

PREDICTING MOVIE RATINGS AT IMDb

Project Report submitted in fulfillment of the requirement for the degree of
Bachelor of Technology

in

Information Technology

By

Gaurvi Lal (121418)

Anirudh Prasher(121429)

Under the supervision of

Dr. Pardeep Kumar

to



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Wanknaghat, Solan-173234,
Himachal Pradesh

Certificate Candidate's Declaration

I hereby declare that the work presented in this report entitled “ **Predicting Movie Ratings at IMDb**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2015 to December 2015 under the supervision of **Dr. Pardeep Kumar** (Assistant Professor, Senior Grade, Department of Computer Science & Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
Gaurvi Lal, 121418
Anirudh Prasher, 121429

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Dr. Pardeep Kumar
Assistant Professor (Senior Grade)
Department of Computer Science & Engineering
Date

Acknowledgement

We place on record and warmly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our guide **Dr. Pardeep Kumar**, Assistant Professor (Senior Grade), Department of Computer Science & Engineering at Jaypee University of Information Technology, Waknaghat in bringing this report to a successful completion.

We would also like to express our sincere gratitude to **Prof. Dr. Satya Prakash Ghrera**, Head, Department of Computer Science & Engineering, Jaypee University of Information Technology, Waknaghat for permitting us to make use of the facilities available in the department to carry out the project successfully.

We also express our humble gratitude to the technical and non-teaching staff of the Department of Computer Science & Engineering for their constant assistance and co-operation.

(Student Signature)
Gaurvi Lal, 121418
Anirudh Prasher, 121429

Table of Contents

| | |
|--|------------|
| Title Page | |
| Certificate..... | i |
| Acknowledgement..... | ii |
| Table of Contents..... | iii |
| List of Abbreviations..... | iv |
| Abstract..... | iv |
| 1. Introduction..... | 1 |
| 1.1 Introduction..... | 1 |
| 1.2 Problem Statement..... | 1 |
| 1.3 Objectives..... | 1 |
| 1.4 Methodology..... | 2 |
| 1.5 Organization..... | 2 |
| 2. Literature Survey..... | 8 |
| 3. System Development..... | 13 |
| 3.1 Analysis/Design/Development/Algorithm..... | 13 |
| 3.2 Model Development..... | 18 |
| 4. Performance Analysis..... | 35 |
| 5. Conclusions..... | 50 |
| 5.1 Conclusions..... | 50 |
| 5.2 Future Scope..... | 50 |
| 5.3 Applications Contributions..... | 50 |
| 6. References..... | 51 |
| 7. Appendices..... | 52 |

List of Abbreviations

IMDb-Internet Movie Database

IDE-Interactive Development Environment

.csv-Comma Separated Values

KDD-Knowledge Discovery in Databases

ML-Machine Learning

AI-Artificial Intelligence

CLI-Command Line Interface

GUI-Graphical User Interface

URL-Uniform Resource Locator

SQL-Structured Query Language

JDBC-Java Database Connectivity

ARFF-Attribute Relation File Format

€- belongs to

Abstract

In this project, we are interested in predicting the movie ratings at IMDb. We apply the predictive data mining techniques of classification and to the database. Among the various attributes of movies like year of release, length (running time), number of votes, and genres, we determine which attributes of a movie affect its rating the most. The prototype model is based on the decision tree (J48) based classification using WEKA 3.7 and Java (Netbeans IDE 8.1)

Chapter-1 INTRODUCTION

1.1 Introduction

The International movie industry produces a large number of movies per year at 50,000/year, produced by Hollywood and her foreign equivalents [1]. However, very few movies taste success and are ranked high.

1.2 Problem Statement

Given the low success rate [2], models and mechanisms to reliably predict the ranking and / or box office collections of a movie can help de-risk the business significantly and increase average returns. Various stakeholders such as actors, financiers, directors etc. can use these predictions to make more informed decisions.

1.3 Objectives

Some of the questions that can be answered using prediction models are:

1. Is the genre of the movie a key determinant of rank or success of a movie?
2. Does the length (running time) matter?
3. Does the year of release matter?
4. Do the number of votes affect the ranking?

Further, a DVD rental agency or a distribution house could use these predictions to determine which titles to stock or promote respectively.

The objective of the project is to predict the ranking of movies on www.imdb.com using information available from different sources including IMDB itself.

1.4 Methodology

Predictive data mining technique of classification -decision tress (J48) was applied to the dataset that was obtained from trusted sources [3] that provide reliable data about the movies and their ratings. Among the various attributes of movies like the year of release, length (running time), number of votes, and genres, we determine which attributes of a movie affect its rating the most.

1.5 Organization

1.5.1 What is IMDb? [4]

IMDb started in 1990 as a hobby project by an international group of movie and TV fans.

IMDb is now the world's most popular and authoritative source for movie, TV and celebrity content. It offers a searchable database of more than 185 million data items including more than 3.5 million movies, TV and entertainment programs and more than 7 million cast and crew members.

Consumers rely on the information IMDb provides -- including local movie showtimes, ticketing, trailers, critic and user reviews, personalized recommendations, photo galleries, entertainment news, quotes, trivia, box-office data, editorial feature sections and a universal Watchlist – when deciding what to watch and where to watch it. IMDb's portfolio of leading entertainment apps includes its popular "Movies & TV" app for iPhone, iPad, Kindle Fire, Android phones, Android tablets and its mobile-optimized website. To date, there have been more than 115 million downloads of IMDb's mobile apps worldwide. IMDb's X-Ray for Movies & TV Shows is a feature that revolutionizes the viewing experience by bringing the power of IMDb directly to Kindle Fire HD, Fire TV and Fire TV Stick. IMDb's Facebook page and official Twitter account are followed by more than 7 million passionate entertainment fans. IMDbPro is a subscription version of IMDb designed exclusively for professionals who work in the entertainment industry. IMDbPro provides a casting service, contact information, in production listings for film and television projects, exclusive


STARMeter rankings that are determined by user searches on IMDb, and a mobile optimized website. Additionally, IMDb owns and operates Withoutabox, the premier submission service for film festivals and filmmakers, and Box Office Mojo, the leading online box-office reporting service. IMDb.com is operated by IMDb.com, Inc., a wholly owned subsidiary of Amazon.com, Inc. (NASDAQ:AMZN).

1.5.2 How does IMDb calculate the vote displayed on a film or show page? [5]

IMDb takes all the individual votes cast by IMDb registered users and use them to calculate a single rating. They don't use the arithmetic mean of the votes (although they do display the mean and average votes on the votes breakdown) -- the rating displayed on a film's page is a **weighted average**.

IMDb displays **weighted** vote averages rather than raw data averages. Various filters are applied to the raw data in order to eliminate and reduce attempts at "vote stuffing" by individuals more interested in changing the current rating of a movie or TV show than giving their true opinion of it.

Although the raw mean and median are shown under the detailed vote breakdown graph on the ratings pages, the user rating vote displayed on a film / show's page is a weighted average.



Watch It

at Amazon

or

Buy it at Amazon

More at IMDb Pro

Discuss in Boards

Add to Watchlist

Update Data

User ratings for
National Treasure (2004) [More at IMDbPro »](#)

[login to vote](#)

252939 IMDb users have given a **weighted average** vote of 6.9 / 10

Demographic breakdowns are shown below.

| Votes | Percentage | Rating |
|-------|------------|--------|
| 18617 | 7.4% | 10 |
| 18838 | 7.4% | 9 |
| 49780 | 19.7% | 8 |
| 82674 | 32.7% | 7 |
| 47586 | 18.8% | 6 |
| 19128 | 7.6% | 5 |
| 7895 | 3.1% | 4 |
| 3769 | 1.5% | 3 |
| 2049 | 0.8% | 2 |
| 2603 | 1.0% | 1 |

Arithmetic mean = 7.0. Median = 7

This page is updated daily.

See user ratings report for:

| | Votes | Average |
|--------------------|--------|---------|
| Males | 180706 | 6.8 |
| Females | 33230 | 7.0 |
| Aged under 18 | 959 | 7.0 |
| Males under 18 | 771 | 6.9 |
| Females under 18 | 179 | 7.3 |
| Aged 18-29 | 103347 | 7.0 |
| Males Aged 18-29 | 84818 | 6.9 |
| Females Aged 18-29 | 17731 | 7.0 |
| Aged 30-44 | 85959 | 6.7 |
| Males Aged 30-44 | 74077 | 6.7 |
| Females Aged 30-44 | 11072 | 6.9 |
| Aged 45+ | 19121 | 6.9 |
| Males Aged 45+ | 15668 | 6.8 |
| Females Aged 45+ | 3187 | 7.2 |
| IMDb staff | 14 | 6.5 |
| Top 1000 voters | 780 | 6.5 |
| US users | 55399 | 7.1 |
| Non-US users | 126885 | 6.8 |
| IMDb users | 252939 | 6.9 |

Quicklinks

user ratings ▼

Top Links

- trailers and videos
- full cast and crew
- trivia
- official sites
- memorable quotes

Overview

- main details
- combined details
- full cast and crew
- company credits

Awards & Reviews

- user reviews
- external reviews
- awards
- user ratings
- parents guide

First of all, the same formula is applied universally across the database to all movies and shows without exception so there is no bias in when and where the scheme operates. The objective of the scheme is to present a more representative rating which is immune from abuse by subsets of individuals who have combined together with the aim of influencing (either up or down) the ratings of specific movies or shows. This includes people involved in the production of a movie / TV show and their friends or fans trying to unduly raise the rating far above that of where the typical IMDb users would rate it.

The scheme combines a number of well-known and proven statistical methods, including a trimmed mean to reduce extreme influences and, most importantly a complex voter

weighting system to make sure that the final rating is representative of the general voting population and not subject to over influence from individuals who are not regular participants in the poll. The scheme has been developed internally over the 25 years which the poll has been in operation and tuned on a regular basis to make sure it remains fair.

A weighted average, is defined as "an average that takes into account the proportional relevance of each component, rather than treating each component equally". It's a very simple, universally accepted statistical method, commonly used in a wide variety of fields (from financial analysis to student reports).

For example, an automobile magazine reviewing a new car may give it high marks in several categories (appearance, comfort, fuel efficiency, price, number of cup holders). However a model with high ratings in all those categories may still get a low overall score if it gets a low vote in just one or two other areas (like speed or safety): even the cheapest, nicest-looking and most fuel-efficient car isn't worth buying if the fuel tank explodes when you hit the brakes or if its top speed is only 15 miles per hour. Clearly, a high (or low) vote in some categories has more **weight** than the same exact vote cast in another category, so the final rating takes these differences into account.

IMDb's calculations follow a similar principle: some votes have more weight than others. The idea is to give a more objective rating and neutralize attempts to artificially inflate or deflate the average user rating on a film or show.

This scheme is applied uniformly to all films listed in the database, without exception; and has proved to be very effective.

1.5.3 How/where does IMDb gets its information? How accurate/reliable is it? [6]

The information in IMDb comes from various sources. While IMDb actively gathers information from and verifies items with studios and filmmakers, the bulk of their information is submitted by people in the industry and the IMDb official site visitors.

In addition to using as many sources as they can, their data goes through consistency checks to ensure it's as accurate and reliable as possible. However, there's absolutely no substitute for an international team of entertainment fans with an encyclopedic knowledge of

trivia and a large assortment of reference works (and they include in this group many of their loyal contributors). Their sources of information include, but are not limited to, on-screen credits, press kits, official bios, autobiographies, and interviews.

Given the sheer volume and the nature of the information they list, occasional mistakes are inevitable and, when spotted/reported, they are promptly verified and fixed. That's why they welcome corrections and submissions.

1.5.4 IMDb Dataset Description [7]

Movies were selected for inclusion if they had a known length and had been rated by at least one IMDb user. The data set contains the following fields:

- **year.** Year of release.
- **length.** Length in minutes.
- **votes.** Number of IMDB users who rated this movie.
- **genre-**
 - ❖ **Action**
 - ❖ **Comedy**
 - ❖ **Documentary**
 - ❖ **Drama**
 - ❖ **Romance**
 - ❖ **Animation**
 - ❖ **ActCom**(Action|Comedy)
 - ❖ **Action|Drama**
 - ❖ **Action|Romance**
 - ❖ **Comedy|Drama**
 - ❖ **Drama|Romance**
 - ❖ **RomCom**(Romantic Comedy)
 - ❖ **Multi-genre.**Movies with more than two genres.
- **Ranking.** IMDB user ranking

- ❖ low:1-3
 - ❖ medium:4-6
 - ❖ high:7-10
- IMDB offers a rating / ranking scale that allows users to rate films by choosing one of ten categories in the range 1–10, with each user able to submit one rating. The points of reference given to users of these categories are the descriptions "1 (awful)" and 3
 - "10 (excellent)"; and these are the only descriptions of categories.
 - There are a total of 36,543 movie records in the dataset.
 - The dataset is stored in a file with .csv format.

```

year,length,votes,Genre,ranking
1928,86,231,ActCom,high
1933,70,316,ActCom,medium
1939,117,1988,ActCom,high
1939,72,48,ActCom,medium
1940,62,13,ActCom,medium
1952,105,831,ActCom,high
1960,84,16,ActCom,medium
1962,91,539,ActCom,medium
1963,105,782,ActCom,high
1963,192,5891,ActCom,high
1964,122,122,ActCom,high
1964,108,199,ActCom,high
1964,115,225,ActCom,medium
1964,100,6,ActCom,medium
1964,83,6,ActCom,low
1965,92,302,ActCom,medium
1965,160,2248,ActCom,medium
1965,138,1064,ActCom,medium
1965,88,67,ActCom,medium
1966,90,257,ActCom,medium
1966,106,28,ActCom,medium
1966,102,78,ActCom,medium
1966,100,52,ActCom,medium
1966,102,395,ActCom,medium
1966,92,47,ActCom,medium
1966,72,117,ActCom,low
1966,70,1144,ActCom,low
1967,114,935,ActCom,medium
1967,107,133,ActCom,medium
1967,131,4173,ActCom,medium
1967,102,17,ActCom,medium
1968,89,14,ActCom,medium
1968,86,39,ActCom,medium
1969,99,4459,ActCom,high
1969,110,329,ActCom,medium

```

Chapter-2 LITERATURE SURVEY [8]

Recent years have witnessed an explosive growth in the amounts of data collected, stored, and disseminated by various organizations. Examples include:

- ❖ the large volumes of point-of-sale data amassed at the checkout counters of grocery stores,
- ❖ the continuous streams of satellite images produced by Earth-observing satellites,
- ❖ the avalanche of data logged by network monitoring software, etc.

One immediate difficulty encountered in these domains is how to extract useful information from massive data sets.

The sheer size of the data simply overwhelms our ability to manually sift through the data, hoping to find useful information. Fueled by the need to rapidly analyze and summarize the data, researchers have turned to data mining techniques. In a nutshell, **data mining is the task of discovering interesting knowledge automatically from large data repositories.**

Data mining is often considered to be an integral part of another process, called Knowledge Discovery in Databases. KDD refers to the overall process of turning raw data into interesting knowledge and consists of a series of transformation steps, including :

- data preprocessing,
 - data mining, and
 - postprocessing.
-
- ❖ The objective of data preprocessing is to convert data into the right format for subsequent analysis by selecting the appropriate data segments and extracting attributes that are relevant to the data mining task (feature selection and construction). For many practical applications, more than half of the knowledge discovery efforts are devoted to data preprocessing.
 - ❖ Postprocessing includes all additional operations performed to make the data mining results more accessible and easier to interpret. For example, the results can be sorted or filtered according to various measures to remove

uninteresting patterns. In addition, visualization techniques can be applied to help analysts explore data mining results.

Data mining tasks are often divided into two major categories:

- ❖ **Predictive.** The goal of predictive tasks is to use the values of some variables to predict the values of other variables.
- ❖ **Descriptive.** The goal of descriptive tasks is to find human-interpretable patterns that describe the underlying relationships in the data.

Data mining tasks can be accomplished using a variety of data mining techniques:

- Data Clustering
- Predictive Modeling
- Anomaly Detection
- Association Rule Mining

➤ **Predictive modelling** is used primarily for predictive data mining tasks. The input data for predictive modelling consists of two distinct types of variables:

- (1) **explanatory variables**, which define the essential properties of the data, and
- (2) **one or more target variables**, whose values are to be predicted.

Predictive modelling techniques can be further divided into two categories: classification and regression.

1. **Classification techniques** are used to predict the values of discrete target variables.
 2. **Regression techniques** are used to predict the values of continuous target variables.
- **Association rule mining** seeks to produce a set of dependence rules that predict the occurrence of a variable given the occurrences of other variables.
 - **Cluster analysis** finds groupings of data points so that data points that belong to one cluster are more similar to each other than to data points belonging to a different cluster.
 - **Anomaly detection** identifies data points that are significantly different than the rest of the points in the data set.

Challenges of Data Mining

There are several important challenges in applying data mining techniques to large data sets:

- **Scalability.** Scalable techniques are needed to handle the massive size of some of the datasets that are now being created. Such datasets typically require the use of efficient methods for storing, indexing, and retrieving data from secondary or even tertiary storage systems. Furthermore, parallel or distributed computing approaches are often necessary if the desired data mining task is to be performed in a timely manner.
- **Dimensionality.** In some application domains, the number of dimensions (or attributes of a record) can be very large, which makes the data difficult to analyze.
- **Complex Data.** Traditional statistical methods often deal with simple data types such as continuous and categorical attributes. However, in recent years, more complicated types of structured and semi-structured data have become more important. One example of such data is graph-based data representing the linkages of web pages, social networks, or chemical structures. Traditional data analysis techniques often need to be modified to handle the complex nature of such data.
- **Data Quality.** Many data sets have one or more problems with data quality, e.g., some values may be erroneous or inexact, or there may be missing values. As a result, even if a 'perfect' data mining algorithm is used to analyze the data, the information discovered may still be incorrect. Hence, there is a need for data mining techniques that can perform well when the data quality is less than perfect.
- **Data Ownership and Distribution.** For a variety of reasons, e.g., privacy and ownership, some collections of data are distributed across a number of sites. In many such cases, the data cannot be centralized, and thus, the choice is either distributed data mining or no data mining. Challenges involved in developing distributed data mining solutions include the need for efficient algorithms to cope with the distributed and possibly, heterogeneous data sets, the need to minimize the cost of communication, and the need to accommodate data security and data ownership policies.

Data Mining and the Role of Data Structures and Algorithms

Research in data mining is motivated by a number of factors:

- Efficiency
- Flexibility
- Accuracy

The development and success of new data mining techniques is heavily dependent on the creation of the proper algorithms and data structures to address these needs.

Sometimes, currently existing data structures and algorithms can be directly applied, e.g., data access methods can be used to efficiently organize and retrieve data.

Classification

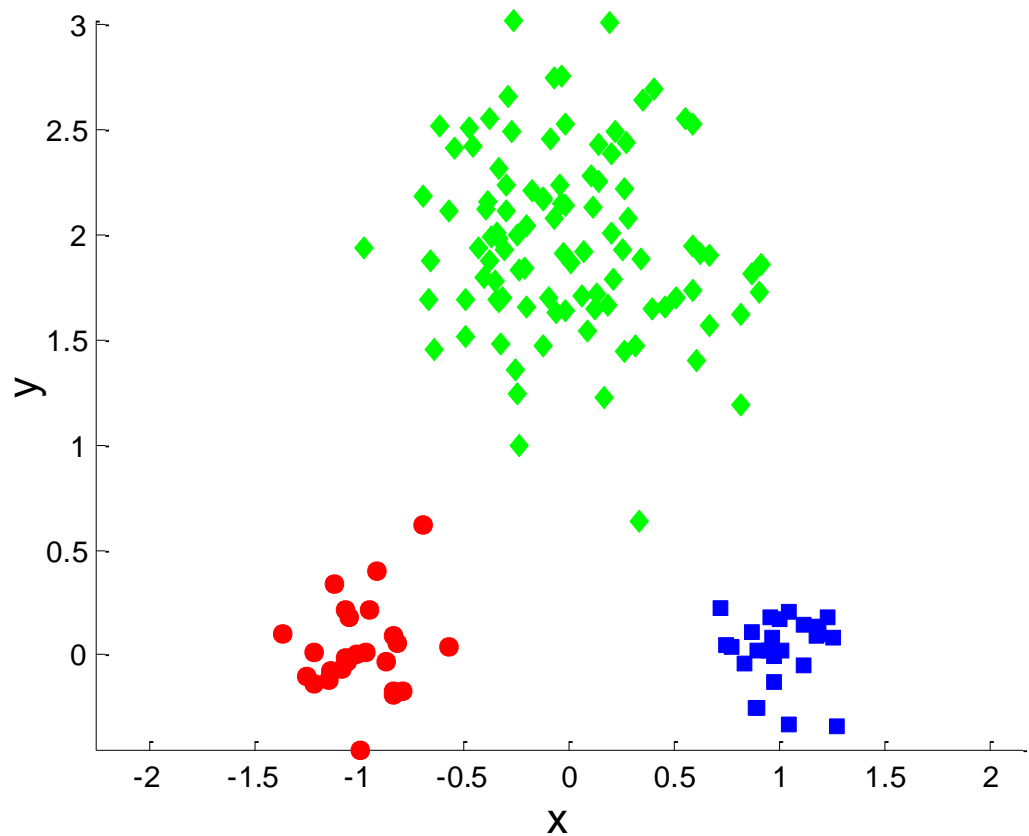
- Classification is the task of assigning objects to their respective categories.
- The data set in a classification problem typically consists of a collection of records or data objects.
- Each record, also known as an instance or example, is characterized by a tuple (x, y) , where x is the set of explanatory variables associated with the object and y is the object's class label. A record is said to be labeled if the value of y is known; otherwise, the record is unlabeled.
- Each attribute $x_k \in x$ can be discrete or continuous. On the other hand, the class label y must be a discrete variable whose value is chosen from a finite set $\{y_1, y_2, \dots, y_c\}$. If y is a continuous variable, then this problem is known as regression.

Association Analysis

- An important problem in data mining is the discovery of association patterns present in large databases.
- This problem was originally formulated in the context of market basket data, where the goal is to determine whether the occurrence of certain items in a transaction can be used to infer the occurrence of other items.
- If such interesting relationships are found, then they can be put to various profitable uses such as marketing promotions, shelf management, inventory management, etc.

Clustering

- Cluster analysis groups data objects based on information found in the data that describes the objects and their relationships.
- The goal is that the objects in a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups.
- The greater the similarity (or homogeneity) within a group, and the greater the difference between groups, the 'better' or more distinct the clustering.



Chapter-3 SYSTEM DEVELOPMENT

3.1 Analysis/Design/Development/Algorithm

Special System Requirements

1. Weka :[9]

- ❖ Weka is a collection of Machine Learning algorithms.
- ❖ ML is a form of Artificial Intelligence.
- ❖ AI is a form of computer science with software capable of self-modification: programs capable of changing themselves, programs capable of improving themselves (self-learning and self-improvement).

2. Dataset:

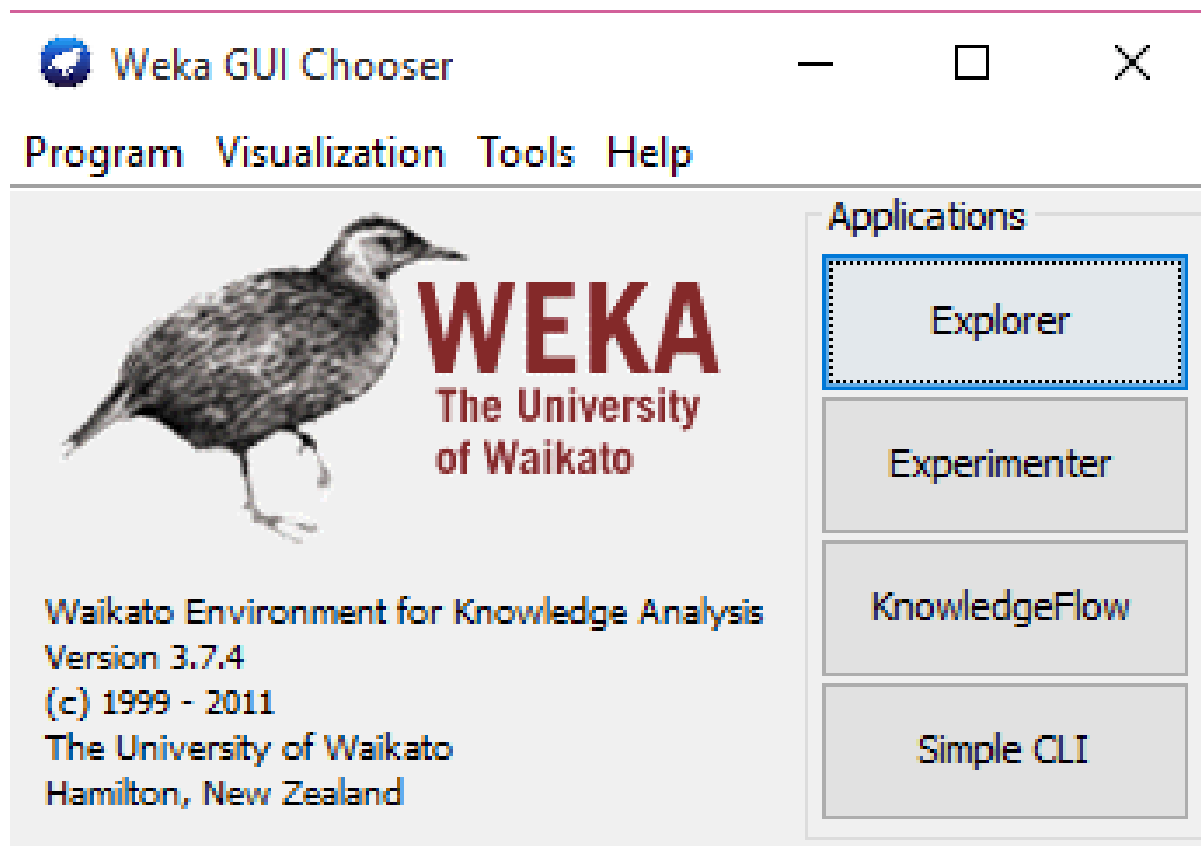
- ❖ Dataset in .csv format is required for interpretation and analysis in Weka. This is achieved by saving an excel file with a .csv extension.

Working on Weka [10]:

WEKA is a data mining system developed by the University of Waikato in New Zealand that implements data mining algorithms. WEKA is a state-of-the-art facility for developing machine learning techniques and their application to real-world data mining problems. It is a collection of machine learning algorithms for data mining tasks. The algorithms are applied directly to a dataset. WEKA implements algorithms for data preprocessing, classification, regression, clustering, association rules; it also includes visualization tools. The new machine learning schemes can also be developed with this package.

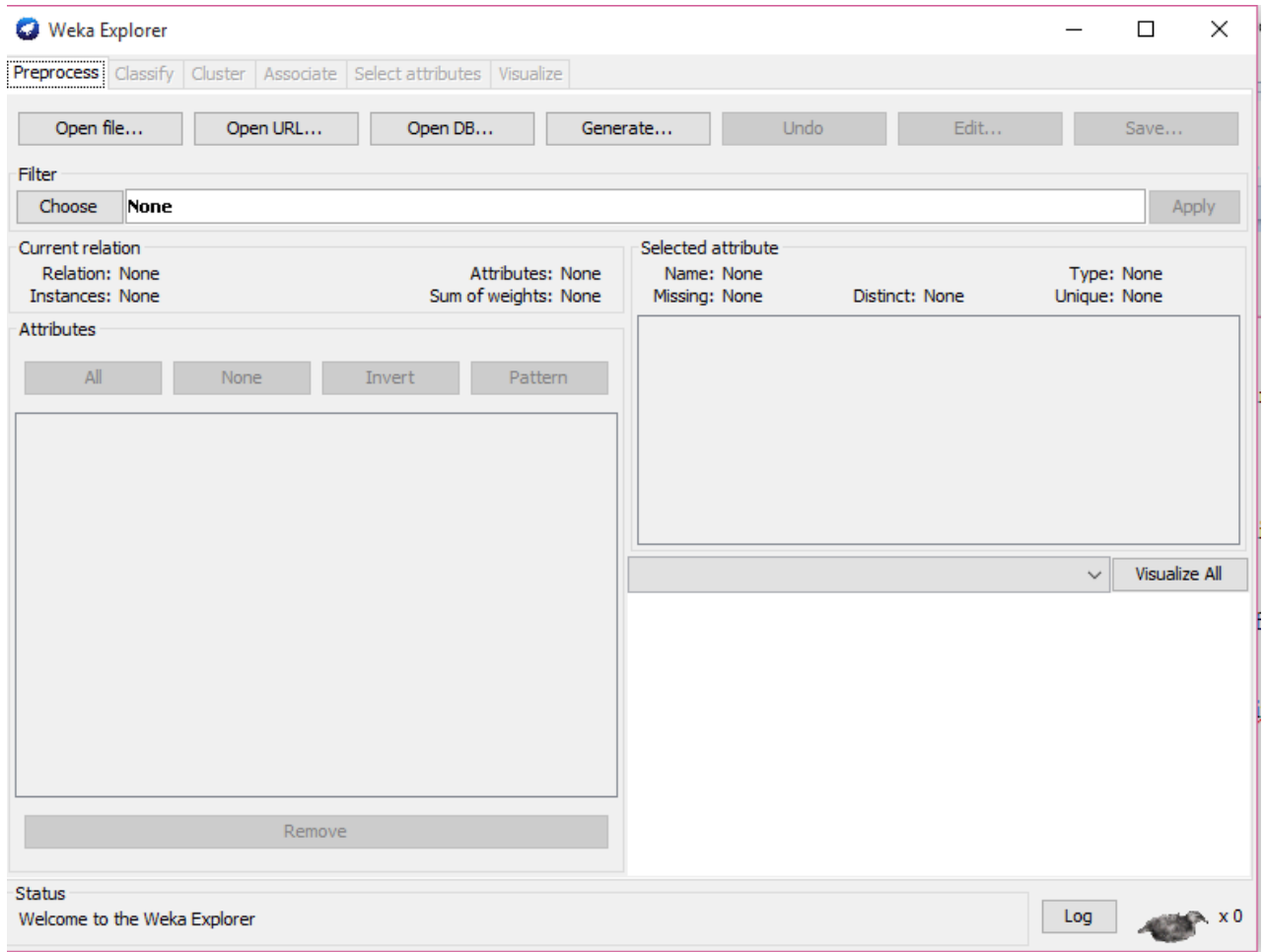
Launching WEKA Explorer

We can launch Weka from C:\Program Files directory



- **Simple CLI** provides a simple command-line interface and allows direct execution of Weka commands.
- **Explorer** is an environment for exploring data.
- **Experimenter** is an environment for performing experiments and conducting statistical tests between learning schemes.
- **KnowledgeFlow** is a Java-Beans-based interface for setting up and running machine learning experiments.

Click on 'Explorer' button in the 'WEKA GUI Chooser' window. 'WEKA Explorer' window appears on a screen.



Preprocessing Data

- At the very top of the window, just below the title bar there is a row of tabs. Only the first tab, 'Preprocess', is active at the moment because there is no dataset open.
- The first three buttons at the top of the preprocess section enable us to load data into WEKA.
- Data can be imported from a file in various formats: ARFF, CSV, C4.5, binary, it can also be read from a URL or from an SQL database (using JDBC).
- The easiest and the most common way of getting the data into WEKA is to store it as ARFF.

Loading data

- Load the data and look what is happening in the ‘Preprocess’ window.
- The most common and easiest way of loading data into WEKA is from ARFF file, using ‘Openfile...’.
- Click on ‘Open file...’ button and choose file from the local filesystem.
- Note, the data can be loaded from CSV file as well because some databases have the ability to convert data only into CSV format.

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

Filter: Choose **None** Apply

Current relation: Relation: movies, Instances: 36543, Attributes: 5, Sum of weights: 36543

Selected attribute: Name: year, Missing: 0 (0%), Distinct: 96, Type: Numeric, Unique: 2 (0%)

| Statistic | Value |
|-----------|----------|
| Minimum | 1901 |
| Maximum | 2005 |
| Mean | 1977.979 |
| StdDev | 22.232 |

Attributes: All | None | Invert | Pattern

| No. | Name |
|-----|--|
| 1 | <input checked="" type="checkbox"/> year |
| 2 | <input type="checkbox"/> length |
| 3 | <input type="checkbox"/> votes |
| 4 | <input type="checkbox"/> Genre |
| 5 | <input type="checkbox"/> ranking |

Remove

Class: ranking (Nom) Visualize All

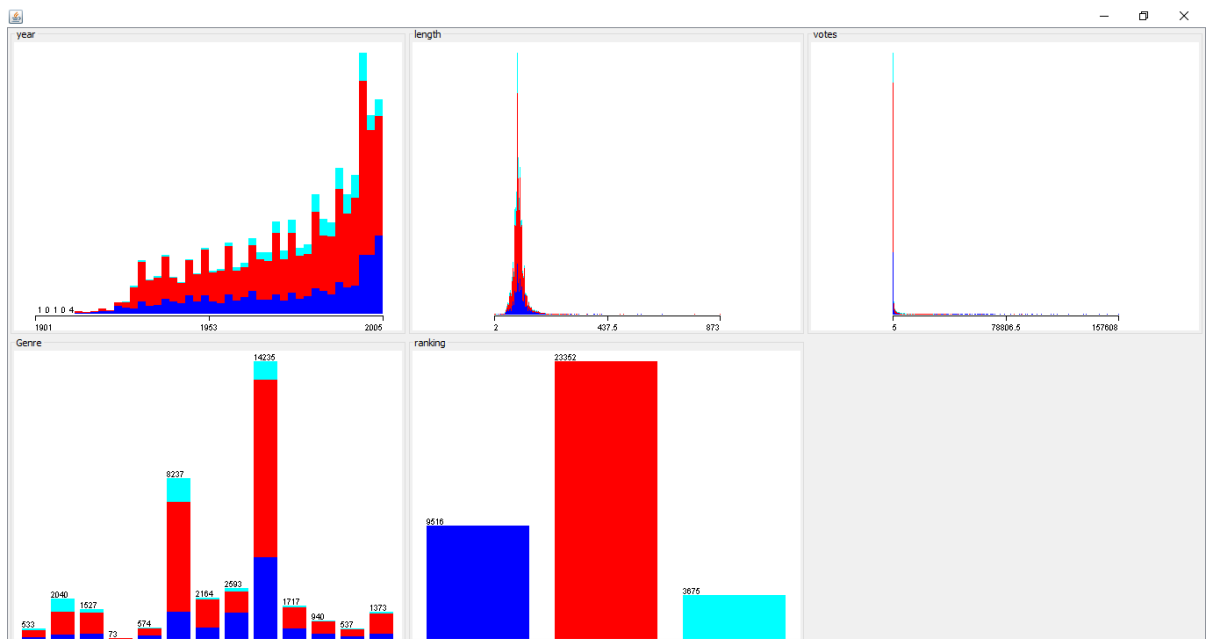
Status: OK

Log x 0

- Once the data is loaded, WEKA recognizes attributes that are shown in the ‘Attribute’ window.
- Left panel of ‘Preprocess’ window shows the list of recognized attributes.
- **No.** is a number that identifies the order of the attribute as they are in data file,
- **Selection tick boxes** allow us to select the attributes for working relation,
- **Name** is a name of an attribute as it was declared in the data file.

- The ‘Current relation’ box above ‘Attribute’ box displays the base relation (table) name and the current working relation and the number of attributes.
- During the scan of the data, WEKA computes some basic statistics on each attribute.
- The following statistics are shown in ‘Selected attribute’ box on the right panel of ‘Preprocess’ window:
 - ❖ **Name** is the name of an attribute,
 - ❖ **Type** is most commonly Nominal or Numeric, and
 - ❖ **Missing** is the number (percentage) of instances in the data for which this attribute is unspecified,
 - ❖ **Distinct** is the number of different values that the data contains for this attribute, and,
 - ❖ **Unique** is the number (percentage) of instances in the data having a value for this attribute that no other instances have.

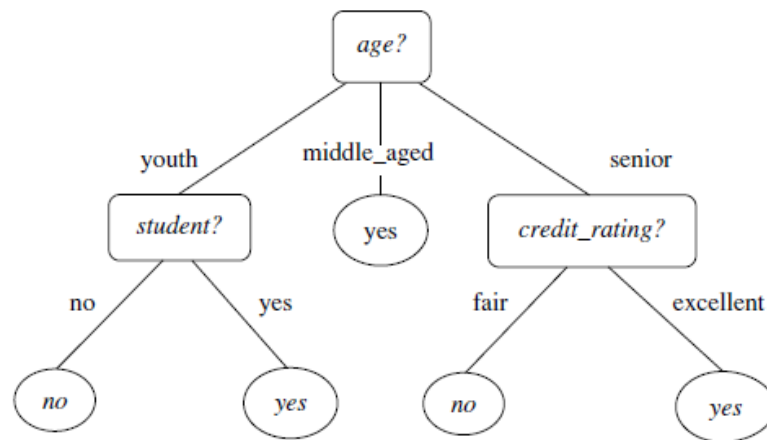
We can visualize the attributes based on selected class. One way is to visualize selected attribute based on class selected in the ‘Class’ pull-down window, or visualize all attributes by clicking on ‘Visualize All’ button.



Model Development (Analytical)[11]

Decision Tree Induction

- **Decision tree induction** is the learning of decision trees from class-labeled training tuples.
- A **decision tree** is a flowchart-like tree structure, where each
 - ❖ **internal node** (non-leaf node) denotes a test on an attribute
 - ❖ **branch** represents an outcome of the test
 - ❖ **leaf node** (or *terminal node*) holds a class label
- The topmost node in a tree is the **root** node.



A decision tree for the concept *buys_computer*, indicating whether an *AllElectronics* customer is likely to purchase a computer. Each internal (nonleaf) node represents a test on an attribute. Each leaf node represents a class (either *buys_computer* = *yes* or *buys_computer* = *no*).

How are decision trees used for classification?

- Given a tuple, X , for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree.
- A path is traced from the root to a leaf node, which holds the class prediction for that tuple.
- Decision trees can easily be converted to classification rules.

Why are decision tree classifiers so popular?

- The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore is appropriate for exploratory knowledge discovery.
- Decision trees can handle multidimensional data.
- Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate.
- The learning and classification steps of decision tree induction are simple and fast.
- In general, decision tree classifiers have good accuracy.
- However, successful use may depend on the data at hand.
- Decision tree induction algorithms have been used for classification in many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology.
- Decision trees are the basis of several commercial rule induction systems.
- During tree construction, *attribute selection measures* are used to select the attribute that best partitions the tuples into distinct classes.
- When decision trees are built, many of the branches may reflect noise or outliers in the training data.
- *Tree pruning* attempts to identify and remove such branches, with the goal of improving classification accuracy on unseen data.
- Scalability issues arise for the induction of decision trees from large databases.

Decision Tree Induction

- ID3, C4.5, and CART adopt a greedy (i.e., nonbacktracking) approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner.
- Most algorithms for decision tree induction also follow a top-down approach, which starts with a training set of tuples and their associated class labels.
- The training set is recursively partitioned into smaller subsets as the tree is being built.

- The strategy is as follows.
 - ❖ The algorithm is called with three parameters: D , *attribute list*, and *Attribute selection method*.
 - ❖ We refer to D as a data partition. Initially, it is the complete set of training tuples and their associated class labels.
 - ❖ The parameter *attribute list* is a list of attributes describing the tuples.
 - ❖ *Attribute selection method* specifies a heuristic procedure for selecting the attribute that “best” discriminates the given tuples according to class. This procedure employs an attribute selection measure such as information gain or the Gini index.
 - ❖ Whether the tree is strictly binary is generally driven by the attribute selection measure.
 - ❖ Some attribute selection measures, such as the Gini index, enforce the resulting tree to be binary.
 - ❖ Others, like information gain, do not, therein allowing multiway splits (i.e., two or more branches to be grown from a node).
 - ❖ The tree starts as a single node, N , representing the training tuples in D (step 1). The partition of class-labeled training tuples at node N is the set of tuples that follow a path from the root of the tree to node N when being processed by the tree. This set is sometimes referred to in the literature as the *family* of tuples at node N .

Algorithm: Generate decision tree. Generate a decision tree from the training tuples of data partition, D .

Input:

- Data partition, D , which is a set of training tuples and their associated class labels;
- *attribute list*, the set of candidate attributes;
- *Attribute selection method*, a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes. This criterion consists of a *splitting attribute* and, possibly, either a *split-point* or *splitting subset*.

Output: A decision tree.

Method:

- (1) create a node N ;
- (2) **if** tuples in D are all of the same class, C , **then**
- (3) return N as a leaf node labeled with the class C ;
- (4) **if** *attribute list* is empty **then**
- (5) return N as a leaf node labeled with the majority class in D ; // majority voting
- (6) apply **Attribute selection method**(D , *attribute list*) to **find** the “best” *splitting criterion*;
- (7) label node N with *splitting criterion*;
- (8) **if** *splitting attribute* is discrete-valued **and**
 multiway splits allowed **then** // not restricted to binary trees
- (9) *attribute list* ← *attribute list* \square *splitting attribute*; // remove *splitting attribute*
- (10) **for each** outcome j of *splitting criterion*
 // partition the tuples and grow subtrees for each partition
- (11) let D_j be the set of data tuples in D satisfying outcome j ; // a partition
- (12) **if** D_j is empty **then**
- (13) attach a leaf labeled with the majority class in D to node N ;
- (14) **else** attach the node returned by **Generate decision tree**(D_j , *attribute list*) to node N ;
- endfor**
- (15) return N ;

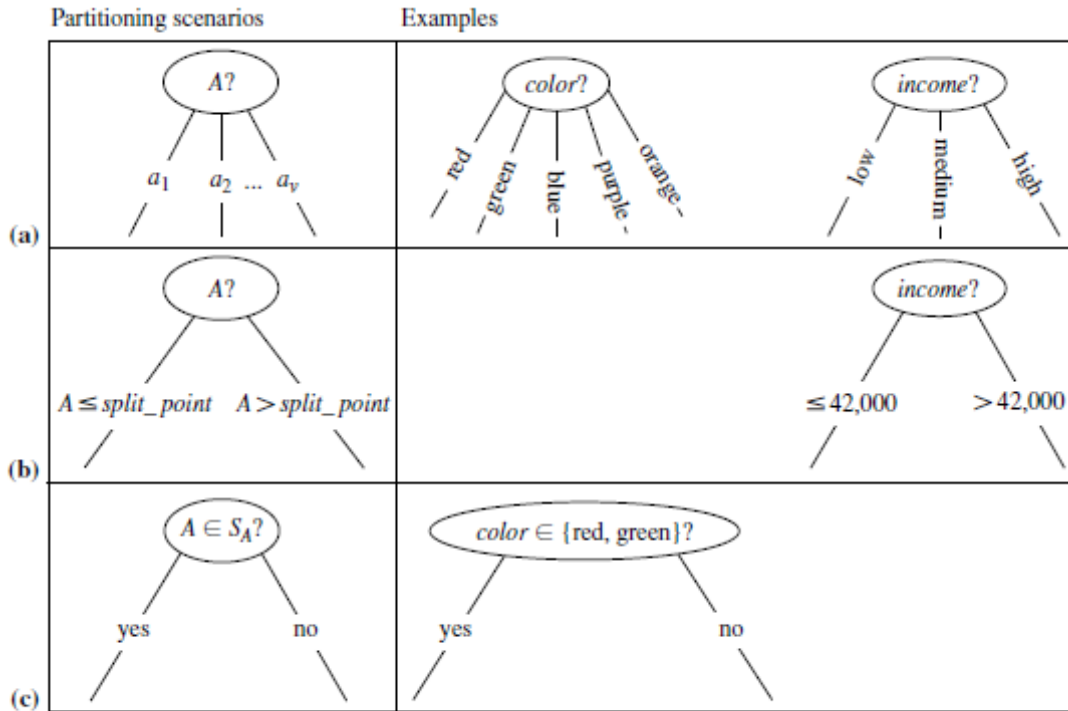
- If the tuples in D are all of the same class, then node N becomes a leaf and is labeled with that class (steps 2 and 3). Note that steps 4 and 5 are terminating conditions. All terminating conditions are explained at the end of the algorithm.
- Otherwise, the algorithm calls *Attribute selection method* to determine the **splitting criterion**. The splitting criterion tells us which attribute to test at node N by determining the “best” way to separate or partition the tuples in D into individual classes (step 6). The splitting criterion also tells us which branches to grow from node N with respect to the outcomes of the chosen test. More specifically, the splitting criterion indicates the **splitting attribute** and may also indicate either a **split-point** or

a **splitting subset**. The splitting criterion is determined so that, ideally, the resulting partitions at each branch are as “pure” as possible. A partition is **pure** if all the tuples in it belong to the same class. In other words, if we split up the tuples in D according to the mutually exclusive outcomes of the splitting criterion, we hope for the resulting partitions to be as pure as possible.

- The node N is labeled with the splitting criterion, which serves as a test at the node (step 7). A branch is grown from node N for each of the outcomes of the splitting criterion. The tuples in D are partitioned accordingly (steps 10 to 11). There are three possible scenarios. Let A be the splitting attribute. A has v distinct values, $[a_1, a_2, \dots, a_v]$, based on the training data.

1. *A is discrete-valued*: In this case, the outcomes of the test at node N correspond directly to the known values of A . A branch is created for each known value, a_j , of A and labeled with that value. Partition D_j is the subset of class-labeled tuples in D having value a_j of A . Because all the tuples in a given partition have the same value for A , A need not be considered in any future partitioning of the tuples. Therefore, it is removed from *attribute list* (steps 8 and 9).

2. *A is continuous-valued*: In this case, the test at node N has two possible outcomes, corresponding to the conditions $A \leq \textit{split point}$ and $A > \textit{split point}$, respectively where *split point* is the split-point returned by *Attribute selection method* as part of the splitting criterion. (In practice, the split-point, a , is often taken as the midpoint of two known adjacent values of A and therefore may not actually be a preexisting value of A from the training data.) Two branches are grown from N and labeled according to the previous outcomes. The tuples are partitioned such that D_1 holds the subset of class-labeled tuples in D for which $A \leq \textit{split point}$, while D_2 holds the rest.



This figure shows three possibilities for partitioning tuples based on the splitting criterion, each with examples. Let A be the splitting attribute. (a) If A is discrete-valued, then one branch is grown for each known value of A . (b) If A is continuous-valued, then two branches are grown, corresponding to $A \leq \textit{split_point}$ and $A > \textit{split_point}$. (c) If A is discrete-valued and a binary tree must be produced, then the test is of the form $A \in S_A$, where S_A is the splitting subset for A .

3. A is discrete-valued and a binary tree must be produced (as dictated by the attribute selection measure or algorithm being used): The test at node N is of the form “ $A \in S_A$?” where S_A is the splitting subset for A , returned by *Attribute selection method* as part of the splitting criterion. It is a subset of the known values of A . If a given tuple has value a_j of A and if $a_j \in S_A$, then the test at node N is satisfied. Two branches are grown from N . By convention, the left branch out of N is labeled *yes* so that D_1 corresponds to the subset of class-labeled tuples in D that satisfy the test. The right branch out of N is labeled *no* so that D_2 corresponds to the subset of class-labeled tuples from D that do not satisfy the test.

- The algorithm uses the same process recursively to form a decision tree for the tuples at each resulting partition, D_j , of D (step 14).

- The recursive partitioning stops only when any one of the following terminating conditions is true:
 1. All the tuples in partition D (represented at node N) belong to the same class (steps 2 and 3).
 2. There are no remaining attributes on which the tuples may be further partitioned (step 4). In this case, **majority voting** is employed (step 5). This involves converting node N into a leaf and labeling it with the most common class in D . Alternatively, the class distribution of the node tuples may be stored.
 3. There are no tuples for a given branch, that is, a partition D_j is empty (step 12). In this case, a leaf is created with the majority class in D (step 13).
 - The resulting decision tree is returned (step 15).

The computational complexity of the algorithm given training set D is $O(n \times |D| \times \log(|D|))$, where n is the number of attributes describing the tuples in D and $|D|$ is the number of training tuples in D . This means that the computational cost of growing a tree grows at most $n \times |D| \times \log(|D|)$ with $|D|$ tuples.

Attribute Selection Measures

- An **attribute selection measure** is a heuristic for selecting the splitting criterion that “best” separates a given data partition, D , of class-labeled training tuples into individual classes.
- If we were to split D into smaller partitions according to the outcomes of the splitting criterion, ideally each partition would be pure (i.e., all the tuples that fall into a given partition would belong to the same class).
- Conceptually, the “best” splitting criterion is the one that most closely results in such a scenario.
- Attribute selection measures are also known as **splitting rules** because they determine how the tuples at a given node are to be split.
- The attribute selection measure provides a ranking for each attribute describing the given training tuples.

- The attribute having the best score for the measure is chosen as the *splitting attribute* for the given tuples.
- If the splitting attribute is continuous-valued or if we are restricted to binary trees, then, respectively, either a *split point* or a *splitting subset* must also be determined as part of the splitting criterion.
- The tree node created for partition D is labeled with the splitting criterion, branches are grown for each outcome of the criterion, and the tuples are partitioned accordingly.
- Let D , the data partition, be a training set of class-labeled tuples.
- Suppose the class label attribute has m distinct values defining m distinct classes, C_i (for $i = 1, \dots, m$).
- Let $C_{i,D}$ be the set of tuples of class C_i in D .
- Let $|D|$ and $|C_{i,D}|$ denote the number of tuples in D and $C_{i,D}$, respectively.

Information Gain

- ID3(Iterative Dichotomiser) uses **information gain** as its attribute selection measure.
- Let node N represent or hold the tuples of partition D .
- The attribute with the highest information gain is chosen as the splitting attribute for node N .
- This attribute minimizes the information needed to classify the tuples in the resulting partitions and reflects the least randomness or “impurity” in these partitions.
- Such an approach minimizes the expected number of tests needed to classify a given tuple and guarantees that a simple (but not necessarily the simplest) tree is found.
- The expected information needed to classify a tuple in D is given by

$$Info(D) = - \sum_{i=1}^m p_i \log_2(p_i)$$

where p_i is the non-zero probability that an arbitrary tuple in D belongs to class C_i and is estimated by $|C_{i,D}|/|D|$. A log function to the base 2 is used, because the information

is encoded in bits.

- $Info(D)$ is just the average amount of information needed to identify the class label of a tuple in D .
- At this point, the information we have is based solely on the proportions of tuples of each class.
- $Info(D)$ is also known as the **entropy** of D .
- Now, suppose we were to partition the tuples in D on some attribute A having v distinct values, $\{a_1, a_2, \dots, a_v\}$, as observed from the training data.
- If A is discrete-valued, these values correspond directly to the v outcomes of a test on A .
- Attribute A can be used to split D into v partitions or subsets, $\{D_1, D_2, \dots, D_v\}$, where D_j contains those tuples in D that have outcome a_j of A .
- These partitions would correspond to the branches grown from node N .
- Ideally, we would like this partitioning to produce an exact classification of the tuples. That is, we would like for each partition to be pure. However, it is quite likely that the partitions will be impure (e.g., where a partition may contain a collection of tuples from different classes rather than from a single class).
- How much more information would we still need (after the partitioning) to arrive at an exact classification? This amount is measured by

$$Info_A(D) = \sum_{j=1}^v \frac{|D_j|}{|D|} \times Info(D_j)$$

- The term $|D_j|/|D|$ acts as the weight of the j^{th} partition.
- $Info_A(D)$ is the expected information required to classify a tuple from D based on the partitioning by A .
- The smaller the expected information (still) required, the greater the purity of the partitions.
- Information gain is defined as the difference between the original information requirement (i.e., based on just the proportion of classes) and the new requirement (i.e., obtained after partitioning on A). That is,

$$Gain(A) = Info(D) - Info_A(D)$$

- In other words, $Gain(A)$ tells us how much would be gained by branching on A .
- It is the expected reduction in the information requirement caused by knowing the value of A .
- The attribute A with the highest information gain, $Gain(A)$, is chosen as the splitting attribute at node N . This is equivalent to saying that we want to partition on the attribute A that would do the “best classification,” so that the amount of information still required to finish classifying the tuples is minimal (i.e., minimum $Info_A(D)$).

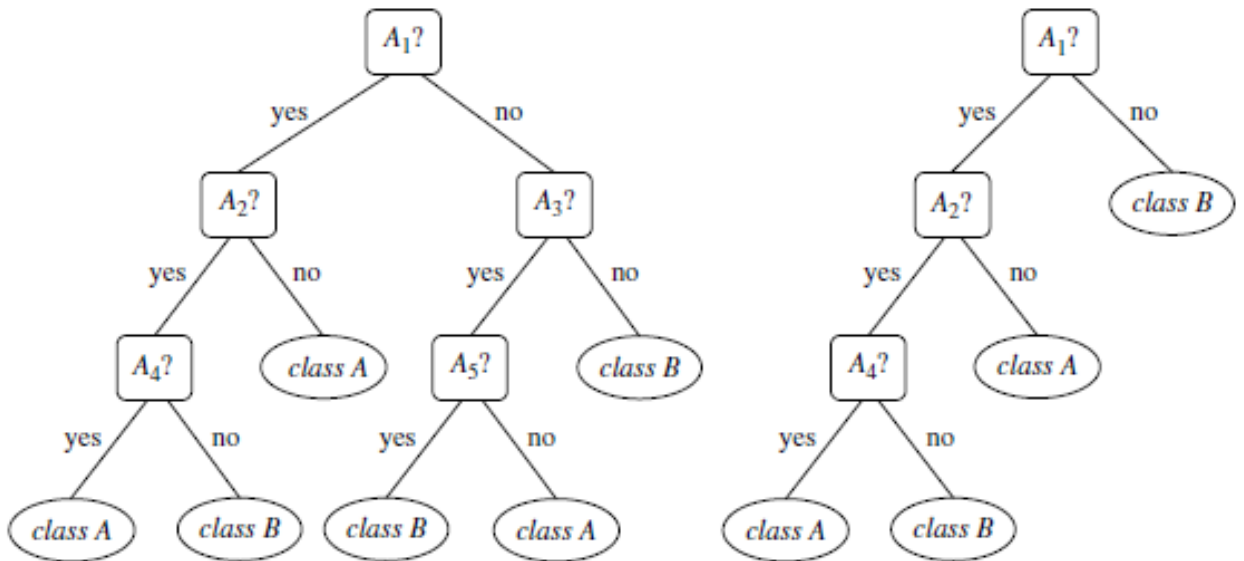
Tree Pruning

- When a decision tree is built, many of the branches will reflect anomalies in the training data due to noise or outliers.
- Tree pruning methods address this problem of *overfitting* the data. Such methods typically use statistical measures to remove the least-reliable branches.
- Pruned trees tend to be smaller and less complex and, thus, easier to comprehend.
- They are usually faster and better at correctly classifying independent test data (i.e., of previously unseen tuples) than unpruned trees.

How does tree pruning work?

- There are two common approaches to tree pruning: *prepruning* and *postpruning*.
- In the **prepruning** approach, a tree is “pruned” by halting its construction early (e.g., by deciding not to further split or partition the subset of training tuples at a given node).
- Upon halting, the node becomes a leaf. The leaf may hold the most frequent class among the subset tuples or the probability distribution of those tuples.
- When constructing a tree, measures such as information gain, can be used to assess the goodness of a split.
- If partitioning the tuples at a node would result in a split that falls below a pre-specified threshold, then further partitioning of the given subset is halted.

- There are difficulties, however, in choosing an appropriate threshold. High thresholds could result in oversimplified trees, whereas low thresholds could result in very little simplification.
- The second and more common approach is **postpruning**, which removes subtrees from a “fully grown” tree.
- A subtree at a given node is pruned by removing its branches and replacing it with a leaf.
- The leaf is labeled with the most frequent class among the subtree being replaced.



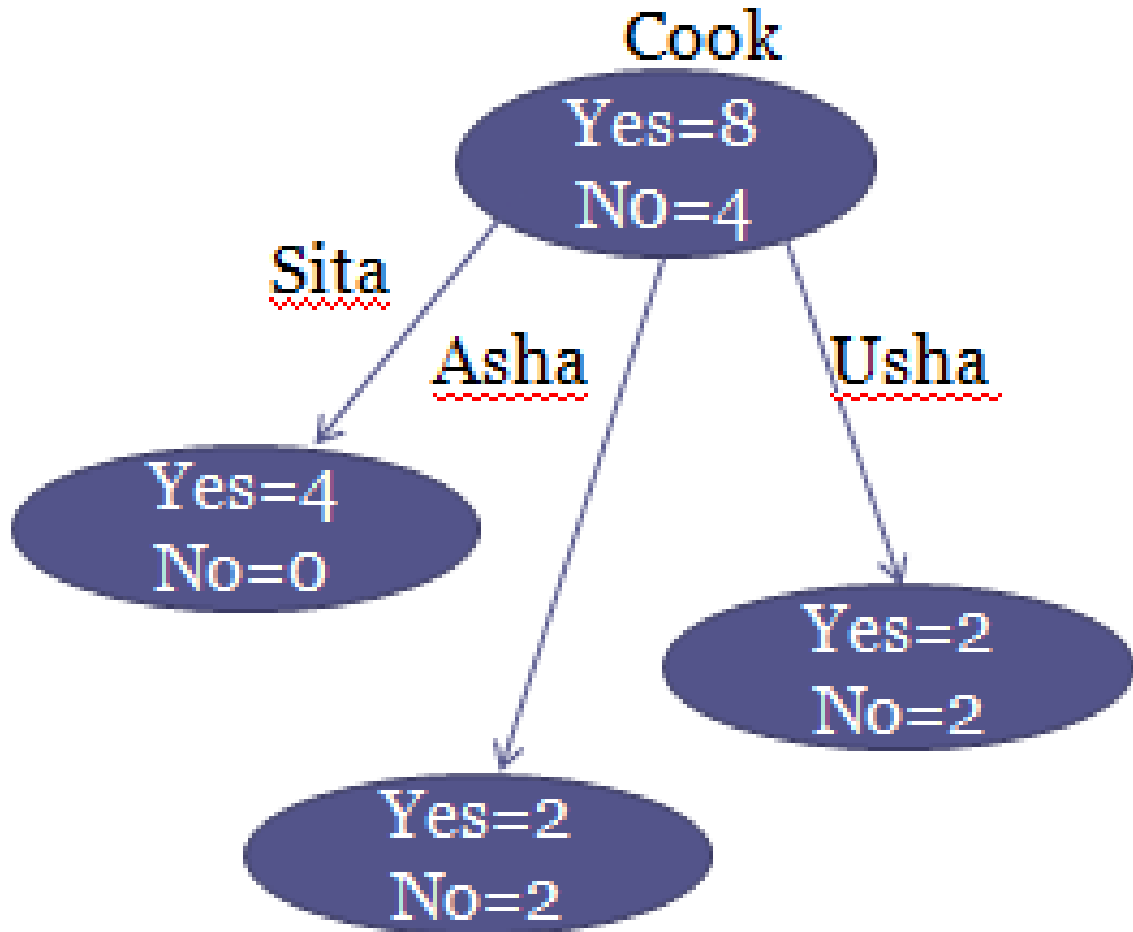
An unpruned decision tree and a pruned version of it.

Example of Decision Tree Induction

Training Set

| Cook | Mood | Cuisine | Tasty |
|------|------|-------------|-------|
| Sita | Bad | Indian | Yes |
| Sita | Good | Continental | Yes |
| Asha | Bad | Indian | No |
| Asha | Good | Indian | Yes |
| Usha | Bad | Indian | Yes |
| Usha | Bad | Continental | No |
| Asha | Bad | Continental | No |
| Asha | Good | Continental | Yes |
| Usha | Good | Indian | Yes |
| Usha | Good | Continental | No |
| Sita | Good | Indian | Yes |
| Sita | Bad | Continental | Yes |

- Entropy at Parent node Tasty
- $I(\text{Tasty}) = -4/12 \log_2 4/12 - 8/12 \log_2 8/12 = 1.8089$
- Consider the three attributes and find the attribute with the highest gain.



1. Cook

Cook = Sita

$$i(N_a) = -1\log 1 - 0\log 0 = 0$$

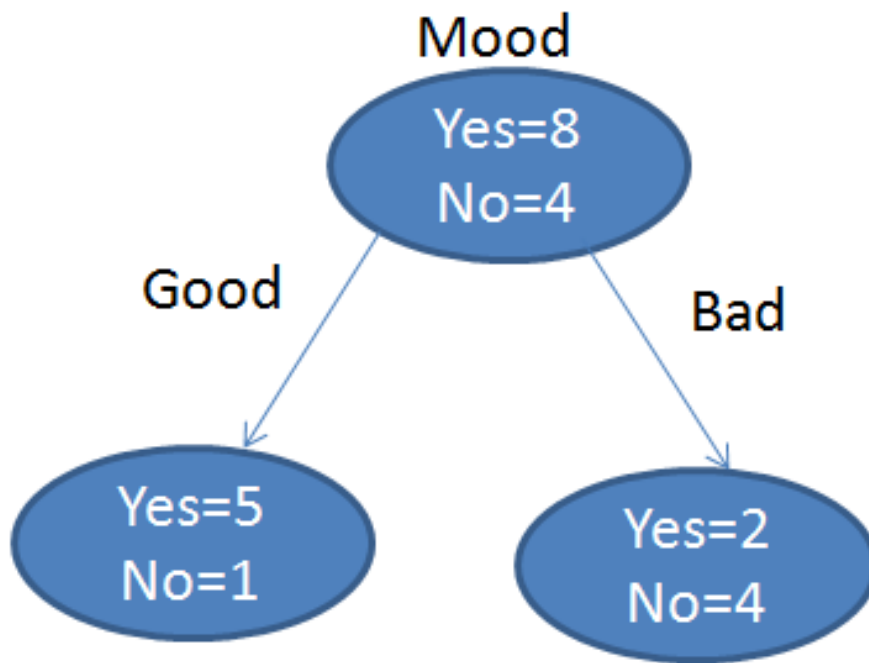
Cook = Asha

$$i(N_a) = -2/4\log 2/4 - 2/4\log 2/4 = 1$$

Cook = Usha

$$i(N_a) = -2/4\log 2/4 - 2/4\log 2/4 = 1$$

$$\text{Gain at Cook} = 1.8089 - 4/12 \times 1.0 - 4/12 \times 1.0 = 1.4223$$



2.Mood

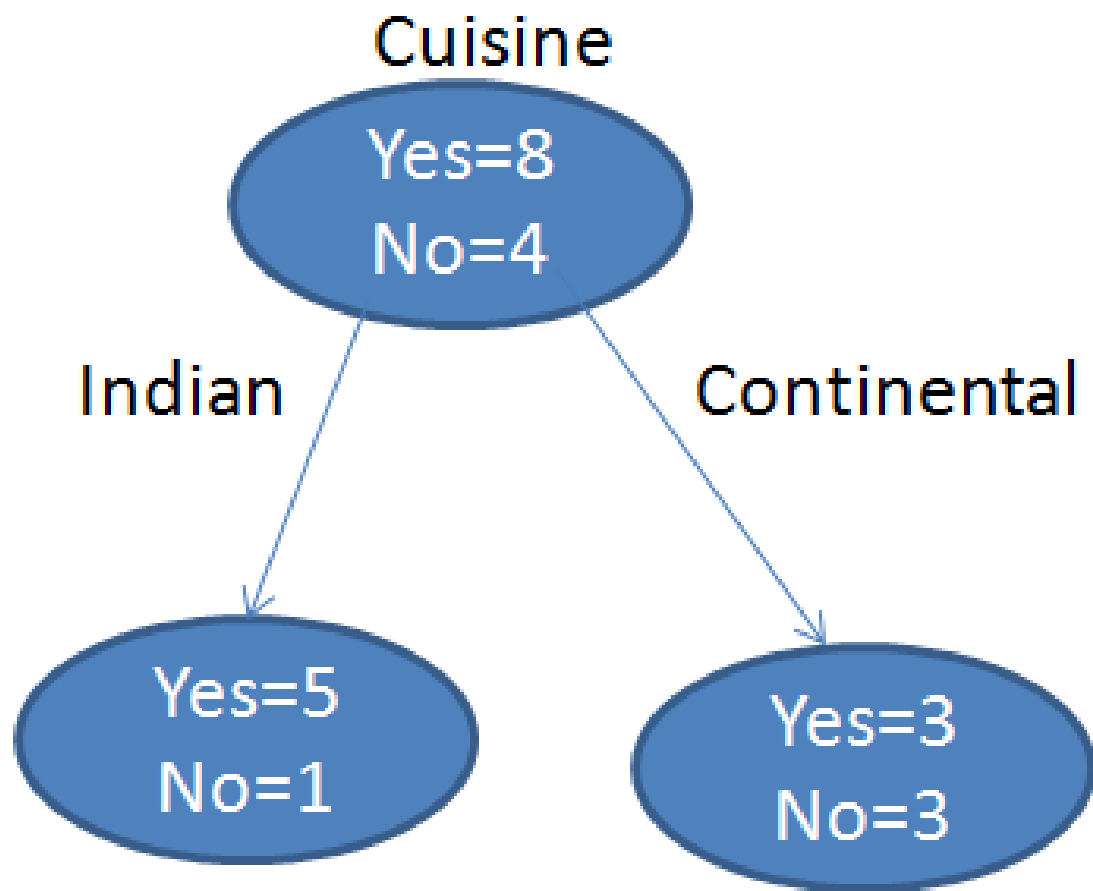
Mood= Bad

$$i(N_a) = -3/6 \log 3/6 - 3/6 \log 3/6 = 1.0$$

Mood = good

$$i(N_a) = -1/6 \log 1/6 - 5/6 \log 5/6 = 2.45$$

$$\text{Gain at Mood} = 1.8089 - 6/12 \times 2.45 - 6/12 \times 1.0 = 0.0795$$



3. Cuisine

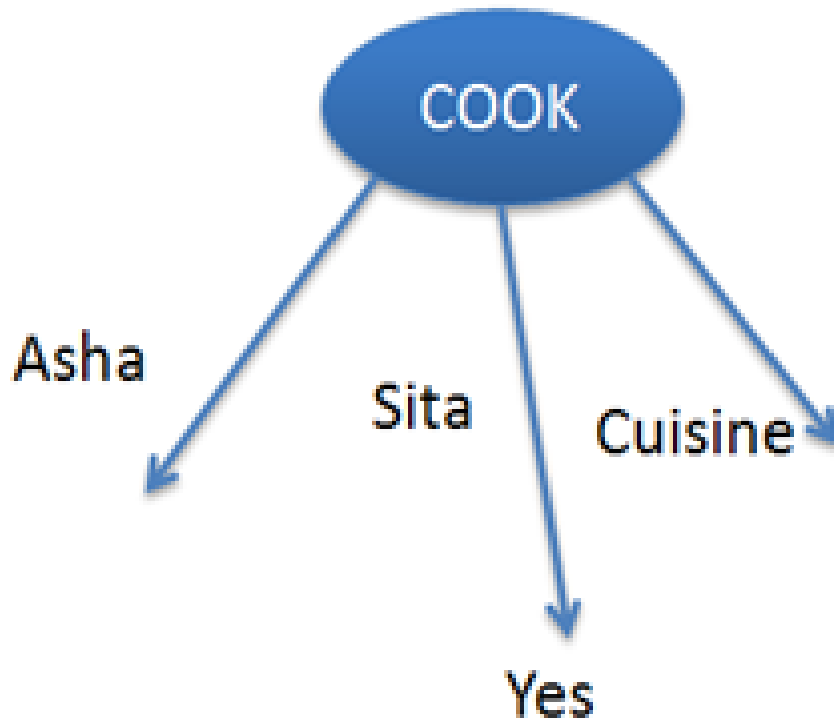
Cuisine= Indian

$$i(N_a) = -1/6 \log 1/6 - 5/6 \log 5/6 = 2.4$$

Cuisine = Continental

$$i(N_a) = -3/6 \log 3/6 - 3/6 \log 3/6 = 1.0$$

$$\text{Gain at Cuisine} = 1.8089 - 6/12 \times 2.45 - 6/12 \times 1.0 = 0.0795$$



Step 2: Choose Cook as a starting point

1. Cook = Sita

Branch has reached the leaf node.

2. Cook = Asha

Entropy at parent Cook = Asha

$$I(\text{Cook} = \text{Asha}) = -2/4 \log_2 2/4 - 2/4 \log_2 2/4 = 1.0$$

(a) Mood

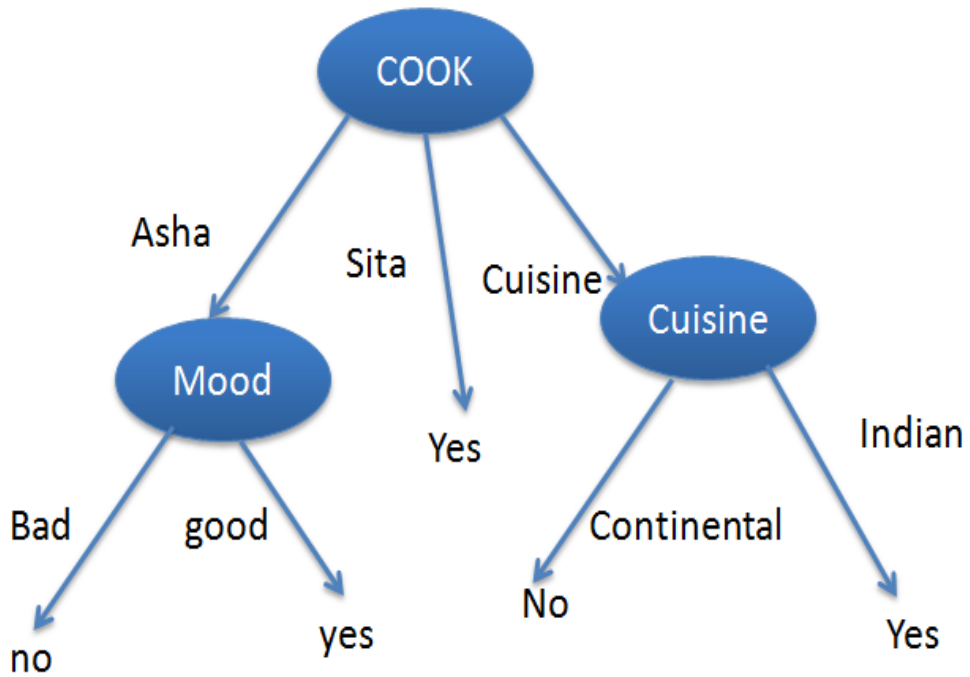
Mood = bad has 2 examples belonging to Tasty = no and 0 examples belonging to Tasty = yes, giving an entropy of 0. Mood = good has 2 examples belonging to Tasty = yes and 0 examples belonging to Tasty = no, giving entropy of 0. The gain for Mood is therefore 1.0

(b) Cuisine

$$\text{Gain}(\text{Cuisine}) = 1.0 - 2/4 \times 1.0 - 2/4 \times 1.0 = 0$$

Fix Mood as child of Cook

Final Tree



Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class.
- Stop expanding a node when all the records have similar attribute values.

Chapter-4 PERFORMANCE ANALYSIS

1. Analysis by Decision Trees

The following figures show the pruned J48 decision tree structure of the dataset:

WEKA Output

```
Classifier output
=== Run information ===

Scheme:          weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:        movies
Instances:       36543
Attributes:      5
                 year
                 length
                 votes
                 Genre
                 ranking
Test mode:       evaluate on training data

=== Classifier model (full training set) ===

J48 pruned tree
-----
Genre = Drama
|
| votes <= 237
| |
| | length <= 109
| | |
| | | year <= 1957
| | | |
| | | | votes <= 84
| | | | |
| | | | | year <= 1929
| | | | | |
| | | | | | votes <= 28: high (131.0/65.0)
| | | | | | votes > 28
| | | | | | |
| | | | | | | votes <= 51: medium (41.0/8.0)
| | | | | | | votes > 51
| | | | | | | |
| | | | | | | | length <= 93
| | | | | | | | |
| | | | | | | | | length <= 76: medium (5.0)
| | | | | | | | | length > 76
| | | | | | | | | |
| | | | | | | | | | length <= 85: high (3.0)
| | | | | | | | | |
| | | | | | | | | | length > 85
| | | | | | | | | | |
| | | | | | | | | | | votes <= 57: high (3.0/1.0)
| | | | | | | | | | | votes > 57: medium (6.0/1.0)
| | | | | | | | | | | length > 93: high (7.0)
| | | | | | | | | | |
| | | | | | | | | | | year > 1929: medium (2259.0/477.0)
| | | | | | | | | | |
| | | | | | | | | | | votes > 84
| | | | | | | | | | | |
| | | | | | | | | | | | length <= 73
| | | | | | | | | | | | |
| | | | | | | | | | | | | year <= 1929: high (12.0/2.0)
| | | | | | | | | | | | | year > 1929
| | | | | | | | | | | | | |
| | | | | | | | | | | | | | votes <= 127: medium (27.0/2.0)
| | | | | | | | | | | | | | votes > 127
| | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | votes <= 157: high (5.0/2.0)
| | | | | | | | | | | | | | | votes > 157
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | year <= 1953: medium (9.0/1.0)
| | | | | | | | | | | | | | | | year > 1953: low (2.0)
| | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | length > 73
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | votes <= 165
| | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | year <= 1928: high (13.0/3.0)
| | | | | | | | | | | | | | | | | | year > 1928: medium (281.0/108.0)
| | | | | | | | | | | | | | | | | | votes > 165
| | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | votes <= 204: high (45.0/10.0)
| | | | | | | | | | | | | | | | | | | votes > 204
| | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | year <= 1943: medium (14.0/2.0)
| | | | | | | | | | | | | | | | | | | | year > 1943: high (22.0/7.0)
| | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | year > 1957: medium (6696.0/2160.0)
| | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | length > 109
| | | | | | | | | | | | | | | | | | | | length <= 139: medium (1652.0/623.0)
| | | | | | | | | | | | | | | | | | | | length > 139
| | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | year <= 1997: high (259.0/93.0)
| | | | | | | | | | | | | | | | | | | | | year > 1997
| | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | year <= 2002
| | | | | | | | | | | | | | | | | | | | | | length <= 177: medium (30.0/4.0)
```



```

| | | | | | length > 177: high (11.0/4.0)
| | | | | | year > 2002: high (28.0/11.0)
| votes > 237
| | year <= 1965: high (597.0/133.0)
| | year > 1965
| | | votes <= 9116
| | | | length <= 130
| | | | | votes <= 803: medium (899.0/277.0)
| | | | | votes > 803
| | | | | year <= 1986: high (185.0/74.0)
| | | | | year > 1986
| | | | | | year <= 2002: medium (502.0/186.0)
| | | | | | year > 2002
| | | | | | | year <= 2003: high (39.0/18.0)
| | | | | | | year > 2003
| | | | | | | | year <= 2004
| | | | | | | | | votes <= 1619: medium (12.0/2.0)
| | | | | | | | | votes > 1619
| | | | | | | | | | length <= 112: medium (13.0/6.0)
| | | | | | | | | | length > 112: high (6.0)
| | | | | | | | | | year > 2004: medium (6.0/1.0)
| | | | | | | | | length > 130: high (204.0/72.0)
| | | | | | | | votes > 9116: high (211.0/40.0)
Genre = Documentary
| votes <= 223
| | year <= 2000
| | | votes <= 22
| | | | votes <= 8
| | | | | year <= 1986: medium (162.0/83.0)
| | | | | year > 1986: high (227.0/92.0)
| | | | votes > 8
| | | | | year <= 1962

```

```

| | | | | | votes <= 20: medium (47.0/15.0)
| | | | | | votes > 20: high (5.0)
| | | | | year > 1962
| | | | | | year <= 1994
| | | | | | | year <= 1985: medium (183.0/87.0)
| | | | | | | year > 1985
| | | | | | | | length <= 216: high (134.0/64.0)
| | | | | | | | length > 216: medium (6.0/1.0)
| | | | | | | year > 1994
| | | | | | | | year <= 1998
| | | | | | | | | year <= 1996: medium (51.0/23.0)
| | | | | | | | | year > 1996
| | | | | | | | | | year <= 1997
| | | | | | | | | | | length <= 173: low (33.0/16.0)
| | | | | | | | | | | length > 173: medium (3.0)
| | | | | | | | | | | year > 1997
| | | | | | | | | | | | votes <= 11
| | | | | | | | | | | | votes <= 10: high (7.0/3.0)
| | | | | | | | | | | | votes > 10: low (3.0)
| | | | | | | | | | | | votes > 11: medium (31.0/14.0)
| | | | | | | | year > 1998
| | | | | | | | | year <= 1999: medium (49.0/17.0)
| | | | | | | | | year > 1999
| | | | | | | | | | votes <= 15
| | | | | | | | | | | votes <= 9: medium (8.0/2.0)
| | | | | | | | | | | votes > 9
| | | | | | | | | | | | votes <= 14: high (19.0/5.0)
| | | | | | | | | | | | votes > 14
| | | | | | | | | | | | | length <= 91
| | | | | | | | | | | | | | length <= 84: high (3.0/1.0)
| | | | | | | | | | | | | | length > 84: medium (3.0)
| | | | | | | | | | | | | | length > 91: low (2.0/1.0)

```

```

| | | | | | | | | | votes > 15
| | | | | | | | | | | length <= 79: high (6.0)
| | | | | | | | | | | | length > 79
| | | | | | | | | | | | length <= 90: low (3.0)
| | | | | | | | | | | | | length > 90: high (3.0/1.0)
| | | | | | | | | | votes > 22
| | | | | | | | | | | year <= 1965: high (65.0/16.0)
| | | | | | | | | | | year > 1965
| | | | | | | | | | | | length <= 93
| | | | | | | | | | | | | votes <= 25: medium (52.0/11.0)
| | | | | | | | | | | | | votes > 25
| | | | | | | | | | | | | | length <= 60
| | | | | | | | | | | | | | | votes <= 28
| | | | | | | | | | | | | | | | votes <= 26
| | | | | | | | | | | | | | | | | length <= 56: high (5.0/1.0)
| | | | | | | | | | | | | | | | | length > 56: medium (2.0)
| | | | | | | | | | | | | | | | | votes > 26: medium (6.0)
| | | | | | | | | | | | | | | | | votes > 28: high (56.0/25.0)
| | | | | | | | | | | | | | | | | length > 60: medium (268.0/111.0)
| | | | | | | | | | | | | | | length > 93
| | | | | | | | | | | | | | | | length <= 109: medium (101.0/46.0)
| | | | | | | | | | | | | | | | | length > 109: high (63.0/20.0)
| | | | | | | | | | | year > 2000: high (795.0/319.0)
| | | | | | | | | | | votes > 223: high (192.0/41.0)
| | | | | | | | | | Genre = Comedy
| | | | | | | | | | | year <= 1960
| | | | | | | | | | | | votes <= 203
| | | | | | | | | | | | | votes <= 16
| | | | | | | | | | | | | | year <= 1929
| | | | | | | | | | | | | | | votes <= 5
| | | | | | | | | | | | | | | | length <= 67: high (2.0)
| | | | | | | | | | | | | | | | | length > 67: low (2.0)

```

```

| | | | | | | | | | | votes > 5
| | | | | | | | | | | | length <= 79
| | | | | | | | | | | | | year <= 1924
| | | | | | | | | | | | | | year <= 1920
| | | | | | | | | | | | | | | votes <= 12: high (3.0)
| | | | | | | | | | | | | | | votes > 12: medium (2.0)
| | | | | | | | | | | | | | | year > 1920: medium (5.0)
| | | | | | | | | | | | | | | year > 1924: high (17.0/4.0)
| | | | | | | | | | | | | | | length > 79
| | | | | | | | | | | | | | | | length <= 82: low (3.0/1.0)
| | | | | | | | | | | | | | | | | length > 82: medium (4.0/2.0)
| | | | | | | | | | | year > 1929
| | | | | | | | | | | | length <= 103: medium (774.0/192.0)
| | | | | | | | | | | | | length > 103
| | | | | | | | | | | | | | votes <= 13: medium (36.0/11.0)
| | | | | | | | | | | | | | votes > 13
| | | | | | | | | | | | | | | length <= 129: high (7.0/1.0)
| | | | | | | | | | | | | | | | length > 129: medium (2.0)
| | | | | | | | | | | votes > 16
| | | | | | | | | | | | year <= 1927
| | | | | | | | | | | | | year <= 1918: medium (4.0)
| | | | | | | | | | | | | year > 1918: high (27.0/9.0)
| | | | | | | | | | | | year > 1927
| | | | | | | | | | | | | votes <= 104: medium (999.0/161.0)
| | | | | | | | | | | | | | votes > 104
| | | | | | | | | | | | | | | year <= 1938
| | | | | | | | | | | | | | | | votes <= 192: high (29.0/11.0)
| | | | | | | | | | | | | | | | | votes > 192: medium (5.0/2.0)
| | | | | | | | | | | | | | | | | year > 1938: medium (114.0/17.0)
| | | | | | | | | | | votes > 203
| | | | | | | | | | | | year <= 1937: high (41.0/3.0)
| | | | | | | | | | | | year > 1937

```

```

| votes <= 783
|   year <= 1948
|     length <= 96
|       length <= 83
|         votes <= 446: high (16.0/3.0)
|         votes > 446: medium (5.0/1.0)
|       length > 83: medium (18.0/5.0)
|     length > 96: high (12.0/1.0)
|   year > 1948: medium (71.0/22.0)
| votes > 783
|   year <= 1955: high (19.0)
|   year > 1955
|     votes <= 1879: high (4.0)
|     votes > 1879: medium (5.0/1.0)
year > 1960
  votes <= 15
    year <= 2000
      length <= 88
        votes <= 10
          votes <= 8
            votes <= 7
              votes <= 6
                length <= 85
                  length <= 83: medium (60.0/34.0)
                  length > 83: high (25.0/11.0)
                  length > 85: medium (31.0/14.0)
                votes > 6
                  year <= 1968
                    year <= 1967: high (15.0/8.0)
                    year > 1967: low (2.0)
                  year > 1968
                    length <= 83: medium (23.0/5.0)

```

```

|   length > 83
|     year <= 1976
|       length <= 85: low (2.0)
|       length > 85: medium (3.0/1.0)
|     year > 1976
|       year <= 1989: high (4.0/1.0)
|       year > 1989: low (2.0)
| votes > 7
|   year <= 1982: low (33.0/19.0)
|   year > 1982: high (13.0/6.0)
| votes > 8
|   length <= 70: low (9.0/1.0)
|   length > 70
|     year <= 1965
|       length <= 80: medium (4.0)
|       length > 80: high (5.0/1.0)
|     year > 1965
|       length <= 81
|         length <= 77: low (5.0/1.0)
|         length > 77
|           year <= 1982: high (4.0)
|           year > 1982
|             year <= 1996: low (7.0/1.0)
|             year > 1996: high (2.0)
|       length > 81
|         votes <= 9
|           year <= 1975
|             length <= 83: medium (5.0/1.0)
|             length > 83
|               length <= 84: high (2.0)
|               length > 84: low (4.0/1.0)
|           year > 1975

```

```

      year <= 1982: low (3.0)
      year > 1982
      | length <= 86: low (6.0/2.0)
      | length > 86: medium (4.0)
      votes > 9
      | length <= 86: low (16.0/8.0)
      | length > 86
      | length <= 87: medium (3.0/1.0)
      | length > 87: high (3.0/1.0)
votes > 10
  length <= 78
  | length <= 61: high (3.0)
  | length > 61
  | year <= 1968
  | | length <= 76: low (6.0/1.0)
  | | length > 76: high (2.0)
  | year > 1968
  | | year <= 1990
  | | | length <= 74: medium (6.0/1.0)
  | | | length > 74: high (6.0/3.0)
  | | year > 1990
  | | | year <= 1998: low (5.0)
  | | | year > 1998: medium (3.0)
  length > 78
  | votes <= 11
  | | year <= 1966: medium (4.0)
  | | year > 1966: low (30.0/13.0)
  | votes > 11
  | | length <= 79: low (3.0)
  | | length > 79: medium (80.0/41.0)
length > 88: medium (772.0/320.0)
year > 2000

```

```

votes <= 5
  | length <= 89: high (13.0/2.0)
  | length > 89: medium (15.0/5.0)
votes > 5
  votes <= 11
  | length <= 97: high (87.0/38.0)
  | length > 97
  | | length <= 103: medium (11.0/1.0)
  | | length > 103
  | | | year <= 2003
  | | | | votes <= 7
  | | | | | length <= 109: medium (3.0)
  | | | | | length > 109: low (4.0/2.0)
  | | | | votes > 7: high (5.0/2.0)
  | | | year > 2003
  | | | | votes <= 8: high (4.0/1.0)
  | | | | votes > 8: medium (3.0/1.0)
votes > 11
  year <= 2002: medium (19.0/5.0)
  year > 2002
  | year <= 2003
  | | votes <= 14
  | | | length <= 89
  | | | | length <= 82: medium (4.0/1.0)
  | | | | | length > 82: high (2.0)
  | | | | length > 89: medium (5.0/2.0)
  | | | votes > 14: high (2.0)
  year > 2003
  | length <= 101
  | | votes <= 13
  | | | votes <= 12: high (2.0)
  | | | votes > 12: low (5.0/2.0)

```



```

| | | | | | | | | | length > 105: medium (23.0/3.0)
| | | | | | | | | | length > 109
| | | | | | | | | | year <= 1981: high (3.0)
| | | | | | | | | | year > 1981
| | | | | | | | | | | length <= 112
| | | | | | | | | | | | length <= 111
| | | | | | | | | | | | | votes <= 89: medium (15.0/2.0)
| | | | | | | | | | | | | votes > 89: low (3.0)
| | | | | | | | | | | | | length > 111: low (3.0/1.0)
| | | | | | | | | | | | | length > 112
| | | | | | | | | | | | | votes <= 26
| | | | | | | | | | | | | year <= 1993: high (5.0/1.0)
| | | | | | | | | | | | | year > 1993: medium (2.0/1.0)
| | | | | | | | | | | | | votes > 26
| | | | | | | | | | | | | year <= 1994: medium (24.0/5.0)
| | | | | | | | | | | | | year > 1994: high (3.0/1.0)
| | | | | | | | | | | | | year > 1995: medium (124.0/24.0)
| | | | | | | | | | | | | length > 136
| | | | | | | | | | | | | year <= 1980: high (5.0)
| | | | | | | | | | | | | year > 1980
| | | | | | | | | | | | | length <= 143: medium (6.0)
| | | | | | | | | | | | | length > 143
| | | | | | | | | | | | | length <= 169
| | | | | | | | | | | | | length <= 158
| | | | | | | | | | | | | length <= 151: high (2.0)
| | | | | | | | | | | | | length > 151: medium (3.0)
| | | | | | | | | | | | | length > 158: high (3.0)
| | | | | | | | | | | | | length > 169: medium (4.0)
| | | | | | | | | | | | | votes > 131
| | | | | | | | | | | | | votes <= 16333: medium (508.0/132.0)
| | | | | | | | | | | | | votes > 16333
| | | | | | | | | | | | | votes <= 24791

```

```

| | | | | | | | | | votes <= 20966: high (6.0)
| | | | | | | | | | votes > 20966: medium (7.0/1.0)
| | | | | | | | | | votes > 24791: high (9.0)
Genre = Comedy|Drama
| | | | | | | | | | votes <= 156
| | | | | | | | | | | year <= 1928: high (33.0/9.0)
| | | | | | | | | | | year > 1928
| | | | | | | | | | | | votes <= 9
| | | | | | | | | | | | | year <= 1995
| | | | | | | | | | | | | votes <= 5
| | | | | | | | | | | | | length <= 88
| | | | | | | | | | | | | | year <= 1934: low (3.0/1.0)
| | | | | | | | | | | | | | year > 1934: medium (9.0)
| | | | | | | | | | | | | | length > 88: high (18.0/6.0)
| | | | | | | | | | | | | | votes > 5: medium (92.0/32.0)
| | | | | | | | | | | | | | year > 1995
| | | | | | | | | | | | | | length <= 98: high (44.0/20.0)
| | | | | | | | | | | | | | length > 98: medium (16.0/7.0)
| | | | | | | | | | | | | | votes > 9: medium (1129.0/279.0)
| | | | | | | | | | | | | | votes > 156
| | | | | | | | | | | | | | votes <= 12681
| | | | | | | | | | | | | | year <= 1968
| | | | | | | | | | | | | | | votes <= 643: medium (63.0/30.0)
| | | | | | | | | | | | | | | votes > 643: high (28.0/1.0)
| | | | | | | | | | | | | | | year > 1968
| | | | | | | | | | | | | | | year <= 1999
| | | | | | | | | | | | | | | votes <= 2806: medium (374.0/83.0)
| | | | | | | | | | | | | | | votes > 2806
| | | | | | | | | | | | | | | | year <= 1995: high (60.0/23.0)
| | | | | | | | | | | | | | | | year > 1995
| | | | | | | | | | | | | | | | | votes <= 2890: high (2.0)
| | | | | | | | | | | | | | | | | votes > 2890: medium (37.0/11.0)

```

```

| | | | | year > 1999
| | | | | | length <= 113: medium (176.0/60.0)
| | | | | | length > 113: high (41.0/13.0)
| | | | | votes > 12681: high (39.0/3.0)
Genre = Action|Drama
| | | | | length <= 106
| | | | | | votes <= 425: medium (946.0/262.0)
| | | | | | votes > 425
| | | | | | | year <= 1971
| | | | | | | | length <= 92: high (8.0/1.0)
| | | | | | | | length > 92
| | | | | | | | | length <= 103: medium (13.0/5.0)
| | | | | | | | | length > 103: high (4.0)
| | | | | | | year > 1971: medium (122.0/28.0)
| | | | | length > 106
| | | | | | votes <= 29762
| | | | | | | year <= 1965
| | | | | | | | votes <= 2266
| | | | | | | | | votes <= 10
| | | | | | | | | | votes <= 5: medium (4.0/2.0)
| | | | | | | | | | votes > 5: high (7.0)
| | | | | | | | | votes > 10: medium (40.0/17.0)
| | | | | | | | votes > 2266: high (11.0)
| | | | | | | year > 1965: medium (352.0/97.0)
| | | | | | votes > 29762: high (20.0)
Genre = Action
| | | | | length <= 107
| | | | | | year <= 1981: medium (438.0/139.0)
| | | | | | year > 1981
| | | | | | | votes <= 261
| | | | | | | | votes <= 9
| | | | | | | | length <= 81: high (10.0/4.0)

```

```

| | | | | | length > 81
| | | | | | | votes <= 6
| | | | | | | | year <= 2000: low (43.0/20.0)
| | | | | | | | year > 2000
| | | | | | | | | length <= 87: low (3.0/1.0)
| | | | | | | | | length > 87: high (4.0/1.0)
| | | | | | | votes > 6
| | | | | | | | length <= 94
| | | | | | | | | length <= 85: low (5.0/2.0)
| | | | | | | | | length > 85
| | | | | | | | | | year <= 1992: medium (15.0/3.0)
| | | | | | | | | | year > 1992: high (7.0/3.0)
| | | | | | | | length > 94
| | | | | | | | | year <= 1989: low (6.0/2.0)
| | | | | | | | | year > 1989: medium (7.0/1.0)
| | | | | votes > 9
| | | | | | length <= 99
| | | | | | | year <= 2001
| | | | | | | | length <= 87
| | | | | | | | | year <= 1998: low (179.0/63.0)
| | | | | | | | | year > 1998: medium (13.0/4.0)
| | | | | | | | length > 87
| | | | | | | | | length <= 90: medium (172.0/82.0)
| | | | | | | | | length > 90: low (336.0/153.0)
| | | | | | | year > 2001
| | | | | | | | length <= 77
| | | | | | | | | votes <= 23: medium (3.0)
| | | | | | | | | votes > 23: high (2.0)
| | | | | | | | length > 77
| | | | | | | | | votes <= 19
| | | | | | | | | | year <= 2003: low (2.0)
| | | | | | | | | | year > 2003: high (3.0)

```



```

Genre = Multi-genre
|   votes <= 13542
|   |   year <= 1949
|   |   |   votes <= 472
|   |   |   |   year <= 1929: high (7.0/1.0)
|   |   |   |   year > 1929: medium (85.0/13.0)
|   |   |   |   votes > 472: high (19.0)
|   |   |   year > 1949
|   |   |   |   year <= 1967
|   |   |   |   |   votes <= 1692: medium (82.0/24.0)
|   |   |   |   |   votes > 1692: high (10.0)
|   |   |   |   |   year > 1967: medium (704.0/201.0)
|   |   |   votes > 13542: high (33.0/6.0)
Genre = Romance
|   length <= 97
|   |   length <= 73
|   |   |   votes <= 5: low (3.0/1.0)
|   |   |   |   votes > 5
|   |   |   |   |   votes <= 181: medium (36.0/13.0)
|   |   |   |   |   votes > 181: high (4.0)
|   |   |   |   length > 73: medium (260.0/77.0)
|   |   length > 97
|   |   |   length <= 126
|   |   |   |   votes <= 1718: medium (153.0/45.0)
|   |   |   |   |   votes > 1718
|   |   |   |   |   |   year <= 1970: high (9.0)
|   |   |   |   |   |   |   year > 1970
|   |   |   |   |   |   |   |   votes <= 6624: medium (4.0)
|   |   |   |   |   |   |   |   |   votes > 6624: high (3.0/1.0)
|   |   |   |   length > 126
|   |   |   |   |   year <= 1990: high (47.0/15.0)
|   |   |   |   |   year > 1990

```

```

|   |   |   |   |   year <= 1999: medium (8.0)
|   |   |   |   |   year > 1999
|   |   |   |   |   |   length <= 133: high (2.0)
|   |   |   |   |   |   |   length > 133
|   |   |   |   |   |   |   |   votes <= 29: high (3.0/1.0)
|   |   |   |   |   |   |   |   |   votes > 29: medium (5.0)

```

```

Genre = Animation
|   votes <= 6530
|   |   year <= 1985
|   |   |   length <= 68
|   |   |   |   year <= 1981
|   |   |   |   |   length <= 47: medium (3.0/1.0)
|   |   |   |   |   length > 47: high (17.0)
|   |   |   |   |   year > 1981
|   |   |   |   |   |   year <= 1983
|   |   |   |   |   |   |   votes <= 11: low (3.0/1.0)
|   |   |   |   |   |   |   |   votes > 11: medium (4.0/1.0)
|   |   |   |   |   |   |   |   |   year > 1983: high (6.0/1.0)
|   |   |   |   length > 68
|   |   |   |   |   votes <= 9
|   |   |   |   |   |   length <= 79: medium (4.0/1.0)
|   |   |   |   |   |   |   length > 79: high (16.0/2.0)
|   |   |   |   |   |   |   votes > 9
|   |   |   |   |   |   |   |   votes <= 3265: medium (133.0/42.0)
|   |   |   |   |   |   |   |   |   votes > 3265
|   |   |   |   |   |   |   |   |   |   length <= 87: high (9.0)
|   |   |   |   |   |   |   |   |   |   |   length > 87: medium (2.0)
|   |   |   |   |   |   |   |   |   |   |   |   year > 1985: medium (333.0/91.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   votes > 6530
|   |   |   |   |   |   |   |   |   |   |   |   |   |   votes <= 17900
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   year <= 1988: high (10.0)
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   year > 1988

```

```

| | | | | length <= 89
| | | | | | year <= 1997: medium (3.0/1.0)
| | | | | | year > 1997: high (7.0)
| | | | | length > 89: medium (6.0/1.0)
| | | | | votes > 17900: high (18.0)
Genre = ActCom
| | | | | length <= 97
| | | | | | votes <= 2694
| | | | | | | length <= 85
| | | | | | | | year <= 1995
| | | | | | | | | year <= 1962: medium (4.0)
| | | | | | | | | year > 1962
| | | | | | | | | | votes <= 35: low (15.0/1.0)
| | | | | | | | | | votes > 35
| | | | | | | | | | | votes <= 103: medium (7.0/1.0)
| | | | | | | | | | | votes > 103: low (11.0/3.0)
| | | | | | | | | | | year > 1995
| | | | | | | | | | | length <= 83
| | | | | | | | | | | | year <= 2003: medium (12.0/4.0)
| | | | | | | | | | | | year > 2003: high (3.0)
| | | | | | | | | | | length > 83: medium (9.0/2.0)
| | | | | | | | | | | length > 85: medium (195.0/61.0)
| | | | | | | | | | | votes > 2694: medium (49.0/7.0)
| | | | | | | | | | | length > 97
| | | | | | | | | | | | year <= 1964: high (8.0/2.0)
| | | | | | | | | | | | year > 1964
| | | | | | | | | | | | | votes <= 20
| | | | | | | | | | | | | | length <= 117
| | | | | | | | | | | | | | | year <= 1983: medium (4.0)
| | | | | | | | | | | | | | | year > 1983
| | | | | | | | | | | | | | | | year <= 1991: high (5.0/1.0)
| | | | | | | | | | | | | | | | year > 1991

```

```

| | | | | | | | | | | length <= 108
| | | | | | | | | | | | votes <= 10
| | | | | | | | | | | | | year <= 2002: medium (3.0)
| | | | | | | | | | | | | year > 2002: high (2.0)
| | | | | | | | | | | | | | votes > 10: low (3.0/1.0)
| | | | | | | | | | | | | length > 108: medium (3.0)
| | | | | | | | | | | | | length > 117: high (4.0)
| | | | | | | | | | | | | votes > 20: medium (196.0/40.0)
Genre = Action|Romance
| | | | | | | | | | | length <= 143: medium (69.0/27.0)
| | | | | | | | | | | length > 143: high (4.0/1.0)

```

Number of Leaves : 339

Size of the tree : 666

=== Evaluation on training set ===

=== Summary ===

| | | |
|------------------------------------|-----------|-----------|
| Correctly Classified Instances | 25656 | 70.2077 % |
| Incorrectly Classified Instances | 10887 | 29.7923 % |
| Kappa statistic | 0.2993 | |
| Mean absolute error | 0.2874 | |
| Root mean squared error | 0.3791 | |
| Relative absolute error | 83.92 % | |
| Root relative squared error | 91.609 % | |
| Coverage of cases (0.95 level) | 99.138 % | |
| Mean rel. region size (0.95 level) | 84.4959 % | |
| Total Number of Instances | 36543 | |

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---------------|---------|---------|-----------|--------|-----------|----------|--------|
| | 0.929 | 0.685 | 0.706 | 0.929 | 0.802 | 0.689 | medium |
| | 0.15 | 0.01 | 0.62 | 0.15 | 0.242 | 0.794 | low |
| | 0.359 | 0.056 | 0.693 | 0.359 | 0.474 | 0.749 | high |
| Weighted Avg. | 0.702 | 0.453 | 0.694 | 0.702 | 0.66 | 0.716 | |

=== Confusion Matrix ===

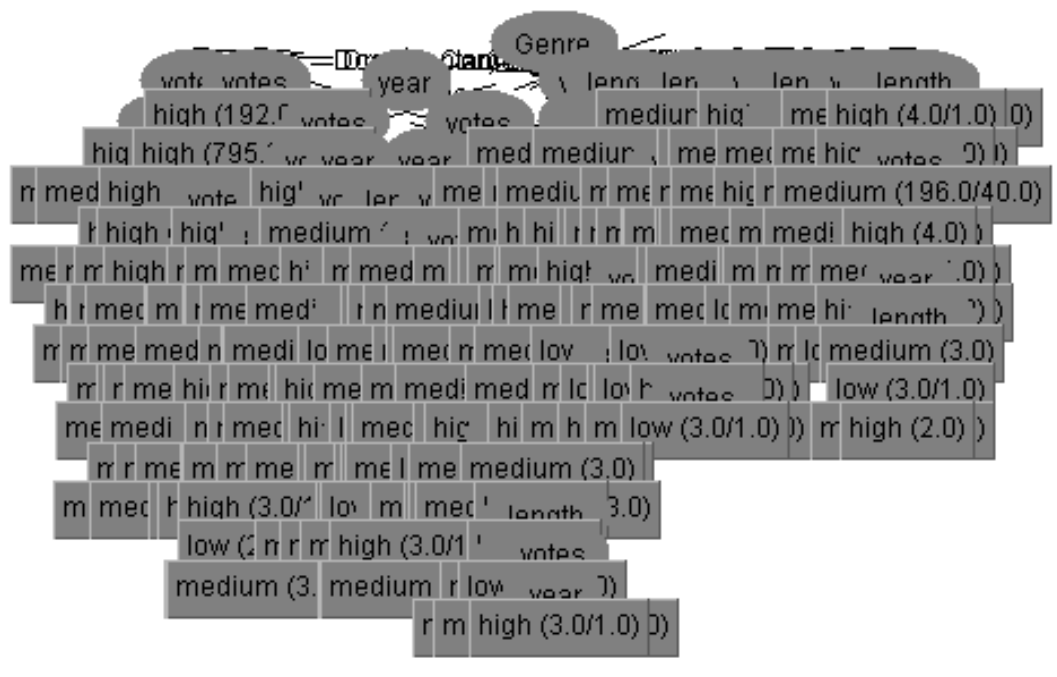
| a | b | c | <-- classified as |
|-------|-----|------|-------------------|
| 21683 | 301 | 1368 | a = medium |
| 2979 | 552 | 144 | b = low |
| 6058 | 37 | 3421 | c = high |

The correctly classified instances are 70.2077%

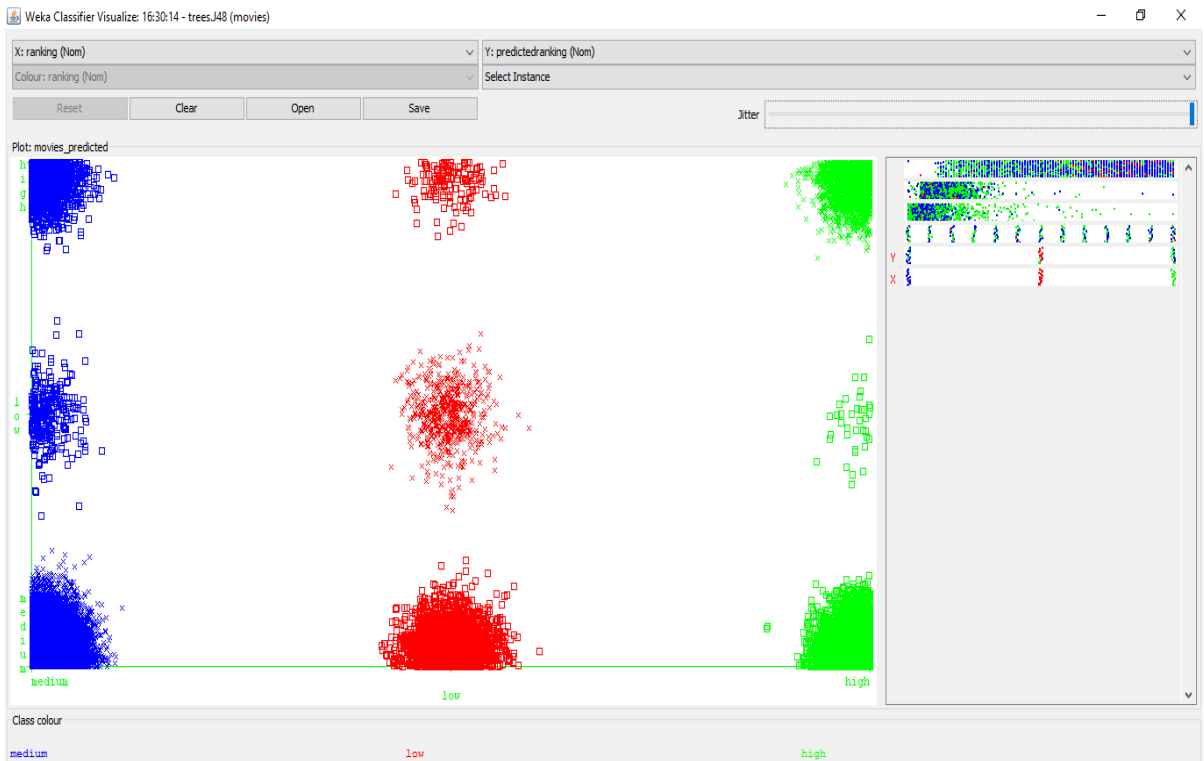
The incorrectly classified instances are 29.7923%

The error rate is high due to the high complexity of the dataset.

Tree View



Classifier error



Java Output

```
run:
Time to construct decision tree = 6054 ms
Target attribute = ranking
Other attributes = year length votes Genre
DECISION TREE
  votes->
    21305
      =medium
    11988
      =high
    2306
      =medium
    2305
      =medium
    34623
      =high
    2303
      Genre->
        Action
          =low
        Comedy|Drama
          =medium
    2302
      =high
    2301
      =medium
    2300
      =high
    4961
      Genre->
        Comedy|Drama
          =high
        Drama
          =medium
    3630
      =medium
  .
  .
  .
  .
```

```
.  
.  
    =medium  
3613  
    =medium  
3610  
    =medium  
15326  
    =medium  
4942  
    =medium  
3619  
    =high  
4943  
    =medium  
3618  
    length->  
    108  
        =high  
    93  
        =medium  
    97  
        =high  
15333  
    =medium  
4958  
    =high  
3622  
    =high  
3620  
    =medium  
4953  
    =medium  
4954  
    =medium
```

```
The class that is predicted is: null  
BUILD SUCCESSFUL (total time: 20 seconds)
```

Chapter-5 CONCLUSIONS

5.1 Conclusions

- The prediction model arrived at provides some key insights on the palate of the audience of a movie and gives a sense for what governs the movie ranking on IMDb.
- Genre related predictors: While slapstick comedy alone doesn't improve a movie's rank, movies with comedy laced with romance (RomCom) have a higher rank on average. Given the number of movies with comedy as a central theme, this result could serve as a useful way to differentiate a movie from its competition and gain the popular vote. Further, masala movies hold their own with a higher rank amongst other Indian movies.
- It can also be seen that better ranked movies have higher votes.

5.2 Future Scope

- Data mining is a rapidly evolving field, with new problems continually arising, and old problems being looked at in the light of new developments.
- These developments pose new challenges in the areas of data structures and algorithms.
- Some of the most promising areas in current data mining research include multi-relational data mining, mining streams of data, privacy preserving data mining, and mining data with complicated structures or behaviors.

5.3 Applications Contributions

- Various stakeholders such as actors, financiers, directors etc. can use these predictions to make more informed decisions.
- Further, a DVD rental agency or a distribution house could use these predictions to determine which titles to stock or promote respectively.

REFERENCES

- [1] <http://www.themovingarts.com/50000-movies-are-made-every-year-is-that-too-many/>, Accessed on Dec 18, 2015
- [2] <http://www.galitshmueli.com/data-mining-project/predicting-indian-movie-ratings-imdb>, Accessed on Dec 18, 2015
- [3] <https://vincentarelbundock.github.io/Rdatasets/datasets.html>, Accessed on Dec 6, 2015
- [4] <http://www.imdb.com/pressroom/>, Accessed on Dec 16, 2015
- [5] http://www.imdb.com/help/show_leaf?votes, Accessed on Dec 16, 2015
- [6] http://www.imdb.com/help/show_leaf?infosource, Accessed on Dec 16, 2015
- [7] <https://vincentarelbundock.github.io/Rdatasets/doc/ggplot2/movies.html>, Accessed on Dec 6, 2015
- [8] <http://www.cse.msu.edu/~ptan/papers/dshandbook.pdf>, Accessed on Dec 18, 2015
- [9] <https://www.youtube.com/watch?v=m7kpIBGEdkI>, Accessed on Dec 19, 2015
- [10] Svetlana S. Aksenova, "Machine Learning with WEKA-WEKA Explorer Tutorial for WEKA Version 3.4.3", 2004
- [11] Jiawei Han & Micheline Kamber, "Data Mining Concepts and Techniques", 3rd Edition, 2012

APPENDICES

Confusion matrix (evaluation of classifiers): example

Consider a study evaluating a new test that screens people for a disease. Each person taking the test either has or does not have the disease. The test outcome can be:

- **positive** (predicting that the person has the disease) or
- **negative** (predicting that the person does not have the disease)

Confusion matrix in this setting produces four different outcomes:

- **True positive:** Sick people correctly diagnosed as sick
- **False positive:** Healthy people incorrectly identified as sick
- **True negative:** Healthy people correctly identified as healthy
- **False negative:** Sick people incorrectly identified as healthy

Given training set on medical data set, let function F is learnt

- Our test set consists of four data points
 - ❖ $(z_1, 1)$,
 - ❖ $(z_2, -1)$,
 - ❖ $(z_3, 1)$,
 - ❖ $(z_4, -1)$
 - ❖ $(z_5, 1)$

Where,

Class label 1 : sick people

Class label -1 : healthy people

- We apply F on the four data points (without labels) and we get
 - ❖ $F(z_1) = 1$,
 - ❖ $F(z_2) = 1$,
 - ❖ $F(z_3) = -1$
 - ❖ $F(z_4) = -1$
 - ❖ $F(z_5) = 1$

- Then F correctly classified z_1 , z_4 and z_5 but incorrectly classified z_2 and z_3 .
- $TP = 2$ $FP = 1$ $TN = 1$ $FN = 1$

Based on TP, FP, TN and FN we measure performance of classifier.

Confusion matrix (example)

Let the number of test samples be N

| | Actual Label (1) | Actual Label (-1) |
|----------------------|----------------------|---------------------|
| Predicted Label (1) | True Positive (N1) | False Positive (N2) |
| Predicted Label (-1) | False Negatives (N3) | True Negatives (N4) |

1. $N = N1 + N2 + N3 + N4$
2. Precision : the fraction of retrieved instances that are relevant: $N1/(N1+N2)$
Precision= $TP/(TP+FP)$
3. Recall : is the fraction of relevant instances that are retrieved : $N1/(N1+N3)$
Recall= $TP/(TP+FN)$

| | Actual Label (1) | Actual Label (-1) |
|----------------------|------------------|-------------------|
| Predicted Label (1) | 10 | 3 |
| Predicted Label (-1) | 2 | 20 |

- ❖ Precision = 10/13
- ❖ Recall = 10/12