A

PROJECT REPORT

On

CARDSAFE FORENSICS - MACHINE LEARNING IN COMBATING FINANCIAL DECEPTIONS

Submitted to

JAWAHARLAL NEHRU TECHNOLOGYICAL UNIVERSITY ANANTHAPUR,

ANANTHAPURAM.

For partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN
INFORMATION TECHNOLOGY

Submitted by

A. VARSHITHA	**-20781A1205**
B. PRAVEEN KUMAR REDDY	**-20781A1209**
B. KISHORE BABU	**-20781A1210**
D. MADHURI	**-20781A1211**
K. SAI TARUN YADAV	**-20781A1222**

Under the Esteemed Guidance of

Mr. P. NANDA KUMAR, M.E., (Ph. D)

Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY

Affiliated to JNTUA, Anantapur, Approved by AICTE, New Delhi, Accredited by NBA, New Delhi & NAAC with A+ Grade, Bangalore. An ISO 9001-2000 Certified Institution), R.V.S NAGAR, CHITTOOR-517127

2023-2024

A

PROJECT REPORT

On

CARDSAFE FORENSICS - MACHINE LEARNING IN COMBATING FINANCIAL DECEPTIONS

Submitted to

JAWAHARLAL NEHRU TECHNOLOGYICAL UNIVERSITY ANANTHAPUR,

ANANTHAPURAM.

For partial fulfilment of the requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

A. VARSHITHA	-20781A1205
B. PRAVEEN KUMAR REDDY	-20781A1209
B. KISHORE BABU	-20781A1210
D. MADHURI	-20781A1211
K. SAI TARUN YADAV	-20781A1222

Under the Esteemed Guidance of

Mr. P. NANDA KUMAR, M.E., (Ph. D)

Assistant Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

SRI VENKATESWARA COLLEGE OF ENGINEERING & TECHNOLOGY

Affiliated to JNTUA, Anantapur, Approved by AICTE, New Delhi, Accredited by NBA, New Delhi & NAAC with A+ Grade, Bangalore. An ISO 9001-2000 Certified Institution), R.V.S NAGAR, CHITTOOR-517127

2023 - 2024

SRI VENKATESWARACOLLEGE OF ENGINEERING AND TECHNOLOGY

(Affiliated to JNTUA, Anantapur, Approved by AICTE, New Delhi, Accredited by National Board of Accreditation (NBA), New Delhi & NAAC, Bangalore. An ISO 9001:2000 Certified Institution), RVS NAGAR, CHITTOOR -517127.

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that this project entitled "**CARDSAFE FORENSICS - MACHINE LEARNING IN COMBATING FINANCIAL DECEPTION**" is a bonafide record of the work done by **A.VARSHITHA (20781A1205), B.PRAVEEN KUMAR REDDY (20781A1209),B.KISHOREBABU (20781A1210), D.MADHURI (20781A1211), K.SAI TARUN YADAV(20781A1222)** in partial fulfilment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY** in **INFORMATION TECHNOLOGY** to Jawaharlal Nehru Technological University, Anantapur during the academic year 2023-2024.

Signature of Project Guide	**Signature of the HOD**
Mr. P. NANDA KUMAR M.E., (Ph. D)	Mr. S.R. RAJKUMAR B.E., M. Tech
Assistant Professor	Head of the Department
Department of Information Technology	Department of Information Technology
SVCET	SVCET

Date of Viva-Voce: _____

Internal Examiner **External Examiner**

SRI VENKATESWARACOLLEGE OF ENGINEERING AND TECHNOLOGY

(Affiliated to JNTUA, Anantapur, Approved by AICTE, New Delhi, Accredited by National Board of Accreditation (NBA), New Delhi & NAAC, Bangalore. An ISO 9001:2000 Certified Institution), RVS NAGAR, CHITTOOR -517127.

DEPARTMENT OF INFORMATION TECHNOLOGY

DECLARATION

We hereby declare that the project entitled "**CARDSAFE FORENSICS - MACHINE LEARNING IN COMBATING FINANCIAL DECEPTIONS"** submitted by us under the guidance of Mr P NANDA KUMAR, M.E., (Ph. D), in the Department of INFORMATION TECHNOLOGY in the fulfilment of award of degree of **BACHELOR OF TECHNOLOGY** program at **SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY,** in the **INFORMATION TECHNOLOGY.**

Finally, we declare that the report has not been submitted to any other Institution/University for any other degree

Name of the Student	Roll No.	Signature
A. VARSHITHA	-20781A1205	
B. PRAVEEN KUMAR REDDY	-20781A1209	
B. KISHORE BABU	-20781A1210	
D. MADHURI	-20781A1211	
K. SAI TARUN YADAV	-20781A1222	

ACKNOWLEDGEMENT

We feel great pleasure while submitting the mini project report titled as "**CARDSAFE FORENSICS - MACHINE LEARNING IN COMBATING FINANCIAL DECEPTIONS".**

Heartfelt thanks to Sri **R. VENKATASWAMI, CHAIRMAN** and Sri **R.V. SRINIVAS, VICE-CHAIRMAN** of the Institute for providing education in his esteemed institution. We express our gratitude and esteemed regards to our beloved **PRINCIPAL Dr. Matam Mohan Babu, M. Tech, Ph.D.,** for providing facilities and encouragement throughout completing the project successfully.

We take great pleasure in conveying our sincere thanks to **Mr S R RAJKUMAR** Head of the Department- Information Technology, for his valuable support in completing our project successfully.

We express our gratitude and esteemed regards to our project guide **Mr P NANDA KUMAR**, Assistant Professor in Department of Information Technology, for providing us valuable inspiration in carrying out our project work and his constant support and encouragement enabled successfully. to complete this work.

We convey our thanks to our beloved parents and our faculty who helped us directly and indirectly in bringing out this project successfully.

Finally, we express our thanks to all our team member and friends who encouraged us a lot during our project work.

Table of Content

LIST OF FIGURES

ABSTRACT

Credit card fraud is a significant problem, with billions of dollars lost each year. Machine learning can be used to detect credit card fraud by identifying patterns that are indicative of fraudulent transactions. Credit card fraud refers to the physical loss of a credit card or the loss of sensitive credit card information. Many machine learning algorithms can be used for detection. This project proposes to develop a machine-learning model to detect credit card fraud. The model will be trained on a dataset of historical credit card transactions and evaluated on a holdout dataset of unseen transactions. Credit card fraud detection involves identifying unauthorized or fraudulent transactions, typically resulting from either the physical loss of a credit card or the compromise of sensitive credit card information. To combat this, a machine learning model can be developed, leveraging various algorithms to analyze transaction data and detect patterns indicative of fraudulent activity. These models utilize features such as transaction amounts, frequency, location, and user behavior to identify anomalies and flag potentially fraudulent transactions in real-time, helping financial institutions and users mitigate losses and safeguard against fraudulent activities.

Keywords: Credit Card Fraud Detection, Fraud Detection, Fraudulent Transactions, K- Nearest Neighbors, Support Vector Machine, Logistic Regression, Decision Tree.

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The main aim of this project is the detection of fraudulent credit card transactions, as it is essential to figure out the fraudulent transactions so that customers do not get charged for the purchase of products that they did not buy. Fraudulent Credit card transactions will be detected with multiple ML techniques. Then, a comparison will be made between the outcomes and results of each method to find the best and most suited model for detecting fraudulent credit card transactions; graphs and numbers will also be provided. In addition, it explores previous literature and different techniques used to distinguish Fraud within a dataset.

As financial transactions migrate to online platforms and digital mediums, the methods employed by fraudsters have become increasingly sophisticated, necessitating innovative approaches to combat them effectively. Amidst this backdrop, the integration of machine learning into forensic analysis emerges as a potent tool in the fight against financial deceptions.

CARDSAFE Forensics represents a ground breaking initiative aimed at leveraging the power of machine learning to detect, prevent, and mitigate financial fraud. Focused primarily on safeguarding card-based transactions, this project stands at the forefront of technological innovation, poised to revolutionize the way financial institutions, businesses, and consumers address fraudulent activities

Credit card companies must distinguish fraudulent from non-fraudulent transactions so that their customers' accounts will not get affected and charged for products they did not buy (Maniraj et al., 2019).

Many financial Companies and institutions lose massive amounts of money because of Fraud and fraudsters that are seeking different approaches continuously to violate the

rules and commit illegal actions; therefore, systems of fraud detection are essential for all banks that issue credit cards to decrease their losses (Zareapoor et al., 2012). Multiple methods are used to detect fraudulent behaviours, such as Neural Networks (N.N.), Decision Trees, K-nearest neighbour algorithms, and Support Vector Machines (SVM). Those ML methods can be applied independently or collectively with ensemble or meta-learning techniques to develop classifiers (Zareapoor et al., 2012).

1.2 OBJECTIVES OF THE PROJECT

The primary objective of CARDSAFE FORENSICS is to develop and implement advanced machine learning algorithms capable of identifying patterns, anomalies, and indicators of fraudulent behavior within card transactions. By harnessing the vast troves of data generated by financial transactions, these algorithms will discern subtle deviations from normal behavior, enabling swift and accurate detection of fraudulent activities.

This objective arises from the pressing need to counteract the rising tide of financial fraud in an increasingly digitalized world. Traditional methods of fraud detection often struggle to keep pace with the sophistication of modern fraud tactics, highlighting the necessity for innovative solutions grounded in cutting-edge technology.

This objective aligns with the overarching goal of enhancing the security and integrity of financial systems, thereby safeguarding the interests of financial institutions, businesses, and consumers alike. Through the proactive identification of fraudulent behavior, CARDSAFE FORENSICS aims to mitigate financial losses, protect sensitive information, and preserve trust in the digital economy.

1.3 PROBLEM STATEMENT

Credit card fraud is a significant concern for financial institutions and consumers alike. Traditional methods of fraud detection often fall short in identifying fraudulent transactions accurately and efficiently. With the increasing volume and sophistication

of fraudulent activities, there is a growing need for more advanced and automated solutions to detect and prevent fraud.

The problem statement for "CardSafe Forensics — Machine Learning in Combatting Financial Deceptions" involves developing a robust and effective system that can accurately identify fraudulent transactions in real-time. This system should be capable of analyzing large volumes of transaction data and detecting patterns indicative of fraudulent behavior.

Key objectives of the project include:

1. Building a machine learning model capable of distinguishing between legitimate and fraudulent credit card transactions with high accuracy.

2. Developing algorithms that can handle imbalanced datasets, as fraudulent transactions are typically rare compared to legitimate ones.

3. Implementing real-time monitoring and detection capabilities to identify fraudulent transactions as soon as they occur.

4. Ensuring the system's scalability and efficiency to handle large volumes of transaction data without compromising performance.

5. Incorporating interpretability and explainability features to understand the factors contributing to the model's decisions and improve transparency

CHAPTER 2

LITERATURE REVIEW

2.1 INTRODUCTION:

Financial fraud and deception pose significant threats to individuals, businesses, and financial institutions worldwide. With the increasing digitization of financial transactions, perpetrators of fraud have adapted their methods, making detection and prevention more challenging. In response to this evolving landscape, the integration of machine learning techniques into forensic analysis has emerged as a promising approach to combat financial deceptions effectively.

This literature review aims to explore the existing research on the utilization of machine learning in the context of CardSafe Forensics, a hypothetical system designed to detect and prevent fraudulent activities related to card-based transactions. By synthesizing insights from previous studies, this review seeks to identify key trends, challenges, and opportunities in the application of machine learning algorithms for enhancing the security and integrity of financial transactions.

The review begins by providing an overview of the current state of financial fraud and the role of machine learning in addressing this issue. It then delves into specific methodologies and algorithms employed in CardSafe Forensics, examining their strengths and limitations. Furthermore, the review discusses relevant case studies and empirical evidence to illustrate the effectiveness of machine learning-based approaches in detecting various forms of financial deception.

The literature review aims to contribute to the growing body of knowledge in the field of financial forensics and machine learning. By synthesizing insights from diverse sources, it seeks to inform future research directions and guide the development of more robust and efficient tools for combating financial fraud and deception.

Zareapoor and his research team used multiple techniques to determine the best performing model for detecting fraudulent transactions, which was established using the Accuracy of the model, the speed of detection and the cost. The models used were Neural Network, Bayesian Network, SVM, KNN and. The comparison table in the research paper showed that the Bayesian Network was high-speed in finding fraudulent transactions with high Accuracy. The N.N. performed well, as the detection was fast, with a medium accuracy. KNN's speed was good with medium Accuracy, and finally, SVM scored one of the lower scores, as the speed was low, and the Accuracy was medium. As for the cost, all models built were expensive (Zareapoor et al.,3 2012).[1]

The model used by Alenzi and Aljehane to detect Fraud in credit cards was Logistic Regression. Their model scored 97.2% in Accuracy, 97% sensitivity and 2.8% Error Rate. A comparison was performed between their model and two other classifiers, which are3 They were voting Classifier and KNN. V.C. scored 90% in Accuracy, 88% sensitivity and 10% error rate, as for KNN where k = 1:10, the Accuracy of the model was 93%, the sensitivity 94%and 7% for the error rate (Alenzi Aljehane, 2020).[2]

Maniraj's team built a model to recognize if any new transaction is Fraud or non-fraud. Their goal was to get 100% in detecting fraudulent transactions and try to minimize the incorrectly classified fraud instances. Their model has performed well as they got 99.7% of the fraudulent transactions (Maniraj et al., 2019).[3]

The classification approach used by Dheepa and Dhanapal was the behaviour-based classification approach, using a Support Vector Machine, where the behavioural patterns of the customers were analyzed to distinguish credit card fraud, such as the amount, date, time, place, and frequency of card usage.

The Accuracy achieved by their approach was more than 80% (Dheepa Dhanapal, 2012).[4]

Mailini and Pushpa proposed using KNN and Outlier detection in identifying credit card fraud. After performing their model oversampled data, the authors found that the most suitable method for detecting and determining target instance anomaly is KNN, which showed that it is most suited to detecting Fraud with memory limitation. As for Outlier detection, the computation and memory required for credit card fraud detections much less in addition to working faster and better in large online datasets. However, their work and results showed that KNN was more accurate and efficient (Malini Pushpa,2017).[5]

Maes and his team proposed using Bayesian and Neural Networks to detect credit card fraud. Their results showed that Bayesian performance is 8% more effective in detecting Fraud than ANN, meaning BBN sometimes detects 8% more fraudulent transactions. In addition to the Learning times, ANN can go up to several hours, whereas BBN takes only 20 minutes (Maes et al., 2002).[6]

Jain's team used several ML techniques to distinguish credit card fraud; three of 4 them are SVM, ANN and KNN. Then, to compare the outcome of each model, the calculated the true positive (T.P.), false Negative (F.N.), false positive (F.P.), and true negative (T.N.) generated. ANN scored 99.71% accuracy, 99.68% precision, and 0.12% false alarm rate. SVM accuracy is 94.65%, 85.45% for the precision, and 5.2% false alarm rate. Moreover, finally, the Accuracy of KNN is 97.15%, the precision is 96.84%, and the false alarm rate is 2.88% (Jain et al., 2019).[7]

Dighe and his team used KNN, Logistic Regression and Neural Networks, multi-layer perceptron and Decision Tree in their work, then evaluated the results regarding numerous accuracy metrics. Of all the models created, the best performing one is KNN, which scored 99.13%, then in second place performing model at 96.40% and in last place is logistic Regression with 96.27% (Dighe et al., 2018).[8]

Sahin and Duman's study on detecting credit card fraud using Support Vector Machine (SVM) methods is significant for its high accuracy rates in both training and testing

phases. Their research employed four different SVM techniques: Support Vector Machine with RBF (Radial Basis Function), Polynomial, Sigmoid, and Linear Kernel.

The training phase achieved an impressive accuracy score of 99.87%, indicating that the models were highly effective in learning from the provided data. This level of accuracy suggests that the SVM models were able to accurately classify the majority of instances in the training dataset, demonstrating their capability to capture complex patterns inherent in credit card transaction data.

However, the true test of a model's performance lies in its ability to generalize to new, unseen data. In the testing phase, the SVM models maintained a commendable accuracy score of 83.02%. While this is slightly lower than the training accuracy, it still indicates a strong performance in correctly identifying fraudulent transactions in real-world scenarios.

The high accuracy rates achieved by Sahin and Duman's SVM models underscore the potential of machine learning techniques, particularly SVM, in fraud detection applications. Such systems hold promise for financial institutions seeking robust solutions to combat the growing threat of fraudulent activities in electronic transactions.

CHAPTER 3

SYSTEM STUDY

3.1 EXISTING SYSTEM

In existing System, research about a case study involving credit card detection, where data normalization is applied before Cluster Analysis and with results obtained from the use of Cluster Analysis and Artificial Neural Networks on fraud detection has shown that by clustering attributes neuronal inputs can be minimized. And promising results can be obtained by using normalized data and data should be MLP trained.

This research was based on unsupervised learning. Significance of this paper was to find new methods for fraud detection and to increase the accuracy of results. The data set for this paper is based on real life transactional data by a large European company and personal details in data is kept confidential. Accuracy of an algorithm is around 50%. Significance of this paper was to find an algorithm and to reduce the cost measure. The result obtained was by 23% and the algorithm they find was Bayes minimum risk.

DISADVANTAGES

- In this paper a new collative comparison measure that reasonably represents the gains and losses due to fraud detection is proposed.
- A cost sensitive method which is based on Bayes minimum risk is presented using the proposed cost measure.

3.2 PROPOSED SYSTEM

In proposed System, we are applying Random Forest algorithm, Bernoulli Naive Bayes Model and Logistic Regression algorithm for classification of the credit card dataset. Random Forest is an algorithm for classification and regression. Summarily, it is a collection of decision tree classifiers. Random forest has advantage over decision tree as it corrects the habit of over fitting to their training set. A subset of the training set is sampled randomly so that to train each individual tree and then a decision tree is built,

each node then splits on a feature selected from a random subset of the full feature set. Even for large data sets with many features and data instances training is extremely fast in random forest and because each tree is trained independently of the others. The Random Forest algorithm has been found to provide a good estimate of the generalization error and to be resistant to over fitting. We propose a Machine learning model to detect fraudulent credit card activities in online financial transactions. Analysing fake transactions manually is impracticable due to vast amounts of data and its complexity. However, adequately given informative features, could make it is possible using Machine Learning. This hypothesis will be explored in the paper. To classify fraudulent and legitimate credit card transaction by supervised learning Algorithm such as Random Forest. To help us to get awareness about the fraudulent and without loss of any financially.

ADVANTAGES

- Random forest ranks the importance of variables in a regression or classification problem in a natural way can be done by Random Forest.
- The ‘amount’ feature is the transaction amount. Feature ‘class’ is the target class for the binary classification and it takes value 1 for positive case (fraud) and 0 for negative case (not fraud).

CHAPTER 4
METHODOLOGY

The methodology for "CardSafe Forensics Using Machine Learning" involves several sequential steps to develop a robust and effective fraud detection system. Initially, a comprehensive dataset containing historical credit card transaction data, encompassing both legitimate and fraudulent transactions, is collected. Following this, thorough data preprocessing techniques are applied, including exploratory data analysis (EDA) to understand the data distribution, handling missing values, normalizing numerical features, and encoding categorical variables. Feature engineering is then employed to extract relevant features from the dataset and 10abelled10lly create new ones to improve model performance.

Moving forward, appropriate machine learning algorithms are selected for binary classification tasks, considering factors such as logistic regression, decision trees, random forests, support vector machines (SVM), and ensemble methods. Deep learning architectures like artificial neural networks (ANNs) or convolutional neural networks (CNNs) may also be explored for complex datasets. These models are trained on a split dataset, with hyperparameters tuned using techniques like cross-validation or grid search to prevent overfitting. Model performance is rigorously evaluated using metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic (ROC) curve, with considerations for the dataset's imbalance.

Upon selecting the most suitable model, it is deployed into a production environment capable of handling real-time credit card transactions. Integration with existing banking infrastructure and workflows is essential for seamless operation and decision-making. Additionally, monitoring mechanisms are implemented to track the model's performance over time and detect any drift or degradation.

Model interpretability techniques, such as feature importance analysis or SHAP (Shapley Additive explanations) values, are utilized to understand the factors contributing to fraud detection and ensure transparency in the model's decisions. Overall' this methodology aims to develop a reliable and transparent credit card fraud detection system that effectively mitigates financial losses and safeguards the interests of both financial institutions and cardholders.

4.1 DATASET DESCRPTION

The first figure below shows the structure of the dataset where all attributes are shown, with their type, in addition to a glimpse of the variables within each Attribute; as shown at the end of the figure, the Class type is integer, which I needed to change to factor and identify the 0 as Not Fraud and the one as Fraud to ease the process of creating the modeland obtain visualizations.The second figure shows the class distribution; the red bar, which contains 284,315variables, represents the non-fraudulent transactions, and the blue bar, with 492 variables, represents the fraudulent transactions.

"CardSafe Forensics" employs advanced machine learning techniques to combat financial deceptions, particularly those related to card fraud and financial crimes. The dataset used in this project comprises a diverse range of financial transactions, including credit card transactions, ATM withdrawals, online purchases, and more. Each transaction entry includes various features such as transaction amount, location, timestamp, card type, merchant information, and any flagged indicators of suspicious activity. Additionally, the dataset encompasses both genuine and fraudulent transactions, providing a comprehensive training ground for the machine learning algorithms to learn and distinguish between legitimate and fraudulent activities

Through the analysis of this dataset, CardSafe Forensics aims to develop robust predictive models capable of accurately detecting and preventing financial deceptions

in real-time, thus safeguarding financial institutions and consumers alike from potential losses and fraudulent activities.

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
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	2.848070e+05
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15
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	1.098632e+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	-1.343407e+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	-6.430976e-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	-5.142873e-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	5.971390e-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	1.559499e+01

Figure 1: Data Description

4.2 ANALYSIS

In order to understand the process, programmers do something called ‘analysis. ‘ Analysis is defined as breaking something down into its inter-related components in order to understand it. The object of the analysis can be raw data, a business processor learning how to program.

```
RangeIndex: 284807 entries, 0 to 284807
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)
```

Figure 2 : Data Structure

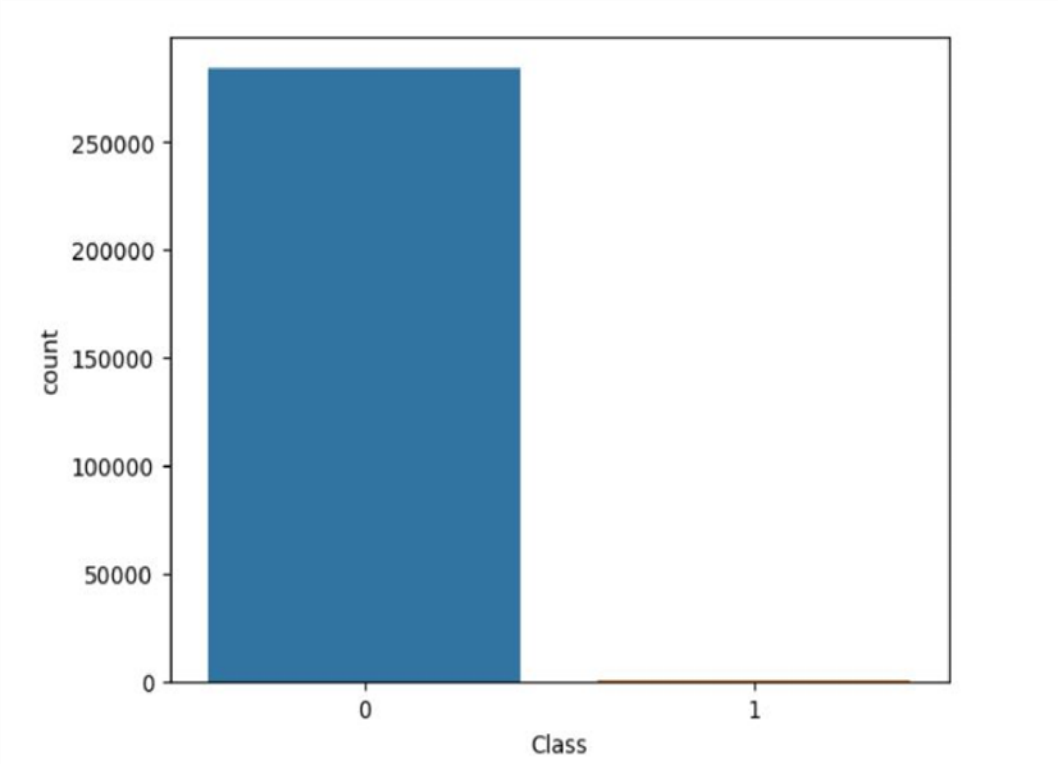


Figure 3: Class Count

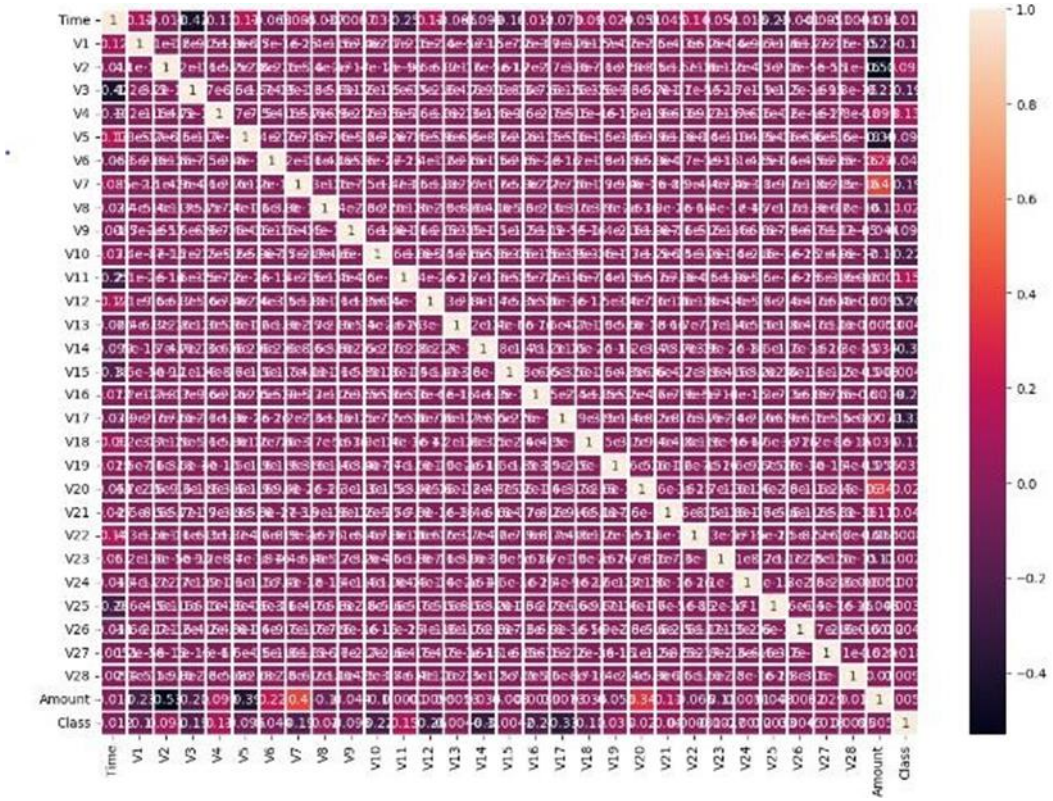


Figure 4: Data Correlation

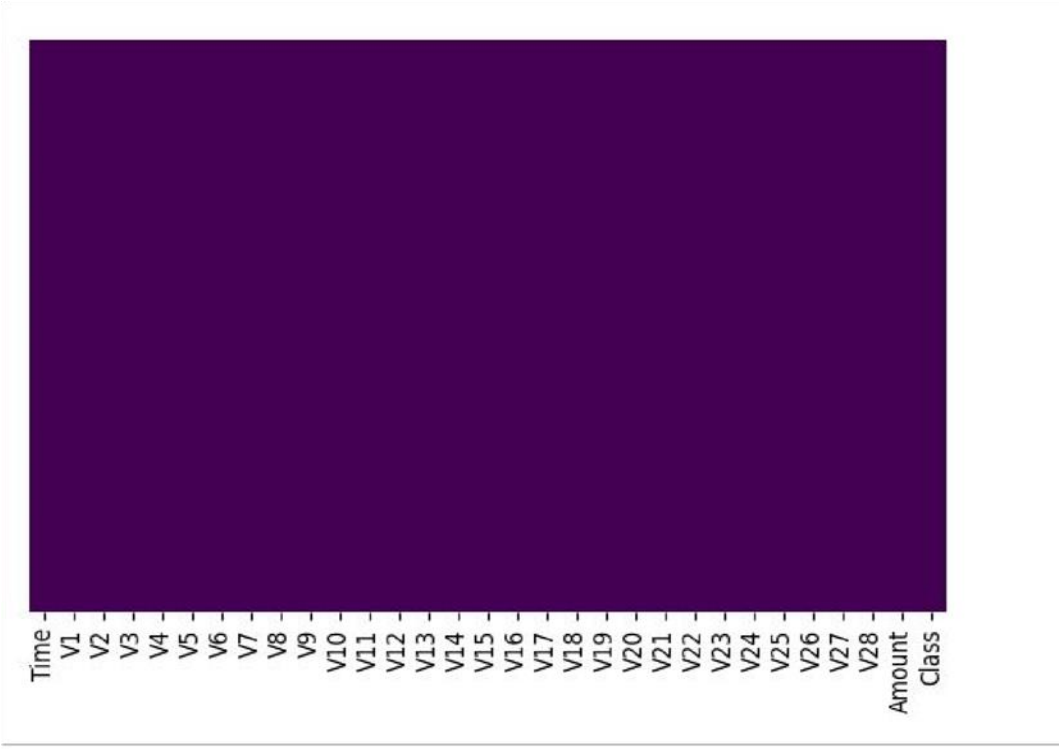


Figure 5: Null data

CHAPTER 5

SYSTEM REQUIREMENTS

5.1 HARDWARE REQUIREMENTS

➢**Processor (CPU):**

- Minimum: Intel Core i5 or equivalent.

- Recommended: Intel Core i7 or higher, or equivalent in AMD processors.

➢**Memory (RAM):- Minimum:** 8GB.

- Recommended: 16GB or more for handling larger datasets or complex computations.

➢**Storage:**

- SSD (Solid State Drive) for faster data access.

- Minimum: 256GB.

- Recommended: 512GB or more, depending on the size of your datasets.

➢**Graphics Card (GPU):**

- For basic machine learning tasks: Integrated GPU may suffice.

- For deep learning and more intensive tasks: Dedicated NVIDIA GPU (preferably with CUDA support) such as the GTX 1050 Ti or better.

5.2 SOFTWARE REQUIREMENTS

- **Operating System:**
  - Windows 10 or later, macOS, or a Linux distribution (e.g., Ubuntu).
  - 64-bit OS is recommended for better performance
- **Python:**
  - Python 3.7 or later.
  - Installed as part of the Anaconda distribution.
- **Collab:**
  - Google Collab, which includes Python, Jupyter Notebook, and many useful data science libraries**.**

- **Machine Learning Libraries:**
  - TensorFlow, PyTorch, scikit-learn, etc. (Install via Anaconda or pip).
  - CUDA Toolkit and cuDNN for GPU support (if using NVIDIA GPU).

