CREDIT CARD FRAUD DETECTION

PHASE 4:SUBMISSION

- there are a few other things to keep in mind when loading and preprocessing data for a credit card fraud detection project:

❖ Feature engineering:

- ✓ It may be helpful to create new features from the existing data. For example, you could create a feature that represents the total amount spent by a customer in the past week.

❖ Imbalanced data:

- ✓ Credit card fraud detection datasets are often imbalanced, meaning that there are many more non-fraudulent transactions than fraudulent transactions. This can make it difficult for machine learning algorithms to learn to detect fraudulent transactions. There are a variety of ways to handle imbalanced data, such as oversampling the minority class or undersampling the majority class.

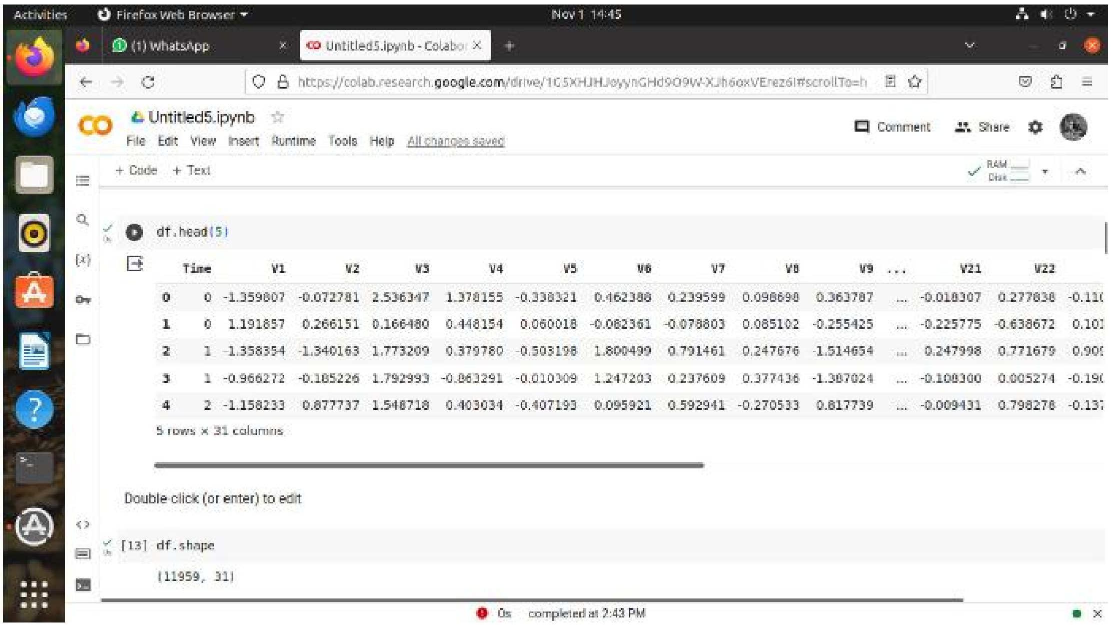
❖ Data security:

- ✓ It is important to keep in mind that credit card fraud detection data is sensitive. It is important to take steps to protect this data, such as encrypting the data and storing it in a secure location.

DATASET LINK:"https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud"

◆ HEAD ()

- ✓ The head() function in a dataset is used to view the first few rows of the data. This can be useful for getting a quick overview of the data and its structure. For example, you can use the head() function to see the column names, data types, and the first few rows of the data.

