Height vs Weight

August 25, 2022

[1]:
```
import pandas as pd
```

[2]:
```
import numpy as np
```

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

[4]:
```
print(data)
```

```
       Index  Height(Inches)  Weight(Pounds)
0          1        65.78331        112.9925
1          2        71.51521        136.4873
2          3        69.39874        153.0269
3          4        68.21660        142.3354
4          5        67.78781        144.2971
…        …              …               …
24995  24996        69.50215        118.0312
24996  24997        64.54826        120.1932
24997  24998        64.69855        118.2655
24998  24999        67.52918        132.2682
24999  25000        68.87761        124.8742

[25000 rows x 3 columns]
```

[5]:
```
data.head(10)
```

[5]:

	Index	Height(Inches)	Weight(Pounds)
0	1	65.78331	112.9925
1	2	71.51521	136.4873
2	3	69.39874	153.0269
3	4	68.21660	142.3354
4	5	67.78781	144.2971
5	6	68.69784	123.3024
6	7	69.80204	141.4947
7	8	70.01472	136.4623
8	9	67.90265	112.3723
9	10	66.78236	120.6672

[6]:
```
data.tail(10)
```

[6]:

	Index	Height(Inches)	Weight(Pounds)
24990	24991	69.97767	125.3672
24991	24992	71.91656	128.2840
24992	24993	70.96218	146.1936
24993	24994	66.19462	118.7974
24994	24995	67.21126	127.6603
24995	24996	69.50215	118.0312
24996	24997	64.54826	120.1932
24997	24998	64.69855	118.2655
24998	24999	67.52918	132.2682
24999	25000	68.87761	124.8742

