Report

Exploratory Data Analysis Using Python: Credit Card Fraud Detection Dataset

MANUSHRI M

LOYOLA-ICAM COLLEGE OF ENGINEERING AND TECHNOLOGY

https://github.com/ManushriManavalane/EDA-Using-Python-Credit-Card-Fraud-Detection-Dataset/tree/4d2f0307eef4b42a698e5d38039ad1863a530243

Table of Contents

Introduction

The following report presents an in-depth analysis of a Credit Card Fraud Detection Dataset using Exploratory Data Analysis (EDA) techniques in Python. Credit card fraud has become a significant concern for individuals and financial institutions alike, with fraudulent transactions posing a substantial financial and security risk. EDA plays a crucial role in understanding the dataset's characteristics, identifying patterns, and uncovering insights that can aid in the development of effective fraud detection models.

The report aims to provide a comprehensive examination of the dataset, exploring various aspects such as data quality, distribution of fraudulent and non-fraudulent transactions, feature analysis, and class imbalance. By conducting a thorough EDA, we can gain valuable insights into the data, enabling us to make informed decisions and develop robust fraud detection strategies.

Dataset Overview

The Credit Card Fraud Detection Dataset used in this analysis is a collection of credit card transaction data. The dataset aims to provide insights into fraudulent and non-fraudulent credit card transactions, allowing for the development of effective fraud detection models. Through Exploratory Data Analysis (EDA), we explore the dataset's characteristics, patterns, and relationships between features to gain a deeper understanding of credit card fraud detection.

Initial Data Exploration

data.head() - displays the first few records of the dataset.

In [7]:
```
#Exploratory Data Analysis
#Initial Data Exploration
data.head()
```

Out[7]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V2
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	0.066928	0.12853
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	-0.339846	0.16717
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	-0.689281	-0.32764
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	-1.175575	0.64737
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	0.141267	-0.20601

5 rows × 31 columns

data.tail() - shows the last few records of the dataset.

In [8]:
```
data.tail()
```

Out[8]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797

5 rows × 31 columns

data.shape - returns the number of rows and columns in the dataset.

(284807, 31)

data.columns - **provides the list of column names in the dataset.**

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',

       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',

       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',

       'Class'],
      dtype='object')
```

data.nunique() - **provides the count of unique values in each column of the dataset.**

```
Time      124592
V1        275663
V2        275663
V3        275663
V4        275663
V5        275663
V6        275663
V7        275663
V8        275663
V9        275663
V10       275663
V11       275663
V12       275663
V13       275663
V14       275663
V15       275663
V16       275663
V17       275663
V18       275663
V19       275663
V20       275663
V21       275663
V22       275663
V23       275663
V24       275663
V25       275663
V26       275663
V27       275663
V28       275663
Amount     32767
Class          2
```

data.info() - **gives an overview of the dataset, including the data types and missing values.**

