TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

BÁO CÁO TỔNG KẾT

HỌC PHẦN HỌC MÁY CƠ BẢN

NĂM HỌC 2024-2025

ỨNG DỤNG THUẬT TOÁN ĐỂ XÂY DỰNG MÔ HÌNH DỰ ĐOÁN GIÁ BẤT ĐỘNG SẢN

Sinh viên thực hiện

Trịnh Thành Nam	**CQ.62.CNTT**	**Bộ môn Công nghệ thông tin**
Phan Tấn Thịnh	CQ.62.CNTT	Bộ môn Công nghệ thông tin

Người hướng dẫn: TS. Lê Ngọc Hiếu

Thành phố Hồ Chí Minh, 2024

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

BÁO CÁO TỔNG KẾT

HỌC PHẦN HỌC MÁY CƠ BẢN

NĂM 2024

ỨNG DỤNG THUẬT TOÁN ĐỂ XÂY DỰNG MÔ HÌNH DỰ ĐOÁN GIÁ BẤT ĐỘNG SẢN

Sinh viên thực hiện

Trịnh Thành Nam	**Nam, Nữ: Nam**	**Dân tộc: Kinh**
Lớp: CQ.62.CNTT	**Khoa: Bộ môn Công nghệ thông tin**	**Năm thứ: 4/4**
Phan Tấn Thịnh	Nam, Nữ: Nam	Dân tộc: Kinh
Lớp: CQ.62.CNTT	Khoa: Bộ môn Công nghệ thông tin	Năm thứ: 4/4

Ngành học: Công nghệ thông tin

Người hướng dẫn: TS. Lê Ngọc Hiếu

Thành phố Hồ Chí Minh, 2024

Tóm tắt

Bài báo cáo này trình bày việc áp dụng các thuật toán học máy trong xây dựng mô hình dự đoán giá bất động sản, với mục tiêu cung cấp công cụ hỗ trợ chính xác cho các quyết định đầu tư và định giá. Đối tượng nghiên cứu là các dữ liệu về đặc điểm bất động sản như diện tích, vị trí, loại hình bất động sản và các yếu tố kinh tế xã hội liên quan. Phương pháp nghiên cứu bao gồm việc thu thập, xử lý và phân tích dữ liệu từ các nguồn mở, sử dụng các thuật toán học máy như hồi quy tuyến tính, cây quyết định, và mạng nơ-ron nhân tạo để xây dựng mô hình dự đoán. Kết quả huấn luyện cho thấy các mô hình hồi quy tuyến tình và mạng nơ-ron đều cho ra tỉ lệ dự đoán khá tốt nhưng vẫn còn một số điểm yếu cần phải khắc phục. Kết luận cho thấy các thuật toán học máy có tiềm năng lớn trong việc dự đoán giá bất động sản, đặc biệt là khi kết hợp với các yếu tố đặc thù của thị trường. Các kết quả này có ý nghĩa quan trọng trong việc tối ưu hóa chiến lược đầu tư và phát triển bất động sản.

Từ khoá

Dự đoán giá bất động sản, học máy, hồi quy tuyến tính, cây quyết định, mạng nơ-ron nhân tạo, dữ liệu bất động sản, xử lý dữ liệu.

Abstract

This report presents the application of machine learning algorithms in building real estate price prediction models, with the goal of providing accurate support tools for investment and valuation decisions. The research object is data on real estate characteristics such as area, location, type of real estate and related socio-economic factors. Research methods include collecting, processing, and analyzing data from open sources, using machine learning algorithms such as linear regression, decision trees, and artificial neural networks to build models. prediction shape. The training results show that linear regression models and neural networks both produce quite good prediction rates but there are still some weaknesses that need to be overcome. The conclusion shows that machine learning algorithms have great potential in predicting real estate prices, especially when combined with market-specific factors. These results have important implications in optimizing real estate investment and development strategies.

Keywords

Real estate price prediction, machine learning, linear regression, decision trees, artificial neural networks, real estate data, data processing.

TABLE OF CONTENTS

TABLE OF IMAGES

TABLE OF ABBREVIATIONS

ID	Abbreviation	Full word
1	ANN	Artificial Neural Networks
2	SVM	Support Vector Machine
3	LSTM	Long Short-Term Memory
4	MSE	Mean Squared Error
5	RMSE	Root Mean Square Error
6	CBD	Central Business District
7	MSE	Mean Square Error
8	CART	Classification And Regression Tree
9	ASM	Attribute Selection Measurement
10	AF	Activation Function
11	MLP	Multi-Layer Perceptron
12	NLP	Natural Language Processing
13	AI	Artificial Intelligence
14	SVC	Support Vector Classification
15	RBF	Radial Basis Function
16	RFR	Random Forest Regression
17	RNN	Recurrent Neural Networks

LIST OF TABLES

Instructor's comments on the scientific contributions of students carrying out the topic *(this part is recorded by the instructor):*

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Date …… month …… 2024

Instructor

(Sign, fullname)

CHAPTER 1: OVERVIEW

1.1. Introduction to the topic

The real estate market is one of the most important sectors in the economy of most countries. Real estate values not only influence investment decisions but also directly impact urban development policies and financial markets. However, determining the exact value of real estate is always a big challenge due to the constant fluctuations of market factors such as location, property characteristics, macroeconomic situation and buyer demand.

In this context, building an accurate and effective real estate price prediction model is extremely important. Traditional methods such as manual analysis or comparison of values between similar real estates can provide reference results, but they are not powerful enough to handle the complexity and volatility of today's market. Therefore, the application of machine learning algorithms to build prediction models has become a popular trend in the real estate research and application industry.

Machine learning, with its ability to process and analyze large volumes of data, can help identify factors that affect real estate values more accurately and improve the reliability of value predictions. Machine learning algorithms such as linear regression, decision trees, and artificial neural networks have been widely applied to build real estate price prediction models, providing positive results and opening up new opportunities for the industry.

The objective of this paper is to study and apply machine learning algorithms to build a real estate price prediction model. Factors such as area, location, property type and other factors will be used as features in the model building process. This paper will also analyze the effectiveness of different algorithms, thereby drawing conclusions and suggestions for practical applications of real estate value prediction models in the current market environment.

1.2. Introducing the author group

Regarding team members, our team consists of 2 people:

- Trinh Thanh Nam: Responsible for building the model with 4 algorithms:

Linear Regression, Gradient Boosting, Decision Tree Regression, ANN.

- Phan Tan Thinh: Responsible for building the model with 3 algorithms: SVM, Random Forest Regression, LSTM.

1.3. Reason for choosing the topic

The real estate market plays an important role in the economy of each country, with a great influence on investment decisions and social development. However, predicting real estate values is always a complex issue, due to the impact of many factors such as economic fluctuations, policies, and market trends. Accurately forecasting real estate prices not only helps investors make wise decisions but also creates development opportunities for businesses in the industry.

In the context of the increasingly unpredictable and volatile real estate market, traditional methods such as comparative analysis and manual evaluation are no longer effective enough to meet the requirements of accuracy and speed. Therefore, the need to apply new technologies, especially machine learning, in predicting real estate values has become a prominent trend. Machine learning algorithms are capable of processing large volumes of data and automatically identifying potential patterns and relationships between factors affecting real estate value.

This topic is chosen to study and apply machine learning algorithms to build a real estate price prediction model. The application of techniques such as linear regression, decision trees and artificial neural networks can improve the accuracy of forecasts and support investors in making more accurate decisions. Moreover, machine learning models have the ability to automatically adjust and improve over time, in line with the constant fluctuations of the real estate market.

In addition to its urgency and practicality in the industry, the study of real estate price prediction models also contributes to the development of knowledge and application of machine learning in various fields, especially in industries related to big data and predictive analytics. Therefore, this topic is not only academic but also has high application value, creating opportunities for optimizing real estate investment and development strategies.

For the above reasons, the topic "Application of machine learning algorithms to build real estate price prediction models" is a research direction with high feasibility and profound practical significance.

1.4. Research objectives

The objective of this topic is to apply machine learning algorithms to build a real estate price prediction model, in order to provide accurate and effective support tools in making real estate investment and development decisions. Specifically, the research objectives include:

- Building real estate price prediction models: Apply machine learning algorithms such as linear regression, decision trees, and artificial neural networks to build real estate price prediction models based on input factors such as area, location, property type, and related socio-economic factors.

- Evaluate the performance of machine learning algorithms: Compare the accuracy of predictive models using evaluation metrics such as accuracy, mean square error (MSE, RMSE), and other model evaluation metrics (such as R^2). This will determine which algorithm provides the best results for predicting real estate prices.

- Analyze factors affecting real estate value: Use machine learning models to identify key factors affecting real estate value. This will help make judgments about factors to focus on when making investment decisions.

- Provide recommendations on practical applications: Based on the model results, propose specific recommendations on how to apply real estate price prediction models in practice to support investors and businesses in the real estate industry.

1.5. Research methods

To train the model to produce the highest performance prediction results, we conducted the following procedures:

1.5.1. Prepare and clean the dataset

The dataset named Melbourne Housing Snapshot is collected from the Kaggle website. The input data includes information about real estate such as area, location,

number of bedrooms, number of bathrooms, ... and other economic factors. To ensure the accuracy and quality of the dataset, data cleaning is done through the following steps:

- Handling missing values by replacing them with average values, descriptive values, or ignoring missing data samples if they are too few.
- Removing noise and invalid data. Unrealistic values or erroneous data will be removed from the dataset.
- Converting data types if there are inconsistent data lines or data that do not fit the model (such as non-numeric variables) will be converted to appropriate forms.

1.5.2. Data processing

After the data is cleaned, the next step is to process the data so that it can be used in the machine learning model. Some common data processing methods we apply are:

- Data normalization and preparation: Numerical features will be normalized to ensure that their values are within the same range, avoiding the influence of different units of measurement. The data normalization method used is StandardScaler to transform the features into a normal distribution with mean = 0 and standard deviation = 1.
- Categorical variable conversion: Categorical variables such as real estate type, sales method will be encoded as numeric values using One-Hot Encoding or Label Encoding techniques.

1.5.3. Model training method

After data preparation and processing, machine learning models are trained to predict property values. The machine learning algorithms applied include: [1].

- Linear Regression: This is a basic model that helps determine the linear relationship between the features and the value of the property. This model will be used as a basis for comparison with more complex models.
- Decision Tree: Decision Tree helps analyze decisions based on input features and allows the model to easily explain the value prediction

decisions. This is a powerful tool in handling non-linear data.

- Neural Networks: Used to model complex relationships between factors affecting the value of the property. Neural networks help improve the accuracy of the model thanks to the ability to learn hidden features in the data.

- Boosted models: Boosting models such as Gradient Boosting and XGBoost are used to improve the accuracy of the model. Boosting models combine multiple weak learners, such as small decision trees, to create a stronger model by correcting the errors of the previous model. This is a powerful method for reducing model bias and variance, especially in complex prediction problems such as real estate prices. Boosted models are trained in multiple iterations, each attempting to correct the errors of the previous model, thereby improving the overall accuracy.

1.5.4. Optimize the model

After training, the models will be optimized to improve accuracy and minimize prediction errors. The optimization process includes the following steps:

- Model parameter optimization: The model parameters will be adjusted to achieve the best performance. Grid Search or Randomized Search [2] techniques will be used to find the optimal parameter set for the models, especially for Boosted models such as XGBoost or Gradient Boosting. Important parameters such as learning rate, number of trees, and max depth will be adjusted to minimize model errors.[3]

- EarlyStopping technique: To avoid overfitting and help the model learn more effectively, the EarlyStopping technique will be applied during the training process, especially for Boosting models and artificial neural networks. This technique stops the training process when the accuracy on the test set no longer improves after a certain number of training rounds (epochs). This helps to minimize the risk of the model overfitting to the training data and improve the generalization ability of the model.[4]

- Regularization techniques: To further optimize the model and avoid overfitting, regularization techniques such as L1 (Lasso) and L2 (Ridge) [5] will be applied to linear models and decision trees. These techniques help to adjust the weights of the features and minimize the influence of unimportant features [6].

- Cross-validation: The optimization process will also be combined with the k-fold cross-validation technique to evaluate the accuracy of the model on different subsets of the data. This helps to ensure that the model not only fits the training set but also has good generalization ability on unseen data [7]-[8].

1.5.5. Test the model through parameters

After the models are trained and optimized, they will be tested through accuracy evaluation indexes such as [9]:

- Mean Mean Square Error (MSE, RMSE): Evaluate the accuracy of the model by measuring the error between the predicted value and the actual value.

- R^2 (R-squared): This index helps evaluate the suitability of the model, showing the proportion of variation in the actual value explained by the model.

- MaE (Mean Absolute Error): Evaluate the average absolute deviation between the actual value and the predicted value of the model.

1.5.6. Reasons for choosing algorithms

In this study, machine learning algorithms are selected based on their ability to handle complex data and require high accuracy in predicting real estate values. The algorithms used include:

- Linear regression: This is a basic and easy-to-understand model that helps determine the linear relationship between input features and predicted values. Linear regression is the first choice for building a model and comparing it with more complex models.

- Decision tree: Decision tree is a powerful algorithm in handling nonlinear relationships. This model is capable of analyzing decisions based on input features and helps explain the model's decisions clearly and easily. This is a useful method when the data is categorical or nonlinear.

- Neural Networks: Neural networks are used to handle complex and nonlinear relationships between features and real estate values. In particular, neural networks have the ability to learn hidden features in data that other models may miss, improving prediction accuracy.

- Boosting algorithms, especially Gradient Boosting: The Gradient Boosting algorithm was chosen because of its powerful ability to improve model accuracy by combining multiple weak learners. This model uses an iterative learning method, with each subsequent training round correcting the errors of the previous model. Gradient Boosting helps reduce bias and variance, thereby improving the overall accuracy of the model. This algorithm is well suited for complex problems such as real estate price prediction, where the data has many nonlinear and uncertain factors.

CHAPTER 2: THEORETICAL OVERVIEW

To serve the research work of building this real estate prediction model, we have studied the advanced process methods and related research in this field to ensure that the built model is not only highly accurate but also practical. We study the process regression methods such as computational linear regression to capture the way to build relationships between the base variables in mathematical prediction. Next, modern machine learning models such as Random Forest, Gradient Boosting, and especially Artificial Neural Networks (Artificial Neural Networks) have been considered and analyzed because of their ability to handle nonlinear data and complex relationships. Finally, we have come to the conclusion and will use the following models to apply in this research: Linear Regression, Gradient Boosting, Decision Tree, Artificial Neural Network, Random Forest, LSTM and Support Vector Machine. Below we will summarize all the theories related to these models.

2.1. Linear Regression

What is Linear Regression ?

Linear Regression is a supervised learning algorithm in Machine Learning, it is a statistical method used to estimate the relationship between independent variables (input features) and dependent variables (output target) [10]. Linear Regression assumes that the correlation between variables is linear thereby finding the best linear function to represent this relationship. This algorithm predicts the value of the output variable from the values of the input variables.

Why is linear regression important?

The interpretability of linear regression is a notable strength. The model's equation provides clear coefficients that elucidate the impact of each independent variable on the dependent variable, facilitating a deeper understanding of the underlying dynamics. Its simplicity is a virtue, as linear regression is transparent, easy to implement, and serves as a foundational concept for more complex algorithms.

Linear regression is not merely a predictive tool; it forms the basis for various advanced models. Techniques like regularization and support vector machines draw inspiration from linear regression, expanding its utility. Additionally, linear regression is a cornerstone in assumption testing, enabling researchers to validate key assumptions about the data.

Types of Linear Regression

There are two main types of Linear Regression: Simple Linear Regression and Multiple Linear Regression [11]. Simple Linear Regression has only one independent variable (input feature) that describes the linear relationship between the dependent variable (output target) and the independent variable. The equation of Simple Linear Regression is in the form $y=a+bx+\epsilon$, where a is the intercept with the y-axis (independent index), b is the slope (slope) of the line and ϵ is the error term. Multiple Linear Regression has more than one independent variable, representing the linear relationship between independent variables and dependent variables. The equation of Multiple Linear Regression has the form $y = a+b_1x_1 + b_2x_2 + \cdots + b_nx_n + \epsilon$, where a is the point of intersection with the vertical axis, $b_1, b_2, .., b_n$ is the angular coefficient and ϵ is the error.

What is the best Fit Line?

The main goal when using linear regression is to determine the best fit line, which means that the error between the predicted value and the actual value should be kept to a minimum. There will be the least error in the best fit line. The Best Fit Line equation provides a straight line that represents the relationship between the dependent and independent variables. The slope of the line indicates how much the dependent variable changes for a one unit change in the independent variables.

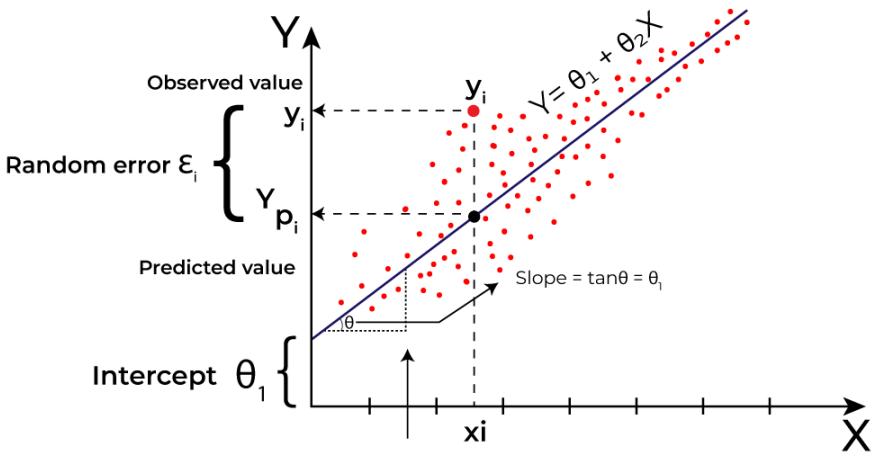

Fig 1: Best Fit Line in Linear Regression

Here Y is called the dependent variable or target variable and X is called the independent variable also known as the predictor of Y. There are many types of functions or modules that can be used for regression. Linear function is the simplest type of function. Here, X can be a feature or multiple features that represent the problem.

