Template đồ án tiểu luận bằng tiếng anh

công nghệ thông tin (Đại học Tôn Đức Thắng)

VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

NGUYEN LAM DUY – 521H0499
TRAN HUU NHAN – 521H0507
NGUYEN HOANG PHUC – 521H0510

MIDTERM REPORT INTRODUCTION TO MACHINE LEARNING

HO CHI MINH CITY, YEAR 2023

VIETNAM GENERAL CONFEDERATION OF LABOUR

TON DUC THANG UNIVERSITY

FACULTY OF INFORMATION TECHNOLOGY

NGUYEN LAM DUY – 521H0499
TRAN HUU NHAN – 521H0507
NGUYEN HOANG PHUC – 521H0510

MIDTERM REPORT

INTRODUCTION TO MACHINE LEARNING

Advised by

Assoc. Prof.Le Anh Cuong

HO CHI MINH CITY, YEAR 2023

ACKNOWLEDGMENT

We would like to express our deepest gratitude to Assoc. Prof. Le Anh Cuong for his invaluable guidance and support throughout the preparation of this report. Your expertise and insights have been instrumental in shaping our understanding and approach to machine learning. Thank you for your time, patience, and dedication.

Ho Chi Minh City, day 22nd month 10 year 2023
Author
(Signature and full name)

Tran Huu Nhan

Nguyen Hoang Phuc

Nguyen Lam Duy

DECLARATION OF AUTHORSHIP

We hereby declare that this thesis was carried out by ourselves under the guidance and supervision of Assoc. Prof. Le Anh Cuong; and that the work and the results contained in it are original and have not been submitted anywhere for any previous purposes. The data and figures presented in this thesis are for analysis, comments, and evaluations from various resources by my own work and have been duly acknowledged in the reference part.

In addition, other comments, reviews and data used by other authors, and organizations have been acknowledged, and explicitly cited.

We will take full responsibility for any fraud detected in our thesis. Ton Duc Thang University is unrelated to any copyright infringement caused on my work (if any).

Ho Chi Minh City, 22nd month 10 year 2023
Author
(Signature and full name)
Tran Huu Nhan
Nguyen Hoang Phuc
Nguyen Lam Duy

ABSTRACT

This report will showcase our group’s research on various machine learning models, such as KNN, Linear Regression, Naive Bayes classifiers, and Decision Tree. We will explain the basic concepts, assumptions, and algorithms of each model, as well as their potential applications in different domains and scenarios. We will also show the advantages and disadvantages of these models in terms of complexity, interpretability, scalability, robustness, and generalization ability.

In the second part of the report, we will demonstrate how to use these models to solve a real-world problem: diagnosing Hepatitis C based on laboratory values and demographic data. We will perform data preprocessing steps such as cleaning, transformation, and normalization to prepare the data for analysis. We will build different machine learning models using scikit-learn library in Python and experiment with different parameters and settings. We will evaluate the performance of the models using various metrics such as accuracy, precision, recall, f1-score.

In the third part of the report, we will discuss one of the common challenges in machine learning: overfitting. Overfitting occurs when the model performs well on the training data but poorly on the test data or new data. It means that the model has learned too much from the noise or specific patterns in the training data that are not generalizable to other data. We will explain the causes and consequences of overfitting, as well as some methods to prevent or mitigate it, such as regularization, cross-validation, pruning, early stopping, ensemble methods, etc.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

KNN	K Nearest Neighbor
NB	Naïve Bayes

CHAPTER 1. INTRODUCTION TO MACHINE LEARNING ALGORITHMS AND APPLICATIONS

1.1 The goal of creating a machine learning model

The primary goal of creating a machine learning model is to build an algorithm that can learn and make predictions based on the given data, it could be either labeled, unlabeled, or mixed data. Different machine learning algorithms are suited to different goals, such as classification or prediction modeling.

1.2 The methods/algorithms for learning models, and what the learning criteria are?

There are various machine learning methods, including supervised learning, unsupervised learning, semi-supervised, and reinforcement learning. Some common algorithms include Support Vector Machines, Decision Trees, Neural Networks, k-Means Clustering, Random Forests, and many others.

Machine learning criteria usually include:

Loss function (measure the distance between the model's prediction and the ground truth data, the lower the result, the more accurate the model)

Base on which algorithm is being used, different evaluation metric can be applied:

Regression: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE)

Classification: confusion matrix, accuracy, precision, recall, F1 score, ...

1.3 Which models are appropriate for what types of problem and data? Their advantages and disadvantages?

- Linear Regression: Suitable for predicting continuous values, e.g., predicting house prices based on area. Simple and interpretable but assumes a linear relationship.

