CHAPTER - 5
DESIGN AND DEVELOPMENT FOR ADSSAP

The astrological prediction using case based reasoning uses various classification techniques. This chapter is going to provide the base for designing the system.

5.1 General Architecture

The general architecture of ADSSAP consists of ten modules: Input Interface, Case Base Interface, Output Interface, Feedback Interface, Data Conversion Module, Case Base Storage, Retrieval Module, Learning Module, Inference Engine, Learning Module.

![Architecture Diagram of ADSSAP](image)

**Figure 5.1:** Architecture Diagram of ADSSAP

i. **Input Interface**

User interacts with the system through input interface to provide the personal data and the prediction he wants to be done.
ii. **Case Based Interface**

   It is the interface provided to input the huge amount of data that is going to be used for training purpose. The data is inputted in the form of cases in the case base storage through this interface. The data inputted will be of known classes.

iii. **Data Conversion Module**

   The data provided by the input interface and case base interface are converted into the planetary position and then stored in the form of records. This change in the usable format is responsibility of data conversion module.

iv. **Case Base Storage**

   All the data for training purpose are stored in Case Base Storage for future use. Data will be stored in the form of relation and attributes defining each case as one record.

v. **Retrieval Module**

   This module retrieve the records from the case base storage based on the input record provided by the user through data conversion module.

vi. **Learning Module**

   Based on the data provided by the retrieval module the learning module uses the algorithm required and pass the rules generated by the algorithm to the Inference Engine.

vii. **Inference Engine**

   Based on the input given by user and the rules provided by the learning module, the inference engine predict the output and provide it to the Output Interface.

viii. **Output Interface**

   This interface is used to provide the output to the user in the user understandable form.
ix. Feed Back Interface

If the user wants to provide the feedback to the system after getting the output then he/she can use Feed Back Interface for that.

x. Testing Module

This module test the output provided by the system against the feedback provided by the user. If the output provided and feedback matches then record is stored in the case base storage other wise it is discarded.

Let I be the new case provided through input interface for prediction.
Let p attributes are there in new case hence $I_1, I_1, \ldots, I_n$

Case Base Storage have number of cases known as $C_1$ to $C_n$.
The Retrieval Module provides the X cases $C_1$ to $C_x$ to the Inference Engine based on similarity with new case I.

- The Learning Module provides the Rules $R_1$ to $R_n$ to be used by Inference Engine.
- These Rules $R_1$ to $R_n$ are generated by using various algorithm on cases $C_1$ to $C_k$ stored in the case based storage.
- The Inference Engine generates the output O by applying rules $R_1$ to $R_n$ on input I.
- The Output O is converted into the format understandable by the human beings through Output Interface.
- If the results of the output are confirmed then the users can input those results to the system to improve the accuracy of prediction. This is done through Inference Engine sending the output O to Testing Module.
- The feedback F provided by the user on the output O is positive then one more case $C_{n+1}$ is stored in the case base by converting the data in compatible format that is understandable by system.
5.2 Flow of ADSSAP

**Step 1:** Cases consisting of previous records will be stored in the database and will form a case base.

**Step 2:** Whenever a new case would come to the system for prediction, the data would be converted in the form of planetary position or birth chart.

**Step 3:** The case base would be checked for similarity with the new case. If similar data is present then the prediction would be generated based on the case retrieved by retrieval module.

**Step 4:** If similar data is not present then learning modules would be used to generate a set of results which are sent to inference Engine.

**Step 5:** Inference Engine infers the output based on data supplied by the learning module.

**Step 6:** The result generated by inference engine would be converted in a manner which is comprehensive to the user and presented to the user through output interface.

**Step 7:** The feedback module would be used to provide the status that whether the output produce is correct or not for new case. The information of correct/incorrect prediction is submitted by the user through feedback module.

**Step 8:** Testing module tests the new case based on feedback provided through feedback module and the information through inference engine. If testing is correct then information is stored in Case Base Storage after proper conversion.
Figure 5.2: Flow Diagram of ADSSAP
5.3 Implementation of Classification and Prediction Techniques

Various classification and prediction algorithms were identified in the previous chapter that can be implemented. Here we are going to present the simulation of the algorithm that we are going to use in our framework.

Simulation of various algorithms is given below. The Table 5.1 shows a snapshot of the records present in the case base storage. Various algorithms are implemented on that case base storage.

**Table 5.1:** Records present in Case Base Storage for implementing various Algorithms for ADSSAP

<table>
<thead>
<tr>
<th>S. No</th>
<th>Moon</th>
<th>Mars</th>
<th>Venus</th>
<th>Saturn</th>
<th>Rahu</th>
<th>.</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td></td>
<td>Nonfamous</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td></td>
<td>Nonfamous</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>11</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td></td>
<td>Nonfamous</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>10</td>
<td>6</td>
<td>11</td>
<td>2</td>
<td></td>
<td>Nonfamous</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>8</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td></td>
<td>Nonfamous</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td></td>
<td>Nonfamous</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td></td>
<td>.</td>
</tr>
<tr>
<td>235</td>
<td>4</td>
<td>11</td>
<td>2</td>
<td>10</td>
<td>10</td>
<td></td>
<td>Famous</td>
</tr>
<tr>
<td>236</td>
<td>2</td>
<td>11</td>
<td>4</td>
<td>12</td>
<td>12</td>
<td></td>
<td>Famous</td>
</tr>
<tr>
<td>237</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>9</td>
<td></td>
<td>Famous</td>
</tr>
<tr>
<td>238</td>
<td>11</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td></td>
<td>Famous</td>
</tr>
<tr>
<td>239</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>10</td>
<td></td>
<td>Famous</td>
</tr>
<tr>
<td>240</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>10</td>
<td></td>
<td>Famous</td>
</tr>
</tbody>
</table>
5.3.1 Implementation of ZeroR

ZeroR is simple algorithm that relies on target and ignores all predictors.

It predict majority Class category. Following is the implementation for ZeroR in the research performed.

