



## MASTERS THESIS

---

### Explaining the Fairness of Deep Neural Networks using Large Language Models

---

*Student:*  
Nalanda Chakrabarti

*First reviewer:*  
Dr. Stefan Heindorf  
*Second reviewer:*  
AOR. Dr. Ilona Horwath

**Department of Computer Science**  
January 5, 2024

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Related Work</b>	<b>5</b>
<b>3</b>	<b>Background</b>	<b>9</b>
3.1	Concepts . . . . .	10
3.1.1	Machine Learning . . . . .	10
3.1.2	LLM(Large language Model) . . . . .	12
3.1.3	Conversational AI . . . . .	14
3.1.4	Neural Network . . . . .	16
3.1.5	Explainability . . . . .	19
3.1.6	Fairness . . . . .	23
3.2	Toolkits . . . . .	27
3.2.1	Streamlit . . . . .	27
3.2.2	GPT-4 . . . . .	28
3.2.3	Langchain . . . . .	31
3.2.4	SHAP . . . . .	34
3.2.5	Fairlearn . . . . .	36
<b>4</b>	<b>Approach</b>	<b>40</b>
4.1	Problem . . . . .	40
4.2	Solution . . . . .	42
4.2.1	Requirements for Solution . . . . .	42
4.2.2	Approach of Solution . . . . .	44
4.2.3	Implementation details . . . . .	54
<b>5</b>	<b>Evaluation</b>	<b>64</b>
5.1	Gender Bias Dataset . . . . .	69
5.1.1	Total Salary Predictions Q/A : . . . . .	69
5.1.2	Explanation of Predictions(SHAP) Q/A . . . . .	75
5.1.3	Fairness of Predictions(Fairlearn) Q/A . . . . .	79
5.1.4	Unfairness Mitigation Q/A . . . . .	81
5.1.5	Single Prediction Q/A . . . . .	82
5.2	Uneven Gender Distribution Dataset . . . . .	84
5.2.1	Total Salary Predictions Q/A . . . . .	84
5.2.2	Explanation of Predictions(SHAP) Q/A . . . . .	87

5.2.3	Fairness of Predictions(Fairlearn) Q/A . . . . .	88
5.2.4	Unfairness Mitigation Q/A . . . . .	89
5.2.5	User Feedback Evaluation . . . . .	92
<b>6</b>	<b>Discussion</b>	<b>95</b>
6.1	Challenges . . . . .	95
6.2	Future Work . . . . .	96
<b>7</b>	<b>Conclusion</b>	<b>97</b>
7.1	Appendix . . . . .	98

## 1 Introduction

'In God we trust; all others must bring data.' W. Edwards Deming. The statistician cum professor has justly put most of the population's thoughts in this quote, apart from atheists and agnostics. They would also want data in favour of God. Understandably, the whole world revolves around data nowadays, and the most efficient way to put this data to use is machine learning. Since the last century, ML has seen a drastic rise in all sectors of life and has become an inherent part of our day-to-day activities. This has also brought about the question of whether ML is, after all, trustworthy at all. In crucial fields like the judgement system and the medical department, ML algorithms are being questioned about their decisions. *In 2013, Loomis was sentenced to prison, the duration largely based on the outputs of an algorithm. He had pleaded guilty to knowingly fleeing a police officer, and at the time of sentencing, the judge consulted an algorithm. The algorithm predicted that Loomis was at 'high risk' of recidivating. Accordingly, the judge imposed a harsh six-year sentence. Loomis, however, sought clarity. He challenged the decision, seeking to understand the basis for the algorithm's determination. His petition was denied, and he was convicted without explanation.*<sup>1</sup>.

Many such devastating examples of an opaque system and its shortcomings are being observed in society in an incremental manner. This brings about the need to explain or interpret this opaque system, or, technically, the black box. "*The need for interpretability arises from an incompleteness in problem formalisation (Doshi-Velez and Kim 2017)*"[27]. In [16], it has been explained how any problem formalisation can be proven incomplete, lacking the criteria for fairness, privacy, reliability, robustness, causality, usability, and trust. It is logically not possible to consider all the weak points of the system in order to come up with an optimal solution. For example, in the above example, a trade-off has been made with the fairness criteria (unbiasedness), which imply that protected groups (explicit or implicit) are not somehow discriminated against[16], which has made the algorithm formalisation incomplete, introducing an unquantified bias. Ethics: The human may want to guard against certain kinds of discrimination, and their notion of fairness may be too abstract to be completely encoded into the system (e.g., one might desire a 'fair' classifier for loan approval). Even if we can encode protections for specific protected classes into the system, there might

---

<sup>1</sup><https://stanfordrewired.com/post/rational-explanations>

be biases that we did not consider a priori (e.g., one may not build gender-biased word embeddings on purpose, but it was a pattern in data that became apparent only after the fact).[16] Similarly, in the case of recidivism in the above example, a bias has entered the algorithm and made it incomplete pertaining to the fairness criteria.

Interpretability can assist in qualitatively ascertaining whether the above criteria are met and will help us understand fair or unfair decisions made by the system and take relevant actions. In order to understand these explanations better, conversational AI systems can be most beneficial, as they can talk to a layman and assist him/her in taking decisions based on meaningful explanations. Conversational AI systems can also assist a blind person with auditory sensors included in it. Often, the results or the working of the AI systems are difficult to understand in simple terms just by looking at plots or graphs or any other kind of data visualization. Conversational AI systems can simplify the explanations and make them easier to understand for any common person even who has minimum domain knowledge on that subject.

In my thesis, the problem of interpreting Fairness metrics and Explanation values of a ML model working with regression data using Conversational AI have been dealt with. A ML model predicting salary has been implemented keeping modularity in mind which predicts salary given certain credentials. The datasets are biased and the solution efficiently find out the bias in the dataset using a chatbot with relevant context.

## 2 Related Work

Explainable AI systems have been on the rise recently, especially with the introduction of conversational AI systems, or intelligent chatbots. The words “*The goal is to turn data into information, and information into insight*” by former HP president Carly Fiorina have actually started becoming reality. A considerable amount of work has been done towards explaining classical ML models to Deep Neural Networks in order to extract meaningful insights on their decisions. These insights or explanations are useful to data scientists or even non expert end users in order to trust the ML model and accept its decisions disregard it. “*Interpretability helps the developer to debug and improve the model, build trust in the model, justify model predictions and gain insights.*”[27] It is a useful debugging tool for detecting bias in machine learning models. For a machine to interact with

us, it may need to shape our emotions and beliefs. Usually, humans are pleased with explanations which confirm their belief system and disregard logical insights in the process. “*Humans tend to ignore information that is inconsistent with their prior beliefs. This effect is called confirmation bias (Nickerson 199813).*”[27] That is why, “*Normative explanations are important because they force us to question the criteria for a decision.*”<sup>2</sup>. A very important point to be made here is that, the more complex a ML model gets, the harder it becomes to explain it as shown in Figure 1. Thus, linear regression models or simple decision trees are intrinsically explainable , but neural networks are not.

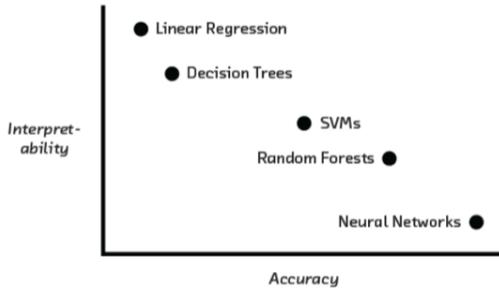


Figure 1: Interpretability is inversely proportional to Algorithm complexity <sup>4</sup>

architecture).[31] It reduces the performance gap between the traditionally used GBDT(Gradient Boosted Decision Tree) on tabular data and normal Deep Learning solutions on tabular data. Decision Trees are intrinsically explainable ML models for decision making. GBDTs are decision tree ensembles specialized for tabular data having high accuracy. NODE can be preferred more as it generalizes CatBoost making the splitting feature choice and decision tree routing differentiable. It also has higher performance rate than GBDTs. Furthermore, NODE allows constructing multi-layer architectures, which resembles ”deep” GBDT that is trained end-to-end, which was **never proposed before**. Basically, it consists of layer wise structure like neural

However, ensemble models have helped bridge the gap between black box and intrinsically explainable models. The research papers [46], <sup>5</sup>, <sup>6</sup> explain how heterogeneous ensemble models containing both black box and white box can give more accurate results. Two or more types of models are intertwined to make a more robust complex model. One such model architecture is Neural Oblivious Decision Ensembles for Deep Learning on Tabular Data (NODE

<sup>2</sup><https://stanfordrewired.com/post/rational-explanations>

<sup>5</sup><https://www.linkedin.com/pulse/ensemble-learning-combining-multiple-models-better-mithilesh-shirsat>

<sup>6</sup><https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9147256/>

network where differentiable oblivious decision trees (ODT) are trained end-to-end by backpropagation. This architecture has also been added to the PyTorch Tabular library [23] which can be easily used for this thesis.

There are many methods to bring out interpretations. Most popular are the post hoc ones as they can be modular in nature and can be applied to any ML model. Some examples are Shap and LIME libraries. Other methods like Feature Visualization, Saliency Maps, Partial Dependence Plots have been used to explain Deep Neural Networks. Partial Dependence Plot (PDP) is a manner to visualize the relationship between different neurons in the feature space. In certain papers, extension of PDP was applied on the tabular dataset which was able to evaluate and visualize the interaction between the neurons and the relationship between the feature and prediction.[34] There is another explainability method called Counterfactual explanations. Its goal is to guide an applicant on how they can change the outcome of a model by providing suggestions for improvement. Given a specific input value (e.g., a data point that is declined by a model), counterfactual explanations attempt to find another input value for which the model would provide a different outcome (essentially get accepted). Such an input value that changes the model outcome is often referred to as a counterfactual [17]. In this paper they propose a model RobX to generate robust counterfactuals for tree-based ensembles, e.g., XGBoost. Another paper [21] introduces Feature Vectors, a new global interpretability method designed for tabular datasets. In addition to providing feature-importance, Feature Vectors discovers the inherent semantic relationship among features via an intuitive feature visualization technique. The CERTIFAI model in paper [37] combines Explainability, Fairness and Robustness altogether for Black-Box models. The framework allows the user to calculate robustness and fairness scores for individual models and generate explanations for individual predictions which provide a means for actionable recourse (changes to an input to help get a desired outcome).

Many of the above methods are not suitable for integration, as some are just introduced in papers with no library implementation and some are not suitable for regression problem on tabular data. Like ALE(Accumulated Local Effects) is a better option than PDP in the Alibi Explain library <sup>7</sup>. The Integrated Gradients method from Alibi Explain library also proves to be useful for many DNN models, as it is gradient based. I plan to compare its results to Kernel Shap explanations from SHAP or Alibi Explain library

---

<sup>7</sup> <https://docs.seldon.io/projects/alibi/en/latest/index.html>

on my model. For counterfactual calculations, only DiCE library <sup>8</sup> proves to be useful, as it works on regression problems.

Certain disagreements can arise on which method to use and when, amongst end users, thus bringing about comparison among XAI libraries. There are also certain evaluation methods to find the optimal explanation method. One of them is OpenXAI, an open-source ecosystem comprising of XAI-ready datasets, implementations of state-of-the-art explanation methods, evaluation metrics, leaderboards and documentation to promote transparency and collaboration around evaluations of post hoc explanations. [6].

It is very essential for the decisions made by the ML models to be fair. There are many fairness libraries and frameworks which are used to measure fairness of such models. The What-If toolkit by Google, AI Fairness 360 by IBM and FairLearn by Microsoft are very popular. In FairLearn, specially the unfairness of a model with respect to harms caused is covered, like Allocation harms – These harms can occur when AI systems extend or withhold opportunities, resources, or information. Some of the key applications are in hiring, school admissions, and lending <sup>9</sup>. There are certain toolkits for mitigating bias like Derivable Conditional Fairness Regularizer and FairRegression. FairSight is a viable fair decision-making system to assist decision makers in achieving fair decision making through the machine learning workflow <sup>10</sup>. Aequitas: Bias and Fairness Audit Toolkit is also an opensource fairness toolkit made for data scientists. There is also an ongoing research on the FATE model [14] whose goal is decision support with use of ongoing human-AI co-learning, explainable AI and fair, bias-free and secure usage of data. Until now, for my prototype model, The dalex library <sup>11</sup> also provides many comparable fairness metrics and gives a chance to mitigate biases using resampling and reweight methods. I have used Fairlearn for its varied fairness methodologies in my thesis.

Even after making an ML model explainable and fair, is it enough for an end user to understand those explanations completely and without any hassle? Explanations are often represented in graphs and maps and other pictorial forms. Making them easily accessible to a layman is a commendable advancement. Human beings are most comfortable having meaningful

---

<sup>8</sup>[https://interpret.ml/DiCE/notebooks/DiCE\\_getting\\_started.html](https://interpret.ml/DiCE/notebooks/DiCE_getting_started.html)

<sup>9</sup><https://techarerearch.com/most-essential-python-fairness-libraries-every-data-scientist-should-know/>

<sup>10</sup><https://murat-durmus.medium.com/an-overview-of-some-available-fairness-frameworks-packages-ff22fde9d2f4>

<sup>11</sup><https://dalex.drwhy.ai>

conversations, and thus, it is most beneficial to have conversational AIs in decision making. We all know how chatting apps like Whatsapp and Telegram have a heavy influence on our lives. This is because we can have easy and meaningful conversations there. Any chatbot, if it is human-like, can become very interesting to converse with and get help from. One conversational XAI model, TalktoModel, has been presented in [40] where, using a LLM (GPT-3) the best explanation on the suggestions made by the ML model are delivered. There are a couple more conversational AIs in the market now. The very recent one is [43] where a Deep Neural Network has been explained with Heat mapping which has been fed to ChatGpt via image captioning prompts. ChatGpt generates expert textual report on the visual explanations. In this thesis, a conversational AI model based on GPT-4 have been be constructed which will differ from the Talkto Model in many ways. TalktoModel can only work on classifiers and uses more technical terms and percentages in its answers as is depicted in Table 1 of the research paper. Also, it does not explain the fairness aspect of the data or model. In my model, a regression problem has been solved with neural network and explained using various methods. These explanations have been fed to the LLM GPT-4 which will provide more human like answers , just like a simple conversation, understandable by a layman with no domain knowledge. I have also explained the results from the fairness module of libraries by passing them through GPT-4. The model in [4] works on pictorial data, so not totally comparable to this thesis, as it is on tabular data. A new addition to OmniXAI<sup>12</sup> model has been introduced few days ago where Shap values and MACE counterfactuals on ML models have been explained using ChatGPT. However, while experimenting it using my dataset and prototype model, I found out that it is not working for regression problems. My main focus is explaining the fairness and bias of the model and the dataset. Along with that, dataset analysis before training with the data is also explained using GPT-4 in my thesis.

### 3 Background

Various concepts have been explored to construct the model. They are as follows:-

---

<sup>12</sup> <https://opensource.salesforce.com/OmniXAI/latest/index.html>

### 3.1 Concepts

#### 3.1.1 Machine Learning

The domain of machine learning has shown remarkable expansion and advancement in recent decades. Machine learning, a fundamental component of artificial intelligence, has emerged as a crucial driver of technological progress, exerting its influence across diverse industries like healthcare, finance, and autonomous systems. The inception of machine learning may be attributed to the mid-20th century, when visionaries like Alan Turing and Arthur Samuel established the fundamental principles for the advancement of intelligent computers. [42] The development of early ideas, such as perceptrons and rule-based systems, laid the foundation for the advancement of increasingly intricate algorithms. [35] The renewed fascination with neural networks, namely deep learning, signified a fundamental change in the field of machine learning. The proliferation of robust technology and the accessibility of extensive datasets have greatly contributed to the triumph of neural networks, facilitating significant advancements in image recognition, natural language processing, and several other intricate jobs. [44]

The core foundations of machine learning consist of three major paradigms: supervised learning, unsupervised learning, and reinforcement learning. Each of these paradigms serves specific objectives in tackling real-world difficulties.

Most practical machine learning is supervised. **Supervised learning** involves using an algorithm to learn the mapping function between input and output variables ( $x$ ,  $Y$ ).  $Y = f(X)$  The purpose is to estimate the mapping function well enough to anticipate the output variables ( $Y$ ) given fresh input data ( $x$ ). It is as if a teacher supervises an algorithm's learning from the training dataset, thus the term supervised learning. We have the right answers, so the algorithm predicts on the training data and the teacher corrects it. When the algorithm performs well, learning ceases. Regression and classification are supervised learning problems. **Classification:** A classification problem has an output variable like "red" or "blue" or "disease" and "no disease". Sometimes numbers reflect these divisions but have no meaning. Just labels. **Regression:** A regression problem involves a real numerical output, such as "dollars" or "weight". Using classification and

regression, recommendation and time series prediction are typical concerns. Popular supervised machine learning algorithms include: Linear regression for regression issues, Random forest for regression and classification and Classification support vector machines.

**Unsupervised learning** uses input data (X) without output variables. Unsupervised learning models data structure or distribution to learn more about it. The absence of an instructor and right replies makes this unattended learning. Algorithms find and display data structure on their own. Unsupervised learning challenges include grouping and association. **Clustering:** A clustering problem involves discovering data groups, such as client purchasing behaviour. **Association:** An association rule learning issue involves discovering rules that explain huge chunks of your data, such as X buyers prefer to purchase Y. Popular unsupervised learning methods include: Clustering with k-means, Association rule learning Apriori algorithm and Topic modelling of text passages using LDA(Latent Dirichlet Allocation) to find and associate keywords.

**Semi-supervised learning** issues involve a lot of input data (X) and just some labelled data (Y). These issues fall between controlled and unsupervised learning. In a photo archive, some photographs are labelled (dog, cat, human) but others are unmarked. This category includes several real-world machine learning challenges. This is because labelling data may need domain specialists and be costly or time-consuming. Unlabeled data is inexpensive and easy to obtain and keep.

Unsupervised learning can reveal input variable structure. Supervised learning may also be used to produce best guess predictions for unlabeled data, feed it back into the algorithm as training data, and utilise the model to predict fresh data. The latest development of language models in machine learning is semi-supervised: From words 1 to N in a phrase, the learning algorithm predicts word N+1. From input (X) comes label (Y).

Reinforcement learning trains agents to make successive judgements in an environment. Markov decision processes, reward functions, and policy optimisation are theoretical underpinnings. Robotics, gaming, and autonomous systems use reinforcement learning. In this part, Q-learning, deep reinforcement learning, and policy gradients are used in dynamic

decision-making. In real-world applications, reinforcement learning faces the exploration-exploitation problem, sample inefficiency, and safety issues. Meta-learning, safe exploration, and hybrid reinforcement learning-ML paradigms are being studied to overcome these issues.

### 3.1.2 LLM(Large language Model)

As humans, we see text as words. Sentences are word sequences. Documents have chapters, sections, and paragraphs. Text is just characters to computers. A recurrent neural network architecture can help machines understand text. Each word or character is processed by this model, which outputs when the input text is finished. This model is good, however it occasionally “forgets” what happened at the beginning when the series ends.

“Attention is All You Need,” by Vaswani et al., [44] established a transformer model in 2017. The attention process underlies it. The attention mechanism lets you perceive the whole text (or paragraph) at once, unlike recurrent neural networks. This helps the transformer model grasp word context. Many modern language processing models use transformers.

Transformer models require tokenization into words before processing text input. Encoding these tokens as integers creates embeddings, vector-space representations that maintain their meaning. Next, the transformer encoder converts all token embeddings into a context vector.

Text string, tokenization, and vector embedding are shown below. Text tokenization can be subwords, such as “nosegay” becoming “nose” and “gay”.

Context vector is core of input. The transformer decoder processes hints using this vector. You may give the transformer decoder the original input as a clue and let it create the next word. Then, use the same decoder with the preceding next-word as the hint. Starting with a leading sentence, repeat this to form a paragraph. This is autoregressive generation. A big language model works similarly, except it is a transformer model with many layers in its encoder and decoder, high context vectors to handle complicated concepts, and extensive input text.

Andrej Karpathy showed in his blog post “Unreasonable Effectiveness of Recurrent Neural Networks” [4] that they can anticipate the next word of a text. Language has redundancy and grammatical restrictions that limit word placement in sentences.

Claude Shannon’s famous article, ”Prediction and Entropy of Printed English,” [36] states that English contains 2.1 bits per letter despite hav-

ing 27 letters (including spaces). The entropy would be 4.8 bits if letters were employed randomly, making human language text prediction simpler. Transformer models and other machine learning models can forecast well.

A transformer model may word-by-word create the paragraph by repeating this approach. However, what is transformer model grammar? Grammar organises words into parts of speech and requires a certain sentence sequence. However, listing all grammatical rules is difficult. The transformer model learns these principles via examples, not directly. The transformer model must be large enough to learn beyond grammatical rules and include concepts from those instances.

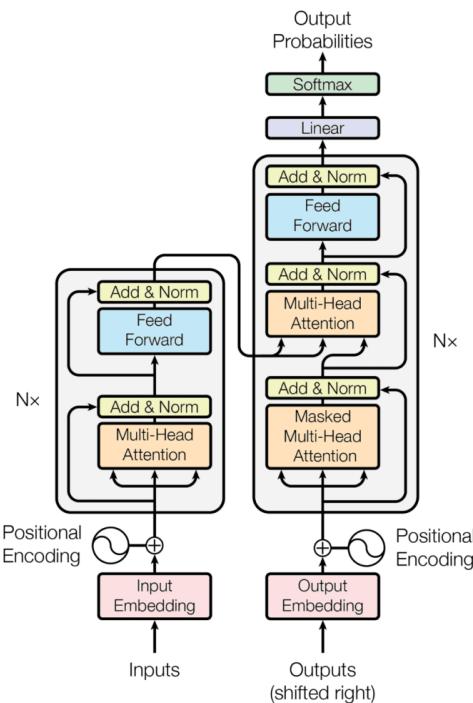


Figure 2: Transformer[41]

summarise, and answer questions, but these are unique “grammars” that match the preceding text, or prompts.

Generative AI encompasses content-generating AI algorithms. Generative AI generates text, code, graphics, video, and music. Midjourney, DALL-E, and ChatGPT are generative AI. Generative AI like large language models

Large-scale language models are transformer models. One computer cannot operate it due of its size. Naturally, it's an API or web interface service. As expected, such a huge model learns linguistic patterns and structures from a lot of material.

The GPT-3 model behind ChatGPT was trained on enormous volumes of internet text data. Sources include books, articles, websites, and more. The model learns the statistical correlations between words, phrases, and sentences during training to provide logical and contextually appropriate replies to prompts and queries.

This large volume of material allows the GPT-3 model to grasp different languages and themes. It may write in numerous styles because of that. You may be astonished that a big language model can translate,

build textual content from text. Generative text AI like ChatGPT is popular. All big language models are generative AI. [28]

Large language models may appear to understand and respond to meaning but they are technical tools and encounter several obstacles.

**Hallucinations:** A LLM creates erroneous output that doesn't fit the user's goal. It may pretend to be human, have feelings, or love the user. Large language models can't fully understand human meaning since they forecast the next syntactically right word or sentence. Sometimes this causes a "hallucination."

**Security:** Improperly maintained or monitored large language models pose significant security threats. They can steal personal data, send spam, and phish. Miscreants can reprogram AI to their beliefs and propagate falsehoods. Global ramifications can be disastrous.

**Bias:** Language model outputs depend on training data. Thus, the big language model will yield similar outcomes if the data is a single population.

**Consent:** Large language models are trained on billions of datasets, some of which may not have been consented to. Large language models have been found to flout copyright licences, plagiarise written content, and reuse proprietary content while collecting online data. Results cannot be traced, and producers are rarely credited, which might lead to copyright infringement. [39] They may also steal subject and photographer names from photo captions, compromising privacy. LLMs have been sued for intellectual property violations, notably by Getty Images. [2]

Scaling and maintaining large language models is time- and resource-intensive. Deployment: Large language models require deep learning, transformer models, distributed software and hardware, and technical skills to be deployed.

### 3.1.3 Conversational AI

Conversational AI refers to the artificial intelligence that enables robots to comprehend, analyse, and react to human language.

Conversational AI may be conceptualised as the cognitive system that drives the functionality of a virtual agent or chatbot. It involves a range of technologies that collaborate to provide efficient and automated communication through text and speech. This is achieved by comprehending the intentions of customers, interpreting language and context, and providing responses that resemble human-like interactions. The significance of Conver-

sational AI has grown significantly in recent years owing to its capacity to enhance any organisation, affecting:

Conversational AI enables organisations to offer round-the-clock customer assistance by utilising chatbots and virtual assistants. This allows them to effectively manage a high influx of client questions and support demands.

**Personalisation:** Conversational AI systems utilise natural language processing (NLP) to comprehend the purpose behind client questions and offer pertinent recommendations, so enabling businesses to personalise the customer experience.

**Efficiency:** Chatbots have the capability to mechanise mundane and repetitive operations, so liberating people to concentrate on intricate and significant assignments, thereby augmenting efficiency and productivity.

Conversational artificial intelligence can be employed to develop user interfaces that are accessible to those with impairments or those who are unable to utilise conventional interfaces owing to illiteracy or inadequate technological proficiency.

**Decision making:** It may also be utilised to gather and evaluate data from consumers and staff in real-time, offering valuable insights that can guide decision-making and enhance corporate performance.

How can conversational AI facilitate the translation of human language into a format that a computer can comprehend and generate responses to in a way resembling human-like communication? At first glance, it may appear to be straightforward - a consumer engages with a virtual assistant and receives a suitable response. However, there are other technologies operating in the background to ensure the seamless execution of this process. The initial stage entails the utilisation of Natural Language Processing (NLP). The role of Natural Language Processing (NLP) is to rectify spelling errors, ascertain synonyms, analyse grammar, detect sentiment, and deconstruct a query into words and phrases, hence facilitating comprehension for the virtual agent.

After the request has been formulated using Natural Language Processing (NLP), a variety of Deep Learning and Machine Learning models are employed. Referred to as Natural

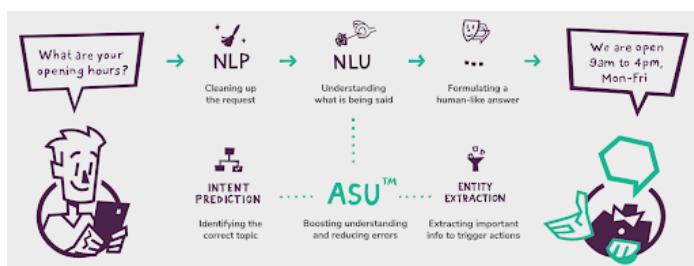


Figure 3: Sequential model architecture [5]

Language Understanding (NLU), these capabilities enable conversational AI systems to accurately determine the intended meaning or subject of a request and extract other significant details that may be utilised to initiate further actions, such as context, account preferences, and entity extraction.

Exclusive algorithms also have a significant impact on improving the overall natural language understanding (NLU) capabilities of a chatbot. Boost.ai's conversational AI platform incorporates Automatic Semantic Understanding (ASU), an algorithm designed to complement Deep Learning models and enhance the accuracy of user intent interpretation, hence minimising the risk of misunderstandings.

A chatbot is a distinct form of conversational AI, whereas conversational AI encompasses a broader array of technologies and methodologies. The differentiation between the two is frequently determined by the extent and concentration, while chatbots primarily concentrate on the creation and implementation of distinct conversational interfaces, while conversational AI primarily focuses on the fundamental technologies and methodologies that enable the existence of such interfaces.

### 3.1.4 Neural Network

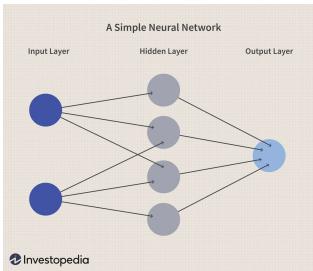
Neural networks use algorithms to simulate the human brain to find underlying links in data. In this respect, neural networks are biological or artificial neuron systems.

Neural networks adapt to changing input to produce the optimal outcome without rethinking output criteria. Neural networks, rooted in artificial intelligence, are becoming prominent in trading system development.

In finance, neural networks help produce time-series forecasting, algorithmic trading, securities categorization, credit risk modelling, proprietary indicators, and price derivatives.

A neural network functions like the brain. A neural network's "neuron" is a mathematical function that gathers and classifies data by architecture. The network resembles curve fitting and regression analysis. Layers of nodes form a neural network. Like multiple linear regression, each node is a per-

Figure 4: Sequential model architecture[22]



ceptron. A multiple linear regression signal is fed into a nonlinear activation function via the perceptron.

Though integrated computers that can think have existed for millennia, neural networks have advanced most in the past 100 years. Warren McCulloch and Walter Pitts from Illinois and Chicago published "A Logical Calculus of the Ideas Immanent in Nervous Activity" in 1943. The study examined how the brain may form complicated patterns and be reduced to a binary logic framework with true/false connections. [26]

Frank Rosenblatt of Cornell Aeronautical Lab. invented perceptron in 1958. His findings gave McColloch and Pitt's work weight, and Rosenblatt used it to show how neural networks could recognise images and draw conclusions.[33]

After a 1970s research drought due to financing issues. Jon Hopfield presented Hopfield Net, a 1982 recurrent neural network article. Backpropagation returned, and many academics saw its promise for neural networks. Paul Werbos' PhD thesis is frequently attributed with the main contribution.[45]

Neural network initiatives for direct use have increased recently. Deep Blue, built by IBM, dominated chess by pushing computers to perform complicated computations. Though famous for beating the world chess champion, these machines are also used to develop novel drugs, analyse financial market trends, and execute large scientific computations.[3] **Multi-layer Perceptron** Multi-layered perceptrons (MLPs) have linked perceptrons. This layer gathers input patterns. Input patterns can translate to output layer classifications or signals. For example, the patterns may list technical indicator numbers for a securities and output "buy," "hold," or "sell."

Hidden layers adjust input weightings to minimise neural network error. Hidden layers are thought to extrapolate important input data characteristics that predict outcomes. This discusses feature extraction, which is useful like principal component analysis.

Neural Network Types are:-

**Feed-forward neural networks** One of the simplest neural networks is feed-forward. Information flows in one way through input nodes until it reaches output mode. Most facial recognition solutions employ feed-forward neural networks with hidden layers for functionality.

**Recurrent neural networks** Recurrent neural networks are more complicated and feed back processing node output. Result: theoretical "learn-

ing” and network enhancement. Each node retains past processes, which are utilised throughout processing.

For networks with inaccurate predictions, the system will try to figure out why the correct outcome happened and modify. Such neural networks are employed in text-to-speech systems.

**Convolutional Nets** Convolutional neural networks (CNNs) organise input into categories in several layers. These networks have input, output, and many hidden convolutional layers. Layers develop feature maps that record picture regions and break them down until they provide useful outputs. Layers can be pooled or coupled, making these networks useful for image recognition.

**A deconvolutional neural network** Simple deconvolutional neural networks reverse convolutional neural networks. The network detects objects that a convolutional neural network could have considered relevant. These elements were presumably rejected during convolutional neural network operation. This neural network is commonly used for image processing and analysis.

**Module-based neural networks** Multiple separate networks make up modular neural networks. These networks do not interact during analysis. Instead, these mechanisms speed up difficult computer tasks. Network independence, like modular real estate, aims to assign each module a specific role in a larger picture.

Financial operations, corporate planning, trade, business analytics, and product maintenance employ neural networks. In forecasting, marketing research, fraud detection, and risk assessment, neural networks are widely used.

A neural network analyses pricing data to find trade opportunities. Networks can detect subtle nonlinear interdependencies and patterns that technical analysis cannot. Research shows neural networks’ stock price prediction accuracy varies. Some algorithms accurately anticipate stock prices 50–60% of the time, while others do 70%. Some say a neural network’s 10% efficiency boost is enough for investors.[30]

Neural networks can handle hundreds of thousands of financial transaction data. This can improve comprehension of trade volume, range, asset correlation, and investment volatility expectations. Since humans cannot efficiently process years of data (often acquired at second intervals), neural networks can identify trends, analyse outcomes, and anticipate asset class value changes.

### 3.1.5 Explainability

AI offers various ways to better private and public lives. Data science uses automated methods to find patterns and structures in massive datasets, which drives applications in computational biology, law, and finance. Despite its good influence, these systems pose substantial challenges: how do we trust their decisions? The underlying workings of data-driven technologies like machine learning and pattern recognition models can be difficult to grasp. Explainability can at least help stakeholders grasp model characteristics, providing insights like in the attached figure:

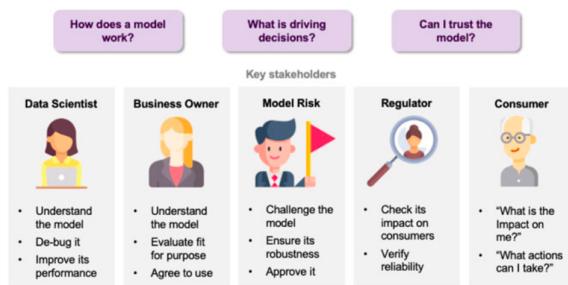


Figure 5: Business roles[10]

- Debugging and improving model performance can aid data scientists.
- Business owners considering model fit with strategy and purpose.
- Model Risk analysts testing the model for robustness and implementation.
- Regulators assessing model reliability and customer effect.

- Consumers demanding clarity about decision-making and their impact.

Looking at explainability another way, the proposed methodologies can assist address the following essential problems when delivering a product or making automated prediction-based decisions:

- Correctness: Are we sure just the relevant variables influenced our decision? Have we eliminated false patterns and correlations from our result? Are we convinced that the model is not vulnerable to modest perturbations, and if so, is that supported by the outcome? Do we trust the model to perform well with missing or noisy data? Can we discover and rectify data-specific biases that unfairly penalise groups of people? What tangible ways might the prediction model be improved? What impact would more training data or feature space have?
- Transferability: How can a prediction model for one application domain be applied to another? What data and model attributes must be changed for transferability? Can we describe the model's

algorithmic machinery to an expert? Perhaps a layperson? Do you think that will help spread the model? The fact that ML models are being used in more enterprises makes this paper extremely relevant. However, as approaches become more common and complicated, business stakeholders have more worries about model flaws, data-specific biases, etc. Data science practitioners may not be aware of academic methodologies or fail to distinguish between methods, thus they use industry standards like SHAP (Lundberg and Lee, 2017) [25]. We surveyed industry practitioners and data scientists to help them grasp explainable machine learning and use the correct tools. Our last sections focus on simplifying problems and methods to explainable machine learning. Human comprehensibility aspects should be considered before discussing explainability methods. We'll start with transparency, meaning people can grasp the model's inner workings. We then discuss model assessment criteria. We conclude by discussing model explanations. These concepts overlap, and a comprehensive definition or formalisation is often absent and hard to agree on. Transparency indicates a human comprehension of the model's inner workings (Lipton, 2016)[24]. Consider three dimensions:

At the first degree of transparency, a model may be simulated by a person. Naturally, only basic, small types qualify. While simplicity is important, a high number of basic rules would prevent a person from calculating the model's choice by thinking. However, basic models like a neural network without hidden layers may fall into this group.

- The second degree of transparency, decomposability, involves breaking down a model into input, parameters, and calculations and explaining them. Unfortunately, not all models meet this requirement. In algorithm Transparency, the third level, is understanding how the model generates output. For instance, a model that classifies instances based on a similarity measure (such as K-nearest neighbours) meets this criterion since the process is clear: select the datapoint most similar to the one under consideration and assign it the same class. However, larger models like neural networks have an illusive loss function and must approximate the training target. One condition for a model to come into this category is that the user can analyse it mathematically. Machine models can be transparent or opaque/black-box, however as seen above, this distinction is not binary. Despite the differences, decision trees, linear regression, and others are seen as simpler, transparent models and random forests, deep learning, as opaque models, partially because contemporary applications seldom employ a single perceptron

neural network. Although first explored for rule extraction approaches (Craven and Shavlik, 1999), we may evaluate models' explainability using the following dimensions:

- Human comprehensibility: How well extracted representations are understood, which relates to transparency.
- Fidelity: How well extracted representations match opaque models.
- Accuracy: Extracted representations can predict unseen examples.
- Scalability: The method's capacity to scale opaque models with huge input spaces and weighted connections.
- Method generality: How much specific training or opaque model limitations are needed.

We repeat that such notions are difficult to quantify but can guide future research. We may examine some post-hoc reasons for opaque models (Arrieta et al., 2019) [8]:

- Text explanations use symbols like real language text to explain. Other examples are propositional symbols that define abstract notions that encapsulate high-level processes to explain model behaviour.
- Visualisations aid model comprehension. Our incapacity to perceive more than three dimensions is a difficulty, but the created methodologies can assist us understand the decision boundary or how features interact. This is why visualisations are usually employed as a supplement, especially for non-experts.
- Local explanations describe how a model works in a specific place. This means that the explanations may not represent the model's global behaviour. Instead, they approximate the model around the instance the user wishes to explain to extract explanations about how the model works in such cases.
- Examples show how the model works using representative occurrences from the training dataset. Humans often use specific instances to explain a more general process. Since random vectors with hundreds of variables may include difficult-to-find information, an example must use human-comprehensible training data like photos.
- Explanations via simplification approximate opaque models with simpler, easier-to-interpret ones. The major problem is that the basic model must be flexible enough to correctly represent the complicated model. Usually, this is determined by comparing the classification accuracy of these two models.
- Feature relevance explanations quantify input variable influences to explain model decisions. Higher significance ratings indicate that the variable was

more essential for the model. These scores alone may not explain the model’s rationale, but they are a start.

The next part distils literature observations and methods. We may not be able to address all dimensions in this part, but they drive further decisions. Five literature-summarizing “frameworks” summarise the rapid growth of explainable machine learning (XAI). These frameworks assist us comprehend by comparing and visualising:

- Identify explainability restrictions of existing models,
- Explore explanation methods, and
- Consider alternate models for redesigned applications.

These frameworks will overlap, as predicted.<sup>3</sup> Our first two frameworks are based on (Arrieta et al., 2019) citearrieta2019explainable but changed. The third and fourth frameworks analyse prominent XAI implementations’ strengths and weaknesses. Our fifth method and pipeline is a “cheat sheet” based on the construction of many libraries for machine learning model analysis and interpretation [Molnar, 2020] [27]. Following figure shows a taxonomy of models by explainability. Based on this taxonomy, the following frameworks will describe transparent and opaque ML models (Transparency framework) and explainability approach capabilities (XAI Capability framework).

## Map of Explainability Approaches

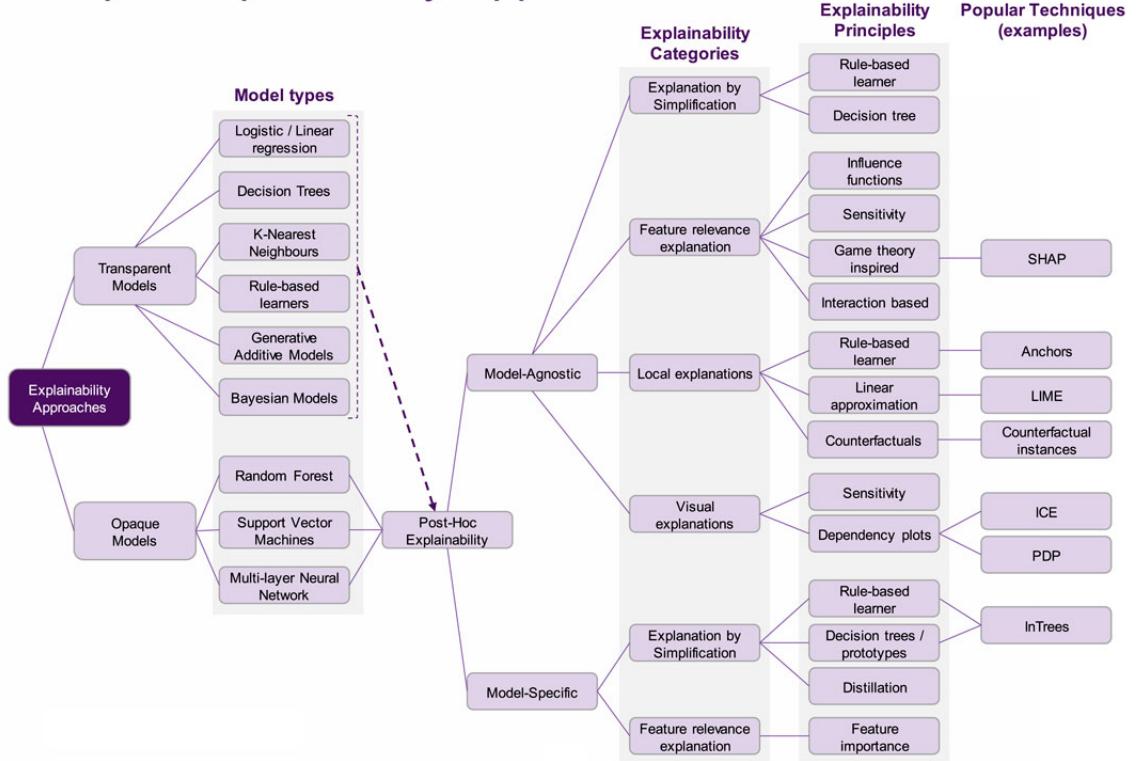


Figure 6: Map of Explainability Approaches[10]

### 3.1.6 Fairness

Introduction Machine Learning (ML) advocates ‘learning by example’. To clarify, a machine learning model analyses data to find patterns and complete the task. However, the model is not explicitly told what patterns to look for. The model often learns undesired assumptions as a result. Specifically, data-driven decision making is only as trustworthy as its data. Machine Learning models are used in sensitive decision-making processes including recruiting, loan approvals, customer selection, etc. These models must be impartial and fair. Only 26% of computer jobs are held by women, according to NCWIT data. Only 3% are African American women. Models based on this data will develop these prejudices and unfairly penalise minorities, even if qualified. Using methods to assess and eliminate biases

from machine learning models is crucial. Machine learning model bias measurement approaches have been covered in several papers. None of them could describe the concepts intuitively. Machine learning should not propagate societal prejudices—good turned bad!

### [38] Protected/Sensitive Features

The frequent bias indicators are: Ethnicity, Gender, Religion, Age, Marital Status, Citizenship. These are sensitive or protected characteristics because machine learning models shouldn't be biased on them. The model may have many more features, depending on choices. This leads to the next crucial idea of PROXY characteristics. Remove these sensitive features from the data before training to make the model unbiased? No, unfortunately, some qualities automatically contain these sensitive attributes' biases. Neighbourhood and zip code biases overlap with ethnicity biases, making them proxy ethnicity traits. These qualities cannot always be eliminated, depending on the situation. One must constantly run fairness tests to ensure their model is unbiased. **Measuring bias**

How to test a model for bias? This is measured by independence, separation, and sufficiency. An intuitive example is better than attempting to comprehend a generic formula with conditional and independence symbols. College entrance decisions is the example here. A table is created to demonstrate simple computations.

S. No	Gender	True Label	Predicted Label
1	M	1	1
2	M	1	1
3	F	0	0
4	M	0	1
5	F	1	0
6	M	1	0
7	M	1	1
8	F	1	1
9	M	0	0
10	F	0	0

Figure 7: Calculations table[38]

Choice is sensitive to gender, which might be male or female. 1 means “You were good enough!” and 0 means “You were good enough, however.” This model mimics the application screening procedure, thus the prediction labels show who would have been accepted or denied. Equations are also provided that meet each of these conditions for a binary sensitive characteristic that

can be readily expanded. This notation will be used:

Protected attribute  $C=C(X, A)$  is the trained classifier.  $Y$  is the target

variable.

### **Independence—Equal outcomes/selection**

Example: Male and female acceptance should be equal! Male acceptance rate is  $4/6 = 66.67\%$ , whereas female acceptance rate is  $1/4 = 25\%$  (measured using the projected label). However, male and female selection rates must be equal to meet independence criterion. The data used to train the model has a male acceptance rate of  $4/6 = 66.67\%$  and a female acceptance rate of  $2/4 = 50\%$ . To satisfy the independence condition, we usually can't get a perfect model with 100% accuracy since the training data doesn't fit the criteria. In statistical terms, classification scores should be independent of the sensitive characteristic. There should be no reciprocal information between classifier scores and sensitive attributes. A General Formula:  $P(C=c | A=\text{male}) = P(C=c | A=\text{female})$

### **tenxtbfSeparation—Error Equality (Threshold-based outcomes)**

Example: Male and female rejection rates while being qualified for admission (false negative) should be equal! This relaxes the separation rule by not requiring the same false positive rate, but nevertheless conveys the idea. This criteria promotes the idea that “similar people should be treated similarly”—that is, there should be no bias when selecting a subset of suitable candidates for admission. Consider separation as a measure of error equality. Thus, each group should have equal chances of forecasting a false positive and false negative. The preceding data shows a  $1/4 = 25\%$  false negative rate for males and  $1/2 = 50\%$  for females. This model fails separation criterion. One benefit of this criteria is that perfect models may be learned. A model with 100% accuracy meets the separation requirement. A General Formula:

**Given a classifier output  $C=c$ ,**

$P(C=c | A = \text{male}, Y = \text{True}) = P(C=c | A = \text{female}, Y = \text{True}).$

$C=c | A = \text{male}, Y = \text{False} = P(C=c | A = \text{female}, Y = \text{False}).$  Separation may be better than Independence since the sensitive trait and target variable may be correlated. For instance, a corporation may argue that based on CLV (Customer Lifetime Value), Default Rate, etc., picking clients from diverse categories is a commercial requirement (cannot fulfil independence standards). Separation only permits correlation to the extent of the target variable, therefore there should be no biases in the client group. Cal-

ibration—Choices indicate similar accuracy per group Example: Given the admission choice, males and girls should have equal odds of qualifying. This criteria clarifies the sensitive attribute and target variable Y from context. The chance of the real variable being 1 should be the same for each group given a score. Our forecast for whether an applicant should be admitted matches their credentials. If it's the same for men and women, the algorithm can learn what makes a successful applicant without gender prejudice. One may also learn a perfect model using this criterion. The sufficiency formula is less obvious than the other two requirements, therefore we shall present it before calculating.

Given a classifier output  $C=c$ ,  $P(Y = \text{True} | A = \text{male}, C = c) = P(Y = \text{True} | A = \text{female}, C = c)$ .

$P(Y = 1 | c=1, A = \text{Male}) = \frac{3}{4} = 75\%$ ;

$P(Y = 1 | c=1, A = \text{Female}) = 1/1 = 100\%$ .

Probabilities show the sufficiency requirement was not satisfied. These figures suggest that the model favours guys in forecasting their credentials. Rectifying this benefits the college since it admits better pupils.

**Impossibility : The Fairness Theorem** How can I choose a criterion? Can I satisfy multiple? These questions arise after knowing these three conditions. First, only one of three conditions can be met, unless the sensitive attribute and target variable are uncorrelated (a degenerate/redundant scenario).

Ensuring equity in machine learning models is crucial for constructing ethical and accountable AI systems. Organisations may reduce prejudice by using many tactics, including pre-processing, in-processing, and post-processing techniques. In addition, they may adhere to MLOps best practices, including ongoing monitoring and review, iterative enhancement of models, and rigorous model governance. Engaging in collaboration with specialists in certain fields and teams with a variety of backgrounds is crucial for comprehending the social consequences of machine learning models and resolving issues related to fairness. Interdisciplinary collaboration involves the integration of specialists from different disciplines to provide a wide range of viewpoints, recognise prejudices, and create efficient measures to address them. Engaging stakeholders is essential to ensure that machine learning models are in line with their requirements and principles. Given the ongoing development of AI, it is crucial to prioritise the concept of fairness and take steps to reduce biases. This is necessary in order to establish confidence in machine learning models and encourage ethical

decision-making. Organisations may foster a more inclusive and fair future by implementing these measures.

Various technologies and toolkits have been used to construct the model. The list is as follows:-

## 3.2 Toolkits

### 3.2.1 Streamlit

Streamlit is a costless and open-source framework written entirely in Python. It allows data scientists to efficiently create interactive dashboards and online applications for machine learning without needing any prior knowledge of front-end web programming.

Streamlit is more advantageous for the application when:

1. Usability: Streamlit is intentionally meant to be accessible to beginners and demands just a minimal amount of code to develop interactive web applications. If you need a rapid and direct method to construct applications that utilise data, Streamlit might be an excellent option.
2. Streamlit offers native integration with popular data visualisation libraries such as Matplotlib, Plotly, and Altair, enabling the creation of data dashboards and interactive visualisations.
3. Streamlit is highly effective for prototyping and swiftly building web apps. Streamlit is a helpful tool for rapidly creating proof of concepts or prototypes.
4. Python Integration: If you are already proficient in Python and wish to utilise your Python programming abilities for web development, Streamlit is an ideal option as it is developed using Python.
5. Streamlit possesses a burgeoning community and an ecosystem of extensions and custom components that augment its usefulness. The Streamlit community offers a diverse range of tools and assistance.

Instances in which Streamlit may not be the optimal selection:

1. For the development of intricate and feature-packed online applications that demand substantial customisation on the user interface and server-side logic beyond what Streamlit can offer, it is advisable to explore alternative

frameworks such as Flask, Django, or full-stack JavaScript solutions.

2. Scalability: Streamlit is primarily intended for the development of prototypes and applications of moderate scale. If you foresee the requirement for an application that can handle a large amount of data and is ready for use in a real-world environment, you may want a more resilient framework.

3. Customisation: Streamlit provides a restricted range of choices for customising the visual and functional aspects of your application. If you need extensive customisation, it may be necessary to utilise a web framework that offers greater flexibility.

4. Team Collaboration: Streamlit is frequently employed by data scientists and independent developers. If you are engaged in a project that involves a larger team, you may find it more advantageous to utilise a framework that offers a wider range of collaboration and project management functionalities.

5. Novelty: Streamlit is a very recent framework, which consequently translates into a smaller community and less available resources compared to more established frameworks. There is a possibility that the option has been withdrawn. If you create something intricate, you may face adverse effects in the future. Streamlit for Community

To handle the UI part of my solution, I wanted some easy framework which I can get up and running in no time. Streamlit proved to be useful for that purpose over other UI frameworks like Gradio <sup>13</sup> and Flask <sup>14</sup>

### 3.2.2 GPT-4

The introduction of GPT-4 <sup>15</sup> represents a noteworthy achievement in the realm of artificial intelligence, specifically in the domain of natural language processing. Generative Pre-trained Transformers (GPT) are a class of deep learning models employed for the purpose of generating text that closely resembles human language. Typical applications encompass : Responding to inquiries, condensing written material, converting text into different languages, producing computer programming instructions, creating blog articles, narratives, dialogues, and many forms of content.

GPT models have several uses and may be further optimised by fine-tuning them with particular data, resulting in enhanced outcomes. Utilising

---

<sup>13</sup><https://www.gradio.app>

<sup>14</sup><https://flask.palletsprojects.com/en/3.0.x/>

<sup>15</sup><https://openai.com/research/gpt-4>

transformers enables cost savings in computing, time, and other resources.

The advent of transformer models, beginning with Google’s BERT in 2017 [15], has enabled the current AI revolution in natural language processing. Prior to this, text production was conducted using alternative deep learning models, such as recursive neural networks (RNNs) and long short-term memory neural networks (LSTMs). While they were effective at producing individual words or brief sentences, they were unable to generate lengthier text that appeared realistic.

The transformer methodology employed by BERT represents a significant advancement due to its utilisation of unsupervised learning methods. In other words, it did not need a costly annotated dataset for its training. Google employed BERT to comprehend natural language queries, but it lacks the ability to produce content based on a given prompt.

In 2018, OpenAI released a research paper titled ”Improving Language Understanding by Generative Pre-Training,” which discussed the application of their GPT-1 language model for enhancing natural language comprehension. This model served as a demonstration of feasibility and was not made available to the general public.

In the subsequent year, OpenAI released a further publication (Language Models are Unsupervised Multitask Learners) detailing their most recent model, GPT-2. On this occasion, the model was made accessible to the machine learning community and gained traction in text creation tasks. GPT-2 frequently produced a few phrases before experiencing a malfunction. This was cutting-edge technology in 2019.

In 2020, OpenAI released a new publication titled ”Language Models are Few-Shot Learners,” which focuses on their GPT-3 model. The model has a magnitude of 100 more parameters than GPT-2 and underwent training on a significantly bigger text dataset, hence yielding superior model performance. The model underwent continuous enhancements via many iterations, collectively referred to as the GPT-3.5 series. One notable iteration within this series is the ChatGPT, which specifically emphasises conversational capabilities.

This iteration captivated the global audience by astonishingly showcasing its capacity to produce extensive passages of text that closely resemble human writing. ChatGPT achieved unprecedented growth as it amassed a staggering 100 million users within a mere two-month timeframe, making it the fastest-growing online application in history.

GPT-4 has been created with the purpose of enhancing model "alignment" - the capacity to accurately comprehend and adhere to user intentions, while simultaneously ensuring more truthfulness and minimising the generation of objectionable or hazardous content. GPT-4 surpasses GPT-3.5 models in terms of the accuracy of its replies. GPT-4 has a reduced amount of "hallucinations," characterised by factual or reasoning mistakes, compared to GPT-3.5. It achieves a 40% better score on OpenAI's internal factual performance test.

Additionally, it enhances "manoeuvrability," which refers to the capacity to modify its actions in response to human commands. For instance, you have the ability to instruct it to produce text in a certain style, tone, or voice. Begin prompts with either "You are a loquacious data specialist" or "You are a succinct data specialist" and request an explanation of a data science term. Further information on crafting effective prompts for GPT models may be found [here](#).

An further enhancement lies in the model's strict compliance with safety measures. When presented with an unlawful or morally objectionable task, the system demonstrates a higher proficiency in declining the request.

Benchmarks play a crucial role in advancing AI. However, they also possess several limitations. The primary objective of the newly developed GPT-Fathom benchmark suite is to mitigate certain drawbacks and shortcomings. LLMs heavily rely on benchmarks for their growth, however, the absence of standardised metrics and criteria sometimes hinders the ability to effectively compare the findings of various research.

GPT-4 exhibits substantial enhancements in performance compared to other models across the majority of benchmarks. The most superior model that is not developed by OpenAI is Claude 2. Meta's Llama models demonstrate comparable performance to GPT-3.5 in terms of language comprehen-

sion and knowledge-based assignments. However, they notably trail behind in multilingual capabilities, mathematics problem-solving, and coding activities. [9] These reasons instigated me to use GPT-4 as the LLM for my solution.

### 3.2.3 Langchain

In essence, LangChain is a freely available framework designed for creating applications that utilise language models. The purpose of this system is to streamline the construction of these applications by offering a range of tools and abstractions that facilitate the integration of language models with other data sources, enable interaction with the surrounding environment, and support the development of intricate applications.

LangChain is implemented using the Python and JavaScript programming languages, and it provides support for a diverse range of language models, such as GPT-4, Hugging Face<sup>16</sup>, and others.

In order to utilise LangChain, it is necessary to own a language model. One option is to utilise a publicly available language model, such as GPT-4. Alternatively, you have the option to develop and train your own language model. After acquiring a language model, you may begin the development of apps using LangChain. LangChain offers a diverse range of tools and APIs that facilitate the integration of language models with different data sources, enable interaction with the surrounding environment, and support the development of intricate applications. LangChain operates by sequentially connecting a set of elements, referred to as links, in order to establish a systematic process. Every link in the chain carries out a distinct function, such as:

1. Applying formatting to the user's input
2. Retrieving information from a data repository
3. Invoking a linguistic model
4. Analysing the results generated by the language model

The links in a chain are interconnected in a sequential manner, with the output of one link serving as the input for the subsequent link. This enables the chain to do intricate operations by linking together elementary jobs. For instance, a rudimentary chatbot may be developed by using a sequence of connections that execute the following operations:

1. It is possible to tailor the functions of the links in a chain and rearrange

---

<sup>16</sup> <https://huggingface.co>

their sequence to generate various workflows. LangChain possesses a high degree of adaptability, enabling its use in the construction of many applications.

2. LangChain deconstructs the pipeline of natural language processing into distinct components, allowing developers to customise processes based on their requirements.

The fundamental elements of LangChain are Templates for prompts

Prompt templates are employed to structure user input in a manner that facilitates comprehension by the language model. They serve the purpose of providing context to the user's input or specifying the job that the language model should do. An illustration of a prompt template for a chatbot may consist of the user's name and their inquiry. Legal Master's Degrees

LLMs, or large language models, are extensively trained on vast datasets containing both text and code. They have the capability to execute a range of functions, including text generation, language translation, and question answering.

Indexes are repositories that preserve information regarding the training data of the LLM. The information can encompass the textual content of the documents, their metadata, and the interconnections among them. Indexes are utilised by search engines to locate the pertinent material in response to a user's query.

Retrievers are computational methods designed to search an index for targeted information. They can be utilised to locate papers that are pertinent to a user's inquiry, or to identify the most similar documents to a specified document. Retrievers play a crucial role in enhancing the promptness and precision of the LLM's reactions.

Output parsers have the task of formatting the replies that are produced by the LLM. They have the ability to eliminate undesirable content, incorporate more details, or modify the structure of the answer. Output parsers play a crucial role in ensuring that the replies generated by the LLM are comprehensible and user-friendly.

Agents are computational entities capable of employing logical thinking to analyse issues and decompose them into smaller, more manageable sub-tasks. Agents can be employed to regulate the progression of a sequence and to determine which jobs to execute. For instance, an agent might be employed to ascertain if a user's inquiry is more suitably addressed by a language model or by a person specialist. Stores that specialise in selling vector products.

Vector stores are repositories that store vectors, which are mathematical representations of words and sentences. Vector repositories may be utilised to identify analogous words and phrases, which can be advantageous for applications such as interrogative resolution and condensation. For instance, a vector repository may be employed to identify all the words that bear resemblance to the term "cat".

Benefits of using LangChain:-

1. Usability: LangChain offers a sophisticated API that simplifies the process of integrating language models with various data sources and constructing intricate applications.
2. Versatility: The framework has the capability to construct a diverse range of applications, encompassing chatbots and question-answering systems.
3. Scalability: LangChain enables the development of applications with the capacity to efficiently process substantial volumes of data.
4. Open-source: LangChain is a framework that is freely available for usage and may be modified by anybody.
5. Community support: LangChain boasts a substantial and dynamic community of users and developers who are readily available to offer assistance and support.
6. Documentation: The documentation is thorough and user-friendly.
7. LangChain has the capability to seamlessly incorporate with several frameworks and libraries, like Flask and TensorFlow.
8. Extensibility: The framework possesses the capability to be expanded, allowing developers to incorporate their own distinct features and functions.

These are the reasons to use LangChain in my solution to integrate the Streamlit UI and GPT-4 LLM.

### 3.2.4 SHAP

<sup>17</sup> Shapley Values refer to a concept in cooperative game theory that quantifies the contribution of each player in a coalition by considering all possible permutations of players.

Shapley values are a useful tool for elucidating the results of a machine learning model. The Shapley value is a fundamental notion in game theory that is employed to ascertain the individual contribution of each participant in a coalition or a cooperative game. Presume that collaborative effort is essential to complete a project. The team, denoted as T, consists of p members. Given the unequal contributions of team members during the project, how may we allocate the entire value, denoted as v=v(T), created through this collaborative effort among the team members? The Shapley value, xm (v), represents the equitable allocation or distribution of rewards to individual team members, denoted as m. The function xm (v) is formally defined as :

$$\varphi_m(v) = \frac{1}{p} \sum_S \frac{[v(S \cup \{m\}) - v(S)]}{\binom{p-1}{k(S)}}, \quad m = 1, 2, 3, \dots, p$$

The summing is performed across all subsets S of the team T=1,2,3,...,p that may be formed by eliminating a particular member m. In the given equation, k(S) represents the size of S, v(S) denotes the value obtained by subteam S, and v(SUm) represents the actual value once m joins S.

A fair allocation is defined as an allocation that meets the following criteria:

1. Efficiency: The sum of individual contributions is equivalent to the actual value achieved by the team:

$$\sum_{m=1}^{m=p} \varphi_m(v) = v(T)$$

2. Symmetry: If two team members contribute the same value, they will receive an equal portion.

If the valuation of the set S union m is equal to the valuation of the set S union n, then the function xm (v) is equal to the function xn (v).

---

<sup>17</sup> <https://c3.ai/glossary/data-science/shapley-values/>

3. Linearity: When a team is involved in many projects (let's say two projects), each resulting in  $v(T)$  and  $u(T)$ , the combined share of each team member in the different projects is equivalent to determining their share using the total gain  $v(T) + u(T)$ . Put simply, the shares can be combined.

The equation  $xm(v + u) = xm(v) + xm(u)$  represents the property of linearity for the function  $xm$ .

The equation  $xm(av) = a xm(v)$  holds true.

Shapley values have been utilised in several domains. In machine learning, they may be utilised to elucidate the contributions of features, where the features represent the players or team members, and the model prediction corresponds to the payout of the game or collaboration.

In order to determine the significance of feature  $j$ , we may conceptualise the procedure as follows: for each iteration, we randomly choose feature values for all features except for feature  $j$ . Subsequently, we measure the discrepancy in prediction when feature  $j$  is included vs when it is excluded. The Shapley value is calculated by determining the average difference over all possible combinations. The Shapley value may be defined as the mean incremental impact of a characteristic, taking into account all potential combinations.

The primary benefit of the Shapley value is in its ability to allocate contributions to features in a fair manner, supported by a mathematically rigorous theory and well-defined properties of its mechanism. Nevertheless, the algorithm's drawback lies in its intricate nature, since the number of characteristics increases rapidly. Thus, in a real-world situation, the Shapley value can only be approximated by utilising a subset of combinations.

### **Shapley Additive exPlanations (SHAP)**

SHAP, an acronym for Shapley Additive exPlanations, is an interpretability technique that use Shapley values. It was developed by Lundberg and Lee in 2017 with the purpose of explaining individual predictions made by machine learning models. When attempting to elucidate intricate models like ensemble techniques or deep networks, it is customary to employ smaller local explanation models that serve as a comprehensible approximation

of the original model. The explanation model in SHAP is represented by a linear model, which is an additive feature attribution approach. It can also be described as the summation of the existing characteristics in the coalition game. Additionally, SHAP provides different methods for calculating Shapley Values.

KernelSHAP is an estimating technique that combines the ideas of local interpretable model-agnostic explanations (LIME) and Shapley values. KernelSHAP calculates the Shapley value by employing a weighted linear model, with the weights being proportional to those utilised in Shapley value estimation. This is hence used in my solution.

SHAP also offers a method for global interpretation by aggregating Shapley values. The calculation of feature significance involves the computation of Shapley values for each data point, followed by the summation of the absolute values over all data points. Dependence plots, created by combining SHAP visualisation with feature values, are useful for uncovering the connections between features and model predictions.

The significance of Shapley values lies in their ability to accurately allocate the contributions of individual players in cooperative games.

Model explainability enables us to analyse the decision-making process of a model, both on a broad scale and at a specific level. At a global scale, we can ascertain the specific characteristics that contribute to the conclusion of the model and the degree to which they impact the model's choice. At a micro level, we may investigate the specific reasons behind the model's conclusion and offer explanations if necessary. Local explainability enables us to investigate a model's limitations in terms of its ability to generalise across data and understand the underlying causes of its failures.

### 3.2.5 Fairlearn

Many general metrics are used from this library in my solution :  
Precision: Precision is a measure of a machine learning model's performance, specifically referring to the accuracy of positive predictions provided by the model. Precision is calculated by dividing the number of true positives by the sum of true positives and false positives. It measures the accuracy of

positive forecasts.

Recall: Recall, commonly referred to as the true positive rate (TPR), is the proportion of data samples that a machine learning model accurately classifies as belonging to a certain class of interest, known as the "positive class," out of the total number of samples for that class.

F1 score: The F1 score is a metric that quantifies the harmonic mean of accuracy and recall. The F1 score is frequently employed as an assessment measure in binary and multi-class classification tasks. It combines accuracy and recall to provide a comprehensive evaluation of model performance.

Selection rate :The selection rate refers to the proportion of data points in each class that are categorised as 1 in binary classification, or the distribution of prediction values in regression.

These metrics have been used to see how well the data fits with the model.

Demographic parity refers to the principle of equal representation of different demographic groups in a given context or situation. It is a fairness statistic that aims to guarantee that a machine learning model's predictions are not influenced by an individual's membership in a sensitive group. Demographic parity is attained when the likelihood of a certain prediction is not influenced by the existence of a sensitive group. Demographic parity, in the context of binary categorization, pertains to the equal rates of selection among different groups. For instance, within the framework of a resume screening approach, equal selection would include ensuring that the ratio of candidates chosen for a job interview is the same across all categories. Not achieving demographic parity might result in allocation harms. Allocation harms arise when AI systems distribute opportunities, resources, or information unequally among various groups. For instance, an AI recruiting system may exhibit a bias by favouring male candidates over female applicants, irrespective of their qualifications. Demographic parity serves as a measure to evaluate the degree of unfair allocation by assuming that resources should be distributed evenly across different groups. Among the metrics discussed in this section, it is the most straightforward to implement. However, the implementation of fairness through demographic parity relies on a few assumptions. Firstly, it assumes that the dataset does not accurately reflect the real-world scenario. For example, a resume

assessment system may exhibit a bias towards male applicants, filtering out qualified female applicants due to organisational preferences, regardless of their skill level. Secondly, it assumes that the dataset accurately represents the phenomena being modelled, but acknowledges that the phenomena itself is unjust. For instance, in the case of predictive policing, a system designed to predict crime rates may correctly identify higher crime rates in certain areas. However, it fails to consider that these higher rates may be a result of disproportionate policing and excessive criminalization in those areas. These assumptions may not accurately reflect reality. The dataset may accurately depict the phenomena, or the phenomena being modelled may not be biased. If either assumption is false, then demographic parity may not yield a relevant or helpful assessment of the fairness of a model's predictions.

Demographic parity, which is a fairness indicator, may be employed as an optimisation constraint while training machine learning models. Nevertheless, demographic parity may not be suitable for this objective as it does not impose certain criteria on the distribution of forecasts in relation to other significant factors.

**Equalized Odds ratio:** The acknowledgment of false positive rates recognises that various populations encounter distinct expenses due to misclassification. For instance, when a model predicts a negative result (such as the likelihood of reoffending) that already has a greater impact on minority communities, false positive predictions highlight the existing disparity in outcomes between minority and majority populations. Equalised chances ensures that the accuracy is uniformly high across all demographic groups, penalising models that excel exclusively on the majority groups.

If a machine learning model exhibits disparate performance across different groups, it has the potential to cause allocation or quality-of-service detriments. Equalised odds can be employed to identify both allocation harms and quality-of-service problems. The negative consequences of allocation are thoroughly examined in the preceding section on demographic parity. Quality-of-service harms arise when an AI system exhibits differential performance between different groups, such as face recognition systems that are more prone to errors when processing images of persons with dark skin. To obtain further insights on the negative impacts of AI, please refer to the

section under "Types of Harms".

The concept of equalised odds is valuable in identifying allocation harms since it aims to guarantee that a machine learning model performs equally well for all demographic groups. Equalised odds may be understood by comparing it to demographic parity. Demographic parity evaluates the overall distribution of resources, whereas equalised odds specifically examines the distribution of resources that were actually given to individuals within a particular group (as indicated by the positive target variable  $Y=1$ ). Nevertheless, equalised odds relies on the assumption that the target variable  $Y$  accurately measures the events being modelled. However, this assumption may not be true if the measurement fails to meet the criteria of construct validity.

These two metrics are important to find out the bias in the data and that is why has been used in my solution. They are the most popular metrics for fairness evaluation

**Adversarial unfairness mitigation :** Adversarial debiasing involves constructing two models. The initial phase is forecasting the desired outcome by utilising the feature engineering and pre-processing techniques that have been applied to the training data. The second model functions as an adversary, aiming to forecast the sensitive characteristic by leveraging the predictions made by your initial model. In an ideal scenario devoid of prejudice, this adversarial model should have limited predictive capability for the sensitive trait. The adversarial model is used to modify the original model by adjusting its parameters and weights in order to reduce its ability to accurately forecast the protected characteristics based on the results.

One primary benefit of this approach is the ability to actively intervene during the learning phase of the modelling procedure. Furthermore, it is applicable to both classification and regression tasks. Unlike reweighing, which we examined before and restricted the available modelling choices to those capable of handling weights, this method may be employed with any modelling strategy, including gradient-based learning models. Moreover, this strategy is also suitable for many fairness definitions.

It looked like a suitable mitigation technique for my solution and hence has been used.

## 4 Approach

### 4.1 Problem

A simple Conversational AI which can explain the underlying workings of a ML model and interpret if it works fairly or not is the goal of this thesis.

An experiment to make non-technical people with little domain knowledge understand the workings of machine learning models and explaining if the results need human intervention or not, is needed to demonstrate the problem. Simulation of a salary predicting process which has future use case adaptability needs to be constructed as an example for the experiment. The non-technical people should be able to interact with the model with ease and understand the salary-predicting process in detail through conversations. Now, how do we solve this problem? Let us break it into subproblems.

- Dataset creation - Two distinct datasets are given that reflect different biases and potential implications for the AI's fairness, accuracy, and overall performance for the candidate salary-predicting scenario.
- Development of the ML model - Developing a salary predicting ML model which can accurately predict the potential candidate salary.
- Training the ML model - It will be done using each of the two prepared datasets, creating a pre-trained version for each dataset.
- Results - Getting prediction for 1 instance
- Explainability - Explaining the salary prediction made by the ML model using explainability algorithms
- Fairness - Finding out if the model is fair on particular sensitive data given, using fairness algorithms
- Modularity - It has to be ensured the model can be adapted for future use cases, i.e. the model is modular in nature.
- Interactability - (move to future work) There are 3 subgroups of end users: development, deployment, use. An intuitive interface for each subgroup (development, deployment, and use) that allows them to interact with the AI system has to be made.

So, what is needed to create such a model? Assuming a huge tabular dataset for future use, a complex ML model like Deep Neural Network has to be taken into account. Then, it has to be trained to get desirable results. Also, the fairness criteria have to be measured in both these cases: what will happen if fairness methods are implemented to mitigate the bias and what will happen if biases are not mitigated. Explanations for a layman has to be given for every process being executed. Let us assume an imaginary conversation between the AI and user:-

User : Can this person be hired in this organisation?

AI : No, she cannot be hired.

User : Can you explain why?

AI : She is a female and it has been observed based on data, that less female employees have been hired for better functioning.

This example shows a bias based on the dataset provided as it discriminates based on gender which is considered socially unethical. What will happen when the bias will be mitigated? The conversation might change to:-

User : Can this person be hired in this organisation?

AI : No, she cannot be hired.

User : Can you explain why?

AI : Because she has low grades in Maths. The threshold is 80 out of 100. Now, this seems to be a more logical reason for not hiring someone. There are 2 aspects of this problem:-

1. The ML model is biased due to the data provided for its training

2. The explanation is consecutively biased based on the predictions of the ML model

Finally in the fairness evaluation part it can be found out if the data, along with the ML model and explanations are biased or not.

However, the explainability algorithm works correctly on the given data and provides explanations about the predictions. It is not its fault if the data itself is biased.

An option to evaluate the model before and after unfairness mitigation needs to be present. In addition, humans are more comfortable having conversations. This can be a mode of interaction for the user. All these problems have been individually and sometimes partially solved in previous

papers, but what is new in this problem statement is: The modularity of the whole model:

### **DNN\_for\_tabular\_data+Explainability+Fairness+Coversational\_Ai**

All these segments have never been handled totally in a modular manner where every segment can be replaced with another of its kind. A different ML Model working on a classification or regression problem, a different explainability method, a different fairness method, or a different conversational AI can be used, producing different combinations with minimal changes.

Conversational AI: Currently, there is no conversational AI explaining the fairness evaluation process and how the explainability methods work on a regression problem using DNN in a proper manner.

A conversational AI is needed here to better explain the reason behind the predictions to a layman. Usually, explainability methods highlight texts or show graph representations about how a ML model arrives to a prediction, but often it is not enough to properly understand the underlying interpretations of those representations. Conversations can help anyone understand the underlying interpretation of the same. Simple conversations are always easier to convey important information.

## **4.2 Solution**

The technical goal is to create a salary-predicting deep neural network for tabular data with the primary focus being fairness for groups based on sensitive features [1] like age, race, gender, and sexual orientation in the data with explanations of biased and unbiased predictions and their workings through a conversational AI based on a Large Language Model. For this, we need a layered architecture like in figure 8:

### **4.2.1 Requirements for Solution**

The layered architecture has some requirements which should be fulfilled in order to make it work as it is successfully.

1. A regression problem has to be provided to the ML model.

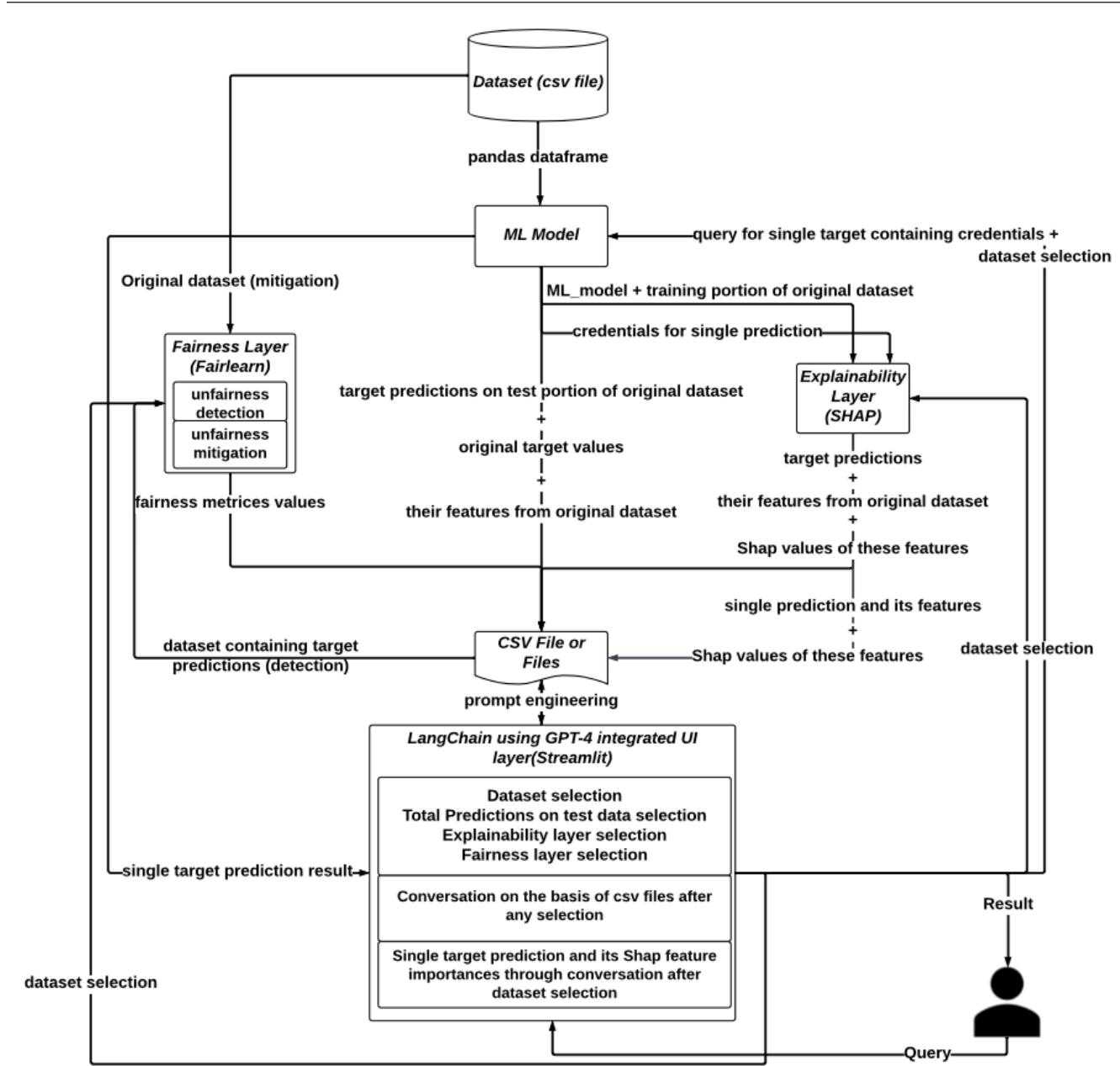


Figure 8: Solution Architecture

2. The ML model supports categorical and numerical features. Features need to be mentioned explicitly while providing them to the model.

3. The ground truth should be provided for training the model, as it is supervised learning. It is the value which results from the combination of different feature values and is known to be true. After training the model with a portion of this ground truth combined with feature values, the model will be able to predict target values, given the combination of needed feature values for the rest portion. The predicted values can then be compared to the ground truth values and checked if the model is working properly.

4. For a special case which will be discussed later (specific fairness evaluation), the target of the dataset has to be integer or float.

5. It is better if the features and target have real names, as it will be used to explain the relation amongst themselves to the end user.

6. The output of ML model, explainability and fairness layers have to be in pandas dataframes<sup>18</sup> which can be stored as csv files. The reason for this will be discussed later.

7. Prior knowledge of the sensitive feature in the dataset is required. Only 1 sensitive feature per dataset is supported at the moment. Sensitive feature is the column based on which the dataset can be biased, for example age, gender, race, sexual orientation.[1]

#### 4.2.2 Approach of Solution

Lets explain each of these layers consecutively and how they are connected to each other.

**The ML Model :** - In this layer, dataset selection is received from the user. The data is then extracted from the respective csv file. This action is denoted by the arrow from the dataset(csv file) to the ML Model. It is then passed through the column transformer, which transforms the categorical data to one-hot-encoded data.

Categorical data is a kind of data that may be stored and recognised using

---

<sup>18</sup> <https://pandas.pydata.org>

names or labels. It is a sort of qualitative data that may be classified rather than quantified statistically, for example, gender,hometown etc. **It needs to be transformed to numerical data through one-hot-encoding as the ML model here can handle only numerical data.** One-hot is used to transform each category value into a new categorical column and give it a binary value of 1 or 0. A binary vector is used to express each integer value. The values are all zero, and the index is denoted by a 1.

Then, the data is prepared for train-test split by separating the **features** and the **target**. The training data is used to train the ML model, and the test data is used to evaluate the performance of the ML model. The train-test split is a model validation technique that enables the simulation of a model's performance on fresh or unknown data. [20] The process operates as in figure 9:

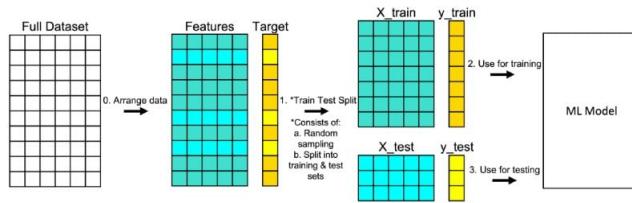


Figure 9: train-test split process [20]

Here the dataset is divided into two distinct halves - a training set and a testing set. This involves selecting rows from a dataset without replacing them, usually more than 70% of the total rows, and adding them to the training set. The remainder of the data is allocated to the test set. It is important to observe that the colours in the sections labelled "Features" and "Target" represent the specific destinations ("X\_train," "X\_test," "y\_train," "y\_test") for their respective data during a train-test split. This operation is performed to prevent the data from overfitting the model and hence accurately evaluating it.

Overfitting is a detrimental behaviour in machine learning when the model accurately predicts the training data but fails to do so for fresh data. Data scientists use machine learning models to make predictions, first training the model using a known dataset. Subsequently, using this knowledge, the model attempts to forecast results for new datasets. An overfit model is prone to producing erroneous predictions and lacks the ability to function well on di-

verse sets of fresh data. [7]

Accurate predictions can only be obtained when the machine learning model demonstrates generalisation across all data types inside its domain. Overfitting arises when the model lacks the ability to generalise and instead fits too tightly to the training dataset. It occurs for various reasons, including:

- Insufficient training data size, which fails to adequately represent all potential input data values.

- Presence of excessive irrelevant information, known as noisy data, throughout the training data.
- Prolonged training on a single set of data.
- High model complexity, resulting in the incorporation of noise from the training data.

After the train-test split, the ML model needs to be constructed. Any ML framework, Tensorflow<sup>19</sup> or Torch<sup>20</sup> can be used to make a dnn. Here, using the Keras<sup>21</sup> deep learning API from Tensorflow, a Sequential model is created. Tensors are multi-dimensional data structures.

The Sequential Model enables one to properly define a neural network that progresses sequentially from input to output, traversing a sequence of neural layers in a consecutive manner. [19] Figure10

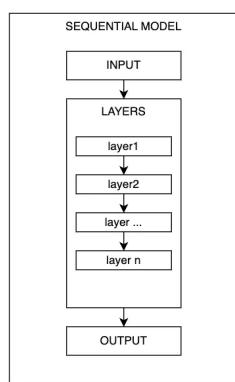


Figure 10: Sequential model architecture[19]

Here, an input layer, a second layer, and an output layer have been taken. All layers are of Dense type containing densely connected neurons. Each Dense layer output = activation(dot(input, kernel) + bias) where "input" refers to the data that is provided as input to the model. The "kernel" represents weights used in the model. The "dot" represents dot product of the input data and its associated weights, calculated using the numpy<sup>22</sup> library. The "bias" is a value used in machine learning to optimise the model by introducing a bias towards certain outcomes. The "activation" refers to the activation function, which is a mathematical function used on the output of a neuron to determine its final output. Activation function helps neurons to change their state. [18]

<sup>19</sup><https://www.tensorflow.org>

<sup>20</sup><https://pytorch.org/docs/stable/torch.html>

<sup>21</sup><https://keras.io>

<sup>22</sup><https://numpy.org>

The model is then compiled, fitted using training data from the train-test split outcome and used to predict the target given the test data feature columns as inputs. The predictions can be compared to the ground truth given in the original dataset, later.

The predictions along with the ground truth and all the features are put in a pandas dataframe and stored as csv file for later use. The arrow which goes from the ML Model to the csv file icon, in the architecture, denotes this action. The ML model and column transformer are also stored as sav and pickle files respectively for later use.

***The Explainability Layer :*** This layer represents the explainability method applied to the ML model in order to explain the feature importances of the dataset. Feature importance is a way to increase model understanding. It allows for the estimation of the individual contributions of each feature in the data towards the model's prediction. [32] By conducting feature importance tests, one may determine the characteristics that have the most influence on the model's decision-making process. To address this, the elements that have less influence on the model's predictions may be eliminated and one can concentrate on enhancing the more substantial features. This has the potential to significantly boost the performance of the model.

There are numerous methods for determining the importance of a feature. A novel and enhanced method known as Shapely Additive Explanations (ShAP) has been developed.<sup>23</sup> It has been widely used and is more consistent than traditional methods and hence has been used in the Explainability layer. The underlying principle of Shapely Additive Explanations is to estimate the significance of a feature by comparing the model's performance with and without the feature for each feature combination. It computes the local importance of features for each individual observation. It is possible for the significance of a feature to vary considerably among different data points. Thus, local feature importance determines, for each data point, the significance of each feature. A global measure is a singular summary of all of the features pertaining to the model. The Shapley value denotes the mean marginal contribution of a feature value when considering all conceivable feature combinations. The global measure can be derived through the aggregation of the local feature importances for every data point. [13]

---

<sup>23</sup>[https://shap.readthedocs.io/en/latest/example\\_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html](https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html)

For the Explainability layer to work, predictions from the ML Model for the respective dataset have to be present. The Explainability layer takes in the dataset selection from the user. The respective dataset is then extracted and sent to the ML Model. The ML Model returns the model instance, the training data and the predictions. This action is denoted by the arrow from the ML Model to the Explainability layer. Then, samples of required length are made out of training data. A considerable sample which can represent the whole dataset is needed to get correct explanations, but the samples should not be too long, in order to save computational efforts and time. The Kernel Explainer method from ShAP library is used to explain the feature importance. The KernelExplainer [12] constructs a weighted linear regression utilising the training data sample and the function responsible for generating the predicted values. The importance values of the features are calculated using the coefficients from a local linear regression and the Shapley values from game theory. It also works better than the Deep Explainer which can be conventionally used for dnn; so it has been used. [25]The Shapley values for all features are then stored in a dataframe with the column name prefix "Shap\_value\_of\_" for each feature column. These are joined with the predictions to form a dataset and stored as a csv file for later use.

***The Fairness Layer :*** The Fairness layer consists of 2 parts: The unfairness detection and the unfairness mitigation. Fairness concepts have been discussed in the background section. Here, the solution will be discussed. The Fairlearn library <sup>24</sup> has been chosen to detect unfairness in the data and consequently the ML model, and then to mitigate it. Unfairness can be introduced in the data in different manners. Minority groups turn to sensitive features like race, age, gender, sexual orientation, in the dataset. Any bias against these sensitive features result in unfairness of data and hence the ML model.

**1. Unfairness Detection :** The fairness layer too, takes in the dataset selection from the user. The ML model has to be put to action before detecting the unfairness. With the dataset selection, the predictions from the ML Model which had been stored earlier as csv file, is extracted. The prediction dataset is sorted in ascending order with respect to the prediction values. Here, the special case arises, where only regression problem with

---

<sup>24</sup><https://fairlearn.org>

target as continuous numeric value can be solved, as sorting cannot be done on binary predictions of classification problem. However, the ground truth and prediction have to be converted to binary data with respect to average value of the predicted results, as the metrics of Fairlearn can only work on Classification problem. There are 2 parts of this evaluation:-

- i) The sorted data is sliced according to 4 quantile ranges (25%,50%,75%,100%) and ranked(0,1,2,3). Fairness metrics are calculated on each slice.
- ii) The fairness metrics are calculated over the whole extracted prediction dataset.

The fairness metrics which detects the unfairness in the data and model are Demographic Parity Ratio and Equalized Odds Ratio.

**Demographic Parity Ratio** - It brings in the mathematical definition of fairness. [11] Its main notion is that, in an ideal scenario, the disparity between the groups experiencing a favourable result should be non-existent. Such ideal cases are rare in real world; so this metric serves the purpose appropriately. Mathematically speaking, Demographic Parity requires that the result is not influenced by the protected category, that is, the sensitive feature in the dataset. Demographic Parity should be used as a criterion for justice in situations when :-

- i) We want to enhance the existing status of the environment, such as increasing the representation of minority groups in top positions.
- ii) We acknowledge that our data may have been influenced by prior biases, such as a machine learning solution that was trained to employ software engineers and had a significant under-representation of women in its previous hiring practices.
- iii) We have implemented a strategy to assist the underprivileged demographic and to mitigate the perpetuation of historical prejudices (e.g. enacting laws that punish exclusionary conduct within the board).

The value of the ratio is scalar. It is the proportion between the lowest and greatest selection rates. A result of 1 indicates the presence of full demographic parity.

**Equalized Odds Ratio** - It is a more restrictive concept compared to the above metric. [11] This notion posits that the model should not only accurately detect the positive outcome at the same rate for all groups, but also make an equal number of incorrect identifications of the positive outcome for all groups (resulting in the same percentage of false positives for all groups). Equalised Odds should be employed as a criterion for fairness in situations when

- i) There is a significant focus on accurately predicting positive outcomes, such as correctly identifying individuals who should receive a loan in order to generate profits.
- ii) Additionally, it is crucial to prioritise the reduction of costly false positives, such as minimising the approval of loans to individuals who would be unable to repay them.
- iii) The goal variable is deemed objective, as subjective factors like prejudice might influence the labelling.
- iv) The reward function of the model is little affected (e.g.: the revenue or profit function for the firm stays at a high level).

Due to the reliance of equalised odds on both the true-positive and false-positive rates, an additional step is necessary to produce a singular scalar outcome. Equalized Odds ratio computes the ratio of true positive rate and false positive rate individually. Subsequently, the greater of these two disparities is returned. 1 refers to full equal opportunity.

These 2 metrics take in the arrays of binary labels containing ground truth and predictions separately and the array of corresponding sensitive feature as any datatype. Only one sensitive feature per dataset is supported in this solution. Once the metrics have been calculated, they are stored in csv files for later use.

**2. Unfairness Mitigation :** If the values of the Demographic Parity Ratio and Equalized Odds Ratio are less than 1, there is a chance of unfairness mitigation. However, there is usually a trade-off between the accuracy of the ML Model and the fairness. It depends from case to case, to which we should give more priority. Here, fairness is the priority, as it is the purpose of the thesis to explain the bias and its after-mitigation results. The Adversarial mitigation technique is used in this functionality. [47] Its objective is to train a model that can accurately predict the target variable based on the input data while also ensuring fairness by imposing restrictions related to sensitive attributes. This functionality takes in the dataset selection similarly from the user. After extracting the selected dataset, the ground truth in it is converted to binary label array, since the input has to be provided as binary data arrays for the Adversarial Classifier. Then the dataset is split into training and test data. Two models have to be instantiated, the predictor model and the adversary model, other specifications need to be provided and the training values passed along with the sensitive features. The predictions provided by the Adversarial Classifier are then evaluated using the Demographic Parity Ratio and Equalized Odds

Ratio metrics. The metric values are then stored as csv files for later use.

***LangChain using GPT-4 integrated UI layer*** All these functionalities needed to be wrapped up and presented to a non-technical user in a more accessible manner in the form of a Conversational AI. A user interface layer has been created keeping this purpose in mind. The Streamlit<sup>25</sup> and Langchain<sup>26</sup> frameworks have been used to create a chatbot with the provision of different datasets and functionality selections. The end user will be able to interact with the AI easily using this chatbot.

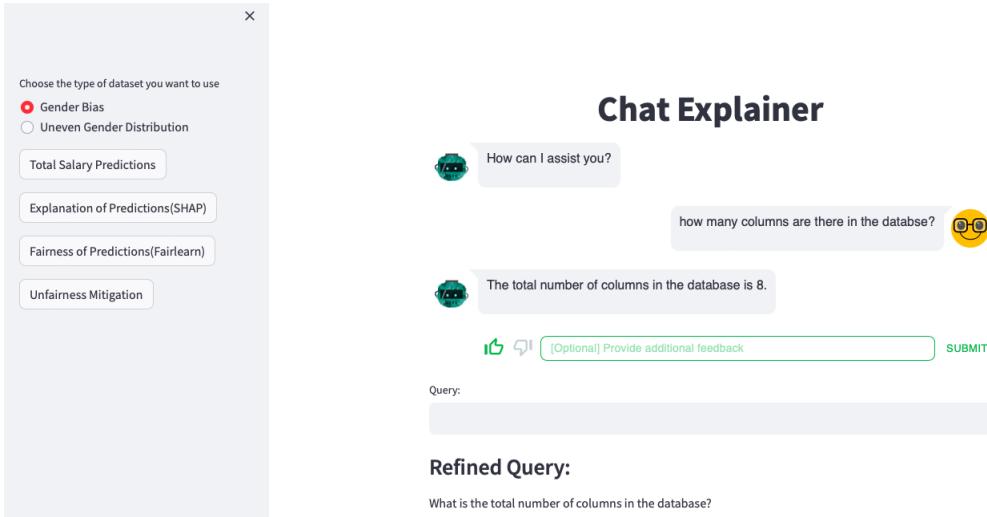


Figure 11: User Interface with Streamlit, powered by Langchain using GPT-4

The figure in 11 shows an example of the chatbot which is a part of the solution. Streamlit has been used to create the UI or forefront and Langchain powered by GPT-4 has been used to create the underlying connections with the ML Model.

As has been mentioned in the UI Layer in the architecture in 8, the sidebar of the UI contains the selection options for dataset and the other functionalities. Every functionality works only when a dataset is chosen. By default the first dataset is kept as the chosen one. Then, the total predictions

---

<sup>25</sup> <https://streamlit.io>

<sup>26</sup> <https://www.langchain.com>

on the test data of the dataset can be retrieved using the button. Questions regarding the dataset and its predictions to the chatbot are welcome only after the retrieval. The Shap Explanations button retrieve the feature importances of the predictions for the test dataset according to Shapley values and gets the chatbot ready to answer questions on that context. The Fairness of prediction button is dependent on the 1st button for retrieving the predictions. It uses the stored predictions which is generated when the 1st button is pressed. Once the Fairness of prediction button is pressed, Fairlearn evaluates the model and its predictions based on the metrics mentioned before and the results are used up by the chatbot as context to answer related questions about fairness. The Unfairness mitigation button connects the Adversarial mitigation method working on the biased dataset and the modified metric measures returned are used up by the chatbot to answer questions regarding mitigation processes in general and the improved metrics.

The feedback component representing the Thums up/down and the textbox beside it for additional feedback, have been incorporated to evaluate user feedback in another Streamlit app called Trubrics<sup>27</sup>. Example evaluation results will be discussed in the following sections.

Now let us delve into the underlying Langchain layer. LangChain is a Python framework specifically created to simplify the creation of AI applications. It places emphasis on efficiently handling real-time data processing and seamlessly integrating with Large Language Models (LLMs). It provides functionalities for data exchange, creation of vector embeddings, and streamlines the interaction with LLM, making it highly effective for AI developers. It is very effective in creating LLM powered chatbots. Here the GPT-4 (Generative Pre-trained Transformer 4) model has been chosen as the LLM.

The Langchain layer here has helped to build different functionalities based on different contexts, using GPT-4. It also connects with Streamlit seamlessly to create an effective UI.

For each button on the UI, Langchain Tools have been created to carry out the underlying related context selection to extract appropriate results. Tools are mechanisms that agents can utilise to engage with the environment. These tools can encompass generic utilities such as search functions, as well as various chains or agents. The basic idea behind agents is to utilise a LLM to select a sequence of actions: identifying which actions to carry out based

---

<sup>27</sup> <https://trubrics.streamlit.app>

on a given context and in what order, taking those actions, examining the outcomes and continuing the procedure iteratively until a desirable result is obtained. The Tool and agent features have already been further discussed in the Background section. When the "Total Salary Prediction" button is pressed and a relevant question about the dataset is asked in the chatbot, the tool called PredictedDatasetTool is invoked in the Langchain layer. This tool carries out extraction of the relevant csv file for predicted results on the test data along with the other feature columns as pandas dataframe, uses it as context for an agent, which it runs on the user query to return relevant results to the user. Similarly, the ShapExplainPredictedResultTool starts working to give relevant answers based on the csv files saved with Shap feature importance results on model predictions, when the "Explanation of Predictions(SHAP)" button is pressed and questions asked to the chatbot on that context. After pressing the "Fairness of Predictions(FAirlearn)" button and asking relevant questions to the chatbot, the FairlearnFairnessTool is selected which extracts the csv containing all the Fairlearn evaluation results as dataframes and feeds it as context to the agent to provide user with a justified result in a similar manner. The "Unfairness Mitigation" button, when pressed, displays the improved metrics after bias has been mitigated for the dataset selected. When relevant questions regarding the results are asked to the chatbot, the UnfairnessMitigationTool is invoked which extracts the relevant csv files containing the displayed improved metrics and passes it as context to the agent to receive relevant answers from the LLM to pass it on to the user.

Another additional functionality has also been introduced through Langchain Tool, which can be accessed via the chatbot only. One can get a single prediction in context of the dataset selected, given the relevant features to the chatbot in a query format. Once such a question is asked to the chatbot, the SinglePredictionTool is selected which parses the user query and passes the feature values on to the single prediction function of the ML Model. The arrow in figure 8 from the Langchain layer to te ML Model layer represents this action. The prediction from the model is then passed on to the user. The user can also ask questions about the feature importances of the single prediction based on Shapley values. In such a case, the SingleExplanationTool is invoked which retrieves feature importances from the csv file already filled up after predicting the single target, in the ML Model. This csv serves as the context for the agent which is run in the Tool to provided relevant answers to the user. Relevant questions which can be asked by the user have

been discussed in the Evaluation section.

Langchain also helps to maintain a chain of conversation in the chatbot by using memory to remember previous chats. It uses both Langchain memory and Streamlit memory to implement this functionality. It seamlessly connects to OpenAI APIs <sup>28</sup> and creates an abstraction to construct such functionalities in a simple manner. Another OpenAI API helps to create the Refined Query [29] in figure 11 by transforming the user input through LLM to a relevant proper question based on previous chats which will be provided as the final query to the LLM to extract appropriate results. GPT-4 by OpenAI <sup>29</sup> serves as the LLM for this purpose. The Trubrics library also integrates smoothly with the LLM through Langchain saving feedbacks with respect to queries from the user and responses from the LLM, in the Trubrics app for further evaluation.

#### 4.2.3 Implementation details

Here the technical implementation details of the solution will be discussed. Several python libraries need to be installed before starting off with the implementation.

##### *Libraries and their required versions :*

streamlit-chat=0.1.1 <sup>30</sup>  
langchain=0.0.345 <sup>31</sup>  
langchain-experimental=0.0.43 <sup>32</sup>  
trubrics=1.6.2 <sup>33</sup>  
pyparsing=3.0.9 <sup>34</sup>  
scikit-learn=1.1.3 <sup>35</sup>  
keras=2.12.0 <sup>36</sup>  
pickleshare=0.7.5 <sup>37</sup>

---

<sup>28</sup> <https://openai.com>

<sup>29</sup> <https://openai.com/gpt-4>

<sup>30</sup> <https://pypi.org/project/streamlit-chat/>

<sup>31</sup> <https://pypi.org/project/langchain/>

<sup>32</sup> <https://pypi.org/project/langchain-experimental/>

<sup>33</sup> <https://pypi.org/project/trubrics/>

<sup>34</sup> <https://pypi.org/project/pyparsing/>

<sup>35</sup> <https://pypi.org/project/scikit-learn/>

<sup>36</sup> <https://pypi.org/project/keras/>

<sup>37</sup> <https://pypi.org/project/pickleshare/>

shap=0.41.0 <sup>38</sup>  
fairlearn=0.8.0 <sup>39</sup>  
python-dotenv=1.0.0 <sup>40</sup>  
tensorflow=2.12.0 <sup>41</sup>  
openai=0.27.7 <sup>42</sup>  
streamlit-feedback=0.1.2 <sup>43</sup>  
tabulate=0.9.0 <sup>44</sup>  
imbalanced-learn=0.11.0 <sup>45</sup>

Now, let us go layer-wise discussing the implementation details referring to the architecture in figure 8 from top to bottom.

**Dataset & ML Model:** The tabular dataset is stored in csv file format making it easier to transform into pandas dataframe for the ML Model to use. Dataframes are two-dimensional data structures which are labelled and consists of columns that may have distinct types. ML Models in general use this data structure when dealing with tabular data.

The dataset selection from the user comes in as a String filenumber. The data is then extracted from the csv file representing the filenumber as a pandas dataframe. This action is denoted by the arrow from the dataset(csv file) to the ML Model. After that the target column(y) and the predictor/feature columns(X) are specified from the dataframe. Then the scikit-learn library <sup>46</sup> is introduced. Scikit-learn, often known as Sklearn, is a very valuable and resilient Python toolkit for machine learning. The library offers a range of effective tools for machine learning and statistical modelling, such as classification, regression etc. These tools are accessible through a unified interface in Python. Scikit-learn leverages on the NumPy <sup>47</sup>, SciPy <sup>48</sup>, and Matplotlib <sup>49</sup> Python libraries. The categorical columns of the dataframe are passed to the scikit-learn ColumnTransformer to change them into numeric form using

---

<sup>38</sup> <https://pypi.org/project/shap2/>

<sup>39</sup> <https://pypi.org/project/fairlearn/>

<sup>40</sup> <https://pypi.org/project/python-dotenv/>

<sup>41</sup> <https://pypi.org/project/TensorFlow1/>

<sup>42</sup> <https://pypi.org/project/openai/>

<sup>43</sup> <https://pypi.org/project/streamlit-feedback/>

<sup>44</sup> <https://pypi.org/project/tabulate/>

<sup>45</sup> <https://imbalanced-learn.org/stable/>

<sup>46</sup> <https://scikit-learn.org/stable/>

<sup>47</sup> <https://numpy.org>

<sup>48</sup> <https://scipy.org>

<sup>49</sup> <https://matplotlib.org>

the fit\_transform method.

After that, the scikit-learn train\_test\_split is used to split the data into train and test portions (X\_train, X\_test, y\_train, y\_test). This method takes in the data to be split, test\_size and random\_state. The test\_size parameter specifies the ratio of the original dataset that should be allocated to the test split. For this instance, it has been designated as 0.3, indicating that 30% of the data will be allocated for the test set, while the remaining 70% will be assigned to the training set. The random\_state option is utilised to initialise the internal random number generator, which determines how the data will be divided into train and test indices. This is done to guarantee the ability to replicate the results. Not specifying or passing an integer to the random\_state option might result in varying outputs each time the data is split due to shuffling.

Then the keras Sequential model is created as had been mentioned in the Approach of Solution section. All three layers(input,2nd,output) are of Dense type with parameters kernel\_initializer='normal' and activation='relu'. kernel\_initializer initializes the kernel weight matrix. activation is the activation function specified here as relu(The rectified linear activation function which is a piecewise linear function that outputs the input directly if it is positive, and zero otherwise. An activation function is a mechanism incorporated into an artificial neural network to facilitate the network's ability to comprehend intricate patterns within the data being processed. When comparing with a neuron-based model in human brains, the activation function plays a crucial role in determining which signals are sent to the next neuron.

Then the Sequential model is compiled using parameters loss = 'mean\_squared\_error', optimizer='adam'. The compiler verifies the presence of format errors and specifies the loss function, optimizer or learning rate, and metrics. It is a necessary step before training the model. The mean\_squared\_error (MSE) loss function is calculated by taking the average of the squared differences between the actual value and the predicted value. This is the regression loss function that is most often utilised. The Adam optimizer, also known as "Adaptive Moment Estimation," is an iterative optimisation technique employed to minimise the loss function when training neural networks. Adam may be regarded as a fusion of RMSprop and Stochastic Gradient Descent with momentum. After that the model is fit using the training data from the train-test split outcome (X\_train,y\_train) corresponding to features and target consecutively. The hyperparameters used here are batch\_size = 20, epochs = 50 and verbose=1. A sample refers

to a solitary instance of data arranged in a single row. The batch size is a hyperparameter that specifies the number of samples to process before changing the internal model parameters. The epoch count determines the number of iterations the learning algorithm will perform on the complete training dataset. the verbose determines how we want to 'see' the training progress for each epoch; verbose=1 will show us an animated progress bar.

After the model is fit, its time to predict the target values for the test data (X\_test). The predictions, along with their feature columns from (X\_test) and original target values/ground truth(y\_test) are sorted in ascending order and stored in a csv file called total\_prediction\_results.csv. The fitted model and column transformer are also saved in sav and pickle formats respectively for later use.

**Explainability Layer (Shap) :** Shap Explainability library has been used for the Explainability layer. This layer gets the dataset selection from the user as String filenumber. Using the read\_csv() method from pandas, the data is extracted from the csv file corresponding to the filenumber, as a pandas dataframe. Then, the training and test dataframes, an instance of the ML Model and the predictions for the dataset selected are extracted by passing the data to the ML Model. After that, the Shap library comes into action. Using all the training data-points as background data samples for Shapley value evaluation could cause slower run times. So, smaller samples of the train and test data (X\_train, X\_test) are selected for the Shapley value evaluations of the features from the data. The Shap sample() method is a basic encapsulation of scikit-learn's shuffle function. Its primary purpose is to downsample X, which is then utilised as a background dataset in the SHAP Explainer and its subclasses. I have taken 1000 data-points each for train and test. Then the Shap Kernel Explainer is instantiated using the predict function of the model instance and the training sample of 1000 data-points. After that the instance of the Kernel Explainer calls the shap\_values() method on the testing sample of 1000 data-points to evaluate the Shapley values of the features. These values are then inserted correctly into a pandas dataframe and joined to the original testing dataframe to store as a single csv file called shap\_result.csv for later use. The Kernel Explainer is also saved in pickle format for later use.

**Fairness Layer (Fairlearn) :** As already discussed, Fairness layer has two parts : Unfairness detection, Unfairness Mitigation. Implementation

of both the solutions will be discussed here.

**i) Unfairness Detection :** The user selects a dataset and the selection is passed to this layer as a String filenumber. The sorted dataset predictions corresponding to the filenumber are extracted as a pandas dataframe. There are 2 ways in which the fairness of the dataset is evaluated as discussed earlier :

- i) The sorted dataframe is sliced according to 4 quantile ranges (25%,50%,75%,100%) and ranked(0,1,2,3). Fairness metrics are calculated on each slice.
- ii) The fairness metrics are calculated over the whole extracted prediction dataset.

Since, multiclass classification is not supported by the Fairlearn library, I have manually divided the data into 4 Quantile Ranks to bring an effect of multiclass classification.

For option i), after slicing, array of mentioned sensitive feature for each quantile rank are extracted from the data. The predictions and the actual target values for each quantile rank are converted from continuous values to binary values/labels for the Fairlearn metrics to use. The continuous values below average predicted target value are saved as 0 and the rest as 1. Then, for each quantile rank, the Fairlearn demographic\_parity\_ratio metric is used to find out the parity of the mentioned sensitive feature in the data. This metric takes in the predicted target label array, actual target label array and the sensitive feature array as inputs, and gives a value between 0 and 1 as output. The more the value is closer to 1, the fairer the data and hence the model is. Same goes for the Fairlearn equalized\_odds\_ratio. It is a stricter version of demographic\_parity\_ratio metric and gives a value within 0 and 1. These metrics values for all the Quantile Ranges are put in a single dataframe and stored as a csv file called Fairlearn\_results\_2.csv.

For option ii), the Fairlearn demographic\_parity\_ratio and equalized\_odds\_ratio metrics are calculated and along with them, the Fairlearn MetricFrame is also used to evaluate and store precision\_score, recall\_score, f1\_score, selection\_rate and count\_of\_instances for each sensitive label in a matrix form for the whole dataset. The demographic\_parity\_ratio and equalized\_odds\_ratio for the whole database are put in a dataframe and stored as a csv file called Fairlearn\_results\_1.csv. The MetricFrame values are put in another dataframe and stored as a csv file called Fairlearn\_results\_5.csv. The Fairlearn\_results\_4.csv is used to store general idea of Demographic Parity ratio and Equalized Odds ratio and some data analytics values on the data

provided based on the sensitive feature labels.

**ii) Unfairness Mitigation :** In this part, the original dataset selected by the user corresponding to the String filenumber passed to the layer is extracted from the csv file. Then, the continuous valued target column is transformed into binary labels based on the average target value. If values in that column are less than the average value, they are transformed to 0, or else 1. The sensitive feature column is also extracted from the dataset as an array. After that, the categorical features of the dataset are transformed into One-hot encodings using Pipeline inside mak\_column\_transformer method of scikit-learn. Then the data is split into training and test sets similarly like in the ML Model layer. This time, some extra parameters are passed to the train\_test\_split method : The sensitive feature array and stratify=target\_array\_labels, along with training data X and target label array Y. Then, the Fairlearn AdversarialFairnessClassifier is instantiated with the parameters backend="tensorflow", predictor\_model= leaky-relu with 50 layers, adversary\_model = leaky-relu with 3 layers, batch\_size=2<sup>8</sup>, progress\_updates = 0.5 and a random-state. It is the mitigator instance which will help remove the bias in the data and give fair predictions. This instance is fit using the X\_train, Y\_train and sensitive data array training data (Z\_train). The prediction function of the mitigator is used to predict the target values for X\_test and these predictions along with the Z\_test and X\_test are passed to the demographic\_parity\_ratio, equalized\_odds\_ratio and the MetricFrame to evaluate the improvement in their values. These results are then saved as csv files named unfairness\_mitigation\_result\_1.csv and unfairness\_mitigation\_result\_2.csv.

**LangChain using GPT-4 integrated UI layer (Streamlit) :** In this layer, two libraries, LangChain and Streamlit have been combined to create the chatbot. Let us delve into the implementation details for this layer.

The UI has been divided into two Streamlit components: sidebar and main screen. In the sidebar component, a dataset selection option through radio-buttons and context for chat selection options through buttons are available. The context buttons are 'Total Salary Predictions', 'Explanation of Predictions(SHAP)', 'Fairness of Predictions(Fairlearn)', 'Unfairness Mitigation'. These functionalities, which have already been discussed before, serves one at a time as the context for the chatbot displayed on the main screen. The radio-button selection goes in as the database selection for each

of the buttons and afterwards, the buttons, when clicked, turns the respective context indicator positive and invokes the respective layer responsible for the context generation (ML Model, Explainability layer, Fairness layer). After clicking the button, when questions are asked about that context, the respective indicator is retrieved and correspondingly the respective csv files are selected. These are the files containing results of the ML Model, Explainability layer and Fairness layer, which store the results after execution of each layer. Some buttons also display a few results returned from the respective layer, when clicked. Some tips to operate the chatbot have been shared under "How To : " in the sidebar.

Now, let us discuss about the main screen containing the chatbot. When a question is asked to the chatbot, it passes through a query refining functionality which completes the query referring to the previous chat context stored using message component of streamlit.chat and session.state in Streamlit, and passes it over to the LangChain layer [29]. This is implemented using the ChatCompletion component of the openai library. The create() function of this component helps to complete the query by taking in parameters model="gpt-4", temperature=0.7, max\_tokens=100, n=1, messages = system-role, where model is the LLM model being used, temperature helps in managing the variability and range of the LLM responses. It can be in the range 0-2. Higher values like 0.7 will make the response more random. The max\_tokens restrict the length of the response to 100, n corresponds to the number of responses one wants to produce; here we only want 1 response. The messages parameter corresponds to an array of a collection of message objects, where each object contains two properties: "role" (which can be "system", "user", or "assistant") and "content" (representing the text of the message). Here, only the system role has been used with the context being the previous conversations. The "system" role often entails delivering instructions or messages that establish the overall context. It provides instructions for the assistant's conduct during the conversation. The reason why these roles are needed for chat completion is simple. The memory or parameters of language models such as GPT remain static and do not undergo updates during each iteration of conversation. Hence, it is imperative to prompt the model with its prior replies in order to sustain the discussion. Indeed, the model lacks awareness of the fact that it is generating responses based on its own previous output. Instead, its sole purpose is to sustain the conversation, as it is the fundamental aim of language models. The completed query is then displayed on the UI main screen and passed on to the

LangChain layer for response generation using the LLM.

In the next step, a LangChain agent is created using `initialize_agent()` method with the parameters `agent = 'chat-conversational-react-description'`, `tools`, `llm`, `llm_chain = conversation`, `verbose = True`, `max_iterations = 3`, `early_stopping_method = 'generate'`, `memory = Streamlit buffer_memory`, `handle_parsing_errors = True`.

The '`agent`' represents the agent type, which is '`chat-conversational-react-description`' here. This agent is specifically built for use in chatbot conversational contexts. The system use the ReAct framework to choose the appropriate tool to utilise and utilises memory to retain past conversation engagements.

The '`tools`' parameter refers to the set of tools which have been developed specifically for each context corresponding to the UI buttons. These tools have been discussed earlier, namely : `PredictedDatasetTool()`, `ShapExplain-PredictedResultTool()`, `FairlearnFairnessTool()`, `UnfairnessMitigationTool()`, `SinglePredictionTool()`, `SingleExplanationTool()`. The latter two tools are for single target prediction corresponding to chatbot query. They are not connected to any button. The tools connected to buttons have name, description and a `_run` function representing name of the tool, description instructing the LLM when the tool should be used and `_run` function to generate LLM results. Inside the `_run` function extraction of the appropriate context from csv result files as dataframes are carried out at first. Then, these dataframes are fed to LangChain `create_pandas_dataframe_agent` along with the parameters `llm`, `verbose = True`, `agent_type = OPENAI_FUNCTIONS` and `handle_parsing_errors = True`. Among these parameters, `llm`, `verbose = True` and `handle_parsing_errors = True` are common with the `initialize_agent()` function parameters and hence will be discussed in the later paragraphs. The remaining is the `agent_type` which has been selected as the `OPENAI_FUNCTIONS`. The primary objective of the OpenAI Function APIs is to enhance the accuracy and utility of function calls compared to a general text completion or chat API. The `OPENAI_FUNCTIONS` agent type enables the dataframe agent to leverage OpenAI's language model functions for advanced analysis and production of responses from the context dataframes provided to it, rather than just deterministic answers. Pandas dataframe agent, as the name suggests, specializes in analyzing dataframes leveraging a LLM and returning appropriate results for the dataframe related questions. Multiple dataframes can be passed to this agent as context. This agent is created and run with the refined query to interpret the data given and find out correct response.

The SinglePredictionTool(), SingleExplanationTool() are created in a different manner. When the user asks the chatbot to calculate the target value for given feature credentials, the SinglePredictionTool is invoked. The description of the tool tells it to calculate the target from the given credentials which it will accept as comma separated values and also shows an example of the query given by the user. Then it parses the query and passes the credentials to the ML Model where the model instance and the column transformer instance which were previously saved as .sav and .pickle files are used to predict the target value. The Shap explainer instance which was previously saved as a pickle file is also used here to calculate the feature importances for this target value. All the credentials, the prediction and the Shapley values are put in a dataframe and stored as a csv file total\_result.csv to be used for the next question by the user. When the user can asks the chatbot questions regarding the Shap values of the features for this target, the SingleExplanationTool is invoked by the agent. This tool extracts the total\_result.csv and uses a pandas dataframe agent from LangChain to run the query asked by the user and give appropriate answers.

Coming back to the next parameter for the initialize\_agent() method, llm. The llm is the chat-model selected for the chatbot and is created using ChatOpenAI component of LangChain with various parameters like temperature = 0, model\_name='gpt-4', openai\_api\_key and callbacks if needed. The temperature is set as 0, as we want deterministic results from the context without any randomness. A callback is also added for feedback, which will be discussed later.

The memory parameter takes in the Streamlit buffer memory which has already been instantiated with ConversationBufferWindowMemory component of LangChain memory using parameters memory-key, k = 5 and return\_messages = True. The ConversationBufferWindowMemory maintains a chronological record of the interactions occurring during the conversation. It exclusively utilises the most recent K interactions. This can be advantageous for maintaining a dynamic range of the most recent chats, preventing the buffer from becoming excessively huge. Additionally, it is possible to obtain the history in the form of a message list by setting return\_messages as True, which becomes advantageous when utilising this feature with a chat model.

Next is the llm\_chain parameter. Chains are formed by linking together one or more large language models (LLMs) in a coherent manner. A solitary LLM represents the most basic form of a chain. A LLM Chain of type

ConversationChain is a mechanism that facilitates chatting and retrieves relevant information from memory. For this the ConversationChain component of LangChain is instantiated with a memory, prompt, the previously mentioned and instantiated llm (chat-model) and verbose as True. The memory here is the ReadOnlySharedMemory component of LangChain memory which utilizes the Streamlit buffer memory mentioned before as a read only memory so that it cannot be changed while the agent refers to previous conversations. Next parameter is the prompt which is an instance of the ChatPromptTemplate component of LangChain. This instance calls its from\_messages function with the parameters SystemMessagePromptTemplate, HumanMessagePromptTemplate and MessagesPlaceholder from LangChain to create a chat template. SystemMessagePromptTemplate is a message format which uses its from\_template function to utilize a general instruction to the chatbot as a string on how it should behave, to create an instance of its self. The HumanMessagePromptTemplate takes in the user input to create an instance of its self. MessagesPlaceholder takes in the chat-history keyword and validates if it already is instantiated with memory. At the end, verbose, as usual is set as True to monitor the steps taken by the ConversationChain.

Also for the initialize\_agent() function, verbose is set to be True to monitor the intermediate steps the agent takes in order to reach the final answer and max\_iterations puts a check on the number of these steps taken by the agent. The agent can take 3 steps at max in this case. The early\_stopping\_method is set as 'generate' to generate final results within the given number of iterations. Lastly, at times, the LLM faces difficulty in determining the appropriate course of action because of the incorrect formatting of its outputs, which cannot be processed by the output parser. In this scenario, the agent automatically produces errors. However, one can effortlessly manage and manipulate this capability by utilising the handle\_parsing\_errors feature, setting it to be True.

After all the steps, the answer from running a specific tool is redirected to the main agent and is parsed finally making it ready to be displayed to the user. For additional error handling, try-catch blocks are utilized at this part of the implementation.

Once the chatbot displays an answer to the user, a feedback component represented by a thumbs up/down pops up beside the response. When clicked on either of the thumbs, an optional textbox component to give additional written feedback pops up. Once these components are used, and the feedback submitted, it gets saved to the profile created in the Trubrics app for

the experiment. To use Trubrics, one has to create a profile in the app to store feedbacks from the user.

The feedback component for evaluation of user reaction is incorporated seamlessly as it is another Streamlit app. Trubrics uses the LangChain callback functionality in the background implementation while instantiating the LLM using ChatOpenAI to integrate its feedback component TrubricsCallbackHandler. LangChain offers a callbacks system that enables one to integrate with different phases of the LLM application. This is beneficial for the purposes of recording, observing, transmitting etc. The TrubricsCallbackHandler is created using parameters project\_name, tags=[“chat model”], user\_id, some\_metadata, session\_id. It helps to save the user query and chatbot responses corresponding to a particular user in a particular session for a particular project in the Trubrics profile created for the LLM app. The tags refer to the type of LLM app and the some\_metadata parameter helps to add metadata to the query and responses to be stored under the project.

Another Trubrics component called FeedbackCollector is instantiated which uses its st\_feedback() function to collect the user feedback from the UI. This function takes in parameters : component = the project name in the Trubrics app, feedback\_type = “thumbs”, open\_feedback\_label = “[Optional] Provide additional feedback” - the textbox for additional feedback, model=“gpt-4” - model name, prompt\_id= the respective id for each prompt saved using the TrubricsCallbackHandler, key = a key to represent this feedback, align = “flex-end” - alignment of the component. This completes the feedback collection both through the UI and the backend. The user can finally smoothly use this chatbot in its full swing.

## 5 Evaluation

Two datasets have been produced with synthetic data to be used in this experiment.

1. Gender Bias Dataset
2. Uneven Gender Distribution Dataset

The 1<sup>st</sup> one represents a dataset where female employees are being paid less than the male employees due to their gender. The 2<sup>nd</sup> one represents a dataset where the number of female employees is very low.

	Age	Gender	Experience	Education	Interview Score	Test Score	Salary
0	24	f	11	PhD	6	6	6000.0
1	28	m	13	PhD	7	6	6000.0
2	24	f	11	PhD	7	6	6000.0
3	30	m	13	PhD	7	7	6000.0
4	25	f	11	PhD	6	7	6000.0

Figure 12: Example dataset

Each dataset has been evaluated using all the functionalities in the chatbot. Some sample questions have been asked to the chatbot regarding each context to each dataset. The general steps are as follows:-

1. Select the dataset using the radio button.
2. Press the Total Salary Predictions.
3. Ask questions to the chatbot regarding the dataset and its predictions. (Questions will be discussed later)
4. Move on to the next button Explanation of Predictions(SHAP)
5. Ask questions to the chatbot regarding the feature importances with Shap values for the dataset selected to the chatbot. (Questions will be discussed later)
6. Move on to the next button Fairness of Predictions(Fairlearn)
7. Ask questions to the chatbot regarding the displayed values of metrics and fairness of the dataset selected. (Questions will be discussed later)
8. Move on to the next button Unfairness Mitigation.
9. Ask questions to the chatbot regarding the displayed metrics, their values and unfairness mitigation technique of the Fairlearn library with respect to the selected dataset.
10. Ask "how to predict salary" to the chatbot. It returns a template to calculate a single salary prediction using given credentials.
11. This template is modified and sent as a query to the chatbot in order to get a salary prediction.
12. Once the answer is received, other questions about the feature importances with Shap values for the single prediction is asked to the chatbot.

A few users have provided feedbacks for various responses during this evaluation which will be analysed using the Trubrics app.

Numeric answers have been evaluated manually in the test\_file to tally with the LLM results.

The csv files carrying the results of each layer (ML Model, Explainability layer, Fairness layer) look as the following example figures 13, 14, 15, 17 :-

```
1 Age,Gender,Experience,Education,Interview Score,Test Score,Actual Salary,PredictedSalary
2 25,f,7,Ma,6,7,3059.7,2548.5015
3 24,f,7,Ma,6,8,3207.66,2574.2163
4 22,f,6,Ma,8,9,3235.05,2576.671
5 29,f,6,Ma,6,7,2674.79,2608.4927
6 24,f,8,Ma,6,7,3354.95,2627.58
7 21,f,8,Ma,7,8,3054.33,2634.8474
8 25,f,6,Ma,8,8,2902.5,2643.6694
9 24,f,7,Ma,6,9,3376.75,2646.288
10 22,f,9,Ma,6,7,2971.04,2660.3022
```

Figure 13: Predicted salary column joined to the original dataset columns and stored in the csv file named total\_database\_prediction.csv



```

1 Database,Demographic Parity Ratio,Equalized Odds Ratio
2 overall database,0.03525922527364489,0.030429866974573623
3

1 QuantileRank,Demographic Parity Ratio,Equalized Odds Ratio
2 0,1,0,1,0
3 1,0.4428841161198962,0.4081841432225064
4 2,0,0,0,0
5 3,1,0,1,0
6

1 Age,Gender,Experience,Education,Interview Score,Test Score,Actual Salary,PredictedSalary,QuantileRank
2 25,f,7,Ma,6,7,3059.7,2548.5015,0
3 24,f,7,Ma,6,8,3287.66,2574.2163,0
4 22,f,6,Ma,8,9,3235.05,2576.671,0
5 29,f,6,Ma,6,7,2674.79,2608.4927,0

1 Gender,precision_score,recall_score,f1_score,selection_rate,count of instances
2 f,0.32653061224489793,0.13114754098360556,0.18713450292397663,0.03349282296659718,1463.0
3 m,0.9363013690630137,0.9613220815752461,0.948646773074254,0.9499024072869225,1537.0
4

```

Figure 15: Fairness metrics for overall dataset and Quantile ranges stored in csv files named Fairlearn\_results\_1.csv, Fairlearn\_results\_2.csv, Fairlearn\_results\_3.csv, Fairlearn\_results\_5.csv. Fairlearn\_results\_4.csv contains a long String context on the metrics used.

```

Gender mitigated precision_score(Adversarial Mitigation Technique) \
0 f 0.072515
1 m 0.936471

mitigated recall_score(Adversarial Mitigation Technique) \
0 0.529915
1 0.573074

mitigated f1_score(Adversarial Mitigation Technique) \
0 0.127572
1 0.711032

mitigated selection_rate(Adversarial Mitigation Technique) \
0 0.567352
1 0.569324

present count of instances precision_score recall_score f1_score \
0 1507.0 0.326531 0.131148 0.187135
1 1493.0 0.936301 0.961322 0.948647

selection_rate count of instances
0 0.033493 1463.0
1 0.949902 1537.0
Database Demographic Parity Ratio Equalized Odds Ratio \
0 overall database 0.035259 0.030429

Demographic Parity Ratio after Adversarial Mitigation Technique \
0 0.996538

Equalized Odds Ratio after Adversarial Mitigation Technique
0 0.910127

```

Figure 16: Fairness metrics after unfairness mitigation using Adversarial technique stored in csv files named unfairness\_mitigation\_result\_1.csv and unfairness\_mitigation\_result\_2.csv

```
0  Age Gender  Experience Education Interview Score  Test Score \
0  30 f          8 Ma           7 9
0  predicted salary Shap_value_of_Gender_female Shap_value_of_Gender_male \
0  3124.130127 -234.025549 -343.189936
0  Shap_value_of_Education_Masters Shap_value_of_Education_PhD \
0  -52.064317 -179.971957
0  Shap_value_of_Age Shap_value_of_Experience Shap_value_of_Interview Score \
0  126.090362 -260.905892 -31.191647
0  Shap_value_of_Test Score
0  45.405274
```

Figure 17: Salary prediction for a single data-point given the credentials are stored along with its Shapley values of features in total\_result.csv

## 5.1 Gender Bias Dataset

### 5.1.1 Total Salary Predictions Q/A :

#### 1. How many columns are there in the database?

Manual result : The number of columns is calculated using the code

```
len (data . axes [1])
```

which gives the result "No. of columns 8" . len() function calculates the length of its input. data is the dataframe and axes[1] represents the columns.

LLM result : It calculates the number of columns using

```
df . shape [1]
```

which gives the result "The total number of columns in the given dataset is 8." The df represents the dataframe and shape of axis 1 returns the total number of columns.

```
> Entering new AgentExecutor chain...
2024-01-04 16:37:04.615 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```json
{
  "action": "Dataset QA",
  "action_input": "What is the total number of columns in the given dataset?"
}..
entered

> Entering new AgentExecutor chain...
2024-01-04 16:37:08.827 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
Invoking: `python_repl_ast` with `{'query': 'df.shape[1]'}`

2024-01-04 16:37:11.552 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
8The total number of columns in the given dataset is 8.

> Finished chain.

Observation: The total number of columns in the given dataset is 8.
Thought:2024-01-04 16:37:17.156 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```json
{
  "action": "Final Answer",
  "action_input": "The total number of columns in the given dataset is 8."
}..
```
> Finished chain.
the response isThe total number of columns in the given dataset is 8.
```

Figure 18: Dataset 1: Number of columns

Here we can see a chain of actions which are produced by the LangChain AgentExecutor. When the agent's run function on the query is called, the AgentExecutor starts executing the query as input in the chain. The 1<sup>st</sup> action has the query as input. 2<sup>nd</sup> step invokes the corresponding Python function for the calculation and 3<sup>rd</sup> step gives the Final result. All the user prompts and the intermediate steps are saved to Trubrics. We also get the final result within 3 steps as had been mentioned in the agent initialization.

## 2. What is the max value of numeric columns in the database?

Manual result : The result is derived using the code

```
DataFrame .max( numeric_only=True )
```

which gives the result

```
"Age 35.000
Experience 15.000
Interview Score 10.000
Test Score 10.000
Actual Salary 6000.000
PredictedSalary 5910.029"
```

This function calculates the maximum values of numeric data in the dataframe.

LLM result : It calculates the maximum values using the same function as manual. The result is hence same : "The maximum value of each numeric column in your dataset is as follows: Age is 35, Experience is 15, Interview Score is 10, Test Score is 10, Actual Salary is 6000, and Predicted Salary is 5910.029."

```

> Entering new AgentExecutor chain...
2024-01-04 17:36:23.168 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```
{
  "action": "Dataset Q4",
  "action_input": "What is the maximum value of each numeric column in the dataset?"
}
```
entered

> Entering new AgentExecutor chain...
2024-01-04 17:36:26.649 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.

Invoking: `python_repl_last` with `{'query': 'df.max(numeric_only=True)'}`

Age      35.000
Experience 15.000
Interview Score 10.000
Test Score 10.000
Actual Salary 6000.000
PredictedSalary 5910.029
dtype: float642024-01-04 17:36:36.055 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
s.
The maximum value of each numeric column in the dataset is as follows:

- Age: 35
- Experience: 15
- Interview Score: 10
- Test Score: 10
- Actual Salary: 6000
- Predicted Salary: 5910.029

> Finished chain.

Observation: The maximum value of each numeric column in the dataset is as follows:

- Age: 35
- Experience: 15
- Interview Score: 10
- Test Score: 10
- Actual Salary: 6000
- Predicted Salary: 5910.029
Thought:2024-01-04 17:36:45.785 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```
{
  "action": "Final Answer",
  "action_input": "The maximum value of each numeric column in your dataset is as follows: Age is 35, Experience is 15, Interview Score is 10, Test Score is 10, Actual Salary is 6000, and Predicted Salary is 5910.029."
}
```

> Finished chain.
the response isThe maximum value of each numeric column in your dataset is as follows: Age is 35, Experience is 15, Interview Score is 10, Test Score is 10, Actual Salary is 6000, and Predicted Salary is 5910.029.

```

Figure 19: Dataset 1: Max of numeric columns

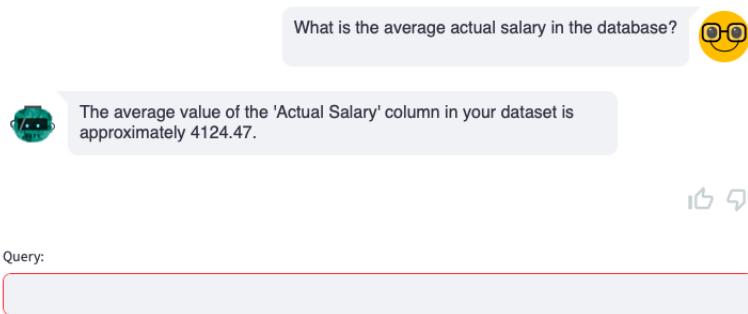
### 3. What is the average actual salary in the database?

Manual result : The result is derived using the code

```
df[ 'Actual Salary' ].mean()
```

where df is the dataframe "Actual Salary" is the column name and mean() function gives average of the values in this column. Result : avg act sal: 4124.469103333333

LLM result : The result is calculated using the same function in the same way, and the result is displayed as follows:-



#### Refined Query:

What is the average value of the Actual Salary column in the dataset?

Figure 20: Dataset 1: Average Actual Salary

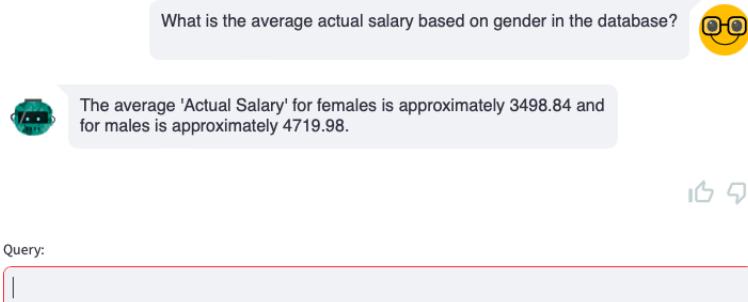
### 4. What is the average actual salary based on gender in the database?

Manual result : The result is derived using the code

```
df.groupby( 'Gender' )[ 'Actual Salary' ].mean()
```

where df is the dataframe, grouby() function separates the gender labels and mean() gives the average actual salary for each label. Result : "sal based on gen: Gender  
f 3498.839665  
m 4719.977150"

LLM result : LLM uses the same function like manual and hence produces the same result as follow :-



#### Refined Query:

What is the average 'Actual Salary' for each gender in the database?

Figure 21: Dataset 1: Average Actual Salary for each Gender label

#### 5. How many females are there in the database?

Manual result : The function

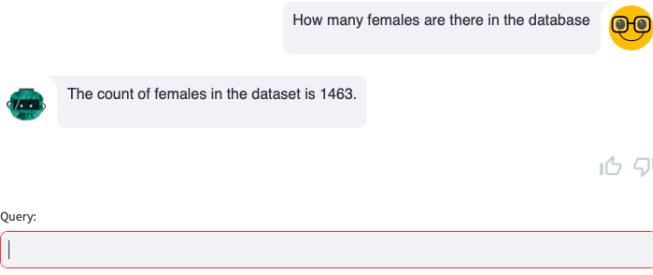
```
len(data[data['Gender'] == 'f'])
```

gives the number of females in the dataset, where data is the dataframe, the nested part subsets the dataframe based on female gender and len gives the total number of contents of its input. Result : "no. of female : 1463"

LLM result : The number of females is calculated by the LLM using the code

```
df[df['Gender'] == 'f'].shape[0]
```

where the shape[0] counts the number of rows, 0 being the axis index. The result as follows is same :-



Query:

|

**Refined Query:**

What is the count of females in the dataset?

Figure 22: Dataset 1: Number of females in the dataset

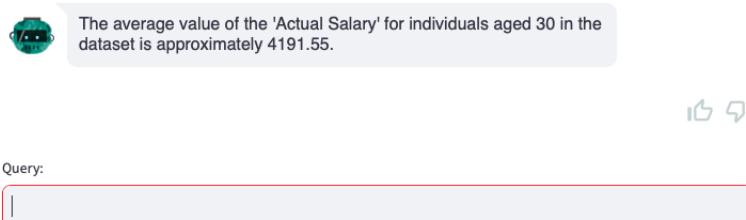
**6. What is the average actual salary for 30 year olds in the database?**

Manual result : The code

```
data [ data [ 'Age' ] == 30 ][ 'Actual Salary' ].mean()
```

gives the result "avg act sal for 30 yr olds : 4191.5508794788275"  
where data is the dataframe, inner part = subset of the data with age 30,  
and then mean() calculates the average actual slary for the subset.

LLM result : LLM uses the same function to calculate the result, and produces the same result as in figure 23:-



Query:

|

**Refined Query:**

What is the average value of the 'Actual Salary' column for individuals aged 30 in the dataset?

Figure 23: Dataset 1: Average Actual salary for 30 year olds

## 7. What is the average predicted salary for 32 year old phd holders in the database?

Manual result : The code

```
data[(data['Age'] == 32)
& (data['Education'] == 'PhD')]
['PredictedSalary'].mean()
```

gives the result "avg sal for 32 yr old phd holders : 4548.882857777778". Here data is the dataframe, subset with age = 32 has been created and condition that these people should also hold a Phd has been added. On this, mean/average of the predicted salary column has been calculated.

LLM result : The same Python code is used by the LLM to calculate the result which is same as in figure 24:-



### Refined Query:

What is the average value of the 'Predicted Salary' for individuals aged 32 with a PhD in the dataset?

Figure 24: Dataset 1: Average Predicted salary for 32 year olds holding Phd

### 5.1.2 Explanation of Predictions(SHAP) Q/A

#### 1. What are the most important features of the dataset according to Shap values?

Manual result : Following code snippet is used to calculate the results from the dataframe containing the Shap results.

```
shap_res=[i for i in data_exp if i.startswith('Shap')]
avg_data=data_exp[shap_res].abs().mean()
```

shap\_res is the dataframe containing the Shap columns. It is extracted using the startswith function as the columns start with the String "Shap". Shap global values are absolute mean of the local Shapley values for each data point.

Result:-

|                                 |            |
|---------------------------------|------------|
| Shap_value_of_Gender_female     | 245.123174 |
| Shap_value_of_Gender_male       | 362.993008 |
| Shap_value_of_Education_Masters | 48.004670  |
| Shap_value_of_Education_PhD     | 182.336800 |
| Shap_value_of_Age               | 93.573061  |
| Shap_value_of_Experience        | 155.736647 |
| Shap_value_of_Interview Score   | 55.640577  |
| Shap_value_of_Test Score        | 56.007937  |

LLM result : LLM uses the same idea to calculate the Shap values. The difference is that it uses startwith() function with the String "Shap\_value" and sorts the mean absolute values in descending order to get the most important features. Figure 25 show the LLM computation.

```

> Entering new AgentExecutor chain...
2024-01-04 20:37:14.454 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```json
{
  "action": "SHAP Explainability QA",
  "action_input": "What are the most important features in the dataset based on SHAP values?"
}..
entered

> Entering new AgentExecutor chain...
2024-01-04 20:37:27.888 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.

Invoking: `python_repl_ast` with `{'query': "shap_columns = [col for col in df.columns if 'Shap_value' in col]\nmean_abs_shap_values = df[shap_columns].abs().mean()\nsort_values(ascending=False)\nmean_abs_shap_values"}'
responded: The importance of features in a dataset based on SHAP values can be determined by looking at the absolute values of the SHAP values. The higher the absolute SHAP value, the more important the feature is.

To find the most important features, we can calculate the mean absolute SHAP value for each feature across all instances in the dataset. Let's calculate this.

Shap_value_of_Gender_male      362.993008
Shap_value_of_Gender_female    245.123174
Shap_value_of_Education_PhD    182.336800
Shap_value_of_Experience       155.736647
Shap_value_of_Age              93.573861
Shap_value_of_Test_Score       56.007937
Shap_value_of_Interview_Score  55.640577
Shap_value_of_Education_Masters 48.004670
dtype: float64
Thought:2024-01-04 20:37:34.812 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.

Based on the mean absolute SHAP values, the most important features in the dataset are:
1. Gender (male)
2. Gender (female)
3. Education (PhD)
4. Experience
5. Age

These features have the highest impact on the model's output according to the SHAP values.

> Finished chain.

Observation: Based on the mean absolute SHAP values, the most important features in the dataset are:
1. Gender (male)
2. Gender (female)
3. Education (PhD)
4. Experience
5. Age

These features have the highest impact on the model's output according to the SHAP values.
Thought:2024-01-04 20:37:38.989 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```json
{
  "action": "Final Answer",
  "action_input": "Based on the analysis, the most important features in the dataset that impact the model's output are, in order: Gender (male), Gender (female), Education (PhD), Experience, and Age."
}..
Thought:2024-01-04 20:37:40.000 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.

> Finished chain.
the response isBased on the analysis, the most important features in the dataset that impact the model's output ar e, in order: Gender (male), Gender (female), Education (PhD), Experience, and Age.
```

```

Figure 25: Dataset 1: Most important features (Shap)

## 2. What are the most important features of the 10th observation of the database according to Shap values? Use whole sentence

Manual result : The shap values of the 10th observation is calculated with the following code

```
shap_res=[i for i in data_exp if i.startswith('Shap')]
data_exp[shap_res].iloc[9]
```

where iloc[9] corresponds to the 10th observation. Result :

shap 10 :

|                                 |             |
|---------------------------------|-------------|
| Shap_value_of_Gender_female     | -234.679867 |
| Shap_value_of_Gender_male       | -347.527972 |
| Shap_value_of_Education_Masters | -48.490488  |
| Shap_value_of_Education_PhD     | -184.182224 |
| Shap_value_of_Age               | 67.659104   |
| Shap_value_of_Experience        | -253.023944 |
| Shap_value_of_Interview Score   | -37.192898  |
| Shap_value_of_Test Score        | 43.601515   |

LLM result : The LLM at first retrieves the 10<sup>th</sup> observation using the same function from both the predictions dataframe and the Shap dataframe, using 2 different corresponding tools. Then it gives an analysis of the facts it encounters in these 2 dataframes in the figure 26

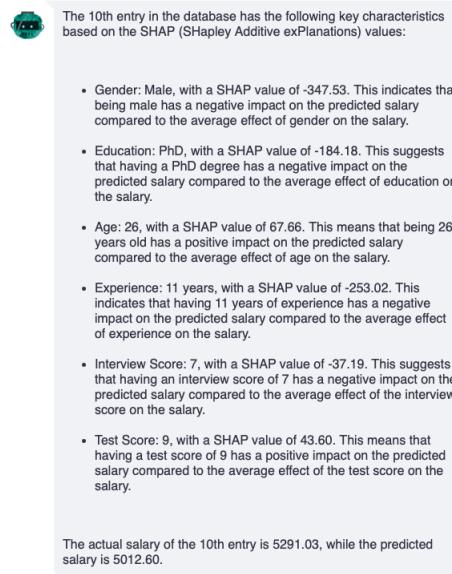


Figure 26: Dataset 1: Most important features (Shap) for the 10<sup>th</sup> observation

### 5.1.3 Fairness of Predictions(Fairlearn) Q/A

When the button is pressed the following fairness metrics values are displayed:-

|   | Database         | Demographic Parity Ratio | Equalized Odds Ratio |
|---|------------------|--------------------------|----------------------|
| 0 | overall database | 0.0353                   | 0.0304               |
|   |                  |                          |                      |

Figure 27: Dataset 1: Fairlearn fairness metrics

#### 1. What can be interpreted from the Fairlearn results?

Manual results are not possible for this type of question as it is the LLM's interpretation based on the given contexts which will not have any distinct numeric answer.

LLM result : The LLM though process analyzes each and every dataframe and interprets that the model is biased over Gender. Males are preferred over females according to this dataset. The response is clear and understandable by a non technical person. Detailed questions about the metrics used for fairness evaluation can be asked to better understand the response. The following figures display the thought process and final results of the LLM:-

```
> Entering new AgentExecutor chain...
2024-01-05 01:47:35.306 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```json
{
  "action": "Fairlearn Fairness QA",
  "action_input": "What insights can be derived from the Fairlearn analysis of the dataset?"
}.. entered

> Entering new AgentExecutor chain...
2024-01-05 01:48:07.656 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
The Fairlearn analysis provides several insights into the fairness of the machine learning model used on the dataset:
1. **Demographic Parity and Equalized Odds**: The demographic parity ratio and equalized odds ratio are used to measure the fairness of the model. A ratio of 1 indicates perfect fairness, meaning all groups have the same selection rate or true positive and false positive rates. In this case, the overall database has a demographic parity ratio of 0.035 and an equalized odds ratio of 0.030, indicating a significant disparity between groups. However, for certain quantile ranks (0 and 3), these ratios are 1, indicating fairness within these specific groups.
2. **Gender Disparity**: The analysis also reveals a disparity between genders. The average salary for females is significantly lower than that for males (3478.23 for females vs. 4690.32 for males). Furthermore, the precision, recall, and F1 scores are much lower for females than for males, indicating that the model performs better in predicting salaries for males.
3. **Quantile Ranks**: The dataset has been divided into 4 quantile ranges. The demographic parity ratio and equalized odds ratio vary significantly across these ranges. For instance, range 2 has a ratio of 0 for both metrics, indicating a complete lack of fairness.
4. **Model Performance**: The precision, recall, and F1 scores provide insights into the model's performance. For instance, the model has a high precision and recall for males, indicating that it is good at predicting salaries for this group. However, the scores are significantly lower for females, suggesting that the model's performance is poorer for this group.
In conclusion, the Fairlearn analysis suggests that the model may be biased, performing better for certain groups (e.g., males) and specific quantile ranks. This could potentially lead to unfair outcomes, such as lower salary predictions for females.
> Finished chain.
```

Figure 28: Dataset 1: Thought process of fairness analysis

What can be interpreted from the Fairlearn results of the dataset?



The analysis of the dataset reveals several insights into the fairness of the machine learning model used. It was found that there is a significant disparity between different groups in the dataset. For instance, the average salary for females is significantly lower than that for males. Furthermore, the model's performance, as measured by precision, recall, and F1 scores, is much better for males than for females. The dataset has been divided into different quantile ranges, and the fairness metrics vary significantly across these ranges. In conclusion, these findings suggest that the model may be biased, performing better for certain groups and specific quantile ranks. This could potentially lead to unfair outcomes, such as lower salary predictions for females.

Query:

**Refined Query:**

What insights can be derived from the Fairlearn analysis of the dataset?

Figure 29: Dataset 1: Display of final result

#### 5.1.4 Unfairness Mitigation Q/A

##### 1. What can be inferred from the unfairness mitigation results?

LLM result : The LLM analyses the context provided and provides a reasonable interpretation of the unfairness mitigation results. The results have improved to a great extent using the Adversarial Mitigation Technique. The model has turned out to be fair and can give fair prediction for the data provided. The thought process and final result of the LLM is as follows:-

```
> Entering new AgentExecutor chain...
2024-01-05 03:37:36.825 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
  json
{
  "action": "Unfairness Mitigation QA",
  "action_input": "What conclusions can be drawn from the results of the unfairness mitigation methods?"
}, entered

> Entering new AgentExecutor chain...
2024-01-05 03:38:07.218 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
From the results, it can be concluded that the Adversarial Mitigation Technique has significantly improved the fairness of the model.

Before mitigation, the Demographic Parity Ratio and Equalized Odds Ratio were 0.0352592 and 0.0304299 respectively. These values are far from 1, indicating a high level of bias in the model.

After applying the Adversarial Mitigation Technique, the Demographic Parity Ratio improved to 0.996538 and the Equalized Odds Ratio to 0.910127. These values are much closer to 1, indicating that the model is now more fair and less biased.

In terms of gender, the mitigation technique has improved the precision, recall, and f1 scores for both genders. For females, the precision score improved from 0.326531 to 0.0725146, the recall score from 0.131148 to 0.529915, and the f1 score from 0.187135 to 0.127572. For males, the precision score improved from 0.936301 to 0.936471, the recall score from 0.961322 to 0.573074, and the f1 score from 0.948647 to 0.711032.

However, it's important to note that while the mitigation technique has improved fairness, it has also resulted in a decrease in some of the performance metrics. This is a common trade-off when addressing bias in machine learning models.
```

Figure 30: Dataset 1: Thought process of LLM

What can be inferred from the unfairness mitigation results? 

 The results indicate that the Adversarial Mitigation Technique has significantly improved the fairness of the model. Initially, the Demographic Parity Ratio and Equalized Odds Ratio were 0.0352592 and 0.0304299 respectively, indicating a high level of bias. After applying the mitigation technique, these values improved to 0.996538 and 0.910127 respectively, indicating a more fair and less biased model. In terms of gender, the mitigation technique improved the precision, recall, and f1 scores for both genders. However, it's important to note that while the mitigation technique has improved fairness, it has also resulted in a decrease in some of the performance metrics. This is a common trade-off when addressing bias in machine learning models.

Query:

#### Refined Query:

What conclusions can be drawn from the results of the unfairness mitigation methods?

Figure 31: Dataset 1: Display of final result

Follow up questions on the metrics used to evaluate the mitigated results and how they are related to the Fairlearn library can be asked to the chatbot to understand the debiasing technique in a better manner.

#### 5.1.5 Single Prediction Q/A

##### 1. How to predict salary?

LLM result : The LLM reaches the ML Model through function call to get the answer. The thought process and final responses regarding the single prediction and the Shap feature importances of the data-point are displayed below:-

```
> Entering new AgentExecutor chain...
2024-01-05 04:06:57.223 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```json
{
  "action": "Single Prediction QA",
  "action_input": "30,f,7,Ma,7,9"
}
1/1 [=====] - 0s 261ms/step
result prediction: tf.Tensor([[2998.6948]], shape=(1, 1), dtype=float32)
[2998.6948]
1/1 [=====] - 0s 88ms/step
794/794 [=====] - 1s 1ms/step
| 0/1 [00:00<?, ?it/s]
100%|██████████| 1/1 [00:01<00:00, 1.80s/it]
shape of shapley value array (1, 1, 8)
shape of swapped shapley value array (1, 8, 1)
  Age Gender Experience Education Interview Score Test Score \
0   30         f           7           Ma          7            9 \
predicted salary Shap_value_of_Gender_female Shap_value_of_Gender_male \
0   2998.694824           -234.025516           -343.189933
Shap_value_of_Education_Masters Shap_value_of_Education_PhD \
0   -52.064284             -179.971921
Shap_value_of_Age Shap_value_of_Experience Shap_value_of_Interview Score \
0   126.090407            -386.34139             -31.191626
Shap_value_of_Test Score
0   45.405298

Observation: [2998.6948]
Thought: 2024-01-05 04:07:07.111 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```json
{
  "action": "Final Answer",
  "action_input": "The predicted salary for a 30-year-old female with a Master's degree, 7 years of experience, an interview score of 7, and a test score of 9 is approximately $2998.69."
}..
> Finished chain.
```

Figure 32: Dataset 1: Thought process for single target prediction

```
> Entering new AgentExecutor chain...
2024-01-05 04:08:20.519 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
```json
{
  "action": "Single SHAP Explainability QA",
  "action_input": "What are the most significant features for the single salary prediction based on SHAP values according to our knowledge base?"
}..
entered

> Entering new AgentExecutor chain...
2024-01-05 04:08:40.609 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
The most significant features for the single salary prediction based on SHAP values are the ones with the highest absolute SHAP values. In this case, they are:
1. Shap_value_of_Experience: -386.341
2. Shap_value_of_Gender_male: -343.19
3. Shap_value_of_Gender_female: -234.026
4. Shap_value_of_Education_PhD: -179.972
5. Shap_value_of_Age: 126.09

Please note that the sign of the SHAP value doesn't indicate the feature's importance. It indicates the direction of the relationship between the feature and the target variable. A positive SHAP value means that the feature contributes to increasing the predicted value, while a negative SHAP value means that the feature contributes to decreasing the predicted value.

> Finished chain.
```

Figure 33: Dataset 1: Shapley value of features analysed

how to predict salary? 

For a single prediction: max age=35,gender=(m,f),max experience=15,education=(PhD,Ma),max interview score=10,max test score=10. Example question template='Get single prediction for 32,'l',7,'Ma',8,9. dont change the these values'

Get single prediction for 30,'l',7,'Ma',7,9. don't change the these values 

The predicted salary for a 30-year-old female with a Master's degree, 7 years of experience, an interview score of 7, and a test score of 9 is approximately \$2998.69.

what are the most important features for the single prediction according to shap values? 

The most significant features for the single salary prediction, based on SHAP values, are Experience, Gender (both male and female), Education (specifically PhD level), and Age. The sign of the SHAP value indicates the direction of the relationship between the feature and the target variable. A positive SHAP value means that the feature contributes to increasing the predicted value, while a negative SHAP value means that the feature contributes to decreasing the predicted value.

Figure 34: Dataset 1: Display of final result

## 5.2 Uneven Gender Distribution Dataset

### 5.2.1 Total Salary Predictions Q/A

The same questions are asked after selecting this database.

LLM results : The LLM thought processes are same like the previous dataset. The final results are in the figures 35, 36:-

How many columns are there in the dataset?

The dataset used for salary prediction and bias mitigation consists of 8 columns.

What is the max value of numeric columns in the selected dataset?

The maximum values for each numeric column in the salary prediction and bias mitigation dataset are: Age is 35, Experience is 15, Interview Score is 10, Test Score is 10, Actual Salary is 6000.0, and Predicted Salary is 5198.101.

What is the average actual salary in this dataset?

The mean value of the 'Actual Salary' column in the salary prediction and bias mitigation dataset is approximately 4124.47.

What is the average actual salary based on gender in the dataset?

The average actual salary for each gender in the salary prediction and bias mitigation dataset is approximately 4141.10 for females and approximately 4120.37 for males.

Figure 35: Dataset 2: Database related questions

Manual results : The manual results are produced using the code snippet

```
no_col=len(data.axes[1])
print("No. of columns",no_col)
print("numeric max: ",data.max(numeric_only=True))
print("avg act sal: ",data['Actual Salary'].mean())
print("sal based on gen: ",data.groupby('Gender')
      ['Actual Salary'].mean())
print("no. of female : ",len(data[data['Gender'] == 'f']))
print("avg act sal for 30 yr olds : ",
      data[data['Age'] == 30]['Actual Salary'].mean())
print("avg sal for 32 yr old phd holders : ",
      data[(data['Age'] == 32) & (data['Education'] == 'PhD')]['PredictedSalary'].mean())
.
```

The Manual Results are :-

How many females are there in the dataset? 

The count of female entries in the salary prediction and bias mitigation dataset is 593. 

What is the average actual salary for 30 year olds in the dataset? 

The average actual salary for individuals who are 30 years old in the salary prediction and bias mitigation dataset is approximately 4191.55. 

What is the average predicted salary for 32 year old phd holders in the dataset? 

The average predicted salary for 32-year-old individuals with a PhD in the salary prediction and bias mitigation dataset is approximately 4538.75. 

! ↴

Query:

### Refined Query:

What is the average predicted salary for 32-year-old individuals with a PhD in the salary prediction and bias mitigation dataset?

Figure 36: Dataset 2: Database related questions

```
No. of columns 8
numeric max: Age           35.000
Experience      15.000
Interview Score 10.000
Test Score      10.000
Actual Salary   6000.000
PredictedSalary 5198.101
dtype: float64
avg act sal: 4124.469103333333
sal based on gen: Gender
f 4141.097352
m 4120.372489
Name: Actual Salary, dtype: float64
no. of female : 593
avg act sal for 30 yr olds : 4191.550879478827
avg sal for 32 yr old phd holders : 4538.7525911111115
```

Figure 37: Dataset 2: Manually got results for Database related questions

### 5.2.2 Explanation of Predictions(SHAP) Q/A

The same questions about Shapley values of the dataset features are asked for this dataset.

LLM results : The results are calculated in the same way as in dataset 1. The final results are in the following figures :

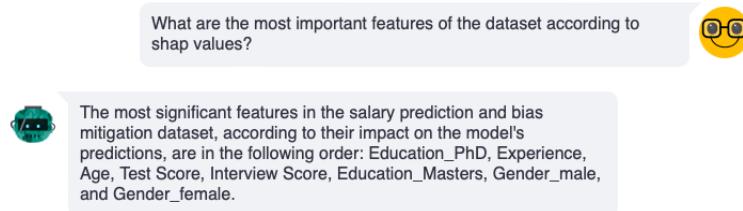


Figure 38: Dataset 2: Shap Q/A

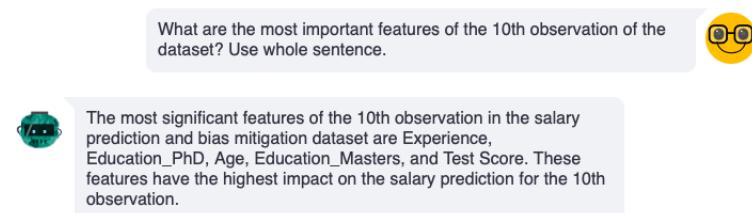


Figure 39: Dataset 2: Shap Q/A

Manual results : The corresponding manual results are produced using the code snippet

```
shap_res=[i for i in data_exp if i.startswith('Shap')]
avg_data=data_exp[shap_res].abs().mean()
print(avg_data)
print("shap for 10th observation :",
      data_exp[shap_res].iloc[9])
```

. The Results are as follows:-

global Shap values for features :	Shap_value_of_Gender_female	26.364051
Shap_value_of_Gender_male	29.812294	
Shap_value_of_Education_Masters	49.938330	
Shap_value_of_Education_PhD	129.795471	
Shap_value_of_Age	123.269093	
Shap_value_of_Experience	124.608437	
Shap_value_of_Interview_Score	53.523211	
Shap_value_of_Test_Score	55.110949	
dtype:	float64	
shap for 10th observation :	Shap_value_of_Gender_female	-16.863794
Shap_value_of_Gender_male	19.069456	
Shap_value_of_Education_Masters	-50.443741	
Shap_value_of_Education_PhD	-131.109084	
Shap_value_of_Age	89.131129	
Shap_value_of_Experience	-202.450260	
Shap_value_of_Interview_Score	-35.777556	
Shap_value_of_Test_Score	42.903201	

Figure 40: Dataset 2: Shap results for manual process

### 5.2.3 Fairness of Predictions(Fairlearn) Q/A

LLM Response : The response is generated by the LLM using the various fairness metrics results given as dataframe contexts. The thought process and the final result are as follows:-

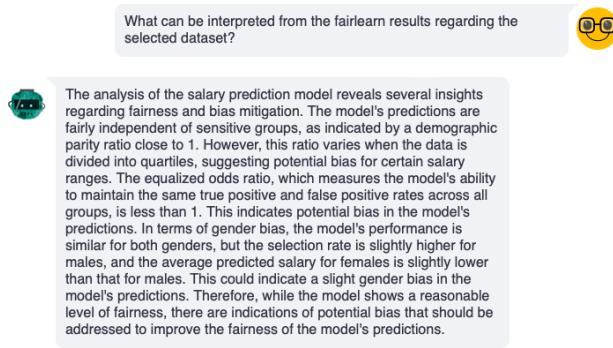


Figure 41: Dataset 2: Final result for fairness interpretation

```
> Entering new AgentExecutor chain...
2024-01-05 05:28:13.109 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
The Fairlearn results provide several insights into the fairness of the salary prediction model:

1. **Demographic Parity Ratio**: This ratio measures the model's ability to make predictions that are independent of the influence by sensitive groups. A ratio of 1 means all groups have the same selection rate. The overall database has a demographic parity ratio of 0.957, which is close to 1, indicating that the model's predictions are fairly independent of the sensitive groups. However, when the data is divided into quartiles, the demographic parity ratio varies from 0.904 to 0.991. This suggests that there may be some bias in the model's predictions for certain salary ranges.

2. **Equalized Odds Ratio**: This ratio ensures that the model's predictions are independent of sensitive groups and that all groups in the dataset have the same true positive and false positive rates. The overall database has an equalized odds ratio of 0.917, which is less than 1, indicating that there may be some bias in the model's predictions. The equalized odds ratio for the quartiles ranges from 0.827 to 0.959, suggesting that the model's predictions may be more biased for certain salary ranges.

3. **Gender Bias**: The precision, recall, and F1 scores for males and females are quite similar, suggesting that the model's performance is similar for both genders. However, the selection rate is slightly higher for males than for females (0.500 vs. 0.478), indicating that the model may be slightly biased towards predicting higher salaries for males.

4. **Salary Predictions**: The average predicted salary for females is slightly lower than that for males. This could be a sign of gender bias in the model's predictions.

In conclusion, while the model shows a reasonable level of fairness in its predictions, there are indications of potential bias, particularly in relation to gender and certain salary ranges. These issues should be addressed to improve the fairness of the model's predictions.

> Finished chain.
```

Figure 42: Dataset 2: LLM Thought process for fairness interpretations

It shows that even if the number of females are less in this dataset, the dataset is not that much biased towards the males.

#### 5.2.4 Unfairness Mitigation Q/A

LLM response : The same question is asked to the chatbot given the fairness metrics values after unfairness mitigation using Adversarial debiasing method. The thought process and final result are as follows:

```
> Entering new AgentExecutor chain...
2024-01-05 05:46:09.904 | INFO    | trubrics.platform:log_prompt:102 - User prompt saved to Trubrics.
From the given data, we can draw the following insights regarding fairness and bias mitigation:

1. The Adversarial Mitigation Technique seems to have improved the fairness of the model. This is evident from the increase in the Demographic Parity Ratio and Equalized Odds Ratio after applying the technique. The Demographic Parity Ratio increased from 0.957444 to 0.97459, and the Equalized Odds Ratio increased from 0.917938 to 0.924571. Both these metrics are closer to 1 after the mitigation, indicating a more fair model.

2. The mitigation technique has slightly reduced the precision score for both genders but has increased the recall score. This suggests that the technique has reduced the model's bias towards predicting a particular class, leading to a more balanced prediction.

3. The selection rate, which indicates the proportion of instances that are predicted as positive, is more balanced across genders after applying the mitigation technique. This is another indication of reduced bias.

4. The count of instances for both genders remains the same before and after the mitigation, indicating that the mitigation technique does not affect the distribution of the data.

In conclusion, the Adversarial Mitigation Technique appears to have successfully reduced bias in the model, leading to more fair and balanced predictions.
```

Figure 43: Dataset 2: LLM Thought process for unfairness mitigation results interpretations

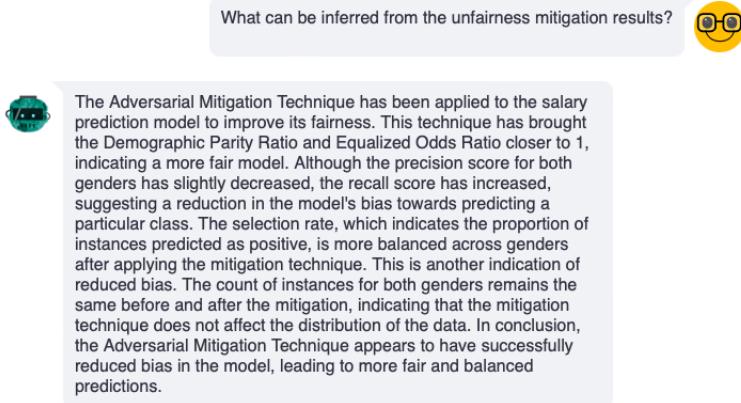


Figure 44: Dataset 2: Final response after unfairness mitigation

Even though the final result speaks highly of the mitigation technique, there was not much to be done with the data, as it was not much biased, as can be seen in the thought process values. The count of instances for females also remain less than males even after mitigation. Hence, it can be inferred that this mitigation technique has been very less effective on this dataset.

To find out if balancing the number of instances affect the fairness metrics and other data analysis metrics too, a test unfairness mitigation approach using SMOTE (Synthetic Minority Oversampling Technique) algorithm has been implemented.

SMOTE is an algorithm that enhances data by generating artificial data points derived from the actual data points. SMOTE can be seen as a more sophisticated kind of oversampling or as a dedicated technique for data augmentation. A benefit of SMOTE is that it generates synthetic data points that are distinct from the original data points, rather than producing duplicates.

SMOTE from imbalanced-learn library has been used to implement this solution for this specific dataset. At first, the dataset is extracted from the corresponding csv and the target variable is set as 'Gender'. The target is then converted to binary labels to feed the SMOTE algorithm. SMOTE generates synthetic data for the Gender label which is under-represented. This data is then joined to the original dataset and the target is now set as

'Salary'. The train-test split is performed on this newly modified data. Then the ML Model saved as .sav file is extracted and the newly modified training data is used to train the model again. After that the model is used to generate prediction for the new testing data. The fairness and other metrics are calculated based on these predictions. Below are the metrics values for the original dataset, the dataset after Adversarial debiasing mitigation and the dataset after SMOTE unfairness mitigation.

```
Value of demographic parity ratio for overall dataset: 0.98
Value of equal odds ratio: 0.98
precision_score recall_score f1_score selection_rate \
Gender
f          0.601010    0.574879  0.587654    0.502113
m          0.578203    0.573906  0.576046    0.493229

count of instances
Gender
f            2366.0
m            2437.0
```

Figure 45: Dataset 2: Fairness metrics after SMOTE Resampling

```
Gender mitigated precision_score(Adversarial Mitigation Technique) \
0      f          0.569930
1      m          0.503836

mitigated recall_score(Adversarial Mitigation Technique) \
0                  0.53268
1                  0.49250

mitigated f1_score(Adversarial Mitigation Technique) \
0                  0.550676
1                  0.498104

mitigated selection_rate(Adversarial Mitigation Technique) \
0                  0.496528
1                  0.483911

present count of instances  precision_score  recall_score  f1_score \
0           576.0        0.595070    0.524845  0.557756
1         2424.0        0.605482    0.571765  0.588140

selection_rate count of instances
0      0.478921        593.0
1      0.500208        2407.0
```

Figure 46: Dataset 2: Data analysis metrics after unfairness mitigation

```
Database Demographic Parity Ratio Equalized Odds Ratio \
0 overall database          0.957444        0.917938

Demographic Parity Ratio after Adversarial Mitigation Technique \
0                                0.97459

Equalized Odds Ratio after Adversarial Mitigation Technique
0                                0.924571
```

Figure 47: Dataset 2: Fairness metrics after unfairness mitigation

It can be clearly seen that the SMOTE resampling gives better scores for most of the metrics and some metrics values stay almost the same. Overall, for unbalanced datasets, SMOTE resampling can be suggested to be used on the dataset before training the ML Model, to produce unbiased target predictions.

### 5.2.5 User Feedback Evaluation

Trubrics has been used to store the user feedback. The dashboard looks like the following after feedback collection.

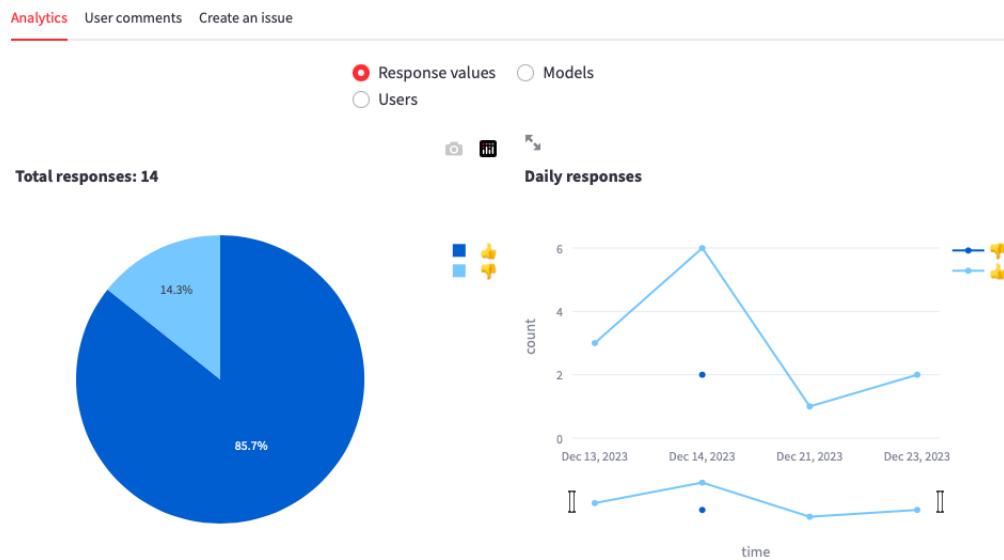
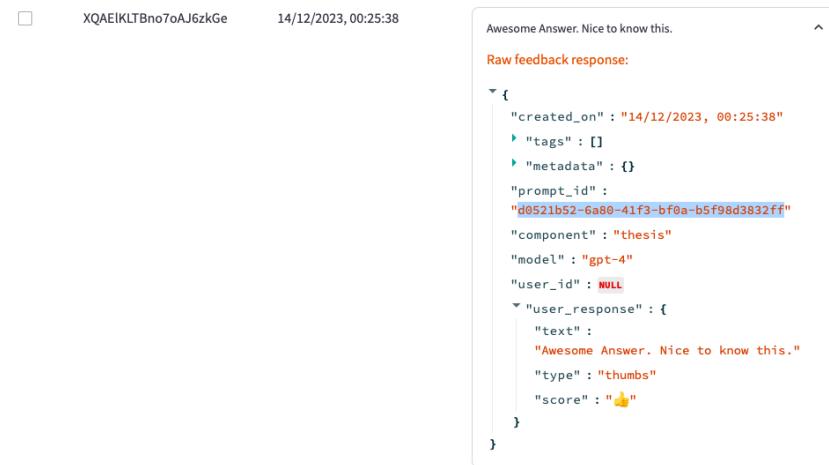


Figure 48: Trubrics Feedback Dashboard

The additional text feedbacks look like the following.

<input checked="" type="checkbox"/>	ID	Created on	None
<input type="checkbox"/>	Yf1LQSNKoad1baW4L5k6	23/12/2023, 18:55:57	this is a better answer!
<input type="checkbox"/>	CaatlJko0apInsUwL1x	21/12/2023, 05:13:51	x
<input type="checkbox"/>	DXDHgldfC0wBA9oWL6Ek	14/12/2023, 00:59:38	wrong explanation again!
<input type="checkbox"/>	OYasNFEQssNKBcLcSOf4	14/12/2023, 00:55:40	wrong explanation!
<input type="checkbox"/>	L9PPvuragB1eJ6d7y0BY	14/12/2023, 00:43:35	great!
<input type="checkbox"/>	XQAEIKLTBno7oAJ6zkGe	14/12/2023, 00:25:38	Awesome Answer. Nice to know this.
<input type="checkbox"/>	3zfAkDURkfMxVpDfZcBD	14/12/2023, 00:22:05	Great answers. But takes time to generate. Optimization can be enhanced
<input type="checkbox"/>	VWz6tgZdHd5UBT3Mhdzq	14/12/2023, 00:17:37	Very clear explanation but takes time to generate.
<input type="checkbox"/>	3UMNM5900YJmcFSRos5y	14/12/2023, 00:14:05	A nice explanation.
<input type="checkbox"/>	EYPGBJfnAKlcDJyzXTT5	14/12/2023, 00:09:21	Nice, works as expected
<input type="checkbox"/>	Cnc7XfKOk6JL24mmXXY6	13/12/2023, 14:33:36	Nice to know

Figure 49: Trubrics Text Feedback



```

 XQAEIKLTBno7oAJ6zkGe 14/12/2023, 00:25:38
      Awesome Answer. Nice to know this.

Raw feedback response:
{
  "created_on": "14/12/2023, 00:25:38",
  "tags": [],
  "metadata": {},
  "prompt_id": "d0521b52-6a80-41f3-bf0a-b5f98d3832ff",
  "component": "thesis",
  "model": "gpt-4",
  "user_id": null,
  "user_response": {
    "text": "Awesome Answer. Nice to know this.",
    "type": "thumbs",
    "score": "👉"
  }
}

```

Figure 50: Session id to Prompt id

For each feedback, the prompt id refers to the session id of the prompt. The corresponding prompt by the user/user question and the corresponding LLM answer are displayed below.

id	user_id	session_id	prompt
fdg7f6Rvh9GlnMASEwea	user-id-1234	d0521b52-6a80-41f3-bf0a-b5f98d3832ff	TOOLS ----- Assistant can ask the user to use tools to look up information that may be available online or in a database. This is useful for tasks like entity resolution or fact verification.

Figure 51: Selected session id values for a prompt

```
```json
{
  "action": "Fairlearn Fairness QA",
  "action_input": "How can fairness be evaluated for a machine learning model when there is an uneven distribution of gender in the data set?"
}
```

```

Figure 52: Corresponding question to the Session id

There are several strategies to mitigate unfairness in machine learning model predictions:

1. Pre-processing Techniques: These techniques are applied before the training of the model. They include re-sampling the dataset to ensure balance.
2. In-processing Techniques: These techniques are applied during the training of the model. They include modifying the existing algorithms to incorporate fairness constraints.
3. Post-processing Techniques: These techniques are applied after the model has been trained. They include adjusting the classification threshold or applying post-hoc fairness filters.
4. Adversarial Debiasing: This is a type of in-processing technique where a classifier is trained to predict the outcome while an adversary is simultaneously trying to change the prediction.
5. Fairness through Awareness: This approach involves explicitly including the sensitive attribute in the model and adding a fairness constraint that penalizes unfair predictions.
6. Fairness through Unawareness: This approach involves completely ignoring the sensitive attribute during the model training. However, this approach requires careful validation to ensure fairness.
7. Regularization Techniques: These techniques add a penalty term to the loss function that increases as the unfairness of the model predictions increases.
8. Interpretability and Transparency: This involves making the model more understandable and transparent, so that it can be checked for any potential biases or unfairness.
9. Audit and Monitoring: This involves regularly checking and monitoring the model predictions for any signs of unfairness or discrimination.
10. Legal and Ethical Guidelines: Following legal and ethical guidelines can also help in mitigating unfairness. This includes respecting data privacy and avoiding discriminatory practices.

Figure 53: Corresponding answer to the question

These evaluations show that mostly the responses produced by the chatbot when asked relevant questions on the given context, are correct. The chatbot has an accuracy of 85.7% as can be seen in the pie-chart based on 14 user feedbacks.

## 6 Discussion

While implementing this solution for the given problem, I have faced a few challenges and would like to discuss them in details. I would also like to discuss about some future improvements which can be added to this solution to make the chatbot more robust.

### 6.1 Challenges

Firstly, I had tried to use Pinecone<sup>50</sup> to create a vector database for the tabular datasets provided using embeddings so that it becomes cheaper to use the GPT-4 as less tokens would have been needed to extract the required information. But, this process turned out to be lengthy and did not give fruitful results using the basic subscription plan of Pinecone. The queries on the dataset vector embeddings were not properly answered, wrong answers were provided sometime. I dropped the idea of vector embeddings and turned to LangChain dataframe agents to access the LLM after that.

The next challenge was to extract the Shapley values from the explainer results. They were not in the proper form for each data-point and the dataframe needed to be transformed with respect to a certain axis in order to finally extract the local Shapley values and place them in a correct sequence for each data-point.

After that, while searching for appropriate Fairness libraries for a regression problem, I encountered very few which were mostly git-repos and not meant to be integrated with any other project. I also tried experimenting by using my data for a few fairness libraries, mentioned in the Related work section, but that did not produce correct results, as most were meant for classification problems. Even in Fairlearn, even though the needed metrics or mitigation methods are apparently present for regression problems, they did not work appropriately when used, for example AdversarialFairnessRegressor in Fairlearn. Internally it uses the methods which ultimately give classification results even when provided with regression data. I did not find any tips or examples to finetune this method to work for regression data without changing anything. The Demographic Parity Ratio and Equalized

---

<sup>50</sup><https://www.pinecone.io>

Odds Ratio metrics also take in only classification data for evaluation. It seems that mostly classification problems deal with unfairness and hence mitigation techniques have been created keeping that in mind.

Next, for the 2<sup>nd</sup> dataset, the Adversarial debiasing technique didn't work as I had imagined, and thus I had to use resampling for testing purpose.

Lastly, finding suitable candidates and collecting user feedbacks from them turned out to be quite challenging as reminding them and explaining them the Tool instructions needed too much more time than expected.

## 6.2 Future Work

There are many scopes for improvement in this solution. Firstly, the ML Model can be improved and to NODE architecture mentioned in the Related Work section. This architecture seems to fit well for tabular datasets according to its documentation. For lack of time, I could not utilize this architecture and create a more optimised model to predict salary. Also, NODE is present in PyTorch library. The other libraries I integrated with the keras model work smoothly, and I was not sure how these libraries will be integrating with PyTorch. I would like to try this out in future

Next, a few more explainability methods can be used in the solution to find out the optimal one. Like ALE(Accumulated Local Effects) is a better option than PDP in the Alibi Explain library<sup>51</sup>. The Integrated Gradients method from Alibi Explain library also proves to be useful for many DNN models, as it is gradient based. In future its results can be compared to Kernel Shap explanations from SHAP or Alibi Explain library on my model. For counterfactual calculations, only DiCE library<sup>52</sup> proves to be useful, as it works on regression problems. Hence, it can be used in future. OpenXAI can prove to be useful in this thesis in future if we want to evaluate different explainability methods specifically for given datasets and find out the best one.

While exploring Fairness libraries, AI Fairness 360 by IBM can be used in future and the results from both Fairlearn and AI Fairness 360 can be compared to choose the optimal library for the problem. Measuring Counter-

---

<sup>51</sup> <https://docs.seldon.io/projects/alibi/en/latest/index.html>

<sup>52</sup> [https://interpret.ml/DiCE/notebooks/DiCE\\_getting\\_started.html](https://interpret.ml/DiCE/notebooks/DiCE_getting_started.html)

factual Fairness using fat-fairness library<sup>53</sup> and Individual fairness metrics from AI Fairness 360<sup>54</sup> can be used in future for further fairness evaluation. There are some questions which the chatbot cannot answer, like general context questions, showing SHAP visualizations for better interpretability and extensive fairness metric questions whose context has not been provided to the LLM. These drawbacks can be overcome in the future using finetuning on more efficient LLMs.

In the conversational AI field, recent LLMs can be used and prompts can be finetuned to generate more appropriate results. The UI can be made more interactive using recent Streamlit components. The chatbot can be optimized for speed and performance. The code can be made more modular in nature in future.

## 7 Conclusion

The solution successfully satisfies all the criteria mentioned in the problem of salary prediction using regression datasets. The application can now efficiently interpret Fairness and explainability results for a ML Model using a conversational AI. The results and the user feedbacks deem it fairly usable for research and educational purposes. The solution is also modular and its parts can be replaced with any other of its kind on the go with very few code alterations.

The Conversational AI can help understand ML models and their predictions for laymen using the application. Since it is a chatbot, it can easily be acceptable among non-technical users. It is the beginning of exploring ML fairness and interpretability using Conversational AIs. This app will serve as a stepping stone for many other such applications for research and industry purposes. The deployment of this app in the Streamlit cloud for selected users to access it make it portable and adds another feather to its pros list. There are few shortcoming like robustness, delay etc, but those can be solved in the near future using various new libraries which will be available in the market.

Lastly, the evaluation of database predictions, its explanations, fairness and unfairness mitigations have all been successfully incorporated in this app which will help common users to understand ML and its implications on the

---

<sup>53</sup> <https://fat-fairness.org/index.html>

<sup>54</sup> <https://aif360.readthedocs.io/en/stable/>

society in a better manner.

## 7.1 Appendix

Quillbot<sup>55</sup> and ChatGpt have been used as help in paraphrasing certain parts of the thesis.

## References

- [1] Article 21 - Non-discrimination — fra.europa.eu. <https://fra.europa.eu/en/eu-charter/article/21-non-discrimination>
- [2] Getty Images Statement — newsroom.gettyimages.com. <https://newsroom.gettyimages.com/en/getty-images/getty-images-statement>, [Accessed 05-01-2024]
- [3] IBM Archives: Deep Blue — ibm.com. [https://www.ibm.com/ibm/history/exhibits/vintage/vintage\\_4506VV1001.html](https://www.ibm.com/ibm/history/exhibits/vintage/vintage_4506VV1001.html), [Accessed 05-01-2024]
- [4] The Unreasonable Effectiveness of Recurrent Neural Networks — karpathy.github.io. <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>, [Accessed 05-01-2024]
- [5] What is conversational AI, anyway? — boost.ai. <https://boost.ai/blog/what-is-conversational-ai/>, [Accessed 05-01-2024]
- [6] Agarwal, C., Krishna, S., Saxena, E., Pawelczyk, M., Johnson, N., Puri, I., Zitnik, M., Lakkaraju, H.: Openxai: Towards a transparent evaluation of model explanations (2023)
- [7] Amazon: What is overfitting? - overfitting in machine learning explained - aws, <https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20is%20an%20undesirable%20machine,on%20a%20known%20data%20set.>
- [8] Arrieta, A.B., Díaz-Rodríguez, N., Ser, J.D., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R.,

---

<sup>55</sup> <https://quillbot.com>

- Chatila, R., Herrera, F.: Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai (2019)
- [9] Bastian, M.: GPT-4 ”crushes” other LLMs according to new benchmark suite — the-decoder.com. <https://the-decoder.com/gpt-4-crushes-other-l1ms-according-to-new-benchmark-suite/#:~:text=GPT%2D4%20shows%20significant%20performance,%2C%20mathematical%2C%20and%20coding%20tasks.>, [Accessed 05-01-2024]
  - [10] Belle, V., Papantonis, I.: Principles and practice of explainable machine learning (Jul 2021). <https://doi.org/10.3389/fdata.2021.688969>, <https://doi.org/10.3389/fdata.2021.688969>
  - [11] Cortez, V.: How to define fairness to detect and prevent discriminatory outcomes in machine learning (Dec 2021), <https://towardsdatascience.com/how-to-define-fairness-to-detect-and-prevent-discriminatory-outcomes-in-machine-learning-ef23fd408ef2#:~:text=Demographic%20Parity%20states%20that%20the,positive%20outcome%20at%20equal%20rates.>
  - [12] Dataman, C.K.: Explain any models with the shap values — use the kernelexplainer (Jan 2023), <https://towardsdatascience.com/explain-any-models-with-the-shap-values-use-the-kernelexplainer-79de9464897a>
  - [13] Dataman, C.K.: Explain your model with the shap values - dataman in ai - medium (Jan 2023), <https://medium.com/dataman-in-ai/explain-your-model-with-the-shap-values-bc36aac4de3d>
  - [14] de Boer, M., Vethman, S., Bakker, R., Adhikari, A., Marcus, M., de Greff, J., van der Waa, J., van Zoelen, E., Kamphorst, B., More Authors: The fate system iterated: Fair, transparent and explainable decision making in a juridical case. CEUR Workshop Proceedings **3121** (2022), aAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence, AAAI-MAKE 2022 ; Conference date: 21-03-2022 Through 23-03-2022
  - [15] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2019)

- [16] Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning (2017)
- [17] Dutta, S., Long, J., Mishra, S., Tilli, C., Magazzeni, D.: Robust counterfactual explanations for tree-based ensembles (2022)
- [18] D'Agostino, A.: Introduction to neural networks — weights, biases and activation (Mar 2022), <https://medium.com/mlearning-ai/introduction-to-neural-networks-weights-biases-and-activation-270ebf2545aa>
- [19] D'Agostino, A.: Get started with tensorflow 2.0 — introduction to deep learning (Feb 2023), <https://towardsdatascience.com/a-comprehensive-introduction-to-tensorflows-sequential-api-and-model-for-deep-learning-c5e31aee49fa#:~:text=The%20sequential%20model%20allows%20us,for%20building%20deep%20learning%20models.>
- [20] Galarnyk, M.: Understanding train test split (Jul 2022), <https://builtin.com/data-science/train-test-split>
- [21] Ghorbani, A., Berenbaum, D., Ivgi, M., Dafna, Y., Zou, J.: Beyond importance scores: Interpreting tabular ml by visualizing feature semantics (2021)
- [22] Jiang, S.: What Is a Neural Network? — investopedia.com. <https://www.investopedia.com/terms/n/neuralnetwork.asp>, [Accessed 05-01-2024]
- [23] Joseph, M.: Pytorch tabular: A framework for deep learning with tabular data (2021)
- [24] Lipton, Z.C.: The mythos of model interpretability (2017)
- [25] Lundberg, S.M., Lee, S.: A unified approach to interpreting model predictions. CoRR **abs/1705.07874** (2017), <http://arxiv.org/abs/1705.07874>
- [26] McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics **5**(4), 115–133 (Dec 1943). <https://doi.org/10.1007/bf02478259>, <http://dx.doi.org/10.1007/bf02478259>

- [27] Molnar, C.: Interpretable Machine Learning. 2 edn. (2022), <https://christophm.github.io/interpretable-ml-book>
- [28] Myer, M.: Are Generative AI And Large Language Models The Same Thing? — quiq.com. <https://quiq.com/blog/generative-ai-vs-large-language-models/>, [Accessed 05-01-2024]
- [29] Nichite, P.: Building an interactive chatbot with langchain, chatgpt, pinecone, and streamlit (May 2023), <https://blog.futuresmart.ai/building-an-interactive-chatbot-with-langchain-chatgpt-pinecone-and-streamlit>
- [30] Pang, X., Zhou, Y., Wang, P., Lin, W., Chang, V.: An innovative neural network approach for stock market prediction. *The Journal of Supercomputing* **76**(3), 2098–2118 (Jan 2018). <https://doi.org/10.1007/s11227-017-2228-y> <http://dx.doi.org/10.1007/s11227-017-2228-y>
- [31] Popov, S., Morozov, S., Babenko, A.: Neural oblivious decision ensembles for deep learning on tabular data (2019)
- [32] Rathi, P.: A novel approach to feature importance — shapley additive explanations (Dec 2021), <https://towardsdatascience.com/a-novel-approach-to-feature-importance-shapley-additive-explanations-d18af30fc21b>
- [33] Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review* **65**(6), 386–408 (1958). <https://doi.org/10.1037/h0042519>, <http://dx.doi.org/10.1037/h0042519>
- [34] Saleem, R., Yuan, B., Kurugollu, F., Anjum, A., Liu, L.: Explaining deep neural networks: A survey on the global interpretation methods. *Neurocomputing* **513**, 165–180 (2022). <https://doi.org/https://doi.org/10.1016/j.neucom.2022.09.129>, <https://www.sciencedirect.com/science/article/pii/S0925231222012218>
- [35] Samuel, A.L.: Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development* **3**(3), 210–229 (1959). <https://doi.org/10.1147/rd.33.0210>

- [36] Shannon, C.E.: Prediction and entropy of printed english. *The Bell System Technical Journal* **30**(1), 50–64 (1951). <https://doi.org/10.1002/j.1538-7305.1951.tb01366.x>
- [37] Sharma, S., Henderson, J., Ghosh, J.: CERTIFAI. In: Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society. ACM (feb 2020). <https://doi.org/10.1145/3375627.3375812>, <https://doi.org/10.1145%2F3375627.3375812>
- [38] Shekhar, R.: An Introduction to Fairness in Machine Learning — medium.com. <https://medium.com/analytics-vidhya/an-introduction-to-fairness-in-machine-learning-62ef827e0020>, [Accessed 05-01-2024]
- [39] Sheng, E.: In generative AI legal Wild West, the court-room battles are just getting started — cnbc.com. <https://www.cnbc.com/2023/04/03/in-generative-ai-legal-wild-west-lawsuits-are-just-getting-started.html>, [Accessed 05-01-2024]
- [40] Slack, D., Krishna, S., Lakkaraju, H., Singh, S.: Talktomodel: Explaining machine learning models with interactive natural language conversations (2023)
- [41] Tam, A.: What are Large Language Models - MachineLearningMastery.com — machinelearningmastery.com. <https://machinelearningmastery.com/what-are-large-language-models/>, [Accessed 05-01-2024]
- [42] TURING, A.M.: I.—COMPUTING MACHINERY AND INTELLIGENCE. *Mind* **LIX**(236), 433–460 (10 1950). <https://doi.org/10.1093/mind/LIX.236.433>, <https://doi.org/10.1093/mind/LIX.236.433>
- [43] Tursun, O., Denman, S., Sridharan, S., Fookes, C.: Towards self-explainability of deep neural networks with heatmap captioning and large-language models (2023)
- [44] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2023)

- [45] Werbos, P., John, P.: Beyond regression : new tools for prediction and analysis in the behavioral sciences / (01 1974)
- [46] Xiao, J., Xiao, Y., Huang, A., Liu, D., Wang, S.: Feature-selection-based dynamic transfer ensemble model for customer churn prediction. *Knowledge and Information Systems* **43**, 29–51 (04 2014). <https://doi.org/10.1007/s10115-013-0722-y>
- [47] Zhang, B.H., Lemoine, B., Mitchell, M.: Mitigating unwanted biases with adversarial learning. *CoRR* **abs/1801.07593** (2018), <http://arxiv.org/abs/1801.07593>