Upload the Dataset

```
from google.colab import files

uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving student-mat.csv to student-mat.csv
```

Load the Dataset

```
import pandas as pd

# Read the dataset
df = pd.read_csv('student-mat.csv', sep=';')
```

Data Exploration

```
# Display first few rows
df.head()
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	freetime	goout	Dalc	Walc	health	absences	G1	G2	G3
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	3	4	1	1	3	6	5	6	6
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	3	3	1	1	3	4	5	5	6
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	3	2	2	3	3	10	7	8	10
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	2	2	1	1	5	2	15	14	15
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	3	2	1	2	5	4	6	10	10

5 rows × 33 columns

```
# Shape of the dataset
print("Shape:", df.shape)

# Column names
print("Columns:", df.columns.tolist())

# Data types and non-null values
df.info()

# Summary statistics for numeric features
df.describe()
```

```
Shape: (395, 33)
Columns: ['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime', 'failures', 'schoolsup', 'f
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 395 entries, 0 to 394
Data columns (total 33 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   school      395 non-null    object
 1   sex         395 non-null    object
 2   age         395 non-null    int64
 3   address     395 non-null    object
 4   famsize     395 non-null    object
 5   Pstatus     395 non-null    object
 6   Medu        395 non-null    int64
 7   Fedu        395 non-null    int64
 8   Mjob        395 non-null    object
 9   Fjob        395 non-null    object
 10  reason      395 non-null    object
 11  guardian    395 non-null    object
 12  traveltime  395 non-null    int64
 13  studytime   395 non-null    int64
 14  failures    395 non-null    int64
 15  schoolsup   395 non-null    object
 16  famsup      395 non-null    object
 17  paid        395 non-null    object
 18  activities  395 non-null    object
 19  nursery     395 non-null    object
 20  higher      395 non-null    object
 21  internet    395 non-null    object
 22  romantic    395 non-null    object
 23  famrel      395 non-null    int64
 24  freetime    395 non-null    int64
 25  goout       395 non-null    int64
 26  Dalc        395 non-null    int64
 27  Walc        395 non-null    int64
 28  health      395 non-null    int64
 29  absences    395 non-null    int64
 30  G1          395 non-null    int64
 31  G2          395 non-null    int64
 32  G3          395 non-null    int64
dtypes: int64(16), object(17)
memory usage: 102.0+ KB
```

	age	Medu	Fedu	traveltime	studytime	failures	famrel	freetime	goout	Dalc	Walc	health	absences	G1	
count	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000000	395.000(
mean	16.696203	2.749367	2.521519	1.448101	2.035443	0.334177	3.944304	3.235443	3.108861	1.481013	2.291139	3.554430	5.708861	10.908861	10.713(

std	1.276043	1.094735	1.088201	0.697505	0.839240	0.743651	0.896659	0.998862	1.113278	0.890741	1.287897	1.390303	8.003096	3.319195	3.761
min	15.000000	0.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	3.000000	0.000
25%	16.000000	2.000000	2.000000	1.000000	1.000000	0.000000	4.000000	3.000000	2.000000	1.000000	1.000000	3.000000	0.000000	8.000000	9.000
50%	17.000000	3.000000	2.000000	1.000000	2.000000	0.000000	4.000000	3.000000	3.000000	1.000000	2.000000	4.000000	4.000000	11.000000	11.000
75%	18.000000	4.000000	3.000000	2.000000	2.000000	0.000000	5.000000	4.000000	4.000000	2.000000	3.000000	5.000000	8.000000	13.000000	13.000
max	22.000000	4.000000	4.000000	4.000000	4.000000	3.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	75.000000	19.000000	19.000

Check for Missing Values and Duplicates

```
# Check for missing values
print(df.isnull().sum())

