In [1]:
```
#Import neccesary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

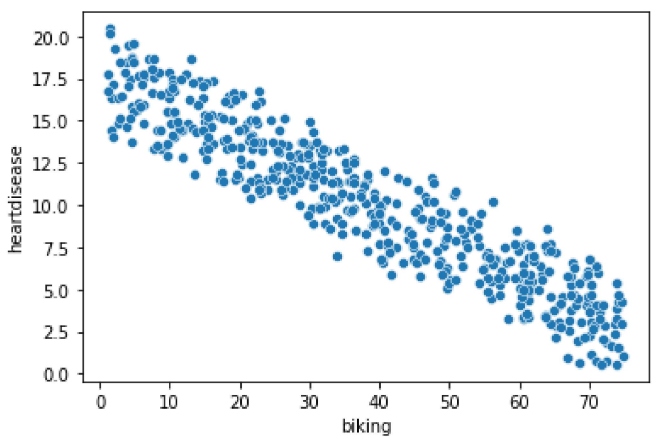
In [2]:
```
#Load dataset and show top 5 rows
df = pd.read_csv("heart.csv")
df.head ()
```

Out[2]:

	SN	biking	smoking	heartdisease
0	1	30.801246	10.896608	11.769423
1	2	65.129215	2.219563	2.854081
2	3	1.959665	17.588331	17.177803
3	4	44.800196	2.802559	6.816647
4	5	69.428454	15.974505	4.062224

In [3]:
```
#scatterplot heart disease vs. biking
y = df['heartdisease']
X = df['biking']
sns.scatterplot(x=X,y=y)
```

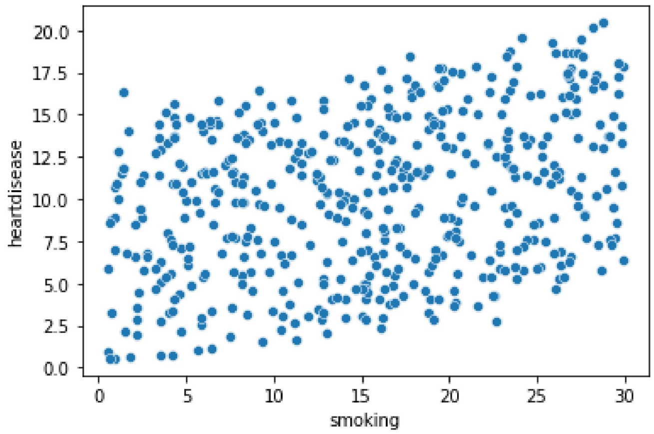
Out[3]: <AxesSubplot:xlabel='biking', ylabel='heartdisease'>



In [4]:
```
#This scatter plot reveals that there is a highly negative
#correlation between individuals who bike and the risk of
#heart disease. This will reflect on the model by providing
#accurate measures of correlation with the subject matter.
```

In [5]:
```
#scatterplot heart disease vs. smoking.
y = df['heartdisease']
X = df['smoking']
sns.scatterplot(x=X,y=y)
```

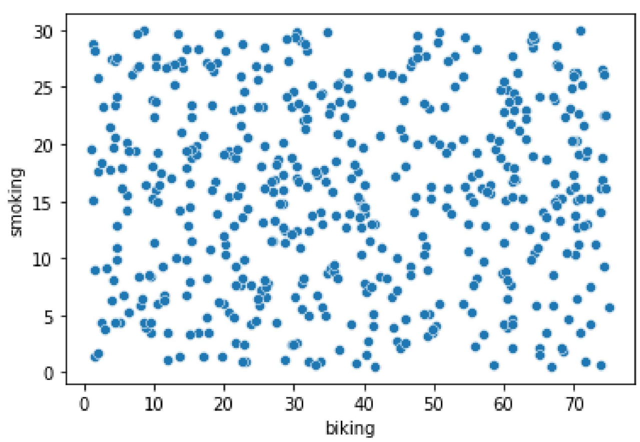
Out[5]: <AxesSubplot:xlabel='smoking', ylabel='heartdisease'>



In [6]:
```
#This scatter plot reveals that there is a weak positive
#correlation between individuals who smoke and the risk
#of heart disease. This will reflect on the model by prompting
#to conduct more analysis between smoking and heart disease
#to further collect more information on the complexity within
#the correlation
```

In [7]:
```
#scatterplot smoking vs biking
y = df['smoking']
X = df['biking']
sns.scatterplot(x=X,y=y)
```

Out[7]: <AxesSubplot:xlabel='biking', ylabel='smoking'>



In [8]:
```
#This scatter plot reveals that there is no correlation between
#individuals who smoke and the individuals who bike. This will reflect
#on the model by reporting that further analysis drawn between biking
#and smoking will not be necessary for the remainder of the model.
```

In [9]:
```
#drop unnecessary columns
X = df.drop(['heartdisease','SN'],axis=1)
y = df['heartdisease']
```

In [10]:
```
#split data into training and test sets
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=15)
```

In [11]:
```
#the purpose of splitting is to avoid the issue of over and under fitting
#so we can train the model by using the training set and apply it to
#properly evaluate the performance of the model
```

In [12]:
```
#Training a linear regression model
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

