SDSS GALAXY CLASSIFICATION USING MACHINE LEARNING

AN INDUSTRY ORIENTED MINI REPORT

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING(DS)

Submitted By

ABHINAY THEDLA	**21UK1A6792**
ANJALI BOLLAM	**21UK1A67C2**
AKANKSHA POTHINENI	**21UK1A6782**
SHIVAMANI MARLA	**22UK5A6710**

Under the guidance of

Mr. N. Rajesh

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) –

506005

DEPARTMENT OF

COMPUTER SCIENCE AND ENGINEERING(DS)

VAAGDEVI ENGINEERING COLLEGE(WARANGAL)

CERTIFICATE OF COMPLETION
INDUSTRY ORIENTED MINI PROJECT

This is to certify that the UG Project Phase-1 entitled “SDSS GALAXY USING MACHINE” is being submitted **by ABHINAY THEDLA (21UK1A6792), ANJALI BOLLAM (21UK1A67C2), AKANKSHA POTHINENI (21UK1A6782), SHIVAMANI MARLA (22UK5A6710)** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025.

Project Guide	**HOD**
Mr. N. Rajesh	**Dr. K. Sharmila Reddy**
(Assistant professor)	(Professor)

External

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. PRASAD RAO,** Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this UG Project Phase-1 in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILA**, Head of the Department of CSE, Vaagdevi Engineering College for providing us necessary infrastructure and thereby giving us freedom to carry out the UG Project Phase-1.

We express heartfelt thanks to Smart Bridge Educational Services Private Limited, all for their constant supervision as well as for providing necessary information regarding the UG Project Phase-1 and for their support in completing the UG Project phase-1.

We express heartfelt thanks to the guide N. RAJESH**,** Assistant professor, Department of CSE for his constant support and giving necessary guidance for completion of this UG Project Phase-1.

Finally, we express our sincere thanks and gratitude to my family members, friends for their encouragement and outpouring their knowledge and experience throughout the thesis.

ABHINAY THEDLA	**21UK1A6792**
ANJALI BOLLAM	**21UK1A67C2**
AKANKSHA POTHINENI	**21UK1A6782**
SHIVAMANI MARLA	**22UK5A6710**

ABSTRACT

In recent decade, large scale sky surveys such as Sloan Digital Sky Survey (SDSS) has resulted in generation of tremendous amount of data. The classification of this enormous amount of data by astronomers is time consuming. To simplify this process, in 2007 a volunteer, based citizen science project called "Galaxy Zoo" was introduced which has reduced the time for classification by a good extent.

However, in this modern era of deep learning, automating this classification task is highly beneficial as it reduces the time for classification. Since last few years, many algorithms have been proposed which happen to do a phenomenal job in classifying galaxies into multiple classes. But all these algorithms tend to classify galaxies into less than 6 classes. However, after considering the minute information which we know about galaxies, it is necessary to classify galaxies into more than 8 classes.

In this study, a neural network model is proposed so as to classify SDSS data into 10 classes from an extended Hubble Tuning Fork. Great care is given to disk edge and disk face galaxies, distinguishing between a variety of substructures and minute features which are associated to each class. The proposed model consists of convolution layers to extract features making this method fully automatic.

The achieved test accuracy is 84.73 % which happens to be promising after considering such minute details in classes. Along with convolution layers, the proposed model has 3 more layers responsible for classification which makes the algorithm consume less time.

- Key words: galaxies: general - Galaxy: structure - methods: miscellaneous - surveys - techniques: image

TABLE OF CONTENTS: -

1.INTRODUCTION

1.1. OVERVIEW

The Sloan Digital Sky Survey (SDSS) stands as a cornerstone of modern astronomical research, offering an unparalleled repository of data on millions of celestial objects, including an extensive cate log of galaxies. This wealth of information has provided the scientific community with unprecedented opportunities to explore and understand the universe. However, the massive scale and complexity of the SDSS dataset present significant challenges for traditional data analysis methods.

In this context, machine learning has emerged as a transformative approach, capable of handling large datasets and extracting meaningful patterns with high efficiency.
Machine learning algorithms, particularly those used in supervised learning

1.2. PURPOSE

The purpose of SDSS galaxy classification using machine learning can be articulated across several dimensions:

1. **Informing the Public:** By accurately classifying galaxies, we can enhance public understanding of the universe's structure, evolution, and diversity, fostering scientific literacy and curiosity.
2. **Health Protection:** Although indirectly related, advances in technology and understanding the cosmos can inspire innovation in other scientific fields, potentially leading to medical advancements or broader technological improvements.
3. **Government Regulation:** Understanding our universe can also provide insights into cosmic phenomena that might impact Earth, such as asteroids or solar events, prompting the development of policies to protect against such risks.

4. **Environmental Monitoring:** Certain aspects of astronomy, such as studying cosmic radiation or cosmic dust, can have implications for understanding environmental phenomena on Earth, contributing to environmental monitoring efforts.
5. **Economic Impact:** Advances in space science and technology often have spin-off benefits for various industries, contributing to economic growth through innovations in technology, materials science, and data analysis techniques.

These purposes collectively underscore the multidimensional benefits of SDSS galaxy classification using machine learning, extending beyond pure astronomical curiosity to impact various facets of society and scientific inquiry.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

One existing problem in SDSS galaxy classification using machine learning revolves around the challenge of classifying rare or poorly represented galaxy types accurately. Here are some key aspects of this problem:

1. **Class Imbalance:**
    - In datasets derived from surveys like SDSS, certain galaxy types may be significantly less common than others. Traditional machine learning models can struggle to correctly classify these minority classes due to inadequate representation in the training data.
    - Addressing this issue often requires specialized techniques such as data augmentation, resampling methods (like oversampling minority classes), or using algorithms that are inherently robust to class imbalance.
2. **Complex Feature Space:**
    - Galaxies are complex astronomical objects characterized by diverse features, including photometric data (brightness across different wavelengths), spectroscopic data (emission and absorption lines), and spatial distributions.
    - Machine learning models must effectively capture and interpret these multidimensional features to make accurate classifications. Feature selection and extraction methods are critical to handle this complexity effectively.

3.**Interpretability and Generalization:**

- Deep learning models, while powerful in learning intricate patterns, often lack interpretability in how they arrive at their classifications. Understanding why a model assigns a certain label to a galaxy is crucial for validating scientific findings and ensuring robustness across different datasets.
- Ensuring that models generalize well across different surveys and observational conditions is another challenge, as variations in data collection methods and instruments can introduce biases that affect classification performance.

4.**Integration with Astrophysical Knowledge:**

- Incorporating domain knowledge and astrophysical insights into machine learning models can enhance classification accuracy and interpretability. Ensuring that AI-driven classifications align with existing astrophysical taxonomies and theories is essential for producing scientifically meaningful results.

Addressing these challenges requires interdisciplinary collaboration between astronomers, data scientists, and machine learning experts. Innovative approaches that combine advanced machine learning techniques with domain-specific knowledge are key to advancing SDSS galaxy classification and pushing the boundaries of astronomical research.

2.2 PROPOSED SOLLUTION

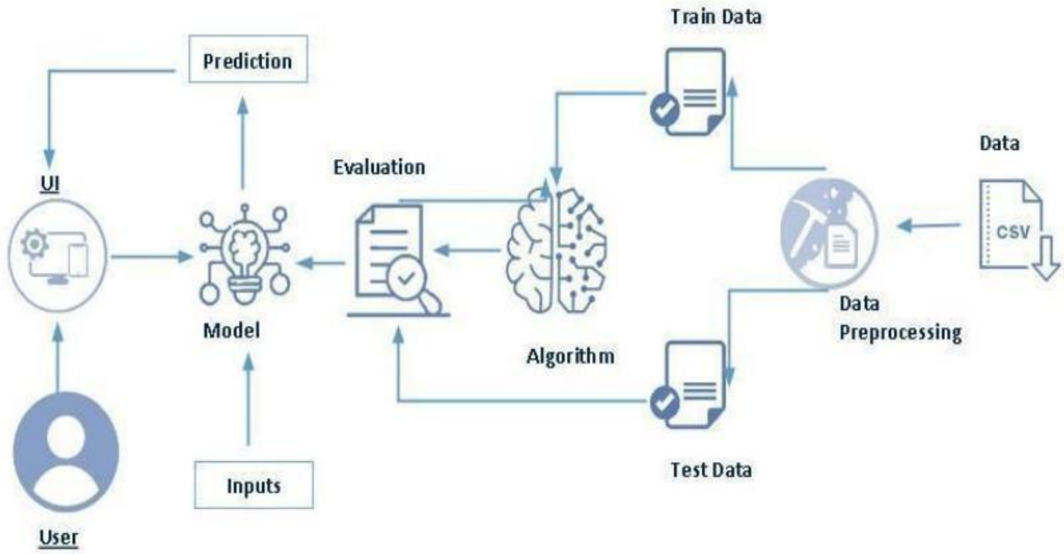
The Sloan Digital Sky Survey (SDSS) has amassed a vast amount of astronomical data, including detailed images and spectra of millions of celestial objects. This data is invaluable for understanding the universe, and galaxy classification is a key step in this process. Traditional methods of galaxy classification are time-consuming and often rely on human expertise. Machine learning offers a promising alternative, enabling the automatic and efficient classification of galaxies.