- Logistic Regression: Used for binary classification problems, e.g., email spam detection. Linear model with interpretable results.
- Decision Trees: Suitable for both classification and regression tasks, easy to understand, and can handle non-linear relationships, but prone to overfitting.
- Random Forests: Improve decision tree's generalization by combining multiple trees. Robust and less prone to overfitting.
- Neural Networks: Suitable for various problems, especially in computer vision and natural language processing. Can model complex relationships but may require large amounts of data and computation.
- Support Vector Machines: Useful for classification and regression, especially when data is linear or can be linearly transformed. Effective in high-dimensional spaces.
- k-Means Clustering: Used for data clustering, e.g., customer segmentation. Simple but sensitive to the choice of the number of clusters (k).
- Reinforcement Learning: Suitable for sequential decision-making tasks, such as autonomous driving or game playing. Can learn from interactions but often requires extensive training.

1.4 Analyze and compare models

1.4.1 K-Nearest Neighbors (KNN)

Introduction

The K-Nearest Neighbors (KNN) model is a supervised learning method that uses training data to predict labels for new data points. It stores training data and their labels, and when classifying a new point, it calculates distances to known points and uses a voting method among the nearest neighbors to determine the label.

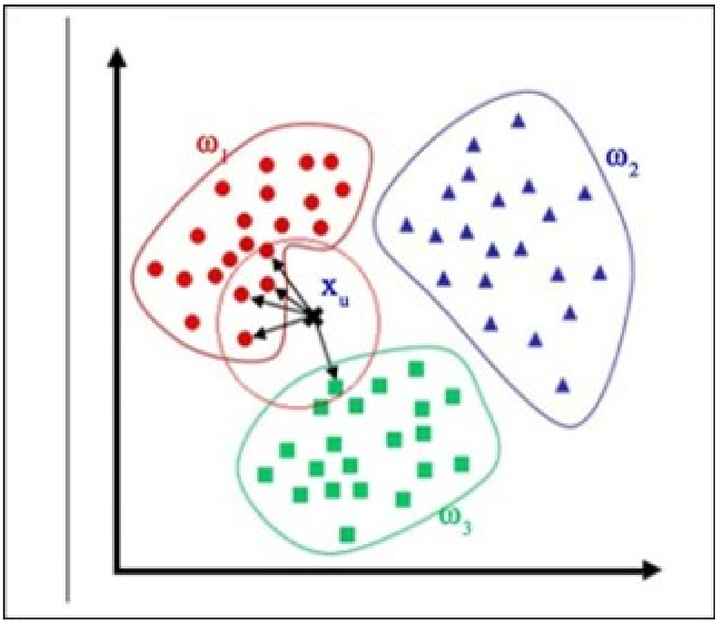


Figure 1-1 *An illustration of K nearest neighbor model. (Zhang, 2017)*

For classification, KNN considers the labels of its k nearest neighbors. The label is determined by either majority voting, where each neighbor contributes one vote and the label with the most votes is assigned, or weighted voting, where votes are weighted based on proximity to the data point.

Euclidean distance:

$$D(p,q)=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+\ldots+(p_n-q_n)^2}$$

Equation 1.1 Euclidean distance formula

Where p and q are subjects to be compared with n characteristics. There are also other methods to calculate distance such as Manhattan distance.

For regression, KNN typically determines the output of a data point by taking the average or weighted average of the output values of its nearest neighbors. When K = 1, the output of the nearest data point is directly assigned as the predicted output. When K > 1, the predicted output is calculated as the average or weighted average of the outputs of the K nearest neighbors.

Applicability of KNN

KNN can be used to solve both classification and linear regression problems.

- Image classification: KNN is used to classify images based on visual similarities. For example, identifying handwritten digits or detecting diseased plants.
- Document classification: KNN can classify text documents into different categories like spam/not-spam based on their word frequencies.
- Gene expression classification: In bioinformatics, KNN is used to classify genes with similar expression patterns that may be co-regulated or share similar functions.
- Time series forecasting: KNN regression can make predictions for time series data based on similar historical patterns. For example, stock price prediction.
- Anomaly detection: By finding data points with low proximity to their nearest neighbors, KNN can identify outliers and anomalies.
- Credit scoring: Banks use KNN for credit scoring to classify loan applicants as low/high risk based on similarities to previous applicants.
- Handwriting recognition: Recognizing handwritten digits and letters by comparing an input image to labeled examples.

Pros of KNN:

- Simple and easy to understand.
- Effective for both classification and regression tasks.
- Makes no assumptions about data distribution.
- Can model complex nonlinear decision boundaries.

Cons of KNN:

- Computationally expensive for large datasets.

- Performance degrades with high dimensional data due to curse of dimensionality.
- Sensitive to irrelevant or noisy features.
- Requires feature scaling for meaningful distances.

In summary, while KNN is flexible, simple, and versatile for both classification and regression tasks, it scales poorly compared to Naive Bayes and Linear Regression. Decision Trees are interpretable but can overfit easily. Despite these limitations, KNN serves as a good baseline approach for many tasks.

1.4.2 Linear Regression

Introduction

Linear Regression is a supervised learning method that models the relationship between a dependent variable and one or more independent variables. It assumes that the dependent variable is a linear function of the independent variables, plus some random errors. It can be used for both regression and classification tasks.

