Generate randomly select 5 items from a list Close

```
1. Upload the Dataset


from google.colab import files
uploaded = files.upload()
```

Choose Files creditcard.csv

- **creditcard.csv**(text/csv) - 46221802 bytes, last modified: 9/20/2019 - 100% done

```
Saving creditcard.csv to creditcard.csv
```

2. Load the Dataset

```
import pandas as pd

# Load the dataset
df = pd.read_csv('creditcard.csv')

# Display basic information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87399 entries, 0 to 87398
Data columns (total 31 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Time    87399 non-null  int64
 1   V1      87399 non-null  float64
 2   V2      87399 non-null  float64
 3   V3      87399 non-null  float64
 4   V4      87399 non-null  float64
 5   V5      87399 non-null  float64
 6   V6      87399 non-null  float64
 7   V7      87399 non-null  float64
 8   V8      87399 non-null  float64
 9   V9      87399 non-null  float64
 10  V10     87399 non-null  float64
 11  V11     87399 non-null  float64
 12  V12     87399 non-null  float64
 13  V13     87399 non-null  float64
 14  V14     87399 non-null  float64
 15  V15     87399 non-null  float64
 16  V16     87399 non-null  float64
 17  V17     87398 non-null  float64
 18  V18     87398 non-null  float64
 19  V19     87398 non-null  float64
 20  V20     87398 non-null  float64
 21  V21     87398 non-null  float64
 22  V22     87398 non-null  float64
 23  V23     87398 non-null  float64
 24  V24     87398 non-null  float64
```

```
 25  V25     87398 non-null  float64
 26  V26     87398 non-null  float64
 27  V27     87398 non-null  float64
 28  V28     87398 non-null  float64
 29  Amount  87398 non-null  float64
 30  Class   87398 non-null  float64
dtypes: float64(30), int64(1)
memory usage: 20.7 MB
```

3. Data Exploration

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

# Statistical summary
df.describe()
```

	Time	V1	V2	V3	V4	
count	87399.000000	87399.000000	87399.000000	87399.000000	87399.000000	87399.0000
mean	39249.095470	-0.264160	-0.039340	0.679331	0.162933	-0.2775
std	15860.727322	1.876285	1.666534	1.358770	1.360733	1.3714
min	0.000000	-56.407510	-72.715728	-33.680984	-5.172595	-42.1478
25%	31895.500000	-1.027975	-0.602547	0.184402	-0.720231	-0.8969
50%	41618.000000	-0.260941	0.070761	0.761799	0.186068	-0.3124
75%	51639.000000	1.152778	0.726532	1.388911	1.039671	0.2563
max	61670.000000	1.960497	18.902453	4.226108	16.715537	34.8016

8 rows × 31 columns

4. Check for Missing Values and Duplicates

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

# Check for duplicates
df.duplicated().sum()
```