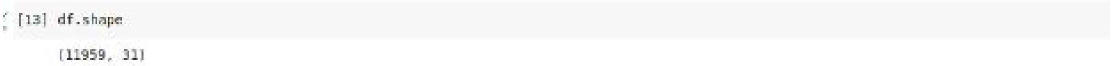

Activities
Firefox Web Browser
Nov 1 14:45
(1) WhatsApp
Untitled5.ipynb - Colabo
https://colab.research.google.com/drive/1G5XHJHJoyynGHd9O9W-XJh6oxVErez6l#scrollTo=h
Untitled5.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Double click (or enter) to edit
0s completed at 2:43 PM

```
df.head(5)

   Time        V1        V2        V3        V4        V5        V6        V7        V8        V9  ...       V21       V22
0     0 -1.359807 -0.072781  2.536347  1.378155 -0.338321  0.462388  0.239599  0.098698  0.363787  ... -0.018307  0.277838 -0.11
1     0  1.191857  0.266151  0.166480  0.448154  0.060018 -0.082361 -0.078803  0.085102 -0.255425  ... -0.225775 -0.638672  0.10
2     1 -1.358354 -1.340163  1.773209  0.379780 -0.503198  1.800499  0.791461  0.247676 -1.514654  ...  0.247998  0.771679  0.90
3     1 -0.966272 -0.185226  1.792993 -0.863291 -0.010309  1.247203  0.237609  0.377436 -1.387024  ... -0.108300  0.005274 -0.19
4     2 -1.158233  0.877737  1.548718  0.403034 -0.407193  0.095921  0.592941 -0.270533  0.817739  ... -0.009431  0.798278 -0.13
5 rows × 31 columns
```

```
[13] df.shape

(11959, 31)
```

◆ SHAPE

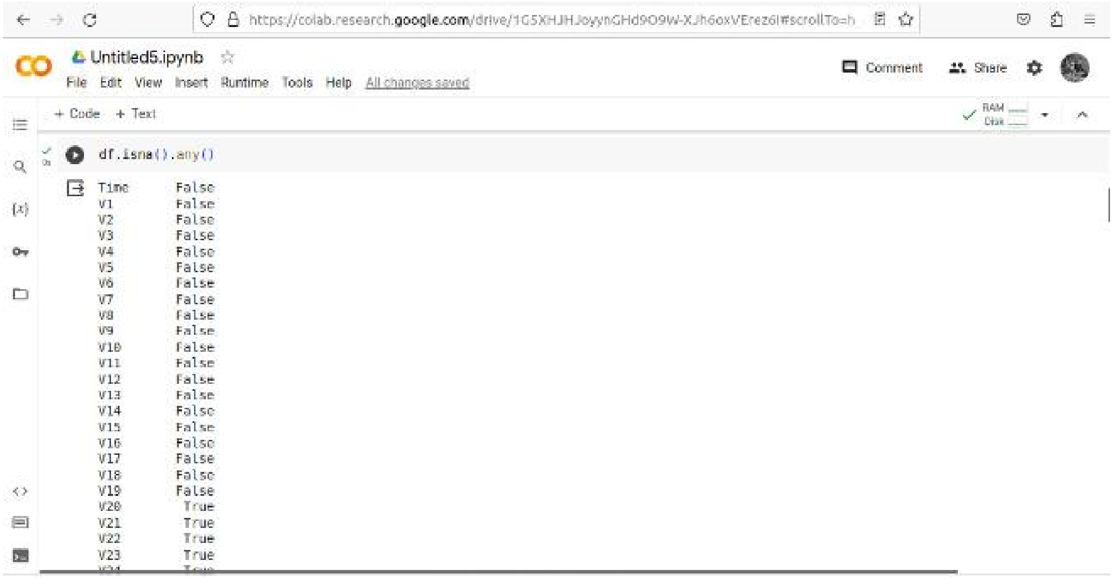
- ✓ The df.shape function can be used in conjunction with other functions to perform various tasks, such as:
- ✓ Checking to see if a dataset is the correct size for a specific machine learning algorithm.
- ✓ Calculating the number of samples in a dataset.
- ✓ Calculating the number of features in a dataset.
- ✓ Slicing and dicing a dataset based on its shape.



```
[13] df.shape

(11959, 31)
```

◆ ISNA().ANY()

- ✓ The df.isna().any() function in a dataset returns a boolean value indicating whether there are any missing values in the dataset. This function is useful for checking the quality of the data and identifying any potential problems.
- ✓ If the df.isna().any() function returns True, then there are missing values in the dataset. If the df.isna().any() function returns False, then there are no missing values in the dataset.

