**VIETNAM GENERAL CONFEDERATION OF LABOR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**REPORT**

**DESIGN AND ANALYSIS OF ALGORITHMS**

*Instructor*: **NGUYEN CHI THIEN**

*Student*: **BUI ANH PHU - 521H0508**

**BUI HAI DUONG - 521H0220**

**NGUYEN HOANG PHUC - 521H0511**

*Class*: **21H50302**

**HO CHI MINH CITY, 2023**

**VIETNAM GENERAL CONFEDERATION OF LABOR**

**TON DUC THANG UNIVERSITY**

**FACULTY OF INFORMATION TECHNOLOGY**



**REPORT**

**DESIGN AND ANALYSIS OF ALGORITHMS**

*Instructor*: **NGUYEN CHI THIEN**

*Student*: **BUI ANH PHU - 521H0508**

**BUI HAI DUONG - 521H0220**

**NGUYEN HOANG PHUC - 521H0511**

*Class*: **21H50302**

**HO CHI MINH CITY, 2023**

**ACKNOWLEDGEMENT**

To complete this essay, besides our own efforts, we have received a lot of help in terms of knowledge, experience, and skills from the school and teachers. First and foremost, we would like to express my special gratitude to Nguyen Chi Thien - the lecturer of Design and Analysis of Algorithms course who has taught us valuable knowledge of the subject. That knowledge is the foundation for us to continue learning and effectively apply it to this essay. Additionally, We would like to thank the teacher for allowing us to complete this essay, which has helped us to further develop my understanding of the subject. Thank you for your guidance and support in helping us to complete this essay to the best of my ability. Moreover, we would also like to express our gratitude to the school and the teachers who have compiled the Design and Analysis of Algorithms materials, providing me with useful resources for research and essay writing. Thank you sincerely!

**COMPLETION OF THESIS**

**AT TON DUC THANG UNIVERSITY**

We here by certify that this thesis is my/our own work and was conducted under the guidance of Nguyen Chi Thien. The research and results presented in this thesis are truthful and have not been published previously in any form. The data presented in tables and figures used for analysis, comments, and evaluations were collected by the author from various sources and are clearly cited in the reference section.

Moreover, this thesis includes some comments, evaluations, and data from other authors and organizations, which are properly cited and referenced.

If any misconduct is detected, I fully take responsibility for the content of my thesis. Ton Duc Thang University is not liable for any copyright infringement that may occur during the thesis completion process.

*Ho Chi Minh City, October 22, 2023*

*Author*

*(signature and full name)*

ACKNOWLEDGEMENT AND EVALUATION SECTION BY INSTRUCTOR

**Instructor's Acknowledgement Section**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh City, 2023

(signature and full name)

**Instructor's Evaluation Section**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ho Chi Minh City, 2023

(signature and full name)

SUMMARY

This report covers essential aspects of machine learning models, their suitability for various problems, and their pros and cons. It also demonstrates practical problem-solving with real-world data, including data preprocessing, model construction, and feature selection. Finally, it explores the critical issue of overfitting and methods to address it.

**TABLE OF CONTENTS**

[ACKNOWLEDGEMENT i](#_Toc27334)

[ACKNOWLEDGEMENT AND EVALUATION SECTION BY INSTRUCTOR iii](#_Toc20266)

[SUMMARY v](#_Toc12672)

[TABLE OF CONTENTS 1](#_Toc19563)

[CHAPTER 1 – THE ASPECTS OF MACHINE LEARNING MODELS 4](#_Toc9367)

[1.1 K-Nearest Neighbors Algorithm (KNN) 4](#_Toc5436)

[1.1.1 Standard KNN 4](#_Toc15356)

[1.1.1.1 The goal of creating a Standard KNN model 4](#_Toc28962)

[1.1.1.2 The methods/algorithms for learning the Standard KNN model 4](#_Toc23343)

[1.1.1.3 Aprropriate, advantages and disadvantages 5](#_Toc12155)

[1.1.2 Weighted KNN 6](#_Toc11886)

[1.1.2.1 The goal of creating a Weighted KNN model 6](#_Toc25728)

[1.1.2.2 The methods/algorithms for learning the Weighted KNN model 6](#_Toc30263)

[1.1.2.3 Aprropriate, advantages and disadvantages 7](#_Toc32569)

*[1.1.3 Radius-based KNN](#_Toc23535)* [8](#_Toc23535)

[1.1.3.1 The goal of creating a Radius-based KNN model 8](#_Toc2060)

[1.1.3.2 The methods/algorithms for learning the Radius-based KNN model 9](#_Toc172)

[1.1.3.3 Aprropriate, advantages and disadvantages 11](#_Toc5331)

[1.2 Linear Regression 13](#_Toc16250)

*[1.2.1 The goal of creating a Linear Regression model](#_Toc9887)* [13](#_Toc9887)

*[1.2.2 The methods/algorithms for learning the Linear Regression model](#_Toc9872)* [13](#_Toc9872)

*[1.2.3 Aprropriate, advantages and disadvantages](#_Toc1681)* [16](#_Toc1681)

[1.3 Naive Bayes Classifiers 16](#_Toc1862)

*[1.3.1 Gaussian Naive Bayes](#_Toc8811)* [16](#_Toc8811)

[1.3.1.1 The goal of creating a Gaussian Naive Bayes model 16](#_Toc15071)

[1.3.1.2 The methods/algorithms for learning the Gaussian Naive Bayes model 17](#_Toc11981)

[1.3.1.3 Aprropriate, advantages and disadvantages 21](#_Toc27219)

*[1.3.2 Multinomial Naive Bayes](#_Toc17467)* [22](#_Toc17467)

[1.3.2.1 The goal of creating a Multinomial Naive Bayes model 22](#_Toc22980)

[1.3.2.2 The methods/algorithms for learning the Multinomial Naive Bayes model 22](#_Toc5464)

[1.3.2.3 Aprropriate, advantages and disadvantages 26](#_Toc31811)

*[1.3.3 Bernoulli Naive Bayes](#_Toc4092)* [28](#_Toc4092)

[1.3.3.1 The goal of creating a Bernoulli Naive Bayes model 28](#_Toc31501)

[1.3.3.2 The methods/algorithms for learning the Bernoulli Naive Bayes model 28](#_Toc2073)

[1.3.3.3 Aprropriate, advantages and disadvantages 28](#_Toc29548)

*[1.3.4 Categorical Naive Bayes](#_Toc17021)* [30](#_Toc17021)

[1.3.4.1 The goal of creating a Categorical Naive Bayes model 30](#_Toc2269)

[1.3.4.2 The methods/algorithms for learning the Categorical Naive Bayes model 30](#_Toc9975)

[1.3.4.3 Aprropriate, advantages and disadvantages 31](#_Toc5419)

[1.4 Decision Tree 32](#_Toc23667)

*[1.4.1 The goal of creating a Decision Tree model](#_Toc20128)* [32](#_Toc20128)

*[1.4.2 The methods/algorithms for learning the Decision Tree model](#_Toc26362)* [32](#_Toc26362)

*[1.4.3 Aprropriate, advantages and disadvantages](#_Toc15483)* [38](#_Toc15483)

[CHAPTER 2 – MODELS IMPLEMENTATION 41](#_Toc24256)

[1.1 Features description 41](#_Toc21572)

[1.2 Model description 43](#_Toc20652)

[CHAPTER 3 – OVERFITTING 45](#_Toc11974)

[3.1 What is Overfitting? 45](#_Toc7157)

[3.2 Cause of Overfitting 45](#_Toc31143)

[3.3 Solutions 46](#_Toc32198)

*[3.3.1 Regularization](#_Toc13051)* [46](#_Toc13051)

[3.3.1.1 L1 regularization 47](#_Toc17173)

[3.3.1.2 L2 regularization (Weight Decay) 48](#_Toc26187)

[3.3.1.3 Dropout 49](#_Toc6664)

*[3.3.2 Cross-validation](#_Toc7904)* [50](#_Toc7904)

*[3.3.3 Early stopping](#_Toc1948)* [51](#_Toc1948)

*[3.3.4 Prunning](#_Toc8529)* [52](#_Toc8529)

*[3.3.5 Data augmentation](#_Toc27497)* [53](#_Toc27497)

*[3.3.6 Ensembling](#_Toc20138)* [54](#_Toc20138)

[REFERENCES 56](#_Toc2751)

CHAPTER 1 – THE ASPECTS OF MACHINE LEARNING MODELS

* 1. K-Nearest Neighbors Algorithm (KNN)
     1. Standard KNN
        1. The goal of creating a Standard KNN model

KNN stands for k-nearest neighbors, is a non-parametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point.

