Pregunta 8

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [2]:
```
# Evaluacion
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

# Modelo
from sklearn.linear_model import LinearRegression

# Visualizacion
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
```

In [3]:
```
df = pd.read_csv('fat.csv')
df
```

Out[3]:

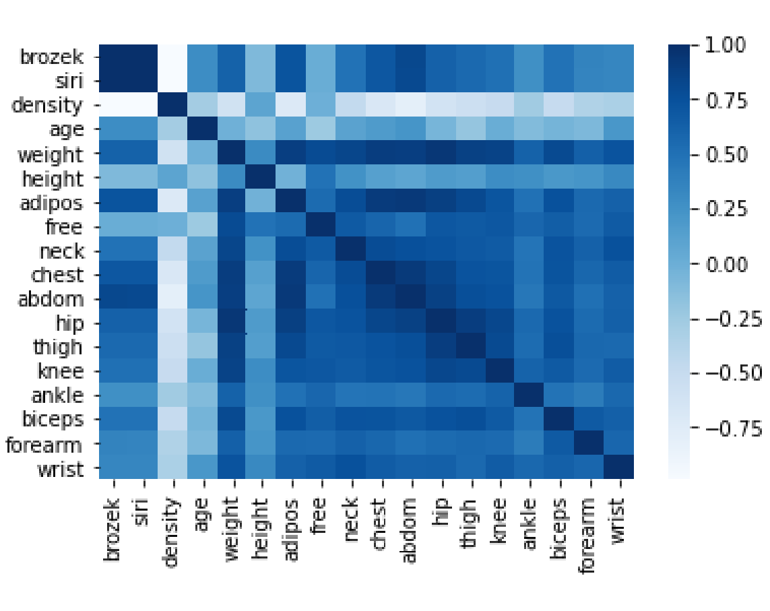
	brozek	siri	density	age	weight	height	adipos	free	neck	chest	abdom	hip	thigh	knee	ankle	biceps	forearm	wrist
0	12.6	12.3	1.0708	23	154.25	67.75	23.7	134.9	36.2	93.1	85.2	94.5	59.0	37.3	21.9	32.0	27.4	17.1
1	6.9	6.1	1.0853	22	173.25	72.25	23.4	161.3	38.5	93.6	83.0	98.7	58.7	37.3	23.4	30.5	28.9	18.2
2	24.6	25.3	1.0414	22	154.00	66.25	24.7	116.0	34.0	95.8	87.9	99.2	59.6	38.9	24.0	28.8	25.2	16.6
3	10.9	10.4	1.0751	26	184.75	72.25	24.9	164.7	37.4	101.8	86.4	101.2	60.1	37.3	22.8	32.4	29.4	18.2
4	27.8	28.7	1.0340	24	184.25	71.25	25.6	133.1	34.4	97.3	100.0	101.9	63.2	42.2	24.0	32.2	27.7	17.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
247	11.5	11.0	1.0736	70	134.25	67.00	21.1	118.9	34.9	89.2	83.6	88.8	49.6	34.8	21.5	25.6	25.7	18.5
248	32.3	33.6	1.0236	72	201.00	69.75	29.1	136.1	40.9	108.5	105.0	104.5	59.6	40.8	23.2	35.2	28.6	20.1
249	28.3	29.3	1.0328	72	186.75	66.00	30.2	133.9	38.9	111.1	111.5	101.7	60.3	37.3	21.5	31.3	27.2	18.0
250	25.3	26.0	1.0399	72	190.75	70.50	27.0	142.6	38.9	108.3	101.3	97.8	56.0	41.6	22.7	30.5	29.4	19.8
251	30.7	31.9	1.0271	74	207.50	70.00	29.8	143.7	40.8	112.4	108.5	107.1	59.3	42.2	24.6	33.7	30.0	20.9

