

# CREDIT CARD FRAUD DETECTION WITH MACHINE LEARNING

## Phase 5:

In this phase, the entire project is documented

### Problem Definition:

Develop a machine learning model for credit card fraud detection that effectively identifies and prevents fraudulent transactions while minimizing false alarms. The objective is to protect both cardholders and financial institutions from financial losses by accurately distinguishing between legitimate and fraudulent transactions in real-time. The model should leverage historical transaction data, incorporate advanced anomaly detection and classification techniques, and continuously adapt to evolving fraud patterns

### Design Thinking:

**Data Collection:** Data collection in credit card fraud detection involves gathering historical transaction data and relevant information that can be used to train and evaluate machine learning models for fraud detection.

**Data Preprocessing:** Clean and preprocess the data, handle missing values, and convert categorical features into numerical representations.

**Exploratory Data Analysis:** Explore the data to understand its characteristics, identify trends, and outliers.

**Statistical Analysis:** Perform statistical tests to analyze vaccine efficacy, adverse effects, and distribution across different populations.

**Visualization:** Create visualizations (e.g., bar plots, line charts, heatmaps) to present key findings and insights.

## **Data collection:**

The first step is to collect data that is relevant to the product demand prediction task. Once the data is collected, it needs to be cleaned and prepared for modeling.

The given dataset:

<https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud>

## **Data Preprocessing:**

Data preprocessing is the initial step in preparing data for machine learning. It involves tasks like handling missing values, removing outliers, encoding variables, and scaling features. The goal is to clean, transform, and structure the data to improve its quality and make it suitable for training predictive models, ultimately enhancing their accuracy and performance.

In context of creditcard fraudulent detection, we are going to loadbalance the dataset provided which consists of legitimate and illegitimate transactions.

In data preprocessing of this step of this project we are going to balance the number of legitimate and illegitimate transactions by nearly bringing the transaction count equal in coding phase and then next phase is carried out

The dataset is cleaned

## **Exploratory Data Analysis:**

### **Loading dataset**

Colab interface showing the initial steps of loading the dataset:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# loading dataset
credit_card=pd.read_csv("/content/creditcard.csv")

#first five rows of dataset
credit_card.head()
```

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

5 rows x 31 columns

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

Colab interface showing the classification of transactions:

```
Name: Class, dtype: int64

[13] legit=credit_card[credit_card.Class==0]
      fraud=credit_card[credit_card.Class==1]

print(legit.shape)
print(fraud.shape)

(15862, 31)
(73, 31)

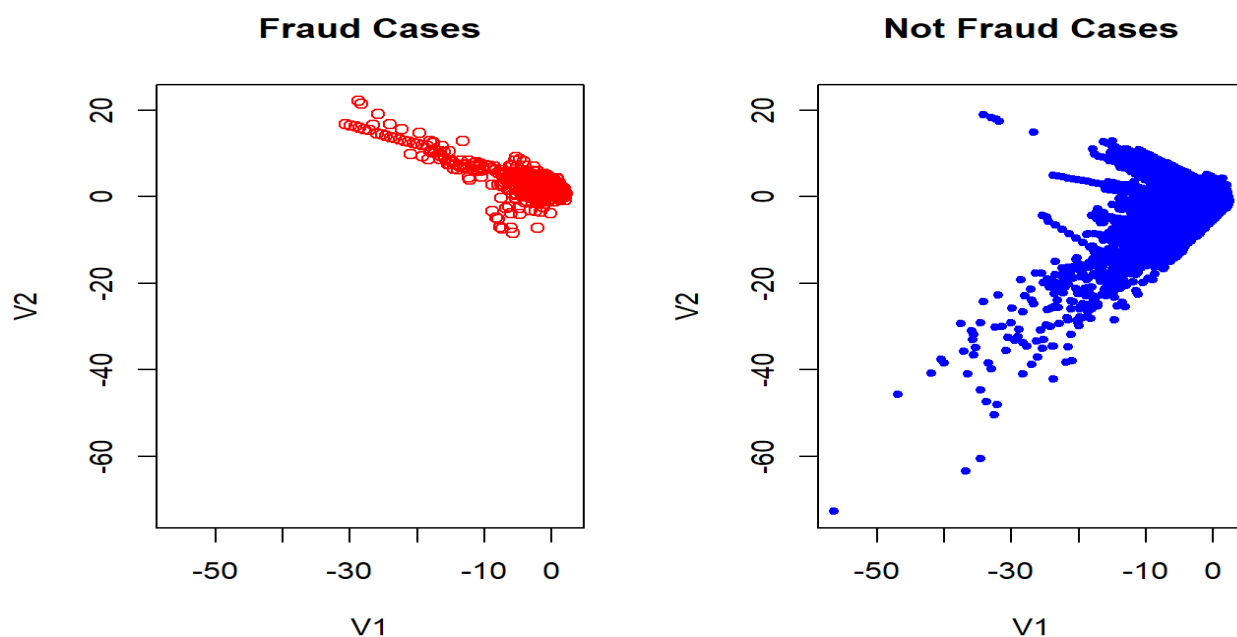
[15] legit.Amount.describe()

