Material and Methods
3.1 Overview

This chapter is focused on the strategies used in the development of different prediction methods described in this thesis. These include the decisive steps followed in development of a biological prediction method like:-

1) **Preparation of a raw dataset**: The first requirement for developing any bioinformatics prediction method is the availability of experimentally validated sequence data. One should create/use dataset from latest release of databases in order to have an up-to-date and accurate dataset. Moreover, larger the dataset used for training, more accurate, reliable and less error prone will be the prediction method.

2) **Preparation of a non-redundant dataset**: Presence of highly similar sequences in a large dataset may make the prediction method biased towards that group of sequences. Various strategies are described in this chapter which could be used to remove duplicate or highly homologous sequences to create a non-redundant dataset.

3) **Techniques to develop the prediction method**: Most of the tools developed during this thesis work are based on artificial intelligence techniques, which are capable of recognizing the hidden patterns from biological data.

4) **Converting the primary protein sequence into fixed length input patterns**: As machine learning techniques are unable to handle the variations in protein sequence length, therefore there is a requirement of strategies for encapsulating the information from proteins of variable lengths to a fixed length format.

5) **Cross validation**: All the cross-validation techniques and performance measures used to assess the performance of various methods have been described.

6) **Measures of performance**: Evaluating the performance of prediction methods after its development is an important step to test the reliability of that method. Various threshold dependent and independent measures are described in this chapter in detail.

Similarly for developing any QSAR model in chemoinformatics, the following steps were undertaken:

1) **Preparation/collection of dataset**: The first requirement for developing any QSAR model
for prediction is the availability of experimentally validated chemical data. One should create/use dataset from latest release of databases in order to have an up-to-date and accurate dataset. Moreover, larger the dataset used for training, more accurate, reliable and less error prone will be the prediction method.

2) **Techniques to develop the prediction method:** Most of the tools developed during this thesis work are based on machine-learning techniques, which are capable of recognizing the hidden patterns from chemical data.

3) **Preprocessing of dataset:** During this step molecules were drawn in ChemBioOffice and then converted into 3D and energy minimization was done for these molecules.

4) **Converting the information of primary molecules in input patterns (Molecular descriptor calculation):** The molecular descriptors of chemical molecules were calculated which were further used in different machine learning techniques for QSAR model development.

5) **Feature Selection:** Since number of these descriptors are very high and all of these could not be used in machine learning techniques, different techniques were needed to select most significant molecular descriptors for model development.

6) **Cross validation:** All the cross-validation techniques and performance measures used to assess the performance of various methods have been described.

7) **Measures of performance:** Evaluating the performance of prediction methods after its development is an important step to test the reliability of that method. Various threshold dependent and independent measures and regression parameters which are used in this thesis are described in detail in this chapter.

8) Finally, the software and hardware used for data modeling and prediction method have been described.

### 3.2 Bioinformatics

#### 3.2.1 Preparation of dataset

Any prediction tool in bioinformatics requires obligatory creation of a clean and high quality experimentally annotated dataset. Extraction of unprocessed experimental dataset (raw dataset)
from different preexisting sources requires a great deal of work and also involves several critical
decisions and pitfalls. The first and the most important step is the selection of database from
which dataset has to be compiled because all training examples should have unambiguous
experimental evidence for the desired function/property. The annotation should not be inferred
either by similarity or existing prediction methods. In present thesis, only non-ambiguous,
experimentally annotated full-length protein sequences from databases like Uniprot (Apweiler,
Bairoch et al. 2004; Bairoch, Apweiler et al. 2005; Wu, Apweiler et al. 2006), SwissProt
(Gasteiger, Jung et al. 2001; Boeckmann, Bairoch et al. 2003) for Glutathione S-Transf-erase
(GST) protein prediction (Mishra, Kumar et al. 2007) at protein level have been used. FAD
binding proteins from PDB (Berman, Battistuz et al. 2002) were used to develop prediction
methods for FAD interacting residue prediction. Moreover, the designing of accurate method
requires data about both negative and positive examples.

3.2.1.1 Non-redundant Dataset

Performance of a prediction method will be biased and unrealistic if the dataset used for its
training have high similarity/homology within sequences. Such type datasets are known as
redundant datasets. Therefore, to avoid any biasness while developing a realistic method, it is
starting point to create a non-homologous/non-redundant dataset. Non-homologous dataset for
testing is absolutely necessary for a reliable estimate of performance of any prediction method.
The purpose can be served by keeping a single sequence from a cluster of homologous sequences
by applying certain threshold cut-off (E - value) or removing duplicates, if any. In this thesis, all
duplicate sequences were removed while creating clean datasets for peptide prediction methods.
The peptides with unnatural amino acids were also removed from the dataset. The dataset
redundancy was reduced to $n\%$ using freely available software CD-HIT (Li and Godzik 2006;
Huang, Niu et al. 2010). This means that none of the two sequences of dataset had more than
$>n\%$ identity with each other. In present thesis, 90% and 40% identity has been used. The dataset
formed in this way is called as non-redundant dataset.

3.2.1.2 Independent dataset/Blind dataset

It was observed that cross-validation is not true blind test for assigning the performance of
newly developed methods. In the it shown in several studies that there is bias in the performance
of the methods in trained and tested on same dataset despite m-fold cross-validation (Soeria-
Atmadja, Wallman et al. (2005). Therefore, performance of the newly developed method should be evaluated on a naïve dataset which is neither used for training nor for test. The performance of a method on dataset can be assessed by calculating the threshold dependent or threshold independent performance or by using both.

### 3.2.2 CD-HIT (Li and Godzik 2006)

CD-HIT stands for Cluster Database at High Identity with Tolerance. The program (CD-hit) takes a fasta format sequence database as input and produces a set of 'non-redundant' (nr) representative sequences as output. In addition CD-hit output as a cluster file, documenting the sequence 'groupies' for each nr sequence representative. The idea is to reduce the overall size of the database without removing any sequence information by only removing 'redundant' (or highly similar) sequences. This is why the resulting database is called non-redundant (nr). Essentially, CD-hit produces a set of closely related protein families from a given fasta sequence dataset. CD-uses a greedy incremental algorithm, that uses the 'longest sequence first' list removal algorithm to remove sequences above a certain identity threshold. Additionally the algorithm implements a very fast heuristic to find high identity segments between sequences, and thus avoids many costly full alignments.