Hypothesis function in Linear Regression

The hypothesis function in Linear Regression is defined as follows [12]:

$$\hat{y} = \theta_1 + \theta_2 x.$$

The model gets the best regression fit line by finding the best θ_1 and θ_2 values. To achieve the best-fit regression line, the model aims to predict the target value $\hat{y}$ such that the error difference between the predicted value $\hat{y}$ and the true value *y* is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimizes the error between the predicted y value (pred) and the true y value (y).

Cost function for Linear Regression (MSE)

The cost function in Linear Regression is used to measure the difference between the actual value and the predicted value from the model.It is defined as [12]:

$$\textit{Cost function (J)} = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)$$

Utilizing the MSE function, the iterative process of gradient descent is applied to update the values of θ_1 and θ_2. This ensures that the MSE value converges to the global minima, signifying the most accurate fit of the linear regression line to the dataset. This process involves continuously adjusting the parameters θ_1 and θ_2 based on the gradients calculated from the MSE. The final result is a linear regression line that minimizes the overall squared differences between the predicted and actual values, providing an optimal representation of the underlying relationship in the data.

Gradient Descent for Linear Regression

A linear regression model can be trained using the optimization algorithm gradient descent by iteratively modifying the model's parameters to reduce the mean squared error (MSE) of the model on a training dataset [13]. To update θ_1 and θ_2 values in order to reduce the Cost function and achieve the best-fit line the model uses Gradient Descent. The idea is to start with random θ_1 and θ_2 values and then iteratively update the values, reaching minimum cost. A gradient is nothing but a derivative that defines the effects on outputs of the function with a little bit of variation in inputs. The parameter update procedure in Gradient Descent is defined as follows:

$$\theta_j := \theta_j - a \frac{\partial J(\theta)}{\partial \theta_j}$$

where θ_j is the parameter to be optimized, $J(\theta)$ is the cost function, a is a value that determines the amount of change in each update, also known as the learning rate, and $\frac{\partial J(\theta)}{\partial \theta_j}$ is the derivative of the loss function with respect to parameter θ_j.

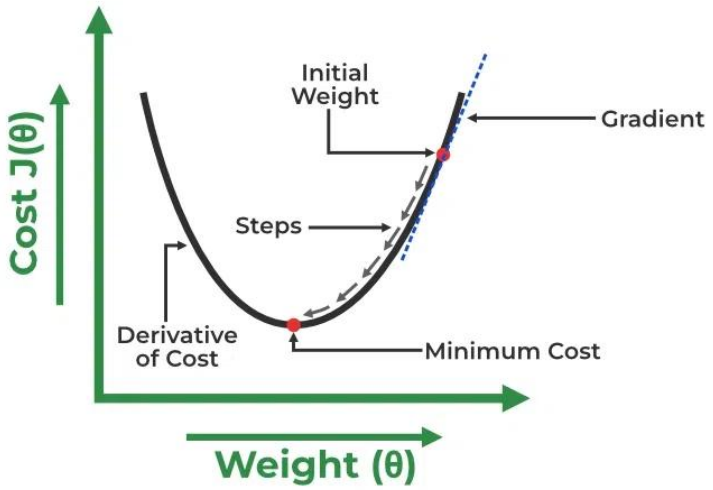

Fig 2: Gradient Descent for Linear Regression

Regularization Techniques for Linear Models

Regularization techniques are used in linear models to prevent overfitting by adding a penalty term to the loss function. This penalty prevents assigning too much importance (large number system) to any feature, improving its chemical ability on unknown data [14]. Here are some commonly used regularization techniques:

Lasso Regression (L1 Regularization)

Lasso Regression is a technique used for regularizing a linear regression model, it adds a penalty term to the linear regression objective function to prevent overfitting. The objective function after applying lasso regression is:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i) + \lambda \sum_{j=1}^{n} |\theta_j|$$

where λ is the regularization parameter that adjusts the strength of the penalty. Lasso Regression produces sparse models because many of the coefficients θ_j can be reduced to exactly zero. At the same time, Lasso automatically selects important features.

Regularization L2 (Ridge Regression)

Ridge regression is a linear regression technique that adds a regularization term to the standard linear objective. Again, the goal is to prevent overfitting by penalizing

large coefficient in linear regression equation. It useful when the dataset has multicollinearity where predictor variables are highly correlated. The objective function after applying ridge regression is:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i) + \lambda \sum_{j=1}^{n} \theta_j{}^2$$

Ridge Regression shrinks all coefficients θ_j close to zero but not all the way to zero and minimizes the effects of multicollinearity.

Early Stopping

Early Stopping stops training early if the error on the test set no longer decreases after a certain number of iterations. It helps reduce the risk of overfitting and is often used with iterative optimization algorithms (like Gradient Descent).

Applications of Linear Regression

Linear regression is used in many different fields, including finance, economics, and psychology, to understand and predict the behavior of a particular variable. For example, in finance, linear regression can be used to understand the relationship between a company's stock price and its earnings or to predict the future value of a currency based on its past performance. And in this very paper, a linear regression model is used to build a real estate price prediction model.

Advantages & Disadvantages of Linear Regression

Advantages of Linear Regression

Linear Regression is a simple and intuitive algorithm that is easy to understand and implement. Its coefficients can be interpreted to provide valuable insights into the relationships between variables, such as the effect of a one-unit change in an independent variable on the dependent variable. It is computationally efficient, making it well-suited for handling large datasets and real-time applications. Furthermore, Linear Regression is relatively robust to outliers compared to other machine learning algorithms, with minimal impact on overall performance. It serves as an excellent baseline model for evaluating more complex machine learning methods. Additionally,

its widespread availability in various machine learning libraries and software makes it highly accessible to practitioners and researchers.

Disadvantages of Linear Regression

Linear Regression has several limitations that affect its applicability in complex scenarios. It assumes a linear relationship between the independent and dependent variables, which may not hold true in real-world data. The algorithm is sensitive to multicollinearity, where highly correlated independent variables can distort the model's coefficients. Linear Regression also struggles with non-linear relationships, as it cannot capture complex interactions among features without transformations. Moreover, it is sensitive to outliers, which can disproportionately influence the model's predictions. Additionally, the assumption of normally distributed residuals and homoscedasticity (constant variance of errors) can be restrictive, limiting its effectiveness when these assumptions are violated.

2.2. Gradient Boosting

What is Boosting?

Before diving into any type of Gradient Boosting, we need to understand a few things to get a better understanding. There is a logic in building machine learning models: "*If one model cannot solve a problem by itself, let many models solve it together*". That is, if one model cannot solve a problem, the solution is to use many weak models to build a high-performing model. In machine learning, to solve the problem of bias-variance trade off, the general solution will be to build many small models and then combine these models to create a superior model.

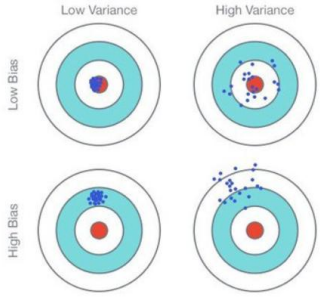

Fig 3: Understanding Bias-Variance Trade-off

One solution that has been proposed at the moment is Bagging [15], which aims to reduce variance, applied to models that already have bias and high variance. Build a large number of models (often of the same type) on different subsamples from the *training_dataset*. These models will be trained independently and in parallel, but their outputs will be averaged to produce the final result. However, Bagging has some weaknesses. First, the models in Bagging are all learned separately, unrelated or have no influence on each other, which in some cases can lead to bad results when the models can learn the same result. Second, Bagging cannot build weak models that can support each other, learn from each other to avoid making the mistakes of previous models. To solve these limitations of Bagging, Boosting was born. The basic idea is that Boosting will create a series of weak models that learn to complement each other. In other words, in Boosting, later models will try to learn to limit the mistakes of previous models.

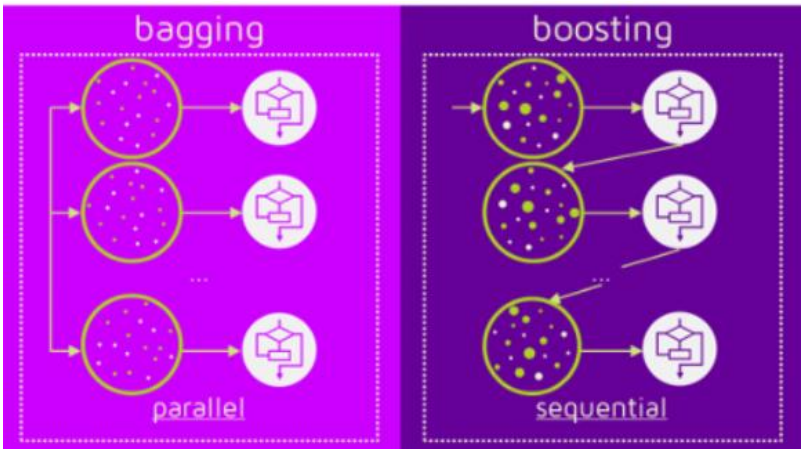

Fig 4: Comparison between Bagging and Boosting

Type of Boosting?

Boosting is divided into two main types, which are AdBoost and Gradient Boosting. AdaBoost trains new models based on reweighting current data points, to help new models focus more on data samples that are being mislearned, thereby reducing value loss of the model. Meanwhile, Gradient Boosting is an optimized version of AdaBoost, which is capable of using more flexible loss functions, optimized for gradient descent, and is more customizable to help solve more complex problems.

Both AdaBoost and Gradient Boosting are algorithms built to solve the following optimization problem:

$$min_{c_n=1:N, w_n=1:N}\, L(y\,, \sum_{n=1}^{N} c_n\, w_n)$$

where L is the loss function value, y is the label, c_n is the confidence score of the nth weak learner, w_n is the nth weak learner.

Adaptive Boosting

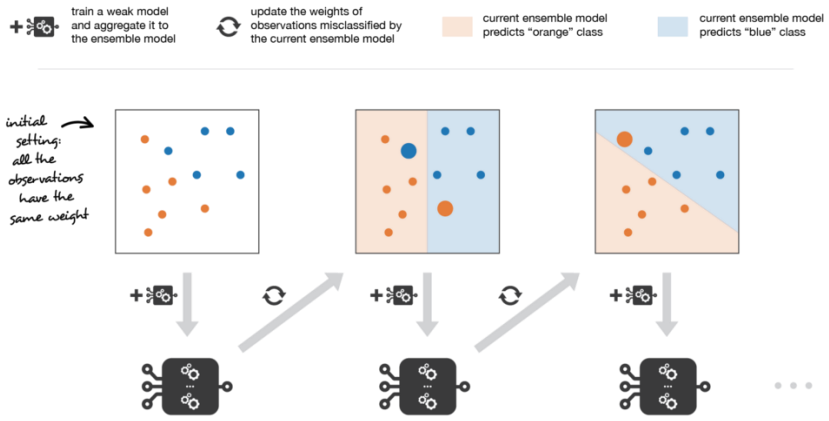

Fig 5: AdaBoost's working process

AdaBoost optimizes through minimizing an exponential loss function, focusing on adjusting the weights of mispredicted samples. Here is the general loss function [16]:

$$L(y, F(x)) = \sum_{i=1}^{m} w_i \cdot exp\,(-y_i \cdot F(x_i))$$

where, w_i is the weight of the ith sample, y_i is the actual label, $F(x_i)$ is the output of the combined model

Optimization steps:

Step 1: Initialization

- Initial sample weight $w_i = \frac{1}{m}$ (each sample has equal weight).

Step 2: Training the weak model

- Train a weak learning model $h_t(x)$ on a dataset with weights w_i

Step 3: Weak model error

Step 4: Calculate the weights a_t of the weak model

Step 5: Update the sample weights

Step 6: Final model

$$F(x) = \sum_{t=1}^{T} a_t \cdot h_t(x)$$

Gradient Boosting

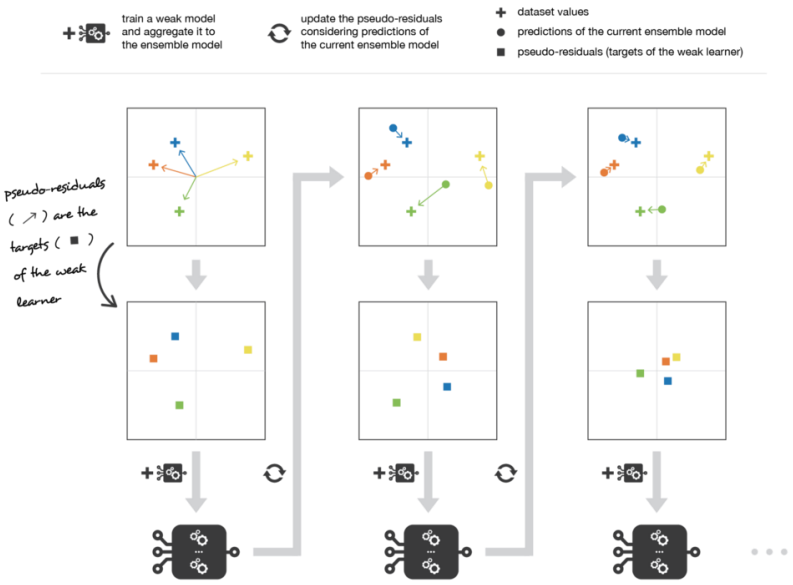

Fig 6: Gradient Boosting's working process

Gradient Boosting optimizes a general loss function *L(y,F(x))* [16]:

$$L(y,F(x)) = \sum_{i=1}^{m} l(y_i, F(x_i))$$

where $l(y_i, F(x_i))$ is a local loss function, such as MSE for regression and Log-Loss for binary classification.

Optimization steps in Gradient Boosting:

Step 1: Initialization

$$F_0(x) = arg\,min \sum_{i=1}^{m} l(y_i, \theta)$$

Step 2: Calculate gradients (pseudo-residuals):

$$r_i{}^{(t)} = -\frac{\partial \mathrm{l}(y_i, F_{t-1}(x_i))}{\partial F_{t-1}(x_i)}$$

Step 3: Training the weak model

Step 4: Model update:

$$F_t(x) = F_{t-1}(x) + v \cdot h_t(x)$$

Step 5: Final model

$$F_T(x) = \sum_{t=1}^{T} v \cdot h_t(x)$$

Compare Adaboost and Gradient Boosting [17]

Characteristic	AdaBoost	Gradient Boosting
Principle	Update misclassified data sample weights	Learn from residuals and improve models via gradient descent
Loss function	Exponential loss	Many different loss functions can be used
Model update method	Add weight to misclassified samples	Add a model to learn residuals (errors) from the previous model
Optimization method	Weighted voting	Gradient descent

Table 1: Compare Adaboost and Gradient Boosting

Applications of Boosting

Boosting algorithms, such as AdaBoost and Gradient Boosting, are widely applied across various domains due to their high accuracy and versatility. In finance, boosting is used for credit scoring, fraud detection, and risk assessment. In healthcare, it helps predict disease risks, patient outcomes, and drug responses. Boosting is also employed in marketing for customer segmentation, churn prediction, and personalized recommendations. In real estate, these algorithms are used for price prediction and property value assessment. Additionally, in natural language processing and image recognition, boosting enhances classification and regression tasks by combining multiple weak learners into a strong predictive model. The adaptability and robustness of boosting make it a cornerstone in modern machine learning pipelines.

Advantages & Disadvantages of Boosting

Advantages of Boosting

Boosting is a powerful ensemble technique that combines multiple weak learners to create a strong predictive model, leading to high accuracy and robust performance. It excels in handling complex relationships and patterns in data, making it suitable for both classification and regression tasks. Boosting reduces bias and variance by iteratively improving the model's predictions, addressing errors made by previous learners. It adapts dynamically to focus on difficult-to-predict samples, ensuring balanced learning. Boosting algorithms like Gradient Boosting, XGBoost, and LightGBM are scalable and can efficiently handle large datasets, making them ideal for real-world applications across various domains.

Disadvantages of Boosting

Boosting, while highly effective, comes with certain disadvantages. It is computationally expensive, especially for large datasets, as it trains models sequentially. This iterative process can lead to longer training times compared to simpler algorithms. Boosting is also sensitive to noisy data and outliers, which can disproportionately influence the model and degrade its performance. Additionally, tuning hyperparameters, such as learning rate and number of iterations, is crucial but can be time-consuming and

complex. Overfitting is another concern if the model is allowed to train for too many iterations without proper regularization. Despite these challenges, careful implementation can mitigate many of these drawbacks.

2.3. Decision Tree

What is a Decision Tree?

A decision tree is a supervised machine learning algorithm. It is used in both classification and regression algorithms. The goal of using a Decision Tree is to create a training model that can use to predict the class or value of the target variable by learning simple decision rules inferred from prior data.

Structure of a Decision Tree

The structure of a regularly defined tree consists of components such as [18]:

- *Root Node*: Represents the entire dataset and the initial decision to be made.
- *Internal Nodes*: Represent decisions or tests on attributes. Each internal node has one or more branches.
- *Branches*: Represent the outcome of a decision or test, leading to another node.
- *Leaf Nodes*: Represent the final decision or prediction. No further splits occur at these nodes.