- **Additional Tools and Libraries:**
  - Libraries for data manipulation and visualization (e.g., Pandas, NumPy, Matplotlib).

5.3 SOFTWARE FEATURES

➢PYTHON

Python is a Programming language that is interpreted, object-oriented, and considered to be high-level too. Python is one of the easiest yet most useful programming languages which is widely used in the software industry. People use Python for Competitive Programming, Web Development, and creating software. Due to its easiest syntax, it is recommended for beginners who are new to the software engineering field. Its demand is growing at a very rapid pace due to its vast use cases in Modern Technological fields like Data Science, Machine learning, and Automation Tasks. For many years now, it has been ranked among the top Programming languages.

 # Python Program to print Hello World

print(“Hello World! I Don’t Give a Bug”)

```
# Python Program to print Hello World
print("Hello World! I Don't Give a Bug")
```

Key Features of Python

- Python is known for its ease of learning, readability, and maintainability. It boasts a broad standard library that supports various platforms, making it highly portable. Python's interactive mode allows for real-time testing and debugging, while its extendable nature enables integration with low-level modules for enhanced functionality.

Python Web Frameworks

- Python offers robust web frameworks like Flask and Django, which streamline web application development by providing URL routing, HTML templating, database integration, and security features. These frameworks empower developers to build scalable and maintainable web applications efficiently.
    - Python's versatility, simplicity, and extensive ecosystem make it a preferred choice for developers across various domains. Whether it's scripting, web development, data science, or machine learning, Python's rich features and supportive community continue to drive its widespread adoption and innovation in the programming landscape.

➢ GOOGLE COLAB

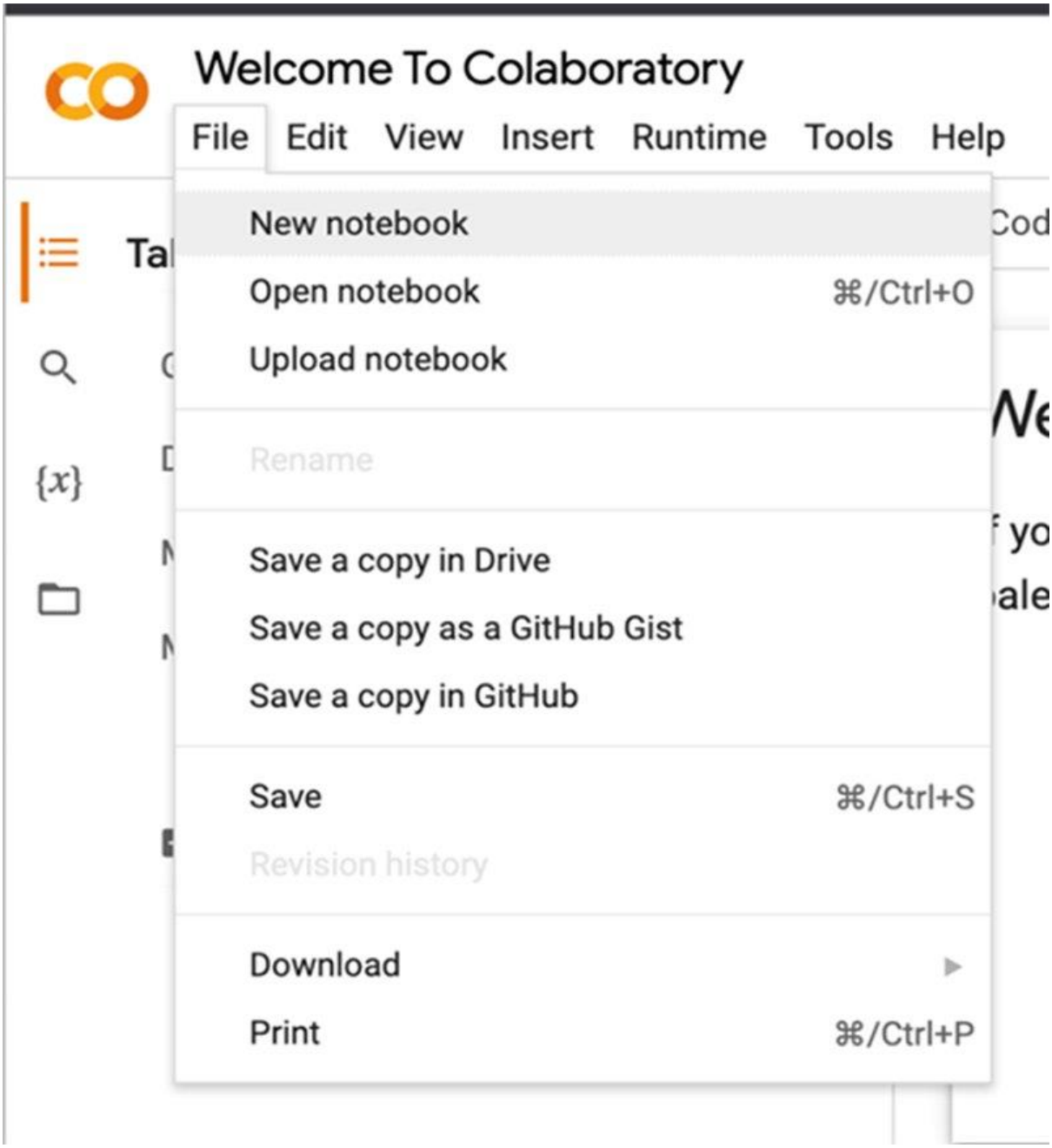
- Google Collaboratory, or Colab, is an as-a-service version of Jupyter Notebook that enables you to write and execute Python code through your browser.
- Jupyter Notebook is a free, open-source creation from the Jupyter Project. A Jupyter notebook is like an interactive laboratory notebook that includes not just notes and data, but also code that can manipulate the data. The code can be executed within the notebook, which, in turn, can capture the code output. Applications such as MATLAB and Mathematica pioneered this model, but unlike those applications, Jupyter is a browser-based web application.

- Google Colab is built around Project Jupyter code and hosts Jupyter notebooks without requiring any local software installation. But while Jupyter notebooks support multiple languages, including Python, Julia and R, Colab currently only supports Python.
- Colab notebooks are stored in a Google Drive account and can be shared with other users, similar to other Google Drive files. The notebooks also include an autosave feature, but they do not support simultaneous editing, so collaboration must be serial rather than parallel.
- Colab is free, but has limitations. There are some code types that are forbidden, such as media serving and crypto mining. Available resources are also limited and vary depending on demand, though Google Colab offers a pro version with more reliable resourcing. There are other cloud services based on Jupyter Notebook, including Azure Notebooks from Microsoft and SageMaker Notebooks from Amazon.

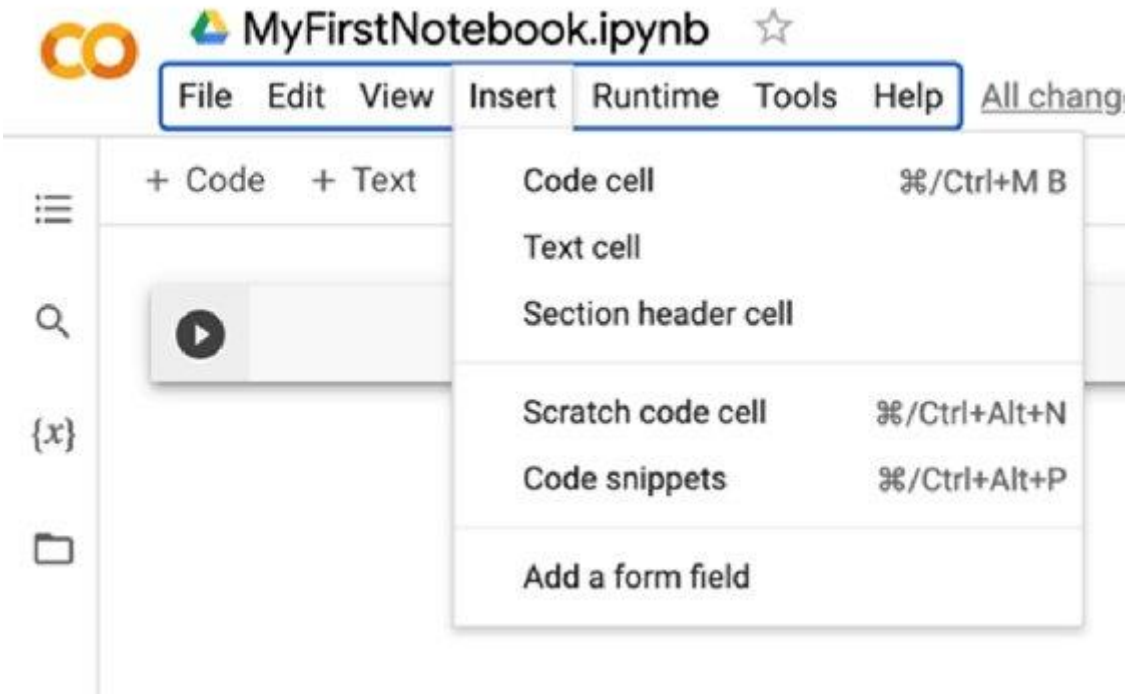
The benefits of Google Colab

- Enterprise data analysts and analytics developers can use Colab to work through data analytics and manipulation problems in collaboration. They can write, execute, and revise core code in a tight loop, developing the documentation in Markdown format, LaTeX, or HTML as they go.
- Notebooks can include embedded images as part of the documentation or as generated output. In addition, you can copy finished analytics code, with documentation, into other platforms for production use once sufficiently tested and debugged.

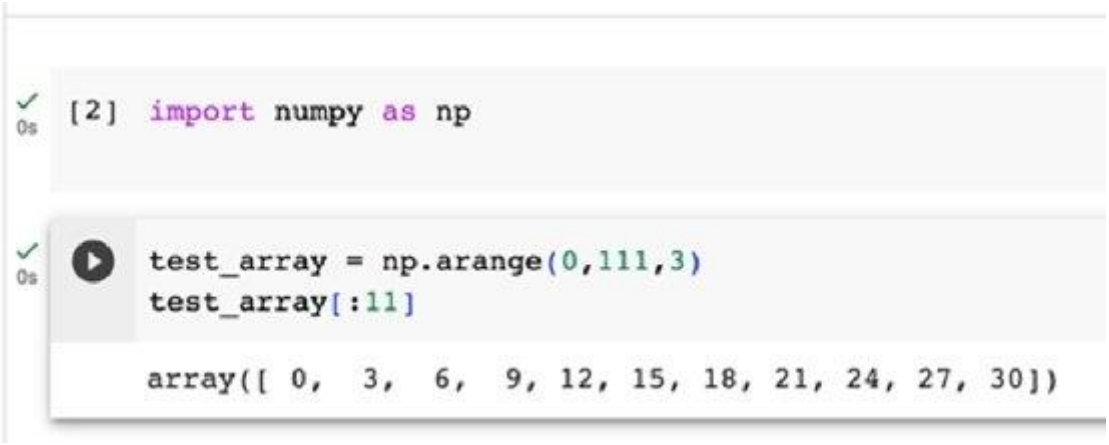
- Google Colab eliminates the need for complex configuration setup and installation, as it runs right in the browser. It also includes pre-installed Python libraries that require no setup to use.


Welcome To Colaboratory
File
Edit
View
Insert
Runtime
Tools
Help
New notebook
Open notebook
⌘/Ctrl+O
Upload notebook
Rename
Save a copy in Drive
Save a copy as a GitHub Gist
Save a copy in GitHub
Save
⌘/Ctrl+S
Revision history
Download
Print
⌘/Ctrl+P

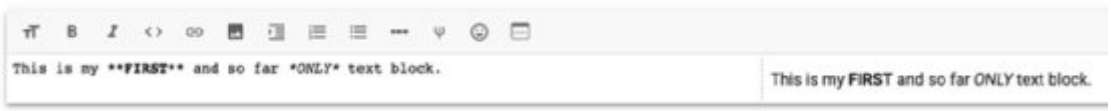
Step 1:

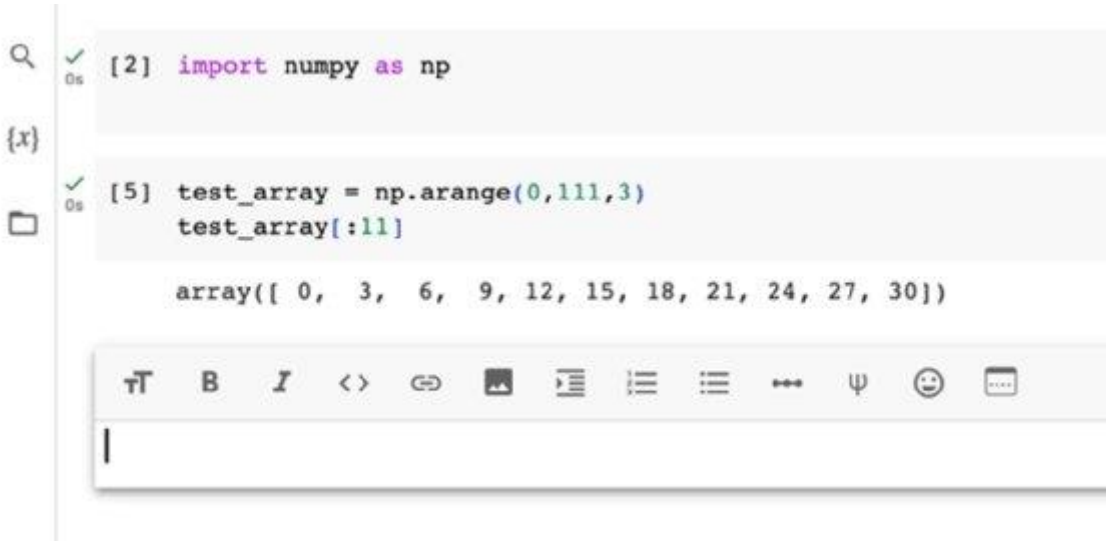

MyFirstNotebook.ipynb
File Edit View Insert Runtime Tools Help All chang
+ Code + Text
Code cell ⌘/Ctrl+M B
Text cell
Section header cell
Scratch code cell ⌘/Ctrl+Alt+N
Code snippets ⌘/Ctrl+Alt+P
Add a form field

Step 2:

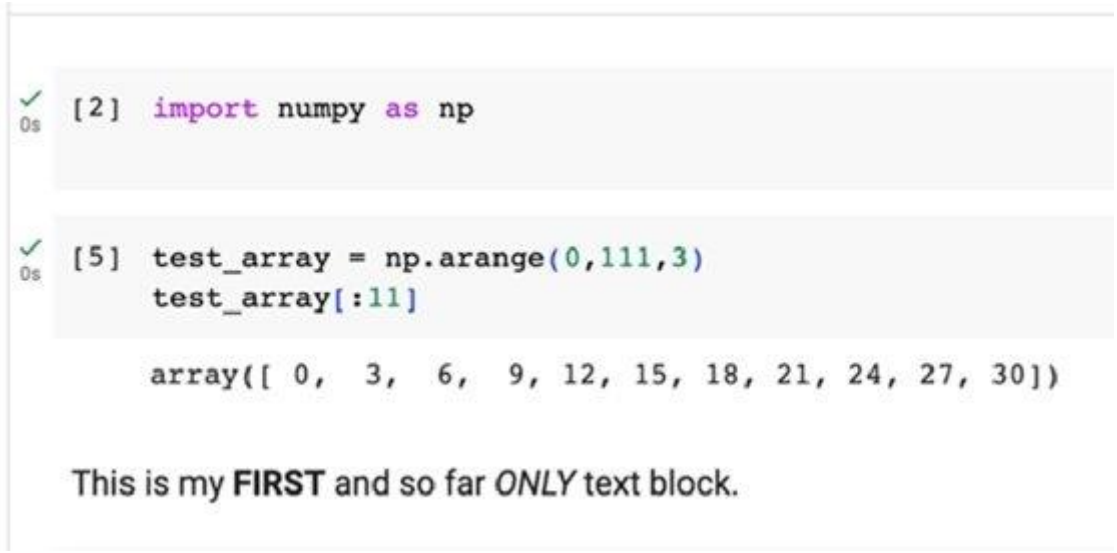

[2] import numpy as np
test_array = np.arange(0,111,3)
test_array[:11]
array([0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30])

Step 3:


This is my **FIRST** and so far *ONLY* text block.
This is my FIRST and so far ONLY text block.


[2] import numpy as np
[5] test_array = np.arange(0,111,3)
test_array[:11]
array([0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30])

Step 4:


[2] import numpy as np
0s
[5] test_array = np.arange(0,111,3)
test_array[:11]
0s
array([0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30])
This is my FIRST and so far ONLY text block.

Step 5:


from google.colab import drive
drive.mount('/my_drive')

Step 6:

➢ VISUAL STUDIO

Visual Studio Code (VS Code) is a free open-source text editor by Microsoft. VS Code is available for Windows, Linux, and macOS. Although the editor is relatively lightweight, it includes some powerful features that have made VS Code one of the most popular development environment tools in recent times.

➢ MACHINE LEARNING

- Machine Learning is a field of artificial intelligence (AI) that focuses on developing algorithms and techniques that enable computers to learn from data and make predictions or decisions without being explicitly programmed for each task. It involves creating models that can learn patterns and relationships from large datasets, allowing computers to perform tasks or make decisions based on that learned knowledge.
    - The primary goal of machine learning is to develop algorithms that can improve their performance over time as they are exposed to more data. This process of learning from data is often referred to as training, where the algorithm adjusts its parameters or structure to optimize its performance on a specific task**.**
    - There are several types of machine learning algorithms, including:

1.Supervised Learning: In supervised learning, the algorithm is trained on labelled data, where each example in the dataset is paired with a corresponding label or outcome. The algorithm learns to map inputs to outputs based on the provided examples, allowing it to make predictions on new, unseen data. Through exposure to these labelled examples, the algorithm learns to recognize patterns and relationships between inputs and outputs.

By leveraging this learned knowledge, the algorithm can then make predictions or classifications on new, unseen data. For instance, in a supervised learning task of

recognizing handwritten digits, the algorithm learns from a dataset where each digit image is labelled with its corresponding numerical value. Over time, the algorithm becomes proficient at recognizing digits and can accurately classify new handwritten digits it has not encountered before, demonstrating the effectiveness of supervised learning in teaching machines to perform tasks based on labelled examples.

2. Unsupervised Learning: Unsupervised learning involves training algorithms on unlabelled data, where the algorithm must discover patterns or structures in the data without explicit guidance. This type of learning is often used for tasks such as clustering similar data points together or dimensionality reduction.

The primary objective is to discern inherent patterns or structures within the dataset. This mode of learning is instrumental in tasks like clustering similar data points together or reducing the complexity of the dataset through techniques like dimensionality reduction. For instance, in clustering, the algorithm identifies groups of data points that exhibit similar characteristics, aiding in data segmentation or organization. Dimensionality reduction techniques, on the other hand, streamline complex datasets by extracting essential features, facilitating easier analysis or visualization. Unsupervised learning thus empowers algorithms to uncover insights and underlying structures within data autonomously, without the need for explicit supervision or labels.

3.Reinforcement Learning: Reinforcement learning is a type of learning where an agent learns to interact with an environment by performing actions and receiving feedback or rewards. The goal of the agent is to learn a policy that maximizes its cumulative reward over time, leading to behaviors that are optimal for the given task.

In this framework, the agent takes actions within the environment, which then responds with feedback in the form of rewards or penalties. The agent’s objective is to learn a strategy, known as a policy, that maximizes its cumulative reward over time.

Through iterative exploration and exploitation, the agent refines its policy, leading to behaviors that are increasingly optimized for the given task. Reinforcement learning finds applications in a myriad of domains, from training autonomous agents to play games like chess or Go to optimizing complex industrial processes. This learning paradigm mirrors the trial-and-error learning process observed in humans and animals, enabling machines to autonomously acquire skills and make decisions in dynamic environments.

APPLICATION OF MACHINE LEARNING

1.Predictive Analytics:

Predicting future outcomes or trends based on historical data is a fundamental task in various fields, including finance, marketing, and economics. This process, often referred to as time series forecasting, involves analyzing past data points to identify patterns, trends, and relationships that can be used to make predictions about future events. In contexts like sales forecasting or stock price prediction, historical data such as past sales figures or stock prices are analyzed using statistical methods or machine learning algorithms to generate forecasts. Techniques commonly employed for time series forecasting include autoregressive integrated moving average (ARIMA), exponential smoothing methods, and machine learning models like recurrent neural networks (RNNs) or long short-term memory (LSTM) networks. By leveraging historical data and advanced analytical techniques, organizations can make informed decisions and plan strategies to anticipate future outcomes and trends.

2. Image and Speech Recognition:

Image and speech recognition are pivotal technologies that enable machines to automatically identify objects, people, or speech patterns within images or audio recordings, respectively. In image recognition, algorithms analyze visual data to detect and classify objects, faces, scenes, and other relevant features present in images.

This technology finds wide applications in fields like computer vision, autonomous vehicles, medical imaging, and surveillance systems. Similarly, speech recognition involves analyzing audio data to transcribe spoken words or identify specific patterns in speech. This capability is crucial for applications such as virtual assistants, voice-controlled devices, speech-to-text systems, and automated customer service. Both image and speech recognition rely on advanced machine learning techniques, including convolutional neural networks (CNNs) for image recognition and recurrent neural networks (RNNs) for speech recognition, among others. As these technologies continue to evolve, they play an increasingly significant role in enhancing human-computer interaction and driving innovation across various industries.

3. Natural Language Processing:

Understanding and generating human language, enabling tasks such as translation, sentiment analysis, and chatbots.

Natural Language Processing (NLP) is a branch of artificial intelligence that focuses on enabling computers to understand, interpret, and generate human language. It encompasses a wide range of tasks, including language translation, sentiment analysis, text summarization, named entity recognition, and chatbots. NLP algorithms analyze and process textual data in order to extract meaningful information, derive insights, and perform various language-related tasks. For instance, in language translation, NLP systems translate text from one language to another by understanding the semantics and grammar of both languages. Sentiment analysis involves determining the sentiment or opinion expressed in a piece of text, which is valuable for understanding customer feedback, social media trends, and market sentiments. Chatbots utilize NLP techniques to interpret user input, generate appropriate responses, and engage in human-like conversations. With advancements in deep learning and natural language understanding, NLP continues to play a critical role in improving human-computer interaction, automating language-related tasks, and

unlocking new opportunities in areas such as customer service, healthcare, finance, and education.

4. Healthcare:

Assisting in medical diagnosis, personalized treatment recommendations, and drug discovery. Machine learning is used for disease diagnosis, personalized treatment plans, drug discovery, and patient monitoring. It can analyze medical images, such as X-rays and MRIs, to detect abnormalities and assist radiologists in diagnosis.

In healthcare, machine learning plays a pivotal role in revolutionizing various aspects of medical diagnosis, treatment, and drug discovery. By leveraging vast amounts of medical data, machine learning algorithms can assist in disease diagnosis by analyzing patient symptoms, medical history, and diagnostic tests to provide accurate and timely diagnoses. Additionally, machine learning enables personalized treatment recommendations by considering individual patient characteristics, genetic profiles, and treatment responses to tailor interventions for optimal outcomes. In drug discovery, machine learning techniques expedite the identification of potential drug candidates by analyzing molecular structures, biological interactions, and clinical trial data to predict drug efficacy and safety profiles.

Moreover, machine learning facilitates patient monitoring by analyzing real-time health data from wearable devices, electronic health records, and medical imaging to detect abnormalities, monitor disease progression, and intervene when necessary. Particularly in medical imaging, machine learning algorithms can analyze images such as X-rays and MRIs to assist radiologists in detecting abnormalities, improving diagnostic accuracy, and expediting patient care. Overall, machine learning empowers healthcare professionals with powerful tools to enhance medical decision-making, improve patient outcomes, and advance medical research and innovation.

5. Autonomous Vehicles:

Enabling vehicles to perceive their environment, make decisions, and navigate safely without human intervention.

They process data from sensors such as cameras, lidar, and radar to detect obstacles, pedestrians, and road signs.

Overall, machine learning plays a crucial role in enabling computers to perform complex tasks and make decisions in ways that mimic human intelligence, revolutionizing various industries and domains.

CONCEPT OF MACHINE LEARNING

- At its core, machine learning is all about enabling computers to learn from data. Traditionally, in programming, humans write explicit instructions for computers to follow. However, with machine learning, we take a different approach. Instead of providing explicit instructions, we provide the computer with a set of data and let it learn from that data to improve its performance on a specific task.
- Imagine you're trying to teach a computer to recognize images of cats. In traditional programming, you might have to manually define what features make up a cat, such as pointy ears, whiskers, and a tail. However, with machine learning, you would feed the computer a large dataset of images labeled as either "cat" or "not cat." The machine learning algorithm then learns patterns from these images, without being explicitly told what those patterns are. Through this process, the algorithm gradually improves its ability to correctly identify cats in new, unseen images.

5.4 DATAFLOW DIAGRAM

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

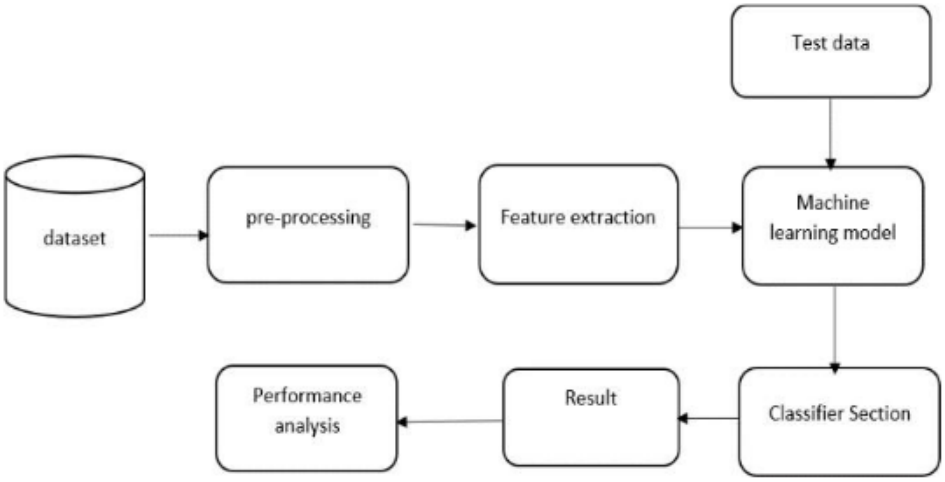


Figure 6. System architecture

They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually "say" things that would be hard to explain in words, and they work for both technical and nontechnical audiences, from developer to CEO.

That is why DFDs remain so popular after all these years. While they work well for data flow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

5.5 IMPLEMENTATION TOOLS

For implementing "CardSafe Forensics — Machine Learning and Combating Financial Deceptions," several tools and frameworks are commonly used. Here's a list of some popular ones:

1.Python:

Python stands as a preeminent programming language in the realm of machine learning and data science, owing to its rich ecosystem of libraries and straightforward syntax. Its popularity stems from the accessibility and versatility it offers to practitioners, enabling them to seamlessly implement complex algorithms and workflows. Python boasts an extensive array of libraries such as NumPy, pandas, and matplotlib, which provide robust support for numerical computing, data manipulation, and visualization, respectively. Additionally, frameworks like TensorFlow, PyTorch, and scikit-learn augment Python's capabilities by offering specialized tools for deep learning, neural network construction, and machine learning model development. The language's user-friendly nature, coupled with its vast community support and continuous development, solidifies Python's position as the language of choice for tackling machine learning and data science challenges across various domains and industries.

2. Scikit-learn:

Scikit-learn stands out as a cornerstone in the Python ecosystem for machine learning, renowned for its simplicity and effectiveness in facilitating data mining and analysis tasks. Boasting a comprehensive suite of algorithms, it empowers users with a diverse array of tools for classification tasks, encompassing logistic regression, decision trees, random forests, and support vector machines (SVM). Its intuitive interface and well-documented API streamline the process of model building, evaluation, and deployment, making it a preferred choice for both beginners and seasoned practitioners alike. With its commitment to simplicity and efficiency, scikit-learn remains a go-to library for harnessing the power of machine learning in Python across various domains and applications.

3. TensorFlow / Keras:

TensorFlow, an open-source machine learning framework created by Google, serves as a robust foundation for developing various deep learning models.

It provides a flexible ecosystem for constructing, training, and deploying neural networks across different platforms. Keras, on the other hand, operates as a user-friendly interface built on top of TensorFlow, simplifying the process of building neural networks by offering high-level abstractions. This abstraction layer enables developers to rapidly prototype neural network architectures without delving into the intricate details of TensorFlow's core functionalities. By seamlessly integrating with TensorFlow, Keras enhances accessibility and productivity, making it an ideal choice for both beginners and experienced practitioners in the field of deep learning.

4. PyTorch:

PyTorch stands out as an open-source machine learning framework renowned for its efficient tensor computation capabilities, empowered by robust GPU acceleration. It offers a versatile platform for developing deep neural networks, underpinned by a tape-based autograd system. This system enables automatic differentiation of computational graphs, facilitating seamless and efficient gradient computation crucial for training complex neural networks.

PyTorch's dynamic computation graph paradigm allows for flexible and intuitive model building, fostering rapid prototyping and experimentation. Its integration with Python further enhances its usability, making it a preferred choice for both researchers and practitioners in the machine learning community.

5. Pandas:

Pandas is a versatile and powerful Python library designed for effortless data manipulation and analysis, making it exceptionally well-suited for managing tabular data like credit card transaction datasets. Its intuitive and expressive syntax allows users to easily load data from various sources, including CSV files, Excel spreadsheets, SQL databases, and more, into DataFrame objects—a primary data structure within Pandas. With DataFrame, users can efficiently clean, transform, filter, and analyze data, thanks to its extensive array of built-in functions and methods. Whether it's exploring trends,

detecting anomalies, or conducting statistical analyses, Pandas provides a comprehensive toolkit that streamlines the entire data workflow, from initial data loading to final insights extraction. Its seamless integration with other libraries like NumPy and Matplotlib further enhances its capabilities, making Pandas a go-to solution for data professionals and enthusiasts alike.

6. NumPy:

NumPy stands as a cornerstone in the Python ecosystem for scientific computing, offering indispensable tools for numerical analysis, data manipulation, and computation-heavy tasks. Its core functionality revolves around the efficient handling of multi-dimensional arrays and matrices, facilitating operations like element-wise computations, linear algebra, statistical analysis, and more. By leveraging optimized algorithms and data structures, NumPy empowers researchers, engineers, and data scientists to efficiently tackle complex mathematical problems and process vast datasets with ease.

Its seamless integration with other scientific libraries makes it a pivotal component in the Python ecosystem, fostering innovation and advancement across various domains, including physics, engineering, machine learning, and beyond.

7. Matplotlib / Seaborn:

Matplotlib and Seaborn are essential Python libraries renowned for their prowess in data visualization, enabling users to craft a wide array of plots and charts for comprehensive data analysis and model evaluation. Matplotlib serves as the foundation for creating static, interactive, and publication-quality visualizations, offering fine-grained control over every aspect of the plot’s appearance. On the other hand, Seaborn operates as a high-level interface to Matplotlib, providing a more straightforward syntax and producing aesthetically pleasing statistical graphics with minimal code.