# Check for duplicates
print("Duplicate rows:", df.duplicated().sum())
```

```
school        0
sex           0
age           0
address       0
famsize       0
Pstatus       0
Medu          0
Fedu          0
Mjob          0
Fjob          0
reason        0
guardian      0
traveltime    0
studytime     0
failures      0
schoolsup     0
famsup        0
paid          0
activities    0
nursery       0
higher        0
```

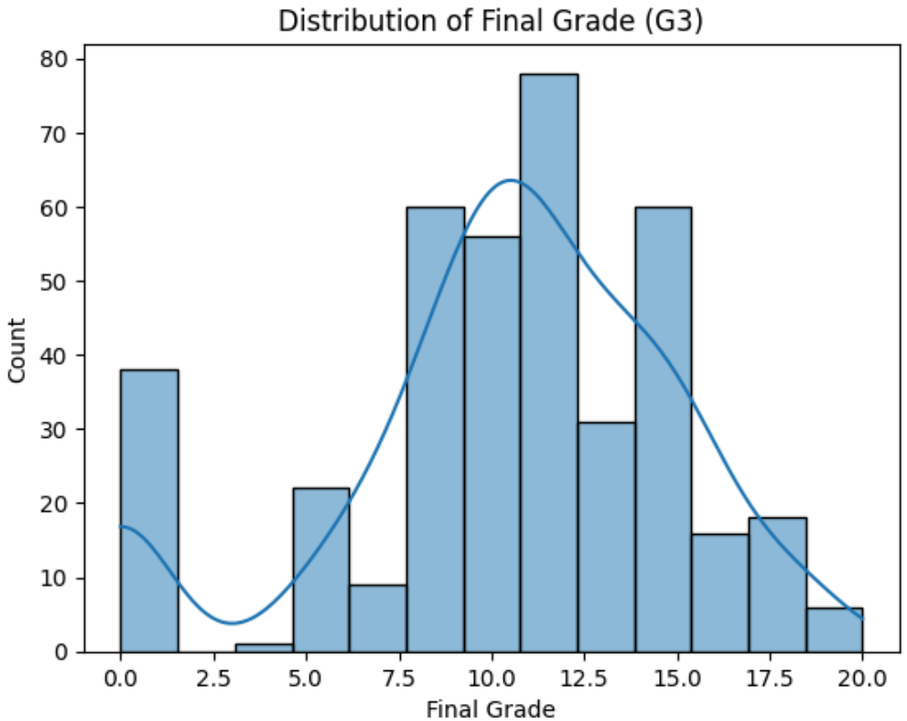
```
internet      0
romantic      0
famrel        0
freetime      0
goout         0
Dalc          0
Walc          0
health        0
absences      0
G1            0
G2            0
G3            0
dtype: int64
Duplicate rows: 0
```

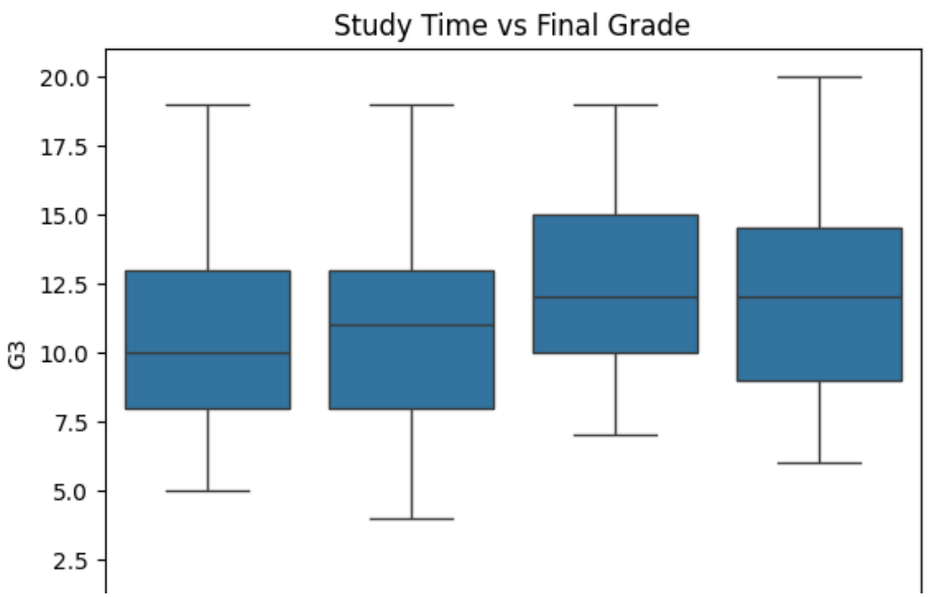
Visualize a Few Features

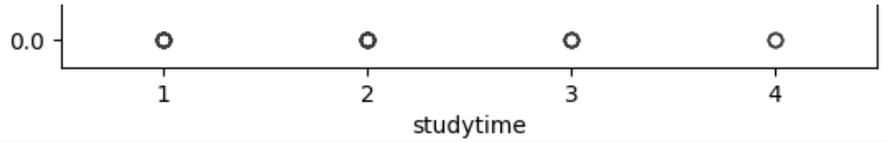
```
import seaborn as sns
import matplotlib.pyplot as plt

# Distribution of final grades
sns.histplot(df['G3'], kde=True)
plt.title('Distribution of Final Grade (G3)')
plt.xlabel('Final Grade')
plt.show()

