0.1953
Date:                Mon, 22 Jul 2024   Prob (F-statistic):                       0.664
Time:                        07:48:56   Log-Likelihood:                         -110.03
No. Observations:                  17   AIC:                                      222.1
Df Residuals:                      16   BIC:                                      222.9
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
x1            17.2982     39.138      0.442      0.664     -65.671     100.267
==============================================================================
Omnibus:                        0.135   Durbin-Watson:                   0.002
Prob(Omnibus):                  0.935   Jarque-Bera (JB):                0.203
Skew:                          -0.166   Prob(JB):                        0.904
Kurtosis:                       2.581   Cond. No.                         1.00
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] 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 an
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

```
#prediction for new data
regression.predict(scaler.transform([[72]]))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but StandardScaler was f
  warnings.warn(
array([[155.97744705]])
```

Start coding or generate with AI.