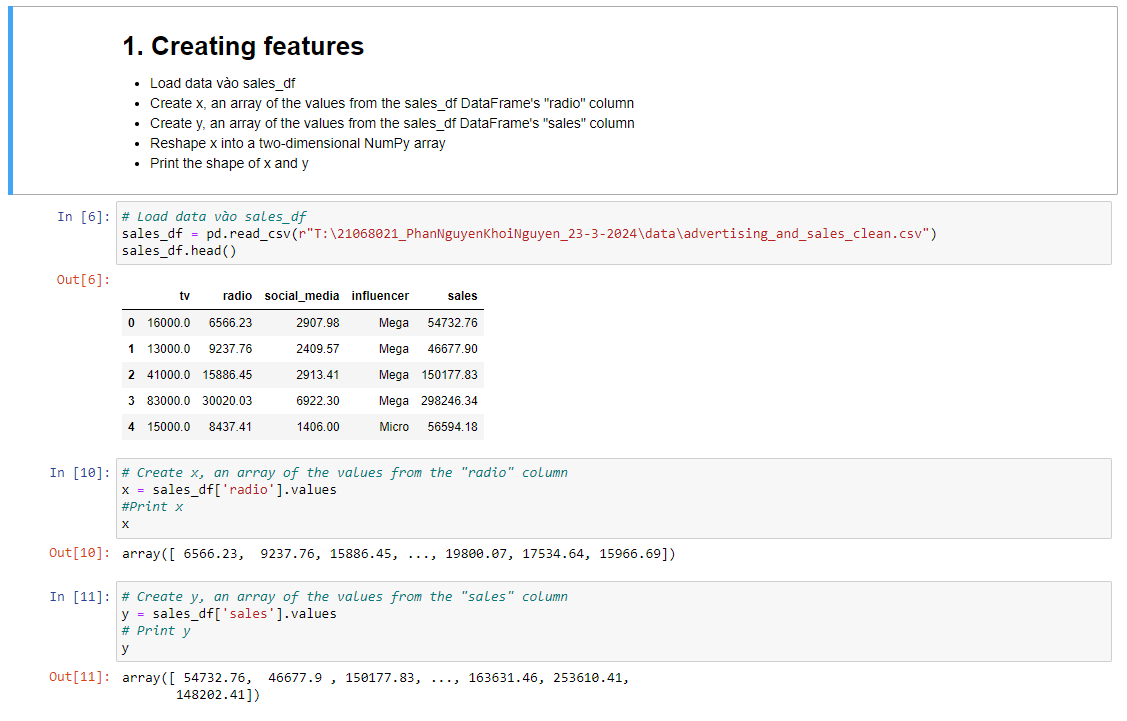
**MSSV** : 21068021

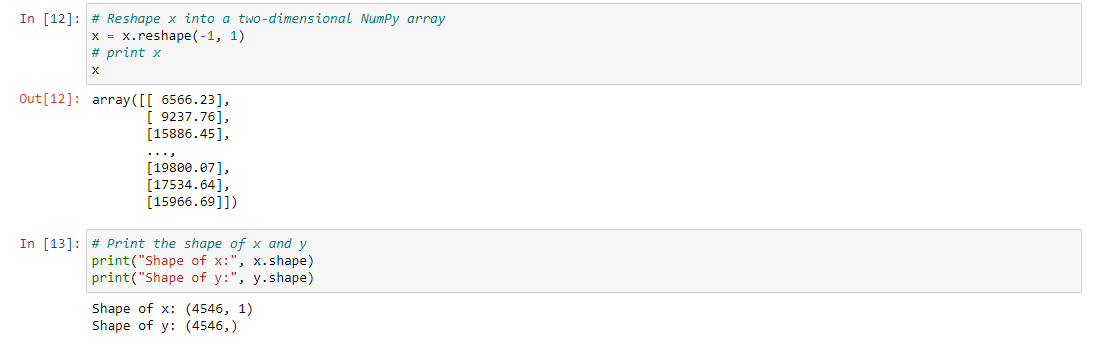
**Họ và Tên** : Phan Nguyễn Khôi Nguyên

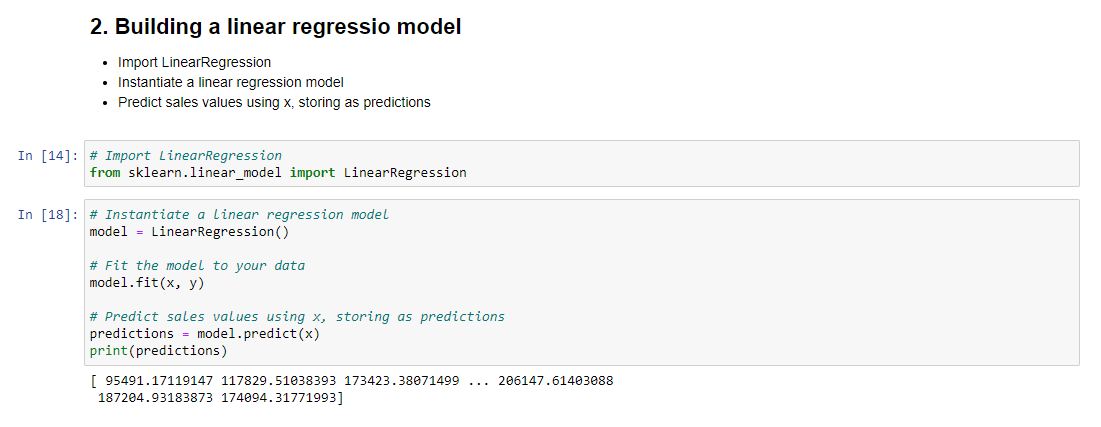
**Ca TH** : Thứ 7 tiết 10-12

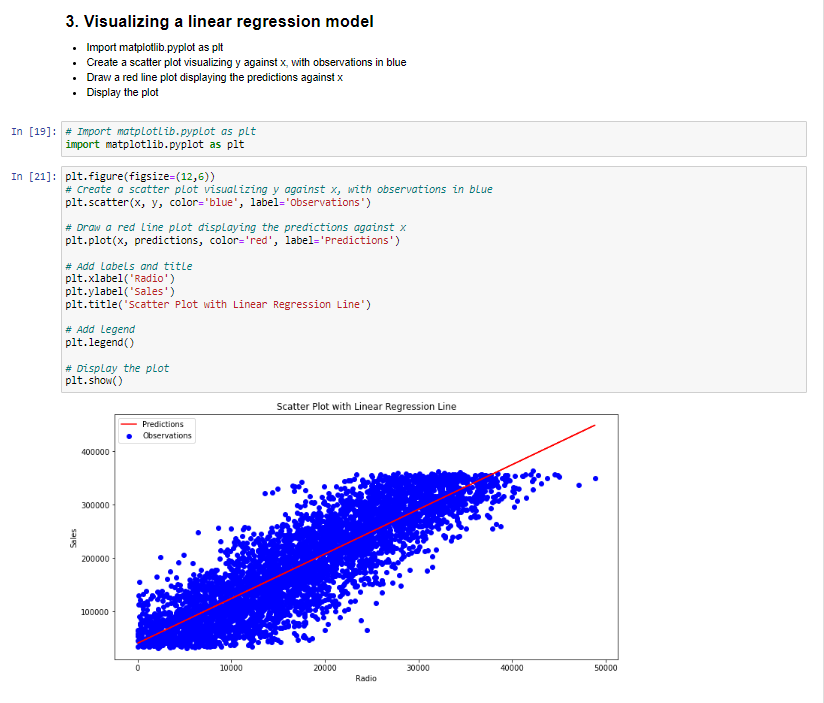
**BÀI LÀM**

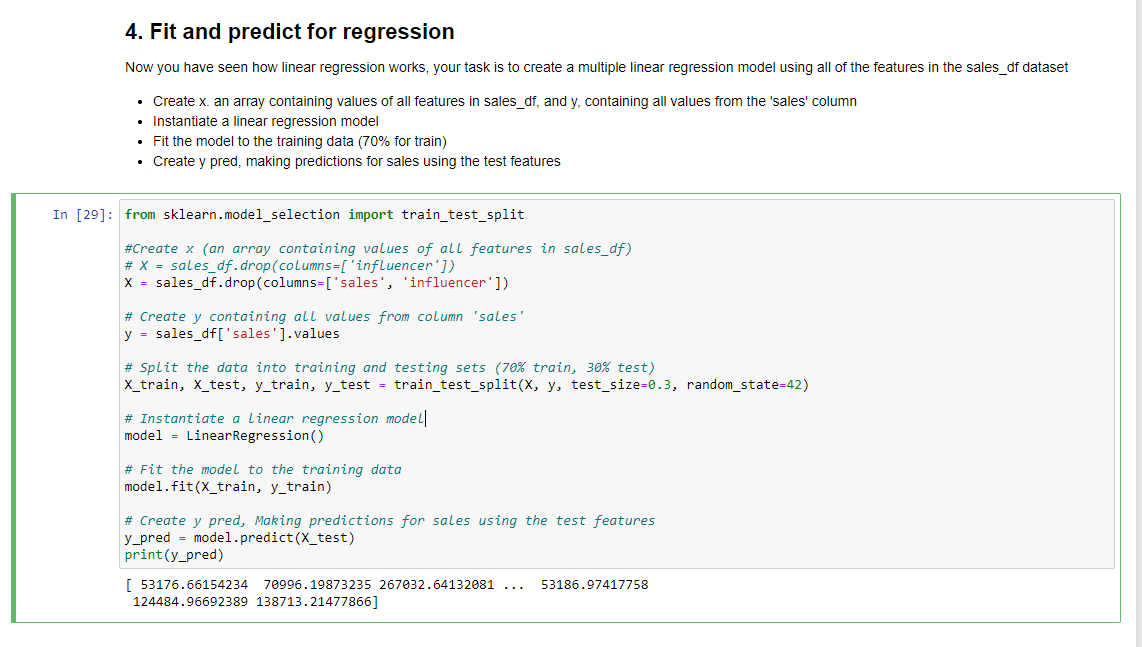
**Hình chụp bài làm :**



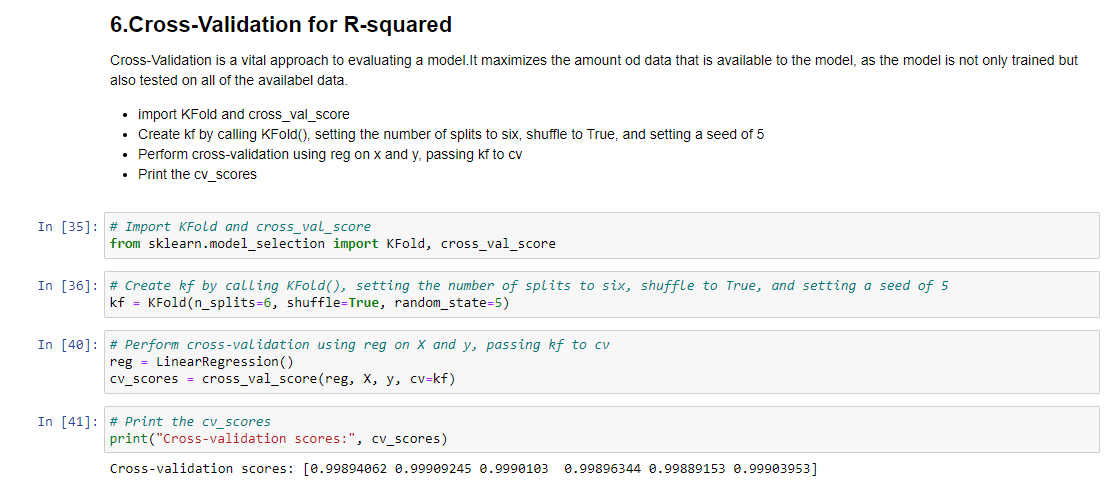