Out[12]:
```
LinearRegression()
```

In [13]:
```
#display coefficients of the regression model
modelCoef = model.coef_
coeff_df = pd.DataFrame(modelCoef,X.columns,columns=['Coefficient'])
coeff_df
```

Out[13]:

	Coefficient
biking	-0.200524
smoking	0.176604

In [14]:
```
#Calculating the coefficient is a way to define to relationship between
#predictor variables and the response variables
```

In [15]:
```
#The metrics that are going to be computed are the mean absolue error,
#mean squarred error, R^2 and the root mean square error
```

In [16]:
```
#generate test predictions using the model
test_predictions = model.predict(X_test)
```

In [17]:
```
#import metrics and calculate MAE, MSE, and RMSE
from sklearn.metrics import mean_absolute_error,mean_squared_error
MAE = mean_absolute_error(y_test,test_predictions)
MSE = mean_squared_error(y_test,test_predictions)
RMSE = np.sqrt(MSE)
```

In [18]:
```
MAE
```

Out[18]:
```
0.5059659645518587
```

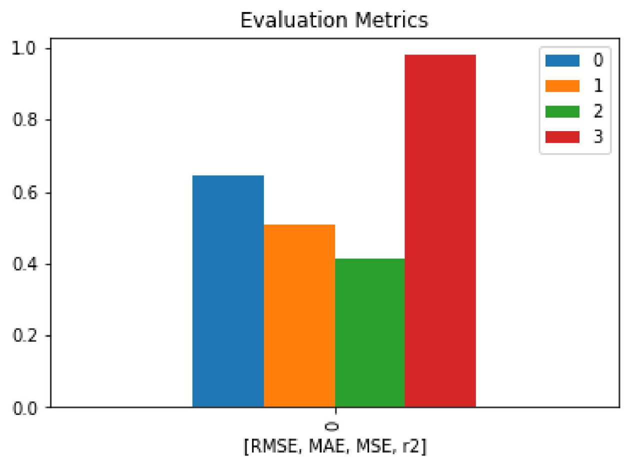
In [19]:
```
MSE
```

Out[19]:
```
0.41549278914354126
```

In [20]:
```
RMSE
```
Out[20]:
```
0.6445873014135023
```

In [21]:
```
#calculate and display r-squared value
from sklearn.metrics import r2_score
r2=r2_score(y_test, test_predictions)
r2
```
Out[21]:
```
0.9792389107198726
```

In [22]:
```
#Create a bar chart to visualize RMSE, MSE, MAE, r2
RMSE = pd.Series(RMSE)
MSE = pd.Series(MSE)
MAE = pd.Series(MAE)
r2 = pd.Series(r2)
metrics = pd.concat([RMSE, MAE, MSE, r2], axis = 1)
metrics.plot.bar(xlabel=['RMSE','MAE','MSE','r2'])
plt.title('Evaluation Metrics')
plt.show()
```



In [23]:
```
#The root square mean error is .644 which means the model can predict
#the data accurately because the measure falls between 0.2 and 0.6.
#The r2 is .979 which means this model is reading a high correlation between
#the independent data and the dependent data..
#The mean absolute error is .505 which means the data is medially
#reliable because the closer the number is to 0, the more reliable the model is.
#The mean squared error is .415 meaning the model is reliable because the
#closer this measure is to 0, the less error the model has.
```

In [24]:
```
#Create a DataFrame for new subjects and predict heart disease based on biking and smoking dat
subjects = [[3.99,13.00], [19.11,15.17], [44.43,19.66],
            [55.13,7.05],[34.22,15.05]]
subjects = pd.DataFrame(subjects, columns = ['biking', 'smoking',])
model.predict(subjects)
```
Out[24]:
```
array([16.51119814, 13.86249862,  9.57817036,  5.20558621, 10.81138186])
```