There are two values of Class: Famous and Nonfamous

In above example the number of records of Nonfamous are more irrespective of the attributes then the class will predict for each Record Class value as Nonfamous.

5.3.2 Implementation of Simple Cart

In this algorithm Classification and Regression tree both can be generated based on the type of variable. In our example we have Binary and Nominal attributes so classification tree is created.

In our example the target Class is Famous and Nfamous.

We have 120 samples of Famous and 120 samples of Nfamous.

Hence Gini Index= 1-((120/240)^2-(120/240)^2

Gini(t) =1-.25-.25=.5

For Gender attribute

<table>
<thead>
<tr>
<th>Gender</th>
<th>Target Value</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Famous</td>
<td>Nfamous</td>
</tr>
<tr>
<td>Female</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>Male</td>
<td>70</td>
<td>80</td>
</tr>
<tr>
<td>Total</td>
<td>120</td>
<td>120</td>
</tr>
</tbody>
</table>

Spliting the code based on Gender variable

Gini(s, t)=Gini(t)-PLGINI(tl)- PRGINI(tr)

Gini(tl)=1-((50/90)^2-(40/90)^2 =1-.309-.197=0.494
Design and Development for ADSSAP

Gini(tr) = 1 - \((70/150)^2 - (80/150)^2\) = 1 - 0.218 - 0.284 = 0.498

Gini(s, t) = 0.5 - \((90/240)*0.494 - (150/240)*0.498\)

Gini(s, t) = 0.5 - 0.185 - 0.311 = 0.004

Similarly Gini Index value for all split points are calculated for a variable and best variable is selected to split the input node

The tree generated by above example is as shown

![Figure 5.3: Tree for Simple Cart Algorithm in ADSSAP](image)

5.3.3 Implementation of ID3

Let U is finite set of states, called the universe

U = \{T1, T2, ..., T_{239}, T_{240}\} is a set of 240 Tuples

A is set of attributes

A = \{Moon, Mars, Venus, Saturn, Rahu, ..., Class\} out of it Class is the Decision Attribute remaining all are Condition Attributes.

V is the set of values of an attribute.

For Moon, Mars, Venus, Saturn, Rahu ... V = 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
For Gender $V =$ Male, Female
For Class $V =$ Famous, Nonfamous

Suppose $S$ is a set of 240 examples in which one of the attributes is Gender. The values of Gender can be Male or Female. Similarly Moon can be any one from 1 to 12. The classification of these 240 examples are 120 Famous and 120 Nfamous.

For attribute Gender, suppose there are 150 occurrences of Male 90 occurrences of Female. For Gender = Male, 80 of them are Famous and 70 are Nfamous. For Gender = Female, 50 are Famous and 40 are Nfamous. Therefore

$$\text{Entropy}(S) = -(\text{No. of Famous class/Total case}) \log_2(\text{No. of Famous class/Total case}) \cdot -(\text{No. of NFamous class/Total case}) \log_2(\text{No. of NFamous class/Total case})$$

$$\text{Entropy}(S) = -(120/240) \log_2(120/240) \cdot -(120/240) \log_2(120/240)$$

$$\text{Entropy}(S_{\text{Male}}) = -(80/150) \log_2(80/150) \cdot -(70/150) \log_2(70/120)$$

$$\text{Entropy}(S_{\text{Female}}) = -(50/90) \log_2(50/90) \cdot -(40/90) \log_2(40/90)$$

$$\text{Gain} (S, \text{Gender}) = \text{Entropy}(S) - (150/240) \cdot \text{Entropy}(S_{\text{Male}}) - (90/240) \cdot \text{Entropy}(S_{\text{Female}})$$

Similarly for Moon and other attribute gain is calculated

Gain $(S, \text{Moon})$ is highest so Moon become First node to partition and a branch is generated for each value of Moon.

Then again algorithm is run on each sub partition and so on.

![Decision Tree by ID3 Algorithm for ADSSAP](image-url)

**Figure 5.4:** Decision Tree by ID3 Algorithm for ADSSAP
5.3.4 Implementation of Naïve Bayes Classification

Let U is finite set of states, called the universe

U= \{T1, T2, ..., T_{239}, T_{240}\} is a set of 240 Tuples

A is set of attributes

A= \{Moon, Mars, Venus, Saturn, Rahu, ..., Class\} out of it Class is the Decision Attribute remaining all are Condition Attributes.

V is the set of values of an attribute.

For Moon, Mars, Venus, Saturn, Rahu ... V= 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

For Class V= Famous, Nonfamous.

From above table we can see that

Total No. of records are 240

No of records of class Famous = 120

No. of records of class Nonfamous = 120

Hence Probability of P (Famous) = 120/240 = .5

Probability of P (Nonfamous) = 120/240 = .5

**Table 5.3**: Counts of various attributes and sub attributes for Class attributes in ADSSAP

<table>
<thead>
<tr>
<th>Gender</th>
<th>Famous</th>
<th>Nonfamous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>70</td>
<td>80</td>
</tr>
<tr>
<td>Female</td>
<td>50</td>
<td>40</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Aries</th>
<th>Famous</th>
<th>Nonfamous</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

From above table we can see that

For Attribute Gender 70 males are Famous and 80 Males are Nonfamous

For Attribute Aries 5 in 1st house are Famous and 3 in 1st house are Nonfamous
Table 5.4: Probabilities of various attributes for given classes in ADSSAP

<table>
<thead>
<tr>
<th>Gender</th>
<th>Probability Famous</th>
<th>Probability Nonfamous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>70/120 = .583</td>
<td>80/120 = .667</td>
</tr>
<tr>
<td>Female</td>
<td>50/120 = .417</td>
<td>40/120 = .333</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Aries</th>
<th>Probability Famous</th>
<th>Probability Nonfamous</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5/120 = .042</td>
<td>3/120 = .025</td>
</tr>
<tr>
<td>2</td>
<td>9/120 = .075</td>
<td>9/120 = .075</td>
</tr>
</tbody>
</table>

