HUTECH UNIVERSITY

FACULTY of INFORMATION TECHNOLOGY

HUTECH

Đại học Công nghệ Tp.HCM

Comprehensive Machine Learning & Deep Learning Approaches for Hotel Room Review Score Prediction & Influenza Proportion Forecasting: Regression, Clustering, Classification, & Time Series Analysis

Final Project in Statistical Machine Learning

Module Code **CMP1045**

Instructor **Quang-Phuc Nguyen**

Student **Hoang-Khang Nguyen**

Student ID **2186400244**

Ho Chi Minh City, November 1, 2024

HUTECH UNIVERSITY
FACULTY of INFOMATION TECHNOLOGY

——————o0o——————

Final Project in Statistical Machine Learning

Comprehensive Machine Learning & Deep Learning Approaches for Hotel Room Review Score Prediction & Influenza Proportion Forecasting: Regression, Clustering, Classification, & Time Series Analysis

Instructor: **Quang-Phuc Nguyen**

Student: **Hoàng-Khang Nguyen**

Student ID: **2186400244**

Ho Chi Minh City, November 1, 2024

Contents

List of Figures

List of Tables

Acronyms & Abbreviations

Table 1: List of Abbreviations and their Definitions

Abbreviation	**Definition**
DDI	Drug-drug Interaction
MLP	Multi-layer Perceptron
GAT	Graph Attention Network
SAGPooling	Self-Attention Pooling
GCN	Graph Convolutional Network
DGNN-DDI	Dual Graph Neural Network for drug–drug interactions prediction
SA-DMDNN	Directed Message Passing Neural Network with Substructure Attention Mechanism
IDOLpro	Inverse Design of Optimal Ligands for Protein Pockets
SBDD	Structure-based Drug Design
DDPM	Denoising Diffusion Probabilistic Model

Chapter 1

OVERVIEW

1.1 Overview

1.1.1 Previous Works

Machine Learning (ML) and Deep Learning (DL) techniques have been extensively applied in both hospitality and healthcare industries to address various challenges. In the hospitality domain, previous works have focused on using ML models for customer satisfaction prediction, market segmentation, and automated classification of services. Traditional models like Linear Regression, Decision Trees, and clustering techniques such as K-means have been widely used for understanding customer preferences and optimizing marketing strategies. Recent advancements include the use of Deep Learning models, such as Convolutional Neural Networks (CNNs) for image analysis and Recurrent Neural Networks (RNNs) for temporal data analysis. In the healthcare sector, ML and DL have been instrumental in disease prediction, trend analysis, and resource management. Time series models, including ARIMA and LSTMs, have been successfully used for forecasting the spread of infectious diseases like influenza. The integration of exogenous features has further improved the accuracy of these models in predicting public health trends, enabling better planning and intervention.

The goal of this project is to develop a comprehensive analysis and prediction framework for various problems related to hotel room reviews and public health data using both Machine Learning (ML) and Deep Learning (DL) techniques. The project focuses on four main tasks: Regression for predicting hotel room review scores, Clustering to segment hotel rooms, Classification to categorize rooms, and Time Series Analysis to predict the proportion of influenza patients in the United States. The data used in this project includes hotel room data crawled from Booking.com and health data related to influenza from the Health_US.csv dataset. This section provides an in-depth overview of the datasets, the problems being addressed, and the methodologies used.

1.1.2 Problem Statements

Regression Analysis for Hotel Room Review Scores

The first problem tackled in this project is to predict review scores for hotel rooms based on data collected from Booking.com. The dataset contains features such as price, location, amenities, and room specifications, which are used to predict the review score given by customers. The regression models employed include both Machine Learning (Linear Regression) and Deep Learning approaches to capture different patterns in the data and compare their performance.

Clustering for Hotel Room Segmentation

The second problem aims to segment hotel rooms into distinct clusters based on their features. This involves grouping rooms that have similar characteristics, such as price range, amenities, and review scores, to help understand different market segments. Clustering methods are used to achieve this, allowing for better targeting of customer preferences and optimization of room offerings.

Classification of Hotel Rooms

The third problem is to classify hotel rooms into categories based on their attributes, such as "Luxury," "Standard," or "Budget." This classification task uses both supervised Machine Learning techniques, which require labeled training data, and ensemble methods to enhance accuracy. The goal is to provide an automated system for categorizing rooms based on input features, which helps in streamlining hotel management and marketing efforts.

Time Series Analysis for Influenza Patients Proportion

The fourth problem focuses on analyzing the proportion of influenza patients in the United States using the Health_US.csv dataset. The dataset contains information about the proportion of influenza patients, referred to as the OT (Influenza Patients Proportion), collected over time. The goal is to forecast future trends in influenza prevalence to help with healthcare resource allocation and planning. Time series models are applied to capture temporal patterns and predict the proportion of influenza patients.

1.1.3 Motivation

The motivation for this project stems from the increasing demand for effective decision-making tools in both the hospitality and healthcare industries. In the hospitality industry, understanding customer preferences and enhancing the quality of services is crucial for maintaining

competitive advantage. Predicting customer satisfaction through review scores, segmenting hotel rooms based on features, and automating room classification are key challenges that can be effectively addressed using Machine Learning and Deep Learning techniques. By providing actionable insights, this project aims to contribute to improved customer experiences and better management strategies.

In the healthcare sector, accurate forecasting of public health trends is essential for resource allocation and intervention planning. The analysis of influenza patient trends using time series models helps public health authorities prepare for potential outbreaks and allocate healthcare resources efficiently. The motivation here is to harness data-driven approaches to support informed decision-making, ultimately contributing to better public health outcomes.

1.1.4 Datasets

Hotel Room Dataset from Booking.com

The dataset used for hotel room analysis was crawled from Booking.com and contains a wide range of features that describe each room. These features include:

- **Review Score:** The rating given by customers, which serves as the dependent variable for the regression problem.
- **Price:** The cost of booking a room, which is used for both regression and clustering.
- **Amenities:** Information about available amenities such as "Free Parking," "Breakfast Included," "Pool," "No Smoking Room," "Family Rooms," "Room Service," "Free Parking," etc., which play a significant role in clustering, classification, and customer satisfaction prediction.
- **Location:** Information about the location of the hotel, such as address and proximity to popular landmarks, which is a key factor in determining the room's attractiveness and is used in both regression and clustering.
- **Star Rating:** The official star rating of the hotel, which helps classify the room type and influence the predicted review score.
- **Images:** URLs of images for each room, which will be used as input for a Deep Learning model (ResNet18) to extract image features. These image features will be combined with the other features to predict the review score, providing a more comprehensive approach to understanding customer satisfaction.
- **Room Type:** Information about the type of room (e.g., single, double, suite), which helps in the clustering and classification tasks.

- **Customer Reviews:** Textual reviews left by customers, which can be processed to extract sentiment scores and used in combination with other features for regression and classification tasks.
- **Availability:** Information about room availability, which can help in clustering rooms based on occupancy rates and availability trends.

The dataset used for hotel room analysis was crawled from Booking.com and contains a wide range of features that describe each room. These features include:

- **Review Score:** The rating given by customers, which serves as the dependent variable for the regression problem.
- **Price:** The cost of booking a room, which is used for both regression and clustering.
- **Amenities:** Information about available amenities such as "Free Parking," "Breakfast Included," "Pool," etc., which play a role in clustering and classification.
- **Location:** Information about the location of the hotel, which is considered a key factor in determining the room's attractiveness.
- **Images:** URLs of images for each room, which will be used as input for a Deep Learning model (ResNet18) to extract image features. These image features will be combined with the other features to predict the review score, providing a more comprehensive approach to understanding customer satisfaction.

The dataset used for hotel room analysis was crawled from Booking.com and contains a wide range of features that describe each room. These features include:

- **Review Score:** The rating given by customers, which serves as the dependent variable for the regression problem.
- **Price:** The cost of booking a room, which is used for both regression and clustering.
- **Amenities:** Information about available amenities such as "Free Parking," "Breakfast Included," "Pool," etc., which play a role in clustering and classification.
- **Location:** Information about the location of the hotel, which is considered a key factor in determining the room's attractiveness.

The goal of using this dataset is to analyze various aspects of hotel room offerings and their impact on customer satisfaction, which is quantified through review scores.

HealthUS.csv Dataset for Influenza Analysis

The Health_US.csv dataset contains data related to public health in the United States, specifically focusing on the proportion of patients diagnosed with influenza, labeled as "OT." The dataset includes temporal information, which makes it suitable for time series analysis. The main features are:

- **OT (Influenza Patients Proportion):** The percentage of patients diagnosed with influenza over time.
- **Date:** The time periods during which the data was collected, which serves as the basis for the time series analysis.

The purpose of using this dataset is to build a predictive model for forecasting influenza trends, which is critical for planning public health interventions.

1.1.5 Methodology

Regression and Deep Learning for Review Score Prediction

To predict hotel room review scores, we use both Linear Regression and Deep Learning models. Linear Regression provides a baseline model, allowing for interpretability and a straightforward approach to understanding feature impacts. On the other hand, Deep Learning models, such as fully connected neural networks, capture non-linear relationships and complex patterns, aiming to improve prediction accuracy.
The performance of the models is evaluated using metrics such as Mean Squared Error (MSE) and R-squared (R^2) to determine how well the model fits the data and how much of the variability in review scores can be explained by the features.

Clustering for Room Segmentation

For clustering, unsupervised learning techniques such as K-means and DBSCAN are applied to segment hotel rooms based on their features. The goal is to identify groups of rooms that share similar characteristics, thus providing insights into different customer segments and enabling more personalized marketing strategies.
The quality of the clusters is assessed using metrics such as Silhouette Score and Davies-Bouldin Index, which measure the cohesion and separation of the formed clusters.

Classification of Hotel Rooms

Classification models, including Decision Trees, Random Forest, and Support Vector Machines (SVM), are used to categorize hotel rooms into predefined categories. The models are trained

on labeled data, with accuracy, precision, recall, and F1-score used as evaluation metrics to determine the effectiveness of the classifiers in distinguishing between different room categories.

Time Series Analysis for Influenza Prediction

The time series analysis for predicting influenza patient proportion is conducted using models such as ARIMA (AutoRegressive Integrated Moving Average) and LSTM (Long Short-Term Memory networks). These models are chosen to capture temporal dependencies in the data and predict future trends in the OT variable.
The performance of the models is evaluated using metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to assess prediction accuracy and model robustness.

Chapter 2

DATA PREPROCESSING

2.1 Data Preprocessing

Data preprocessing is a crucial step in any machine learning pipeline as it ensures that the data is clean, relevant, and ready for model training. In this chapter, we describe the various steps undertaken to clean and preprocess the data collected from Booking.com and Health_US.csv datasets. These steps include handling missing values, encoding categorical features, scaling numerical features, and data visualization to understand the relationships between different variables.

2.1.1 Data Cleaning

The initial data cleaning process involved handling missing values, removing duplicates, and dealing with any inconsistencies in the dataset.

Handling Missing Values

Missing values were addressed by applying different strategies based on the type of feature and the amount of missing data. For numerical features such as **Price** and **Review Score**, missing values were imputed using the mean value of the respective feature. For categorical features like **Amenities** and **Room Type**, missing values were filled using the most frequent category.

The missing values in the **Health_US.csv** dataset, specifically in the **OT (Influenza Patients Proportion)** and **Region** fields, were handled using forward-fill techniques to maintain temporal continuity in the time series data. This ensured that any gaps in the influenza data did not disrupt the overall trend, which is essential for accurate time series modeling.

Removing Duplicates

The dataset was checked for duplicate entries. Duplicate rows were dropped to avoid redundant data that could bias the model. For instance, duplicate hotel room listings were identified based on the combination of extbfHotel Name, extbfLocation, and extbfRoom Type, and subsequently removed.

2.1.2 Feature Engineering

Feature engineering was conducted to extract more informative features and improve the predictive power of the models. The following steps were taken:

Encoding Categorical Variables

Categorical features, such as **Amenities**, **Location**, and **Room Type**, were transformed into numerical form using different encoding techniques. For features with a high cardinality, such as **Location**, target encoding was used to capture the impact of each location on the **Review Score**. Other categorical features were encoded using one-hot encoding to represent each category as a binary vector, which helped the models understand the presence or absence of specific features.

Image Feature Extraction

The **Images** feature in the **Booking.com** dataset contained URLs of room images. These images were processed using a pre-trained ResNet18 model to extract high-level image features. The extracted image embeddings were then concatenated with the other features to create a more comprehensive feature set for predicting **Review Score**. This approach helped capture visual aspects of the rooms that might influence customer satisfaction, such as room aesthetics and cleanliness.

Sentiment Analysis of Customer Reviews

The **Customer Reviews** feature, consisting of textual data, was analyzed to extract sentiment scores. Natural Language Processing (NLP) techniques were employed, using a pre-trained sentiment analysis model to convert each review into a numerical sentiment score. These scores were then used as additional features in the regression and classification models to better understand the relationship between customer feedback and review scores.

Feature Scaling

Numerical features, such as **Price**, **Star Rating**, and extracted image features, were scaled using StandardScaler to standardize the values with a mean of zero and a standard devia-

tion of one. This ensured that all features contributed equally to the model training process, preventing features with larger numerical ranges from dominating the learning process.

2.1.3 Data Visualization

Data visualization was employed to gain insights into the relationships between various features and the target variables.

Correlation Analysis

A heatmap was created to visualize the correlation between numerical features in the dataset. The correlation matrix showed a strong positive correlation between **Price** and **Star Rating**, indicating that higher-rated hotels tend to have higher prices. Additionally, **Review Score** was found to be positively correlated with features like **Amenities** and **Location**, suggesting that better amenities and favorable locations lead to higher customer satisfaction.

Distribution Plots

The distribution of **Review Score** was visualized to understand its spread across different rooms. The histogram showed that the majority of the review scores were skewed towards higher values, indicating overall positive customer experiences. Similar plots were created for features like **Price** and **Star Rating**, providing insights into the range and skewness of these variables.

Box Plots for Outlier Detection

Box plots were used to detect outliers in features like **Price** and **Review Score**. Significant outliers were identified and analyzed to determine whether they were genuine or errors in data entry. For instance, extremely high prices were cross-referenced with the corresponding room types and amenities to determine if they were justified or needed to be removed.

2.1.4 Time Series Analysis Preparation

For the **Health_US.csv** dataset, specific preprocessing steps were taken to prepare the data for time series analysis.

Temporal Consistency

The **Date** feature was converted to a datetime format to ensure temporal consistency, which is crucial for time series modeling. The data was then resampled to ensure that all time points were equally spaced, which is essential for ARIMA and LSTM models.

Handling Seasonality and Trends

Seasonality and trend components were identified using decomposition techniques. The **OT (Influenza Patients Proportion)** variable was decomposed into trend, seasonal, and residual components to better understand underlying patterns. This decomposition helped in determining the appropriate parameters for the ARIMA model and in preparing the data for LSTM training.

Chapter 3

PROPOSED MACHINE LEARNING METHODS for REGRESSION PROBLEMS

3.1 Linear Regression

Linear regression is one of the most fundamental models in machine learning, primarily used for predicting a continuous dependent variable. Mathematically, the model can be represented as follows:

$$y = \mathbf{X}\boldsymbol{\beta} + \varepsilon \tag{3.1}$$

where:

- y is the $n \times 1$ vector of observed values of the response variable.
- $\mathbf{X}$ is the $n \times p$ matrix of predictors, where each row corresponds to one observation and each column corresponds to one feature.
- $\boldsymbol{\beta}$ is the $p \times 1$ vector of regression coefficients.
- ε is the $n \times 1$ vector of random errors, assumed to be normally distributed with mean $\mathbf{0}$ and variance $\sigma^2\mathbf{I}$.

The goal is to estimate the vector $\boldsymbol{\beta}$ that minimizes the sum of squared residuals:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} (y - \mathbf{X}\boldsymbol{\beta})^T (y - \mathbf{X}\boldsymbol{\beta}) \tag{3.2}$$

The closed-form solution, often referred to as the Ordinary Least Squares (OLS) estimate, is given by:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T y \tag{3.3}$$

3.1.1 Assumptions of Linear Regression

Linear regression is based on several key assumptions that need to be satisfied for the model to be valid and for the estimates to be unbiased and efficient:

- **Linearity:** The relationship between the independent variables and the dependent variable is linear.
- **Independence:** The residuals are independent, meaning there is no correlation between consecutive residuals.
- **Homoscedasticity:** The residuals have constant variance across all levels of the independent variables.
- **Normality:** The residuals are normally distributed.
- **No Multicollinearity:** The independent variables are not highly correlated with each other.

3.1.2 Estimating and the Gauss-Markov Theorem

The Ordinary Least Squares (OLS) estimator $\hat{\boldsymbol{\beta}}$ is known to be the Best Linear Unbiased Estimator (BLUE) under the assumptions of the Gauss-Markov theorem. This means that $\hat{\boldsymbol{\beta}}$ has the smallest variance among all linear unbiased estimators. The Gauss-Markov theorem relies on the following assumptions:

- The model is linear in parameters.
- The expectation of the error term is zero: $E[\varepsilon] = 0$.
- Homoscedasticity: $\text{Var}(\varepsilon) = \sigma^2 I$.
- No perfect multicollinearity among the predictors.

3.2 Sum of Squares in Linear Regression

In linear regression, the total variation in the response variable y can be decomposed into different components using the concept of sum of squares:

3.2.1 Total Sum of Squares (SST)

The Total Sum of Squares (SST) measures the total variation in the observed data and is defined as:

$$SST = \sum_{i=1}^{n}(y_i - \bar{y})^2 \tag{3.4}$$

where $\bar{y}$ is the mean of the observed response values. SST represents the total variability in the response variable y.

3.2.2 Regression Sum of Squares (SSR)

The Regression Sum of Squares (SSR) measures the variation explained by the regression model and is defined as:

$$SSR = \sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2 \tag{3.5}$$

where $\hat{y}_i$ is the predicted value for the i-th observation. SSR represents the variability in y that is explained by the model.

3.2.3 Residual Sum of Squares (SSE)

The Residual Sum of Squares (SSE), also known as the Sum of Squared Errors, measures the unexplained variation after fitting the regression model and is defined as:

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.6}$$

SSE represents the variability in y that is not explained by the model.

3.2.4 Relationship between SST, SSR, and SSE

The relationship between these sums of squares is given by:

$$SST = SSR + SSE \tag{3.7}$$

This equation represents the partitioning of the total variability in the response variable into the portion explained by the model (SSR) and the portion that remains unexplained (SSE).

3.2.5 Coefficient of Determination (R^2)

The coefficient of determination, R^2, is a measure of the proportion of the variance in the response variable that is explained by the regression model. It is defined as:

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} \tag{3.8}$$

R^2 ranges from 0 to 1, where a value closer to 1 indicates that a larger proportion of the variance is explained by the model, implying a better fit.

3.2.6 Adjusted R^2

The Adjusted R^2 is a modified version of R^2 that adjusts for the number of predictors in the model. It is used to prevent overestimation of the model's explanatory power, especially when adding more predictors. Adjusted R^2 is defined as:

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{3.9}$$

where n is the number of observations and p is the number of predictors. Unlike R^2, the Adjusted R^2 can decrease if the added predictors do not improve the model, making it a more reliable metric for model comparison.

3.3 Multicollinearity and Variance Inflation Factor (VIF)

Multicollinearity occurs when two or more predictor variables are highly correlated, leading to instability in the estimated regression coefficients. Mathematically, this results in $\mathbf{X}^T\mathbf{X}$ being nearly singular, which makes the OLS solution very sensitive to small changes in the data. Multicollinearity inflates the variance of the coefficient estimates, making them unreliable.
One common method to detect multicollinearity is the Variance Inflation Factor (VIF). For a given predictor x_j, the VIF is defined as:

$$\text{VIF}(x_j) = \frac{1}{1 - R^2_j} \tag{3.10}$$

where R^2_j is the coefficient of determination when x_j is regressed on all the other predictors. A high VIF value (typically greater than 10) indicates severe multicollinearity.

3.3.1 Consequences of Multicollinearity

The main consequences of multicollinearity include:

- Increased standard errors of the regression coefficients, leading to wider confidence intervals.
- Decreased statistical power to detect significant predictors.
- Coefficient estimates that may change erratically in response to small changes in the model or data.

To mitigate multicollinearity, some common techniques include:

- Removing highly correlated predictors.

- Combining correlated predictors using dimensionality reduction techniques like Principal Component Analysis (PCA).
- Applying regularization techniques such as Ridge Regression or Lasso.

3.4 Root Mean Squared Error (RMSE)

The performance of the linear regression model is often evaluated using metrics like the Root Mean Squared Error (RMSE), which measures the average magnitude of the residuals:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3.11}$$

where y_i and $\hat{y}_i$ are the observed and predicted values, respectively. RMSE provides a straightforward interpretation of the model's prediction error in the same units as the response variable.

3.5 Ridge Regression

To address the problem of multicollinearity, Ridge Regression introduces a regularization term to shrink the regression coefficients. The Ridge objective function is given by:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[(y - \mathbf{X}\boldsymbol{\beta})^T(y - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}\right] \tag{3.12}$$

where $\lambda > 0$ is the regularization parameter. By adding $\lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$, Ridge Regression penalizes the magnitude of the coefficients, making them smaller and reducing the variance of the model. The Ridge solution can be expressed as:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T y \tag{3.13}$$

3.5.1 Choosing the Regularization Parameter

The regularization parameter λ controls the trade-off between fitting the data well and keeping the model coefficients small. Cross-validation is often used to select an optimal value for λ.

3.6 Elastic Net Regression

Elastic Net Regression combines both Ridge and Lasso regularization to encourage both small coefficients and feature selection. The Elastic Net objective function is given by:

$$\hat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left[(y - \mathbf{X}\boldsymbol{\beta})^T (y - \mathbf{X}\boldsymbol{\beta}) + \lambda_1 \boldsymbol{\beta}^T \boldsymbol{\beta} + \lambda_2 \sum_{j=1}^{p} |\beta_j| \right] \tag{3.14}$$

where λ_1 and λ_2 are the Ridge and Lasso regularization parameters, respectively. This combination allows Elastic Net to balance the shrinkage effect from Ridge Regression and the variable selection property from Lasso.

3.6.1 Properties of Elastic Net

Elastic Net is particularly useful when dealing with datasets that have:

- Highly correlated predictors: Elastic Net tends to select groups of correlated predictors, unlike Lasso, which may select only one.
- A need for both regularization and feature selection: Elastic Net provides a compromise between Ridge and Lasso by incorporating both L_1 and L_2 penalties.

Chapter 4

PROPOSED DEEP LEARNING METHODS for REGRESSION PROBLEMS

4.1 Convolutional Neural Networks and ResNet Architectures

Convolutional Neural Networks (CNNs) are a type of deep neural network architecture designed specifically for tasks involving grid-like data, such as images. They have been extremely successful in computer vision tasks, enabling state-of-the-art performance on a wide range of applications including image classification, object detection, and segmentation. This section provides a detailed mathematical explanation of CNNs and explores how they evolve into more advanced architectures such as ResNet13 and ResNet18.

4.1.1 Convolutional Neural Networks

A Convolutional Neural Network consists of several layers, each designed to extract increasingly abstract features from the input data. The key components of a CNN are convolutional layers, pooling layers, and fully connected layers. In this section, we will focus on the mathematical formulation of these components.

Convolutional Layer

The convolutional layer is the core building block of a CNN. It applies convolutional filters to the input to extract features. Mathematically, let $\mathbf{X} \in \mathbb{R}^{H \times W \times C}$ represent the input tensor, where H, W, and C denote the height, width, and number of channels, respectively. A convolutional filter $\mathbf{K} \in \mathbb{R}^{k \times k \times C}$ of size $k \times k$ is applied to $\mathbf{X}$ to produce an output feature map $\mathbf{Y} \in \mathbb{R}^{H' \times W'}$.

The output feature map is computed as follows:

$$\mathbf{Y}_{ij} = \sum_{c=1}^{C} \sum_{u=1}^{k} \sum_{v=1}^{k} \mathbf{K}_{uvc} \cdot \mathbf{X}_{(i+u)(j+v)c} \tag{4.1}$$

where i and j denote the spatial locations of the output feature map.

Activation Function

After convolution, an activation function is typically applied element-wise to introduce non-linearity. The most commonly used activation function in CNNs is the ReLU (Rectified Linear Unit) function, defined as:

$$\text{ReLU}(z) = \max(0, z) \tag{4.2}$$

This helps to avoid the vanishing gradient problem and makes the network learn faster.

Pooling Layer

Pooling layers are used to reduce the spatial dimensions of the feature maps, thereby reducing the number of parameters and computation in the network. The most common pooling operation is max pooling, which takes the maximum value within a sliding window of size $p \times p$.
Mathematically, given an input feature map $\mathbf{Y}$, the output after max pooling is given by:

$$\mathbf{Z}_{ij} = \max_{u,v \in p \times p} \mathbf{Y}_{(i+u)(j+v)} \tag{4.3}$$

Fully Connected Layer

The fully connected layer is used at the end of the network to produce the final output, typically for classification. The output of the final convolutional or pooling layer is flattened into a vector $\mathbf{h}$, and a weight matrix $\mathbf{W}$ is used to compute the final output $\mathbf{o}$:

$$\mathbf{o} = \mathbf{W}\mathbf{h} + \mathbf{b} \tag{4.4}$$

where $\mathbf{b}$ is the bias term.

4.1.2 Residual Networks (ResNet)

Residual Networks (ResNet) were introduced to address the problem of vanishing gradients that occurs when training very deep networks. The key idea of ResNet is the introduction of shortcut connections that skip one or more layers. This allows the network to learn residual functions instead of directly learning the desired underlying mapping.

Residual Block

A residual block is the fundamental building block of ResNet. It consists of two convolutional layers, followed by batch normalization and ReLU activation. The key difference from traditional CNNs is the addition of a shortcut (or skip) connection that bypasses these layers. Mathematically, let $\mathbf{X}$ be the input to a residual block. The output $\mathbf{Y}$ is given by:

$$\mathbf{Y} = \mathbf{X} + F(\mathbf{X}, \mathbf{W}) \tag{4.5}$$

where $F(\mathbf{X}, \mathbf{W})$ represents the transformation applied by the two convolutional layers, and $\mathbf{W}$ denotes the weights of these layers. The addition operation helps to propagate gradients through the network more effectively, making it easier to train deeper architectures.

ResNet Architectures

ResNet13 and ResNet18 are two popular versions of the ResNet architecture with different numbers of layers. The ResNet18 architecture, for example, consists of 17 convolutional layers and 1 fully connected layer, with multiple residual blocks.
The forward propagation in ResNet can be described mathematically as follows. Let $\mathbf{X}_0$ be the input to the network, and let $\mathbf{X}_l$ represent the output of the l-th layer. For a residual block, the output is:

$$\mathbf{X}_{l+1} = \mathbf{X}_l + F(\mathbf{X}_l, \mathbf{W}_l) \tag{4.6}$$

The final output of the network is obtained by passing the output of the last residual block through a global average pooling layer followed by a fully connected layer for classification.