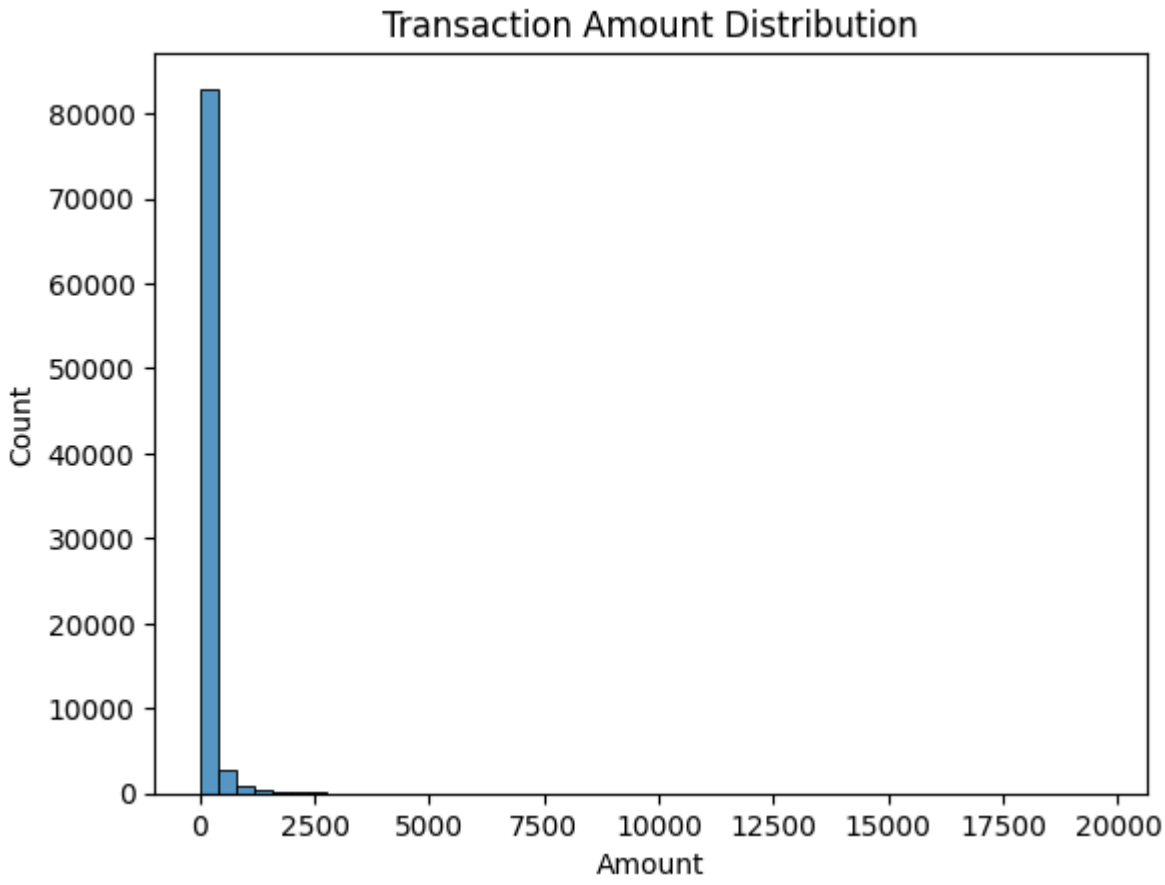
```
np.int64(319)
```

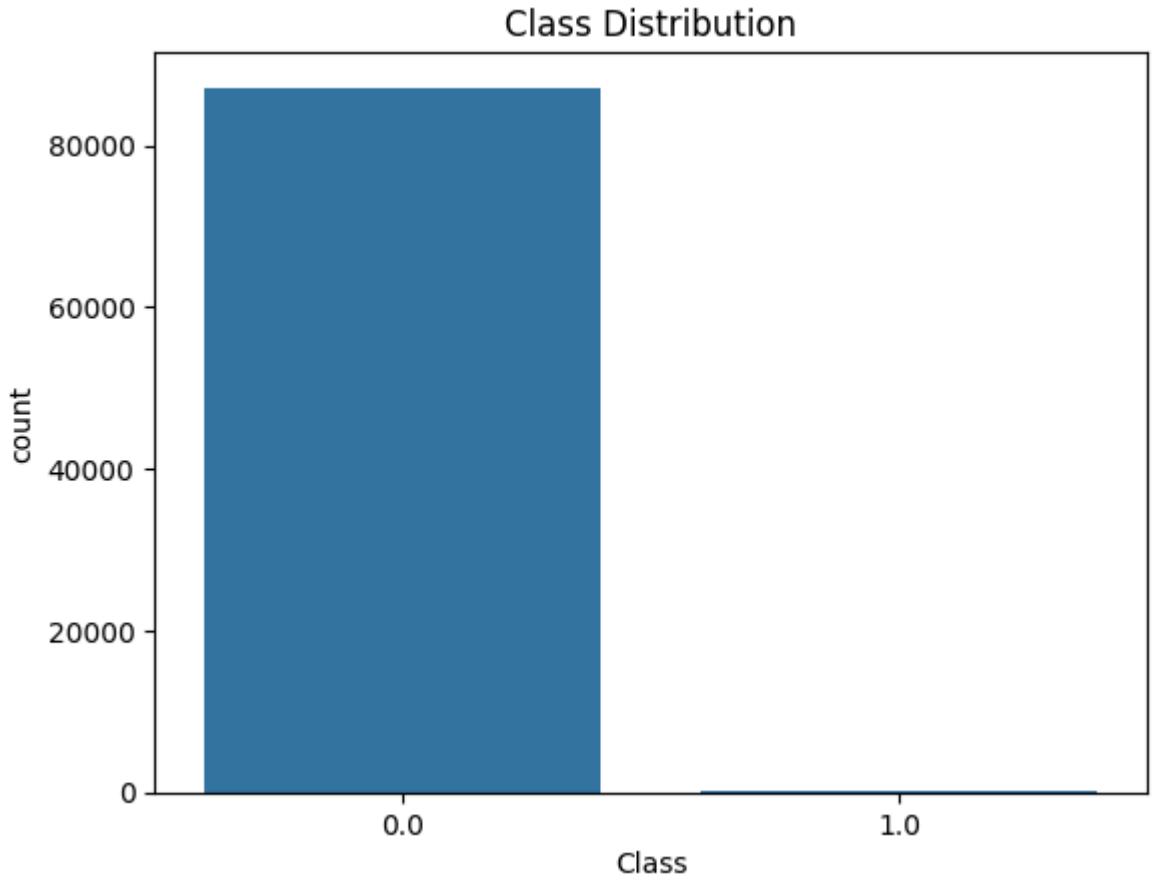
5. Visualize a Few Features

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

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

# Count of fraud vs. non-fraud
sns.countplot(x='Class', data=df)
plt.title('Class Distribution')
plt.show()
```





6. Identify Target and Features

```
# Features and target
X = df.drop('Class', axis=1)
```

```
y = df['Class']
```

7.Feature Scaling

```
from sklearn.preprocessing import StandardScaler

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

8.Train-Test Split


```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Assuming 'df' is your DataFrame
# ... (Your previous code to load and preprocess the data) ...

# Features and target
X = df.drop('Class', axis=1)
y = df['Class']


# Check for NaN values in 'y' and handle them
# Remove rows with NaN in 'y'
X = X[~y.isnull()]  # Keep rows where 'y' is not NaN in the features
y = y[~y.isnull()]
```

9.Model Building

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Assuming 'X' and 'y' are your features and target
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train) # Now X_train and y_train are defined
```

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

10. Evaluation

```
from sklearn.metrics import classification_report, confusion_matrix

# Make predictions
y_pred = model.predict(X_test)

# Display evaluation metrics
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[17437     0]
 [    9    34]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     17437
         1.0       1.00      0.79      0.88        43

    accuracy                           1.00     17480
   macro avg       1.00      0.90      0.94     17480
weighted avg       1.00      1.00      1.00     17480
```

11. Make Predictions from New Input

Double-click (or enter) to edit

```
import numpy as np

