🛡️ Credit Card Fraud Detection (2023, Europe)

🔍 Project Overview

This project builds a machine learning pipeline to detect fraudulent credit card transactions using anonymized financial data collected in Europe in 2023. The dataset has been balanced to enable fair model training and evaluation.

Developer: Agba Daniel
Role: AI/ML Practitioner | Data Scientist
Certifications:

- Machine Learning Specialization (Coursera)
- Deep Learning Specialization (Coursera)

🎯 Objective

To design and evaluate a robust classification pipeline that can accurately detect fraud in real-time credit card transactions.

✅ Key Goals

- Build a complete supervised ML pipeline
- Explore and visualize class distributions
- Apply data preprocessing, scaling, and feature analysis
- Train and evaluate multiple classifiers
- Use accuracy and classification metrics for evaluation

🛠️ Tools & Libraries

- Python, Pandas, NumPy
- Scikit-learn, XGBoost, LightGBM
- Matplotlib, Seaborn
- Jupyter Notebook

📊 Model Performance (Validation Set)

Model	Accuracy
Decision Tree	99.72%

Model	Accuracy
Random Forest	99.99%
Gradient Boosting	97.99%
XGBoost	99.97%
LightGBM	99.92%

Note: Balanced dataset — accuracy is meaningful here.

🧠 Skills Demonstrated

- ✔ Exploratory Data Analysis (EDA)
- ✔ Data Cleaning & Preprocessing
- ✔ Feature Scaling & Engineering
- ✔ Supervised Classification
- ✔ Model Selection & Evaluation
- ✔ Git & GitHub Workflow

👨‍🦲 About Me

Agba Daniel

AI/ML Specialist with over a year of hands-on experience in developing classification and regression models, with a focus on real-world applications and clean pipelines.

📧 agbadaniel13@gmail.com
📱 WhatsApp: +2347080157838
🌐 GitHub: github.com/agbadaniel13

📂 Project Folder Structure

In [49]:
```
# 1. Data Loading
import pandas as pd

# Load the dataset (update the path as necessary)
df = pd.read_csv("creditcard_2023.csv")
df.head()
```

Out[49]:

	id	V1	V2	V3	V4	V5	V6	V7	V8	
0	0	-0.260648	-0.469648	2.496266	-0.083724	0.129681	0.732898	0.519014	-0.130006	
1	1	0.985100	-0.356045	0.558056	-0.429654	0.277140	0.428605	0.406466	-0.133118	
2	2	-0.260272	-0.949385	1.728538	-0.457986	0.074062	1.419481	0.743511	-0.095576	
3	3	-0.152152	-0.508959	1.746840	-1.090178	0.249486	1.143312	0.518269	-0.065130	
4	4	-0.206820	-0.165280	1.527053	-0.448293	0.106125	0.530549	0.658849	-0.212660	