252 rows × 18 columns

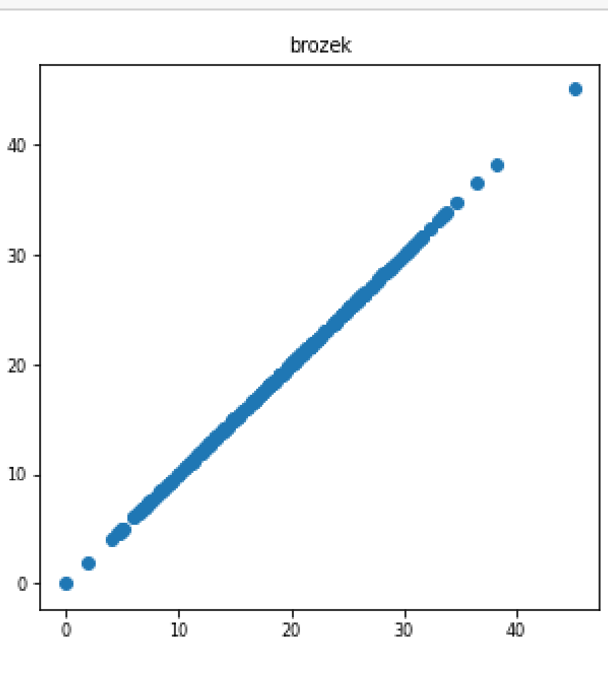
A) Determina la matriz de correlación y muestra las visualizaciones entre la variable target y sus demás variables.

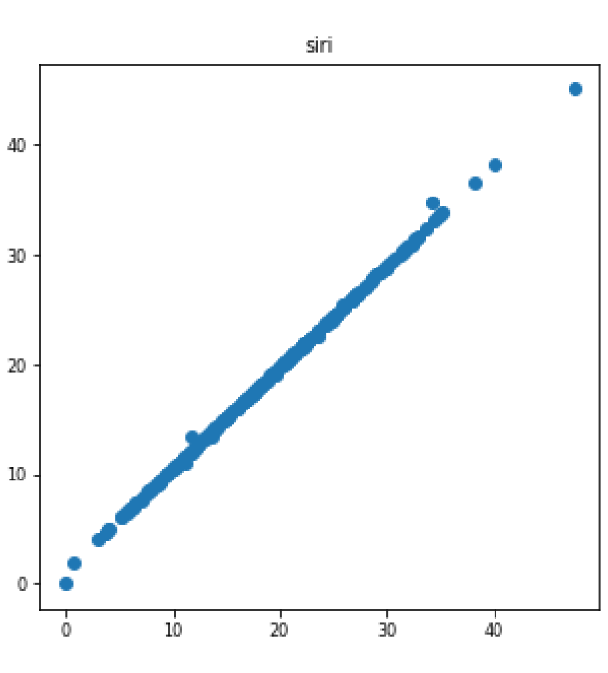
In [4]:
```
# Matriz de correlación
corr = df.corr()
sns.heatmap(corr, cmap="Blues", annot=False)
```