# Example: Predicting a single transaction
new_transaction = np.array([X_test.iloc[0]])  # Use iloc to access the first row by posit
prediction = model.predict(new_transaction)
print("Fraudulent" if prediction[0] == 1 else "Not Fraudulent")
```

```
Not Fraudulent
/usr/local/lib/python3.11/dist-packages/sklearn/utils/validation.py:2739: UserWarning
  warnings.warn(
```

12. Deployment - Building an Interactive App

```
!pip install gradio
import gradio as gr

def predict_fraud(*inputs):
    input_array = np.array(inputs).reshape(1, -1)
    input_scaled = scaler.transform(input_array)
    prediction = model.predict(input_scaled)
```

```
    return "Fraudulent" if prediction[0] == 1 else "Not Fraudulent"
```

```
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-
Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.11
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 (
Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.11
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-package
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/
```

Double-click (or enter) to edit

13. Create a Prediction Function

```
def predict_fraud(*inputs):
    input_array = np.array(inputs).reshape(1, -1)
    input_scaled = scaler.transform(input_array)
    prediction = model.predict(input_scaled)
    return "Fraudulent" if prediction[0] == 1 else "Not Fraudulent"
```

14.Create the Gradio Interface

```
# Assuming the dataset has 30 features excluding 'Class'
feature_names = df.drop('Class', axis=1).columns.tolist()

iface = gr.Interface(
    fn=predict_fraud,
    inputs=[gr.Number(label=feature) for feature in feature_names],
    outputs="text",
    title="Credit Card Fraud Detection"
)

iface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio a

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://a207ad6690659bdaac.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `
```