### **&** User

Here is a dataset about auto MPG: "https://huggingface.co/datasets/scikit-learn/auto-mpg/raw/main/auto-mpg.csv". You should download and load the dataset from the URL, take a look at it, check for missing value, do a train-test split, and train a regression model predicting its MPG. You should use everything except "mpg" and "car name" as the input features.

# **CodeActAgent**

To accomplish this task, I will use the pandas library to download and load the dataset, check for missing values, and perform a train-test split. Then, I will use the scikit-learn library to train a regression model.

```
import pandas as pd
import numpy as np
   from sklearn.model_selection import train_test_split
   from sklearn.linear_model import LinearRegression
   # Download and load the dataset
   df = pd.read_csv(
     "https://huggingface.co/datasets/scikit-learn/auto-mpg/raw/main/auto-mpg.csv")
   # Check for missing values
   print(df.isnull().sum())
   # Remove rows with missing values
                                                 Use Pandas Library to Download,
   df = df.dropna()
                                                     Examine, and Process Data
   # Perform train-test split
   X = df.drop(columns=["mpg", "car name"])
   X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                       test_size=0.2, random_state=42)
   # Train a regression model
   model = LinearRegression()
                                                  Use Scikit-Learn Library to Split
   model.fit(X_train, y_train)
                                                  Data and Train Regression Model
   # Predict MPG for test data
   y_pred = model.predict(X_test)
   # Evaluate the model
   print("Mean squared error: ", np.mean((y_test - y_pred) ** 2))
   print("R^2 score: ", r2_score(y_test, y_pred))
```

# **Environment**

# **c** CodeActAgent

It seems that there is a data type issue in the dataset. The '?' character is present in the data, which is causing the error when trying to convert the data to a float. To fix this, I will remove the rows containing the '?' character.

```
# Remove rows containing '?'

df = df.replace('?', np.nan)

df = df.dropna()

X = df.drop(columns=["mpg", "car name"])

y = df["mpg"]

X_train, X_test, y_train, y_test = train_test_split(

X, y, test_size=0.2, random_state=42
)

model = LinearRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print("Mean squared error: ", np.mean((y_test - y_pred) ** 2))

print("R^2 score: ", r2_score(y_test, y_pred))
```

### **Environment**

Mean squared error: 10.710864418838367 R^2 score: 0.790150038676035

# **CodeActAgent**

The mean squared error is 10.711 and the  $R^2$  score is 0.790, which indicates that the model has a good fit with the data [...]

### **user**

Can you also give me the MSE and  $R^2$  for the training set?

# **CodeActAgent** Able

Able to Answer follow-up Question Interactively

Certainly! Here are the mean squared error (MSE) and R^2 scores [...]

```
from sklearn.metrics import mean_squared_error

# Calculate MSE for training set

mse_train = mean_squared_error(y_train, model.predict(X_train))

# Calculate R^2 score for training set

r2_train = r2_score(y_train, model.predict(X_train))

print("Mean squared error (training set):", mse_train)

print("R^2 score (training set):", r2_train)
```

# **Environment**

Mean squared error (training set): 10.979255842373478 R^2 score (training set): 0.826001578671067

# **Ġ** CodeActAgent

The mean squared error for the training set is 10.979 and the  $R^2$  score is 0.826, which indicates [...]

#### **user**

Can you visualize the regression coefficient? You may access them via model.coef\_ and their names via model.feature\_names\_in\_

# **Ġ** CodeActAgent

Certainly! Here is a visualization of the regression coefficients [...]

```
import matplotlib.pyplot as plt

# Get regression coefficients
coef = model.coef_

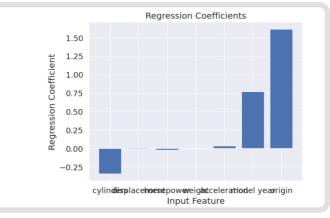
# Get names of input features
feature_names = model.feature_names_in_

# Plot regression coefficients
plt.bar(feature_names, coef)
plt.xlabel('Input Feature')
plt.ylabel('Regression Coefficient')
plt.title('Regression Coefficients')

# Display the plot
plt.show()
Use Matplotlib Library for
Data Visualization
```

# **Environment**

<Figure size 640x480 with 1 Axes>



# **₾** CodeActAgent

The regression coefficients are plotted as a bar chart, with each input feature on the x-axis and the corresponding coefficient on the y-axis [...]

#### **User**

Can you rotate the x ticks by 45 degrees for better readability?

# **©** CodeActAgent

Certainly! Here is the updated bar chart with the x-ticks rotated by 45 degrees for better readability:

```
# Plot regression coefficients

plt.bar(feature_names, coef, tick_labels=False)

plt.xticks(feature_names, feature_names,

rotation=45, ha='right', fontsize=12)

[..same code omitted...]
```

#### **Environment**

<Figure size 640x480 with 1 Axes>

# **Ġ** CodeActAgent

It seems that the tick\_labels argument is not a valid argument for the bar() function in Matplotlib. Instead, we can use the tick\_params() function to rotate the x-ticks [...]

[3 rounds of self-debugging omitted for space]

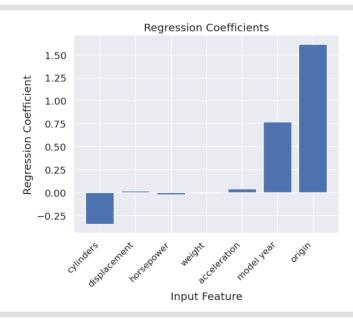
### **Ġ** CodeActAgent

I apologize for the confusion. It seems that the set\_xticks() function is not a valid function in Matplotlib. Instead, we can use the xticks() function to set the x-ticks and rotate them by 45 degrees for better readability. Here is the updated bar chart with the x-ticks rotated by 45 degrees:

```
plt.xticks(feature_names, feature_names, rotation=45, ha='right', fontsize=12)
[...same code omitted...]
```

### **&** Environment

```
<Figure size 640x480 with 1 Axes>
```





[Stops Interaction]