Together, they empower data scientists and analysts to explore data distributions, identify patterns, and communicate insights effectively through intuitive visualizations, thus enhancing understanding and driving informed decision-making in various domains, from exploratory data analysis to model performance evaluation.

8. XGBoost / LightGBM / CatBoost:

Popular gradient boosting libraries such as XGBoost, LightGBM, and CatBoost offer efficient implementations of gradient boosting algorithms, making them go-to choices for various machine learning tasks, including fraud detection. These libraries excel in handling imbalanced datasets, a common characteristic in fraud detection scenarios, by employing techniques such as weighted sampling and advanced regularization. Their ability to iteratively improve model performance by minimizing loss functions makes them particularly suitable for detecting subtle patterns indicative of fraudulent activities within large and complex datasets.

Moreover, their flexibility in accommodating various data types, feature engineering techniques, and hyperparameter tuning options empowers data scientists to tailor models to specific fraud detection challenges, ultimately leading to high accuracy and robust performance in real-world applications.

9. Jupyter Notebook / Google Colab:

Jupyter Notebook and Google Colab revolutionize the way Python code is written and executed, particularly in the realm of machine learning. Jupyter Notebook provides an interactive platform where users can create and share documents containing live code, equations, visualizations, and narrative text. It fosters a seamless workflow for prototyping and experimenting with machine learning models by allowing users to iteratively write and execute code blocks, instantly visualizing results. On the other hand, Google Colab takes this functionality a step further by providing a cloud-based environment with free access to GPU and TPU resources, enabling faster computation for training complex models.

Both platforms offer collaborative features, facilitating teamwork and knowledge sharing within the machine learning community. Together, they empower practitioners and researchers to explore, develop, and deploy cutting-edge machine learning solutions with ease and efficiency.

10. SQL / NoSQL Databases:

Depending on the scale of the data and the requirements of the project, SQL or NoSQL databases may be used for storing and querying large volumes of transaction data efficiently.

These tools and frameworks provide a comprehensive set of capabilities for implementing credit card fraud detection systems using machine learning techniques, enabling developers to build accurate, scalable, and efficient solutions.

Algorithm and Performance Analysis

1.K-Nearest Neighbor (KNN):

This supervised learning technique achieves consistently high performance compared to other fraud detection techniques of supervised statistical pattern recognition. Three factors majorly affect its performance distance to identify the least distant neighbours. There are some rules to deduce a categorization from the k-nearest neighbour and the count of neighbours to label the new sample. This algorithm classifies transactions by computing the least distant point to this particular transaction. If this least distant neighbour is classified as fraudulent, the latest marketing is also labelled as fraudulent. Euclidean distance is an excellent choice to calculate the distances in this scenario. This technique is fast and results in fault alerts. Its performance can be improved by distance metric optimization.

KNN is a non-parametric and lazy learning algorithm, meaning it doesn’t make any assumptions about the underlying data distribution and doesn’t learn a model during the training phase.

CHAPTER 6

APPLICATIONS

6.1.Banking and Financial Services:

Financial institutions such as banks and credit card companies rely on advanced technology like machine learning algorithms to detect fraudulent transactions in real-time. These algorithms are designed to analyze vast amounts of data quickly and efficiently to identify patterns and anomalies that may indicate fraudulent activity.

By continuously monitoring transactions as they occur, these algorithms can flag suspicious behavior, such as unusually large purchases, transactions from unfamiliar locations, or unusual spending patterns. Once flagged, these transactions can be further investigated or blocked outright to prevent financial losses and protect the assets of the institution and its customers.

Utilizing machine learning for fraud detection allows financial institutions to stay ahead of increasingly sophisticated fraudsters who are constantly evolving their tactics. By leveraging these algorithms, banks and credit card companies can enhance their security measures and provide customers with greater peace of mind knowing that their accounts are being actively monitored for any signs of fraudulent activity.

6.2. E-commerce and Online Retail:

Online retailers and e-commerce platforms operate within a digital landscape fraught with the constant threat of credit card fraud. With a staggering volume of transactions processed daily, the risk is omnipresent. However, machine learning models serve as invaluable tools in mitigating this risk. These models leverage vast datasets to detect patterns and anomalies indicative of fraudulent activity, enabling swift identification and prevention of suspicious orders. By analyzing factors such as purchase history, transaction frequency, and user behavior, these algorithms can accurately flag potentially fraudulent transactions in real-time, reducing the likelihood of chargebacks

and financial losses. Moreover, the implementation of robust fraud detection systems instills confidence in consumers, assuring them of a secure shopping environment and fostering long-term trust and loyalty. Thus, machine learning not only safeguards businesses from financial harm but also upholds the integrity of the e-commerce ecosystem, ensuring a seamless and secure experience for all stakeholders involved.

6.3. Payment Gateways and Processors:

Payment gateways and processors play a pivotal role in facilitating secure transactions within the realm of e-commerce. To combat the persistent threat of credit card fraud, these entities leverage sophisticated machine learning algorithms integrated into their systems. These algorithms are designed to assess the risk associated with each transaction by analyzing a myriad of factors. One crucial aspect scrutinized is the transaction history associated with the user's account, including past purchasing behavior and patterns. By examining this data, machine learning models can identify deviations from typical spending habits or recurring patterns indicative of fraudulent activity.

Furthermore, payment gateways and processors analyze device information such as IP addresses, geolocation data, and device type to establish the legitimacy of transactions. Discrepancies between the location of the user and the origin of the transaction, or the use of unfamiliar devices, can raise red flags prompting further scrutiny. Additionally, user behavior is closely monitored and scrutinized. Machine learning algorithms analyze the velocity of transactions, the time of day, and any unusual purchasing behaviors that may suggest fraudulent intent. For instance, a sudden surge in transaction volume or an unusually high-value purchase may warrant additional verification steps to confirm the authenticity of the transaction.

By integrating these advanced machine learning capabilities into their systems, payment gateways and processors can distinguish between legitimate and fraudulent transactions in real-time. This proactive approach allows them to swiftly mitigate potential losses by

flagging suspicious activity before it escalates into costly chargebacks or financial damages. Moreover, the continuous learning capabilities of these algorithms enable them to adapt to evolving fraud patterns and tactics, ensuring ongoing effectiveness in combating fraudulent activity.

Overall, the integration of machine learning algorithms into payment gateways and processors represents a proactive and dynamic approach to fraud detection and prevention in e-commerce. By analyzing a diverse array of factors and behaviors, these algorithms empower businesses to safeguard their financial interests while simultaneously preserving the trust and confidence of consumers in the security of online transactions.

6.4. Digital Wallets and Mobile Payments:

Digital wallets and mobile payments have revolutionized the way we conduct transactions, offering convenience, security, and efficiency to consumers and businesses alike. At the heart of these innovations lie sophisticated technologies, including encryption protocols and biometric authentication, ensuring the safety and integrity of sensitive financial information. One of the key advantages of digital wallets is their ability to tokenize payment credentials, replacing actual card numbers with unique, randomized tokens that are meaningless to potential attackers. This significantly reduces the risk of credit card fraud, as even if intercepted, these tokens are of no value to cybercriminals.

Moreover, digital wallets often incorporate multifactor authentication methods such as fingerprint or facial recognition, adding an extra layer of security to mobile payments. These biometric authentication measures not only enhance security but also streamline the checkout process, offering a seamless and

frictionless user experience. Additionally, many digital wallets employ machine learning algorithms to analyze user behavior and detect anomalies that may indicate fraudulent activity.

By monitoring factors such as transaction history, spending patterns, and device usage, these algorithms can quickly flag suspicious transactions and prevent unauthorized access to accounts.

From a business perspective, digital wallets and mobile payments offer numerous benefits, including reduced transaction costs, increased customer engagement, and enhanced data security. By embracing these technologies, businesses can streamline the checkout process, reduce cart abandonment rates, and tap into new markets, particularly among tech-savvy consumers who prefer mobile-first payment solutions. Furthermore, the adoption of digital wallets enables businesses to collect valuable data on consumer spending habits and preferences, facilitating targeted marketing efforts and personalized experiences.

6.5. Travel and Hospitality:

The travel and hospitality industry, including airlines, hotels, and booking platforms, often experiences credit card fraud related to reservations and bookings. Machine learning models help identify fraudulent transactions, such as booking cancellations, multiple reservations using stolen cards, and identity theft, thereby minimizing revenue losses and ensuring a secure booking process for customers.
The travel and hospitality industry has undergone significant transformation in recent years, driven by technological advancements and changing consumer preferences. One of the most notable shifts has been the widespread adoption of digital platforms and online booking systems, which have revolutionized the way people plan and book their travel experiences. From flights and accommodations to car rentals and activities, travelers now have access to a vast array of options at their fingertips, thanks to intuitive booking interfaces and user-friendly mobile apps.

In addition to streamlining the booking process, technology has also played a crucial role in enhancing the overall travel experience. For instance, the rise of smart devices and wearable technology has enabled travelers to stay connected and informed

throughout their journey, whether they're checking into their flight, navigating a new city, or accessing personalized recommendations for dining and entertainment. Moreover, the integration of artificial intelligence and machine learning algorithms has allowed travel companies to deliver more personalized and tailored experiences to their customers, from customized travel itineraries to targeted promotions and offers.

Furthermore, the emergence of sharing economy platforms has disrupted traditional models of accommodation and transportation, offering travelers more choice and flexibility at competitive prices. Services like Airbnb and Uber have empowered individuals to monetize their assets, whether it's spare rooms in their homes or idle vehicles, while also providing travelers with unique and authentic experiences that go beyond traditional hotel stays and taxi rides.

However, the travel and hospitality industry also faces its share of challenges, including concerns around data privacy and security, as well as the impact of global events such as pandemics and natural disasters on travel patterns and consumer behavior. Nonetheless, with ongoing innovations in technology and a growing emphasis on sustainability and responsible tourism, the future of the travel and hospitality industry remains bright, promising exciting opportunities for both businesses and travelers alike. As technology continues to evolve and consumer expectations continue to shift, the industry will undoubtedly continue to adapt and innovate, shaping the way we experience and explore the world for years to come.

6.6. Healthcare and Insurance:

Healthcare providers and insurance companies leverage machine learning techniques to detect fraudulent claims and billing activities. By analyzing patterns in medical billing, insurance claims, and patient records, they can identify anomalies indicative of fraudulent behavior and prevent financial losses due to fraudulent claims.

The intersection of healthcare and insurance represents a complex ecosystem shaped by technological advancements, regulatory requirements, and evolving patient needs. In

recent years, technology has played a pivotal role in driving innovation and improving access to care, while also addressing challenges related to cost, efficiency, and patient outcomes.

One of the most significant advancements in healthcare technology is the adoption of electronic health records (EHRs) and digital health platforms, which have streamlined administrative processes, improved data accuracy, and enhanced care coordination among healthcare providers. By digitizing patient information and medical histories, EHRs enable clinicians to access critical information in real-time, leading to more informed decision-making and better outcomes for patients. Additionally, digital health solutions such as telemedicine and remote patient monitoring have expanded access to care, particularly in rural and underserved areas, while also reducing healthcare costs and improving patient satisfaction.

6.7. Government and Regulatory Agencies:

Government agencies and regulatory bodies use machine learning to monitor financial transactions and identify potential instances of money laundering, terrorist financing, and other illicit activities. By analyzing large volumes of transaction data from financial institutions, they can detect suspicious patterns and take appropriate enforcement actions to combat financial crime.

Overall, credit card fraud detection using machine learning has broad applications across industries, helping organizations mitigate financial risks, protect customer assets, and maintain trust in the integrity of financial transactions.

One of the primary functions of government and regulatory agencies is to establish and enforce regulations that safeguard public health and safety. This includes overseeing industries such as healthcare, food and drug administration, transportation, and environmental protection.

CHAPTER 7

ALGORITHM

7.1 Algorithm KNN

- Let m be the number of training data samples. Let p be an unknown point that needs to be classified
- Storing the training samples in an array of data points arr[]. Each element of this array denotes a tuple (x, y).
- for i = 0 to m do
- Calculating distance d(arr[i], p)
- end for
- They are making set S of K smallest distances achieved. Each of these distances resembles an already classified data point.
- Returning the majority label among S.

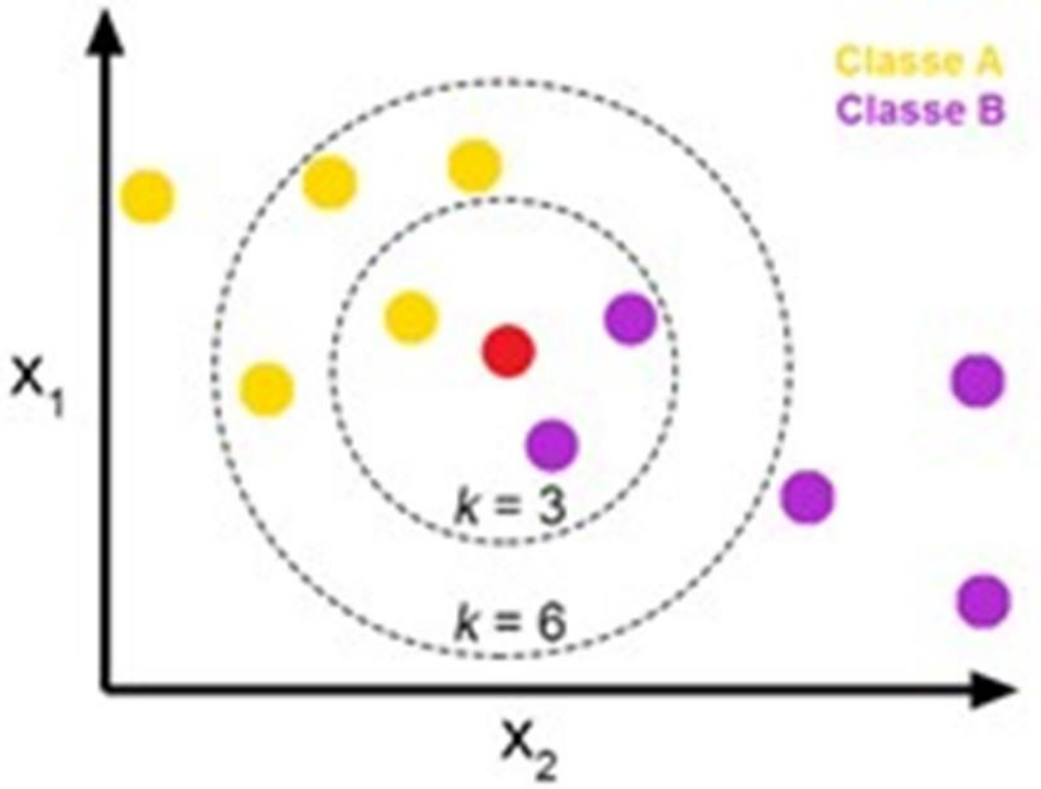


Figure 7: KNN

Two Ks were used to determine the best KNN model, K=3 and K =7.

- K = 3 While making the KNN model, We created two models: K =3 and K =7. Figure 5 shows the model created in Jupiter Notebook; the model scored an accuracy of 100% and identified 85,443 transactions correctly and missed 131.
- K = 7
- There was a slight decrease in the Accuracy of the model created in Jupiter Notebook as it scored 100% when K is 7, and the model miss classified 131 fraudulent transactions as no fraudulent. As for the Accuracy is the same as K=3 100% with 52 misclassified transactions; the only difference is within the classifications.

7.2 Logistic Regression (L.R.):

This statistical classification model based on probabilities detects Fraud using a logistic curve. Since the value of this logistic curve varies from 0 to 1, it can be

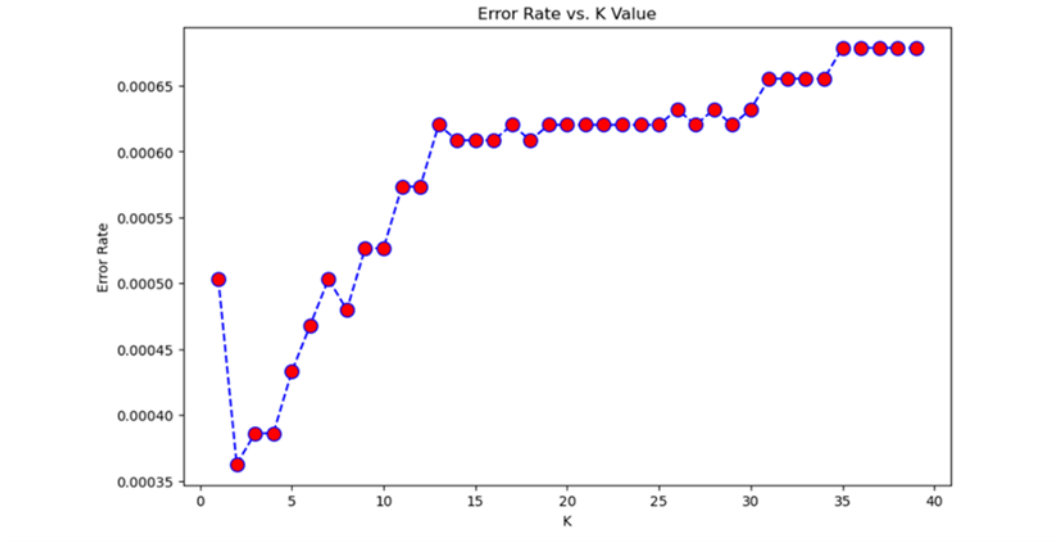


Figure 8 : Error Rate

used to interpret class membership probabilities. The dataset fed as input to the model is classified for training and testing the model. Post-model activity is tested for some minimum threshold cut-off value for prediction. Based on some threshold probabilities, the logistic Regression can divide the plane using a single line and divide dataset points into exactly two regions.

Logistic Regression (LR) is a fundamental statistical technique used for binary classification tasks, where the goal is to predict the probability of an event occurring based on one or more independent variables. Despite its name, logistic regression is a linear model that employs the logistic function to map input features to a binary outcome.

The last model created using Jupiter Notebook is Logistic Regression; the model managed to score an Accuracy on Training data of 93.51% , while it scored an Accuracy score on Test Data of 91.88%.

7.3 Support Vector Machine (SVM):

Support vector machines or SVMs are linear classifiers, as stated in that work in high dimensionality because, in high dimensions, a non-linear task in input becomes linear. Hence, this makes SVMs highly useful for detecting Fraud. Its two most important features that is a kernel function to represent the classification function in the dot product of input data point projection and the fact that it tries finding a hyperplane to maximize separation between classes while minimizing overfitting of training data; it provides a very high generalization capability.

Finally, the model Support Vector Machine, as shown in Figure 9, scored 97.59 % for the Accuracy.

```
WITH k=3


[[85307     5]
 [   28   103]]

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85312
           1       0.95      0.79      0.86       131

    accuracy                           1.00     85443
   macro avg       0.98      0.89      0.93     85443
weighted avg       1.00      1.00      1.00     85443
```

Figure 9:Support vector machine accuracy

7.4 Decision Tree (D.T.):

A supervised learning algorithm is a decision tree in the form of a tree structure consisting of the root node and other nodes split in a binary or multi-split manner further into child nodes, with each tree using its algorithm to perform the splitting process. With the tree growing, there may be possibilities of overfitting the training data with possible anomalies in branches, some errors or noise. Hence, pruning is used for improving classification performance of the tree by removing specific nodes. Ease in use and the flexibility that the decision trees provide to handle different data types of attributes make them quite popular.

Decision Trees (DT) are powerful and interpretable machine learning models used for both classification and regression tasks. They operate by recursively partitioning the input space into subsets based on the values of input features, with the goal of maximizing the purity of the resulting subsets.

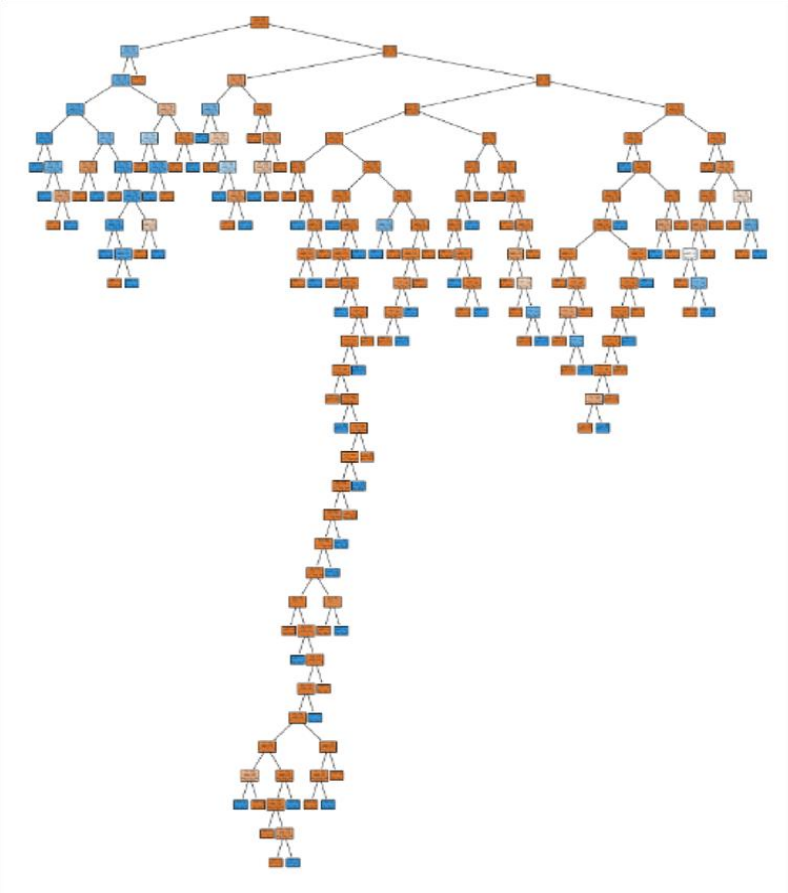


Figure 10: Decision tree

Algorithm: of Decision Tree.

- Create (T)
- Calculate frequencies (Ci, T)
- If all instances belong to the same class, the returning leaf
- for every Attribute, a test is set for splitting criteria. An attribute that satisfies the test is test node K
- Repeating Create (Ti) on each partition Ti.Adding those nodes as children of node K

```
WITH k=7


[[85300    12]
 [   31   100]]


              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85312
           1       0.89      0.76      0.82       131

    accuracy                           1.00     85443
   macro avg       0.95      0.88      0.91     85443
weighted avg       1.00      1.00      1.00     85443
```

Figure 11: Decision tree accuracy

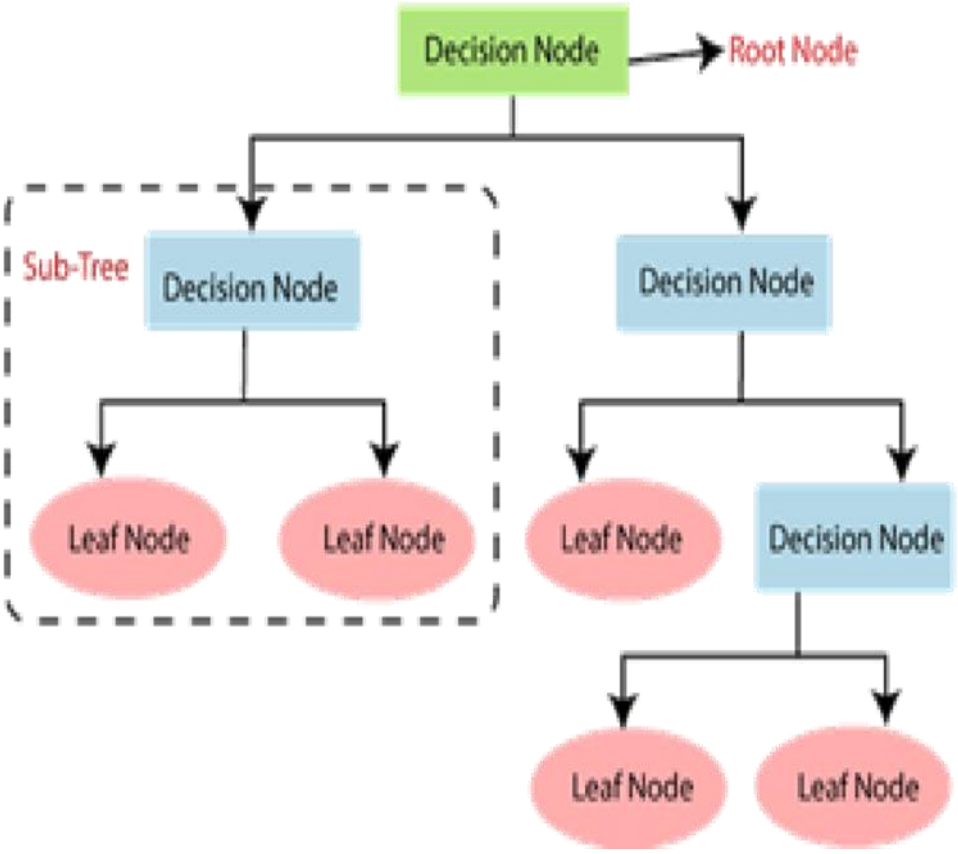


Figure 12: Decision Tree Algorithm

CHAPTER 8

SYSTEM TESTING

8.1 TESTING

- Testing the analysis of source/executable code and the controlled execution of executable code to reveal defects that compromise a python programs executable integrity. Defects often lead to erratic behavior or the premature termination of an executing program.
- The system should be tested experimentally with test data so as to ensure that the system works according to the required specifications. Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. After the coding phase, computer programs are available that can be executed for testing purpose.
- This process encompasses various aspects, including functionality testing to verify that all features and functionalities of the machine learning-based fraud detection system are working correctly and reliably. Additionally, performance testing is conducted to assess the system's responsiveness, scalability, and reliability under different loads and conditions. Security testing is crucial to identify and address potential vulnerabilities that could be exploited by malicious actors. Moreover, integration testing ensures seamless interaction between different components of the system, including data sources, machine learning algorithms, and user interfaces. Throughout the testing process, rigorous validation against real-world scenarios and historical data is performed to validate the accuracy and efficacy of the fraud detection mechanisms, ultimately ensuring that "CARDSAFE FORENSICS" can effectively combat financial deceptions and safeguard the integrity of financial transactions.

8.2 UNIT TESTING

Unit testing is the process where you test the smallest functional unit of code. Software testing helps ensure code quality, and it's an integral part of software development. It's a software development best practice to write software as small, functional units then write a unit test for each code unit. You can first write unit tests as code. Then, run that test code automatically every time you make changes in the software code. This way, if a test fails, you can quickly isolate the area of the code that has the bug or error. Unit testing enforces modular thinking paradigms and improves test coverage and quality. Automated unit testing helps ensure you or your developers have more time to concentrate on coding.

Each component of the system, from data preprocessing and feature engineering to model training and evaluation, must undergo rigorous unit testing to validate its functionality and correctness. Unit tests should encompass various scenarios and edge cases, including different types of fraudulent activities and diverse input data characteristics, to verify the robustness and accuracy of the system. Additionally, unit tests should be designed to assess the system's performance metrics, such as precision, recall, and false positive rate, to ascertain its effectiveness in detecting and preventing financial fraud. By meticulously conducting unit testing throughout the development process, the "CARDSAFE FORENSICS" project can instill confidence in its machine learning algorithms and forensic methodologies, ultimately bolstering its capabilities in combatting financial deceptions and protecting the integrity of financial transactions.

8.3 INTEGRATION TESTING

Integration testing is known as the second level of the software testing process, following unit testing. Integration testing involves checking individual components or units of a software project to expose defects and problems to verify that they work together as designed.

This entails testing the integration points between various software modules, databases, and external systems to ensure smooth communication and data flow. Specifically, integration testing for "CARDSAFE FORENSICS" would focus on validating the integration of machine learning algorithms with the fraud detection system, ensuring that the models accurately analyze transaction data and flag suspicious activities. Additionally, it involves testing the integration of forensic tools and data visualization components to enable investigators to interpret and act upon the insights provided by the machine learning models effectively. Through rigorous integration testing, the project aims to ensure the reliability, accuracy, and effectiveness of the overall system in combating financial deceptions while maintaining the integrity of the forensic process.

CHAPTER 9

CODE

9.1. LIBRARIES

NumPy: For numerical computing.

NumPy is a fundamental library for numerical computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy's array data structure allows for fast and efficient computation, making it essential for numerical tasks such as linear algebra, statistical analysis, and signal processing.

One of the key features of NumPy is its ndarray (n-dimensional array) object, which represents homogeneous arrays of fixed-size items. These arrays enable vectorized operations, where mathematical operations are applied element-wise across the entire array, without the need for explicit looping constructs. This vectorized approach leads to faster execution compared to traditional Python loops and is particularly beneficial when dealing with large datasets.

Additionally, NumPy provides a wide range of mathematical functions for array manipulation, including basic arithmetic operations, statistical functions, linear algebra routines, Fourier transforms, and more. Its versatility and performance make it a cornerstone of scientific computing and machine learning in Python, forming the foundation upon which many other libraries and frameworks are built.

Pandas: For data manipulation and analysis.

Pandas is a powerful Python library for data manipulation and analysis. It offers data structures like DataFrame and Series that allow for easy handling and manipulation of structured data. In the code, Pandas is used for reading data from CSV files, exploring dataset characteristics (e.g., viewing data, checking null values), selecting specific columns, and merging datasets.

Pandas is a powerful library in Python designed for data manipulation and analysis. It provides easy-to-use data structures and functions for working with structured data, making tasks like loading, cleaning, transforming, and analyzing data more efficient and intuitive.

At the core of Pandas are two primary data structures: Series and DataFrame. Series represents a one-dimensional array-like object containing an array of data (of any NumPy data type) and an associated array of labels, called the index. DataFrame, on the other hand, is a two-dimensional labeled data structure with columns of potentially different types. It can be thought of as a spreadsheet or SQL table, where each column is a Series.

Pandas offers a wide range of functionalities for data manipulation, including indexing, slicing, merging, reshaping, grouping, and aggregating data. It allows for efficient handling of missing data, data alignment, and data I/O operations with various file formats such as CSV, Excel, SQL databases, and more. Moreover, Pandas integrates seamlessly with other libraries like NumPy, Matplotlib, and Scikit-learn, enabling a smooth workflow for data analysis and machine learning tasks.

Matplotlib: For data visualization.

Matplotlib is a comprehensive library in Python used for creating static, interactive, and publication-quality visualizations. It provides a wide range of functionalities for generating various types of plots and charts, including line plots, scatter plots, bar plots, histograms, heatmaps, and more. Matplotlib's flexibility and customization options make it suitable for a diverse range of visualization needs in data analysis, scientific research, and presentation of results.

One of the key features of Matplotlib is its object-oriented interface, which allows users to create and customize plots by directly interacting with plot elements such as axes, figures, and artists. This interface provides fine-grained control over every aspect of the plot, enabling users to tailor visualizations to their specific requirements.

Additionally, Matplotlib integrates seamlessly with other Python libraries such as NumPy and Pandas, allowing for easy visualization of data stored in arrays or DataFrames. It also supports various output formats, including interactive environments like Jupyter notebooks, static image files (e.g., PNG, PDF), and graphical user interfaces (GUIs), making it suitable for different usage scenarios.

