Credit Card Fraud Detection

Objective:

The primary objective of the Credit Card Fraud Detection project is to build a robust and accurate machine learning model capable of identifying fraudulent credit card transactions. By detecting and preventing fraudulent transactions, the project aims to enhance the security of credit card usage and minimize financial losses for both consumers and financial institutions.

Project Description:

Credit card fraud is a significant concern for financial institutions and consumers alike. With the increasing volume of online transactions, detecting fraudulent activities in real-time has become crucial. This project aims to build a robust fraud detection system using machine learning algorithms to identify and flag suspicious transactions.

Importing Libraries

In [25]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from imblearn.over_sampling import RandomOverSampler
from sklearn.metrics import accuracy_score
import pickle
```

In [11]:
```
# Loading dataset
df = pd.read_csv('creditcard.csv')
```

In [12]:
```
df.head()
```

Out[12]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-(
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-(
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	(
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-(
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-(

5 rows × 31 columns

In [13]:
```
# checking for null values
df.isnull().sum()
```

Out[13]:
```
Time      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
```

In [14]:

```
# Class distribution
class_counts = df['Class'].value_counts()
print(class_counts)
```

```
Class
0    284315
1       492
Name: count, dtype: int64
```

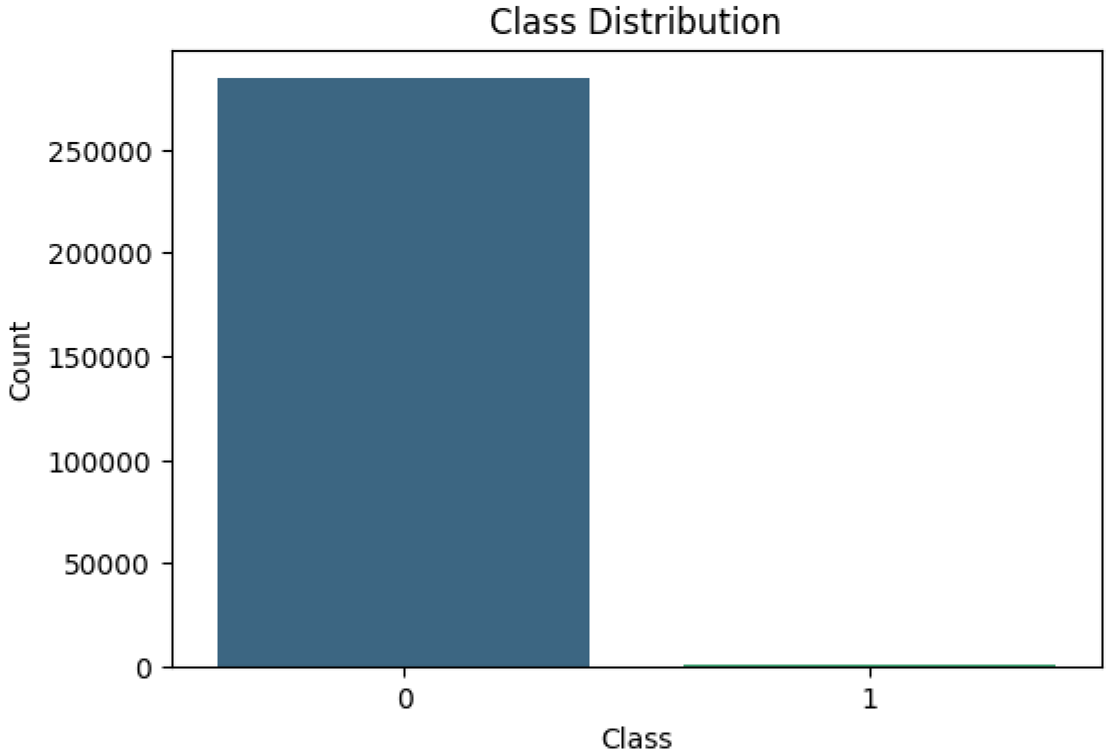
0 is real

1 is fraud

In [15]:

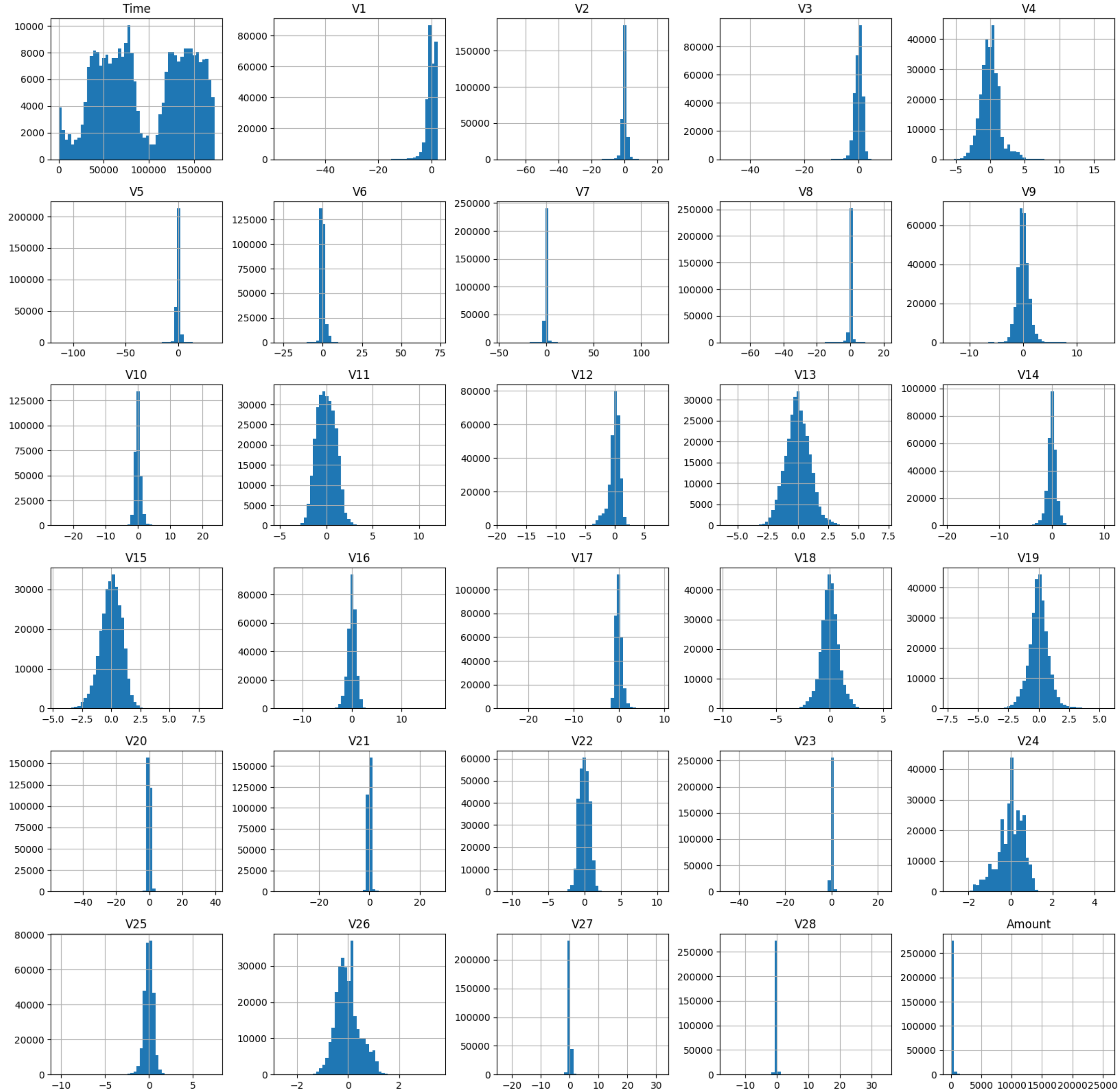
```
plt.figure(figsize=(6,4))
sns.barplot(x=class_counts.index, y=class_counts.values, palette='viridis')
plt.title('Class Distribution')
plt.xlabel('Class')
plt.ylabel('Count')
plt.show()
```

```
<ipython-input-15-2c6009a85595>:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x=class_counts.index, y=class_counts.values, palette='viridis')
```



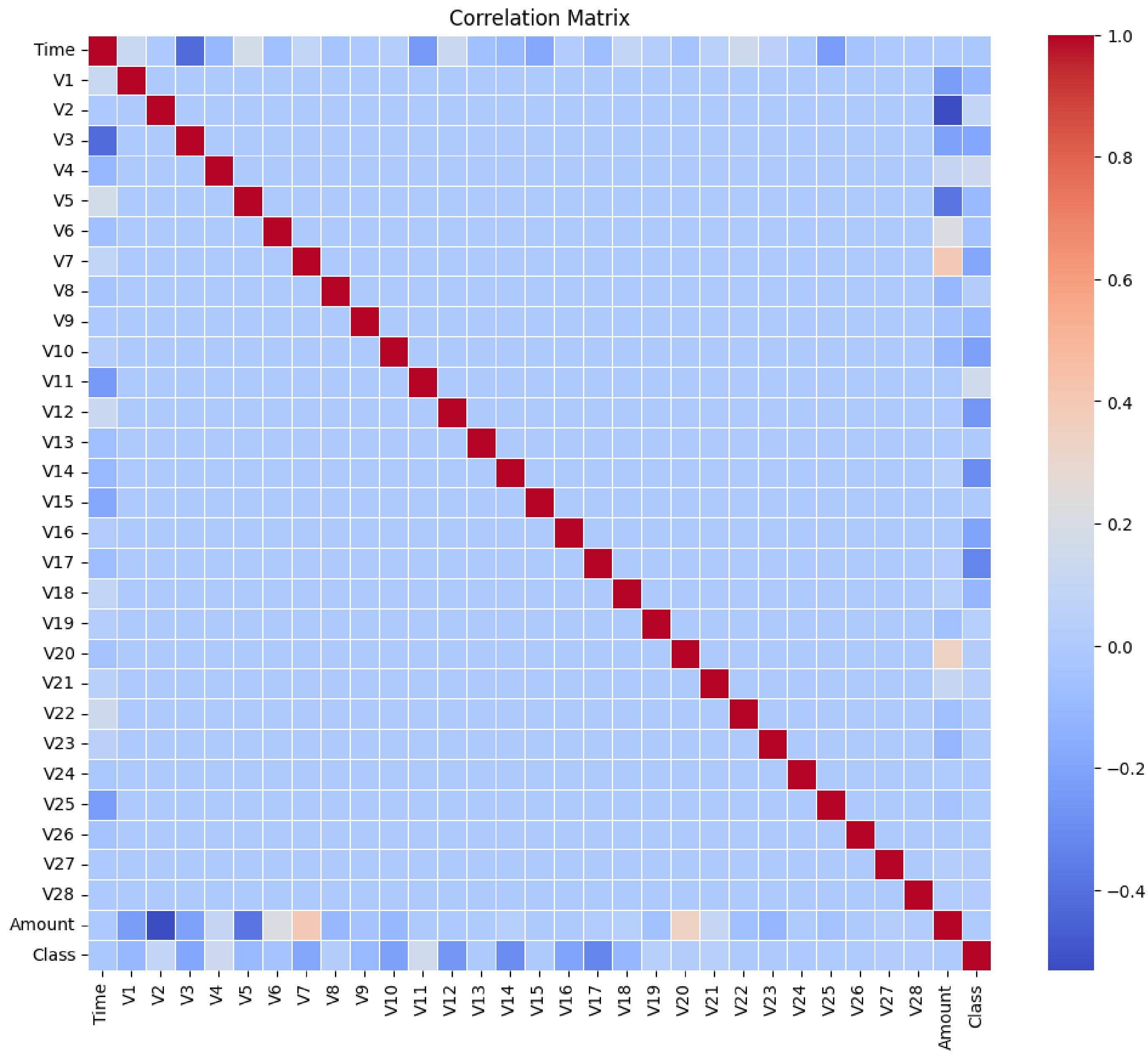
This dataset is imbalacned

In [16]:
```
# Plot distribution of features
df.drop('Class', axis=1).hist(figsize=(20,20), bins=50)
plt.show()

