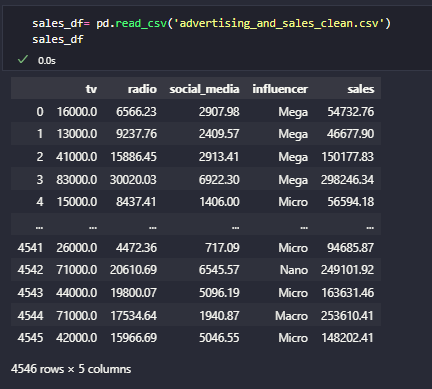
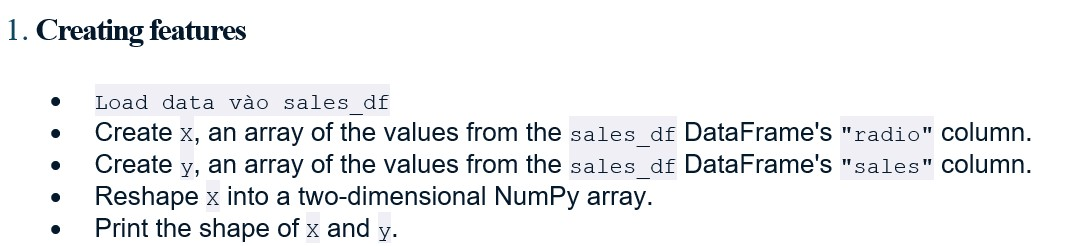
MSSV: 21010611 Họ và Tên: Trần Minh Tiến ca TH: CN t10-12

**Load dataset**





# Tạo mảng x, giá trị cột radio

x = sales\_df['radio'].values

y = sales\_df['sales'].values

# Mảng 2 chiều x

x = x.reshape(-1,1)

print("Mang x", x)

print("Mang y", y)

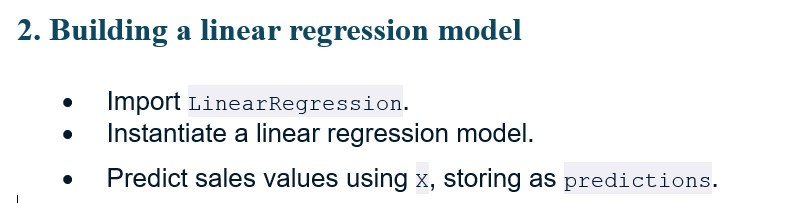
print("In hình dạng cua x", x.shape)

print("In hình dạng cua y", y.shape)

**Kết quả:**

A computer screen shot of a number

Description automatically generated



from sklearn.linear\_model import LinearRegression

model = LinearRegression() # Create a new linear regression model

model.fit(x, y) # Fit the model to the data

# Predict sales values using x, storing as predictions.

Kết quả:

A blue rectangle with a light border

Description automatically generated

predictions = model.predict(x)

print(predictions)

Kết quả:

A screenshot of a computer

Description automatically generated

A close-up of a text

Description automatically generated

plt.scatter (x,y)

plt.plot(x, predictions, *color*='red')

plt.xlabel ("Radio")

plt.ylabel ("Sales")

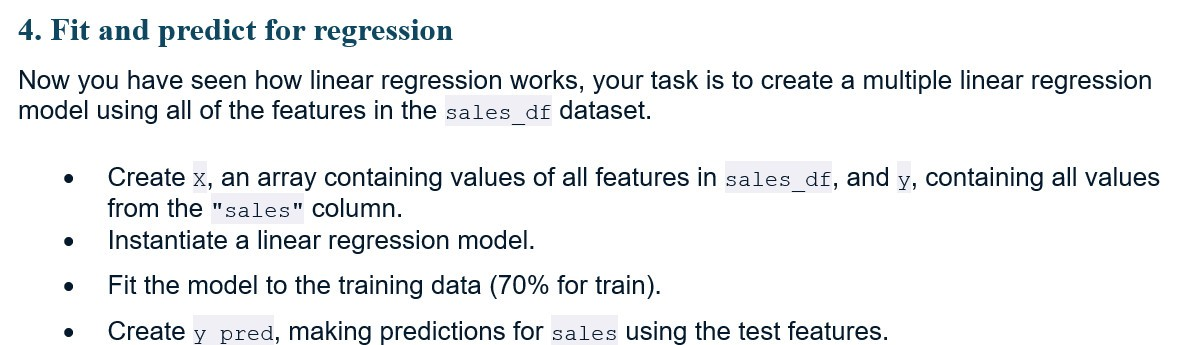
plt.title ("Radio vs Sales")

plt.show()