Part of a family of “lazy learning” models, meaning that it only stores a training dataset, all the computation occurs when a classification or prediction is being made.

1.1.1.2 The methods/algorithms for learning the Standard KNN model

Knn is instance-based or memory-based learning method as it heavily depends on the dataset.

**Distance metrics:** determining which data points are closest to query point, these distances must be calculated.

**Euclidean distance** (p=2):

**Manhattan distance** (p=1):

**Minkowski distance:**

**Hamming distance:**

**Learning criteria:**

Data must be properly labeled (not containing Nan, null) and noise-free (clean and accurate).

Choosing k:  square root of n, where n is total number of data points.

Choosing odd value for k when solving binary classification problem.

Using cross-validation technique.

1.1.1.3 Aprropriate, advantages and disadvantages

**How this model can be applied:**

It can be used for both regression and classification, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Regression problems use a similar concept as classification problem, but in this case, the average the k nearest neighbors is taken to make a prediction about a classification.

**Advantages** **and Disadvantages:**

Easy for implementing, Adapts easily, few hyper parameters.

Does not perform well with high dimensional data inputs.

Prone to overfitting.

Can not find generalized pattern in data.

* + 1. Weighted KNN
       1. The goal of creating a Weighted KNN model

Weighted KNN model is a modified KNN model. It has the same goal of the standard KNN model by predicting new data points based on the similarity of surrounding data points. Moreover, the weighted KNN overcomes the disadvantage of the KNN model by adding weight to data points nearby through distance. That would make the prediction become better accurately.

1.1.2.2 The methods/algorithms for learning the Weighted KNN model

The method used in the weighted KNN model is basically similar to the KNN model by grouping surrounding data points to predict new data points. However, the weighted KNN uses a function called kernel function to add weight to neighbor data points based on the distance from the new data points. The closer the neighbor data point be, the bigger weight is added.

This is the training set of the observation Xi with the given class yi. This is the new data point that the model has to predict by the k nearest training data with the given class y

This is the distance between the query point X to the training data point Xi, whose class label has to be predicted.

Choosing the set D' D, which represents the k nearest training data of the new observation (query point).

Based on the distance-weighted voting, with v represents for the class labels, the prediction can be done by applying this formula:

This example will show deeply about how the W-KNN work:

A graph with red and green dots

Description automatically generated

Based on the data points which are represented by the figure, the red labels indicate the class 0 points, and the green labels indicate class 1 points. The white point in the middle is the query point which the class label should be predicted.

The W-KNN uses a function called the kernel function to give weight to the k nearest training points of the query point. This function gives more weight to the closer training points and less weight to the further points.

**Learning criteria:**

This is basically the same as the KNN model.

**Choose k:** Select the number of nearest neighbors (k) to consider.

**Select a Distance Metric:** Decide on a distance metric to measure data point similarity.

1.1.2.3 Aprropriate, advantages and disadvantages

**Problems and data:** Weighted k-Nearest Neighbors (k-NN) is a versatile machine learning algorithm that can work with various types of data, both for classification and regression tasks. It is well-suited for situations where the data exhibits certain characteristics.

**In Classification problems:** W-KNN model can be applied as a standard KNN model by predicting new data points based on the surrounding data points.

E.g., Classify cat and dog or classify normal email and spam email, etc

**In Regression problems:** W-KNN model can be applied as a standard KNN model to predict continuous values.

E.g., Predicting house price, etc.

**Advantages:**

It is a non-parametric algorithm (No assumptions).

It is useful for both classification and regression, but mostly for classification.

We don’t have a training step here, so it is much faster than any other algorithm.

Easy to implement.

Convert the multiclass classification to a binary classification.

**Disadvantages:**

Lazy learner (Testing is heavily time-consuming)

It is a Distance-based algorithm since we need feature scaling.

Sensitive to outliers

Get affected if there is an imbalanced dataset.

Cannot be used with high-dimensional datasets because of lazy learning.

* + 1. ***Radius-based KNN***
       1. The goal of creating a Radius-based KNN model

Radius Neighbors is a technique grounded in instance-based learning. Models following the instance-based learning approach extend their knowledge beyond the training data. They commence by storing the training examples. When a novel instance, or text example, emerges, they promptly establish associations between the archived training examples and this fresh instance to assign a target value for the new instance. Instance-based techniques are at times referred to as “lazy learning” methods since they defer learning until they encounter a new instance for prediction.

Contrary to estimating the hypothetical function or target function once for the entire dataset, these methods compute it locally and differently for each new instance to be predicted.

* + - 1. The methods/algorithms for learning the Radius-based KNN model

**Radius Neighbors Classifier:**

Basic Assumptions

All instances correspond to points in the n-dimensional space where n represents the number of features in any instance.

The neighbors of an instance are defined in terms of the Euclidean distance.

An instance can be represented by Euclidean distance between two instances and is given by

Euclidean Distance

Radius Neighbors Classifier begins by saving the training examples. When it needs to make a prediction for a new instance (or test example), it calculates how many training instances fall within a defined fixed radius (r) centered at the test instance. This radius (r) is set as a floating-point value chosen by the user. Subsequently, it assigns the class that appears most frequently among the training instances located within this radius to the test instance.

The optimal choice for r is by validating errors on test data.

A black and white image of a circle with a arrow pointing to the center

Description automatically generated

Radius Neighbors Classification Graphical Representation

In the above figure, “+” symbols represent training instances labeled as 1, and “-” symbols represent training instances labeled as 0. In this context, we are making a classification for the test instances xt based on the most frequently occurring class among the training instances located inside the circular region. It’s important to note that the radius of this circle (denoted as “r”) is chosen by the users. In the depicted scenario, the circle predominantly contains positive instances, so the test instance xt is classified as “+”, which corresponds to the label 1.

**Learning criteria:** The learning criteria for a Radius Neighbors Classifier involve determining the class label of a new data point based on the majority class within a specified radius of that data point. In summary:

***Radius*** ***Selection:*** The user defines the radius (r), a floating-point value that sets the neighborhood size for predictions.

***Classification:*** For a given test instance, the classifier calculates the number of training instances within the specified radius.

***Majority Voting:*** The learning criterion is to assign the class label that is most frequent among the training instances within the specified radius to the test instance. It follows a majority voting mechanism.

***Radius Neighbors Regressor:***

Radius Neighbors Regressor initially saves the training examples. When it’s time to make a prediction for a new instance (or test example), it calculates how many training instances are located within a specified fixed radius (r) centered at the test instance. This radius (r) is determined by the user and is represented as a floating-point value. Subsequently, it assigns the average value of the training instances within this radius to the test instance as its predictions.

The optimal choice for r is by validating errors on test data.

**Learning criteria:** The learning criteria for Radius Neighbors Regressor involve estimating a numerical value for a new data point based on the average (mean) of the target values of training instances within a specified radius of that data point. In summary:

***Radius Selection:*** The user specifies the radius (r), a floating-point value, which determines the neighborhood size for making predictions.

***Regression:*** When predicting the target value for a given instance, the regressor calculates the number of training instances within the specified radius.

***Mean Estimation:*** The learning criterion is to assign the average (mean) of the target values of training instances within the specified radius as the predicted target value for the test instance.

1.1.3.3 Aprropriate, advantages and disadvantages

**Appropriate for:**

**Classification and Regression:**Radius k-NN is appropriate for both classification and regression tasks. It can predict class labels for classification problems or estimate numeric values for regression problems.

**Spatial Data Analysis:**It is particularly useful in applications where spatial relationships are important, such as geospatial analysis, image processing, and location-based services.

**Localized Patterns:**When the data exhibits localized patterns, and the relationship between data points varies across different regions of the feature space, Radius k-NN can be a suitable choice.

**Small to Medium Datasets:**It is more practical for smaller to medium-sized datasets due to the computational cost of finding neighbors within a specified radius.

**Advantages:**

**Non-Parametric and Flexibility*:*** Radius k-NN is a non-parametric method, meaning it doesn’t make strong assumptions about the underlying data distribution. It can handle data with complex patterns and non-linear relationships effectively.

**Local Adaptation:**The model estimates the target function locally for each prediction, which allows it to adapt to varying data patterns in different parts of the feature space.

**Customizable Neighborhood*:*** The radius (r) parameter is user-defined, making it adaptable to different scenarios. You can choose the radius that suits the problem’s characteristics.

**Simple Concepts*:*** It’s an easy-to-understand algorithm, making it suitable for educational purpose and as a staring point for understanding instance-based learning.

**Disadvantages:**

**Sensitivity to Radius Choice:**The performance of the model is highly dependent on the choice of the radius (r). Selecting an inappropriate radius can lead to underfitting or overfitting, affecting the quality of predictions.