# Correlation matrix
corr_matrix = df.corr()
```



In [17]:

```
# Plot correlation matrix
plt.figure(figsize=(12,10))
sns.heatmap(corr_matrix, cmap='coolwarm', annot=False, fmt='.2f', linewidths=0.5)
plt.title('Correlation Matrix')
plt.show()
```



In [18]:
```
# Checking correlation with the target variable
print(corr_matrix['Class'].sort_values(ascending=False))
```

```
Class     1.000000
V11       0.154876
V4        0.133447
V2        0.091289
V21       0.040413
V19       0.034783
V20       0.020090
V8        0.019875
V27       0.017580
V28       0.009536
Amount    0.005632
V26       0.004455
V25       0.003308
V22       0.000805
V23      -0.002685
V15      -0.004223
V13      -0.004570
V24      -0.007221
Time     -0.012323
V6       -0.043643
V5       -0.094974
V9       -0.097733
```

```
V1       -0.101347
V18      -0.111485
V7       -0.187257
V3       -0.192961
V16      -0.196539
V10      -0.216883
V12      -0.260593
V14      -0.302544
V17      -0.326481
Name: Class, dtype: float64
```

In [19]:
```python
# Separating the features and the target
X = df.drop(columns=['Class'])
y = df['Class']
```

In [20]:
```python
# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [21]:
```python
# Handling class imbalance using RandomOverSampler
ros = RandomOverSampler()
X_resampled, y_resampled = ros.fit_resample(X_scaled, y)

# Spliting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=
```

In [22]:
```python
# Training a logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

Out[22]:
```
□      LogisticRegression
LogisticRegression(max_iter=1000)
```

In [23]:
```python
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')
```

```
Accuracy: 0.9493607442449395
```

In [26]:
```python
# Saving the trained model and scaler
with open('fraud_detection_model.pkl', 'wb') as file:
    pickle.dump(model, file)
with open('scaler.pkl', 'wb') as file:
    pickle.dump(scaler, file)
```

In [29]:
```python
# Function to predict fraud on new data
def predict_fraud(transaction, scaler, model, feature_names):
  # Ensuring the transaction data is in a DataFrame with the correct feature names
    transaction_df = pd.DataFrame([transaction], columns=feature_names)

    # Preprocess the new transaction
    transaction_scaled = scaler.transform(transaction_df)

    # Predict
    prediction = model.predict(transaction_scaled)
    return prediction[0]
    # Loading the model and scaler
    with open('fraud_detection_model.pkl', 'rb') as file:
        model = pickle.load(file)
    with open('scaler.pkl', 'rb') as file:
```

```
        scaler = pickle.load(file)

    # Predict
    prediction = model.predict(transaction_scaled)
    return prediction[0]

# Testing the prediction function
new_transaction = X_test[0]
feature_names = df.drop(columns=['Class']).columns
print(f'Prediction for new transaction: {predict_fraud(new_transaction, scaler, model, f
```

```
Prediction for new transaction: 1
```

In [ ]:

```

```