Objectives

- ➢ **Efficiency**: Develop a machine learning model that can quickly classify galaxies into different morphological categories (e.g., elliptical, spiral, irregular) with high accuracy.
- ➢ **Scalability**: Ensure the model can handle the large volume of data provided by SDSS and future astronomical surveys.
- ➢ **Interpretability**: Provide insights into the features and patterns that the model uses for classification to advance our understanding of galaxy morphology.
- ➢ **Accessibility**: Create a user-friendly interface for astronomers and the public to access and utilize the classification results.

3.THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. SOFTWARE DESIGNING

REQUIREMENTS	SPECIFICATIONS
Anaconda Navigator	You must have anaconda installed in your device prior to begin.
➢ GOOGLECOLLA, JUPYTER Notebook, Flask ➢ Frame work .	➢ One should have GOOGLE COLLAB, and JUPYTER notebook. ➢ One should install flask framework through Anaconda prompt for running their web application. ➢ We need to build the mode; using JUPYTER notebook with all the imported packages.
Web browser	For all Web browsers, the following must be enabled: ● Cookies ● Java script

Hardware requirements:

REQUIREMENTS	**SPECIFICATIONS**
Operating system	➢ Microsoft windows ➢ Unix ➢ Linux
Processing	Minimum: 4 CPU cores for one user. For each deployment, a sizing exercise is highly recommended.
RAM	Minimum 8 GB.
Operating system specifications	File descriptor limit set to 8192 on UNIX and Linux
Disk space	A minimum of 7 GB of free space is required to install the software.

4.EXPERIMENTAL INVESTIGATION

An experimental investigation of SDSS (Sloan Digital Sky Survey) galaxy classification using machine learning (ML) typically involves several key steps:

1. Data Collection and Preprocessing:

Dataset: Use the SDSS database to collect images and/or spectra of galaxies. SDSS provides extensive photometric and spectroscopic data.

Preprocessing: Clean the data by handling missing values, normalizing the data, and possibly augmenting images if using image data.

2. Feature Extraction:

Extract relevant features from the data. For images, this might involve using techniques like PCA (Principal Component Analysis) or using pre-trained convolutional neural networks (CNNs) to extract features. For spectra, it could involve extracting specific lines or features in the spectra.

3. Labelling

Use existing classifications from SDSS or other astronomical catalogs to label the data. This often involves labels like galaxy types (e.g., elliptical, spiral) or morphological features.

4. Model Selection:

Choose appropriate ML models for classification. Common choices include:

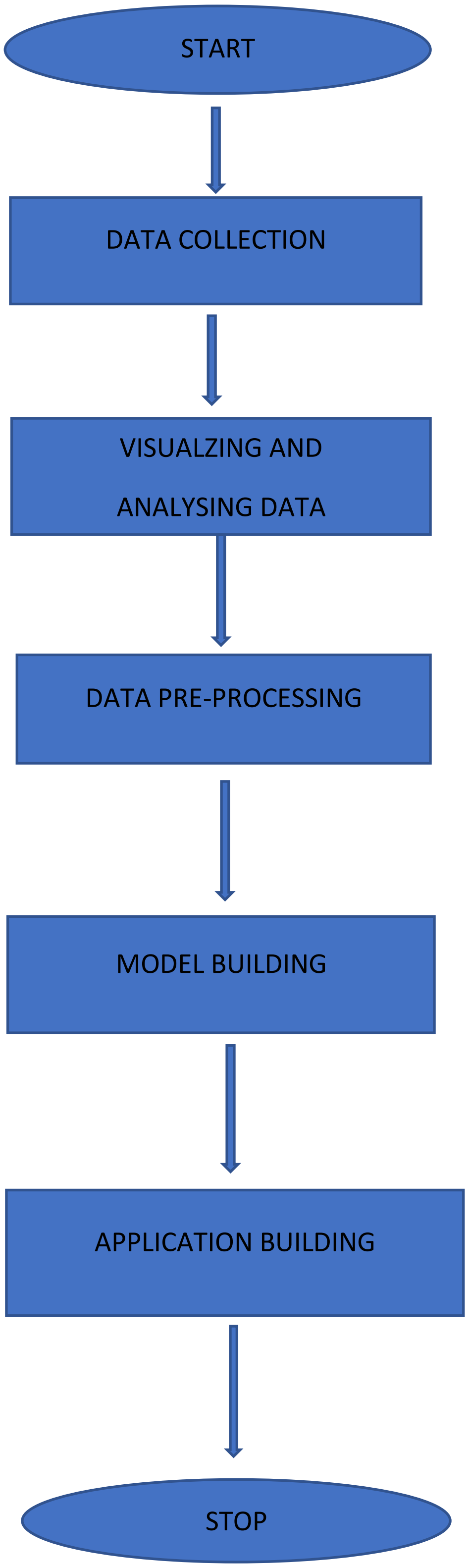
Supervised Learning Models: Decision Trees, Random Forests, Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and neural networks.

Deep Learning Models*: Convolutional Neural Networks (CNNs) are particularly effective for image classification tasks.

For the dataset we selected, it consists of more than the columns we want to predict it. So, we have chosen the feature drop it contains the columns that we are going to predict the AQI value.

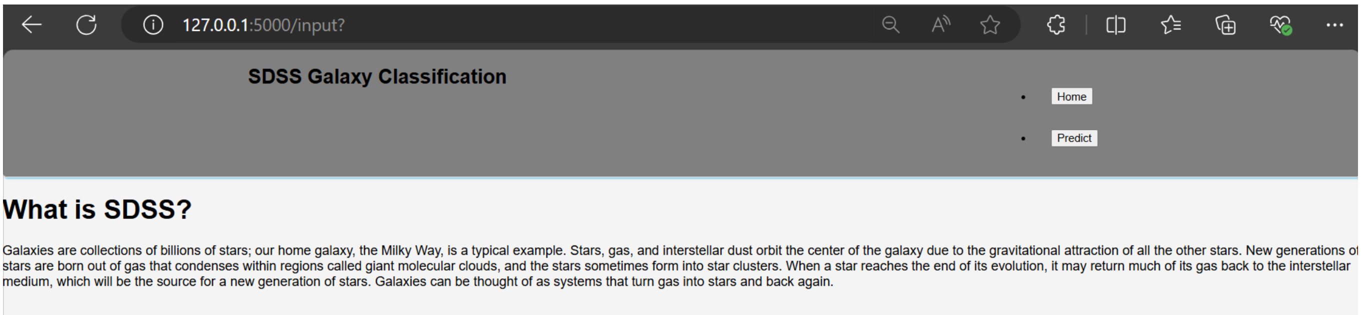
- ➢ Feature drop means it drops the columns that we don’t want in our dataset.
- ➢ Feature drop = ['PM10','NH3','Benzene','Toluene','Xylene','index']