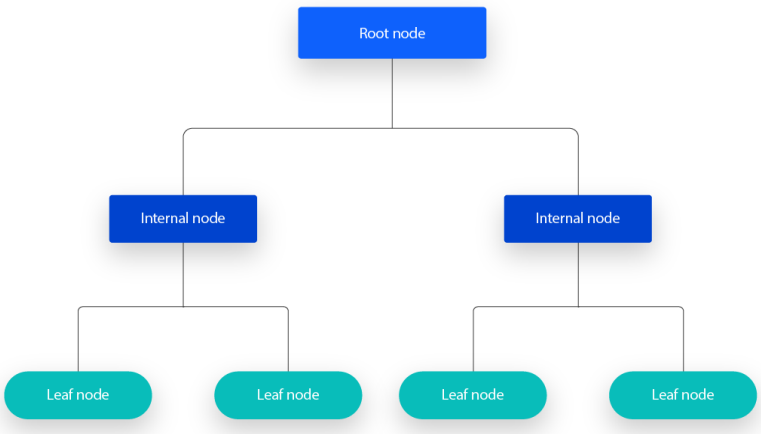

Fig 7: Basic structure of a decision tree

Types of Decision Trees

The types of decision trees are based on the type of target variable we have.

It can be of two types: Classification Tree and Regression Tree [19]. Decision Tree which has a categorical target variable then it called a Classification Tree (right). Decision Tree has a continuous target variable then it is called a Regression Tree (left) in Fig 8 .

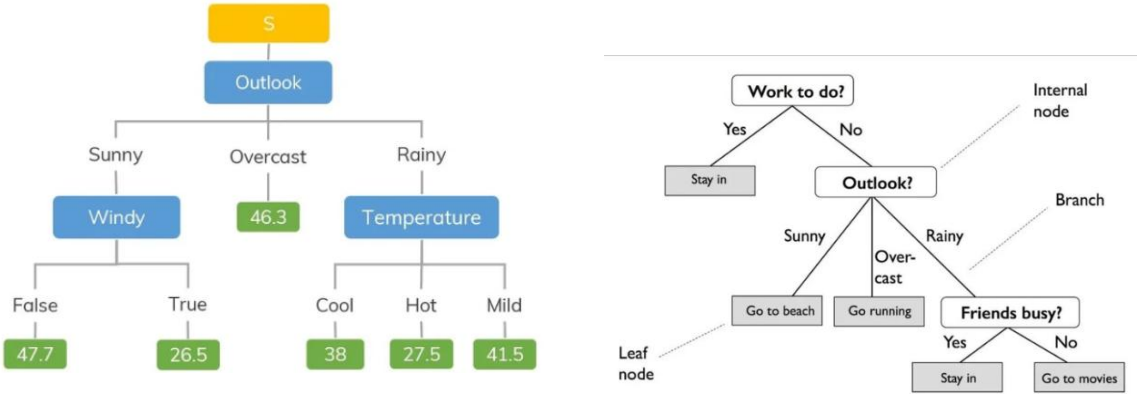

Fig 8: Regression Tree and Classification Tree.

How do Decision Trees work?

Step 1: Start the tree with the root node (S), which contains the complete data set.

Step 2: Find the best attribute in the data set using Attribute Selection Measurement (ASM).

Step 3: Divide S into subsets containing possible values for the best attributes.

Step 4: Create a decision tree node containing the best attribute.

Step 5: Recursively create a new decision tree using subsets of the dataset created in step -3. Continue this process until you reach a stage where you cannot classify more nodes and call the last node a leaf node.

Classical algorithms used in Decision Trees

Decision Trees are built upon a series of classical algorithms that determine the most optimal splits for data at each node. These algorithms are designed to select features and thresholds that maximize the model's predictive performance while maintaining interpretability. Classical approaches often rely on metrics like Gini

Impurity, Information Gain, and Variance Reduction to evaluate splits. Understanding these algorithms provides insight into the foundational mechanics of Decision Trees and highlights their practical applications in both classification and regression tasks.

CART (Classification & Regression Tree)

CART(Classification and Regression Trees) is a variation of the decision tree algorithm. It can handle both classification and regression tasks. Scikit-Learn uses the Classification and Regression Tree (CART) algorithm to train Decision Trees (also called “growing” trees). CART was first produced by Leo Breiman, Jerome Friedman, Richard Olshen, and Charles Stone in 1984 [20]. CART is a predictive algorithm used in Machine learning and it explains how the target variable’s values can be predicted based on other matters. It is a decision tree where each fork is split into a predictor variable and each node has a prediction for the target variable at the end. The term CART serves as a generic term for the following categories of decision trees. With CART classification trees are used to determine which "class" the target variable is most likely to fall into when it is continuous. With regression trees they are used to predict the value of a continuous variable.

CART for Classification

A classification tree is an algorithm where the target variable is categorical. The algorithm is then used to identify the “Class” within which the target variable is most likely to fall. Classification trees are used when the dataset needs to be split into classes that belong to the response variable(like yes or no). For classification in decision tree learning algorithm that creates a tree-like structure to predict class labels. The tree consists of nodes, which represent different decision points, and branches, which represent the possible result of those decisions. Predicted class labels are present at each leaf node of the tree. CART for classification works by recursively splitting the training data into smaller and smaller subsets based on certain criteria. The goal is to split the data in a way that minimizes the impurity within each subset. Impurity is a measure of how mixed up the data is in a particular subset. For classification tasks, CART uses Gini impurity.

$$Gini = 1 - \sum_{i=1}^{C}(p_i)^2$$

The Gini index is a metric for the classification tasks in CART. It stores the sum of squared probabilities of each class. It computes the degree of probability of a specific variable that is wrongly being classified when chosen randomly and a variation of the Gini coefficient. It works on categorical variables, provides outcomes either "successful" or "failure" and hence conducts binary splitting only. The degree of the Gini index varies from 0 to 1, where 0 depicts that all the elements are allied to a certain class, or only one class exists there. Gini index close to 1 means a high level of impurity, where each class contains a very small fraction of elements, and a value of 1-1/n occurs when the elements are uniformly distributed into n classes and each class has an equal probability of 1/n [20].

CART for Regression

CART is a method that constructs decision trees for both classification and regression tasks. For regression, CART builds a tree structure by splitting the dataset recursively to minimize the variance or error in the dependent variable within each resulting subset. CART for regression problem is performed through the following steps:

Step 1: Define the Splitting Criterion

For regression, CART uses the reduction in variance (or Mean Squared Error) as the criterion for splitting. The goal is to split the dataset at a point where the variance of the target variable yyy in each subset is minimized.

$$Variance = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

where y_i ís target values in the subset and $\bar{y}$ is mean of y_i in the subset.

Step 2: Iterative Splitting:

- CART considers all possible splits for each feature to find the one that results in the greatest reduction in variance.

- At each split, the data is divided into two subsets based on the feature value that minimizes the variance.

Step 3: CART stopping criteria

CART stops splitting when a pre-defined maximum tree depth is reached, the number of samples in a node is below a specified threshold. Further splitting does not result in a significant reduction in variance.

Step 4: Prediction

For regression tasks, the leaf nodes of the tree contain the average value of the target variable *y* for all samples in that node. This average is used as the prediction for any new data point falling into that node.

C4.5 Algorithm

The C4.5 algorithm is one of the machine learning algorithms used to build Decision Trees, and is an improved version of the ID3 algorithm, with a number of outstanding features to improve performance and reduce overfitting problem. C4.5 is mainly used in classification problems, where the goal is to group objects into different classes. C4.5 is considered an upgraded version of ID3 with the ability to handle continuous data (continuous data) and define data and algorithm decision tree symbols. C4.5 uses the additional formula Gain ratio to solve the solution of ID3's Information Gain points in choosing the optimal branch. About the formula Entropy, Information Gain and Gain rate we will introduce right after.

With the Entropy formula, this is considered the basis for building the C4.5 algorithm. Entropy is a method of selecting the optimal branching method based on maximizing the amount of information received "Information maximization" which means minimizing the chaos and turbulence in each node "Entropy reduction". The branching nodes selected by this method must represent the maximum information needed so that the decision tree can accurately classify data objects into subsets containing the core, which is the value/probability of the input variable. Similar to the Gini index, if all objects in a node, a subset have the value of the branch, or the subset

itself, then Entropy has a value of 0, on the contrary, if the objects contain many different values, Entropy will be the highest and equal to 1. Entropy is a term associated with Information gain in Information theory and is similar to the "Kullback-Leibler divergence" theory in Machine Learning [21].

$$Entropy\ (t) = -\sum_j p(j|t)\ log_2\ p(j|t)$$

With $p(j|t)$ being the probability of a data object with attribute j of the target variable appearing in node t. As $p(j|t)$ gets larger, $\log_2$ of $p(j|t)$ will have a negative value and approach 0, multiplying by the negative value $p(j|t)$ at the beginning of the formula which is always less than 1, will give a positive value approaching 0. Similar to Gini Index, the smaller the Entropy value is, the more data objects the node or subset contains with any attribute j ($p(j|t)$ will be larger). Below is the Gini Index formula:

$$GINI(t) = 1 - \sum_j [p(j|t])]^2$$

GINI and Entropy have similar performance, but in practice Gini index is often preferred due to simpler calculation, when using the square formula, while Entropy uses the logarithmic function, which can lead to higher computational costs when the data set to be analyzed is very large and diverse in input variables.

Next we learn about Information Gain, which is a formula for determining which of several branching methods will provide the most, clearest, and most complete basis for classifying data objects according to the available values, groups, and analysis classes of the target variable::

$$Information\ Gain = GAIN_{split} = Entropy(p) - (\sum_{i=1}^{k} \frac{n_i}{n}\ Entropy(i))$$

However, many experts believe that Information Gain still has certain limitations, for example, in the case where the variable to be considered has too many distinct values, for example, the variable information about the bank account number with high Information Gain is selected as a branching node. But how can we branch according to the bank account number of each customer? If we do it right away (each branch is an account number), the model may be "overfitting" or completely matching the training

data, so when new customers with different bank account numbers are put into the model, the model cannot be classified because these bank account numbers are not the same as the account numbers when training the model. Information Gain tends to be biased or "biased" for variables containing many different data values. The Gain ratio was therefore born as an upgraded table of Information Gain and limits its shortcomings when considering both the number and size of branches during calculation:

$$GainRATIO_{split} = \frac{GAIN_{split}}{SplitNFO}$$

with $SplitNFO$ has the following formula:

$$SplitNFO = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

with Splitinfo is “available information” about the number of data objects in each branch.

How to avoid Overfitting in Decision Trees?

Decision trees are powerful machine learning models that have been widely used in various domains for their interpretability and effectiveness in classification and regression tasks. However, like any other machine learning algorithm, decision trees are susceptible to overfitting. Overfitting occurs when a decision tree model becomes overly complex, capturing noise or irrelevant patterns in the training data, and fails to generalize well to unseen data. A decision tree can easily become overfit due to its inherent nature of recursively partitioning the feature space. The model may create too many branches or leaf nodes, leading to highly specific rules that only apply to the training set. Some of the effects of Decision Tree Overfitting can be listed as [22]:

- **Poor Generalization**: An overfit decision tree tends to perform poorly on new, unseen data. It becomes excessively specialized in the training set, losing its ability to make accurate predictions on real-world examples.

- **Sensitivity to Noise**: Decision trees can be sensitive to noise and outliers. Overfitting exacerbates this sensitivity, causing the model to incorporate noise patterns into the decision-making process.
- **Increased Complexity**: An overfit decision tree is usually more complex and challenging to interpret. It may involve numerous levels of branches, making it harder to extract meaningful insights from the model.

Based on the effects of this Overfitting, preventive measures were also created. There are many methods used to avoid Overfitting in Decision Trees, but in this research paper, we would like to quote 2 basic and popular measures that are widely used [23]:

Pre-Pruning (Early Stopping)

Pre-pruning, also known as early stopping, involves halting the growth of the decision tree before it becomes fully developed. In this approach, we impose certain conditions (constraints) to stop the tree from growing beyond a certain depth or complexity. This way, the tree is restricted from learning the noise in the training data. The tree reaches a predefined maximum depth. The stopping criterion in Pre-Pruning is met when any of the following conditions are met:

- The tree reaches a predefined maximum depth.
- The number of samples in a node is below a minimum threshold.
- The information gain or Gini impurity reduction from further splits is less than a specified threshold.
- The node becomes pure (all samples in the node belong to the same class).

In pre pruning, we tune hyperparameters like max_depth, max_features etc

Post-Pruning (Pruning After Full Growth)

Post-pruning, also known as cost-complexity pruning, is the process of growing the decision tree fully, allowing it to overfit the training data, and then trimming back unnecessary branches that do not contribute to improving accuracy on unseen data. How pruning works is described as follows:

- **Fully Grown Tree**: Initially, the decision tree is grown to its maximum depth, often leading to overfitting as it captures all the variations, including noise, in the training data.
- **Pruning Process**: Afterward, branches or subtrees are removed if they do not improve performance on a validation set or if they increase the model's complexity without significant improvement in accuracy.
- **Error Complexity Pruning**: Each subtree is replaced with a leaf node if pruning reduces the error on a validation dataset or based on a cost-complexity trade-off.
- **Validation Set**: A portion of the training data is typically reserved as a validation set to guide the pruning process.

Applications of Decision Trees

Decision Trees are versatile algorithms widely used across various domains due to their interpretability and flexibility. In healthcare, they assist in diagnosing diseases and predicting patient outcomes by analyzing medical records. In finance, Decision Trees are employed for credit scoring, risk assessment, and fraud detection. In marketing, they enable customer segmentation, churn prediction, and targeted advertising. In education, they help analyze student performance and recommend personalized learning paths. Decision Trees are also utilized in environmental science for predicting weather patterns and classifying ecological data. Their ability to handle both categorical and numerical data makes them a powerful tool for classification and regression tasks in diverse fields.

Advantages & Disadvantages of Decision Trees

Advantages of Decision Trees

Decision Trees are highly intuitive and easy to interpret, making them suitable for non-technical users. They can handle both numerical and categorical data effectively, requiring minimal preprocessing like normalization or scaling. Decision Trees provide a clear representation of feature importance, helping to understand the relationships between variables. They are versatile, supporting both classification and regression tasks. Furthermore, Decision Trees can handle missing values and outliers without significant performance loss. Their simplicity and transparency make them an excellent

choice for exploratory data analysis and baseline models in machine learning projects.

Disadvantages of Decision Trees

Decision Trees are prone to overfitting, especially when they are deep and capture noise in the training data, leading to poor generalization on unseen data. They are sensitive to small changes in the dataset, which can result in different tree structures and predictions. Decision Trees may also favor features with more levels or categories, potentially introducing bias. Additionally, while they are computationally efficient for small datasets, their complexity increases with large datasets, making them less efficient compared to some other algorithms. Lastly, Decision Trees can sometimes struggle with modeling complex relationships without additional preprocessing or ensemble methods like Random Forests or Gradient Boosting.

2.4. Artificial Neural Networks (ANN)

What is ANN ?

A neural network is a method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning (ML) process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously. Like human brains, artificial neural networks are made up of neurons that are connected like brain cells. These neurons process and receive information nearby from neurons before sending it to other neurons.

Structure of ANN

ANN takes its inspiration from the way the human brain works - making the right connections. Therefore, ANN uses silicon and wires to make its living neurons and dendrites. In the human body, a part of the brain already consists of 86 billion neurons and they are connected to thousands of other cells through Axons. Because humans have many different information inputs from the senses, the body also has many dendrites to help transmit this information.They will create electrical impulses to move, transmit

information in this network of neurons. And the same is true for ANN artificial neural networks - When it needs to process different problems, neurons will send a message to another neuron.

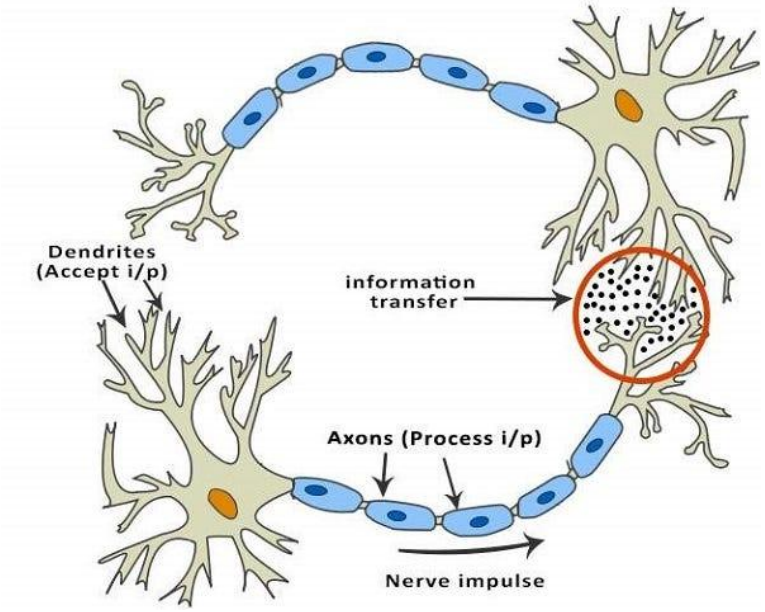

Fig 9: Basic structure of artificial neural network ANN

Therefore, we can say that ANN will consist of many internal nodes, which mimic the biological neurons inside the human brain. ANN networks will connect these neurons with connections and they will interact with each other. The nodes in ANN are used to take input data. Moreover, performing operations on the data is also very simple. After performing operations on the data, these operations are passed on to other neurons. The output at each node is called its activation value or node value.

There are three layers in the network architecture: the input layer, the hidden layer (more than one), and the output layer includes only one layer. A typical feedforward network processes information in one direction, from input to output. Because of the numerous layers are sometimes referred to as the MLP (Multi-Layer Perceptron).

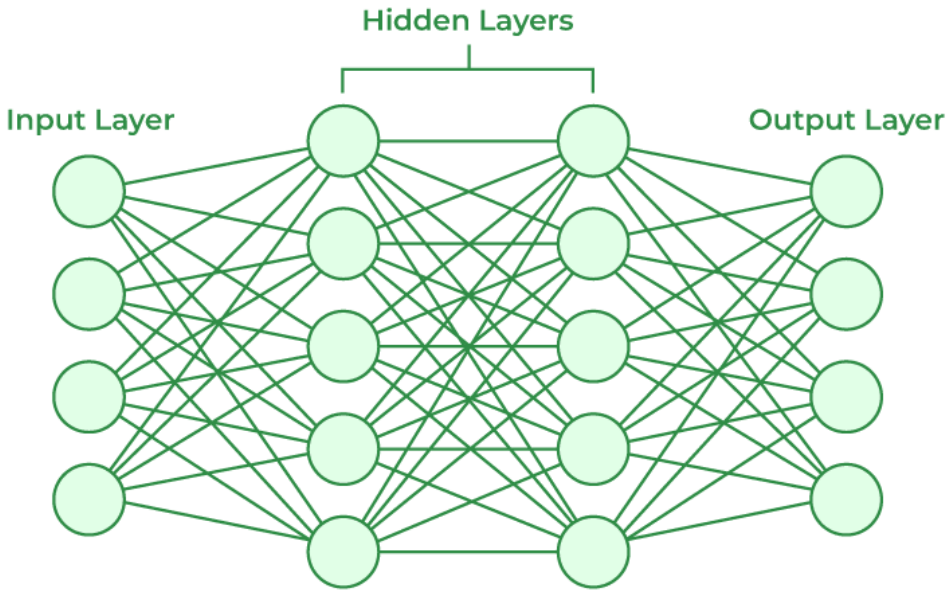

Fig 10:Neural network architecture

The hidden layer can be thought of as a “distillation layer” [24], extracting some of the most relevant patterns from the input and sending them to the next layer for further analysis. It speeds up and improves the efficiency of the network by recognizing only the most important information from the input and discarding redundant information.

Activation function in ANN

The Activation Functions are essential in artificial neural networks (ANNs) as they introduce non-linearity, enabling the network to learn complex, non-linear relationships in data. Without them, the model would behave as a simple linear regression, regardless of the depth. They determine whether a neuron is activated and its contribution to the next layer. Functions like ReLU mitigate vanishing gradient issues, improving training stability in deep networks. Others like Sigmoid and Tanh normalize outputs, enhancing consistency. Choosing the right activation function significantly impacts model performance and convergence, making it crucial for accurate and efficient learning. Below we will list the activation functions used in ANN.

Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

A sigmoid function is a bounded and differentiable function that is nondecreasing and has exactly one inflection point. Roughly speaking, a sigmoid function is a smooth, "S-shaped" curve. Because sigmoid functions "squash" the real values into a bounded interval, they are sometimes called squashing functions. Sigmoid activation has a long-standing tradition in the theory and practice of neural networks [25]. As shown in equation and Fig 11, the Sigmoid function has a structure that produces values between 0 and 1. Hence, it compresses the data between 0 and 1. The Sigmoid function has a derivative. The Sigmoid is a nonlinear differentiable function and is generally used in output layers in binary classification problems

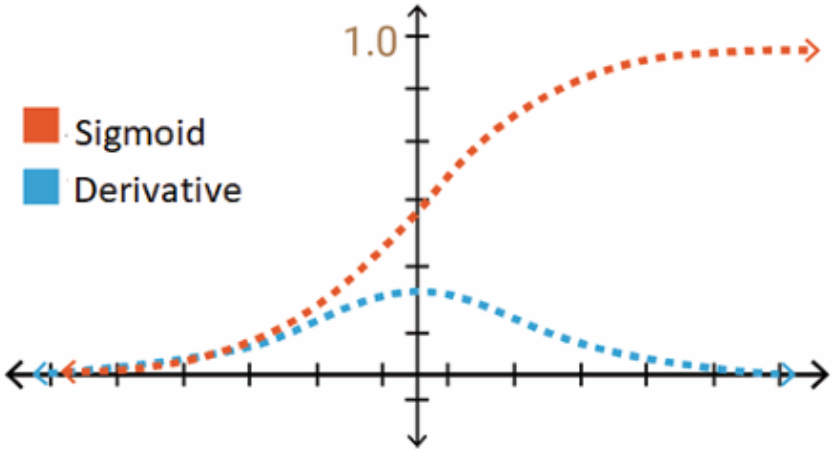

Fig 11:Sigmoid function and derivative

Tanh

$$f(x) = tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Tanh is one of the AFs commonly used instead of the Sigmoid to train ANNs in regression problems from the late 1990s to the early 2000s. It is nonlinear and differentiable everywhere . The Tanh is similar in shape to the Sigmoid, but the Sigmoid has limits from 0 to 1, while the Tanh has limits from -1 to +1. Compared to the Sigmoid, the gradient is steeper. It is preferred over the Sigmoid because it has gradients that are not limited to moving in a particular direction and is zero-centered [26].

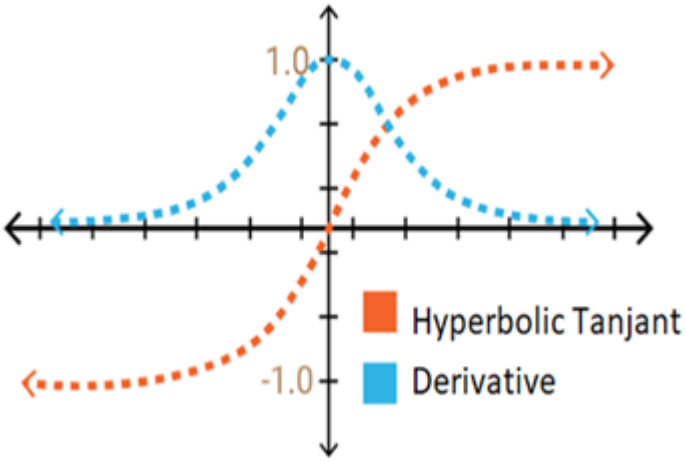

Fig 12:Tanh function and derivative

As shown in equation and Fig 12, the Tanh function has a structure that produces values between -1 and 1. Tanh is quite similar to the Sigmoid. The Tanh, whose derivative is steeper than the Sigmoid, provides more efficient learning because it can take more values. Therefore, it is almost always superior to the Sigmoid. Tanh is a nonlinear differentiable function and is generally used in hidden layers in binary classification problems.

ReLU

$$f(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases}$$

ReLU is one of the most used AFs in ANNs. At input values less than zero, the output is equal to zero, and at input values greater than zero, the output is equal to the input. Since the ReLU only propagates gradients when the input is positive, activating ANNs with the ReLU is simpler than Sigmoid and Tanh. Therefore, using the ReLU makes ANN less regular compared to others . It has become the default AF, having been widely used in ANNs since 2012. Although the ReLU is popular, the fact that input values less than zero are exactly zero prevents negative values from propagating throughout the network [26].

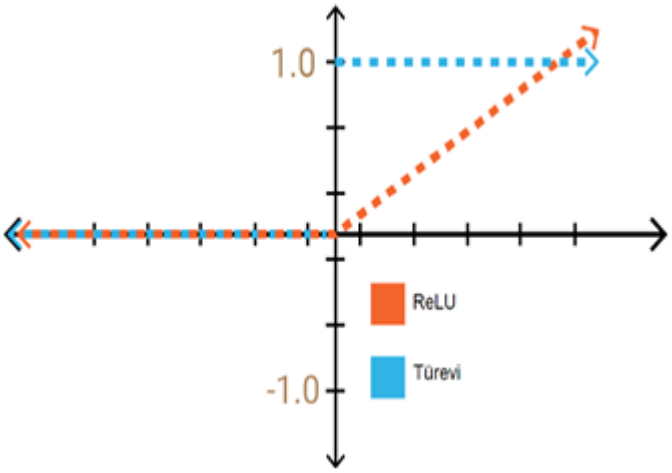

Fig 13:ReLU function and derivative

As shown in equation and Fig 13, the ReLU function has a structure that produces values between 0 and ∞. It produces zero at values of x less than zero, and x at values of x greater than zero. At negative values, neurons take the value zero, which significantly reduces the computational load. Therefore, the ReLU is preferred over Sigmoid and Tanh in multilayer ANNs. It is a very popular and effective AF. On the one hand, the fact that neurons take the value of zero at negative values is a disadvantage of the ReLU. Because, at values of x less than zero, the derivative is zero, so learning does not occur in these regions. ReLU is a nonlinear differentiable function and is generally used in hidden layers in multilayer ANNs.

Leaky ReLU

$$f(x) = \begin{cases} 0.01x, x < 0 \\ x, x \geq 0 \end{cases}$$

Leaky ReLU is a continuous, unbounded, zero-centered, and low-computational AF . It is a modified version of the classic ReLU. At input values less than zero, the output will have a small slope. Although this causes slowness, it can solve the problem of neurons not learning by reducing the problem of dead neurons . The Leaky ReLU, which is fast and simple to calculate, helps prevent saturation problems that may occur in Sigmoid and Tanh. Like the ReLU, the Leaky ReLU is also a very popular AF. Its difference from the ReLU is that the Leaky ReLU can also be differentiated at values less than zero. The Leaky ReLU, which was developed to prevent the problem of

vanishing gradients in training ANNs, performs similarly to the ReLU. This shows that the non-zero slope of the Leaky ReLU over the entire domain does not significantly affect ANN training [26].

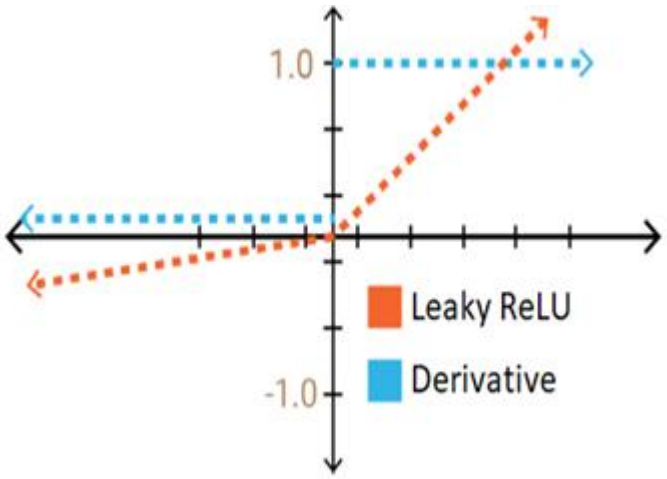

Fig 14:Leaky ReLU function and derivative

As shown in equation and Fig 14 , the Leaky ReLU function has a structure that produces values between -∞ and ∞. It produces a value, albeit small, at values of x less than zero. Since the Leaky ReLU, unlike the ReLU, also processes values less than zero, learning occurs at every stage, but the processing load is high. Although the Leaky ReLU is slightly better than the ReLU because learning occurs at each stage, the ReLU is more popular due to the higher processing load. The Leaky ReLU is a nonlinear differentiable function and is generally used in hidden layers in multilayer ANNs

How do Artificial Neural Networks Learn?

Here is the Steps to learn AI neural Network:

Step 1: Starting Point: Imagine you're building a robot brain, but initially, it knows nothing. So, you randomly assign some strengths to the connections between its "neurons" (like how our brain's neurons are connected).

Step 2: Seeing Data: Now, show the robot some examples of what you want it to learn. For instance, if you're teaching it to recognize cats, show it lots of pictures of cats.

Step 3: Guessing and Checking: The robot tries to imagine what it's seeing based on the strengths of its connections. At first, it'll make lots of mistakes because it's just guessing randomly.

Step 4: Getting Feedback: You tell the robot how wrong its guesses are. For example, you say, "No, that's not a cat; it's a dog." This helps the robot understand where it went wrong and adjust through feedback loops.

Step 5: Adjusting Strengths: The robot tweaks the strengths of its connections based on the feedback. If it guessed wrong, it changes the connections to be a bit stronger or weaker so that next time it might make a better guess. This learning process helps the robot improve its accuracy over time.

Step 6: Practice Makes Perfect: The robot keeps looking at more examples, guessing, getting feedback, and adjusting until it gets better and better at recognizing cats.

Step 7: Testing Skills: Once the robot has seen lots of examples and adjusted its connections a lot, you give it a new picture it hasn't seen before to see if it can correctly identify whether it's a cat or not.

Types of Artificial Neural Networks

Artificial neural networks include the following basic types of artificial neural networks:

- **Feedforward Neural Networks (FNNs)**: These are straightforward networks where information flows in one direction, like from the input to the output. They're used for tasks like identifying patterns in data or making predictions, making them ideal for pattern recognition.
- **Convolutional Neural Networks (CNNs):** Think of these as networks designed specifically for understanding images. They're great at recognizing patterns in pictures, making them perfect for tasks like identifying objects in photos or videos.

- **Recurrent Neural Networks (RNNs)**: These networks are good with sequences, like predicting the next word in a sentence or understanding the context of words. They remember previous information, which helps them understand the current data better.
- **Long Short-Term Memory Networks (LSTMs):** LSTMs are a type of RNN that are really good at remembering long sequences of data. They're often used in tasks where understanding context over time is important, like translating languages or analyzing time-series data.
- **Generative Adversarial Networks (GANs):** These networks are like artists. One part of the network generates new data, like images or music, while the other part critiques it to make sure it looks or sounds realistic. GANs are a key technology in generative AI. GANs are used for creating new content, enhancing images, or even generating deepfakes.

Application of Artificial Neural Networks

Artificial Neural Networks (ANNs) are versatile tools widely applied across various fields. In healthcare, they assist in diagnosing diseases, analyzing medical images, and predicting patient outcomes. In finance, ANNs are employed for credit scoring, fraud detection, and stock market prediction. In marketing, they enable customer segmentation, personalized recommendations, and churn analysis. ANNs play a crucial role in natural language processing (NLP) tasks such as language translation, sentiment analysis, and speech recognition. In computer vision, they are used for object detection, facial recognition, and autonomous driving. The adaptability and learning capabilities of ANNs make them indispensable in solving complex, real-world problems.

Advantages & Disadvantages of Artificial Neural Networks

Advantages of Artificial Neural Networks

Artificial Neural Networks (ANNs) are powerful tools capable of modeling complex, non-linear relationships, making them highly effective for solving real-world problems. They are versatile and can be applied to diverse domains such as image recognition, natural language processing, and predictive analytics. ANNs automatically

learn features from raw data, reducing the need for manual feature engineering. They are scalable and can handle large datasets effectively. Additionally, ANNs are resilient to noise and partial data, making them robust for various applications. Their adaptability and ability to generalize enable them to achieve high accuracy in tasks that involve intricate patterns and relationships.

Disadvantages of Artificial Neural Networks

Despite their power and versatility, Artificial Neural Networks (ANNs) have several drawbacks. They are computationally intensive, requiring significant processing power and memory, especially for large networks. Training ANNs can be time-consuming, and they often demand extensive labeled data for optimal performance. ANNs are prone to overfitting, especially with insufficient regularization or small datasets. Additionally, they function as "black-box" models, making it difficult to interpret or explain their decision-making process. Tuning hyperparameters, such as learning rate, network architecture, and activation functions, is complex and can greatly influence model performance, adding to the challenges of using ANNs effectively.

2.5. Support Vector Machines (SVM)

What is SVM?

Support Vector Machines (SVM) is a supervised machine learning algorithm commonly used in classification and regression problems. In regression problems, the Support Vector Regression (SVR) method is developed based on the principle of SVM with the goal of optimizing the prediction error within the tolerance range ϵ.

How SVM works?

SVM for classification problem (SVC - Support Vector Classification) [27]

Step 1: Find the Separating Hyperplane

The goal of SVM in a classification problem is to find a hyperplane that separates the classes so that the distance between the hyperplane and the nearest points of each class is maximized. These nearest points are called support vectors.

The separating hyperplane can have the equation: $w^T x + b = 0$

where: w is the weight vector (normal vector) of the hyperplane.

b is the bias, which helps to adjust the position of the hyperplane.

Step 2: Maximize Margin

Margin is the distance from the nearest data points (support vectors) to the separating hyperplane. SVM tries to maximize the margin to get better classification ability.

$$Margin = \frac{2}{||w||}$$

The goal of the SVM algorithm is to find w and b such that this margin is maximized, that is:

$$min \frac{1}{2} ||w||^2$$

Step 3: Handling Margin Violations with Slack Variables

In practice, not all data points can be separated without error. Therefore, SVM allows some data points to violate the margin. To do this, the algorithm introduces a slack error variable (ξi) for each data point, allowing them to fall within or exceed the margin.

Objective function with slack variable:

$$min \frac{1}{2} ||w||^2 + C \sum_{i=1}^{N} \xi i$$

where C is a parameter that adjusts the degree of error tolerance. If C is high, the model will try to separate every point correctly; if C is low, the model will accept some error to maximize the margin.

Step 4: Prediction

Once the model is trained, we can use the hyperplane $w^T x + b = 0$ to predict the labels

of new data points. The prediction is done by calculating:

$$\hat{y} = sign\,(w^T x + b)$$

If $(w^T x + b)$>0, the prediction is class +1

If $(w^T x + b)$<0, the prediction is class -1

SVM for regression problems (SVR - Support Vector Regression) [28]

In a regression problem, the goal of SVM is to predict continuous values instead of classifying them into classes. However, the basic steps are still the same as in the classification problem, but instead of separating classes, SVM will search for a hyperplane such that the error between the predicted value and the actual value of the majority of data points do not exceed a permissible threshold ϵ.