### 3.2.3 pLogo (Schneider and Stephens 1990)

pLogo is an advanced graphical method presented for displaying the patterns in a set of aligned sequences (Schneider and Stephens 1990). The characters representing the sequence are stacked on top of one other for each position in the aligned sequences. The height of each letter is made proportional to its frequency, and the letters are sorted so the most common one is on top. The height of the entire stack is then adjusted to signify the information content of the sequences at that position. From these 'sequence logos', one can observe not only the consensus sequence but also the relative frequency of bases and the information content (measured in bits) at every position in a site or sequence. The logo displays both significant residues and subtle sequence patterns. pLogo were drawn to look for the positional preference of certain residues for the N- and C-termini of antibacterial peptides while developing the antibacterial peptide prediction methods.
3.2.4 Techniques used to develop prediction methods

After the creation of a clean non-redundant dataset for development of methods, careful selection of computational tools plays an important role in the development of an accurate and reliable method. The methods developed in this study can be classified into two major classes based on the techniques used in their development. These classes are similarity search methods and machine learning methods. The techniques and methods have been described in following section.

3.2.4.1 Similarity search algorithm

BLAST (Altschul, Gish et. al. 1990)

BLAST (Basic Local Alignment Search Tool) (Altschul, Gish et al. 1990; Altschul, Madden et al. 1997) and FASTA (Pearson and Lipman 1988) are the two most popular tools for searching similar sequences in database. BLAST generates a list of short words (default word size is 3 for protein and 11 for nucleotide) from a query sequence. These short words are searched in target sequence database and matching words are extended until the algorithms found a match whose score is below the user defined threshold. BLAST software is available both in form of web-server at NCBI (http://www.ncbi.nlm.nih.gov/BLAST) and standalone version (ftp://ftp.ncbi.nlm.nih.gov/blast/documents/blast.html). To fulfill different searching requirements several modification of BLAST programs are available which is described in table 3.1.

Table 3.1: Different form of BLAST on the basis of query sequence and target database.

<table>
<thead>
<tr>
<th>BLAST</th>
<th>Type of Query sequence</th>
<th>Nature of Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLASTP</td>
<td>Protein Sequence</td>
<td>Protein Sequence</td>
</tr>
<tr>
<td>BLASTN</td>
<td>Nucleotide Sequence</td>
<td>Nucleotide Sequence</td>
</tr>
<tr>
<td>BLASTX</td>
<td>Nucleotide Sequence</td>
<td>Protein sequence</td>
</tr>
<tr>
<td>TBLASTN</td>
<td>protein query sequence</td>
<td>six-frame translated nucleotide sequence</td>
</tr>
<tr>
<td>TBLASTX</td>
<td>six-frame translated nucleotide sequence</td>
<td>six-frame translated nucleotide sequence</td>
</tr>
</tbody>
</table>
**PSI-BLAST (Altschul, Madden et al. 1997)**

Position-Specific Iterated (PSI)-BLAST is the most sensitive BLAST program, making it useful for finding very remotely related proteins or new members of a protein family. The first round of PSI-BLAST is a standard protein-protein BLAST search. The program builds a position-specific scoring matrix (PSSM or profile) from a multiple alignment of the sequences returned with expect values better (lower) than the inclusion threshold (default=0.005). The PSSM will be used to evaluate the alignment in the next iteration of search. Any new database hits, which have E-values below the inclusion threshold, are included in the construction of the new PSSM. A PSI-BLAST search is said to have converged when no more matches to new database sequences are found in subsequent iterations. The module of PSI-BLAST (Altschul, Madden et al. 1997) was designed, in which query sequences in test datasets were searched against proteins in training datasets using PSI-BLAST. Three iterations of PSI-BLAST were carried out at a cut-off E-value of 0.001. The module could predict any of the functions depending upon the similarity of the query protein to the protein in the dataset. Command-line options used in the study are shown in Table 3.2.

**Table 3.2: Command-line options used in PSI-BLAST**

<table>
<thead>
<tr>
<th>Command</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>formatdb</td>
<td><code>-i fastafile -p T -0 T -n databasefilename</code></td>
</tr>
</tbody>
</table>

where `fastafile` is a file containing sequences in FASTA format, and `databasefilename` is the source database.

| blastpgp  | `-e E-value -j 3 -d databasefilename -i queryfilename -o outfilename` |

E-value used were $10^{-9}$, $10^{-6}$, $10^{-3}$, $10^{-1}$ and 1.

**3.2.4.1 Machine learning based methods**

In this study, two main machine learning techniques; artificial neural network (ANN) and support vector machine (SVM) have been used extensively.
Support vector machine (SVM)

Support Vector machines (SVM) is a class of statistical learning algorithms explained by Vapnik (Vapnik 1995). SVM and related kernel methods have become extremely popular and are being voraciously implemented in the field of computational biology for protein secondary structure prediction (Ward, McGuffin et al. 2003), MHC-binding peptide prediction (Donnes and Elofsson 2002), microarray analysis, antibacterial peptide prediction (Lata, Mishra et al.; Lata, Sharma et al. 2007) and many other classification problems (Brown, Grundy et al. 2000; Kriegl, Arnhold et al. 2005; Kim, Nam et al. 2006; Li, Allinson et al. 2006; Kong and Choo 2007; Leong 2007; Kumar, Gromiha et al. 2008). SVM is successful in avoiding the problems that are often encountered with other machine learning approaches. For example, in case of ANN, too many parameters have to be optimized, over-fitting is hard to avoid and it is difficult to find the best structures particularly size of hidden layer (Zavaljevski, Stevens et al. 2002). The effectiveness of SVM in overcoming these problems has proved it to be a promising method in practice. SVM has superior generalization capability when the number of input features is large as compared to number of training samples (Cai, Wang et al. 2003; Chou and Cai 2003; Cai, Ricardo et al. 2004). The basic idea and theory of two class and multiclass SVM has been described in following section:

a. SVM for two class classification problem

SVM can be easily implemented for two class classification problem such as GST and non-GST or FAD interacting and non-FAD interacting residues. Let us assume that we have a series of examples (or input vectors) \( x_i \in \mathbb{R}^d \) \((i=1,2,...,N)\) with corresponding labels \( y_i \in \{ +1, -1 \}\) \((i=1,2,...,N)\). For example in case of antibacterial peptides, \( x_i \) corresponds to the amino acids sequence of the peptides and \( y_i \) represents positive/negative samples. The SVM performs the classification task as follows (Figure 3.1)

i) Mapping of the input vectors \( x_i \) into a high dimensional features space \( \phi(x_i) \in \mathbb{H} \).