**Computational Intensity:**Calculating distances for all data points in the dataset can be computationally intensive, especially for large datasets. Efficient data structures, like KD-Trees or Ball Trees, can help mitigate this issue.

**Curse of Dimensionality:**As the number of features (dimensions) increases, the effectiveness of the Radius k-NN algorithm can degrade due to the “curse of dimensionality”. High-dimensional data requires a larger radius to capture meaningful neighbors, which may not be practical.

* 1. Linear Regression

***1.2.1 The goal of creating a Linear Regression model***

Mostly used for predicting and forecasting problem.

Simply means fitting a straight line to data, based on historical data, we find a pattern of information in form of a straight line showing linear relationship of some features over a numerical variable that we want to predict, the pattern is in form of this

The ultimate goal of the model is to output a vector of coefficients that are then used in predicting the target value

***1.2.2 The methods/algorithms for learning the Linear Regression model***

Cost Function using MAE (Mean Absolute Error) or MSE (Mean Squared Error ): Cost function measures the performance of a machine learning model for given data. Cost function quantifies the error between predicted and expected values and present that error in the form of a single real number.

**MAE:**

**MSE:**

**Form of cost function:**

**Gradient Descent:** An optimization Algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost). It will repeatedly update the model’s parameters to find the local or global (rare) minima of cost function value. Here is the visualization of how gradient descent works:

**A graph showing a person walking on a map

Description automatically generated with medium confidence**

Updating parameters part:

A white board with text and numbers

Description automatically generated

must be simultaneously updated

is the learning rate. This one has its own complex method for determining the best value for the model. But it will not be included in the scope of this research.

Choosing too small learning rate would make the learning process longer to reach convergence.

Choosing too large learning rate would result in overshooting or never reach minimum value.

In each epoch, will be adjusted a small portion equals derivatives part of cost function over w and b, here is how gradient descent adjusting w for example:

A diagram of a graph

Description automatically generated

⇒ At convergence, gradient descent will output a vector of coefficients that minimizes the cost function and those are used as model’s parameters.

**Learning criteria**

The input data must be numerical and continuous data, if they were categorical or binary then they should be encoded.

No outliers as linear regression is sensitive to outliers, these must be carefully handled as they could make the result skew.

***1.2.3 Aprropriate, advantages and disadvantages***

**When to apply this model:**

When it comes to regression problem (we want to predict a numerical value out of thousands of continuous value)

**Advantages:**

* Simple to implement and easy to interpret the output coefficients.
* Overfitting of this problem can be tackled by using dimensional reduction techniques, regularization technique (L1 and L2) and cross validation.
* Linear regression fits linearly separable datasets almost perfectly and is often used to find the nature of the relationship between variables.

**Disadvantage:**

* Sensitive to outliers
* It assumes that data features are independent to each other.
* Being a simple model, linear regression might not capture complex relations in the data as effectively as some non-linear model.

**1.3 Naive Bayes Classifiers**

***1.3.1 Gaussian Naive Bayes***

1.3.1.1 The goal of creating a Gaussian Naive Bayes model

Gaussian Naive Bayes is one of the Naive Bayes Classifiers techniques. It is a supervised machine learning algorithm and basically used in classification tasks.

In the Naive Bayes Classifiers models, features of the dataset do not make any effect on the others, which means they are independent.

Gaussian Naive Bayes is a popular classification algorithm that is well-suited for tasks with continuous data. Continuous data is data that can take on any value within a certain range such as height, weight, etc... It is a variant of Naive Bayes that assumes that the features are distributed according to a Gaussian distribution (normal distribution).

1.3.1.2 The methods/algorithms for learning the Gaussian Naive Bayes model

In Gaussian Naive Bayes, the X’s is assumed to be the continuous variable and it follows the Normal distribution (or Gaussian distribution).

With:

is the input value

is the given class of feature in the dataset

is the standard deviation of the class c

is the mean value of the class c

The above formula is to calculate the Class-Conditional Probabilities (Likelihoods) of the feature given the class c with the input value X.

We can calculate the probability of each input values for each class through frequency based on this formula by computing the standard deviation and the mean of X computed for a class c of Y.

The formula for calculating the mean:

With

is the input value

is the number of instances

The formula for calculating the standard deviation:

With

is the input value

is the number of instances

Using Bayes' theorem to calculate the posterior numerator of each class given the observed feature values by using this formula:

With

is the prior probability of class

is the likelihood of x given class Y

The result of the prediction is the class with the higher posterior numerical.

Example of applying Gaussian Naive Bayes:

|  |  |
| --- | --- |
| **Status** | **PSA** |
| Cancer | 4.1 |
| Cancer | 3.4 |
| Cancer | 2.9 |
| Healthy | 2.5 |
| Healthy | 2.0 |
| Healthy | 1.7 |

This table shows the PSA of a reality group of people. Each PSA value is estimated to correspond to each person in the group.

First, we calculate the mean and standard deviation of PSA on each class label based on the presented formula above:

The mean of each class:

The standard deviation of each class:

Prior Probability of each class:

A diagram of a function

Description automatically generated

Normal distribution of Cancer class

A pink cone shaped graph

Description automatically generated

Normal distribution of Healthy class

These graphs above are the presentation of the normal distribution in Healthy class and Cancer class.

Picking a random PSA value as 3(x = 3) for the prediction whether the person whose PSA equals to 3 is being Cancer or not.

The likelihood of the person with PSA = 3:

Calculating the posterior numerator of each class:

Because is higher. Which mean the result is belong to the Cancer class.

**Leaning criteria:**

**Class Priors:** Calculate the prior probabilities of each class in the training dataset. This is done by counting the number of data points in each class and dividing by the total number of data points.

**Likelihood:** For each class, estimate the class-conditional probabilities for each feature. In Gaussian Naive Bayes, it is assumed that the features follow a Gaussian (normal) distribution. To estimate these probabilities, calculate the mean and standard deviation of each feature for each class.

1.3.1.3 Aprropriate, advantages and disadvantages

**Type of data:**

A Gaussian Naive Bayes based model is a classification model and it is able to work extremely strongly with data which contains continuous variables. It can solve image classification problems, spam filtering problems and so on…

**About advantages:**

Gaussian Naive Bayes is simple to implement and understand. This is typically confirmative for a beginner to approach and practice.

Gaussian Naive Bayes has a rapid speed. It works effectively even on a big dataset.

**About disadvantages:**

Gaussian Naive Bayes is biased toward a dominant class in the training data. Therefore, the prediction of the Gaussian Naive Bayes based models returns an unexpected result, which belongs to the dominant class, even if it is not.

***1.3.2 Multinomial Naive Bayes***

1.3.2.1 The goal of creating a Multinomial Naive Bayes model

There are thousands of softwares or tools for the analysis of numerical data but there are very few for texts. Multinomial Naive Bayes is one of the most popular supervised learning classifications that is used for the analysis of the categorical text data.

Text data classification is gaining popularity because there is an enormous amount of information available in email, documents, websites, etc. that needs to be analyzed. Knowing the context around a certain type of text helps in finding the perception of a software or product to users who are going to use it.

The Multinomial Naive Bayes algorithm is a probabilistic learning technique primarily employed in Natural Language Processing (NLP). It operates based on the Bayes theorem and is utilized to predict the category or label of text, such as an email or a newspaper article. The algorithm computes the likelihood of each possible label for a given sample and then selects the label with the highest likelihood as the final output.

The Naive Bayes classifier is a family of diverse algorithms, all sharing a common underlying principle: each feature being analyzed is considered independent of all other features. This means that the presence or absence of one feature does not influence the presence or absence of any other feature.

1.3.2.2 The methods/algorithms for learning the Multinomial Naive Bayes model

Naive Bayes is a robust algorithm employed in the analysis of text data and particularly useful for problems involving multiple classes. To grasp how the Naive Bayes theorem operates, it's essential to have a prior understanding of the concept of the Bayes theorem, as it builds upon the foundation of the latter.

The Bayes theorem, originally devised by Thomas Bayes, is a mathematical approach that calculates the likelihood of an event happening, considering prior information about conditions associated with the event. It is grounded in the following formula:

Where we are calculating the probability of class A when predictor B is already provided.

= prior probability of B

= prior probability of class A

= occurrence of predictor B given class A probability

This formula helps in calculating the probability of the tags in the text.

