```
import math
A = 16
B = 3.14
print(math.sqrt(A))
print(math.sin(B))
print(math.cos(A))
```

```
4.0
0.0015926529164868282
-0.9576594803233847
```

```
distance=[10,15,17,26]
time=[.20,.47,.55,1.20]
import numpy as np
np_distance = np.array(distance)
np_time = np.array(time)
speed=np_distance/np_time
speed
```

```
array([50.        , 31.91489362, 30.90909091, 21.66666667])
```

```
import numpy as np
a = np.array([1,2,3])
b = np.array([1,15,20])
print(a.shape)
print(b.shape)
```

```
(3,)
(3,)
```

```
import numpy as np
a = np.array([1,2,3,4,5,6,7,8,9,10,11,12])
b = np.array([23,4,5,6,7,77,7,44566,6,4535,6,5,63535,467788,75656])
newa = a.reshape(4,3)
newb = b.reshape(5,3)
print(newa)
print(newb)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[[    23      4      5]
 [     6      7     77]
 [     7  44566      6]
 [  4535      6      5]
 [ 63535 467788  75656]]
```

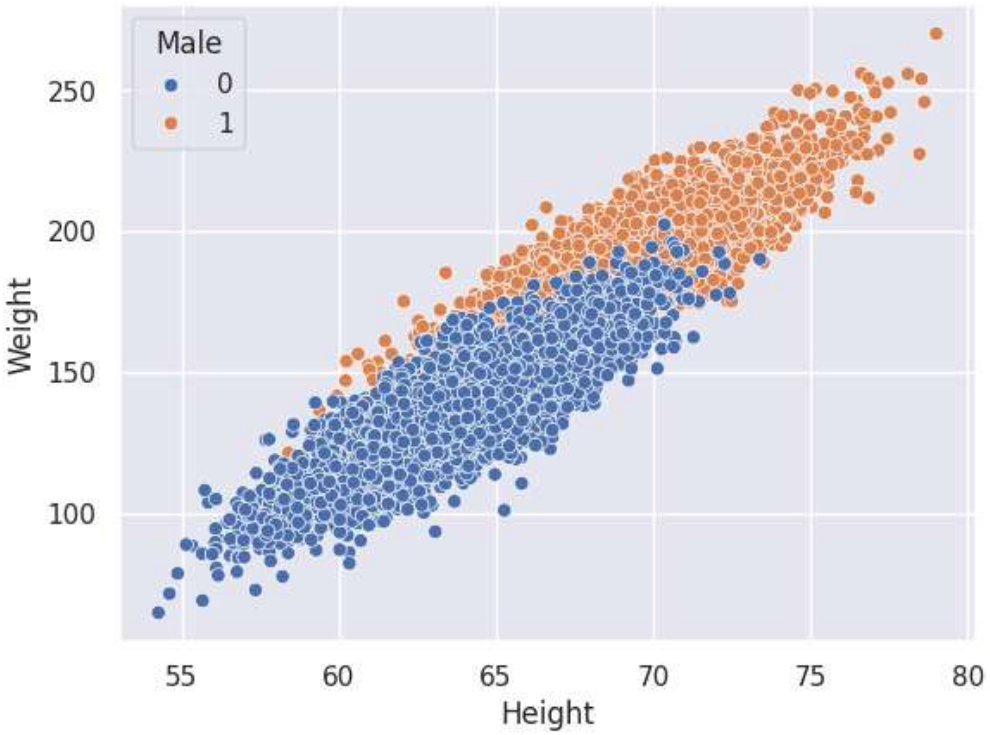
```
import numpy as np
import pandas as pd
## load data
file_name = "/content/heights_weights.csv"
df = pd.read_csv(file_name)
# visualize data
df.head()
```

	Height	Weight	Male
0	73.847017	241.893563	1
1	68.781904	162.310473	1
2	74.110105	212.740856	1
3	71.730978	220.042470	1
4	69.881796	206.349801	1