Now suppose a new case of person come with value
Gender = Male
Aries = 1
.
.
To identify that person will be Famous or Nonfamous

\[ P(\text{Famous}/X) = P(\text{Famous}) \times P(\text{Male}/\text{Famous}) \times P(1/\text{Famous}) \ldots \]
\[ P(\text{Famous}/X) = .5 \times .583 \times .042 \ldots \]
\[ P(\text{NonFamous}/X) = P(\text{NonFamous}) \times P(\text{Male}/\text{NonFamous}) \times P(1/\text{NonFamous}) \ldots \]
\[ P(\text{NonFamous}/X) = .5 \times .667 \times .025 \ldots \]

If value of \( P(\text{Famous}/X) > P(\text{NonFamous}/X) \)
Then person will be Famous Else Nonfamous

5.3.5 Implementation of Decision Table

Decision table[6] is system S consisting of four attributes \( \{U, A, V, F\} \) where:

- \( U \) is finite set of states, called the universe
- \( A \) is set of attributes, attributes are of two types- \( C \) is the set of Conditions attributes and \( D \) is the set of decisions attributes.
- \( V \) is the set of values of an attribute.
- \( F \) is a set of decision functions called rules.
In the above implementation \( U = \{T_1, T_2, \ldots, T_{240}\} \) is a set of 240 Tuples

\[ A = \{\text{Moon}, \text{Mars}, \text{Venus}, \text{Saturn}, \text{Rahu}, \ldots, \text{Class}\} \]

total 23 attributes out of it Class is the decision attribute remaining all are condition attributes.

\( V \) (Values or Domain of Attributes) Moon, Mars, Venus, Saturn, Rahu …… has value between 1 and 12.

Class has value Famous or Nonfamous.

Gender has value Male or Female

The rules are generated from above example

Rule 1 - (Moon=12) \& (Mars=6) \& (Venus=6) \& (Saturn=3) \& (Rahu=1) \rightarrow (Class=Nonfamous)

In above Rule, if we check the value of Rahu for Non famous class we find that Rahu holds 4, 9, 2, 3, 6 values so the Rule 1 becomes

That means

Rule 1-(Moon=12) \& (Mars=6) \& (Venus=6) \& (Saturn=3) \rightarrow (Class=Nonfamous)

Now we consider value of Saturn, it takes value 3, 6, 10, 11, 2, 3 so again rule is reduced

So Rule 1-(Moon=12) \& (Mars=6) \& (Venus=6) \rightarrow (Class=Nonfamous)

5.3.6 Implementation of Decision Stump

Let \( U \) is finite set of states, called the universe

\[ U = \{T_1, T_2, \ldots, T_{239}, T_{240}\} \] is a set of 240 Tuples

\( A \) is set of attributes

\[ A = \{\text{Moon}, \text{Mars}, \text{Venus}, \text{Saturn}, \text{Rahu}, \ldots, \text{Class}\} \]

out of it Class is the Decision Attribute remaining all are Condition Attributes.

\( V \) is the set of values of an attribute.

For Moon, Mars, Venus, Saturn, Rahu … \( V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\} \)

For Class \( V = \) Famous, Nonfamous.

According to the concept of decision stump single node tree are created for each attribute.

Then each tree is checked and the tree that provide best split is taken as final decision stump tree.
Results for above example is

Venus = 8: Famous
Venus = 8: Nonfamous

![Decision Stump For ADSSAP](image)

**Figure 5.5**: Decision Stump For ADSSAP

### 5.3.7 Implementation of DTNB

Proposed by Hall and Frank 21, DTNB is a combination of Decision Tables and Naïve Bayes approaches. The model is a Bayesian Network in which the conditional probabilities are represented with Decision Tables. In the DTNB algorithm, the attributes are divided into two subsets by applying the gain function. These two subsets are used to create Decision Tables and Naïve Bayes model, respectively. The algorithm is based on a forward selection procedure, where all attributes are initially modelled by Decision Trees and the selected attributes are provided to a Naïve Bayes model. In order to generate the overall class probability, the class probabilities estimated by Decision Tables and Naïve Bayes have to be combined.

Suppose S is a set of 240 examples in which one of the attributes is Gender. The values of Gender can be Male or Female. Similarly Moon can be any one from 1 to 12. The classification of these 240 examples are 120 Famous and 120 Nfamous. For attribute Gender, suppose there are 150 occurrences of Male 90 occurrences of Female. For Gender= Male, 80 of them are Famous and 70 are Nfamous. For Gender = Female, 50 are Famous and 40 are Nfamous. Therefore

\[
\text{Entropy}(S) = \frac{-\text{No. of Famous class}}{\text{Total case}} \log_2 \frac{\text{No. of Famous class}}{\text{Total case}} - \frac{-\text{No. of NFamous class}}{\text{Total case}} \log_2 \frac{\text{No. of NFamous class}}{\text{Total case}}
\]

\[
\text{Entropy}(S) = -(\frac{120}{240}) \log_2 (\frac{120}{240}) - (\frac{120}{240}) \log_2 (\frac{120}{240})
\]
Entropy(S\text{Male}) = -(80/150) \times \log_2(80/150) - (70/150) \times \log_2(70/120)

Entropy(S\text{Female}) = -(50/90) \times \log_2(50/90) - (40/90) \times \log_2(40/90)

Gain(S, Gender) = Entropy(S) - (150/240) \times \text{Entropy}(S_{\text{Male}}) - (90/240) \times \text{Entropy}(S_{\text{Female}})

Similarly for Moon and other attributes, Gain is calculated.

Based on the value of Gain, two subsets of attributes are created.

One set is passed to Decision Table Algorithm and the other through Naïve Bayes.

**Decision Table**

Steps defined in 5.3.6 Section “Implementation of Decision Table for ADSSAP” is performed.

**Naïve Bayes**

Steps defined in section 5.3.5 “Implementation of Naïve Bayes Classification ADSSAP” is performed.

Overall class probability is computed as

\[ Q(y/X) = \alpha \times Q_{\text{DT}(y/X^+)} \times Q_{\text{NB}(y/X^-)} / Q(y) \]  \hspace{1cm} (5.1)

Where \( Q_{\text{DT}(y/X^+)} \) and \( Q_{\text{NB}(y/X^-)} \) are the class probability estimates obtained from the DT and NB respectively, \( \alpha \) is a normalization constant and \( Q(y) \) is prior probability of class.

