**I.K Gujral Punjab Technical University**

**Amritsar Campus**



**Major Project-1:**

**ML-Preprocessor**

Department of Computer Science and Engineering

**Submitted by:**

Aditya Sharma (2023708)

**Abstract**

This Machine Learning (ML) Preprocessing project is designed to improve the foundation of datasets through a systematic approach. Recognizing the important role data quality plays in shaping the efficiency of ML models, this project amalgamates standalone .py files and class-based structures to create a comprehensive preprocessing solution. With a primary focus on enhancing flexibility, modularity, and user-friendly interactions, the project seamlessly integrates into various ML workflows.

The project commences with a thorough data description, unraveling the intricacies of the dataset and providing users with a comprehensive understanding of its structural nuances. Subsequently, the system undertakes the critical task of handling null values, offering users the flexibility to choose between mean, mode, and median strategies based on their dataset's unique characteristics.

Categorical data encoding, a fundamental preprocessing step, is executed through hot encoding, ensuring compatibility with a broad spectrum of ML models. The project's core lies in the feature scaling module, where normalization and standardization techniques are applied to numerical features, optimizing their range and ensuring convergence for ML algorithms.

This project sets the stage for future developments, encouraging exploration into emerging preprocessing techniques, dynamic outlier handling, and the integration of advanced visualization tools for enhanced user interactions. Its continual evolution ensures alignment with the dynamic landscape of ML and data preprocessing, providing a reliable foundation for constructing robust machine learning models

**Acknowledgement**

We are deeply grateful to my Mentor **Dr. Vipul Sharma** who was all the time helpful and supportive to me for the completion of this project. Being truly without his guidance, support, suggestions, and encouragement We would not have completed this study.

A semester project is a golden opportunity for learning and self-development. I consider myself very lucky and honored to have so many wonderful people lead me through in completion of this project.

I would like to express my special thanks to **Dr. Amit Sarin** Sir for being a constant source of Inspiration.

**Table of Content**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Content** | **Page Number** |
| **1** | Introduction | 5 |
| **2** | Scope and Objective | 6 |
| **3** | Requirement Analysis and System Specification | 8 |
| **4** | Implementation | 9 |
| **5** | Results | 15 |
| **6** | Conclusion | 18 |
| **7** | Future Scope | 19 |

**Introduction**

In the rapidly advancing realm of machine learning, the efficacy of models hinges significantly on the quality of input data. The preprocessing phase stands as a critical precursor, addressing data imperfections and refining datasets to ensure optimal model performance. This project delves into the intricate process of ML data preprocessing, offering a comprehensive solution that combines the efficiency of standalone .py files with the organizational structure of classes.

Recognizing the diverse challenges posed by real-world datasets, this project aims to provide a versatile toolkit for practitioners and researchers alike. The introduction of this project signifies a commitment to streamlining and enhancing the initial stages of machine learning workflows. By seamlessly integrating into existing pipelines, the project endeavors to simplify the often intricate process of preparing data for model training.

With a focus on modularity, the project allows users to tailor their preprocessing steps based on dataset-specific requirements. Whether it's descriptive data analysis, handling null values, encoding categorical features, or scaling numerical attributes, the system is designed to cater to a spectrum of preprocessing needs. The introduction of this project marks a stride towards fostering accessibility, efficiency, and adaptability in the intricate landscape of machine learning data preprocessing.

**Scope:**

The ML preprocessing project extends its utility across various domains and applications, providing a versatile toolkit for data scientists, researchers, and practitioners involved in machine learning endeavors. The scope of the project encompasses:

**Diverse Dataset Compatibility:** The project is designed to handle a wide array of datasets, accommodating different data types, structures, and sizes. Its modular nature allows users to adapt preprocessing steps according to the unique characteristics of their datasets.

**Flexibility in Preprocessing Techniques:** The system's modular design provides users with the flexibility to choose and execute specific preprocessing techniques based on their requirements. Whether it's descriptive analysis, null value handling, categorical encoding, or feature scaling, the project caters to diverse needs.

**Educational Applications**: The project serves as a valuable resource for educational purposes, offering a hands-on approach to understanding fundamental concepts of data preprocessing in machine learning

**Scalability and Performance Optimization:** As datasets continue to grow in size and complexity, the project is designed to scale effectively. The incorporation of efficient algorithms ensures that preprocessing tasks are executed with minimal computational overhead, optimizing performance.

