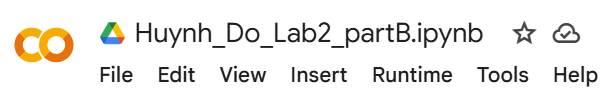
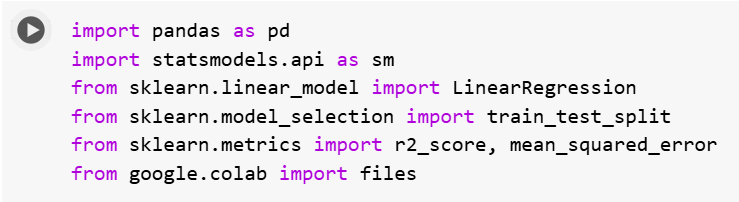
**Huynh Do Lab#2 Part B:**Objective: Use Python in Google Colab to download your dataset "bikes.csv", to explore and summarize the Regression



1. **Import libraries**

  
The code above imports several Python libraries commonly used in data analysis.

1. import pandas as pd: Brings in Pandas, r go-to library for working with tabular data (like CSV files).

* Use it to load, clean, manipulate, and analyze datasets.
* Example: df = pd.read\_csv("data.csv")

1. import statsmodels.api as sm :

* This loads Statsmodels, a library for doing deep-dive statistical analysis.Useful for regression with detailed output: coefficients, p-values, confidence intervals, etc.
* Commonly used for OLS (Ordinary Least Squares) regression.

1. from sklearn.linear\_model import LinearRegression :

* model.fit(X\_train, y\_train)
* Grabs the Linear Regression model from Scikit-learn.
* Faster, simpler, great for prediction.

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

* Split r dataset into training and testing sets.
* Ensures 're testing r model on unseen data.
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2)

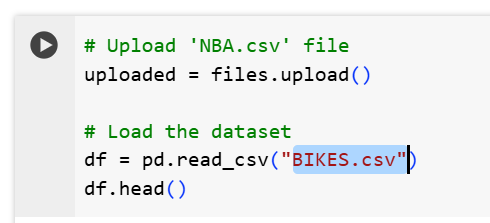
1. from google.colab import files

* Used in Google Colab to upload/download files from your local system within a Colab notebook.

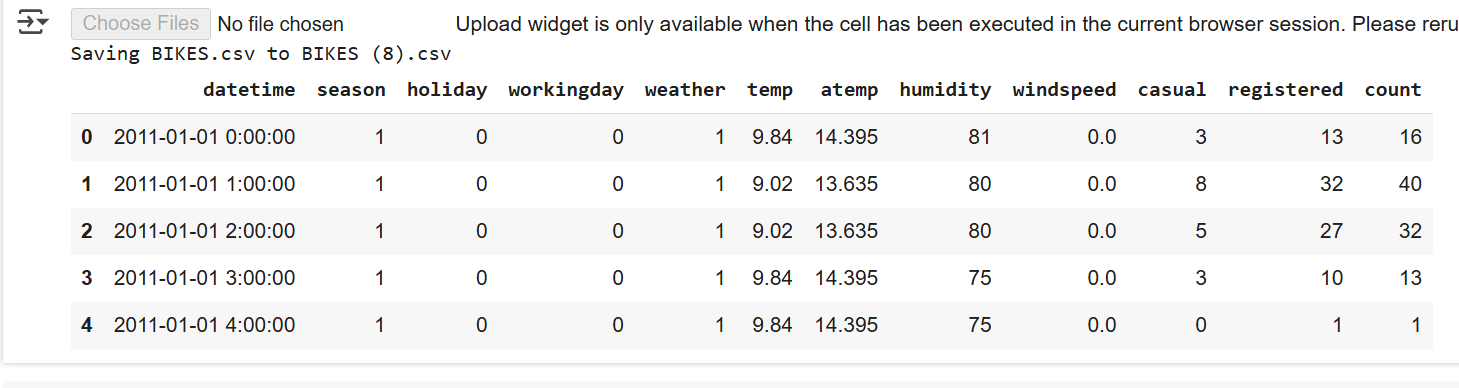
1. from sklearn.metrics import r2\_score, mean\_squared\_error

* These are tools to evaluate r model’s performance.
* r2\_score: How well r model explains the variance (ranges from 0 to 1; higher is better).
* mean\_squared\_error: Average of the squared differences between predicted and actual values (lower is better).

