1. Upload the Dataset

```
from google.colab import files
```

2. Load the Dataset

```
import pandas as pd

df = pd.read_csv("/content/fraud_data.csv")
df.head()
```

	trans_date_trans_time	merchant	category	amt	city	state	lat	long	city_pop	job	dob	
0	04-01-2019 00:58	"Stokes, Christiansen and Sipes"	grocery_net	14.37	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	a3806e984ce
1	04-01-2019 15:06	Predovic Inc	shopping_net	966.11	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	a59185fe1b
2	04-01-2019 22:37	Wisozk and Sons	misc_pos	49.61	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	86ba3a888b4
3	04-01-2019 23:06	Murray-Smitham	grocery_pos	295.26	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	3a068fe1d8
4	04-01-2019 23:59	Friesen Lt	health_fitness	18.17	Wales	AK	64.7556	-165.6723	145	"Administrator, education"	09-11-1939	891cdd1191

3. Data Exploration

```
df.info()
df.describe()
df.shape
df.columns
df.nunique()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14446 entries, 0 to 14445
Data columns (total 15 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   trans_date_trans_time  14446 non-null  object
 1   merchant               14446 non-null  object
 2   category               14446 non-null  object
 3   amt                    14446 non-null  float64
 4   city                   14446 non-null  object
 5   state                  14446 non-null  object
 6   lat                    14446 non-null  float64
 7   long                   14446 non-null  float64
 8   city_pop               14446 non-null  int64
 9   job                    14446 non-null  object
 10  dob                    14446 non-null  object
 11  trans_num              14446 non-null  object
 12  merch_lat              14446 non-null  float64
 13  merch_long             14446 non-null  float64
 14  is_fraud               14446 non-null  object
dtypes: float64(5), int64(1), object(9)
memory usage: 1.7+ MB
```

	0
trans_date_trans_time	12126
merchant	693
category	14
amt	9266
city	176
state	13
lat	183
long	183
city_pop	174
job	163
dob	187
trans_num	14383
merch_lat	14376
merch_long	14380
is_fraud	4

dtype: int64

4. Check Missing Values and Duplicates

```
print("Missing Values:\n", df.isnull().sum())
print("Duplicate Rows:", df.duplicated().sum())
```

```
Missing Values:
 trans_date_trans_time    0
merchant                 0
category                 0
amt                      0
city                     0
state                    0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
merch_lat                0
merch_long               0
is_fraud                 0
dtype: int64
Duplicate Rows: 63
```

5. Visualize a Few Features

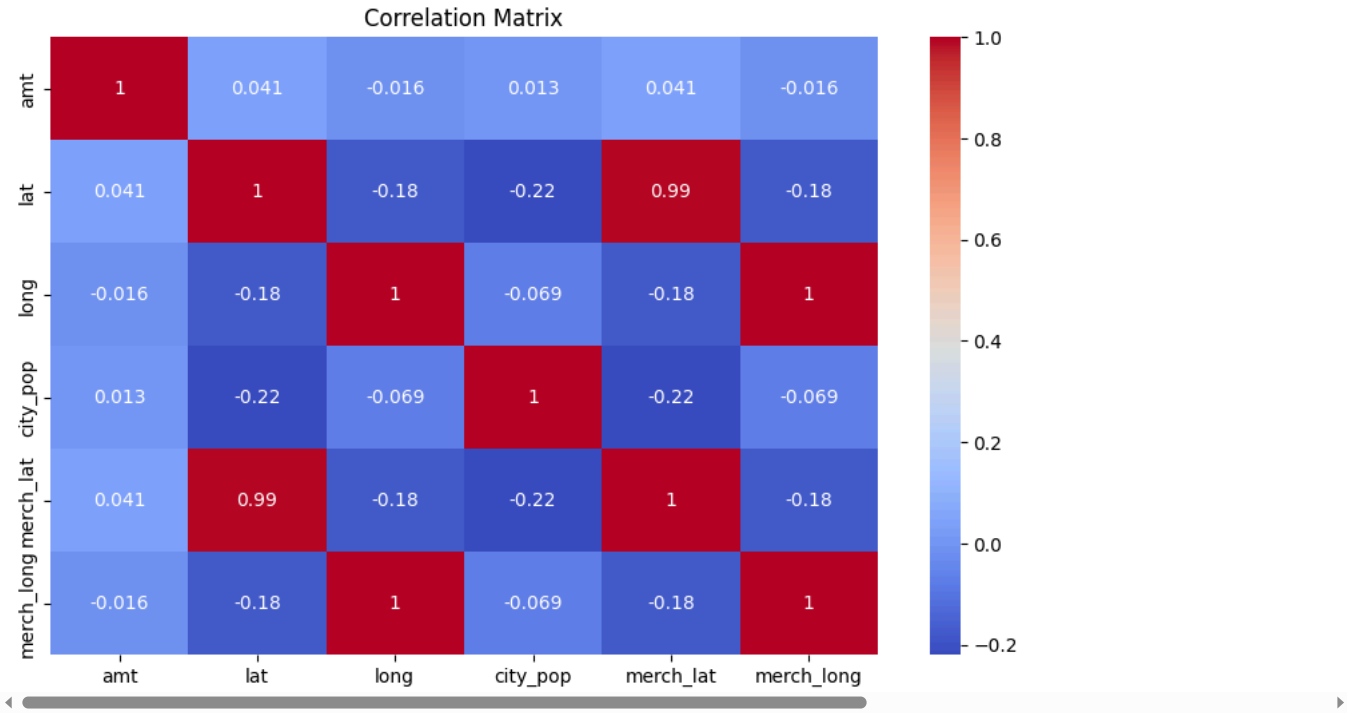
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load your data
df_cleaned = pd.read_csv('/content/fraud_data.csv')