Overall, Matplotlib is a versatile and widely-used library for data visualization in Python. Its rich set of features, combined with its flexibility and ease of use, make it an essential tool for exploring data, communicating insights, and presenting findings in a visually appealing manner.

Seaborn: For statistical data visualization based on Matplotlib.

Seaborn is a Python visualization library built on top of Matplotlib and closely integrated with Pandas data structures. It provides a high-level interface for creating attractive and informative statistical graphics, with an emphasis on concise syntax and built-in functionality for common visualization tasks.

One of the key features of Seaborn is its ability to generate complex statistical plots with minimal code. It includes functions for creating visually appealing plots such as scatter plots, box plots, violin plots, pair plots, and heatmaps, among others. These plots are designed to highlight patterns, relationships, and distributions in the data, making them valuable tools for exploratory data analysis and presentation of findings.

Seaborn also offers support for visualizing categorical data through features like categorical plots, swarm plots, and factor plots, allowing for the examination of data distributions across different categories or groups. Moreover, Seaborn provides options for customization and theming, allowing users to easily modify the appearance of plots to suit their preferences or adhere to specific styling guidelines.

Overall, Seaborn serves as a complementary library to Matplotlib, offering additional capabilities and convenience for statistical data visualization tasks. Its integration with Pandas and Matplotlib, along with its intuitive API and rich set of features, make it a popular choice for researchers, data scientists, and analysts seeking to create informative and visually appealing plots from their data.

Scikit-learn: For machine learning tasks, including:

Scikit-learn, often abbreviated as sklearn, is a widely-used Python library for machine learning tasks, offering a comprehensive suite of tools and algorithms for building predictive models, performing data preprocessing, and evaluating model performance. One of Scikit-learn's key strengths lies in its user-friendly and consistent API, which makes it accessible to both novice and experienced machine learning practitioners.

Scikit-learn provides implementations of a vast array of machine learning algorithms, including supervised and unsupervised learning methods, as well as tools for model selection, feature extraction, and dimensionality reduction. For supervised learning, it offers algorithms for classification, regression, and ensemble methods like random forests and gradient boosting. For unsupervised learning, it includes algorithms for clustering, manifold learning, and decomposition.

Furthermore, Scikit-learn emphasizes model evaluation and validation through robust and efficient techniques such as cross-validation, grid search, and metrics for assessing model performance. These tools enable users to fine-tune model hyperparameters, handle overfitting, and make informed decisions about model selection based on objective evaluation criteria.

Overall, Scikit-learn serves as a versatile and reliable library for tackling a wide range of machine learning tasks, from simple classification and regression problems to more complex tasks like clustering and dimensionality reduction. Its ease of use, extensive documentation, and active community support make it an essential tool for practitioners in academia, industry, and research, driving innovation and advancement in the field of machine learning.

Sklearn.model_selection: This module provides functions for splitting datasets into train and test sets. It includes various methods for cross-validation and model selection. The sklearn.model_selection module in scikit-learn is a crucial component for partitioning datasets into training and testing subsets, facilitating the evaluation and validation of machine learning models. It offers functions like train_test_split(), which efficiently divides data into separate training and testing sets based on user-defined ratios. Additionally, this module provides advanced techniques for cross-validation, such as K-fold and Stratified K-fold cross-validation, enabling robust estimation of model performance by systematically partitioning the data into multiple subsets for training and evaluation. Furthermore, it includes utilities for hyperparameter tuning through methods like Grid Search CV and Randomized Search CV, allowing users to systematically search through a hyperparameter space to find the optimal model configuration. Overall, sklearn.model_selection is essential for ensuring reliable model evaluation and selection in machine learning workflows.

Sklearn.tree: This module contains the Decision Tree Classifier class, which is used to create decision tree models for classification tasks. It also includes functions for visualizing decision trees. The sklearn.tree module in scikit-learn provides powerful tools for building decision tree models and visualizing their structures. The Decision Tree Classifier class within this module allows users to create decision tree models specifically tailored for classification tasks. Decision trees are intuitive models that partition the feature space into regions, making predictions based on the majority class within each region. Additionally, the module includes functions for visualizing decision trees, allowing users to gain insights into the decision-making process of the model. Visualizations can include details such as feature importance, node splits, and class distributions, providing valuable interpretability for understanding model predictions and feature relationships. Overall, the DecisionTree Classifier class and associated visualization functions in the sklearn.tree module enable users to build, analyze, and interpret decision tree models effectively for classification tasks.

Sklearn.linear_model: This module includes the Logistic Regression class, which is used to perform logistic regression for binary classification tasks. Logistic regression is a linear model used when the target variable is categorical. The sklearn.linear_model module in scikit-learn incorporates the Logistic Regression class, which is a fundamental tool for performing logistic regression in binary classification tasks. Logistic regression is a type of linear model used when the target variable is categorical, specifically binary (having two classes). The logistic regression algorithm estimates the probability that a given instance belongs to a particular class using a logistic function (also known as the sigmoid function), which maps the output of a linear combination of features to a probability score between 0 and 1. The Logistic Regression class in scikit-learn provides various options for regularization, such as L1 and L2 regularization, to prevent overfitting and improve model generalization. Overall, the LogisticRegression class in sklearn.linear_model is a versatile and powerful tool for binary classification tasks, providing interpretable results and straightforward implementation.

Sklearn.neighbors: This module contains the K Neighbors Classifier class, which implements the k-nearest neighbors algorithm. K-nearest neighbors is a non-parametric method used for classification and regression tasks, where the output is based on the majority vote of its neighbors.

Sklearn.svm: This module includes the SVC (Support Vector Classifier) class, which implements Support Vector Machines for classification tasks. SVMs are supervised learning models used for classification, regression, and outlier detection.

The sklearn.svm module in scikit-learn contains the SVC (Support Vector Classifier) class, which is used to implement Support Vector Machines (SVMs) for classification tasks. SVMs are powerful supervised learning models that can be used for classification, regression, and outlier detection. In the context of classification, SVMs work by finding the hyperplane that best separates the classes in the feature space, maximizing the margin between the classes. The SVC class in scikit-learn offers various options for

kernel functions, such as linear, polynomial, and radial basis function (RBF), allowing users to model complex relationships in the data. SVMs are particularly effective in high-dimensional spaces and cases where the data is not linearly separable. Additionally, SVMs can be extended to handle multi-class classification and incorporate regularization techniques to prevent overfitting. Overall, the SVC class in sklearn.svm provides a versatile and robust tool for solving classification problems across different domains.

sklearn.metrics: This module provides various metrics for evaluating machine learning models' performance, including classification accuracy, precision, recall, F1-score, confusion matrix, ROC curve, and AUC (Area Under the ROC Curve).The sklearn.metrics module in scikit-learn is a crucial component for assessing the performance of machine learning models across a wide range of tasks, particularly in classification problems. It offers a comprehensive suite of evaluation metrics that enable users to quantify different aspects of model performance. These metrics include classification accuracy, which measures the proportion of correctly classified instances, precision, which quantifies the accuracy of positive predictions, recall (also known as sensitivity), which gauges the model's ability to capture all positive instances, and the F1-score, which balances precision and recall into a single metric.

Here's a summary of the libraries imported in each snippet:

DECISION TREE

pandas: for data manipulation and analysis.

Numpy: for numerical computations.

Seaborn: for data visualization.

Matplotlib: for plotting graphs.

Sklearn: for machine learning algorithms and evaluation metrics.

Logistic Regression:

numpy as np: For numerical operations.

Pandas as pd: For data manipulation and analysis.

Train_test_split from sklearn.model_selection: For splitting the dataset into train and test sets.

LogisticRegression from sklearn.linear_model: For logistic regression modeling.

Accuracy_score from sklearn.metrics: For calculating accuracy scores.

K-Nearest Neighbour:

pandas as pd: For data manipulation and analysis.

Numpy as np: For numerical operations.

Matplotlib.pyplot as plt: For data visualization.

Seaborn as sns: For statistical data visualization.

RandomUnderSampler from imblearn.under_sampling: For undersampling imbalanced data.Train_test_split from sklearn.model_selection: For splitting the dataset into train and test sets.

SVC from sklearn.svm: For Support Vector Machine modeling.

Various metrics functions from sklearn.metrics: For evaluating model performance.

Support Vector Machines:

pandas as pd: For data manipulation and analysis.

Numpy as np: For numerical operations.

Matplotlib.pyplot as plt: For data visualization.

Seaborn as sns: For statistical data visualization.

RandomUnderSampler from imblearn.under_sampling: For undersampling imbalanced data.

Train_test_split from sklearn.model_selection: For splitting the dataset into train and test sets.

SVC from sklearn.svm: For Support Vector Machine modeling.

Various metrics functions from sklearn.metrics: For evaluating model performance.

Machine Learning Libraries:

sklearn.model_selection: Train/test splitting.

Sklearn.tree: Decision Tree Classifier and visualization.

Sklearn.linear_model: Logistic Regression.

Sklearn.neighbors: K-Nearest Neighbors.

Sklearn.svm: Support Vector Machines.

Sklearn.metrics: Classification metrics and evaluation.

9.2. CODE

DECISION TREE

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline
df = pd.read_csv('creditcard.csv')
df.columns

df.head(3)

df.info()

df.describe()

df.isnull().sum()

sns.pairplot(df,hue='Class',palette='Set1')

from sklearn.model_selection import train_test_split

X = df.drop('Class',axis=1)

y = df['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier(criterion='entropy', random_state=0)

dtree.fit(X_train,y_train)

predictions = dtree.predict(X_test)

predictions

array([0, 0, 0, …, 0, 0, 0], dtype=int64)

from sklearn.metrics import classification_report,confusion_matrix
```

```
print(classification_report(y_test,predictions))
print(confusion_matrix(y_test,predictions))
from sklearn import tree
plt.figure(figsize=(20,25))
tree.plot_tree(dtree,feature_names=X.columns,class_names=['Class-1', 'Class-0'],rounded=True, # Rounded node edges
        filled=True, # Adds color according to class
        proportion=True
       )
plt.show()
```

LOGISTIC REGRESSION

```
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
credit_card_data = pd.read_csv('creditcard.csv') credit_card_data.head()
credit_card_data.tail()
credit_card_data.info() credit_card_data.isnull().sum()
credit_card_data['Class'].value_counts()
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]
print(legit.shape)
print(fraud.shape)
 legit.Amount.describe() fraud.Amount.describe()
```

```
credit_card_data.groupby('Class').mean() legit_sample = legit.sample(n=492)

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

new_dataset.head()

new_dataset.tail() new_dataset['Class'].value_counts()
new_dataset.groupby('Class').mean()

X = new_dataset.drop(columns='Class', axis=1)

Y = new_dataset['Class']

print(X)

print(Y)

 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y,
random_state=2)

print(X.shape, X_train.shape, X_test.shape) model = LogisticRegression()

model.fit(X_train, Y_train)

X_train_prediction = model.predict(X_train)

training_data_accuracy = accuracy_score(X_train_prediction, Y_train)

print('Accuracy on Training data : ', training_data_accuracy)

X_test_prediction = model.predict(X_test)

test_data_accuracy = accuracy_score(X_test_prediction, Y_test)

print('Accuracy score on Test Data : ', test_data_accuracy)
```

K- Nearest Neighbour

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

credit_card_data = pd.read_csv('creditcard.csv')
```

```
credit_card_data.keys()
credit_card_data.head()
credit_card_data.tail()
 credit_card_data.info()
credit_card_data.isnull().sum()
credit_card_data['Class'].value_counts()
 credit_card_data = credit_card_data.drop("Time", axis=1)
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
credit_card_data['std_Amount'] =
scaler.fit_transform(credit_card_data['Amount'].values.reshape (-1,1))
credit_card_data = credit_card_data.drop("Amount", axis=1)
sns.countplot(x="Class", data=credit_card_data)
import imblearn
from imblearn.under_sampling import RandomUnderSampler
undersample = RandomUnderSampler(sampling_strategy=0.5)
cols = credit_card_data.columns.tolist()
cols = [c for c in cols if c not in ["Class"]]
target = "Class"
X = credit_card_data[cols]
Y = credit_card_data[target]
X_under, Y_under = undersample.fit_resample(X, Y)
from pandas import DataFrame
test = pd.DataFrame(Y_under, columns = ['Class'])
#visualizing undersampling results
fig, axs = plt.subplots(ncols=2, figsize=(13,4.5))
sns.countplot(x="Class", data=credit_card_data, ax=axs[0])
```

```
sns.countplot(x="Class", data=test, ax=axs[1])
fig.suptitle("Class repartition before and after undersampling")
a1=fig.axes[0]
a1.set_title("Before")
a2=fig.axes[1]
a2.set_title("After")
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_under, Y_under, test_size=0.2,
random_state=1)
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc
from sklearn.metrics import precision_recall_curve
model = SVC()
model.fit(X_train,y_train)
model2 = SVC(probability=True, random_state=2)
svm = model2.fit(X_train, y_train)
y_pred_svm = model2.predict(X_test)
print("Accuracy SVM:",metrics.accuracy_score(y_test, y_pred_svm))
print("Precision SVM:",metrics.precision_score(y_test, y_pred_svm))
print("Recall SVM:",metrics.recall_score(y_test, y_pred_svm))
print("F1 Score SVM:",metrics.f1_score(y_test, y_pred_svm))
matrix_svm = confusion_matrix(y_test, y_pred_svm)
cm_svm = pd.DataFrame(matrix_svm, index=['not_fraud', 'fraud'],
```

```
columns=[‘not_fraud’, ‘fraud’])
sns.heatmap(cm_svm, annot=True, cbar=None, cmap=”Blues”, fmt = ‘g’)
plt.title(“Confusion Matrix SVM”), plt.tight_layout()
plt.ylabel(“True Class”), plt.xlabel(“Predicted Class”)
plt.show()
y_pred_svm_proba = model2.predict_proba(X_test)[::,1]
fpr_svm, tpr_svm, _ = metrics.roc_curve(y_test,  y_pred_svm_proba)
auc_svm = metrics.roc_auc_score(y_test, y_pred_svm_proba)
print(“AUC SVM :”, auc_svm)
plt.plot(fpr_svm,tpr_svm,label=”SVM, auc={:.3f})”.format(auc_svm))
plt.plot([0, 1], [0, 1], ‘k—')
plt.xlabel(‘False positive rate’)
plt.ylabel(‘True positive rate’)
plt.title(‘SVM ROC curve’)
plt.legend(loc=4)
plt.show()
svm_precision, svm_recall, _ = precision_recall_curve(y_test, y_pred_svm_proba)
no_skill = len(y_test[y_test==1]) / len(y_test)
plt.plot([0, 1], [no_skill, no_skill], linestyle=’—', color=’black’, label=’No Skill’)
plt.plot(svm_recall, svm_precision, color=’orange’, label=’SVM’)
plt.xlabel(‘Recall’)
plt.ylabel(‘Precision’)
plt.title(‘Precision-Recall curve’)
plt.legend()
plt.show()
```

Support Vector Machines

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
credit_card_data = pd.read_csv('creditcard.csv')
credit_card_data.keys()
credit_card_data.head()
credit_card_data.tail()
 credit_card_data.info()
credit_card_data.isnull().sum()
credit_card_data['Class'].value_counts()
 credit_card_data = credit_card_data.drop("Time", axis=1)
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
credit_card_data['std_Amount'] =
scaler.fit_transform(credit_card_data['Amount'].values.reshape (-1,1))
credit_card_data = credit_card_data.drop("Amount", axis=1)
sns.countplot(x="Class", data=credit_card_data)
import imblearn
from imblearn.under_sampling import RandomUnderSampler
undersample = RandomUnderSampler(sampling_strategy=0.5)
cols = credit_card_data.columns.tolist()
cols = [c for c in cols if c not in ["Class"]]
```

```
target = “Class”
X = credit_card_data[cols]
Y = credit_card_data[target]
X_under, Y_under = undersample.fit_resample(X, Y)
from pandas import DataFrame
test = pd.DataFrame(Y_under, columns = [‘Class’])
#visualizing undersampling results
fig, axs = plt.subplots(ncols=2, figsize=(13,4.5))
sns.countplot(x=”Class”, data=credit_card_data, ax=axs[0])
sns.countplot(x=”Class”, data=test, ax=axs[1])
fig.suptitle(“Class repartition before and after undersampling”)
a1=fig.axes[0]
a1.set_title(“Before”)
a2=fig.axes[1]
a2.set_title(“After”)
 from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_under, Y_under, test_size=0.2,
random_state=1)
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
%matplotlib inline
df = pd.read_csv('creditcard.csv')
df.columns

df.head(3)

df.info()

df.describe()

df.isnull().sum()

sns.pairplot(df,hue='Class',palette='Set1')

from sklearn.model_selection import train_test_split

X = df.drop('Class',axis=1)

y = df['Class']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30)

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier(criterion='entropy', random_state=0)

dtree.fit(X_train,y_train)

predictions = dtree.predict(X_test)

predictions

array([0, 0, 0, …, 0, 0, 0], dtype=int64)

from sklearn.metrics import classification_report,confusion_matrix

print(classification_report(y_test,predictions))

print(confusion_matrix(y_test,predictions))

from sklearn import tree

plt.figure(figsize=(20,25))

tree.plot_tree(dtree,feature_names=X.columns,class_names=['Class-1', 'Class-
0'],rounded=True, # Rounded node edges