In regression, the goal is to estimate the coefficients of the linear function that best fits the data. The general form of the multiple linear regression model is:

$$y=\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p+\varepsilon$$

Equation 1.2 multiple linear regression formulation

Where y is the dependent variable, $\beta_0,\beta_1,\beta_2,\ldots,\beta_p$are the regression coefficients $x_0,x_1,x_2,\ldots,x_n$are independent variables in the model. In the classical regression setting it is usually assumed that the error term εfollows the normal distribution with $E(\varepsilon)=0$ and a constant variance $Var(\varepsilon)=\sigma^2$. *(Xin Yan, 2009)*

Applicability of Linear Regression

- Linear Regression can be used to solve various problems that involve predicting a continuous or categorical outcome based on numerical or categorical features. Here are some examples of Linear Regression being used in different domains:

- Predicting house prices based on features such as size, location, number of rooms, etc.
- Predicting customer satisfaction based on features such as service quality, product quality, price, etc.
- Predicting credit risk based on features such as income, debt, credit history, etc.
- Predicting student grades based on features such as attendance, homework, test scores, etc.

Pros of Linear Regression

- Simple and easy to interpret. The coefficients indicate the direction and magnitude of the effect of each feature on the outcome.
- Fast to train and predict. Computational complexity is low compared to other methods.
- Good for linear data. It can capture linear relationships between features and outcome.

Cons of Linear Regression:

- Prone to underfitting. It may not capture nonlinear or complex patterns in the data.
- Makes strong assumptions about data distribution. It assumes that the error term is normally distributed and independent of the features.
- Sensitive to outliers and multicollinearity. Outliers can distort the regression line and inflate the error. Multicollinearity can cause instability in the coefficient estimates and reduce interpretability.

1.4.3 Naive Bayes Classifiers

Introduction

Naive Bayes classifiers are simple probabilistic classifiers based on Bayes' theorem with strong independence assumptions among features. They are scalable, requiring parameters linear to the number of features. Training can be done through a closed-form expression in linear time, avoiding costly iterative approximation.

Despite their simplicity, they can achieve higher accuracy when combined with kernel density estimation.

The formula for Bayes' theorem is:

$$P(A \vee B) = \frac{P(B \vee A)P(A)}{P(B)}$$

Equation 1.3 Bayes's theorem formula

where:

A and B are events

P(A) is the prior probability of A

P(B) is the prior probability of B

P(A|B) is the posterior probability of A given B

P(B|A) is the likelihood of B given A

Applicability of Naive Bayes classifiers

Naive Bayes can be used for binary and multiclass classification problems. They have been highly successful in text classification problems, such as spam filtering and sentiment analysis, due to their ability to handle an extremely large number of features. Here are some application:

- Spam Filtering: Naive Bayes spam filtering is a baseline method for dealing with spam that can tailor itself to the email needs of individual users and give low false positive spam detection rates that are generally acceptable to users.
- Product Recommendation: Naive Bayes is also used in product recommendation based on product attributes and user preferences.
- Document Categorization: Naive Bayes text classification is considered a good choice for this task. For example, it can be used for face recognition in computer vision.

Pros of Naive Bayes

- It is easy and fast to predict the class of the test data set. It also performs well in multi-class prediction.
- When the assumption of independence holds, a Naive Bayes classifier performs better compared to other models like logistic regression.
- Performs well in the case of categorical input variables compared to numerical variables. For numerical variables, a normal distribution is assumed (bell curve, which is a strong assumption).

Cons of Naive Bayes

- Zero Frequency: If a category in the test data wasn't in the training data, the model assigns it zero probability, making predictions impossible. Smoothing techniques like Laplace estimation can help.
- Bad Estimator: Naive Bayes isn't reliable for probability outputs.
- Assumption of Independence: It assumes predictors are independent, which is rarely true in real life.

In summary, Naive Bayes classifiers are great tools for quick and easy binary or multiclass classification tasks. They're especially useful for text classification tasks and work well with high-dimensional datasets. However, they do make strong assumptions about your data, so they won't work well for every problem.

1.4.4 Decision trees

Introduction

Decision Trees are a form of Supervised Machine Learning that continuously divides data based on a specific parameter. The tree consists of two elements: decision nodes and leaves. Leaves represent the decisions or results, while decision nodes are points where the data is divided.

