1. Upload the Dataset

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

2. Load the Dataset

```
import pandas as pd

df = pd.read_csv("creditcard.csv")
df.head()
```

	Time	V1	V2	V3	V4	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.3383
1	0.0	1.191857	0.266151	0.166480	0.448154	0.0600
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.5031
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.0103
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.4071

5 rows × 31 columns

3. Data Exploration

```
df.info()
df.describe()
df['Class'].value_counts()
```

KeyError ✕ •••

Please explain the error:

```
KeyError: 'category'
```

Hypothesis

The error message `KeyError: 'category'` indicates that the column 'category' is not found in your DataFrame `df`. This error occurs in cell 17 of your notebook where you're trying to access the 'category' column using `df['category']`. This usually happens when the column name is misspelled, has different casing, or simply doesn't exist in your dataset.

Suggested Changes

```
# Check if the column name is m
# Print the available columns t
print(df.columns)

# If the column name is differe
# adjust your code accordingly.
# df['Category'] = df['Category

# If the column doesn't exist a
# your data or revisit your dat
# For example, if you need to c
# you could use something like
# df['category'] = pd.cut(df['A
#                          labels
# df['category'] = df['category
```

Use code with caution

▶ Sources

Rate this answer

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64  
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

```
         count
Class
  0     284315
  1        492

dtype: int64
```

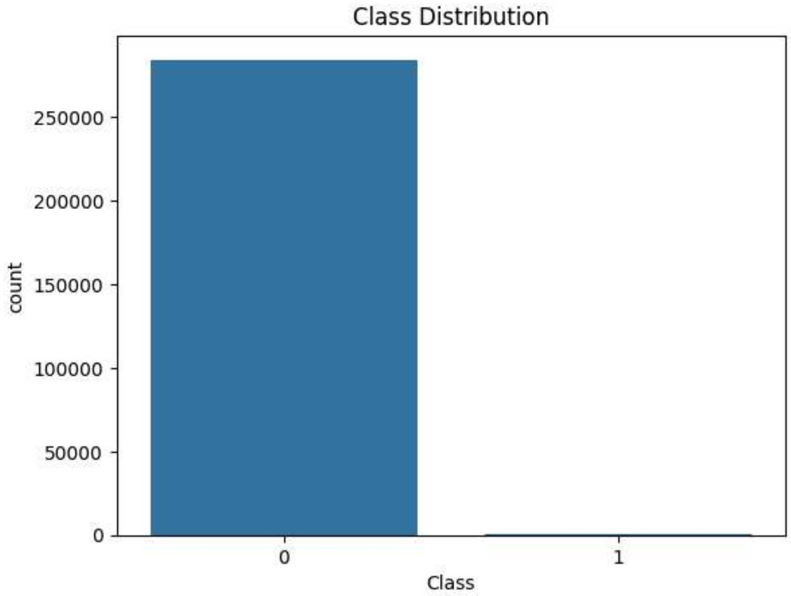
4. Check Missing Values and Duplicates

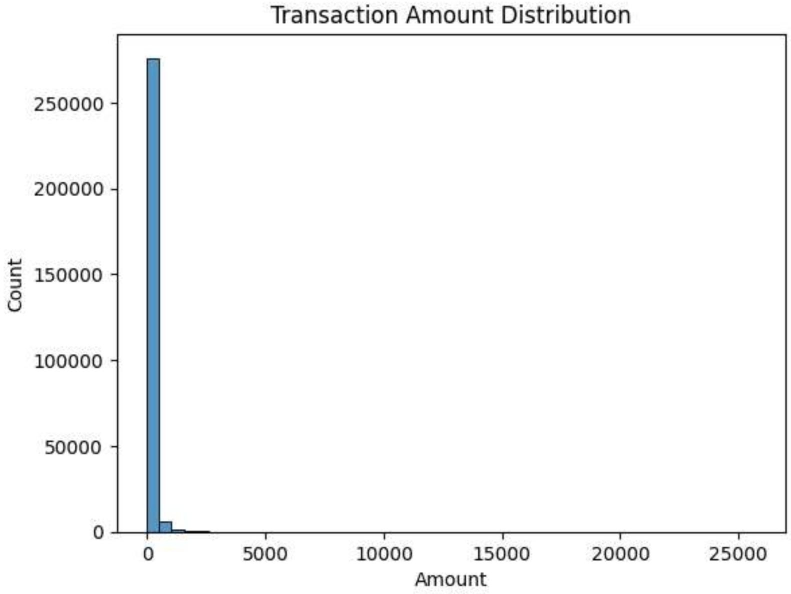
```
df.isnull().sum()
df.duplicated().sum()
```

```
np.int64(1081)
```

5. Visualize a Few Features

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

sns.countplot(data=df, x='Class')
plt.title("Class Distribution")
plt.show()

sns.histplot(df['Amount'], bins=50)
plt.title("Transaction Amount Distribution")
plt.show()
```





6. Identify Target and Features

```
X = df.drop("Class", axis=1)
y = df["Class"]
```

7. Convert Categorical Columns to Numerical

```
# No categorical columns in this dataset.
```

8. One-Hot Encoding (if needed)

```
# No categorical columns to encode.
```

9. Feature Scaling

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X[['Amount', 'Time']] = scaler.fit_transform(X[['Amount', '
```