ii) Construction of an optimal separating hyperplane (OSH) in the higher dimensional feature space. The OSH is the hyperplane with the maximum distance to the nearest data points of each class in the feature space \( \mathbb{H} \). The choice of mapping feature \( \phi \) which is defined by a
kernel function $K(x_i, x_j)$ is crucial in SVM-based classification. The decision function used by SVM is shown in equation 1.

$$f(x) = \text{sign} \left( \sum_{i=1}^{N} y_i \alpha_i K(x_i, x_j + b) \right)$$ \[1\]

Where $K(x_i, x_j)$ is the kernel function that defines the feature space, coefficient $b$ is the bias value, $i$ is the number obtained by solving the quadratic programming (QP) problem that gives the maximum margin hyper plane. Quadratic programming is a convex optimization problem, which ensures a global optimum hyperplane. Kernel function determines feature space that means different kernels represent the input vectors in different ways. The kernels tested for prediction in the present study are linear, polynomial and radial basis function (RBF).

![Figure 3.1: An illustration of SVM based classification. The red and green circles represent positive and negative examples respectively.](image)

The example of the radial basis function and polynomial kernels has been shown by sin equations 2 and 3 respectively. The problem of choosing the most suitable kernel for an SVM is analogous to the problem of choosing the architecture for a neural network.
Chapter 3: Material and methods

\[ K(x_i, x_j) = \exp(-\gamma \| x_i - x_j \|^2) \]  

[2]

\[ K(x_i, x_j) = (\| x_i \cdot x_j \|)^d \]  

[3]

For actual implementation, we have used SVM-Light package written by Joachims (Joachims 1999), which is freely available at http://svmlight.joachims.org for scientific use (Joachims 1999). The software can be used in classification or regression mode. The classification mode is important for solving the two class problem such as positive and negative examples. The regression mode can be used to predict targeted value instead of +1/-1. For example in case of melting point prediction the real value of binding affinity can be a targeted value.

b. SVM for multiclass problem

The above explained SVM is not able to handle the multiclass problem, where the numbers of classes are more than two such as classification of proteins into families and sub-families. The simplest procedure to handle the multiclass classification problem is to reduce the multiclassification to a series of binary classifications. For \( N \) class classification, \( N \) SVMs have been constructed. The \( i \)th SVM has been trained with all samples of \( i \)th subfamily with positive label and samples of all other subfamilies as negative label. The SVMs trained in this way have been referred to as one versus rest 1-v-r SVMs (Hua and Sun 2001). In this classification approach, each of the unknown protein achieves \( N \) scores. An unknown protein will be classified into the subfamily that correspond to the 1-v-r SVM with highest output score (Hua and Sun 2001).

Hidden Markov Model (HMM)

Proteins are often composed of domains. These are polypeptide regions that can adopt independent compact three-dimensional (3D) structures and are often found in diverse molecular contexts. Search against database of protein domain families often produce more precise results than BLAST search because a protein containing one or more common domains can produce a list of BLAST hits when searched against simple sequence databases like GenBank, Swissprot, PIR. Protein family databases are typically based on multiple sequence alignments of known family members. Conserved features are given higher weight during searching, which make the
comparison more sensitive than pair-wise alignment approaches. In this thesis work, Pfam domain database has been used which contains multiple alignments and hidden Markov model based profiles (HMM-profiles) of complete protein domains.

A hidden Markov model (HMM) is a statistical model in which modeled system is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters (Krogh, Brown et al. 1994). In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters. It can be simplified as follows: suppose that at time \( t \) a Markovian random variable is in state \( E_t \). If the probability that at time \( t+1 \) it is in state \( E_k \) is \( P_{jk} \) (called transition probability from state \( E_j \) to \( E_k \)). If transition probabilities of movement from one state to second state are represented in form of matrix \( P \) (transition probability matrix) then \( P \) will looks as follow

\[
P = \begin{bmatrix}
(p_{11}) & (p_{12}) & (p_{13}) & \cdots & (p_{1s}) \\
(p_{21}) & (p_{22}) & (p_{23}) & \cdots & (p_{2s}) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
(p_{s1}) & (p_{s2}) & (p_{s3}) & \cdots & (p_{ss})
\end{bmatrix}
\]

Row in the matrix corresponds to the state from which the transition is made, and column in the matrix corresponds to the state to which the transition is made. Thus the probabilities in any particular row in the matrix \( P \) must sum to 1. However, the probabilities in any given column do not have to sum to anything in particular. Besides transition probabilities there will be also some initial probability distribution for the various states in the Markov chain. That is, it is assumed that there is some probability \( \pi_t \) that at the initial time point the Markovian random variable is in state \( E_t \). The initial distribution arises when it is known that the random variable starts in state \( E_t \); in this case \( \pi_t = 1 \), while \( \pi_j = 0 \) for \( j \neq t \). The initial probability distribution along with transition matrix \( P \) determines the probability for any event in the entire process.

A hidden Markov model is similar to a Markov chain, but is more generalized, and hence
more flexible. It allows to model phenomena that cannot be modeled with a regular Markov chain model. In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. The main addition is that when a state is visited by the Markov chain, the state “emits” a time independent, but state-dependent, probability distribution over the alphabet. When the HMM runs there is, first, a sequence of states visited, which we denote by $q_1, q_2, q_3, \ldots$, and second, a sequence of emitted symbols, denoted by $o_1, o_2, o_3, \ldots$. This generation can be visualized as:

![HMM diagram](image)

We can denote entire sequence of $q_i$'s by $Q$ and the entire sequence of $o_i$'s by $O$, which means “the observed sequence $O = o_1, o_2, o_3, \ldots$” and “the state sequence $Q = q_1, q_2, q_3, \ldots$”. Often we know the emitted sequence $O$ but do not know the sequence $Q$. It means sequence $Q$ is “hidden”.

Use of HMM in biological problem (Krogh, Brown et al. 1994) can be understood by this example. Let there be a DNA sequence that starts in an intron, contains one 5' splice site and ends in an intron. Here the problem is to predict the point where exon ends and intron begins i.e. prediction of 5' splice site (5'-SS). Again consider that exon has a uniform base composition (25% each base), introns are A/T rich (40% each for A/T, 10% each for C/G) and 5'-SS consensus nucleotide is almost always a G (95% G and 5% A). All this information is pictorially depicted by an HMM diagram (Figure 3.2). The three states: E (exon), 5 (5'-SS) and I (intron) has its own emission probabilities (shown above the states), which model the base composition of exons, introns and the consensus G at the 5'-SS. Each state also has the transition probabilities (arrows) i.e. the probabilities of moving from one state to a new state. The transition probabilities describe the linear order in which the states are expected to occur: one or more Es, one 5, one or more Is. Now suppose the HMM begins to generate a sequence. When it visits a state, it emits a residue on the basis of the state’s emission probability distribution. The visit to next state occurs according to the state’s transition probability distribution. The model thus generates two strings of information. One is the underlying state path (the labels) the other is the observed sequence (DNA), each residue being emitted from one state in the state path.
Figure 3.2: The diagrammatic representation of HMM generated for 5' - splice site prediction.

The state path is a Markov chain, meaning that what state comes next, depends only on the present state. Since only observed sequence is given hence the underlying state path is hidden. It means residue label has to be inferred on the basis of nucleotide sequence. Thus the state path is a hidden Markov chain.

The probability $P(S, \pi | HMM, \theta)$ that an HMM with parameters $\theta$ generates a state path $\pi$ and an observed sequence $S$ is the product of all the emission probabilities and transition probabilities that were used. For 26-nucleotide sequence shown in Figure 3.2, where there are 27 transitions and 26 sequence alphabet (nucleotide) emissions. Hence an HMM means four things (i) the symbol alphabet, $K$ different symbols (e.g., $ATGC$, $K = 4$); (ii) the number of states in the model, $M$; (iii) emission probabilities $E_i(x)$ for each state $i$, that sum to one over $K$ symbols $x$, $\sum_x E_i(x) = 1$; and (iv) transition probabilities $t_{i,j}(j)$ for each state $i$ going to any other state $j$ (including itself) that sum to one over the $M$ sites $j$, $\sum_j t_{i,j} = 1$.

In the present work the HMM based searching was implemented using a publicly available software package HMMER (http://selab.janelia.org/).

### 3.2.5 Techniques used creating patterns from proteins

Machine learning techniques are highly successful for residue state prediction where the input data for network consists of fixed length. Therefore, various strategies are adopted for providing
global information about whole proteins of variable length in a fixed length format. The approaches used in this study to obtain the patterns of fixed length from proteins of variable lengths have been described in following section:

### 3.2.5.1 Binary coding

Each amino acid was represented by a unique pattern containing binary numbers (i.e. 19 0’s and single 1) e.g. A “100000000000000000000” C “010000000000000000000” so on (table 3.3).

**Table 3.3:** Binary or sparse encoding of amino acids.

<table>
<thead>
<tr>
<th>Amino acid</th>
<th>Binary Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alanine (A)</td>
<td>100000000000000000000</td>
</tr>
<tr>
<td>Cysteine (C)</td>
<td>010000000000000000000</td>
</tr>
<tr>
<td>Aspartate (D)</td>
<td>001000000000000000000</td>
</tr>
<tr>
<td>Glutamate (E)</td>
<td>000100000000000000000</td>
</tr>
<tr>
<td>Phenylalanine (F)</td>
<td>000010000000000000000</td>
</tr>
<tr>
<td>Glycine (G)</td>
<td>000001000000000000000</td>
</tr>
<tr>
<td>Histidine (H)</td>
<td>000000100000000000000</td>
</tr>
<tr>
<td>Isoleucine (I)</td>
<td>000000010000000000000</td>
</tr>
<tr>
<td>Lysine (K)</td>
<td>000000001000000000000</td>
</tr>
<tr>
<td>Leucine (L)</td>
<td>000000000100000000000</td>
</tr>
<tr>
<td>Methionine (M)</td>
<td>000000000010000000000</td>
</tr>
<tr>
<td>Asparagine (N)</td>
<td>000000000001000000000</td>
</tr>
<tr>
<td>Proline (P)</td>
<td>000000000000100000000</td>
</tr>
<tr>
<td>Glutamine (Q)</td>
<td>000000000000010000000</td>
</tr>
<tr>
<td>Arginine (R)</td>
<td>000000000000001000000</td>
</tr>
<tr>
<td>Serine (S)</td>
<td>000000000000000100000</td>
</tr>
<tr>
<td>Threonine (T)</td>
<td>000000000000000010000</td>
</tr>
<tr>
<td>Valine (V)</td>
<td>000000000000000001000</td>
</tr>
<tr>
<td>Tryptophan(W)</td>
<td>000000000000000000100</td>
</tr>
<tr>
<td>Tyrosine (Y)</td>
<td>000000000000000000010</td>
</tr>
<tr>
<td>Protein Terminal (X)</td>
<td>000000000000000000001</td>
</tr>
</tbody>
</table>

### 3.2.5.2 Amino acid composition

Amino acid composition is the most common way to represent all the features of protein with few discrete numbers. This approach is widely used to predict protein structural classes (Bhasin and Raghava 2004; Bhasin and Raghava 2004; Lata and Raghava 2008; Lata and Raghava 2009), subcellular localization (Bhasin, Garg et al. 2005; Garg, Bhasin et al. 2005; Rashid, Saha et al. 2007; Garg and Raghava 2008; Kaundal and Raghava 2009) and enzyme families (Hua and
Sun 2001; Cai, Wang et al. 2003). The amino acid composition is fraction of each amino acid in a protein. The fraction of all natural 20 amino acids is obtained by using the equation 4. The global feature of protein is represented in terms of 20 scalar values. The amino acid composition of protein will miss all the sequence order and sequence length effect.

\[
\text{Fraction of } AA_i = \frac{\text{total number of amino acid } i}{\text{total number of amino acids of protein}}
\]

[4]

Where, \( i \) can be any amino acid out of 20 natural amino acids.

### 3.2.5.3 Dipeptide Composition

Dipeptide composition is an approach used to encapsulate the global information about each protein sequence in the fixed length format. This gives a fixed pattern length of 400 (equation 5). This representation encompasses the information about amino acid composition along local order of amino acid.

\[
\text{Fraction of dipep (i)} = \frac{\text{total number of dipep(i)}}{\text{total number of all possible dipeptides}}
\]

[5]

Where dipep(i) is a dipeptide i out of 400 dipeptides.

### 3.2.5.4 Tripeptide Composition

Tripeptide composition was used to encapsulate the global information about each amino acid sequence, which gives fixed pattern length of 8,000. This information encompasses the information about amino acid composition along the local order of amino acids. The fraction of each tripeptide was calculated using following equation:

\[
\text{Fraction of tripep (i)} = \frac{\text{total number of tripep(i)}}{\text{total number of all possible dipeptides}}
\]

[6]

Where tripep (i) is a tripeptide i out of the 8,000 tripeptides.
3.2.5.5 Position Specific Scoring Matrix (PSSM)

PSI-BLAST is a very popular tool for detection of distantly related proteins. PSI-BLAST refers to a feature of BLAST in which a profile or position specific scoring matrix (PSSM) is constructed from the multiple alignment of the highest scoring hits in an initial BLAST search (inclusion E-value=0.001). Highly conserved positions receive high scores and weakly conserved positions receive scores near zero. The PSSM thus generated contained the probability of occurrence of each type of amino acid residues at each position along with insertion/deletion. The newly generated profile is used to perform subsequent BLAST search and the result of each iteration used to refine the profile. In this work, the intermediate PSI-BLAST generated PSSM have been used as a direct input to the machine learning techniques. The PSSM has $21 \times M$ elements, where $M$ the length of target sequence and each element is represents the frequency of occurrence of each of the 20 amino acids at one position in the alignment. We normalized each value within 0-1 range using equation:

$$Val = \frac{1}{1 + (2.7182)^{-Val}}$$

(7)

Where val is the PSSM score and Val is its normalized value.

In this thesis work PSI-BLAST was run against NR (non-redundant) protein database with the BLOSUM62 matrix. The intermediate PSSM was used as input for development of FAD interacting proteins and residue prediction methods.

3.2.6 Cross-validation

Cross-validation is a practical and reliable way for testing the predicting power of methods. In $k$-fold cross-validation, the relevant dataset is partitioned randomly into $k$ equally sized partitions. The method of development and evaluation is carried out $k$ times, each time using one distinct partition as the testing set and the remaining $k-1$ partitions as the training set. The pictorial representation of commonly used 5-fold cross-validation has been shown in Figure 3.3.
When the number of groups omitted is equal to the number of the samples in the dataset, the procedure is named "leave one out" or jack-knife validation test. It is the most accurate and extensive method to evaluate the performance of method. This cross-validation test is limited due to lot of time consumption when the size of dataset is large. To reduce the time consumption, more limited cross-validation is performed.

In the present study, Leave-One-Out Cross-Validation (LOOCV) or jack-knife technique has been used to estimate the performance of methods of CYP isoform specificity selection and melting point prediction. For GST prediction, FAD interacting residue prediction, toxicity prediction, melting point prediction, toxicity prediction & classification of CYP 450 isoform specificity, a 5-fold cross-validation has been used. In 5-fold cross-validation, 20% of data will be used for testing of method and remaining 80% will be used for training of method. For the prediction of melting point, 25-fold, 9 fold cross-validation, respectively have been used. The performance of methods is computed as the average of the total runs. This procedure prevents artificially inflated performance values.

3.2.7 Measures for accessing the prediction performance

Evaluating the performance of prediction methods after its development is an important step to test the reliability of that method. Performance of prediction is assessed by counting the
number of correct positive examples (true positive, TP); correct negative examples (true negatives, TN); incorrect positive examples (false positives, FP); and incorrect negative examples (false negatives, FN) as shown in Figure 3.4.

Figure 3.4: A representation of assignment of categories (TP, TN, FP, and FN) to peptides by comparing prediction results with experimental finding.

In case of GST protein prediction, the assignment of categories to peptides is made as follows:

i) TP = True Positive. (Positive examples that are correctly predicted as positives [predicted score > threshold]).

ii) TN = True Negative. (Negative examples that are correctly predicted as negatives [predicted score < threshold]).

iii) FP = False Positive. (Negative examples that are wrongly predicted as positives [predicted score > threshold]).
iv) FN = False Negative. (Positive examples that are wrongly predicted as negatives [predicted score< threshold]).

The performance measures for the measurement of success of prediction methods have been derived from these four categories. The performance measures can be divided in two- threshold dependent or threshold independent measures.

3.2.7.1 Threshold independent measures – ROC

The receiver operating characteristic is a threshold independent measure that has been developed as signal processing technique. The term refers to the performance ('operating characteristics') of a mechanical observer (the receiver) engaged in assigning cases to classes. For a prediction method, ROC plot is obtained by plotting all the sensitivity values on y-axis against their equivalent (1-specificity) values for all available thresholds on x-axis. The curve always goes through two points (0, 0 and 1, 1). 0, 0 is where classifier finds no positives. In this situation, all the negative cases are predicted correct but all the positive cases are predicted wrong. The second point is 1, 1 where everything is classified as positive. So the classifier predicts all the positive cases right, but predicts all the negative cases wrong. The method that randomly guesses has the ROC which lies somewhere along the diagonal connecting the line of 0, 0 and 1, 1. The area under an individual ROC curve (AUC) is a good measure of overall performance of a prediction system. Values of A between 0.7 and 0.9 indicate good and those above 0.9 indicate excellent accuracy of predictions. Values of A between 0.5 and 0.7 indicate poor accuracy. In this study, we have not used the threshold independent measure to estimate the performance of method.

The statistical program ‘Statistical Package for the Social Sciences’ SPSS version 11.0 for windows) has been used to calculate ROC values of prediction methods. It generates a single ROC curve and calculates the area under the curve.

3.2.7.2 Threshold dependent measures

The performance is assessed at different threshold or cutoff values using different parameters such as sensitivity, specificity, accuracy, MCC as described in literature (Brusic, Rudy et al.
Sensitivity: It is the percent of GST protein that are correctly predicted as GST proteins. It is also known as percent coverage. Higher the sensitivity means that almost all of the potential positive example will be included in the predicted results. However, at the same time, some of the non-GST proteins will also be predicted as GST proteins.