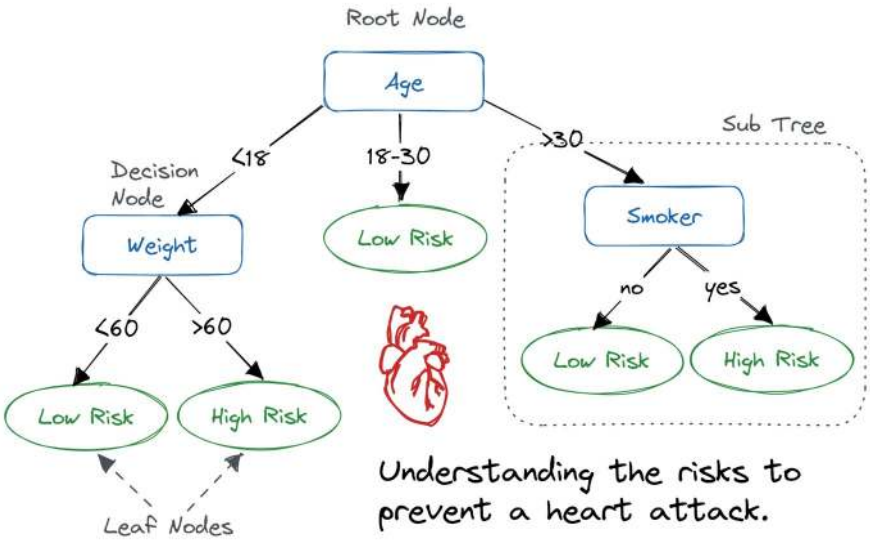


Figure 1-2 Decision tree example in heart attack (Abid Ali Awan)

Applicability of Decision Trees

Decision Trees are a simple and useful form of machine learning that can be used for both classification and regression problems. Here are some examples:

- Medical Diagnosis: Decision trees are used to predict patients' likelihood of having a certain disease based on specific characteristics such as age, gender, and symptoms.
- Credit Risk: Banks use decision trees to predict whether a loan applicant is a high-risk or low-risk customer based on their income, employment status, credit history, etc.
- Customer Segmentation: Businesses use decision trees to segment customers into different groups based on their purchasing behavior, demographics, etc.

Pros of Decision Trees

- Easy to Understand: Decision trees output rulesets that are easy for humans to understand.
- Less Data Cleaning Required: They require less data cleaning compared to some other modeling techniques.

- Data Type as Not a Constraint: Decision trees are versatile and capable of handling both numerical and categorical variables without any limitations.
- Non-parametric Method: Decision trees are considered a non-parametric method, which means that decision trees have no assumptions about the space distribution and the classifier structure.

Cons of Decision Trees

- Overfitting: This issue can be addressed by imposing constraints on model parameters and employing pruning techniques.
- Challenges with Continuous Variables: Decision trees encounter difficulties when dealing with continuous numerical variables, as they tend to lose valuable information during the categorization process.

In summary, Decision Trees are simple to understand and interpret, and are useful for both classification and regression. However, they can easily overfit the data and therefore need tuning. They also lose information when working with continuous variables.

CHAPTER 2. APPLYING MACHINE LEARNING MODELS TO REAL-WORLD PROBLEMS

2.1 Introduction

Hepatitis C is a liver disease that affects millions worldwide. Machine learning is increasingly being used in healthcare for early detection and diagnosis can analyze comprehensive health data, hospital databases, to facilitate early detection and diagnosis of diseases.

2.2 Materials and methods

2.2.1 Dataset

The dataset used in this study was obtained from UCI dataset. It contained information related to the values of blood donors and Hepatitis C patients and demographic values like age.

Shape of dataset: 615 instances and 12 Features. The target attribute for classification is Category (blood donors vs. Hepatitis C, including its progress: 'just' Hepatitis C, Fibrosis, Cirrhosis).

Table 1 Features description of dataset

#	No. Features	Description	Types
1	Age	The age of patient	Numerical
2	Sex	Male, female	Binary(m, f)
3	Albumin Blood Test (ALB)	Measures the amount of albumin in your blood. Low albumin levels can indicate liver or kidney disease or another medical condition.	Continuous
4	Alkaline Phosphatase (ALP)	The test measures the amount of ALP in your blood. ALP is an enzyme found in many parts of your body. Each part of your body produces a different type of ALP.	Continuous
5	Aspartate aminotransferase (AST)	It is an enzyme found mostly in the liver but also in muscles and other organs in your body. When damaged cells contain AST, they release the AST into your blood.	Continuous
6	BIL	Amount of Bilirubin in your blood.	Continuous
7	(Cholinesterase) CHE	An enzyme that helps the nervous system function properly.	Continuous

8	(Cholesterol) CHOL	A type of fat found in your blood. High levels can indicate a risk for heart disease.	Continuous
9	(Creatinine) CREA	A waste product that forms when creatine, found in muscle, breaks down. High levels may indicate kidney damage.	Continuous
10	(Gamma-glutamyl Transferase) CGT	An enzyme mostly found in the liver. High levels may indicate liver disease or damage to the bile ducts.	Continuous
11	(Protein) PROT	Proteins serve as building blocks for many organs, hormones, and enzymes. Hight or low levels can indicate various health condition.	Continuous
12	Category	Target column. (values: '0=Blood Donor', '0s=suspect Blood Donor', '1=Hepatitis', '2=Fibrosis', 3=Cirrhosis'	Categorical
13	(Alanine Transaminase) ALT	An enzyme is mainly found in the liver. High levels may in indicate liver damage.	Continuous