count    15862.000000
mean      66.280151
std       188.898885
min        0.000000
25%        5.522500
50%       15.950000
75%       53.890000
max      7712.430000
Name: Amount, dtype: float64
```

The above image represents the number of legit and fraud transactions.

## Statistical Analysis

As per the statistics



## Machine learning

Machine learning (ML) is a popular use of artificial intelligence since it automates the system and allows it to learn and improve from diverse experiences without being programmed. Computer programs can teach how to learn by giving them access to data and allowing them to utilize it for learning in ML. The learning process in ML begins with seeing the data through examples or instructions that humans offer; these observations enable ML to look for patterns in order to make the best predictions. Five different ML models were used to train the classifier and evaluate classification performance using the test dataset. These are discussed below.

## Machine learning Techniques

There are several techniques in Machine Learning including

- Random Forest

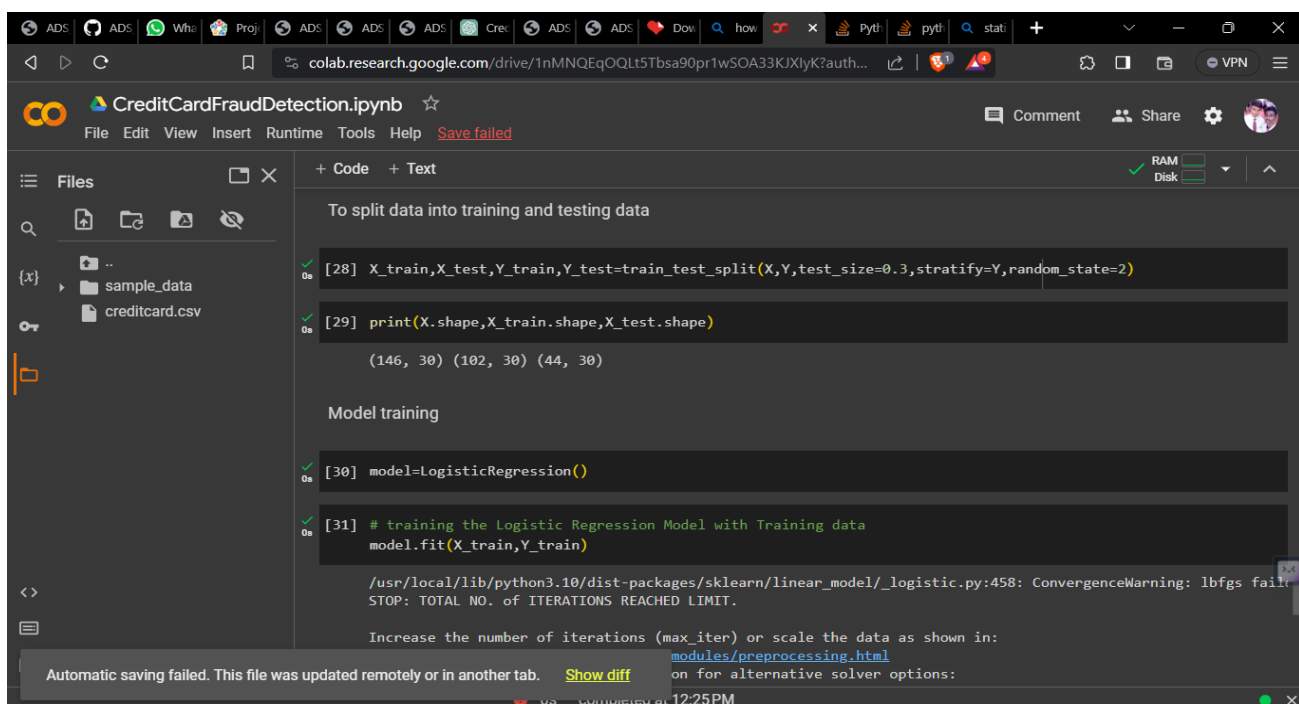
- Naive Bayes
- Decision Tree
- Logistic Regressions
- Support Vector Machine

Our choice of Machine Learning technique is Logistic Regression

## Logistic Regression

Logistic Regression is a statistical approach to data analysis in which one or more variables are utilized to determine the outcome. When the target variable is categorical, the optimum learning model to utilize is LR, which is the regression model that was used to estimate the likelihood of class members. Linear Regression uses a logistic function to estimate probabilities for the association between the categorical dependent variable and one or more independent variables.

Once the data is being pre-processed and got load balanced the entire data set is split into training and testing data and the model is trained successfully.