5.FLOWCHAT

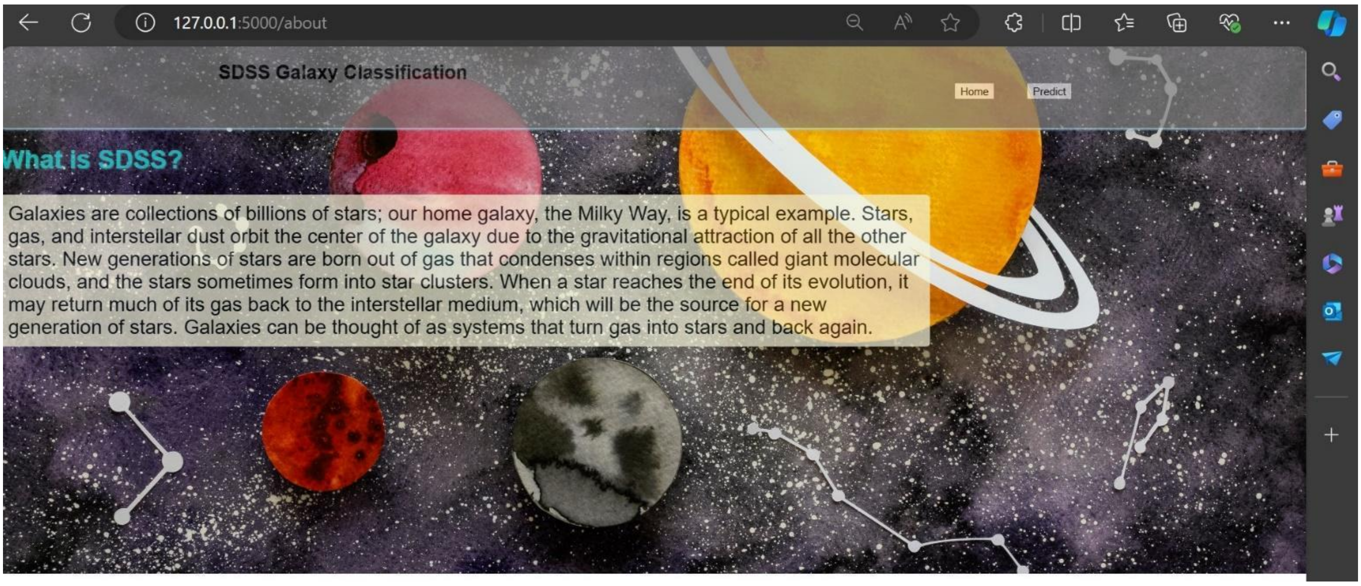

START
DATA COLLECTION
VISUALZING AND
ANALYSING DATA
DATA PRE-PROCESSING
MODEL BUILDING
APPLICATION BUILDING
STOP

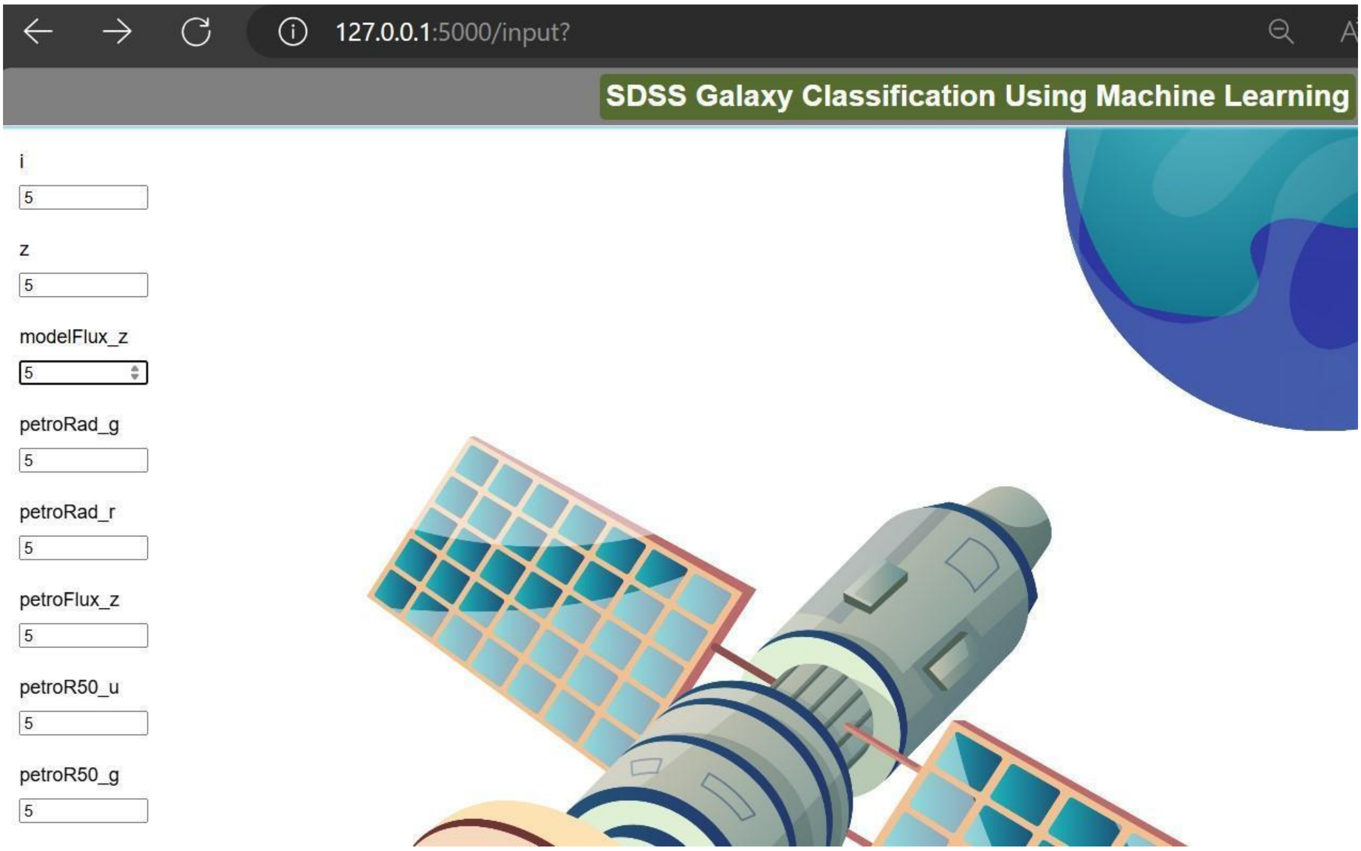
6.RESULT

HOME PAGE



PREDICTIONS


127.0.0.1:5000/about
SDSS Galaxy Classification
Home
Predict
What is SDSS?
Galaxies are collections of billions of stars; our home galaxy, the Milky Way, is a typical example. Stars, gas, and interstellar dust orbit the center of the galaxy due to the gravitational attraction of all the other stars. New generations of stars are born out of gas that condenses within regions called giant molecular clouds, and the stars sometimes form into star clusters. When a star reaches the end of its evolution, it may return much of its gas back to the interstellar medium, which will be the source for a new generation of stars. Galaxies can be thought of as systems that turn gas into stars and back again.


127.0.0.1:5000/input?
SDSS Galaxy Classification Using Machine Learning
i
5
z
5
modelFlux_z
5
petroRad_g
5
petroRad_r
5
petroFlux_z
5
petroR50_u
5
petroR50_g
5

7.ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

1. **Speed and Efficiency**:
    - **Automated Processing**: Machine learning algorithms can process vast amounts of data much faster than human classification, leading to quicker analysis and discoveries.
    - **Scalability**: The ability to handle large datasets, such as those from SDSS, makes machine learning suitable for extensive astronomical surveys.
2. **Accuracy and Consistency**:
    - **Reduced Human Error**: Machine learning models can maintain consistent accuracy without the variability introduced by human classifiers.
    - **Pattern Recognition**: Algorithms can identify subtle patterns and features in the data that may be overlooked by humans.
3. **Predictive Capabilities**:
    - **Generalization**: Once trained, models can generalize from known data to classify new, unseen data.
    - **Anomaly Detection**: Machine learning can be used to identify unusual or rare objects, leading to potential new discoveries.
4. **Resource Efficiency**:
    - **Cost-Effective**: Automating the classification process reduces the need for extensive human labor, potentially lowering costs.
    - **Continuous Improvement**: Machine learning models can be continuously improved with new data, enhancing their accuracy over time.
5. **DISADVANTAGES:**

1. **Data Quality and Bias**:
    - **Training Data Dependence**: The quality and accuracy of the model depend heavily on the quality and representativeness of the training data.

    - **Bias**: Models can inherit and amplify biases present in the training data, leading to skewed or inaccurate classifications.
2. **Complexity and Interpretability**:
    - **Black Box Nature**: Many machine learning models, especially deep learning, can be difficult to interpret, making it challenging to understand the reasoning behind classifications.
    - **Model Complexity**: Developing and tuning machine learning models can be complex and require significant expertise.
3. **Resource Intensive**:
    - **Computational Cost**: Training machine learning models, particularly deep learning models, can be computationally expensive and require substantial hardware resources.
    - **Data Requirements**: Large amounts of labeled data are often necessary to train effective models, which can be a limitation.
4. **Generalization Issues**:
    - **Overfitting**: Models can sometimes overfit the training data, performing well on known data but poorly on new, unseen data.
    - **Domain Specificity**: Models trained on specific datasets may not generalize well to different datasets or classification tasks without retraining

8.APPLICATIONS

1. **Public Health Protection**: Empowering individuals to make informed decisions regarding outdoor activities, reducing exposure to poor air quality, and minimizing health risks.
2. **Environmental Monitoring**: Assessing the impact of air pollution on the environment, ecosystems, and natural habitats, aiding in conservation efforts.
3. **Government Policy**: Assisting governments and regulatory bodies in setting air quality standards, formulating pollution control policies, and conducting effective urban planning
4. **Public Awareness**: Raising public awareness about the importance of air quality and its impact on health, influencing behavior and lifestyle choices.

9.CONCLUSION

In this study, we propose a convolutional neural network to classify galaxies in 10 classes. This is one of the initial, work wherein the galaxies are classified in 10 classes by considering such minute details. This detailed classification happens to be of need after considering the theoretical knowledge we have. This was initially done by professional astronomers but due to large amount of data we have, it was impossible for them to continue. Further, different citizen scientists were trained to do this task, but in modern era the data we have is huge in number compared to that of available volunteers. Hence, this algorithm happens to solve this problem. Also, the time taken by algorithm to classify large volume of dataset is less than 10 minutes which is another advantage of using the automated algorithms over manual classification. The proposed algorithm gives accuracy of 84.5 % which is good after considering such minute details in, classification**.**

10.FUTURE SCOPE

Future Scope of SDSS Galaxy Classification

The future of SDSS galaxy classification holds significant potential for advancing our understanding of the universe. Here are some key areas where this field is expected to grow and evolve:

1. **Improved Classification Algorithms**
   - **Machine Learning and AI:** Continued development and integration of advanced machine learning techniques, such as deep learning and neural networks, will enhance the accuracy and efficiency of galaxy classification.
   - **Automated Classification:** The use of automated systems will reduce human bias and error, allowing for the consistent classification of large datasets.
2. **Integration with Multi-Wavelength Data**
   - **Cross-Survey Analysis:** Combining SDSS data with information from other astronomical surveys (e.g., X-ray, radio, infrared) will provide a more comprehensive understanding of galaxy properties and evolution.
   - **Enhanced Feature Extraction:** Multi-wavelength data can help in identifying and characterizing different features of galaxies that are not visible in optical wavelengths alone.
3. **Real-Time Data Processing**
   - **Streamlined Pipelines:** Development of real-time data processing pipelines will enable the immediate classification and analysis of incoming data from new SDSS observations.