# Relationship between study time and final grade
sns.boxplot(x='studytime', y='G3', data=df)
plt.title('Study Time vs Final Grade')
plt.show()
```


Distribution of Final Grade (G3)
Count
Final Grade
0
10
20
30
40
50
60
70
80
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0


Study Time vs Final Grade
G3
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0



Identify Target and Features

```
target = 'G3'
features = df.columns.drop(target)
print("Features:", features)
```

```
Features: Index(['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu', 'Fedu',
       'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime', 'studytime',
       'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery',
       'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc',
       'Walc', 'health', 'absences', 'G1', 'G2'],
      dtype='object')
```

Convert Categorical Columns to Numerical

```
# Identify categorical columns
categorical_cols = df.select_dtypes(include=['object']).columns
print("Categorical Columns:", categorical_cols.tolist())
```

```
Categorical Columns: ['school', 'sex', 'address', 'famsize', 'Pstatus', 'Mjob', 'Fjob', 'reason', 'guardian', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher',
```

One-Hot Encoding

```
df_encoded = pd.get_dummies(df, drop_first=True)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_encoded.drop('G3', axis=1))
y = df_encoded['G3']
```

Train-Test Split

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)
```

Model Building

```
# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict
y_pred = model.predict(X_test)
```

Evaluation

```
# Evaluate
```

```
print("MSE:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
MSE: 5.656642833231218
R² Score: 0.7241341236974024
```

Make Predictions from New Input

```
# Sample input (replace values with any other valid values from the original dataset)
new_student = {
    'school': 'GP',             # 'GP' or 'MS'
    'sex': 'F',                 # 'F' or 'M'
    'age': 17,                  # Integer
    'address': 'U',            # 'U' or 'R'
    'famsize': 'GT3',          # 'LE3' or 'GT3'
    'Pstatus': 'A',            # 'A' or 'T'
    'Medu': 4,                 # 0 to 4
    'Fedu': 3,                 # 0 to 4
    'Mjob': 'health',          # 'teacher', 'health', etc.
    'Fjob': 'services',
    'reason': 'course',
    'guardian': 'mother',
    'traveltime': 2,
    'studytime': 3,
    'failures': 0,
    'schoolsup': 'yes',
    'famsup': 'no',
    'paid': 'no',
    'activities': 'yes',
    'nursery': 'yes',
    'higher': 'yes',
    'internet': 'yes',
    'romantic': 'no',
    'famrel': 4,
    'freetime': 3,
    'goout': 3,
    'Dalc': 1,
    'Walc': 1,
    'health': 4,
    'absences': 2,
```

```
    'G1': 14,
    'G2': 15
}
```

Convert to DataFrame and Encode

```
import numpy as np

# Convert to DataFrame
new_df = pd.DataFrame([new_student])

# Combine with original df to match columns
df_temp = pd.concat([df.drop('G3', axis=1), new_df], ignore_index=True)

# One-hot encode
df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)

# Match the encoded feature order
df_temp_encoded = df_temp_encoded.reindex(columns=df_encoded.drop('G3', axis=1).columns, fill_value=0)

# Scale (if you used scaling)
new_input_scaled = scaler.transform(df_temp_encoded.tail(1))
```

Predict the Final Grade

```
predicted_grade = model.predict(new_input_scaled)
print("🎓 Predicted Final Grade (G3):", round(predicted_grade[0], 2))
```

```
🎓 Predicted Final Grade (G3): 14.94
```

Deployment-Building an Interactive App

```
!pip install gradio
```

```
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.2)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.9.0->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.9.0->gradio) (15.0.1)
```

```
Downloading ruff-0.11.7-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 78.8 MB/s eta 0:00:00
Downloading safehttpx-0.1.6-py3-none-any.whl (8.7 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 6.1 MB/s eta 0:00:00
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Downloading uvicorn-0.34.2-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 4.3 MB/s eta 0:00:00
Downloading ffmpy-0.5.0-py3-none-any.whl (6.0 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub, uvicorn, tomlkit, semantic-version, ruff, python-multipart, groovy, ffmpy, aiofiles, starlette, safehttpx, gradio-client, fastapi, grad
Successfully installed aiofiles-24.1.0 fastapi-0.115.12 ffmpy-0.5.0 gradio-5.27.0 gradio-client-1.9.0 groovy-0.1.2 pydub-0.25.1 python-multipart-0.0.20 ruff-0.11.7 safehttpx
```