Step 1: Find the Regression Hyperplane

The regression hyperplane can be described by the equation: $y = w^T x + b$. The goal of SVR is to find w and b such that the deviation between the actual value y_i and the predicted value $w^T x_i + b$ is within ϵ.

Step 2: Error Bounds With ϵ

SVR only cares about points whose errors exceed ϵ. Points with errors smaller than ϵ are not considered violations, and have no impact on the model. Therefore, SVR will find a hyperplane such that the majority of data points are within the error range of ϵ.

Step 3: Slack Error Variables

SVR also uses slack error variables ξ_i and ξ_i^* to handle points that violate the error ϵ. These error variables allow some points to exceed the error ϵ but must minimize the total error.

The objective function of SVR is: $\min \frac{1}{2}\left||w|\right|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$

With constraints : $y_i - (w^T x_i + b) \leq \epsilon + \xi i$

$$(w^T x_i + b) - y_i \leq \epsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \geq 0$$

Step 4: Prediction

Once the model is trained, the predicted value for the new data point x is calculated as follows:

$$\hat{y}(x) = w^T x + b$$

Using Kernel In SVM :

When the data is not linearly separable in the original space, SVM uses a kernel to map the data into a higher space where the data can be separated more easily. Common kernel functions include [29]:

- Linear Kernel: K(x,y) = $x^T y + c$
- Second-tier kernel (Polynomial): K(x,y) = $(\alpha x^T y + c)^d$
- Gaussian Kernel (RBF): K(x,y) = $\exp(-\frac{||x-y||^2}{2\sigma^2})$

Using kernels allows the model to solve more complex problems without having to directly transform the data into higher space, optimizing computation.

When should you use SVM?

Small or Medium Data

- Data size: SVM works well on small to medium data sets (a few hundred to a few tens of thousands of samples), since the algorithm requires computing the kernel matrix.
- Data distribution: The data does not need to be too large for SVM to work well.

Binary Classification

- SVMs are particularly strong in binary classification problems where the classes are well or nearly well separated.

Highly complex data

- Non-linear data: When the data cannot be separated by a linear hyperplane, the SVM kernel helps map the data to a higher feature space to solve the problem.
- Moderately noisy data: With the appropriate C parameter and kernel, SVM can handle noisy data effectively.

High Accuracy Required

- SVM is often chosen in applications that require high accuracy such as disease detection, image classification, or face recognition.

High-dimensional data

- SVM works well with data that has more features than samples (high-dimensional data), such as text, genetics, or images.

When overfitting is needed

- SVM has good overfitting control by optimizing the margin distance.

Application of Support Vector Machines

Support Vector Machines (SVM) are widely used in various domains due to their robustness and accuracy. In healthcare, SVMs are employed for disease diagnosis, such as detecting cancer or diabetes, by analyzing medical datasets. In finance, they assist in credit scoring, fraud detection, and risk assessment. SVMs are also widely applied in image processing tasks, including facial recognition, handwriting analysis, and object detection. In natural language processing, they are used for text classification, spam email filtering, and sentiment analysis. Additionally, SVMs play a key role in bioinformatics, helping in gene classification and protein structure prediction. Their versatility in handling both linear and non-linear data makes them a powerful tool for a wide range of applications.

Advantages & Disadvantages of Support Vector Machines

Advantages of Support Vector Machines

Support Vector Machines (SVM) are highly efficient for small and complex datasets as they are less dependent on data size, provided the data is of high quality. They excel in handling complex or non-linear data through the use of kernel functions. SVMs have strong generalization capabilities by focusing on support vectors near the decision boundary, reducing the risk of overfitting, particularly in imbalanced or noisy datasets. The flexibility of SVMs lies in their ability to use various kernels (linear, polynomial, RBF, sigmoid), enabling them to tackle complex problems by mapping data into higher-dimensional spaces. Additionally, SVMs effectively handle non-linear data and provide a well-balanced approach to accuracy and simplicity through the C parameter, which adjusts the trade-off between maximizing the margin and minimizing classification errors.

Disadvantages of Support Vector Machines

Support Vector Machines (SVM) face challenges with large datasets due to their reliance on large memory for storing the kernel matrix and the computational complexity of quadratic optimization, leading to longer training times. They are highly sensitive to hyperparameters and kernel selection, requiring extensive experimentation and domain knowledge to tune parameters such as C, ϵ, and kernel-specific settings like γ in RBF. For non-linear kernels, SVM models are difficult to interpret as they do not directly represent features in the original space. Additionally, SVMs are sensitive to noisy data, as outliers can significantly impact model performance, particularly with suboptimal parameter settings. For multi-class problems, SVMs rely on strategies like One-vs-One (OvO) or One-vs-Rest (OvR), increasing complexity and training time. Lastly, SVMs struggle with severely imbalanced datasets, often failing to properly classify smaller classes.

2.6. Random Forest Regression

What is Random Forest Regression?

Random Forest Regression (RFR) is a machine learning method belonging to the ensemble learning group, using multiple decision trees to build a more powerful prediction model. The goal of Random Forest Regression is to predict continuous values (regression) based on input features.

How Random Forest Regression works? [30]

Step 1: Forest initialization

Input: A training dataset D = $\{(X_1, y_1),(X_2, y_2),\ldots,(X_n, y_n)\}$, where X_i are the features and y_i is the target value.

Bootstrap Sampling: Random Forest uses bootstrap technique to generate N subsets of data by randomly sampling with replacement from D. Expression of each bootstrap subset:

$$D_b = \{ (X'_1, y'_1), (X'_2, y'_2), \ldots, (X'_m, y'_m) , m \leq n$$

Step 2: Build each decision tree

For each subset D, Random Forest builds a decision tree following these steps:

- **Select random features at each node:**
  - At each node, randomly select m_{try} features from the total p features to find the best split point.
  - Formula to determine m_{try} in regression problem: $m_{try} = \sqrt{P}$
- **Find the best split point:**
  - Splitting criteria based on Variance Reduction.
  - Variance at node t: $Var(t) = \frac{1}{|S|}\sum_{i \in S}(y_i - \bar{y})^2$

    Where:

    - S : Data set at node t.
    - y_i: Target value.
    - $\bar{y}$: The mean value of y_i

- Reduce variance after node splitting:

$$\Delta Var = Var(t) - (\frac{|S_L|}{|S|}Var(S_L) + \frac{|S_R|}{|S|}Var(S_R))$$

Where:

- S_L : The left dataset after splitting.
- S_T: The right dataset after splitting
- |S|: Total size of the dataset at the node.

- The best split point is the x point that optimizes ΔVar.

- **Continue building the tree:**
  - Continue the splitting process until the stopping condition is met:
    - The tree reaches its maximum depth.
    - The number of samples in the node is less than the minimum value (min_samples_leaf).
- **Prediction at leaf node:**
  - At each leaf node, the predicted value is the average of the target values in the node: $\hat{y}_{leaf} = \frac{1}{|S_{leaf}|}\sum_{j \in S_{leaf}} y_j$

Step 3: Combining predictions from trees:

Once N trees have been built, Random Forest combines the results of all the trees to produce a final prediction:

- Prediction from each tree: Each tree T_i predicts a value $\hat{y_i}$ based on the average at the corresponding leaf node.
- Ensemble prediction: The final prediction value $\hat{y_{final}}$ is calculated by averaging the predictions from all N trees: $\hat{y_{final}} = \frac{1}{N}\sum_{i=1}^{N}\hat{y}_i$

When should you use RFR?

Random Forest Regressor (RFR) is ideal for tasks involving non-linear relationships, high-dimensional datasets, noisy or incomplete data, and situations where feature importance analysis is needed. It handles complex interactions effectively, making it suitable for applications like house price prediction or energy consumption modeling. RFR is robust to outliers and can deal with datasets containing missing values, such as sales forecasts with incomplete records. Additionally, it prevents overfitting by averaging predictions across multiple trees, providing stable results. However, it is best used in scenarios where real-time predictions are not required due to its computational intensity [31].

Applications of RFR?

Random Forest Regressor (RFR) is a versatile and robust algorithm widely used across various domains for predictive modeling tasks. Its ability to handle non-linear relationships, high-dimensional data, and noisy datasets makes it a popular choice for complex real-world problems. Below are some key applications of RFR:

- **Real Estate**: Predicting property prices based on features like location, size, and amenities.
- **Finance**: Forecasting stock prices and analyzing credit risk.
- **Healthcare**: Predicting health metrics such as blood sugar levels or patient recovery times.
- **Retail and E-commerce**: Sales prediction and demand forecasting for inventory management.
- **Environmental Science**: Modeling temperature, air quality, and energy consumption patterns.
- **Bioinformatics**: Predicting gene expression levels or identifying significant biomarkers.
- **Marketing**: Customer behavior analysis and revenue prediction for targeted campaigns.

Advantages & Disadvantages of Random Forest

Advantages of Random Forest

Random Forest achieves high accuracy by aggregating predictions from multiple decision trees, reducing variance and improving precision compared to a single tree. Its use of bootstrap sampling and random feature selection minimizes the risk of overfitting, making the model more stable. Random Forest can handle non-linear relationships, making it suitable for datasets with complex feature-target interactions. It is robust to noise and outliers, as different trees treat such data inconsistently, diluting their overall impact. Additionally, Random Forest supports high-dimensional data, effectively managing heterogeneous or inconsistent features. It also provides feature importance, offering insights into the significance of various factors in the model.

Disadvantages of Random Forest

Random Forest requires significant computational resources due to the construction of multiple decision trees and the processing of random subsets of data, especially for large datasets with numerous features. It is considered a "black-box" model, lacking the interpretability of simpler models like linear regression. The performance of Random Forest decreases with sparse data or datasets with few important features. Additionally, it is unsuitable for real-time applications due to its longer training and prediction times. While robust to noise, excessive outliers in the data can still impact the average predictions, reducing model reliability.

2.7. Long Short-Term Memory

What is Long Short-Term Memory?

Long Short-Term Memory (LSTM) is a special form of Recurrent Neural Networks (RNNs), designed to learn and remember sequential information over long periods of time without being affected by the vanishing gradient problem.

How Long Short-Term Memory works?

LSTM works by processing information through gates. At each time step, the state is updated by combining new information from the input, the previous hidden state, and the memory state. Here is how LSTM works step by step, along with the relevant expressions [32].

Step 1: Get Input Data

- Input at time t:
  - x_t: Input data at step t (e.g., a word or time value).
  - h_{t-1}: Hidden state from previous step.
  - C_{t-1}: Memory state from previous step.

LSTM combines x_t and h_{t-1} to decide how to update the hidden memory and output states.

Step 2: Forget Gate: This gate determines which information from the previous memory state C_{t-1} should be retained or discarded.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- f_t : Forgetting coefficient (value from 0 to 1), indicates what information to forget..
- W_f, b_f: Weights and bias of the forget gate..
- $[h_{t-1}, x_t]$: Combines the input and the previous hidden state.
- σ: Sigmoid function, returns the result to the interval (0, 1).

Step 3: Input Gate: This gate decides what new information from x_t và h_{t-1} should be stored in the memory state.

- **Calculate the memory coefficient***:* $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

  i_t: Memorization coefficient, which determines the level of memorization of new information.

  W_i, b_i: Weights and biases of memristive gates.

- **Create candidate status:** $\tilde{C_t} = \tanh(W_C . [h_{t-1}, x_t] + b_C)$

  $\tilde{C_t}$: Candidate status, which is potential information to be stored in memory.

W_C, b_C: Weights and biases for candidate status.

tanh: Hyperbolic tangent activation function, values in the range (-1, 1).

Step 4: Update Cell State :

Combine the old memory state C_{t-1} with the new information.

$$C_t = f_t . C_{t-1} + i_t . \tilde{C_t}$$

- C_t: New memory state at time t.
- $f_t . C_{t-1}$: Old information is retained.
- $i_t . \tilde{C_t}$: New information is written into memory

Step 5: Output Gate

This gate decides which information from the memory state C_t will be used to generate the hidden output h_t.

Calculate output factor: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

- o_t: Output coefficient, which determines the influence of C_t on h_t.
- W_o , b_o: Weights and biases of output gates.

Calculate hidden output: $h_t = o_t . \tanh(C_t)$

- h_t: The hidden output at time t, is used as input for the next step or prediction result.

Step 6: Iterate Over Time Steps :

The above process is repeated for each time step in the data series. Each step uses h_t and C_t as input to the next step.

When should you use LSTM?

LSTMs are ideal for tasks involving sequential data and long-term dependencies. They excel in time series forecasting, such as predicting stock prices or weather patterns, and in natural language processing (NLP) tasks like sentiment analysis, machine translation, and text generation. LSTMs are also effective in speech and audio processing, enabling applications like voice recognition and genre classification. They are commonly used in sequential image analysis for tasks like video data processing and multimodal applications such as video captioning. LSTMs handle long-term dependencies better than traditional RNNs, making them suitable for datasets where context spans across extended sequences.

Application of LSTM?

LSTMs are widely used in tasks involving sequential data. In Natural Language Processing (NLP)**,** they excel in language modeling, machine translation, and text generation. They are also effective in speech recognition, transcribing audio into text by modeling temporal dependencies. LSTMs are applied in handwriting recognition, processing pen stroke sequences, and image captioning, generating descriptive sentences for images by integrating visual and sequential data. Additionally, they are utilized in time series prediction, forecasting future trends in data like stock prices or weather patterns. These applications highlight LSTMs' versatility in handling complex sequential problems across various fields [33].

Advantages & Disadvantages of Long Short-Term Memory

Advantages of Long Short-Term Memory

LSTM effectively addresses the "vanishing gradient" problem in traditional RNNs, enabling it to learn long-term dependencies in sequential data. Its use of gates (Forget, Input, Output) allows for flexible management of information, making it capable of efficiently storing, recalling, or discarding data as needed. LSTMs excel in handling sequential tasks like time series forecasting, natural language processing (NLP), and speech or sequential image recognition. They can be seamlessly integrated with other models, such as CNNs, to improve performance on complex datasets like

video or multimodal text-image data. Additionally, LSTMs are versatile, supporting both regression and classification tasks across various data types.

Disadvantages of Long Short-Term Memory

LSTMs require higher computational resources compared to traditional RNNs due to the complexity of operations within their gates, leading to longer training times and challenges when deploying on resource-constrained devices. Their performance is heavily dependent on large datasets to effectively learn long-term dependencies; insufficient data can result in overfitting or suboptimal performance. Parameter tuning in LSTMs can be challenging, requiring significant expertise and experimentation. Additionally, while LSTMs handle long-term dependencies better than traditional RNNs, they are not ideal for extremely long dependencies, where advanced models like Transformers often outperform. Without regularization techniques like dropout, LSTMs are prone to overfitting, memorizing training data instead of generalizing well.

CHAPTER 3: INTRODUCTION THE DATASET

The dataset used in training this model is called Melbourne Housing Snapshot, collected from Domain.com.au by *Tony Pino* and uploaded to Kaggle in 2017. The dataset includes detailed information about properties and related attributes, which play an important role in predicting property prices [34].

3.1. Main characteristics of the data set:

The data set includes the following attributes:

1. **Suburb**: name of the suburb.
2. **Address**: address of the real estate.
3. **Rooms**: Number of rooms in the property.
4. **Price**: The selling price of the property (target variable), in dollars.
5. **Method**: Method of sale (eg auction, private sale).
6. **Type**: Type of real estate (eg house, apartment, land).
7. **SellerG**: name of the real estate seller.
8. **Date**: date of real estate sale.
9. **Distance**: Distance from the property to the city center (CBD).
10. **Postcode**: postal code of real estate.
11. **Regionname**: General region where the property is located (eg North, South).
12. **Propertycount**: Total number of properties in the area or suburb.
13. **Bedroom2**: Number of bedrooms, collected from additional sources.
14. **Bathroom**: Number of bathrooms.
15. **Car**: Number of available parking spaces.
16. **Landsize**: Land area (unit: square meter).
17. **BuildingArea**: Construction area (unit: square meter).
18. **YearBuilt**: year of real estate construction.
19. **CouncilArea**: Local council manages real estate.
20. **Latitude**: latitude of real estate.
21. **Longitude**: longitude of real estate.

3.2. Characteristics of the dataset

- Number of samples: The dataset consists of 13519 rows, each row representing a single property.
- Number of attributes: The dataset has 21 attributes, including both numerical and categorical variables.
- Time: This dataset was created in September 2017.
- Missing data: Some attributes such as *Car*, *BuildingArea*, *CouncilArea*, *YearBuilt* and *Landsize* have missing values, and they were handled during preprocessing.

3.3. Purpose of using data

This dataset is used to train and evaluate machine learning models to predict real estate prices. Attributes such as Rooms, Distance, Landsize, and BuildingArea are expected to have strong correlations with the target variable Price.

3.4. Data processing

Prior to training the model, the following data processing steps were performed:

1. Handling missing values.
2. Encoding categorical variables using techniques such as One-Hot Encoding.
3. Normalizing numerical variables to standardize the range of values.
4. Splitting the data into training, testing, and validation sets to evaluate the performance.

This dataset provides a rich source of information to understand the determinants of real estate prices and is the basis for developing and testing predictive models.

CHAPTER 4: EXPERIMENT AND RESULTS

Experiments were conducted on the prepared and processed real estate dataset. Machine learning models including Linear Regression, Decision Tree, Artificial Neural Network, and Gradient Boosting were trained and optimized through methods such as Grid Search, Regularization, and EarlyStopping. The data was divided into training set (80%) and test set (20%) to evaluate the model performance.

4.1. Results achieved:

The prediction performance of the models is compared based on common evaluation indicators in regression problems, including:

- Mean Squared Error (MSE): Measures the mean squared deviation between the predicted value and the actual value.
- Root Mean Squared Error (RMSE): Square root of MSE, helps bring the index to the same units as the data.
- Mean Absolute Error (MAE): Measures the average absolute error.
- R^2 (R-squared): Evaluate the fit of the model.

Model	MSE	RMSE	MAE	R^2
ANN	114.391.816.405,28	338218,59	205195,23	0.72
Decision Tree Regression	124.608.178.228,19	352998,836	214535,55	0.70
Gradient Boosting	80.360.621.250	283479.49	173824,7	0.804
Linear Regression	161.148.400.000	401432,9	270674,6	0.608
LSTM	124.245.888.852,34	352485,3	224569,14	0.69
Random Forest Regression	92.486.985.561,82	304116,73	183320,36	0.77
SVM	312.264.592.700	558806,40	341654,57	0.24

Table 2: Model performance evaluation table

4.2. Overview:

- Gradient Boosting achieves the highest performance with $R^2 = 0.804$, showing good prediction ability and data fit. This model excels due to its ability to minimize errors over many training iterations.

- Artificial Neural Networks also achieve good results with $R^2 = 0.72$ but require more resources and training time.

- Linear Regression and Decision Trees have lower performance with R^2 in the range of 0.6 to 0.7 but are still notable for their simplicity and ease of understanding.

- However, with the SVM model, the prediction performance is very low with $R^2 = 0.24$, showing that this model is not suitable for the problem of predicting real estate prices in current data. The reason may come from the fact that SVM does not handle well data with complexity and strong nonlinear relationships like in this problem, especially when there is no optimal parameter adjustment step or when the data has too many noisy variables.

4.3. Visualization chart:

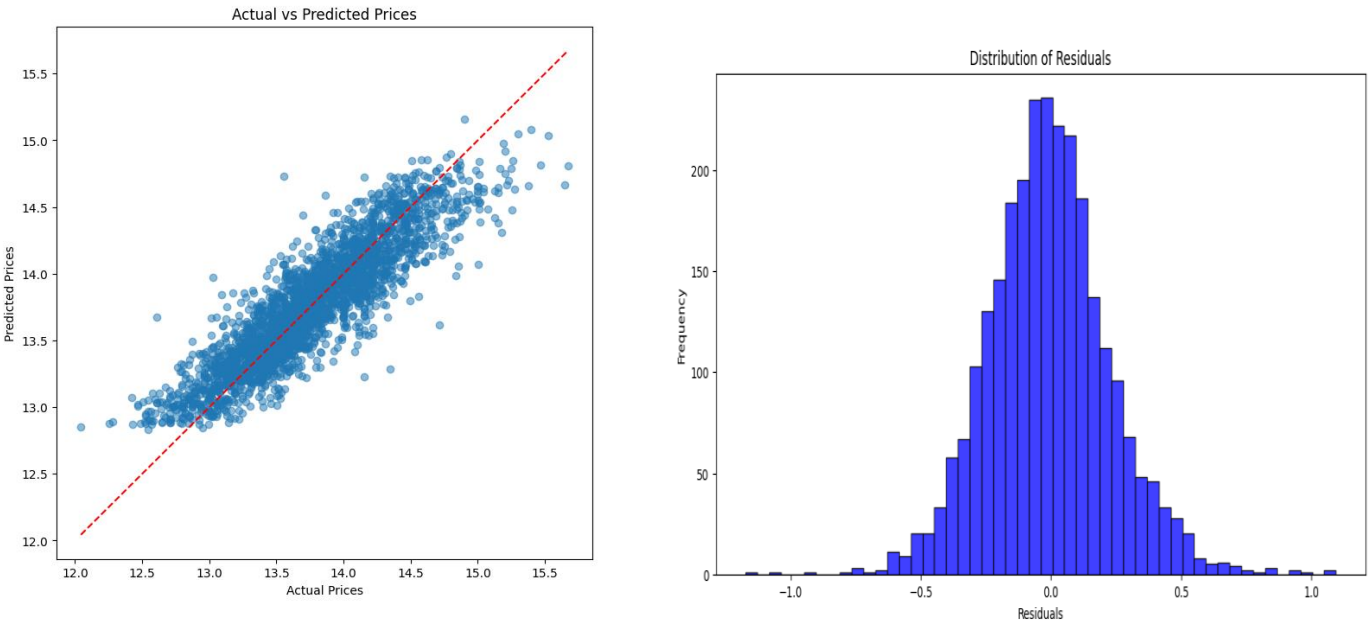

Fig 15: ANN model performance evaluation chart

"Actual vs Predicted Prices" chart:

- The model predicts well with most data points located near the diagonal.
- The error is larger in the extremely low and extremely high value regions.
- Using log-transformation for house prices brings many important meanings in data analysis and modeling, it compresses larger values, making the data distribution closer to the normal distribution, helping to Regression and machine learning models perform better on data with a near normal distribution, this technique helps reduce the negative impact of outliers and makes the model more focused on common values thereby The model becomes more stable and reliable when predicting [35] .

"Distribution of Residuals" chart:

- The residual distribution is normal and symmetric around 0, proving that the model is not biased.
- Some large errors at both ends of the distribution.

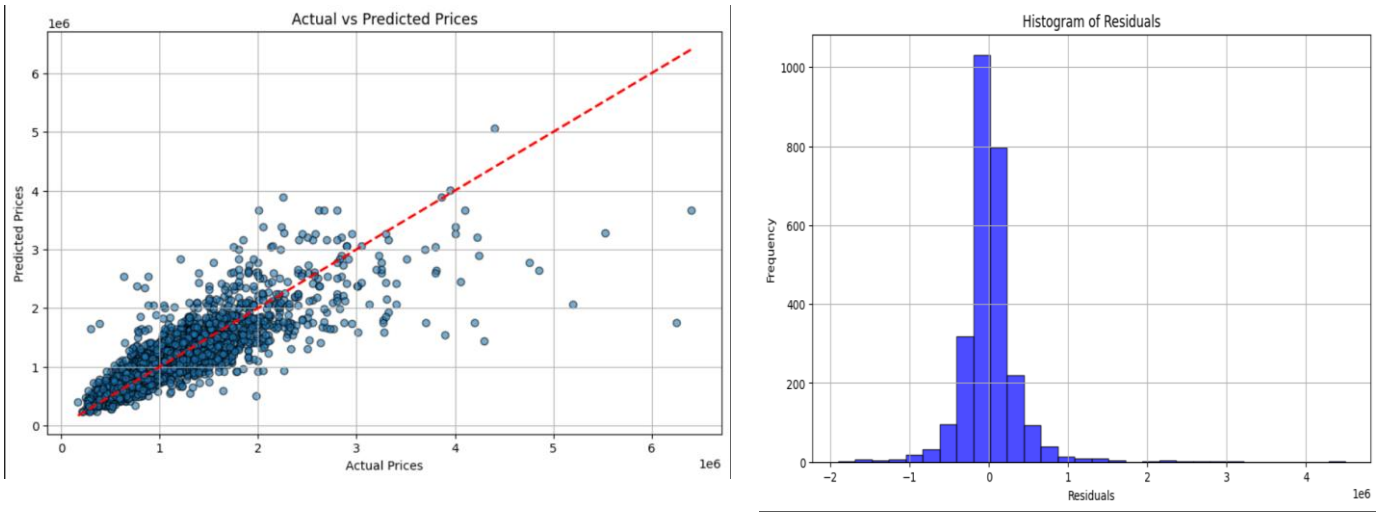

Fig 16: Decision Tree Regression model performance evaluation chart

Actual vs Predicted Prices chart:

- Most predictions are close to actual values, but the model has difficulty with high values (outliers).
- The model tends to lose performance when predicting large values.

Histogram of Residuals chart:

- The residuals are concentrated around the value 0 and close to a normal distribution, showing that the model is unbiased.
- However, there are some large residuals, showing high errors in some samples.

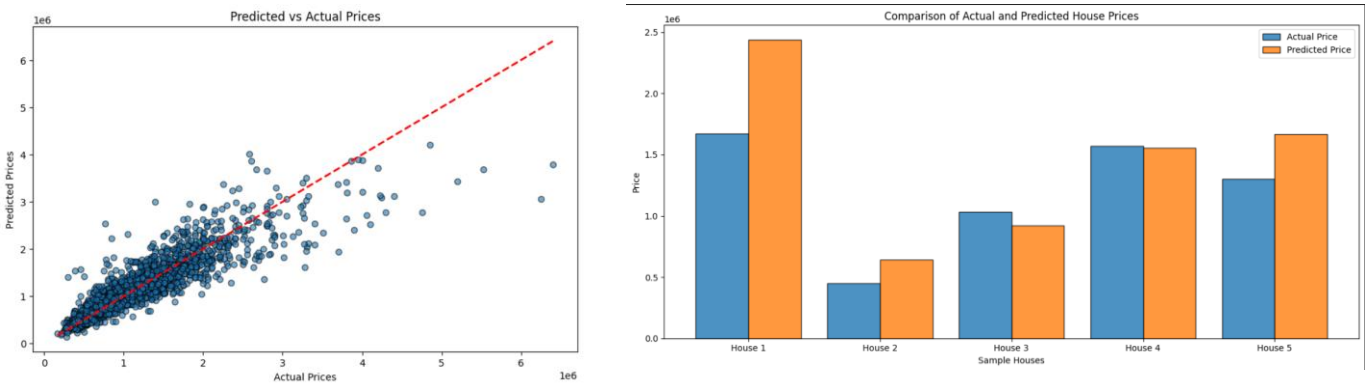

Fig 17: Gradient Boosting model performance evaluation chart

Predicted vs Actual Prices chart:

- The data points are concentrated near the y = x line (red line), showing that the Gradient Boosting model has the ability to predict quite accurately with most of the data.
- However, there are some deviations from the y = x line, especially at high house values (prices greater than 4 million), indicating that the model may not perform well at outliers or anomalous values .

- Gradient Boosting model shows good prediction performance for the majority of data but needs further optimization in exceptional cases.

Comparison of Actual and Predicted Prices chart:

- For any 5 houses taken to test the performance, the predicted price is close to the actual price, demonstrating that the model performs well in these cases.
- However, there are still differences in predicted prices that are significantly lower than actual prices, indicating that the model has not fully identified influencing factors in some cases.
- Gradient Boosting model works well on most samples, but performance may degrade on some specific data samples.

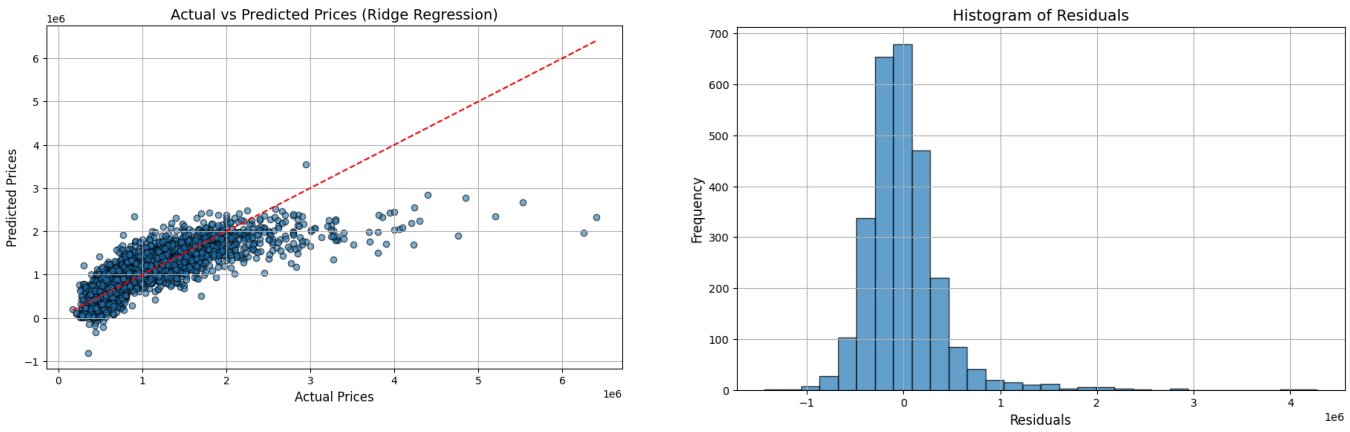

Fig 18: Linear Regression model performance evaluation chart

"Actual vs Predicted Prices" chart:

- The data points (actual and predicted prices) are concentrated quite close to the red diagonal line, showing that the prediction model is relatively accurate for the majority of the data.
- However, at high values (above 2 million) the dispersion increases indicating that the model does not predict accurately at high house values.
- This may be because the data is not evenly distributed or lacks better descriptive

information for high-priced homes.

- The model performs well for most average house price data but performance drops when predicting high house prices.

"Histogram of Residuals" chart:

- The error distribution (residuals) has a relatively standard shape (bell-shaped) with the center near 0, showing that the model is not biased.
- However, there are some large error values (outliers) at both ends of the distribution, especially the positive value side, which shows that the model tends to predict lower than the actual price at some points.
- The model works well with small errors in most data but still has large errors in some cases.

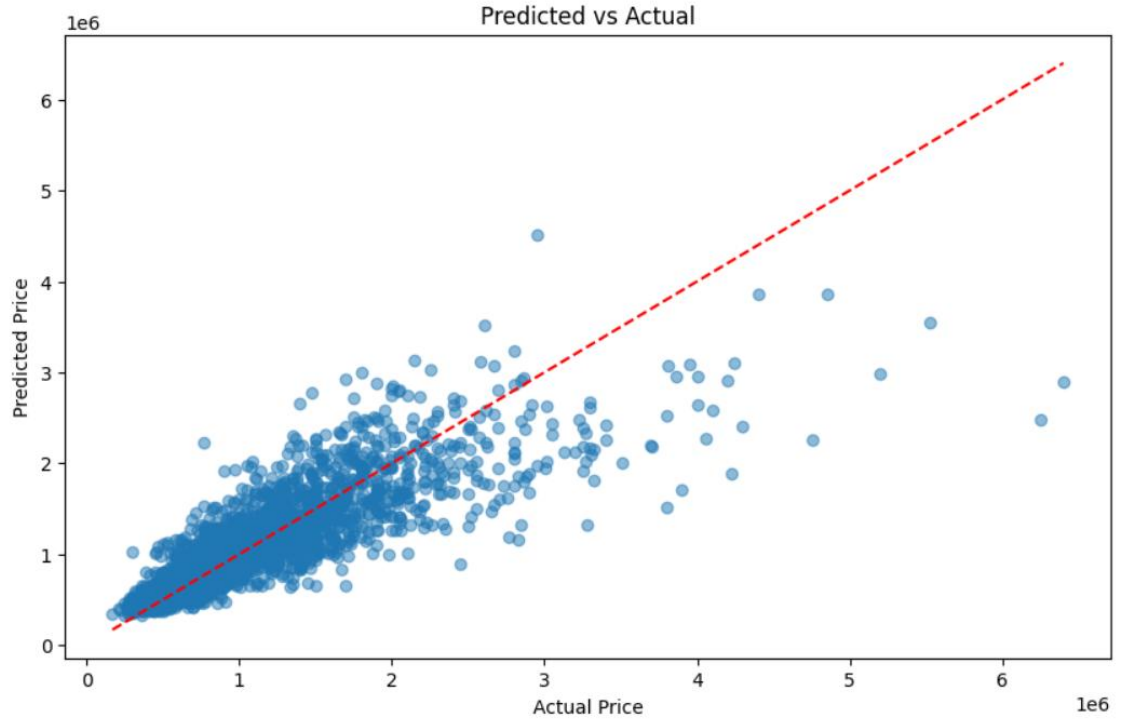

Fig 19: LSTM model performance evaluation chart

Chart description:

- **X axis**: Actual value of house price.

- **Y axis**: Predicted value from LSTM model.
- **Dashed red line**: Diagonal line, representing the ideal relationship between actual and predicted values.

Comment:

- The majority of data points are concentrated near the baseline showing that the model performs well in predicting house prices in the average range (about 0 to 2 million).
- Some data points, especially in the upper right of the chart (values greater than 4 million) stray far from the baseline. This indicates that the model may predict poorly with larger values or possibly because this data contains outliers.
- The model captured the general trend between actual and predicted values with most predictions falling close to the ideal line.

Performance rating:

Advantage:

- The model works well with mid-range values as shown by the distribution close to the standard line.
- This is consistent with the R^2 value (about 0.7 or higher) achieved by the model.

Disadvantages:

- Accuracy is lower at high values, which can increase MSE and MAE.
- The data in the upper right can be outliers that influence the model.

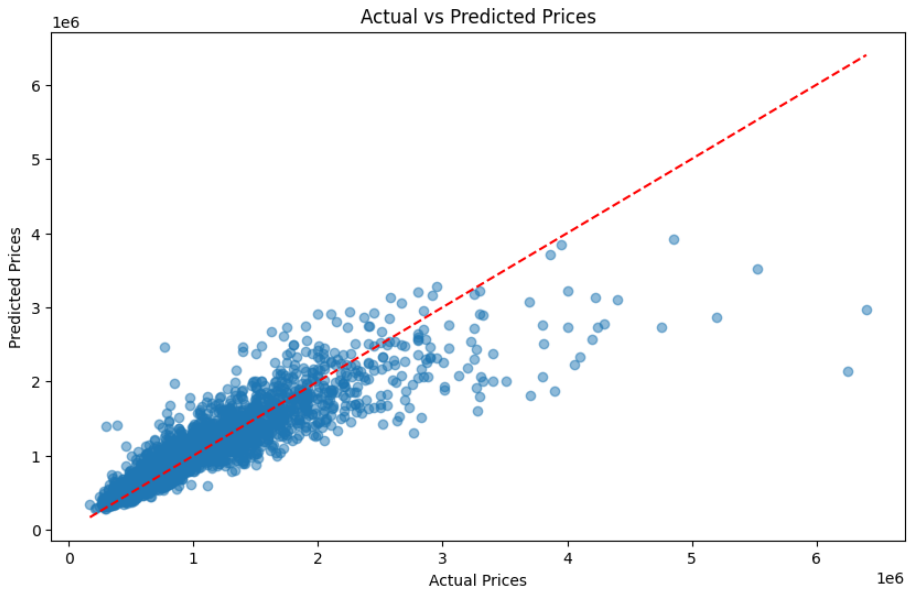

Fig 20: Random Forest Regression model performance evaluation chart

Chart description:

- **X axis**: Actual value of house price.
- **Y axis**: Predicted value from Random Forest Regression model.
- **Dashed red line**: Reference line represents the ideal relationship between actual and predicted values.

Comment:

- Most points lie near the red line, proving that the model works well in predicting house prices.
- At low price ranges (below about 3 million), the points are evenly distributed and closer to the diagonal representing higher accuracy in this segment.
- At high actual values (above 4 million), some points far from the red line show larger errors.
- The model tends to underestimate high values and slightly overestimate low values, which is a common problem in Random Forest due to the averaging nature of the model.

Performance rating:

Advantage:

- The model predicts accurately in the 0–3 million price range, where the majority of test data is concentrated.
- Random Forest works well with data that has outliers and provides reliable predictions.
- Important features of the model (feature importance) help analyze factors affecting house prices.

Disadvantages:

- The model has large errors in predicting high values (>4 million), which may be due to lack of data or insufficiently robust features.
- The model tends to average, leading to poor prediction of outlying values.
- Random Forest with many trees and optimization parameters consumes a lot of time and resources.

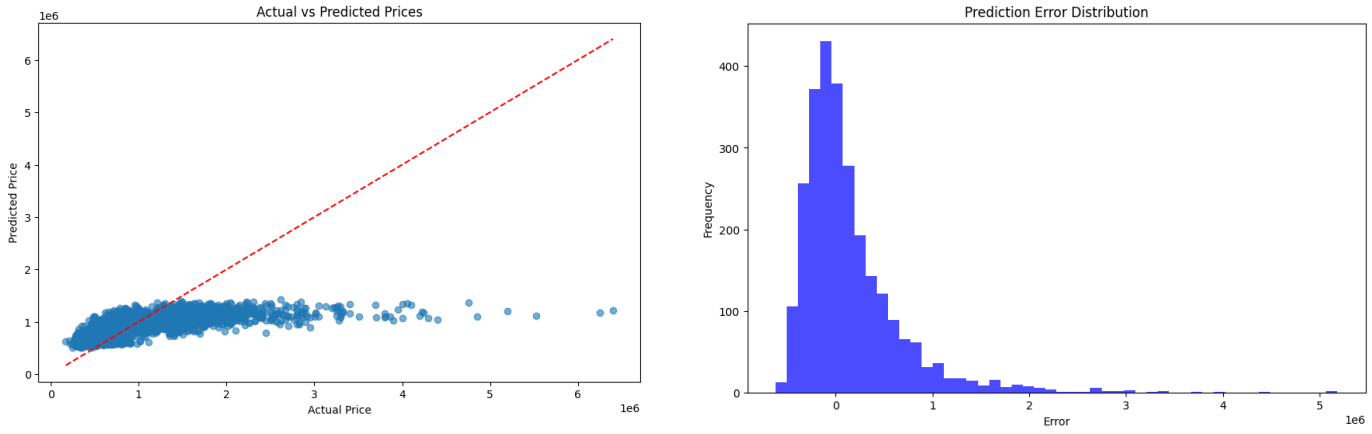

Fig 21: SVM model performance evaluation chart

"Actual vs Predicted Prices" chart:

- The red dotted line represents the ideal situation where the predicted value matches the actual value exactly.
- Most of the points are below the ideal line, showing that the model tends to predict lower than the actual value.
- The model tends to concentrate near the origin. Lower values (below 1 million) are relatively better predicted than high values.

"Prediction Error Distribution" chart:

- The error is mainly in a small range (most predictions are close to the actual value).
- A distribution skewed to the right indicates some large errors, especially with high home values where the model does not predict accurately.
- Appearance of significant errors (greater than 1 million) shows that the model is not very suitable for predicting outliers.

CHAPTER 5: EXPLANATION AND DISCUSSION

5.1. Analyze model results

The experimental results show that Gradient Boosting is the most effective model with $R^2 = 0.804$, outperforming other models such as Artificial Neural Network ($R^2 = 0.72$), Linear Regression ($R^2 = 0.608$), and Decision Tree ($R^2 = 0.7$). Meanwhile, SVM shows low performance with $R^2 = 0.24$, proving that it is not suitable for the current problem.

This result confirms the ability of Boosting models to minimize errors over many iterations and exploit complex data structures well. The high performance of Gradient Boosting also shows the important role of optimizing model parameters (learning rate, max depth, number of trees) and using techniques such as EarlyStopping.

5.2. Effects of data preprocessing

Data preprocessing plays an important role in improving model performance. Handling missing values with variables such as BuildingArea and Landsize added or removed reduced noise and improved data integrity. Normalizing variables using StandardScaler helped Gradient Boosting and the Artificial Neural Network converge faster and achieve better results. Variables such as Method and Type were processed using One-Hot Encoding, making it easier for the model to extract information. These steps demonstrate that clean and well-normalized data is the foundation for high performance of predictive models.

Why data preprocessing is important?

Data preprocessing is an important and indispensable step in every machine learning project [36]. The quality of the data and how it is processed directly affects the performance and accuracy of the model. Here are the main reasons:

Improve data quality

Real-world data often contains many problems such as missing values, noisy data, or invalid values [37]. Steps like:

- Handling missing values (missing values).

- Eliminate or minimize interference.
- Standardize or renormalize features (features scaling), which helps ensure that data is clean and suitable for use.

Increase model accuracy

Data that is not thoroughly processed can lead to models that do not converge or poor performance. In a data set, categorical variables need to be encoded for the model to understand and use them. Normalization or standardization helps the model learn more effectively, especially with algorithms like Gradient Descent.

Handling heterogeneous data

Real-world data is often heterogeneous in type (numeric data, categorical data) or value range. Preprocessing such as One-Hot Encoding or data normalization helps handle this heterogeneity, increasing the model's ability to learn relationships in the data [38].

Reduce the risk of overfitting

Preprocessing helps reduce the impact of unimportant or noisy factors in the data, thereby reducing the risk of overfitting. Techniques like feature selection eliminate unnecessary variables, helping the model focus on important factors.

Speed up training

Standardized and clean data helps algorithms converge faster, reducing training time. For example, normalizing data helps Gradient Descent work more effectively because the features have similar scales.

Ensure data integrity

A predictive model is only effective when trained on quality data. Preprocessing ensures that the input data is reliable, consistent, and does not contain errors that could affect the prediction results [39].

5.3. Significance of the results for the research problem

This research has important implications for several areas of social life. In terms of practical significance, the Gradient Boosting model can be used to support accurate real estate valuation, helping investors, developers and buyers make more effective decisions. Regarding scientific significance, the research strengthens the role of advanced machine learning techniques such as Boosting in solving complex nonlinear and multivariate problems. In addition, the method can be applied to other real estate markets with similar data or other fields such as finance, asset valuation, etc. Experimental results show that with models such as Gradient Boosting, ANN, Decision Tree and Linear Regression the predicted house price is correlated with the actual house price and the scatter plot of the actual house price is gathered and distributed on both sides of the predicted house price. It further demonstrates the reliability of the model in this paper and also illustrates the importance of the study. These results can provide a better multiple regression model to predict real estate prices and help buyers make better reference.

5.4. Limitations of the research

Despite achieving positive results, the research still has some limitations that need to be overcome. As for data quality and size, the existing data is limited in quantity and has missing values in some attributes, affecting the integrity and generalizability of the model. In addition, social and natural factors can impact housing prices every year, making the model not always applicable to predict housing prices of all cities [40]. Artificial Neural Networks and Gradient Boosting require a lot of computational resources [41], making them difficult to implement in resource-constrained systems. The models have not been tested with macro factors such as economic fluctuations or state policies, which could further improve prediction capabilities.

5.5. Scalability and improvement

Since machine learning models have not really achieved high results, there are a number of potential development directions to improve model performance [42]:

Scaling up data

Improving data quality by collecting more diverse data from many different sources, data from many geographical areas or macroeconomic factors such as interest

rates and unemployment rates will help the model become more effective. should be more comprehensive and accurate. Real estate data is a type of data that changes over time, so it is necessary to establish a real-time data update system to maintain the accuracy of the prediction model. If the data is large, it can be used with tools like Apache Spark or Dask to efficiently process big data.

Advanced pre-processing techniques

Handling spatial and temporal factors by integrating spatial (location, region) and temporal (price trends over the years) into the model can significantly improve prediction performance. Apply noise filtering techniques to ensure that input data is not affected by unusual data points or measurement errors.

Improve current model

For the current model, parameters should be optimized, using more modern optimization methods such as Bayesian Optimization or Hyperband to find the best set of parameters for the model. In addition, you can consider using hybrid models by combining different algorithms such as Gradient Boosting combined with Neural Networks to take advantage of the advantages of each algorithm.

Expand the scope of research

Analyze the impact of external factors, further research the impact of macroeconomic factors (such as GDP, interest rate policy) on real estate prices. Expand research to different types of real estate such as townhouses, apartments, or land to increase the practical applicability of the model.

Improved model interpretability

Use model explanation tools like SHAP or LIME to explain important factors in the model's prediction decisions, helping users better understand how the model works.

Enhanced reliability

Combining the ensemble model will help minimize errors and increase the stability of prediction results. In addition, cross-validation techniques with methods

such as k-fold cross-validation to ensure the model has good generalization ability on different data sets.

5.6. Comparison with other research

The topic "Building a real estate price prediction model with machine learning models" is a quite familiar topic to researchers and forecasters, so there have been many previous studies focusing on solving problems. solve this problem. Below are some comparisons between this study and related works:

Linear Regression

Description of the dataset

The research model uses the Ames Housing Dataset, collected by *Dean De Cock*, which is a popular data set for real estate price prediction problems [43]. This dataset includes house price data in Ames, Iowa, from January 2006 to July 2010. With 1,460 samples in the training set and 1,459 samples in the testing set. The target variable used is *SalePrice* and the explanatory variables include characteristics such as: *YearBuilt*, *RoofStyle*, *LotFrontage*, in addition to special characteristics such as quality of swimming pool, fence, and heating system.

Performance rating

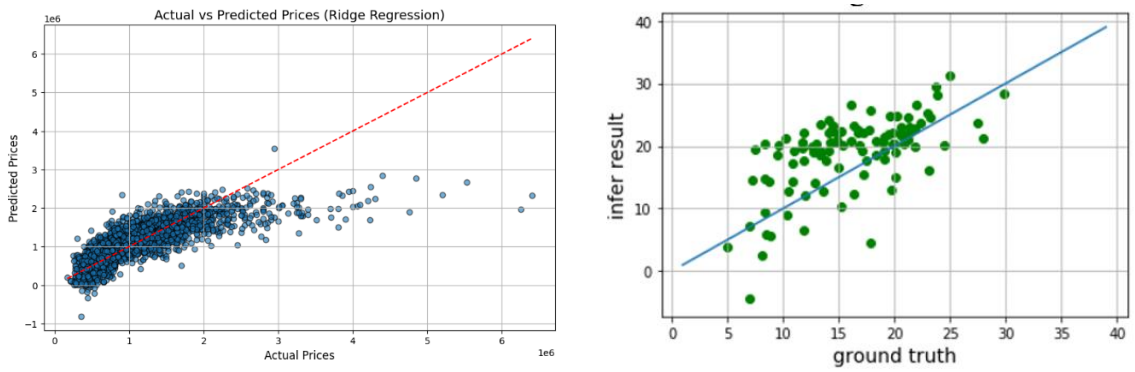

Fig 22: Comparison between two Linear Regression models

Based on the visualization between the two charts, it can be seen that the research uses the Ames Housing data set shown by *Tingjun Mao* (right) with advanced data processing methods and machine learning algorithms to build. The house price prediction model has good performance and high stability, many points are close to the

standard line showing that the model has quite good prediction ability in some cases. However, some points are far from the standard line, especially at the two ends (large or small values) showing that the model does not predict well for values outside the average range. The author of this research also points out the reason why the model has not achieved the desired performance, such as the data has certain limitations and cannot represent the general trend of the entire real estate industry. Because social and natural factors can impact housing prices every year, the model is not always applicable to predict housing prices of all cities. Overall the model has acceptable performance and will be upgraded in the future.

Gradient Boosting

Description of the dataset

Author Esthernjihia's research uses Kaggle's available data set with 80 characteristics of 1460 houses. This data set includes both a training set and a test set to serve model training [44].

Performance rating

```
some_data = housing_train.iloc[:5]
some_labels = housing_label.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)
predictions =gbrt.predict(some_data_prepared)
print("Predictions:\t" ,predictions )
```

```
Predictions:     [207285.1737694  180712.62463749 222139.85344347 142072.05013378
 255736.14750758]
```

```
diff_in_labels = some_labels - predictions
print("The difference between prediction and the actual prices:\t", diff_in_labels )
```

```
The difference between prediction and the actual prices:         0    1214.826231
1     787.375363
2    1360.146557
3   -2072.050134
4   -5736.147508
Name: SalePrice, dtype: float64
```

Fig 23: Research results of the reference Gradient Boosting model

This is the result of research by author *Esthernjihia*, it can be seen that the rate of real estate price prediction with the Gradient Boosting model achieves quite high performance when the prediction error is very small. Thereby, it can be seen that the use of the Gradient Boosting model is an effective method in the problem of real estate price

prediction. Gradient Boosting with the ability to learn from the errors of previous models through the construction of sequential decision trees, has proven its superiority in handling nonlinear and complex relationships between features and target values..

Decision Tree

Description of the dataset

Author Muhammad Ihsan's research uses Kaggle's available data set with 80 characteristics of 1460 houses. This data set includes both a training set and a test set to serve model training [45].

Performance rating

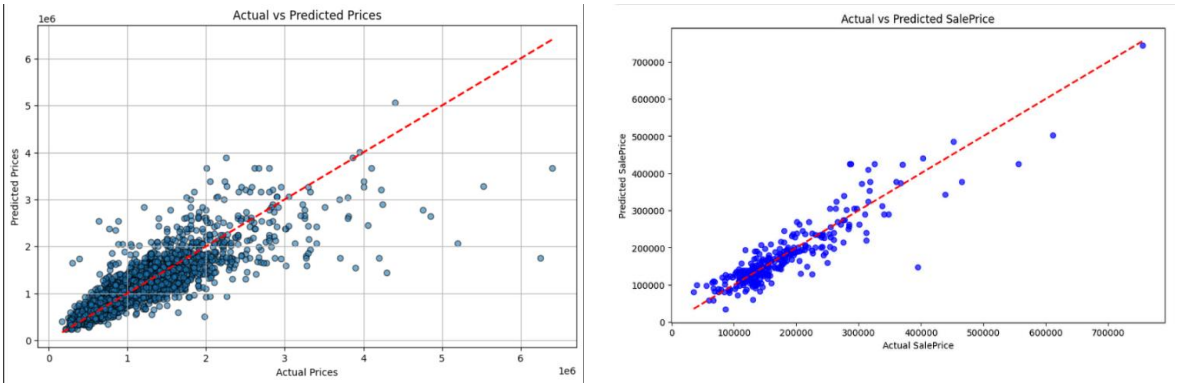

Fig 24: Comparison between two Decision Tree Regression models

The research model using Decision Tree Regressor by author *Muhammad Ihsan* (right) to build a real estate price prediction model achieved a relatively high performance (R^2 = 0.7) approximately equal to the R^2 ratio in this research paper in Table 2. The chart also shows that the model has a fairly good prediction ability in some cases. However, at high values (over 500,000), the points are scattered far from the ideal line showing large errors in prediction. At prices below 300,000, most of the predicted points are close to the ideal line showing that the model works well with common real estate values in the data set. At larger values (over 400,000), the model has difficulty in predicting accurately. Some points deviate far from the ideal line showing that the model may not handle outliers or uncommon values well.

Decision Tree models are prone to overfitting, especially without pruning. The Decision Tree Regressor performs reasonably well at predicting house prices at common (medium and low) price points but is not suitable for large or outlier values.

Artificial Neural Networks (ANN)

Description of the dataset

The paper "*Artificial Neural Networks for Predicting Real Estate Prices*" uses a real estate dataset from the city of Malaga, Spain. The data source was collected from the real estate market in Malaga city, Spain, this is one of the areas with a strongly developed real estate market, with many significant fluctuations in recent years. The data set contains more than ten thousand apartment and condominium transactions including variables: *Area* , *Number of rooms*, *Number of bathrooms*, *Housing condition*, *Geographic location* with target variable *SalesPrice*. Data includes both numerical variables and categorical variables. The dataset focuses on a specific region, Malaga, thus reflecting well the characteristics of the local real estate market [46].