```
File Edit View Insert Runtime Tools Help Save failed

CreditCardFraudDetection.ipynb ☆
Comment Share Settings

Files
sample_data
creditcard.csv

+ Code + Text
RAM
Disk

To split data into training and testing data

[28] X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.3,stratify=Y,random_state=2)
[29] print(X.shape,X_train.shape,X_test.shape)

(146, 30) (102, 30) (44, 30)

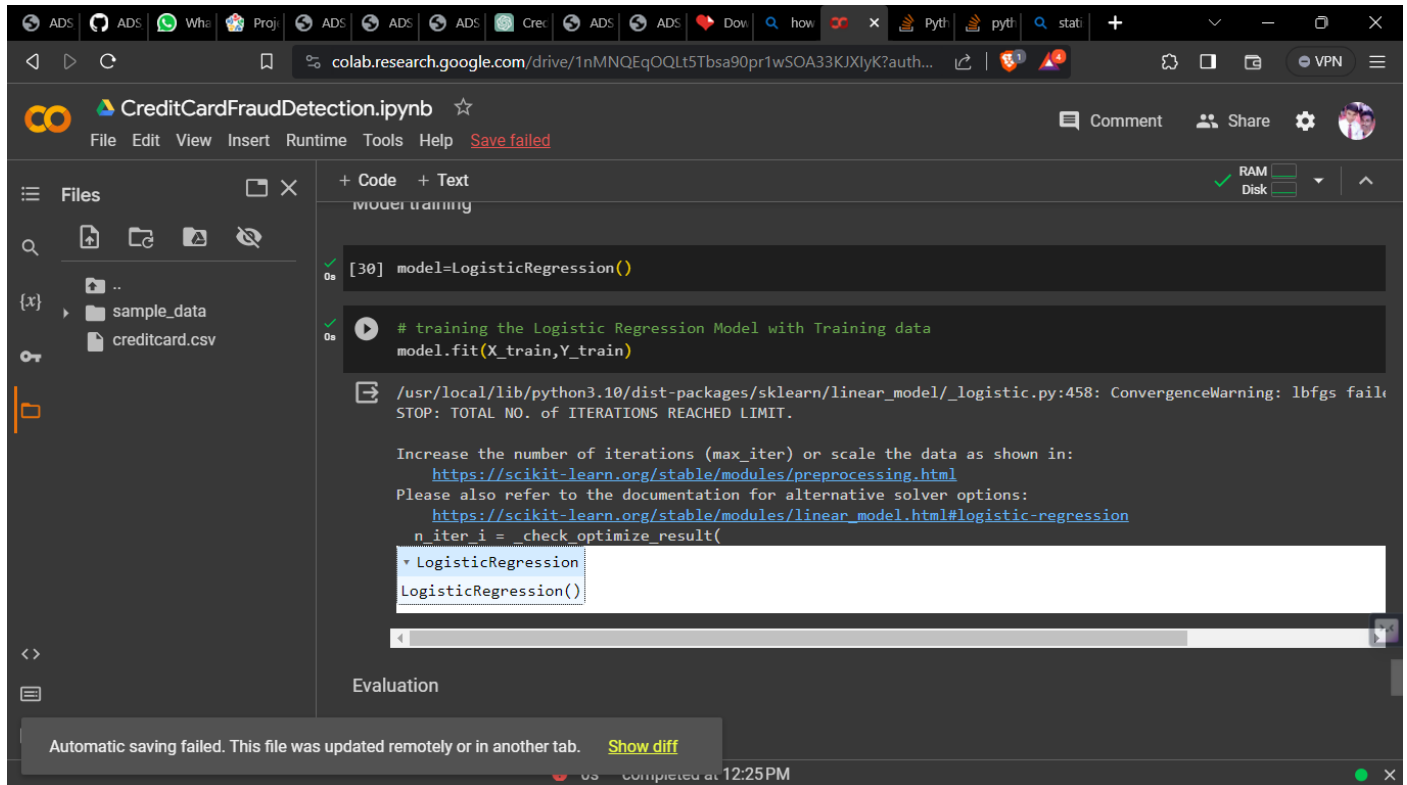
Model training

[30] model=LogisticRegression()
[31] # training the Logistic Regression Model with Training data
model.fit(X_train,Y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (50 iterations total). Increase the number of iterations (max_iter) or scale the data as shown in:
modules/preprocessing.html
on for alternative solver options:

Automatic saving failed. This file was updated remotely or in another tab. Show diff
Completed at 12:25PM
```

## Model training:



```
[30] model=LogisticRegression()

# training the Logistic Regression Model with Training data
model.fit(X_train,Y_train)

/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge. Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
  LogisticRegression
  LogisticRegression())
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

## Model Evaluation:

There are many evaluation metrics our choice is Accuracy score

Accuracy is a common performance metric used in machine learning, particularly for classification tasks. It measures the proportion of correctly classified instances out of the total instances in the dataset. In other words, it calculates how many predictions made by the model are correct.

The accuracy score is calculated as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

The accuracy on training and testing data as follows:

Colab interface showing a Jupyter Notebook titled "CreditCardFraudDetection.ipynb". The notebook contains Python code for calculating accuracy scores on training and test data. The output shows training accuracy as 0.975609756097561 and test accuracy as 0.9811320754716981. A message at the bottom states: "Automatic saving failed. This file was updated remotely or in another tab. Show diff".

```
[ ] # accuracy on training data
X_train_prediction=model.predict(X_train)
training_data_accuracy=accuracy_score(X_train_prediction,Y_train)

[ ] print("Accuracy on Trainind data:",training_data_accuracy)

Accuracy on Trainind data: 0.975609756097561

[ ] # accuracy on test data
X_test_prediction=model.predict(X_test)
test_data_accuracy=accuracy_score(X_test_prediction,Y_test)

[ ] print(test_data_accuracy)

0.9811320754716981
```