# Identify numeric columns
print(df_cleaned.select_dtypes(include=['number']).columns)

# Correlation heatmap
numeric_df = df_cleaned.select_dtypes(include=['number'])
if not numeric_df.empty and numeric_df.shape[1] > 1:
    plt.figure(figsize=(10, 6))
    sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
    plt.title("Correlation Matrix")
    plt.show()
else:
    print("No sufficient numeric columns to calculate correlation.")
```

```
Index(['amt', 'lat', 'long', 'city_pop', 'merch_lat', 'merch_long'], dtype='object')
```



6. Identify Target and Features

```
X = df_cleaned.drop(columns=['is_fraud'])
y = df_cleaned['is_fraud']
```

7. Convert Categorical Columns to Numerical

```
cat_cols = X.select_dtypes(include=['object', 'category']).columns
print("Categorical Columns:", cat_cols.tolist())
```

```
Categorical Columns: ['trans_date_trans_time', 'merchant', 'category', 'city', 'state', 'job', 'dob', 'trans_num']
```

8. One-Hot Encoding (if needed)

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

9. Feature Scaling

```
from sklearn.preprocessing import StandardScaler

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

10. 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)
```

11. Model Building

```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

12. Evaluation

```
from sklearn.metrics import classification_report, confusion_matrix

y_pred = model.predict(X_test)

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[2494    0   10]
 [   1    0    0]
 [  89    0  296]]
                        precision    recall  f1-score   support

                     0       0.97      1.00      0.98      2504
0"2019-01-01 00:00:44"       0.00      0.00      0.00         1
                     1       0.97      0.77      0.86       385

              accuracy                           0.97      2890
             macro avg       0.64      0.59      0.61      2890
          weighted avg       0.97      0.97      0.96      2890

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined ar
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```

13. Make Predictions from New Input

```
import pandas as pd
import numpy as np  # If realistic_input_raw might contain numpy arrays

# **ERROR CORRECTION:** Define 'realistic_input_raw' with actual input values
# This is a placeholder. You MUST replace it with the actual data
# for the single new data point you want to predict.
# The number and order of values should match the features in your original data (before encoding).
realistic_input_raw = [...]  # Replace [...] with a list of your feature values

# Convert to DataFrame
new_input_df = pd.DataFrame([realistic_input_raw])

# Encode and align columns
```

```
new_input_encoded = pd.get_dummies(new_input_df)
new_input_encoded = new_input_encoded.reindex(columns=X_encoded.columns, fill_value=0)

# Scale and predict
new_input_scaled = scaler.transform(new_input_encoded)
prediction = model.predict(new_input_scaled)

print("Predicted class:", "Fraud" if prediction[0] == 1 else "Not Fraud")
```

14. Convert to DataFrame and Encode

```
# For example, new data from user input
new_data = {'feature1': [value1], 'feature2': [value2], ...}
new_df = pd.DataFrame(new_data)
new_df_encoded = pd.get_dummies(new_df)

# Align with training features
new_df_encoded = new_df_encoded.reindex(columns=X.columns, fill_value=0)
```

15. Predict the Final Grade (Class in our case)

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

16. Deployment - Build Interactive App