To illustrate the Naive Bayes algorithm, let's use an example. In the table provided below, we have a dataset representing different weather conditions: sunny (S), overcast (O), and rainy (R). Our objective is to forecast the likelihood of players engaging in a game, considering these weather conditions.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Weather | S | O | R | S | S | O | R | R | S | R | S | O | O | R |
| Play | N | Y | Y | Y | Y | Y | N | N | Y | Y | N | Y | Y | N |

This can be easily calculated by following the below given steps:

Build a frequency table using the training dataset provided in the problem statement above. Specifically, record the number of occurrences for each weather condition alongside their corresponding weather conditions.

|  |  |  |
| --- | --- | --- |
| Weather | Yes | No |
| Sunny | 3 | 2 |
| Overcast | 4 | 0 |
| Rainy | 2 | 3 |
| Total | 9 | 5 |

Find the probabilities of each weather condition and create a likelihood table.

|  |  |  |  |
| --- | --- | --- | --- |
| Weather | Yes | No |  |
| Sunny | 3 | 2 | = 5/14 (0.36) |
| Overcast | 4 | 0 | = 4/14 (0.29) |
| Rainy | 2 | 3 | = 5/14 (0.36) |
| Total | 9 | 5 |  |
|  | = 9/14 (0.64) | = 5/14 (0.36) |  |

Determine the posterior probability for each weather condition using the Naive Bayes theorem. The weather condition with the highest probability will determine whether the players will engage in the activity or not.

Use the following equation to calculate the posterior probability of all the weather conditions:

After replacing variables in the above formula, we get:

Take the values from the above likelihood table and put it in the above formula.

Hence,

Take the values from the above likelihood table and put in the above formula.

The probability of playing in sunny weather conditions is elevated. Therefore, if the weather is sunny, the player is likely to participate.

In a similar manner, we can compute the posterior probability for rainy and overcast conditions. By considering the condition with the highest probability, we can make a prediction regarding the player's participation.

**Learning criteria:**

***Class Prior Probability (Priors):*** Estimate the prior probabilities of each class label in the training data. This is done by counting the frequency of each class in the training set and calculating the proportion of each class relative to the total number of samples.

***Probability (Likelihood):*** Estimate the likelihood of observing each feature (word or term) given each class. In text classification, this often involves counting the frequency of each term within each class. For numerical features, you can use probability density estimation techniques, such as the Multinomial distribution for discrete data.

***Class-Conditional Probability:*** Calculate the probability of observing each class given the features. This is achieved by applying Bayes' theorem, which combines the prior probabilities and likelihoods for each class.

***Feature Independence Assumption:*** One of the main assumptions in Naive Bayes is that features are conditionally independent given the class. This simplifying assumption, known as the "naive" assumption, is applied to make the computations more tractable. While this assumption doesn't always hold in practice, it is often effective for text classification.

1.3.2.3 Aprropriate, advantages and disadvantages

**Appropriate Problems and Data:**

***Text Classification:*** Multinomial Naive Bayes is commonly used for text classification tasks such as sentiment analysis, spam detection, topic categorization, and document classification.

***Natural Language Processing (NLP):*** It's a fundamental model in NLP and is suitable for tasks that involve working with text data.

***High-Dimensional Data:*** It performs well in high-dimensional spaces, making it efficient for problems with a large number of features or words.

**Advantages:** The Naive Bayes algorithm offers several advantages, including:

***Ease of Implementation:*** Implementing the Naive Bayes algorithm is straightforward because it mainly involves calculating probabilities. Its simplicity makes it accessible to a wide range of users.

***Applicability to Various Data Types:*** Naive Bayes can be used with both continuous and discrete data, making it versatile for various types of data analysis, including text classification and numeric data analysis.

***Real-time Predictions:*** This algorithm is well-suited for real-time applications, where predictions need to be made quickly. Its efficiency in making predictions can be valuable for applications that require immediate responses.

***Scalability:*** Naive Bayes is highly scalable and can efficiently handle large datasets. It doesn't suffer from performance issues when working with substantial amounts of data, making it a practical choice for big data scenarios..

**Disadvantages:** The Naive Bayes algorithm does come with certain limitations, including:

***Lower Prediction Accuracy:*** In some cases, Naive Bayes may have lower prediction accuracy compared to more complex machine learning algorithms, particularly when the independence assumption doesn't hold well in the data. However, it often performs well, especially in text classification tasks.

***Incompatibility with Regression:*** Naive Bayes is primarily designed for classification tasks, not regression. It is not suitable for predicting numeric values or performing regression tasks. Other algorithms, like linear regression or decision trees, are better suited for regression problems.

***1.3.3 Bernoulli Naive Bayes***

1.3.3.1 The goal of creating a Bernoulli Naive Bayes model

Using for problem where data dataset having binary features. Each feature is assumed to follow a Bernoulli distribution.

Usually, Bernoulli Naive Bayes can be applied to Text Classification, Sentiment Analysis, Document Categorization,…

1.3.3.2 The methods/algorithms for learning the Bernoulli Naive Bayes model

Based on Bernoulli distribution, this is the only difference from the Gaussian NB

1.3.3.3 Aprropriate, advantages and disadvantages

Bernoulli Naive Bayes (BNB) is tailored for boolean/binary data. As with any machine learning classifier, it comes with its own set of advantages and disadvantages:

**Advantages:**

**Simplicity and Speed:** The BNB classifier is easy to understand and implement. Its training and prediction processes are fast due to the simplicity of the underlying calculations.

**Scalability:** BNB scales well to large datasets because of its linear complexity with respect to the number of features and data points.

**Good Baseline Model:** Due to its simplicity, BNB can be used as an initial baseline model in many classification tasks.

**Performance in Low Data Regimes:** BNB can perform decently even when there's relatively little training data, especially when compared to more complex models which might overfit.

**Insensitivity to Irrelevant Features:** While not always the case, BNB can sometimes be less sensitive to irrelevant features when compared to some more complex algorithms.

**Laplace Smoothing:** BNB has built-in mechanisms like Laplace (or add-one) smoothing to handle features that don't appear in the training set for a particular class, which prevents zero probability issues.

**Works with Binary Data:** BNB is explicitly designed to work with binary data and can effectively model the presence or absence of specific features.

**Disadvantages**:

**Naive Assumption:** The biggest disadvantage of BNB (and other Naive Bayes classifiers) is the assumption that all features are conditionally independent given the class label. This assumption is rarely true in real-world data and can be a limitation.

**Binary Features Only:** BNB is designed for binary features. If the data has categorical or continuous features, they need to be binarized, which can result in a loss of information.

**Loss of Frequency Information:** BNB considers only the presence or absence of features and not their frequencies. For some datasets, especially in text classification, the frequency of features (like word counts) can be valuable, and this information is ignored in BNB.

**Performance Ceiling:** While BNB can be a good starting point and can perform surprisingly well in certain situations, it might not achieve as high accuracy as some more sophisticated models on complex datasets.

**Sensitive to Irrelevant Features:** While BNB might be less sensitive than some algorithms, the presence of many irrelevant features can still negatively impact its performance. Proper feature selection or engineering can help alleviate this.

In conclusion, while Bernoulli Naive Bayes has its strengths, especially with binary data and as a baseline, it's crucial to evaluate its performance in the context of the specific problem and dataset you're dealing with and consider it as one of the many tools in your machine learning toolbox.

***1.3.4 Categorical Naive Bayes***

1.3.4.1 The goal of creating a Categorical Naive Bayes model

Categorical Naive Bayes is one of the Naive Bayes Classifiers techniques. It is a supervised machine learning algorithm and basically used in classification tasks.

In the Naive Bayes Classifiers models, features of the dataset do not make any effect on the others, which means they are independent.

Categorical Naive Bayes is suitable for classification with discrete features that are categorically distributed. The categorical data refers to a form of information that can be stored and identified based on their names or labels. It is a type of qualitative data that can be grouped into categories instead of being measured numerically.

Categorical Naive Bayes is a very popular classification algorithm, especially for tasks with categorical data. It is often used as a baseline algorithm for classification tasks, as it is easy to implement and train.

1.3.4.2 The methods/algorithms for learning the Categorical Naive Bayes model

**Data Encoding:** The process of transforming categorical values into numerical representations based on classes. Common methods include one-hot encoding and label encoding.

**Calculate Feature Likelihoods:** After encoding, compute the likelihood of each feature occurring within each class. This step quantifies the frequency of feature-category combinations for class-conditional probabilities.

**Bayesian Probability Estimation:** Utilize Bayes' theorem to determine the posterior probability of each class for a given instance. This step involves multiplying the class prior probability by the product of class-conditional probabilities for each feature.

**Final Classification:** The class with the highest posterior probability is selected as the final prediction for the instance.

1.3.4.3 Aprropriate, advantages and disadvantages

**Suitability for Categorical Data:** Categorical Naive Bayes is an ideal choice when working with datasets that feature categorical data with discrete, non-numeric features. It is particularly effective for text classification and other applications where features represent categories or labels.