2.2.2 Using python to apply

Data Prepare

Download HCV dataset from UCI. Link download: [HCV data - UCI Machine Learning Repository]

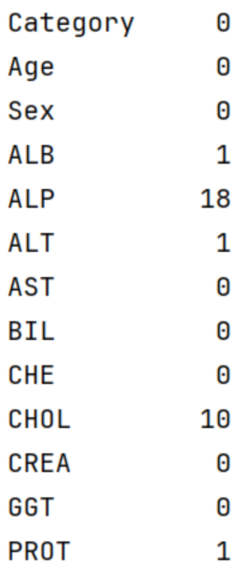
Read data using pandas

Explore dataset

Using function of pandas data frame to print feature columns, shape, dtypes, description, info of dataset

Data preprocessing

Cleaning: data have missing values that can replace with mean() of data.



```
Category     0
Age          0
Sex          0
ALB          1
ALP         18
ALT          1
AST          0
BIL          0
CHE          0
CHOL        10
CREA         0
GGT          0
PROT         1
```

Figure 2-3 Missing values in dataset

Because data has 2 columns with categories values (category, sex) and these column does not contain missing values so that I cut 2 columns out of dataset to fill missing value. After that, append 2 columns (category, sex) to dataset.

Also check duplicate row but all rows are distinct.

Encode: We use both Label Encoding and One hot encoding. But binary such as sex suitable with one hot encoder than Label encoder so that We chose One Hot encoder.

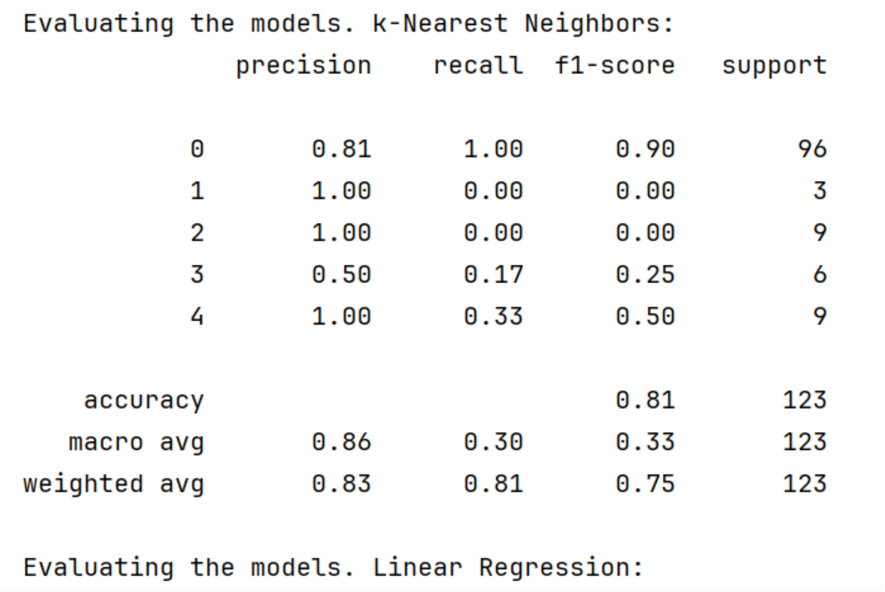
Normalization

Using Standard Scaler

Train Model

Using KNN, Linear Regression, Naïve Bayes, Decision Tree.

2.3 Evaluating the models.



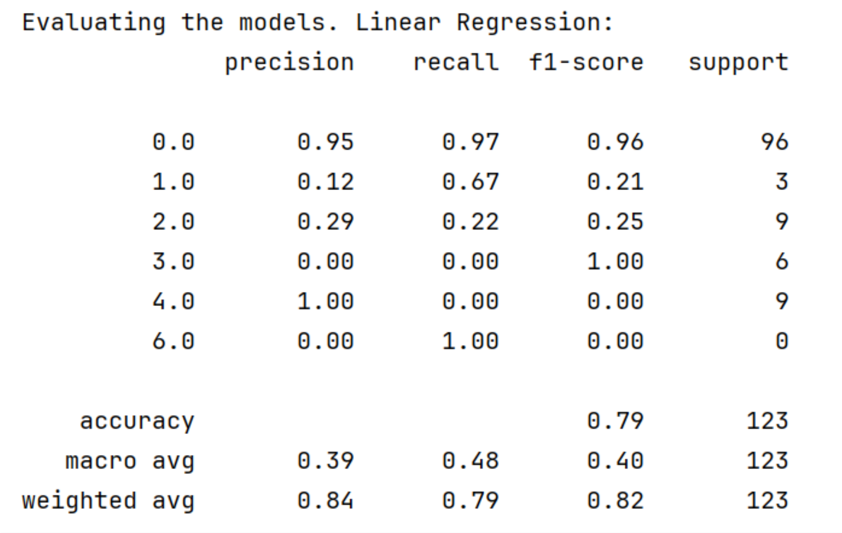
```
Evaluating the models. k-Nearest Neighbors:
              precision    recall  f1-score   support

           0       0.81      1.00      0.90        96
           1       1.00      0.00      0.00         3
           2       1.00      0.00      0.00         9
           3       0.50      0.17      0.25         6
           4       1.00      0.33      0.50         9

    accuracy                           0.81       123
   macro avg       0.86      0.30      0.33       123
weighted avg       0.83      0.81      0.75       123

Evaluating the models. Linear Regression:
```

