Feature selection using Machine learning

Course No.: CSE-400

Submitted By-

Md.Anas
Student ID: 1505087

Md.Sayeduzzaman
Student ID: 0405035

Submitted To-

Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering.

Supervised by-

Dr. Md. Monirul Islam
Professor
Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology (BUET)
Dhaka-1000, Bangladesh
February , 2021

CERTIFICATION

This is to certify that the work presented in the thesis is an outcome of the investigation carried out by the authors under the supervision of Professor Dr. Md. Monirul Islam, Department of Computer Science and Engineering , BUET.

Authors

Md. Anas

Sayed Uzzaman

Signature of the Supervisor

Dr. Md. Monirul Islam

Professor
Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)

ACKNOWLEDGMENTS

We would like to express our deepest gratitude to our supervisor Dr. Md. Monirul Islam for his guidance on this thesis showing us the path of conducting successful research and above all for always being there as our mentor. He shared his wisdom with us in analyzing subject matters and at the same time valued our thinking approach to synthesize those topics. His suggestions drove us towards better ways of thinking, his reviews enriched us in solving problems, and his support gave us strength at the time of our disappointment. We shall forever cherish the memories of working with him. We deeply thank our friends and families for always believing in us even at the moment when we were losing our confidence.

ABSTRACT

The quality of the data being analyzed is a critical factor that affects the accuracy of data mining algorithms. There are two important aspects of the data quality, one is relevance and the other is data redundancy. The inclusion of irrelevant and redundant features in the data mining model results in poor predictions and high computational overhead. This thesis book presents an efficient method concerning both the relevance of the features and the pairwise features correlation in order to improve the prediction and accuracy of our data mining algorithm. Our approach takes into consideration not only the dependency among the features, but also their dependency with respect to a given data mining task. Our analysis shows that the correlation relationship among features depends on the decision task and, thus, they display different behaviors as we change the decision task. We have applied our method on various popular machine learning datasets and compared them with other papers.

Keywords: Correlation measure, Feature Selection, Machine learning.

TABLE OF CONTENTS

References **42**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

1.1.1 Feature selection

Feature Subset Selection is an essential pre-processing task in Data Mining. Feature selection process refers to choosing subset of attributes from the set of original attributes. This technique attempts to identify and remove as much irrelevant and redundant information as possible It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model. Statistical-based feature selection methods involve evaluating the relationship between each input variable and the target variable using statistics and selecting those input variables that have the strongest relationship with the target variable. These methods can be fast and effective, although the choice of statistical measures depends on the data type of both the input and output variables. As such, it can be challenging for a machine learning practitioner to select an appropriate statistical measure for a dataset when performing filter-based feature selection.
Feature selection is also related to dimensionally reduction techniques in that both methods seek fewer input variables to a predictive model. The difference is that feature selection select features to keep or remove from the dataset, whereas dimensionality reduction create a projection of the data resulting in entirely new input features. As such, dimensionality reduction is an alternate to feature selection rather than a type of feature selection.
We can summarize feature selection as follows.

- **Feature Selection:** Select a subset of input features from the dataset.
  - **Unsupervised:** Do not use the target variable (e.g. remove redundant variables)
    - Correlation
  - **Supervised:** Use the target variable (e.g. remove irrelevant variables)
    - **Wrapper:** Search for well-performing subsets of features.
    - **Filter:** Select subsets of features based on their relationship with the target.
    - **Intrinsic:** Algorithms that perform automatic feature selection during training.
- **Dimensionality Reduction:** Project input data into a lower-dimensional feature space

Feature Selection with Statistical Measures

We can use correlation type statistical measures between input and output variables, which can then be used as the basis for filter feature selection.The choice of statistical measures highly depends upon the variable data types. Common variable data types include:

- Numerical such as height
- Categorical such as a label

Both of the variable data types are subdivided into many categories, which are as under: Numerical variables are divided into the following:

- Integer Variables
- Float Variables

On the other hand, categorical variables are divided into the following:

- Boolean Variables
- Nominal Variables
- Ordinal Variables

Univariate Feature Selection

In feature-based filter selection, the statistical measures are calculated considering only a single input variable at a time with a target (output) variable. These statistical measures are termed as univariate statistical measures, which means that the interaction between input variables is not considered in the filtering process. Univariate feature selection selects the best features on the basis of univariate statistical tests. We compare each feature to the target variable in order to determine the significant statistical relationship between them. Univariate feature selection is also called analysis of variance (ANOVA). The majority of the techniques are univariate means that they perform the predictor evaluation in isolation. The existence of the correlated predictors increases the possibility of selecting significant but redundant predictors. Consequently, a large number of predictors are chosen, which results in the rise of collinearity problems. In univariate feature selection methods, we examine each feature individually to determine the features‚Äô relationship with the response variable.

1.2 Related works

1.2.1 Dependency and Correlation Analysis for Features

The quality of the data being analyzed is a critical factor that affects the accuracy of data mining algorithms. There are two important aspects of the data quality, one is relevance and the other is data redundancy. The inclusion of irrelevant and redundant features in the data mining model results in poor predictions and high computational overhead. Paper in [1] presents an method concerning both the relevance of the features and the pairwise features correlation in order to improve the prediction and accuracy of data mining algorithm. It introduced a new feature correlation metric $Q_Y(X_i, X_j)$ and feature subset merit measure $e(S)$ to quantify the relevance and the correlation among features with respect to a desired data mining task.They applied their data mining approach to network security and validated it using the DARPA KDD99 benchmark data set. Their results showed that, using the new decision dependent correlation metric, they could efficiently detect rare network attacks such as User to Root (U2R) and Remote to Local (R2L) attacks. The best reported detection rates for U2R and R2L on the KDD99 data sets were 13.2 percent and 8.4 percent with 0.5 percent false alarm, respectively. For U2R attacks, their approach can achieve a 92.5 percent detection rate with a false alarm of 0.7587 percent. For R2L attacks,their approach can achieve a 92.47 percent detection rate with a false alarm of 8.35 percent.

1.2.2 Conditional mutual information approach

Thu Zar proposed a new feature subset selection algorithm based on conditional mutual information approach [2].First calculate $I(C;X_i)$ for all $X_i \in X.Then, the i^{th}$ feature that has maximum $I(C;X_i)$ is chosen as first relevant feature as it provide highest class information among other features. In the next steps, repeat feature selection process until the feature set X becomes empty. Select and remove feature one by one by using the proposed criteria. Information Theoretical concepts is used to implement the effective feature selection algorithm and then implement the entropy, joint entropy and Mutual Information to produce the first most relevant feature to the class. After that, Conditional Mutual Information is used to reduce redundancy and to produce the other most effective features to the class. First calculate $I(C;X_i)$ for all $X_i \in X.Then, the i^{th}$ feature that has maximum $I(C;X_i)$ is chosen as first relevant feature as it provide highest class information among other features. In the next steps, repeat feature selection process until the feature set X becomes empty. Select and remove feature one by one by using the proposed criteria. Problem with their approach is correlated features may exist in final selected feature subset hence introduce redundancy.

1.2.3 Feature selection based on Bayesian Network

Fengli Zhang [3] proposed a feature selection approach based on Bayesian Network classifier for network-based intrusion detection system (IDS).With the intrusion detection benchmark dataset (NSL-KDD), the performance of the proposed approach is evaluated and compared with other commonly used feature selection methods. It is shown by empirical results that features selected by our approach have decreased the time to detect attacks and increased the classification accuracy as well as the true positive rates significantly

1.2.4 Features Selection using Fuzzy ESVDF

Safaa and Fakhri [4] introduced an algorithm for features selection based on a Support Vector Decision Function (SVDF) and Forward Selection (FS) approach with a fuzzy inferencing model. In the new algorithm, Fuzzy Enhancing Support Vector Decision Function (Fuzzy ESVDF), features are selected stepwise, one at a time, by using SVDF to evaluate the weight value of each specified candidate feature, then applying FS with the fuzzy inferencing model to rank the feature according to a set of rules based on a comparison of performance. Using a fast and simple approach, the Fuzzy ESVDF algorithm produces an efficient features set and, thus, provides an effective solution to the dimensionality reduction problem in general. they had examined the feasibility of their approach by conducting several experiments using five different datasets. The experimental results indicate that the proposed algorithm can deliver a satisfactory performance in terms of classification accuracy, False Positive Rate (FPR), training time, and testing time.

1.2.5 Feature selection based on Symmetric Uncertainty and Ant Colony Optimization

Imran and Waseem [5] proposed a feature selection method based on symmetric uncertainty and Ant colony optimization. The proposed method is a pure filter based feature subset selection technique that incurs less computational cost and highly efficient in terms of classification accuracy. Moreover, along with high accuracy the proposed method requires less number of features in most of the cases. In the proposed method the issue of feature ranking and threshold value selection is addressed. The proposed method adaptively selects number of features as per the worth of an individual feature in the dataset. An extensive experimentation is performed, comprised of a number of benchmark datasets over three well known classification algorithms.

1.2.6 Other related works

H. Liu et al. proposed a consistency based feature selection mechanism. It evaluates the worth of a subset of attributes by the level of consistency in the class values when the training instances are projected onto the subset of attributes. Consistency of any subset can never be lower than that of the full set of attributes; hence the usual practice is to use this subset evaluator in conjunction with a Random or Exhaustive search which looks for the smallest subset with consistency equal to that of the full set of attributes [6].
M. Hall presented a new correlation based approach to feature selection (CFS) in different datasets and demonstrated how it can be applied to both classification and regression problems for machine

learning [7].
Anirut Suebsing, Nualsawat Hiransakolwong proposed Euclidean distance measure to use as score in feature selection for KDD dataset. High score features that is greater than defined threshold value were selected as best feature subsets [8].
Zahra Karimi, Mohammas Mansour, presented a hybrid feature selection methods by combining symmetric uncertainty measure and gain measure. Both SU and gain measures for each feature-class correlation were calculated first and then rank feature according to average score value. High ranked feature greater than a threshold values was selected. They evaluated their system using KDD dataset and Naive Bayes algorithm [9].
A. Chaudhary, et. al. presented the performance evaluation of three feature selection methods with optimized Naive Bayes is performed on mobile device. Correlation based method, Gain Ratio method and Information Gain method methods were used in this work [10].

1.3 Motivation

Data quality affects the accuracy of data mining algorithms. Proper feature set can improve accuracy significantly.
Two important aspects:

- data relevance
- data redundancy

Concerns

- Redundant , irrelevant features affect accuracy of learning algorithms
- Improvement of accuracy of data mining algorithm
- Pairwise features correlation
- Relevance of features
- Allow algorithms to operate faster

1.4 Objectives

The objective of our thesis is to find the best possible feature subset from a given dataset that increase accuracy and decrease execution time. We need to find most important features and remove redundant features as much as possible. We want our final feature subset size as small as possible so that execution time decreases. We have studied about correlation between features and decision variable and correlation between features themselves. We have proposed a hybrid method that resolves most important concerns of feature selection namely feature relevance and feature redundancy. We have applied our method on various datasets and compared them with other papers.

1.5 Thesis organization

In this thesis book we have introduced our work on feature selection using machine learning. We have discussed various feature selection methods briefly. Then we have discussed about related past works on similar topic. In related works section we have included some great papers we have found on feature selection. We have discussed methodology they used and how well they did. We also discussed about our background where we talked about machine learning and various concepts we have needed for our thesis work. We have talked about classifier and various classification algorithms.
In chapter 3 we have discussed our proposed method. We have discussed each steps broadly with the theoretical concepts we have used in our proposed method. Then we have showed our method using flowchart and finally we have provided pseudocode of our proposed algorithm.
We have applied our method on various datasets and in chapter 4 we have shown the result and analysis. Finally we have discussed about our future work in conclusion chapter.

CHAPTER 2

BACKGROUND

2.1 Machine Learning

Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data such algorithms overcome following strictly static program instructions by making data driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms is infeasible; example applications include spam filtering, detection of network intruders or malicious insiders working towards a data breach, optical character recognition (OCR), search engines and computer vision. Machine learning tasks are typically classified into three broad categories, depending on the nature of the learning ‚Äúsignal‚Äù or ‚Äúfeedback‚Äù available to a learning system. These are

Supervised learning
Supervised learning describes a class of problem that involves using a model to learn a mapping between input examples and the target variable. Applications in which the training data comprises examples of the input vectors along with their corresponding target vectors are known as supervised learning problems. Models are fit on training data comprised of inputs and outputs and used to make predictions on test sets where only the inputs are provided and the outputs from the model are compared to the withheld target variables and used to estimate the skill of the model. Learning is a search through the space of possible hypotheses for one that will perform well, even on new examples beyond the training set. To measure the accuracy of a hypothesis we give it a test set of examples that are distinct from the training set There are two main types of supervised learning problems: they are classification that involves predicting a class label and regression that involves predicting a numerical value.

- **Classification:** Supervised learning problem that involves predicting a class label.
- **Regression:** Supervised learning problem that involves predicting a numerical label.

Both classification and regression problems may have one or more input variables and input variables may be any data type, such as numerical or categorical.

An example of a classification problem would be the MNIST handwritten digits dataset where the inputs are images of handwritten digits (pixel data) and the output is a class label for what digit the image represents (numbers 0 to 9).

An example of a regression problem would be the Boston house prices dataset where the inputs are variables that describe a neighborhood and the output is a house price in dollars.

Some machine learning algorithms are described as ‚Äús upervised‚Äù machine learning algorithms as they are designed for supervised machine learning problems. Popular examples include: decision trees, support vector machines, and many more. Our goal is to find a useful approximation $f(x)$ to the function $f(x)$ that underlies the predictive relationship between the inputs and outputs. Algorithms are referred to as ‚Äús upervised‚Äù because they learn by making predictions given examples of input data, and the models are supervised and corrected via an algorithm to better predict the expected target outputs in the training dataset.

The term supervised learning originates from the view of the target y being provided by an instructor or teacher who shows the machine learning system what to do.Some algorithms may be specifically designed for classification (such as logistic regression) or regression (such as linear regression) and some may be used for both types of problems with minor modifications (such as artificial neural networks).

Unsupervised learning

Unsupervised learning describes a class of problems that involves using a model to describe or extract relationships in data.

Compared to supervised learning, unsupervised learning operates upon only the input data without outputs or target variables. As such, unsupervised learning does not have a teacher correcting the model, as in the case of supervised learning.

In unsupervised learning, there is no instructor or teacher, and the algorithm must learn to make sense of the data without this guide. There are many types of unsupervised learning, although there are two main problems that are often encountered by a practitioner: they are clustering that involves finding groups in the data and density estimation that involves summarizing the distribution of data.

- **Clustering:** Unsupervised learning problem that involves finding groups in data.
- **Density Estimation:** Unsupervised learning problem that involves summarizing the distribution of data.

An example of a clustering algorithm is k-Means where k refers to the number of clusters to discover in the data. An example of a density estimation algorithm is Kernel Density Estimation that involves using small groups of closely related data samples to estimate the distribution for new points in the problem space. The most common unsupervised learning task is clustering: detecting potentially useful clusters of input examples. For example, a taxi agent might gradually develop a concept of ‚Äúgood traffic days‚Äù and ‚Äúbad traffic days‚Äù without ever being given labeled examples of each by a teacher.Clustering and density estimation may be performed to learn about the patterns in the data.

Additional unsupervised methods may also be used, such as visualization that involves graphing or plotting data in different ways and projection methods that involves reducing the dimensionality of the data.

- **Visualization:** Unsupervised learning problem that involves creating plots of data.
- **Projection:** Unsupervised learning problem that involves creating lower-dimensional representations of data.

An example of a visualization technique would be a scatter plot matrix that creates one scatter plot of each pair of variables in the dataset. An example of a projection method would be Principal Component Analysis that involves summarizing a dataset in terms of eigenvalues and eigenvectors, with linear dependencies removed.

The goal in such unsupervised learning problems may be to discover groups of similar examples within the data, where it is called clustering, or to determine the distribution of data within the input space, known as density estimation, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

Reinforcement Learning

Reinforcement learning describes a class of problems where an agent operates in an environment and must learn to operate using feedback.

Reinforcement learning is learning what to do ,Äî how to map situations to actions,Äîso as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them.The use of an environment means that there is no fixed training dataset, rather a goal or set of goals that an agent is required to achieve, actions they may perform, and feedback about performance toward the goal.

Some machine learning algorithms do not just experience a fixed dataset. For example, reinforcement learning algorithms interact with an environment, so there is a feedback loop between the learning system and its experiences.It is similar to supervised learning in that the model has some response from which to learn, although the feedback may be delayed and statistically noisy, making it challenging for the agent or model to connect cause and effect.

An example of a reinforcement problem is playing a game where the agent has the goal of getting a high score and can make moves in the game and received feedback in terms of punishments or rewards.In many complex domains, reinforcement learning is the only feasible way to train a program to perform at high levels. For example, in game playing, it is very hard for a human to provide accurate and consistent evaluations of large numbers of positions, which would be needed to train an evaluation function directly from examples. Instead, the program can be told when it has won or lost, and it can use this information to learn an evaluation function that gives reasonably accurate estimates of the probability of winning from any given position. Impressive recent results include the use of reinforcement in Google,Äôs AlphaGo in out-performing the world,Äôs top Go player. Some popular examples of reinforcement learning algorithms include Q-learning, temporal-difference learning, and deep reinforcement learning.

Semi-Supervised Learning

Semi-supervised learning is supervised learning where the training data contains very few labeled

examples and a large number of unlabeled examples. The goal of a semi-supervised learning model is to make effective use of all of the available data, not just the labelled data like in supervised learning. In semi-supervised learning we are given a few labeled examples and must make what we can of a large collection of unlabeled examples. Even the labels themselves may not be the oracular truths that we hope for. Making effective use of unlabelled data may require the use of or inspiration from unsupervised methods such as clustering and density estimation. Once groups or patterns are discovered, supervised methods or ideas from supervised learning may be used to label the unlabeled examples or apply labels to unlabeled representations later used for prediction.

Unsupervised learning can provide useful cues for how to group examples in representation space. Examples that cluster tightly in the input space should be mapped to similar representations. It is common for many real-world supervised learning problems to be examples of semi-supervised learning problems given the expense or computational cost for labeling examples. For example, classifying photographs requires a dataset of photographs that have already been labeled by human operators.

Many problems from the fields of computer vision (image data), natural language processing (text data), and automatic speech recognition (audio data) fall into this category and cannot be easily addressed using standard supervised learning methods.

Self-Supervised Learning

Self-supervised learning refers to an unsupervised learning problem that is framed as a supervised learning problem in order to apply supervised learning algorithms to solve it.

Supervised learning algorithms are used to solve an alternate or pretext task, the result of which is a model or representation that can be used in the solution of the original (actual) modeling problem.

The self-supervised learning framework requires only unlabeled data in order to formulate a pretext learning task such as predicting context or image rotation, for which a target objective can be computed without supervision.A common example of self-supervised learning is computer vision where a corpus of unlabeled images is available and can be used to train a supervised model, such as making images grayscale and having a model predict a color representation (colorization) or removing blocks of the image and have a model predict the missing parts (inpainting).

In discriminative self-supervised learning, which is the main focus of this work, a model is trained on an auxiliary or ‚Äòpretext‚Äô task for which ground-truth is available for free. In most cases, the pretext task involves predicting some hidden portion of the data (for example, predicting color for gray-scale images. A general example of self-supervised learning algorithms are autoencoders. These are a type of neural network that is used to create a compact or compressed representation of an input sample. They achieve this via a model that has an encoder and a decoder element separated by a bottleneck that represents the internal compact representation of the input.

An autoencoder is a neural network that is trained to attempt to copy its input to its output. Internally, it has a hidden layer h that describes a code used to represent the input. These autoencoder models are trained by providing the input to the model as both input and the target output, requiring that the model reproduce the input by first encoding it to a compressed

representation then decoding it back to the original. Once trained, the decoder is discarded and the encoder is used as needed to create compact representations of input.

Although autoencoders are trained using a supervised learning method, they solve an unsupervised learning problem, namely, they are a type of projection method for reducing the dimensionality of input data.

Traditionally, autoencoders were used for dimensionality reduction or feature learning. Another example of self-supervised learning is generative adversarial networks, or GANs. These are generative models that are most commonly used for creating synthetic photographs using only a collection of unlabeled examples from the target domain.

GAN models are trained indirectly via a separate discriminator model that classifies examples of photos from the domain as real or fake (generated), the result of which is fed back to update the GAN model and encourage it to generate more realistic photos on the next iteration.

Multi-Instance Learning

Multi-instance learning is a supervised learning problem where individual examples are unlabeled; instead, bags or groups of samples are labeled.

In multi-instance learning, an entire collection of examples is labeled as containing or not containing an example of a class, but the individual members of the collection are not labeled.Instances are in bags rather than sets because a given instance may be present one or more times, e.g. duplicates.

Modeling involves using knowledge that one or some of the instances in a bag are associated with a target label, and to predict the label for new bags in the future given their composition of multiple unlabeled examples.

In supervised multi-instance learning, a class label is associated with each bag, and the goal of learning is to determine how the class can be inferred from the instances that make up the bag. Simple methods, such as assigning class labels to individual instances and using standard supervised learning algorithms, often work as a good first step.

Inductive Learning

Inductive learning involves using evidence to determine the outcome. Inductive reasoning refers to using specific cases to determine general outcomes, e.g. specific to general.Most machine learning models learn using a type of inductive inference or inductive reasoning where general rules (the model) are learned from specific historical examples (the data).Fitting a machine learning model is a process of induction. The model is a generalization of the specific examples in the training dataset.

A model or hypothesis is made about the problem using the training data, and it is believed to hold over new unseen data later when the model is used.

Deductive Inference

Deduction or deductive inference refers to using general rules to determine specific outcomes.We can better understand induction by contrasting it with deduction. Deduction is the reverse of induction. If induction is going from the specific to the general, deduction is going from the general to the specific. Deduction is a top-down type of reasoning that seeks for all premises to be met before determining the conclusion, whereas induction is a bottom-up type of reasoning that uses available data as evidence for an outcome.

In the context of machine learning, once we use induction to fit a model on a training dataset, the model can be used to make predictions. The use of the model is a type of deduction or deductive inference.

Transductive Learning

Transduction or transductive learning is used in the field of statistical learning theory to refer to predicting specific examples given specific examples from a domain.It is different from induction that involves learning general rules from specific examples, e.g. specific to specific.Induction, deriving the function from the given data. Deduction, deriving the values of the given function for points of interest. Transduction, deriving the values of the unknown function for points of interest from the given data.Unlike induction, no generalization is required; instead, specific examples are used directly. This may, in fact, be a simpler problem than induction to solve.

The model of estimating the value of a function at a given point of interest describes a new concept of inference: moving from the particular to the particular. We call this type of inference transductive inference. Note that this concept of inference appears when one would like to get the best result from a restricted amount of information.

A classical example of a transductive algorithm is the k-Nearest Neighbors algorithm that does not model the training data, but instead uses it directly each time a prediction is required.

Multi-Task Learning

Multi-task learning is a type of supervised learning that involves fitting a model on one dataset that addresses multiple related problems. It involves devising a model that can be trained on multiple related tasks in such a way that the performance of the model is improved by training across the tasks as compared to being trained on any single task. Multi-task learning is a way to improve generalization by pooling the examples (which can be seen as soft constraints imposed on the parameters) arising out of several tasks. Multi-task learning can be a useful approach to problem-solving when there is an abundance of input data labeled for one task that can be shared with another task with much less labeled data. For example, it is common for a multi-task learning problem to involve the same input patterns that may be used for multiple different outputs or supervised learning problems. In this setup, each output may be predicted by a different part of the model, allowing the core of the model to generalize across each task for the same inputs.

A popular example of multi-task learning is where the same word embedding is used to learn a distributed representation of words in text that is then shared across multiple different natural language processing supervised learning tasks.

Active Learning

Active learning is a technique where the model is able to query a human user operator during the learning process in order to resolve ambiguity during the learning process. Active learning: The learner adaptively or interactively collects training examples, typically by querying an oracle to request labels for new points. Active learning is a type of supervised learning and seeks to achieve the same or better performance of so-called ‚Äúpassive‚Äù supervised learning, although by being more efficient about what data is collected or used by the model.The key idea behind active learning is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns. An active learner may pose queries, usually in the form of unlabeled data instances to be labeled by an oracle (e.g., a human annotator).It is not unreasonable to view active learning as an approach to solving semi-supervised learning problems, or an alternative paradigm for the same types of problems.Active learning is a useful approach when there is not much data available and new data is expensive to collect or label.

The active learning process allows the sampling of the domain to be directed in a way that minimizes the number of samples and maximizes the effectiveness of the model.

Online Learning

Online learning involves using the data available and updating the model directly before a prediction is required or after the last observation was made. Online learning is appropriate for those problems where observations are provided over time and where the probability distribution of observations is expected to also change over time. Therefore, the model is expected to change just as frequently in order to capture and harness those changes. This approach is also used by algorithms where there may be more observations than can reasonably fit into memory, therefore, learning is performed incrementally over observations, such as a stream of data. Online learning is helpful when the data may be changing rapidly over time. It is also useful for applications that involve a large collection of data that is constantly growing, even if changes are gradual.Generally, online learning seeks to minimize regret, which is how well the model performed compared to how well it might have performed if all the available information was available as a batch.
One example of online learning is so-called stochastic or online gradient descent used to fit an artificial neural network.
The fact that stochastic gradient descent minimizes generalization error is easiest to see in the online learning case, where examples or minibatches are drawn from a stream of data.

Transfer Learning

Transfer learning is a type of learning where a model is first trained on one task, then some or all of the model is used as the starting point for a related task. In transfer learning, the learner must perform two or more different tasks, but we assume that many of the factors that explain the variations in P1 are relevant to the variations that need to be captured for learning P2.It is a useful approach on problems where there is a task related to the main task of interest and the related task has a large amount of data.

It is different from multi-task learning as the tasks are learned sequentially in transfer learning, whereas multi-task learning seeks good performance on all considered tasks by a single model at the same time in parallel.An example is image classification, where a predictive model, such as an artificial neural network, can be trained on a large corpus of general images, and the weights of the model can be used as a starting point when training on a smaller more specific dataset, such as dogs and cats. The features already learned by the model on the broader task, such as extracting lines and patterns, will be helpful on the new related task.If there is significantly more data in the first setting (sampled from P1), then that may help to learn representations that are useful to quickly generalize from only very few examples drawn from P2. Many visual categories share low-level notions of edges and visual shapes, the effects of geometric changes, changes in lighting, etc.As noted, transfer learning is particularly useful with models that are incrementally trained and an existing model can be used as a starting point for continued training, such as deep learning networks.

Ensemble Learning

Ensemble learning is an approach where two or more modes are fit on the same data and the predictions from each model are combined. The field of ensemble learning provides many ways of combining the ensemble members,Äô predictions, including uniform weighting and weights chosen on a validation set. The objective of ensemble learning is to achieve better performance with the ensemble of models as compared to any individual model. This involves both deciding how to create models used in the ensemble and how to best combine the predictions from the ensemble members. Ensemble learning can be broken down into two tasks: developing a population of base learners from the training data, and then combining them to form the composite predictor. Ensemble learning is a useful approach for improving the predictive skill on a problem domain and to reduce the variance of stochastic learning algorithms, such as artificial neural networks.

2.2 Concepts

2.2.1 Feature Selection

In many supervised learning tasks the input is represented by a very large number of features, many of which are not needed for predicting the labels. Feature selection (variously known as subset selection, attribute selection or variable selection) is the task of choosing a small subset of features that is sufficient to predict the target labels well. The four main reasons to use feature selection are:

- **Reduced computational complexity:** Feature selection reduces the computational complexity of learning and prediction algorithms. Many popular learning algorithms become computationally intractable in the presence of huge numbers of features, both in the training step and in the prediction step. A preceding step of feature selection can solve the problem.
- **Economy:**Feature selection saves on the cost of measuring non selected features. Once we have found a small set of features that allows good prediction of the labels, we do not have to measure the rest of the features any more. Thus, in the prediction stage we only have to measure a few features for each instance. Imagine that we want to predict whether a patient has a specic disease using the results of medical checks. There are a huge number of possible medical checks that might be predictive; let's say that there are 1000 potential checks and that each of them costs ten dollars to perform. If we can nd a subset of only 10 features that allows good performance, it saves a lot of money, and may turn the whole thing from an infeasible into a feasible procedure
- **Improved accuracy**In many situations, feature selection can also enhance prediction accuracy by improving the signal to noise ratio. Even state-of-the-art learning algorithms cannot overcome the presence of a huge number of irrelevant or weakly relevant features. On the other hand once a small set of good features has been found, even very simple learning algorithms may yield good performance. Thus, in such situations, an initial step of feature selection may improve accuracy dramatically
- **Problem understanding**Another benet of feature selection is that the identity of the selected features can provide insights into the nature of the problem at hand. This is signicant since in many cases the ability to point out the most informative features is more important than the ability to make a good prediction in itself. Imagine we are trying to predict whether a person has a specic type of cancer using gene expression data. While we can know whether the individual is sick or not in other ways, the identity of the genes which are informative for prediction may give us a clue to the disease mechanism involved, and help in developing drugs

Thus feature selection is an important step in efficient learning of large multi-featured data sets. Regarding reasons 2 and 4 above, feature selection has an advantage over other general dimensionality reduction methods. Computing general functions of all the input features means that we must always measure all the features first, even if in the end we calculate only a few functions of them. In many problems, different features vary in terms of their nature and their

units (e.g. body temperature in Celsius, yearly income in dollars). In these cases feature selection is the natural formulation and it enables a better interpretation of the results, as the meaning of the combination of the features is not very clear.
Many works dene the task of feature selection is detecting which features are relevant and which are irrelevant. In this context we need to dene relevancy. This is not straightforward, because the eect of a feature depends on which other features we have selected. Almuallim and Dietterich [11] provide a denition of relevance for the binary noise-free case. They dene a feature to be relevant if it appears in any logical formula that describes the target concept. Gennari et al. [12] suggest a probabilistic definition of relevancy, which defines a feature to be relevant if it affects the conditional distribution of the labels. John et al [13] define the notion of strong and weak relevance of a feature. Roughly speaking, a strongly relevant feature is a useful feature that cannot be replaced by any other feature (or set of features) whereas a weakly relevant feature is a feature which is useful, but can be replaced by another feature (or set of features). They also show that, in general, relevance does not imply optimality and that optimality does not imply relevance.

2.2.2 Paradigms for feature selection

Many different algorithms for the task of feature selection have been suggested over the last few decades both in the Statistics and in the Learning community. Different algorithms present different conceptual frameworks. However, in the most common selection paradigm an evaluation function is used to assign scores to subsets of features and a search algorithm is used to search for a subset with a high score. Different selection methods can be different both in the choice of evaluation function and the search method. The evaluation function can be based on the performance of a specific predictor (wrapper model, [14]) or on some general (typically cheaper to compute) relevance measure of the features to the prediction (filter model, [14]). The evaluation function is not necessarily a black box and in many cases the search method can use information on the evaluation function in order to perform an efficient search. In most common wrappers, the quality of a given set of features is evaluated by testing the predictor performance on a validation set. The main advantage of a wrapper is that you optimize what really interest you - the predictor accuracy. The main drawback of such methods is their computational deficiency that limits the number of sets that can be evaluated. The computational complexity of wrappers is high since we have to re-train the predictor in each step. Common filters use quantities like conditional Variance (of the features given the labels), Correlation Coefficients or Mutual Information as a general measure. In any case (wrapper or filter), an exhaustive search over all feature sets is generally intractable due to the exponentially large number of possible sets. Therefore, search methods are employed which apply a variety of heuristics. Two classic search methods are [15]:

- **Forward selection:**start with an empty set of features and greedily add features one at a time. In each step, the feature that produces the larger increase of the evaluation function (with respect to the value of the current set) is added
- **Backward Elimination:** Start with a set of features that contains all the features and greedily remove features one at a time. In each step the feature whose removal results in the larger increase (or smaller decrease) in the evaluation function value is removed

Backward elimination has the advantage that when it evaluates the contribution of a feature it takes into consideration all the other potential features. On the other hand, in forward selection, a feature that was added at one point can become useless later on and vice versa. However, since evaluating small sets of features is usually faster than evaluating large sets, forward selection is much faster when we are looking for small number of features to select. Moreover, if the initial number of features is very large, backward elimination become infeasible. A combination of the two is also possible of course, and has been used in many works. Many other search methods such as stochastic hill climbing, random permutation [16] and genetic algorithms [17] can be used as search methods as well.

Individual feature ranking

Other feature selection methods simply rank individual features by assigning a score to each feature independently. These methods are usually very fast, but inevitably fail in situations where only a combined set of features is predictive of the target function. Two of the most common rankers are:

- **Infogain** [18], which assigns to each feature the Mutual Information between the feature value and the label, considering both the feature and the label as random variables and using any method to estimate the joint probability. Formally, let P be an estimation of the joint probability for a feature, i.e., Pij represents the probability that both f equal its i's possible value and the label is j, then we get the following formula for the infogain of the feature f:

  (2.1) $$IG(f) = \sum_{i,j} P_{ij} \log \frac{P_{ij}}{\sum_i P_{ij} \sum_j P_{ij}}$$

  The most common way to estimate P is by simply counting the percent of the instances that present each value pair (the empiric distribution) with some kind of zero correction (e.g., adding a small constant to each value, and re-normalizing)

- **Correlation Coefficients**, which assign each feature the Pearson correlation between the vector of values the feature got in the training set (v) and the vector of the labels (y) [19] i.e

  (2.2) $$CC(f) = \frac{E(v - E(y)(y - E(y)))}{Var(y)\sqrt{Var(v)}}$$

  where the expectation and the variance are estimated empirically of course.

Corrcoef has the advantage (over infogain) that it can be used with continuous values (of features or the label) without any need of quantization. On the other hand, Infogain has the advantage that it does not assume any geometric meaning of the values, and thus can work for any kind of features and label, even if they are not numerical values.

Filter methods

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here. For basic guidance, you can refer to the following table for defining correlation co-efficients.

Feature	Continuous	Categorical reduction
Continuous	Pearson's correlation	LDA
Categorical	Anova	Chi-Square

1. **Pearson Correlation:** It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1.
2. **LDA:** Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.
3. **ANOVA:** ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.
4. **Chi-Square:** It is a is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

Wrapper Methods

In wrapper methods, we try to use a subset of features and train a model using them. Based on the inferences that we draw from the previous model, we decide to add or remove features from your subset. The problem is essentially reduced to a search problem. These methods are usually computationally very expensive.

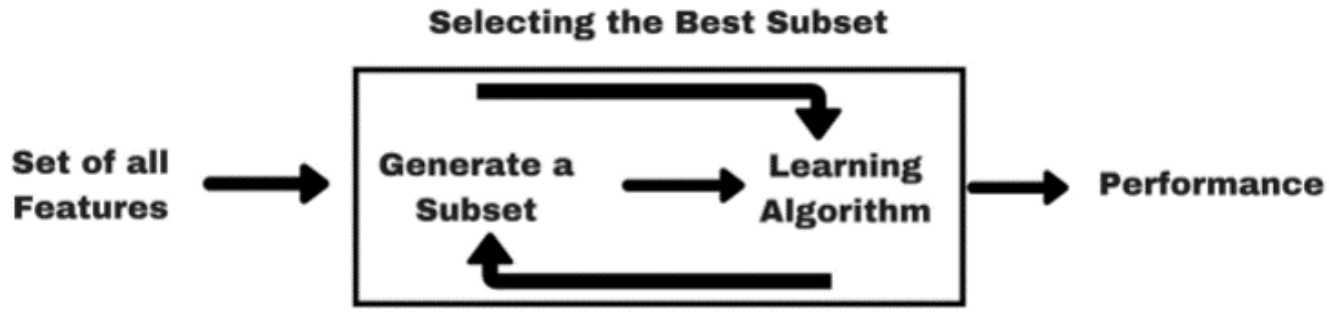

Figure 2.1: Wrapper method

Some common examples of wrapper methods are forward feature selection, backward feature elimination, recursive feature elimination, etc.

- **Forward Selection:** Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.

- **Backward Elimination:** In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.

- **Recursive Feature elimination:** It is a greedy optimization algorithm which aims to find the best performing feature subset. It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.

One of the best ways for implementing feature selection with wrapper methods is to use Boruta package that finds the importance of a feature by creating shadow features.
It works in the following steps:

- Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).

- Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.

- At every iteration, it checks whether a real feature has a higher importance than the best of its shadow features (i.e. whether the feature has a higher Z-score than the maximum Z-score of its shadow features) and constantly removes features which are deemed highly unimportant.

- Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specified limit of random forest runs.

Embedded methods

The term Embedded methods is usually used to describe selection which is done automatically by the learning algorithm. Note that almost any wrapper can be considered as an embedded method, because selection and learning are interleaved. Decision tree learning can also be considered to be an embedded method, as the construction of the tree and the selection of the features are interleaved, but the selection of the feature in each step is usually done by a simple filter or ranker. Other authors use the term embedded method to refer only to methods where the selection is done by the learning algorithm implicitly. A prominent example of such a method is the L1-SVM [20] Embedded methods combine the qualities‚Äô of filter and wrapper methods. It‚Äôs implemented by algorithms that have their own built-in feature selection methods.
Some of the most popular examples of these methods are LASSO and RIDGE regression which

have inbuilt penalization functions to reduce overfitting.

- Lasso regression performs L1 regularization which adds penalty equivalent to absolute value of the magnitude of coefficients.
- Ridge regression performs L2 regularization which adds penalty equivalent to square of the magnitude of coefficients.

Filter vs Wrapper

The main differences between the filter and wrapper methods for feature selection are:

- Filter methods measure the relevance of features by their correlation with dependent variable while wrapper methods measure the usefulness of a subset of feature by actually training a model on it.
- Filter methods are much faster compared to wrapper methods as they do not involve training the models. On the other hand, wrapper methods are computationally very expensive as well.
- Filter methods use statistical methods for evaluation of a subset of features while wrapper methods use cross validation.
- Filter methods might fail to find the best subset of features in many occasions but wrapper methods can always provide the best subset of features.
- Using the subset of features from the wrapper methods make the model more prone to overfitting as compared to using subset of features from the filter methods.

ReliefF and K-means methods

ReliefF method is another significant method to select feature by ranking features according to a criterion. Before ReliefF method, there is another method named relief. The relief method can only process the data set which only has two class labels. However, ReliefF can process the data set of multiple class labels. K-means clustering is a method of vector quantization, with the purpose of partitioning samples into clusters. There are three criteria to determine whether the clustering is good or bad: homogeneity, completeness and V-measure. Firstly, homogeneity means that a cluster contains only one category of samples. Secondly, completeness means that similar samples are categorized into the same cluster. Thirdly, V-measure is theweighted average of homogeneity and completeness. In this paper, we combine thesetwo formulas to get V-measure by giving their different weights.

Symmetrical uncertainty

Symmetric Uncertainty is one of the best feature selection methods and most feature selection system based on mutual information use this measure. SU is a correlation measure between the features and the class.

$$SU = \frac{H(X) + H(Y) - H(X)}{H(X) + H(Y)} \tag{2.3}$$

where H(X) and H(Y) are the entropies based on the probability associated with each feature and class value respectively and H(X,Y), the joint probabilities of all combinations of values of X and Y [21]

2.2.3 Classifier

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y). For example, spam detection in email service providers can be identified as a classification problem. This is s binary classification since there are only 2 classes as spam and not spam. A classifier utilizes some training data to understand how given input variables relate to the class. In this case, known spam and non-spam emails have to be used as the training data. When the classifier is trained accurately, it can be used to detect an unknown email.Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).
For example, spam detection in email service providers can be identified as a classification problem. This is s binary classification since there are only 2 classes as spam and not spam. A classifier utilizes some training data to understand how given input variables relate to the class. In this case, known spam and non-spam emails have to be used as the training data. When the classifier is trained accurately, it can be used to detect an unknown email.

Lazy learners

Lazy learners simply store the training data and wait until a testing data appear. When it does, classification is conducted based on the most related data in the stored training data. Compared to eager learners, lazy learners have less training time but more time in predicting. Ex. k-nearest neighbor, Case-based reasoning

Eager learners

Eager learners construct a classification model based on the given training data before receiving data for classification. It must be able to commit to a single hypothesis that covers the entire instance space. Due to the model construction, eager learners take a long time for train and less time to predict. Ex. Decision Tree, Naive Bayes, Artificial Neural Networks

2.2.4 Classification algorithms

There is a lot of classification algorithms available now but it is not possible to conclude which one is superior to other. It depends on the application and nature of available data set. For example, if the classes are linearly separable, the linear classifiers like Logistic regression, Fisher,Äôs linear discriminant can outperform sophisticated models and vice versa.

Decision Tree

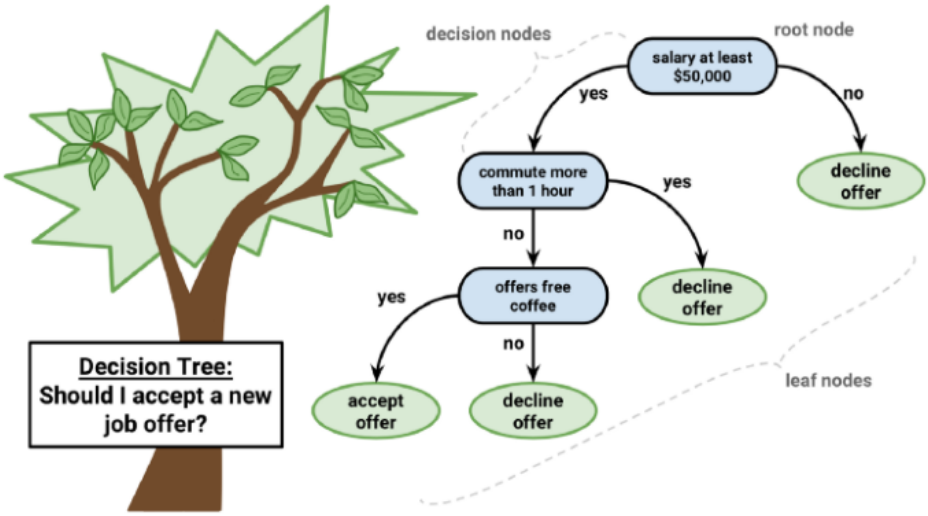

Figure 2.2: Decision tree

Decision tree builds classification or regression models in the form of a tree structure. It utilizes an if-then rule set which is mutually exclusive and exhaustive for classification. The rules are learned sequentially using the training data one at a time. Each time a rule is learned, the tuples covered by the rules are removed. This process is continued on the training set until meeting a termination condition. The tree is constructed in a top-down recursive divide-and-conquer manner. All the attributes should be categorical. Otherwise, they should be discretized in advance. Attributes in the top of the tree have more impact towards in the classification and they are identified using the information gain concept.
A decision tree can be easily over-fitted generating too many branches and may reflect anomalies due to noise or outliers. An over-fitted model has a very poor performance on the unseen data even though it gives an impressive performance on training data. This can be avoided by pre-pruning.

Naive Bayes

Naive Bayes is a probabilistic classifier inspired by the Bayes theorem under a simple assumption which is the attributes are conditionally independent.

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \tag{2.4}$$

The classification is conducted by deriving the maximum posterior which is the maximal $P(C_i|X)$ with the above assumption applying to Bayes theorem. This assumption greatly reduces the computational cost by only counting the class distribution. Even though the assumption is not valid in most cases since the attributes are dependent, surprisingly Naive Bayes has able to perform impressively.Naive Bayes is a very simple algorithm to implement and good results have obtained in most cases. It can be easily scalable to larger datasets since it takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.
Naive Bayes can suffer from a problem called the zero probability problem. When the conditional probability is zero for a particular attribute, it fails to give a valid prediction.

Artificial Neural Networks

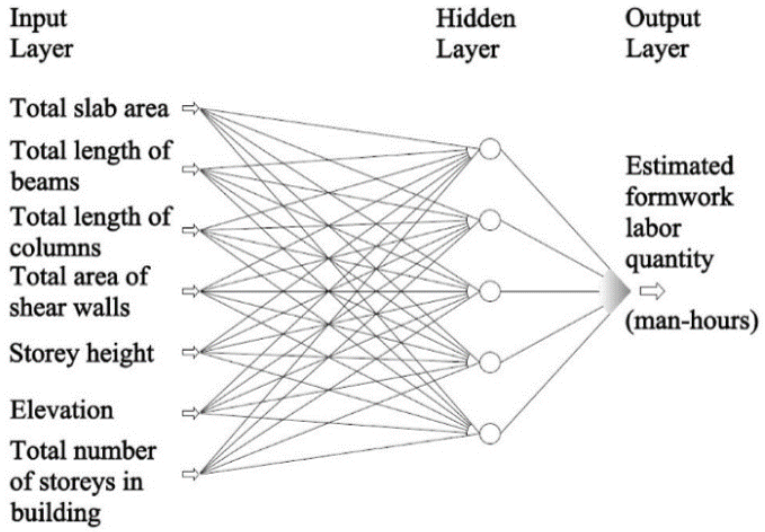

Figure 2.3: ANN

Artificial Neural Network is a set of connected input/output units where each connection has a weight associated with it started by psychologists and neurobiologists to develop and test computational analogs of neurons. During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input tuples.
There are many network architectures available now like Feed-forward, Convolutional, Recurrent etc. The appropriate architecture depends on the application of the model. For most cases feed-forward models give reasonably accurate results and especially for image processing applications, convolutional networks perform better.
There can be multiple hidden layers in the model depending on the complexity of the function

which is going to be mapped by the model. Having more hidden layers will enable to model complex relationships such as deep neural networks.
However, when there are many hidden layers, it takes a lot of time to train and adjust wights. The other disadvantage of is the poor interpretability of model compared to other models like Decision Trees due to the unknown symbolic meaning behind the learned weights.
But Artificial Neural Networks have performed impressively in most of the real world applications. It is high tolerance to noisy data and able to classify untrained patterns. Usually, Artificial Neural Networks perform better with continuous-valued inputs and outputs.

k-Nearest Neighbor (KNN)

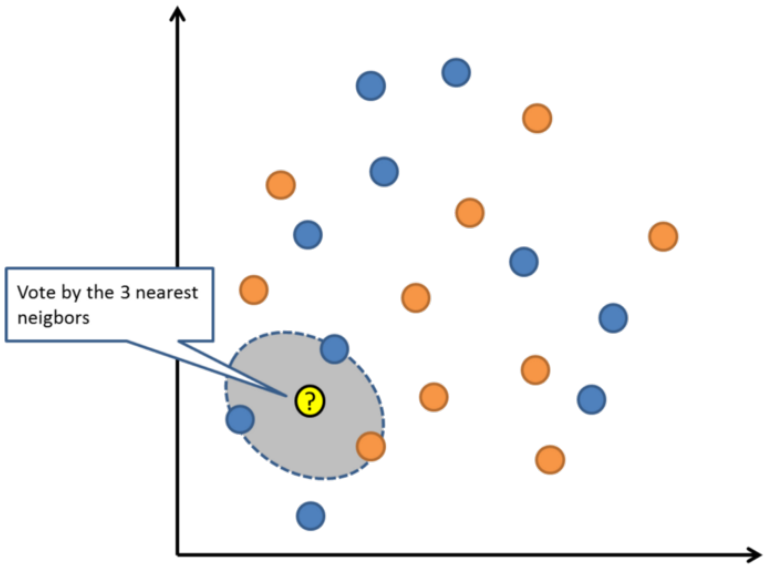

Figure 2.4: KNN

k-Nearest Neighbor is a lazy learning algorithm which stores all instances correspond to training data points in n-dimensional space. When an unknown discrete data is received, it analyzes the closest k number of instances saved (nearest neighbors)and returns the most common class as the prediction and for real-valued data it returns the mean of k nearest neighbors.
In the distance-weighted nearest neighbor algorithm, it weights the contribution of each of the k neighbors according to their distance using the following query giving greater weight to the closest neighbors.

$$W \equiv \frac{1}{d(x_q, x_i)^2} \tag{2.5}$$

Usually KNN is robust to noisy data since it is averaging the k-nearest neighbors.

Support Vector Machine (SVM)

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they,Äôre able to categorize new text.

A support vector machine takes these data points and outputs the hyperplane (which in two dimensions it,Äôs simply a line) that best separates the tags. This line is the decision boundary: anything that falls to one side of it we will classify as blue, and anything that falls to the other as red.

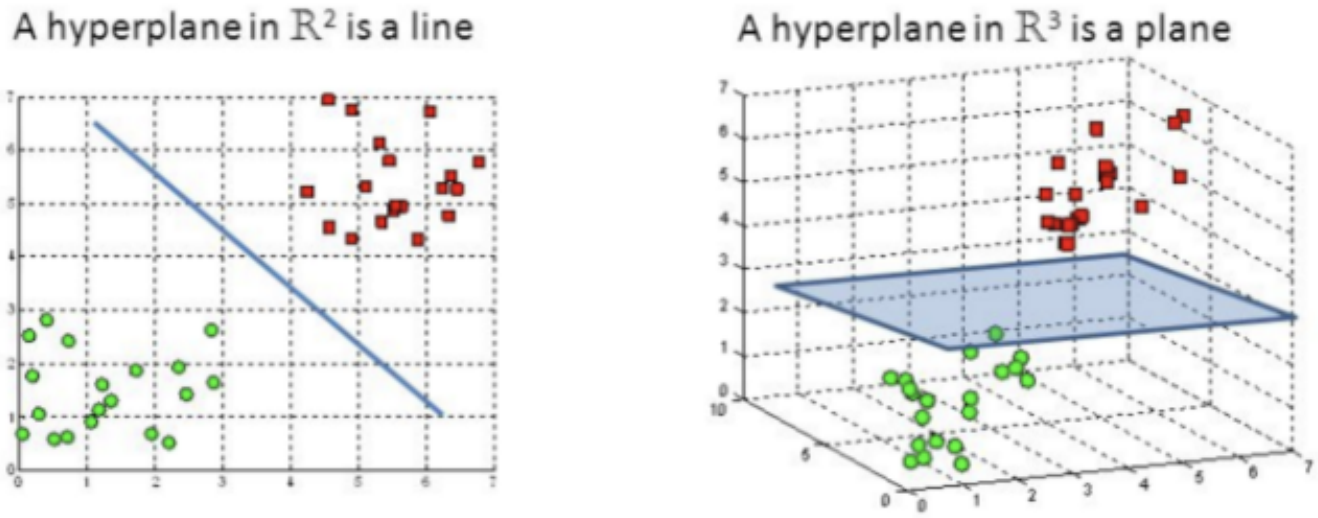

Figure 2.5: Hyperplanes in 2D and 3D feature space

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

SVM doesn,Äôt perform well when we have large data set because the required training time is higher.But It is effective in high dimensional spaces. It is effective in cases where the number of dimensions is greater than the number of samples. It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

2.2.5 Evaluating a classifier

After training the model the most important part is to evaluate the classifier to verify its applicability.

Holdout method

There are several methods exists and the most common method is the holdout method. In this method, the given data set is divided into 2 partitions as test and train 20% and 80% respectively. The train set will be used to train the model and the unseen test data will be used to test its predictive power.

Cross-validation

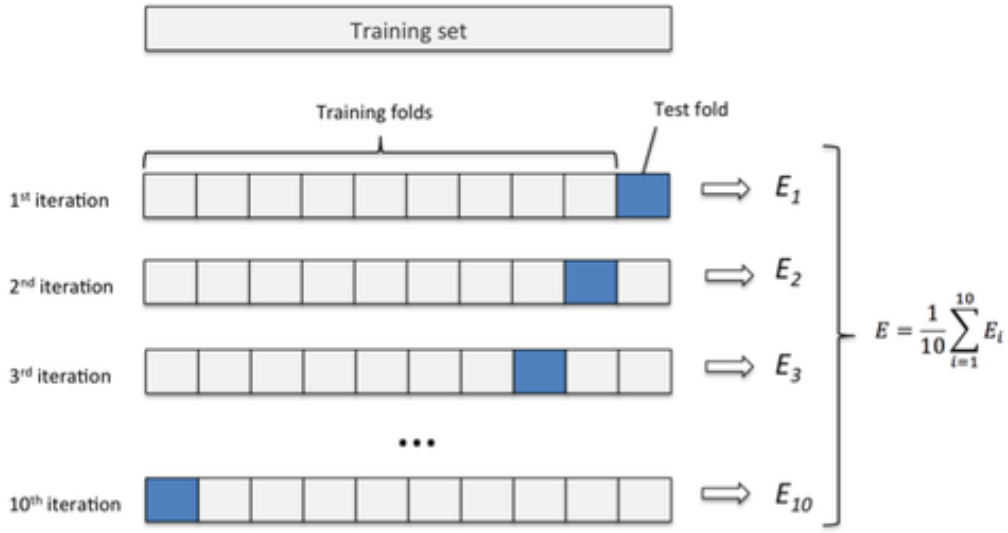

Figure 2.6: 10 fold cross validation

Over-fitting is a common problem in machine learning which can occur in most models. k-fold cross-validation can be conducted to verify that the model is not over-fitted. In this method, the data-set is randomly partitioned into k mutually exclusive subsets, each approximately equal size and one is kept for testing while others are used for training. This process is iterated throughout the whole k folds.

Precision and Recall

Precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. Precision and Recall are used as a measurement of the relevance.

ROC curve (Receiver Operating Characteristics)

ROC curve is used for visual comparison of classification models which shows the trade-off between the true positive rate and the false positive rate. The area under the ROC curve is a measure of the accuracy of the model. When a model is closer to the diagonal, it is less accurate and the model with perfect accuracy will have an area of 1.0

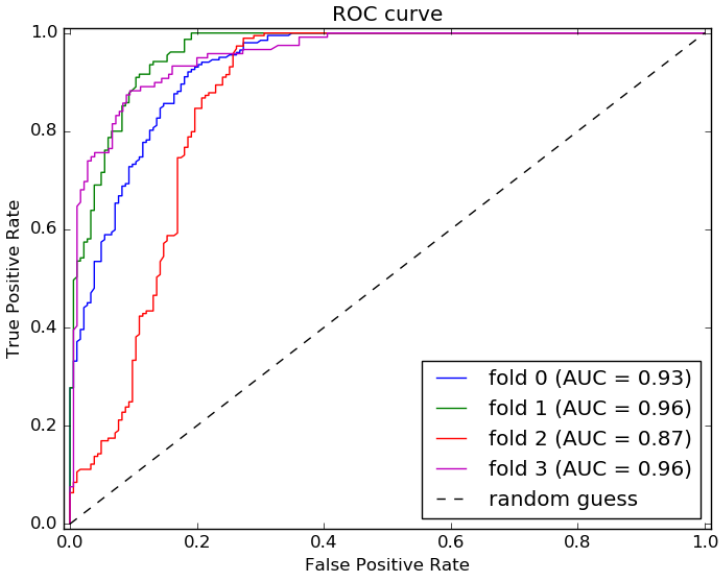

Figure 2.7: ROC curve example

CHAPTER 3

OUR METHOD

3.1 Introduction

Our goal is to select a feature subset that provides maximum classification accuracy with minimum possible size. We've focused on two major aspects, relevance and non redundancy. Relation between feature variables shows how relevant a feature is with the decision variable. We've used mutual information for that. We have removed redundant feature pairs using correlation analysis. In this chapter we have discussed how our method resolves the above mention aspects. First, we will describe our method step by step with the help of a flowchart and then we will see the detail algorithm.

3.2 Methodology

We have proposed a hybrid method which has two part, filter and wrapper. In filter part, we have used Mutual Information (MI) and Decision Dependent Correlation (DDC). Mutual information is used to select the most relevant features and Decision dependent correlation is used to eliminate correlated feature pairs to remove redundancy. Features selected from filter part are used in wrapper model. In wrapper model we generate all possible subsets and evaluate them using appropriate classifier.

The algorithm is show below with a flowchart. At first we discretize data. It is important because dataset may have continuous data. Then we normalize the data which is a good practice while applying machine learning model. Then we calculate mutual information for all features and select top ten among them. We add more features to our goal feature set ensuring that they are as less correlated as possible with the already taken features. This is just to ensure that we are not taking redundant features.

Then finally we generate all subset from the filtered shortlisted feature set. We evaluate each of them and take the best among them.

3.2.1 Flowchart

In figure 1.1 we have shown our method. Here threshold indicates the maximum size of the intial filtered feature set. It is very important as we are generating all subsets. After several experiments on various datasets we've decided to set the threshold value to 15.

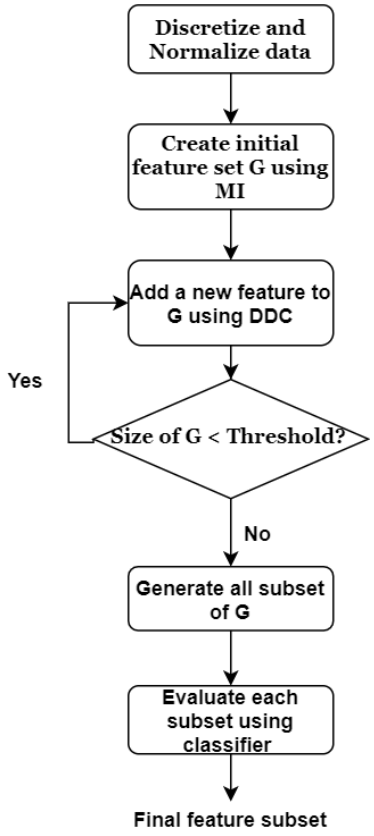

Figure 3.1: Flowchart of our method

3.2.2 Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges.We have used min max normalization

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Figure 3.2: Min Max Normalization

3.2.3 Discretization

Discretization is the process through which we can transform continuous variables, models or functions into a discrete form. We do this by creating a set of contiguous intervals (or bins) that go across the range of our desired variable/model/function.
Mathematical problems with continuous data have an infinite number of DoF. Such a problem would entail having limited degrees of freedom (DoF) since our calculations cannot go on forever. Data Scientists require using Discretization for a number of reasons. Many of the top contributions on Kaggle use discretization for some of the following reasons:

- Often, it is easier to understand continuous data (such as weight) when divided and stored into meaningful categories or groups. For example, we can divide a continuous variable, weight, and store it in the following groups : Under 100 lbs (light), between 140,Äì160 lbs (mid), and over 200 lbs (heavy) We would consider the structure useful if we see no objective difference between variables falling under the same weight class.In our example, weights of 85 lbs and 56 lbs convey the same information (the object is light). Therefore, discretization helps make our data easier to understand if it fits the problem statement.

- Continuous features have a smaller chance of correlating with the target variable due to infinite degrees of freedom and may have a complex non-linear relationship. Thus, it may be harder to interpret an such a function. After discretizing a variable, groups corresponding to the target can be interpreted.

- Certain models may be incompatible with continuous data, for example, alternative decision-tree models such as a Random-Forest model is not suitable for continuous features. Feature engineering methods, for example any entropy-based methods may not work with continuous data, thus we would discretize variables to work with different models methods.

- When we discretize a model, we are fitting it to bins and reducing the impact of small fluctuation in the data. Often, we would consider small fluctuations as noise. We can reduce

this noise through discretization. This is the process of ‚Äús‚Äùmoothing‚Äù, wherein each bin smoothens fluctuations, thus reducing noise in the data.

Continuous features in the data can be discretized using a uniform discretization method. Discretization considers only continues features, and replaces them in the new data set with corresponding categorical features.We have used KBinsDiscretizer to discretize our continuous data.

3.2.4 Mutual Information (MI)

Mutual information is calculated between two variables and measures the reduction in uncertainty for one variable given a known value of the other variable.The mutual information between two random variables X and Y can be stated formally as follows:

- I(X;Y) = H(X) - H(X | Y)

Where I(X ; Y) is the mutual information for X and Y, H(X) is the entropy for X and H(X | Y) is the conditional entropy for X given Y. The result has the units of bits.

Mutual information is a measure of dependence or ‚Äúmutual dependence‚Äù between two random variables. As such, the measure is symmetrical, meaning that I(X ; Y) = I(Y ; X).

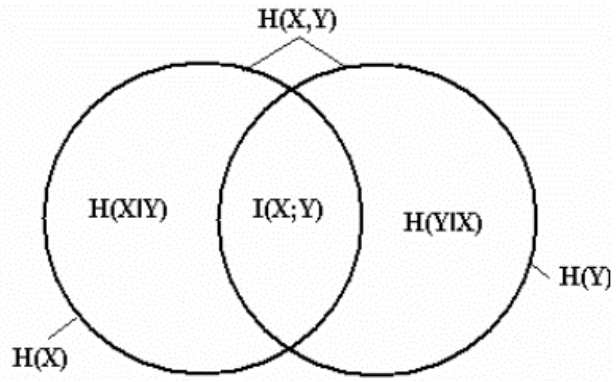

Figure 3.3: Relation between entropy and mutual information

We have generated initial feature set G using MI. First we calculated mutual information values for each feature. Then we select top 10 features with highest mutual information values. Mutual information for each feature denotes how much this feature is important with respect to certain decision variable. By taking the features with high mutual information values we have ensured that our initial shortlisted feature are relevant to out decision variable.

3.2.5 Decision Dependent Correlation (DDC)

Dependency measure or correlation measures qualify the accuracy of decision to predict the value of one variable. The main shortcomings of classical linear correlations are the assumption of linear correlation between the features and the requirement that all features contain numerical values. To overcome these shortcomings, several information theory-based measures of association were introduced for the feature-class correlations and feature intercorrelations, such as the gain ratio and information gain, the symmetrical uncertainty coefficient, and several others based on the minimum description length principle. Good results were acquired through using the gain ratio for feature-class correlations and symmetrical uncertainty for feature intercorrelations. However, the symmetrical uncertainty measure is not accurate enough to quantify the dependency among features with respect to a given decision. A critical point was neglected that the correlation or redundancy between features is strongly related with the decision variable under consideration. Let X_i and X_j be two features. When there is a decision Y associated with the features, we say the correlation between them is decision dependent correlation (DDC).We define a correlation measure to quantify the information redundancy between Xi and Xj with respect to Y as follows :

$$Q_Y(X_i, X_j) = \frac{I(Y;X_i) + I(Y;X_j) - I(Y;X_i,X_j)}{H(Y)}$$

We used DDC to make sure that highly correlated features doesn't co-exist in out final feature set.Thus we have eliminated redundant features. Let

3.2.6 Wrapper part

In wrapper methods, the feature selection process is based on a specific machine learning algorithm that we are trying to fit on a given dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion.
As we generate all possible subsets, it is computationally expensive. So we had to make sure that initial filtered feature set from filter part is as minimal as possible. After several experiments on different datasets using different classifier, we have decided to keep the filtered feature set size at most 15. We've applied different classifiers such as naive bayes, decision tree, support vector machine.

3.2.7 The Algorithm

Algorithm 1 Feature selection

procedure

G ←top 10 features with highest mutual information values
F ←all other features from dataset
while $|G|$ <Threshold **do**
  minCorrelation ← ∞
  minCorrelatedFeature ← 0
  for each feature f in F **do**
    maxCorrelation ← 0
    for each feature g in G **do**
      maxCorrelation = max(maxCorrelation, DDC(f,g))
    end for
    if minCorrelation > maxCorrelation **then**
      minCorrelatedFeature ←f
      minCorrelation ← $maxCorrelation$
    end if
  end for
  F ← $F - \{f\}$
  G ← $G + \{f\}$
end while
S ←all subsets of G
max_accuracy ←0
Final_feature_set ← {}
for each subset s from S **do**
  curr_accuracy ← $Classify(s)$
  if curr_accuracy > max_accuracy **then**
    max_accuracy ← $curr\_accuracy$
    Final_feature_set ← s
  end if
end for

end procedure

3.2.8 How it works

In this subsection we will discuss how our algorithm works and how it solves the problems related to feature selection. As we have said earlier that our algorithm has two parts :

- **Filter part (line 1-18):** In filter part we have tried to select most relevant features excluding the redundant one. So how exactly our algorithm ensures feature relevance. Well as we have discussed earlier about mutual information, which is a measurement of relevance between a feature variable. In our algorithm we compute all the mutual information values and sort them in non-increasing order. Then we take top few features from them as an initial top important features. Then from the rest of the features we keep taking one by one using decision dependent correlation measure (line 3-18) which ensures that no redundant feature exists in our final selection. If a feature is highly correlated with already taken feature(s) than we don't take this feature. This is how our filter part resolves two major concerns of feature selection, feature relevance and feature redundancy.
- **Wrapper part (line 19-28):** The major concern of applying wrapper model is execution time as it generates all subset. Here we could safely use wrapper model as the feature subset size is reduced significantly in our filter part. So generation of all subsets will not be a problem at all. But why do we need to generate all subset? Well, as we know some there might be some interdependent features. So we need to check all subsets. In our algorithm we have generated all subset using bit manipulation which is very fast. Then we evaluate each subset using popular machine learning learning classifiers like SVM, DT and NB.

CHAPTER 4

EXPERIMENTAL STUDIES

4.1 Introduction

An experiment is a procedure carried out to support, refute, or validate a hypothesis. Experiments provide insight into cause-and-effect by demonstrating what outcome occurs when a particular factor is manipulated. Experiments vary greatly in goal and scale, but always rely on repeatable procedure and logical analysis of the results. In engineering and the physical sciences, experiments are a primary component of the scientific method. They are used to test theories and hypotheses about how physical processes work under particular conditions. Typically, experiments in these fields focus on replication of identical procedures in hopes of producing identical results in each replication. No theory can have a real significance without an experimentation on that theory. To establish some kind of proposition we must have conducted some experiments and analysis it,Äôs output results. It is true for our proposed heuristics also. That,Äôs why we have arranged some experiments to test our proposed hybrid method.
In this chapter, at first, we have introduced the data sets on which experiments are conducted. Then we have shown our experimental result on different datasets using various classifiers.

4.2 Datasets

We have used some popular datasets for machine learning tasks. Initially we work on only KDD'99 dataset but later we moved on to NSL KDD and few other datasets such as WDBC, SPECT, Hill_valley, Liver_disorder, pageblocks, waveform and sonar_signal. We have briefly discussed them in the following subsections.