[7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Index            25000 non-null  int64
 1   Height(Inches)   25000 non-null  float64
 2   Weight(Pounds)   25000 non-null  float64
dtypes: float64(2), int64(1)
memory usage: 586.1 KB
```

[8]:
```
data.rename(columns={'Height(Inches)':'height','Weight(Pounds)':'weight'},␣
↪inplace = 1)
```

[9]:
```
data.head(2)
```

[9]:

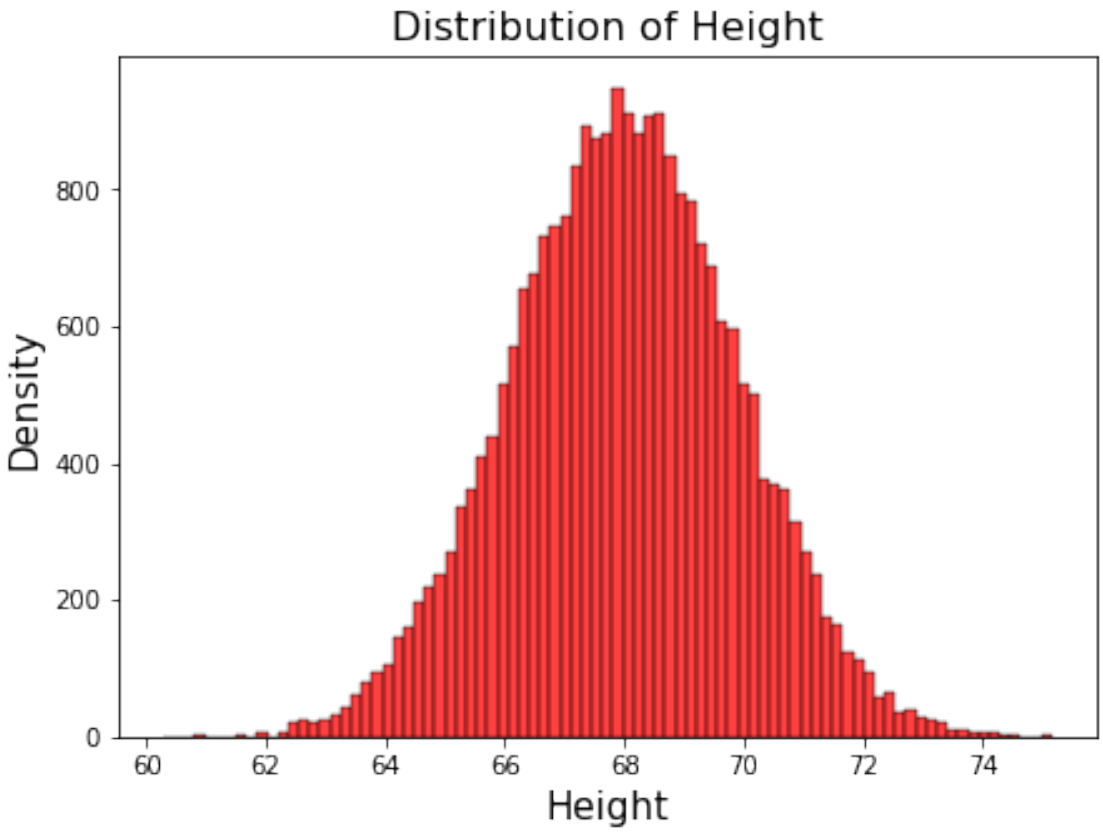
	Index	height	weight
0	1	65.78331	112.9925
1	2	71.51521	136.4873

[10]:
```
data.shape
```

[10]:
```
(25000, 3)
```

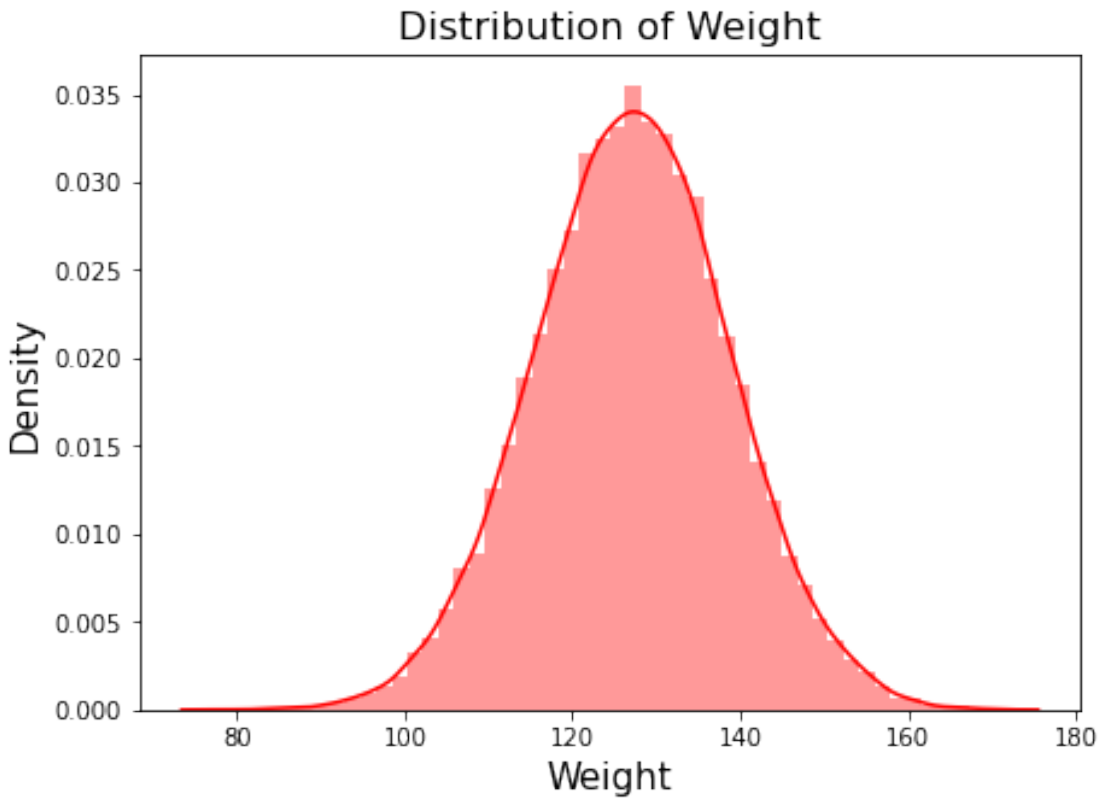
[11]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

[12]:
```
plt.figure(figsize=(7,5))
sns.histplot(data['height'],color='red')
plt.title('Distribution of Height', fontsize=16)
plt.xlabel('Height', fontsize=15)
plt.ylabel('Density', fontsize=15)
plt.show()
```



[13]:
```
plt.figure(figsize=(7,5))
sns.distplot(data['weight'],color='red')
plt.title('Distribution of Weight', fontsize=16)
plt.xlabel('Weight', fontsize=15)
plt.ylabel('Density', fontsize=15)
plt.show()
```

```
C:\Users\Sushan Shivagiri\AppData\Local\Programs\Python\Python310\lib\site-
packages\seaborn\distributions.py:2619: FutureWarning: `distplot` is a
deprecated function and will be removed in a future version. Please adapt your
code to use either `displot` (a figure-level function with similar flexibility)
or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



[14]:
```
data.isnull().any()
```

[14]:
```
Index     False
height    False
weight    False
dtype: bool
```

[15]:
```
data = data.drop(['Index'], axis = 'columns')
```

[16]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   height  25000 non-null  float64
 1   weight  25000 non-null  float64
dtypes: float64(2)
memory usage: 390.8 KB
```

[17]:
```
from sklearn.model_selection import train_test_split
from sklearn import linear_model
```

[18]:
```
x = data.drop(columns=['weight'])
y = data['weight']
```

[19]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25,␣
↪random_state =1)
```

[20]:
```
regressor = linear_model.LinearRegression()
regressor.fit(x,y)
```

[20]:
```
LinearRegression()
```

[21]:
```
regressor.score(x,y) * 100
```

[21]:
```
25.28666917428807
```

[22]:
```
from sklearn.ensemble import RandomForestRegressor
```

[23]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.30,␣
↪random_state=40)
```

[24]:
```
regre = RandomForestRegressor(max_depth=10, random_state=0)
```

[25]:
```
regre.fit(x_train, y_train)
```

[25]:
```
RandomForestRegressor(max_depth=10, random_state=0)
```

[26]:
```
regre.score(x_test, y_test)*100
```

[26]:
```
23.888324494614434
```