**Advantages:**

**Simplicity:** Easy to understand and implement.

**Efficiency:** Quick training and prediction.

**Text-Friendly:** Ideal for text data (e.g., document classification).

**Tolerates Irrelevant Features:** Handles irrelevant features gracefully.

**Small Datasets:** Works well with limited data.

**Disadvantages:**

**Independence Assumption:** It assumes feature independence, which may not hold in all cases.

**Limited Expressiveness:** May not capture complex relationships in data.

**Sensitive to Feature Quality:** It relies heavily on the quality of feature engineering.

**Not Suitable for Continuous Data:** Designed for categorical features, not continuous ones.

**Sensitivity to Class Imbalance:** Performance may suffer with imbalanced class distributions.

**Suboptimal for Some Tasks:** In some cases, more complex models may outperform it.

**1.4 Decision Tree**

***1.4.1 The goal of creating a Decision Tree model***

The Decision Tree algorithm falls under the category of supervised learning algorithms. What sets it apart from other supervised learning algorithms is its versatility in tackling both regression and classification tasks.

The primary objective when employing a Decision Tree is to construct a training model that can make predictions regarding the class or value of the target variable. This is achieved by learning straightforward decision rules derived from historical data, often referred to as training data.

In the context of Decision Trees, predicting a class label for a data record begins at the root of the tree. Here, we compare the values of the root attribute to the attributes of the record. Depending on the outcome of this comparison, we then proceed down the branch corresponding to that value, leading to the next node in the tree.

***1.4.2 The methods/algorithms for learning the Decision Tree model***

Types of decision trees are based on the type of target variable we have. It can be of two types:

***Categorical Variable Decision Tree:*** Decision Tree which has a categorical target variable then it called a Categorical variable decision tree.

***Continuous Variable Decision Tree:*** Decision Tree has a continuous target variable then it is called Continuous Variable Decision Tree.

For instance, consider a scenario where the task at hand is to forecast whether a customer will renew their premium with an insurance company, denoted as either 'yes' or 'no.' In this case, it's understood that a customer's income is a crucial factor influencing their decision. However, the insurance company lacks income information for all of its customers. Given the significance of income as a variable, a decision tree can be constructed to estimate customer incomes using factors like occupation, product preferences, and other relevant variables. In this particular instance, the aim is to predict values for continuous variables.

**Important Terminology related to Decision Trees**

***Root Node:*** This node represents the complete population or sample and is subsequently divided into two or more homogeneous subsets.

***Splitting:*** It refers to the procedure of dividing a node into two or more sub- nodes.

***Decision Node:*** When a sub-node further divides into additional sub-nodes, it is referred to as a decision node.

***Leaf / Terminal Node:*** Nodes that do not undergo further splitting are known as leaf or terminal nodes.

***Pruning:*** When we eliminate sub-nodes from a decision node, this process is called pruning. It can be seen as the reverse process of splitting.

***Branch / Sub-Tree:*** A segment of the entire tree is termed a branch or sub-tree.

***Parent and Child Node:*** A node that is divided into sub-nodes is identified as a parent node for those sub-nodes, while the sub-nodes are considered the children of the parent node.

A diagram of a diagram

Description automatically generated

Decision trees classify instances by guiding them through the tree structure, starting from the root and eventually arriving at a leaf or terminal node, which assigns a classification to the instance.

Every node within the tree serves as a criterion to evaluate a specific attribute, and the branches stemming from the node represent the potential outcomes of this attribute assessment. This recursive procedure is iterated for each subtree emanating from the current node.

The accuracy of a tree is greatly influenced by the strategic decisions regarding node splitting. These criteria differ between classification and regression trees.

Decision trees employ various algorithms to determine when to divide a node into two or more sub-nodes. This subdivision process aims to enhance the uniformity within the resulting sub-nodes, effectively increasing their purity concerning the target variable. The decision tree evaluates all available variables for node splitting and selects the split that yields the most homogeneous sub-nodes.

The choice of algorithm is also contingent on the nature of the target variables. Now, let's examine some of the algorithms utilized in Decision Trees:

**ID3** → (extension of D3)

**C4.5** → (successor of ID3)

**CART** → (Classification and Regression Tree)

**CHAID** → (Chi-square automatic interaction detection Performs multi-level splits when computing classification trees)

**MARS** → (multivariate adaptive regression splines)

The ID3 algorithm constructs decision trees by following a top-down and forward-looking strategy in exploring potential branches without revisiting previous choices.

Being a 'greedy' algorithm, it consistently opts for the choice that appears most favorable at the current point in time.

**Steps in ID3 algorithm:**

1. It starts with the initial set S, which serves as the root node of the tree.
2. In each iteration of the algorithm, it goes through the attributes in set S that 3. haven't been used yet. For each of these attributes, it computes both the **Entropy (H)** and **Information Gain (IG)**.
3. The algorithm then chooses the attribute with the smallest Entropy or the largest Information Gain.
4. It divides the set S into subsets based on the selected attribute, creating a subset of the data.
5. The algorithm recursively proceeds with the same process on each subset, considering only attributes that have not been chosen before.

**Learning criteria:**

Determining the appropriate attribute to place at the root or at various levels within the decision tree can be a challenging step, especially when the dataset contains a large number of attributes. Simply selecting a node at random to be the root of the tree is not a reliable solution, as it can lead to suboptimal results with lower accuracy.

To address this challenge of attribute selection, researchers have put considerable effort into developing solutions. They have proposed using specific criteria and methods to guide the selection process: **Entropy, Information gain, Gini index, Gain Ratio, Reduction in Variance, Chi-Square.**

These criteria involve the calculation of values for each attribute. These values are then sorted, and attributes are positioned within the tree in accordance with this order. In the case of using Information Gain as the criterion, we treat attributes as categorical, whereas for the Gini index, we consider attributes as continuous.

***Entropy:*** Entropy is a measure of the unpredictability of the state, or equivalent, of its average information content.A graph with a blue line

Description automatically generated

, then

***Information gain:*** Information gain is the decrease in entropy resulting from dividing a dataset based on attributes and identifying the best candidate that yields the maximum reduction:

where is a random variable and is the entropy of given the value of attribute .

***Gini index:*** It is computed by deducting the sum of squared probabilities for each class from one. This approach tends to prioritize larger partitions and is straightforward to apply, while information gain, in contrast, gives preference to smaller partitions with distinct values.

A feature with a lower Gini index is chosen for a split.

The classic CART algorithm uses the Gini Index for constructing the decision tree.

***Gain Ratio:*** The information gain ratio is the ratio between the information gain and the Split Information value:

***Reduction in Variance:*** The reduction in variance is a technique employed in regression problems when dealing with continuous target variables. This method applies the standard variance formula to identify the optimal split. It chooses the split with the least variance as the criterion for dividing the dataset:

Above is the mean of the values, is actual and n is the number of values.

***Chi-Square:*** CHAID, which stands for Chi-squared Automatic Interaction Detector, is one of the earliest classification methods based on decision trees. Its primary function is to determine the statistical significance of distinctions between child nodes and the parent node. This is assessed by measuring the sum of squared standardized differences between the observed and expected frequencies of the target variable. CHAID is designed to operate with a categorical target variable, typically labeled as 'Success' or 'Failure.' It is capable of executing two or more splits, and a higher Chi-Square value indicates a greater level of statistical significance in the differences between sub-nodes and the parent node.

It generates a tree called CHAID (Chi-square Automatic Interaction Detector).

Mathematically, Chi-squared is represented as:

Where:

= Chi-Squre obtained

∑ = the sum of

= observed score

= expected score

***1.4.3 Aprropriate, advantages and disadvantages***

**Appropriate**:

***Categorical Data:*** Decision trees are well-suited for problems with categorical data, such as customer segmentation based on demographics.

***Binary Classification:*** Decision trees are often used for binary classification problems, like spam detection.

***Exploratory Data Analysis:*** Decision trees are useful for exploring data and identifying important features.

***Feature Selection:*** Decision trees can be used for feature selection by assessing feature importance.

***Teaching and Education:*** Decision trees are commonly used in teaching machine learning concepts due to their interpretability.

***Ensemble Learning:*** Decision trees can serve as base models in ensemble methods, where their weaknesses are mitigated by combining multiple trees (e.g., Random Forests).

**Advantages:**

* It can be applied to both continuous and categorical data.
* It has the capability to yield multiple results.
* This method provides accurate results that can be quantified and trusted, ensuring the reliability of the generated trees.
* Using this approach, one can efficiently explore data, identify important variables, establish relationships between variables to enhance target variables, and create new features in a significantly shorter time.
* It is straightforward to comprehend and explain to others.
* It is useful for data cleaning, and unlike other methods, it doesn't require extensive time for handling missing values and outliers beyond a certain point.
* The efficiency and performance of decision trees remain unaffected by non-linear relationships between features.
* Data preparation is quick as it doesn't necessitate tasks like missing value replacement or data normalization.
* It is a non-parametric technique, and it is unrelated to the design and arrangement of classifiers.

**Disadvantages:**

* Certain users might construct decision trees that become excessively intricate, exceeding their personal preferences. Such complex trees lack the ability to generalize data as effectively as simpler ones.
* The prevalence of certain classes often leads to the development of biased trees. This underscores the vital importance of balancing the sample data before utilizing it.
* On occasion, these trees exhibit instability. Fluctuations in the data can result in the creation of trees that don't meet the desired criteria. This phenomenon is known as variance and can be mitigated through methods like boosting and bagging.
* Relying solely on greedy algorithms doesn't guarantee the creation of the optimal decision tree. To address this issue, it is advisable to train multiple trees.

CHAPTER 2 – MODELS IMPLEMENTATION

* 1. Features description

This is the House\_price\_predicting dataset:

* SalePrice - the property's sale price in dollars. **This is the target variable that you're trying to predict.**
* MSSubClass: The building class
* MSZoning: The general zoning classification
* LotFrontage: Linear feet of street connected to property
* LotArea: Lot size in square feet
* Street: Type of road access
* Alley: Type of alley access
* LotShape: General shape of property
* LandContour: Flatness of the property
* Utilities: Type of utilities available
* LotConfig: Lot configuration
* LandSlope: Slope of property
* Neighborhood: Physical locations within Ames city limits
* Condition1: Proximity to main road or railroad
* Condition2: Proximity to main road or railroad (if a second is present)
* BldgType: Type of dwelling
* HouseStyle: Style of dwelling
* OverallQual: Overall material and finish quality
* OverallCond: Overall condition rating
* YearBuilt: Original construction date
* YearRemodAdd: Remodel date
* RoofStyle: Type of roof
* RoofMatl: Roof material
* Exterior1st: Exterior covering on house
* Exterior2nd: Exterior covering on house (if more than one material)
* MasVnrType: Masonry veneer type
* MasVnrArea: Masonry veneer area in square feet
* ExterQual: Exterior material quality
* ExterCond: Present condition of the material on the exterior
* Foundation: Type of foundation
* BsmtQual: Height of the basement
* BsmtCond: General condition of the basement
* BsmtExposure: Walkout or garden level basement walls
* BsmtFinType1: Quality of basement finished area
* BsmtFinSF1: Type 1 finished square feet
* BsmtFinType2: Quality of second finished area (if present)
* BsmtFinSF2: Type 2 finished square feet
* BsmtUnfSF: Unfinished square feet of basement area
* TotalBsmtSF: Total square feet of basement area
* Heating: Type of heating
* HeatingQC: Heating quality and condition
* CentralAir: Central air conditioning
* Electrical: Electrical system
* 1stFlrSF: First Floor square feet
* 2ndFlrSF: Second floor square feet
* LowQualFinSF: Low quality finished square feet (all floors)
* GrLivArea: Above grade (ground) living area square feet
* BsmtFullBath: Basement full bathrooms
* BsmtHalfBath: Basement half bathrooms
* FullBath: Full bathrooms above grade
* HalfBath: Half baths above grade
* Bedroom: Number of bedrooms above basement level
* Kitchen: Number of kitchens
* KitchenQual: Kitchen quality
* TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
* Functional: Home functionality rating
* Fireplaces: Number of fireplaces
* FireplaceQu: Fireplace quality
* GarageType: Garage location
* GarageYrBlt: Year garage was built
* GarageFinish: Interior finish of the garage
* GarageCars: Size of garage in car capacity
* GarageArea: Size of garage in square feet
* GarageQual: Garage quality
* GarageCond: Garage condition
* PavedDrive: Paved driveway
* WoodDeckSF: Wood deck area in square feet
* OpenPorchSF: Open porch area in square feet
* EnclosedPorch: Enclosed porch area in square feet
* 3SsnPorch: Three season porch area in square feet
* ScreenPorch: Screen porch area in square feet
* PoolArea: Pool area in square feet
* PoolQC: Pool quality
* Fence: Fence quality
* MiscFeature: Miscellaneous feature not covered in other categories
* MiscVal: $Value of miscellaneous feature
* MoSold: Month Sold
* YrSold: Year Sold
* SaleType: Type of sale
* SaleCondition: Condition of sale
  1. **Model description**

In this session, we use various approach with different models for solving regression problem of trying to predict houses prices based on various features.

**Model prediction goal:** Predicting house price based on the given related features of that house.

**Preprocessing:**

**Cleaning data:**

* Specifying those columns that have missing values over 40% the total rows of the dataset, and drop them as they have to many missing values to properly handle.

**Cleaning data:** Using two different approach: dropping row that has missing values and Using imputation (Using SimpleImputer from sklearn.impute)

+ Two cleaned datasets are: **data\_train\_drop** and **data\_train\_imputed**

**Transformation data:** Transforming all the categorical data to numerical data by applying **OrdinalEncoder** and **BinaryEncoder** from **category\_encoders.**

More specifically, those features which has less than 3 unique classes will be encoded by BinaryEncoder and the remains using OrdinalEncoder

**Normalizing data:** Scaling all the values using MinMaxScaler.

**Training model:**

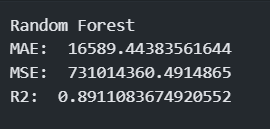
Doing Train-Test Splitting on the dataset with 75% training data, then applying

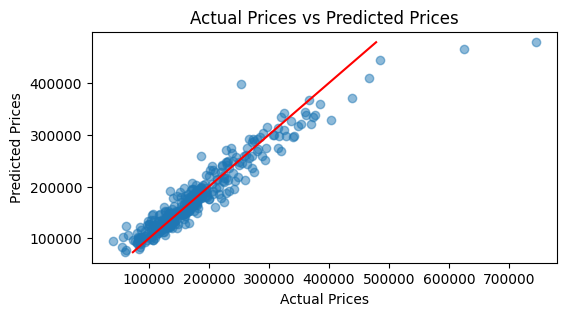
Linear Regression, Decision Tree Regressor, Random Forest Regressor, KNN regressor, Ridge Regressor respectively.

**Evaluating:**

After comparing MAE, MSE and R^2 score, The chosen model is Random Forest Regressor.

Evaluation and visualization of the model:





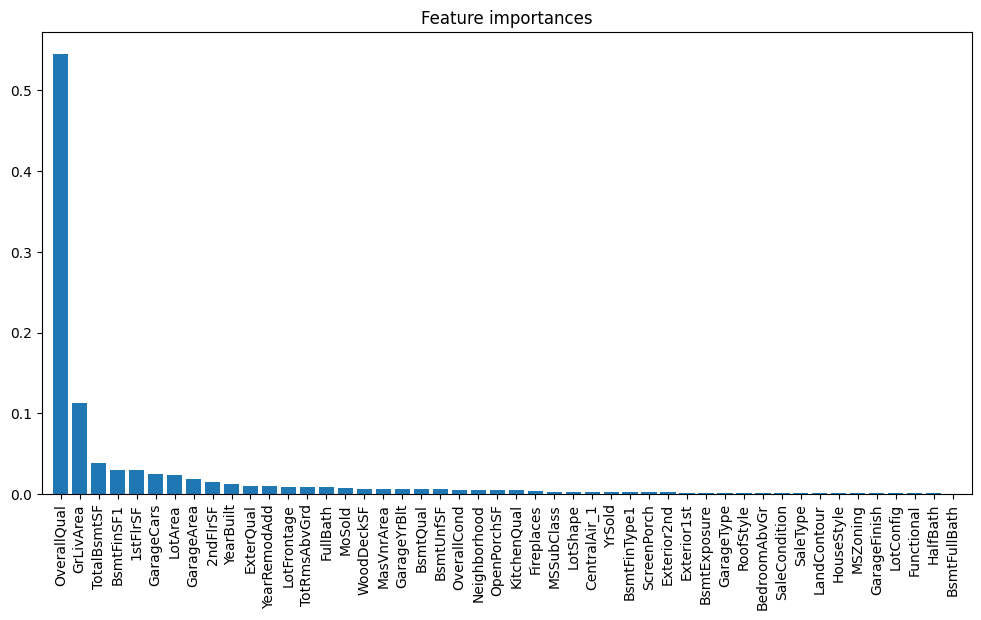
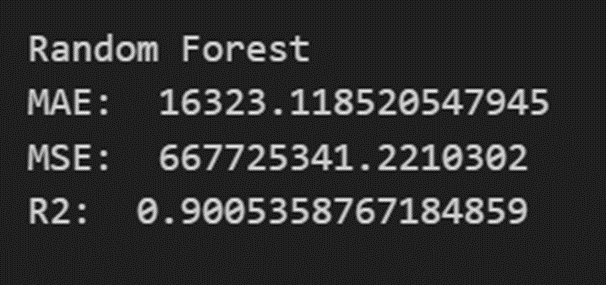
**Feature selection:** Applying for loop from i equal to 15 to numbers of all feature in dataset to choose the number of feature wanted to apply in RFE function.