- **Big Data Technologies:** Utilizing big data technologies will help manage and process the enormous volumes of data generated by SDSS and other astronomical surveys.

4. **Expanded Data Sets**
   - **Next-Generation Surveys:** Future surveys, such as the Large Synoptic Survey Telescope (LSST), will provide even larger datasets, requiring more sophisticated classification techniques.
   - **Increased Data Depth:** Higher sensitivity and resolution in future observations will allow for the classification of fainter and more distant galaxies.

11.BIBILOGRAPHY

SDSS galaxy classification project involves listing the key sources you've used or plan to use. Here are some essential references to include:

Books

- **Astrophysics for Physicists** by Arnab Rai Choudhuri (2010).
- **Galaxy Formation and Evolution** by Houjun Mo, Frank van den Bosch, and Simon White (2010).

Research Papers

1. **York, D. G., et al. (2000).** The Sloan Digital Sky Survey: Technical Summary. *The Astronomical Journal, 120*(3), 1579-1587. doi:10.1086/301513.
2. **Abazajian, K. N., et al. (2009).** The Seventh Data Release of the Sloan Digital Sky Survey. *The Astrophysical Journal Supplement Series, 182*(2), 543-558. doi:10.1088/0067-0049/182/2/543.
3. **Blanton, M. R., & Roweis, S. (2007).** K-Corrections and Filter Transformations in the Ultraviolet, Optical, and Near-Infrared. *The Astronomical Journal, 133*(2), 734-754. doi:10.1086/510127.
4. **Stoughton, C., et al. (2002).** Sloan Digital Sky Survey: Early Data Release. *The Astronomical Journal, 123*(1), 485-548. doi:10.1086/324741.
5. **Eisenstein, D. J., et al. (2001).** Spectroscopic Target Selection for the Sloan Digital Sky Survey: The Luminous Red Galaxy Sample. *The Astronomical Journal, 122*(4), 2267-2280. doi:10.1086/323717.

Websites

- Sloan Digital Sky Survey (SDSS) official website
- Astrophysics Data System (ADS) by NASA
- arXiv.org e-Print archive

[1] Anderson H. R., R.W. Atkinson, J. L. Peacock, M. J. Sweeting and L. Marston. Ambient Particulate matter and health effect; Publication bias in studies of short-term association. Epidemiol 16; 2005:

12.APPENDIX

Model building :

1)Dataset
2)Google colab and Anaconda Application Building
   1. HTML file (Index file, second file, final file)
   1. CSS file
   2. Models in pickle format

SOURCE CODE:

INDEX.HTML

```
!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About</title>
    <link rel="stylesheet" type="text/css" href="{{url_for ('static',
filename='style.css')}} “>
</head>
<body>
    <div class="main">
        <div class="nav">
            <nav>
                <h2>SDSS Galaxy Classification</h2>
                <ul>
                    <li>
                        <form action="{{url_for('index')}} “>
                            <button type="submit">Home</button>
                        </form>
                    </li>
                    <li>
                        <form action="{{url_for('input')}} “>
                            <button type="submit">Predict</button>
                        </form>
                    </li>
                </ul>
            </nav>
        </div>
        <div>
            <h1>What is SDSS?</h1>
            <p>Galaxies are collections of billions of stars; our home galaxy, the
Milky Way, is a typical example. Stars, gas, and interstellar dust orbit the centre
of the galaxy due to the gravitational attraction of all the other stars. New
```

```
generations of stars are born out of gas that condenses within regions called giant
molecular clouds, and the stars sometimes form into star clusters. When a star
reaches the end of its evolution, it may return much of its gas back to the
interstellar medium, which will be the source for a new generation of stars. Galaxies
can be thought of as systems that turn gas into stars and back again.</p>
        </div>
    </div>
</body>
</html>
```

SECOND.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>About</title>
    <link rel="stylesheet" type="text/css" href="{{url_for ('static',
filename='style.css')}} “>
</head>
<body>
    <div class="main">
        <div class="nav">
            <nav>
                <h2>SDSS Galaxy Classification</h2>
                <ul>
                    <li>
                        <form action="{{url_for('index')}} “>
                            <button type="submit">Home</button>
                        </form>
                    </li>
                    <li>
                        <form action="{{url_for('input')}} “>
                            <button type="submit">Predict</button>
                        </form>
                    </li>
                </ul>
            </nav>
        </div>
        <div>
            <h1>What is SDSS?</h1>
            <p>Galaxies are collections of billions of stars; our home galaxy, the
Milky Way, is a typical example. Stars, gas, and interstellar dust orbit the center
of the galaxy due to the gravitational attraction of all the other stars. New
generations of stars are born out of gas that condenses within regions called giant
molecular clouds, and the stars sometimes form into star clusters. When a star
reaches the end of its evolution, it may return much of its gas back to the
```

```
interstellar medium, which will be the source for a new generation of stars. Galaxies
can be thought of as systems that turn gas into stars and back again. </p>
        </div>
    </div>
</body>
</html>
```

FINAL.HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Prediction Result</title>
    <link rel="stylesheet" type="text/css" href="{{url_for ('static',
filename='style.css')}} “>
    <style>
        body {
            background: url('static\\bg3.jpg');
            background-position: centre;
            background-size: cover;
            background-repeat: no-repeat;
        }
        h1{
            color: rgb(26, 161, 134);
            filter: drop-shadow(1px 1px 1px black);
        }
        h3{
            font-size: 30px;
            color: rgb(219, 11, 11);
            filter: drop-shadow (1px 1px 1px rgb (9, 1, 1));
        }
    </style>
</head>
<body >
    <div class="container"> <centre>
        <h1>Prediction Result</h1>
        <h3 class="output">{{ prediction }}</h3> </centre>
    </div>
</body>
</html>
```

MAIN.PY

```
from flask import Flask, request, render_template
import pickle
import pandas as pd
```

```
# Load the trained model
with open ('RF.pkl', 'rb') as file:
    model = pickle. load(file)

app = Flask(__name__)

@app. Route ("/")
def index ():
    return render_template("index.html")

@app. route ("/about", methods=["POST", "GET"])
def about ():
    return render_template("about.html")

@app.route("/input")
def input():
    return render_template("second.html")

@app. route ('/submit', methods=["POST"]) # Specify POST method
def submit ():
    # Reading input values from the form
    input_feature = [float(x) for x in request. form. Values ()]
    names = ['i', 'z', 'modelFlux_z', 'petroRad_g', 'petroRad_r', 'petroFlux_z',
'petroR50_u', 'petroR50_g', 'petroR50_i', 'petroR50_r']

    print ("Number of columns in names:", len(names))
    print ("Number of columns in input_feature:", len(input_feature))
    print ("Column names:", names)

    data = pd.DataFrame([input_feature], columns=names)

    # Make prediction
    prediction = model.predict(data)

    # Render the output template with the prediction result
    if prediction [0] == 0:
        print(prediction)
        return render_template ('final.html', prediction='starforming')
    else:
        return render_template ('final.html', prediction='starbursting')

if __name__ == "__main__":
    app.run(debug=True)
```

CODE SNIPPETS

MODEL BUILDING

```
!pip install -q kaggle
```

```
rm -rf /root/.kaggle
```

```
!mkdir ~/.kaggle
```

```
!cp kaggle.json ~/.kaggle
```

```
!chmod 600 /root/.kaggle/kaggle.json
```

```
!kaggle datasets download -d bryancimo/sdss-galaxy-classification-dr18
```

```
Dataset URL: https://www.kaggle.com/datasets/bryancimo/sdss-galaxy-classification-dr18
License(s): CC0-1.0
Downloading sdss-galaxy-classification-dr18.zip to /content
 27% 5.00M/18.4M [00:00<00:00, 21.3MB/s]
100% 18.4M/18.4M [00:00<00:00, 63.1MB/s]
```

```
!unzip /content/sdss-galaxy-classification-dr18.zip
```

```
Archive:  /content/sdss-galaxy-classification-dr18.zip
replace sdss_100k_galaxy_form_burst.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename:
```