## Visualization:



## Project Overview in development phase

colab.research.google.com/drive/1nMNQEqOQLt5Tbsa90pr1wSOA33KJXlyK?auth... | VPN

### CreditCardFraudDetection.ipynb

File Edit View Insert Runtime Tools Help Save failed

Comment Share

Files

- sample\_data
- creditcard.csv

```
[4] # importing dependencies
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

# loading dataset
credit_card=pd.read_csv("/content/creditcard.csv")

#first five rows of dataset
credit_card.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	1
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.013
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247
3		-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108			

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

colab.research.google.com/drive/1nMNQEqOQLt5Tbsa90pr1wSOA33KJXlyK?auth...

### CreditCardFraudDetection.ipynb

File Edit View Insert Runtime Tools Help Save failed

+ Code + Text

RAM ✓ Disk ✓

Files

- sample\_data
- creditcard.csv

```
[6] credit_card=pd.read_csv("../content/creditcard.csv")

#first five rows of dataset
credit_card.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.0183
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.2251
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.2471
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.1083
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.0094

5 rows x 31 columns

```
[8] # lats five rows
credit_card.tail()
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)



Colab interface showing a Jupyter Notebook titled "CreditCardFraudDetection.ipynb". The notebook is open to the "Code" tab. The file explorer on the left shows a folder named "sample\_data" containing a file named "creditcard.csv".

The code cell contains the following Python code:

```
# lats five rows
credit_card.tail()
```

The output of the code is a DataFrame showing the last five rows of the "creditcard.csv" file. The DataFrame has 31 columns: "Time", "V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", and "...". The rows are indexed 15931 to 15935.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
15931	27369	-1.160116	-0.244177	0.744250	-0.192350	1.156356	-1.931383	0.409670	-0.364716	-0.516156	...
15932	27369	-3.058318	3.099206	-4.932555	1.924138	-1.576032	-2.135383	-0.830098	2.228617	-0.312343	...
15933	27369	-0.661806	0.315385	2.011194	-0.438757	-0.554990	-0.668072	0.424651	0.079141	0.126057	...
15934	27370	1.525348	-1.231442	0.420095	-1.551218	-1.376006	0.100758	-1.455755	0.134876	-1.319056	...
15935	27371	1.385680	-0.590076	-0.569197	-0.939441	-0.196015	-0.486685	-0.102496	-0.237930	-0.928028	...

5 rows x 31 columns

The code cell also contains the following Python code:

```
[9] credit_card.info()
```

The output of the code is the following information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15936 entries, 0 to 15935
```

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

Connected to Python's Google Compute Engine backend

Colab interface showing a Jupyter Notebook titled "CreditCardFraudDetection.ipynb". The notebook is open to the "Code" tab. The file explorer on the left shows a folder named "sample\_data" containing a file named "creditcard.csv".

The code cell contains the following Python code:

```
credit_card.isnull().sum()
```

The output of the code is a DataFrame showing the number of missing values for each column in the "creditcard.csv" file. The DataFrame has 21 columns: "Time", "V1", "V2", "V3", "V4", "V5", "V6", "V7", "V8", "V9", "V10", "V11", "V12", "V13", "V14", "V15", "V16", "V17", "V18", "V19", and "V20". The rows are indexed 0 to 20.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19	V20
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

Connected to Python's Google Compute Engine backend

Colab interface showing the execution of a Jupyter Notebook titled "CreditCardFraudDetection.ipynb". The notebook is connected to a Python 3 Google Compute Engine backend. The file explorer on the left shows a folder named "sample\_data" containing a file named "creditcard.csv".