Performance rating

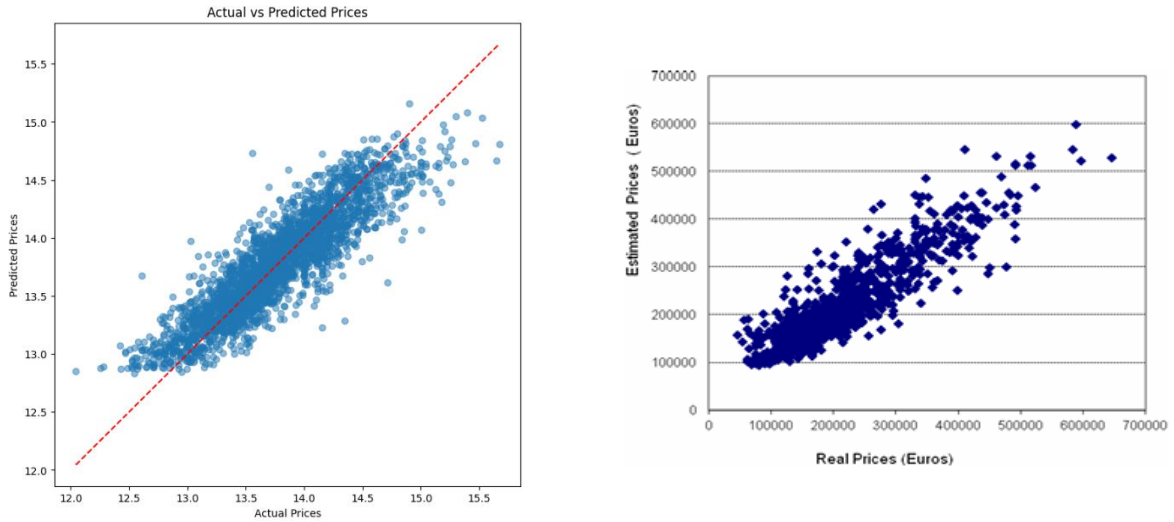

Fig 25: Comparison between two ANN models

Base on the results obtained from the author's research using the real estate data set from Malaga city, Spain (right), it can be seen that the ANN model has quite good performance. Most of the points are close to the ideal line $y=x$, showing that the model is able to predict relatively accurate real estate prices in the majority of cases. The tight

distribution around the ideal line shows relatively small errors for most of the data. With low to medium prices (under 400,000 Euro), the model predicts quite accurately, most points are concentrated close to the ideal line. At high prices (over 400,000 Euros), some points are spread out further, showing larger errors when predicting high values. This could be due to asymmetric data or a lack of representative samples of high values in the training set. Some predicted values are lower than the actual price (underestimating) or higher than the actual price (overestimating), but in general the difference is not too large.

	R^2	RMSE	RESIDUAL SE	MAE	RME
2002	90.21%	21652.25	21634.04	14900.48	13.78%
2003	84.75%	25884.77	25738.99	18999.93	14.75%
2004	90.23%	30983.78	28908.32	24157.36	16.26%
2005	81.12%	32825.44	32755.04	23562.59	13.82%
2006	86.05%	39540.36	39102.13	28551.34	13.69%

Table 3: Goodness of fit measures for the yearly ANN models

The different levels of coverage of the statistics are presented by the authors in Table 3, corresponding to the sample from each year considered. The results were similar over the period considered. Over the years with the collapse of the real estate market, the amount of available data has decreased to a level that makes comparisons over time difficult. However, even though the amount of data has decreased, the ANN model still shows relatively stable prediction ability over the years, with fluctuations mainly coming from changes in external factors such as fiscal policy. interest rates and changing trends in real estate demand.

Although the data is no longer as rich as before, the ANN model still maintains its effectiveness in determining the exact values of factors affecting house prices. Indices such as R^2, MAE, MSE, and RMSE are still at acceptable levels. However, some small errors still appear due to data shortages during crisis periods. In particular, as transaction volume decreases, predictions become less accurate for regions with low data availability.

Long Short-Term Memory

Description of the dataset

This study was authored by *Kuei-Chen Chiu* using a real-world dataset collected with relevant data such as Taiwan housing price index data from 2002 to 2020 to identify the important factors affecting housing prices in Taiwan [47].

Performance rating

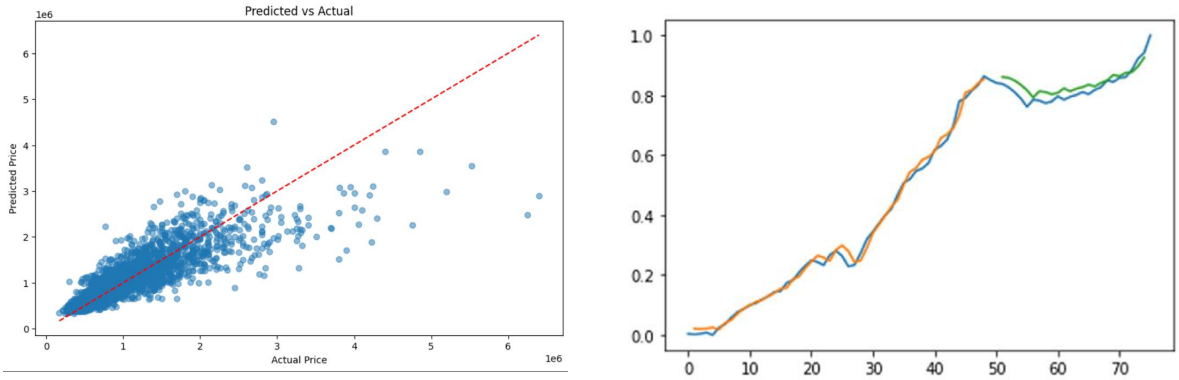

Fig 26: Comparison between two LSTM models

The results of this research (right) were published by author *Kuei-Chen Chiu* in the article, with an accuracy of up to 83.33% and an almost absolute $R^2 = 0.995$, it can explain most of the fluctuations of the dependent variable (target variable) based on the independent variables (explanatory variables). Looking at Figure 26 (right), the X-axis is the period, starting from the beginning of the first quarter of 2002 and divided into quarters, and the Y-axis is the standardized housing price index in Taiwan, the blue line represents the actual value, the orange line shows the predicted value, and the use of the LSTM model to predict housing prices in Taiwan is a good fit.

Random Forest

Description of the dataset

In her research, author *Sarah Lea* uses the California Housing dataset available in the scikit-learn library. With the 'fetch' command, we load the California Housing dataset, which is already included in the sklearn library. Therefore, the dataset is ideal

for starting with new models. Then, the author uses 'pd. DataFrame()' to convert the loaded dataset into a ganda DataFrame. By assigning to 'data.target', the author saves MedHouseVal as the target variable. With this command, we add these values to the DataFrame as a new column [48].

Performance rating

To evaluate the performance of the model, the author uses the MAPE index [49] which is a popular index in evaluating the performance of regression models, especially in value prediction problems. MAPE calculates the absolute error of each prediction as a percentage of the actual value and then calculates the average of all these percentage errors. With the published MAPE index of 19.29%, it shows that the predicted value of the model deviates by an average of 19.29% from the actual value. This means that, in most cases, the predicted value of the model can deviate by about 19.29% from the actual value. This can be considered acceptable for complex problems such as house price prediction, where there are many uncertain factors.

Support Vector Machine

Description of the dataset

In this study the dataset used is the Boston housing dataset, available in the Scikit-Learn library.

Performance rating

The study used a trained SVR model to make predictions on the test data and evaluated the performance of the model using the R^2 score. The results were published with $R^2 = 0.75$ [50] showing that the model was able to explain 75% of the variation in the target variable in the test data set. This shows that the model has reasonable accuracy in predicting house prices based on input features but there are still some points that need to be improved. However, using the SVR model is not considered the optimal choice for the prediction problem, SVM is susceptible to noise and outliers, has low performance with large and volatile data, and requires careful parameter tuning, which is time-consuming [51].

CHAPTER 6: CONCLUSION

In summary, this study investigates a better prediction model for predicting housing prices based on models such as linear regression, reinforcement learning, decision trees and support vectors, and verifies the reliability of the model. Specifically, this paper selects the Melbourne housing price dataset and uses supervised machine learning algorithms to predict real estate prices. Basically, the data is preprocessed with basic processing techniques, and then the price data is analyzed by histograms to confirm the availability of data. To further analyze the relationship between independent variables and dependent variables, the paper uses a matrix graph to conduct correlation analysis and optimizes the anomalies and missing data. Finally, the model is trained and tested. The experimental results show that the actual housing price scatter plots are collected and distributed on both sides of the predicted housing price, which further proves the reliability of the proposed model. However, this study also has certain limitations, that is, other powerful algorithms have not been used for comparison, and the generality of the model has not been further verified. In the future, more advanced deep learning models and more flexible and extensive updated datasets will be applied to the study to achieve stable real-time prediction of housing prices. The main significance of this study is to provide suggestions for housing price forecasting and provide better reference for investors to choose. Overall, these results provide guidance for providing better multiple regression models for housing price prediction and better reference for buyers.

REFERENCES

[1] Pinakin Ariwala. Deep Dive into Predictive Analytics Models and Algorithms. Internet:https://marutitech.com/predictive-analytics-models-algorithms/, 06/12/2024.

[2] Bergstra, J., & Bengio, Y. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research 13*, 2012, p281-305.

[3] Model selection and evaluation. Internet:3. Model selection and evaluation — scikit-learn 1.5.2 documentation, 06/12/2024.

[4] Prechelt, L. *Neural Networks: Tricks of the Trade*, Springer, 2012, p55-69.

[5] Ridge coefficients as a function of the L2 Regularization. Internet:Ridge coefficients as a function of the L2 Regularization — scikit-learn 1.5.2 documentation, 06/12/2024.

[6] Tibshirani, R. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society.* Series B (Methodological), Vol.58, p267-288, 05th Dec-2018.

[7] Kohavi, R. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *14th International Joint Conference on Artificial Intelligence*, 1995, p1137-1143.

[8] Arlot, S., & Celisse, A. A Survey of Cross-Validation Procedures for Model Selection. *Statistics Surveys*, 4, p40-79, 2010.

[9] GeeksforGeeks. Machine Learning Model Evaluation. Internet: Machine Learning Model Evaluation - GeeksforGeeks, 06/12/2024.

[10] Jim Frost. Linear Regression Explained with Examples,Internet: https://statisticsbyjim.com/regression/linear-regression/, 10/12/2024.

[11] Linear Regression (Hồi quy tuyến tính), Internet:https://trituenhantao.io/machine-learning-co-ban/bai-3-linear-regression-hoi-quy-tuyen-tinh/, 10/12/2024.

[12] GeeksforGeeks. Linear Regression in Machine learning, Internet:Linear Regression in Machine learning - GeeksforGeeks, 10/12/2024.

[13] Rukshan Pramoditha. Linear Regression with Gradient Descent. *Data Science*, p1 37, Jun-14-2020

[14] Deanna Schreiber-Gregory, Henry M Jackson. Regulation Techniques for Multicollinearity: Lasso, Ridge, and Elastic Nets. *WUSS, SESUG, SCSUG, MWSUG, PharmaSUG, SAS Global Forum*, 2018, p6.

[15] Bui Tien Tung. Gradient Boosting. Internet:Gradient Boosting - Tất tần tật về thuật toán mạnh mẽ nhất trong Machine Learning, 10/12/2024.

[16] AdaBoost and Gradient Boost – Comparitive Study Between 2 Popular Ensemble Model Techniques, Internet:https://www.analyticsvidhya.com/blog/2020/10/adaboost-and-gradient-boost-comparitive-study-between-2-popular-ensemble-model-techniques/, 10/12/2024.

[17]Gradient Boosting vs AdaBoost: Battle of the Algorithms. Internet:https://dataheadhunters.com/academy/gradient-boosting-vs-adaboost-battle-of-the-algorithms/, 10/12/2024.

[18] GeeksforGeeks. Decision Tree. Internet:https://www.geeksforgeeks.org/decision-tree/, 10/12/2024.

[19] Sean LeSuer. What is a decision tree (parts, types & algorithm examples). Internet:https://slickplan.com/blog/what-is-a-decision-tree#:~:text=There%20are%20two%20types%20of,purpose%20of%20your%20decision%20tree, 10/12/2024.

[20] GeeksforGeeks. CART (Classification And Regression Tree) trong Machine Learning.Internet:https://www.geeksforgeeks.org/cart-classification-and-regression-tree-in-machine-learning/, 10/12/2024.

[21] Thuật toán cây quyết định (P.3): C4.5(Entropy), Internet:https://bigdatauni.com/tin-tuc/thuat-toan-cay-quyet-dinh-p-3-c4-5-entropy.html, 10/12/2024.

[22] Joel Bhaskar Nadar. Overfitting in Decision Tree Models: Understanding and Overcoming the Pitfalls. Internet: Overfitting in Decision Tree Models: Understanding and Overcoming the Pitfalls | by JOEL BHASKAR NADAR | Medium, 10/12/2024.

[23] Abhishek Jain. Pre-Pruning and Post-Pruning in Decision Trees: A Comprehensive Guide.Internet:https://medium.com/@abhishekjainindore24/pre-pruning-and-post-pruning-in-decision-trees-a-comprehensive-guide-391fd3682883, 10/12/2024.

[24] Gourav Guru. Introduction to Artificial Neural Networks. Internet: https://www.analyticsvidhya.com/blog/2021/09/introduction-to-artificial-neural-networks/, 11/12/2024.

[25] Johannes Lederer. Activation Functions in Arti cial Neural Networks: A Systematic Overview. *arXiv,* p3, January 26, 2021

[26] İsmail Akgül. *Academic Studies in Engineering.* Gece Kitaplığı, Octoeber-2010, pp.41-58.

[27] Jakub Nalepa & Michal Kawulok. Selecting training sets for support vector machines: a review. *Springer Nature Link,* Vol52, pages857-900, 2019

[28] Alex J. Smola & Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing,* p200-218, August -2004

[29] Wei Zhang, Wanyi Su, Zeyu Hu, Jinyao Lu, Yin Yu Kevani Chow. A Practical Guide to Support Vector Machines (SVM), Internet: https://medium.com/sfu-cspmp/a-practical-guide-to-support-vector-machines-svm-ccd6a4d4dd04, 07/12/2024.

[30] Varun Samarth. What is Random Forest In Data Science and How Does it Work?. Internet: What is Random Forest In Data Science and How Does it Work?, 07/12/2024.

[31] Christina Ellis. When to use random forests. Internet: When to use random forests - Crunching the Data, 07/12/2024.

[32] GeeksforGeeks. What is LSTM – Long Short Term Memory?. Internet: https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/, 07/12/2024.

[33] Zachary C. Lipton, John Berkowitz, Charles Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv*, p17-31, Jun-5th-2015.

[34]DANB.MelbourneHousingSnapshot,Internet:https://www.kaggle.com/datasets/dansbecker/melbourne-housing-snapshot/data, 07/12/2024.

[35] Log Transformation in Machine Learning (with Python Examples). Internet: Log Transformation in Machine Learning (with Python Examples) | PythonProg, 08/12/2024

[36] Nazri Mohd Nawi, Walid Hasen Atomi, M. Z. Rehman. The Effect of Data Pre-Processing on Optimized Training of Artificial Neural Networks. *The 4th International Conference of Electrical Engineering and Informatics (ICEEI 2013)*, 2013, p32-39.

[37] Tlamelo Emmanuel , Thabiso Maupong, Dimane Mpoeleng, Thabo Semong, Banyatsang Mphago and Oteng Tabona. A survey on missing data in machine learning. *Journal of Big Data*, 140, 2021.

[38] Vasudev. What is One Hot Encoding? Why And When do you have to use it?. Internet:https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f, 08/12/2024

[39] Rahm, E., & Do, H. H. Data Cleaning: Problems and Current Approaches. *ResearchGate,* 2000.

[40] Felix Gauger, Jan-Oliver Strych, Andreas Pfnür. Linking real estate data with entrepreneurial ecosystems: Coworking spaces, funding and founding activity of start-ups. *Data Brief,* 37, p3, 2021.

[41] Han, S., Mao, H., & Dally, W. J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *International Conference on Learning Representations (ICLR),*2016.

[42] Sunil. 8 Ways to Improve Accuracy of Machine Learning Models, Internet:8 Best Ways to Increase Accuracy of Machine learning model, 08/12/2024.

[43]. Tingjun Mao. Real Estate Price Prediction Based on Linear Regression and Machine Learning Scenarios. *BCP Business & Management*, Vol.38, pp401, 2023.

[44] Esthernjihia. Predicting house prices: GradientBoostingRegressor Algorithm. Internet:Predicting house prices: GradientBoostingRegressor Algorithm. | by Esthernjihia | Nur: The She Code Africa Blog | Medium, 08/12/2024.

[45] Muhammad Ihsan. House Price Prediction with Decision Tree Regressor. Internet:https://emhaihsan.medium.com/house-price-prediction-with-decision-tree-regressor-9728064de7da, 08/12/2024.

[46] Julia M. Núñez Tabales, José María Caridad y Ocerin & Francisco J. Rey Carmona. Artificial Neural Networks for Predicting Real Estate Prices. *Revista de Métodos Cuantitativos para la Economía y la Empresa*, Vol.15, p29-44, 2013

[47] Kuei-Chen Chiu. A long short-term memory model for forecasting housing prices in Taiwan in the post-epidemic era through big data analytics. Internet: https://www.sciencedirect.com/science/article/pii/S1029313223000623 ,07/12/2024.

[48] Sarah Lea. Beginner's Guide to Predicting House Prices with Random Forest: Step-by-Step Introduction with a Built-in scikit-learn Dataset. Internet: https://medium.com/@schuerch_sarah/beginners-guide-to-predicting-house-prices-with-random-forest-step-by-step-introduction-with-a-aee81daae3ee, 07/12/2024

[49] Amber Roberts. Mean Absolute Percentage Error (MAPE): What You Need To Know.Internet:https://arize.com/blog-course/mean-absolute-percentage-error-mape-what-you-need-to-know/, 09/12/2024.

[50] Dr. Soumen Atta, Ph.D. Predicting House Prices with Support Vector Regression in Python.Internet: https://medium.com/@soumenatta/predicting-house-prices-with-support-vector-regression-in-python-46ce87686eb7, 07/12/2024

[51] Theodoros Evgeniou & Massimiliano Pontil. Support Vector Machines: Theory and Applications. *Machine Learning and Its Applications, Advanced Lectures*, Sep-2001.