IMPORTING THE LIBRARIES

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE
```

READ THE DATASET

```
df=pd.read_csv('/content/sdss_100k_galaxy_form_burst.csv',header=1)
```

```
df.head()
```

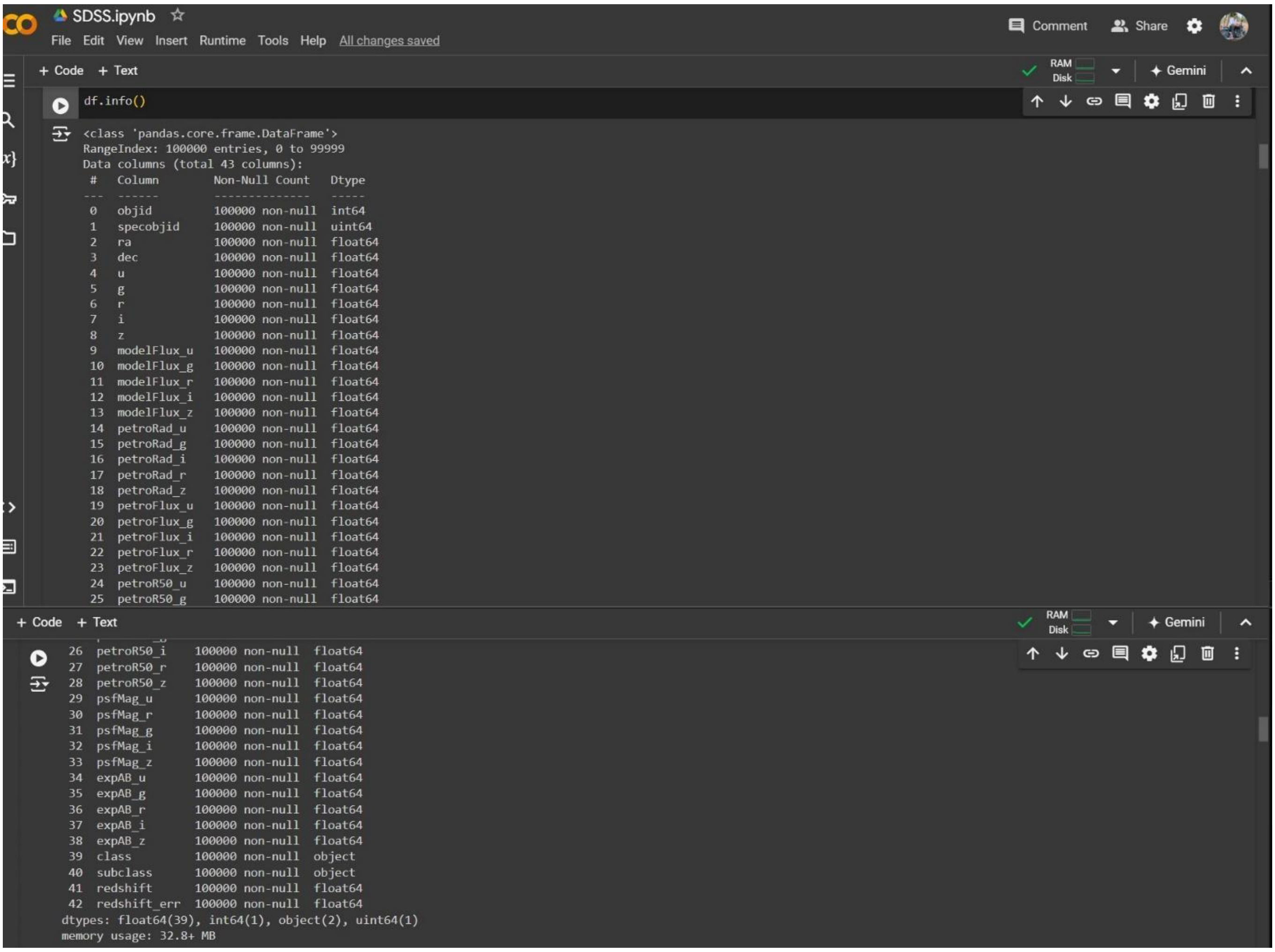
	objid	specobjid	ra	dec	u	g	r	i	z	modelFlux_u	...	psfMag_z	expAB_u	expAB_
0	1237646587710669400	8175185722644649984	82.038679	0.847177	21.73818	20.26633	19.32409	18.64037	18.23833	2.007378	...	19.43575	0.099951	0.31186
1	1237646588247540577	8175186822156277760	82.138894	1.063072	20.66761	19.32016	18.67888	18.24693	18.04122	5.403369	...	18.85012	0.366549	0.51687
2	1237646588247540758	8175187097034184704	82.028510	1.104003	23.63531	21.19671	19.92297	19.31443	18.68396	0.295693	...	19.42235	0.050000	0.41713
3	1237648702973083853	332152325571373056	198.544469	-1.097059	20.12374	18.41520	17.47202	17.05297	16.72423	8.920645	...	18.03204	0.310763	0.35682
4	1237648702973149350	332154249716721664	198.706864	-1.046217	-9999.00000	-9999.00000	18.37762	18.13383	17.78497	0.000000	...	19.02880	-9999.000000	-9999.00000

5 rows × 43 columns

HANDING MISSING VALUES

```
df.shape
```

```
(100000, 43)
```



```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 43 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   objid         100000 non-null  int64
 1   specobjid     100000 non-null  uint64
 2   ra            100000 non-null  float64
 3   dec           100000 non-null  float64
 4   u             100000 non-null  float64
 5   g             100000 non-null  float64
 6   r             100000 non-null  float64
 7   i             100000 non-null  float64
 8   z             100000 non-null  float64
 9   modelFlux_u   100000 non-null  float64
 10  modelFlux_g   100000 non-null  float64
 11  modelFlux_r   100000 non-null  float64
 12  modelFlux_i   100000 non-null  float64
 13  modelFlux_z   100000 non-null  float64
 14  petroRad_u    100000 non-null  float64
 15  petroRad_g    100000 non-null  float64
 16  petroRad_i    100000 non-null  float64
 17  petroRad_r    100000 non-null  float64
 18  petroRad_z    100000 non-null  float64
 19  petroFlux_u   100000 non-null  float64
 20  petroFlux_g   100000 non-null  float64
 21  petroFlux_i   100000 non-null  float64
 22  petroFlux_r   100000 non-null  float64
 23  petroFlux_z   100000 non-null  float64
 24  petroR50_u    100000 non-null  float64
 25  petroR50_g    100000 non-null  float64
 26  petroR50_i    100000 non-null  float64
 27  petroR50_r    100000 non-null  float64
 28  petroR50_z    100000 non-null  float64
 29  psfMag_u      100000 non-null  float64
 30  psfMag_r      100000 non-null  float64
 31  psfMag_g      100000 non-null  float64
 32  psfMag_i      100000 non-null  float64
 33  psfMag_z      100000 non-null  float64
 34  expAB_u       100000 non-null  float64
 35  expAB_g       100000 non-null  float64
 36  expAB_r       100000 non-null  float64
 37  expAB_i       100000 non-null  float64
 38  expAB_z       100000 non-null  float64
 39  class         100000 non-null  object
 40  subclass      100000 non-null  object
 41  redshift      100000 non-null  float64
 42  redshift_err  100000 non-null  float64
dtypes: float64(39), int64(1), object(2), uint64(1)
memory usage: 32.8+ MB
```

SDSS.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
df.isnull().sum()
```

```
objid            0
specobjid        0
ra               0
dec              0
u                0
g                0
r                0
i                0
z                0
modelFlux_u      0
modelFlux_g      0
modelFlux_r      0
modelFlux_i      0
modelFlux_z      0
petroRad_u       0
petroRad_g       0
petroRad_i       0
petroRad_r       0
petroRad_z       0
petroFlux_u      0
petroFlux_g      0
petroFlux_i      0
petroFlux_r      0
petroFlux_z      0
petroR50_u       0
petroR50_g       0
petroR50_i       0
petroR50_r       0
petroR50_z       0
psfMag_u         0
psfMag_r         0
psfMag_g         0
psfMag_i         0
psfMag_z         0
expAB_u          0
expAB_g          0
expAB_r          0
expAB_i          0
expAB_z          0
class            0
subclass         0
redshift         0
redshift_err     0
dtype: int64
```

SDSS.ipynb

File Edit View Insert Runtime Tools Help All changes saved

CHANHING THE DATATYPE OF SUBCLASS FROM OBJECT TO INT

```
df['subclass'].replace(['STARFORMING','STARBURST'],[0,1],inplace=True)
df['class'].replace(['GALAXY'],[0],inplace=True)
```