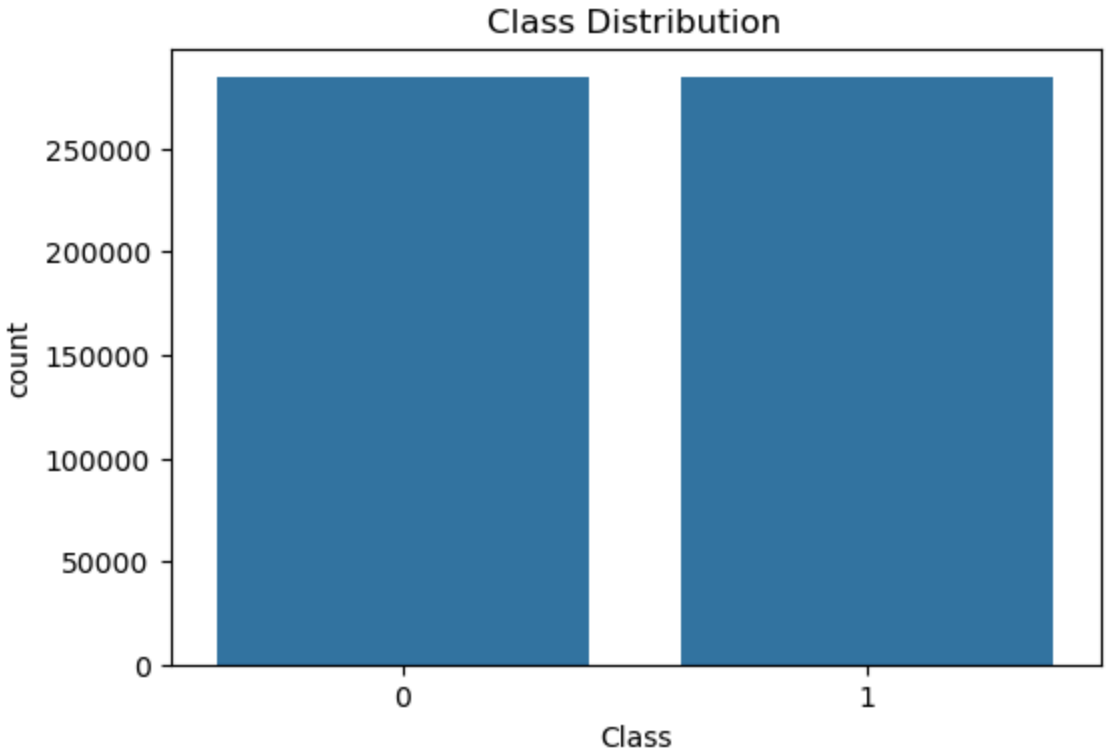
5 rows × 31 columns

In [51]:
```
# 2. Exploratory Data Analysis (EDA)
print("\nDataset Info:")
df.info()

print("\nMissing Values:")
print(df.isnull().sum())

print("\nDescriptive Statistics:")
print(df.describe())

print("\nClass Distribution:")
print(df['Class'].value_counts())

import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(6,4))
sns.countplot(data=df, x='Class')
plt.title("Class Distribution")
plt.show()
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568630 entries, 0 to 568629
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   id      568630 non-null  int64  
 1   V1      568630 non-null  float64
 2   V2      568630 non-null  float64
 3   V3      568630 non-null  float64
 4   V4      568630 non-null  float64
 5   V5      568630 non-null  float64
 6   V6      568630 non-null  float64
 7   V7      568630 non-null  float64
 8   V8      568630 non-null  float64
 9   V9      568630 non-null  float64
 10  V10     568630 non-null  float64
 11  V11     568630 non-null  float64
 12  V12     568630 non-null  float64
 13  V13     568630 non-null  float64
 14  V14     568630 non-null  float64
 15  V15     568630 non-null  float64
 16  V16     568630 non-null  float64
 17  V17     568630 non-null  float64
 18  V18     568630 non-null  float64
 19  V19     568630 non-null  float64
 20  V20     568630 non-null  float64
 21  V21     568630 non-null  float64
 22  V22     568630 non-null  float64
 23  V23     568630 non-null  float64
 24  V24     568630 non-null  float64
 25  V25     568630 non-null  float64
 26  V26     568630 non-null  float64
 27  V27     568630 non-null  float64
 28  V28     568630 non-null  float64
 29  Amount  568630 non-null  float64
 30  Class   568630 non-null  int64  
dtypes: float64(29), int64(2)
memory usage: 134.5 MB

Missing Values:
id        0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
```

```
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64

Descriptive Statistics:
                  id            V1            V2            V3            V4  \
count  568630.000000  5.686300e+05  5.686300e+05  5.686300e+05  5.686300e+05
mean   284314.500000 -5.638058e-17 -1.319545e-16 -3.518788e-17 -2.879008e-17
std    164149.486121  1.000001e+00  1.000001e+00  1.000001e+00  1.000001e+00
min         0.000000 -3.495584e+00 -4.996657e+01 -3.183760e+00 -4.951222e+00
25%    142157.250000 -5.652859e-01 -4.866777e-01 -6.492987e-01 -6.560203e-01
50%    284314.500000 -9.363846e-02 -1.358939e-01  3.528579e-04 -7.376152e-02
75%    426471.750000  8.326582e-01  3.435552e-01  6.285380e-01  7.070047e-01
max    568629.000000  2.229046e+00  4.361865e+00  1.412583e+01  3.201536e+00

                 V5            V6            V7            V8            V9  \
count  5.686300e+05  5.686300e+05  5.686300e+05  5.686300e+05  5.686300e+05
mean   7.997245e-18 -3.958636e-17 -3.198898e-17  2.109273e-17  3.998623e-17
std    1.000001e+00  1.000001e+00  1.000001e+00  1.000001e+00  1.000001e+00
min   -9.952786e+00 -2.111111e+01 -4.351839e+00 -1.075634e+01 -3.751919e+00
25%   -2.934955e-01 -4.458712e-01 -2.835329e-01 -1.922572e-01 -5.687446e-01
50%    8.108788e-02  7.871758e-02  2.333659e-01 -1.145242e-01  9.252647e-02
75%    4.397368e-01  4.977881e-01  5.259548e-01  4.729905e-02  5.592621e-01
max    4.271689e+01  2.616840e+01  2.178730e+02  5.958040e+00  2.027006e+01

       ...           V21           V22           V23           V24  \
count  ...  5.686300e+05  5.686300e+05  5.686300e+05  5.686300e+05
mean   ...  4.758361e-17  3.948640e-18  6.194741e-18 -2.799036e-18
std    ...  1.000001e+00  1.000001e+00  1.000001e+00  1.000001e+00
min    ... -1.938252e+01 -7.734798e+00 -3.029545e+01 -4.067968e+00
25%    ... -1.664408e-01 -4.904892e-01 -2.376289e-01 -6.515801e-01
50%    ... -3.743065e-02 -2.732881e-02 -5.968903e-02  1.590123e-02
75%    ...  1.479787e-01  4.638817e-01  1.557153e-01  7.007374e-01
max    ...  8.087080e+00  1.263251e+01  3.170763e+01  1.296564e+01

                V25           V26           V27           V28         Amount  \
count  5.686300e+05  5.686300e+05  5.686300e+05  5.686300e+05  568630.000000
mean  -3.178905e-17 -7.497417e-18 -3.598760e-17  2.609101e-17   12041.957635
std    1.000001e+00  1.000001e+00  1.000001e+00  1.000001e+00    6919.644449
min   -1.361263e+01 -8.226969e+00 -1.049863e+01 -3.903524e+01      50.010000
25%   -5.541485e-01 -6.318948e-01 -3.049607e-01 -2.318783e-01    6054.892500
50%   -8.193162e-03 -1.189208e-02 -1.729111e-01 -1.392973e-02   12030.150000
```

```
75%    5.500147e-01  6.728879e-01  3.340230e-01  4.095903e-01   18036.330000
max    1.462151e+01  5.623285e+00  1.132311e+02  7.725594e+01   24039.930000

          Class
count  568630.0
mean        0.5
std         0.5
min         0.0
25%         0.0
50%         0.5
75%         1.0
max         1.0

[8 rows x 31 columns]

Class Distribution:
Class
0    284315
1    284315
Name: count, dtype: int64
```