Training the Random Forest Regressor model with features chosen by RFE and evaluate the model each time in the loop for looking up i best features bring up the best evaluation.

**Evaluating after applying feature selection method:**

Using MAE, MSE and R^2 score for evaluating. After applying feature selection,

the evaluation is higher than before using only 48 particular features.



**CHAPTER 3** – **OVERFITTING**

**3.1 What is Overfitting?**

Overfitting is an unfavorable occurrence in machine learning where a model provides precise predictions for the data it was trained on but struggles to make accurate predictions on new, unseen data. When data scientists employ machine learning models to make predictions, they typically begin by training the model on a known dataset. Subsequently, the model attempts to forecast outcomes for novel datasets. An overfitted model is prone to providing inaccurate predictions and does not generalize well to diverse types of new data.

**3.2 Cause of Overfitting**

**Lack of Data:** Inadequate training data can lead to overfitting as the model has limited examples to learn from. With a small dataset, the model may attempt to fit every data point exactly, which can result in a lack of generalization.

**Noisy Data:** Noise in a dataset represents irrelevant or random information. When a dataset is overly noisy, the model may mistakenly memorize the noise rather than discerning the true underlying patterns. This can lead to an overly complex and overfitted model.

**Model Complexity:** Overfitting can occur if a model is too complex relative to the amount of data available for training. Models with excessive complexity can memorize training data rather than truly understanding the underlying relationships.

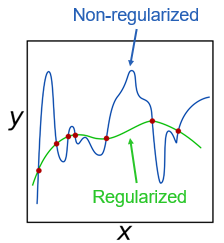
**Unbalanced Data:** If the training data is unbalanced, where some features or classes are significantly more prevalent than others, the model may disproportionately focus on the dominant features, leading to overfitting and poor generalization.

**Overlearning:** Overlearning, or overfitting through excessive training, can happen when a model is trained for too many epochs or iterations. This can lead the model to memorize the training data rather than identifying the broader, more general patterns.

**3.3 Solutions**

***3.3.1 Regularization***

In most cases, a model's output is influenced by numerous features. As the number of features grows, the model's complexity increases. In the case of overfitting, the model tends to consider all the features, regardless of whether some of them have minimal impact on the ultimate output. In more extreme cases, some of these features may be irrelevant noise that adds no meaningful value to the output.



Regularization and Non-regularization

In order to limit these cases, we have two kinds of solutions:

1. Select only the useful features and remove the useless features from our model

2. Minimize the weights of the features which have little influence on the final classification

To do this, we introduce a "penalty term" known as a regularizer into the cost function, as depicted in the following formula:

where:

: the original cost function

: the weight

: the training set

: the labeled value (true value)

: the size of training set

: the regularization coefficient

: the penalty term

Here we can use the “Gradient-Descent” method to find out the set of weights.

3.3.1.1 L1 regularization

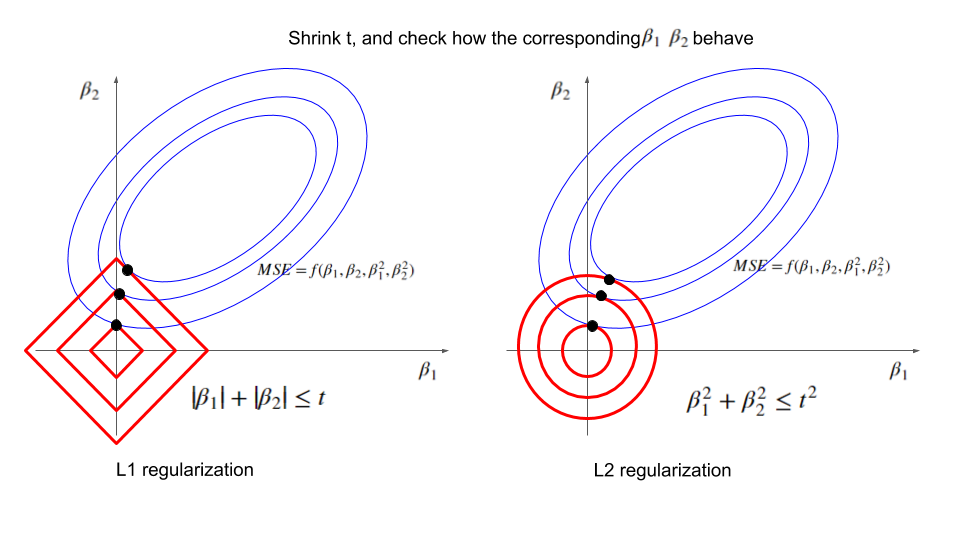
To determine the minimum value of the cost function, L1 regularization employs the principles of Lasso Regression, which is a linear regression theory. In this technique, the penalty term is calculated as the sum of the absolute values of all the weights, often referred to as the taxi-cab distance.

To reduce the cost function, it's necessary to assign a weight of zero to certain features. In simpler terms, we eliminate some features from our model and retain only the ones that are more valuable. This process results in a more straightforward model that is easier to interpret. However, it comes at the cost of discarding some useful features that have a lesser impact on the final output.

3.3.1.2 L2 regularization (Weight Decay)

L2 regularization is based on the "Ridge Regression" theory. In this method, the penalty term is determined using the Euclidean distance.

Unlike L1 regularization, this method encourages the network to prioritize learning features with small weights. Instead of discarding less valuable features, they are assigned lower weights. This way, we retain as much information as we can. Higher weights are reserved for features that significantly enhance the initial cost function.

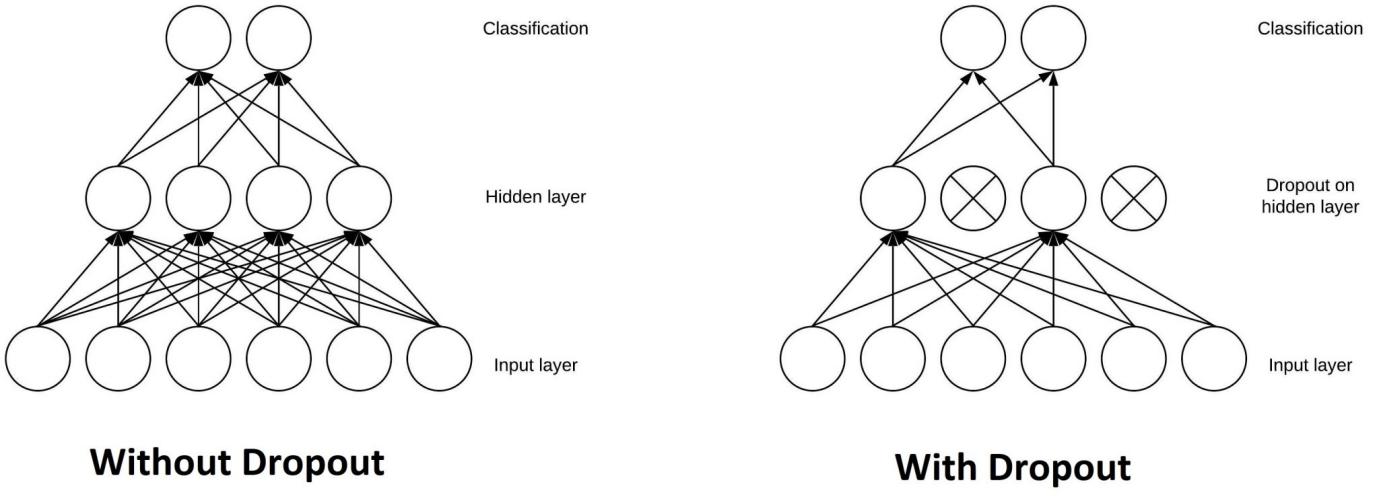


L1 regularization vs L2 regularization

3.3.1.3 Dropout

Dropout is a widely embraced and effective technique for addressing overfitting in neural networks. The core concept behind dropout is to introduce randomness by selectively deactivating units and their corresponding connections in the neural network during training. This helps prevent units from becoming overly reliant on one another. The general procedure is as follows:

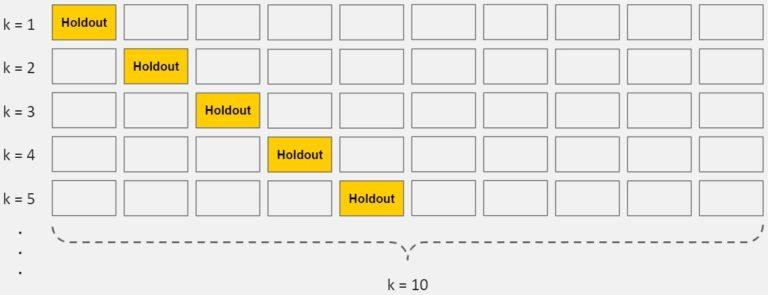
1. Randomly deactivate around half of the hidden neurons, creating a simpler network.
2. Train this simplified network using stochastic gradient descent. Gradients for each parameter are computed by averaging across the training cases in each mini-batch. If a training case doesn't involve a parameter, it contributes a gradient of zero for that parameter.
3. Reactivate the previously deactivated neurons.
4. Randomly deactivate roughly half of the hidden neurons in the reactivated network to establish a new simplified network.
5. Reiterate the above process until achieving an optimal set of parameters.