Kết quả:

A screen shot of a graph

Description automatically generated



from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

# Tao x

x = sales\_df

#Tạo y chua gia tri cua sales và y

y = sales\_df['sales'].values

#  chia du lieu thanh 2 phan train va test

x=sales\_df.drop(['sales','influencer'], *axis*=1).values

y=sales\_df['sales'].values

# KFold: Lớp chia dữ liệu thành các tập huấn luyện và tập kiểm tra theo phương pháp K-Fold

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, *test\_size*=0.3, *random\_state*=40)

#khoi tao mo hinh tuyen tinh

model = LinearRegression()

model.fit(x\_train, y\_train)

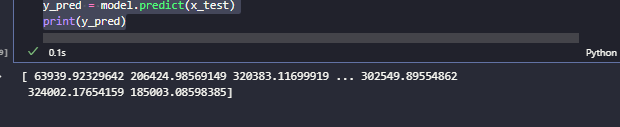
# Huấn luyện mô hình hồi quy tuyến tính trên tập huấn luyện

predictions = model.predict(x\_test)

y\_pred = model.predict(x\_test)

print(y\_pred)

Kết quả



A close-up of a text

Description automatically generated

from sklearn.metrics import mean\_squared\_error

# Nhập lỗi bình phương trung bình mean squared error

r\_squared = model.score(x\_test, y\_test)

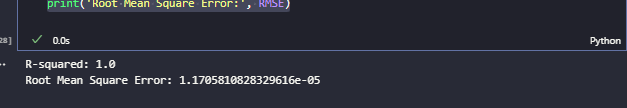
#RMSE = sqrt(MSE)

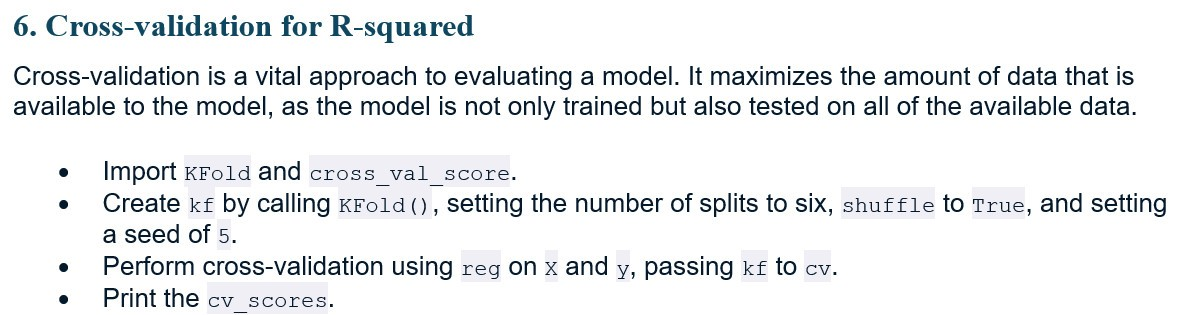
RMSE = np.sqrt(mean\_squared\_error(y\_test, y\_pred, *squared*=False))

print('R-squared:', r\_squared)

print('Root Mean Square Error:', RMSE)

Kết quả





from sklearn.model\_selection import KFold, cross\_val\_score

kf=KFold(*n\_splits*=6,*shuffle*=True,*random\_state*=5)