      filled=True, # Adds color according to class

      proportion=True

    )
```

```
plt.show()
from sklearn.metrics import auc
from sklearn.metrics import precision_recall_curve
model = SVC()
model.fit(X_train,y_train)
model2 = SVC(probability=True, random_state=2)
svm = model2.fit(X_train, y_train)
y_pred_svm = model2.predict(X_test)
print("Accuracy SVM:",metrics.accuracy_score(y_test, y_pred_svm))
print("Precision SVM:",metrics.precision_score(y_test, y_pred_svm))
print("Recall SVM:",metrics.recall_score(y_test, y_pred_svm))
print("F1 Score SVM:",metrics.f1_score(y_test, y_pred_svm))
matrix_svm = confusion_matrix(y_test, y_pred_svm)
cm_svm = pd.DataFrame(matrix_svm, index=['not_fraud', 'fraud'],
columns=['not_fraud', 'fraud'])
sns.heatmap(cm_svm, annot=True, cbar=None, cmap="Blues", fmt = 'g')
plt.title("Confusion Matrix SVM"), plt.tight_layout()
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
plt.show()
y_pred_svm_proba = model2.predict_proba(X_test)[::,1]
fpr_svm, tpr_svm, _ = metrics.roc_curve(y_test,  y_pred_svm_proba)
auc_svm = metrics.roc_auc_score(y_test, y_pred_svm_proba)
print("AUC SVM :", auc_svm)
plt.plot(fpr_svm,tpr_svm,label="SVM, auc={:.3f})".format(auc_svm))
plt.plot([0, 1], [0, 1], 'k—')
plt.xlabel('False positive rate')
```

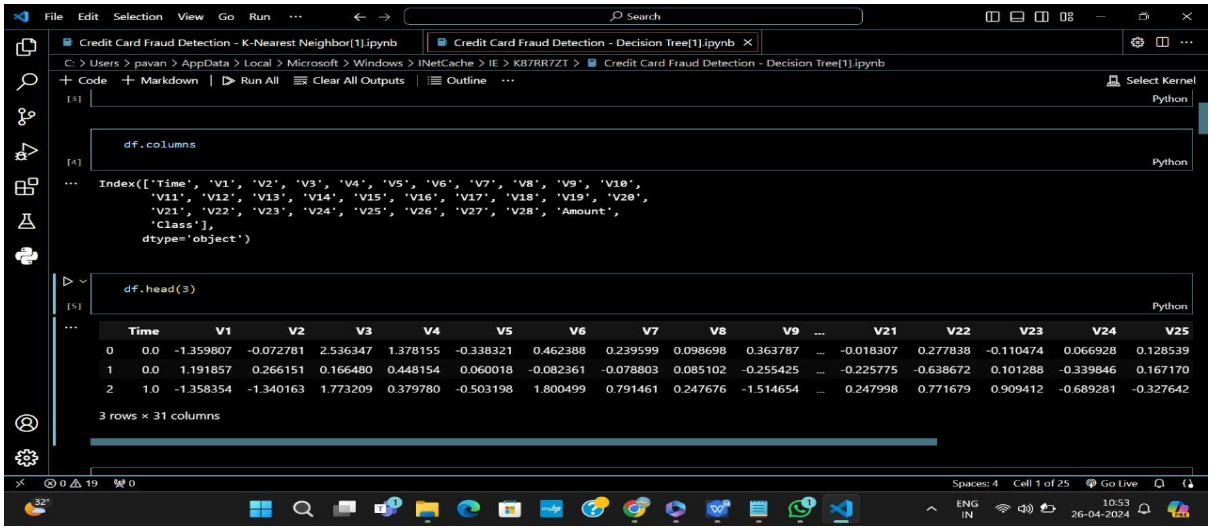
```
plt.ylabel('True positive rate')
plt.title('SVM ROC curve')
plt.legend(loc=4)
plt.show()
svm_precision, svm_recall, _ = precision_recall_curve(y_test, y_pred_svm_proba)
no_skill = len(y_test[y_test==1]) / len(y_test)
plt.plot([0, 1], [no_skill, no_skill], linestyle='—', color='black', label='No Skill')
plt.plot(svm_recall, svm_precision, color='orange', label='SVM')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall curve')
plt.legend()
plt.show()
```

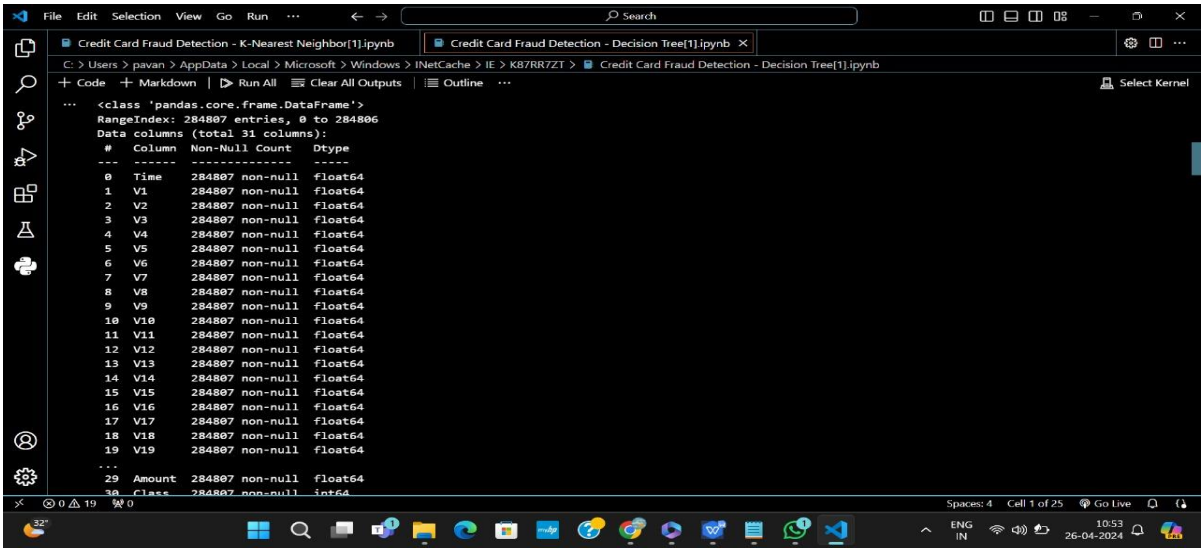
CHAPTER 10

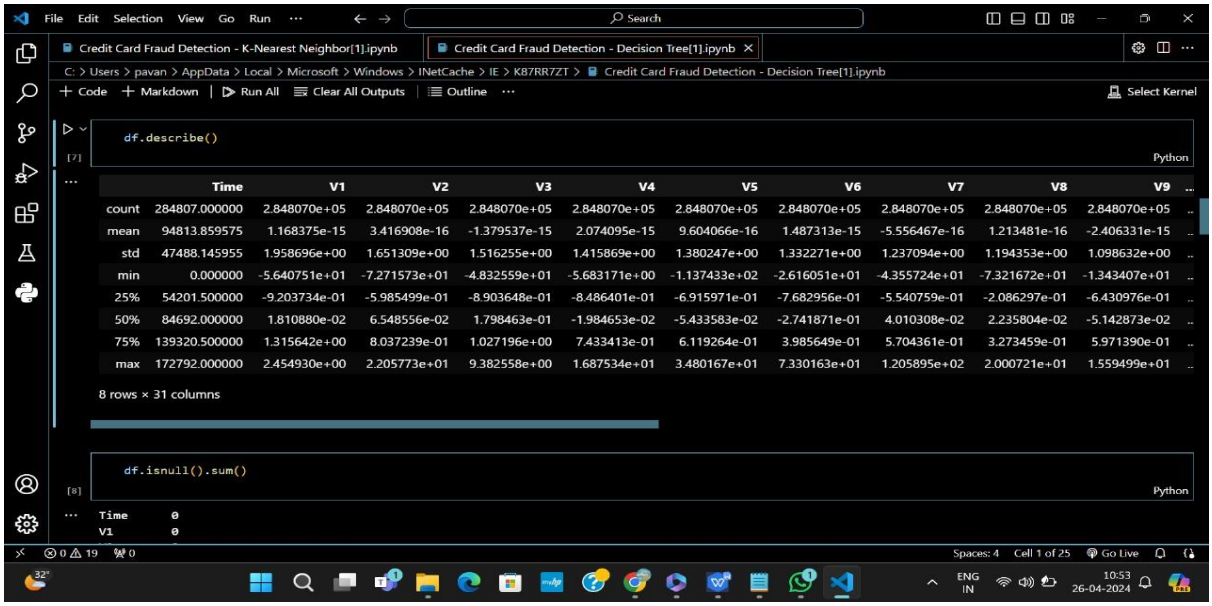
RESULT

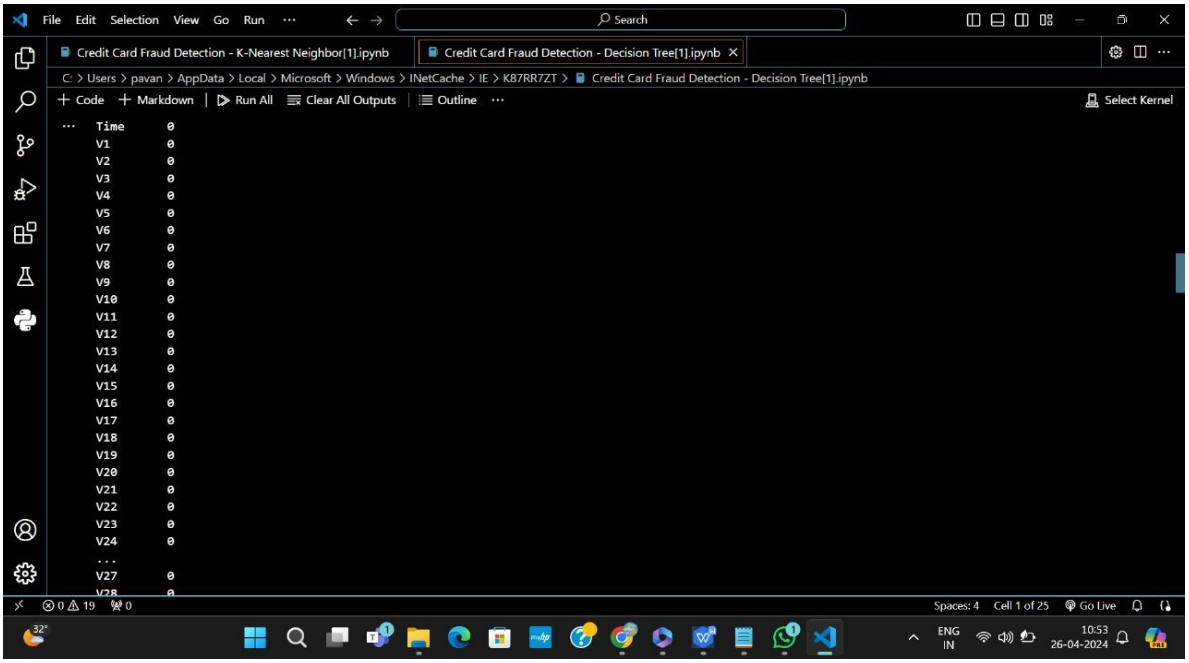
10.1 OUTPUT:

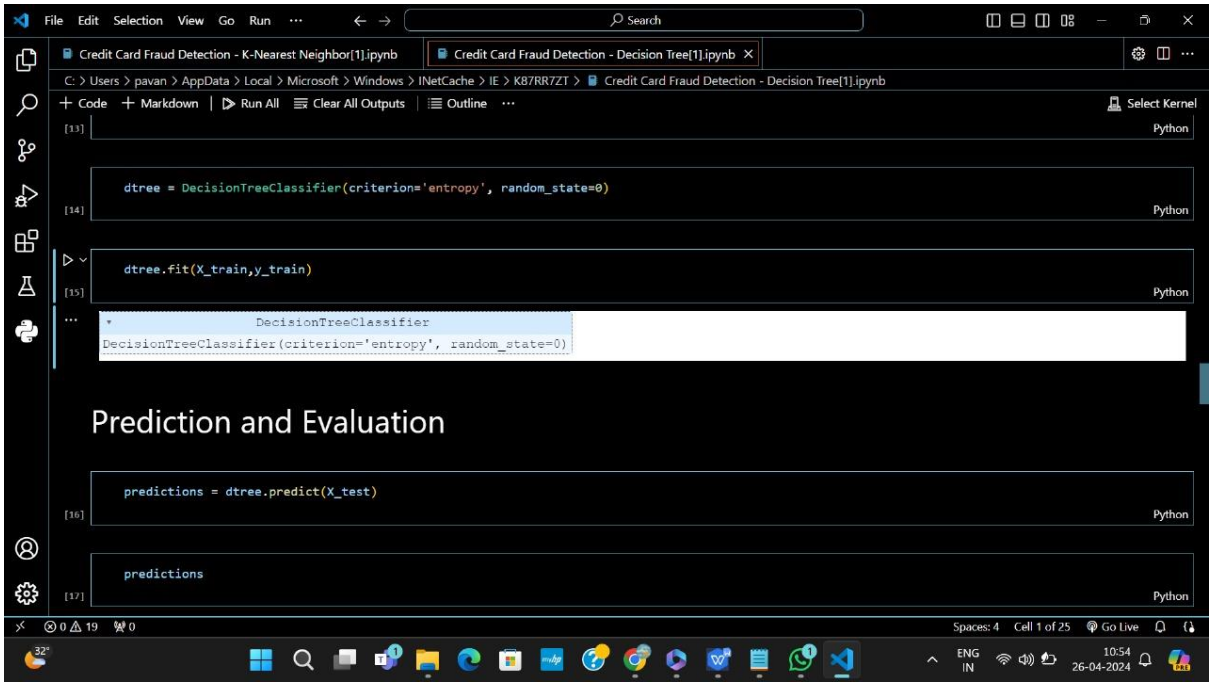
DECISION TREE

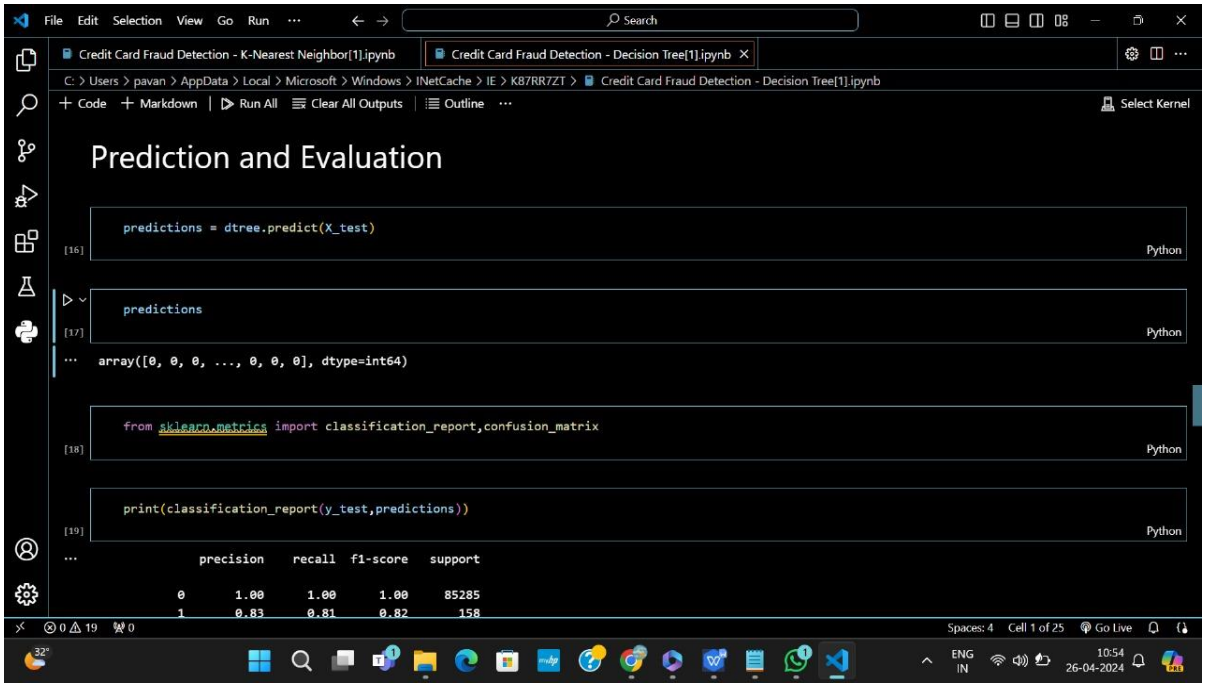


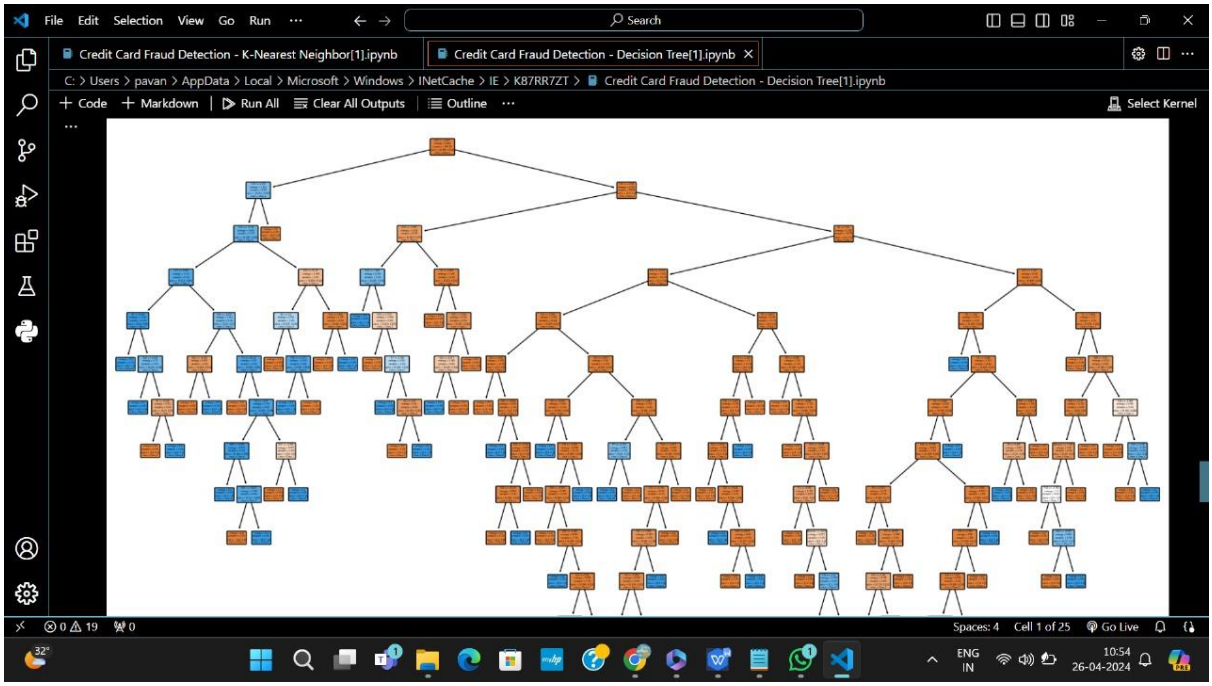


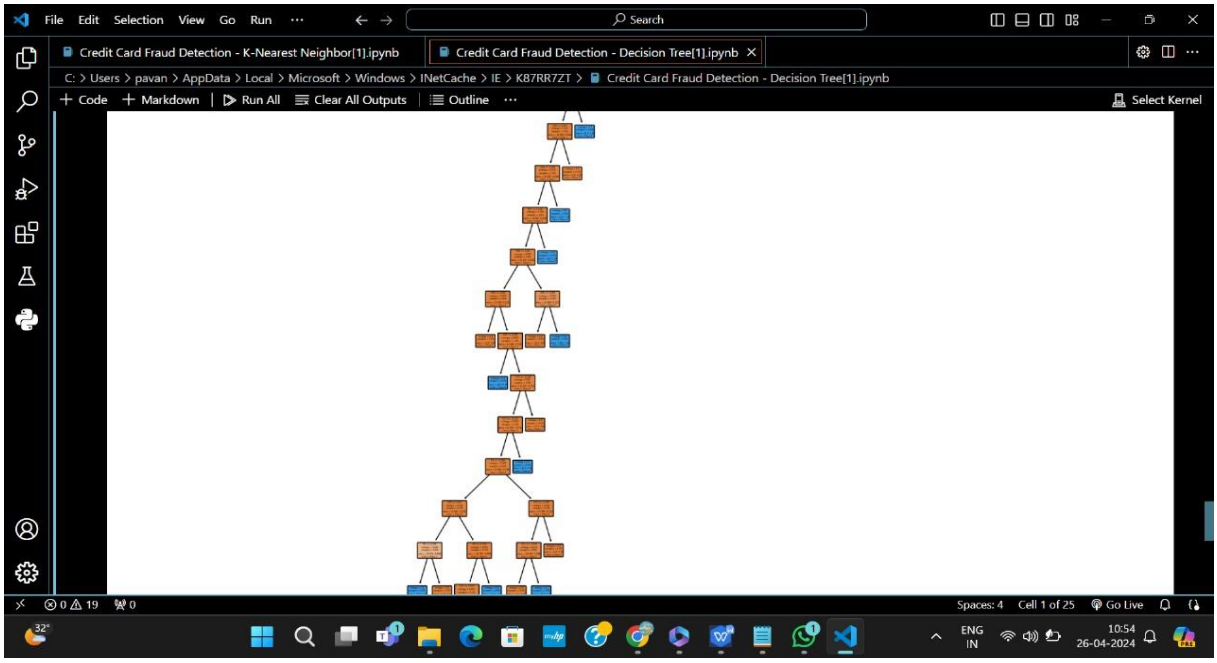

File Edit Selection View Go Run
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
Credit Card Fraud Detection - Decision Tree[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > K87RR7ZT > Credit Card Fraud Detection - Decision Tree[1].ipynb
+ Code + Markdown | Run All Clear All Outputs Outline
Select Kernel
df.describe()
Python
Time V1 V2 V3 V4 V5 V6 V7 V8 V9
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 2.848070e+05
mean 94813.859575 1.168375e-15 3.416908e-16 -1.379537e-15 2.074095e-15 9.604066e-16 1.487313e-15 -5.556467e-16 1.213481e-16 -2.406331e-15
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 1.098632e+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 -1.343407e+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 -6.430976e-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 -5.142873e-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 5.971390e-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 1.559499e+01
8 rows × 31 columns
df.isnull().sum()
Python
Time 0
V1 0
Spaces: 4 Cell 1 of 25 Go Live
ENG IN
10:53 26-04-2024


File Edit Selection View Go Run
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
Credit Card Fraud Detection - Decision Tree[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > K87RR7ZT > Credit Card Fraud Detection - Decision Tree[1].ipynb
+ Code + Markdown | Run All Clear All Outputs Outline
Select Kernel
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
...
V27 0
Spaces: 4 Cell 1 of 25 Go Live
ENG IN
10:53 26-04-2024

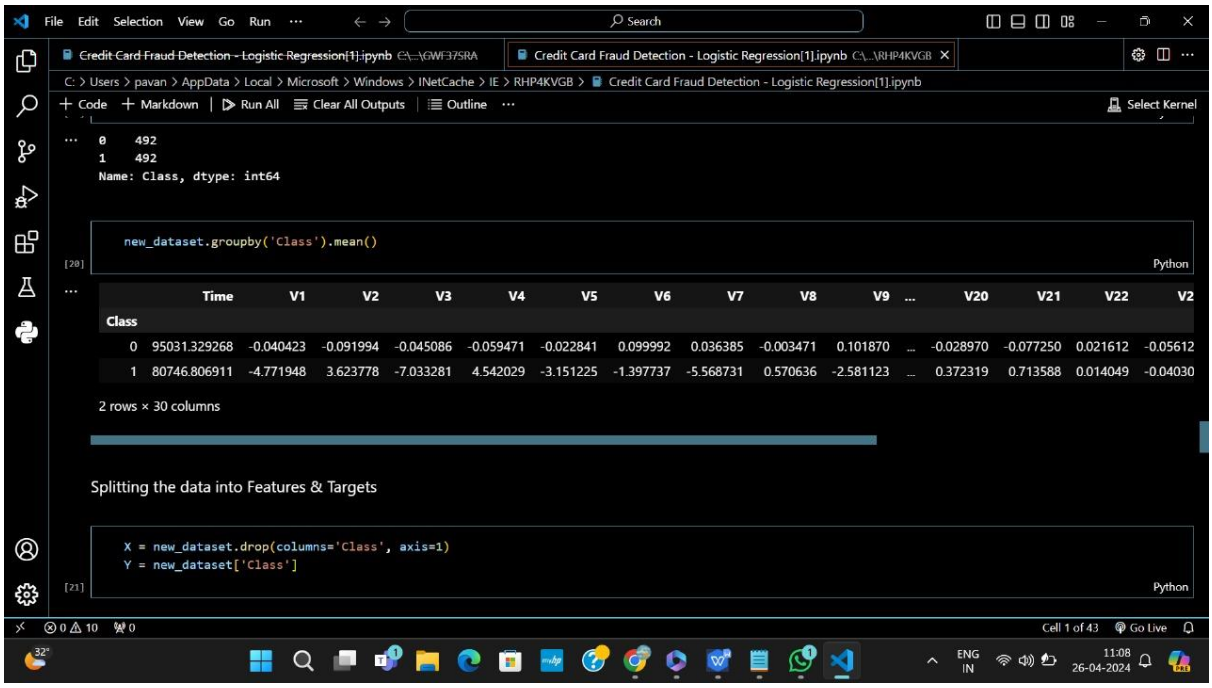

Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
Credit Card Fraud Detection - Decision Tree[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
[13]
Python
dtree = DecisionTreeClassifier(criterion='entropy', random_state=0)
[14]
Python
dtree.fit(X_train,y_train)
[15]
Python
DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', random_state=0)
Prediction and Evaluation
predictions = dtree.predict(X_test)
[16]
Python
predictions
[17]
Python

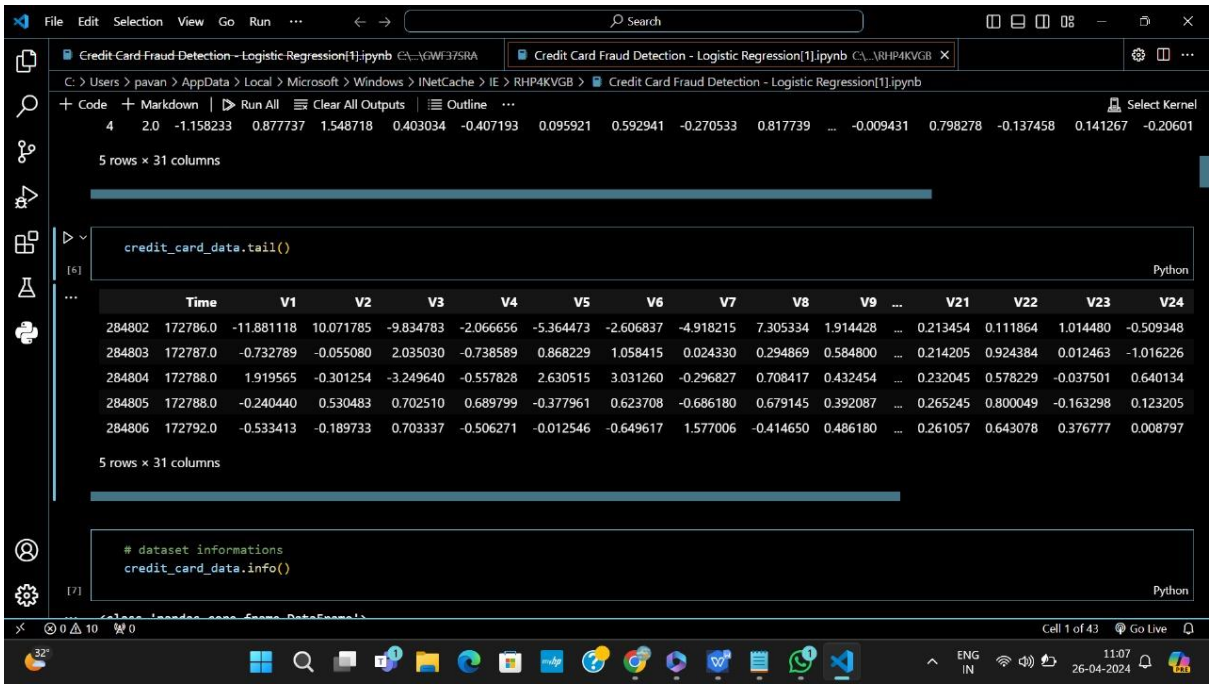

Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
Credit Card Fraud Detection - Decision Tree[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
Prediction and Evaluation
predictions = dtree.predict(X_test)
[16]
Python
predictions
[17]
Python
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
from sklearn.metrics import classification_report,confusion_matrix
[18]
Python
print(classification_report(y_test,predictions))
[19]
Python
precision recall f1-score support
0 1.00 1.00 1.00 85285
1 0.83 0.81 0.82 158

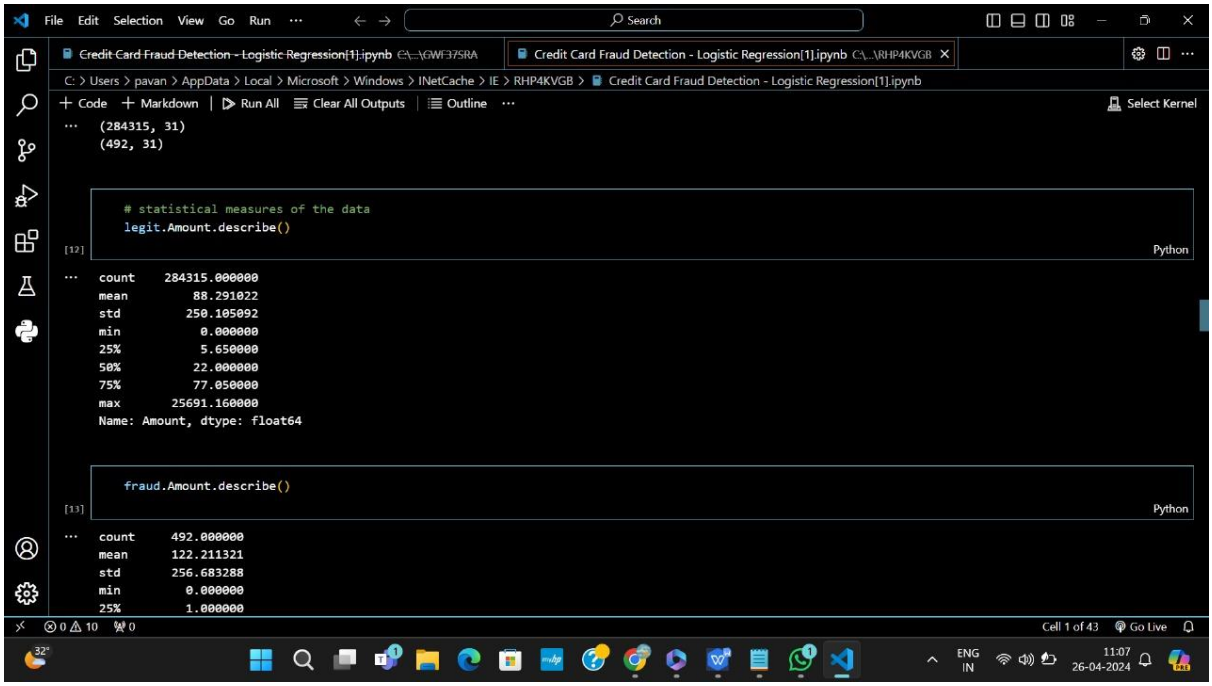

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
Credit Card Fraud Detection - Decision Tree[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > K87RR7ZT > Credit Card Fraud Detection - Decision Tree[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
Spaces: 4 Cell 1 of 25 Go Live
ENG IN 10:54 26-04-2024

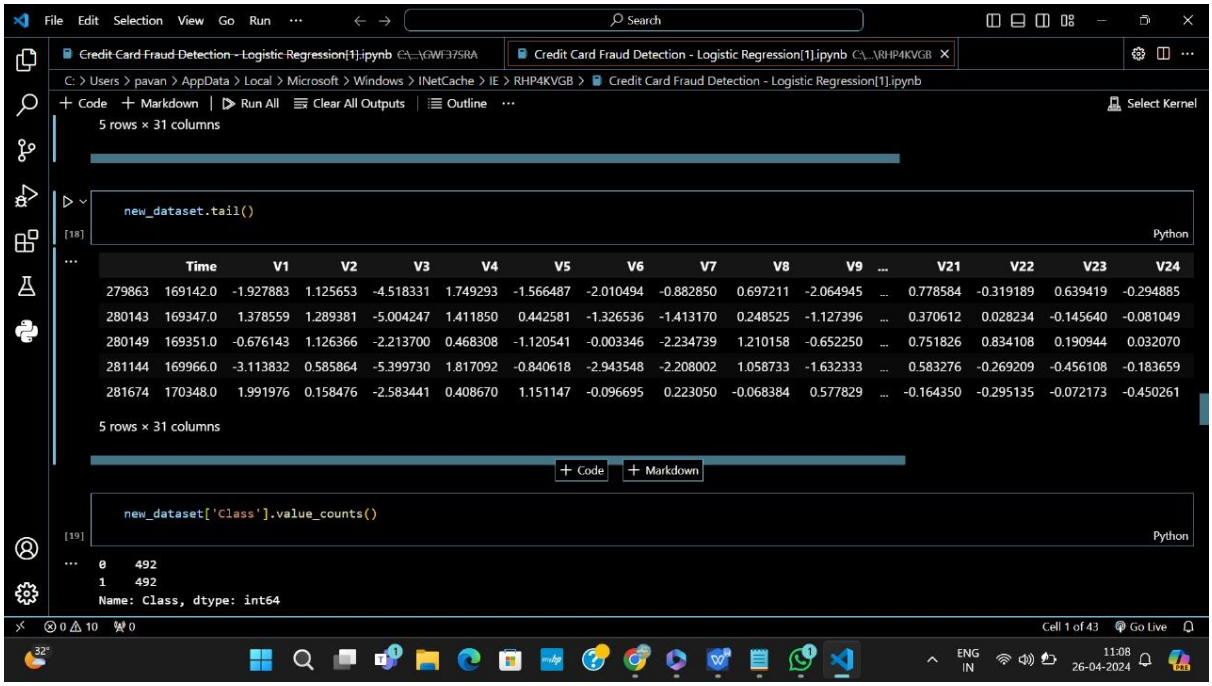

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
Credit Card Fraud Detection - Decision Tree[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > K87RR7ZT > Credit Card Fraud Detection - Decision Tree[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
Spaces: 4 Cell 1 of 25 Go Live
ENG IN 10:54 26-04-2024

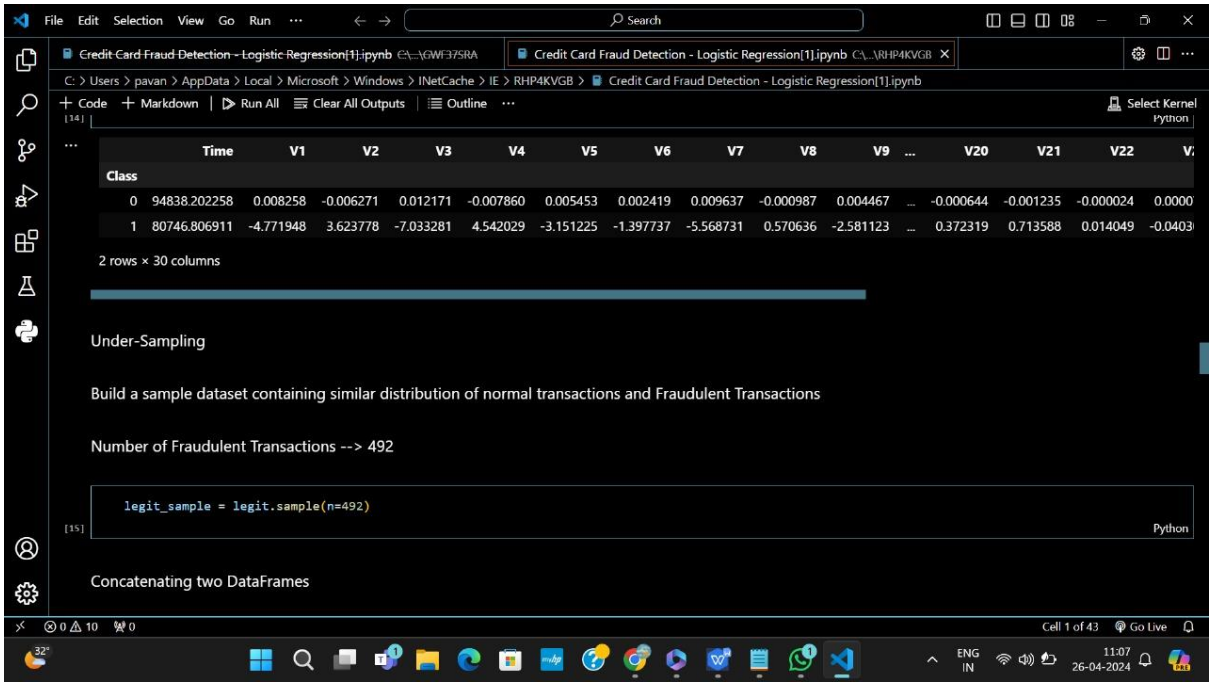
LOGISTIC REGRESSION



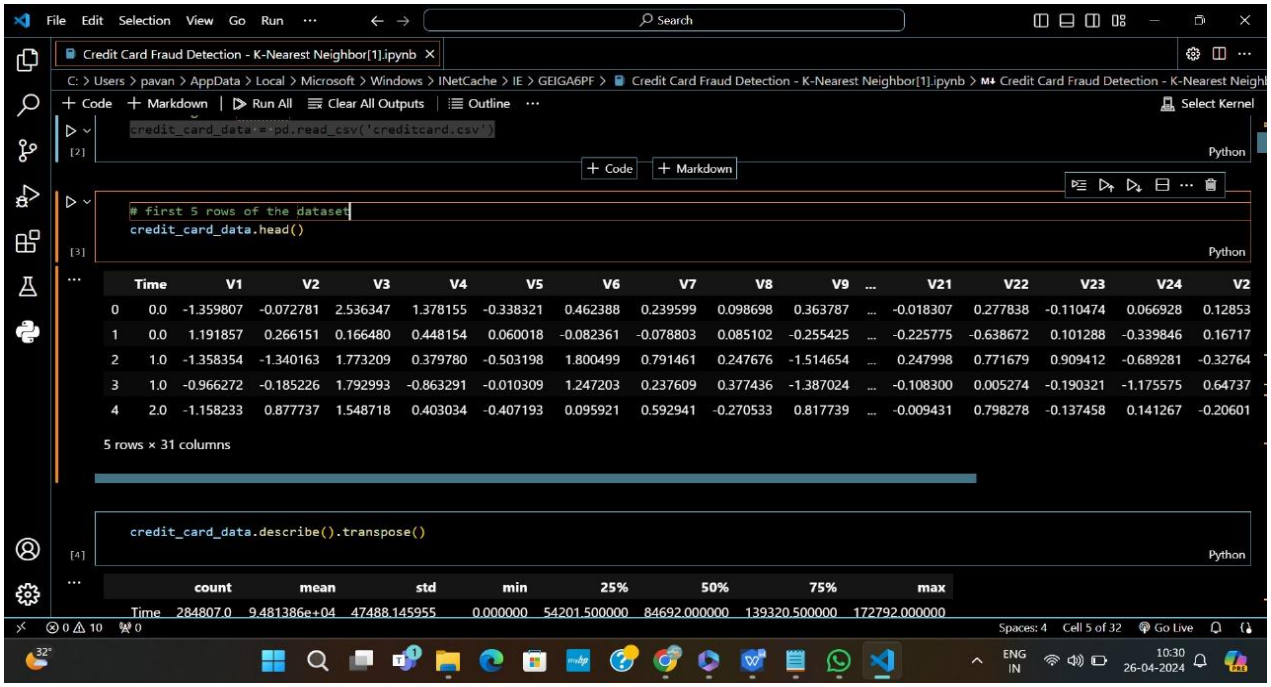


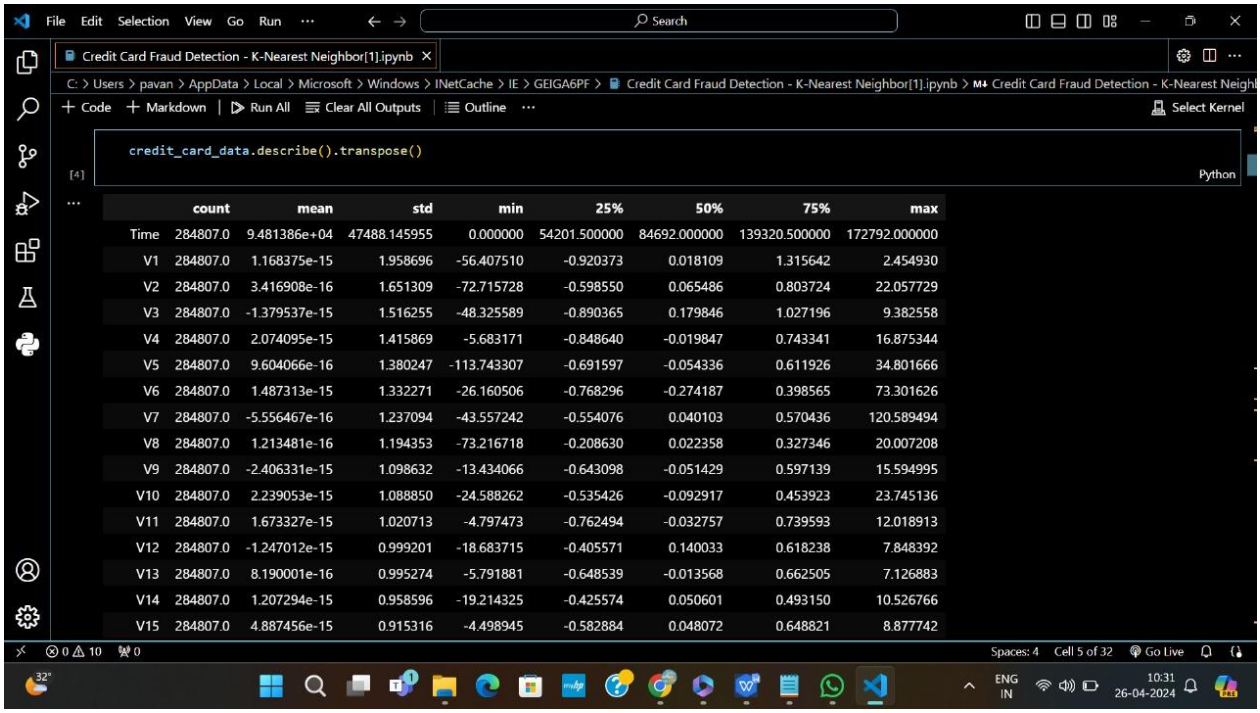

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - Logistic Regression[1].ipynb
C:\Users\pavan\AppData\Local\Microsoft\Windows\INetCache\IE\RHP4KVGB\Credit Card Fraud Detection - Logistic Regression[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
(284315, 31)
(492, 31)
# statistical measures of the data
legit.Amount.describe()
[12]
Python
count 284315.000000
mean 88.291022
std 250.105092
min 0.000000
25% 5.650000
50% 22.000000
75% 77.050000
max 25691.160000
Name: Amount, dtype: float64
fraud.Amount.describe()
[13]
Python
count 492.000000
mean 122.211321
std 256.683288
min 0.000000
25% 1.000000
Cell 1 of 43
Go Live
ENG IN
11:07
26-04-2024

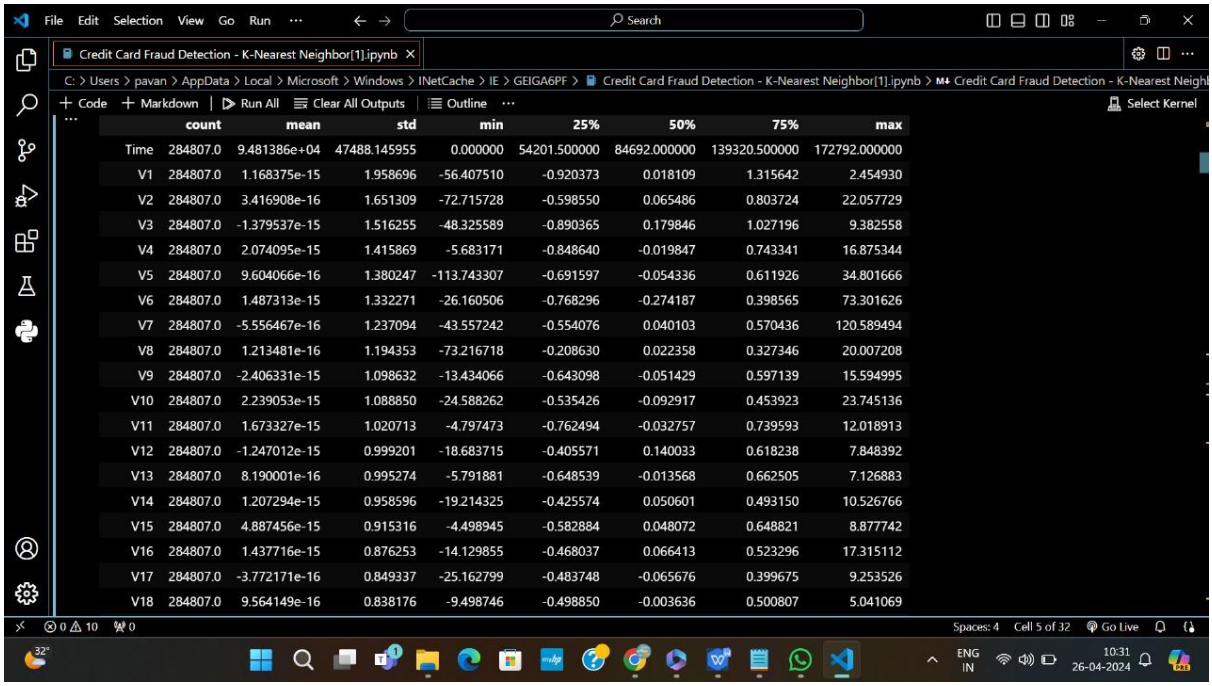

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - Logistic Regression[1].ipynb
C:\Users\pavan\AppData\Local\Microsoft\Windows\INetCache\IE\RHP4KVGB\Credit Card Fraud Detection - Logistic Regression[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
5 rows × 31 columns
new_dataset.tail()
[18]
Python
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... V21 V22 V23 V24
279863 169142.0 -1.927883 1.125653 -4.518331 1.749293 -1.566487 -2.010494 -0.882850 0.697211 -2.064945 ... 0.778584 -0.319189 0.639419 -0.294885
280143 169347.0 1.378559 1.289381 -5.004247 1.411850 0.442581 -1.326536 -1.413170 0.248525 -1.127396 ... 0.370612 0.028234 -0.145640 -0.081049
280149 169351.0 -0.676143 1.126366 -2.213700 0.468308 -1.120541 -0.003346 -2.234739 1.210158 -0.652250 ... 0.751826 0.834108 0.190944 0.032070
281144 169966.0 -3.113832 0.585864 -5.399730 1.817092 -0.840618 -2.943548 -2.208002 1.058733 -1.632333 ... 0.583276 -0.269209 -0.456108 -0.183659
281674 170348.0 1.991976 0.158476 -2.583441 0.408670 1.151147 -0.096695 0.223050 -0.068384 0.577829 ... -0.164350 -0.295135 -0.072173 -0.450261
5 rows × 31 columns
+ Code + Markdown
new_dataset['Class'].value_counts()
[19]
Python
0 492
1 492
Name: Class, dtype: int64
Cell 1 of 43
Go Live
ENG IN
11:08
26-04-2024


File Edit Selection View Go Run
Search
Credit Card Fraud Detection - Logistic Regression[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > RHP4KVGB > Credit Card Fraud Detection - Logistic Regression[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
Python
[14]
Time V1 V2 V3 V4 V5 V6 V7 V8 V9 ... V20 V21 V22
Class
0 94838.202258 0.008258 -0.006271 0.012171 -0.007860 0.005453 0.002419 0.009637 -0.000987 0.004467 ... -0.000644 -0.001235 -0.000024
1 80746.806911 -4.771948 3.623778 -7.033281 4.542029 -3.151225 -1.397737 -5.568731 0.570636 -2.581123 ... 0.372319 0.713588 0.014049
2 rows × 30 columns
Under-Sampling
Build a sample dataset containing similar distribution of normal transactions and Fraudulent Transactions
Number of Fraudulent Transactions --> 492
[35]
legit_sample = legit.sample(n=492)
Python
Concatenating two DataFrames
Cell 1 of 43
Go Live
ENG
IN
11:07
26-04-2024

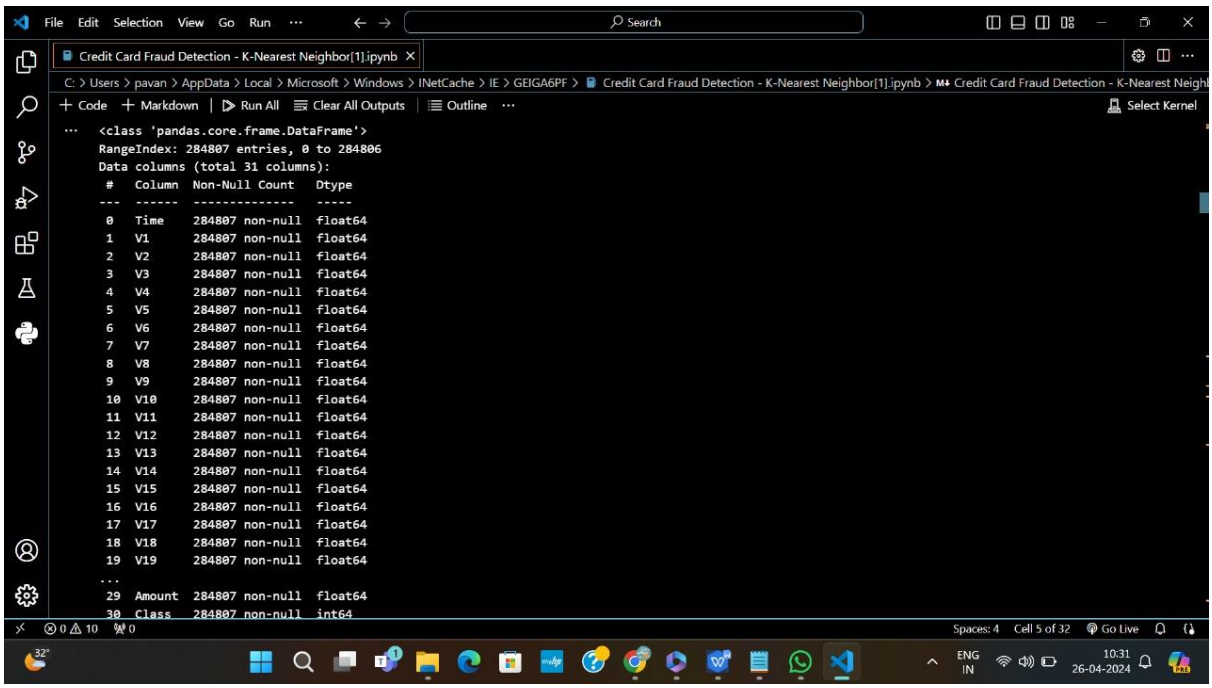
K – NEAREST NEIGHBOR



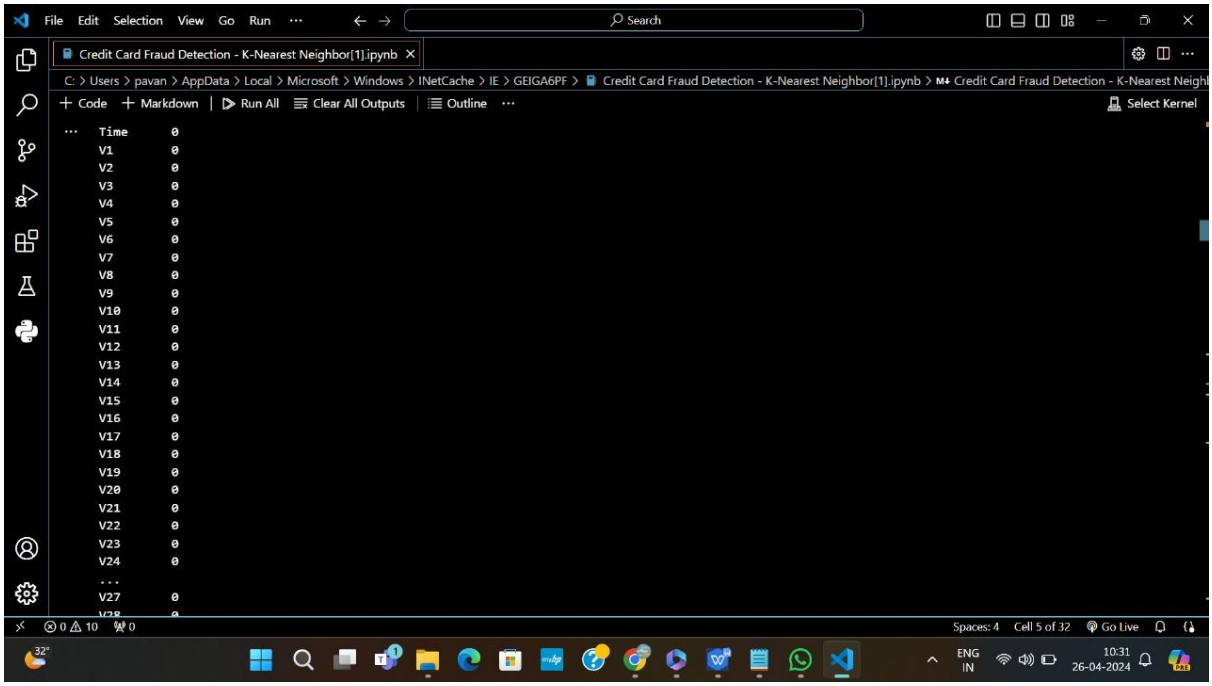


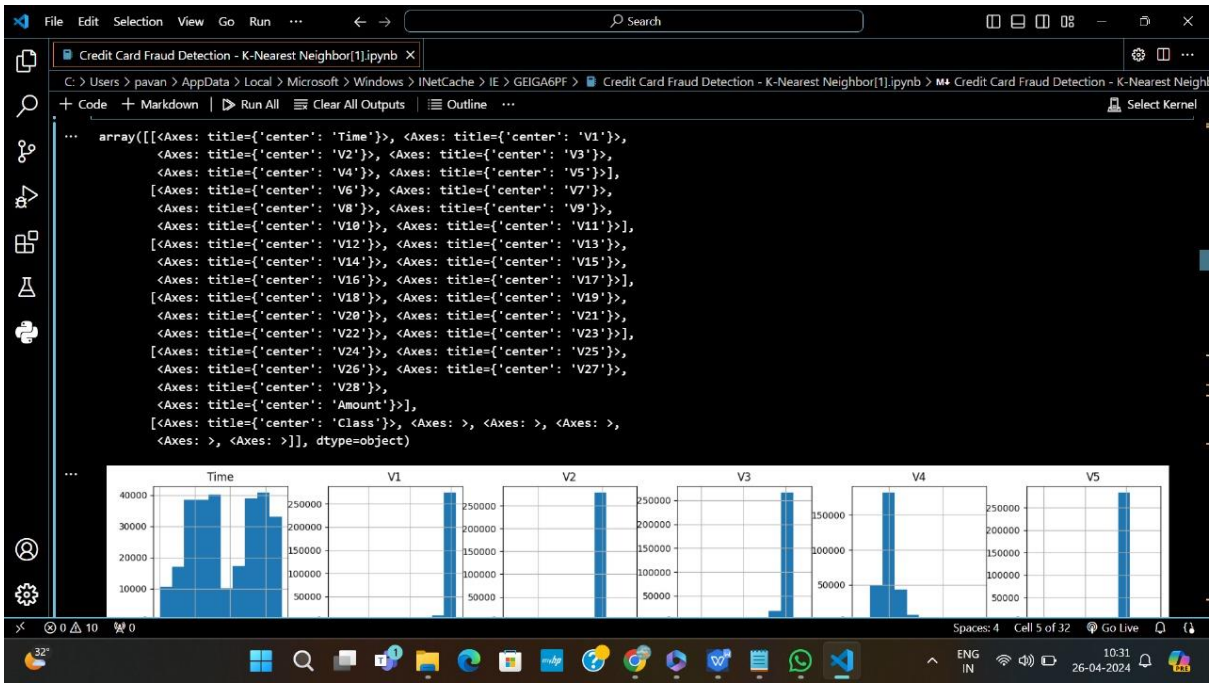


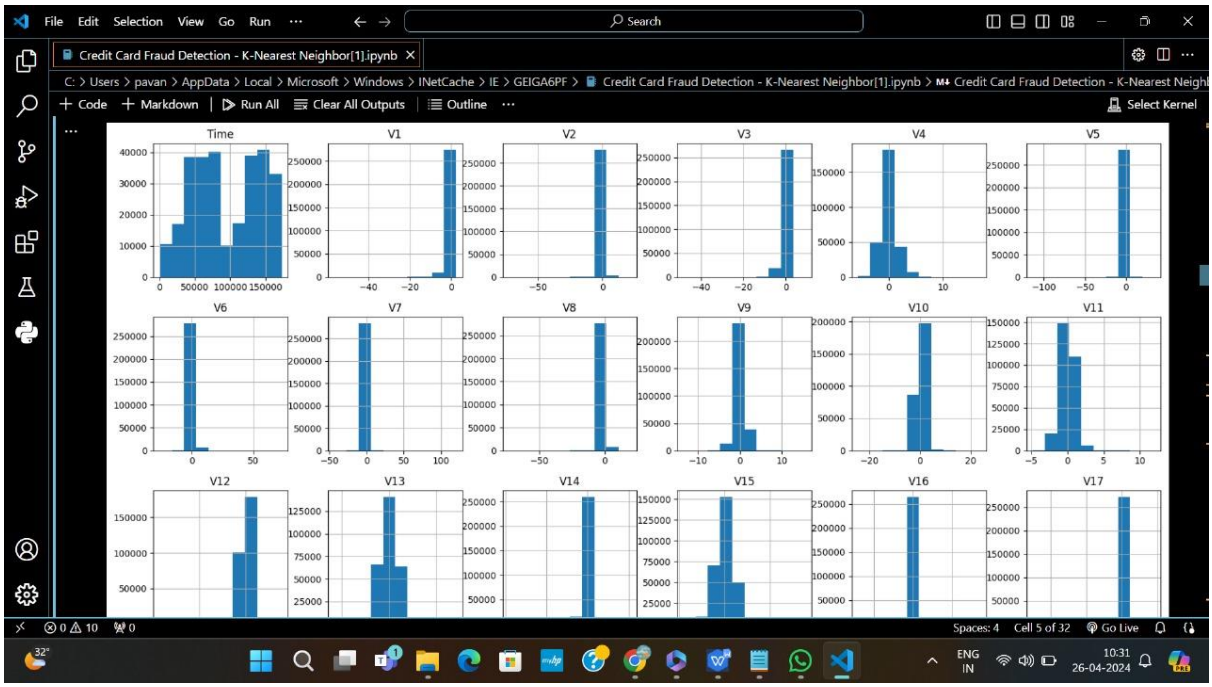
	count	mean	std	min	25%	50%	75%	max
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000	139320.500000	172792.000000
V1	284807.0	1.168375e-15	1.958696	-56.407510	-0.920373	0.018109	1.315642	2.454930
V2	284807.0	3.416908e-16	1.651309	-72.715728	-0.598550	0.065486	0.803724	22.057729
V3	284807.0	-1.379537e-15	1.516255	-48.325589	-0.890365	0.179846	1.027196	9.382558
V4	284807.0	2.074095e-15	1.415869	-5.683171	-0.848640	-0.019847	0.743341	16.875344
V5	284807.0	9.604066e-16	1.380247	-113.743307	-0.691597	-0.054336	0.611926	34.801666
V6	284807.0	1.487313e-15	1.332271	-26.160506	-0.768296	-0.274187	0.398565	73.301626
V7	284807.0	-5.556467e-16	1.237094	-43.557242	-0.554076	0.040103	0.570436	120.589494
V8	284807.0	1.213481e-16	1.194353	-73.216718	-0.208630	0.022358	0.327346	20.007208
V9	284807.0	-2.406331e-15	1.098632	-13.434066	-0.643098	-0.051429	0.597139	15.594995
V10	284807.0	2.239053e-15	1.088850	-24.588262	-0.535426	-0.092917	0.453923	23.745136
V11	284807.0	1.673327e-15	1.020713	-4.797473	-0.762494	-0.032757	0.739593	12.018913
V12	284807.0	-1.247012e-15	0.999201	-18.683715	-0.405571	0.140033	0.618238	7.848392
V13	284807.0	8.190001e-16	0.995274	-5.791881	-0.648539	-0.013568	0.662505	7.126883
V14	284807.0	1.207294e-15	0.958596	-19.214325	-0.425574	0.050601	0.493150	10.526766
V15	284807.0	4.887456e-15	0.915316	-4.498945	-0.582884	0.048072	0.648821	8.877742
V16	284807.0	1.437716e-15	0.876253	-14.129855	-0.468037	0.066413	0.523296	17.315112
V17	284807.0	-3.772171e-16	0.849337	-25.162799	-0.483748	-0.065676	0.399675	9.253526
V18	284807.0	9.564149e-16	0.838176	-9.498746	-0.498850	-0.003636	0.500807	5.041069

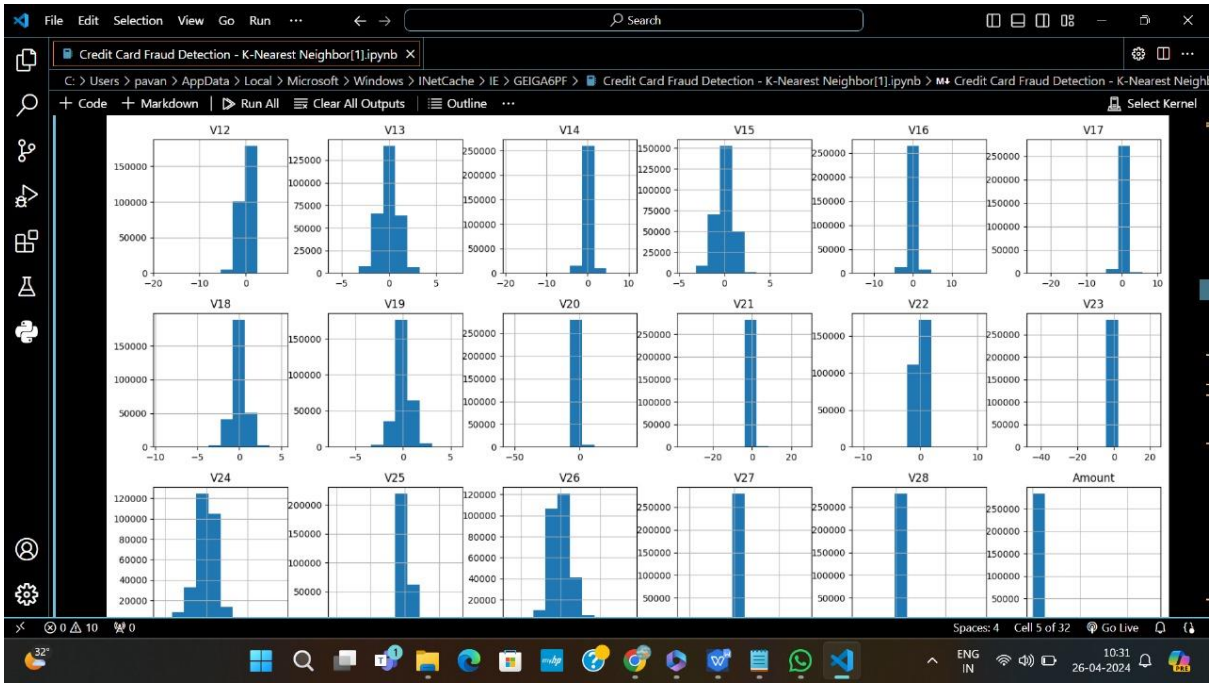


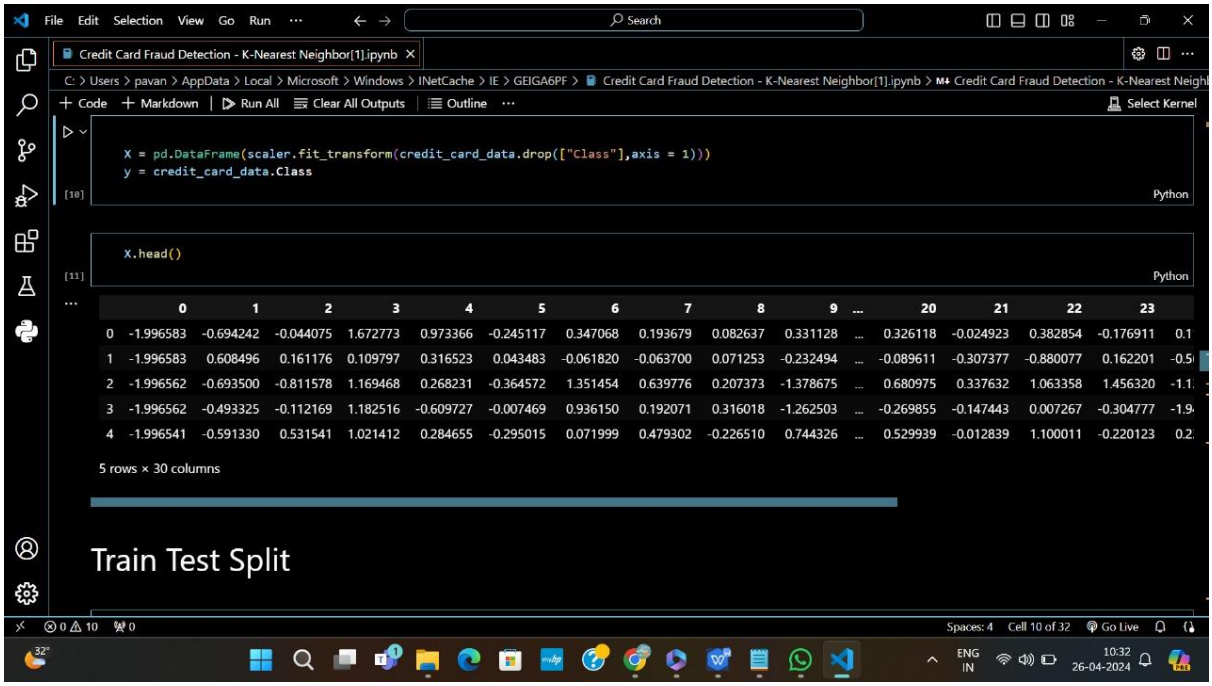

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 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
...
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
```

