EXPLAINABLE ARTIFICIAL INTELLIGENCE

Department of Mathematics
IIT MADRAS

Name of the Project Guides
Prof. Dr. S. Sundar
Dr. Sri Vallabha Deevi(Tiger Analytics)

Submitted by:
Kavita Sonker (MA21M014)

INDIAN INSTITUTE OF TECHNOLOGY MADRAS
DEPARTMENT OF MATHEMATICS

NAME: Kavita Sonker

Roll No: MA21M014

PROGRAMME: M.Tech.

DATE OF JOINING: 28.07.2021

Guide: Prof. Dr. S. Sundar

Co-guide Dr. Sri Vallabha Deevi

Certificate

This is to certify that the report **"Explainable Artificial Intelligence"**, submitted by **Kavita Sonker (MA21M014)**, in partial fulfillment of the requirement for the award of the degree of Master of Technology in Industrial Mathematics and Scientific Computing, Indian Institute of Technology Madras is the record of the work done by her during the academic year 2022-2023 in the Department of Mathematics, IIT Madras, India, under my supervision.

Prof. Dr. S. Sundar
Department of Mathematics
IIT Madras

Dr. Sri Vallabha Deevi
Tiger Analytics
Chennai, India

Acknowledgment

I would like to thank Prof. Dr. S. Sundar and Dr. Sri Vallabha Deevi for their constant help and support, without which the completion of my project for this academic year would be impossible. They have always been there to guide me throughout the course and have patiently solved all the queries associated with the project. Lastly, I would like to thank the Department of Mathematics, IIT Madras, and Tiger Analytics for providing me with an opportunity to work in this exciting field.

Kavita Sonker MA21M014
Department of Mathematics
IIT Madras

Abstract

Explainable Artificial Intelligence (XAI) has become increasingly important as machine learning(ML) models continue to gain widespread use in various domains. To increase the interpretability and transparency of these ML models, several techniques are used here, such as SHapley Additive exPlanations (SHAP), Local Interpretable Model-agnostic Explanations (LIME), and Gradient-weighted Class Activation Mapping (GradCAM). Shapley values are required for the interpretations of global models. Global interpretation approaches include summary plots, interactions, feature dependence, and feature importance. LIME generates interpretable explanations for any model by approximating the model's behavior locally using a simpler, more transparent model. GradCAM is a visualization technique that highlights the regions of an image that are most important for a neural network prediction. This work explores the use of SHAP, LIME, and GradCAM to improve the interpretability of machine learning models and demonstrate their effectiveness in various use cases, including image classification and natural language processing.

CONTENTS

1 Introduction **8**

2 Importance Of Explainable AI **10**

2.1 Transparency And Accountability 10

2.2 Trust And Acceptance 11

2.3 Safety And Security 11

2.4 Legal And Regulatory Compliance 11

3 Types Of Explainable AI **12**

3.1 Model-Based Expainability 13

3.2 Post Hoc Models 14

3.2.1 White box model 15

3.2.2 Black box model 15

4 Explainability Methods For Machine Learning Models **17**

4.1 Local Interpretable Model-Agnostic Explanations(LIME) 17

4.2 Shapley Additive exPlanations(SHAP) 20

5 Explainability Methods For Deep Learning Models **24**

5.1 Integrated gradient 25

5.2 Expected Gradient 27

5.3 Class Activation Map 29

5.4 Gradient-Weighted Class Activation Mapping(Grad-Cam) 31

6 Datasets **33**
6.1 California House Price Prediction Dataset . . . 33
6.2 Breast Cancer Diagnostic Dataset . . . 34
6.3 Imagenet Dataset . . . 35
6.4 IMDB Dataset . . . 37

7 Implementation **38**
7.1 Linear Regression . . . 38
7.1.1 Find The Best Fit Line: . . . 40
7.1.2 Least square method: . . . 40
7.1.3 Model Evaluation Metrics: . . . 41
7.2 Decision Tree . . . 46
7.3 Random Forest Algorithm . . . 51
7.3.1 Python implementation to check Explainability with LIME: . . . 52
7.4 Convolutional Neural Network . . . 54

8 Explainability Techniques For Natural Language Processing **61**
8.1 Vectorization Techniques: . . . 63
8.1.1 Bag-of-words(BoW): . . . 63
8.1.2 Term Frequency-Inverse Document Frequency (TF-IDF): . . . 63
8.2 Algorithms: . . . 64
8.2.1 Support vector machine: . . . 64
8.2.2 Multinomial Naive Bayes: . . . 64
8.3 Explainability with SHAP: . . . 66

9 Conclusion **69**

CHAPTER 1

INTRODUCTION

AI systems are being used in a variety of applications in the real world today, ranging from personal assistants like Siri and Alexa to complex applications in healthcare, finance, transportation, and many other industries. Some of the most significant uses of AI systems and their implications are discussed below:

1. **Healthcare:** AI systems are being used in healthcare to help doctors and healthcare professionals diagnose diseasis, predict outbreaks, and personalize treatment plans for individual patients. AI systems can analyze large amounts of data from medical records, lab results and imaging studies to provide accurate and timely diagnoses. However, there are concerns about the privacy of patient data and the potential for biased algorithms.

2. **Finance:** AI systems are being used in finance to detect fraud, make investment decisions and provide personalized financial advice to consumers. AI systems can analyze large amounts of financial data to identify patterns and make predictions about market trends. However, there are concerns about the potential for AI systems to make risky or biased investment decisions.

3. **Transportation:** AI systems are used in transportation to optimize traffic flow, improve safety and reduce emissions. AI systems can analyze real-time traffic data to adjust traffic signals, reroute traffic and alert drivers about potential hazards. However, there are concerns about the potential for AI systems to malfunction and cause accidents.

4. **Education:** AI systems are being used in education to personalize learning experiences, provide feedback to students, and assist teachers in grading assignments. AI systems can analyze student data to identify strengths and weaknesses and provide personalized learning plans. However, there are concerns about the potential for AI systems to reinforce biases and limit creativity.

5. **Customer service:** AI systems are being used in customer service to provide automated responses to common inquiries and assist human representatives in solving complex issues. AI systems can analyze customer data to provide personalized recommendations and anticipate customer needs. However, there are concerns about the potential for AI systems to misunderstand or misinterpret customer inquiries and provide inappropriate responses.

Artificial intelligence (AI) has many benefits, but it also has several drawbacks that need to be addressed. Some of the drawbacks of AI like a lack of transparency (i.e., many AI models are considered "black boxes"), meaning that they are difficult for humans to understand. There is some real-world an example where artificial intelligence system has gone wrong:

Stock market predictions: AI algorithms have been used to predict stock market trends, but they have not always been accurate. For example, in 2020, an AI-powered hedge fund lost 36 percent of its value in just one week, highlighting the unpredictability of financial markets.

Self-driving car crashes: Self-driving cars have been involved in several accidents, some of which have been fatal. In 2018, a pedestrian was killed by an Uber self-driving car in Arizona, highlighting the limitations of current AI technology.

So, XAI is one solution to these challenges, it provides transparency, interpretability, and accountability in AI decision-making. By doing so, XAI can help to build trust in AI and ensure that it is being used in a responsible and ethical manner, ultimately unlocking its full potential to benefit society.

CHAPTER 2

IMPORTANCE OF EXPLAINABLE AI

Explainable Artificial Intelligence (XAI) is an increasingly important concept in the field of AI. It refers to the ability of an AI system to provide clear and understandable explanations for its decisions and actions. The importance of XAI lies in the following:

2.1 Transparency And Accountability

With the increasing use of AI in decision-making processes, it is crucial that the outcomes and reasoning behind the AI's decisions are transparent and understandable to the users. This not only helps to ensure that the decisions made are fair and ethical, but also provides accountability in cases where the decisions are questioned or challenged. For example, a credit scoring model is a type of AI model that uses statistical algorithms to analyze an applicant's credit history and other relevant factors, such as income, employment history, and outstanding debts. On the basis of this analysis, the model generates a credit score, which is a numerical representation of an applicant's creditworthiness.

2.2 Trust And Acceptance

When users can understand how an AI system arrives at its decisions, they are more likely to trust and accept those decisions. This is particularly important in cases where the AI system's decisions have a significant impact on the user's life, such as in healthcare or finance. For example, a deep learning algorithm could be trained on a large dataset of medical images to recognize patterns associated with a particular disease, such as lung cancer. When presented with a new image, the algorithm can compare it to the patterns it has learned and provide a diagnosis or probability that the patient has the disease. Another example is that, in 2018, DeepMind Health launched an AI-powered diagnostic tool that uses deep learning algorithms to analyze retinal scans and detect early signs of age-related macular degeneration and diabetic retinopathy, two common eye diseases that can cause blindness if left untreated.

2.3 Safety And Security

In critical areas such as healthcare, finance, and transportation, AI systems must be safe and secure. If an AI system's decision-making process is not explainable, it becomes difficult to identify and fix errors or security vulnerabilities. Explainability ensures that the system is transparent, making it easier to identify and address potential problems.

2.4 Legal And Regulatory Compliance

Many industries have legal and regulatory requirements for transparency and accountability in decision-making processes. For example, the European Union's General Data Protection Regulation (GDPR) requires that individuals have the right to an explanation of any decision made by an AI system. XAI helps organizations to comply with these requirements. In general, XAI is essential to build trust in AI systems and ensure that they are used ethically and effectively.

CHAPTER 3

TYPES OF EXPLAINABLE AI

There are different types or categories of explainable AI (XAI), based on various parameters. Here are a few of the most common types:

- **Intrinsic explainability:** This refers to the extent to which an AI algorithm inherently provides explanations for its decisions. An example of an intrinsically explainable AI algorithm is a decision tree, which provides a set of rules that are easy to interpret and understand.
- **Model agnostic explainability:** This refers to methods that can be applied to any AI model, regardless of its complexity or type. Examples of model-agnostic explainability methods include LIME (Local Interpretable Model-Agnostic Explanations) and SHAP (Shapley Additive Explanations).
- **Local explainability:** This refers to the level of detail in the explanations provided by an AI algorithm. Local explainability focuses on individual predictions or decisions, providing more specific and detailed explanations for each decision made by the AI system.
- **Global explainability:** This considers the overall functioning of the AI system, providing a higher-level view of how the AI algorithm works. Global explainability methods

include sensitivity analysis, which assesses the impact of input variables on the output of the AI system.

3.1 Model-Based Expainability

Model-Based Explainability is an approach to explaining how an artificial intelligence (AI) model works by analyzing its structure, parameters, and inputs/outputs. One of the key advantages of model-based explainability is that it can provide more accurate and detailed explanations compared to other XAI methods. By examining the model's inner workings, one can gain insights into how it arrives at its decisions and what factors it considers most important. Model-based explainability methods include sensitivity analysis, which involves systematically varying the input variables to observe their impact on the output of the model. Another approach is to use visualization techniques to represent the model's internal structure and highlight the most important features. There are several model-based explainability methods, some of which are:

Linear models: These models are simple and transparent, and their coefficients can be easily interpreted. In linear regression, the coefficients associated with each input variable represent the degree of influence that variable has on the model's output(target variable). The magnitude and sign of the coefficients in the linear regression are used to assess the importance of the input variables. A positive coefficient indicates that an increase in the corresponding input variable will result in an increase in the model output(target variable), while a negative coefficient indicates that an increase in the input variable will result in a decrease in the output. The absolute value of the coefficients can be used to classify the importance of the input variables. The larger the coefficient, the more important the input variable is in influencing the model's output. Therefore, the coefficients in linear regression can be used to identify which input variables are most critical in the model's decision-making process. They are often used as a baseline to compare with more complex models.

Decision tree: These models split data into different branches and nodes, making it easy to understand how the model is making decisions. For example, consider a decision tree built to predict whether a person is likely to purchase a product based on their demographic information and browsing behavior. The decision tree may start by splitting the data based on the person's age group, with one branch for individuals aged 18-25 and another for those aged 26-35. The next split may be based on the person's browsing behavior, with one branch for individuals who spent more than 10 minutes on the website and another for those who spent less. Each subsequent split further refines the predictions made by the model until a

final prediction is made for each individual in the data set. Furthermore, decision trees can be used to identify the most significant input variables that influence the model predictions. For example, if the split on age was found to be the most significant in the example above, it suggest's that age is a critical factor in predicting whether a person is likely to purchase a product.

Rule-based systems: These models use a set of rules to make predictions, making them transparent and interpretable. However, they can be limited by the number of rules and their complexity.

Bayesian models: These models use probabilities to make predictions, making them transparent and easy to interpret. They can also handle uncertainty and variability in the data.

In summary, model-based explainability methods aim to provide transparency and interpretability to AI models, making them more trustworthy and understandable to humans.

3.2 Post Hoc Models

In the context of explainable artificial intelligence (XAI), a post hoc model refers to an explainability method that is developed after the original model has been trained. Post hoc model is used to provide information about the behavior and decision-making process of the original model. The goal of a post hoc model is to provide a more interpretable and transparent representation of the original model. The simplified model can be trained using a variety of techniques, such as feature selection or dimensionality reduction, to identify the most important features and decision rules that are driving the original model's predictions.

For example, the post hoc model can be used to identify which features are most important for the original model's predictions or to explain why the original model made a particular prediction for a specific input. Post hoc models can take various forms, such as decision trees, rule-based systems, or linear models. This information can be used to improve the original model, to identify potential biases or errors, or communicate the behavior of the model to stakeholders. However, it is important to note that post hoc models are not a substitute for transparent and interpretable models that are developed from the outset. Post-Hoc methods are partitioned into two main concepts: model agnostic/ model specific and local/ global techniques.

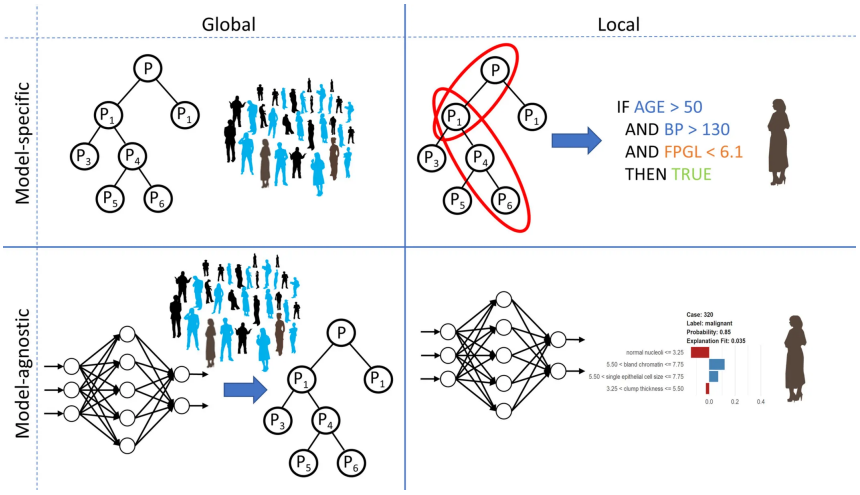


Figure:1 Visual representation of interpretability approaches for machine learning-based predictive modeling in healthcare [(Ref[23])]

3.2.1 White box model

A white box model is a type of machine learning model in which the internal workings of the model are transparent and easily interpretable. In other words, the model's architecture and parameters are known and can be analyzed by the user. This is in contrast to black box models, where the inner workings of the model are opaque, and it's difficult to understand how the model arrives at its predictions. In a white box model, the user can examine the weights and biases of the model, as well as the logic behind its decision-making process. This makes white box models particularly useful in situations where the user needs to understand the reasoning behind the model's output, such as in critical applications like healthcare or finance. Examples of white box models include linear regression, decision trees, and logistic regression. These models are often used in academic research and in industries where transparency and interpretability are important.

3.2.2 Black box model

A black box model is a type of machine learning model where the internal workings of the model are opaque and not easily interpretable. In other words, the user does not have access to the internal mechanisms of the model and it can be difficult to understand how the model arrives at its predictions. In a black box model, the input features are processed through complex algorithms and numerous layers, and the output is generated without any insight

into the internal workings of the model. Examples of black box models include deep neural networks, support vector machines, and ensemble models.

While black box models can often achieve high accuracy on complex tasks such as image recognition or natural language processing, they can be problematic in certain applications where interpretability and transparency are crucial, such as in healthcare or finance. In these cases, it can be important to understand why the model is making a particular decision, and black box models can be difficult to explain.

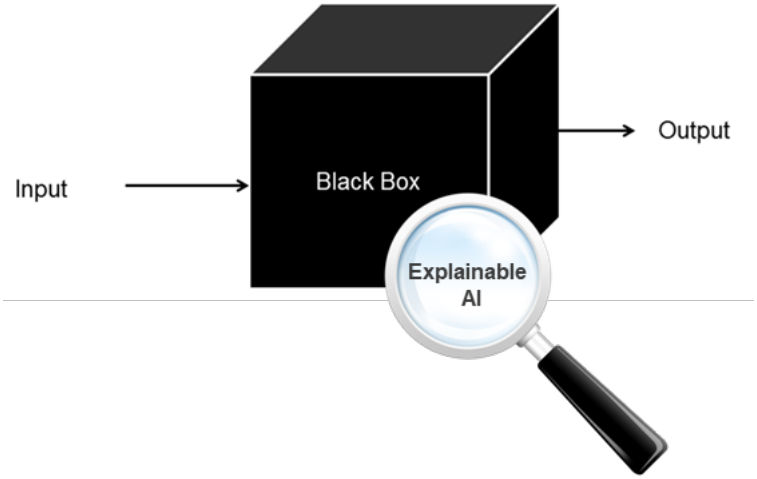


Figure:2 Black-box model [Ref([1])]

CHAPTER 4

EXPLAINABILITY METHODS FOR MACHINE LEARNING MODELS

Explainability methods for machine learning models are techniques that help to make machine learning models more transparent and interpretable to humans. These methods are designed to help humans understand how the model is making its predictions or decisions, particularly for complex or black box models.

4.1 Local Interpretable Model-Agnostic Explanations(LIME)

Local interpretable model-agnostic explanations (LIME) ([28]) is a technique for explaining the predictions of machine learning models. It is an interpretable machine learning approach that can provide insights into the decision-making process of complex models. LIME works by creating a simpler and interpretable model (such as a linear model) that approximates the behavior of the complex black-box model for a specific instance. This simpler model is trained to predict the output of the complex model for that instance, and the weights of the simpler model are chosen to minimize the difference between the output of the simpler model and the output of the complex model. Once the simpler model has been trained, the weights of the features in the simpler model are used to explain how the complex model is

making its prediction for that instance. The features with the highest absolute weights are considered the most important features in the prediction, and their signs (positive or negative) indicate whether they are positively or negatively associated with the predicted outcome.

LIME can be used with a wide range of machine learning models, including decision trees, random forests, and neural networks. It has been used in a variety of applications, including image classification, text classification, and recommendation systems. LIME perfectly summarizes the three basic ideas behind this explanation method:

Model-agnosticism: In other words, model independent, which means that LIME does not make any assumptions about the model whose prediction is being explained. It treats the model as a black box, so the only way that it has to understand its behaviour is by perturbing the input and seeing how the predictions change.

Interpretability: The interpretability of LIME comes from the ability to generate these local explanations. The explanations are presented as a set of weighted features, with positive or negative weights indicating the importance of the feature in the prediction. These explanations can be used to understand how the model is making its predictions and to identify any biases or errors that may be present.

Locality: LIME produces an explanation by approximating the black-box model with an interpretable model in the neighborhood of the instance we want to explain.

Mathematics behind LIME ([28])

The mathematics behind LIME involves three main steps:

1. Sample selection:

LIME first selects a subset of the original input datapoints (e.g., words in a text document, pixels in an image) that are most relevant to the prediction made by the model. This is done using a process called "sampling," which generates a large number of new data points by randomly perturbing the original input in small ways.

2. Model fitting:

LIME then fits a simpler and interpretable model (e.g., linear regression, decision tree) to these new data points, using the model predictions as labels. The objective is to find a model

that approximates the complex model as closely as possible while still being interpretable.

3. Feature importance:

Finally, LIME assigns an importance score to each feature based on how much it contributed to the prediction made by the simpler model. This score reflects the degree to which the original model relied on that feature to make its prediction and can be used to explain the model's decision to a human.

So, LIME provides a way to "localize" the explanation of a model's prediction to a particular input instance by approximating the model's decision-making process using a simpler, more interpretable model. This makes it a powerful tool for XAI, particularly in cases where the original model is too complex to be easily understood by humans.

The mathematical formulation for LIME can be expressed as follows: Given a data instance x and a black box model f, LIME aims to find an interpretable model $g(x')$ that approximates $f(x)$ in a local region around x. The local region is defined by a set of similar instances, which are obtained by perturbing x. Let $D(x)$ be a distribution over the perturbed instances, and let $D'(x')$ be a similar distribution over the corresponding transformed instances $x^{'}$ in the interpretable feature space.

The goal of LIME is to minimize the distance between $f(x)$ and $g(x^{'})$ over the local region, subject to the constraint that $g(x^{'})$ is as simple and interpretable as possible. This can be formulated as an optimization problem:

$$\text{minimize } L(f, g, D) + \Omega(g)$$

subject to

$$E[x' - D'(x)] = x$$

where $L(f, g, D)$ is a loss function that measures the distance between $f(x)$ and $g(x^{'})$ in the local region, and $\Omega(g)$ is a regularization term that encourages simplicity and interpretability of the model g. The constraint ensures that the interpretable model g is faithful to the original data instance x. The solution to this optimization problem gives the locally interpretable model $g(x^{'})$ that best approximates the black box model f in the local region around x. This model can then be used to provide explanations for the prediction of f at x by identifying the features that are most important for the prediction according to $g(x)$.

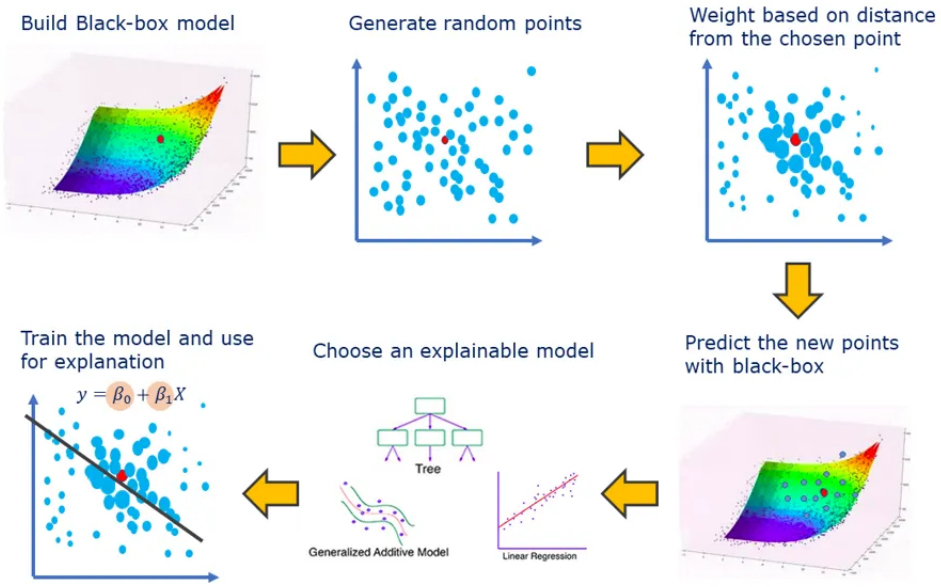


Figure:3 Steps of the LIME algorithm [Ref[8]]

4.2 Shapley Additive exPlanations(SHAP)

Shapley values come from game theory. Shapley values were invented by **LI**oyd S Shapley ([19]) as a way of providing a fair solution to the following question like- **If we have a coalition $'C'$ that collaborates to produce a value $'V'$. So how much did each individual member contribute to the final value?**

Suppose, we have a coalition $'C'$, a group of cooperating members that work together to produce some value V, called the coalition value. This could be something like a corporation of employees that together generate a certain profit or a dinner group running up a restaurant bill. There need to calculate how much each member of the dinner party owes to settle the bill.

To find a fair answer to this question that takes into account these interaction effects, can compute the shapley values for each member of the coalition. So need to calculate the shapley value for member 1. The way this is done by sampling a coalition that contains member 1 and then looking at the coalition formed by removing that member. Then look at the respective values of these two coalitions and compare the differences between these two. This difference is $V_{1234} - V_{234}$ is called the marginal contribution of member 1 to C_{234}.

The goal is to calculate the contribution of members 1 to that specific group. So we then enumerate all such pairs of coalition that is, all pairs of a coalition that only differ based on whether or not membe 1 is included and gets all the marginal contributions for each.
So mean marginal contribution is the Shapley values of that member. It can proceed by the same process for each member of the coalition and find a fair solution to the original question. Mathematically, the formula for calculating the Shapley value of member i with n members is:

$$\phi_i = \frac{1}{n!} * \sum_{\sigma \in \Sigma} (V(C_{\sigma(i)} - V(C_{\sigma \setminus i}))$$

Where,

- ϕ_i is the Shapley value of member, i
- n is the total number of members
- Σ is the set of all possible orderings of the members
- σ is one of these orderings, called a permutation.
- C_{σ_i} is the set of members that contribute before member i in the ordering.
- $C_{\sigma \setminus i}$ is the set of members that contribute after member i in the ordering.
- $V(C)$ is the value of the coalition

The average amount of contribution that a particular member makes to the coalition value is known as Shapley value. Now, translating this concept to the model explainability is relatively straightforward, and that is exactly what **SCOTT LUNDBERG and SU-IN-LEE** [14] did in 2017 with their paper. They introduced the Shap, Shap refers to the Shapley additive explanations. Where look at how members of a coalition contribute to the output of a model output.
Meaning of additive in the Shapley Additive Explanations, Additive feature attribution is defined as if a set of inputs x, and a model f(x). Then define a set of simplified local inputs ($x^{'}$), which usually means that turning a feature vector into a discrete binary vector, where features are either included or excluded. There is also define an explanatory model g. What need to ensure is that

$$x \approx x^{'}$$

$$f(x) \approx g(x^{'})$$

$$g(x^{'}) = \phi_0 + \sum_{i=1}^{N} \phi_i x_i^{'}$$

where ϕ_0 is the null output of the model and ϕ_i is explained effect of features. How much feature changes the model's output, is called attribution. The advantage of this form of explanation is really easy to interpret. To see the exact contribution and importance of each feature just by looking at the ϕ values. Three desirable properties of this additive feature method are local accuracy, missingness, and consistency.

Local accuracy: Local accuracy just simply means that if the input and simplified input is roughly the same, then the actual model and the explanatory model should produce roughly the same output.

Missingness: Missingness states that if a feature is excluded from the model, its attribution must be zero. **That is the only thing that can affect the output of the explanation model**.

Consistency: The Shapley additive value method should be consistent across different subsets of the data. If the method is consistent, then it should provide similar predictions for similar data points, regardless of which subset of the data they are in. This is important because consistency ensures that the method is reliable and can be applied to different subsets of the data without producing widely varying results.

Other variants of SHAP are also present like:

Max SHAP: Max-SHAP is a method for computing the SHAP values for non-linear models by taking the maximum of the SHAP values of the individual features over a set of linear models. The idea is to approximate the non-linear model with a set of linear models and then use the SHAP values from these models to estimate the SHAP values for the non-linear model. The method works by partitioning the feature space into regions and then fitting a linear model to each region. The SHAP values for each feature are then calculated for each linear model, and the maximum value is used as an approximation for the nonlinear model.

Linear SHAP: Linear SHAP is a method to approximating the SHAP values for non-linear models using a linear model. The idea is to fit a linear model to the predictions of the nonlinear model, and then use the coefficients of the linear model to estimate the SHAP values. This approach works by assuming that the non-linear model can be approximated by a linear model in the vicinity of a specific instance and then using the coefficients of the linear model to estimate the contribution of each feature to the prediction.

Deep SHAP: This variant of SHAP is designed specifically for deep learning models, such as convolutional neural networks and recurrent neural networks. It uses a combination of gradient-based and perturbation-based methods to compute the SHAP values.

Tree SHAP: This variant of SHAP is designed specifically for tree-based models, such as decision trees and random forests. It uses a fast algorithm for computing and can be applied to any type of machine learning model.

Kernal SHAP: This variant of SHAP uses a Monte Carlo approximation of the Shapley values, which makes it faster and more scalable than the standard SHAP algorithm. It is particularly well-suited for explaining the predictions of deep learning models. "kernel SHAP" is the one among them that is universal and can be applied to any type of machine learning model.

CHAPTER 5

EXPLAINABILITY METHODS FOR DEEP LEARNING MODELS

Explainability is the ability to understand and interpret the decision-making process of a machine learning model, as mentioned in the previous chapter. In the context of deep learning, which refers to a class of machine learning algorithms that are based on neural networks with many layers, explainability can be particularly challenging due to the complexity of the models. Several approaches can be used to achieve explainability for deep learning models, including:

- **Model visualization:** This involves creating visual representations of the model structure, such as heat maps or activation maps, to help understand how the model makes decisions.
- **Feature importance:** This involves identifying the most important features or variables that are used by the model to make predictions. This can be done using techniques such as the importance of permutation features or SHAP values.
- **Model transparency:** This involves designing the model with interpretability in mind, such as by using simpler architectures or incorporating domain knowledge into the model.

- **Post-hoc interpretation:** This involves using external methods to interpret the model's predictions, such as by training an additional interpretable model on top of the original model or using rule-based systems.

There are several techniques to explain the working of deep learning models:

5.1 Integrated gradient

Integrated Gradient [25] is a technique used to explain the predictions of deep learning models. The basic idea behind Integrated Gradients is to compute the integral of the gradients of the output with respect to the input, which gives an estimate of how important each input feature is for the model's output. The method involves calculating the gradient of the model output with respect to the input at each point along the straight-line path from the baseline input to the actual input. The gradients are integrated along this path to obtain the final explanation for the prediction of the model. The resulting attribution scores indicate how much each input feature contributes to the model's prediction. These scores can be used to identify which features are most important for the model's decision and to gain insight into the model's inner workings. It is applicable to any differentiable model like image, text, or structured data.
In case of image classification , Integrated Gradient is based on mainly two axioms:
1. Sensitivity
2. Implementation invariance

Sensitivity: Sensitivity analysis is a technique that involves testing the sensitivity of the Integrated Gradients[25] results to changes in the baseline input. Using different baseline inputs and comparing the resulting attribution scores, it is possible to assess the robustness of the method and identify any potential biases or artifacts in the results. This technique can be particularly useful for detecting hidden dependencies or confounding factors in the input data.
To implement sensitivity techniques in Integrated Gradients, one approach is to perform a Monte Carlo analysis, where the method is run multiple times with different baseline inputs and/or different implementations. The resulting attribution scores can then be compared and analyzed to assess the reliability and robustness of the method.

Implementation invariance: Implementation Variance is a technique that involves testing the robustness of the Integrated Gradients results to changes in the implementation of the method. This can include variations in the path definition, integration method, or normaliza-

tion scheme. By testing different implementations and comparing the resulting attribution scores, it is possible to identify the sources of variance and select the most appropriate implementation for the specific use case.

Calculating and visualizing Integrated Gradients[25]:

- **Step 1 :** Choose a baseline input such that the baseline input should be a reference input that represents the absence of any features. It can be set to all zeros or random noise. It is important that the baseline input is consistent between all instances and experiments. For example, if you are working with an image, the baseline could be a black image.
- **Step 2 :** Generate a linear interpolation between the baseline and the original input. This means creating a series of small steps α in the feature space between the baseline and the input, with each step continuously increasing in intensity. This can be represented mathematically as:

$$x = x^{'} + \alpha(x - x^{'})$$

Where,

  - α - interpolation constant to perturbed featured by
  - x - Input image tensor
  - $x^{'}$ - Baseline image tensor

Figure:4 Interpolated image [Ref[21]]

- **Step 3 :** For each interpolated image, compute the gradients of the model output with respect to the input. This can be done using backpropagation or any other gradient-based method. The gradients tell us how sensitive the model's output is to changes in the input.

- **Step 4 :** Average the gradients across all interpolated images to obtain an approximation of the integral of the gradients along the path from baseline to input. This gives us the attribution of each pixel in the input to the model's output.
- **Step 5 :** Scale the integrated gradient by multiplying it by the difference between the input and the baseline. This ensures that the attribution values are calculated in the same units as the input.
- **Step 6 :** Represent the integrated gradient on the input image by assigning pixel importance based on their attribution values. This can be done by colouring the pixels according to their importance or by overlaying a heatmap of the attribution values on top of the input image.

Figure:5 Integrated Gradient on the input image with different pixels [Ref[21]].

Finally, the conclusion is that Integrated Gradient can help to explain the deep learning model in which a prediction is made by highlighting the importance of the feature. It does this by calculating the gradient of the model's prediction output to its input features. It does not need any modification to the original deep neural network and can be applied to images, text as well as structured data.

5.2 Expected Gradient

Expected gradient [11] is a term used in machine learning and explainable artificial intelligence to measure the contribution of each input feature to the output of a model. It is essentially the expected change in the output when a specific input feature is changed. To

calculate expected gradient, we take the partial derivative of the model's output with respect to each input feature, and then average those partial derivatives across all instances in the dataset. This yields a vector of gradients, where each element represents the expected change in the output when the corresponding input feature is changed.

Expected gradient can be used to identify the importance of different input features in a model, as well as to understand the behaviour and potential biases of the model. By analyzing the expected gradient values, XAI practitioners can gain insight into the model's decision-making process and make informed decisions about how to improve or adjust the model. For example, if a model is trained on a dataset that contains biased data, the expected gradient can help identify which features are driving the bias and how to correct it.

Mathematics behind expected gradient [11]

Expected gradient can be mathematically defined as the expected value of the gradient of the model's output with respect to its inputs. Let us assume that we have a dataset of N instances, each with d input features and a corresponding output y. The expected gradient for a specific input feature x_i can be calculated as follows:

$$\text{Expected Gradient} = E[\bigtriangledown_{x_i} f(x, y)]$$

where E denotes the expectation operator, and $\bigtriangledown_{x_i} f(x, y)$ is the gradient of the output f(x,y) with respect to the input feature x_i. To calculate this expected gradient, we first need to calculate the partial derivative of the output with respect to each input feature for each instance in the dataset:

$$\frac{\partial f(x_j, y_j)}{\partial x_i}$$

where x_j is the input features of the j-th instance, and y_j is the corresponding output. This gives us a matrix of partial derivatives with dimensions $N * d$. There can take the mean of the partial derivatives with respect to each input feature i to get the expected gradient for that feature:

$$E[\bigtriangledown_{x_i} f(x, y)] = \frac{1}{N} * \sum_{j=1}^{N} \frac{\partial f(x_j, y_j)}{\partial x_i}$$

where $\sum$ represents summation over all instances in the dataset.
In summary, the expected gradient of an input feature is the average of the partial derivatives of the output with respect to that feature, computed over all instances in the dataset.

5.3 Class Activation Map

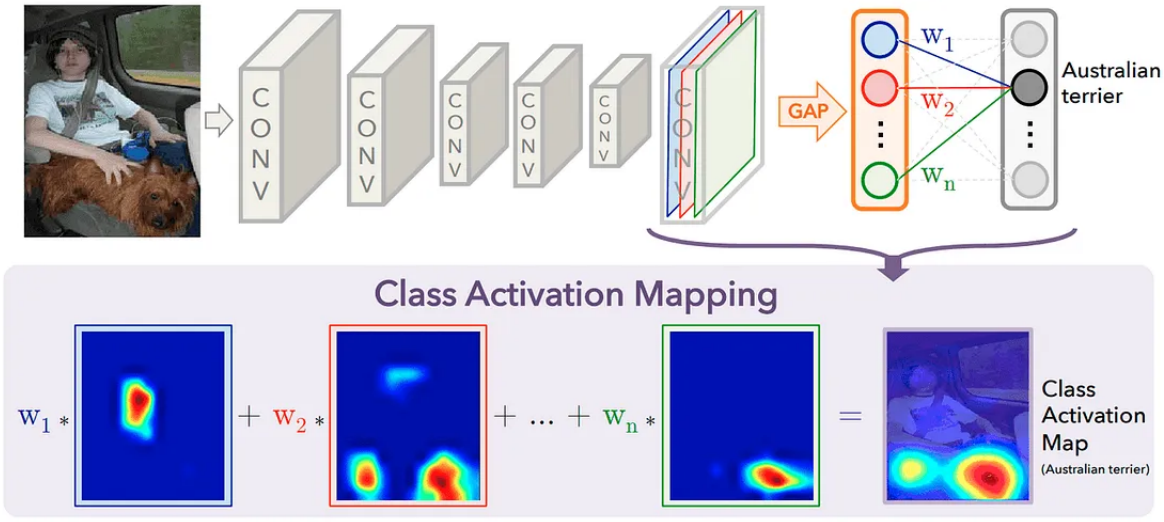


Figure:6 Class activation mapping: the predicted class score is mapped back to the previous convolution layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions [Ref[30]].

Class Activation Map(CAM) [30] is typically used in convolutional neural networks (CNNs), where the final convolutional layer is followed by a global average pooling layer and a fully connected layer. The weights of the final fully connected layer are then used to generate the class activation map.
To generate a CAM, the weights of the final fully connected layer are multiplied with the feature maps of the last convolutional layer to generate a weighted feature map. This weighted feature map is then averaged across all feature maps to obtain a single activation map. The activation map is then upsampled to the size of the input image, and the heat map is generated by applying a colour map to the activation values.

CAMs can be used to understand which parts of an image are most important for a specific classification decision, and they can also be used to generate saliency maps, which highlight the most visually important regions of an image. CAMs are often used in computer vision

applications such as object detection, image segmentation, and visual question answering.

Mathematical intuition behind CAM [30]:

Let X be the input image and let $f(X)$ be the output of the final convolutional layer. The output of the global average pooling layer is denoted by a vector a, and the weights of the final fully connected layer are denoted by a matrix W. The class score for class c can be calculated as:

$$S_c = a^T W_c$$

where W_c is the c-th column of the weight matrix W.

To generate the CAM for class c, we can obtain the class activation map A_c by computing the weighted sum of the feature maps f_i:

$$A_c = \sum_i w_c^i f_i$$

where w_c^i is the weight corresponding to the i-th feature map for class c. The weights can be obtained by taking the gradient of the class score S_c with respect to the feature maps f_i:

$$w_c^i = \frac{\partial S_c}{\partial f_i}$$

The activation map A_c can then be upsampled to the size of the input image, and a colour map can be applied to generate the final CAM visualization.
Hence, the CAM formula can be expressed as:

$$CAM_{c(x,y)} = \sum_i w_c^i f_{i(x,y)}$$

where $CAM_{c(x,y)}$is the activation value for class c at spatial location (x, y) in the input image, and w_c^i and $f_{i(x,y)}$ are the weights and feature maps, respectively.

CAM is limited in its ability to pinpoint the exact location of the important regions in the image.Gradient-weighted Class Activation Mapping (GradCAM) is an improvement over CAM that addresses this limitation. It uses the gradients of the output class with respect to

the feature maps of the last convolutional layer to generate a localization map that highlights the important regions of the image.

5.4 Gradient-Weighted Class Activation Mapping(Grad-Cam)

Gradient Weighted Class Activation Mapping (GradCam) is a technique used in computer vision to visualize the regions of an image that are important for a neural network's prediction. It was introduced in a 2016 paper by Selvaraju et al [20] which combines both **GBP** and **GradCAM**. To explain the idea of GradCAM, need to split it into separate components. In traditional visualizations of neural network predictions, such as heatmaps or saliency maps, the visualization is generated by analyzing the gradients of the output with respect to the input. However, these visualizations do not provide any information about the specific class that the network predicts.

GradCam addresses this issue by generating a heatmap that is specific to a particular class. To do this, it uses the gradients of the output with respect to the final convolutional layer of the network, which represents the feature maps of the input image. It then applies a global average pooling operation to these gradients, producing a weight for each feature map. These weights are then used to compute a weighted sum of the feature maps, which produces the Grad-Cam heat map.

The resulting heat map highlights the regions of the image that are most important for the network's prediction of the given class. GradCam has been used in various applications, such as object localization, image captioning, and visual question answering.

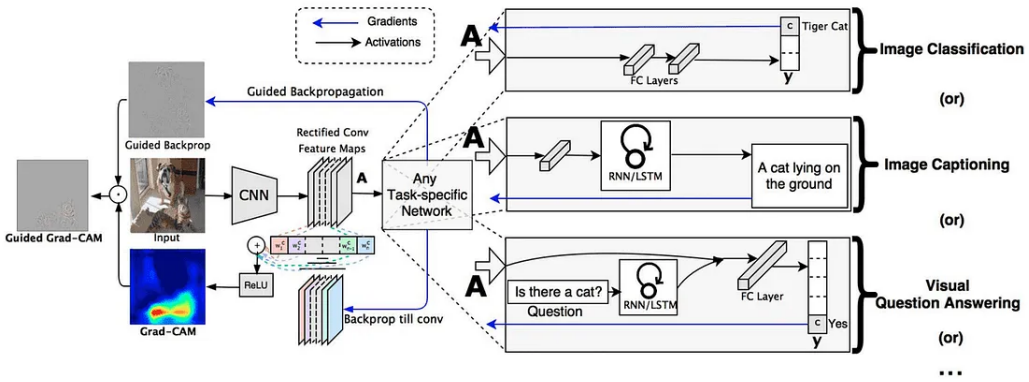


Figure:7 Guided GradCAM computation process [Ref[20]].

The advantage of GradCAM is that it can be applied to any layer in the model, not just the last convolutional layer. This allows for more flexibility in visualizing the important features in the image and can provide additional insights into the model's decision-making process. The key difference between CAM and GradCAM is that GradCAM takes into account the gradients of the output class, which makes it more accurate and precise in localizing the important regions of the image. It has been applied to a wide range of applications, including medical image analysis, object detection, and autonomous driving.

CHAPTER 6

DATASETS

To demonstrate explainability, four datasets are considered in this study. In numerical data, two datasets are studied, one for regression and one for classification. Classification task on one image dataset and one text dataset is also considered.

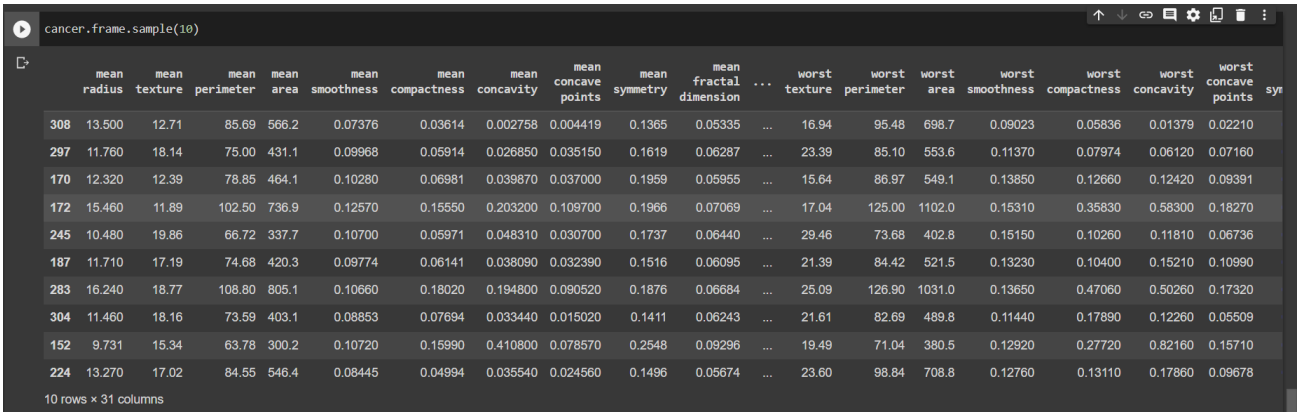
6.1 California House Price Prediction Dataset

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Figure:6.1 Screenshot of California House Price dataset

California House Price Prediction as Figure 6.1 Dataset [3] has 20640 number of instances and there are 8 numeric, independent attributes and the target attribute. Attributes are MedInc, HouseAge, AveRooms, AveBedrms, population, AveOccup. Latitude, Longitude, MedHouseVal. The target variable is the median house value for California districts, expressed in hundreds of thousands of dollars ($100,000).
This dataset was derived from the 1990 U.S. census, using one row per census block group. A block group is the smallest geographical unit for which the U.S. Census Bureau publishes sample data (a block group typically has a population of 600 to 3,000 people). A household is a group of people residing within a home. Since the average number of rooms and bedrooms in this dataset are provided per household, these columns may take surprisingly large values for block groups with few households and many empty houses, such as vacation resorts.

6.2 Breast Cancer Diagnostic Dataset



```
cancer.frame.sample(10)
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	sy
308	13.500	12.71	85.69	566.2	0.07376	0.03614	0.002758	0.004419	0.1365	0.05335	...	16.94	95.48	698.7	0.09023	0.05836	0.01379	0.02210	
297	11.760	18.14	75.00	431.1	0.09968	0.05914	0.026850	0.035150	0.1619	0.06287	...	23.39	85.10	553.6	0.11370	0.07974	0.06120	0.07160	
170	12.320	12.39	78.85	464.1	0.10280	0.06981	0.039870	0.037000	0.1959	0.05955	...	15.64	86.97	549.1	0.13850	0.12660	0.12420	0.09391	
172	15.460	11.89	102.50	736.9	0.12570	0.15550	0.203200	0.109700	0.1966	0.07069	...	17.04	125.00	1102.0	0.15310	0.35830	0.58300	0.18270	
245	10.480	19.86	66.72	337.7	0.10700	0.05971	0.048310	0.030700	0.1737	0.06440	...	29.46	73.68	402.8	0.15150	0.10260	0.11810	0.06736	
187	11.710	17.19	74.68	420.3	0.09774	0.06141	0.038090	0.032390	0.1516	0.06095	...	21.39	84.42	521.5	0.13230	0.10400	0.15210	0.10990	
283	16.240	18.77	108.80	805.1	0.10660	0.18020	0.194800	0.090520	0.1876	0.06684	...	25.09	126.90	1031.0	0.13650	0.47060	0.50260	0.17320	
304	11.460	18.16	73.59	403.1	0.08853	0.07694	0.033440	0.015020	0.1411	0.06243	...	21.61	82.69	489.8	0.11440	0.17890	0.12260	0.05509	
152	9.731	15.34	63.78	300.2	0.10720	0.15990	0.410800	0.078570	0.2548	0.09296	...	19.49	71.04	380.5	0.12920	0.27720	0.82160	0.15710	
224	13.270	17.02	84.55	546.4	0.08445	0.04994	0.035540	0.024560	0.1496	0.05674	...	23.60	98.84	708.8	0.12760	0.13110	0.17860	0.09678	

10 rows × 31 columns

Figure:6.2 Screenshot of breast cancer dataset

The breast cancer dataset as in Figure:6.2 [2] is a classic and very easy binary classification dataset. This dataset contains 569 instances and 30 numeric, independent attributes.

Attribute Information:

- radius (mean of distances from the center to points on the perimeter)
- texture (standard deviation of gray-scale values)

- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The target attributes have two classes, Malignant(212) and Benign(357).

6.3 Imagenet Dataset

ImageNet dataset [5] is a large-scale image label dataset that has been widely used for image classification, object detection, and other computer vision tasks. It contains over 1.2 million images with a resolution of at least 256 pixels in the shortest dimension, divided into 1000 classes.

Class ID	Class Name
0	tench, Tinca tinca
1	goldfish, Carassius auratus
2	great white shark, white shark, man-eater, man-eating shark, Carcharodon caharias',
3	tiger shark, Galeocerdo cuvieri
4	hammerhead, hammerhead shark
5	electric ray, crampfish, numbfish, torpedo
6	stingray
7	cock
8	hen
9	ostrich, Struthio camelus
10	brambling, Fringilla montifringilla
11	goldfinch, Carduelis carduelis
12	house finch, linnet, Carpodacus mexicanus
13	junco, snowbird
14	indigo bunting, indigo finch, indigo bird, Passerina cyanea
15	robin, American robin, Turdus migratorius
16	bulbul
17	jay
18	magpie
19	chickadee

Figure:6.3 Screenshot of imagenet dataset

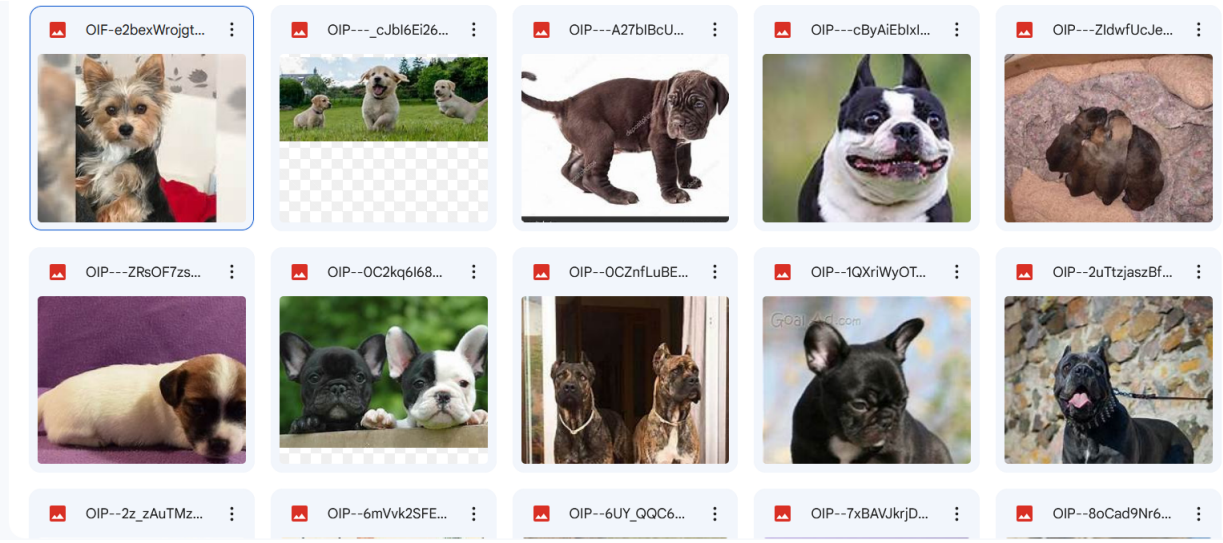


Figure:6.4 Screenshot of images of different classes in imagenet dataset

The images were collected from the web and hand-labeled with a variety of object categories, such as animals, vehicles, and household objects. The dataset as Figure 6.4 has been expanded over the years, and the latest version (ImageNet-21K) contains over 14 million images with more than 21,000 categories.

6.4 IMDB Dataset

The IMDB [6] movie review dataset is a widely used dataset for sentiment analysis. Figure 6.5 contains a collection of 50,000 movie reviews from the Internet Movie Database (IMDB) labeled as positive or negative. The dataset is split evenly into 25,000 training reviews and 25,000 testing reviews.

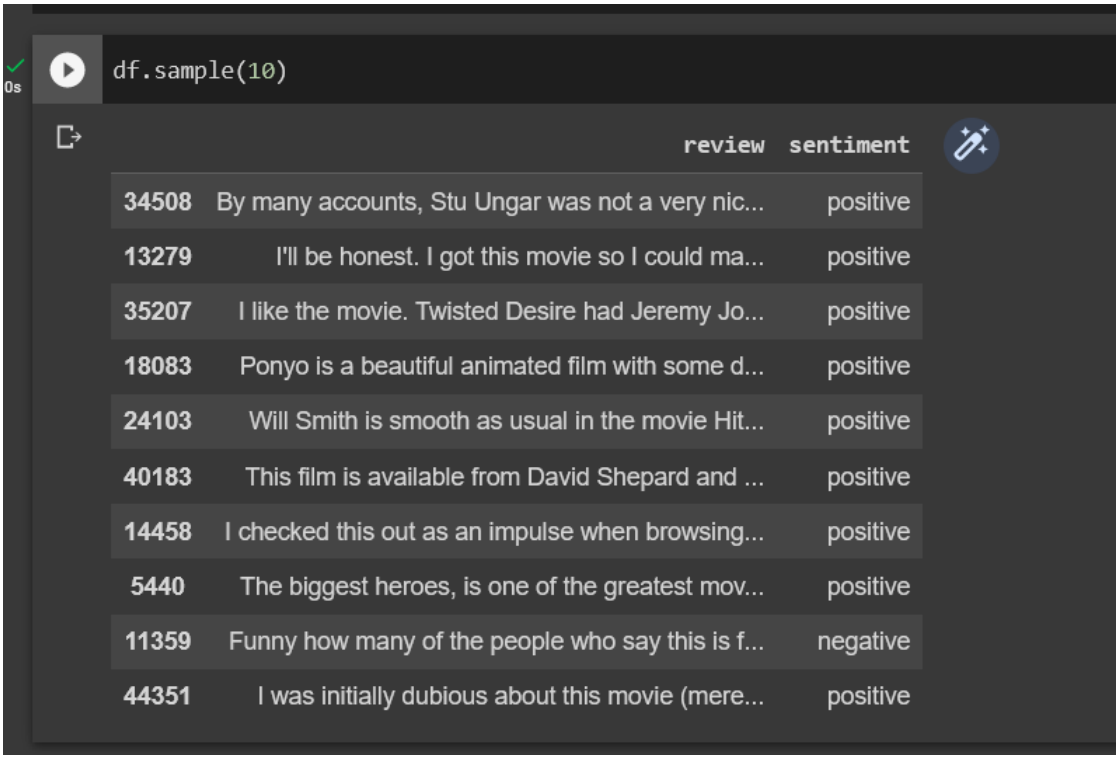


Figure:6.5 Screenshot of IMDB dataset

CHAPTER 7

IMPLEMENTATION

7.1 Linear Regression

Linear regression [24] is a statistical method that is used to analyze the relationship between a dependent variable and one or more independent variables. The main aim of linear regression is to find the best-fit line that describes the relationship between the variables. Mathematically, linear regression represents as:

$$y = mx + b$$

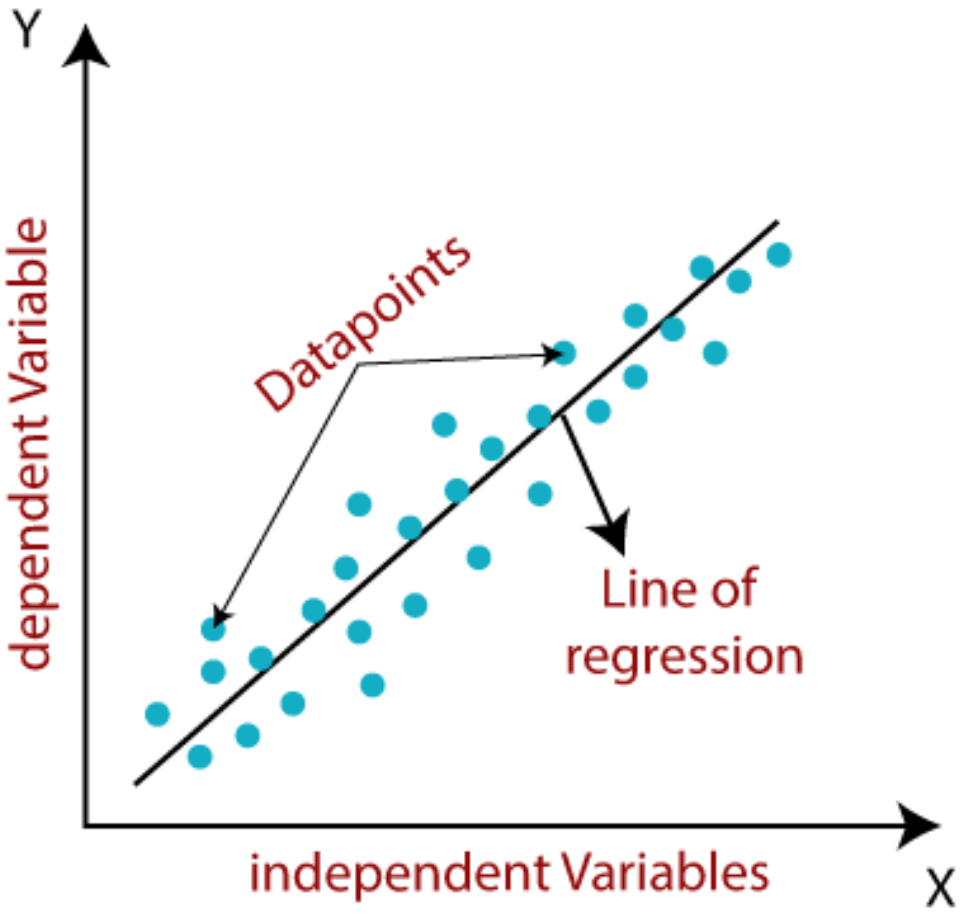


Figure:8 Linear regression [Ref[7]]

Linear regression is further divided into two types of algorithm:

1. simple linear regression
2. multiple linear regression

A linear line that shows the relationship between the dependent and independent variables is known as a regression line.

7.1.1 Find The Best Fit Line:

There are several approaches to estimating the line of best fit line. The simplest method involves visually estimating such a line on a scatter plot and drawing it to the best ability. The more precise method involves the least squares method and another is the gradient descent algorithm. This is a statistical procedure to find the best fit for a set of data points by minimizing the sum of the offsets or residuals of points from the plotted curve.

7.1.2 Least square method:

The ordinary least squares [17] (OLS) method is a linear regression method for estimating model parameters that are unknown. The approach is based on reducing the sum of squared residuals between the actual and predicted values. The difference between the actual value and the predicted value is known as the residual. Error is a synonym for residual. The residual sum of squares (RSS) is another name for the sum of the squared differences. The OLS method reduces RSS by identifying the coefficient values that produce the smallest RSS. The line that results is known as the regression line, and it is the line that best fits the data. This is the coefficient calculated by this method:

$$m = \frac{(y_i - \bar{y})x_i - \bar{x}}{(x_i - \bar{x})^2}$$

$$b = \bar{y} - m * \bar{x}$$

where,

- x is independent variable
- $\bar{x}$ is average of independent variable
- y is dependent variable
- $\bar{y}$ is average of dependent variable

Cost Function: Cost function is used to calculate the error between predicted values and actual values, represented as a single real number. The cost function for linear regression is root mean squared error or mean squared error. Mathematically, it is written as

$$J = \frac{1}{n}\sum_{i=1}^{n}(y_p - y_i)^2$$

where,

- n is total number of data points
- y_i is the actual value of the dependent variable for the ith data point
- y_p is the predicted value of the dependent variable

and the distance between the predicted values and true values.

7.1.3 Model Evaluation Metrics:

To check the performance of a linear regression model, there are several metrics that can be used:
1. R-squared (R^2) value: This is a statistical measure that represents the proportion of variance in the dependent variable that is explained by the independent variables. The R^2 value ranges between 0 and 1, where 0 indicates that the model explains none of the variance, and 1 indicates that the model explains all of the variance. A higher value of R^2 indicates a better fit of the model.

$$R^2 = 1 - \frac{(y_p - y_i)^2}{(\bar{y} - y_i)^2}$$

where, n is the number of data points,
y_i is the actual value of the dependent variable for the ith data point
y_p is the predicted value of the dependent variable
$\bar{y}$ is the mean of the actual values of the dependent variable.

2. Mean Squared Error (MSE): This is the average of the squared difference between the actual and predicted values of the dependent variable. A lower MSE value indicates a better fit of the model.

$$J = \frac{1}{n}\sum_{i=1}^{n}(y_p - y_i)^2$$

3. Root Mean Squared Error (RMSE): This is the square root of the MSE value and provides a measure of the standard deviation of the errors. A lower RMSE value indicates a better fit of the model.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_p - y_i)^2}$$

4. Mean Absolute Error (MAE): This is the average absolute difference between the actual and predicted values of the dependent variable. A lower MAE value indicates a better fit of the model.

$$MAE = \frac{1}{n}|y_p - y_i|$$

To calculate these metrics, these are evaluated on the test data, then compare the predicted values with the actual values of the dependent variable and calculate the metrics mentioned above. The closer the predicted values are to the actual values, the better the model performance.

Python implementation and results:

Examine the model coefficient: The best way to understand a linear model in scaled data is to examine the coefficients learned for each feature. These coefficients tell us how much the model output changes when we change each of the input features:

MedInc	0.45769
HouseAge	0.01153
AveRooms	-0.12529
AveBedrms	1.04053
Population	5e-05
AveOccup	-0.29795
Latitude	-0.41204
Longitude	-0.40125

Check explainability:

To check the explainability of linear regression for California house price prediction dataset , by using Shapley additive explaination method. The SHAP values provide an estimate of the global contribution by using the feature importances.
Here are the steps to apply SHAP in explainability in linear regression:
1. Train a linear regression model using your dataset.
2. Create a shap.Explainer object using the trained linear regression model and the training dataset. This object will be used to calculate the SHAP values for each feature.
3. Calculate the SHAP values for the features of interest using the shap values method of the **shap.Explainer** object. This method takes the test dataset as input and returns the SHAP values.

Visualization using partial dependence plots: To visualize the importance of the feature for a linear model, from Figure 9 the classical partial dependence plot which shows the distribution of the feature values as a histogram on the x-axis:

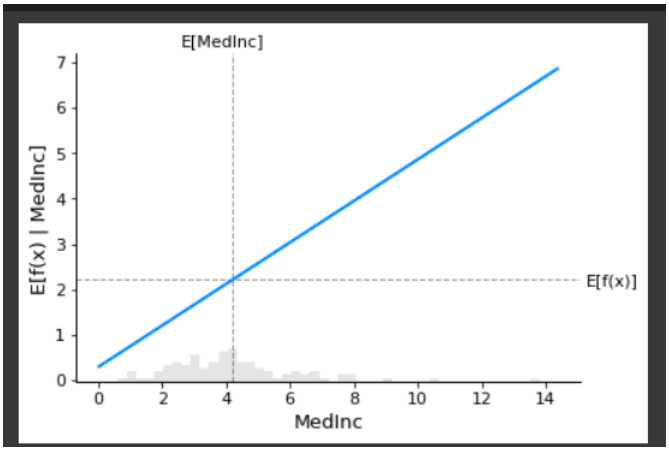


Figure:9 Partial dependence plot

In Figure 9, the partial dependence plot includes the expected value lines for the model output and the target feature. The horizontal gray line represents the expected value of the model output when all features are set to their average values, while the vertical gray line represents the average value of the "MedInc" feature in the dataset. The blue partial dependence plot line represents the average value of the model output when the "MedInc" feature is fixed to a given value. The intersection of the two gray expected value lines represents the point where the "MedInc" feature and the model output are at their expected values, and the blue line passes through this point. This point can be considered as the "center" of the partial dependence plot with respect to the data distribution.

SHAP values from partial dependence plots:

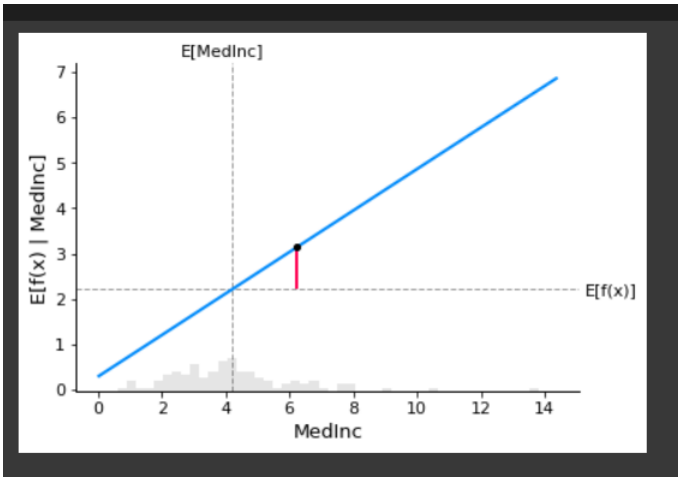


Figure:10 partial dependence plot

Figure 10 shows that, partial dependence plot close correspondence between the classic partial dependence plot and SHAP values means that plot the SHAP value for a specific feature across a whole dataset, exactly traces out a mean centered version of the partial dependence plot for a specific feature.

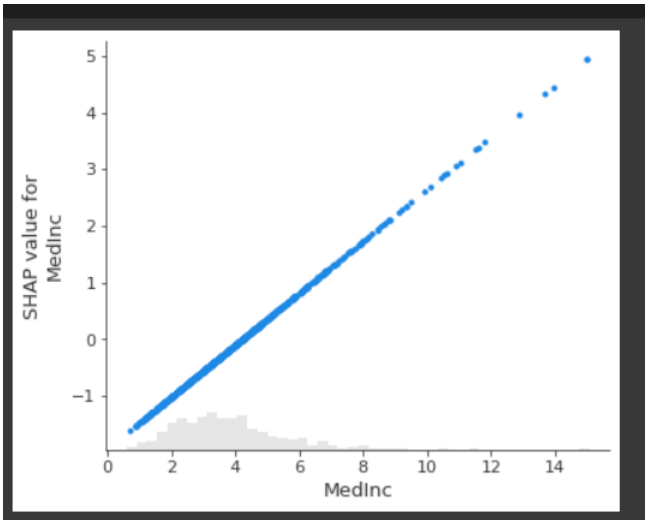


Figure:11 scatter plot for SHAP values

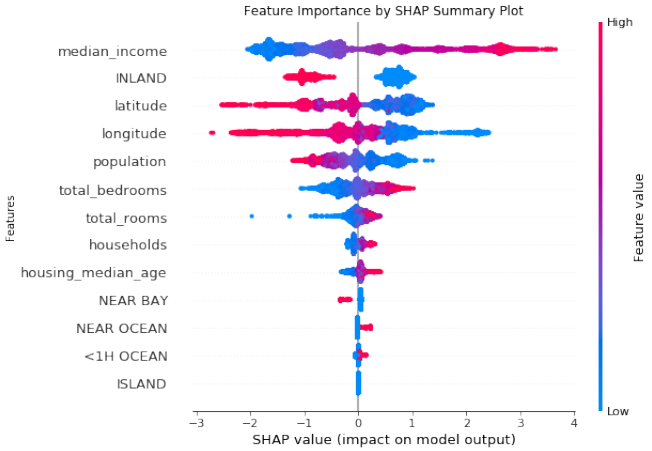


Figure:12 SHAP summary plot

From Figure 12, the horizontal axis represents the SHAP value, and colored points represent the observation having a higher or a lower value, compared to other observations. Higher latitudes and longitudes have a negative impact on the prediction, while lower values have a positive impact.

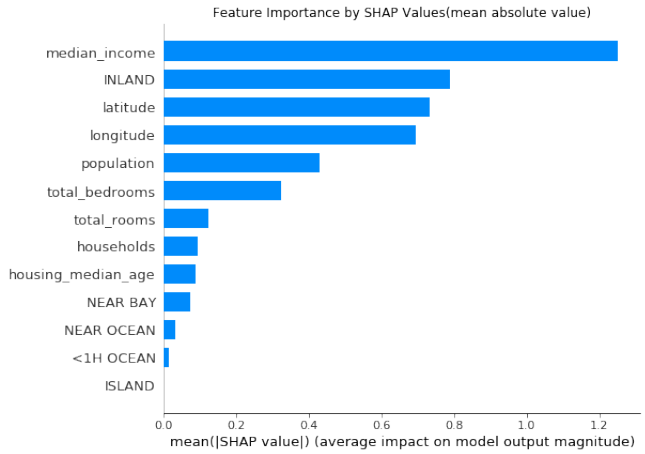


Figure:13 Bar plot for SHAP values

Figure 13 shows that, The features are listed here in order of their impact on the prediction, from highest to lowest. It takes into account the absolute SHAP value, so it is irrelevant whether the feature has a positive or negative impact on the prediction.

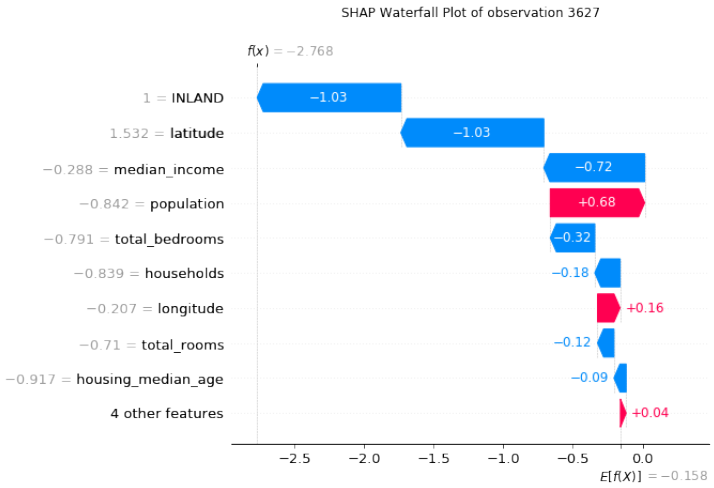


Figure:14 Waterfalls plot

From Figure 14, the waterfalls plot is shows how the sum of all SHAP values equals to the difference between the prediction $f(x)$ and the expected value $E[f(x)]$.

7.2 Decision Tree

The decision tree [16] algorithm is a popular machine learning algorithm that is used for both classification and regression tasks. It is a tree-based model that recursively splits the data into smaller subsets based on the values of the features. The basic idea of a decision tree is to start with a root node that represents the entire dataset, and then recursively partition the data into smaller subsets based on the feature values. Each internal node of the tree represents a decision based on a feature, and each leaf node represents a classification or regression result. The decision tree algorithm can handle both categorical and numerical data, and it can be used for both binary and multiclass classification problems. In addition, decision trees are easy to interpret and visualize, making them a popular choice for machine learning tasks. There are several different variations of the decision tree algorithm, including the ID3, and CART algorithms. Each of these algorithms has its strengths and weaknesses, and the choice of which algorithm to use depends on the specific problem and data set being analyzed.

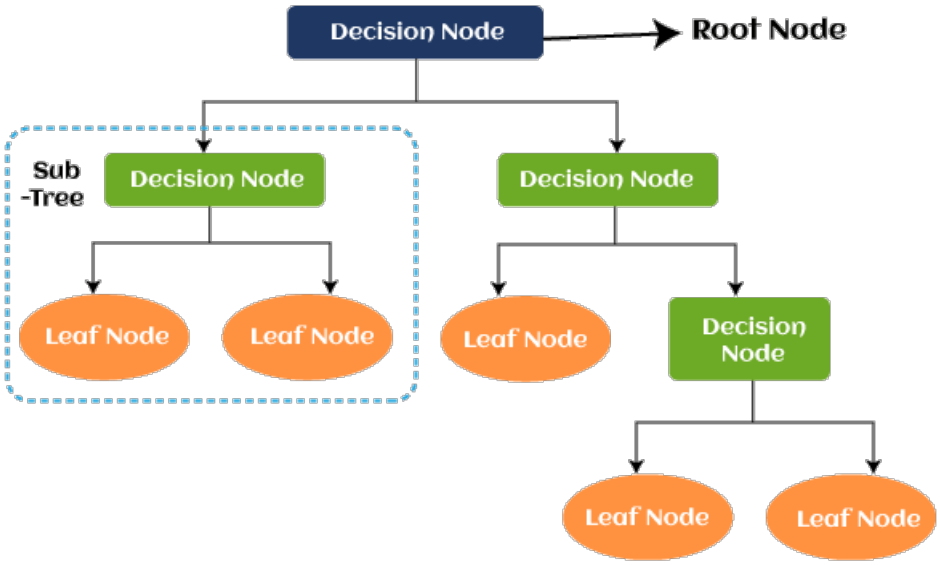


Figure:15 Decision Tree [Ref[4]]

Entropy, Gini Index, and Information Gain are all measures used in the process of building decision trees, which are widely used in machine learning and data mining. The accuracy of the decision tree is calculated as the number of correct predictions divided by the total number of predictions made.
Accuracy $= \frac{TN+TP}{TN+FP+FN+TP}$

where, True positives (TP): The model correctly predicts a positive value.
True negatives (TN): The model correctly predicts a negative value.
False positives (FP): The model incorrectly predicts a positive value.
False negatives (FN): The model incorrectly predicts a negative value.

- **Entropy [26]:** Entropy is a measure of the impurity of a set of data. In the context of decision trees, entropy is used to measure the uncertainty or randomness of a set of examples. The entropy is calculated on the basis of the proportion of positive and negative examples in the set. The higher the entropy, the more uncertain the set of data.
  The advantage of using entropy is that it is sensitive to changes in the distribution of the data. This means that if there is a change in the dataset, the decision tree can be updated to reflect the new information. The mathematical formula for entropy:

$$E = \sum_{i=1}^{n} p_i \log_2(p_i)$$

where , p_i is probability of randomly selecting an example in class i
and Entropy always lies between 0 and 1.

- **Gini Index [26]:** The Gini index is a measure of impurity and it is used to determine the best-split point for a given feature. The split that results in the lowest Gini index is chosen as the optimal split. This is because the split that minimizes the Gini index results in the greatest purity of the resulting subsets. To compute the Gini index for a given split, we first calculate the Gini index for each resulting subset, and then compute a weighted average of the Gini indices, where the weights are proportional to the size of each subset. The formula for the Gini index is:

$$G = 1 - \sum_{i=1}^{n} p_i^2$$

where, n is the number of classes and p_i is the proportion of attributes belonging to class i in the subset.

- **Information Gain [26]:** Information gain is used to determine the importance of a feature in distinguishing between different classes. It is calculated by taking the difference between the entropy of the parent node and the weighted sum of the entropies of the child nodes after splitting for that feature. Information gain is then defined as

$$\text{Information gain } (S, A) = E(S) - \textstyle\sum_{v \in values(A)} \frac{|S_v|}{|S|} E(S_v)$$

where, A is the feature being considered, $v \in values(A)$ is the set of possible values for that feature, S_v is the subset of examples in S that have value v for feature A, and $|S_v|$ and $|S|$ are the sizes of S_v and S, respectively.

ID3 algorithm: The ID3 algorithm is a decision tree algorithm used for classification tasks. It was introduced by Ross Quinlan [13] in 1986 and stands for Iterative Dichotomiser 3. The algorithm follows a top-down, greedy approach that refers to each possible value of the chosen feature, creates a new branch in the tree. Partitions the dataset accordingly, and partitions the input dataset based on the feature that provides the most information gain at each level of the tree. Here are the basic steps of the ID3 algorithm:

1. Calculate the entropy of the target variable for the input dataset.

2. For each feature in the dataset, calculate the information gain when that feature is used to partition the dataset. Choose the feature with the highest information gain as the root of the decision tree.

3. For each possible value of the chosen feature, create a new branch in the tree and partition the dataset accordingly.

4. Repeat steps 2 and 3 for each branch until all the leaves of the tree are pure, i.e., they contain only instances of a single class.

5. Prune the tree by removing any subtrees that do not contribute significantly to the overall accuracy of the model.

Python implementation to check explainability:

- Implementation of the ID3 algorithm in Python in the breast cancer dataset.
- Classify breast cancer as either malignant or benign by using the decision tree classifier from the Scikit-learn library, which uses the ID3 algorithm.
- Removed missing values from the data and split the dataset to train and test.
- Initialize a decision tree classifier using tree.DecisionTreeClassifier() set hyperparameter max-depth =3.
- Fit the decision tree classifier to the training data
- Use the trained model to make predictions on the testing dataset.
- Evaluate the performance of the decision tree classifier and visualize the important features by the bar graph in Figure 16 of the decision tree.

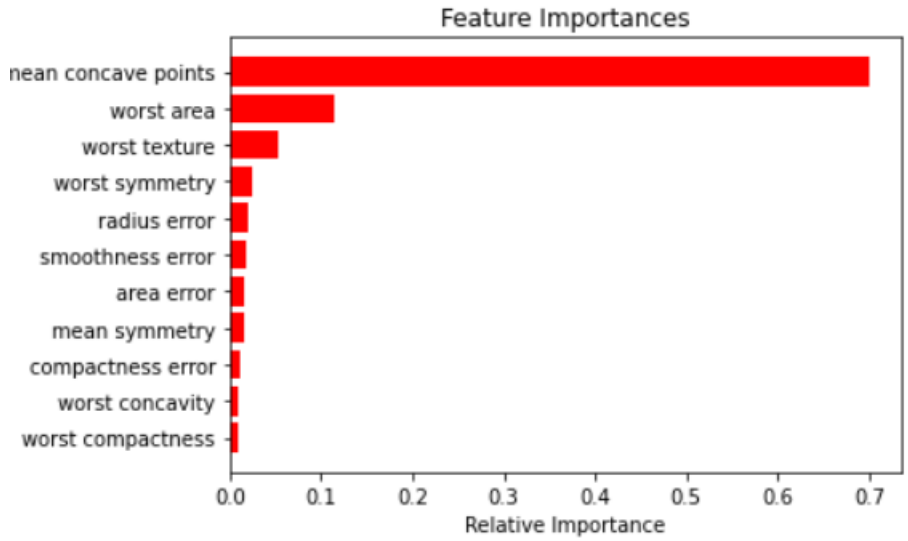


Figure:16 Bar plot

In conclusion, from Figure 16, only the top 11 features are being used. The other features are not being used. Their importance is zero.
Visualize the first three levels of the decision tree, max depth=3.

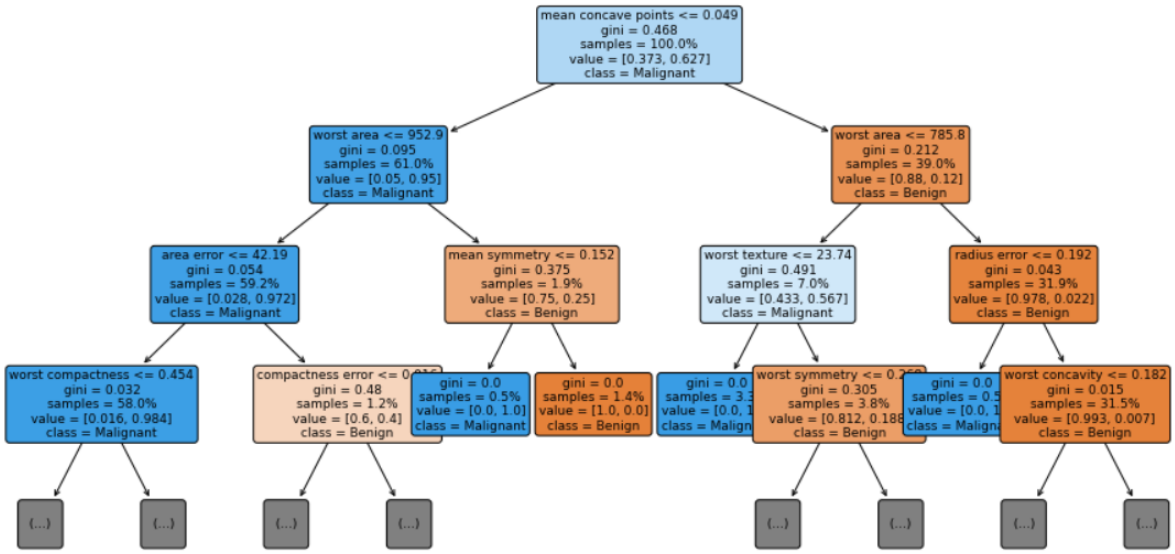


Figure:17 decision tree with depth-3

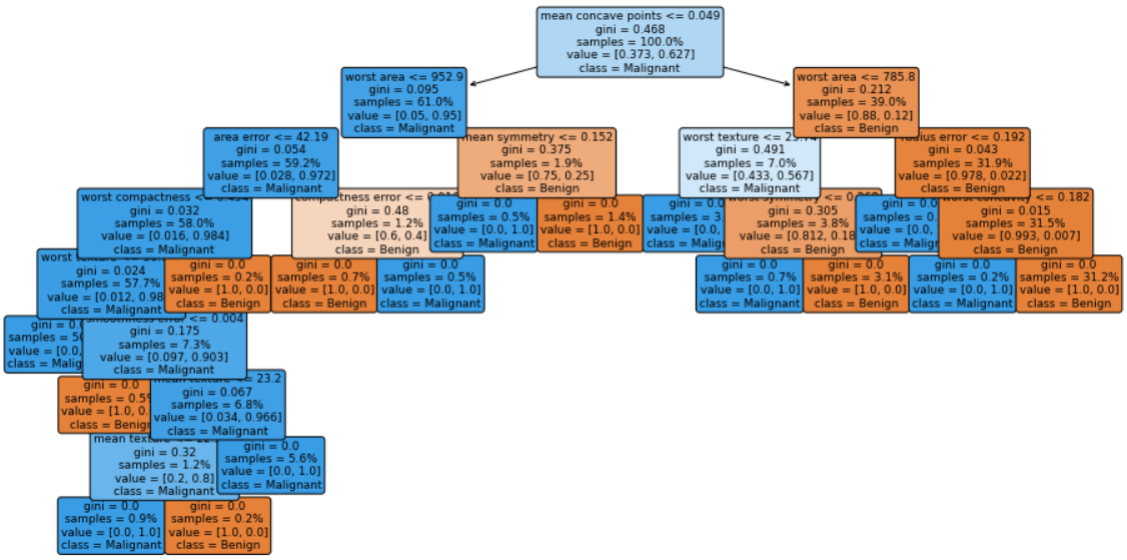


Figure:18 decision tree with depth-8

It is clear from Figure 17 that tree depth 3 is more explainable because it has fewer levels and is easier to visualize and understand. In contrast, as seen in Figure 18 the tree with depth 8 may have many nodes and branches, making it more difficult to interpret and explain. A Gini score greater than zero implies that the samples contained within that node belong to different classes. In Figures 17 and 18, the leaves have a Gini score of zero, meaning that the samples in each leaf belong to a single class. Note that when purity is high, the node/leaf is darker in color.

7.3 Random Forest Algorithm

Random Forest [10] is a popular machine learning algorithm used for both classification and regression tasks. It belongs to the family of ensemble learning algorithms, which combines multiple weaker models to create a stronger one. In a Random Forest model, a large number of decision trees are trained independently on random subsets of data and subsets of features, and the output of each tree is aggregated to produce the final prediction. The randomness comes from the fact that each tree is trained on a different subset of the data and a different subset of the features, this helps to reduce overfitting and increase the model's generalization ability. The algorithm works as follows:

- Random subsets of the data are created by sampling with replacement from the original data.

- A decision tree is trained on each of these subsets using a randomly selected subset of the features.
- During prediction, the output of each decision tree is aggregated to produce the final prediction.

7.3.1 Python implementation to check Explainability with LIME:

Here are the general steps to apply LIME to the breast cancer dataset using a random forest algorithm:

- Load the Breast Cancer dataset and preprocess the data by cleaning the missing values and encoding categorical variables.
- Train a Random Forest model on the dataset:
  - Split the dataset into training and testing sets.
  - Train a random forest model on the training data.
  - Evaluate the performance of the model on the testing data.
- Use LIME to explain the predictions of the model:
  - Select a few instances from the testing set and generate LIME explanations for them.
  - Use the LimeTabularExplainer class of the Lime package to generate local explanations for each instance.
  - Visualize the explanations using a heatmap or other relevant visualization techniques.
  - Compare the importance of features across different instances and draw general insights about the model's behavior.

Results:

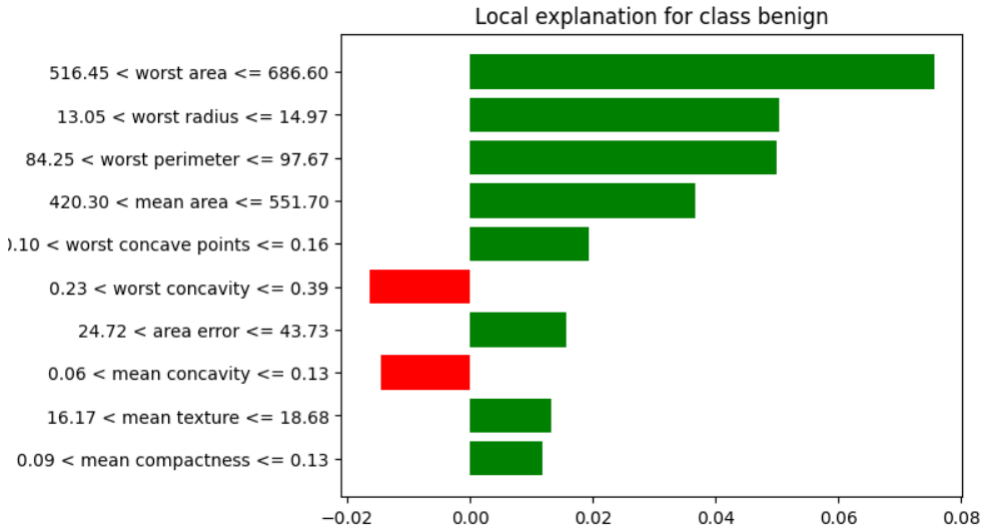


Figure:19 Bar plot

From Figure 19, it is clear in this binary classification problem that the true class and the predicted class are both benign. The values indicate the importance scores assigned to each feature. From the top 5 important features, it appears that the size and shape of the tumor (worst area, worst perimeter, worst radius and worst concave points) were the most important factors in predicting a benign tumor. Other features such as mean concavity, symmetry error, and worst texture had less impact on the classification decision.

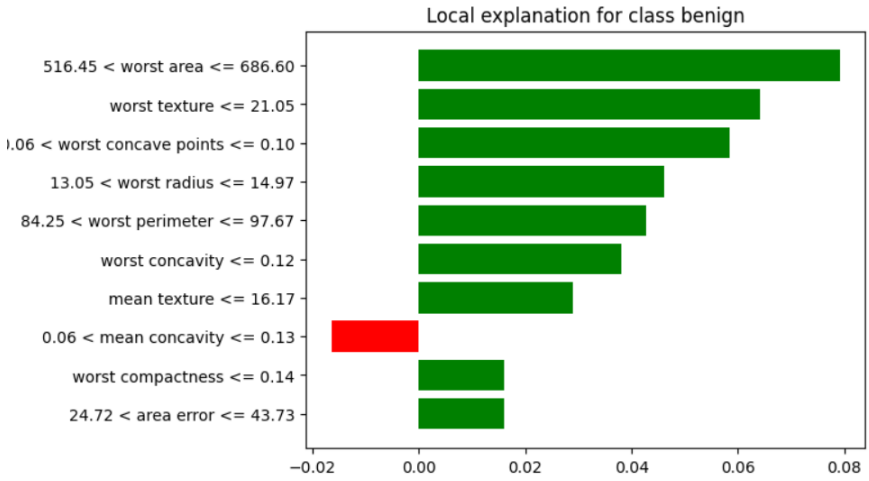


Figure:20 Bar plot

From Figure 20, in this binary classification problem, the true class is malignant and the predicted class is benign. From the top 5 important features, the model is misclassifying the sample as benign when it is actually malignant. The most important features for the malignant class (such as worst concave points, worst texture, and mean texture) have positive importance scores, indicating that they would be expected to contribute to a malignant classification. However, these features are not having a strong enough influence on the classification decision to override the influence of the top features for the benign class (such as worst area and worst texture).

7.4 Convolutional Neural Network

A Convolutional Neural Network (CNN) [15] is a deep learning algorithm used for image recognition. It is a type of neural network that uses convolutional layers to extract features from input images and then uses fully connected layers to classify them. CNNs are inspired by the structure of the visual cortex in the human brain [15]. The filters extract different features from the input image, such as edges, corners, and textures. Filters are also known as kernels or feature maps, are used to extract different features from the input image. Here are some types of filters commonly used as in Figure 21.

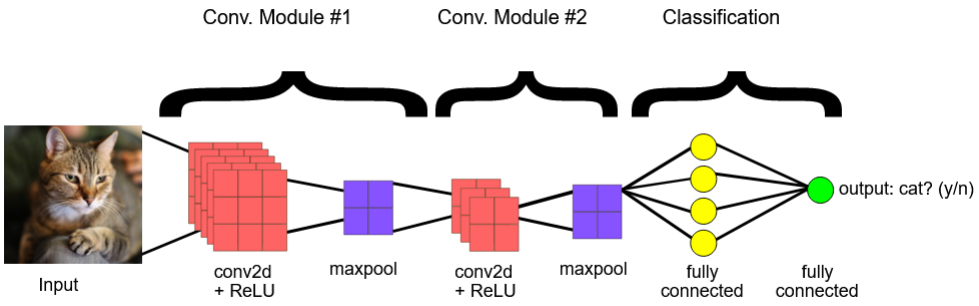


Figure:21 CNN model [Ref[27]]

- **Identity filter [27]:** The identity filter does not perform any convolution operation and simply passes the input through to the next layer. It is used when the input and output sizes of a layer are the same.
- **Edge detection filter [27]:**The edge detection filter is used to detect edges in an image. It consists of a horizontal, vertical, or diagonal kernel that highlights changes in pixel values along edges.

- **Gaussian filter [27]:** The Gaussian filter is used to smooth an image and reduce noise. It applies a Gaussian function to the input image to blur out high-frequency components.
- **Max-pooling filter [27]:** The max-pooling filter is used to reduce the size of the input by taking the maximum value in a set of non-overlapping subregions of the input. This helps reduce the computational cost of the network and prevent overfitting.
- **Average-pooling filter [27]:** The average-pooling filter is similar to the max-pooling filter, but instead of taking the maximum value in each subregion, it takes the average value. This can help to reduce the effect of outliers in the input.

After the convolutional layer, a nonlinear activation function is applied to the output to introduce nonlinearity into the network. This is followed by a pooling layer, which reduces the spatial size of the input, making the network more efficient. The output of the convolutional and pooling layers is then flattened and fed into fully connected layers, which perform the final classification based on the features extracted from the input image.

Explainability using GradCam [20] method: Explainability in Convolutional Neural Networks (CNNs) refers to the ability to understand and interpret the decisions made by the network. Especially in cases like medical diagnosis or autonomous driving, the network's decision-making process is not immediately obvious or intuitive. GradCAM is a useful technique for achieving explainability in CNNs and can help to build trust in these models by providing insights into their decision-making process.

GradCAM [20] (Gradient-weighted Class Activation Mapping) is a technique used in Convolutional Neural Networks (CNNs) to visualize which parts of an image were most important for a network's prediction. To apply GradCAM, one must first select an input image and pass it through the network to obtain the predicted class score. Then, the gradients are computed with respect to the feature maps of the last convolutional layer, and the resulting heatmap is generated. Finally, the heatmap is overlaid onto the original image to provide a visual explanation of the network's decision.

Python implementation:
To check the explainability using GradCAM in the imagenet dataset, first need to take a pre-trained CNN model on the imagenet dataset. Concept is called transfer learning. The name of the classes used in the dataset are the scientific names of the animal species given as:

scientific name	regular name
cane	dog
cavallo	horse
elefante	element
farfalla	butterfly
gallina	chicken
gatto	cat
mucca	cow
pecora	sheep
scoiattolo	squirrel
ragno	spider

Table:1 Scientific name vs Regular name

- Split the dataset into two parts training and validation set with validation split is 0.15. Get that 22257 images belonging to 10 classes in training and 3922 images belonging to 10 classes in validation.
- Preprocessing the data involves resizing images to a common size, scaling pixel values to be between 0 and 1, and one-hot encoding the labels and using the architecture VGG16.
- define the loss function = categorical cross entropy, optimizer = Adam, learning-rate=0.0001 for model.
- Fit the model on the training data, and evaluate it on the validation set to monitor its performance. Once the model has been trained, use it to make predictions on new, unseen data.
- Then it gives the validation accuracy is 96.07 percent and validation loss is 0.1312.

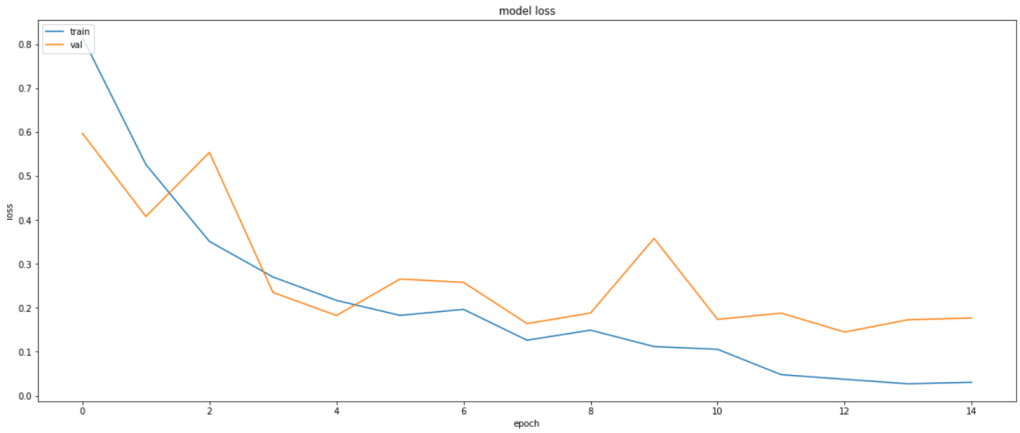


Figure:22 validation loss vs epochs

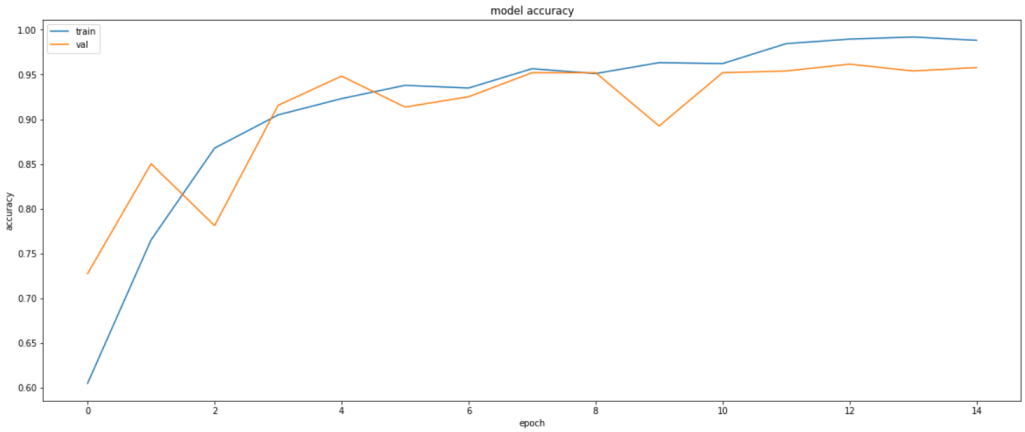


Figure:23 validation accuracy vs epochs

Classification report:

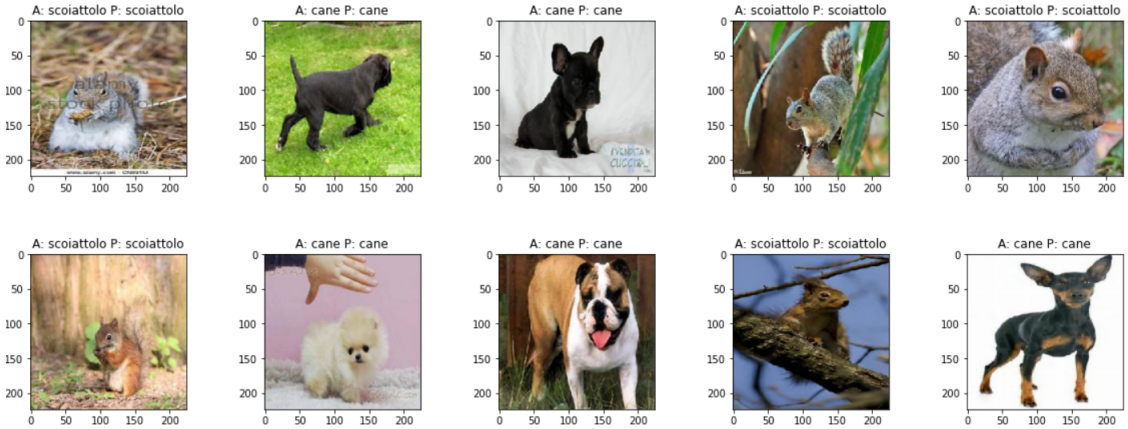
	precision	recall	f1-score	support
0	0.93	0.97	0.95	237
5	1.00	0.20	0.33	5
9	0.97	0.96	0.97	279
accuracy			0.96	521
macro avg	0.97	0.71	0.75	521
weighted avg	0.96	0.96	0.95	521

Table:2

Image Visualization:

- Correctly Classified (Actual(A) and Predicted(P) are same)
- Misclassified (A and P are different)

Correctly classified:



Misclassified:

Grad-CAM Computation:

- Gradient computed [20] of the final softmax layer with respect to the last convolution layer.

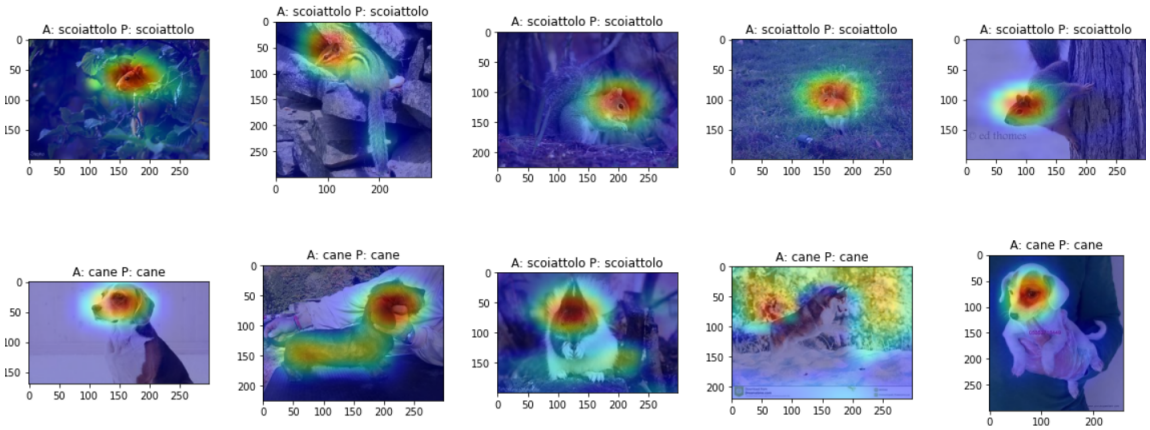
- The fourth last layer of the model is the final convolutional block in the VGG-16 finetuned architecture.
- The mean of the gradients is then multiplied by the last convolution layer's output.
- The output is then passed through a relu function.

Grad-CAM Projection:

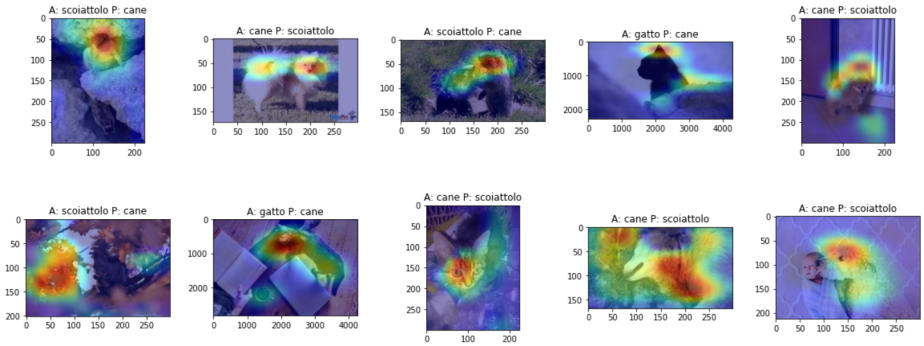
- The heat map produced using the Grad-CAM computation is then projected onto the original image.
- The heat map highlights the important parts of the image that lead to the activation of a class in the final softmax layer.

Result:

Correctly classified:



Misclassified:



Hence it is clear from the above screenshot of the Python code result, some features have been incorrectly detected on some of these animal pictures.

- Distant image of a Pomerian breed dog is misinterpreted as a chicken due to its small size
- Furry cat in the distance is misinterpreted as a sheep
- Body of animals like sheep and horse are being misinterpreted as cow due to the sideways orientation of the animal in the image
- A close up sideways picture of a furry dog is misinterpreted as sheep etc.

These challenges can be resolved by changing the setting of augmentation techniques in the future.

CHAPTER 8

EXPLAINABILITY TECHNIQUES FOR NATURAL LANGUAGE PROCESSING

Sentiment analysis is the process of using natural language processing (NLP) techniques to automatically detect the sentiment (positive, negative, or neutral) expressed in a piece of text. Sentiment analysis can be applied to a wide variety of texts, including social media posts, customer reviews, news articles, etc.

Movie review analysis can be considered a type of customer feedback analysis, as it involves analyzing the sentiment expressed in reviews of movies, which are written by customers or viewers. The steps for creating a sentiment analysis model for movie reviews would involve similar techniques to those used for other types of customer feedback analysis, such as text preprocessing, feature extraction, and classification. The goal would be to train a machine learning model on a dataset of movie reviews and their associated sentiment labels (positive, negative, or neutral) so that it can accurately predict the sentiment of new, unseen movie reviews. The resulting model could be used to analyze the sentiment of movie reviews on a large scale, identify trends and insights about viewers' opinions of different movies or genres, and also check their explainability using the method SHAP and LIME.
Preprocessing is an essential step in sentiment analysis that involves cleaning and transforming the raw text data into a format that is suitable for analysis by machine learning algo-

rithms. The following are some of the common preprocessing steps used in sentiment analysis:

- **Lowercasing:** Convert all the text data to lowercase to avoid redundancy and mismatch caused by case sensitivity.
- **Tokenization:** Breaking down the text into words or smaller units, called tokens. This process makes it easier to analyze the text and extract useful features.
- **Stop word removal:** Stop words are commonly used words in a language that do not carry any significant meaning,like and, the, it etc. These words can be removed to reduce the feature set and avoid overfitting.
- **Stemming/Lemmatization:** Stemming is the process of reducing a word to its root form, whereas lemmatization involves converting the word into its base form, called a lemma. Both of these processes can help to reduce the number of unique features and simplify the analysis. For example, stemming the word 'Caring' would return 'Car', and lemmatizing the word 'Caring' would return 'Care'.
- **Removing special characters and punctuation:** Removing special characters and punctuation from the text data can help to standardize the text and avoid any mismatch caused by differences in formatting.
- **Handling spelling mistakes:** Spelling mistakes can cause inconsistencies in the text data, which can impact the accuracy of the sentiment analysis. Therefore, it is important to handle spelling mistakes by using techniques such as spell checking and correction.
- **Removing HTML tags:** If the text data is obtained from web pages, it may contain HTML tags that need to be removed before analysis.

These preprocessing steps can be performed in different orders and with different variations depending on the specific requirements of the sentiment analysis task. By performing these preprocessing steps, the text data can be transformed into a format that can be used as input to a machine learning model, which can then be trained to predict the sentiment of the text data.

8.1 Vectorization Techniques:

Vectorization is an important step in sentiment analysis that involves converting the pre-processed text data into a numerical format that can be used as input to machine learning algorithms. The following are some of the common vectorization techniques used in sentiment analysis:

8.1.1 Bag-of-words(BoW):

BoW is a simple vectorization technique that involves creating a matrix where each row represents a document or text sample, and each column represents a word in the vocabulary. The cells of the matrix contain the frequency of occurrence of each word in the corresponding document. This technique does not take into account the order or context of words in the text, but it is simple and effective for many applications.

8.1.2 Term Frequency-Inverse Document Frequency (TF-IDF):

TF-IDF[18] is a vectorization technique that assigns weights to each word based on its frequency in the document and its frequency across all the documents in the corpus. This technique takes into account the importance of rare words that can have a high discriminative power. The resulting matrix is a sparse matrix with each row representing a document and each column representing a weighted word from the vocabulary.
$TF(t)$ = (Number of times that term t appears in a document) / (Total number of terms in the document).
$IDF(t) = \log_e$(Total number of documents / Number of documents with term t in it).

$$TF(t) - IDF(t) = TF(t) * IDF(t)$$

8.2 Algorithms:

8.2.1 Support vector machine:

SVM [12] is a binary classification algorithm that works by finding the optimal hyperplane that maximizes the margin between two classes. In sentiment analysis, SVM can be used to classify texts into positive or negative categories based on the presence of specific features. SVM has been shown to be effective in sentiment analysis because it can handle high-dimensional data and find complex decision boundaries.

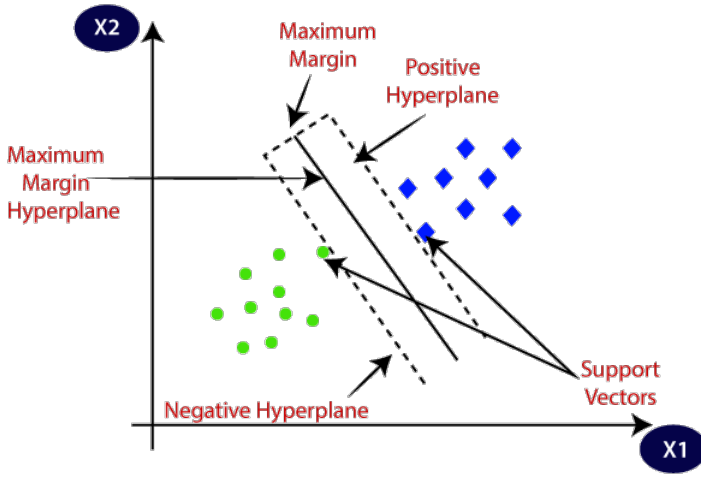


Figure:24 Support vector machine[Ref[9]]

8.2.2 Multinomial Naive Bayes:

MNB [29], is a probabilistic algorithm that is based on the Bayes theorem. It works by estimating the probability of each feature given a class and using these probabilities to classify texts into positive or negative categories. MNB has been shown to be effective in sentiment analysis because it is simple, fast, and can handle sparse data.

Accuracy of the model: Python implementation to train SVM and MNB models using both bag of words (BOW) and term frequency-inverse document frequency (tfidf) features, and obtain the accuracy for each approach
mnb-bow-core : 0.751
mnb-tfidf-score : 0.750

Visualizing the classification report:

Confusion matrix [22]**:** A confusion matrix is a table that is used to evaluate the performance of a machine learning model for a classification problem. It is a way to visualize the number of correct and incorrect predictions made by the model, and to compare them with the actual values. A confusion matrix consists of four parts:

- **True Positives (TP):** the number of positive instances that are correctly predicted by the model.
- **False Positives (FP):** the number of negative instances that are incorrectly predicted as positive by the model.
- **False Negatives (FN):** the number of positive instances that are incorrectly predicted as negative by the model.
- **True Negatives (TN):** the number of negative instances that are correctly predicted by the model.

The matrix is usually arranged in a 2x2 table, with the actual class labels along one axis and the predicted class labels along the other axis. The diagonal entries of the matrix represent the number of correct predictions, while the off-diagonal entries represent the number of incorrect predictions. The accuracy of a model (through a confusion matrix) is calculated using the given formula below.
Accuracy = $\frac{TN+TP}{TN+FP+FN+TP}$

Here's an example of a confusion matrix for a binary classification problem:

	Actual Positive	Actual Negative
Predictive Positive	True Positive (TP)	False Positive(FP)
Predictive Negative	False Negative(FN)	True Negative(TN)

Table:3

Classification report for bag of words and tfidf:

For BoW	precision	recall	f1-score	support
Positive	0.75	0.76	0.75	4993
Negative	0.75	0.75	0.75	5007
accuracy			0.75	10000
macro avg	0.75	0.75	0.75	10000
weighted avg	0.75	0.75	0.75	10000
For tfidf	**precision**	**recall**	**f1-score**	**support**
Positive	0.75	0.76	0.75	4993
Negative	0.75	0.74	0.75	5007
accuracy			0.75	10000
macro avg	0.75	0.75	0.75	10000
weighted avg	0.75	0.75	0.75	10000

Table:4

By analyzing the entries of the confusion matrix, calculate various performance metrics of the model such as accuracy, precision, recall, F1-score, etc as in Table 4. These metrics help to understand how well the model performs and identify areas where it can be improved.

8.3 Explainability with SHAP:

Once the model is trained, use SHAP to explain the predictions made by the model on the test set. Compute SHAP values for each feature (i.e., each word in the review) for a given prediction, and visualize these values using a SHAP summary plot or a SHAP scatter plot.

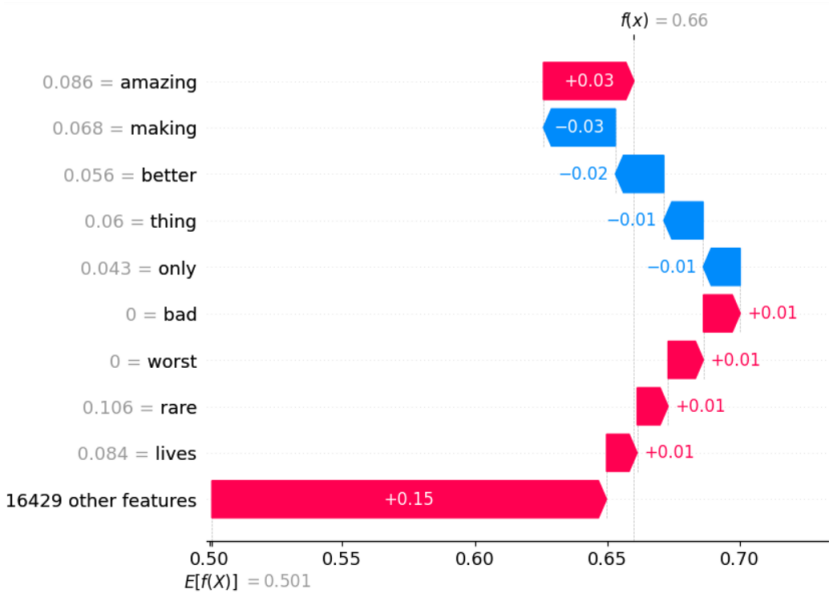


Figure:25 SHAP summary plot for SHAP values

From figure 25, $E[f(x)] = 0.501$ is the average prediction of the model over the test set, and $f(x)$ =0.66 indicates that the prediction of the model for specific review number 6 is positive because the value is strictly greater than 0.5.

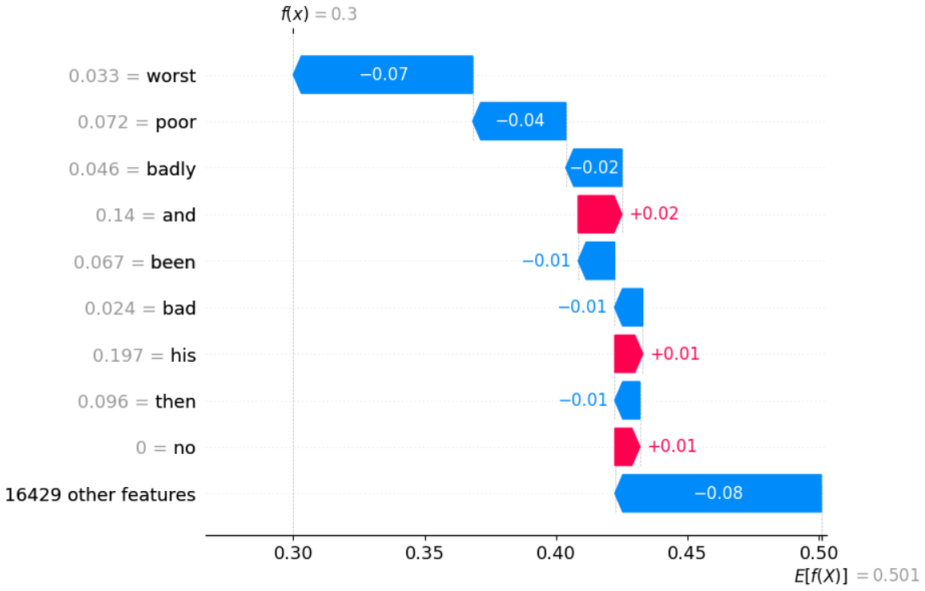


Figure:26 SHAP summary plot for SHAP values

In Figure 26, $E[f(x)] = 0.501$ is the average prediction of the model over the test set, and $f(x)$ =0.3 indicates that the prediction of the model for the specific review number 1222 is negative because the value is strictly less than 0.5.

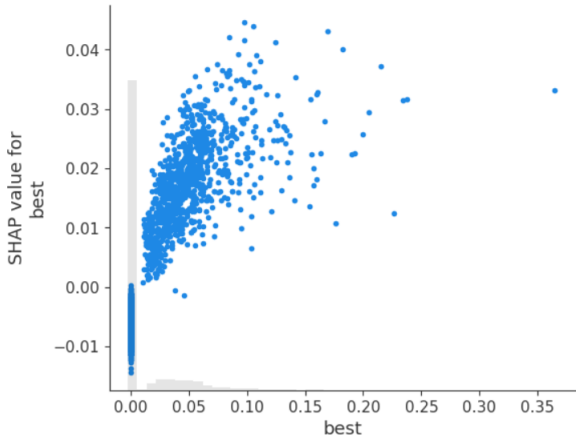


Figure:27 SHAP values of the word "best" for each sample in the test set.

Figure 27 shows that, each point in this scatter plot is representing the word "best" in a different sample of the test set. Here x-axis represents the TF-IDF value of the specific sample, while the y-axis represents its SHAP value.

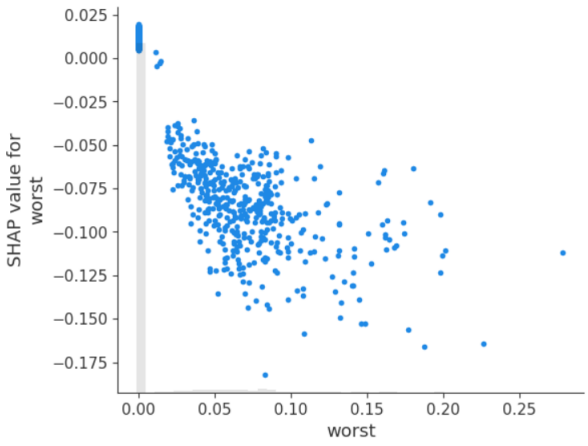


Figure:28 SHAP values for the word 'worst' for each sample in the test set.

From Figure 28, the word 'worst' has a negative SHAP value if present in the review and a small positive SHAP value if not present, and the SHAP values of 'worst' have a higher magnitude concerning the SHAP values of 'best'.

CHAPTER 9

CONCLUSION

Overall, this thesis highlights the use of Explainable Artificial Intelligence (XAI) techniques in combination with machine learning models. On different datasets, such as California house price prediction, Breast cancer, ImageNet and IMDB dataset for text classification, etc., various machine learning models explored here using XAI techniques. Results showed that the combination of machine learning models and XAI techniques can lead to more reliable and trustworthy systems, particularly in domains where the decisions made by the models have significant impacts on human lives, such as healthcare and finance. By providing interpretable explanations for machine learning models, XAI helps ensure that these models are used responsibly and ethically. Another possible direction for future work is to investigate the transferability of machine learning models across human languages. This involves training a model in one language and testing it in another human language to see how well it performs. Transfer learning can be used to leverage the knowledge gained from one language to improve the performance in another language would be a part of future work.

BIBLIOGRAPHY

[1] black box image,. URL.

[2] Breast cancer dataset. URL.

[3] California house price prediction dataset. URL.

[4] Decision tree. URL.

[5] Imagenet dataset. URL.

[6] Imdb dataset. URL.

[7] Linear regression. URL.

[8] steps of lime algorithm,. URL.

[9] Svm image. URL.

[10] Bin Dai, Rung-Ching Chen, Shun-Zhi Zhu, and Wei-Wei Zhang. Using random forest algorithm for breast cancer diagnosis. In *2018 International Symposium on Computer, Consumer and Control (IS3C)*, pages 449–452, 2018.

[11] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[12] Marti Hearst, S.T. Dumais, E. Osman, John Platt, and B. Scholkopf. Support vector machines. *Intelligent Systems and their Applications, IEEE*, 13:18 – 28, 08 1998.

[13] Badr Hssina, Abdelkarim MERBOUHA, Hanane Ezzikouri, and Mohammed Erritali. A comparative study of decision tree id3 and c4.5. *(IJACSA) International Journal of Advanced Computer Science and Applications*, Special Issue on Advances in Vehicular Ad Hoc Networking and Applications, 07 2014.

[14] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[15] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[16] Harsh Patel and Purvi Prajapati. Study and analysis of decision tree based classification algorithms. *International Journal of Computer Sciences and Engineering*, 6:74–78, 10 2018.

[17] Florin Marius Pavelescu et al. Features of the ordinary least square (ols) method. implications for the estimation methodology. *Journal for Economic Forecasting*, 1(2):85–101, 2004.

[18] Juan Ramos. Using tf-idf to determine word relevance in document queries, 1999.

[19] Alvin E. Roth and Lloyd S. Shapley. The shapley value : essays in honor of lloyd s. shapley. *Economica*, 101:123, 1991.

[20] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[21] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *International conference on machine learning*, pages 3145–3153. PMLR, 2017.

[22] Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89, 1997.

[23] Gregor Stiglic, Primoz Kocbek, Nino Fijacko, Marinka Zitnik, Katrien Verbert, and Leona Cilar. Interpretability of machine learning-based prediction models in healthcare. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(5):e1379, 2020.

[24] Xiaogang Su, Xin Yan, and Chih-Ling Tsai. Linear regression. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(3):275–294, 2012.

[25] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[26] T. Suryakanthi. Evaluating the impact of gini index and information gain on classification using decision tree classifier algorithm*. *International Journal of Advanced Computer Science and Applications*, 11, 01 2020.

[27] Tahir Syed, Jalaluddin Qureshi, and Syed Rizvi. Real time retail petrol sales forecasting using 1d-dilated cnns on iot devices. 11 2019.

[28] Giorgio Visani, Enrico Bagli, and Federico Chesani. Optilime: Optimized lime explanations for diagnostic computer algorithms. *arXiv preprint arXiv:2006.05714*, 2020.

[29] Shuo Xu, Yan Li, and Wang Zheng. Bayesian multinomial naïve bayes classifier to text classification. pages 347–352, 05 2017.

[30] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.