Figure 2-4 Classification report KNN

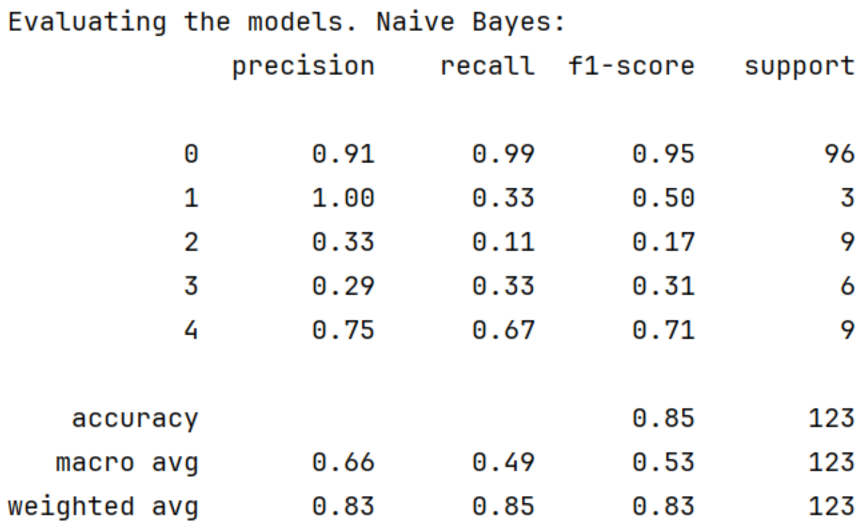


```
Evaluating the models. Linear Regression:
              precision    recall  f1-score   support

         0.0       0.95      0.97      0.96        96
         1.0       0.12      0.67      0.21         3
         2.0       0.29      0.22      0.25         9
         3.0       0.00      0.00      1.00         6
         4.0       1.00      0.00      0.00         9
         6.0       0.00      1.00      0.00         0

    accuracy                           0.79       123
   macro avg       0.39      0.48      0.40       123
weighted avg       0.84      0.79      0.82       123
```

Figure 2-5 Classification report Linear Regression

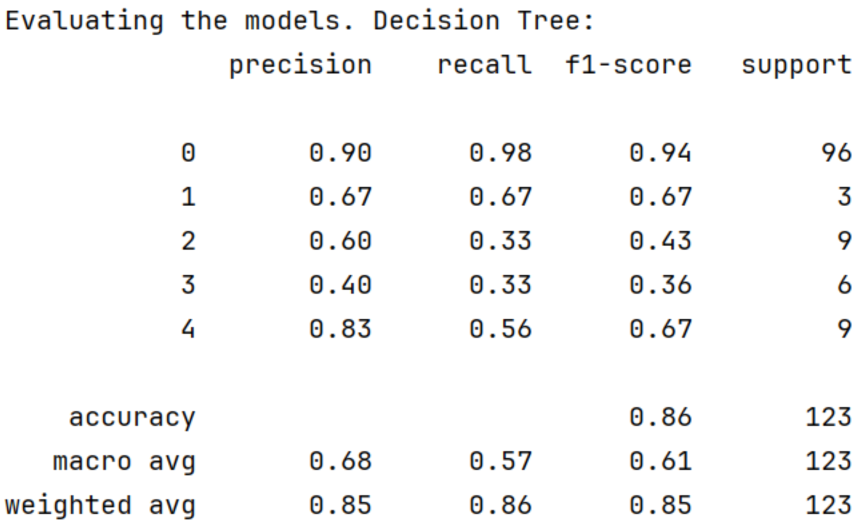


```
Evaluating the models. Naive Bayes:
              precision    recall  f1-score   support

           0       0.91      0.99      0.95        96
           1       1.00      0.33      0.50         3
           2       0.33      0.11      0.17         9
           3       0.29      0.33      0.31         6
           4       0.75      0.67      0.71         9

    accuracy                           0.85       123
   macro avg       0.66      0.49      0.53       123
weighted avg       0.83      0.85      0.83       123
```

Figure 2-6 Classification report Naive Bayes



```
Evaluating the models. Decision Tree:
              precision    recall  f1-score   support

           0       0.90      0.98      0.94        96
           1       0.67      0.67      0.67         3
           2       0.60      0.33      0.43         9
           3       0.40      0.33      0.36         6
           4       0.83      0.56      0.67         9

    accuracy                           0.86       123
   macro avg       0.68      0.57      0.61       123
weighted avg       0.85      0.86      0.85       123
```

Figure 2-7 Classification report Decision Tree

The **Decision Tree** model appears to perform the best overall in terms of accuracy (0.86) and weighted average of f1-score (0.85). This model seems to have a good balance between precision and recall across most classes.