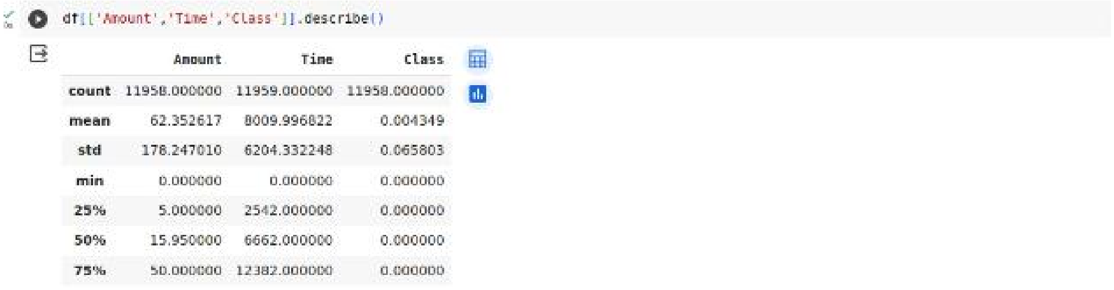

https://colab.research.google.com/drive/1G5XHJHJoyynGHd9O9W-XJh6oxVErez6l#scrollTo=h
Untitled5.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk

```
df.isna().any()
```

```
Time      False
V1        False
V2        False
V3        False
V4        False
V5        False
V6        False
V7        False
V8        False
V9        False
V10       False
V11       False
V12       False
V13       False
V14       False
V15       False
V16       False
V17       False
V18       False
V19       False
V20        True
V21        True
V22        True
V23        True
```

◆ DESCRIBE()

- ➢ The df[['Amount','Time','Class']].describe() function in Python returns a summary of the statistical properties of the specified columns in the DataFrame. This information can be useful for understanding the distribution of the data and identifying any potential outliers.
- ➢ The summary includes the following metrics for each column:
  - ✓ Count: The number of non-null values in the column.
  - ✓ Mean: The average value in the column.
  - ✓ Std: The standard deviation of the values in the column.
  - ✓ Min: The minimum value in the column.
  - ✓ 25%: The 25th percentile of the values in the column.
  - ✓ 50%: The 50th percentile of the values in the column (also known as the median).
  - ✓ 75%: The 75th percentile of the values in the column.
  - ✓ Max: The maximum value in the column.



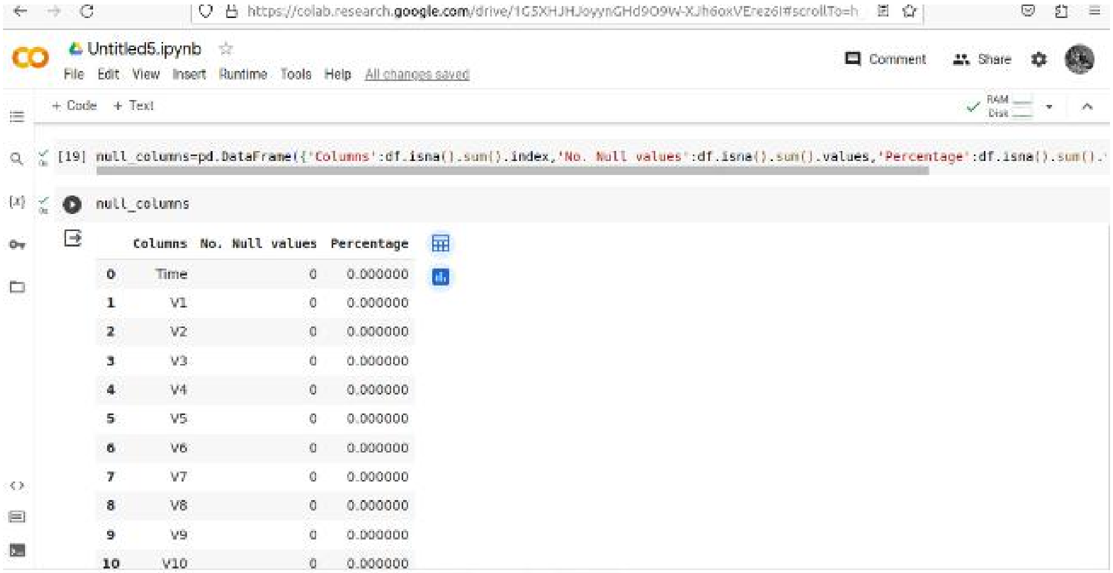
```
df[['Amount','Time','Class']].describe()
```