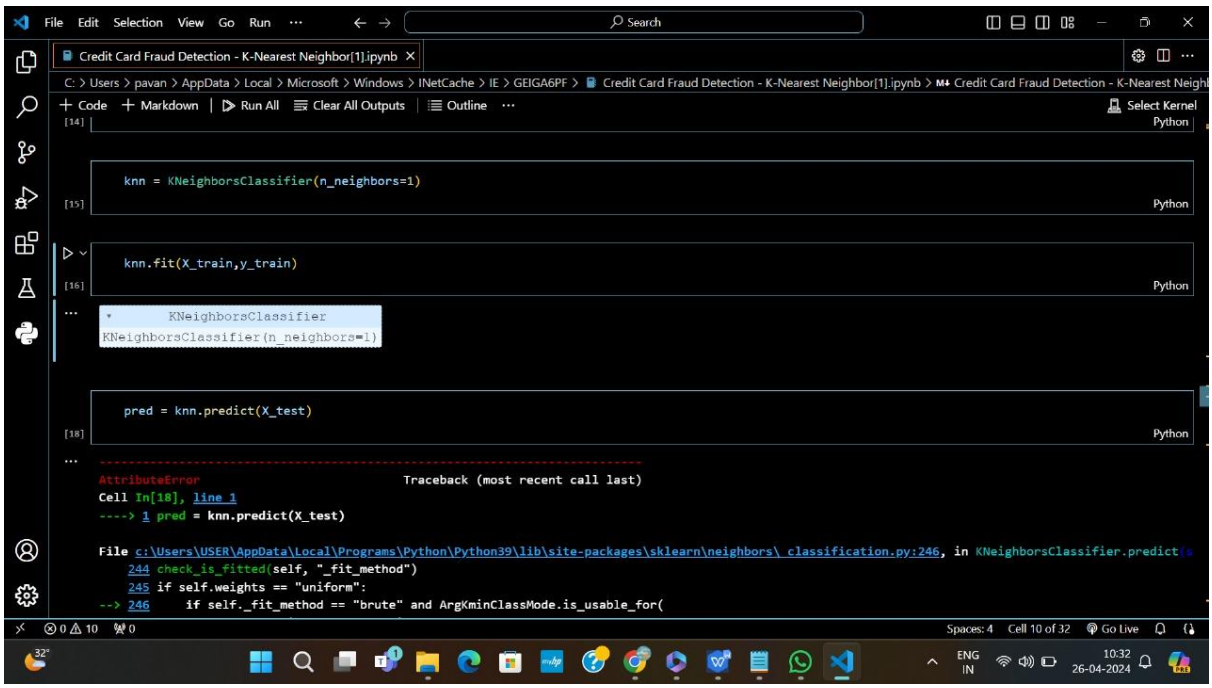

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > GEIGA6PF > Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb > Credit Card Fraud Detection - K-Nearest Neigh
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
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
V27 0
Spaces: 4 Cell 5 of 32 Go Live
ENG IN 10:31 26-04-2024

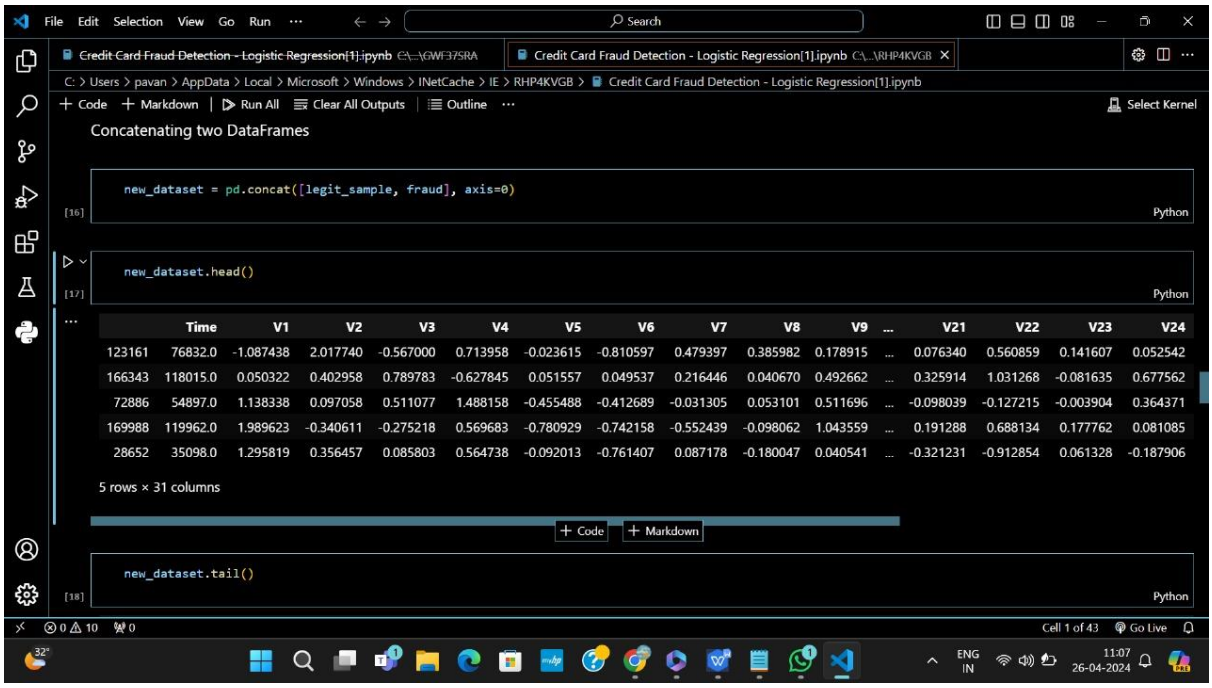

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > GEIGA6PF > Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb > Credit Card Fraud Detection - K-Nearest Neigh
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
array([[<Axes: title={'center': 'Time'}>, <Axes: title={'center': 'V1'}>,
<Axes: title={'center': 'V2'}>, <Axes: title={'center': 'V3'}>,
<Axes: title={'center': 'V4'}>, <Axes: title={'center': 'V5'}>],
[<Axes: title={'center': 'V6'}>, <Axes: title={'center': 'V7'}>,
<Axes: title={'center': 'V8'}>, <Axes: title={'center': 'V9'}>,
<Axes: title={'center': 'V10'}>, <Axes: title={'center': 'V11'}>],
[<Axes: title={'center': 'V12'}>, <Axes: title={'center': 'V13'}>,
<Axes: title={'center': 'V14'}>, <Axes: title={'center': 'V15'}>,
<Axes: title={'center': 'V16'}>, <Axes: title={'center': 'V17'}>],
[<Axes: title={'center': 'V18'}>, <Axes: title={'center': 'V19'}>,
<Axes: title={'center': 'V20'}>, <Axes: title={'center': 'V21'}>,
<Axes: title={'center': 'V22'}>, <Axes: title={'center': 'V23'}>],
[<Axes: title={'center': 'V24'}>, <Axes: title={'center': 'V25'}>,
<Axes: title={'center': 'V26'}>, <Axes: title={'center': 'V27'}>,
<Axes: title={'center': 'V28'}>,
<Axes: title={'center': 'Amount'}>],
[<Axes: title={'center': 'Class'}>, <Axes: >, <Axes: >,
<Axes: >, <Axes: >]], dtype=object)
Time
V1
V2
V3
V4
V5
Spaces: 4 Cell 5 of 32 Go Live
ENG IN 10:31 26-04-2024

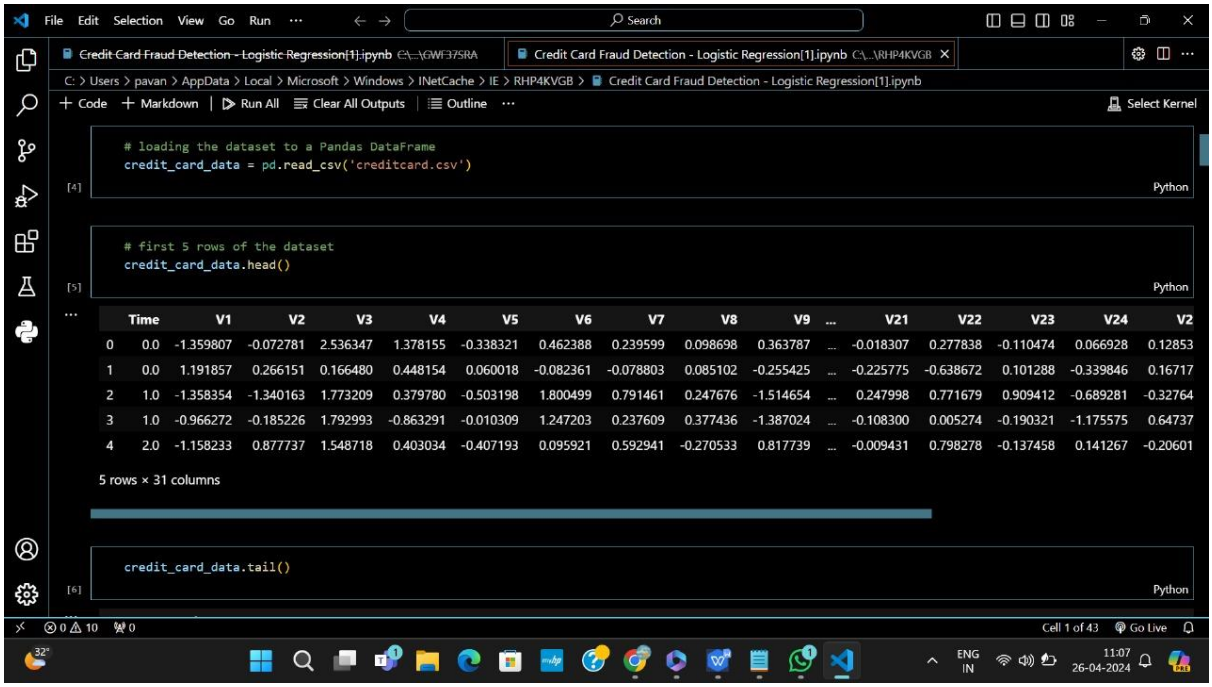

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > GEIGA6PF > Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb > Credit Card Fraud Detection - K-Nearest Neigh
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
Spaces: 4 Cell 5 of 32 Go Live
ENG IN
10:31 26-04-2024