Next steps: Generate code with df View recommended plots

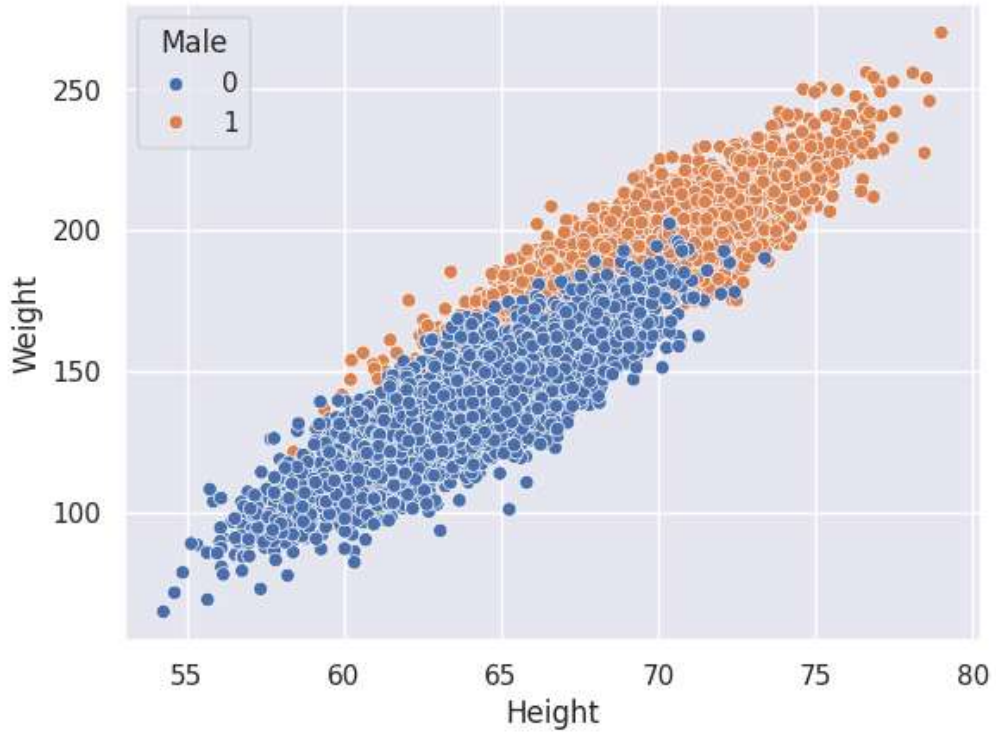
```
# plot the data
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt

ax = sns.scatterplot(x="Height",y="Weight",hue="Male",data=df)
```



```
# plot the data
import seaborn as sns; sns.set()
import matplotlib.pyplot as plt

ax = sns.scatterplot(x="Height",y="Weight",hue="Male",data = df)
```




```
x = df.iloc[:,0:2].values
y = df.iloc[:,2].values

from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size=0.3,random_state=0)


from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score


logreg_clf = LogisticRegression()
logreg_model = logreg_clf.fit(X_train,Y_train)
logreg_prediction = logreg_clf.predict(X_test)


print("Accuracy {0:.2f}%".format(100*accuracy_score(logreg_prediction,Y_test)))
print(confusion_matrix(logreg_prediction,Y_test))
print(classification_report(logreg_prediction,Y_test))
```

```
Accuracy 91.87%
[[1385  140]
 [ 104 1371]]
```

```
                    precision    recall  f1-score   support

                 0       0.93      0.91      0.92      1525
                 1       0.91      0.93      0.92      1475

          accuracy                           0.92      3000
         macro avg       0.92      0.92      0.92      3000
      weighted avg       0.92      0.92      0.92      3000
```

```
logreg_clf = RandomForestClassifier()
logreg_model = logreg_clf.fit(X_train,Y_train)
logreg_prediction = logreg_clf.predict(X_test)
```

```
print("Accuracy {0:.2f}%".format(100*accuracy_score(logreg_prediction,Y_test)))
print(confusion_matrix(logreg_prediction,Y_test))
print(classification_report(logreg_prediction,Y_test))
```

```
     Accuracy 90.70%
     [[1377  167]
      [ 112 1344]]
                   precision    recall  f1-score   support

                0       0.92      0.89      0.91      1544
                1       0.89      0.92      0.91      1456

         accuracy                           0.91      3000
        macro avg       0.91      0.91      0.91      3000
     weighted avg       0.91      0.91      0.91      3000
```

```
logreg_clf = SVC()
logreg_model = logreg_clf.fit(X_train,Y_train)
logreg_prediction = logreg_clf.predict(X_test)
```

```
print("Accuracy {0:.2f}%".format(100*accuracy_score(logreg_prediction,Y_test)))
print(confusion_matrix(logreg_prediction,Y_test))
print(classification_report(logreg_prediction,Y_test))
```

```
     Accuracy 91.40%
     [[1382  151]
      [ 107 1360]]
                   precision    recall  f1-score   support

                0       0.93      0.90      0.91      1533
                1       0.90      0.93      0.91      1467

         accuracy                           0.91      3000
        macro avg       0.91      0.91      0.91      3000
     weighted avg       0.91      0.91      0.91      3000
```

```
logreg_clf = KNeighborsClassifier()
logreg_model = logreg_clf.fit(X_train,Y_train)
logreg_prediction = logreg_clf.predict(X_test)
```

```
print("Accuracy {0:.2f}%".format(100*accuracy_score(logreg_prediction,Y_test)))
print(confusion_matrix(logreg_prediction,Y_test))
```

```
print(classification_report(logreg_prediction,Y_test))
```

```
Accuracy 90.27%
[[1367  170]
 [ 122 1341]]
              precision    recall  f1-score   support

           0       0.92      0.89      0.90      1537
           1       0.89      0.92      0.90      1463

    accuracy                           0.90      3000
   macro avg       0.90      0.90      0.90      3000
weighted avg       0.90      0.90      0.90      3000
```

```
print(X_train.shape)
print(Y_train.shape)
print(X_test.shape)
print(Y_test.shape)
```

```
(7000, 2)
(3000,)
(3000, 2)
(7000,)
```