	Amount	Time	Class
count	11958.000000	11959.000000	11958.000000
mean	62.352617	8009.996822	0.004349
std	178.247010	6204.332248	0.065803
min	0.000000	0.000000	0.000000
25%	5.000000	2542.000000	0.000000
50%	15.950000	6662.000000	0.000000
75%	50.000000	12382.000000	0.000000

◆ NULL_COLUMNS

- ➢ The null_columns variable contains a list of all the columns in the dataset that contain missing values. This variable can be used to identify and handle missing values in the dataset.
- ➢ There are a variety of ways to handle missing values in a dataset. Some common methods include:

- ✓ Dropping the columns with missing values: This is the simplest method, but it can lead to a loss of data.
- ✓ Imputing the missing values: This involves replacing the missing values with a default value, such as the mean or median value of the column.
- ✓ Using a more sophisticated method like K-nearest neighbors (KNN) imputation: This method involves finding the K most similar rows to the row with the missing value and using the values in those rows to impute the missing value.

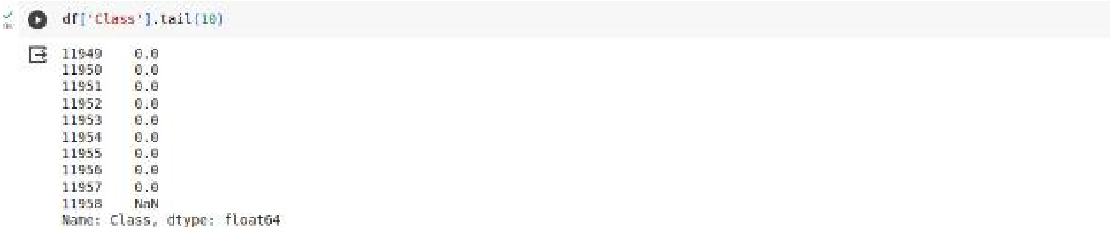

https://colab.research.google.com/drive/1G5XHJHJoyynGHd9O9W-XJh6oxVErez6I#scrollTo=h
Untitled5.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk

```
[19] null_columns=pd.DataFrame({'Columns':df.isna().sum().index,'No. Null values':df.isna().sum().values,'Percentage':df.isna().sum().

null_columns
```

	Columns	No. Null values	Percentage
0	Time	0	0.000000
1	V1	0	0.000000
2	V2	0	0.000000
3	V3	0	0.000000
4	V4	0	0.000000
5	V5	0	0.000000
6	V6	0	0.000000
7	V7	0	0.000000
8	V8	0	0.000000
9	V9	0	0.000000
10	V10	0	0.000000

◆ TAIL()

- ✓ The tail(10) function in Python is a function that returns the last 10 rows of a pandas DataFrame. This function is useful for getting a quick overview of the end of the data and identifying any potential outliers.
- ✓ For example, you could use the tail(10) function to see the last 10 rows of a credit card fraud detection dataset to identify any potential fraudulent transactions.



```
df['Class'].tail(10)

11949    0.0
11950    0.0
11951    0.0
11952    0.0
11953    0.0
11954    0.0
11955    0.0
11956    0.0
11957    0.0
11958    NaN
Name: Class, dtype: float64
```