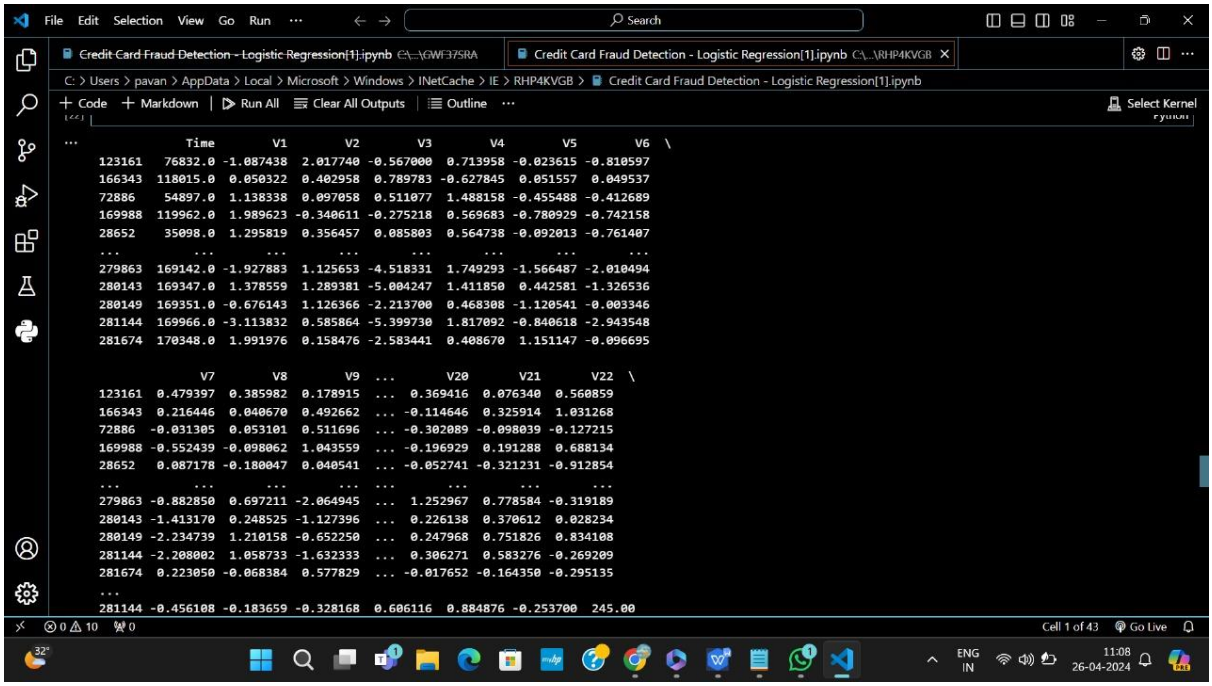

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > GEIGA6PF > Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb > Credit Card Fraud Detection - K-Nearest Neigh
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Spaces: 4 Cell 5 of 32 Go Live
ENG IN
10:31 26-04-2024


File Edit Selection View Go Run
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
X = pd.DataFrame(scaler.fit_transform(credit_card_data.drop(["Class"],axis = 1)))
y = credit_card_data.Class
[10]
Python
X.head()
[11]
Python
0 1 2 3 4 5 6 7 8 9 ... 20 21 22 23
0 -1.996583 -0.694242 -0.044075 1.672773 0.973366 -0.245117 0.347068 0.193679 0.082637 0.331128 ... 0.326118 -0.024923 0.382854 -0.176911 0.1
1 -1.996583 0.608496 0.161176 0.109797 0.316523 0.043483 -0.061820 -0.063700 0.071253 -0.232494 ... -0.089611 -0.307377 -0.880077 0.162201 -0.5
2 -1.996562 -0.693500 -0.811578 1.169468 0.268231 -0.364572 1.351454 0.639776 0.207373 -1.378675 ... 0.680975 0.337632 1.063358 1.456320 -1.1
3 -1.996562 -0.493325 -0.112169 1.182516 -0.609727 -0.007469 0.936150 0.192071 0.316018 -1.262503 ... -0.269855 -0.147443 0.007267 -0.304777 -1.9
4 -1.996541 -0.591330 0.531541 1.021412 0.284655 -0.295015 0.071999 0.479302 -0.226510 0.744326 ... 0.529939 -0.012839 1.100011 -0.220123 0.2
5 rows × 30 columns
Train Test Split
Spaces: 4 Cell 10 of 32 Go Live
ENG IN 10:32 26-04-2024


File Edit Selection View Go Run
Credit Card Fraud Detection - K-Nearest Neighbor[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
[14]
Python
knn = KNeighborsClassifier(n_neighbors=1)
[15]
Python
knn.fit(X_train,y_train)
[16]
Python
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=1)
pred = knn.predict(X_test)
[18]
Python
AttributeError Traceback (most recent call last)
Cell In[18], line 1
----> 1 pred = knn.predict(X_test)
File c:\Users\USER\AppData\Local\Programs\Python\Python39\lib\site-packages\sklearn\neighbors_classification.py:246, in KNeighborsClassifier.predict
244 check_is_fitted(self, "_fit_method")
245 if self.weights == "uniform":
--> 246 if self._fit_method == "brute" and ArgKminClassMode.is_usable_for(
Spaces: 4 Cell 10 of 32 Go Live
ENG IN 10:32 26-04-2024

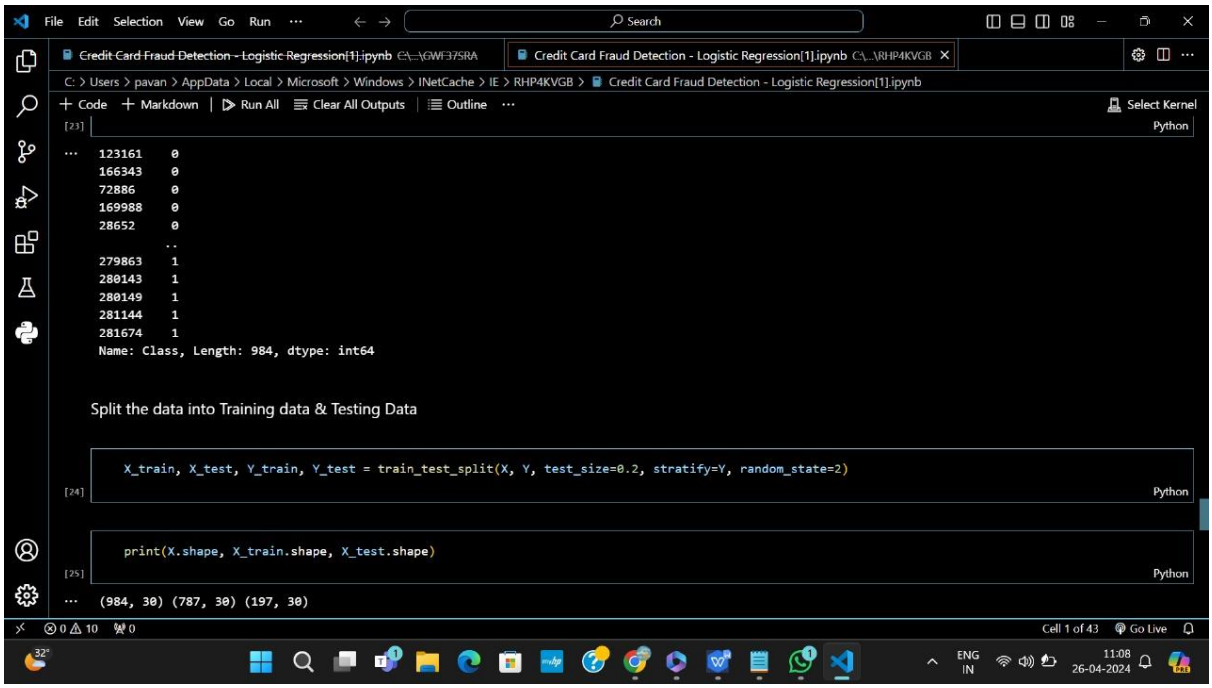

Concatenating two DataFrames
new_dataset = pd.concat([legit_sample, fraud], axis=0)
new_dataset.head()
new_dataset.tail()
5 rows × 31 columns


# loading the dataset to a Pandas DataFrame
credit_card_data = pd.read_csv('creditcard.csv')
# first 5 rows of the dataset
credit_card_data.head()
credit_card_data.tail()
5 rows × 31 columns



```
             Time        V1        V2        V3        V4        V5        V6  \
123161   76832.0 -1.087438  2.017740 -0.567000  0.713958 -0.023615 -0.810597
166343  118015.0  0.050322  0.402958  0.789783 -0.627845  0.051557  0.049537
72886    54897.0  1.138338  0.097058  0.511077  1.488158 -0.455488 -0.412689
169988  119962.0  1.989623 -0.340611 -0.275218  0.569683 -0.780929 -0.742158
28652    35098.0  1.295819  0.356457  0.085803  0.564738 -0.092013 -0.761407
...          ...       ...       ...       ...       ...       ...       ...
279863  169142.0 -1.927883  1.125653 -4.518331  1.749293 -1.566487 -2.010494
280143  169347.0  1.378559  1.289381 -5.004247  1.411850  0.442581 -1.326536
280149  169351.0 -0.676143  1.126366 -2.213700  0.468308 -1.120541 -0.003346
281144  169966.0 -3.113832  0.585864 -5.399730  1.817092 -0.840618 -2.943548
281674  170348.0  1.991976  0.158476 -2.583441  0.408670  1.151147 -0.096695

               V7        V8        V9  ...       V20       V21       V22  \
123161  0.479397  0.385982  0.178915  ...  0.369416  0.076340  0.560859
166343  0.216446  0.040670  0.492662  ... -0.114646  0.325914  1.031268
72886  -0.031305  0.053101  0.511696  ... -0.302089 -0.098039 -0.127215
169988 -0.552439 -0.098062  1.043559  ... -0.196929  0.191288  0.688134
28652   0.087178 -0.180047  0.040541  ... -0.052741 -0.321231 -0.912854
...          ...       ...       ...  ...       ...       ...       ...
279863 -0.882850  0.697211 -2.064945  ...  1.252967  0.778584 -0.319189
280143 -1.413170  0.248525 -1.127396  ...  0.226138  0.370612  0.028234
280149 -2.234739  1.210158 -0.652250  ...  0.247968  0.751826  0.834108
281144 -2.208002  1.058733 -1.632333  ...  0.306271  0.583276 -0.269209
281674  0.223050 -0.068384  0.577829  ... -0.017652 -0.164350 -0.295135
...
281144 -0.456108 -0.183659 -0.328168  0.606116  0.884876 -0.253700  245.00
```



```
123161    0
166343    0
72886     0
169988    0
28652     0
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

Split the data into Training data & Testing Data

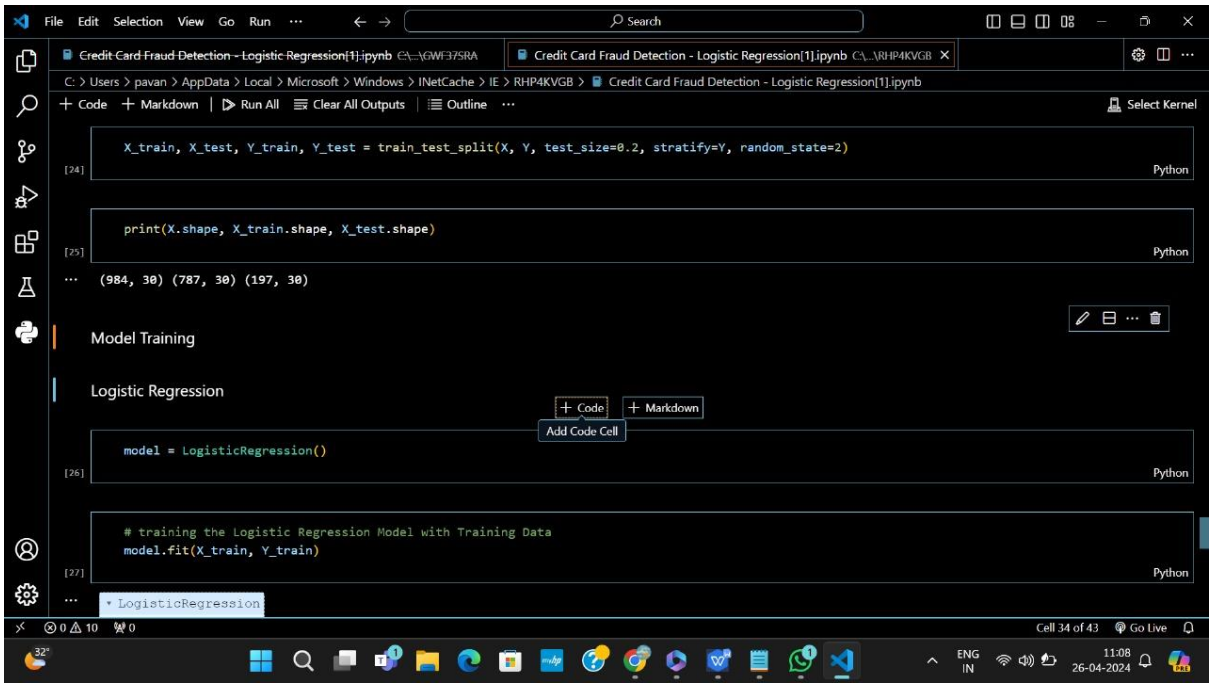
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
```

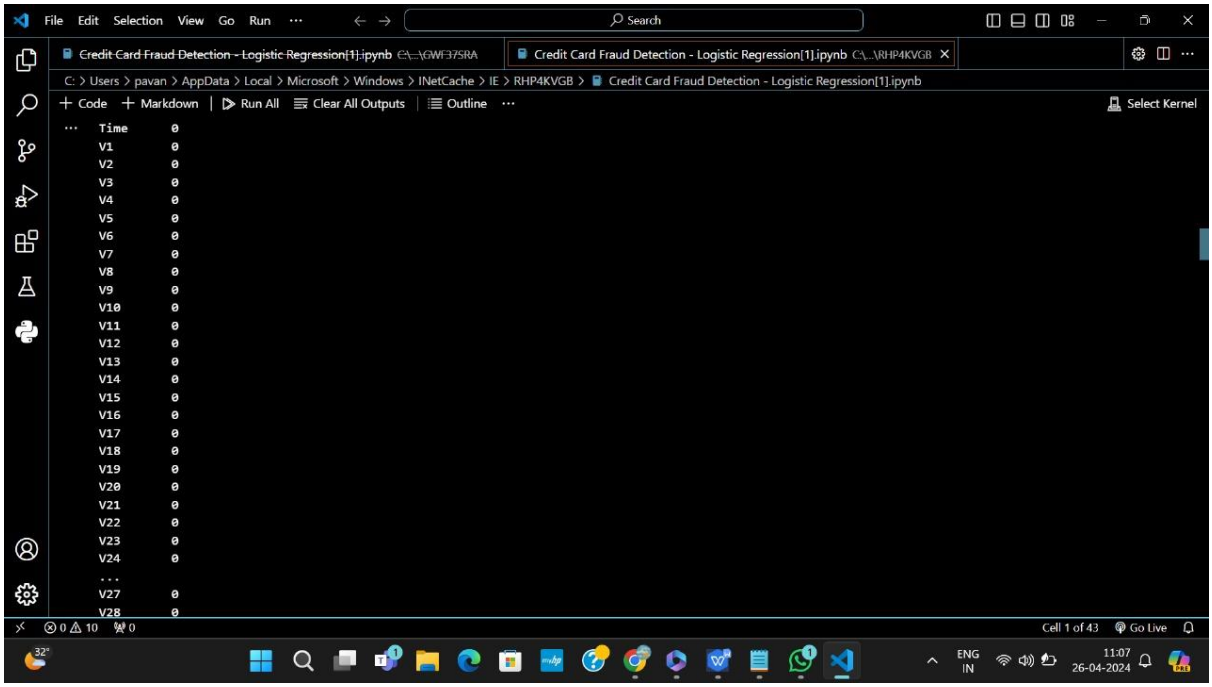
[24] Python

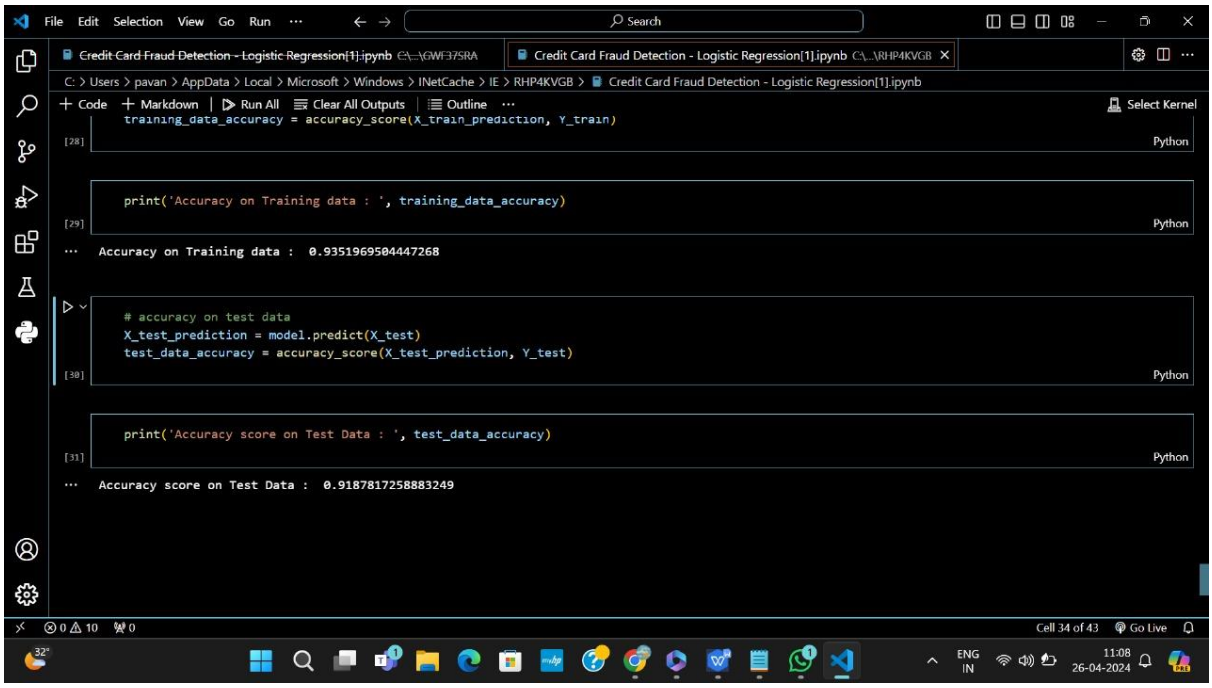
```
print(X.shape, X_train.shape, X_test.shape)
```

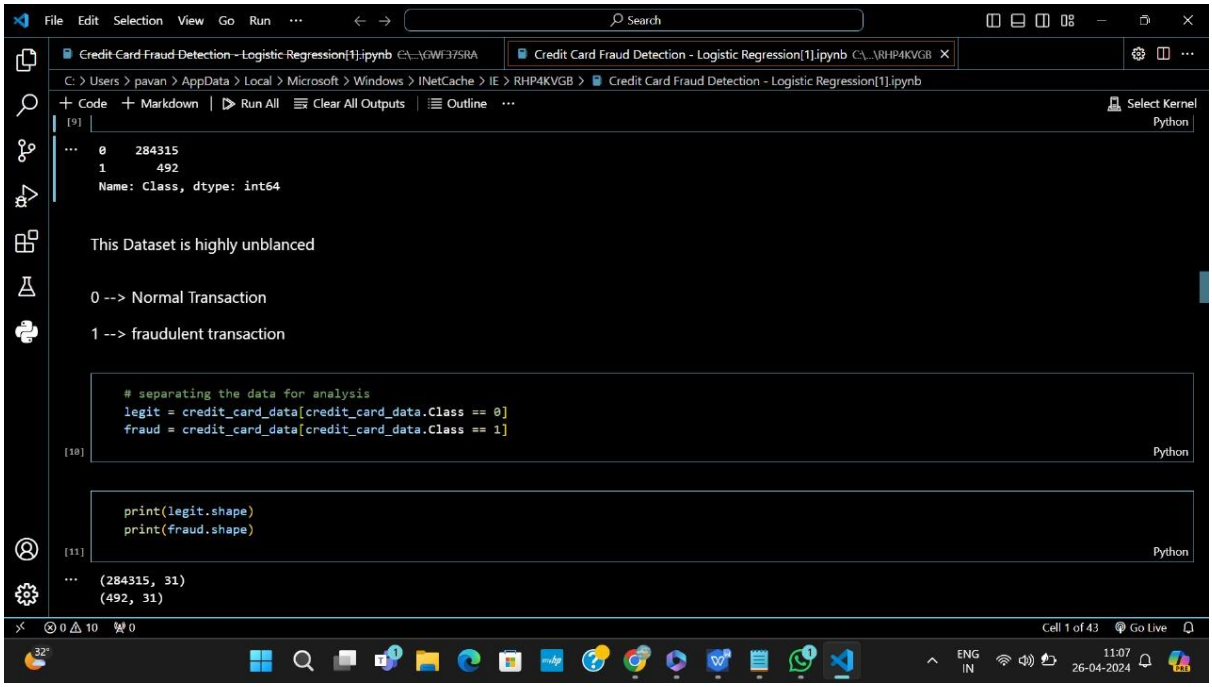
[25] Python

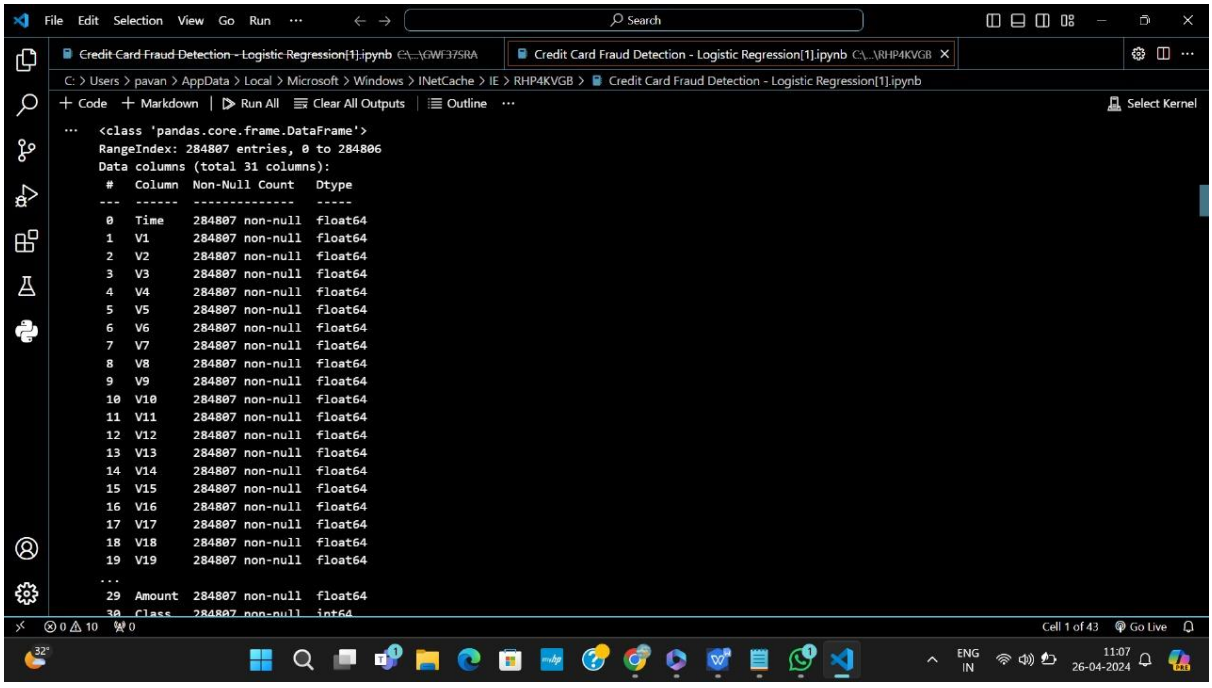
```
(984, 30) (787, 30) (197, 30)
```


File Edit Selection View Go Run
Search
Credit Card Fraud Detection - Logistic Regression[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > RHP4KVGB > Credit Card Fraud Detection - Logistic Regression[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, stratify=Y, random_state=2)
[24]
Python
print(X.shape, X_train.shape, X_test.shape)
[25]
Python
(984, 30) (787, 30) (197, 30)
Model Training
Logistic Regression
+ Code
+ Markdown
Add Code Cell
model = LogisticRegression()
[26]
Python
# training the Logistic Regression Model with Training Data
model.fit(X_train, Y_train)
[27]
Python
LogisticRegression
Cell 34 of 43
Go Live
ENG
IN
11:08
26-04-2024

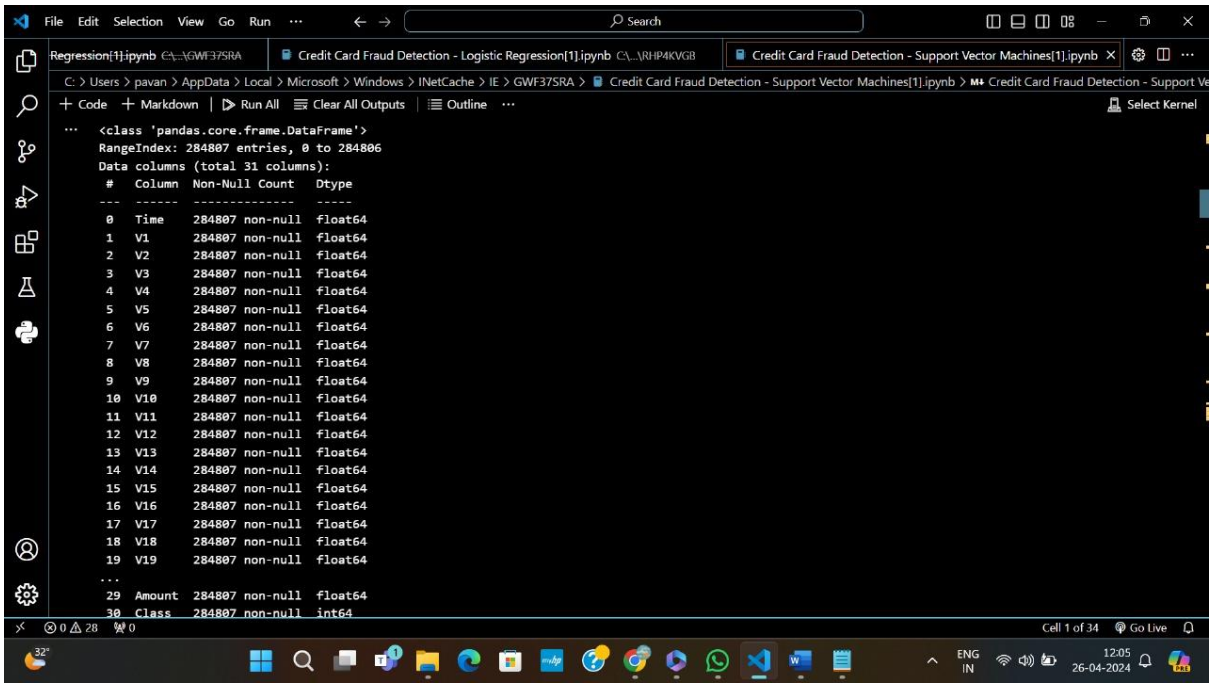

File Edit Selection View Go Run
Search
Credit Card Fraud Detection - Logistic Regression[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > RHP4KVGB > Credit Card Fraud Detection - Logistic Regression[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
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
...
V27 0
V28 0
Cell 1 of 43
Go Live
ENG
IN
11:07
26-04-2024

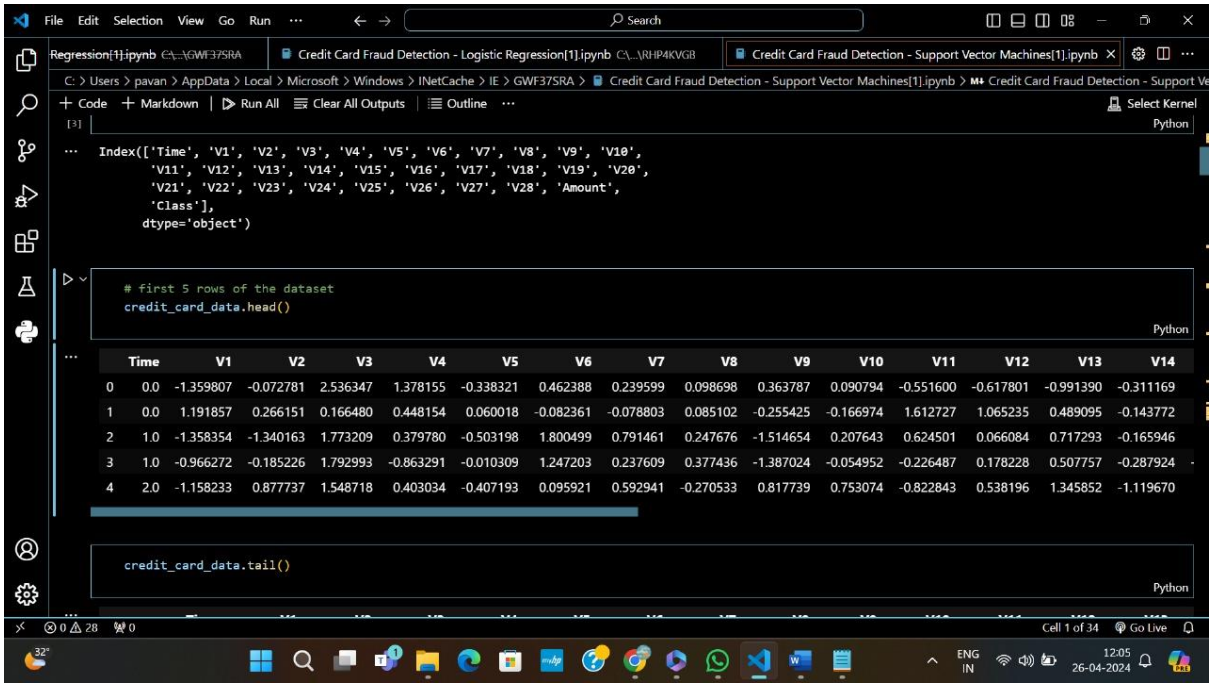

File Edit Selection View Go Run
Credit Card Fraud Detection - Logistic Regression[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
Python
print('Accuracy on Training data : ', training_data_accuracy)
Accuracy on Training data : 0.9351969504447268
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
print('Accuracy score on Test Data : ', test_data_accuracy)
Accuracy score on Test Data : 0.9187817258883249