The broad scope of this ML preprocessing project positions it as a valuable asset for the machine learning community, offering a robust, adaptable, and extensible solution to address the ever-evolving landscape of data preprocessing challenges.

**Objectives**

**Data Understanding and Description:**

Develop a module for descriptive data analysis to provide users with a comprehensive understanding of the dataset's structure, statistics, and key features.

**Null Value Handling:**

Implement strategies to handle null values, including options for users to choose between mean, mode, and median imputation based on the distribution of data.

**Feature Scaling:**

Develop modules for normalization and standardization to scale numerical features, optimizing their ranges for improved convergence and performance of machine learning algorithms.

**Modular Design with Classes:**

Implement a modular structure using Python classes, promoting code reusability, maintainability, and providing users with the flexibility to selectively execute specific preprocessing tasks.

**User-Friendly Interface:**

Design an intuitive user interface that facilitates seamless interaction with the preprocessing functionalities, allowing users to easily navigate and execute the necessary steps.

**Downloadable Modified Dataset:**

Integrate a feature to allow users to download the modified dataset after preprocessing, ensuring straightforward integration with machine learning models

**Efficiency and Performance Optimization:**

Develop efficient algorithms for preprocessing tasks to optimize performance, especially for handling large-scale datasets.

**Requirement Analysis and System Specification**

**Python:**

Use Python as the primary programming language due to its extensive libraries for data manipulation and machine learning.

**Integrated Development Environment (IDE):**

Choose a suitable IDE for Python development, such as Jupyter Notebooks, PyCharm, or VSCode, to streamline coding and debugging.

**Version Control:**

Implement version control using Git to track changes and collaborate with others.

**Libraries and Frameworks:**

Utilize popular Python libraries and frameworks for data manipulation and machine learning, including:

* NumPy and Pandas for data manipulation.
* Scikit-learn for machine learning tasks.

**Class-based Structure:**

Design the project using a class-based structure to encapsulate functionalities, enhance modularity, and promote code reusability.

**Implementation:**

1. **Main.py**

from data\_description import DataDescription

from data\_input import DataInput

from imputation import Imputation

from download import Download

from categorical import Categorical

from feature\_scaling import FeatureScaling

class Preprocessor:

bold\_start = "\033[1m"

bold\_end = "\033[0;0m"

# The Task associated with this class. This is also the main class of the project.

tasks = [

'1. Data Description',

'2. Handling NULL Values',

'3. Encoding Categorical Data',

'4. Feature Scaling of the Dataset',

'5. Download the modified dataset'

]

data = 0

def \_\_init\_\_(self):

self.data = DataInput().inputFunction()

print("\n\n" + self.bold\_start + "WELCOME TO THE MACHINE LEARNING PREPROCESSOR CLI!!!\N{grinning face}" + self.bold\_end + "\n\n")

*# function to remove the target column of the DataFrame.*

def removeTargetColumn(self):

print("Columns\U0001F447\n")

for column in self.data.columns.values:

print(column, end = " ")

while(1):

column = input("\nWhich is the target variable:(Press -1 to exit) ").lower()

if column == "-1":

exit()

choice = input("Are you sure?(y/n) ")

if choice=="y" or choice=="Y":

try:

self.data.drop([column], axis = 1, inplace = True)

except KeyError:

print("No column present with this name. Try again......\U0001F974")

continue

print("Done.......\U0001F601")

break

else:

print("Try again with the correct column name...\U0001F974")

return

def printData(self):

print(self.data)

*# main function of the Preprocessor class.*

def preprocessorMain(self):

self.removeTargetColumn()

while(1):

print("\nTasks (Preprocessing)\U0001F447\n")

for task in self.tasks:

print(task)

while(1):

try:

choice = int(input("\nWhat do you want to do? (Press -1 to exit): "))

except ValueError:

print("Integer Value required. Try again.....\U0001F974")

continue

break

if choice == -1:

exit()

*# moves the control into the DataDescription class.*

elif choice==1:

DataDescription(self.data).describe()

*# moves the control into the Imputation class.*

elif choice==2:

self.data = Imputation(self.data).imputer()

*# moves the control into the Categorical class.*

elif choice==3:

self.data = Categorical(self.data).categoricalMain()

*# moves the control into the FeatureScaling class.*

elif choice==4:

self.data = FeatureScaling(self.data).scaling()

*# moves the control into the Download class.*

elif choice==5:

Download(self.data).download()

else:

print("\nWrong Integer value!! Try again..\U0001F974")

*# obj is the object of our Preprocessor class(main class).*

obj = Preprocessor()

# the object 'obj' calls the main function of our Preprocessor class.

obj.preprocessorMain()

1. **Download.py:**

import pandas as pd

class Download:

bold\_start = "\033[1m"

bold\_end = "\033[0;0m"

def \_\_init\_\_(self, data):

self.data = data

*# download the modified DataFrame as .csv file*

def download(self):

toBeDownload = {}

for column in self.data.columns.values:

toBeDownload[column] = self.data[column]

newFileName = input("\nEnter the " + self.bold\_start +"FILENAME" + self.bold\_end +" you want? (Press -1 to go back): ")

if newFileName=="-1":

return

newFileName = newFileName + ".csv"

*# index=False as this will not add an extra column of index.*

pd.DataFrame(self.data).to\_csv(newFileName, index = False)

print("Hurray!! It is done....\U0001F601")

if input("Do you want to exit now? (y/n) ").lower() == 'y':

print("Exiting...\U0001F44B")

exit()

else:

return

1. **Data\_Description.py**

import pandas as pd

class DataDescription:

# The Task associated with this class.

tasks = [

'\n1. Describe a specific Column',

'2. Show Properties of Each Column',

'3. Show the Dataset'

]

def \_\_init\_\_(self, data):

self.data = data

*# The function that prints the database on the command line.*

def showDataset(self):

while(1):

try:

rows = int(input(("\nHow many rows(>0) to print? (Press -1 to go back) ")))

if rows == -1:

break

if rows <= 0:

print("Number of rows given must be +ve...\U0001F974")

continue

print(self.data.head(rows))

except ValueError:

print("Numeric value is required. Try again....\U0001F974")

continue

break

return

*# function to print all the columns*

def showColumns(self):

for column in self.data.columns.values:

print(column, end=" ")

*# function to describe the dataset or any specific column.*

def describe(self):

while(1):

print("\nTasks (Data Description)\U0001F447")

for task in self.tasks:

print(task)

while(1):

try:

choice = int(input(("\n\nWhat you want to do? (Press -1 to go back) ")))

except ValueError:

print("Integer Value required. Try again.....\U0001F974")

continue

break

if choice==-1:

break

elif choice==1:

self.showColumns()

while(1):

describeColumn = input("\n\nWhich Column? ").lower()

try:

*# describe() function is used to tell all the info regarding any specific column.*

print(self.data[describeColumn].describe())

except KeyError:

print("No Column present with this name. Try again....\U0001F974")

continue

break

elif choice==2:

*# describe() function is used to tell all the info about the database.*

print(self.data.describe())

print("\n\n")

print(self.data.info())

elif choice==3:

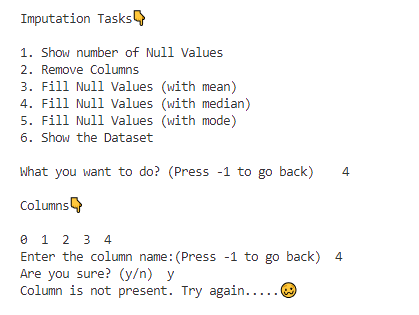
self.showDataset()

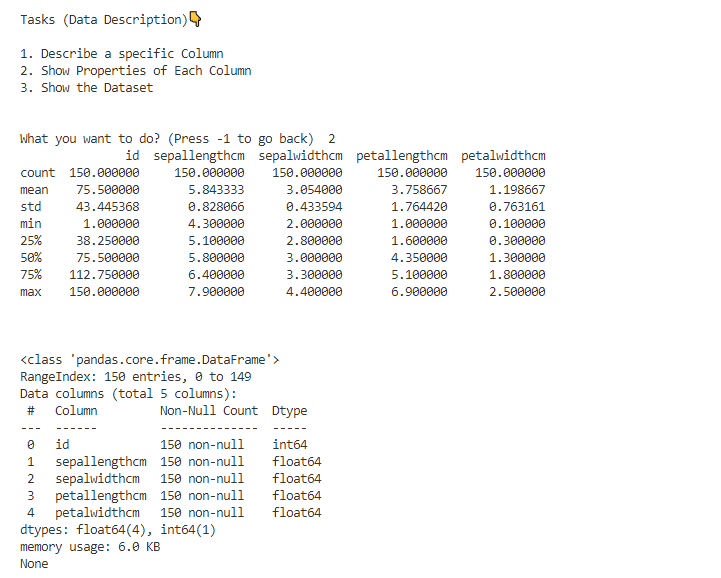
else:

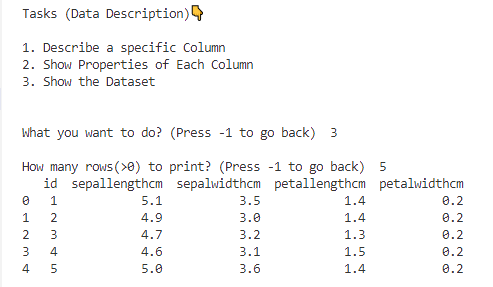
print("\nWrong Integer value!! Try again..\U0001F974")

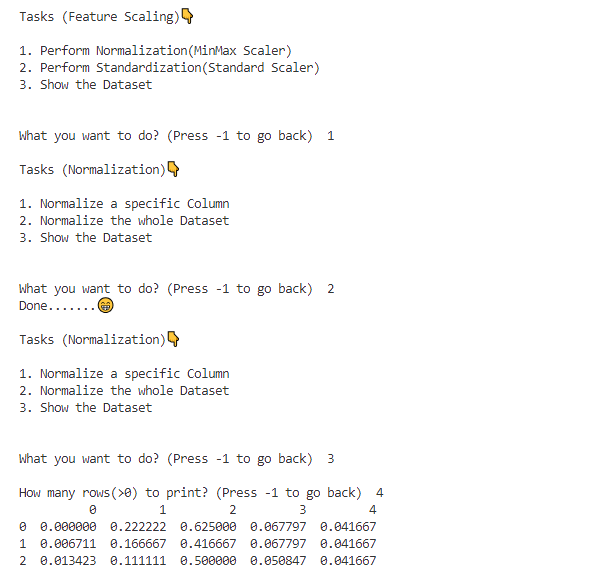
**Result:**











**Conclusion:**

In conclusion, the ML Preprocessing Project represents a significant stride towards fortifying the foundation of machine learning endeavors by systematically addressing the nuances of raw datasets. The project successfully achieves its primary objectives, offering a modular and class-based structure that enhances code reusability, maintainability, and flexibility.

Through comprehensive data understanding, null value handling, categorical encoding, and feature scaling, the project empowers users to seamlessly prepare datasets for machine learning model training. The user-friendly interface ensures accessibility, allowing practitioners and researchers to navigate through preprocessing tasks effortlessly.

The downloadable modified dataset serves as the tangible outcome of the project, now refined and optimized for integration with a variety of machine learning algorithms. The versatility demonstrated in accommodating diverse datasets positions the project as a valuable asset in the data preprocessing landscape.

Looking forward, the future scope of the project includes automation, dynamic outlier handling, enhanced visualization tools, and integration with data quality frameworks, promising continuous improvement and adaptation to emerging challenges. The project's commitment to open-source collaboration invites the community to contribute, fostering a collective effort towards refining and expanding the capabilities of the preprocessing tool.

**Future Scopes:**

**Automated Preprocessing Strategies:**

Explore and implement automated strategies for selecting optimal preprocessing techniques based on dataset characteristics. This could involve leveraging machine learning algorithms to intelligently determine the most suitable imputation methods and scaling techniques.

**Dynamic Outlier Handling:**

Integrate dynamic outlier detection and handling mechanisms to improve robustness in the face of outliers. This could involve the implementation of advanced statistical techniques or machine learning models for outlier detection.

**Enhanced Visualization Tools:**

Incorporate advanced data visualization tools to provide users with more intuitive insights into the data. This includes interactive visualizations and dashboards for a deeper understanding of dataset characteristics.

**Integration with Data Quality Frameworks:**

Explore integration with data quality frameworks to offer a comprehensive solution for ensuring data quality throughout the machine learning pipeline. This may involve incorporating checks for consistency, accuracy, and completeness

.

**Advanced Feature Engineering:**

Extend the project to include advanced feature engineering techniques, such as creating interaction terms, polynomial features, or time-based features, to enhance the predictive power of machine learning models

**Handling Time-Series Data:**

Enhance the project's capabilities to handle time-series data, incorporating preprocessing steps specific to temporal datasets. This includes handling irregular time intervals, lag features, and temporal aggregations.