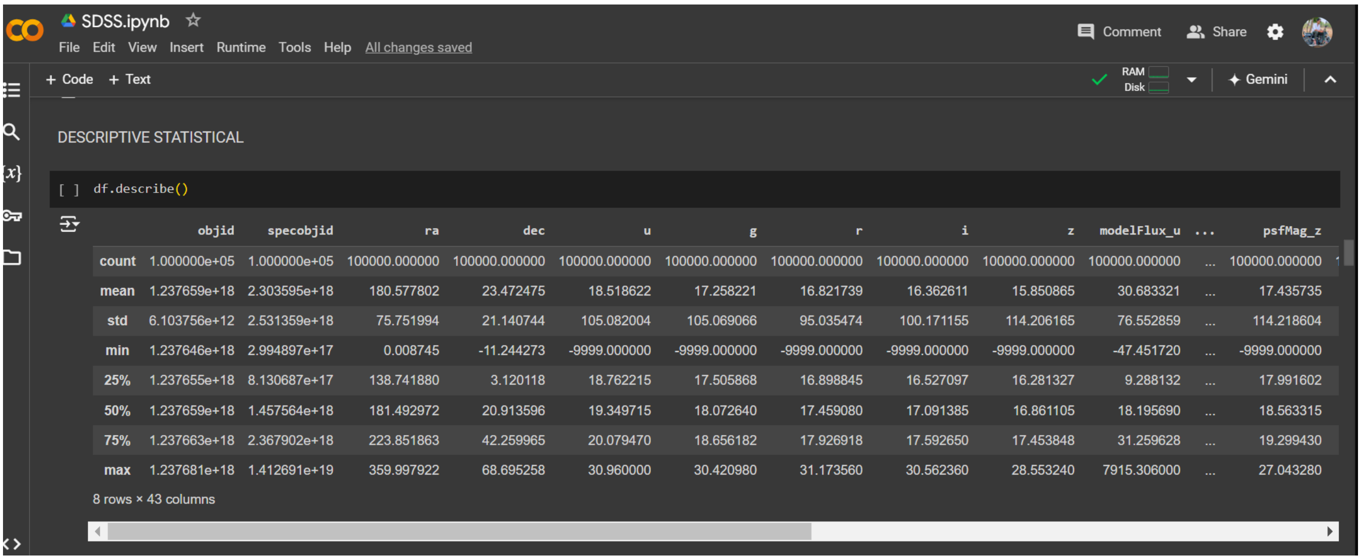
```
df.head()
```

	objid	specobjid	ra	dec	u	g	r	i	z	modelFlux_u	...	psfMag_z	expAB_u	expAB_
0	1237646587710669400	8175185722644649984	82.038679	0.847177	21.73818	20.26633	19.32409	18.64037	18.23833	2.007378	...	19.43575	0.099951	0.31186
1	1237646588247540577	8175186822156277760	82.138894	1.063072	20.66761	19.32016	18.67888	18.24693	18.04122	5.403369	...	18.85012	0.366549	0.51687
2	1237646588247540758	8175187097034184704	82.028510	1.104003	23.63531	21.19671	19.92297	19.31443	18.68396	0.295693	...	19.42235	0.050000	0.41713
3	1237648702973083853	332152325571373056	198.544469	-1.097059	20.12374	18.41520	17.47202	17.05297	16.72423	8.920645	...	18.03204	0.310763	0.35682
4	1237648702973149350	332154249716721664	198.706864	-1.046217	-9999.00000	-9999.00000	18.37762	18.13383	17.78497	0.000000	...	19.02880	-9999.000000	-9999.00000

5 rows × 43 columns

```
df.shape
```

```
(100000, 43)
```

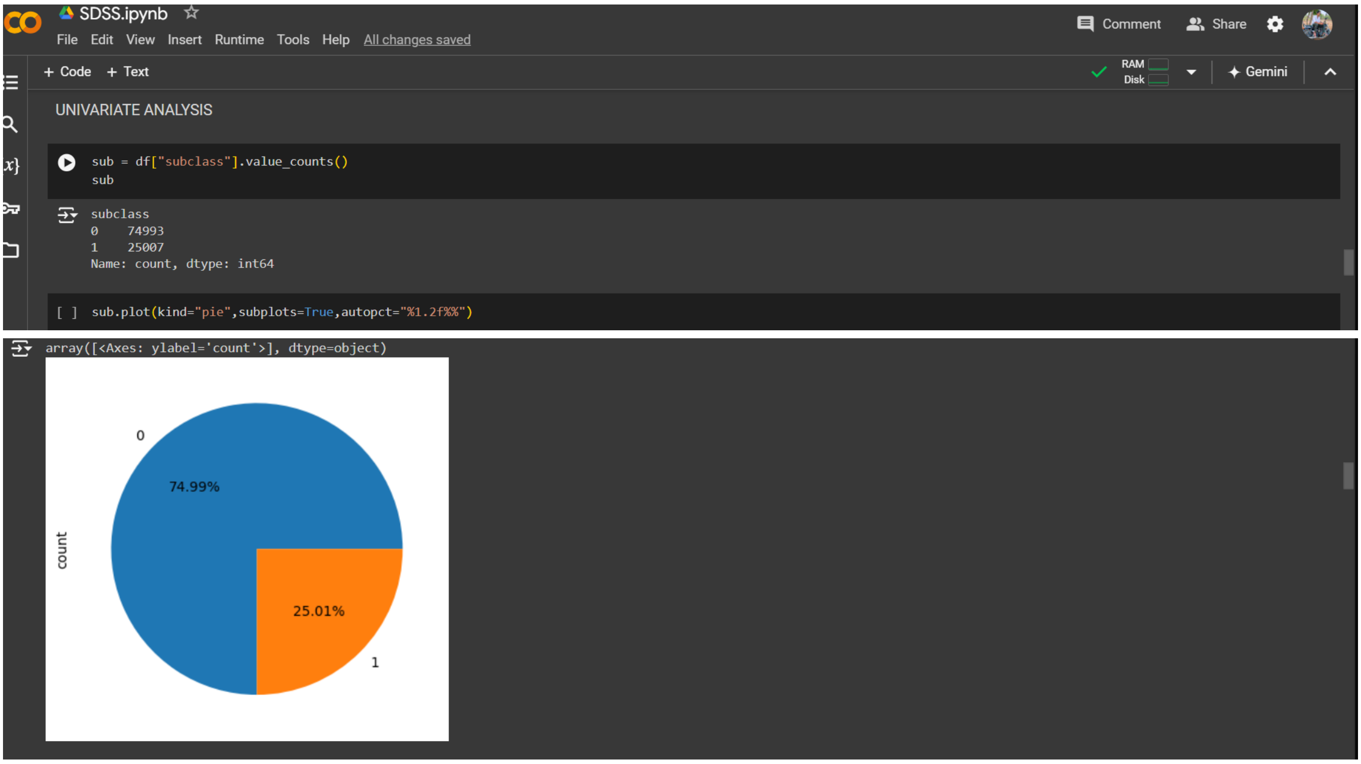


DESCRIPTIVE STATISTICAL

```
df.describe()
```

	objid	specobjid	ra	dec	u	g	r	i	z	modelFlux_u	...	psfMag_z
count	1.000000e+05	1.000000e+05	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	100000.000000	...	100000.000000
mean	1.237659e+18	2.303595e+18	180.577802	23.472475	18.518622	17.258221	16.821739	16.362611	15.850865	30.683321	...	17.435735
std	6.103756e+12	2.531359e+18	75.751994	21.140744	105.082004	105.069066	95.035474	100.171155	114.206165	76.552859	...	114.218604
min	1.237646e+18	2.994897e+17	0.008745	-11.244273	-9999.000000	-9999.000000	-9999.000000	-9999.000000	-9999.000000	-47.451720	...	-9999.000000
25%	1.237655e+18	8.130687e+17	138.741880	3.120118	18.762215	17.505868	16.898845	16.527097	16.281327	9.288132	...	17.991602
50%	1.237659e+18	1.457564e+18	181.492972	20.913596	19.349715	18.072640	17.459080	17.091385	16.861105	18.195690	...	18.563315
75%	1.237663e+18	2.367902e+18	223.851863	42.259965	20.079470	18.656182	17.926918	17.592650	17.453848	31.259628	...	19.299430
max	1.237681e+18	1.412691e+19	359.997922	68.695258	30.960000	30.420980	31.173560	30.562360	28.553240	7915.306000	...	27.043280

8 rows × 43 columns



UNIVARIATE ANALYSIS

```
sub = df["subclass"].value_counts()
sub
```

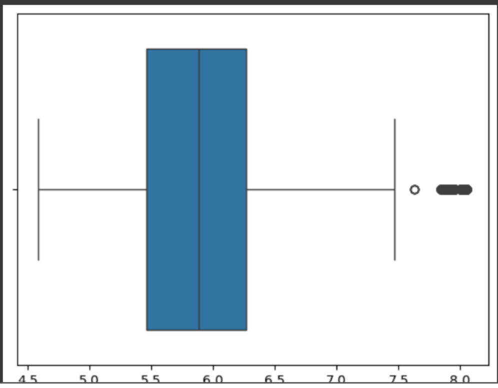
```
subclass
0    74993
1    25007
Name: count, dtype: int64
```

```
sub.plot(kind="pie",subplots=True,autopct="%1.2f%%")
```

```
array([<Axes: ylabel='count'>], dtype=object)
```

HANDLING THE OUTLIERS

```python
def func(col):
  sns.boxplot(x=col,data=df)
  plt.show()
for i in df.columns:
  func(i)
```



