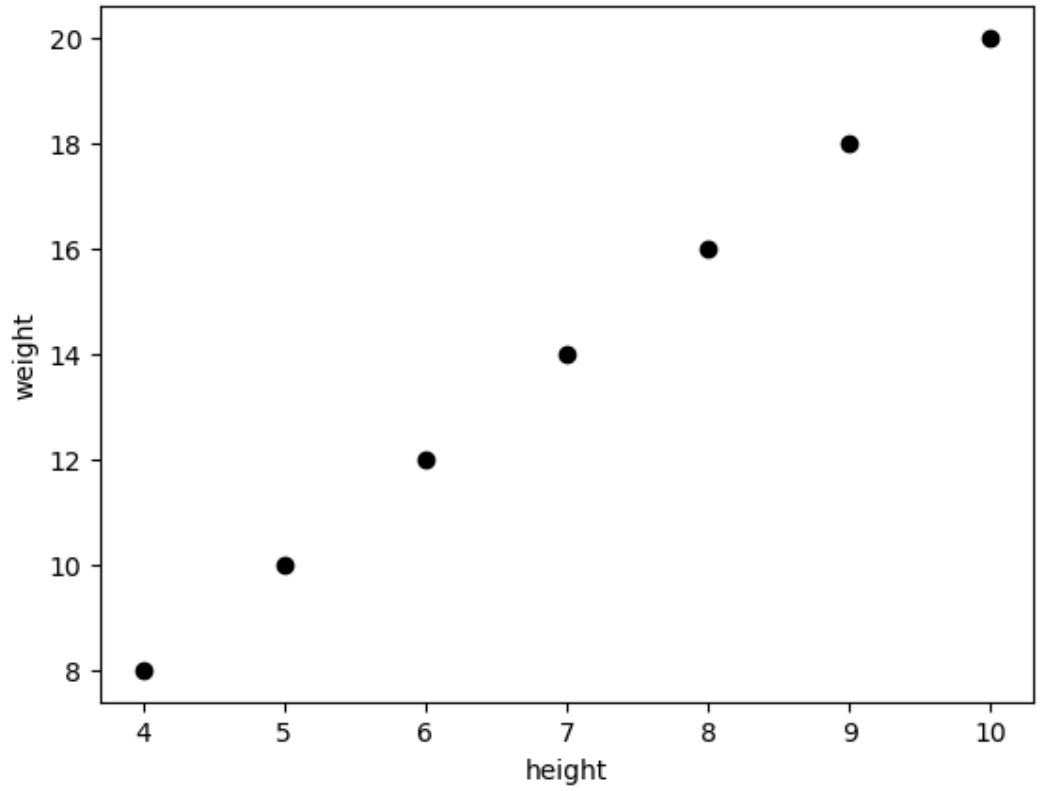
In [2]:
```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

In [5]:
```
import sklearn
from sklearn import linear_model
```

In [5]:
```
height=[[4.0],[5.0],[6.0],[7.0],[8.0],[9.0],[10.0]]
weight=[  8, 10 , 12, 14, 16, 18, 20]
```

In [6]:
```
plt.scatter(height,weight,color='black')
plt.xlabel("height")
plt.ylabel("weight")
reg=linear_model.LinearRegression()
reg.fit(height,weight)
X_height=[[12.0]]
print(reg.predict(X_height))
```

```
[24.]
```



In [8]:
```
x=[[4.0],[5.0],[6.0],[7.0],[8.0],[9.0],[10.0]]
y=[  16, 25 , 36, 49,64,81, 100]
```

In [9]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x,y)
```

Out[9]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [10]:
```
print(lin_reg.predict([[11]]))
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures

polynomial_regression = make_pipeline(
    PolynomialFeatures(degree=2, include_bias=False),
    LinearRegression(),
)
polynomial_regression.fit(x,y)
X_height=[[20.0]]
target_predicted = polynomial_regression.predict(X_height)
print(target_predicted)
```

```
[109.]
[400.]
```