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \times 100
\]  

ii) Specificity or Recall: Specificity is the percentage of non-GST proteins, which are correctly predicted as non-GST proteins.

\[
\text{Specificity} = \frac{TN}{TN + FP} \times 100
\]

iii) Accuracy: Accuracy is the percentage of correctly predicted GST proteins and non-GST proteins. It is a good measure to assess the performance of any method when the dataset is balanced (equal number of positive and negative examples). However, it is biased when data set is imbalanced. The value of accuracy would be higher for thresholds favoring the correct prediction of FAD interacting residues (if number of FAD interacting residues is more than non-FAD interacting residues) or non-FAD interacting residues (if the number of non-FAD interacting residues is more than FAD interacting residues).

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}
\]

iv) Matthews’s correlation coefficient (MCC): It is the best performance measure that accounts for both over and under predictions. Therefore, it is a best measure to compare the performance of imbalanced dataset. The value of MCC ranges from −1 (perfectly anti-correlated) to 1 (perfectly correlated).
Chapter 3: Material and methods

\[ MCC = \frac{(TP \times TN) - (FN \times FP)}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}} \]  

3.3 Chemoinformatics

3.3.1 Preparation of dataset

Any prediction tools in chemoinformatics require creation of a clean and high quality experimentally annotated dataset. In present study, we used chemical dataset which are already reported in literature. For melting point prediction we directly used dataset which is already used by Kartikeyan et al. (Karthikeyan, Glen et al. 2005), in other cases we searched chemical molecules in literature and then used for QSAR model development. We removed all redundant chemical molecules from dataset and used non-redundant dataset for this study.

3.3.2 Preprocessing of dataset

Once we drew these molecules in ChemBioOffice, we saved the mol file of these molecules. Then we converted these molecules in 3D structure using CORINA and minimized energy using MMFF94. We removed those molecules from our dataset which were converted in 3D structure.

3.3.3 Techniques used to develop prediction method

Artificial neural network (ANN)

ANN is an elegant machine learning technique widely used for the classification and pattern recognition tasks. The idea of neural network originated from biological networks. There is an analogy between neurons in brain and ANN in the ability of learning. ANN is connectionist model used extensively for protein secondary structure prediction, gene prediction and analysis of microarray data. The neural network consists of a large number of units, which are similar to neurons in the brain. It is characterized by architecture and the pattern of connection between neuron.

Neural networks are typically organized in layers. These layers are made up of a number of
interconnected “neurons” which contain an “activation function”. Each unit receives the input, which is then multiplied with a certain weight depending on the algorithm. This is also known as synaptic weights. Each unit has an internal state, called its activation function or activity level, which is a function of the inputs it has received. Typically, a neuron sends its activation as a signal to several other neurons. Sequence patterns are presented to the network via the “input layer”, which communicates to one or more “hidden layers” where the actual processing is done via a system of weighed connections. The hidden layers then link to an “output layer” where the answer is an output (Figure 3.5).

For example, let us assume that the neuron $Y$ receives input (activation) from $n$ neurons $X_1$, $X_2$, $X_3$......$X_n$. From each input neuron to the output neuron $Y$, there is a connection with an associated weight; the connection from $X_i$ to $Y$ has weight $w_i$.

To determine the output of neuron $Y$, we sum all the inputs combined with their respective weights (Equation 12):

$$Y_{in} = b + w_1X_1 + w_2X_2 + .... + w_nX_n$$  \[12\]

![Figure 3.5: Structure of a neural network](image)

The bias $b$ allows one to change the output independently of the inputs. Then an activation function to this sum is applied. Many different functions may be used as an activation function. A
common activation function is the logistic sigmoid function (an S-shaped curve) shown in equation 10.

\[ y = f(Y_m) = \frac{1}{1 + \exp(-Y_m)} \]  

[13]

The output of this function is the output (activation) of neuron \( Y \), which is often connected as an input to other neurons.

Most ANNs contain some form of "learning rule" which modifies the weights of the connections according to the input patterns that it is presented with. There are many different kinds of learning rules used by neural networks, the common being the delta rule. The delta rule is often utilized by the most common class of ANNs called "backpropagation neural networks". With backpropagation, learning is a supervised process that occurs with each cycle or epoch (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs and the backwards error propagation of weight adjustments.

In the present work for the neural network implementation, the publicly available free simulation package SNNS version 4.2 from Stuttgart University has been used (Zell 1998). It allows incorporation of the resulting networks into an ANSI C function for use in stand-alone code. A linear activation function is used. At the start of each simulation, the weights are initialized with the random values. The training is carried out using error back-propagation with a sum of square error function (SSE). The magnitude of the error sum in the training and testing set is monitored in each cycle of the training. The ultimate number of cycles is determined, where the network converges. During the testing of network, a cutoff value is set for each network and the output produced by the network is compared with the cutoff value. If the output is greater than the cutoff value, then that residue is taken as true otherwise it is considered as false. The cutoff value is adjusted so that it yields the highest accuracy for the network. In the present work, neural network has been used for the prediction of melting point FAAH inhibitors binding affinity prediction.
K-Nearest Neighborhood (KNN)

In KNN, the binding affinity of each peptide was predicted as the algebraic average activity of its $k$-nearest-neighbor compounds in the training set. In this, Euclidean distances in the input pattern space between the compound and each of the $k$ nearest neighbors are not same. Thus, neighbor which has smaller distance with a peptide is given a higher weight in calculating the predicted activity using given equation-

$$w_i = \frac{\exp(-d_i)}{\sum_{i=0}^{k} \exp(-d_i)}$$  \hspace{1cm} (14)

$$\bar{y} = \sum w_i y_i$$  \hspace{1cm} (15)

Where $d_i$ is the Euclidean distance between the compound and its $k$ nearest neighbors, $w_i$ is the weight for each individual nearest neighbors; $y_i$ is the actual activity value and $\bar{y}$ is the predicted activity value. K-Nearest Neighbor was implemented using the free software TiMBL (Tilburg Memory-Based Learner) for melting point prediction in our study.

### 3.3.4 Input pattern and feature selection

We calculated the molecular descriptors of chemical molecules which were further used in different machine learning techniques for QSAR model development. In present study we used more than 3500 molecular descriptors which are calculated by using Molinspiration, PreADMET, TSAR, ADMEWorks Model builder. Out of these ~3500 molecular descriptors, lot of descriptors having zero value or several descriptors are same. So remove all those descriptors which have missing value, zero in most of the cases and highly correlated. For this feature selection we used ADMEWorks model builder, WEKA and Rapidminer. We further remove molecular descriptors using several other unsupervised methods as Linear forwards selection, Ranker, Random, GA, PCA, Greedy search etc. We removed these highly informative molecular descriptors for QSAR based model development.
3.3.5 Cross-validation

Cross-validation is a practical and reliable way for testing the predicting power of methods. In \( k \)-fold cross-validation, the relevant dataset is partitioned randomly into \( k \) equally sized partitions. The method of development and evaluation is carried out \( k \) times, each time using one distinct partition as the testing set and the remaining \( k-1 \) partitions as the training set. In present study we used different fold cross-validation as four, five, ten, twenty five and leave one out etc.

3.3.6 WEKA (Waikato Environment for Knowledge Analysis)

The Weka 3.6.0 package (Frank, Hall et al. 2004) contains a collection of visualization tools and algorithms for data analysis and predictive modelling, together with graphical user interfaces for easy access to this functionality developed at University of Waikato, New Zealand (Frank, Hall et al. 2004). The system is written in Java, a platform independent object oriented programming language. Weka supports several standard data mining tasks, more specifically, data pre-processing, clustering, classification, regression, visualization, and feature selection. All of Weka's techniques are predicated on the assumption that the data is available as a single flat file or relation, where each data point is described by a fixed number of attributes (normally, numeric or nominal attributes, but some other attribute types are also supported). Weka includes lot of statistical and machine learning techniques as C4.5, SVM, KNN, logistic regression, multiple layer regression (MLR), BayesNet, neural network, RandomForest etc.

3.3.6.1 Weka Data representation

The Weka learning algorithms have been trained on molecular descriptors of small chemical molecules. The data is represented in ARFF (Attribute Relation File Format) that consists of a list of all instances, with the attribute value for each instance being separated by commas. A @relation tag introduces the dataset names, types and value of each attribute for each instance. The data section of the ARFF file begins with a @data tag.

3.3.6.2 Weka: Learning Algorithms

The following learning algorithms have been used in this study in Weka (Frank, Hall et al. 2004):
Genetic Algorithm

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimize and search problems (Rada 1982). Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of evolutionary algorithms (EA) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection and crossover. GA is implemented in a computer simulation in which a population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Greedy Stepwise Search

Greedy stepwise search (Merz and Freisleben 2000) performs a greedy forward or backward search through the space of attribute subsets. It may start searching with no/all attributes or from an arbitrary point in the space, stops when the addition/deletion of any remaining attributes results in a decrease in evaluation. It can also produce a ranked list of attributes by traversing the space from one side to the other and recording the order that attributes are selected.

Logistic Regression

Logistic regression is a variation of simple regression, useful when the observed outcome is restricted to two values, which usually represents the occurrence and non-occurrence of some outcome event, usually coded as 1 or 0 respectively. It provides a formula that predicts the probability of the occurrence of a function of independent variables. It fits a S-shaped curve by taking the linear regression, which could produce any y value between $-\infty$ and $+\infty$, and
transforming it with the function

\[ p = \frac{e^y}{1 + e^y} \]  

which produces p value between 0 (as \( y \) approaches \(-\infty\)) and 1 (as \( y \) approaches \(+\infty\)).

**Multiple Regression (MLR)**

The general purpose of multiple regression is to learn more about the relationship between several independent or predictor variables and a dependent or criterion variable. This is the advanced form of simple linear regression.

**J48 Decision Tree**

The C4.5 algorithm generates a classification-decision tree for the given data set by recursive partitioning of data. The algorithm considers the entire possible test that can split the data set and select a test that gives the best gain in information. For each discrete attribute, one test with outcomes as the number of distinct values of the attribute is considered. For each continuous attribute, binary test involving every distinct value of the attribute are considered. In order to gather the entropy gain of all these binary test efficiently, the training data set belonging to the node in consideration is sorted for the values of the continuous attribute and the entropy gains of the binary cut based on each distinct values are calculated in one scan of the sorted data. This process is repeated for each continuous attribute.

**RandomForest**

Random forest is an ensemble classifier that consists of many decision trees and output is the mode of the class’s output by individual tree. The algorithm for including a random forest was developed by Leo Breiman (Breiman and Leo 2001). The method combines Beriman’s “bagging” idea and random selection of features, introduced independently (Ho and Tin 1995; Amit and Geman 1997; Ho and Tin 1998) in order to construct a collection of decision tree with controlled variation. The selection of a random subset of features is an example of the random
subspace method, which, in Ho’s formulation, is a way to implement stochastic discrimination (Kleinberg and Eugene 1996).

Each tree is constructed using the following algorithm:

1- Let the number of training cases be N, and the number of variables in the classifier be M.

2- We are told the number m of input variables to be used to determine the decision at a node of the tree; m should be much less than M.

3- Choose a training set for this tree by choosing N times with replacement from all N available training cases (i.e. take a bootstrap sample). Use the rest of the cases to estimate the error of the tree, by predicting their classes.

4- For each node of the tree, randomly choose m variables on which to base the decision at that node. Calculate the best split based on these m variables in the training set.

5- Each tree is fully grown and not pruned (as may be done in constructing a normal tree classifier).

BayesNet

A BayesNet is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG). For example, a Bayesian network could represent the probabilistic relationships between biological activity and molecular descriptors. Given descriptors, the network can be used to compute the probabilities of biological activities.

Formally, Bayesian networks are directed acyclic graphs whose nodes represent random variables in the Bayesian sense: they may be observable quantities, latent variables, unknown parameters or hypotheses. Edges represent conditional dependencies; nodes which are not connected represent variables which are conditionally independent of each other. Each node is associated with a probability function that takes as input a particular set of values for the node's
parent variables and gives the probability of the variable represented by the node. For example, if the parents are m Boolean variables then the probability function could be represented by a table of 2m entries, one entry for each of the 2m possible combinations of its parents being true or false.