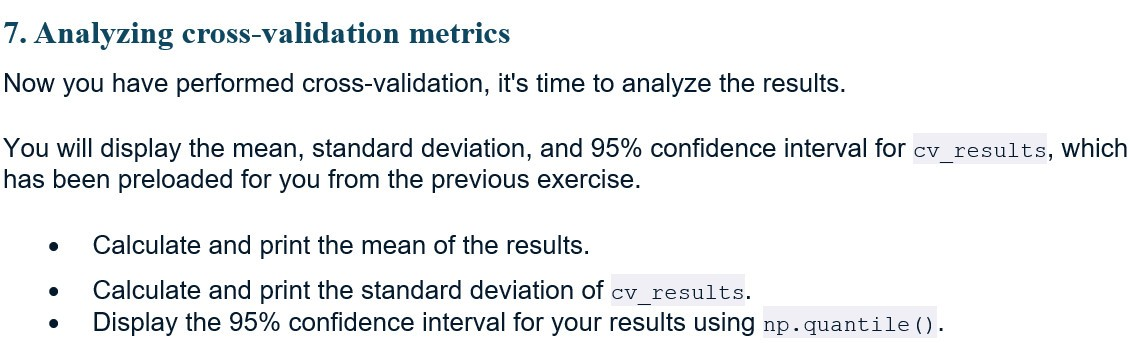
reg=LinearRegression()

cv\_results=cross\_val\_score(reg,x,y,*cv*=kf)

print("Cross Validation Score: ", cv\_results)

Kết quả:





# Trung bình của cv\_results

mean\_cv\_scores = np.mean(cv\_results)

print('mean of cv\_scores: ', mean\_cv\_scores)

#  Do lech chuan của cv\_results

std\_cv\_scores = np.std(cv\_results)

print('standard deviation of cv\_scores:',std\_cv\_scores)

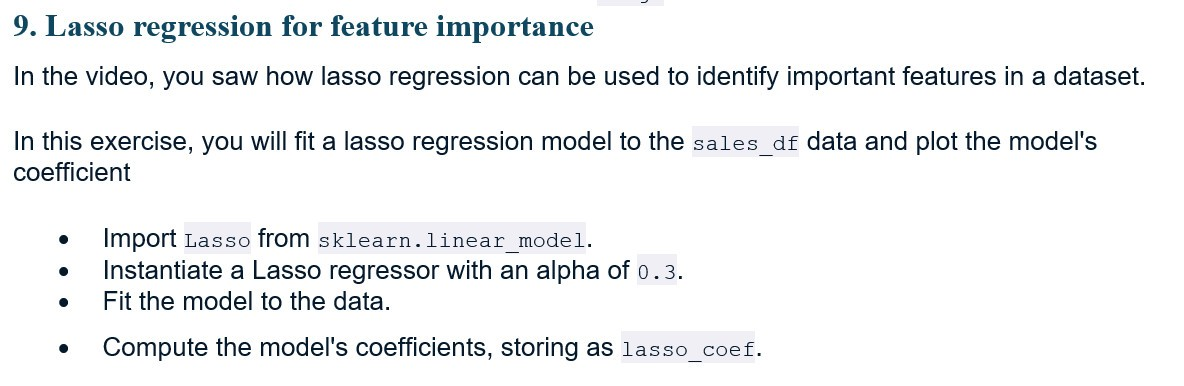
# do tin cay 95%

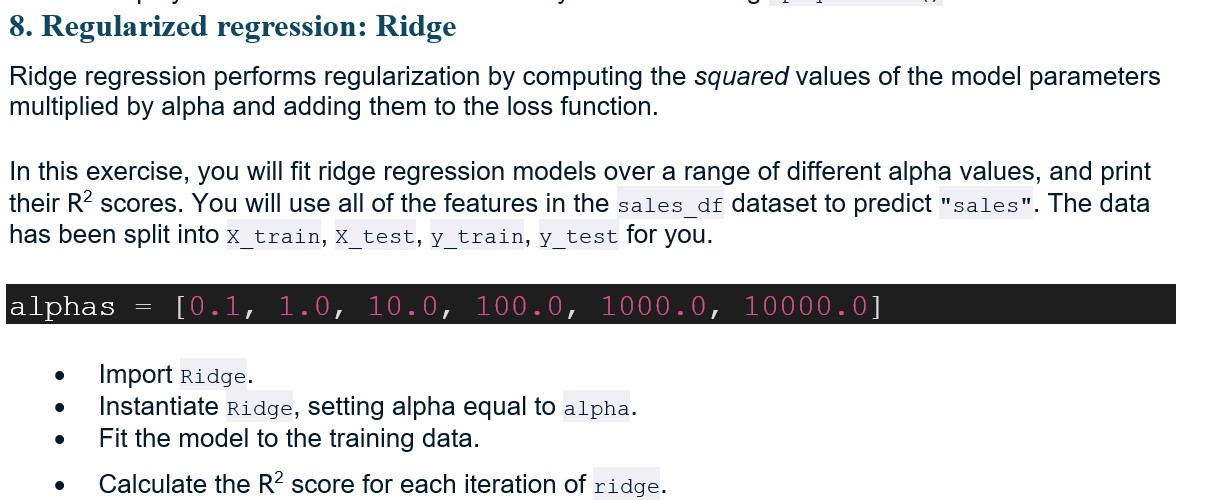
confidene\_interval = np.quantile(cv\_results, [0.025, 0.975])