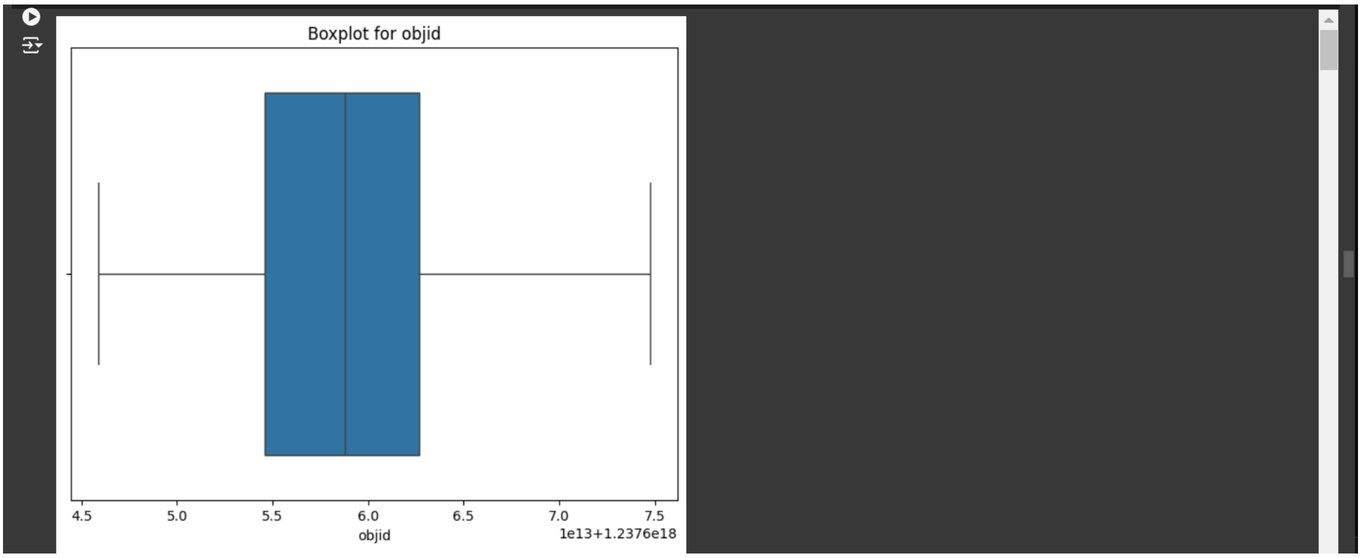
```python
# Iterate over each column
for column in df.columns:
    # Check if the column contains numeric data
    if pd.api.types.is_numeric_dtype(df[column]):
        # Calculate quantiles
        quant = df[column].quantile(q=[0.75, 0.25])
        Q3 = quant.loc[0.75]
        Q1 = quant.loc[0.25]

        # Calculate IQR
        IQR = Q3 - Q1

        # Calculate lower and upper bounds for outliers
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR

        # Replace outliers with values within the bounds
        df[column] = np.where(df[column] < lower_bound, lower_bound, df[column])
        df[column] = np.where(df[column] > upper_bound, upper_bound, df[column])
```

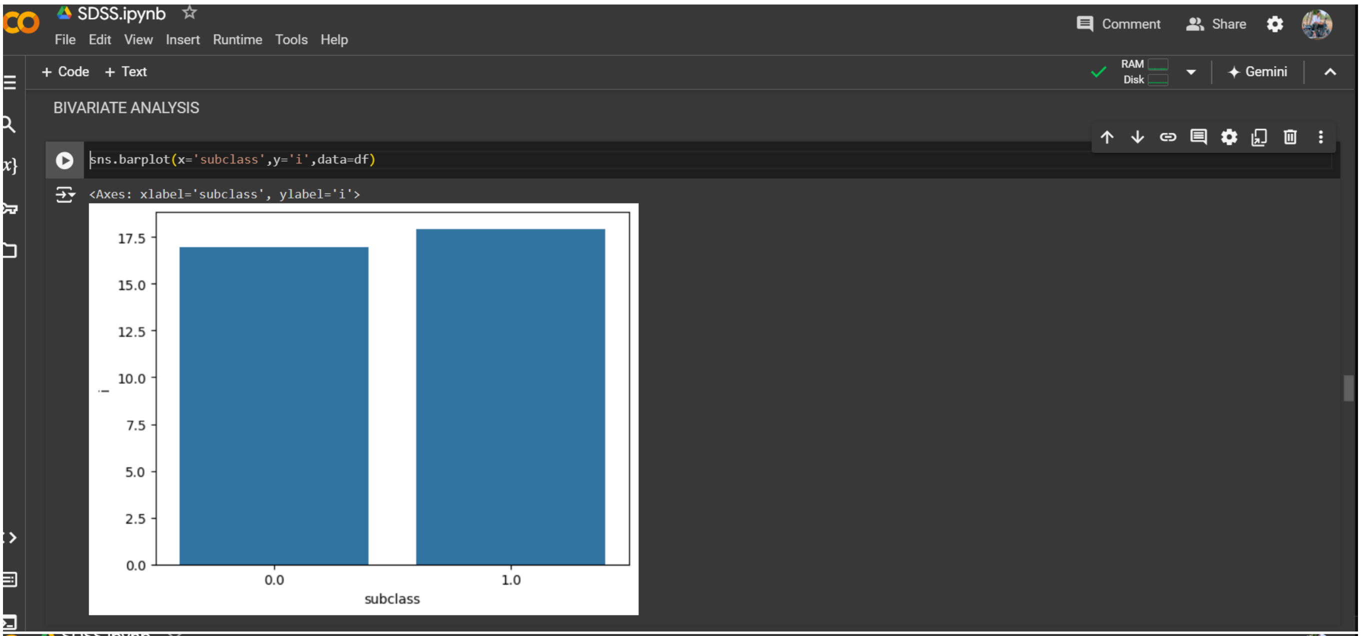
```python
# Iterate over each column and plot boxplot
for column in df.columns:
    plt.figure(figsize=(8, 6))  # Adjust the figure size as needed
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot for {column}')
    plt.xlabel(column)
    plt.show()
```



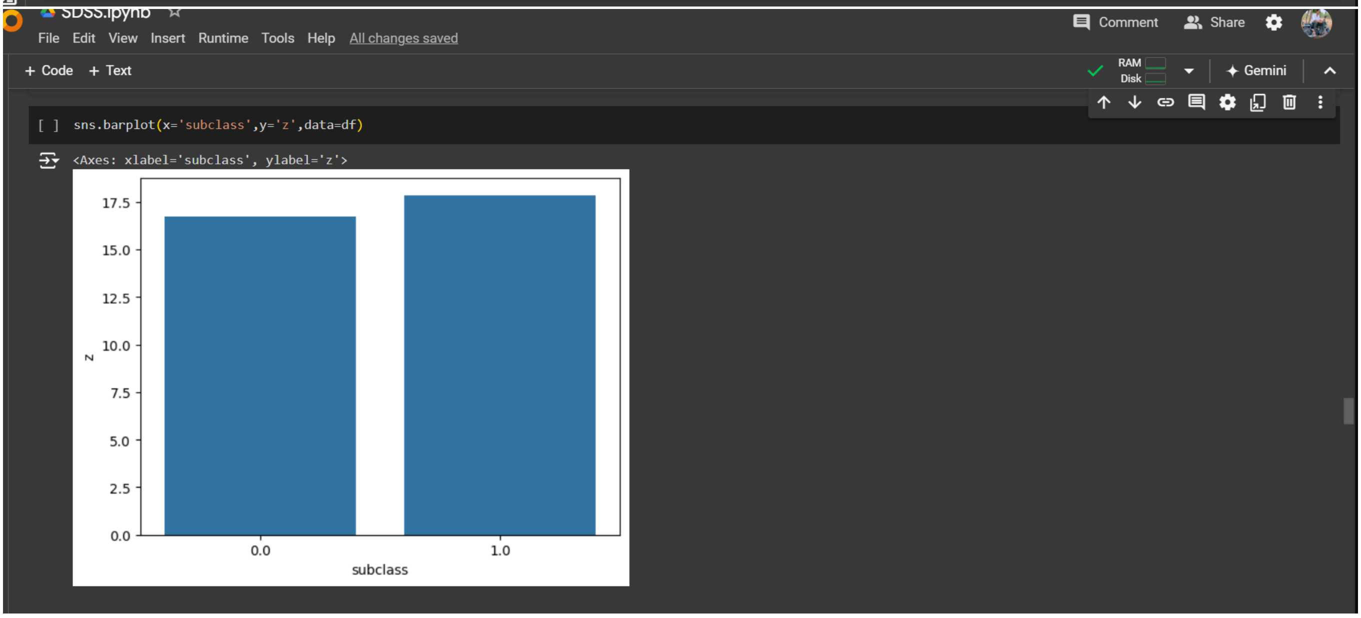
Boxplot for objid
4.5
5.0
5.5
6.0
6.5
7.0
7.5
objid
1e13+1.2376e18

BIVARIATE ANALYSIS

```
sns.barplot(x='subclass',y='i',data=df)
```



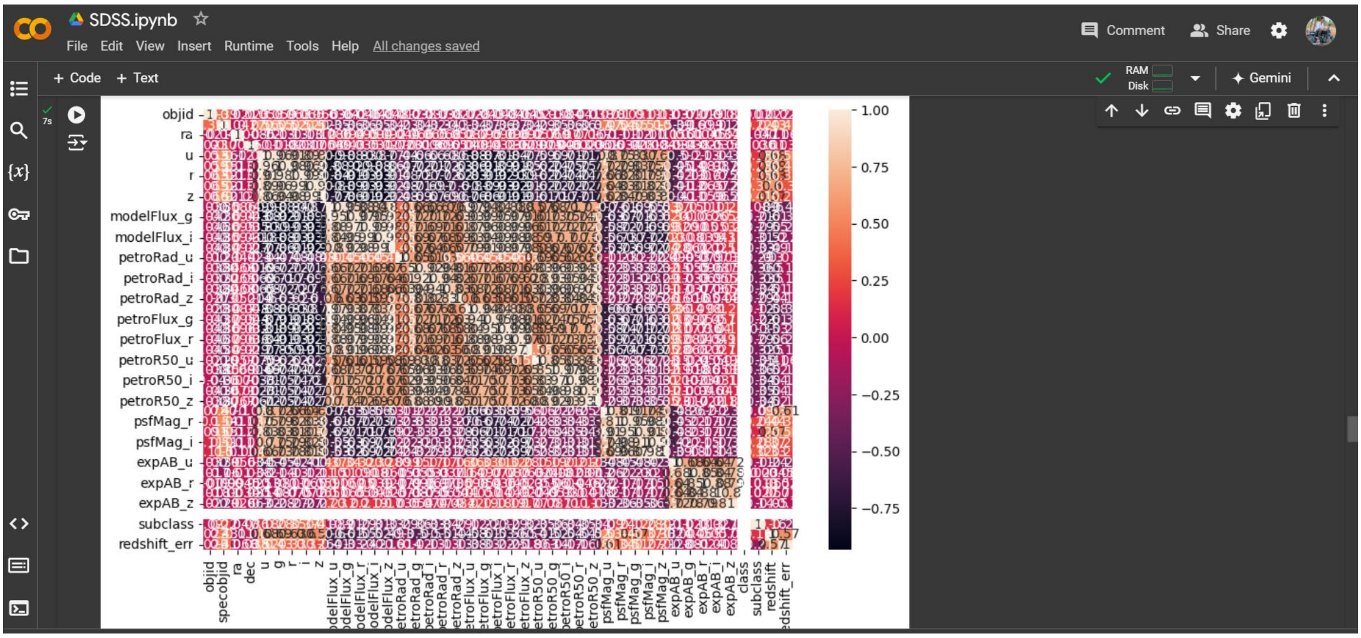

```
sns.barplot(x='subclass',y='z',data=df)
```



HEAT MAP

MULTIVARIATE ANALYSIS

```
[ ] plt.figure(figsize=(30,22))
    sns.heatmap(df.corr(),annot=True)
    plt.show()
```



SELECTING BEST FEATURES USING SELECT K BEST

```
x=df.drop(['subclass',],axis=1)
y=df['subclass']
```

```
#i want to know top best columns in the data frame using selectkBest k=10
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
# Assuming x and y are your data and target variables
selector = SelectKBest(score_func=f_classif, k=10)  # select top 10 features
#selector = selectKBest(score_func=chi2,k=10)  # For classificaton takes with non negative features

#fit selector to the data
x_selected = selector.fit_transform(x,y) # Fixed: Use selector instead of features
# Get the names of the selected features
selected_features = x.columns[selector.get_support()] # Fixed: Use x instead of X

# print the selected features
print("selected features:",selected_features)
```

```
selected features: Index(['r', 'i', 'z', 'petroRad_g', 'petroRad_r', 'petroR50_u', 'petroR50_g',
       'petroR50_i', 'petroR50_r', 'petroR50_z'],
      dtype='object')
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_selection.py:112: UserWarning: Features [39] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/feature_selection/_univariate_selection.py:113: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
```

BALANCING VALUE COUNTS USING SMOTE

```
# Assuming your target column is 'subclass' in your DataFrame 'df'
x = df.drop(['subclass','class'], axis=1)
y = df['subclass']

# Initialize SMOTE
from imblearn.over_sampling import SMOTE
smote = SMOTE() # Initialize the SMOTE object

x_resampled, y_resampled = smote.fit_resample(x,y)

# check the new value counts
print(pd.Series(y_resampled).value_counts())
```

```
subclass
0.0    74993
1.0    74993
Name: count, dtype: int64
```

SDSS.ipynb

File Edit View Insert Runtime Tools Help

\+ Code + Text

SPLITTING DATA INTO TRAIN AND TEST

```
[22] df1=df[['i','z','modelFlux_z','petroRad_g','petroRad_r','petroFlux_z','petroR50_u','petroR50_g','petroR50_i','petroR50_r','subclass']]
```

```
[23] from sklearn.model_selection import train_test_split
     x=df1[['i','z','modelFlux_z','petroRad_g','petroRad_r','petroFlux_z','petroR50_u','petroR50_g','petroR50_i','petroR50_r']]
     y=df1["subclass"]
     x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20)
```

SCALING THE FEATURE VARIABLES USING STANDARDSCALER METHOD

```
from sklearn.preprocessing import StandardScaler
# create a scalar object
sc=StandardScaler()

# Transform your data
scaled_data = sc.fit_transform(x_train)
```

```
[ ] from sklearn.metrics import accuracy_score # Import the accuracy_score function
    print(accuracy_score(y_pred,y_test))
```

```
0.77375
```

```
[ ] print(accuracy_score(y_pred,y_test))
```

```
0.77375
```

TRAINING THE MODEL IN MULTIPLE ALGORITHMS

DECISION TREE CLASSIFIER

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier() # Use the correct class name with capitalization

# Train the classifier on the training data
clf.fit(x_train, y_train)

# make predictions on the testing data
y_pred = clf.predict(x_test)

# Evaluate the classifier
from sklearn.metrics import classification_report # Don't forget to import this module
report = classification_report(y_test, y_pred)
print("classification Repoprt:\n",report)
```

```
classification Repoprt:
               precision    recall  f1-score   support

         0.0       0.84      0.85      0.85     14845
         1.0       0.56      0.55      0.56      5155

    accuracy                           0.77     20000
   macro avg       0.70      0.70      0.70     20000
weighted avg       0.77      0.77      0.77     20000
```

LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, recall_score, precision_score, confusion_matrix, f1_score

lg = LogisticRegression()
log=lg.fit(x_train,y_train)
print("confusion matrix: \n",confusion_matrix(y_test,y_pred))
print("..........................................")
print("classification report:\n",classification_report(y_test,y_pred))
print("..........................................")
print("accuracy score:\n",accuracy_score(y_test,y_pred))
```

```
confusion matrix:
 [[12644  2201]
 [ 2324  2831]]
...........................................
classification report:
               precision    recall  f1-score   support

         0.0       0.84      0.85      0.85     14845
         1.0       0.56      0.55      0.56      5155

    accuracy                           0.77     20000
   macro avg       0.70      0.70      0.70     20000
weighted avg       0.77      0.77      0.77     20000

...........................................
accuracy score:
 0.77375
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
print(accuracy_score(y_pred,y_test))
```

```
0.77375
```

RANDOM FOREST CLASSIFIER

```
from sklearn.ensemble import RandomForestClassifier
RF=RandomForestClassifier

# Train the Random Forest classifier
RF = RandomForestClassifier()
```

```
RF.fit(x_train,y_train)
RFtrain=RF.predict(x_train)
RFtest=RF.predict(x_test)
```

+ Code + Text

```
from sklearn.metrics import confusion_matrix, classification_report # Import necessary functions
```

```
# print classification report , confusion matrix
print(confusion_matrix(RFtrain,y_train))
print(confusion_matrix(RFtest,y_test))
print(classification_report(RFtrain,y_train)) # Fix the typo here
print(classification_report(RFtest,y_test)) # Fix the typo here
```

```
[[60148    93]
 [    0 19759]]
[[13869  2261]
 [  976  2894]]
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     60241
         1.0       1.00      1.00      1.00     19759

    accuracy                           1.00     80000
   macro avg       1.00      1.00      1.00     80000
weighted avg       1.00      1.00      1.00     80000

              precision    recall  f1-score   support

         0.0       0.93      0.86      0.90     16130
         1.0       0.56      0.75      0.64      3870

    accuracy                           0.84     20000
   macro avg       0.75      0.80      0.77     20000
weighted avg       0.86      0.84      0.85     20000
```

```
[ ] print(accuracy_score(RFtrain,y_train))
    print(accuracy_score(RFtest,y_test))

    0.9988375
    0.83815

[ ] from sklearn.metrics import confusion_matrix, classification_report, accuracy_score # Import the accuracy_score function

    print(accuracy_score(RFtrain,y_train))
    print(accuracy_score(RFtest,y_test))

    0.9988375
    0.83815

[ ] RF.fit(x_train,y_train) # Train the model on your training data

    ▾ RandomForestClassifier
    RandomForestClassifier()
```

TEST THE MODEL

```
[ ] RF.predict([[16.946170,16.708910,207.218700,4.180779,4.060687,194.731000,2.141953,2.149080,2.056686,2.055798]])

    /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
      warnings.warn(
    array([0.])

[ ] RF.predict([[17.675285,17.53775,104.25655,3.397512,3.3975512,3.424717,90.717547,1.632243,1.548225,1.596137]])

    /usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomForestClassifier was fitted with feature names
      warnings.warn(
    array([0.])
```

SAVE THE MODEL

```
[ ] import pickle

    pickle.dump(RF,open("RF.pkl","wb"))
```

```
[ ] ys4tn_cls = np.array(pd.cut(ys4tn_pred, bins=intervals, labels=labels))
    ys4tt_cls = np.array(pd.cut(ys4tt_pred, bins=intervals, labels=labels))

[ ] acc_s4tn = accuracy_score(ys_val_train, ys4tn_cls)
    print("Accuracy Score:", acc_s4tn)

    acc_s4tt = accuracy_score(ys_val_test, ys4tt_cls)
    print("Accuracy Score:", acc_s4tt)

    Accuracy Score: 0.7516516841763241
    Accuracy Score: 0.7453259221829207

[ ] # North data metrics
    model_names = ['Linear Regression','Random Forest Regressor', 'SVM', 'Gradient Boosting']
    R2_values = [r2_n1,r2_n2,r2_n3,r2_n4]
    Accuracy_scores = [acc_n1tt,acc_n2tt,acc_n3tt,acc_n4tt]

    metric_north = {'Model': model_names, 'R2 Score': R2_values, 'Accuracy': Accuracy_scores}
    north_metric = pd.DataFrame(metric_north)
```