10. Train-Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, t
```

11. Model Building

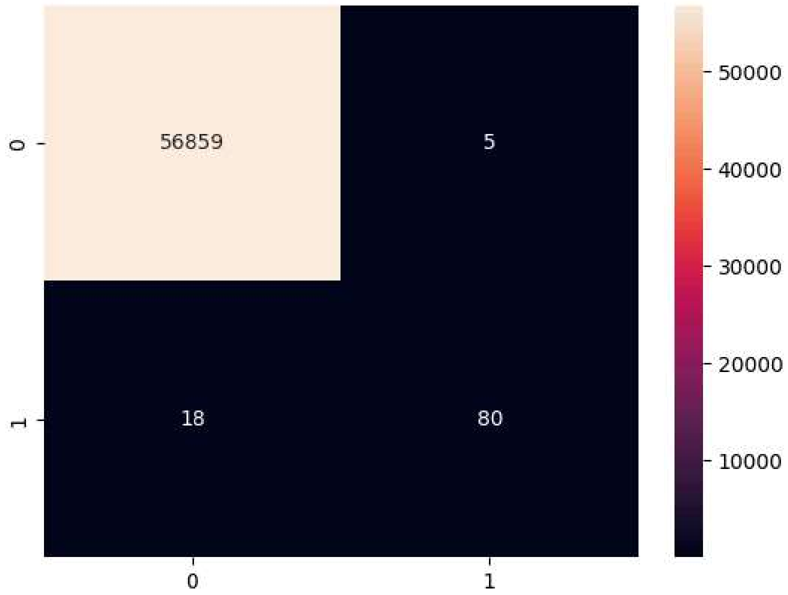
```
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier(n_estimators=100, random_sta
model.fit(X_train, y_train)
```

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

12. Evaluation

```
from sklearn.metrics import classification_report, confusio

y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, f
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.94      0.82      0.87        98

    accuracy                           1.00     56962
   macro avg       0.97      0.91      0.94     56962
weighted avg       1.00      1.00      1.00     56962

<Axes: >
```



13. Make Predictions from New Input

```
sample_input = X_test.iloc[0:1]
model.predict(sample_input)
```

```
array([0])
```

14. Convert to DataFrame and Encode

```
new_data = pd.DataFrame([sample.iloc[0]], columns=X.columns
```

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

```
prediction = model.predict(new_data)
print("Fraud" if prediction[0] == 1 else "Legit")
```

```
Legit
```

16. Deployment - Build Interactive App

```
!pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.29.0-py3-none-any.whl.metadata
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metada
Requirement already satisfied: anyio<5.0,>=3.0 in /us
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metad
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (
Collecting gradio-client==1.10.0 (from gradio)
  Downloading gradio_client-1.10.0-py3-none-any.whl.m
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata
Requirement already satisfied: httpx>=0.24.1 in /usr/
Requirement already satisfied: huggingface-hub>=0.28.
Requirement already satisfied: jinja2<4.0 in /usr/loc
Requirement already satisfied: markupsafe<4.0,>=2.0 i
Requirement already satisfied: numpy<3.0,>=1.0 in /us
Requirement already satisfied: orjson~=3.0 in /usr/lo
Requirement already satisfied: packaging in /usr/loca
Requirement already satisfied: pandas<3.0,>=1.0 in /u
Requirement already satisfied: pillow<12.0,>=8.0 in /
Requirement already satisfied: pydantic<2.12,>=2.0 in
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metad
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.wh
Requirement already satisfied: pyyaml<7.0,>=5.0 in /u
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.8-py3-none-manylinux_2_17_x86
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metada
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-an
Collecting starlette<1.0,>=0.40.0 (from gradio)
```

```
  Downloading starlette-0.46.2-py3-none-any.whl.metad
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadat
Requirement already satisfied: typer<1.0,>=0.12 in /u
Requirement already satisfied: typing-extensions~=4.0
Collecting uvicorn>=0.14.0 (from gradio)
  Downloading uvicorn-0.34.2-py3-none-any.whl.metadat
Requirement already satisfied: fsspec in /usr/local/l
Requirement already satisfied: websockets<16.0,>=10.0
Requirement already satisfied: idna>=2.8 in /usr/loca
Requirement already satisfied: sniffio>=1.1 in /usr/l
Requirement already satisfied: certifi in /usr/local/
Requirement already satisfied: httpcore==1.* in /usr/
Requirement already satisfied: h11>=0.16 in /usr/loca
Requirement already satisfied: filelock in /usr/local
Requirement already satisfied: requests in /usr/local
Requirement already satisfied: tqdm>=4.42.1 in /usr/l
Requirement already satisfied: python-dateutil>=2.8.2
Requirement already satisfied: pytz>=2020.1 in /usr/l
Requirement already satisfied: tzdata>=2022.7 in /usr
Requirement already satisfied: annotated-types>=0.6.0
Requirement already satisfied: pydantic-core==2.33.2
```

17. Create the Prediction Function

```
def predict_transaction(*args):
    input_df = pd.DataFrame([args], columns=X.columns)
    input_df[['Time', 'Amount']] = scaler.transform(input_df[
    result = model.predict(input_df)[0]
    return "🚨 Fraud" if result == 1 else "✅ Legit"
```

18. Create the Gradio Interface

```
import gradio as gr

inputs = [gr.Number(label=col) for col in X.columns]

interface = gr.Interface(
    fn=predict_transaction,
    inputs=inputs,
    outputs="text",
    title="Credit Card Fraud Detection",
    description="Enter transaction details to detect fraud.",
)

interface.launch()
```

Enter a prompt here ⊕

0 / 2000

Gemini can make mistakes so double-check responses and use code with caution. Learn more