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.

Load the Dataset

```
import pandas as pd

# Replace with your file name if different
df = pd.read_csv('/content/sentimentdataset.csv')
df.head()
```

	Unnamed: 0.1	Unnamed: 0	Text	Sentiment	Timestamp	User	Platform	Hashtags	Retweets	Likes	Country	Year	Month	Day	Hour
0	0	0	Enjoying a beautiful day at the park! ...	Positive	2023-01-15 12:30:00	User123	Twitter	#Nature #Park	15.0	30.0	USA	2023	1	15	12
1	1	1	Traffic was terrible this morning. ...	Negative	2023-01-15 08:45:00	CommuterX	Twitter	#Traffic #Morning	5.0	10.0	Canada	2023	1	15	8

Just

Data Exploration

```
# Basic information
df.info()

# Summary statistics
df.describe()

# Column names
df.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 732 entries, 0 to 731
Data columns (total 15 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Unnamed: 0.1  732 non-null    int64
 1   Unnamed: 0    732 non-null    int64
 2   Text          732 non-null    object
 3   Sentiment     732 non-null    object
 4   Timestamp     732 non-null    object
 5   User          732 non-null    object
 6   Platform      732 non-null    object
 7   Hashtags      732 non-null    object
 8   Retweets      732 non-null    float64
 9   Likes         732 non-null    float64
 10  Country       732 non-null    object
 11  Year          732 non-null    int64
 12  Month         732 non-null    int64
 13  Day           732 non-null    int64
 14  Hour          732 non-null    int64
dtypes: float64(2), int64(6), object(7)
memory usage: 85.9+ KB
Index(['Unnamed: 0.1', 'Unnamed: 0', 'Text', 'Sentiment', 'Timestamp', 'User',
       'Platform', 'Hashtags', 'Retweets', 'Likes', 'Country', 'Year', 'Month',
       'Day', 'Hour'],
      dtype='object')
```

Check for Missing Values and Duplicates

```
# Missing values
print(df.isnull().sum())

# Duplicates
print(f"Duplicate rows: {df.duplicated().sum()}")
```

Visualize a Few Features

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

# Example visualization
sns.countplot(data=df, x='some_column')  # Replace with your actual column
plt.show()

# Correlation heatmap
sns.heatmap(df.corr(numeric_only=True), annot=True, cmap='coolwarm')
plt.show()
```

Identify Target and Features

```
# Replace 'target_column' with your actual target column
X = df.drop('target_column', axis=1)
y = df['target_column']
```

Convert Categorical Columns to Numerical

```
# Example: converting all object type columns
for col in X.select_dtypes(include=['object']).columns:
    X[col] = X[col].astype('category').cat.codes
```

One-Hot Encoding

```
X = pd.get_dummies(X, drop_first=True)
```

Feature Scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Train-Test Split

```
from sklearn.model_selection import train_test_split

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

Model Building

```
from sklearn.ensemble import RandomForestClassifier  # or Regressor based on task

model = RandomForestClassifier()  # Change if regression task
model.fit(X_train, y_train)
```

Evaluation

```
from sklearn.metrics import accuracy_score, classification_report

y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

Make Predictions from New Input

```
# Example input
new_data = [[value1, value2, ..., valueN]]  # match input features

# Scale and predict
new_data_scaled = scaler.transform(new_data)
prediction = model.predict(new_data_scaled)
print("Prediction:", prediction)
```

Convert to DataFrame and Encode

```
# Convert raw input to DataFrame and encode just like training
new_df = pd.DataFrame([input_dict])
new_df = pd.get_dummies(new_df)

# Align with training data
new_df = new_df.reindex(columns=X.columns, fill_value=0)
```

Predict the Final Grade (or label)

```
new_scaled = scaler.transform(new_df)
final_prediction = model.predict(new_scaled)
print("Final Grade Prediction:", final_prediction)
```

Deployment - Building an Interactive App

```
!pip install gradio
```

Create a Prediction Function

```
def predict_final_grade(input1, input2, ...):  # replace with actual feature names
    input_df = pd.DataFrame([[input1, input2, ...]], columns=X.columns)
    input_df = input_df.reindex(columns=X.columns, fill_value=0)
    input_scaled = scaler.transform(input_df)
    result = model.predict(input_scaled)
    return f"Predicted Result: {result[0]}"
```

Create the Gradio Interface

```
import gradio as gr

interface = gr.Interface(
    fn=predict_final_grade,
    inputs=[gr.Textbox(label=col) for col in X.columns],
    outputs="text",
    title="Final Grade Predictor"
)
interface.launch()
```