**5.4 Approaches Used to Implement Various Phases of CBR in ADSSAP**

CBR has to be customized specifically to a particular problem domain because there is no specific CBR method suitable for a particular problem domain.

**5.4.1 Case Representation for ADSSAP**

In various experiments performed in the research, the cases were represented by defining various categories or classes for the cases (e.g., Doctors, Singers, Players, and so on). With each category, we store a number of Examples. Finding a case in the case base that matches an input description is done by combining the input features of a problem case into category that shares most of the features.
5.4.2 Case Retrieval for ADSSAP

The case retrieval of the system is done using various classification algorithms such as- ZeroR, Logistic Regression, Decision Trees, Simple Cart Algorithm, ID3 Algorithm, Naïve Bayes Algorithm, Decision Table Algorithm, Decision Stump Algorithm, DNTB Algorithm.

5.4.3 Case Reuse for ADSSAP

The reuse of the retrieved case focuses on the solution by two concept the differences among the past and the current case and what part of a retrieved case can be transferred to the new case.

In Adapt systems have to take into account differences and thus the reused part cannot be directly transferred to the new case but requires an adaptation process that takes into account those differences. There are two main ways to reuse past cases:

5.4.4 Case Revision for ADSSAP

Evaluate the case solution generated by reuse. If results generated are correct then no revision takes place but if the result generated is incorrect then reevaluate the case solution using domain-specific knowledge from case base storage.

5.4.5 Case Retainment for ADSSAP

If prediction performed by reusing the case is correct then retaining of the case is done by simply inserting the case as new case in the case base thus providing index.
5.5 Algorithms for ADSSAP

The Architecture of the ADSSAP is present in section 5.1. The design of frame work for system requires modelling of the algorithms that are going to be used for various operations. Here various algorithms are generated. This algorithms are identified based on the results generated by various algorithms presented in chapter 6. Only those algorithms are implemented for the frame work whose results were good and required to be developed for various implementations.
Various variables used throughout all algorithms are described below with the details of their purpose.

**Global Variables**

- **New_Case** is the input case given by user.
- **Case_Base** is the array of cases in case base storage
- **No_Case** is number of cases in Case Based Storage.
- **K** is are number attributes for each Case
- **R** be set of rules generated by Learning
- **No_R** are number of rules generated by the algorithm

### 5.5.1 Retrieval of Cases from Case Base Storage

The algorithm presented here is responsible to retrieve cases from the case base storage. The notations used by the algorithm are presented below

**Local Variables**

- **Usage** is variable to keep count of number of time a case from case base storage is used.

**Algorithm: Retrieval (New_Case, Case_Base)**

1. Loop for i=1 to No_Case
2. If New_Case = Case_Base[i] then
   i. Let Usage=Usage + 1
   ii. Return Case_Base[i] // Single case is reterived for further use
3. End Loop
4. Return (Case_Base, No_Case) // all cases are reterived for further use

### 5.5.2 Case Creation in Case Base Storage

This algorithm is responsible for the creation of new case in the case base storage. Various notations used by the algorithms are provided below

**Case_Creation( New_Case, Case_Base)**

1. Let No_Case = No_Case + 1
2. Loop for i = 1 to K
3. Let Case_Base[No_Case][i]=New_Case[i]
4. End Loop
5. Exit (Case_Base, No_Case)
5.5.3 Learning

Various rules are used by inference engine for prediction of the data. The rules used by inference engine are provided by learning module. These rules are generated using various algorithms that are identified in chapter 6. Based on the type of prediction to be done one of the algorithms will be activated and learning is done by that algorithm and the rules generated are passed to inference engine.

Local Variables

Algo be variable that can hold any one of values from (ZeroR, Simple_Cart, Decision_Stump, ID3, Navie_Bayes, Decision_Table, DTNB)

Algorithm: Learning (Case_Base, Algo, No_Case)

Step 1 : If Algo = ZeroR then
(i). Call ZeroR(Case_Base, No_Case)
(ii). Return(R, No_R)

Step 2 : If Algo = Simple_Cart then
(i). Call Simple_Cart(Case_Base, No_Case)
(ii). Return(R, No_R)

Step 3 : If Algo = Decision_Stump then
(i). Call Decision_Stump(Case_Base, No_Case)
(ii). Return(R, No_R)

Step 4 : If Algo = ID3 then
(i). Call ID3(Case_Base, No_Case)
(ii). Return(R, No_R)

Step 5 : If Algo = Naïve_Bayes then
(i). Call Navie_Bayes(Case_Base, No_Case)
(ii). Return(R, No_R)

Step 6 : If Algo = Decision_Table then
(i). Call Decision_Table(Case_Base, No_Case)
(ii). Return(R, No_R)

Step 7 : If Algo = DTNB then
(i). Call DTNB(Case_Base, No_Case)
(ii). Return(R, No_R)

Step 8 : End
5.5.4 Inferencing Output

This algorithm takes rules generated by the learning module, cases retrieved by retrieval module and input provided by the user to produce output.

Local Variables

Algo be variable that can hold any one of values from (ZeroR, Simple_Cart, Decision_Stump, ID3, Navie_Bayes, Decision_Table, DTNB)

Result is the output generated by inference engine after applying the rules generated by Learning algorithm on New_Case

Inference(New_Case, Case_Base, R, No_R, Algo)

Step1 : Call Learning(Case_Base, Algo, No_Case)
Step 2 : Loop for i = 1 to No_R // implementing rules generated to get o/p
   (i) Temp = Apply R[i] on New_Case
   (ii) Result = Result + Temp.
Step 3 : Display Result to User
Step 4 : End

5.5.5 Data Conversion

Data provided by the user and provided to the case base storage are not in format that can be directly used. Data conversion module is responsible to convert them in the form required by case base storage and other module.

Local Variables

Time_Birth is the time of birth of the person
Date_Birth is date of birth of the person
Place_Birth is place of birth of the person
Acase_Base is a case having 23 attributes
Longitude(Time_Birth, Place_Birth, Date_Birth) is API for getting longitude
Latitude(Time_Birth, Place_Birth, Date_Birth) is API for getting longitude
Horoscope(Longitude, Latitude) provides the house number for each planet and zodiac

**Algorithm: Data_Conversion(Time_Birth, Date_Birth, Place_Birth)**

Step 1 : Let Longitude = Call Longitude(Time_Birth, Place_Birth, Date_Birth)
Step 2 : Let Latitude = Call Latitude(Time_Birth, Place_Birth, Date_Birth)
Step 3 : Let Acase_Base = Call Horoscope(Longitude, Latitude)
Step 4 : Values of horoscope is converted into 23 attributes for Acase_Base
Step 5 : Return (Acase_Base)

**5.5.6 Testing**

For retaining cases in the case base storage the data to be send to case base storage is tested. If the results of test are positive then it is stored in case base storage else it is discarded.

**Local Variables**

**Result** be the value predicted by the system

**Feedback** be the value provided by the user

**Algorithm: Testing (Result, Feedback, Case_Base, New_Case, No_Case)**

Step 1 : if Result = Feedback then

(i) No_Case=No_Case+1

(ii) Case_Base= New_Case

Step 2 : Return

**5.5.7 ZeroR**

ZeroR is one of the method that require to be implemented.

Let I be case Provided by user for prediction.

**Local Variables**

**Class** is attribute having value Class1 or Class2

**CClass1** is variable used to keep count of Class1

**CClass2** is variable used to keep count of Class2
Algorithm: ZeroR( Case_Base, No_Case)
Step 1 : Loop for i = 1 to No_Case
Step 2 : If Case_Base[i][Class]= Class1 then CClass1=CClass1+1
Step 3 : Else CClass2= CClass2+1
Step 4 : End Loop
Step 5 : If CClass1 >= CClass2 then Return(Class1)
Step 6 : Else Return(Class2)

5.5.8 Simple Cart

Local Variables
Simple cart Algorithm is used to perform the prediction.

Algorithm: Simple_Cart(Case_Base, No_Case)
Step 1 : Loop for i = 1 to No_Case
Step 2 : Pick one of the predictor attribute Ai
Step 3 : Pick the value Vi of Ai attribute such that training data is divided into two parts.
Step 4 : Calculate purity of the split done by different attribute Ai having value Vi
Step 5 : Choose the split that reduces impurity the most
Step 6 : Chosen split points become nodes on the tree
Step 7 : Perform the above procedure for each node till leaf nodes are not created.
Step 8 : Pass the tree as rule to inference engine
Step 9 : End

5.5.9 ID3 in ADSSAP

Local Variable

T is the node of tree

Class is the attribute with value Class1 or Class2

Entropy is an variable that will hold entropy
Entropys is an array that will hold value of entropy of each attribute
Gains is an array that will hold entropy of each attribute
Attrib is a variable that hold the name of the attribute selected
Leaf_Node is the node that will hold the null, Class1 or Class2

Algorithm: ID3(Case_Base, No_Case)

Step 1 : Create a node T for the tree.
Step 2 : If No_Case = 0 then
   (i) Return(null)
Step 2 : If All value of Class=Class1
   (i) Return(Class1)
Step 3 : If All value of Class=Class2
   (i) Return(Class1)
Step 4 : Let Entropy = -(No. of Class1 cases / No_case)Log2(No. of Class1 cases / No_case) - (No. of Class2 cases / No_case)Log2(No. of Class2 / No_case).
Step 5 : Loop for i = 1 to K
   (i) C1Entropy[i]= - (No. of Sub1 Class1/No_Sub1)Log2 - (No. of Sub1 Class1/ No_Sub1) - (No. of Sub2 Class1/ No_Sub1)
   (ii) C2Entropy[i]= -(No. of Sub1 Class2/ No_Sub2)Log2 - (No. of Sub1 Class2/ No_Sub2) - (No. of Sub2 Class2/ No_Sub2)
   (iii) Gains[i] = Entropy-(No of Sub1/No_Case)* C1Entropy[i]- (No of Sub2/No_Case)* C2Entropy[i]
   (iv) End Loop
Step 6 : Loop for i = 1 to K
   (i) if Max < Gains[i] then
   (ii) Max = Gains[i]
   (iii) Attrib = i
Step 7 : End Loop
Step 8 : Loop For each value of Attrib
(i) Add a new tree branch below $T$,
(ii) For each branch find Subset_B of Case_Base belonging to branch
(iii) if no element in branch them Leaf_Node = null
(iv) if Subset_B have all element of same class then add Leaf_Node with the value of that Class.
(v) Else Call ID3(Subset_B, No_Case)

Step 9  :  End Loop
Step 10  :  Return $T$.

5.5.10 Navie Bayes

Local Variables

Class is attribute having value Class1 or Class2

CClass1 is variable used to keep count of no. of element in case based storage belonging to Class1

CClass2 is variable used to keep count of no. of element in case based storage belonging to Class2

Attribute is an array that will hold value of each attribute from case based storage

CValue1Class1 is an array that will hold count of no. of records belonging to Class1 and having Value1 for each attribute

CValue2Class1 is an array that will hold count of no. of records belonging to Class1 and having Value2 for each attribute

CValue1Class2 is an array that will hold count of no. of records belonging to Class2 and having Value1 for each attribute

CValue2Class2 is an array that will hold count of no. of records belonging to Class2 and having Value2 for each attribute

PValue1Class1, PValue2Class1, PValue1Class2, PValue2Class2 are array holding corresponding probability values.

Algorithm: Naïve Bayes (Case_Base, No_Case)

Step 1  :  Loop For i = 1 to No_Case
Step 2  :  If Class= Class1 then CClass1 = CClass1+1
Step 3  :  If Class= Class2 then CClass2 = CClass2+1
Step 4 : End Loop
Step 5 : PClass1 = CClass1 / No_Case
Step 6 : PClass2 = CClass2 / No_Case
Step 8 : Loop For i = 1 to k
Step 7 : Loop For j = 1 to No_Case
Step 8 : If Attribute[i] = Value1 and Class = Class1 then
        (i) CValue1Class1[i] = CValue1Class1[i]+1
Step 9 : If Attribute[i] = Value2 and Class = Class1 then
        (i) CValue2Class1[i] = CValue2Class1[i]+1
Step 10 : If Attribute[i] = Value1 and Class = Class2 then
          (i) CValue1Class2[i] = CValue1Class2[i]+1
Step 11 : If Attribute[i] = Value2 and Class = Class2 then
          (i) CValue2Class2[i] = CValue2Class2[i]+1
Step 12 : End Loop
Step 13 : For i= 1 to K
Step 13 : PValue1Class1[i] = CValue1Class1[i] / CClass1
Step 14 : PValue2Class1[i] = CValue2Class1[i] / CClass1
Step 15 : PValue1Class2[i] = CValue1Class2[i] / CClass2
Step 16 : PValue2Class2[i] = CValue2Class2[i] / CClass2
Step 17 : Return All Probability Values

5.5.11 Decision Table

Local Variables

Class is attribute having value Class1 or Class2

Attribute is an array that will hold each attribute

Algorithm: Decision Table(Case_Base, No_Case)

Step 1 : Create List of Data Attributes
Step 2 : Create List of possible values of each Data Attributes
Step 3 : Identify appropriate condition attributes
Step 4 : Compute Maximum number of Rules by different combination of the condition attributes values.
Step 5 : Identify Possible Actions for implementing decision
Step 6 : Build the Decision Table by entering all possible combinations of condition attribute values in rules column and condition rows
Step 7 : Define Actions for each rules
Step 8 : Resolve any rules for which the actions are not specific
Step 9 : Resolve contradiction in rules.
Step 10 : Simplify the Decision Table by eliminating impossible rules and combining rules with indifferent conditions.
Step 11 : Return the Rules generated to Inference Engine
Step 12 : End

5.5.12 Decision Stump

Local Attributes

T is the node of tree

Leaf_Node is a node that hold either Class1, Class2 or Null

Class is attribute having value Class1 or Class2

Attribute is an array that will hold each attribute

Accuracy is a parameter that hold no of correctly classified instances by each attribute

Final will hold the highest accuracy value

Index will hold the tree having highest accuracy

Algorithm: Decision Stump(Case_Base, No_Case)

Step 1 : Create a node T for the tree.
Step 2 : If No_Case = 0 then
       (i) Return(null)
Step 2 : If All value of Class=Class1
       (i) Return(Class1)
Step 3 : If All value of Class=Class2
       (i) Return(Class1)
Step 4 : Loop for i = 1 to K
       (i) Create a branch for each value of the Attribute[i]
       (ii) Divide the Case_Base into number of records for each branch
(iii) Create a Leaf_Node with value of majority class for each branch
(iv) Calculate Accuracy = Correctly_Classified Records
   if Final<Accuracy then
      (a) Final = Accuracy
      (b) index = i
Step 5 : End Loop
Step 6 : Return T having value index

5.5.13 DTNB Local Variable
PDT Class probability estimates obtained from Decision Tree
PNB Class probability estimates obtained from Decision Tree
Py Prior Probability
Pfinal Overall Class Probability
Entropy is an variable that will hold entropy
Entropys is an array that will hold value of entropy of each attribute
Gains is an array that will hold entropy of each attribute

Algorithm: DTNB (Case_Base, No_Case)
Step 1 : All attributes are passed to Gain_Calculation( ) Algorithm.
Step 2 : Let Entropy = -(No. of Class1 cases / No_case)Log2(No. of Class1 cases / No_case) - (No. of Class2 cases / No_case)Log2(No. of Class2 / No_case).
Step 3 : Loop for i = 1 to K
   (i) C1Entropy[i]= -(No. of Sub1 Class1/No_Sub1)Log2 - (No. of Sub1 Class1/ No_Sub1) -(No. of Sub2 Class1/ No_Sub1)Log2 - (No. of Sub2 Class1/ No_Sub1)
   (ii) C2Entropy[i]= -(No. of Sub1 Class2/ No_Sub2)Log2 - (No. of Sub1 Class2/ No_Sub2) -(No. of Sub2 Class1/ No_Sub2)Log2 - (No. of Sub2 Class2/ No_Sub1)
   (iii) Gains[i] = Entropy-(No of Sub1/No_Case)* C1Entropy[i]- (No of Sub2/No_Case)* C2Entropy[i]
   (iv) End Loop
Step 4: Divide attribute in two parts and store it in array Attribute1 and Attribute 2
Step 5: Create Case_Base_Naive as subset of Case_Based with Attribute1
Step 6: Create Case_Base_Decision as subset of Case_Based with Attribute2
Step 7: PNB= Call Naïve Bayes (Case_Base_Navie, No_Case)
Step 8: PDT= Call Decision_Table(Case_Base_Decision, No_Case)
Step 9: Input Py, α
Step 10: Pfinal = α × PDT × PNB*Py
Step 11: Return Pfinal

5.6 Analysis for ADSSAP

Let P be a person
Let n be the number of persons
So P1 to Pn are the persons whose actual status (output) is known
Let O1 to On be the output values of persons P1 to Pn
Op1 to Opn be the predicted outputs of the persons P1 to Pn
L1, L2, L3 are the three parameter related to date, time and place of birth of a person.
A1 to A21 are the astrological attributes prepared based on parameters L1, L2, L3
Out of which A1 to A12 are zodiac attributes
A13 to A21 are planet attributes
A22 is gender of the person and
A23 is decision attribute that defines the class to which a person belongs.
C1 to Cn be n number of classification technique

![Diagram of Analysis for ADSSAP](image)

**Figure 5.7: Analysis Diagram for ADSSAP**

Above Figure 4.2. describe the hypothesis. It states that data are collected for P1 to Pn persons with known output O1 to On. Op1 to Opn are the predicted output based on Attributes A1 to A22 then.

- If the similarity of predicted output Op1 to Opn $\geq 90\%$ with O1 to on then it is validated that the life events of human are in full relevance with the Astrology and Horoscope.

- If the similarity of predicted output Op1 to Opn $\leq 40\%$ with O1 to on then it is validated that the life events of human has no relevance with the Astrology and Horoscope.

- If the similarity of predicted output Op1 to Opn $< 90\%$ and $\geq 40\%$ with O1 to on then there is some relevance in life events with the Astrology and Horoscope. Hence improvement in the prediction and classification techniques is required.

And for predicting the particular output from the astrological charts Case Based Reasoning along with any one classification technique from C1 to Cn is used.
5.7 Data Collection and Creation for ADSSAP

Data is collection of information to provide results, observation and measurement of phenomena. We perform experiments using various methods and tools on the data to achieve various outcomes or results for the phenomena. So data collection plays an important part in research. For this research around 450 records were collected and then used in various experiments with different number of combinations.

Data collection for the purpose of research was a difficult job. The data were collected from individual persons. As the accuracy of data for the research purpose was very crucial hence first data were collected from the persons those are known. Most of the data were collected from reliable local persons on paper and through emails. The accuracy of data was confirmed by individually verifying with the persons. Some data were collected from Astro Data bank wiki project. Astro Data Bank publishes the huge collection of astrological data collected by Lois Rodden and her cooperators, so that these data can be used for astrological research, for astrological publications and for serious astrological discussion. It is a non-profit community project and is a subdivision of Astrodienst AG, with no commercial interest.

The raw data collected from both the sources consist of profile of the person, Date of Birth, Time of Birth, Place of Birth, Profession, Education, profile and brief biography of the person.

INFORMATION SHEET FOR CASE BASE REASONING
Name :-
Telephone Number :-
Time Of Birth*:-
Date Of Birth(DD/MM/YYYY)*:-
Place Of Birth*:-
Qualification*:-
Specialization*:-
Profession:-
Designation*:-
Internationally Famous or Not* :-
Brief Biography of Person :-

Figure 5.8: Data collection Sheet
Then astrological charts were prepared based on Indian astrology. These planetary charts were constructed using date of birth, time of birth and place of birth of the person. Then these records were stored in the tabular form for future use. Total 23 attributes Aries, Taurus, Gemini, Cancer, Leo, Virgo, Libra, Scorpio, Sagittarius, Capricorn, Aquarius, Pisces, Sun, Moon, Mars, Mercury, Venus, Jupiter, Saturn, Rahu, Ketu, Gender, and Class were selected for research purpose out of various attributes and all of them were nominal data. The data was converted into.arrf format and then WEKA (Waikato Environment for Knowledge Analysis) tool was used for performing the prediction.

Table 5.4 gives a brief description of the attributes created for performing astrological prediction. Attributes are nominal type holding the planetary positions of planets and zodiacs in the horoscope at the time of birth of the person.

To perform the experiments we are using software Weka (Waikato Environment for Knowledge Analysis) (Hall et al., 2008) well known software of machine learning written in Java and developed by University of Waikato. WEKA is software available under the GNU General Public License. It contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to this functionality

**Table 5.5:** Attributes Used For Classification Tasks in ADSSAP

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Aries</td>
<td>Nominal</td>
<td>House number of Aries in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>2.</td>
<td>Taurus</td>
<td>Nominal</td>
<td>House number of Taurus in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>3.</td>
<td>Gemini</td>
<td>Nominal</td>
<td>House number of Gemini in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>4.</td>
<td>Cancer</td>
<td>Nominal</td>
<td>House number of Cancer in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>5.</td>
<td>Leo</td>
<td>Nominal</td>
<td>House number of Leo in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>6.</td>
<td>Virgo</td>
<td>Nominal</td>
<td>House number of Virgo in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>7.</td>
<td>Libra</td>
<td>Nominal</td>
<td>House number of Libra in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>S. No.</td>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>8.</td>
<td>Scorpio</td>
<td>Nominal</td>
<td>House number of Scorpio in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>9.</td>
<td>Sagitarius</td>
<td>Nominal</td>
<td>House number of Sagittarius in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>10.</td>
<td>Capricorn</td>
<td>Nominal</td>
<td>House number of Capricorn in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>11.</td>
<td>Aquarius</td>
<td>Nominal</td>
<td>House number of Aquarius in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>12.</td>
<td>Pisces</td>
<td>Nominal</td>
<td>House number of Pisces in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>13.</td>
<td>Sun</td>
<td>Nominal</td>
<td>House number of Sun in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>14.</td>
<td>Moon</td>
<td>Nominal</td>
<td>House number of Moon in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>15.</td>
<td>Mars</td>
<td>Nominal</td>
<td>House number of Mars in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>16.</td>
<td>Mercury</td>
<td>Nominal</td>
<td>House number of Mercury in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>17.</td>
<td>Venus</td>
<td>Nominal</td>
<td>House number of Venus in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>18.</td>
<td>Jupiter</td>
<td>Nominal</td>
<td>House number of Jupiter in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>19.</td>
<td>Saturn</td>
<td>Nominal</td>
<td>House number of Saturn in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>20.</td>
<td>Rahu</td>
<td>Nominal</td>
<td>House number of Rahu in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>21</td>
<td>Ketu</td>
<td>Nominal</td>
<td>House number of Ketu in the birth chart. Value can be a number from 1 to 12</td>
</tr>
<tr>
<td>22</td>
<td>Gender</td>
<td>Nominal</td>
<td>It will have value M for Male or F for Female.</td>
</tr>
<tr>
<td>23</td>
<td>Class</td>
<td>Nominal</td>
<td>It holds classes of the record based on the experiment performed</td>
</tr>
</tbody>
</table>

### 5.8 Estimation of Performance Parameter

The statistical parameters are calculated for various methods thus determining the accuracy of the results generated. The entire purpose of estimation of parameters is to produce estimations with accuracy to arrive at best results for various predictions done on human life. The parameters are Root Mean Squared Error (RMSE), Mean absolute Error (MAE), Recall & Precision.
5.8.1 Mean Absolute Error

In statistics, the mean absolute error (MAE) is a quantity used to measure how close forecasts or predictions are to the eventual outcomes. The mean absolute error is given by

\[
\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |f_i - y_i| = \frac{1}{n} \sum_{i=1}^{n} |e_i|.
\]  
(5.2)

As the name suggests, the mean absolute error is an average of the absolute errors

\[|e_i| = |f_i - y_i|\]

(5.3)

Where \(f_i\) is prediction and \(y_i\) the true. Note that alternative formulations may include relative frequencies as weight factors.