The **k-Nearest Neighbors** model has a high accuracy (0.81) but its performance varies greatly across different classes. It performs well on class 0 but poorly on others.

The **Linear Regression** model has a lower accuracy (0.79) and its performance is also inconsistent across different classes. It performs well on class 0 but poorly on others, especially class 3 and 4.

The **Naive Bayes** model has good accuracy (0.85) and a decent weighted average of f1-score (0.83). However, its performance is not consistent across different classes.

Accuracy:

Accuracy Score kNN: 81.30%

Accuracy Score Linear Regression: 78.86%

Accuracy Score Naive Bayes: 85.37%

Accuracy Score Decision Tree: 86.18%

2.4 Feature selection

Feature selection is the process of automatically selecting relevant features, meaningful data and eliminating data noise for your machine learning model, based on the problem you need to solve. We do this by re-collecting or excluding features that do not affect the output. This helps reduce data noise and reduce the amount of input data. This process is crucial for several reasons:

- Greatly reduce the amount of input data
- By reduce the amount of input data, the learning algorithm can run faster
- Improve predictive accuracy

There are three general methods: filter methods, wrapper methods, and embedded methods.

- Filter methods: these methods select features from the dataset irrespective of the use of any machine learning algorithm. In terms of computation, they are very fast and inexpensive and are very good for removing duplicated, correlated, redundant features but these methods do not remove multicollinearity.

- Wrapper methods: also referred to as greedy algorithms train the algorithm by using a subset of features in an iterative manner. Based on the conclusions made from training in prior to the model, addition and removal of features takes place. Stopping criteria for selecting the best subset are usually pre-defined by the person training the model such as when the performance of the model decreases or a specific number of features has been achieved.
- Embedded methods: the feature selection algorithm is blended as part of the learning algorithm, thus having its own built-in feature selection methods. Embedded methods encounter the drawbacks of filter and wrapper methods and merge their advantages.

CHAPTER 3. OVERFITTING

3.1 What is overfitting?

Overfitting is a fundamental concept in machine learning, and it occurs when a model learns the training data too well. In other words, the model becomes excessively tailored to the specific training data, to the point where it captures not only the underlying patterns and relationships in the data but also the noise, random fluctuations, and outliers. This results in a model that performs exceptionally well on the training data but fails to generalize effectively to new, unseen data.

3.2 Cause of overfitting and solution

3.2.1 Model complexity

Factor: Model complexity refers to the intricacy of the model, including the number of parameters and features it uses to capture relationships in the data. Overly complex models are prone to overfitting.

In-Depth Analysis: Model complexity is associated with the bias-variance trade-off. Complex models can fit the training data with high accuracy, but they may capture noise and fail to generalize to unseen data. Understanding the concept of Occam's razor, which suggests that simpler models are preferred when they perform equally well, is essential.

Prevention Strategy: Use techniques such as Bayesian model selection, which combines prior knowledge and data likelihood to find the most suitable model complexity. Employ regularization methods, like L1 (Lasso) and L2 (Ridge), to constrain model parameters.

3.2.2 Insufficient data

Factor: Insufficient data refers to a small or biased training dataset, which hinders the model's ability to generalize effectively.

In-Depth Analysis: The curse of dimensionality is particularly problematic when dealing with limited data, as models may struggle to discover underlying patterns in high-dimensional spaces. Insufficient data can lead to memorization, where the model merely stores examples without true understanding.

Prevention Strategy: Implement transfer learning by utilizing pre-trained models on large datasets and fine-tuning them for specific tasks. Use data augmentation techniques to generate synthetic examples.

3.2.3 Noisy data

Factor: Noise in training data refers to random variations, errors, or outliers. Models may overfit to this noise, reducing generalization.

In-Depth Analysis: Noise can arise from various sources, such as measurement errors, data entry mistakes, or natural variability. Handling noisy data requires robust modeling and preprocessing techniques to distinguish genuine patterns from anomalies.

Prevention Strategy: Employ robust loss functions like the Huber loss for regression tasks, which are less sensitive to outliers. Apply advanced outlier detection algorithms like Isolation Forests or Mahalanobis distance.

3.2.4 Feature complexity

Factor: Feature complexity involves the richness and intricacy of input features. Complex or irrelevant features can confuse the model, leading to overfitting.

In-Depth Analysis: Feature engineering plays a crucial role in machine learning. Complex features or the inclusion of irrelevant information can lead to poor generalization. Effective feature selection is vital in preventing overfitting.

Prevention Strategy: Utilize feature important measures from tree-based models or apply dimensionality reduction techniques such as t-SNE to identify and retain the most informative features.

3.2.5 Overtraining

Factor: Overtraining occurs when a model is trained for an excessive number of iterations or epochs, resulting in overfitting, particularly in iterative learning algorithms.