Without Dropout and With Dropout

***3.3.2 Cross-validation***

Cross-validation is a potent strategy to guard against overfitting. The concept is quite ingenious: Utilize your original training data to create several small training and testing subsets. These subsets are employed to fine-tune your model. In the typical k-fold cross-validation method, the data is divided into k subsets, known as folds. Subsequently, we systematically train the algorithm on k-1 folds while reserving the remaining fold as the test set, which is often referred to as the “holdout fold”.

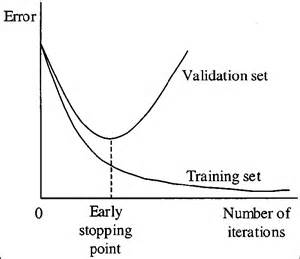


K-Fold Cross-Validation

Cross-validation enables the optimization of hyperparameters using solely the initial training dataset. This ensures that the test dataset remains entirely untouched and serves as a genuinely unseen dataset for the final model selection.

***3.3.3 Early stopping***

When you are training an iterative learning algorithm, you can assess the performance of each iteration of the model. Initially, with each new iteration, the model's performance tends to improve. However, beyond a certain point, the model's capacity to generalize may deteriorate as it starts to overfit the training data. Early stopping entails halting the training process before the learner reaches that critical point.



Early stopping

In contemporary applications, this method is predominantly employed within the domain of deep learning, while other approaches like regularization are favored in traditional machine learning.

***3.3.4 Prunning***

As we understand, the presence of noise in the data is a significant contributor to overfitting. Consequently, noise reduction naturally emerges as a research avenue for mitigating overfitting. In light of this concept, the method of “pruning” has been introduced to reduce the size of the final classifiers in relational learning, particularly in the context of decision tree learning. Pruning represents an important theoretical framework used to decrease the complexity of classification by eliminating less informative or irrelevant data, ultimately preventing overfitting and enhancing classification accuracy.

There are two standard approaches, pre-pruning and post-pruning, employed to address noise:

* Pre-pruning algorithms operate during the learning process. They typically employ stopping criteria to determine when to halt the addition of conditions to a rule or the inclusion of rules in a model description. These criteria may include constraints like limiting the length of encoding based on the evaluation of encoding cost, significance testing to assess differences between the distributions of positive and negative examples, or the use of cutoff stopping criteria defined by a predetermined threshold.
* Post-pruning involves dividing the training set into two subsets: a “growing set” and a “pruning set”. In contrast to pre-pruning, post-pruning algorithms do not directly address overfitting issues during the learning process on the growing set. Instead, they manage overfitting by removing conditions and rules from the model generated during learning. This approach tends to be more accurate but can be less efficient.

***3.3.5 Data augmentation***

Expanding the training dataset through data augmentation is a powerful strategy for reducing overfitting. This technique involves diversifying the data by applying operations like flipping, translation, rotation, scaling, brightness adjustments, and noise addition to increase the number of images in the dataset.

In real-world scenarios gathering large amounts of data is a tedious & time-consuming task, hence the collection of new data is not a viable option.

This technique is shown in the below diagram.

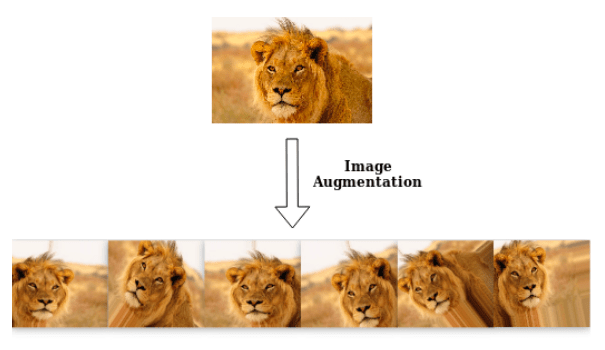


Image Augmentation

Data augmentation is a technique that creates additional images with variations, enabling the network to train on multiple instances of the same object class from different angles. This increases dataset size, reducing overfitting as the model can't overfit on all samples and must generalize.

For instance, data augmentation can produce images with a lion viewed from various angles, even upside down or with portions cropped. The “cutout” augmentation helps the network learn distinctive features, like a male lion's mane.

In essence, data augmentation prevents overfitting by expanding the training dataset, forcing the network to generalize. It does increase training loss since the model may not predict augmented images accurately, but the optimizer adapts to the general trends in the dataset.

***3.3.6 Ensembling***

Ensemble techniques in machine learning involve combining predictions from multiple independent models. Two commonly used ensemble methods are:

* **Bagging:** This method aims to reduce the risk of overfitting complex models. It trains numerous “strong” learners concurrently, where a strong learner is relatively unconstrained. Bagging then combines the outputs of these strong learners to create a more stable prediction.
* **Boosting:** This technique focuses on enhancing the predictive power of simple models. It sequentially trains multiple “weak” learners, where a weak learner is a more constrained model, like decision trees with limited depth. Each learner in the sequence learns from the errors of the previous one. Boosting merges the weak learners into a single, more robust model.

Although both bagging and boosting are ensemble approaches, they take opposite approaches to the problem. Bagging employs complex base models and aims to smooth their predictions, while boosting uses simple base models and seeks to increase their overall predictive power.

**REFERENCES**

1. <https://www.ibm.com/topics/knn>
2. <https://towardsdatascience.com/knn-algorithm-what-when-why-how-41405c16c36f>
3. <https://statusneo.com/getting-cozy-with-knn-your-guide-to-the-intuitive-k-nearest-neighbors-algorithm/>
4. <https://www.geeksforgeeks.org/implementation-of-radius-neighbors-from-sratch-in-python/>
5. <https://www.ibm.com/topics/gradient-descent><https://builtin.com/machine-learning/cost-function>
6. <https://www.geeksforgeeks.org/ml-advantages-and-disadvantages-of-linear-regression/>
7. <https://iq.opengenus.org/gaussian-naive-bayes/>
8. <https://www.upgrad.com/blog/gaussian-naive-bayes/>
9. <https://www.upgrad.com/blog/multinomial-naive-bayes-explained/>
10. <https://www-users.cselabs.umn.edu/Fall-2020/csci5521-001/Notes/DecTreeExplained.pdf>
11. <https://www.upgrad.com/blog/decision-trees-in-machine-learning/>
12. <https://en.wikipedia.org/wiki/Entropy_%28information_theory%29>
13. <https://en.wikipedia.org/wiki/Information_gain_ratio>
14. <https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022/pdf>
15. <https://www.analyticsvidhya.com/blog/2021/06/complete-guide-to-prevent-overfitting-in-neural-networks-part-1/>
16. [https://elitedatascience.com/overfitting-in-machine-learning#:~:text=Cross%2Dvalidation%20is%20a%20powerful,into%20k%20subsets%2C%20called%20folds.](https://elitedatascience.com/overfitting-in-machine-learning" \l ":~:text=Cross%2Dvalidation%20is%20a%20powerful,into%20k%20subsets%2C%20called%20folds.)
17. [https://aws.amazon.com/what-is/overfitting/#:~:text=Overfitting%20occurs%20when%20the%20model,to%20several%20reasons%2C%20such%20as%3A&text=The%20training%20data%20size%20is,all%20possible%20input%20data%20values.](https://aws.amazon.com/what-is/overfitting/" \l ":~:text=Overfitting%20occurs%20when%20the%20model,to%20several%20reasons%2C%20such%20as%3A&text=The%20training%20data%20size%20is,all%20possible%20input%20data%20values.)