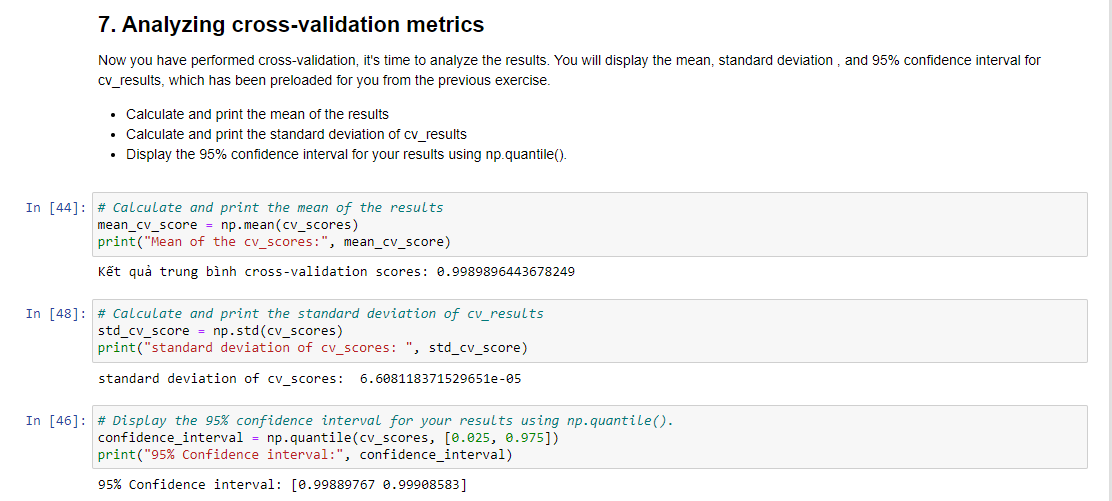




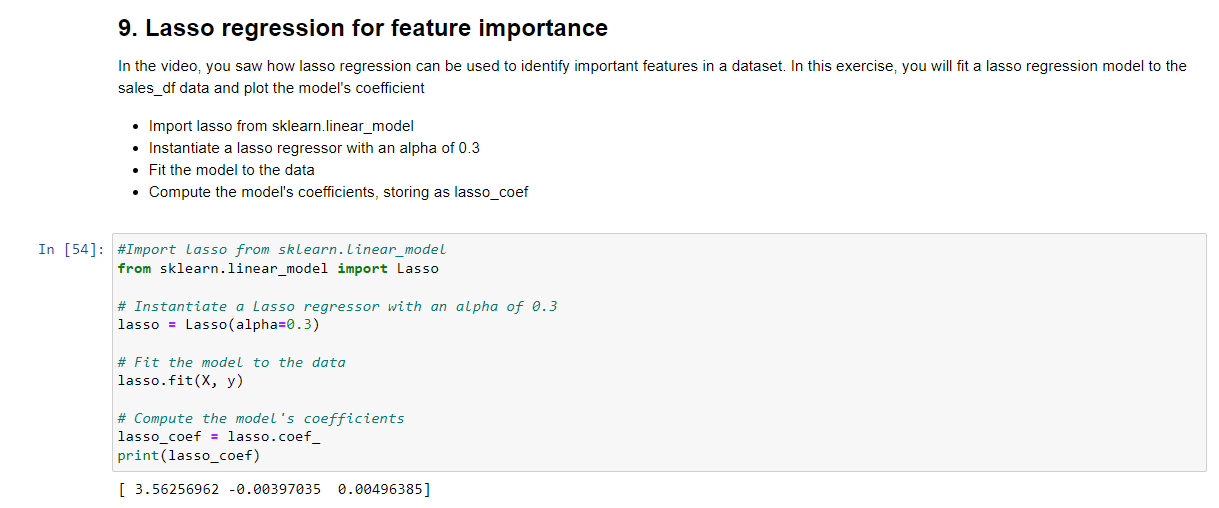


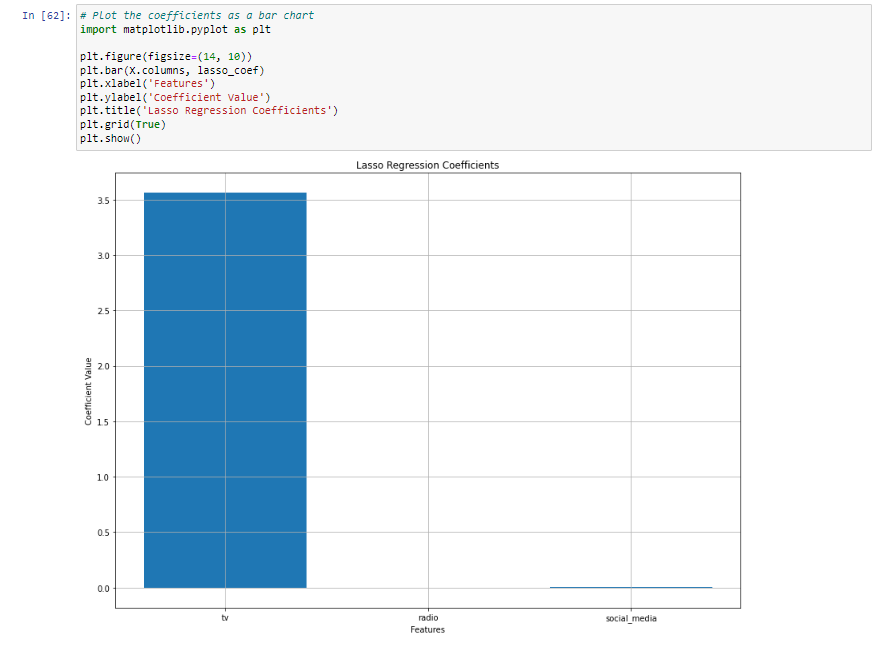












**Code bài làm:**

**Bài 1:**

|  |
| --- |
| # Load data vào sales\_df  sales\_df = pd.read\_csv(r"T:\21068021\_PhanNguyenKhoiNguyen\_23-3-2024\data\advertising\_and\_sales\_clean.csv")  sales\_df.head()  # Create x, an array of the values from the "radio" column  x = sales\_df['radio'].values  #Print x  X  # Create y, an array of the values from the "sales" column  y = sales\_df['sales'].values  # Print y  Y  # Reshape x into a two-dimensional NumPy array  x = x.reshape(-1, 1)  # print x  X  # Print the shape of x and y  print("Shape of x:", x.shape)  print("Shape of y:", y.shape) |

**Bài 2:**

|  |
| --- |
| # Import LinearRegression  from sklearn.linear\_model import LinearRegression  # Instantiate a linear regression model  model = LinearRegression()  # Fit the model to your data  model.fit(x, y)  # Predict sales values using x, storing as predictions  predictions = model.predict(x)  print(predictions) |

**Bài 3:**

|  |
| --- |
| # Import matplotlib.pyplot as plt  import matplotlib.pyplot as plt  plt.figure(figsize=(12,6))  # Create a scatter plot visualizing y against x, with observations in blue  plt.scatter(x, y, color='blue', label='Observations')  # Draw a red line plot displaying the predictions against x  plt.plot(x, predictions, color='red', label='Predictions')  plt.xlabel('Radio')  plt.ylabel('Sales')  plt.title('Scatter Plot with Linear Regression Line')  plt.legend()  # Display the plot  plt.show() |