In [12]:
```
x=[[4.0],[5.0],[6.0],[7.0],[8.0],[9.0],[10.0]]
y=[  64, 125 , 216, 343, 512,729, 1000]
```

In [13]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x,y)
```

Out[13]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [14]:
```
print(lin_reg.predict([[11]]))
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures

polynomial_regression = make_pipeline(
    PolynomialFeatures(degree=2, include_bias=False),
    LinearRegression(),
)
polynomial_regression.fit(x,y)
X_height=[[20.0]]
target_predicted = polynomial_regression.predict(X_height)
print(target_predicted)
```

```
[1043.]
[5894.]
```

In [23]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
X = [[30],[40],[50],[60],[20],[10],[70]]
y = [0,1,1,1,0,0,1]
classifier = DecisionTreeClassifier(criterion = 'entropy', random_stat
classifier.fit(X,y)
X_marks=[[39]]
print(classifier.predict(X_marks))
```

```
[1]
```

In [17]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.svm import SVC
X = [[30],[40],[50],[60],[20],[10],[70]]
y = [0,1,1,1,0,0,1]
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X,y)
X_marks=[[55]]
print(classifier.predict(X_marks))
```

```
[1]
```

In [18]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
X = [[30],[40],[50],[60],[20],[10],[70]]
y = [0,1,1,1,0,0,1]
from sklearn.neighbors import KNeighborsClassifier
classifier= KNeighborsClassifier(n_neighbors=5, metric='minkowski', p=
classifier.fit(X,y)
X_marks=[[50]]
print(classifier.predict(X_marks))
```

```
[1]
```

In [21]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
RandomForestRegModel = RandomForestRegressor()
RandomForestRegModel.fit(X,y)
X_marks=[[70]]
print(RandomForestRegModel.predict(X_marks))
```

```
[0.99]
```

In [25]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("NBD.csv")
df.head()
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.35,ra
x=df.drop('diabetes',axis=1)
y=df['diabetes']
model=GaussianNB()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
y_pred
```

Out[25]:
```
array([1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0,
0, 1,
       1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0,
0, 0,
       0, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
0, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
0, 1,
       0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0,
1, 0,
       1, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1,
1, 0,
       0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
0, 0,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0,
0, 1,
       0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1,
0, 0,
       0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
0, 0,
       1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0,
1, 0,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1,
0, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1,
1, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0,
1, 0,
       0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0,
0, 0,
       1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0],
      dtype=int64)
```

In [28]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv("heart.csv")
df.head()
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.25,ra
x=df.drop('ca',axis=1)
y=df['ca']
model=GaussianNB()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
print(y_pred)
```

```
[209 226 289 295 256 269 299 206 255 174 223 326 235 126 184 268 335
157
 199 229 308 207 284 311 308 175 325 175 407 330 218 308 229 226 235
221
 308 302 303 235 206 200 192 284 225 217 188 223 223 211 175 259 232
282
 325 303 269 206 335 304 250 220 407 211 299 169 325 256 217 252 172
407
 223 192 268 264 299 273 335 192 229 207 327 335 245 330 223 212 229
330
 258 229 199 209 199 208 256 315 288 218 258 281 282 298 299 226 206
227
 229 308 335 308 250 268 199 211 203 235 195 299 394 208 286 206 233
311
 192 274 172 342 308 288 325 258 353 183 176 208 183 203 303 244 306
263
 265 240 175 209 244 286 229 217 192 303 307 249 236 342 237 200 192
178
 281 281 208 303 207 294 270 209 223 222 226 164 242 226 245 298 282
214
 235 247 235 232 223 195 235 335 232 250 277 335 260 232 206 193 223
318
 235 235 277 192 235 207 229 149 325 250 192 208 192 225 182 258 249
205
 249 207 227 353 201 327 218 250 192 192 235 250 326 394 260 299 270
236
 286 299 207 305 214 185 201 319 186 258 282 227 276 258 234 230 192
174
 241 298 206 180 341]
```

In [ ]: