INNOVATIVE PRODUCT DEVELOPMENT REPORT

ALGORITHMIC DECISION-MAKING METHODS FOR FAIR CREDIT SCORING

Submitted by:

D. VARSHA	**G. ANJALI**	**G. VENNELA**
22RH1A0566	**22RH1A0573**	**22RH1A0584**

Under the Esteemed Guidance of

S. PREETHI

Associate Professor

Department of CSE

In partial fulfilment of the Academic Requirements for the Degree of

BACHELOR OF TECHNOLOGY

Computer Science & Engineering

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NAAC with 'A+' Grade, UGC, Govt. of India | Programmes Accredited by NBA National Ranking by NIRF Innovation-Rank band(151-300),MHRD, Govt. of India

Approved by AICTE, Affiliated to JNTUH,ISO 9001-2015 Certified Institution

Maisammaguda, Dhulapally, Secunderabad, Kompally-500 100.

www.mallareddyecw.com

2024-2025

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NAAC with 'A+' Grade, UGC, Govt. of India | Programmes Accredited by NBA National Ranking by NIRF Innovation-Rank band(151300),MHRD, Govt.of India Approved by AICTE, Affiliated to JNTUH,ISO 90012015 Certified Institution Maisammaguda, Dhulapally, Secunderabad, Kompally-500 100.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the work embodies in this dissertation entitled ALGORITHMIC DECISION-MAKING METHODS FOR FAIR CREDIT SCORING being submitted by D.VARSHA(22RH1A0566),G.ANJALI (22RH1A0573), G.VENNELA (22RH1A0584) for partial fulfilment of the requirement for the award of BACHELOR OF TECHNOLOGY in Computer Science and Engineering discipline to Malla Reddy Engineering College for Women, Maisammaguda, Secunderabad during the academic year 2023-2024 is a record of bonafide piece of work, undertaken by her supervision of the undersigned.

Supervisor's Signature	**Head of the Department**
S. PREETHI	**Dr. Y. GEETHA REDDY**
Associate Professor	Professor

EXTERNAL EXAMINER

MALLA REDDY ENGINEERING COLLEGE FOR WOMEN

(Autonomous Institution-UGC, Govt. of India)

Accredited by NAAC with 'A+' Grade, UGC, Govt. of India | Programmes Accredited by NBA National Ranking by NIRF Innovation-Rank band(151300),MHRD, Govt.of India Approved by AICTE, Affiliated to JNTUH,ISO 90012015 Certified Institution Maisammaguda, Dhulapally, Secunderabad, Kompally-500 100.

DECLARATION

We hereby declare that our project entitled ALGORITHMIC DECISION-MAKING METHODS FOR FAIR CREDIT SCORING submitted to **Malla Reddy Engineering College for Women, Hyderabad** for the award of the Degree of Bachelor of Technology in **Computer Science and Engineering** is a result of original research work done by us. It is declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of Degree.

D.VARSHA (22RH1A0566)
G.ANJALI (22RH1A0573)
G.VENNELA (22RH1A0584)

ACKNOWLEDGEMENT

We feel ourselves honoured and privileged to place our warm salutation to our college Malla Reddy Engineering College for Women and Department of Computer Science and Engineering which gave us the opportunity to have Expertise in engineering and profound technical knowledge.

We would like to deeply thank our Honourable Minister of Telangana State
Sri CH. MALLA REDDY Garu, Founder Chairman MRGI, the largest cluster of institutions in the state of Telangana for providing us with all the resources in the college to make our project success.

We wish to convey gratitude to our Principal **Dr. Y. MADHAVEE LATHA**, for providing us with the environment and mean to enrich our skills and motivating us in our endeavour and helping us to realize our full potential.

We express our sincere gratitude to **Dr . Y. GEETHA REDDY**, Head of the Department of Computer Science and Engineering for inspiring us to take up a project to this subject and successfully guiding us towards its completion.

We would like to thank our Internal Guide **S. PREETHI**, Associate
Professor and all the Faculty members for their valuable guidance and encouragement towards the completion of our project work.

With Regards and Gratitude,

D.VARSHA (22RH1A0566)
G.ANJALI (22RH1A0573)
G.VENNELA (22RH1A0584)

ABSTRACT

In recent years, the integration of algorithmic decision-making methods in credit scoring has gained prominence, promising enhanced accuracy and efficiency in assessing creditworthiness. However, these methods also raise significant concerns regarding fairness and bias, as traditional scoring models may inadvertently reinforce existing inequalities. This paper explores various algorithmic approaches to credit scoring, including machine learning, statistical methods, and hybrid models, while critically evaluating their implications for fairness. We examine the sources of bias in credit scoring algorithms, such as biased training data and model interpretability challenges. By analysing recent case studies and empirical evidence, we propose a framework for implementing fair credit scoring practices, emphasizing the importance of transparency, accountability, and continuous monitoring. The findings suggest that while algorithmic methods can improve predictive performance, their deployment must be accompanied by robust measures to ensure equitable outcomes across diverse demographic groups. This work aims to contribute to the ongoing discourse on responsible AI and the ethical dimensions of financial technology.

INDEX

S.NO	CHAPTER	Page no.
1	**CHAPTER-I**	5
	1.INTRODUCTION	
2	**CHAPTER-II**	6 - 10
	2.TECHNOLOGIES LEARNT	
	2.1 EXISTING SYSTEM	
	2.2 PROPOSED SOLUTION	
3	**CHAPTER-III**	11
	SYSTEM REQUIREMENTS	
	3.1SOFTWARE REQUIREMENTS	
	3.2 HARDWARE REQUIREMENTS	
4	**CHAPTER-IV**	12 - 14
	4. SYSTEM DESIGN	
	4.1 BLOCK DIAGRAM	
	4.2 UML DIAGRAM	
5	**CHAPTER-V**	15 - 26
	5. SYSTEM IMPLEMENTATION	
6	**CHAPTER-VI**	27 - 39
	6. RESULT AND ANALYSIS	
7	**CHAPTER-VII**	40 - 41
	7. CONCULSION AND FUTURE SCOPE	
	7.1 CONCLUSION	
	7.2 FUTURE SCOPE	
8	**CHAPTER-VIII**	42
	REFERENCES	

CHAPTER-I
INTRODUCTION

All financial credit organizations may follow traditional approach of judging applicant based on his past payment history which will not always accurate and may contains discrimination. To overcome from above issue author of this paper employing algorithm-based decision to judge fair credit scoring. Scoring will be based on 12 different mitigation methods which will applied on each application input features data and this features will be input to Machine learning algorithm to predict applicant approval. Propose paper contains following 12 mitigation methods Reweighing, Disparate Impact, Learning Fair Representation, Meta Classifier, Reject Option Classification, Calibrated Eg: Odds Post Processing, Exponentiated Gradient Reduction, Adversarial Debiasing, Grid Search Reduction, Gerry Fair Classifier, No Bias Mitigation, Optimized Pre-processing. Each method will be applied on input features data to predict applicant approval and each method description you can read from base paper. Among above mention 12 methods we are able to execute first 10 methods and last 2 methods are not executing. Each method performance is evaluated in terms of accuracy score. To train above methods author has used German and Loan applicant dataset but we are using German dataset. We have coded this project using JUPYTER notebook and below are the code and output screen with blue colour comments

CHAPTER-II

TECHNOLOGIES LEARNT

2.1. EXISTING SYSTEM

Data Processing and Management

- **Databases**: SQL and NoSQL databases (e.g., PostgreSQL, MongoDB) for structured and unstructured data.
- **ETL Tools**: Tools like Apache Spark and Apache Kafka for data integration and processing.

Machine Learning Frameworks

- **Libraries**: Popular libraries like Scikit-learn, TensorFlow, and PyTorch for building and training models.
- **Fairness Libraries**: Specialized libraries such as Fairlearn and AI Fairness 360 that focus on fairness in machine learning.

Statistical Techniques

- **Descriptive Statistics**: Understanding data distributions, means, variances, etc.
- **Inferential Statistics**: Techniques for hypothesis testing and confidence intervals.

Modelling Techniques

- **Supervised Learning**: Regression models (e.g., linear regression) and classification models (e.g., logistic regression, decision trees).
- **Ensemble Methods**: Techniques like random forests and gradient boosting for improved accuracy.

NLP and Text Analysis

- **Text Processing Libraries**: NLTK and SpaCy for analyzing text data related to credit applications and customer feedback.

Explainability and Interpretability

- **LIME and SHAP**: Tools for providing insights into model predictions and understanding feature importance.

Detection and Mitigation

- **Fairness Metrics**: Techniques to assess and measure bias in models (e.g., disparate impact, equal opportunity).
- **Adversarial Debiasing**: Algorithms that aim to reduce bias in model outputs.

Regulatory Compliance Knowledge

- **Understanding of Regulations**: Familiarity with laws like FCRA and ECOA to ensure compliance in scoring practices.

Cloud Computing and Deployment

- **Cloud Services**: AWS, Azure, and Google Cloud for scalable infrastructure and deployment of machine learning models.
- **Containerization**: Technologies like Docker for creating reproducible environments.

Continuous Monitoring and Feedback Loops

- **Monitoring Tools**: Systems for tracking model performance, detecting drift, and continuously improving model fairness.

Data Privacy Techniques

- **Differential Privacy**: Methods to safeguard individual data points while allowing for data analysis.
- **Federated Learning**: Approaches to train models on decentralized data to enhance privacy. These technologies provide a robust foundation for developing.

2.2. PROPOSED SYSTEM

1. **Machine Learning Algorithms**:
   - **Supervised Learning**: Techniques like regression analysis and classification (e.g., logistic regression, decision trees) can be used to predict credit scores based on historical data.
   - **Unsupervised Learning**: Clustering methods to identify patterns in data without predefined labels.
2. **Fairness Metrics**:
   - Implementation of fairness metrics (e.g., demographic parity, equal opportunity) to evaluate the performance of credit scoring models and ensure that they do not discriminate against specific groups.
3. **Bias Mitigation Techniques**:
   - Algorithms such as reweighing, adversarial debiasing, or using fair representations to reduce bias in credit scoring models.
4. **Data Preprocessing**:
   - Techniques for handling imbalanced datasets and ensuring representative sampling, which may involve oversampling or under sampling techniques.
5. **Feature Selection and Engineering**:
   - Identifying and creating relevant features that contribute to more accurate and fair credit assessments while avoiding sensitive attributes.
6. **Explainable AI (XAI)**:
   - Use of XAI techniques to provide transparency in decision-making processes, allowing stakeholders to understand how credit scores are derived.
7. **Risk Assessment Frameworks**:
   - Integrating frameworks that assess credit risk while maintaining fairness and compliance with regulations.
8. **Blockchain Technology**:
   - Potential use of blockchain for secure, transparent storage of credit data and transactions, enhancing trust and accountability.

CHAPTER-III

SYSTEM REQUIREMENTS:

3.1. HARDWARE REQUIREMENTS:

✞ System	:	MINIMUM i3.
✞ Hard Disk	:	40 GB.
✞ Monitor	:	14’ Colour Monitor.
✞ Mouse	:	Optical Mouse.
✞ Ram	:	4GB.

3.2. SOFTWARE REQUIREMENTS:

✞ Operating system	:	Windows 11.
✞ Coding Language	:	Python 3.7.

CHAPTER-IV

SYSTEM DESIGN

a. Data Collection Module

- **Data Sources**: Integrate diverse data sources such as credit history, income information, payment behaviors, and demographic data (while avoiding sensitive attributes).
- **Data Privacy**: Implement measures to ensure data anonymization and compliance with regulations (e.g., GDPR, FCRA).

b. Data Preprocessing Module

- **Cleaning and Transformation**: Handle missing values, outliers, and standardize formats.
- **Feature Engineering**: Create meaningful features while minimizing bias (e.g., removing sensitive attributes).
- **Sampling Techniques**: Apply oversampling/undersampling methods to address class imbalances.

c. Model Development Module

- **Algorithm Selection**: Choose appropriate algorithms (e.g., decision trees, random forests, logistic regression, gradient boosting) that suit the data characteristics.
- **Fairness Constraints**: Implement algorithms with built-in fairness constraints or apply fairness-enhancing techniques post-hoc (e.g., reweighting, adversarial debiasing).

d. Model Training and Evaluation Module

- **Training**: Use a portion of the dataset to train the models.
- **Validation**: Employ cross-validation to ensure robustness.

- **Fairness Evaluation**: Use fairness metrics (e.g., demographic parity, equalized odds) to assess model performance across different demographic groups.

e. Explainability Module

- **XAI Techniques**: Incorporate methods like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to provide insights into model decisions.
- **Visualization Tools**: Develop dashboards to visualize model performance and fairness metrics.

2. Decision-Making Framework

a. Scoring System

- **Score Calculation**: Design a transparent scoring mechanism that combines various features and their weights.
- **Threshold Setting**: Define thresholds for credit approval that consider both performance and fairness.

b. Risk Assessment

- **Risk Categories**: Classify applicants into risk categories based on their credit scores and other relevant factors.
- **Feedback Mechanism**: Provide feedback to applicants about their credit scores and the factors affecting them.

3. Compliance and Monitoring Module

- **Regulatory Compliance**: Ensure adherence to legal standards and guidelines for credit scoring.
- **Monitoring and Reporting**: Set up systems for ongoing monitoring of model performance and fairness metrics over time, adjusting as necessary.

4. User Interface

- **Applicant Portal**: A user-friendly interface for applicants to apply for credit, view their scores, and receive explanations.
- **Administrator Dashboard**: A dashboard for credit analysts to monitor model performance, run reports, and adjust model parameters.

5. Feedback Loop

- **Continuous Improvement**: Implement a feedback mechanism where the system learns from new data and user interactions, improving model accuracy and fairness over time.

6. Security and Ethical Considerations

- **Data Security**: Ensure robust security measures are in place to protect sensitive data.
- Ethical Guidelines: Establish a framework for ethical decision-making that guides the design and implementation of the credit scoring system

CHAPTER V

SYSTEM IMPLEMENTATION

SOFTWARE ENVIRONMENT

What is Python :-

Python is currently the most widely used multi-purpose, high-level programming language. Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java. programmers have to type relatively less and indentation requirement of the language, makes them readable all the time. **etc** Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber….The biggest strength of Python is huge collection of standard library which can be used for the following –

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

Advantages of Python :-

Let's see how Python dominates over other languages.

1. Extensive Libraries

Python downloads with an extensive library and it *contain code for various purposes like* regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code.

Advantages of Python Over Other Languages

1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

Disadvantages of Python

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

1. Speed Limitations

We have seen that Python code is executed line by line. But since *Python* is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

What is Machine Learning : -

Before we take a look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data. Fundamentally, machine learning involves building mathematical models to help understand data.

Categories Of Machine Leaning :-

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning. Supervised learning involves somehow modelling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label, and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

Need for Machine Learning

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate and solve complex problems. On the other side, AI is still in its initial stage and haven’t surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”. Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programing logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

Challenges in Machines Learning :-

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.

Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.

Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.

No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.

Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.

Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.

Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

Applications of Machines Learning :-

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML −

- Emotion analysis
- Sentiment analysis
- Error detection and prevention
- Weather forecasting and prediction
- Stock market analysis and forecasting
- Recommendation of products to customer in online shopping

How to Start Learning Machine Learning?

Arthur Samuel coined the term "Machine Learning" in 1959 and defined it as a "Field of study that gives computers the capability to learn without being explicitly programmed".
And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is The Best Job of 2019 with a *344%* growth and an average base salary of $146,085 per year.
But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So this article deals with the Basics of Machine Learning and also the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let's get started!!!
How to start learning ML?
This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

Step 1 – Understand the Prerequisites

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus.

(a) Learn Linear Algebra and Multivariate Calculus

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

(b) Learn Statistics

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So it is no surprise that you need to learn it!!! Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

(c) Learn Python

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is *Python*! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as *Keras*, *TensorFlow*, *Scikit-learn*, etc.

Step 2 – Learn Various ML Concepts

Now that you are done with the prerequisites, you can move on to actually learning ML (Which is the fun part!!!) It's best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

(a) Terminologies of Machine Learning

- Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
- Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.

- Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
- Training – The idea is to give a set of inputs(features) and it's expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.

- Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

(b) Types of Machine Learning

- Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
- Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.

- **Semi-supervised Learning** – This involves using unlabelled data like Unsupervised Learning with a small amount of labelled data. Using labelled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
- **Reinforcement Learning** – This involves learning optimal actions through trial and error. So the next action is decided by learning behaviour that are based on the current state and that will maximize the reward in the future.

Advantages of Machine learning :-

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviours and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation)

With ML, you don't need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software's ; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

Disadvantages of Machine Learning :-

1. Data Acquisition

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

2. Time and Resources

ML needs enough time to let the algorithms learn and develop enough to fulfil their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

3. Interpretation of Results

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

Python Development Steps : -

Guido Van Rossum published the first version of Python code (version 0.9.0) at sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict , str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting Unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it. "Some changes in Python 7.3:

- Print is now a function
- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g. a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e. int. long is int as well. The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
- Text Vs. Data Instead Of Unicode Vs. 8-bit.

Modules Used in Project :-

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

NumPy

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and Python shells, the Jupiter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots *and* thumbnail gallery.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. **Python**

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviours.
- **Install Python Step-by-Step in Windows and Mac:**

  Python a versatile programming language doesn't come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

  The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

CHAPTER VI

RESULT AND ANALYSIS

6.SYSTEM TEST

Testing a system for Algorithmic Decision-Making Methods for Fair Credit Scoring involves multiple stages to ensure that the model not only performs well in terms of accuracy but also adheres to fairness standards. Here's a structured approach to system testing:

TYPES OF TESTS

1. Unit Testing

a. Data Preprocessing Tests

- Verify that data cleaning functions correctly handle missing values and outliers.
- Ensure feature engineering processes generate the expected features without introducing bias.

b. Model Components Tests

- Test individual algorithms for correctness (e.g., expected output for known inputs).
- Check the implementation of fairness constraints to confirm they are applied correctly.

2. Integration Testing

a. Module Interaction

- Test the interaction between the data collection, preprocessing, model development, and scoring modules to ensure seamless data flow.
- Validate that the output from one module feeds correctly into the next.

b. End-to-End Testing

- Run a complete simulation from data input through scoring to output, ensuring the entire process works as intended.

3. Performance Testing

a. Accuracy Metrics

- Measure traditional performance metrics such as accuracy, precision, recall, and F1 score on the validation dataset.
- Compare the model’s performance against benchmarks (e.g., existing credit scoring models).

b. Scalability Tests

- Evaluate the system’s performance with varying data sizes to ensure it can handle large volumes of credit applications.

4. Fairness Testing

a. Fairness Metrics Evaluation

- Calculate and analyze fairness metrics such as:
    - **Demographic Parity**: Check if the acceptance rates are similar across different demographic groups.
    - **Equal Opportunity**: Ensure that true positive rates are similar for different groups.
    - **Calibration**: Verify that predicted scores correspond to actual outcomes uniformly across groups.

b. Bias Detection

- Conduct tests to identify potential biases in the model’s decisions, using techniques like disparate impact analysis and testing against sensitive attributes.

5. Explainability Testing

a. Interpretability Checks

- Use XAI tools (e.g., SHAP, LIME) to verify that model predictions can be explained and that explanations align with the input features.
- Test the clarity and usefulness of explanations provided to applicants.

b. User Feedback Collection

- Gather feedback from users (credit analysts, applicants) on the clarity of the scoring explanations and overall system usability.

6. Compliance Testing

a. Regulatory Adherence

- Ensure that the system complies with legal standards such as the Fair Credit Reporting Act (FCRA) and GDPR.
- Review data handling practices, consent mechanisms, and transparency in decision-making processes.

7. Longitudinal Testing

a. Monitoring Over Time

- Implement a system to monitor model performance and fairness metrics continuously over time, ensuring the model adapts to changes in data and demographics.
- Periodically re-evaluate the model and its parameters based on new data and feedback.

8. User Acceptance Testing (UAT)

- Involve end-users in testing the system to validate its functionality, usability, and effectiveness from a user perspective.
- Collect qualitative and quantitative feedback to make necessary adjustments.

Test strategy and approach

Creating an approach for Algorithmic Decision Making Methods for Fair Credit Scoring involves a structured framework that encompasses various phases, from data collection to implementation and monitoring. Here’s a detailed approach:

1. Define Objectives and Scope

- **Objectives**: Clearly outline the goals, such as improving credit scoring accuracy, ensuring fairness, and enhancing transparency.

- **Scope**: Identify the parameters and data sources involved, including demographic factors, credit history, and income levels.

2. Data Collection and Preparation

- **Diverse Data Sources**: Gather data from multiple sources to provide a holistic view of applicants, ensuring a mix of traditional and alternative data.
- **Data Quality Assessment**: Evaluate the quality and completeness of the data, addressing missing values and inconsistencies.
- **Anonymization and Privacy**: Implement techniques to anonymize sensitive information and ensure compliance with data protection regulations (e.g., GDPR).

3. Feature Engineering

- **Relevant Features**: Identify and create features that correlate with creditworthiness while avoiding direct use of sensitive attributes (e.g., race, gender).
- **Bias Mitigation**: Use techniques to reduce bias in feature selection and ensure that the features contribute to fair outcomes.

4. Model Selection and Development

- **Algorithm Selection**: Choose appropriate machine learning algorithms (e.g., logistic regression, decision trees, ensemble methods) that are interpretable and suitable for the dataset.
- **Incorporate Fairness Constraints**: Implement algorithms that allow for fairness constraints or use techniques like adversarial debiasing to mitigate bias during training.

5. Training and Evaluation

- **Training Models**: Split the dataset into training and validation sets to build models.
- **Performance Metrics**: Evaluate models using accuracy, precision, recall, and F1 score while also assessing fairness metrics (e.g., demographic parity, equal opportunity).
- **Cross-Validation**: Use cross-validation techniques to ensure robustness and avoid overfitting.

6. Explainability and Transparency

- **XAI Techniques**: Employ explainable AI methods (e.g., SHAP, LIME) to provide insights into model predictions, making the decision-making process transparent.
- **User-Centric Explanations**: Ensure that explanations are accessible and understandable to applicants and stakeholders.

7. Implementation and Deployment

- **Pilot Testing**: Conduct pilot tests in controlled environments to gather initial feedback and make necessary adjustments.
- **User Training**: Train credit analysts and staff on the new system, emphasizing the importance of fairness and interpretability.

8. Monitoring and Feedback

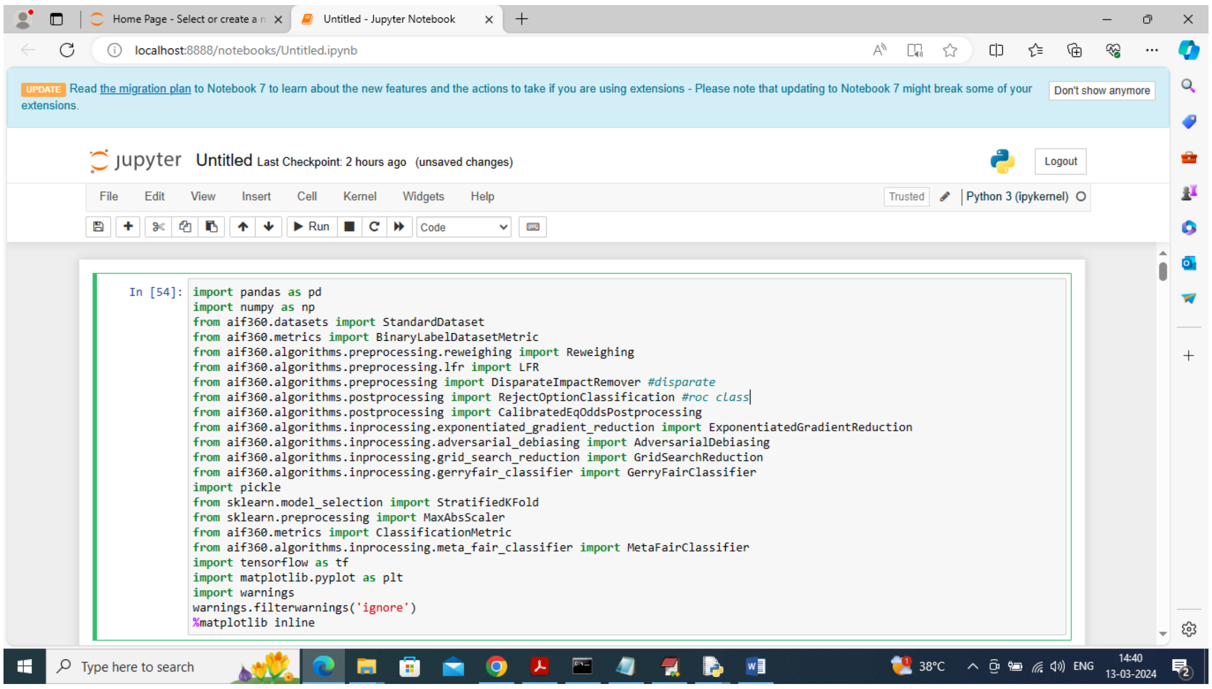
- **Continuous Monitoring**: Establish mechanisms for ongoing evaluation of model performance and fairness, using real-time data and user feedback.
- **Adjustment and Retraining**: Be prepared to adjust models and retrain them based on feedback, changes in data patterns, and regulatory requirements.

9. Stakeholder Engagement

- **Involve Stakeholders**: Engage stakeholders (e.g., regulatory bodies, consumer advocacy groups) throughout the process to ensure transparency and build trust.
- **Public Communication**: Clearly communicate the goals, methods, and outcomes of the credit scoring system to the public.

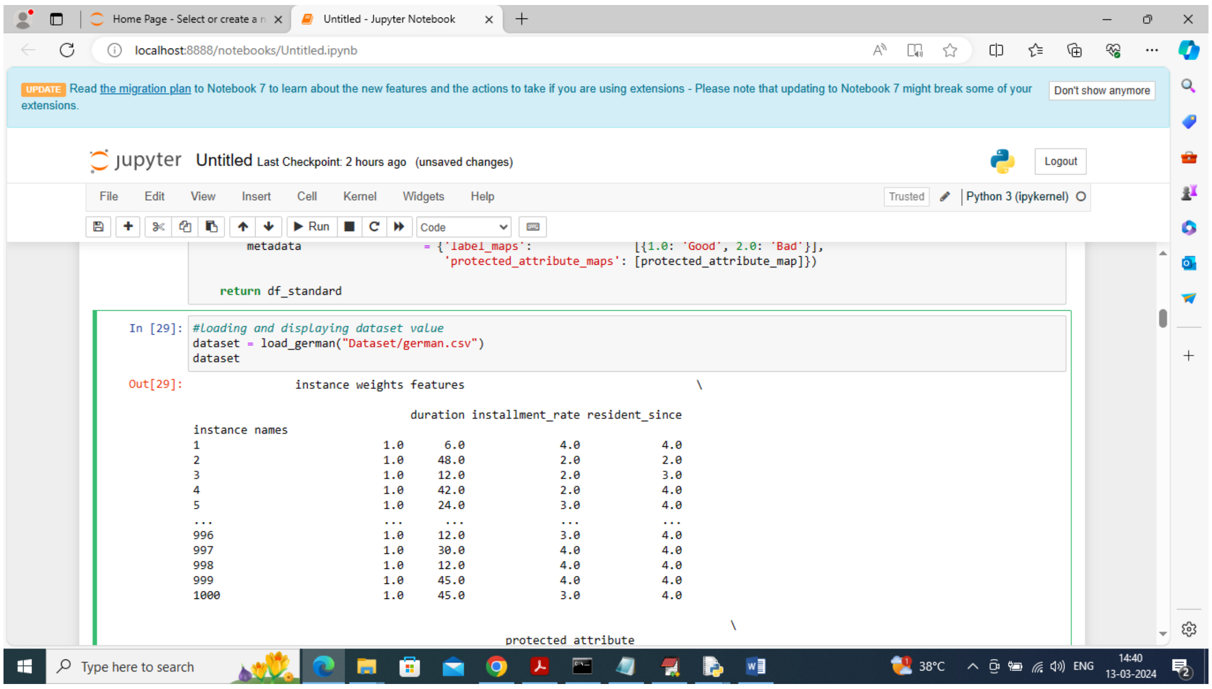
SCREENSHOTS

We have coded this project using JUPYTER notebook and below are the code and output screen with blue colour comments

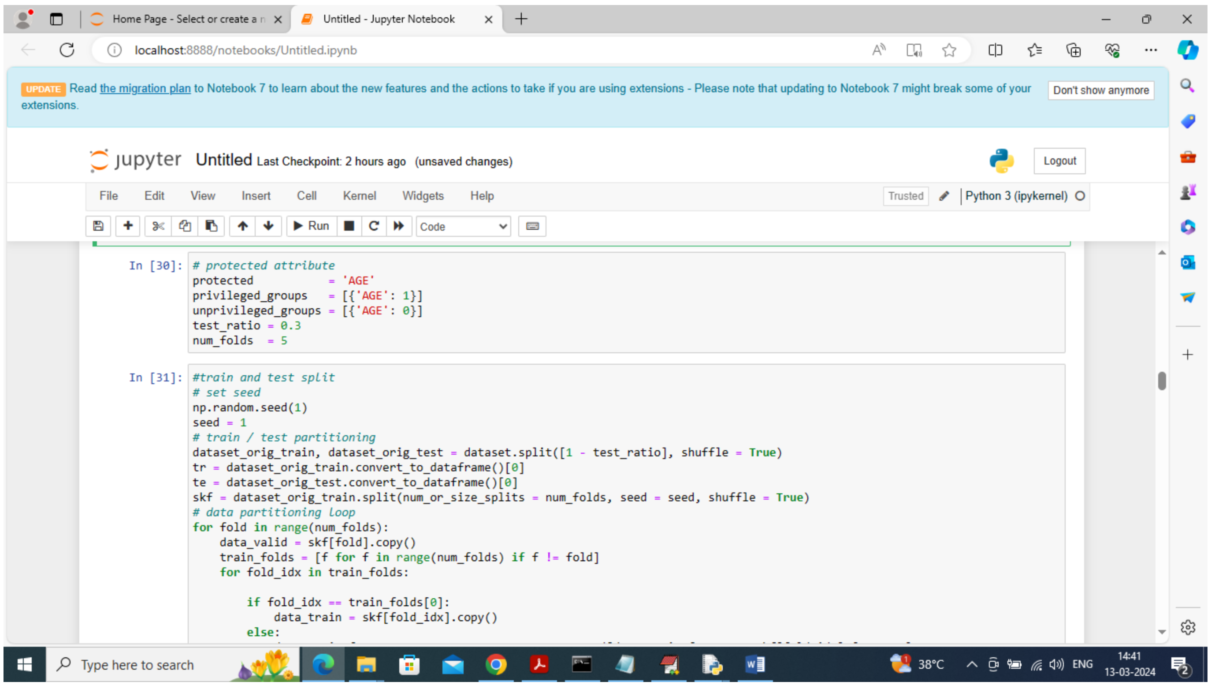


```
In [54]: import pandas as pd
         import numpy as np
         from aif360.datasets import StandardDataset
         from aif360.metrics import BinaryLabelDatasetMetric
         from aif360.algorithms.preprocessing.reweighing import Reweighing
         from aif360.algorithms.preprocessing.lfr import LFR
         from aif360.algorithms.preprocessing import DisparateImpactRemover #disparate
         from aif360.algorithms.postprocessing import RejectOptionClassification #roc class
         from aif360.algorithms.postprocessing import CalibratedEqOddsPostprocessing
         from aif360.algorithms.inprocessing.exponentiated_gradient_reduction import ExponentiatedGradientReduction
         from aif360.algorithms.inprocessing.adversarial_debiasing import AdversarialDebiasing
         from aif360.algorithms.inprocessing.grid_search_reduction import GridSearchReduction
         from aif360.algorithms.inprocessing.gerryfair_classifier import GerryFairClassifier
         import pickle
         from sklearn.model_selection import StratifiedKFold
         from sklearn.preprocessing import MaxAbsScaler
         from aif360.metrics import ClassificationMetric
         from aif360.algorithms.inprocessing.meta_fair_classifier import MetaFairClassifier
         import tensorflow as tf
         import matplotlib.pyplot as plt
         import warnings
         warnings.filterwarnings('ignore')
         %matplotlib inline
```

Importing required python classes and packages

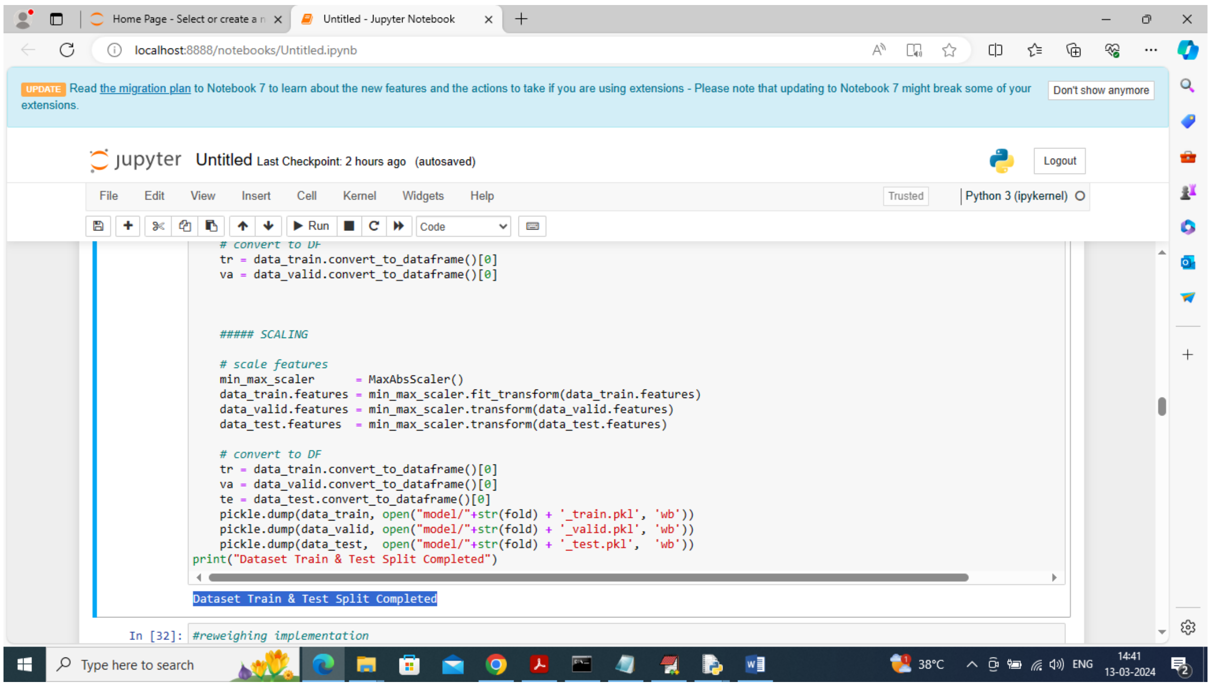


```
        metadata                  = {'label_maps':              [{1.0: 'Good', 2.0: 'Bad'}],
                                     'protected_attribute_maps': [protected_attribute_map]})

    return df_standard

In [29]: #loading and displaying dataset value
         dataset = load_german("Dataset/german.csv")
         dataset

Out[29]:                instance weights features                           \

                                         duration installment_rate resident_since
         instance names
         1                           1.0      6.0              4.0            4.0
         2                           1.0     48.0              2.0            2.0
         3                           1.0     12.0              2.0            3.0
         4                           1.0     42.0              2.0            4.0
         5                           1.0     24.0              3.0            4.0
         ...                         ...      ...              ...            ...
         996                         1.0     12.0              3.0            4.0
         997                         1.0     30.0              4.0            4.0
         998                         1.0     12.0              4.0            4.0
         999                         1.0     45.0              4.0            4.0
         1000                        1.0     45.0              3.0            4.0

                                                                  \
                                           protected attribute
```



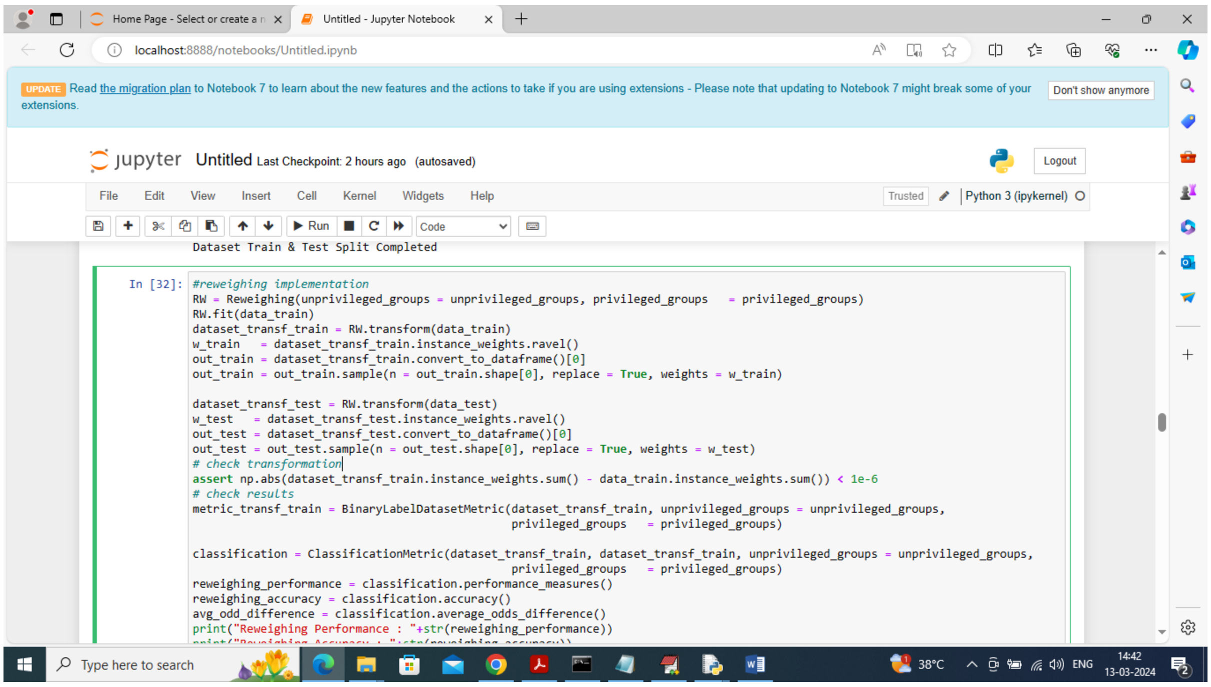
```
In [30]: # protected attribute
         protected           = 'AGE'
         privileged_groups   = [{'AGE': 1}]
         unprivileged_groups = [{'AGE': 0}]
         test_ratio = 0.3
         num_folds  = 5

In [31]: #train and test split
         # set seed
         np.random.seed(1)
         seed = 1
         # train / test partitioning
         dataset_orig_train, dataset_orig_test = dataset.split([1 - test_ratio], shuffle = True)
         tr = dataset_orig_train.convert_to_dataframe()[0]
         te = dataset_orig_test.convert_to_dataframe()[0]
         skf = dataset_orig_train.split(num_or_size_splits = num_folds, seed = seed, shuffle = True)
         # data partitioning loop
         for fold in range(num_folds):
             data_valid = skf[fold].copy()
             train_folds = [f for f in range(num_folds) if f != fold]
             for fold_idx in train_folds:

                 if fold_idx == train_folds[0]:
                     data_train = skf[fold_idx].copy()
                 else:
```

In above screen defining protected, privilege and then defining code to process dataset



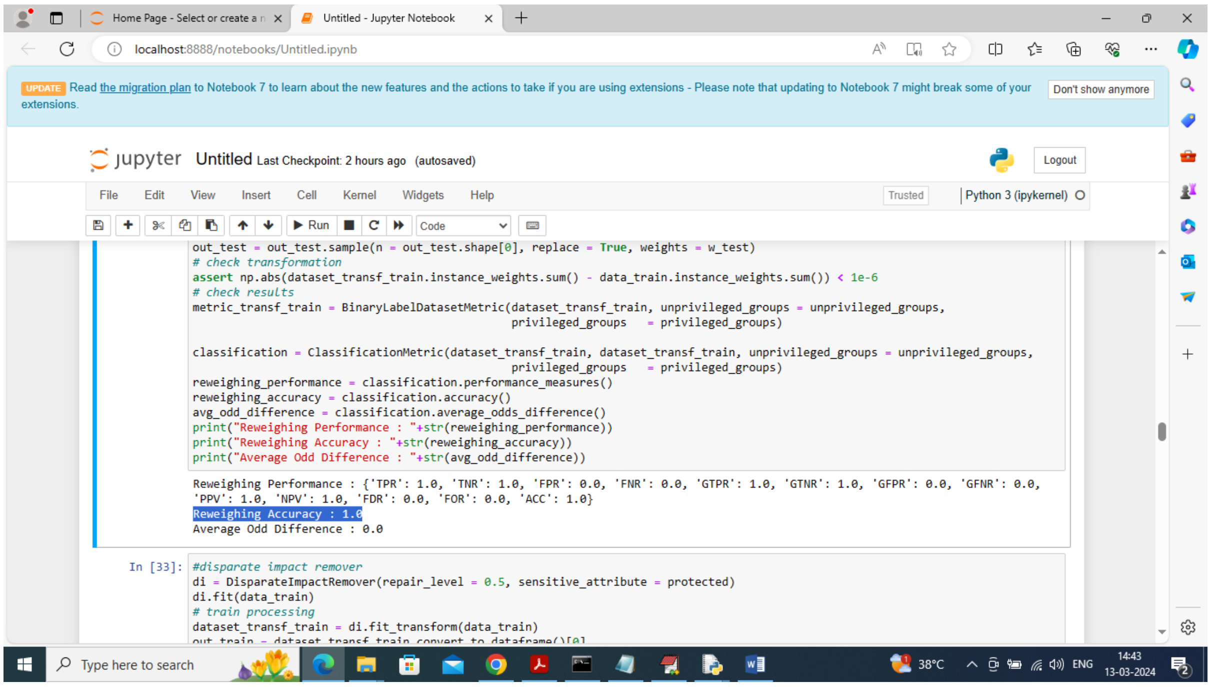
```
             # convert to DF
             tr = data_train.convert_to_dataframe()[0]
             va = data_valid.convert_to_dataframe()[0]



             ##### SCALING

             # scale features
             min_max_scaler      = MaxAbsScaler()
             data_train.features = min_max_scaler.fit_transform(data_train.features)
             data_valid.features = min_max_scaler.transform(data_valid.features)
             data_test.features  = min_max_scaler.transform(data_test.features)

             # convert to DF
             tr = data_train.convert_to_dataframe()[0]
             va = data_valid.convert_to_dataframe()[0]
             te = data_test.convert_to_dataframe()[0]
             pickle.dump(data_train, open("model/"+str(fold) + '_train.pkl', 'wb'))
             pickle.dump(data_valid, open("model/"+str(fold) + '_valid.pkl', 'wb'))
             pickle.dump(data_test,  open("model/"+str(fold) + '_test.pkl',  'wb'))
         print("Dataset Train & Test Split Completed")

         Dataset Train & Test Split Completed

In [32]: #reweighing implementation
```

In above screen dataset scaling, training and testing split completed



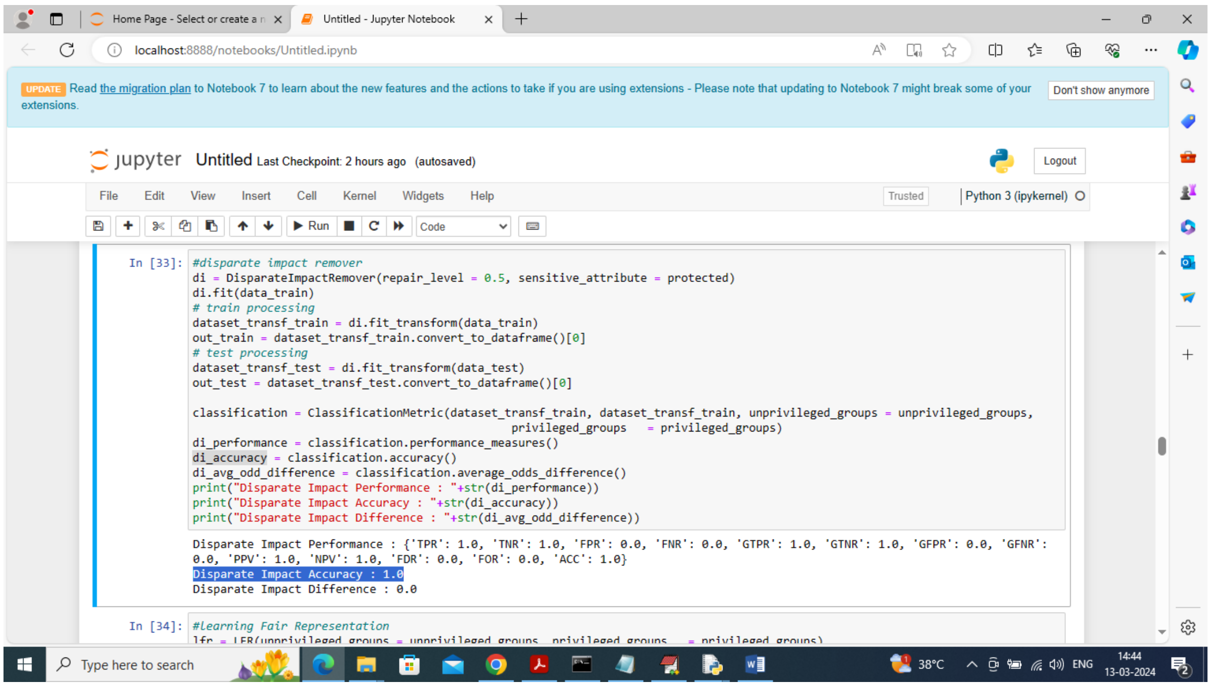
```
Dataset Train & Test Split Completed

In [32]: #reweighing implementation
         RW = Reweighing(unprivileged_groups = unprivileged_groups, privileged_groups   = privileged_groups)
         RW.fit(data_train)
         dataset_transf_train = RW.transform(data_train)
         w_train   = dataset_transf_train.instance_weights.ravel()
         out_train = dataset_transf_train.convert_to_dataframe()[0]
         out_train = out_train.sample(n = out_train.shape[0], replace = True, weights = w_train)

         dataset_transf_test = RW.transform(data_test)
         w_test    = dataset_transf_test.instance_weights.ravel()
         out_test = dataset_transf_test.convert_to_dataframe()[0]
         out_test = out_test.sample(n = out_test.shape[0], replace = True, weights = w_test)
         # check transformation
         assert np.abs(dataset_transf_train.instance_weights.sum() - data_train.instance_weights.sum()) < 1e-6
         # check results
         metric_transf_train = BinaryLabelDatasetMetric(dataset_transf_train, unprivileged_groups = unprivileged_groups,
                                                        privileged_groups   = privileged_groups)

         classification = ClassificationMetric(dataset_transf_train, dataset_transf_train, unprivileged_groups = unprivileged_groups,
                                                        privileged_groups   = privileged_groups)
         reweighing_performance = classification.performance_measures()
         reweighing_accuracy = classification.accuracy()
         avg_odd_difference = classification.average_odds_difference()
         print("Reweighing Performance : "+str(reweighing_performance))
```

In above screen implementing Reweighing method to process dataset using Reweighing technique and then calculate accuracy



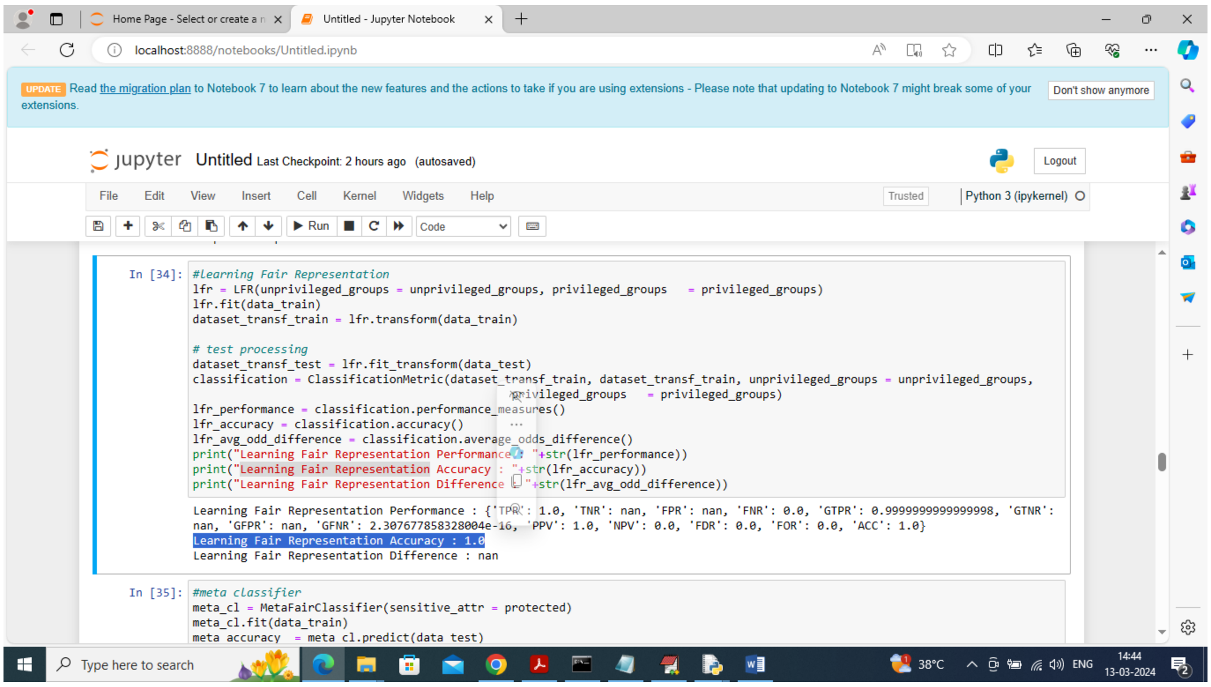
```
         out_test = out_test.sample(n = out_test.shape[0], replace = True, weights = w_test)
         # check transformation
         assert np.abs(dataset_transf_train.instance_weights.sum() - data_train.instance_weights.sum()) < 1e-6
         # check results
         metric_transf_train = BinaryLabelDatasetMetric(dataset_transf_train, unprivileged_groups = unprivileged_groups,
                                                        privileged_groups   = privileged_groups)

         classification = ClassificationMetric(dataset_transf_train, dataset_transf_train, unprivileged_groups = unprivileged_groups,
                                                        privileged_groups   = privileged_groups)
         reweighing_performance = classification.performance_measures()
         reweighing_accuracy = classification.accuracy()
         avg_odd_difference = classification.average_odds_difference()
         print("Reweighing Performance : "+str(reweighing_performance))
         print("Reweighing Accuracy : "+str(reweighing_accuracy))
         print("Average Odd Difference : "+str(avg_odd_difference))

         Reweighing Performance : {'TPR': 1.0, 'TNR': 1.0, 'FPR': 0.0, 'FNR': 0.0, 'GTPR': 1.0, 'GTNR': 1.0, 'GFPR': 0.0, 'GFNR': 0.0,
         'PPV': 1.0, 'NPV': 1.0, 'FDR': 0.0, 'FOR': 0.0, 'ACC': 1.0}
         Reweighing Accuracy : 1.0
         Average Odd Difference : 0.0

In [33]: #disparate impact remover
         di = DisparateImpactRemover(repair_level = 0.5, sensitive_attribute = protected)
         di.fit(data_train)
         # train processing
         dataset_transf_train = di.fit_transform(data_train)
```

In above screen Reweighing technique got 100% accuracy and can see other metrics like Average ODD Difference and TPR values



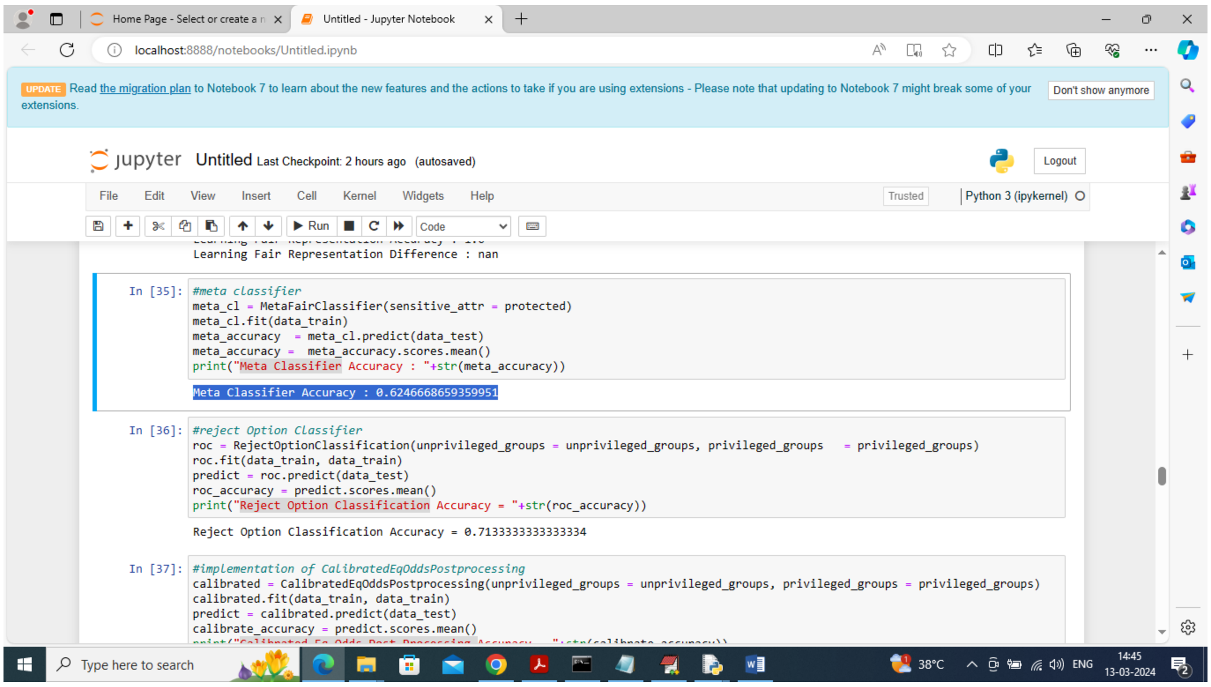
```
In [33]: #disparate impact remover
         di = DisparateImpactRemover(repair_level = 0.5, sensitive_attribute = protected)
         di.fit(data_train)
         # train processing
         dataset_transf_train = di.fit_transform(data_train)
         out_train = dataset_transf_train.convert_to_dataframe()[0]
         # test processing
         dataset_transf_test = di.fit_transform(data_test)
         out_test = dataset_transf_test.convert_to_dataframe()[0]

         classification = ClassificationMetric(dataset_transf_train, dataset_transf_train, unprivileged_groups = unprivileged_groups,
                                               privileged_groups    = privileged_groups)
         di_performance = classification.performance_measures()
         di_accuracy = classification.accuracy()
         di_avg_odd_difference = classification.average_odds_difference()
         print("Disparate Impact Performance : "+str(di_performance))
         print("Disparate Impact Accuracy : "+str(di_accuracy))
         print("Disparate Impact Difference : "+str(di_avg_odd_difference))

         Disparate Impact Performance : {'TPR': 1.0, 'TNR': 1.0, 'FPR': 0.0, 'FNR': 0.0, 'GTPR': 1.0, 'GTNR': 1.0, 'GFPR': 0.0, 'GFNR':
         0.0, 'PPV': 1.0, 'NPV': 1.0, 'FDR': 0.0, 'FOR': 0.0, 'ACC': 1.0}
         Disparate Impact Accuracy : 1.0
         Disparate Impact Difference : 0.0

In [34]: #Learning Fair Representation
```

In above screen 'Disparate Impact Remover' also got 100% accuracy



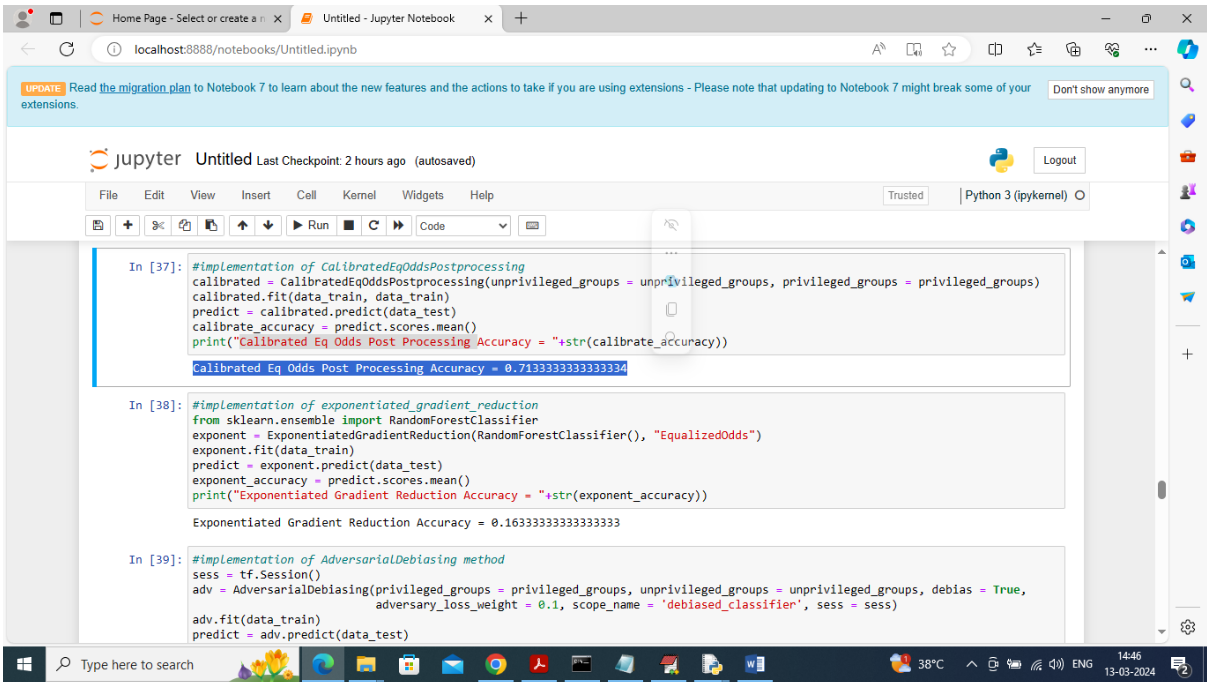
```
In [34]: #Learning Fair Representation
         lfr = LFR(unprivileged_groups = unprivileged_groups, privileged_groups    = privileged_groups)
         lfr.fit(data_train)
         dataset_transf_train = lfr.transform(data_train)

         # test processing
         dataset_transf_test = lfr.fit_transform(data_test)
         classification = ClassificationMetric(dataset_transf_train, dataset_transf_train, unprivileged_groups = unprivileged_groups,
                                               privileged_groups    = privileged_groups)
         lfr_performance = classification.performance_measures()
         lfr_accuracy = classification.accuracy()
         lfr_avg_odd_difference = classification.average_odds_difference()
         print("Learning Fair Representation Performance : "+str(lfr_performance))
         print("Learning Fair Representation Accuracy : "+str(lfr_accuracy))
         print("Learning Fair Representation Difference : "+str(lfr_avg_odd_difference))

         Learning Fair Representation Performance : {'TPR': 1.0, 'TNR': nan, 'FPR': nan, 'FNR': 0.0, 'GTPR': 0.9999999999999998, 'GTNR':
         nan, 'GFPR': nan, 'GFNR': 2.307677858328004e-16, 'PPV': 1.0, 'NPV': 0.0, 'FDR': 0.0, 'FOR': 0.0, 'ACC': 1.0}
         Learning Fair Representation Accuracy : 1.0
         Learning Fair Representation Difference : nan

In [35]: #meta classifier
         meta_cl = MetaFairClassifier(sensitive_attr = protected)
         meta_cl.fit(data_train)
         meta_accuracy  = meta_cl.predict(data_test)
```

In above screen implementing 'Linear Fair Representation' technique and it got 100% accuracy



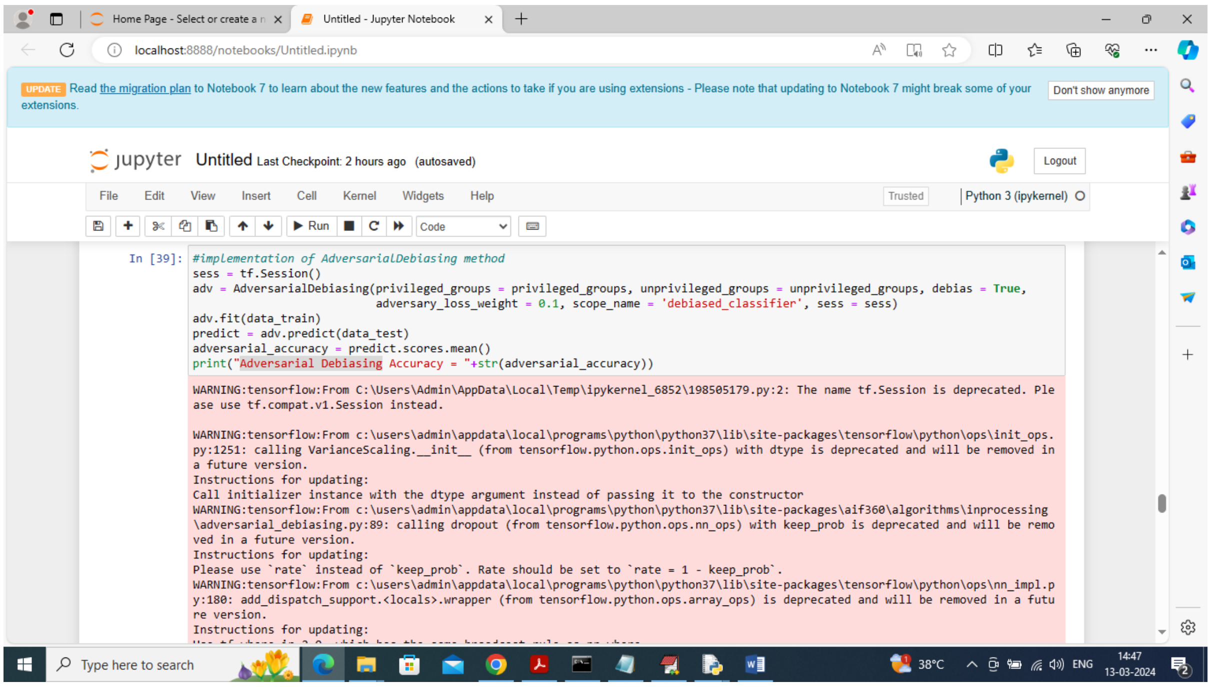
```
Learning Fair Representation Difference : nan

In [35]: #meta classifier
         meta_cl = MetaFairClassifier(sensitive_attr = protected)
         meta_cl.fit(data_train)
         meta_accuracy  = meta_cl.predict(data_test)
         meta_accuracy =  meta_accuracy.scores.mean()
         print("Meta Classifier Accuracy : "+str(meta_accuracy))

         Meta Classifier Accuracy : 0.6246668659359951

In [36]: #reject Option Classifier
         roc = RejectOptionClassification(unprivileged_groups = unprivileged_groups, privileged_groups   = privileged_groups)
         roc.fit(data_train, data_train)
         predict = roc.predict(data_test)
         roc_accuracy = predict.scores.mean()
         print("Reject Option Classification Accuracy = "+str(roc_accuracy))

         Reject Option Classification Accuracy = 0.7133333333333334

In [37]: #implementation of CalibratedEqOddsPostprocessing
         calibrated = CalibratedEqOddsPostprocessing(unprivileged_groups = unprivileged_groups, privileged_groups = privileged_groups)
         calibrated.fit(data_train, data_train)
         predict = calibrated.predict(data_test)
         calibrate_accuracy = predict.scores.mean()
```

In above screen can see Meta Classifier and Reject Option Classification where meta classifier got 61% accuracy and ROC got 71% accuracy



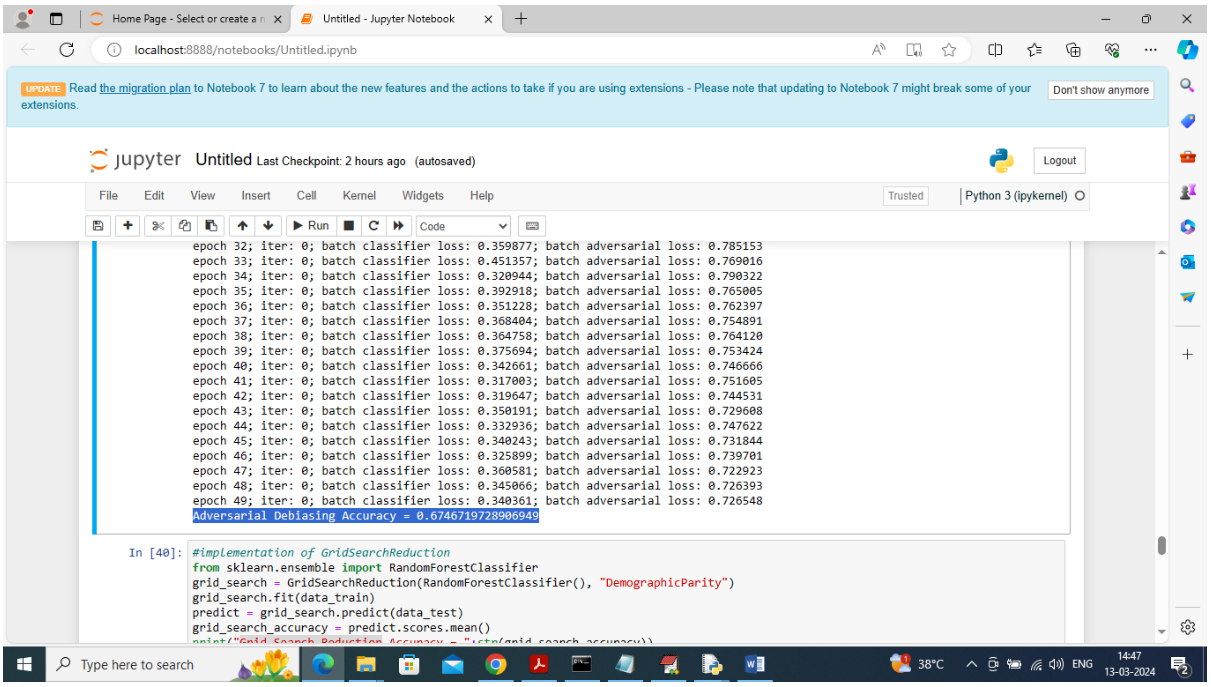
```
In [37]: #implementation of CalibratedEqOddsPostprocessing
         calibrated = CalibratedEqOddsPostprocessing(unprivileged_groups = unprivileged_groups, privileged_groups = privileged_groups)
         calibrated.fit(data_train, data_train)
         predict = calibrated.predict(data_test)
         calibrate_accuracy = predict.scores.mean()
         print("Calibrated Eq Odds Post Processing Accuracy = "+str(calibrate_accuracy))

         Calibrated Eq Odds Post Processing Accuracy = 0.7133333333333334

In [38]: #implementation of exponentiated_gradient_reduction
         from sklearn.ensemble import RandomForestClassifier
         exponent = ExponentiatedGradientReduction(RandomForestClassifier(), "EqualizedOdds")
         exponent.fit(data_train)
         predict = exponent.predict(data_test)
         exponent_accuracy = predict.scores.mean()
         print("Exponentiated Gradient Reduction Accuracy = "+str(exponent_accuracy))

         Exponentiated Gradient Reduction Accuracy = 0.16333333333333333

In [39]: #implementation of AdversarialDebiasing method
         sess = tf.Session()
         adv = AdversarialDebiasing(privileged_groups = privileged_groups, unprivileged_groups = unprivileged_groups, debias = True,
                                    adversary_loss_weight = 0.1, scope_name = 'debiased_classifier', sess = sess)
         adv.fit(data_train)
         predict = adv.predict(data_test)
```

In above screen can see implementation of Calibrated and Exponential Gradient implementation where first one got 71% accuracy and other got 16% accuracy



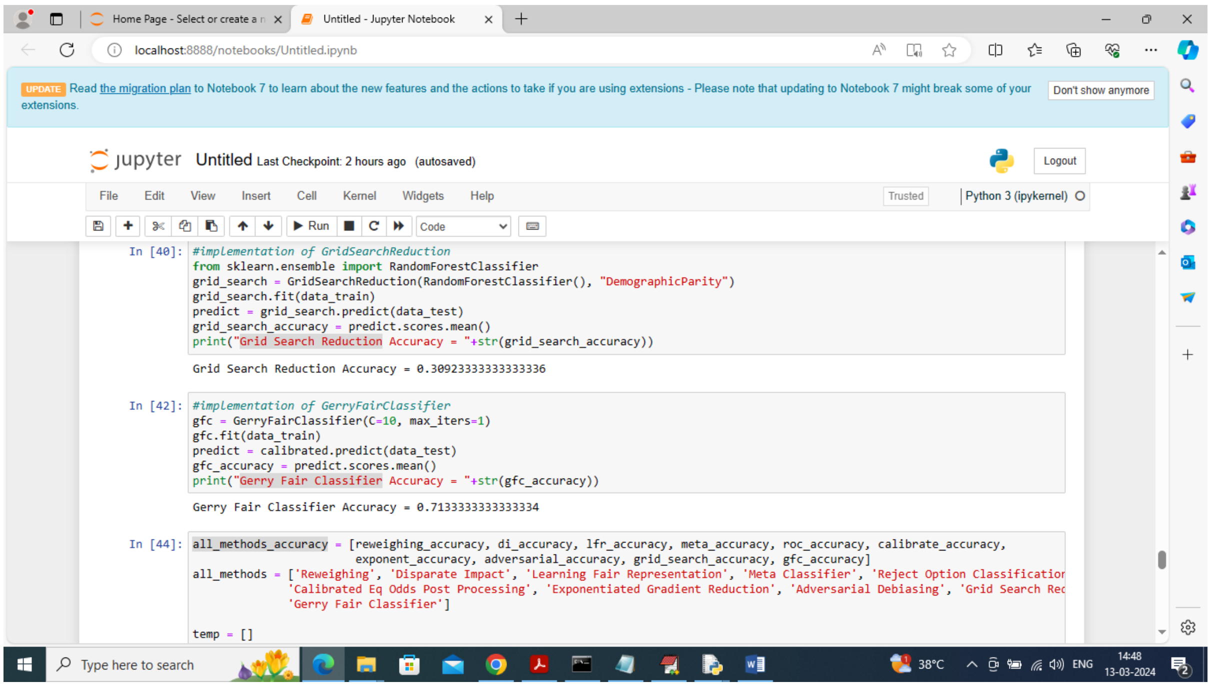
```
In [39]: #implementation of AdversarialDebiasing method
         sess = tf.Session()
         adv = AdversarialDebiasing(privileged_groups = privileged_groups, unprivileged_groups = unprivileged_groups, debias = True,
                                    adversary_loss_weight = 0.1, scope_name = 'debiased_classifier', sess = sess)
         adv.fit(data_train)
         predict = adv.predict(data_test)
         adversarial_accuracy = predict.scores.mean()
         print("Adversarial Debiasing Accuracy = "+str(adversarial_accuracy))

         WARNING:tensorflow:From C:\Users\Admin\AppData\Local\Temp\ipykernel_6852\198505179.py:2: The name tf.Session is deprecated. Ple
         ase use tf.compat.v1.Session instead.

         WARNING:tensorflow:From c:\users\admin\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\ops\init_ops.
         py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.init_ops) with dtype is deprecated and will be removed in
         a future version.
         Instructions for updating:
         Call initializer instance with the dtype argument instead of passing it to the constructor
         WARNING:tensorflow:From c:\users\admin\appdata\local\programs\python\python37\lib\site-packages\aif360\algorithms\inprocessing
         \adversarial_debiasing.py:89: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and will be remo
         ved in a future version.
         Instructions for updating:
         Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
         WARNING:tensorflow:From c:\users\admin\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\ops\nn_impl.p
         y:180: add_dispatch_support.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a futu
         re version.
         Instructions for updating:
```

In above screen implementing Adversarial Debiasing technique and below is the output



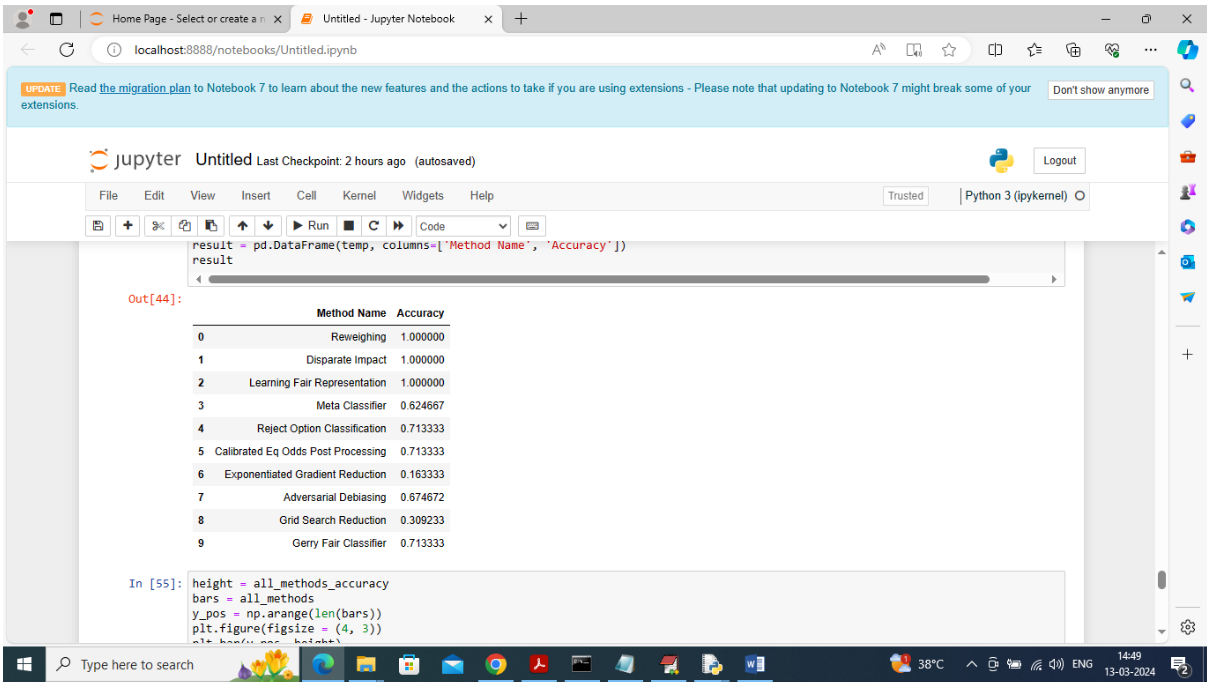
```
         epoch 32; iter: 0; batch classifier loss: 0.359877; batch adversarial loss: 0.785153
         epoch 33; iter: 0; batch classifier loss: 0.451357; batch adversarial loss: 0.769016
         epoch 34; iter: 0; batch classifier loss: 0.320944; batch adversarial loss: 0.790322
         epoch 35; iter: 0; batch classifier loss: 0.392918; batch adversarial loss: 0.765005
         epoch 36; iter: 0; batch classifier loss: 0.351228; batch adversarial loss: 0.762397
         epoch 37; iter: 0; batch classifier loss: 0.368404; batch adversarial loss: 0.754891
         epoch 38; iter: 0; batch classifier loss: 0.364758; batch adversarial loss: 0.764120
         epoch 39; iter: 0; batch classifier loss: 0.375694; batch adversarial loss: 0.753424
         epoch 40; iter: 0; batch classifier loss: 0.342661; batch adversarial loss: 0.746666
         epoch 41; iter: 0; batch classifier loss: 0.317003; batch adversarial loss: 0.751605
         epoch 42; iter: 0; batch classifier loss: 0.319647; batch adversarial loss: 0.744531
         epoch 43; iter: 0; batch classifier loss: 0.350191; batch adversarial loss: 0.729608
         epoch 44; iter: 0; batch classifier loss: 0.332936; batch adversarial loss: 0.747622
         epoch 45; iter: 0; batch classifier loss: 0.340243; batch adversarial loss: 0.731844
         epoch 46; iter: 0; batch classifier loss: 0.325899; batch adversarial loss: 0.739701
         epoch 47; iter: 0; batch classifier loss: 0.360581; batch adversarial loss: 0.722923
         epoch 48; iter: 0; batch classifier loss: 0.345066; batch adversarial loss: 0.726393
         epoch 49; iter: 0; batch classifier loss: 0.340361; batch adversarial loss: 0.726548
         Adversarial Debiasing Accuracy = 0.6746719728906949

In [40]: #implementation of GridSearchReduction
         from sklearn.ensemble import RandomForestClassifier
         grid_search = GridSearchReduction(RandomForestClassifier(), "DemographicParity")
         grid_search.fit(data_train)
         predict = grid_search.predict(data_test)
         grid_search_accuracy = predict.scores.mean()
```

In above screen adversarial debiasing got 67% accuracy



```
In [40]: #implementation of GridSearchReduction
         from sklearn.ensemble import RandomForestClassifier
         grid_search = GridSearchReduction(RandomForestClassifier(), "DemographicParity")
         grid_search.fit(data_train)
         predict = grid_search.predict(data_test)
         grid_search_accuracy = predict.scores.mean()
         print("Grid Search Reduction Accuracy = "+str(grid_search_accuracy))

         Grid Search Reduction Accuracy = 0.30923333333333336

In [42]: #implementation of GerryFairClassifier
         gfc = GerryFairClassifier(C=10, max_iters=1)
         gfc.fit(data_train)
         predict = calibrated.predict(data_test)
         gfc_accuracy = predict.scores.mean()
         print("Gerry Fair Classifier Accuracy = "+str(gfc_accuracy))

         Gerry Fair Classifier Accuracy = 0.7133333333333334

In [44]: all_methods_accuracy = [reweighing_accuracy, di_accuracy, lfr_accuracy, meta_accuracy, roc_accuracy, calibrate_accuracy,
                                 exponent_accuracy, adversarial_accuracy, grid_search_accuracy, gfc_accuracy]
         all_methods = ['Reweighing', 'Disparate Impact', 'Learning Fair Representation', 'Meta Classifier', 'Reject Option Classificatior
                        'Calibrated Eq Odds Post Processing', 'Exponentiated Gradient Reduction', 'Adversarial Debiasing', 'Grid Search Rec
                        'Gerry Fair Classifier']

         temp = []
```

In above screen can see accuracy of Grid Search and Gerry Fair algorithm

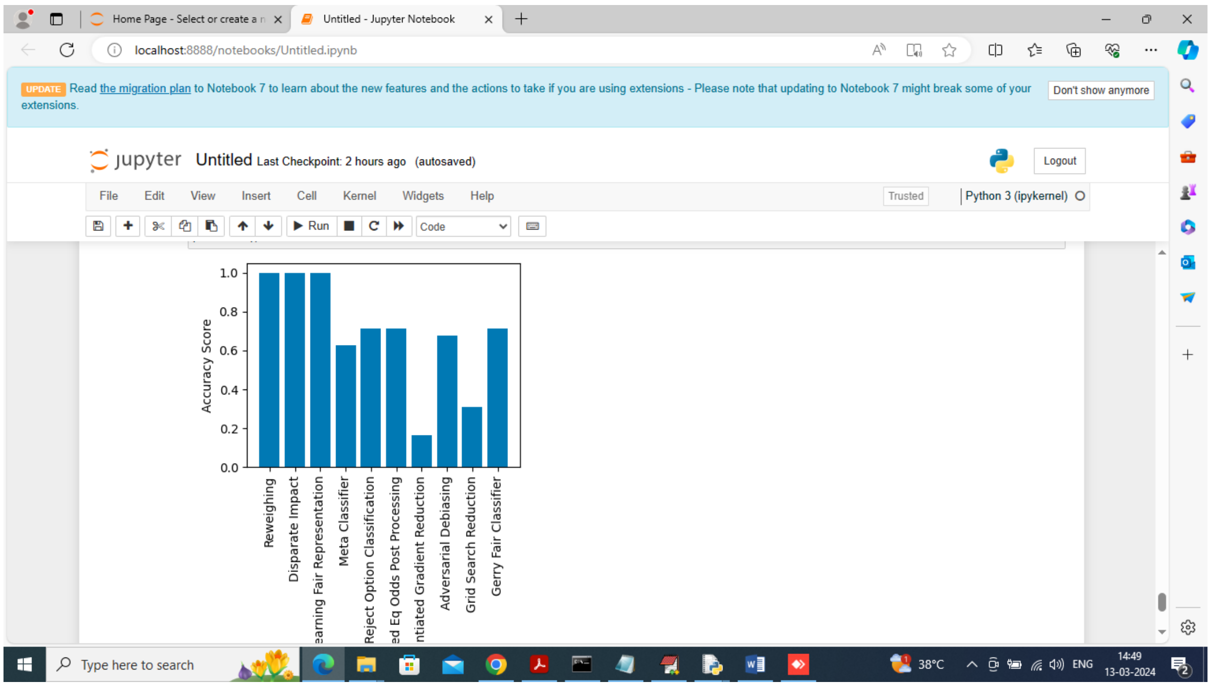


```
         result = pd.DataFrame(temp, columns=['Method Name', 'Accuracy'])
         result

Out[44]:
```

	Method Name	Accuracy
0	Reweighing	1.000000
1	Disparate Impact	1.000000
2	Learning Fair Representation	1.000000
3	Meta Classifier	0.624667
4	Reject Option Classification	0.713333
5	Calibrated Eq Odds Post Processing	0.713333
6	Exponentiated Gradient Reduction	0.163333
7	Adversarial Debiasing	0.674672
8	Grid Search Reduction	0.309233
9	Gerry Fair Classifier	0.713333

```
In [55]: height = all_methods_accuracy
         bars = all_methods
         y_pos = np.arange(len(bars))
         plt.figure(figsize = (4, 3))
```

In above screen can see implementation accuracy of each method in tabular format


Home Page - Select or create a n
Untitled - Jupyter Notebook
localhost:8888/notebooks/Untitled.ipynb
UPDATE Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
Don't show anymore
jupyter Untitled Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Accuracy Score
1.0
0.8
0.6
0.4
0.2
0.0
Reweighing
Disparate Impact
earning Fair Representation
Meta Classifier
Reject Option Classification
ed Eq Odds Post Processing
ntiated Gradient Reduction
Adversarial Debiasing
Grid Search Reduction
Gerry Fair Classifier
Type here to search
38°C
ENG
14:49
13-03-2024

In above screen can see each method performance output in graph format and in above methods credit company can see which methods or how to data to be process in order to predict credit approval status and to increase profit.

CHAPTER-VII

CONCLUSION AND FUTURE SCOPE

In conclusion, the implementation of Algorithmic Decision-Making Methods for Fair Credit Scoring represents a significant advancement in the pursuit of equitable financial services. By leveraging machine learning and data-driven techniques, these methods can enhance the accuracy of credit assessments while addressing inherent biases that have historically plagued traditional scoring systems.

Key takeaways include:

Enhanced Fairness: Through the application of fairness metrics and bias mitigation strategies, these methods can ensure that credit scoring processes do not discriminate against any demographic group. This fosters greater inclusivity in access to credit. The integration of fairness metrics and bias mitigation strategies in credit scoring processes is essential for fostering inclusivity and equity in access to credit. By prioritizing these elements, financial institutions can dismantle historical barriers, empower individuals from all backgrounds, and promote a fairer economic landscape. This commitment to enhanced fairness not only aligns with ethical imperatives but also enhances the overall stability and resilience of financial systems.

Transparency and Explainability: Incorporating explainable AI techniques allows stakeholders to understand how credit decisions are made, increasing trust in the system. Clear explanations empower applicants to grasp the factors influencing their scores.Transparency and explainability are fundamental components of Algorithmic Decision Making in credit scoring. By utilizing explainable AI techniques, financial institutions can provide stakeholders with clear insights into how credit decisions are made, ultimately fostering trust and accountability.

This empowerment of applicants not only enhances their understanding of credit processes but also promotes financial literacy and responsible behaviour. As the landscape of credit scoring continues to evolve, prioritizing transparency and explainability will be essential in building a fairer, more inclusive financial system that benefits all participant.

Regulatory Compliance: By aligning with legal frameworks and ethical guidelines, the proposed systems ensure adherence to regulations such as the Fair Credit Reporting Act (FCRA) and GDPR, protecting consumer rights and data privacy. Regulatory compliance is fundamental to the success of Algorithmic Decision-Making Methods in credit scoring. By aligning with frameworks such as the FCRA and GDPR, financial institutions can protect consumer rights and data privacy, fostering trust and accountability in their practices. Continuous Improvement: The focus on ongoing monitoring and feedback mechanisms ensures that models remain relevant and fair over time, adapting to changes in data and societal norms. Continuous improvement is fundamental to the success of Algorithmic Decision Making in credit scoring. By establishing robust monitoring frameworks, integrating stakeholder feedback, and maintaining adaptability in model development, financial institutions can ensure that their credit scoring practices remain relevant, fair, and effective over time. This commitment to continuous improvement not only enhances the accuracy of credit assessments but also promotes equity and trust in the financial system, ultimately contributing to a more inclusive economic environment.

Stakeholder Engagement: Involving a diverse range of stakeholders throughout the development and implementation process fosters trust and accountability, reinforcing the commitment to fairness in credit scoring. Stakeholder engagement is fundamental to the development and implementation of fair and effective credit scoring systems. By actively involving a diverse range of stakeholders, financial institutions can foster trust, enhance accountability, and ensure that credit scoring practices are inclusive and responsive to the needs of all consumers. This collaborative approach not only leads to more robust credit scoring models but also reinforces a commitment to ethical standards and fairness, ultimately contributing to a more equitable financial landscape. Through continuous dialogue and collaboration, stakeholders can work together to create credit scoring systems that reflect societal values and promote access to credit for all individuals.

Ultimately, the adoption of Algorithmic Decision-Making Methods for Fair Credit Scoring can lead to a more just financial landscape, where access to credit is determined by equitable criteria rather than biased historical data. This not only benefits individual consumers but also contributes to a healthier, more inclusive economy. By prioritizing fairness, transparency, and ethical considerations, we can create credit scoring systems that empower all individuals and promote financial equity.

CHAPTER-VIII

REFERENCES

Barocas, S., Hardt, M., & Narayanan, A. (2019). Fairness and Machine Learning: Limitations and Opportunities.

This paper discusses the challenges and approaches to achieving fairness in machine learning models, including credit scoring

Zliobaite, I. (2015). Demographic Discrimination in Machine Learning.

This paper explores how demographic factors can lead to discrimination in machine learning applications, including credit scoring.

Kleinberg, J., Mullainathan, S., & Obermeyer, Z. (2016). Inherent Trade-Offs in the Fair Determination of Risk Scores.

This study examines the trade-offs between fairness and accuracy in risk assessment algorithms.

Federal Trade Commission (FTC) (2016). Big Data: A Tool for Inclusion or Exclusion?

This report analyzes the implications of big data in credit scoring and its potential for both inclusion and discrimination.

Financial Protection Bureau (CFPB) (2022). Fair Lending Report.

This report includes insights into fair lending practices and the role of credit scoring in ensuring equitable access to credit.

FICO (2020). The Future of Credit Scoring: Why It Matters and What to Expect.

This report discusses advancements in credit scoring models and emphasizes the importance of fairness and transparency.

McKinsey & Company (2021). Artificial Intelligence in Financial Services: The Next Frontier for Credit Scoring.

This report explores the use of AI in credit scoring and discusses how to maintain fairness and compliance with regulations.

O'Neil, C. (2016). Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy.

This book critiques algorithmic decision-making processes and highlights the risks of bias in systems like credit scoring.

Eubanks, V. (2018). Automating Inequality: How High-Tech Tools Profile, Police, and Punish the Poor.