4.2.1 NSL-KDD

NSL-KDD data set, developed by Tavallaee et al (2009), an enhanced version of **KDD-CUP1999** benchmark intrusion detection data set sought to solve the inherent problems of KDDCUP1999 data set. The first important limitation in the KDDCUP1999 data set is the huge number of redundant records in the sense that almost 78% training and 75% testing records are duplicated, as shown in Table 3.1 and Table 3.2. As a result the learning algorithms suffered prejudice to observe most frequent attacks, preventing it from recognizing rare attack records such as U2R and R2L classes. For NSL-KDD the training data set has 125973 patterns, and

testing data set consists of 22544 patterns. This number of records in the NSL-KDD train and test sets are reasonable. This advantage makes it affordable to run the experiments on the complete set without the need to randomly select a small portion.NSL-KDD data set covers four major categories of attacks such as Probing attacks (information gathering attacks), Denial-of-Service (DoS) attacks (deny legitimate requests to a system), User-to-Root (U2R) attacks (unauthorized access to local super-user or root), and Remote-to-Local (R2L) attacks (unauthorized local access from a remote machine). NSL-KDD data set is divided into labeled and unlabeled records and this class attribute has 21 predicated labels for each record. Here is some improvements that NSL-KDD data set has:

1. It does not include redundant records in the train set, so the classifiers will not be biased towards more frequent records.

2. There are no duplicate records in the proposed test sets; therefore, the performance of the learners is not biased by the methods which have better detection rates on the frequent records.

3. The number of selected records from each difficulty level group are inversely proportional to the percentage of records in the original KDD data set. As a result, the classification rates of distinct machine learning methods vary in a wider range, which make it more efficient to have an accurate evaluation of different learning techniques

4. The number of records in the train and test sets are reasonable, which make it affordable to run the experiments on the complete set without the need to randomly select a small portion. Consequently, evaluation results of different research works will be consistent and comparable

Attack Types	Number of Records
Normal	67343
DoS	45927
Probe	11656
R2L	995
U2R	52
Total	125973

Figure 4.1: Distribution of instance in NSL-KDD training data set

4.2.2 Wisconsin Diagnostic Breast Cancer (WDBC)

In this dataset features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. It has two predicting fields, diagnosis: B = benign, M = malignant. It is linearly separable using all 30 input features.

- Number of instances: 569
- Number of attributes: 32 (ID, diagnosis, 30 real-valued input features)
- Class distribution: 357 benign, 212 malignant

4.2.3 SPECT heart data

The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images.Each of the patients is classified into two categories: normal and abnormal.The database of 267 SPECT image sets (patients) was processed to extract features that summarize the original SPECT images.As a result, 44 continuous feature pattern was created for each patient. The pattern was further processed to obtain 22 binary feature patterns. SPECT is a good data set for testing ML algorithms; it has 267 instances that are described by 23 binary attribute.

- Number of instances: 267
- Number of attributes: 23

4.2.4 Hill-Valley Dataset

Each record represents 100 points on a two-dimensional graph. When plotted in order (from 1 through 100) as the Y co-ordinate, the points will create either a Hill (a ‚Äúbump‚Äù in the terrain) or a Valley (a ‚Äúdip‚Äù in the terrain).

- Number of instances: 606
- Number of attributes: 100

4.2.5 Other Datasets

We have tested on some other datasets from UCI such as:

- Liver Disorder
- Sonar signal
- Page blocks
- Waveform

Datasets in brief

Dataset Name	No of classes	No of features	No of records
NSL-kDD	4	41	125973
SPECT	2	21	267
Hill valley	2	100	606
WDBC	2	30	569
Waveform	3	21	5000
Sonar signal	2	60	208
Page blocks	4	11	5473
Liver disorder	2	7	345

4.3 Experiment Result

We have run our experiments on PyCharm IDE and Jupyter Notebook. For testing purpose we have used 10 fold cross validation on dataset. We have shown comparison between results found from our approach and other approaches. Among the other approaches we have selected the best ones.

Dataset(Class)	Classifier	Accuracy (Our approach)	Feature reduction (Our approach)	Accuracy (Other approach)	Feature reduction (Other approach)
WDBC	SVM	97.68%	4/30	96.65%	4/30
NSL-KDD (DOS)	Decision Tree	99.57%	11/41	99.30%	11/41
NSL-KDD (Probe)	Decision Tree	98.52%	11/41	97.40%	11/41
NSL-KDD (U2R)	Decision Tree	45.83%	11/41	59.60%	11/41
NSL-KDD (R2L)	Decision Tree	95.36%	11/41	95%	11/41
SPECT	SVM	72.6%	1/21	76.73%	5/21
Hill Valley	Naive Bayes	52%	2/100	77.93%	11/100
Liver Disorder	Naive Bayes	61.76%	3/7	63.20%	2/7
Page-blocks	Decision Tree	86.99%	5/11	96.90%	8/11
Waveform	Naive Bayes	79%	10/21	81.40%	8/21
Sonar signal	Naive Bayes	74.5%	4/60	87.98%	11/60

4.3.1 Result Analysis

From the result table we can see that we have pretty good accuracy on WDBC, page blocks , waveform datasets. In NSL-kDD dataset We have good accuracy for 3 classes namely DOS, Probe and R2L. The reason behind poor accuracy in case of U2R is class imbalance. We have only 52 data of U2R in the whole dataset. As we have used 10 fold cross validation, it is possible that in some cases the training set has none or very few of this 52 to data in it which naturally reads to low accuracy.
NSL-KDD dataset has a lot of DOS data compapred to other classes. So DOS detection rate is better compare to other classes.

In case of Liver disorder and page blocks datasets we can see that they have only 7 and 11 features respectively. So we were able to generated all possible feature subsets hence it it more dependent on classifier itself then our feature selection algorithm.
In case of Hill valley dataset we have 52% accuracy. It looks quite low but in reality the best result we have found in other papers is around 77%. So it's not that low as it's looks like.

CHAPTER 5

CONCLUSION

5.1 Conclusion

In this thesis work we have proposed a hybrid method for feature selection using machine learning. Our method has two parts namely filter and wrapper. Our method takes care of two major concerns of feature selection and they are feature relevance and feature redundancy. Using mutual information and decision dependent correlation analysis we have ensured that our selected feature subset has all important relevant features and doesn't have redundant features. As two or more feature may be dependent on each other so we took care of this in our wrapper part by generating all subsets and taking the best among them. We have applied our method on various popular machine learning datasets and we have good accuracy on them.

5.2 Future work

Our proposed algorithm improve the machine learning task by extracting the relevant and effective feature set from original feature set. Two or more individual less important feature may turn out to be important when they are paired together. In future we will work on that. Also future work will involve planning to investigate the possibility and feasibility of implementing our approach in real time applications. We are planning also to improve our approach and decrease its execution time.

REFERENCES

[1] M. Y. Guangzhi Qu, Salim Hariri, "A new dependency and correlation analysis for features," *Computers & operations research*, vol. 17, no. 9, 2005.

[2] N. N. O. Thu Zar Phyu, "Performance comparison of feature selection methods," 2016.

[3] D. W. Fengli Zhang, "An effective feature selection approach for network intrusion detection," *School of Computer Science Engineering University of Electronic Science and Technology of China Chengdu, China*, 2013.

[4] S. Zaman and F. Karray, "Features selection using fuzzy esvdf for data dimensionality reduction," 2009.

[5] W. S. Syed Imran Ali, "A feature subset selection method based on symmetric uncertainty and ant colony optimization," *International Journal of Computer Applications (0975 ,Äì 8887)Volume 60,Äì No.11, December 2012*.

[6] H. Liu and R. Setiono, "A probabilistic approach to feature selection - a filter solution," *the 13th International Conference on Machine Learning, pp. 319-327, 1996*.

[7] M. Hall, "Feature selection for discrete and numeric class machine learning,"

[8] A. Suebsing and N. Hiransakolwong, "Euclideanbased feature selection for network intrusion detection," *International Conference on Machine Learning and Computing IPCST, 2011*.

[9] Z. Karimi, M. Mansour, and A. Harounabadi, "Feature ranking in intrusion detection dataset using combination of filtering," *International Journal of Computer Applications,Vol. 78, September 2013*.

[10] S. K. A. Chaudhary and Rajkamal, "Performance evaluation of feature selection methods for mobile devices," *ISSN: 2248-9622, Vol. 3, Issue 6, NovDec 2013, pp. 587-594*.

[11] H. Almuallim and T. G. Dietterich, "Learning with many irrelevant features," *In Proceedings of the Ninth National Conference on Articial Inte lligence (AAAI-91)*, vol. 2, pp. 547–552, 1991.

[12] P. L. J. H. Gennari and D. Fisher, "Models of incremental concept formation," no. 40, pp. 11–61, 1989.

[13] R. K. George H. John and K. Peger, "Irrelevant features and the subset selection problem," *In International Conference on Machine Learning, 1994.*, pp. 121–129.

[14] R. Kohavi and G. John., "Wrapper for feature subset selection.," 1997.

[15] T. Marill and D. Green., "On the effectiveness of receptors in recognition systems.," *IEEE Transactions on Information Theory, 9:1117, 1963*.

[16] C. Papadimitriou and K. Steiglitz., "Combinatorial optimization: Algorithms and complexity," *Prentice Hall, 1982*.

[17] j. Holland, "Adaption in neural and artificial systems.," *University of Michigen Press, 1975*.

[18] J. R. Quinlan, "Induction of decision trees," *Journal of Machine Learning, 1:81-106, 1986*.

[19] W. T. V. William H. Press, Saul A. Teukolsky and B. P. Flannery., "Numerical recipes in c: The art of scientic computing.," *Cambridge University Press, Cambridge, UK, 1988*.

[20] A. Ng., "Feature selection, l1 vs l2 regularization and rotational invariance.," *Proc. 21st International Conference on Machine Learning (ICML), pp 615-622, 2004*.

[21] Z. Karimi, M. Mansour, and A. Harounabadi, "Feature ranking in intrusion detection dataset using combination of filtering," *International Journal of Computer Applications,Vol. 78, September 2013*.