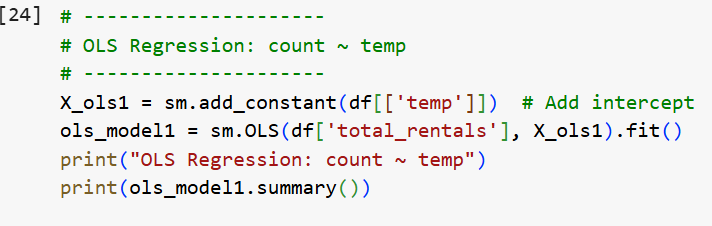
1. **Upload file BIKES.csv**



The above screen shot is used to upload and load a CSV file named **BIKE.csv** into a Pandas Data Frame in a Google Colab environment.

When upload is done:  


1. **OLS Regression: count ~ temp**



1. df[['temp']]

* Selecting the temperature column from the DataFrame df as r independent variable (X).
* Double square brackets [[]] keep it as a DataFrame (not a Series), which is what statsmodels expects.

1. sm.add\_constant(...)

* OLS (Ordinary Least Squares) needs an intercept term — a baseline value when all predictors are 0.
* X\_ols1 = sm.add\_constant(df[['temp']]) adds a column of 1s to r X matrix so r model can learn the intercept (i.e., y = b0 + b1 \* temp).

1. sm.OLS(df['total\_rentals'], X\_ols1

* This sets up an OLS regression model, predicting: total\_rentals ~ temp where: df['total\_rentals'] = dependent variable (what you're trying to predict, aka y)

1. .fit()

This trains the model using data:

It computes the best-fit line using least squares and returns an object (ols\_model1) that stores all the regression results.

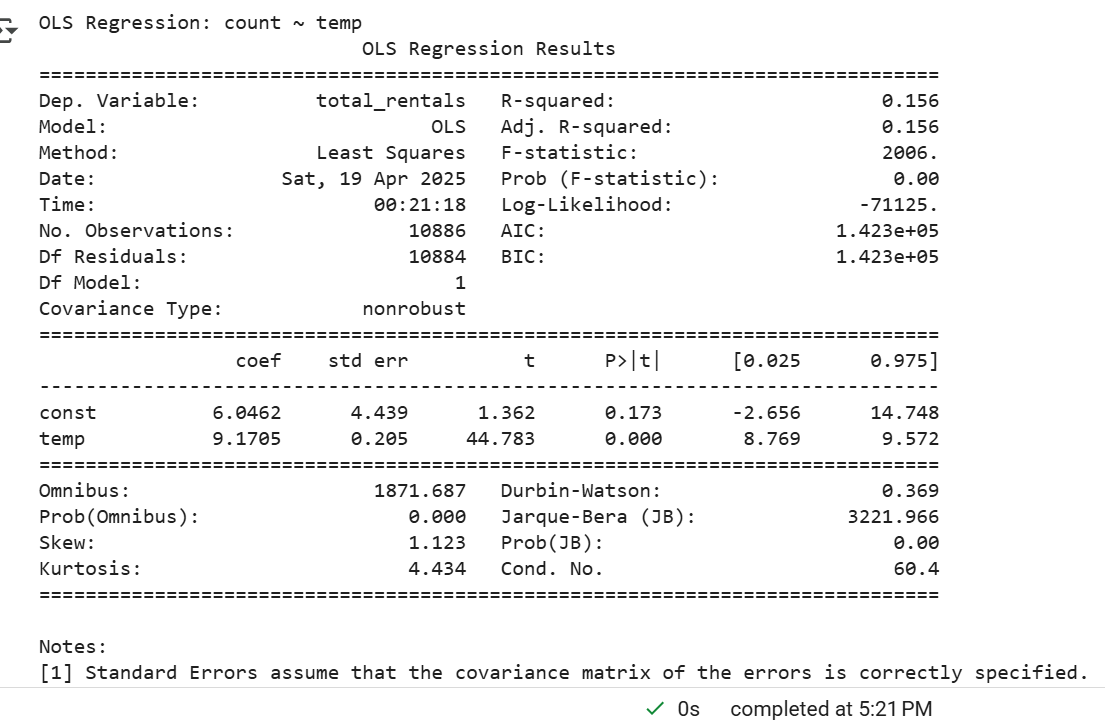
1. print(ols\_model1.summary()):