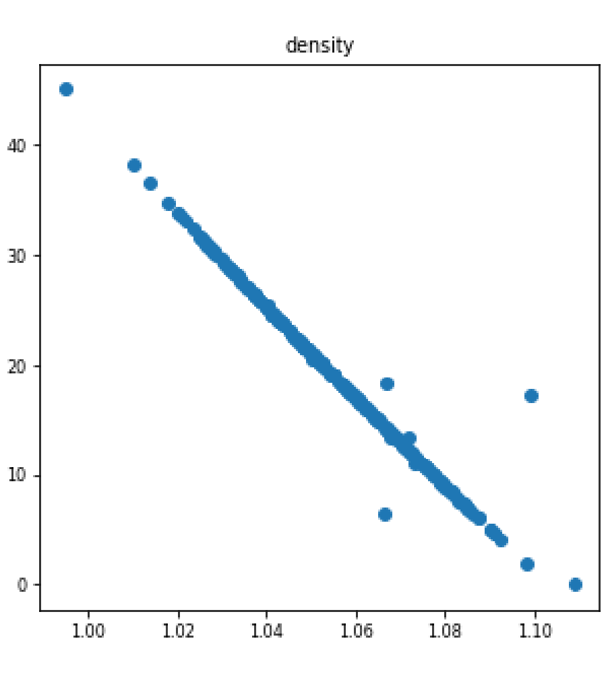
Out[4]:
```
<AxesSubplot:>
```

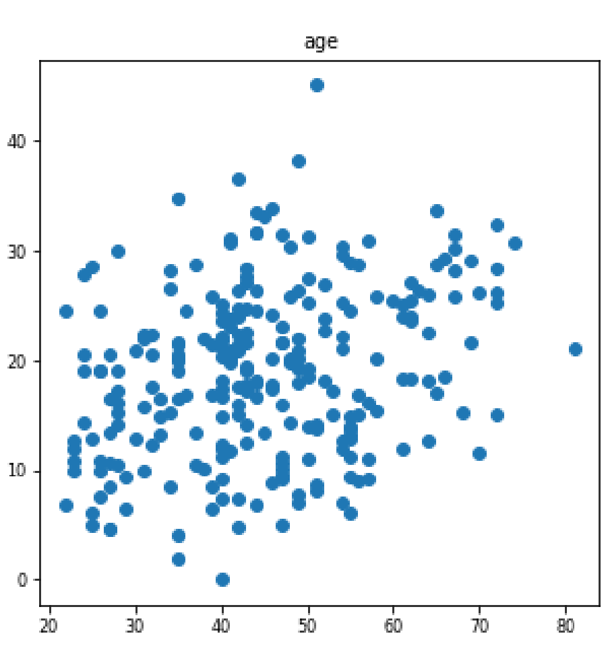


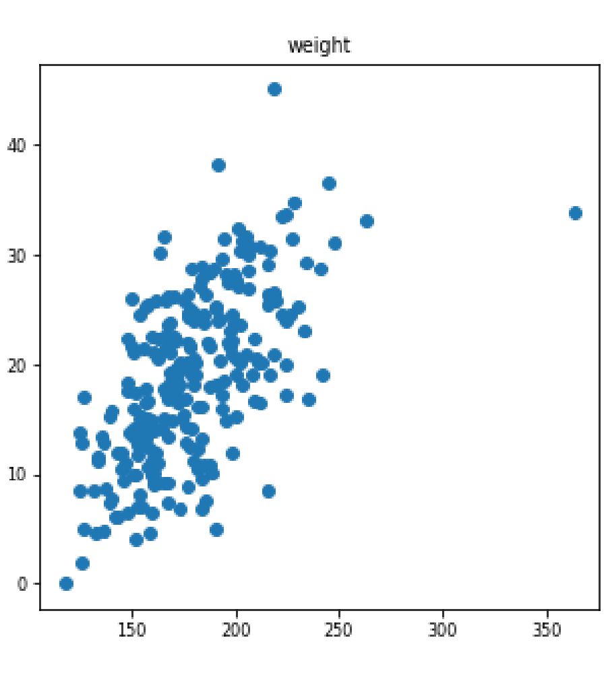
In [5]:
```
# Mostrar las visualizaciones entre la variable target y sus demás variables.
# Definir parametros del grafico

plt.rcParams['figure.figsize'] = [5.0, 5.0]
plt.rcParams['font.size'] = 8

    # Obtener cada columna y hacer comparación con la variable predictora
for column in df:
    plt.figure()
    plt.title(column)
    plt.scatter(df[column],df["brozek"])
```