The mean absolute error is on same scale of data being measured. This is known as a scale-dependent accuracy measure and therefore cannot be used to make comparisons between series on different scales.

5.8.2 Root Mean Squared Error

Root-Mean-Square Error (RMSE) is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values actually observed. The RMSE represents the differences between predicted values and observed values. It is just the square root of the mean square error.

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} e_i^2}.
\]  
(5.4)
5.8.3 Precision

Precision also known as positive predictive value is the fraction of retrieved instances that are relevant. Precision is therefore based on an understanding and measure of relevance.

In a classification task, the precision for a class is the number of true positives (i.e. the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class).

\[
\text{Precision} = \frac{TP}{TP+FP}
\]

Where TP is true positive
FP is false positive

5.8.4 Recall

Recall also known as sensitivity is the fraction of relevant instances that are retrieved. Recall are therefore based on an understanding and measure of relevance. Recall in this context is defined as the number of true positives divided by the total number of elements that actually belong to the positive class (i.e. the sum of true positives and false negatives, which are items which were not labeled as belonging to the positive class but should have been)

\[
\text{Recall} = \frac{TP}{TP+FN}
\]

Where TP is true positive
FN is false negative

5.8.5 Execution Time

Execution Time of algorithm is defined as the time taken by system to perform code of algorithm with the values defined when no other software, except the operating system runs on it. Execution time depends on the computers clock, memory size and the input data size. However, the Execution time is important to identify the efficiency of algorithm.
Experimental Setup using Weka Tool for ADSSAP

The experiments were performed using Weka Tool through various interfaces. Figure 4.1 shows the initial interface that is provided by the weka tool. The arrf format file is uploaded in the tool using this interface. The interface provided after loading the file is presented in Figure 4.2. Once the file is loaded than classification task is performed using the panel provided in Figure 4.3 and the results are generated as shown in the figure. Figure 4.4 presents a sample output file generated by weka tool.

![Interface of Weka Tool for ADSSAP](image1)

**Figure 5.9:** Interface of Weka Tool for ADSSAP

![Weka Interface after loading the.arrf format file for ADSSAP](image2)

**Figure 5.10:** Weka Interface after loading the.arrf format file for ADSSAP
Figure 5.11: Result generated by Weka tool for ADSSAP
=== Run information ===

Scheme: weka.classifiers.trees.DecisionStump
Relation: astrology
Instances: 240
Attributes: 23
  Gender
  Class
  Aries
  Taurus
  Gemini
  Cancer
  Leo
  Virgo
  Libra
  Scorpio
  Sagittarius
  Capricon
  Aquarius
  Pisces
  Sun
  Moon
  Mars
  Mercury
  Jupiter
  Venus
  Saturn
  Rahu
  Ketu
Test mode: 20-fold cross-validation

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

Decision Stump

Classifications

Venus = 8: Famous
Venus != 8: Nfamous
Venus is missing: Nfamous

Class distributions

Venus = 8
Nfamous Famous 0.1875 0.8125
Venus != 8
Nfamous Famous 0.5223214285714286 0.47767857142857145
Venus is missing
Nfamous Famous 0.5 0.5
Time taken to build model: 0 seconds

### Stratified cross-validation ===

### Summary ===

| Correctly Classified Instances | 118 | 49.1667 % |
| Incorrectly Classified Instances | 122 | 50.8333 % |
| Kappa statistic | -0.0167 |
| Mean absolute error | 0.5033 |
| Root mean squared error | 0.5087 |
| Relative absolute error | 100.6507 % |
| Root relative squared error | 101.7421 % |
| Total Number of Instances | 240 |

### Detailed Accuracy By Class ===

<table>
<thead>
<tr>
<th>Class</th>
<th>TP Rate</th>
<th>FP Rate</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
<th>ROC Area</th>
<th>ROC Area Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nfamous</td>
<td>0.942</td>
<td>0.958</td>
<td>0.496</td>
<td>0.942</td>
<td>0.649</td>
<td>0.471</td>
<td>Nfamous</td>
</tr>
<tr>
<td>Famous</td>
<td>0.042</td>
<td>0.058</td>
<td>0.417</td>
<td>0.042</td>
<td>0.076</td>
<td>0.471</td>
<td>Famous</td>
</tr>
<tr>
<td>Weighted Avg.</td>
<td>0.492</td>
<td>0.508</td>
<td>0.456</td>
<td>0.492</td>
<td>0.363</td>
<td>0.471</td>
<td></td>
</tr>
</tbody>
</table>

### Confusion Matrix ===

```
a b <-- classified as
<table>
<thead>
<tr>
<th>Class</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nfamous</td>
<td>113</td>
<td>7</td>
</tr>
<tr>
<td>Famous</td>
<td>115</td>
<td>5</td>
</tr>
</tbody>
</table>
```

**Figure 5.12:** Sample output file produced by Weka Tool for ADSSAP

### 5.10 Summary of Chapter

This chapter concentrates on the design and development part of the work. It describes the architectural design of the system and also describes its execution sequences. After that implementation of various algorithm was simulated on the case base storage. Then it describes the various approaches used to implement case representation and CBR cycle.

Algorithm for various module and classification and prediction techniques were designed and then hypothesis was proposed and presented with diagram.

After designing this chapter talks about the data collection and experimental setup for performing the research thus identifying the necessary attributes. This chapter also describes the interfaces of the weka tool and the sample output presented by the tool.