This prints out a detailed regression summary, including:

* Coefficients (slope and intercept)
* Standard errors

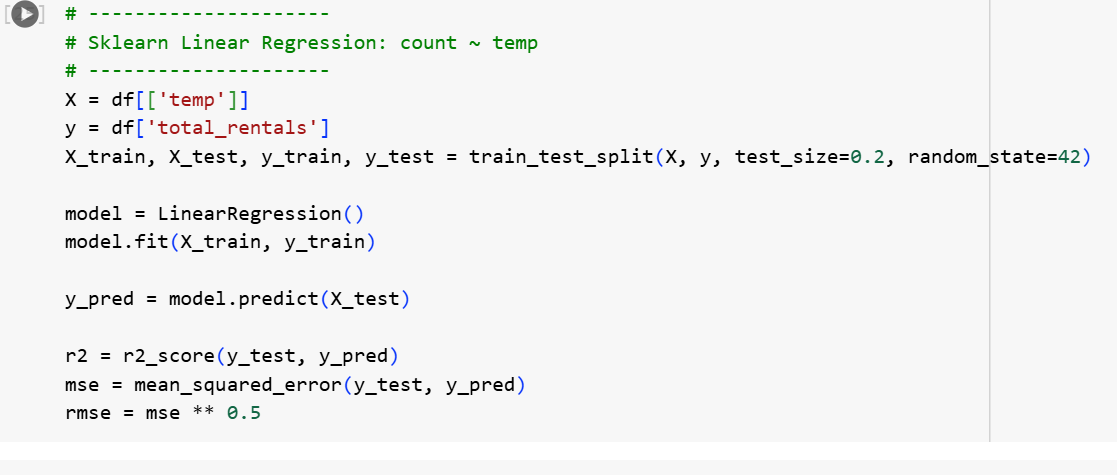
1. Summary

* The result predicts that temp actually has a statistically significant impact — or if it's just hot air.



The above figure is the result of running the OLS Regression python code

1. **Sklearn Linear Regression: count ~ temp**



1. train\_test\_split(...)

* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)
* Splits data:80% for training and 20% for testing
* random\_state=42 ensures reproducibility (so get the same split every time)

1. model = LinearRegression()\
2. model.fit(X\_train, y\_train)

* Trains the model using the training data.
* Learns the best-fit line: total\_rentals ≈ b0 + b1 \* temp

1. y\_pred = model.predict(X\_test)

Predicts rental counts using the test set temperatures.

Gives a list of predicted values based on the learned model.

1. r2 = r2\_score(y\_test, y\_pred)

* Calculates R-squared, which tells how well r model explains the variability in the data.
* Ranges from 0 to 1:
* 1 = perfect prediction
* 0 = no better than guessing the mean

1. mse = mean\_squared\_error(y\_test, y\_pred)

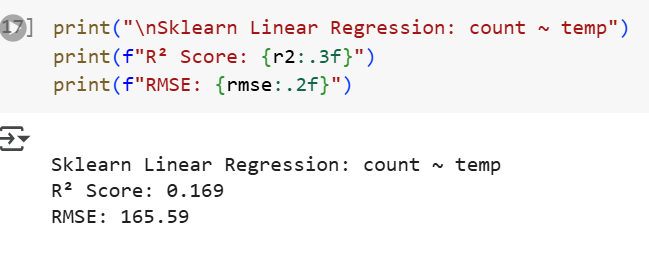
MSE = average of squared prediction errors

Lower = better

Sensitive to outliers due to squaring

1. rmse = mse \*\* 0.5

RMSE = Root Mean Squared Error

****The above figure showed the final results

**Quick summary:**

1. Temperature **does have an effect** on rentals.
2. Warmer days = more rentals. The relationship is statistically significant.