```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import IsolationForest
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score


# Loading the dataset
url = "https://storage.googleapis.com/download.tensorflow.org/data/creditcard.csv"
df = pd.read_csv(url)


# Displaying basic information about the dataset
print(df.info())
```

```
<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
None
```

```
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0

5 rows × 31 columns

```
df.shape
```

```
(284807, 31)
```

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

```
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
```

```
# To Check the distribution of the target variable
print(df['Class'].value_counts())
```

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

```
# Split the data into features (X) and target variable (y)
X = df.drop('Class', axis=1)
y = df['Class']
```

```
# Split the 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)
```

```
# Create a training set with only non-fraudulent examples
X_train_non_fraud = X_train[y_train == 0]
```

```
# Specify the contamination parameter based on the expected percentage of outliers
contamination = len(y_train[y_train == 1]) / len(y_train[y_train == 0])
```

```
# Train the Isolation Forest model
model = IsolationForest(contamination=contamination, random_state=42)
model.fit(X_train_non_fraud)
```

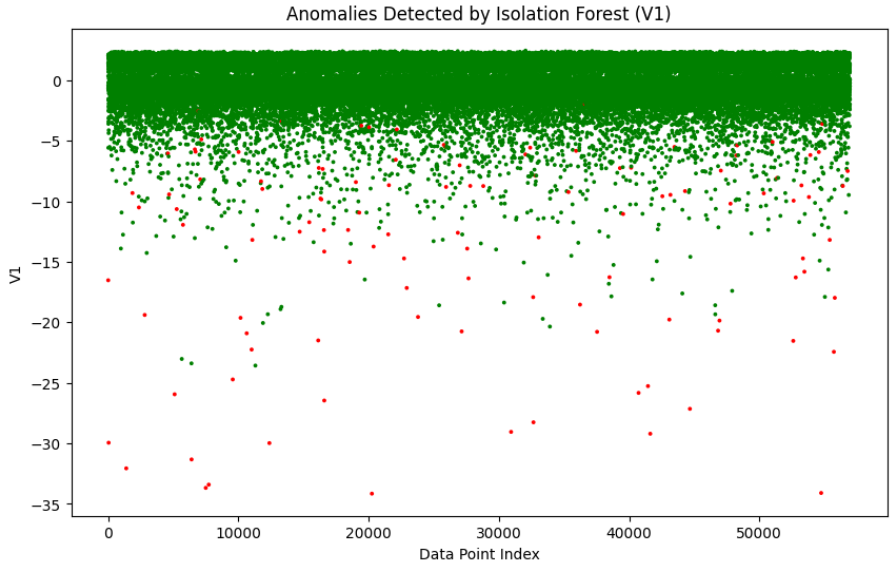
```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not hav
  warnings.warn(
```

```
▾ IsolationForest
IsolationForest(contamination=0.0017322412299792043, random_state=42)
```

```
# Make predictions on the entire dataset
y_pred = model.predict(X_test)
```

```
# Convert predictions to 0 for inliers (non-fraud) and 1 for outliers (fraud)
y_pred = np.where(y_pred == 1, 0, 1)
```

```
# Visualize anomalies
plt.figure(figsize=(10, 6))
colors = np.array(['green', 'red'])
plt.scatter(range(len(y_test)), X_test['V1'], s=3, color=colors[y_pred])
plt.title('Anomalies Detected by Isolation Forest (V1)')
plt.xlabel('Data Point Index')
plt.ylabel('V1')
plt.show()
```




```
# Evaluate the model on the testing set
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
print("\nAccuracy: {:.2f}%".format(accuracy_score(y_test, y_pred) * 100))
print("Precision: {:.2f}%".format(precision_score(y_test, y_pred) * 100))
print("Recall: {:.2f}%".format(recall_score(y_test, y_pred) * 100))
print("F1 Score: {:.2f}%".format(f1_score(y_test, y_pred) * 100))
```

```
Confusion Matrix:
 [[56778    86]
 [   70    28]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.25      0.29      0.26        98

    accuracy                           1.00     56962
   macro avg       0.62      0.64      0.63     56962
weighted avg       1.00      1.00      1.00     56962


Accuracy: 99.73%
Precision: 24.56%
Recall: 28.57%
F1 Score: 26.42%
```