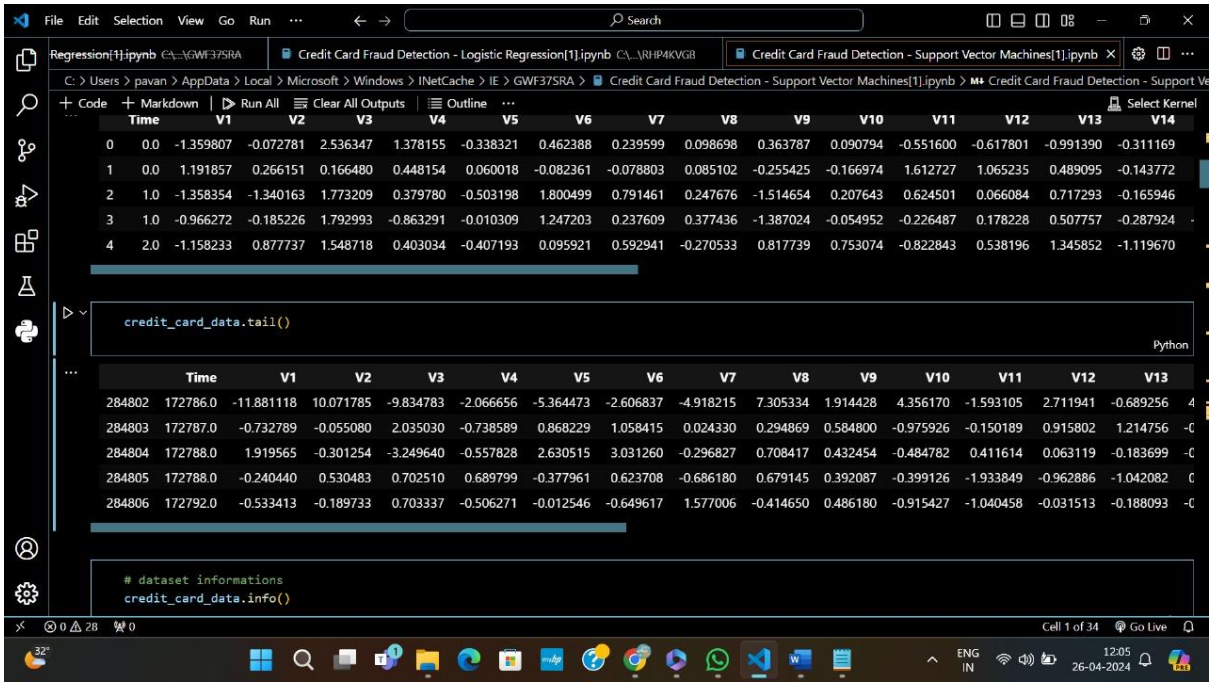

File Edit Selection View Go Run
Credit Card Fraud Detection - Logistic Regression[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
0 284315
1 492
Name: Class, dtype: int64
This Dataset is highly unblanced
0 --> Normal Transaction
1 --> fraudulent transaction
# separating the data for analysis
legit = credit_card_data[credit_card_data.Class == 0]
fraud = credit_card_data[credit_card_data.Class == 1]
print(legit.shape)
print(fraud.shape)
(284315, 31)
(492, 31)


File Edit Selection View Go Run
Search
Credit Card Fraud Detection - Logistic Regression[1].ipynb
Credit Card Fraud Detection - Logistic Regression[1].ipynb
C: > Users > pavan > AppData > Local > Microsoft > Windows > INetCache > IE > RHP4KVGB > Credit Card Fraud Detection - Logistic Regression[1].ipynb
+ Code + Markdown | Run All Clear All Outputs | Outline
Select Kernel
<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
...
29 Amount 284807 non-null float64
Cell 1 of 43
Go Live
ENG IN
11:07 26-04-2024

SUPPORT VECTOR MACHINES







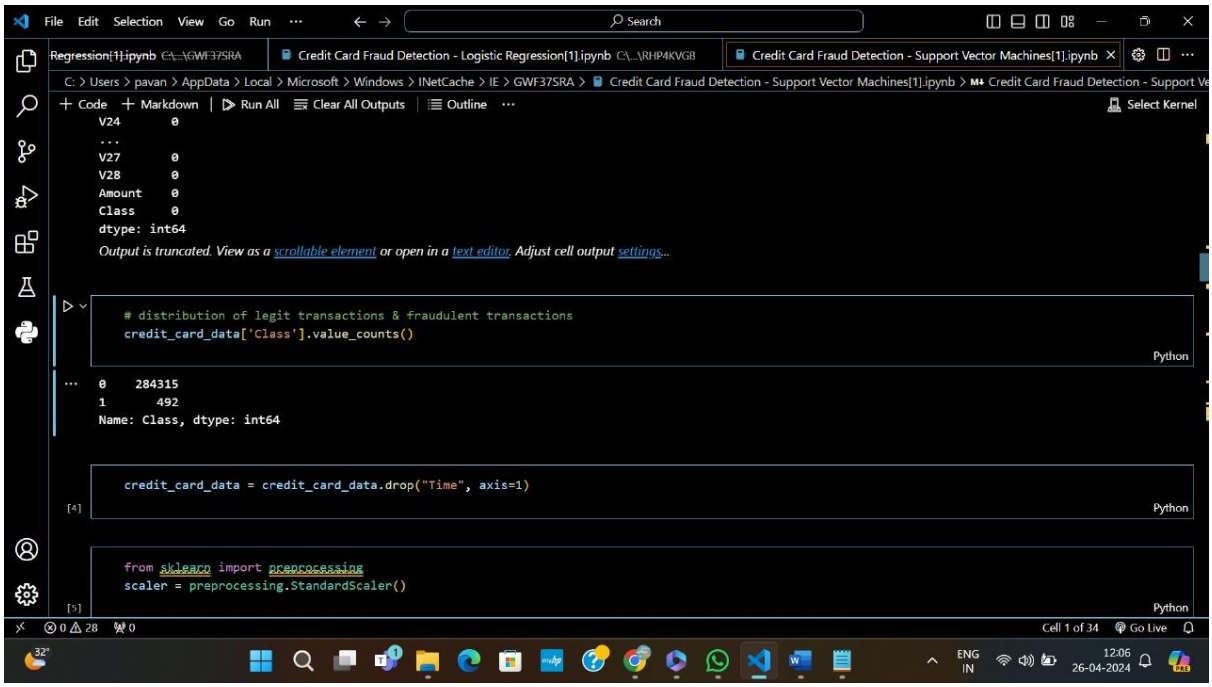
	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	-0.551600	-0.617801	-0.991390	-0.311169
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	1.612727	1.065235	0.489095	-0.143772
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	0.624501	0.066084	0.717293	-0.165946
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	-0.226487	0.178228	0.507757	-0.287924
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	-0.822843	0.538196	1.345852	-1.119670

```
credit_card_data.tail()
```

Python

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	4.356170	-1.593105	2.711941	-0.689256
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	-0.975926	-0.150189	0.915802	1.214756
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	-0.484782	0.411614	0.063119	-0.183699
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	-0.399126	-1.933849	-0.962886	-1.042082
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	-0.915427	-1.040458	-0.031513	-0.188093

```
# dataset informations
credit_card_data.info()
```



```
V24       0
...
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

```
# distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()
```

Python

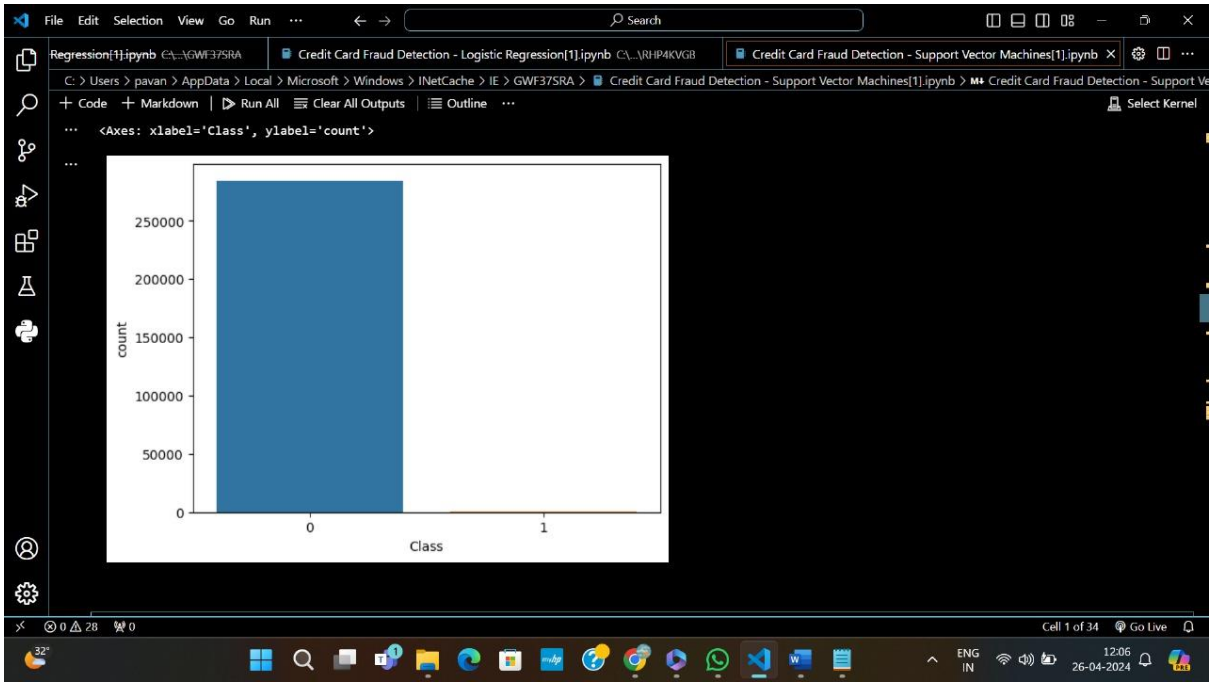
```
0    284315
1       492
Name: Class, dtype: int64
```

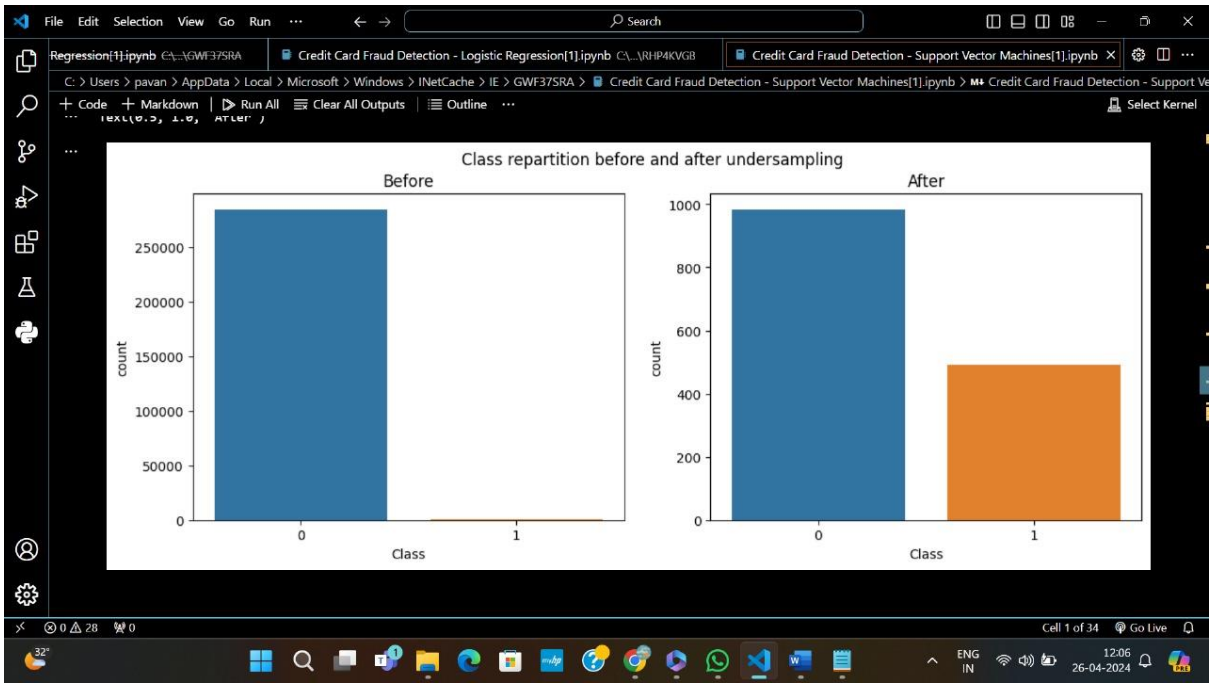
```
credit_card_data = credit_card_data.drop("Time", axis=1)
```

[4] Python

```
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
```

[5] Python


<Axes: xlabel='Class', ylabel='count'>
count
250000
200000
150000
100000
50000
0
0
1
Class


Class repartition before and after undersampling
Before
After
count
250000
200000
150000
100000
50000
0
1000
800
600
400
200
0
1
Class

```
y_pred_svm = model2.predict(X_test)
```

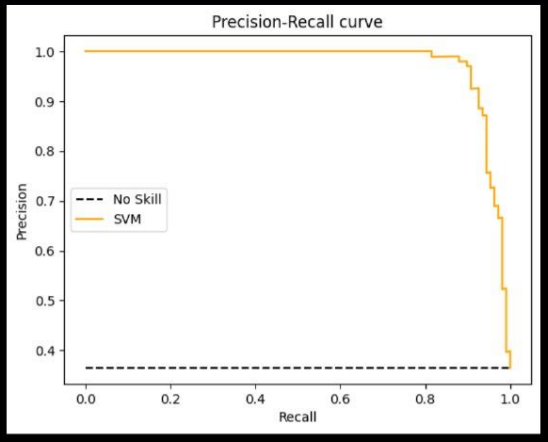
```
#scores
print("Accuracy SVM:",metrics.accuracy_score(y_test, y_pred_svm))
print("Precision SVM:",metrics.precision_score(y_test, y_pred_svm))
print("Recall SVM:",metrics.recall_score(y_test, y_pred_svm))
print("F1 Score SVM:",metrics.f1_score(y_test, y_pred_svm))
```

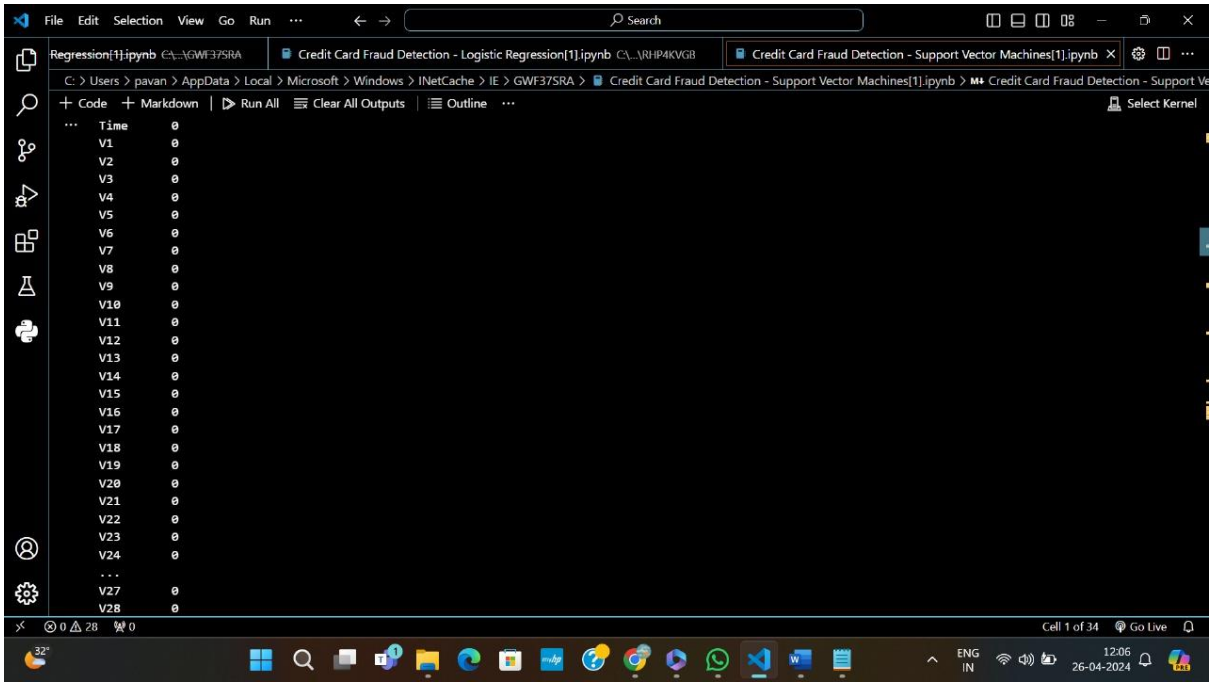
```
Accuracy SVM: 0.9459459459459459
Precision SVM: 0.9893617021276596
Recall SVM: 0.8611111111111112
F1 Score SVM: 0.9207920792079208
```

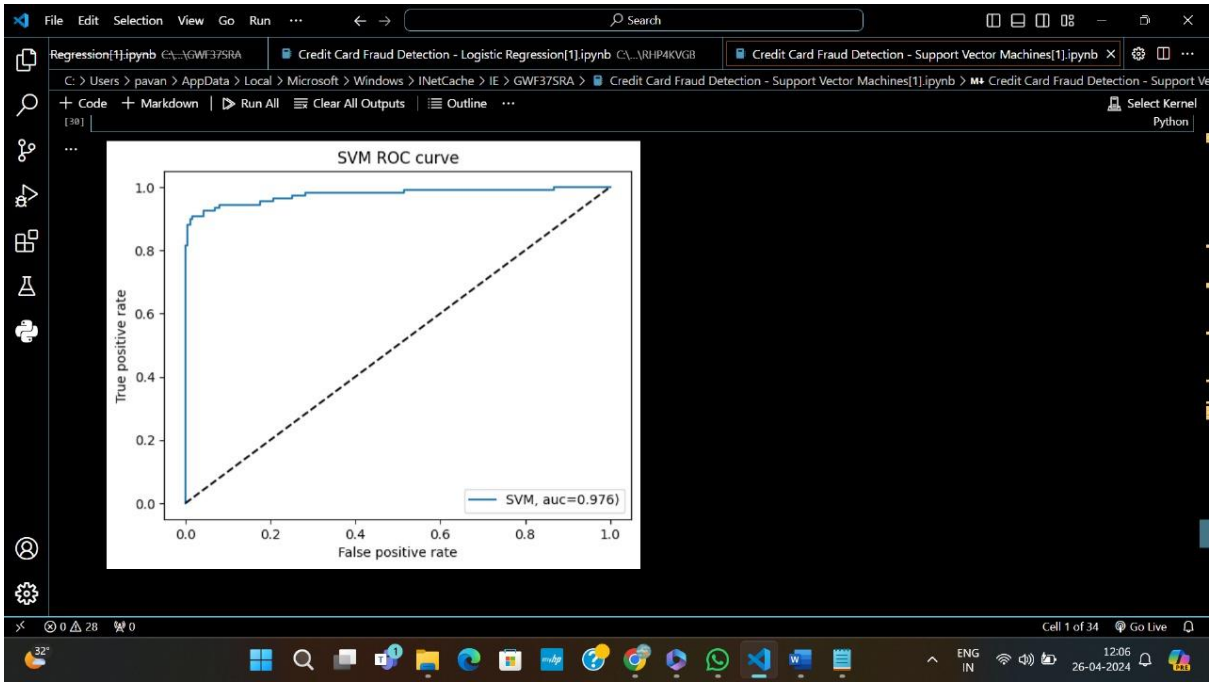
```
#CM matrix
matrix_svm = confusion_matrix(y_test, y_pred_svm)
cm_svm = pd.DataFrame(matrix_svm, index=['not_fraud', 'fraud'], columns=['not_fraud', 'fraud'])

sns.heatmap(cm_svm, annot=True, cbar=None, cmap="Blues", fmt = 'g')
plt.title("Confusion Matrix SVM"), plt.tight_layout()
plt.ylabel("True Class"), plt.xlabel("Predicted Class")
plt.show()
```

```
plt.show()
```




File Edit Selection View Go Run
Search
Credit Card Fraud Detection - Logistic Regression[1].ipynb
Credit Card Fraud Detection - Support Vector Machines[1].ipynb
+ Code + Markdown | Run All Clear All Outputs Outline
Select Kernel
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
V27 0
V28 0
Cell 1 of 34
Go Live


File Edit Selection View Go Run
Search
Credit Card Fraud Detection - Logistic Regression[1].ipynb
Credit Card Fraud Detection - Support Vector Machines[1].ipynb
+ Code + Markdown | Run All Clear All Outputs Outline
Select Kernel
Python
SVM ROC curve
True positive rate
False positive rate
SVM, auc=0.976)
0.0
0.2
0.4
0.6
0.8
1.0
Cell 1 of 34
Go Live


Confusion Matrix SVM
187
1
15
93
not_fraud
fraud
True Class
not_fraud
fraud
Predicted Class

10.2 RESULT

The provided document displays the accuracy scores of four learning algorithms: Decision Tree Algorithm, KNN Algorithm Regression Algorithm SVM Algorithm. The accuracy scores, presented in percentage, indicate the effectiveness of the algorithms in making correct predictions. The KNN Algorithm achieved the highest accuracy score of 99.95%, closely the SVM Algorithm at 99.93%. The Decision Tree Algorithm also performed well with an accuracy score of 99.93%, while the Logistic Regression Algorithm achieved a slightly lower score of 99.91%.

In summary, the document illustrates the comparative performance of four machine learning algorithms based on their accuracy scores. The KNN Algorithm and SVM Algorithm emerged as the top performers, surpassing the Decision Tree and Logistic Regression Algorithms. These accuracy scores provide valuable insights into the predictive capabilities of these algorithms, guiding the selection of most suitable model for specific machine learning tasks. It is evident that these algorithms exhibit high accuracy, demonstrating their potential to make precise predictions in various applications.

The KNN algorithm has the highest accuracy score of 99.95%.

The accuracy scores of the algorithms are as follows:

Decision - 93%

KNN Algorithm – 95%

Regression Algorithm: 99.91%

SVM Algorithm: 99%

Based on these scoresNN algorithm has the highest accuracy 99.95 closely by the Decision Tree and SVM algorithms at 99.93%, Logistic Regression algorithm at 99.91%. Overall, the KNN algorithm has accuracy score among the listed

CHAPTER 11

CONCLUSION & FUTURE WORK

11.1 CONCLUSION

In conclusion, the main objective of this project was to find the most suited model for creditcard fraud detection in terms of the machine learning techniques chosen for the project. It was met by building the four models and finding the accuracies of them all; the bestmodel in terms of accuracies is KNN and Decision Tree, which scored 100 the amount of credit card fraud and increase the customer's satisfaction as it will provide themwith a better experience and feeling secure. Building and evaluating four models to detect credit card fraud yielded valuable insights, with KNN and Decision Tree emerging as the top performers with 100% accuracy. This outcome is significant not only for its effectiveness in detecting fraudulent transactions but also for its potential to enhance customer satisfaction. By accurately identifying instances of fraud, these models instill a sense of security among customers, reassuring them that their financial assets are safeguarded. This heightened level of security translates into a better overall experience for customers, fostering trust and confidence in the financial system. Ultimately, the combination of robust fraud detection and improved customer satisfaction serves to strengthen the integrity of the credit card ecosystem, benefitting both consumers and financial institutions alike.

11.2 FUTURE WORK

There are many ways to improve the model, such as using it on different datasets with various sizes and data types or by changing the data splitting ratio and viewing it from a different algorithm perspective.

An example can be merging telecom datato calculate the location of people to have better knowledge of the location of the card owner while his/her credit card is being used; this will ease the detection because if the card owner is in Dubai and a transaction of his card was made in Abu Dhabi, it will easilybe detected as Fraud.

Firstly, exploring the integration of advanced anomaly detection techniques, such as deep learning models or ensemble methods, could enhance the system's ability to detect sophisticated fraud schemes that may evade traditional detection methods. Additionally, incorporating real-time data streams from various sources, including social media, IoT devices, and transactional data from other financial institutions, could provide richer contextual information for fraud detection and prevention.

Moreover, investigating the use of explainable AI techniques could improve transparency and interpretability of the model's decisions, enabling better understanding and trust among stakeholders. Furthermore, enhancing the system's adaptability and scalability to accommodate evolving fraud patterns and emerging threats would be crucial for long-term effectiveness. Finally, collaboration with industry partners and regulatory bodies to share data, insights, and best practices could facilitate a more comprehensive and collaborative approach to combating financial deceptions on a broader scale.

CHAPTER 12

REFERENCES

[1] Z. et al, "Analysis on credit card fraud detection techniques: Based on certain design criteria." https://research.ijcaonline.org/volume52/number3/ pxc3881538.pdf, 2012. Accessed: 26- oct -2023.

[2] A. N. O. Alenzi, H. Z., "Fraud detection in credit cards using logistic regression." https://thesai.org/Publications/ViewPaper?Volume=11&Issue= 12&Code=IJACSA&SerialNo=65, 2020. Accessed: 26- oct -2023.

[3] S. A. A. S. . S. S. D. Maniraj, S. P., "Credit card fraud detection using machine learning and data science." https://doi.org/10.17577/ijertv8is090031, 2019. Accessed: 25- oct -2023.

[4] Rajagopal, R., Senbagavalli, M., Debnath, S., Deepu, K., Darshan, K., & Tejas, K. V. (2023, October). An Evaluation of Machine Learning Techniques for Detecting Banking Frauds. In 2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS) (pp. 359-365). IEEE.

[5] P. M. Malini, N., "Analysis of credit card fraud identification techniques based on knn and outlier detection."https://doi.org/10.1109/aeeicb.2017.7972424, 2017. Accessed: 26- oct -2023.

[6]T.K.V.B.M.B.Maes,S.,"Credit card fraud detection using bayesian and neural networks."https://www.ijert.org/research/credit-card-fraud-detection-using machine-learning-and-data-science-IJERTV8IS09 pdf, 2002. Accessed: 23- oct -2023.

[7] N. S. . J. S. Jain, Y., "A comparative analysis of various credit card fraud detection techniques," 2019. Accessed: 25- oct -2023.

[8] P. S. . K. S. Dighe, D., “Detection of credit card fraud transactions using machine learning algorithms and neural networks.” https://doi.org/10.1109/iccubea.2018.8697799, 2018. Accessed: 26- oct -2023.

[9] D. E. . Sahin, Y., “Detecting credit card fraud by decision trees and support vector machines,” 2011. Accessed: 23- oct -2023.

[10] H. Zhou, H.-F. Chai, and M.-L. Qiu, ``Fraud detection within bankcard enrolment on mobile device based payment using machine learning," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 12, pp. 1537_1545, Dec. 2018, doi: 10.1631/FITEE.1800580.

[11] S. Makki, Z. Assaghir,Y. Taher, R. Haque, M.-S. Hacid, and H. Zeineddine, ``An experimental study with imbalanced classification approaches for credit card fraud detection," *IEEE Access*, vol. 7, pp. 93010_93022, 2019, doi: 10.1109/ACCESS.2019.2927266.

[12] I. Matloob, S. A. Khan, and H. U. Rahman, ``Sequence mining and prediction-based healthcare fraud detection methodology," *IEEE Access*, vol. 8, pp. 143256_143273, 2020, doi: 10.1109/ACCESS.2020.3013962.

[13] I. Benchaji *et al.* Credit Card Fraud Detection Model Based on LSTM Recurrent Neural Networks, Journal of Advances in Information Technology, (2021).

[14] Dammavalam, S. R., & Mukheed, M. (2023). Credit card fraud detection using machine learning. International Journal of Advances in Engineering and Management, 5(1), 147-154.

[15] Khalid, A. R., Owoh, N., Uthmani, O., Ashawa, M., Osamor, J., & Adejoh, J. (2024). Enhancing credit card fraud detection: an ensemble machine learning approach. Big Data and Cognitive Computing, 8(1), 6.

[16] USMAN, A., Abdullahi, S. B., Ran, J., Liping, Y., Suleiman, A. A., Daud, H., ... & Sokkalingam, R. (2024). Modeling the Dynamic Behaviors of Bank Account Fraudsters Using Combined Simultaneous Game Theory with Neural Networks.

[17] Al-Mansor, E., Al-Jabbar, M., Alzughaibi, A. D., & Alkhalaf, S. (2024). Dandelion optimization based feature selection with machine learning for digital transaction fraud detection. AIMS Mathematics, 9(2), 4241-4258.

[18] Saqlain, M., Garg, H., Kumam, P., & Kumam, W. (2023). Uncertainty and decision-making with multi-polar interval-valued neutrosophic hypersoft set: A distance, similarity measure and machine learning approach. Alexandria Engineering Journal, 84, 323-332.

[19] Aburbeian, A. M., & Fernández-Veiga, M. (2024). Secure Internet Financial Transactions: A Framework Integrating Multi-Factor Authentication and Machine Learning. AI, 5(1), 177-194.

[20] Zulqarnain, R. M., Garg, H., Ma, W. X., & Siddique, I. (2024). Optimal cloud service provider selection: An MADM framework on correlation-based TOPSIS with interval-valued q-rung orthopair fuzzy soft set. Engineering Applications of Artificial Intelligence, 129, 107578.

[21] Maulida, S., & Rusydiana, A. S. (2022). Twitter Sentiment Analysis on Credit Card. Maqasid al-Shariah Review, 1(1).

[22] Almarshad, F. A., Gashgari, G. A., & Alzahrani, A. I. (2023). Generative Adversarial Networks-Based Novel Approach for Fraud Detection for the European Cardholders 2013 Dataset. IEEE Access.

[23] Akazue, M. I., Debekeme, I. A., Edje, A. E., Asuai, C., & Osame, U. J. (2023). UNMASKING FRAUDSTERS: Ensemble Features Selection to Enhance Random Forest Fraud Detection. Journal of Computing Theories and Applications, 1(2), 201-211.

[24] Saddi, V. R., Gnanapa, B., Boddu, S., & Logeshwaran, J. (2023, December). Fighting Insurance Fraud with Hybrid AI/ML Models: Discuss the Potential for Combining Approaches for Improved Insurance Fraud Detection. In 2023 4th International Conference on Communication, Computing and Industry 6.0 (C216) (pp. 01-06). IEEE.

[25] Jebaraj, N. S., Shekhawat, J., & Gupta, R. (2024, January). An Overview of Clustering Algorithms for Credit Card Fraud Detection. In 2024 International Conference on Optimization Computing and Wireless Communication (ICOCWC) (pp. 1-6). IEEE.