Create a Prediction Function

```
import gradio as gr

def predict_grade(school, sex, age, address, famsize, Pstatus, Medu, Fedu,
                  Mjob, Fjob, reason, guardian, traveltime, studytime,
                  failures, schoolsup, famsup, paid, activities, nursery,
                  higher, internet, romantic, famrel, freetime, goout,
                  Dalc, Walc, health, absences, G1, G2):

    # Create input dictionary
    input_data = {
        'school': school, 'sex': sex, 'age': int(age), 'address': address, 'famsize': famsize,
        'Pstatus': Pstatus, 'Medu': int(Medu), 'Fedu': int(Fedu), 'Mjob': Mjob, 'Fjob': Fjob,
        'reason': reason, 'guardian': guardian, 'traveltime': int(traveltime), 'studytime': int(studytime),
        'failures': int(failures), 'schoolsup': schoolsup, 'famsup': famsup, 'paid': paid,
        'activities': activities, 'nursery': nursery, 'higher': higher, 'internet': internet,
        'romantic': romantic, 'famrel': int(famrel), 'freetime': int(freetime), 'goout': int(goout),
        'Dalc': int(Dalc), 'Walc': int(Walc), 'health': int(health), 'absences': int(absences),
        'G1': int(G1), 'G2': int(G2)
    }

    # Create DataFrame
    input_df = pd.DataFrame([input_data])

    # Combine and encode
    df_temp = pd.concat([df.drop('G3', axis=1), input_df], ignore_index=True)
```

```
    df_temp_encoded = pd.get_dummies(df_temp, drop_first=True)
    df_temp_encoded = df_temp_encoded.reindex(columns=df_encoded.drop('G3', axis=1).columns, fill_value=0)

    # Scale and predict
    scaled_input = scaler.transform(df_temp_encoded.tail(1))
    prediction = model.predict(scaled_input)

    return round(prediction[0], 2)
```

Create the Gradio Interface

```
inputs = [
    gr.Dropdown(['GP', 'MS'], label="School (GP=Gabriel Pereira, MS=Mousinho da Silveira)"),
    gr.Dropdown(['M', 'F'], label="Gender (M=Male, F=Female)"),
    gr.Number(label="Student Age"),
    gr.Dropdown(['U', 'R'], label="Residence Area (U=Urban, R=Rural)"),
    gr.Dropdown(['LE3', 'GT3'], label="Family Size (LE3=≤3, GT3=>3 members)"),
    gr.Dropdown(['A', 'T'], label="Parent Cohabitation Status (A=Apart, T=Together)"),
    gr.Number(label="Mother's Education Level (0-4)"),
    gr.Number(label="Father's Education Level (0-4)"),
    gr.Dropdown(['teacher', 'health', 'services', 'at_home', 'other'], label="Mother's Job"),
    gr.Dropdown(['teacher', 'health', 'services', 'at_home', 'other'], label="Father's Job"),
    gr.Dropdown(['home', 'reputation', 'course', 'other'], label="Reason for Choosing School"),
    gr.Dropdown(['mother', 'father', 'other'], label="Guardian"),
    gr.Number(label="Travel Time to School (1-4)"),
    gr.Number(label="Weekly Study Time (1-4)"),
    gr.Number(label="Past Class Failures (0-3)"),
    gr.Dropdown(['yes', 'no'], label="Extra School Support"),
    gr.Dropdown(['yes', 'no'], label="Family Support"),
    gr.Dropdown(['yes', 'no'], label="Extra Paid Classes"),
    gr.Dropdown(['yes', 'no'], label="Participates in Activities"),
    gr.Dropdown(['yes', 'no'], label="Attended Nursery"),
    gr.Dropdown(['yes', 'no'], label="Aspires Higher Education"),
    gr.Dropdown(['yes', 'no'], label="Internet Access at Home"),
    gr.Dropdown(['yes', 'no'], label="Currently in a Relationship"),
    gr.Number(label="Family Relationship Quality (1-5)"),
    gr.Number(label="Free Time After School (1-5)"),
    gr.Number(label="Going Out Frequency (1-5)"),
    gr.Number(label="Workday Alcohol Consumption (1-5)"),
```

```
        gr.Number(label="Weekend Alcohol Consumption (1-5)"),
        gr.Number(label="Health Status (1=Very Bad to 5=Excellent)"),
        gr.Number(label="Number of Absences"),
        gr.Number(label="Grade in 1st Period (G1: 0-20)"),
        gr.Number(label="Grade in 2nd Period (G2: 0-20)")
    ]

    output = gr.Number(label="🎯 Predicted Final Grade (G3)")

    # Launch the app
    gr.Interface(
        fn=predict_grade,
        inputs=inputs,
        outputs=output,
        title="🎓 Student Performance Predictor",
        description="Enter academic and demographic info to predict the final grade (G3) of a student.
    ).launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to wo k, sharing must be enabled. Automatically setting `share=True` (you can turn this

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://37518063c688a89403.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio d ploy` from the terminal in the working directory to deploy to Hugging Face Spaces
```