**NaïveBayes**

NaïveBayes algorithm implements Bayesian classification based on Bayes theorem of conditional probability. The theorem is used to estimate the probability of an example belonging to each of the possible classes of a classification problem. A Bayesian classifier assumes conditional independence of the attributes of a given class (i.e. the attributes are independent of one other). Learning a Naïve Bayes classifier consists of estimating the prior probabilities for each of the possible value of the class and each of the attributes in the example.

**Multilayer Perceptron**

A multilayer perceptron is a feed forward artificial neural network model that maps sets of input data onto a set of appropriate output. It is a modification of the standard linear perceptron in that it uses three or more layers of neurons (nodes) with nonlinear activation functions, and is more powerful than the perceptron in that it can distinguish data that is not linearly separable, or separable by a hyperplane (Cybenko 1989).

**Sequential Minimization Optimization (SMO)**

SMO (Platt 1998; Knebel, Hochreiter et al. 2008) is a new algorithm for training SVM which is implemented in WEKA. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalizes all attributes by default. (In that case the coefficients in the output are based on the normalized data, not the original data. This is important for interpreting the classifier.)

Multi-class problems are solved using pair wise classification. To obtain proper probability estimates, use the option that fits logistic regression models to the outputs of the support vector machine. In the multi-class case the predicted probabilities are coupled using Hastie and Tibshirani's pair wise coupling method. It also performs very well in regression (Shevade,
3.3.7 Measure of performance

3.3.7.1 Pearson's Correlation (R)

The correlation between predicted and experimentally determined binding-affinities of FAAH inhibitors, chemical kinase inhibitors and melting point of small chemical molecules has also been used to assess the performance. It is a useful measure to determine the performance of methods where the prediction results are real values instead of two or three classes. The correlation coefficient is a measure of how well trends in the predicted values follow trends in the actual values. It is a measure of how well the predicted values from a model "fit" with experimentally proven data. The correlation coefficient is a number between 0 and 1. If there is no relationship between the predicted values and the actual values, the correlation coefficient is 0 or very low (the predicted values are no better than random numbers). With the increase in strength of the relationship between the predicted values and actual values, the correlation coefficient also increases. A perfect fit gives a coefficient of 1.0. Thus, a higher the correlation coefficient is better. The value of correlation coefficient is calculated using equation 25.

\[
R = \frac{n(\sum y_i f_i) - (\sum y)(\sum f)}{\sqrt{n(\sum y_i^2) - (\sum y)^2} \sqrt{n(\sum f_i^2) - (\sum f)^2}}
\]  \[17\]

3.3.7.2 Coefficient of determination (R²)

In statistics, the coefficient of determination, \( R^2 \) is used in the context of statistical models whose main purpose is the prediction of future outcomes on the basis of other related information. It is the proportion of variability in a data set that is accounted for by the statistical model. It provides a measure of how well future outcomes are likely to be predicted by the model. \( R^2 \) calculated using the following equation:

\[
R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - f_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}
\]  \[18\]
A data set has values $y_i$, each of which has an associated modeled value $f_i$, and $\bar{y}$ is the average of test set. Here, the values $y_i$ are called the observed values and the modeled values $f_i$ are sometimes called the predicted values. The "variability" of the data set is measured through different sums of squares:

$$SST = \sum_{i=1}^{n} (y_i - \bar{y})^2$$ \[19\]

the total sum of squares (proportional to the sample variance);

$$SSR = \sum_{i=1}^{n} (f_i - \bar{f})^2$$ \[20\]

the regression sum of squares, also called the explained sum of squares.

$$SSE = \sum_{i=1}^{n} (y_i - f_i)^2$$ \[21\]

the sum of squared errors, also called the residual sum of squares.

In the above, $\bar{y}$ and $\bar{f}$ are the means of the observed data and modelled (predicted) values, respectively. That is:

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$ \[22\]

$$\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f_i$$ \[23\]

Where $n$ is the number of observations.
Hence

\[ R^2 = 1 - \frac{SSE}{SST} \]  \hspace{1cm} [24]

\( R^2 \) is a statistic that will give some information about the goodness of fit of a model. In regression, the \( R^2 \) coefficient of determination is a statistical measure of how well the regression line approximates the real data points. An \( R^2 \) of 1.0 indicates that the regression line perfectly fits the data.

### 3.3.7.3 \( Q^2 \)

\( Q^2 \) is a very popular statistical parameter which is commonly used in QSAR studies. Its value ranges from 0 – 1. If value of is \( Q^2 > 0.5 \) then we can say that our model is highly predictive. We calculate \( Q^2 \) using following equation:

\[
Q^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - f_i)^2}{\sum_{i=1}^{n}(y_i - y_{train})^2} \]  \hspace{1cm} [25]

\[
y_{train} = \frac{1}{m} \sum_{i=1}^{m} y_i \]  \hspace{1cm} [26]

Where \( y_{train} \) is the average of training set value.

### 3.3.7.4 Mean Absolute Error (MAE)

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 (MAE) is given by:

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |y - f| \]  \hspace{1cm} [27]
3.3.7.5 Root Mean Square Error (RMSE)

The root mean square error (RMSE) is a frequently-used statistical measure of the differences between values predicted by a model or an estimator and the values actually observed from the thing being modeled or estimated. RMSE is a good measure of predictive power of model. These individual differences are also called residuals, and the RMSE serves to aggregate them into a single measure of predictive power. We can calculate RMSE using this equation:

\[
RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y - f)^2}
\]  

[28]

3.3.7.6 Root Mean Square Cross-validation (RMSECV)

RMSECV is the aggregate root mean squared error (RMSE) of the cross-validation. RMSECV for M fold cross-validation calculated using following equation:

\[
RMSECV = \sqrt{\frac{1}{M} \sum_{i=1}^{M} (RMSE)^2}
\]  

[29]

3.4 Software requirements

All prediction methods in the study are developed using PERL and Java languages. The front end of all methods has been designed using HTML. All the graphics have been generated using Photoshop 5.0. The back end programming has been done using PERL and Java languages. The CGI PERL has been used as an interface between front and back end. All methods have been launched on World Wide Web by implementing apache and tomcat-apache server. All methods are freely available to public from specified URL.