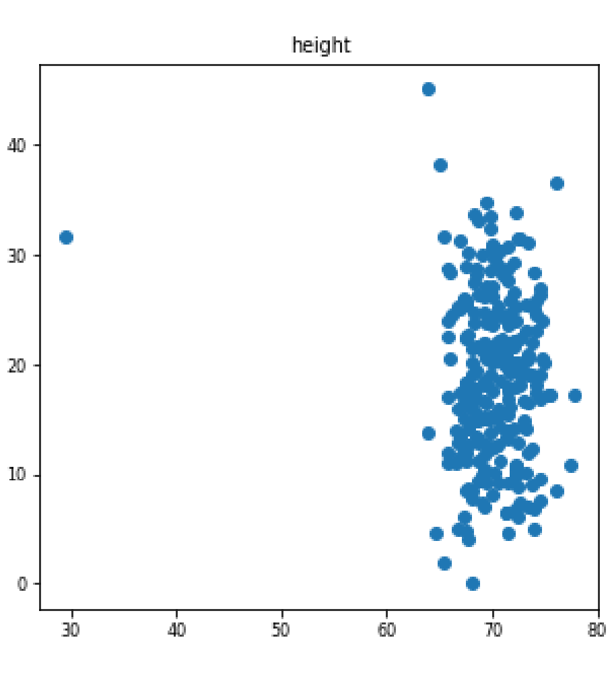


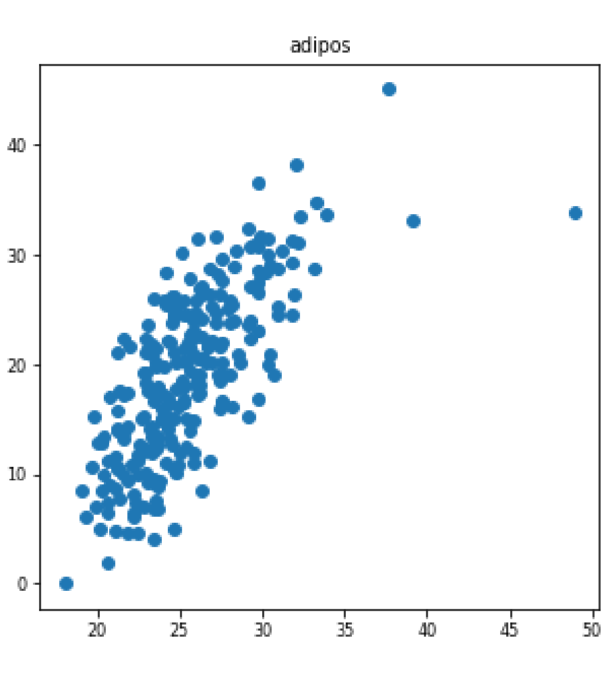


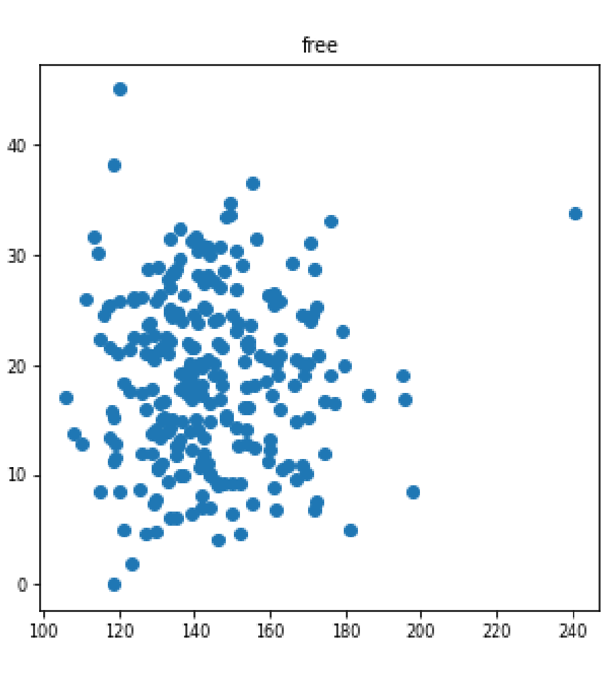


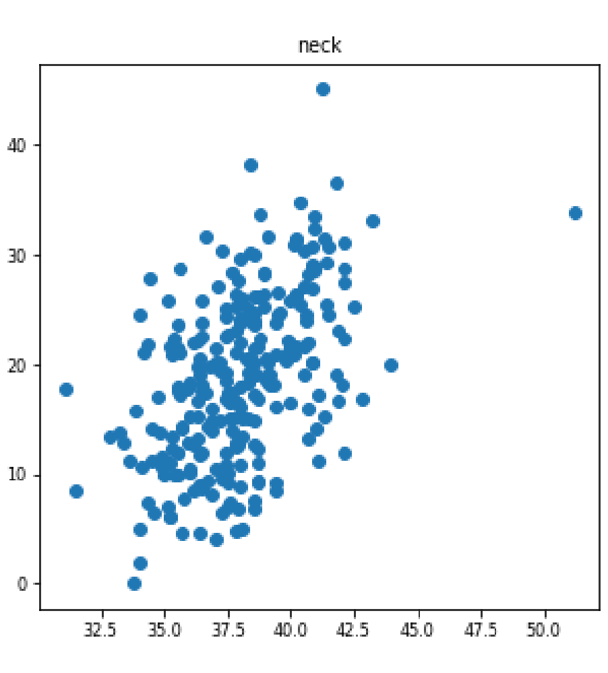


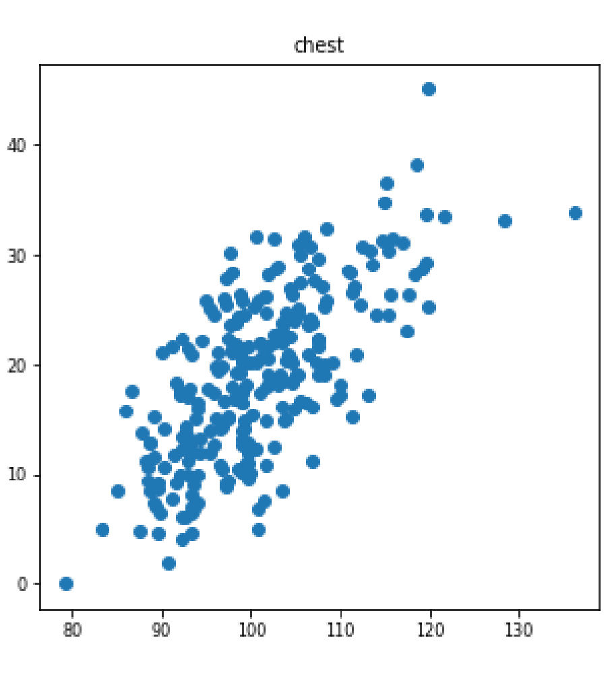


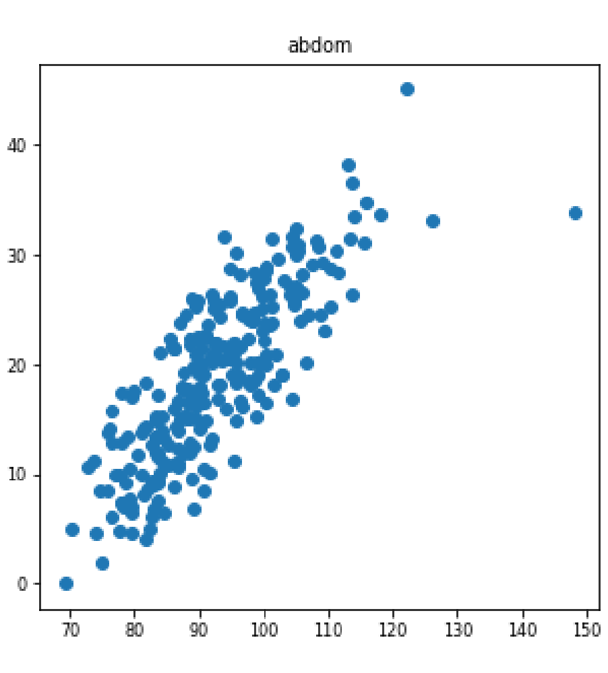


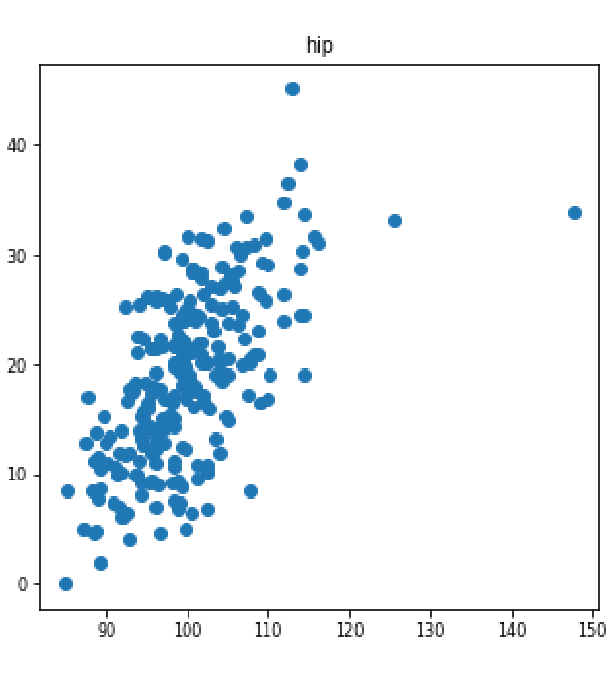


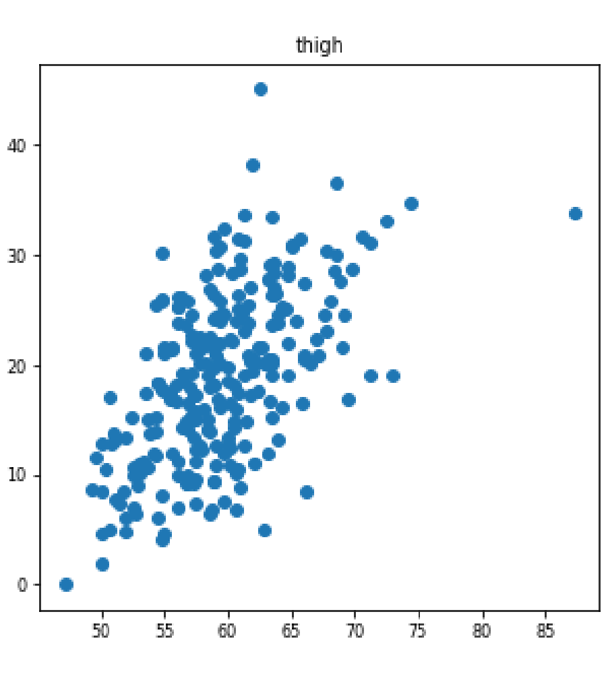


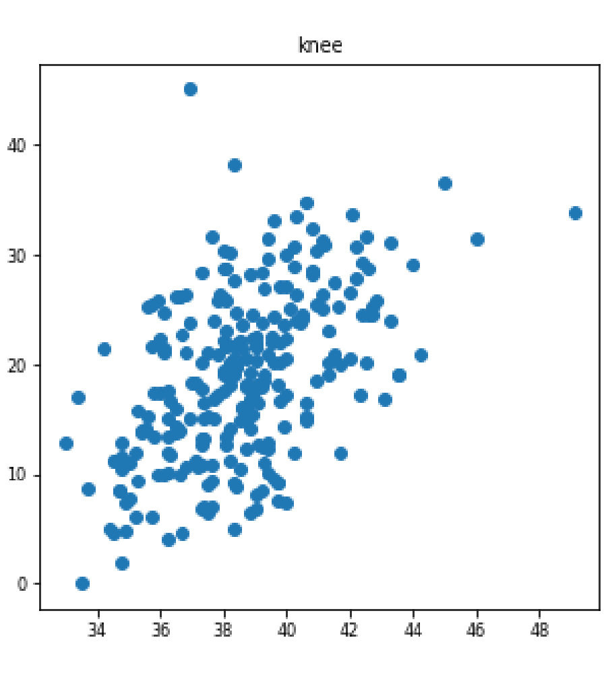


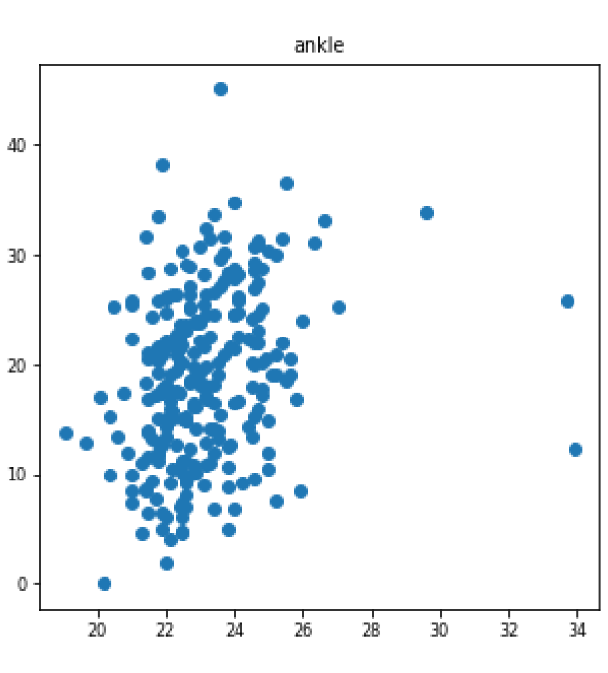


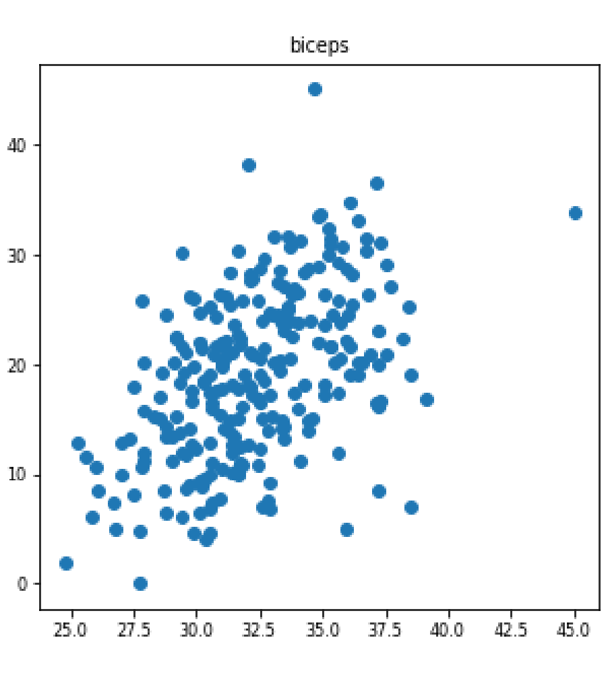


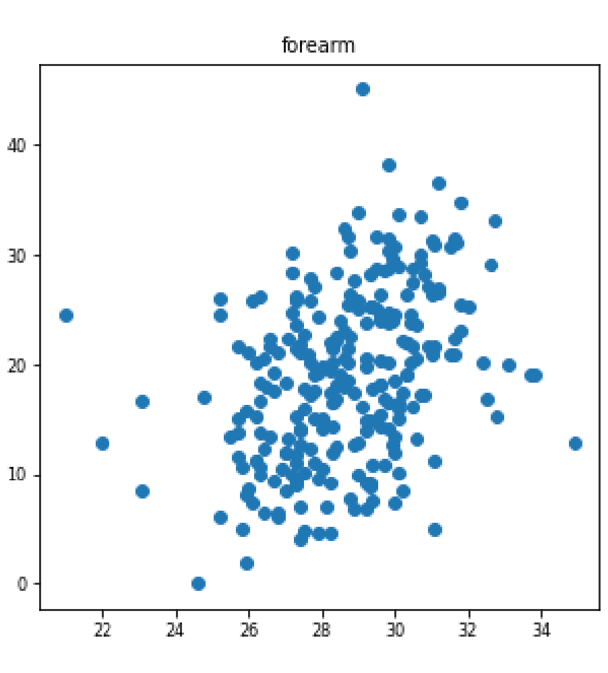


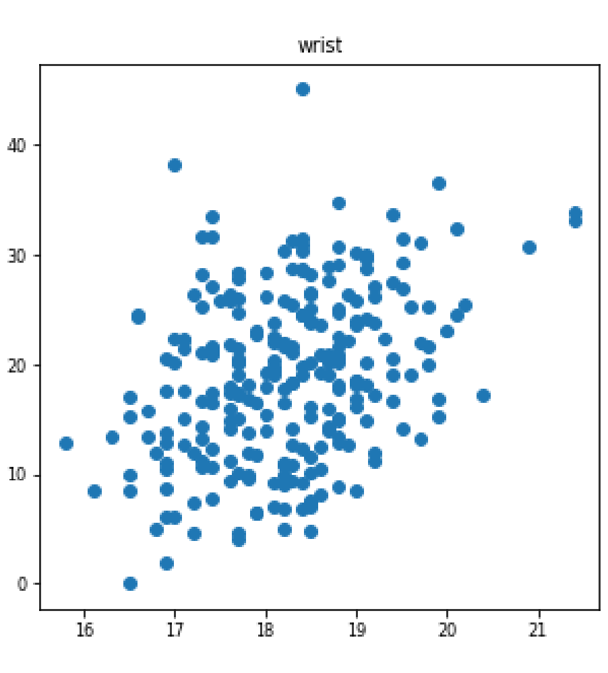












B) Divide el dataset en datos de entrenamiento y prueba.

In [6]:
```
X = df.drop(['brozek'], axis=1) #Categorias
Y = df['brozek'] #Target