```
<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
```

Data Cleaning and Preprocessing

data.duplicated().sum() - returns the total count of duplicated records in the dataset.

In this particular dataset there are 1081 records that are duplicated.

Descriptive Statistics

Descriptive statistics are calculated to gain insights into the dataset.

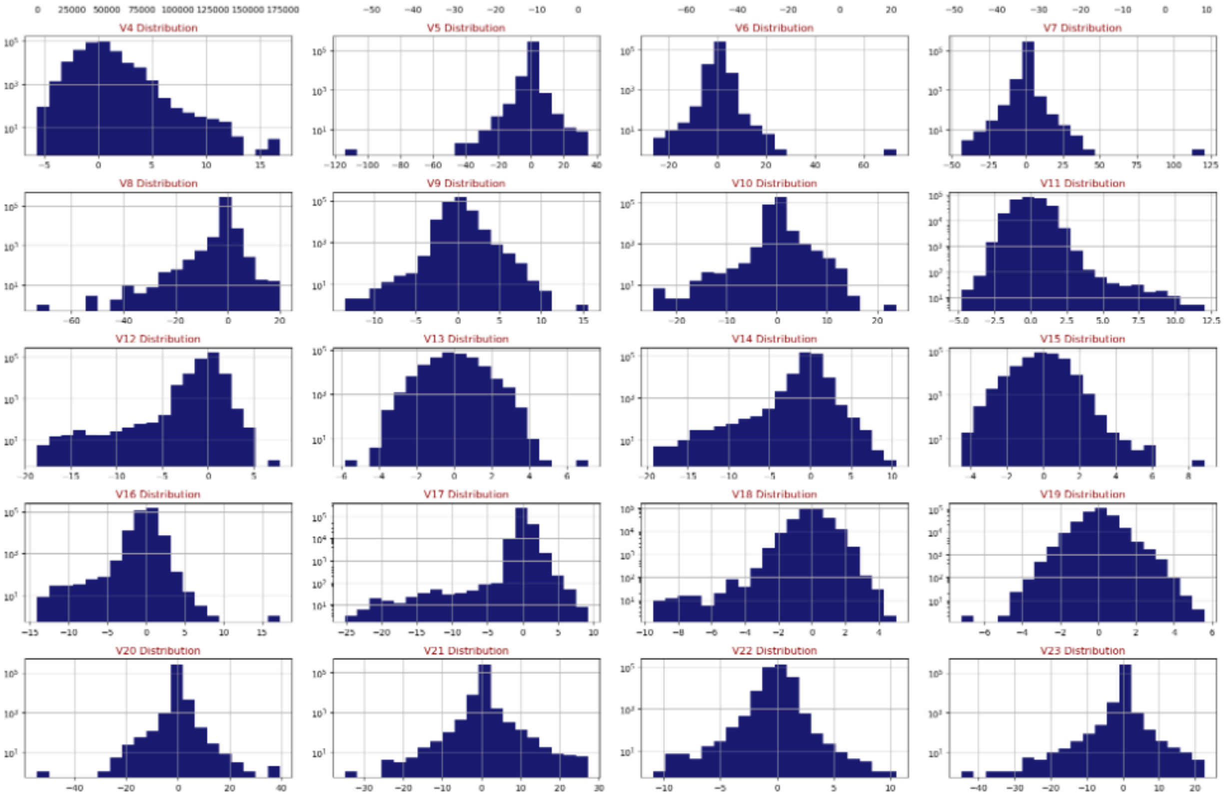
df.describe() - to obtain measures such as mean, median, standard deviation, and quartiles for numerical features.

	Time	V1	V2	V3	V4	V5	V6	V7	V8
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05
mean	94813.859575	3.918649e-15	5.682686e-16	-8.761736e-15	2.811118e-15	-1.552103e-15	2.040130e-15	-1.698953e-15	-1.893285e-16
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01

V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	284807.000000	284807.000000
1.473120e-16	8.042109e-16	5.282512e-16	4.456271e-15	1.426896e-15	1.701640e-15	-3.662252e-16	-1.217809e-16	88.349619	0.001727
7.345240e-01	7.257016e-01	6.244603e-01	6.056471e-01	5.212781e-01	4.822270e-01	4.036325e-01	3.300833e-01	250.120109	0.041527
-3.483038e+01	-1.093314e+01	-4.480774e+01	-2.836627e+00	-1.029540e+01	-2.604551e+00	-2.256568e+01	-1.543008e+01	0.000000	0.000000
-2.283949e-01	-5.423504e-01	-1.618463e-01	-3.545861e-01	-3.171451e-01	-3.269839e-01	-7.083953e-02	-5.295979e-02	5.600000	0.000000
-2.945017e-02	6.781943e-03	-1.119293e-02	4.097606e-02	1.659350e-02	-5.213911e-02	1.342146e-03	1.124383e-02	22.000000	0.000000
1.863772e-01	5.285536e-01	1.476421e-01	4.395266e-01	3.507156e-01	2.409522e-01	9.104512e-02	7.827995e-02	77.165000	0.000000
2.720284e+01	1.050309e+01	2.252841e+01	4.584549e+00	7.519589e+00	3.517346e+00	3.161220e+01	3.384781e+01	25691.160000	1.000000

Data Visualization

Histogram: to identify patterns, distributions, and relationships between variables



In a histogram, various shapes of distribution can be observed, including:

1. Normal Distribution (Gaussian): A symmetrical bell-shaped curve, where the data is evenly distributed around the mean.

2. Skewed Distribution: The data is not evenly distributed and is inclined towards one side. It can be either positively skewed (long tail on the right) or negatively skewed (long tail on the left).

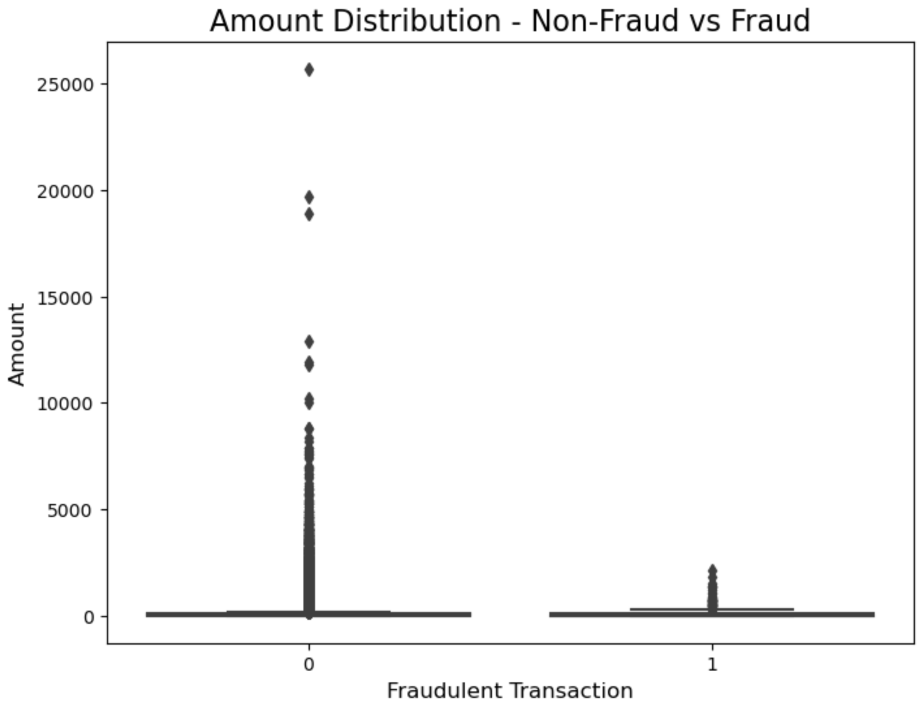
3. Uniform Distribution: The data is evenly distributed, and each value or value range has roughly the same frequency.

4. Bimodal Distribution: The data has two distinct peaks or modes, indicating the presence of two different groups or subpopulations within the dataset.