**Bài 4:**

|  |
| --- |
| from sklearn.model\_selection import train\_test\_split  #Create x (an array containing values of all features in sales\_df)  # X = sales\_df.drop(columns=['influencer'])  X = sales\_df.drop(columns=['sales', 'influencer'])  # Create y containing all values from column 'sales'  y = sales\_df['sales'].values  # Split the data into training and testing sets (70% train, 30% test)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)  # Instantiate a linear regression model  model = LinearRegression()  # Fit the model to the training data  model.fit(X\_train, y\_train)  # Create y pred, Making predictions for sales using the test features  y\_pred = model.predict(X\_test)  print(y\_pred) |

**Bài 5:**

|  |
| --- |
| #Import mean\_squared\_error  from sklearn.metrics import mean\_squared\_error  from math import sqrt  # Calculate R-squared score by passing the test feature values and the test target values to an appropriate method.  r\_squared = model.score(X\_test, y\_test)  # Calculate the model's root mean squared error using y\_test and y\_pred.  rmse = sqrt(mean\_squared\_error(y\_test, y\_pred))  # Print R-squared and RMSE  print("R-squared:", r\_squared)  print("Root Mean Squared Error (RMSE):", rmse) |

**Bài 6:**

|  |
| --- |
| # Import KFold and cross\_val\_score  from sklearn.model\_selection import KFold, cross\_val\_score  # Create kf by calling KFold(), setting the number of splits to six, shuffle to True, and setting a seed of 5  kf = KFold(n\_splits=6, shuffle=True, random\_state=5)  # Perform cross-validation using reg on X and y, passing kf to cv  reg = LinearRegression()  cv\_scores = cross\_val\_score(reg, X, y, cv=kf)  # Print the cv\_scores  print("Cross-validation scores:", cv\_scores) |

**Bài 7:**

|  |
| --- |
| # Calculate and print the mean of the results  mean\_cv\_score = np.mean(cv\_scores)  print("Mean of the cv\_scores:", mean\_cv\_score)  # Calculate and print the standard deviation of cv\_results  std\_cv\_score = np.std(cv\_scores)  print("Standard deviation of cv\_scores: ", std\_cv\_score)  # Display the 95% confidence interval for your results using np.quantile().  confidence\_interval = np.quantile(cv\_scores, [0.025, 0.975])  print("95% Confidence interval:", confidence\_interval) |

**Bài 8:**

|  |
| --- |
| # Import Ridge  from sklearn.linear\_model import Ridge  from sklearn.metrics import r2\_score  # Define a list of alpha values  alphas = [0.1, 1.0, 10.0, 100.0, 1000.0, 10000.0]  # Iterate over each alpha value  for alpha in alphas:  # Instantiate Ridge, setting alpla equal to alpla  ridge\_model = Ridge(alpha=alpha)    # Fit the model to training data  ridge\_model.fit(X\_train, y\_train)  y\_pred = ridge\_model.predict(X\_test) # Make predictions on the test set    # Calculate the R^2 score  r2 = r2\_score(y\_test, y\_pred)    # Print the alpha value and corresponding R^2 score  print(f"Alpha: {alpha}, R^2 Score: {r2}") |

**Bài 9:**

|  |
| --- |
| #Import lasso from sklearn.linear\_model  from sklearn.linear\_model import Lasso  # Instantiate a Lasso regressor with an alpha of 0.3  lasso = Lasso(alpha=0.3)  # Fit the model to the data  lasso.fit(X, y)  # Compute the model's coefficients  lasso\_coef = lasso.coef\_  print(lasso\_coef)  # Plot the coefficients as a bar chart  import matplotlib.pyplot as plt  plt.figure(figsize=(14, 10))  plt.bar(X.columns, lasso\_coef)  plt.xlabel('Features')  plt.ylabel('Coefficient Value')  plt.title('Lasso Regression Coefficients')  plt.grid(True)  plt.show() |