# Dividir test y train
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2, random_state=9)
```

C) Aplica regresión lineal

In [7]:
```
reg = LinearRegression()

# Fit
reg.fit(X_train,Y_train)
```

Out[7]:
```
LinearRegression()
```

D) Determina los coeficientes del modelo

In [8]:
```
print(np.c_[reg.coef_, X.columns])
```

```
[[0.8914969265353478 'siri']
 [-9.812199982913336 'density']
 [-0.00024759129919749157 'age']
 [0.005738263736221327 'weight']
 [-0.0004228701961216663 'height']
 [-0.014135040601558346 'adipos']
 [-0.00620584430942922 'free']
 [-0.0016267540940178016 'neck']
 [0.0016513906368942344 'chest']
 [0.005865374758052155 'abdom']
 [-0.005618409499262178 'hip']
 [0.01685891472580847 'thigh']
 [-0.022691581625959523 'knee']
 [0.00214209287425543 'ankle']
 [-0.014765883684972003 'biceps']
 [0.012798158769390722 'forearm']
 [0.0156688552913401 'wrist']]
```

E) Determina el intercepto del modelo

In [9]:
```
reg.intercept_
```

Out[9]:
```
12.04150900797232
```

F) Una vez entrenado tu modelo, has predicciones.

In [10]:
```
predictions = reg.predict(X_test)
predictions
```

Out[10]:
```
array([24.23098923, 21.03139558, 32.23342775, 20.67881792, 15.92240143,
       26.31947855, 22.42211645, 20.18382041, 12.36452194, 12.50301795,
       28.06431524,  9.37343381, 18.48460805, 23.71913603, 31.36456842,
        6.45374182, 14.86068729, 26.21948592,  6.88771155, 12.22899205,