The code cell [13] defines two variables:

```
legit=credit_card[credit_card.Class==0]
fraud=credit_card[credit_card.Class==1]
```

The code cell [14] prints the shapes of the two datasets:

```
print(legit.shape)
print(fraud.shape)
```

The output shows the shapes of the datasets:

```
(15862, 31)
(73, 31)
```

The code cell [15] displays the summary statistics for the "legit" dataset using `legit.Amount.describe()`:

```
count    15862.000000
mean       66.280151
std       188.898885
min         0.000000
25%        5.522500
50%       15.950000
75%       53.890000
max      7712.430000
Name: Amount, dtype: float64
```

An error message at the bottom states: "Automatic saving failed. This file was updated remotely or in another tab. Show diff".

Colab interface showing the execution of a Jupyter Notebook titled "CreditCardFraudDetection.ipynb". The notebook is connected to a Python 3 Google Compute Engine backend. The file explorer on the left shows a folder named "sample\_data" containing a file named "creditcard.csv".

The code cell [16] displays the summary statistics for the "fraud" dataset using `fraud.Amount.describe()`:

```
count      73.000000
mean       90.307123
std       271.634360
min         0.000000
25%        1.000000
50%        1.000000
75%       99.990000
max      1809.680000
Name: Amount, dtype: float64
```

The code cell [17] displays the mean values for each variable, grouped by "Class" using `credit_card.groupby('Class').mean()`:

```
Class
0.0  12104.432165 -0.219072  0.25000  0.862854  0.272641 -0.105868  0.124522 -0.112681 -0.016178  0.879120
1.0  15559.643836 -7.929807  6.19312 -11.997831  6.555050 -5.474984 -2.480356 -8.354317  3.668478 -3.086988
```

The output shows 2 rows x 30 columns.

An error message at the bottom states: "Automatic saving failed. This file was updated remotely or in another tab. Show diff".

Colab interface showing the initial setup for CreditCardFraudDetection.ipynb. The notebook is titled "CreditCardFraudDetection.ipynb" and is saved to Google Drive. The file explorer on the left shows the "sample\_data" folder containing "creditcard.csv". The code cell [18] defines "legit\_sample" as a sample of 73 normal transactions. Cell [20] concatenates "legit\_sample" and "fraud" to create "new\_dataset". Cell [21] displays the first few rows of "new\_dataset".

```
[18] legit_sample=legit.sample(n=73)
```

```
[20] new_dataset=pd.concat([legit_sample,fraud],axis=0)
```

```
[21] new_dataset.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
12964	22786	-0.853077	-0.123772	2.306695	-1.495506	-0.099336	-0.803062	-0.252171	0.115503	2.594355	...
1791	1395	1.228507	-0.070446	0.198873	0.462775	-0.022367	0.348405	-0.205868	0.109798	0.323381	...
8350	11163	-0.483971	0.482274	1.605304	-0.603256	-0.098256	-0.442161	0.092873	0.041258	1.163763	...
15473	26859	0.067406	1.264429	1.373015	3.389687	0.097276	-0.352311	0.620731	-0.309283	-0.720791	...

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

Connected to Python's Google Compute Engine backend

Colab interface showing the continuation of the CreditCardFraudDetection.ipynb notebook. The code cell [23] displays the value counts for the "Class" variable, showing 73 instances for both 0.0 (normal) and 1.0 (fraudulent). Cell [24] calculates the mean for each feature across the two classes. The resulting table shows the mean values for each feature (Time, V1, V2, V3, V4, V5, V6, V7, V8, V9) for both classes.

```
[23] new_dataset['Class'].value_counts()
```

```
0.0    73
1.0    73
Name: Class, dtype: int64
```

```
[24] new_dataset.groupby('Class').mean()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
Class										
0.0	12505.821918	-0.395269	0.420343	0.391803	0.457549	-0.228428	0.218180	-0.423821	0.295796	0.832196
1.0	15559.643836	-7.929807	6.193120	-11.997831	6.555050	-5.474984	-2.480356	-8.354317	3.668478	-3.086988

2 rows x 30 columns

Splitting the data into Features and Target

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

Connected to Python's Google Compute Engine backend