In-Depth Analysis: Overtraining reflects the need to balance the model's ability to adapt to the training data and its capacity to generalize to new data. Fine-tuning training processes is crucial to achieving this balance.

Prevention Strategy: Employ advanced optimization techniques like Adam or RMSprop, which adaptively adjust learning rates. Implement curriculum learning to gradually introduce complex examples during training.

3.2.6 Lack of regularization

Factor: Failing to apply regularization can lead to overfitting, especially in complex models with many parameters.

In-Depth Analysis: Regularization is essential to constrain the model's freedom in assigning large weights to features. Without it, models may overfit the training data, even if it means capturing noise.

Prevention Strategy: Combine multiple regularization techniques, such as dropout, weight decay, and batch normalization, to stabilize model training and prevent overfitting.

3.2.7 Validation set

Factor: A separate validation set is crucial for tracking the model's performance during training and detecting overfitting.

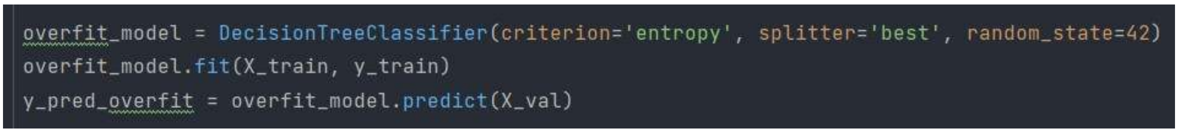
In-Depth Analysis: The validation set offers a reliable assessment of the model's generalization. Monitoring the model's performance on both the training data and the validation data helps identify overfitting.

Prevention Strategy: Utilize advanced validation techniques like k-fold cross-validation or bootstrapping to obtain more robust performance estimates and detect overfitting more effectively.

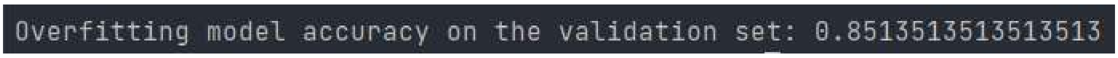
3.3 Example

3.3.1 Cause of overfitting

The 'overfit model' employs the Decision Tree Classifier without any constraints, using the 'criterion' of 'entropy' and 'splitter' set to 'best.' This unrestrained growth allows the Decision Tree to develop intricate decision boundaries, making it highly susceptible to overfitting. Consequently, it exhibits an impressively high accuracy on the training data but performs poorly on the validation data.



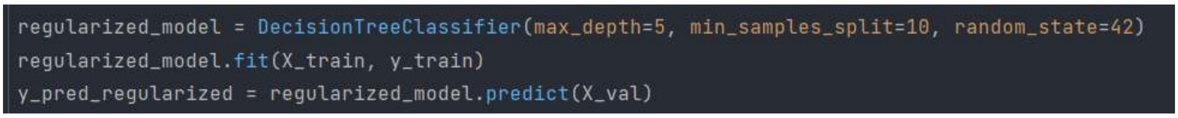
```
overfit_model = DecisionTreeClassifier(criterion='entropy', splitter='best', random_state=42)
overfit_model.fit(X_train, y_train)
y_pred_overfit = overfit_model.predict(X_val)
```



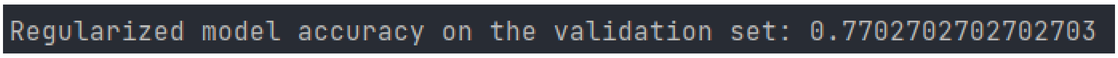
```
Overfitting model accuracy on the validation set: 0.8513513513513513
```

3.3.2 Prevention Strategy

In our 'regularized model' we set specific parameters to guide the Decision Tree's growth. We limit the maximum depth of the tree and mandate that a minimum number of samples are required for further node splitting. This effectively restricts the model's complexity and enforces a balance between fitting the training data and generalizing to unseen data.



```
regularized_model = DecisionTreeClassifier(max_depth=5, min_samples_split=10, random_state=42)
regularized_model.fit(X_train, y_train)
y_pred_regularized = regularized_model.predict(X_val)
```



```
Regularized model accuracy on the validation set: 0.7702702702702703
```

PREFERENCES

Xin Yan, X. S. (2009). Linear Regression Analysis: Theory and Computing. World Scientific, 2009.

Zhang, W. (2017). Machine Learning Approaches to Predicting Company Bankruptcy. Journal of Financial Risk Manager.

Ali, A.M.; Hassan, M.R.; Aburub, F.; Alauthman, M.; Aldweesh, A.; Al-Qerem, A.; Jebreen, I.; Nabot, A. Explainable Machine Learning Approach for Hepatitis C Diagnosis Using SFS Feature Selection. Machines 2023, 11, 391. https://doi.org/10.3390/machines11030391

GeeksforGeeks. (2023, 10 22). Retrieved from https://www.geeksforgeeks.org/feature-selection-techniques-in-machine-learning/