       25.8866731 , 17.18369545, 26.49276222, 23.7705379 ,  6.11984893,
       33.91347466, 17.97031219, 14.75134596, 20.19059905, 14.30596265,
        8.9582139 ,  7.25605226,  4.86211009, 27.51589839, 18.16492   ,
       14.08796394, 33.46450716, 31.35670001, 19.82399272, 13.09412968,
       25.40537445, 18.93937172, 22.42071283,  8.29620259, 10.85721546,
       28.94162923, 18.95325698, 27.08944368, 28.00734179, 16.90096536,
       21.66385469])
```

G) Determina el rendimiento del modelo utilizando dos métricas

In [11]:
```
# Raiz cuadrada del error cuadrático medio
test_rmse = np.sqrt(mean_squared_error(Y_test,predictions))

# Coeficiente de determinacion
test_r2sc = r2_score(Y_test,predictions)

# Remember the lower the value is the better it is
print('Raiz cuadrada del error cuadratico medio:',test_rmse)

# The closer to one the better
print('Coeficiente de determinacion:',test_r2sc)
```

```
Raiz cuadrada del error cuadratico medio: 0.1822230518150871
Coeficiente de determinacion: 0.9994505172476355
```

H) Interpreta la prediccion obtenida.

Ambas métricas dan datos significativos en el momento de la realización del modelo. El coeficiente de deeterminación al ser muy aproximado a 1, nos demuestra que el modelo tiene un buen rendimiento. Y la raiz cuadrada del error cuadratico medio es aproximadamente de 0.1822, este valor es pequeño y eso nos dice que el rendimiento es bueno.