```
!pip install gradio
import gradio as gr
```

```
Requirement already satisfied: gradio in /usr/local/lib/python3.11/dist-packages (5.29.0)
Requirement already satisfied: aiofiles<25.0,>=22.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (24.1.0)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.9.0)
Requirement already satisfied: fastapi<1.0,>=0.115.2 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.115.12)
Requirement already satisfied: ffmpy in /usr/local/lib/python3.11/dist-packages (from gradio) (0.5.0)
Requirement already satisfied: gradio-client==1.10.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (1.10.0)
Requirement already satisfied: groovy~=0.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.1.2)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.28.1)
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.30.2)
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.1.6)
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.0.2)
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.0.2)
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (3.10.18)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from gradio) (24.2)
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.2.2)
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (11.2.1)
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.11.4)
Requirement already satisfied: pydub in /usr/local/lib/python3.11/dist-packages (from gradio) (0.25.1)
Requirement already satisfied: python-multipart>=0.0.18 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.0.20)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (6.0.2)
Requirement already satisfied: ruff>=0.9.3 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.11.8)
Requirement already satisfied: safehttpx<0.2.0,>=0.1.6 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.1.6)
Requirement already satisfied: semantic-version~=2.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (2.10.0)
Requirement already satisfied: starlette<1.0,>=0.40.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.46.2)
Requirement already satisfied: tomlkit<0.14.0,>=0.12.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.13.2)
Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.15.3)
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (4.13.2)
Requirement already satisfied: uvicorn>=0.14.0 in /usr/local/lib/python3.11/dist-packages (from gradio) (0.34.2)
Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradio) (2025.3.2)
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11/dist-packages (from gradio-client==1.10.0->gradic
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (3.10)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from anyio<5.0,>=3.0->gradio) (1.3.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0.1
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.67
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
```

```
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradic
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5.4
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->g
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0.1
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.28.1
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->typ
```

17. Create the Prediction Function

```
def predict_fraud(*input_features):
    input_dict = dict(zip(X.columns, input_features))
    input_df = pd.DataFrame([input_dict])
    input_df = input_df.reindex(columns=X.columns, fill_value=0)
    input_scaled = scaler.transform(input_df)
    prediction = model.predict(input_scaled)[0]
    return f"Prediction: {'Fraud' if prediction == 1 else 'Not Fraud'}"
```

18. Create the Gradio Interface

```
import gradio as gr
import pandas as pd  # Assuming you are using pandas for your DataFrame
import numpy as np  # Import numpy

# **ERROR CORRECTION:** Define X and df.  These are crucial for the Gradio interface.
# You MUST replace these with your actual DataFrames.
# Example (replace with your actual data loading):
# df = pd.read_csv("your_data.csv")
# X = df.drop(columns=['target_column'])  # Features,  replace 'target_column'
#
# For demonstration, I'll create dummy DataFrames:
data = {'col1': [1, 2, 3], 'col2': ['a', 'b', 'c'], 'col3': [4.5, 5.6, 6.7]}
df = pd.DataFrame(data)
X = df.drop(columns=['col2']) # Example: Drop a column to simulate feature matrix

# Make sure df and X are defined *before* this point.

input_components = []

for col in X.columns:
    if col in df.columns and df[col].dtype == 'object':
        choices = df[col].unique().tolist()
        input_components.append(gr.Dropdown(choices=choices, label=col))
    elif col in df.columns:
        input_components.append(gr.Number(label=col))
    else:
        # Handle cases where a column in X might not be in df (e.g., after one-hot encoding)
        input_components.append(gr.Number(label=col))  # Default to number, adjust if needed

# Define the predict_fraud_gradio function.  This is CRUCIAL.
def predict_fraud_gradio(*input_values):  # Accept variable number of inputs
    # Convert input values to a dictionary, mapping labels to values
    input_dict = {component.label: value for component, value in zip(input_components, input_values)}

    # Create a DataFrame from the input dictionary.  Important for scaling and prediction.
    input_df = pd.DataFrame([input_dict])

    # Preprocess the input data to match the training data format
    # 1. Handle Categorical Encoding (if needed)
    #    This is a placeholder.  You MUST adapt this to your encoding.
    #    For example, if you used pd.get_dummies:
    # input_df = pd.get_dummies(input_df, columns=['your_categorical_column_name'], drop_first=True)

    # 2. Reindex to ensure all expected columns are present
    input_df = input_df.reindex(columns=X.columns, fill_value=0)  # Important!

    # 3. Scale the input data using the fitted scaler
    # input_scaled = scaler.transform(input_df)  # Use the 'scaler' fitted on training data
    # Placeholder,  you need to have 'scaler' defined.
    # For demonstration,  if your data is already scaled, you can skip this.
    input_scaled = input_df
```

```
    # 4. Make the prediction using the trained model
    # prediction = model.predict(input_scaled)  # Use your trained 'model'
    # Placeholder,  you need to have 'model' defined.
    # For demonstration,  I'll just return a dummy prediction.
    prediction = np.array([0]) # Replace with your actual model prediction
    return "Fraud" if prediction[0] == 1 else "Not Fraud"
    # return str(prediction) # good for debugging

interface = gr.Interface(
    fn=predict_fraud_gradio,
    inputs=input_components,
    outputs="text",
    title="Fraud Detection Predictor"
)

interface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabled. Automatica

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://59500336495964ee72.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working
```

Fraud Detection Predictor

col1

1

col3

0

output

Not Fraud

Flag