Mathematical Analysis of Residual Learning

The key advantage of residual learning is that it allows the network to learn the identity function more easily. Suppose the optimal mapping for a particular layer is close to the identity function, i.e., $\mathbf{Y} \approx \mathbf{X}$. In traditional networks, the layers would need to learn this identity mapping directly. In ResNet, the network instead learns a residual mapping $F(\mathbf{X}) = \mathbf{Y} - \mathbf{X}$, which is easier to optimize.
The objective function for training a ResNet can be expressed as:

$$\min_{\mathbf{W}} \frac{1}{N} \sum_{i=1}^{N} L(y_i, f(\mathbf{X}_i; \mathbf{W})) \tag{4.7}$$

where $L(y, \hat{y})$ is the loss function, y_i represents the true label, and $f(\mathbf{X}_i; \mathbf{W})$ is the prediction produced by the ResNet with parameters $\mathbf{W}$.

Chapter 5

PROPOSED METHODS for CLUSTERING PROBLEMS

5.1 K-means Clustering

K-means clustering is an unsupervised learning algorithm used to partition a dataset into a set of distinct, non-overlapping groups called clusters. It minimizes the sum of squared distances between the data points and their respective cluster centroids.

5.1.1 Algorithm Steps

a. Initialize K cluster centroids randomly.

b. **Expectation Step (E-Step)**: Assign each data point to the cluster whose centroid is closest. This step minimizes the squared Euclidean distance between data points and centroids.

c. **Maximization Step (M-Step)**: Recompute the centroids of clusters as the mean of the assigned data points:

$$\mu_k = \frac{1}{N_k} \sum_{i:r_{ik}=1} x_i \tag{5.1}$$

where N_k is the number of data points assigned to cluster k.

d. Repeat the E-step and M-step until convergence, meaning that there are no further changes in the assignment of data points to clusters or a maximum number of iterations is reached.

The objective function minimized by K-means is:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2 \tag{5.2}$$

where r_{nk} is a binary indicator that specifies whether data point x_n belongs to cluster k, and μ_k represents the centroid of cluster k.

5.1.2 Challenges

- K-means may converge to a local rather than a global minimum of the objective function.
- The solution is highly sensitive to the initial choice of cluster centroids.

5.1.3 Elbow Method

The Elbow method is used to determine the optimal number of clusters, K. It involves plotting the "inertia," or within-cluster sum of squares (WCSS), against different values of K:

$$WCSS = \sum_{k=1}^{K} \sum_{i:r_{ik}=1} \|x_i - \mu_k\|^2 \tag{5.3}$$

The point at which the decrease in inertia slows down considerably is called the "elbow point." This elbow is considered the optimal K, as increasing the number of clusters beyond this point yields only a marginal decrease in inertia.

5.1.4 Inertia

Inertia represents the sum of squared distances of data points to their closest cluster centroid. It is used to evaluate the quality of clustering, where lower inertia indicates more compact clusters. However, it is sensitive to the number of clusters, which is why we use methods like the Elbow method to choose an optimal value for K.

5.1.5 Silhouette Score

The Silhouette score measures how similar a data point is to its own cluster compared to other clusters. The score is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{5.4}$$

where $a(i)$ is the average distance between the i-th point and all other points in the same cluster, and $b(i)$ is the minimum average distance from the i-th point to points in a different cluster. The score ranges from -1 to 1:

- A value close to 1 indicates that the point is well-clustered.
- A value close to 0 means the point lies between two clusters.
- A value close to -1 indicates that the point may have been assigned to the wrong cluster.

5.2 Expectation-Maximization (EM) Algorithm

The EM algorithm is a general iterative method to find maximum likelihood estimates of parameters in models with latent variables. It maximizes the likelihood function $p(X;\theta)$ when the data X contains missing or hidden variables.

5.2.1 Algorithm Steps

a **Initialization**: Start with initial parameter estimates θ_{old}.

b **E-Step**: Compute the expected value of the complete-data log likelihood, given observed data and current parameter estimates:

$$Q(\theta|\theta_{\text{old}}) = E_{Z|X,\theta_{\text{old}}}[\log p(X, Z; \theta)] \tag{5.5}$$

c **M-Step**: Maximize the expected complete-data log likelihood with respect to the parameters to obtain updated parameters θ_{new}:

$$\theta_{\text{new}} = \arg\max_{\theta} Q(\theta|\theta_{\text{old}}) \tag{5.6}$$

d Repeat the E-step and M-step until convergence, i.e., until the parameters stabilize.

The EM algorithm iteratively increases the likelihood function and converges to a local maximum.

5.2.2 Gaussian Mixture Model (GMM)

The EM algorithm is commonly used for GMMs, where the data is assumed to be generated from a mixture of several Gaussian distributions. The likelihood function for GMM is given by:

$$p(X|\theta) = \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k) \tag{5.7}$$

where π_k are the mixing coefficients, μ_k are the means, and Σ_k are the covariance matrices of the Gaussian components.

The EM steps for GMM involve calculating the responsibility each Gaussian component takes for each data point (E-step) and then updating the parameters of each component based on these responsibilities (M-step).

5.3 DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN is a density-based clustering algorithm that identifies dense regions of data points separated by regions of low density.

5.3.1 Key Parameters

- ϵ: Radius of the neighborhood around a point.
- **MinPts**: Minimum number of points required within the ϵ-neighborhood for a point to be considered a core point.

5.3.2 Algorithm Steps

a. Randomly select an unvisited point.

b. If the point has at least **MinPts** within its ϵ-neighborhood, it becomes a core point and forms the nucleus of a cluster.

c. All points within ϵ distance from a core point are added to the cluster.

d. If a point does not have sufficient neighbors, it is labeled as noise (though it may be later added to a cluster if it falls within the neighborhood of a core point).

e. Repeat until all points are either assigned to clusters or labeled as noise.

5.3.3 Advantages

- It can find arbitrarily shaped clusters.
- It can automatically determine the number of clusters.
- It is robust to outliers as they are labeled as noise.

5.4 Dimensionality Reduction

Dimensionality reduction is a fundamental technique in the field of machine learning and data analysis aimed at mitigating the "curse of dimensionality." As datasets grow in size and complexity, the number of features or variables often increases, posing challenges for computational efficiency and model performance.

Dimensionality reduction methods seek to transform high-dimensional data into a lower-dimensional representation while preserving essential information. Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and Uniform Manifold Approximation and Projection (UMAP) are among the popular techniques used for this purpose. By reducing the number of dimensions, these methods not only enhance computational efficiency but also aid in visualization and interpretation, ultimately improving the performance of machine learning models and facilitating a more meaningful analysis of complex datasets.

Among these techniques, PCA is most used for unsupervised data compression and LDA is a method used for supervised dimensionality reduction - which I'm going to discuss later in this post.

Principal component analysis (PCA) aims to transform high-dimensional data into a lower-dimensional space by identifying the principal components, which are orthogonal vectors capturing the maximum variance in the original dataset. These principal components are linear combinations of the original features and are arranged in descending order of their variance. By selecting a subset of the top principal components, researchers can reduce the dimensionality of the data while retaining the most significant information.

5.4.1 Principal Component Analysis

Proposition 5.1

a PCA learns a representation that has lower dimensionality than the original input. $\mathbf{X} \approx \mathbf{ZW}$, where $\mathbf{Z} \in \mathbb{R}^{N \times p}$,$p < d$.
Optimization Problem:

$$\widehat{\mathbf{Z}}, \widehat{\mathbf{W}} = \arg\max_{\mathbf{Z},\mathbf{W}} \|\mathbf{X} - \mathbf{ZW}\|_{\mathcal{F}}^2$$

b It also learns a representation whose elements have no linear correlation with each other

Suppose we have a dataset $\mathcal{X} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(N)}\}$ with $\mathbf{x}^{(i)} \in \mathbb{R}^d$. If d is sufficiently large, we aim to compress $\mathcal{X}$ into data points with a dimension p such that $p << d$.
Let $\mathcal{B} = \{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_d\}$ be an orthonormal basis of $\mathbb{R}^d$ (the right singular vectors in the decomposition $\mathbf{X} = \mathbf{U\Sigma V}^\top$). For each $\mathbf{x}^{(i)}$, there exists a unique linear representation as

follows:

$$\mathbf{x}^{(i)} = \sum_{j=1}^{d} z_j^{(i)} \mathbf{v}_j \tag{5.8}$$

We seek an approximation:

$$\mathbf{x}^{(i)} \approx \sum_{j=1}^{p} z_j^{(i)} \mathbf{v}_j \tag{5.9}$$

Or

$$\mathbf{X} \approx \mathbf{Z}\mathbf{V}^\top \tag{5.10}$$

where $\mathbf{Z} \in \mathbb{R}^{N \times p}$ and $\mathbf{V} \in \mathbb{R}^{d \times p}$.

Note that each column of $\mathbf{V}$ is a $\mathbf{v}_k$, hence $\mathbf{V}^\top \mathbf{V} = \mathbf{I}_d$.

We minimize the approximation error:

$$(\widehat{\mathbf{V}}, \widehat{\mathbf{A}}) = \arg\min_{\mathbf{V},\mathbf{Z}} \left\{ \left\| \mathbf{Z}\mathbf{V}^\top - \mathbf{X} \right\|_{\mathcal{F}}^2 \right\} \tag{5.11}$$

We have: $\nabla_{\mathbf{Z}} \left\| \mathbf{Z}\mathbf{V}^\top - \mathbf{X} \right\|_{\mathcal{F}}^2 = 2\mathbf{V}^\top(\mathbf{V}\mathbf{Z}^\top - \mathbf{X}^\top) = \mathbf{0}_{p \times N} \Leftrightarrow \mathbf{Z} = \mathbf{X}\mathbf{V}$

Substituting into (5.11), we obtain the optimization problem for a matrix variable $\mathbf{V}$ as follows:

$$\begin{aligned}
\widehat{\mathbf{V}} &= \arg\min_{\mathbf{V}} \left\{ \left\| \mathbf{X}\mathbf{V}\mathbf{V}^\top - \mathbf{X} \right\|_{\mathcal{F}}^2 \right\} \\
&= \arg\min_{\mathbf{V}} \left\{ \mathrm{Tr}((\mathbf{V}\mathbf{V}^\top\mathbf{X}^\top - \mathbf{X}^\top)(\mathbf{X}\mathbf{V}\mathbf{V}^\top - \mathbf{X})) \right\} \\
&= \arg\min_{\mathbf{V}} \left\{ \mathrm{Tr}\left[\mathbf{X}^\top\mathbf{X} - \mathbf{X}^\top\mathbf{X}\mathbf{V}\mathbf{V}^\top - \mathbf{V}\mathbf{V}^\top\mathbf{X}^\top\mathbf{X} + \mathbf{V}\mathbf{V}^\top\mathbf{X}^\top\mathbf{X}\mathbf{V}\mathbf{V}^\top \right] \right\} \\
&= \arg\max_{\mathbf{V}} \left\{ \mathrm{Tr}(\mathbf{V}^\top\mathbf{X}^\top\mathbf{X}\mathbf{V}) \right\} \\
&= \arg\max_{\mathbf{V}} \{ \mathrm{Tr}(\mathbf{V}^\top \mathbb{E}\left[\mathbf{X}^\top\mathbf{X} \right] \mathbf{V}) \} \\
&= \arg\max_{\mathbf{V}} \left\{ \mathrm{Tr}\left[\mathbf{\Sigma}^2 \right] \right\}
\end{aligned}$$

Note that the final equality occurs because we can always assume $\mathbb{E}(\mathbf{X}) = 0$.

Since the covariance matrix $\mathrm{Var}[\mathbf{X}]$ is symmetric and positive semi-definite, it has a unique orthogonal diagonalization:

$$\mathrm{Var}[\mathbf{X}] = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$$

, where $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, ..., \lambda_d\}$ with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_d \geq 0$.

The solution to the main optimization problem is the top p columns of $\mathbf{V}$ (corresponding to the p largest eigenvalues of $\mathrm{Var}[\mathbf{X}]$).

Singular Value Decomposition

Proposition 5.2

KHANG Suppose $\{\mathbf{v}_1, .., \mathbf{v}_n\}$ is an orthonormal basis $\mathbb{R}^n$ consisting eigenvectors of $\mathbf{A}^\top\mathbf{A}$, arranged so that the corresponding eigenvalues of $\mathbf{A}^\top\mathbf{A}$ satisfy $\lambda_1 \geqslant ... \geqslant \lambda_n \geqslant 0$, and suppose $\mathbf{A}$ has r nonzero singular values.

Then $\{\mathbf{A}\mathbf{v}_1, ...\mathbf{A}\mathbf{v}_r\}$ is an orthogonal basis for Col$\{\mathbf{A}\}$, and Rank$(\mathbf{A}) = \dim(\text{Col}(A)) = r$.

Proof. Firstly, we have to demonstrate that $\{\mathbf{A}\mathbf{v}_1, ...\mathbf{A}\mathbf{v}_r\}$ is orthogonal. Indeed, due to $\mathbf{A}\mathbf{v}_i \cdot \mathbf{A}\mathbf{v}_j = \mathbf{v}_i^\top\mathbf{A}^\top\mathbf{A}\mathbf{v}_j = \lambda_j\mathbf{v}_i \cdot \mathbf{v}_j = 0, \forall 1 \leq i \neq j \leq r$.

Finally, it's easy to realize that Span $\{\mathbf{A}\mathbf{v}_1, ...\mathbf{A}\mathbf{v}_r\} = \text{Col}(\mathbf{A})$. ■

Proposition 5.3

Singular Value Decomposition[a]

$$\mathbf{A} = \mathbf{U}_{m\times m}\mathbf{\Sigma}_{m\times n}\mathbf{V}^\top_{n\times n}$$

[a] $\mathbf{\Sigma} := \begin{bmatrix} \text{diag}\{\sigma_1, .., \sigma_r\} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$

Proof. For $1 \leq i \leq r$, we set: $\mathbf{u}_i = \frac{1}{\sigma_i}\mathbf{A}\mathbf{v}_i = \frac{1}{\|\mathbf{A}\mathbf{v}_i\|}$. Then $\{\mathbf{u}_1, .., \mathbf{u}_r\}$ is a orthonormal basis for Col$(\mathbf{A})$. We can add $\mathbf{u}_{r+1}, .., \mathbf{u}_m$ such that $\{\mathbf{u}_1, .., \mathbf{u}_m\}$ is orthonormal basis for $\mathbb{R}^m$.

Build $\mathbf{U} := [\mathbf{u}_1|\mathbf{u}_2|...|\mathbf{u}_m]$, $\mathbf{V} := [\mathbf{v}_1|\mathbf{u}_2|...|\mathbf{u}_n] \Rightarrow \mathbf{A}\mathbf{V} = \mathbf{U}\mathbf{\Sigma}$. ■

5.4.2 Linear Discriminant Analysis

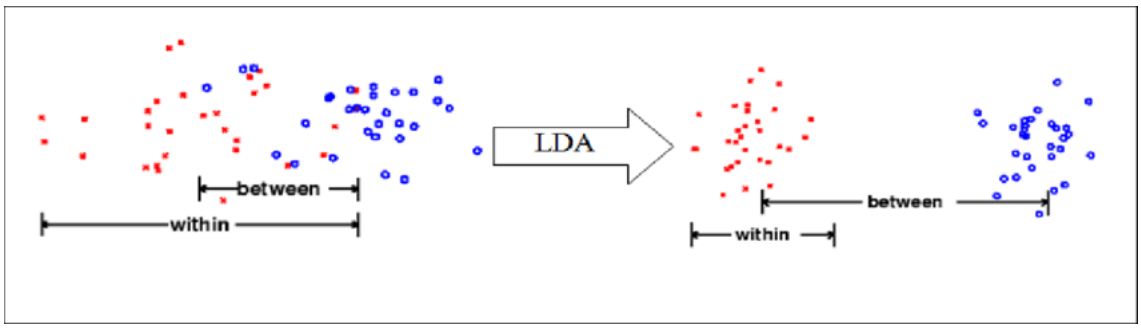

Figure 5.1: Enter Caption

$\mathbf{X} \in \mathbb{R}^{N\times d}, \mathbf{A} \in \mathbb{R}^{d\times k}$

Linear transformation: $\mathbf{Z} = \mathbf{X}\mathbf{A}$. In other words, $\mathbf{z}^{(i)} = \mathbf{A}^\top\mathbf{x}^{(i)}$.

Hence, $\boldsymbol{\mu}_\mathbf{Z} = \mathbf{A}^\top\mathbb{E}[\mathbf{X}] = \mathbf{A}^\top\boldsymbol{\mu}_\mathbf{X}$ and Var$[\mathbf{z}] = \mathbf{A}^\top$ Var$[\mathbf{X}]\mathbf{A}$

$\boldsymbol{\mu}^{c=l}(\mathbf{z}) = \mathbf{A}^\top \boldsymbol{\mu}^{c=l}(\mathbf{x})$; $\text{Var}^{c=l}[\mathbf{z}] = \mathbf{A}^\top \text{Var}^{c=l}[\mathbf{x}]\mathbf{A}$

Broadcasting for expectation: $\mathbf{M}_\mathbf{z} = \mathbf{1}^{N_j \times 1}\boldsymbol{\mu}[\mathbf{z}] = \mathbf{1}^{N_j \times 1}\boldsymbol{\mu}^\top[\mathbf{x}]\mathbf{A} = \mathbf{M}_\mathbf{x}\mathbf{A}$

Broadcast for the mean of each class: $\mathbf{M}_\mathbf{z}^{c=j} = \mathbf{1}^{N_j \times 1}\boldsymbol{\mu}_\mathbf{z}^{c=j,\top} = \mathbf{1}^{N_j \times 1}\boldsymbol{\mu}_\mathbf{x}^{c=j,\top}\mathbf{A} = \mathbf{M}_\mathbf{x}^{c=j}\mathbf{A}$.

Between-Class Variance

$$\begin{aligned}\sum_{j=1}^{C}\left\|\mathbf{M}_\mathbf{z}^{c=j} - \mathbf{M}_j\right\|_\mathcal{F}^2 &= \sum_{j=1}^{C}\left\|(\mathbf{M}_\mathbf{x}^{c=j} - \mathbf{M}_\mathbf{x})\mathbf{A}\right\|_\mathcal{F}^2 \\ &= \sum_{j=1}^{C}\text{Tr}\left\{\mathbf{A}^\top(\mathbf{M}_\mathbf{x}^{c=j,\top} - \mathbf{M}_\mathbf{x}^\top)(\mathbf{M}_\mathbf{x}^{c=j} - \mathbf{M}_\mathbf{x})\mathbf{A}\right\} \\ &= \text{Tr}\left\{\mathbf{A}^\top\mathbf{S}_B\mathbf{A}\right\}\end{aligned}$$

, where $\mathbf{S}_B \overset{\text{def}}{=} \sum_{c=j}^{C}\text{Tr}\left\{(\mathbf{M}_\mathbf{x}^{c=j,\top} - \mathbf{M}_{\mathbf{x}^\top})(\mathbf{M}_\mathbf{x}^{c=j} - \mathbf{M}_\mathbf{x})\right\} = \sum_{c=j}^{C} N_j(\boldsymbol{\mu}_\mathbf{x}^{c=j} - \boldsymbol{\mu}_\mathbf{x})(\boldsymbol{\mu}_\mathbf{x}^{c=j} - \boldsymbol{\mu}_\mathbf{x})^\top$

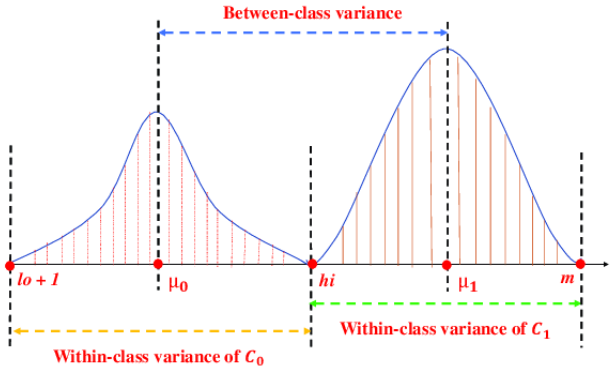

Figure 5.2: Enter Caption

Within-Class Variance

$$\begin{aligned}\sum_{j=1}^{C}\left\|\mathbf{Z}^{c=j} - \mathbf{M}_\mathbf{z}^{c=j}\right\|_\mathcal{F}^2 &= \sum_{j=1}^{C}\left\|(\mathbf{X}^{c=j} - \mathbf{M}_\mathbf{x}^{c=j})\mathbf{A}\right\|_\mathcal{F}^2 \\ &= \sum_{j=1}^{C}\text{Tr}\left\{\mathbf{A}^\top(\mathbf{X}^{c=j,\top} - \mathbf{M}_\mathbf{x}^{c=j,\top})(\mathbf{X}^{c=j} - \mathbf{M}_\mathbf{x}^{c=j})\mathbf{A}\right\} \\ &= \text{Tr}\left\{\mathbf{A}^\top\mathbf{S}_w\mathbf{A}\right\}\end{aligned}$$

, where $\mathbf{S}_w \overset{\text{def}}{=} \sum_{j=1}^{C}(\mathbf{X}^{c=j,\top} - \mathbf{M}_\mathbf{x}^{c=j,\top})(\mathbf{X}^{c=j} - \mathbf{M}_\mathbf{x}^{c=j}) = \sum_{c=j}^{C}\sum_{i=1}^{N}(\mathbf{x}_i^{c=j} - \boldsymbol{\mu}_\mathbf{x}^{c=j})(\mathbf{x}_i^{c=j} - \boldsymbol{\mu}_\mathbf{x}^{c=j})^\top$

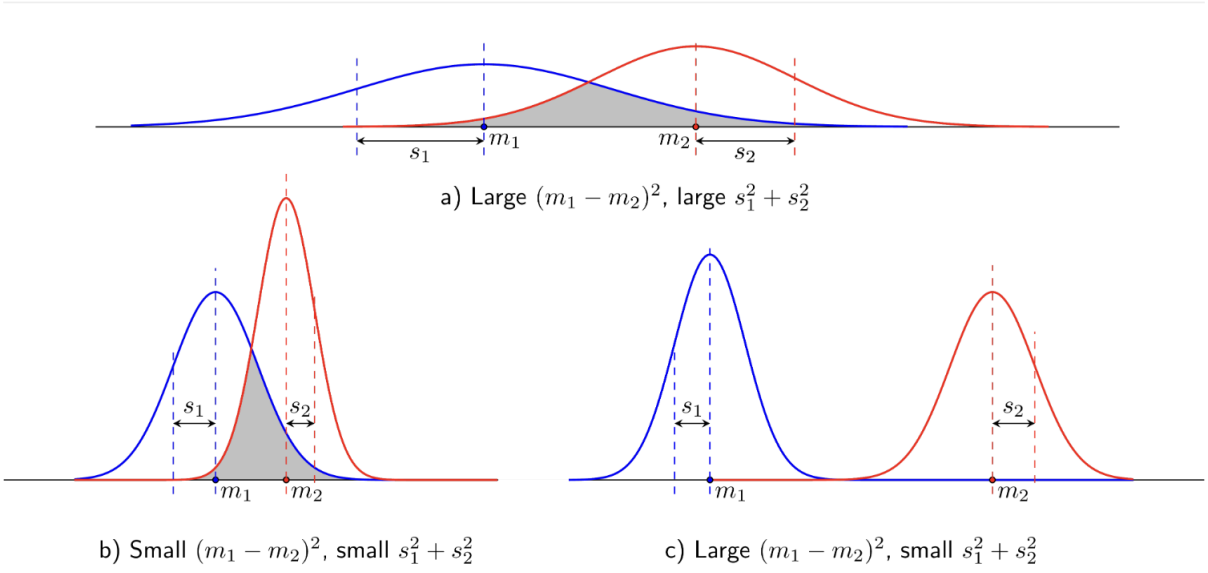

Figure 5.3: Enter Caption

Loss Function: $\ell(\mathbf{A}) = \dfrac{\text{Tr}\left\{\mathbf{A}^\top \mathbf{S}_B \mathbf{A}\right\}}{\text{Tr}\left\{\mathbf{A}^\top \mathbf{S}_W \mathbf{A}\right\}}$.

$$\Rightarrow \nabla_{\mathbf{A}} \ell(\mathbf{A}) = 2\frac{\mathbf{S}_B\mathbf{A}\,\text{Tr}\left[\mathbf{A}^\top \mathbf{S}_W \mathbf{A}\right] - \text{Tr}\left[\mathbf{A}^\top \mathbf{S}_B \mathbf{A}\right]\mathbf{S}_W\mathbf{A}}{\left\{\text{Tr}\left[\mathbf{A}^\top \mathbf{S}_W \mathbf{A}\right]\right\}^2} = 0$$

$$\Rightarrow \mathbf{S}_B\mathbf{A}\,\text{Tr}\left\{\mathbf{A}^\top \mathbf{S}_W \mathbf{A}\right\} = \text{Tr}\left\{\mathbf{A}^\top \mathbf{S}_B \mathbf{A}\right\}\mathbf{S}_W\mathbf{A} \Rightarrow \mathbf{S}_B\mathbf{A} = \ell(\mathbf{A})\mathbf{S}_w\mathbf{A}$$

$$\Rightarrow \mathbf{S}_W^\dagger \mathbf{S}_B\mathbf{A} = \ell(\mathbf{A})\mathbf{A}$$

An Amazing Result: $k < C$

Proof. Indeed, we always find a matrix $\mathbf{R} \in \mathbb{R}^{d\times C}$ so that: $\mathbf{S}_B = \mathbf{R}\mathbf{R}^\top$.

Then: $\mathbf{r}_i = \sqrt{N_i}(\mathbf{m}_i - \mathbf{m}), \forall i = \overline{1, C}$

We have:

$$-\frac{\mathbf{r}_i}{\sqrt{N_i}} = \mathbf{m} - \mathbf{m}_i = \frac{N}{N_i}\mathbf{m} - \mathbf{m}_i - \frac{N - N_i}{N_i}\mathbf{m}$$

$$= \frac{N}{N_i}\mathbf{m} - \frac{N}{N_i}\left(\mathbf{m} - \sum_{1\le j\le C; j\neq i}\frac{N_j}{N}\mathbf{m}_j\right) - \frac{\sum\limits_{j\neq i} N_j}{N_i}\mathbf{m}$$

$$= \sum_{1\le j\le C; j\neq i}\frac{N_j}{N_i}\mathbf{m}_j - \sum_{1\le j\le C; j\neq i}\frac{N_j}{N_i}\mathbf{m}$$

$$= \sum_{1\le j\le C; j\neq i}\frac{N_j}{N_i}(\mathbf{m}_j - \mathbf{m}) = \sum_{1\le j\le C; j\neq i}\frac{\sqrt{N_j}}{N_i}\mathbf{r}_j$$

$\Rightarrow \mathbf{r}_i$ is a linear combination of remaining column vectors of $\mathbf{R}$

$\Rightarrow \text{rank}(\mathbf{R}) < C$.

$\Rightarrow k \le \text{rank}(\mathbf{S}_W^\dagger \mathbf{S}_B) \le \text{rank}(\mathbf{S}_B) \le \text{rank}(\mathbf{R}) < C$ ■

Chapter 6

PROPOSED METHODS for CLASSIFICATION PROBLEMS

6.1 Logistic Regression

Logistic Regression is a classification model that predicts the probability of a binary outcome using a logistic function. Given an input vector $\mathbf{x}$, the model computes the probability $p(y = 1|\mathbf{x})$ as follows:

$$p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w}^T\mathbf{x}}} \tag{6.1}$$

where $\sigma(z)$ is the sigmoid function and $\mathbf{w}$ is the vector of weights. The probability $p(y = 0|\mathbf{x})$ is $1 - p(y = 1|\mathbf{x})$.

6.1.1 Odds and Log-Odds

- **Odds**: The odds represent the ratio of the probability of an event occurring to the probability of it not occurring. If p is the probability of an event, then the odds are defined as:

$$\text{Odds} = \frac{p}{1 - p} \tag{6.2}$$

- **Log-Odds**: Also known as the logit function, log-odds is the natural logarithm of the odds:

$$\text{Log-Odds} = \text{logit}(p) = \log\left(\frac{p}{1 - p}\right) = \mathbf{w}^T\mathbf{x} \tag{6.3}$$

Logistic Regression linearizes the relationship between the input features and the log-odds of the outcome.

6.1.2 Likelihood Function and Maximum Likelihood

The goal of Logistic Regression is to find the weight vector $\mathbf{w}$ that maximizes the likelihood of the observed data. The likelihood function for a dataset $(\mathbf{x}_i, t_i)$ with $t_i \in \{0, 1\}$ is given by:

$$L(\mathbf{w}) = \prod_{i=1}^{N} p(y_i|\mathbf{x}_i;\mathbf{w})^{t_i}(1-p(y_i|\mathbf{x}_i;\mathbf{w}))^{1-t_i} \tag{6.4}$$

The negative log-likelihood function, or the error function, is:

$$E(\mathbf{w}) = -\sum_{i=1}^{N}[t_i \log(p(y_i|\mathbf{x}_i;\mathbf{w})) + (1-t_i)\log(1-p(y_i|\mathbf{x}_i;\mathbf{w}))] \tag{6.5}$$

The gradient of the error function is:

$$\nabla E(\mathbf{w}) = \sum_{i=1}^{N}(p(y_i|\mathbf{x}_i;\mathbf{w}) - t_i)\mathbf{x}_i \tag{6.6}$$

6.2 AIC and BIC

Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) are used to evaluate the quality of statistical models.

- AIC: AIC is used to balance model accuracy and complexity:

$$\text{AIC} = 2k - 2\log(\hat{L}) \tag{6.7}$$

where k is the number of parameters in the model and $\hat{L}$ is the value of the likelihood function at its maximum.

- BIC: Similar to AIC but includes a penalty based on the number of observations N:

$$\text{BIC} = k\log(N) - 2\log(\hat{L}) \tag{6.8}$$

6.3 Multi-Class Logistic Regression and Softmax Regression

To extend Logistic Regression to multi-class classification, we use **Softmax Regression**. Assume there are K classes, the probability of class k is given by:

$$p(C_k|\mathbf{x}) = \frac{e^{\mathbf{w}_k^T\mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^T\mathbf{x}}} \tag{6.9}$$

where $\mathbf{w}_k$ is the weight vector for class k. This is known as the **Softmax function**, which generalizes the sigmoid function to multiple classes.

6.4 Sigmoid and Softmax Functions

- Sigmoid Function: The sigmoid function is used for binary classification to map any real-valued number to the range $(0, 1)$:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{6.10}$$

- Softmax Function**: The softmax function is used for multi-class classification to map a vector of values to a probability distribution:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{6.11}$$

6.5 Parameter Interpretation

- The weight vector $\mathbf{w}$ in Logistic Regression represents the log-odds change for a one-unit change in the corresponding feature, holding all other features constant. - In the case of multi-class logistic regression, $\mathbf{w}_k$ represents the contribution of each feature to the log-odds of class k compared to the baseline.

Logistic Regression provides a simple yet effective way to model binary and multi-class classification problems by transforming linear relationships to probabilities through the use of sigmoid or softmax functions.

6.6 Decision Tree

A Decision Tree is a non-parametric model used for classification and regression. The tree is built by recursively splitting the data based on feature values to reduce impurity in the target variable.

6.6.1 Shannon Entropy

Shannon Entropy is a measure of the uncertainty in a set of outcomes. For a discrete random variable X with possible values $x_1, x_2, \ldots, x_n$ and corresponding probabilities $p_1, p_2, \ldots, p_n$, the entropy $H(X)$ is defined as:

$$H(X) = -\sum_{i=1}^{n} p_i \log(p_i) \tag{6.12}$$

Entropy is used in decision trees to measure the impurity of a node.

6.6.2 Cross-Entropy

Cross-entropy measures the difference between two probability distributions. If $p(x)$ is the true distribution and $q(x)$ is the estimated distribution, the cross-entropy $H(p, q)$ is given by:

$$H(p, q) = -\sum_{x} p(x) \log(q(x)) \tag{6.13}$$

Cross-entropy is often used as a loss function in classification tasks.

6.6.3 Information Gain

Information Gain is a crucial criterion used to determine the best feature to split the dataset at each step of building the decision tree. It is defined as the reduction in entropy after a dataset is split on an attribute. For a dataset D, the information gain $IG(D, A)$ for attribute A is given by:

$$IG(D, A) = H(D) - \sum_{v \in A} \frac{|D_v|}{|D|} H(D_v) \tag{6.14}$$

where $H(D)$ is the entropy of the dataset before the split, and $H(D_v)$ is the entropy of the subset D_v after splitting on attribute A.

6.6.4 Gain Ratio

Gain Ratio is a metric used to address the inherent bias of Information Gain, which tends to favor attributes with many distinct values. It is defined as:

$$GR(D, A) = \frac{IG(D, A)}{H_A(D)} \tag{6.15}$$

where $H_A(D)$ is the intrinsic information of attribute A.

6.6.5 GINI Impurity

GINI impurity is a metric used to quantify the impurity of a node, and it serves as a criterion for determining the best split. For a dataset D with k classes, the GINI impurity $G(D)$ is given by:

$$G(D) = 1 - \sum_{i=1}^{k} p_i^2 \tag{6.16}$$

where p_i is the proportion of samples belonging to class i.

6.6.6 CART (Classification and Regression Trees)

The CART (Classification and Regression Trees) algorithm is a widely-used approach for constructing decision trees. It uses GINI impurity to determine splits for classification tasks and minimizes the mean squared error (MSE) for regression tasks. The CART model is a binary tree, meaning each internal node has two children.

6.6.7 ID3 Algorithm

The ID3 (Iterative Dichotomiser 3) algorithm is a foundational approach to constructing decision trees. It employs a top-down, greedy search strategy to determine the attribute that provides the highest reduction in entropy at each node. It uses Information Gain as the splitting criterion.

6.6.8 C4.5 Algorithm

The C4.5 algorithm is an advancement over ID3, addressing several of its limitations. It handles both continuous and discrete attributes effectively, manages missing values during the training process, and incorporates a pruning mechanism to avoid overfitting after the initial tree construction. It uses Gain Ratio as the splitting criterion.

6.7 Bootstrap Aggregating (Bagging) and Random Forest

Bootstrap Aggregating, or Bagging, is an ensemble method that aims to improve the stability and accuracy of machine learning algorithms by reducing variance and helping to avoid overfitting. Bagging achieves this by generating multiple versions of a training dataset using bootstrapping and then combining the predictions from multiple base models (typically decision trees).

6.7.1 Bootstrapping

Bootstrapping is a statistical resampling technique that involves repeatedly sampling from a dataset with replacement. Let $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ represent the original dataset, where N is the number of samples. To create a bootstrapped dataset $\mathbf{D}^b$, we randomly sample N points from $\mathbf{D}$ with replacement. This means that some samples may appear multiple times in $\mathbf{D}^b$, while others may not appear at all.
Mathematically, the probability that a specific sample from the original dataset is not included in a bootstrapped dataset is given by:

$$P(\text{not included}) = \left(1 - \frac{1}{N}\right)^N \tag{6.17}$$

Taking the limit as $N \to \infty$, we obtain:

$$\lim_{N\to\infty} \left(1 - \frac{1}{N}\right)^N = \frac{1}{e} \approx 0.368 \tag{6.18}$$

Thus, approximately 36.8% of the original dataset will not be included in each bootstrapped dataset. Each bootstrapped dataset $\mathbf{D}^b$ is then used to train a base model, such as a decision tree.

6.7.2 Bagging

Bagging involves training B base models on B different bootstrapped datasets. Let $f_b(\mathbf{x})$ denote the prediction of the b-th base model for an input $\mathbf{x}$. The final prediction for a regression problem is obtained by averaging the predictions from all base models:

$$\hat{y} = \frac{1}{B}\sum_{b=1}^{B} f_b(\mathbf{x}) \tag{6.19}$$

For classification tasks, the final prediction is made by majority voting among the B base models.

Bagging helps to reduce the variance of the model, which is particularly beneficial for high-variance models like decision trees. By aggregating multiple models, bagging creates a more robust and stable prediction.

6.7.3 Random Forest

Random Forest is an extension of Bagging that further improves model diversity by introducing additional randomness during the training of each decision tree. In a Random Forest, each tree is trained on a bootstrapped dataset, but instead of considering all features for each split, only a random subset of features is considered.

Mathematically, let F be the total number of features in the dataset. At each split in a decision tree, Random Forest selects m features from F, where $m < F$. The split is then made using the feature that provides the best split according to a chosen criterion, such as GINI impurity or Information Gain.

The prediction for a Random Forest is given by averaging (in regression) or majority voting (in classification) across all trees:

$$\hat{y}_{\text{RF}} = \frac{1}{B}\sum_{b=1}^{B} f_b(\mathbf{x}) \tag{6.20}$$

6.7.4 Mathematical Analysis of Bagging and Random Forest

Bagging reduces the variance of a model by averaging multiple base models, which effectively reduces the effect of noise in the training data. Let $\hat{f}(\mathbf{x})$ be the prediction of a single model and $\text{Var}(\hat{f}(\mathbf{x}))$ be its variance. In bagging, the variance of the ensemble prediction $\hat{f}_{\text{bag}}(\mathbf{x})$ is given by:

$$\text{Var}(\hat{f}_{\text{bag}}(\mathbf{x})) = \frac{1}{B}\text{Var}(\hat{f}(\mathbf{x})) \tag{6.21}$$

Thus, increasing the number of base models B reduces the overall variance, leading to a more stable prediction.

Random Forest further reduces the correlation between base models by using random feature subsets, which decreases the overall variance without significantly increasing bias. The goal is to ensure that the individual trees are diverse, thereby making the ensemble more robust.

6.7.5 Out-of-Bag (OOB) Error

In Bagging and Random Forest, each bootstrapped dataset leaves out approximately 36.8% of the original samples. These left-out samples are called Out-of-Bag (OOB) samples. The OOB samples can be used to estimate the model's performance without requiring a separate validation set. The OOB error is computed by averaging the prediction error for each sample using only the models that did not see that sample during training.

$$\text{OOB Error} = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{y}_{\text{OOB},i}) \tag{6.22}$$

where L is the loss function, y_i is the true label, and $\hat{y}_{\text{OOB},i}$ is the OOB prediction for sample i.

6.8 Support Vector Machine

6.8.1 Hard-Margin Support Vector Machine

Denote the example in the dataset that is closest to the hyperplane by $\mathbf{x}_a$

Margin

$y^{(i)}(\mathbf{w}\cdot\mathbf{x}^{(i)}+b) \geq 1, \forall i = \overline{1,N}$

Definition 1. *Margin is the distance of the separating hyperplane to the closest examples in the dataset (assuming that the dataset is linearly separable).*

Hay $\textit{Margin} := \min_i \left\{\frac{y^{(i)}(\mathbf{w}\cdot\mathbf{x}^{(i)}+b)}{\|\mathbf{w}\|^2}\right\} = \|\mathbf{w}\|^{-2}$

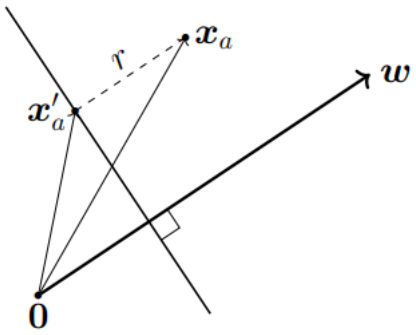

Figure 6.1: Vector addition to express distance to hyperplane: $\mathbf{x}_a = \mathbf{x}'_a + r\frac{\mathbf{w}}{\|\mathbf{w}\|}$

$$\mathbf{x}_a = \mathbf{x}'_a + r\frac{\mathbf{w}}{\|\mathbf{w}\|} \tag{6.23}$$

Refer to 6.23

Require

$$y^{(i)}(\mathbf{w}\cdot\mathbf{x}^{(i)} + b) \geq r, \forall i = \overline{1, N} \tag{6.24}$$

Proposition 6.1

Optimization Problem 1 $\max_{\mathbf{w},b,r} \underbrace{r}_{\text{margin}}$

subject to $\underbrace{y^{(i)}(\mathbf{w}\cdot\mathbf{x}^{(i)} + b) \geq r}_{\text{data fitting}}$, $\underbrace{\|w\| = 1}_{\text{normalization}}$, r>0

Traditional derivation of the Margin $r = \dfrac{1}{\|\mathbf{w}\|}$

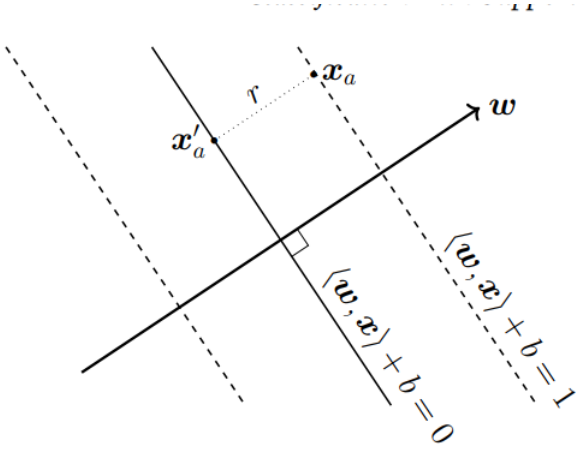

Figure 6.2: Derivation of the margin: $r = \dfrac{1}{\|\mathbf{w}\|}$

Of course, $\mathbf{w}\cdot\mathbf{x}'_a + b = 0 \Rightarrow \mathbf{w}\cdot\left(\mathbf{x}_a - r\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + b = 0 \Rightarrow \mathbf{w}\cdot\mathbf{x}_a + b - r\frac{\mathbf{w}\cdot\mathbf{w}}{\|w\|} = 0$

$\Rightarrow r = \dfrac{1}{\|\mathbf{w}\|}$

Why we can set the Margin $r = 1$**???**

Lemma 1. *6.8.1 is equivalent to scaling data, such that the margin is unity:*

$$\min_{\mathbf{w},b} \underbrace{\frac{1}{2}\|\mathbf{w}\|^2}_{margin} \text{ subject to } \underbrace{y^{(i)}(\mathbf{w}\cdot\mathbf{x}^{(i)} + b) \geq 1.}_{data\ fitting} \tag{6.25}$$

Proof. Kjak ■

Proposition 6.2

Optimization Problem 1

$$\left(\hat{\mathbf{w}}, \hat{b}\right) = \arg\min_{\mathbf{w},b} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 \right\}$$

thỏa mãn: $1 - y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) \leq 0, \forall i = \overline{1, N}$

Chú ý: Bài toán 1 là **Quadratic Programing** và điều kiện Slater dễ dàng được kiểm tra thỏa mãn $\Rightarrow$ Nghiệm của hệ điều kiện **Karush - Kuhn - Tucker** sẽ là nghiệm cho bài toán 1.

Lagrangian & Dual function

Lagrangian: $\mathcal{L}(\mathbf{w}, b, \mathbf{\Lambda}) := \frac{1}{2} \|\mathbf{w}\|^2 + \sum_{i=1}^{N} \lambda_i (1 - y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b))$
Dual function: $\mathcal{D}(\mathbf{\Lambda}) := \min_{\mathbf{w},b} \mathcal{L}(\mathbf{w}, b, \mathbf{\Lambda})$
Để ý rằng $\mathcal{L}$ là một hàm convex.

Ta có: $\begin{cases} \nabla_{\mathbf{w}} \mathcal{L} = \mathbf{w} - \sum_{i=1}^{N} \lambda_i y^{(i)} \mathbf{x}^{(i)} = 0 \\ \nabla_b \mathcal{L} = -\sum_{i=1}^{N} \lambda_i y^{(i)} = 0 \end{cases} \Rightarrow \begin{cases} \mathbf{w} = \sum_{i=1}^{N} \lambda_i y^{(i)} \mathbf{x}^{(i)} \\ \sum_{i=1}^{N} \lambda_i y^{(i)} = 0 \end{cases}$

Thay vào, ta được công thức hàm đối ngẫu tường minh như sau:

$$\mathcal{D}(\mathbf{\Lambda}) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \lambda_i \lambda_j y^{(i)} y_j \mathbf{x}^{(i)} \cdot \mathbf{x}_j$$

Proposition 6.3

Dual problem cho hard-margin S.V.M

$$\hat{\mathbf{\Lambda}} = \arg\max_{\mathbf{\Lambda}} \mathcal{D}(\mathbf{\Lambda}) = \arg\max_{\mathbf{\Lambda}} \left\{ \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \lambda_i \lambda_j y^{(i)} y_j \mathbf{x}^{(i)} \cdot \mathbf{x}_j \right\}$$

thỏa mãn: $\begin{cases} \mathbf{\Lambda} \succeq \mathbf{0} \\ \sum_{i=1}^{N} \lambda_i y^{(i)} = 0 \end{cases}$

Hệ điều kiện Karush-Kunh-Tucker cho Hard-Margin S.V.M

Bộ $(\mathbf{w}, b)$ thỏa mãn hệ điều kiện sau chính là nghiệm của **bài toán 1** (mục 6.8.1):

- Stationary : $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y^{(i)} \mathbf{x}^{(i)}; \sum_{i=1}^{N} \lambda_i y^{(i)} = 0$
- **Complementary Slackness**: $\lambda_i (1 - y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b)) = 0, \forall i = \overline{1, N}$
- Primal Feasibility: $1 - y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) \leq 0, \forall i = \overline{1, N}$

- Dual Feasibility: $\lambda_i \geq 0, \forall i = \overline{1, N}$

Support Vector Set

Từ complementary slackness, kéo theo:

$$\forall i = \overline{1, N}, \lambda_i = 0 \quad \vee \quad \mathbf{w} \cdot \mathbf{x}^{(i)} + b = y^{(i)}$$

Tập $\{\mathbf{x}^{(i)} \mid \mathbf{w} \cdot \mathbf{x}^{(i)} + b = y^{(i)}\}$ được gọi là *support vector set*.
Đặt $\mathcal{R} := \{j \mid \lambda_j > 0\}$

Proposition 6.4

Nghiệm cho bài toán 1

$\mathbf{w} = \sum\limits_{i \in \mathcal{R}} \lambda_i y^{(i)} \mathbf{x}^{(i)}$

$b = \frac{1}{|\mathcal{R}|} \sum\limits_{i \in \mathcal{R}} \left(y^{(i)} - \mathbf{w} \cdot \mathbf{x}^{(i)} \right) = \frac{1}{|\mathcal{R}|} \sum\limits_{i \in \mathcal{R}} \left(y^{(i)} - \sum\limits_{j \in \mathcal{R}} \lambda_j y_j \mathbf{x}_j \cdot \mathbf{x}^{(i)} \right)$

$$\Rightarrow \text{class}(\mathbf{x}) = \text{sign} \left(\sum_{i \in \mathcal{R}} \lambda_i y^{(i)} \mathbf{x}^{(i)} \cdot \mathbf{x} - \frac{1}{|\mathcal{R}|} \sum_{i \in \mathcal{R}} \left(y^{(i)} - \sum_{j \in \mathcal{R}} \lambda_j y_j \mathbf{x}_j \cdot \mathbf{x}^{(i)} \right) \right).$$

6.8.2 Soft-Margin Support Vector Machine

Khi dataset là gần linearly seperable hoặc chứa noise thì hard-margin S.V.M sẽ hoạt động không hiệu quả.
Ta cần 1 slack variable (biến bù) cho mỗi mẫu: $\gamma^{(i)} \geq 0$, đo lường mức độ vi phạm cho phép của quan sát thứ i. Chúng ta có thể giả sử $\gamma^{(i)} := |\mathbf{w} \cdot \mathbf{x}^{(i)} + b - y^{(i)}|$
Soft constraints: $y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) \geq 1 - \gamma^{(i)}, \forall i = \overline{1, N}$
Từ đó, ta được bài toán tối ưu cho soft-margin S.V.M

Proposition 6.5

Optimization Problem 2

$$(\hat{\mathbf{w}}, \hat{b}, \hat{\mathbf{\Gamma}}) = \arg \min_{\mathbf{w}, \mathbf{b}, \mathbf{\Gamma}} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 + \beta \sum_{i=1}^{N} \gamma^{(i)} \right\}$$

$$\text{subject to: } \begin{cases} 1 - y^{(i)}(\mathbf{w} \cdot \mathbf{x}^{(i)} + b) - \gamma^{(i)} \leq 0, \forall i = \overline{1, N} \\ -\gamma^{(i)} \leq 0, \forall i = \overline{1, N} \end{cases}$$

Lagrangian: $\mathcal{L}(\mathbf{w}, b, \mathbf{\Lambda}) := \frac{1}{2} \|\mathbf{w}\|^2 + \beta \sum\limits_{i=1}^{N} \gamma^{(i)} + \sum\limits_{i=1}^{N} \lambda_i (1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - \gamma^{(i)}) - \sum\limits_{i=1}^{N} \mu_i \gamma^{(i)}$ với $\mathbf{\Lambda}, \boldsymbol{\mu} \succeq \mathbf{0}$

Ta có: $\begin{cases} \nabla_{\mathbf{w}}\mathcal{L} = \mathbf{w} - \sum\limits_{i=1}^{N} \lambda_i y_i \mathbf{x}_i = 0 \\ \nabla_b \mathcal{L} = -\sum\limits_{i=1}^{N} \lambda_i y_i = 0 \\ \nabla_{\gamma^{(i)}} = \beta - \lambda_i - \mu_i \end{cases} \Rightarrow \begin{cases} \mathbf{w} = \sum\limits_{i=1}^{N} \lambda_i y_i \mathbf{x}_i \\ \sum\limits_{i=1}^{N} \lambda_i y_i = 0 \\ \lambda_i = \beta - \mu_i \end{cases}$

$\Rightarrow$ Dual function: $\mathcal{D}(\mathbf{\Lambda}) = \sum\limits_{i=1}^{N} \lambda_i - \frac{1}{2} \sum\limits_{j=1}^{N} \sum\limits_{i=1}^{N} \lambda_i \lambda_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$

Proposition 6.6

Dual problem 2

$$\hat{\mathbf{\Lambda}} = \arg\max_{\mathbf{\Lambda}} \left\{ \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y^{(i)} y^{(j)} \mathbf{x}^{(i)} \cdot \mathbf{x}_{(j)} \right\}$$

subject to: $\begin{cases} \mathbf{0} \preceq \mathbf{\Lambda} \preceq \boldsymbol{\beta} \\ \sum\limits_{i=1}^{N} \lambda_i y^{(i)} = 0 \end{cases}$

Bộ $(\mathbf{w}, b, \mathbf{\Gamma})$ thỏa mãn bảng 2.1 (với mọi $i = \overline{1, N}$) chính là nghiệm của **bài toán 2** (xem mục

Table 6.1: Bảng hệ điều kiện K.K.T cho dual problem của soft-margin S.V.M

Name of condition	Equation and Inequation
Stationary	$\mathbf{w} = \sum\limits_{i=1}^{N} \lambda_i y_i \mathbf{x}_i$
	$\sum\limits_{i=1}^{N} \lambda_i y_i = 0$
Complementary Slackness	$\lambda_i(1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - \gamma^{(i)}) = 0$
	$\mu_i \gamma^{(i)} = 0$
Primal Feasibility	$1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - \gamma^{(i)} \leq 0$
	$-\gamma^{(i)} \leq 0$
Dual Feasibility	$\lambda_i \geq 0$
	$\mu_i \geq 0$

Chú ý (*box constraint*): $\lambda_i \in [0; \beta]$

Support vector set cho Soft-margin S.V.M

$$\text{Đặt: } \begin{cases} \mathcal{R} := & \{i \mid \lambda_i > 0\} \\ \mathcal{S} := & \{\mathbf{x}_i \mid y_i(\mathbf{w}\cdot\mathbf{x}_i + b) = 1 - \gamma^{(i)}\} \\ \mathcal{R}_1 := & \{i \mid 0 < \lambda_i < \beta\} \\ \mathcal{S}_1 := & \{\mathbf{x}_i \mid \mathbf{w}\cdot\mathbf{x}_i + b = y_i\} \\ \mathcal{R}_2 := & \{i \mid \lambda_i = \beta\} \\ \mathcal{S}_2 := & \{\mathbf{x}_i \mid y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \leq 1\} \end{cases} \Rightarrow \begin{cases} \mathcal{S} = \{\mathbf{x}_i \mid i \in \mathcal{R}\} \\ \mathcal{S}_1 = \{\mathbf{x}_i \mid i \in \mathcal{R}_1\} \\ \mathcal{S}_2 = \{\mathbf{x}_i \mid i \in \mathcal{R}_2\} \end{cases}$$

Trong đó:

- $\mathcal{S}$ là support vector set, tập chứa các vector đóng góp vào quá trình tối ưu của loss function.
- $\mathcal{S}_1$ là tập các vector "nằm kề" lên lên rìa của margin.
- $\mathcal{S}_2$ là tập các vector nằm trong hoặc nằm trên hai boudaries.

Proposition 6.7

Solution for the problem 2

$$\mathbf{w} = \sum_{i\in\mathcal{R}} \lambda_i y^{(i)} \mathbf{x}^{(i)}$$

$$b = \frac{1}{|\mathcal{R}_1|} \sum_{i\in\mathcal{R}_1} \left(y^{(i)} - \mathbf{w}\cdot\mathbf{x}^{(i)}\right) = \frac{1}{|\mathcal{R}_1|} \sum_{i\in\mathcal{R}_1} \left(y^{(i)} - \sum_{j\in\mathcal{R}} \lambda_j y_j \mathbf{x}_j \cdot \mathbf{x}^{(i)}\right)$$

$$\Rightarrow \text{class}(\mathbf{x}) = \text{sign}\left(\sum_{i\in\mathcal{R}} \lambda_i y_i \mathbf{x}_i \cdot \mathbf{x} - \frac{1}{|\mathcal{R}_1|} \sum_{i\in\mathcal{R}_1} \left(y_i - \sum_{j\in\mathcal{R}} \lambda_j y_j \mathbf{x}_j \cdot \mathbf{x}_i\right)\right).$$

Non-constraint Optimization Problem cho Soft-margin S.V.M

Dễ thấy, **bài toán tối ưu 2** có thể có thể được đưa về dạng tối ưu không ràng buộc như sau:

$$(\hat{\mathbf{w}}, \hat{b}) = \arg\min_{\mathbf{w},\mathbf{b}} \left\{\beta \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)) + \frac{1}{2}\|\mathbf{w}\|^2\right\}$$

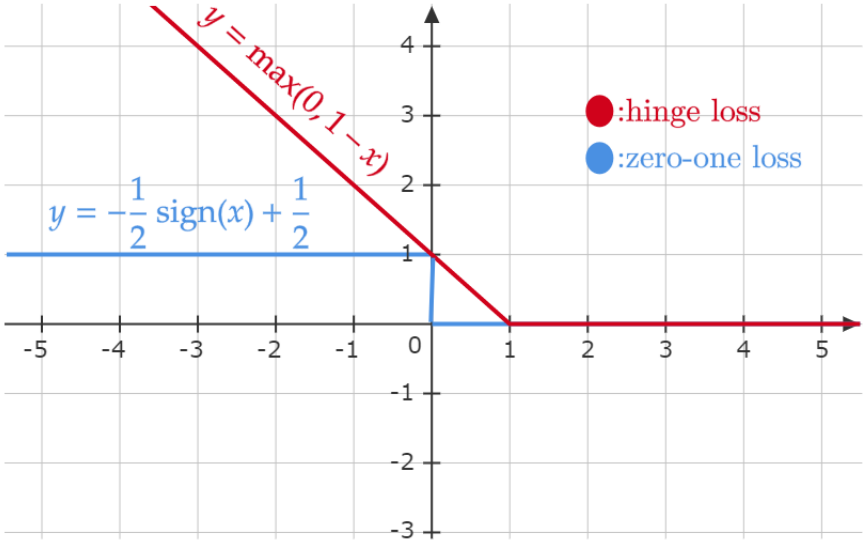

Figure 6.3: So sánh giữa hàm hinge-loss và hàm zero-one loss

6.8.3 Multi-class SVM

- Score matrix: $\mathbf{W}^\top \mathbf{X} + \mathbf{b} \triangleq \mathbf{S} \in \mathbb{R}^{C \times N}$
- Score matrix (sử dụng *bias trick*): $\mathbf{S} = \mathbf{W}^\top \mathbf{X}$
- Loss tại quan sát thứ i: $\mathcal{L}^{(i)}(\mathbf{W}; \mathbf{x}^{(i)}, y^{(i)}) = \sum\limits_{j=1, j \neq y^{(i)}}^{C} \max(0, \mathbf{s}_j^{(i)} - (\mathbf{s}_{y^{(i)}}^{(i)} - \nu))$
- Loss function:
$$\mathcal{J}(\mathbf{W}; \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1, j \neq y}^{C} \max(0, \nu - \mathbf{s}_j^{(i)} - \mathbf{s}_{y^{(i)}}^{(i)})$$
- Loss function (có sử dụng *weight decay*):
$$\widetilde{\mathcal{J}}(\mathbf{W}; \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1, j \neq y^{(i)}}^{C} \max(0, \nu - \mathbf{s}_j^{(i)} - \mathbf{s}_{y^{(i)}}^{(i)}) + \frac{\alpha}{2} \|\mathbf{W}\|_{\mathcal{F}}^2$$

6.9 Ensemble Learning and Stacking

Ensemble learning is a machine learning paradigm where multiple models, often called base learners or weak learners, are combined to solve a particular problem. The idea is that by aggregating multiple models, the ensemble can achieve better predictive performance than any individual model. In this section, we will focus on a specific ensemble technique called Stacking, which combines the predictions of multiple base models using a meta-learner.

6.9.1 Ensemble Learning

Ensemble learning techniques can generally be categorized into three main types: Bagging, Boosting, and Stacking. The key motivation behind ensemble methods is to reduce variance, bias, or improve predictions by leveraging the strengths of different models.
Mathematically, given a dataset $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, an ensemble model aims to combine the predictions from M base learners $f_1, f_2, \ldots, f_M$. The final prediction $\hat{y}$ can be expressed as:

$$\hat{y} = \sum_{m=1}^{M} w_m f_m(\mathbf{x}) \tag{6.26}$$

where w_m represents the weight assigned to the m-th base model, and $\sum_{m=1}^{M} w_m = 1$. The weights can either be fixed or learned based on the performance of the base models.

6.9.2 Stacking

Stacking, or Stacked Generalization, is an ensemble method where the predictions of multiple base models are combined using a meta-learner. Unlike Bagging and Boosting, which typically use the same type of base model, Stacking allows for the use of different types of models to take advantage of their diverse strengths.
The Stacking process involves two main levels:

- **Level-0 (Base Models):** Multiple base models are trained on the original dataset. These models can be of different types, such as decision trees, support vector machines, and neural networks. Let the predictions of the m-th base model be $\hat{y}_m$.
- **Level-1 (Meta-Learner):** A meta-learner is trained to combine the predictions of the base models. The inputs to the meta-learner are the predictions $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_M$ from the base models, and the output is the final prediction $\hat{y}$.

Mathematically, let $\mathbf{F}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})]^T$ be the vector of predictions from the M base models for an input $\mathbf{x}$. The meta-learner g takes $\mathbf{F}(\mathbf{x})$ as input and outputs the final prediction:

$$\hat{y} = g(\mathbf{F}(\mathbf{x})) \tag{6.27}$$

In practice, the dataset used to train the meta-learner is constructed by performing K-fold cross-validation on the original dataset with the base models. For each fold, the base models are trained on $K - 1$ folds, and their predictions on the held-out fold are used as features for training the meta-learner. This ensures that the meta-learner does not overfit by seeing the predictions from base models that were trained on the same data.

6.9.3 Mathematical Analysis of Stacking

Consider an ensemble consisting of M base models, each of which produces a prediction $f_m(\mathbf{x})$. The goal of the meta-learner is to minimize a loss function $L(y, \hat{y})$, where y is the true label and $\hat{y}$ is the final prediction.
The objective function for training the meta-learner can be expressed as:

$$\min_{g} \frac{1}{N} \sum_{i=1}^{N} L(y_i, g(\mathbf{F}(\mathbf{x}_i))) \tag{6.28}$$

where $\mathbf{F}(\mathbf{x}_i) = [f_1(\mathbf{x}_i), f_2(\mathbf{x}_i), \ldots, f_M(\mathbf{x}_i)]^T$ is the vector of predictions from the base models for the i-th sample.

6.9.4 Advantages and Challenges of Stacking

Advantages:

- **Increased Predictive Performance:** By combining the strengths of multiple base models, Stacking can often achieve higher accuracy than any single model.
- **Model Diversity:** Stacking allows for the use of different types of base models, which helps to capture different patterns in the data.

Challenges:

- **Complexity:** Stacking introduces additional complexity compared to simpler ensemble methods like Bagging and Boosting. It requires careful selection of both base models and the meta-learner.
- **Risk of Overfitting:** If the meta-learner is too powerful or if the base models are highly correlated, there is a risk of overfitting.

Algorithm 1 EmerGNN: pair-wise representation learning with flow-based GNN.

Require: $(u, v), L, \delta, \sigma, \{\boldsymbol{W}^{(\ell)}, \boldsymbol{w}^{(\ell)}\}_{\ell=1...L}\}$.

$\{(u, v)$: drug pair; L: the depth of path-based subgraph; δ: activation function; σ: sigmoid function; $\{\boldsymbol{W}^{(\ell)}, \boldsymbol{w}^{(\ell)}\}_{\ell=1...L}\}$: learnable parameters.$\}$

initialize the $u \to v$ pair-wise representation as $\boldsymbol{h}^0_{u,e} = \boldsymbol{f}_u$ if $e = u$, otherwise $\boldsymbol{h}^0_{u,e} = \boldsymbol{0}$;

initialize the $v \to u$ pair-wise representation as $\boldsymbol{h}^0_{v,e} = \boldsymbol{f}_v$ if $e = v$, otherwise $\boldsymbol{h}^0_{v,e} = \boldsymbol{0}$;

for $\ell \leftarrow 1$ to L **do**

  for $e \in \mathcal{V}_{\mathrm{D}}$ **do** {This loop can work with matrix operations in parallel.}

    message for $u \to v$:

    $\boldsymbol{h}^{(\ell)}_{u,e} = \delta\left(\boldsymbol{W}^{(\ell)} \sum_{(e',r,e)\in\mathcal{N}_{\mathrm{D}}} \sigma\left((\boldsymbol{w}^{(\ell)}_r)^\top[\boldsymbol{f}_u; \boldsymbol{f}_v]\right) \cdot \left(\boldsymbol{h}^{(\ell-1)}_{u,e'} \odot \boldsymbol{h}^{(\ell)}_r\right)\right)$;

    message for $v \to u$:

    $\boldsymbol{h}^{(\ell)}_{v,e} = \delta\left(\boldsymbol{W}^{(\ell)} \sum_{(e',r,e)\in\mathcal{N}_{\mathrm{D}}} \sigma\left((\boldsymbol{w}^{(\ell)}_r)^\top[\boldsymbol{f}_u; \boldsymbol{f}_v]\right) \cdot \left(\boldsymbol{h}^{(\ell-1)}_{v,e'} \odot \boldsymbol{h}^{(\ell)}_r\right)\right)$;

  end for

end for

Return $\boldsymbol{W}_{\mathrm{rel}}[\boldsymbol{h}^{(L)}_{u,v}; \boldsymbol{h}^{(L)}_{v,u}]$.

Chapter 7 EXPERIMENTS & EVALUATION

7.1 Regression for Hotel Room Review Score Prediction

This section presents the experimental results and evaluation metrics for predicting hotel room review scores using Linear Regression. The model was trained using K-fold cross-validation to ensure robustness and generalizability of the results. The experiment involved analyzing multicollinearity using the Variance Inflation Factor (VIF), selecting appropriate features, and evaluating the model using metrics such as RMSE, R^2, and Adjusted R^2.

7.1.1 Variance Inflation Factor (VIF) Analysis

To address multicollinearity, VIF analysis was conducted to identify and remove highly correlated features. The initial VIF analysis showed several features with high VIF values, indicating multicollinearity. These features were iteratively removed until all remaining features had VIF values below a reasonable threshold.

Table 7.1: Initial VIF Analysis

Feature	**VIF** heightCleanliness
37.0833 Facilities	31.2937 Comfort
25.6720 Families_room	6.3443 Pool
5.9527 Free_parking	4.5460 24h_front_desk
3.8321 Star	3.5938 Room_service
2.8766 No_smoking_room	2.7074 Airport_shuttle
1.9544 Price	1.6313 Breakfast
1.4830 Review_count	1.1200 height

After removing features with high VIF values, the final set of features selected for training was as follows:

Table 7.2: Final VIF Analysis After Removing Multicollinearity

Feature	**VIF** heightFree_parking
3.6386 24h_front_desk	3.3036 Star
3.2864 Room_service	2.6647 No_smoking_room
2.0670 Airport_shuttle	1.9207 Breakfast
1.4637 Price	1.4609 Facilities
1.4362 Review_count	1.1065 height

The features selected for the regression model after VIF analysis were: `['price', 'review_count', 'Facilities', 'star', 'No_smoking_room', 'Room_service', 'Free_parking', 'Breakfast', '24h_front_desk', 'Airport_shuttle']`. extbfInsights: The initial VIF analysis revealed that certain features, such as `Cleanliness`, extttFacilities, and extttComfort, had very high VIF values, indicating strong multicollinearity. This could distort the model's interpretation of the coefficients, leading to unreliable predictions. By iteratively removing features with high VIF, we were able to reduce multicollinearity and retain a more interpretable and stable set of features for the regression model.

7.1.2 Model Training and Cross-Validation

The Linear Regression model was trained using 5-fold cross-validation. The results for each fold are summarized in the following table:

Table 7.3: 5-Fold Cross-Validation Results

extbfFold	extbfRMSE	extbfR^2	extbfAdjusted R^2 heightFold 1
0.5523	0.8357	0.8284 Fold 2	0.7724
0.6686	0.6539 Fold 3	0.7898	0.6191
0.6022 Fold 4	0.7841	0.6323	0.6160 Fold 5
0.6271	0.7268	0.7147 height**Average**	0.7051
0.6965	0.6830 height		

Insights: The cross-validation results indicate variability in performance across the different folds, with RMSE ranging from 0.5523 to 0.7898. The average R^2 value of 0.6965 suggests that approximately 69.65% of the variance in the review scores can be explained by the features included in the model. The Adjusted R^2 of 0.6830 accounts for the number of predictors used, indicating a slight penalty for adding more features. This suggests that the model has a good but not perfect fit, highlighting areas where additional features or non-linear models might be beneficial.

7.1.3 Model Coefficients

The coefficients of the Linear Regression model, after accounting for multicollinearity, are presented in the table below:

Table 7.4: Linear Regression Coefficients

Feature	**Coefficient** heightPrice
0.0826 Review_count	0.0836 Facilities
1.0158 Star	-0.0280 No_smoking_room
-0.2862 Room_service	0.2798 Free_parking
0.0023 Breakfast	-0.1007 24h_front_desk
-0.3212 Airport_shuttle	0.2126 Intercept
8.0834 height	

Insights: The regression coefficients provide important insights into the relationship between the features and the predicted review score:

- **Facilities** has the largest positive coefficient (1.0158), indicating that better facilities are strongly associated with higher review scores.
- **Review_count** and extbfPrice both have positive coefficients, suggesting that higher review counts and price are linked with better customer satisfaction, possibly due to perceived quality.
- **Star** has a slightly negative coefficient (-0.0280), which could imply that after controlling for other factors, star rating alone might not significantly contribute to customer satisfaction, or there could be a negative bias towards higher-rated hotels due to higher expectations.
- **No_smoking_room**, **Breakfast**, and **24h_front_desk** have negative coefficients, indicating that these features may not significantly improve satisfaction or might even detract from it under certain conditions.

7.1.4 Final Evaluation

The final evaluation of the Linear Regression model on the validation and test sets yielded the following results:

Table 7.5: Final Model Evaluation Results

Dataset	**RMSE**	R^2	**Adjusted** R^2 heightValidation
0.6965	0.6899	0.6672 Test	0.4727
0.8554	0.8448 height		

Insights: The model's performance on the test set is quite strong, with an R^2 value of 0.8554, indicating that 85.54% of the variance in review scores is explained by the model. The lower RMSE (0.4727) on the test set compared to the validation set suggests that the model generalizes well to new data. The high Adjusted R^2 (0.8448) further confirms that the selected features contribute meaningfully to the model's predictive power, with minimal overfitting.
Overall, the Linear Regression model performed well in predicting the hotel room review scores. The careful selection of features through VIF analysis ensured that multicollinearity was reduced, resulting in more stable and interpretable model coefficients. While the model shows promise, future improvements could include the addition of non-linear terms or the exploration of other models such as decision trees or ensemble methods to capture more complex relationships in the data.

7.2 Deep Learning for Hotel Room Review Score Prediction

In this section, we present the results of using a fine-tuned ResNet18 model combined with zero-shot learning for predicting hotel room review scores. The model was trained for three epochs, and the results for training, validation, and test datasets are shown below.

7.2.1 Training and Evaluation Metrics

The results for each epoch, including Train Loss, Validation Loss, Test Loss, and RMSE metrics, are summarized in the following table:

Table 7.6: Training, Validation, and Test Metrics for Fine-Tuned ResNet18

Epoch	**Train Loss**	**Val Loss**	**Test Loss**	**Train RMSE**	**Val RMSE**	**Test RMSE**
1	0.6822	0.3867	0.4392	0.8913	0.6249	0.6733
2	0.5542	0.0902	0.2441	0.7778	0.2932	0.5001
3	0.4349	0.0648	0.1939	0.6836	0.2366	0.4490

7.2.2 Insights

Training Progress: The model's loss decreased significantly over the three epochs, both for the training and validation datasets. This indicates that the model was able to learn

meaningful features effectively, leading to improved performance as training progressed.

RMSE Analysis: The RMSE values for both the validation and test sets also showed a consistent decline across epochs, suggesting that the model's predictive power improved with more training. By the third epoch, the RMSE for the test set was reduced to 0.4490, indicating a reasonable fit.

Zero-Shot Learning: The use of zero-shot learning allowed the model to generalize well to new data, even without specific training examples for each possible scenario. This was evident in the consistently lower validation and test losses compared to the training loss, which implies that the model did not overfit and was able to effectively transfer learned features to unseen data.

Model Performance: The final RMSE of 0.4490 for the test set suggests that the ResNet18 model, after fine-tuning, is capable of capturing complex patterns in the data, including those represented in the images. This is a notable improvement compared to the Linear Regression model, highlighting the value of deep learning techniques for this type of task where visual information plays a crucial role.

Conclusion: The results demonstrate that a fine-tuned ResNet18, when combined with zero-shot learning, can effectively predict hotel room review scores with higher accuracy compared to traditional linear regression. The decrease in RMSE values across epochs, along with the lower validation and test losses, indicate that the model was well-optimized and successfully leveraged visual features to improve predictive performance.

7.3 Classification for Hotel Room Type Prediction using Logistic Regression

In this section, we present the experimental results for predicting hotel room types using a Logistic Regression model. The classification task involves predicting whether a hotel room is a "Normal Room" (review_score < 7), "Good Room" ($7 \leq$ review_score < 9), or "Excellent Room" (review_score ≥ 9). The dataset was split into training, validation, and test sets, and the model was evaluated using K-fold cross-validation.

7.3.1 Training and Data Preparation

After preprocessing, the dataset was split into training, validation, and test sets with the following distributions:

- Training set: 1033 samples, 14 features.
- Validation set: 221 samples, 14 features.

- Test set: 223 samples, 14 features.

Initially, there were four classes (0, 1, 2, 3), but due to the low number of samples in Class 0 (1 sample), it was merged into Class 1. The final distribution of classes in the training set was:

- Class 1: 196 samples (Normal Room)
- Class 2: 613 samples (Good Room)
- Class 3: 224 samples (Excellent Room)

7.3.2 Cross-Validation Results

The Logistic Regression model was trained using 5-fold cross-validation. The metrics for each fold are summarized in the following table:

Table 7.7: 5-Fold Cross-Validation Metrics for Logistic Regression

Fold	Accuracy	Precision	Recall	F1-Score	AUC-ROC	AUC-PR
1	0.9034	0.9108	0.9034	0.9048	0.9552	0.9400
2	0.8454	0.8565	0.8454	0.8464	0.9483	0.9328
3	0.8599	0.8720	0.8599	0.8621	0.9524	0.9349
4	0.7767	0.8341	0.7767	0.7865	0.9077	0.8883
5	0.8350	0.8622	0.8350	0.8398	0.9390	0.9255
Average	0.8441	0.8671	0.8441	0.8479	0.9405	0.9243

Insights: The cross-validation results indicate that the model performed consistently across all folds, with an average accuracy of 84.41%. The AUC-ROC value of 0.9405 suggests that the model has a strong ability to distinguish between the different classes. The model's performance in terms of precision and recall is also quite balanced, which is crucial given the imbalanced nature of the dataset after merging Class 0.

7.3.3 Model Parameters and Coefficients

The final Logistic Regression model was trained on the entire dataset after cross-validation. The following table presents the model coefficients for each feature:

Table 7.8: Logistic Regression Coefficients

Feature	**Class 1**	**Class 2**	**Class 3**
Price	-1.1394	0.4520	0.6875
Review_count	-0.4015	0.1896	0.2119
Comfort	-1.6322	-0.9319	2.5641
Cleanliness	-1.5122	0.4023	1.1099
Facilities	-0.2497	-0.6817	0.9314
Star	0.5740	-0.1361	-0.4379
Pool	0.2249	0.0312	-0.2560
No_smoking_room	0.3253	0.2681	-0.5933
Families_room	-0.0577	0.0165	0.0412
Room_service	-0.3503	0.1926	0.1578
Free_parking	-0.3306	0.1443	0.1863
Breakfast	0.3599	0.0816	-0.4415
24h_front_desk	1.1567	0.1092	-1.2659
Airport_shuttle	-0.3821	-0.1840	0.5661

Insights: The coefficients provide insights into how each feature influences the probability of a room being classified into each category:

- Features like **Comfort** and **Cleanliness** have high positive coefficients for Class 3, indicating that these features significantly increase the likelihood of a room being classified as "Excellent."
- **Price** has a mixed impact across classes, with a positive coefficient for higher classes, suggesting that higher-priced rooms are more likely to be rated as "Good" or "Excellent."
- Features such as **24h_front_desk** and **Airport_shuttle** show contrasting effects across classes, indicating differing customer preferences for these amenities depending on the room type.

7.3.4 Test Set Evaluation

- Class 0 đã được gộp vào Class 1
- Giữ nguyên các labels:
  - Class 1: Phòng bình thường ($review\_score < 7$)
  - Class 2: Phòng tốt ($7 \leq review\_score < 9$)
  - Class 3: Phòng xuất sắc ($review\_score \geq 9$)

Phân phối sau khi gộp

- Class 1: 48 mẫu
- Class 2: 128 mẫu
- Class 3: 47 mẫu

Phân bố classes trong tập test sau khi gộp

- Class 1: 48 mẫu
- Class 2: 128 mẫu
- Class 3: 47 mẫu

Đang predict với Logistic Regression...

Phân phối classes trong tập test

- Class 1: 48 mẫu
- Class 2: 128 mẫu
- Class 3: 47 mẫu

Phân phối classes được dự đoán

- Class 1: 59 mẫu
- Class 2: 112 mẫu
- Class 3: 52 mẫu

Kết quả đánh giá trên tập test

- Accuracy: 0.8475
- Precision: 0.8620
- Recall: 0.8475
- F1-score: 0.8503
- AUC_ROC: 0.9272
- AUC_PR: 0.9009

7.4 Stacking Model for Hotel Room Type Prediction

Preprocessing and Model Initialization

Handling missing values in categorical features...

Normalizing features...

Processing target variable...

Unique classes in target: [0, 1, 2, 3]

Class distribution: [2, 293, 876, 306]

Checking for NaN values after preprocessing:

Features contain NaN: **False**

Targets contain NaN: **False**

Getting train/val/test splits...

Initializing model...

Training set shape: (1033, 14)

Validation set shape: (221, 14)

Test set shape: (223, 14)

Starting Training with K-fold Cross Validation

Actual number of classes: 4

Classes: [0, 1, 2, 3]

Class	Number of Samples
Class 0	1 sample
Class 1	195 samples
Class 2	613 samples
Class 3	224 samples

Table 7.9: Class Distribution in Training Set

Class Merging Information

- Class 0 merged into the nearest class.
- New labels:

- Class 1: Normal Room ($review\_score < 7$)
- Class 2: Good Room ($7 \leq review\_score < 9$)
- Class 3: Excellent Room ($review\_score \geq 9$)

Class	Number of Samples
Class 1	196 samples
Class 2	613 samples
Class 3	224 samples

Table 7.10: Class Distribution After Merging

5-Fold Cross Validation

Fold 1/5:

Training SVM...
Training KNN...
Training Decision Tree...
Training Random Forest...
Meta features shape: (207, 12)
Training meta model...

Metric	Fold 1 Result
Accuracy	0.9758
Precision	0.9760
Recall	0.9758
F1-score	0.9759
AUC-ROC	0.9974
AUC-PR	0.9963

Table 7.11: Metrics for Fold 1

Fold 2/5, Fold 3/5, Fold 4/5, Fold 5/5 follow similar structure...

Final Model Evaluation on Test Set

Predicting with ensemble base models... Meta features shape for test set: (223, 12)

Class	Number of Samples
Class 1	48 samples
Class 2	128 samples
Class 3	47 samples

Table 7.12: Class Distribution in Test Set

Metric	Test Set Result
Accuracy	0.9552
Precision	0.9555
Recall	0.9552
F1-score	0.9550
AUC-ROC	0.9903
AUC-PR	0.9873

Table 7.13: Evaluation Results on Test Set

BIBLIOGRAPHY

[Cox58] David R Cox. "The regression analysis of binary sequences." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 20.2 (1958), pp. 215–232.

[Alt92] Naomi S Altman. "An introduction to kernel and nearest-neighbor nonparametric regression." In: *The American Statistician* 46.3 (1992), pp. 175–185.

[FS95] Yoav Freund and Robert E. Schapire. "A desicion-theoretic generalization of on-line learning and an application to boosting." In: *Computational Learning Theory*. Ed. by Paul Vitányi. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 23–37. ISBN: 978-3-540-49195-8.

[Ho95] Tin Kam Ho. "Random decision forests." In: *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE. 1995, pp. 278–282.

[Bre96] Leo Breiman. "Bagging predictors." In: *Machine learning* 24.2 (1996), pp. 123–140.

[Fri01] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine." In: *Annals of statistics* (2001), pp. 1189–1232.

[Wu+08] Xindong Wu et al. "Top 10 algorithms in data mining." In: *Knowledge and information systems* 14.1 (2008), pp. 1–37.

[RK12] K.H. Rosen and K. Krithivasan. *Discrete Mathematics and Its Applications: With Combinatorics and Graph Theory*. McGraw-Hill Companies, 2012. ISBN: 9780070681880. URL: https://books.google.com.tr/books?id=IQ2jAgAAQBAJ.

[CG16] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System." In: *CoRR* abs/1603.02754 (2016). arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[Gér17] Aurélien Géron. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017. ISBN: 978-1491962299.

[FB19] L. Favero and P. Belfiore. *Data Science for Business and Decision Making*. Elsevier Science, 2019. ISBN: 9780128112168. URL: https://books.google.com.vn/books?id=ToC%5C_swEACAAJ.

[BBG20] P. Bruce, A. Bruce, and P. Gedeck. *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python*. O'Reilly Media, 2020. ISBN: 9781492072898. URL: https://books.google.com.vn/books?id=F2bcDwAAQBAJ.

[DFO20] Marc Peter Deisenroth, A. Aldo Faisal, and Cheng Soon Ong. *Mathematics for Machine Learning*. Cambridge University Press, 2020.

[Cha21] S.H. Chan. *Introduction to Probability for Data Science*. Michigan Publishing, 2021. ISBN: 9781607857464. URL: https://books.google.com.vn/books?id=TKKkzgEACAAJ.