print('95% confidence interval:', confidene\_interval)

A computer screen shot of numbers

Description automatically generated





from sklearn.linear\_model import Ridge

from sklearn.metrics import r2\_score

#Define alphas

alphas = [0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0]

#Interate over alphas

for alpha in alphas:

    #Khởi tạo và điều chỉnh mô hình hồi quy Ridge

    model = Ridge(*alpha*=alpha)

    model.fit(x\_train, y\_train)

    #Dự đoán doanh số bán hàng bằng test features

    y\_pred = model.predict(x\_test)

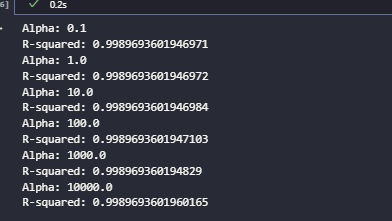
    #Tính R-squared score

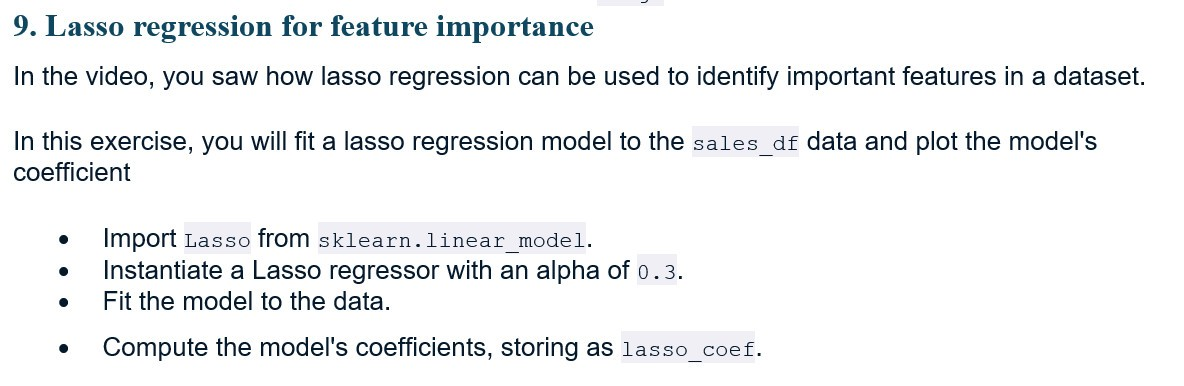
    r2 = r2\_score(y\_test, y\_pred)

    print('Alpha:', alpha)

    print('R-squared:', r2)

Kết quả





from sklearn.linear\_model import Lasso

# Instantite a Lasso regressor with an alpha of 0.3

lasso = Lasso(*alpha*=0.3)

# Dòng này huấn luyện mô hình Lasso trên dữ liệu huấn luyện x\_train và nhãn tương ứng y\_train. Điều này có nghĩa là mô hình sẽ được điều chỉnh để phù hợp với dữ liệu huấn luyện của bạn.lasso.fit(x\_train, y\_train)

# Compute the model's coefficients, doing lasso\_coef

lasso\_coef = lasso.coef\_

print(lasso\_coef)

name = sales\_df.drop(["sales","influencer"], *axis*=1).columns

plt.bar(name, lasso\_coef)

plt.xticks(*rotation*=45)

plt.show()