5. Multimodal Distribution: Similar to bimodal distribution, but with more than two peaks, indicating the presence of multiple groups or subpopulations.

Feature Analysis

Exploring individual features and analyzing their impact on fraud detection.



The above diagram infers that fraudulent cases are occurring when the amount is between 0 - 5000 and when compared to non-fraudulent cases fraudulent are comparatively less.

Class Imbalance Analysis

Class imbalance analysis refers to the examination of the disproportionate distribution of different classes or categories in a dataset, such as the uneven representation of fraud and non-fraud instances in credit card transactions.

Positive (Fraudulent) Instances: 492

Negative (Non-Fraudulent) Instances: 284315
Imbalance Ratio (Positive to Negative): 0.0017304750013189597

Correlation Analysis

Correlation analysis involves quantifying the relationship or association between two or more variables to determine how they are related to each other.

	Time	V1	V2	V3	V4
Time	1.000000	1.173963e-01	-1.059333e-02	-4.196182e-01	-1.052602e-01
V1	0.117396	1.000000e+00	4.135835e-16	-1.227819e-15	-9.215150e-16
V2	-0.010593	4.135835e-16	1.000000e+00	3.243764e-16	-1.121065e-15
V3	-0.419618	-1.227819e-15	3.243764e-16	1.000000e+00	4.711293e-16
V4	-0.105260	-9.215150e-16	-1.121065e-15	4.711293e-16	1.000000e+00
V5	0.173072	1.812612e-17	5.157519e-16	-6.539009e-17	-1.719944e-15
V6	-0.063016	-6.506567e-16	2.787346e-16	1.627627e-15	-7.491959e-16

Conclusion & Inference

The Exploratory Data Analysis (EDA) performed on the Credit Card Fraud Detection Dataset has provided valuable insights into the characteristics and patterns of fraudulent and non-fraudulent credit card transactions. Through various analysis techniques, we have gained a deeper understanding of the dataset, identified potential outliers and explored the relationships between different features and fraud.

Additionally, the class imbalance analysis indicated that the non-fraudulent cases are comparatively higher than the fraudulent cases. Through feature analysis we were able to determine that fraudulent cases are occurring when the amount is between 0-5000. These insights are instrumental in guiding the development of effective fraud detection models.

Overall, this EDA serves as a solid foundation for subsequent steps in credit card fraud detection, including feature engineering, model selection, and evaluation. By leveraging the insights gained from this analysis, it is possible to develop robust and accurate fraud detection models that can help mitigate the financial and security risks associated with fraudulent credit card transactions.