In [52]:
```python
# 3. Data Cleaning
# Drop duplicates if any
df = df.drop_duplicates()

# Fill or drop missing values if present (none in this case, based on EDA)
# Example: df = df.dropna()

# 4. Split Data
from sklearn.model_selection import train_test_split

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

```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random_sta
X_dev, X_test, y_dev, y_test = train_test_split(X_temp, y_temp, test_size=0.5, rand

print("Train shape:", X_train.shape)
print("Dev shape:", X_dev.shape)
print("Test shape:", X_test.shape)
```

```
Train shape: (341178, 30)
Dev shape: (113726, 30)
Test shape: (113726, 30)
```

In [57]:

```
# 6. Model Training
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import classification_report, accuracy_score

# Optional: Use XGBoost and LightGBM if installed
try:
    from xgboost import XGBClassifier
    xgb_installed = True
except ImportError:
    xgb_installed = False

try:
    from lightgbm import LGBMClassifier
    lgbm_installed = True
except ImportError:
    lgbm_installed = False

models = {
    "Decision Tree": DecisionTreeClassifier(random_state=42),
    "Random Forest": RandomForestClassifier(random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(random_state=42)
}

if xgb_installed:
    models["XGBoost"] = XGBClassifier(use_label_encoder=False, eval_metric='logloss
if lgbm_installed:
    models["LightGBM"] = LGBMClassifier(random_state=42)

# 7. Evaluation
for name, model in models.items():
    print(f"\nTraining {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_dev)
    acc = accuracy_score(y_dev, y_pred)
    print(f"Accuracy on Dev Set: {acc:.4f}")
    print(classification_report(y_dev, y_pred))
```

```
Training Decision Tree...
Accuracy on Dev Set: 0.9996
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56863
           1       1.00      1.00      1.00     56863

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726


Training Random Forest...
Accuracy on Dev Set: 0.9999
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56863
           1       1.00      1.00      1.00     56863

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726


Training Gradient Boosting...
Accuracy on Dev Set: 0.9998
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56863
           1       1.00      1.00      1.00     56863

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726


Training XGBoost...
C:\Users\PC\anaconda3\Lib\site-packages\xgboost\training.py:183: UserWarning: [01:1
0:28] WARNING: C:\actions-runner\_work\xgboost\xgboost\src\learner.cc:738:
Parameters: { "use_label_encoder" } are not used.

  bst.update(dtrain, iteration=i, fobj=obj)
```

```
Accuracy on Dev Set: 0.9998
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56863
           1       1.00      1.00      1.00     56863

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726


Training LightGBM...
[LightGBM] [Info] Number of positive: 170589, number of negative: 170589
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing wa
s 0.168829 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7650
[LightGBM] [Info] Number of data points in the train set: 341178, number of used fea
tures: 30
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.500000 -> initscore=0.000000
Accuracy on Dev Set: 0.9998
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56863
           1       1.00      1.00      1.00     56863

    accuracy                           1.00    113726
   macro avg       1.00      1.00      1.00    113726
weighted avg       1.00      1.00      1.00    113726
```

In [ ]: