PD Tool and Comparing Classification Algorithms on Physical Design Datasets

Waseem Ahmed

Graduate Co-op Work Term Report

Department of Computer Science

University of Regina
Regina, Saskatchewan

Supervisor
Dr. Lisa Fan

Date: 20 May 2014

Dr. Lisa Fan
Computer Science Department
University of Regina
Regina, S4S 0A2 Saskatchewan

Dear Dr. Lisa Fan,

Herewith enclosed is the report entitled "PD Tool and Comparing Classification Algorithms on Physical Design Datasets". The report explains one of my major works in Cisco Systems during my co-op work terms in which I have performed studies and experiments to improve the performance of classification algorithms on Physical Design datasets. The report is the representative of my accomplishments over the work terms and demonstrates the ability to integrate programming and data mining skills into a business environment.

At Cisco Systems, I worked with the Hardware Engineering department which is responsible for building, verifying, validating physical design of chips. My role in the team was to developed intelligent tool for data collection and database development for the physical design process. It provides time estimation and forecasting for chip designing projects and helps in design optimization models of current and future projects.

The enclosed report is an entirely original report, created solely by myself based on my work and research I was involved in. Due to confidentiality of work, I am only able to provide the basic insight of PD tool. The second section of report provides implementation of three classification algorithms on physical design datasets and discusses their accuracy and precision results.

Sincerely,

Waseem Ahmed

Executive Summary

PD (Physical Design) tool is designed primarily to assist Cisco's chip design engineers in achieving design optimization. It uses data mining techniques to handle the existing unstructured data repository. The tool extracts the relevant data and load into a well-structured database, so that it may be used for future reference. It also has an archive mechanism that initially creates and then keeps updating an archive repository on a daily basis.

The original input to the PD tool is a completely unstructured datasources which are read by the tool using regular expression based data extraction methodology. By doing this, PD tool converts the input data into the structured tables. This undergoes data cleansing process before fed into the operational DB. By maintaining an archive repository of this, PD tool also ensures data integrity and data validity.

PD tool helps the design engineers to compare, correlate and inter-relate the results of their existing work with the ones done in the past which gives them a clear picture of the progress made and deviations that occurred. Data analysis can be made using various features offered by the tool such as graphical representation and statistical representation.

The second part of the report focuses on the performance of classification method on PD datasets. The classification method used is based on continuous learning approach that uses existing learning algorithms as the backbone. The method classifies the data based on past results which helps saving time and resource which was earlier met with bad designs.

In detail, each chip is made of 4 subchips divided further into 40+ floor-plan blocks normally. Each of these blocks has corners whose design plays a key role in the performance of a chip. While designing a chip, the engineer finishes designing a single corner, and the parameter data is fed through the classifier method which classifies the data into six different subclasses based on its learning. Data records have classified using three classification methods such as Naïve Bayes, Decision Tree and Multilayer Perceptrons classifiers and selects best fit classifier on PD datasets.

Table of Contents

1 Introduction

1.1 Problem Statement

IC (Integrated Circuit) or processing chip manufacturers have been facing challenges in physical design optimization such as size, performance and power efficiencies since last decade. Physical design phase has been taking too long to drive to final (tapeout) release of a model. The aim of this project is to learn from past experiences and efficiently optimize physical design phases for new and under development IC chips. In fact, physical design engineers would be able to interrelate and correlate parameters, estimate and predict physical design models in efficient manner using the results from data mining techniques.

1.2 Physical Design Tool

The PD (physical design) tool details about creation of a single point solution for data gathering, data structuring, error checking and report generation for IC chip production and design. The solution once created is being used as the benchmark for quality tracking and model estimation as the test data from all the chips at the moment of designing, when fed to the proposed solution, aims to generate a successful report without any error. The proposed solution is a great technological leap for the physical design engineers as they no longer have to manually check the test logs from several data servers. Previously, there was no solution available to accomplish this task as the data to be tested was in a raw and poorly structured form.

The tool was written using a combination of different scripting languages such as PHP, PERL and other technologies like Oracle DB, Smart GWT library, JSON and Perl excel library. The solution is designed as an ETL (extract, transform and load) architectural design pattern based solution. Agile programming strategy was chosen for this project and the goal was broken down into two major modules.

The first module performs the tasks needed for data extraction from the unstructured data source followed by data cleansing, data transforming and finally data staging which provides a well-structured dataset to the solution's database. The second module is designed to support user-data controlling, data validation, data visualization and report generation.

The project's ecosystem is the entire physical design department of IC manufacturing unit in CISCO SYSTEMS. The design engineers are currently working on designing ANTARES chip. An

ANTARES chip consists of four subchips each responsible for a specific functionality. Each of these subchips comprises of numerous floorplan blocks. The design engineers are responsible to build as well as verify each floorplan block. They analyze parameters of the chip like the placement of blocks (number of devices known as flopcounts, number of clocks) and standard cell area, metal pins, clock and closure timing, measure clock latency and skewness, layout versus schematic, design rule check, electrical rule check and so on. While testing, the main focus is on Primetime corners such as I/O-setup/hold, Core setup/hold time, Design Rule Checking ICV's, ICC, Layout Versus Schematic (LVS), Primetime Electric Rule Check (ERC).

At the end, the PD engineers store the tapeout values and synopses VCS log files on an unstructured data repository and store text files in a different floorplan blocks directory on the servers. This information is updated by synchronically compile and test logics and designs. A main drawback of this existing set up is that there are no means available for archiving test data.

Efficiency, Power and Performance is the basic underlying parameters to measure chip quality in present day ASICs (application-specific integrated circuits) development process. Physical design engineers are responsible for all phases of chip performance from RTL design (Register Transfer Level) to delivery or the next stage of chip. For example, Cisco's physical design engineers who work from different geographical locations currently use different tools to verify physical design of chip such as Synopses Hercules, LVS, DRC and VCS. They are going to use PD tool to gather and analyze data at central location and PD tool is contributing greatly to optimize their chip's design and processes.

1.3 Introduction of Data Mining

Data mining, in this project, is used to estimate and predict model changes at early stages. In addition, PD engineers would be able to identify and predict the floorplan blocks before marking them as tapeout.

With the help of data mining techniques, we have used the datasets extracted from physical design database and classified this dataset's records using Cisco's given set of rule (which are called subclasses). We then implemented three well-known classification algorithms and selected appropriate classification algorithm under the review of accuracy and precision. Once the model is created, it could be used for estimation and prediction.

2 Background Review and Proposed PD Tool

2.1 Background Review

Cisco System's Hardware engineering teams have been working on under development chip called ANTARES (22 nanometers in size) since last two years. Physical design (PD) engineers have been working from multiple locations and building up the logic in more than 140 floorplan (FP) blocks and verifying and validating the logics and design using Synopsys tool and storing the results on their end servers in logs such as crystal report and excel sheets. Before the PD tool, there was no system available for automation that could gather and extract meaningful data from flat file and store it to centrally located database on daily basis. Most of the time, PD engineering teams worked on different sub-chip levels, which directly or indirectly affects the other parts of chip or its parameters such as power, efficiency and quality. And the problem was the absence of an automation mechanism that could correlate changes and analyze trends between floorplan blocks and their parameters.

Most of the time, Hardware team manager marks FP blocks as tapeout (tapeout is the final result of the design cycle) based on current outcomes. Each block is released in several versions. The PD engineers then assign a unique label to each version. For instance arl_fp block can be assigned labels such as label3537, label3538 and label3539. Each label corresponds to a running experiment in the system. If manager marked label3539 as tapeout because of good outcomes and label3527 and label3538 are still under observation and in future current tapeout version i.e. label3539 destabilizes due to ERC, DRC, setup and hold time violations, then they couldn't interrelate and correlate between experiment versions and tapeout version due to unavailability of historic data. Due to this lack of system, Cisco decided to work on big data analysis of physical design which led to PD tool software and database development. PD tool is a part of big data analysis which is responsible to extract and transform meaningful data from several data sources and load into centrally located database.

PD tool has been collecting data since November 2013. However, legacy system was manually extracting data from flat file and displayed it on html based view. Legacy system had several drawbacks. For example, overall status of PD process could not be checked as the html view only showed limited information. Even, it was not possible to look past run status because every time flat

file data had swapped with new data. At each time request, html view took too much time to render information from file. Even, PD engineers cannot detect interrelated violations.

Ultimately, PD tool provides the feature to store meaningful data in database. In fact, PD tool provides the interface to manager to look at the historic data as well as experiment data which give opportunity to compare and correlate information. This tool also provides the user data interaction with Big Data. With the help of Big Data, manager would be able to estimate and predict the work intensity and performance using data mining techniques.

The second part of report conducted experiment on sixteen datasets fetched from PD database and classified them using three classification algorithms i.e. Naïve Bayes, Decision Tree, and Multilayer Perceptrons (description of datasets and algorithms is given in implementation section). In this project, datasets from PD database at four subchip levels were extracted and every subchip had four types of information which means 4 (subchips) x 4 (types) equal to sixteen datasets. Finally, appropriate classification algorithm by comparing their accuracy and precision gets selected.

Although several studies have been conducted on comparing classification algorithm's performance on different types of datasets, very few have focused on same type of datasets. However, the most extensive work was carried out on numerous datasets in STATLOG project (1995) by King, Feng and Sutherland. They argued on interpretation results, the variation in the data and the algorithm used in different type of datasets [1]. In 1997 Hand and Henley conducted an experiment of different statistical classification approaches used for credit rating datasets [2]. Berardi, Patuwo and Hu presented a principle approach for building and evaluating neural networks models for e-commerce datasets which's done in 2004 [3].And also, Classification algorithms have been used in the medical field. For example, Eftekhar et al. (2005) compared the performance of artificial neural network and multivariable logistic regression models in prediction of outcomes of head trauma [4]. In 2002, J. Eng has done experiment of neural network with one hidden layer and multivariate logistic regression on pulmonary embolism datasets [5]. The objective of the above discussion is to show the work that has been done on multiple datasets with different domains. But my concentration of work is on multiple datasets with same domain.

The main purpose of this report is to determine which classification method would perform better on PD datasets.

2.2 Proposed PD Tool

The PD tool is based on Extract, Transform and Load (ETL) architectural design pattern. ETL design pattern approach is mainly split into three processes:

1) Process of retrieving data (using Regex) from unstructured data sources called **Extraction**.
2) **Transformation** process to convert extracted data according to target database by using set of rules or by mapping, joining and aggregating the data with other data [6].
3) And finally, the process of writing the data into the target database called **loading** or **data staging**.

In the PD tool, we customized these three ETL processes as per industry requirement and we enhanced PD tool with new features such as user data controlling process and UI based data presentation and visualization process.

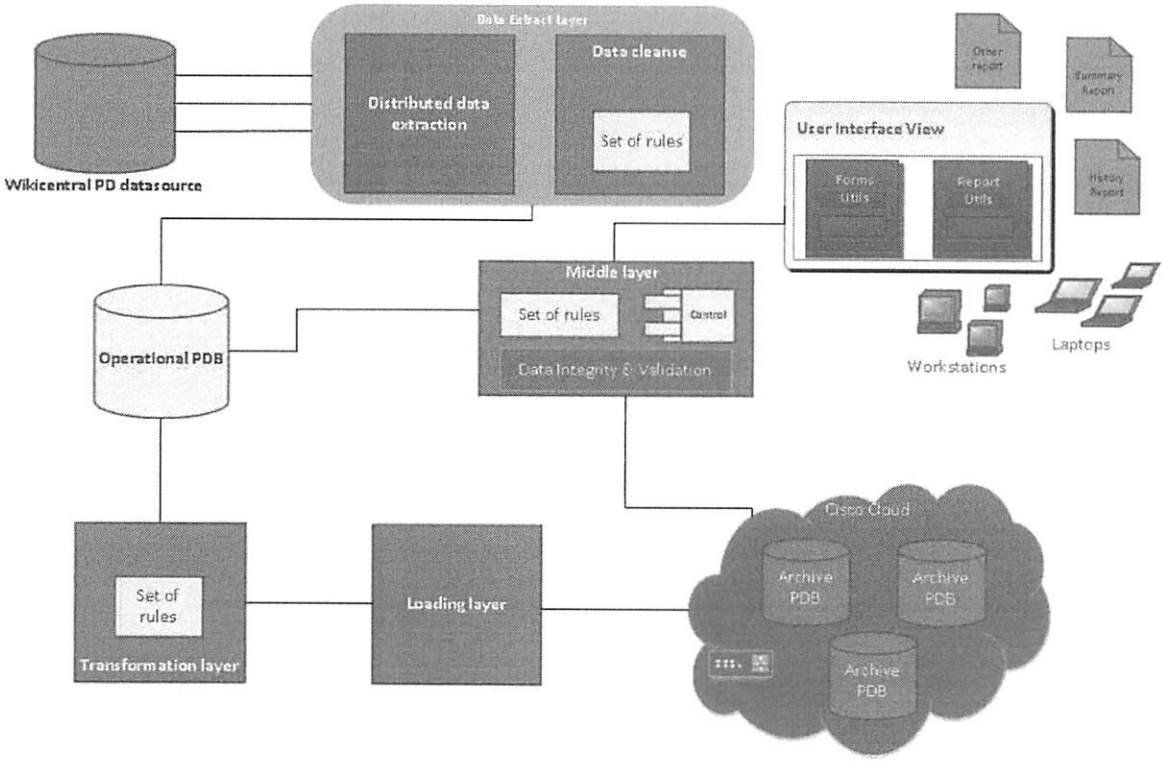

Figure 2.1 PD Tool Structural Framework

Figure 2.1 shows main components and structural layout of PD tool. Wikicentral is the existing physical design datasource which has data in unstructured format such as txt, rpt, xls, xlsm files. This datasource is made of a group of servers connected with nodes. First component of PD tool, shown as "Data extraction layer" in the diagram, is responsible to extract meaningful data values from flat files with the help of regex (regular expressions) and other techniques. Data cleansing process is linked as an extension to data extraction layer which rectifies inappropriate data and inconsistent types with the help of set of rules and predefined functions. Once the cleansing process is complete, the data is pushed down into operational database.

Before transferring selected information from operational database to archive (data warehouse) database, users have the opportunity to verify the quality and integrity of the data by generating summary and detail reports from operational database. In addition to this, extraction layer executes four times a day meaning that the operational database updates data in every six hours. Data controlling layer is used to control data flow by simply changing the status from experimental to tapeout or historic. At the end of day, transformation layer performs processes such as data filtering, data mapping and data aggregation techniques and eventually load the resultant data into archive database. Data visualization, another key component of the PD tool, generates historical data compare reports between dates and revisions. This helps the users to easily find trends and correlate changes using tabular and graphical representation.

2.2.1 Distributed Data Extraction Layer

Distributed data extraction layer consists of modules for data extraction process, data cleansing process and is also responsible to store data into operational database. This layer has the features of a distributed computing environment. In distributed data extraction environment, each process executes on a different server and has its own separate memory (or distributed memory) and processing unit.

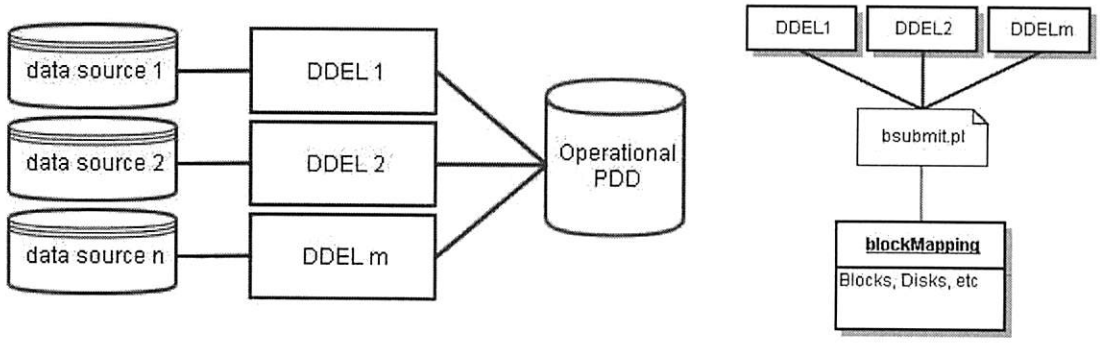

Figure 2.2: Distributed data extraction layer

In Figure 2.2, (n) number of data extraction processes access (m) number of data sources and push down data centrally into operational PD database. This mechanism allows inserting bulk information in less time. This layer executes four times a day using cronjob (Linux scheduler) and this layer also has capability of Load sharing facility (LSF) workload management system.

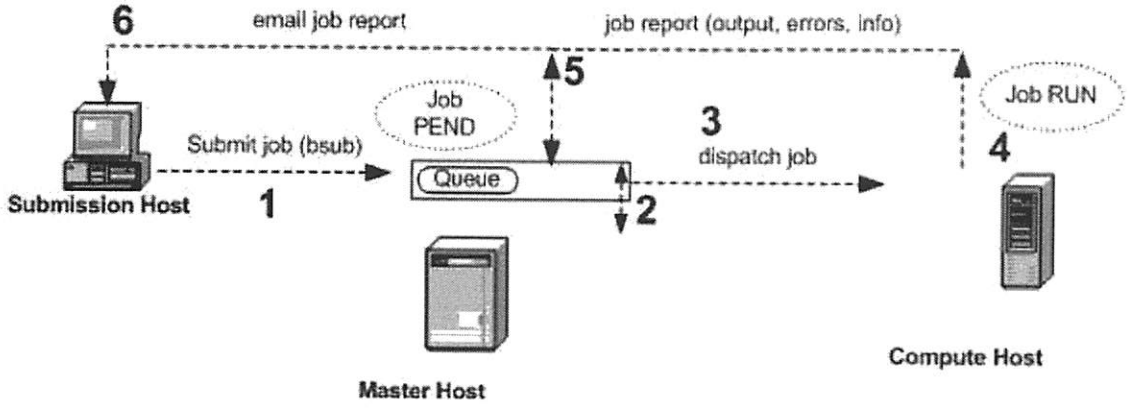

Figure 2.3 Load sharing facility workflow system [7]

Above depicted in Figure 2.3, load sharing facility (LSF) is a workload management platform and job scheduler for distributed HPC environments [8]. It executes batch jobs on networked Linux systems. A job is submitted from one of the head nodes and waits until resources become available on the computational nodes. Once the job is done, it returns the job report (output/error) to user email address.

Algorithm I bsubmit.pl

```
1: procedure submit_job
2:   %records=load (blockMapping_table)    //fetch all record
3:   $cnt=0;
4:   for %records do    //read table by line!
5        //submit job using LSF bsub commands
6:       system ("bsub extractionProcess.pl $_[Blocks] $_[Disk]");
7        $cnt++;
8:  end for
9:  print "$cnt jobs submitted"
10: end procedure
```

Algorithm I: Load sharing facility workflow script

The "bsumbit.pl" script embeds with cronjob and simply executes Perl script every six hours and passes floorplan blocks and disks (server location) as argument which is retrieved from "blockMapping" table and submit job using bsub command. The bsub command submits job to a queue and runs the job on a compute node or nodes that satisfies all requirements of the job, when all conditions on the job, host, queue, and cluster are satisfied.

2.2.1.1 Data Extraction

According to ETL definition, Data extraction is the process in which data is extracted from unstructured text files or poorly structured tables datasources [9].

Each PD datasource has bulk of files which have information about design layout, statistic timing, power optimization and performance of chip. These are distinct set of characteristics that need to be managed in order to extract data effectively for ETL process. Split function and regular expression (Regex) have been mainly used in data extraction process which easily identifies small or large-scale structure [9].

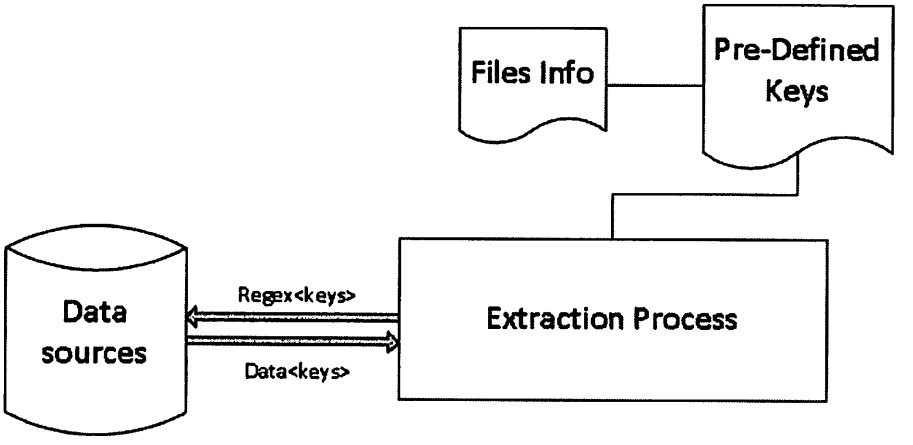

Figure 2.4 Data extraction process system flow diagram

As shown in Figure 2.4, extraction process uses keys/regex data from pre-defined keys object. File info object has information about table name and filename, directory path get as argument from bsubmit.pl. Data key part shows column name and value part shows extracted data-values.

2.2.1.2 Data Cleansing and Operational Database

Data cleansing, also called data cleaning and data scrubbing, deals with finding and rectifying inappropriate and inconsistent data in order to improve the data quality [10]. Data quality problems can occur in operational database due to file missing, file accessing problem, invalid or incomplete data in file. In order to provide accurate and consistent data, PD tool used some predefined set of rule for data cleaning such as Not Applicable or N/A replace by -999, unknown replaced by -1, Error ignored/clean replaced by 0, error message replaced by -777 etc. It simply replaces in coming values

with these set of rule at insertion time. The set of rules can be manipulated from PD form where user can add or delete new rules. Null values are replaced by '--'and type cast the string declare number into integer to make sure getting correct data. Convert the date and time as system standard format such as 'DD/MM/YYYY HH24: MI: SS'

Operational database (based on oracle) completely normalizes database which meet second form of normalization. Each PD process has individual table and appropriate lookups table which help to sort out one-to-many relationship. After the cleansing process, data is ready to be loaded into operational database and iteratively saves the record. If floorplan already exists; it is updated and if new floorplan comes, it creates a new entry.

2.2.2 Data Transformation layer

The second layer in PD tool is data transformation. Typically, the data transformation consists of data cleansing, converging and integration processes which tend to make some cleaning and conforming on incoming selected data to gain data which is correct, complete, consistent and unambiguous[11].

Transformation layer also uses set of rules for example if incoming data have '-'or null or 'No ERC file' then copy the previous record (n-1) values from archive database. Every transformation process has conditional data mapping and data aggregation mapping.

2.2.2.1 Data mapping

PD Operational database saves 663 unique attributes for each floorplan block. However, PD chip has 140 floorplan blocks which leads to 92820 unique data. In data column mapping, we capture 36 data attributes (some of them are aggregate of corners, checks and clocks) in archive database on daily basis.

In Figure 2.5, main table connects with lookup tables using Left Outer Join. The Left Outer Join logical operator returns any rows from the left side table that had no matching rows in the right side tables. The nonmatching rows in the right side tables are returned as null values.

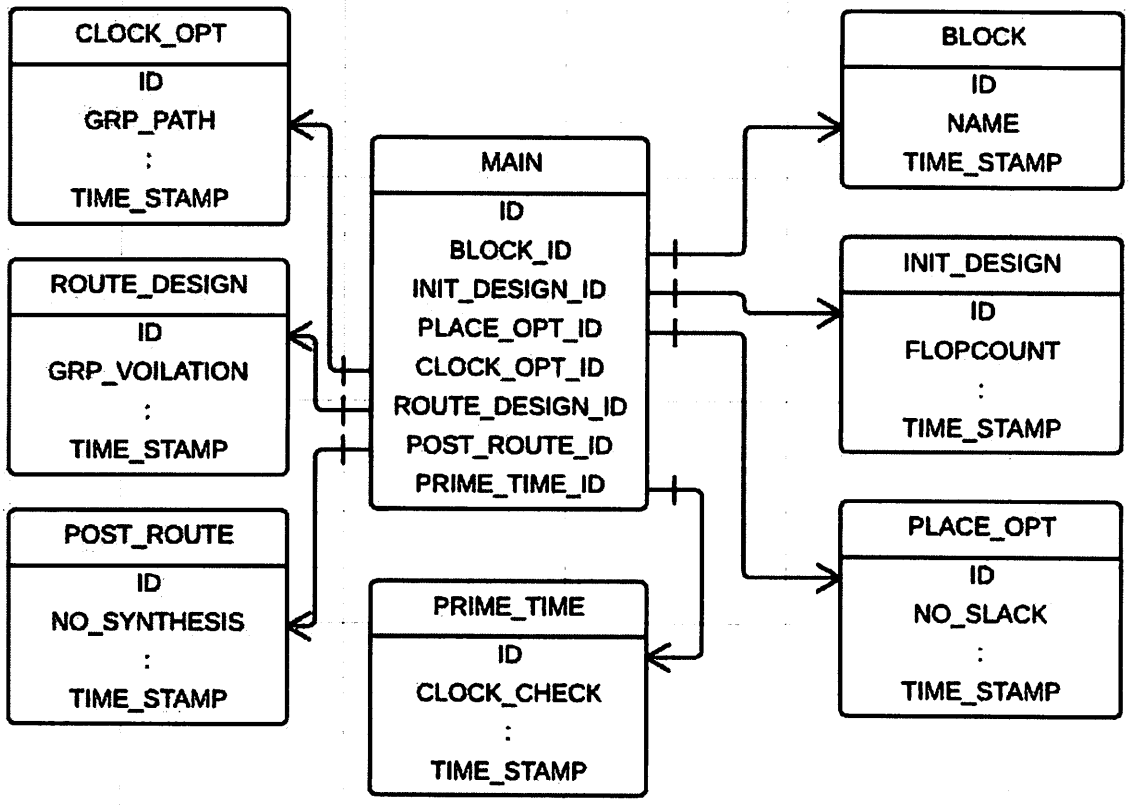

Figure 2.5: ERD diagram of main table

2.2.2.2 Conditional Data Mapping

Sometime the value of a target attribute depends on the value of another attributes. For example, if route designs data is null then fetch data from post route and if place-opt data does not exist then data get from signoff-opt + clock-opt data. Conditional data mapping constantly build data element mappings between two distinct data model.

2.2.2.3 Data Aggregation and user defined formulations mapping

The Data aggregation mapping allows you to perform aggregation such as sums and averages. PD transformation process performs the aggregation on corners and violation checks. Data is aggregated attribute-wise as well as clock-wise. There are other user defined formulas which are used in this project. Data is sorted by ascending floorplan blocks and descending date of insertion.

2.2.3 Data loading layer and Archive Database

Loading the transformed data to the archive database is the final process in ETL framework. In this process, extracted and transformed data is written into the archive database which is assessed by the end users history report system. During load process, it checks for duplicates by date, time and blocks and if records exist, it does nothing else inserts the record. Archive database is based on partitioning tables. With table partitioning, a table can be physically divided into multiple smaller tables, called partitions, while logically it stays as one table. It simply means that while the code stays the same, full partition scans will be executed instead of a full table scan [12].

Archive Database maintains historical data derived from operational PD database. Archive database has partitioning table capabilities to store data in different date chunks and Range Partitioning table is used in it. Achieve Database is placed on Cisco Systems Cloud.

2.2.4 Data validation layer

This layer of PD tool refers to data validity, data integrity and meaningfulness of data, meaning that the data remains consistent and accurate [13]. In the data archiving process, the garbage data problem was encountered i.e. if there is no data integrity in the data archiving process, any resulting report and analysis will not be useful.

PD tool data validation process retrieves and looks at data quality from operational database by generating summary reports. Users compare the operational data from the source file. Figure 2.6 shows comparison of PD data with source file data. Dark grey color shows data is good, light grey color shows data is minor changed and dark grey with white font color shows data is critically changed.

	A	B	C	D	E	F
1	**FP**	**clk_core Freq**	.	**SS Power Slow (mW)**	.	**PTPX Power Slow (mW)**
2	bsd_fp	759	.	202.92	.	316.69
3	**bsd_fp**	**759**	.	**202**	.	**304**
4	cbs0_fp	759	.	545.59	.	584.96
5	**cbs0_fp**	**759**	.	**563**	.	**389**
6	cbs10_fp	759	.	545.59	.	558.46
7	**cbs10_fp**	**759**	.	**572**	.	**337**
8	cbs11_fp	759	.	545.59	.	567.76
9	**cbs11_fp**	**759**	.	**572**	.	**337**

Figure 2.6: Comparison of data using coloring technique

In the beginning, there were some basic relational data problem appeared in PD database such as Referential integrity i.e. child records becomes garbage if parent node does not exist, primary key / unique constraint, identify NOT NULL and NULL-able attributes, valid values i.e. only allowed values are permitted in the database. For example, if an attribute can only have integer values, then alphanumeric values cannot be allowed or system rectify incoming values using set of rules.

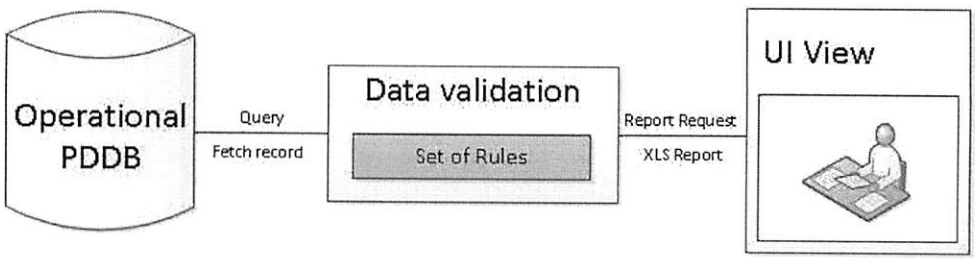

Figure 2.7 Data validation module layout

Figure 2.7 shows data validation module layout. User can generate report from operational database to track data quality record. The system generates report in excel format. Figure2.6 is good example of excel validation report.

2.2.5 User data Controlling UI layer

User data controlling layer provides form driven interface to user that control the "run status", flow of data and assigned target values. This process enables users to move status i.e. tapeout or historic and also enables correspondence with other engineers by posting comments.

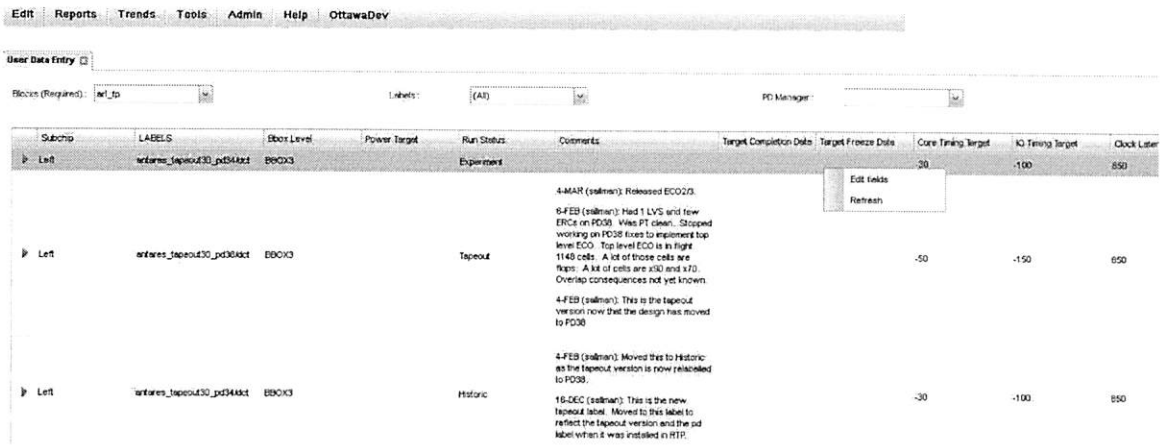

Figure 2.8 User data controlling GUI view

User data controlling layer have report utility feature as well as user can create custom report template and map the information and generate excel report. In fact, user can also enable history archive utility for tracking data.

2.2.6 Data Visualization UI layer

As described earlier, the focus of this tool is working on quality analysis of physical design chip. In order to find quality, we retrieve DRC, ERC, PT-Setup and PT-Hold violations records from archive database and filter information by last three weeks. However, these are aggregating values of 13 corners except design route check which aggregate of 14 checks. Records are sorted by datetime. If records are missing, for example in 1, 2, 4, 5… data record set 3 is missing, PD tool algorithm copy

n-1 value and represent by 3. The benefit of this mechanism is to get consistent data which is good for visualization. Then, convert data into pivot form and display into two representation techniques.

2.2.6.1 Tabular Representation

In tabular representation, we used set of rules to display values in different colors. For example black color font for normal data, **8** (bold grey) shows that some corners data is missing but **-777** (bold grey) shows all corners data is missing, latency max/ skew max (light bold grey) shows that data coming from signoff-opt instead of clock-opt.

	A	B	C	D	E	F	G	H
1	**Block**	**ICC DRC**	**ICV LVS**	**ICV DRCD**	**TOTAL ICV**	**TOTAL PT - ERC**	**Latency Max**	**Skew Max**
2	arl_fp	17	0	0	0	8	0.829	0.156
3	bde_fp	1	0	0	4	1	0.854	0.105
4	bdo_fp	1	0	0	0	2	0.858	0.078
5	bls_fp	1	0	0	0	1	0.84	0.111
6	bsd_fp	5	0	0	0	-777	0.803	0.097

Figure 2.9: Daily tapeout summary report

2.2.6.2 Graph Representation

In graph representation, we used line graph, divergence and convergence techniques based radar charts and other graphical techniques to show the PD information. E.g. in Figure 2.10 shows Divergence graph technique of post route data of slack attribute for clock core.

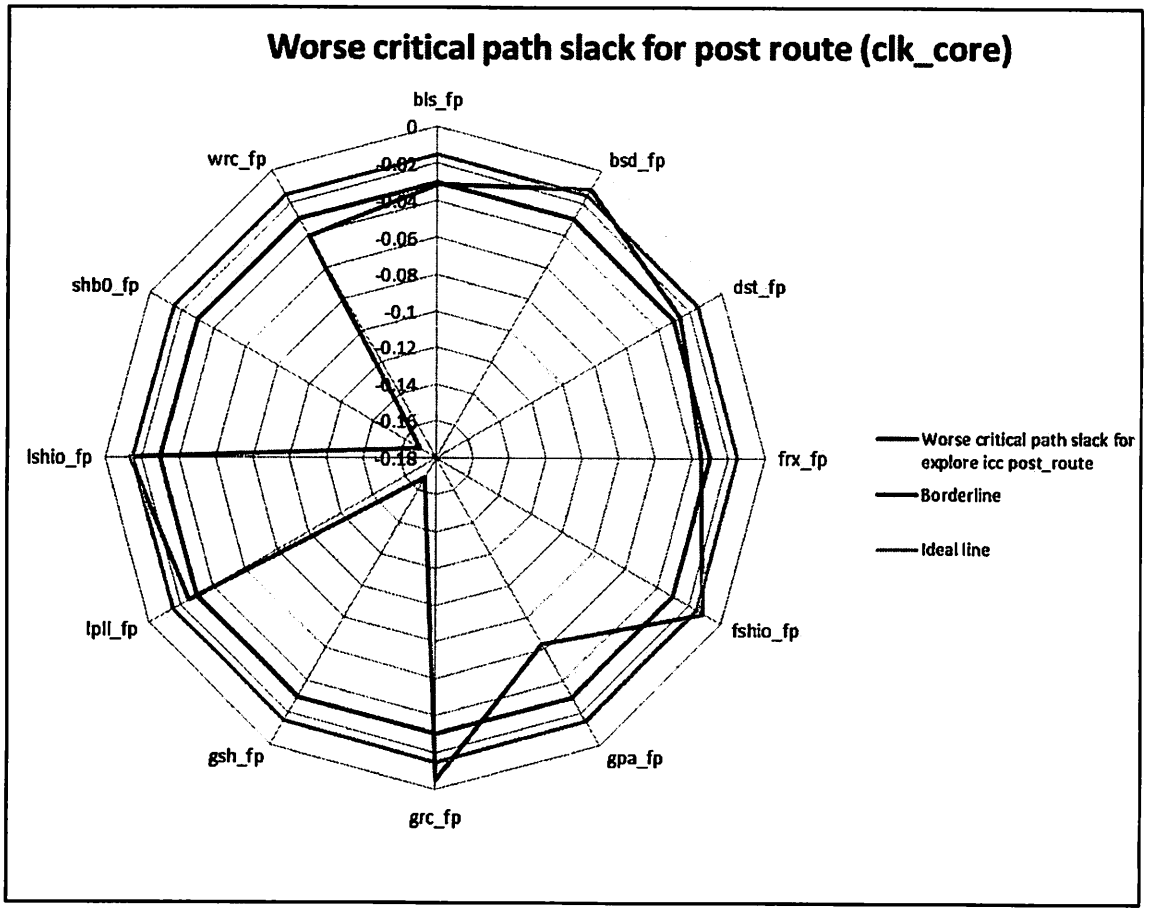

Figure 2.10: Divergence graph of worse critical path slack for post route (clock core)

2.2.6.3 Visualizing correlation

Visualizing correlation technique has great contribution in this project which is achieved with the help of statistical correlation method. Correlation method is used to show the dependency and relationship between two random variables or two sets of data which helps to make predictions [14]. For example, if two variables are correlated, we could predict one based on the other.

Figure 2.11 shows the positive correlated graph between Total Core hold and Total PT-ERC attributes. So when PD engineers want to predict which attribute has impact on floorplan blocks, they will identify the cause which destabilize by correlating the attributes.

General formula of correlation between x and y

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

The value of R is 0.9917. This is a strong positive correlation, which means that high Total Core Hold scores go with high PT-ERC scores (and vice versa).

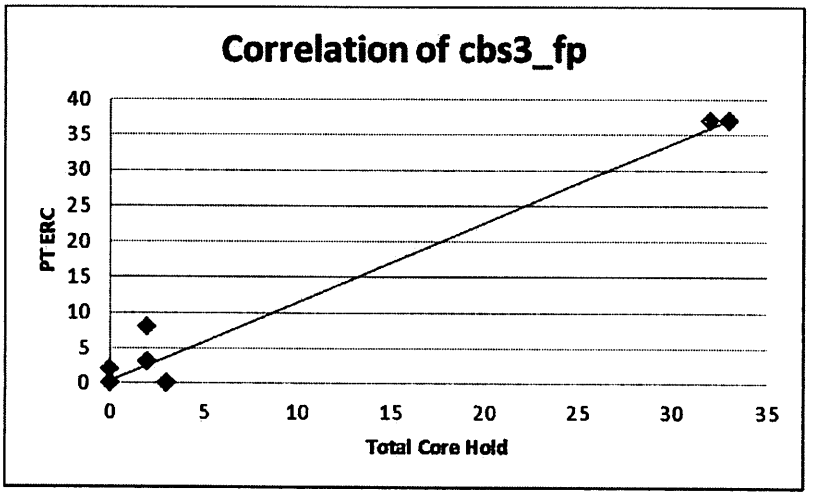

Figure 2.11: Positive correlation of cbs3_fp (block) between PT-ERC and Total Core Hold

Figure 2.12 shows the negative correlation, but weak because the value of R is -0.3374

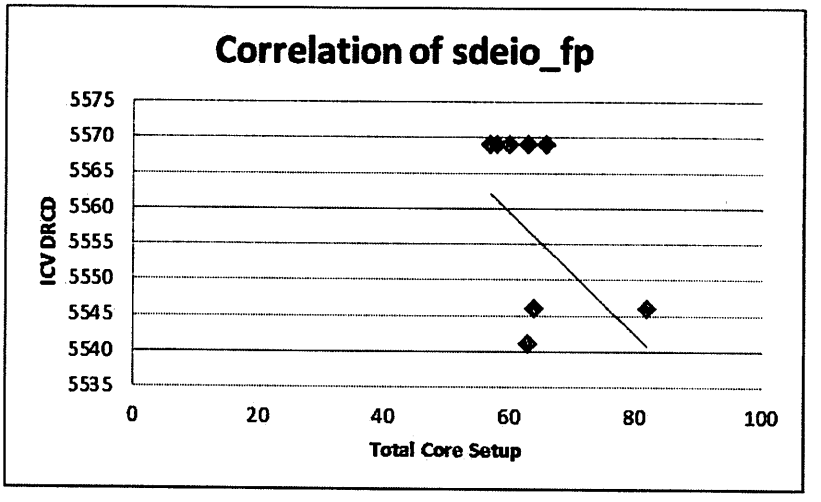

Figure 2.12: Negative correlation of sdeio_fp (block) between DRCD and Total Core Setup

Figure 2.13 shows the correlation of all subchip levels between electrical rule check and total core hold time. Positive dark grey shows positive strong correlation, negative dark grey shows negative strong correlation, positive/negative light grey shows no correlation, positive/negative grey shows weak positive or negative correlation.

bde_fp	bdo_fp	bls_fp	bsd_fp	dmy0_fp	dmy1_fp	dst_fp
0.514	0.380	0.166	0.846	0.746	0.274	-0.579
arl_fp	0.355				-0.233	cbs10_fp
deq1_fp	-0.907				0.238	cbs11_fp
deq2_fp	-0.157				0.445	cbs14_fp
deq3_fp	0.190				0.500	cbs15_fp
dftio_fp	0.905				-0.002	fobio_fp
etb0_fp	0.656	Correlation between PT-ERC			-0.506	fslio_fp
etb1_fp	0.074	and Total Core Hold			0.871	fus_fp
etb2_fp	0.703				0.478	ipe12_fp
etb3_fp	0.150				-0.032	ipe13_fp
etc_fp	0.471				0.862	ipe14_fp
gif_fp	0.365				0.224	ipe15_fp
itc_fp	0.760				0.691	pac_fp
ftx_fp	0.536				-0.812	sdeio_fp
0.101	0.434	0.841	0.915	0.623	0.991	0.215
cbs0_fp	cbs12_fp	cbs13_fp	cbs1_fp	cbs2_fp	cbs3_fp	cbs4_fp

Figure 2.13: Correlation of all subchip levels between PT-ERC and Total Core Hold

User generates visualizing correlation report every week to track the dependency between parameters and rapidly figure out the problem. This technique is completely helping out to resolve problems and optimize physical design process.

The next section relates to comparing performance of classification algorithms using physical design datasets. In future project, classification techniques could help to predict and estimate model at primary stage.

3 Implementation

In this section, we discuss and analyze three classification algorithms: Naïve Bayes, Decision Tree (J48 equivalents to C4.5) and Artificial Neural Network (multiplayer perceptrons). These algorithms are applied on different chip's physical design datasets to build the models and then calculate accuracy and precision of algorithms and determine the appropriate classification algorithm under the accuracy and precision. The above three classification algorithms have been selected within the scope of the project and would be described briefly in section 3.3.

The PD datasets chosen for the project have been collected from PD's data warehouse (archive database). The datasets are described briefly in the next section. The various datasets used are: Bottom (subchip) DRC (design rule check), Bottom (subchip) ERC (electrical rule check) data, Bottom (subchip) setup-time data, Bottom (subchip) hold-time data, Left DRC data, Left ERC data, Left setup data, Left hold data, Top DRC data, Top ERC data, Top setup data, Top hold data, Right DRC data, Right ERC data, Right setup data, Right hold data.

There are four primary stages of this project:

- **Datasets Collection:** Select attributes and data from PD archive database and divide records into bottom, left, top, right subchip levels

- **Preprocessing Datasets:** Eliminate missing values and categorize continuous data class attributes with the help of given (ASIC designer team) set of rules

- **Algorithms Selection:** Choose three classification algorithms and implementation of algorithm on multiple PD datasets

- **Analysis Outcomes:** Evaluate the models using accuracy and precision, and determine the appropriate classification algorithm

Below Figure 3.1 shows methodological approach for this project

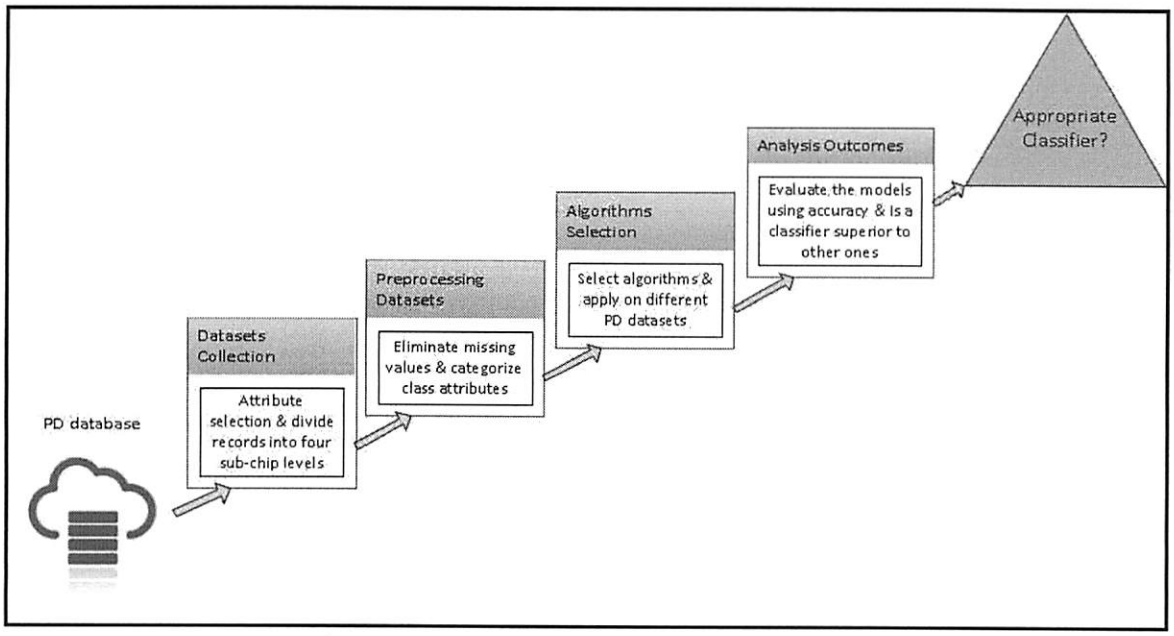

Figure 3.1: Methodological approach

3.1 Datasets overview

In the previous (proposed PD tool) section, we describe that the PD tool helps out to gather data from different data sources and it stores and archives meaningful selected information in PD archive database from past two months. However, we retrieve all datasets from archive database which are related to design quality and performance of chip. Most of information storing in PD database belongs to design rule check (DRC), electrical rule check (ERC), primetime-setup and primetime-hold violations.

Datasets contains overall 18 (nominal and continuous) decision attributes and 4 predict attributes (set of classes) and has more than 3,000 records (without missing and duplicate values) for all subchip levels. All records in database relate to tape-out release (which means final outcomes of physical design blocks). And, split records into four different subchip levels such as bottom subchip, left subchip, top subchip and right subchip. And, each predicted attributes have six sub classes such as clean, below-low, low, below-high, high and very-high.

3.1.1 Bottom subchip datasets

Bottom subchip data are crucial and highly focused by PD Engineers because performance and quality are largely concern with bottom subchip. However, bottom subchip dataset has 1315 instances; and we spilt 18 decision attributes into four dataset's types which are DRC, ERC, setup-time and hold-time and assigned six sub classes to each of these given classes to each of dataset's type.

3.1.2 Other subchips datasets

Similarly, left subchip dataset has 1125 instances; as well as top subchip dataset and right subchip dataset are more than 500 instances each; and we also spilt these dataset further into four dataset's types. Rest of datasets (i.e. top subchip dataset and right subchip dataset) applies similar approach as bottom and left subchip.

3.2 Preprocessing the datasets

The PD datasets have accumulatively more than 3000 records without missing and duplicate values because the data has already passed the cleansing and transformation processes. Although in data visualization process, we have been using duplicate *n-1* row if *n* row does not exist (*n* row determine by data time) but this time we are least concern about duplicate records.

Since design rule check, electrical rule check, primetime setup and primetime hold violations are in continuous numbers. Simply, we categorize the data points into finite number of violation-levels (called subclasses) with the help of given ASIC designer's set of rules. According to set of rules, DRC, ERC, setup time and hold time data ranges are divided into six violation-levels such as clean, below-low, low, below-high, high and very-high. Eventually, we replace selected attributes (i.e. DRC, ERC, and PT-setup and PT-hold) data into these subclasses (violation-levels) by using assign-levels algorithm. We execute algorithm and pass the selected attributes data-column as parameter. This is done independently for each of the selected attributes columns.

Rules	Conditions	Classes (Levels)
1	Equal to 0	CLEAN
2	1 to 5	BELOW-LOW
3	6 to 20	LOW
4	21 to 100	BELOW-HIGH
5	101 to 999	HIGH
6	More then 999	VERY-HIGH

Table 3.1: ASIC designer's set of rules

Algorithm II Assign levels to given dataset columns

```
1:    procedure assign_levels_dataset (data-column)   //data-column is class attributes of datasets
2:     rules = load (set_of_rules)    //fetch rules from table
3:     for each data-column do    //read PD datasets column values by line!
4:          for each rules do    //read all rules by line!
5:               If (data-column-has(rules["condition"]))  //if data-column satisfies condition
6:                    then
7:                    data-column [cur] =rules ["levels"];  //then assign level/class to data
8:               end if
9:          end for
10:    end for
11:  return data-column;
12: end procedure
```

Since the training set and test set are not provided separately, we need to split the dataset into a training set and a test set. We randomly divide dataset into two such that 66% records are in the training set and 34% records are in test set. In 3.4 sections, we simply implement three classification algorithms on the multiple PD datasets. For more information about dataset's attributes see appendix-B

3.3 Algorithms overview:

In this section, we provide the description of Naïve Bayes, Decision Tree (J48) and Artificial Neural Network (multilayer perceptrons) algorithms.

3.3.1 Naïve Bayes.

The Naïve Bayes classification algorithm works on a simple but comparatively intuitive concept which is constructed on Bayesian statistics dealing with a simple probabilistic classification based on applying Bayes' theorem with strong (naive) independence assumptions which means that all features of dataset are independent of each other given the context of the class. The assumption of the conditional probabilities may be expressed as [15].

$$p(X_1 = x_1, X_2 = x_2, \ldots, X_m = x_m|\theta) = \prod_{i=1}^{m} p(X_i = x_i|\theta)$$

So, the Naïve Bayes classification is therefore given in [15].

$$\theta_{\mathrm{NB}} = \arg\max_{\theta} \prod_{i=1}^{m} p(X_i = x_i|\theta)p(\theta)$$

The Naïve Bayes classifier will consider each of these attributes individually when classifying a new instance. So, when checking to see if the new instance is an elephant, the Naïve Bayes classifier will not check whether it has a trunk and has huge tusks and is large. Rather, it will separately check whether the new instance has a trunk, whether it has tusks, whether it is large, etc. [16]. It works under the assumption that one attribute works independently of the other attributes contained by the sample.

3.3.2 (J48) Decision Tree

An efficient and quickest method to build classification model from dataset is to generate a decision tree. A decision tree is a predictive machine-learning model that decides the target value (dependent variable) of a new sample based on various attribute values of the available data. The decision-tree representation is the most widely used logic method [17]. There is a large number of decision-tree induction algorithms described primarily in the machine-learning and applied-statistics literature.

The J48 (which equivalent to C4.5) decision tree classifier follows the following simple algorithm procedures. For classifying a new item, it first has to create a decision tree space based on the feature values of the available training data. It simply rank attributes and to build decision trees where at each node is located the attribute with highest information gain among the attributes not yet considered in the path from the root. In general terms, the expected information gain is the change in information entropy H from a prior state to a state that takes some information as given [18]:

$$IG(Y,X) = H(Y) - H(Y|X)$$

IG (information gain), X denote a set of training examples, the information gain for an attribute Y and is equal to conditional entropy $H(Y|X)$ for an attribute subtracted from the total entropy $H(Y)$.

3.3.3 Multi-layer Perceptrons

A multilayer perceptrons in Weka is a feedforward artificial neural network model. The feedforward neural network begins with an input layer. The input layer may be connected to a hidden layer or directly to the output layer. If it is connected to a hidden layer, the hidden layer can then be connected to another hidden layer or directly to the output layer. There can be any

number of hidden layers, as long as there is at least one hidden layer or output layer provided. In common use, most neural networks will have one hidden layer, and it is very rare for a neural network to have more than two hidden layers [19].

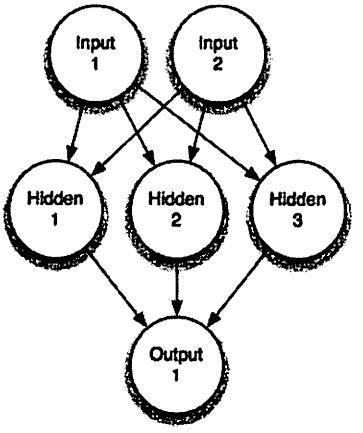

Figure 3.2: A typical feedforward neural network (single hidden layer)

Learning occurs in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result [20]. This is an example of supervised learning, and is carried out through back propagation, a generalization of the least mean squares algorithm in the linear perceptron.

$$\mathcal{E}(n) = \frac{1}{2}\sum_{j} e_j^2(n)$$

We represent the error in output node j in the nth data point by $e_j^2(n) = (d_j(n)-y_j(n))^2$, where d_j is the target value, y_j is the value produced by the perceptron and n is the Epoch learning rate. We then make corrections to the weights of the nodes based on those corrections which minimize the error in the entire output, given by [21]

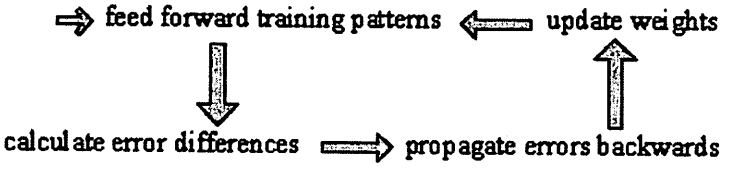

Figure 3.3: The back propagation neural network Epoch learning [22]

3.4 Weka classifiers analysis

Weka is one of valuable open source tool for data mining purpose. The objective is to import all subchip levels datasets in Weka tool and apply Naïve Bayes, decision tree and multilayer perceptrons classifiers, and determine accuracy and precision.

We spilt PD dataset's attributes into four categories (classes) such as design rule check (DRC), electrical rule check (ERC), Primetime setup and Primetime hold. For example, wc_slack_route_design, wc_no_route_design, wc_total_route_design, wc_slack_post_route, wc_no_post_route, and wc_total_post_route are associated to DRC. In the same way, ptwcgrpslowhtrpisfstp, ptwcslowhtrpisfstp, ptwcsumslowhtrpisfstp, ptwcsumfsthtpisf1p13hld, ptercslowhtrpisfstp, ptercfsthtpisf1p13hld are related to ERC. In fact, setup-time and hold-time share some basic attributes from ERC which have closing pattern stp (called setup) and hld (called hold), block_name and time_stamp are common attributes between all of them. Descriptions of these attributes are given in appendix B.

3.4.1 Naïve Bayes implementation on Weka

After preprocessing, we implement Naïve Bayes classifier which could build the model based on the probability of each class given the DRC, ERC, setup-time and hold-time dataset's feature variables. The assumption is that the features are independent of each other.

Below example has been done for DRC of bottom subchip dataset using naïve Bayes classifier and the accuracy and precision are noted down. As we know, bottom subchip dataset has 1315 records without missing and duplicate values. Weka tool spilt 1315 instances into two portion (66% training records and 34% test records) i.e. 868 instances which use to train the model and 447 instances which use to test the generated model.

Evaluation on test split summary

Correctly Classified Instances	327
Incorrectly Classified Instances	120
Kappa statistic	0.6468
Mean absolute error	0.0935
Root mean squared error	0.2647
Relative absolute error	39.7218 %
Root relative squared error	77.0684 %
Total Number of Instances	447

Naïve Bayes classifier is applied on DRC dataset which gives 73.1544% accuracy on 327 out of 447 test instances which mean 120 instances are classified incorrectly.

Confusion Matrix

a	b	c	d	e	f	<-- classified as
114	7	1	5	0	7	a = BELOW-LOW
0	7	0	5	0	0	b = LOW
0	0	52	0	0	0	c = HIGH
0	0	4	35	0	0	d = BELOW-HIGH
0	0	0	0	20	0	e = VERY-HIGH
44	4	5	14	24	99	f = CLEAN

Weka tool builds confusion matrix which is used to determine precision of the classifier. In the next section, the procedure of calculating precision by individual class and overall classes is explained.

Similar approach is applied to bottom subchip of ERC, setup-time and hold-time datasets. In fact, the rest of subchip levels such as left subchip, right subchip and top subchip are tested and the accuracy and precision are calculated by Weka tool. Descriptions of their outcomes have given in appendix B.

3.4.2 J48 (Decision Tree) implementation on Weka

Decision Tree classification algorithm plot hierarchical structure by recursively split the feature attributes based on the entropy that drive to information gain criteria for choosing the best case, and decision tree helps in decision making.

The results gained show Weka tool analysis on DRC of bottom subchip dataset using J48 classification algorithm. Weka tool spilt instances on 66% training records and 34% test records.

Evaluation on test split Summary

Correctly Classified Instances	371
Incorrectly Classified Instances	76
Kappa statistic	0.7616
Mean absolute error	0.083
Root mean squared error	0.2045
Relative absolute error	35.2632 %
Root relative squared error	59.5363 %
Total Number of Instances	447

Decision tree (J48) classifier apply on DRC dataset which gives the 82.9978% accuracy on 371 out of 447 test instances which mean 76 instances are classified incorrectly.

Confusion Matrix

a	b	c	d	e	f	<-- classified as
117	0	0	1	0	16	a = BELOW-LOW
0	6	0	1	0	5	b = LOW
0	0	52	0	0	0	c = HIGH
0	0	4	34	0	1	d = BELOW-HIGH
0	0	0	0	20	0	e = VERY-HIGH
41	0	0	3	4	142	f = CLEAN

Weka tool builds confusion matrix which determines the precision of decision tree. In the next section, the procedure of calculate precision by individual class and overall classes is explained.

Similar approach has been applied to bottom subchip of ERC, setup-time and hold-time datasets. In fact, the rest of subchip levels such as left subchip, right subchip and top subchip have done and written down accuracy and precision by Weka tool. Descriptions of their outcomes are given in appendix B.

3.4.3 Neural Network

Neural Network consists of neurons. Neurons are like a circuit-gate which takes multiple inputs and generates one output. We use multilayer perceptrons which has implementation of back propagation algorithm. Neural Network has main three layers such as input layer, hidden layer and output layer.

The results gained show Weka tool analysis on DRC of bottom subchip dataset using multilayer perceptrons classifier. As similarly, Weka tool spilt instances on 66% training records and 34% test records.

Evaluation on test split summary

Correctly Classified Instances	371
Incorrectly Classified Instances	76
Kappa statistic	0.7616
Mean absolute error	0.083
Root mean squared error	0.2045
Relative absolute error	35.2632 %
Root relative squared error	59.5363 %
Total Number of Instances	447

Multilayer perceptrons classifier apply on DRC dataset which gives the 92.8412% accuracy on 415 out of 447 test instances which mean 32 instances are classified incorrectly.

Confusion Matrix

a	b	c	d	e	f	<-- classified as
117	0	0	1	0	16	a = BELOW-LOW
0	6	0	1	0	5	b = LOW
0	0	52	0	0	0	c = HIGH
0	0	4	34	0	1	d = BELOW-HIGH
0	0	0	0	20	0	e = VERY-HIGH
41	0	0	3	4	142	f = CLEAN

Weka tool builds confusion matrix which is used to determine precision of multilayer perceptrons. In the next section, the procedure of calculating precision by individual class and overall classes is explained.

Similar approach is applied on bottom subchip of ERC, setup-time and hold-time datasets. In fact, the rest of subchip levels such as left subchip, right subchip and top subchip have done and written down accuracy and precision by Weka tool. Descriptions of their outcomes are given in appendix B.

4 Results Analysis and Discussion

In this section the performance of each classification algorithm on each dataset will be discussed and analyzed. The performances of the three classification algorithms were estimated on the PD datasets at four subchip levels.

4.1 Classification Accuracy of PD datasets

This section describes and compares the classification accuracy of three basic classification algorithms. We compare accuracy of Naïve Bayes, Decision Tree (J48) and Neural Network (Multilayer Perceptrons) with each other at subchip levels.

4.1.1 Accuracy of bottom subchip level

The three classification algorithms are implemented on bottom subchip dataset which give us these results; for DRC dataset 73.15%, 82.99% and 92.84% accuracy outcomes of Naïve Bayes, J48 and Multilayer Perceptrons respectively. For hold-time dataset, we obtain 89.70%, 98.65% and 92.84% and for setup-time dataset 96.86, 97.09% and 97.53%. And the accuracy of ERC dataset is 80.53%, 92.84% and 90.38% of Naïve Bayes, J48 and Multilayer perceptrons classifier respectively.

	Naïve Bayes	J48	Multilayer Perceptrons
DRC	73.15	82.99	92.84
HOLD	89.70	98.65	92.84
SETUP	96.86	97.09	97.53
ERC	80.53	92.84	90.38

Table 4.1: Accuracy of bottom subchip dataset in %age

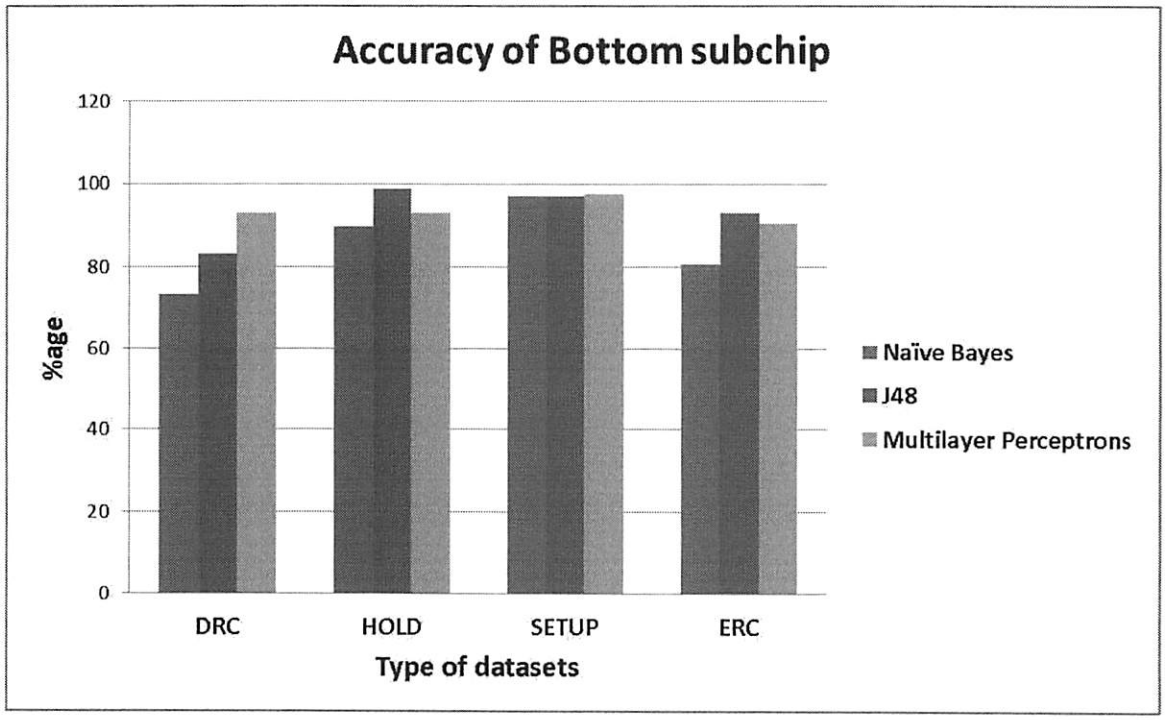

Figure 4.1: Comparison of bottom subchip dataset accuracy

Figure 4.1 provides us information about the quality of classification model in terms of comparison of classification accuracy for bottom subchip dataset. We examine that neural network (multilayer perceptrons) classification method give relatively high accuracy model of this datasets. In DRC dataset, multilayer perceptrons gives 92.84% where Naïve Bayes and J48 find 73.15% and 82.99% respectively.

Based on bottom subchip datasets, we can't assume that multilayer perceptrons is appropriate for physical design datasets. Let's obtain more accuracy outcomes from rest of subchip datasets.

4.1.2 Accuracy of left subchip

Left subchip datasets have slightly lower accuracy than bottom subchip dataset because violations are in high numbers and large number of fluctuations. However, this dataset also categorize level wise using the same set of rules which are used in bottom subchip dataset and simply implement three classification algorithms which give following results; DRC's dataset give us 70.73%, 80.89% and 95.52% correctness of Naïve Bayes, J48 and Multilayer Perceptrons respectively. Similarly, for hold-time we obtain 66.66%, 98.37% and 97.15% and for setup-time

86.17, 96.34% and 99.18%. And we achieve accuracy from ERC dataset are 95.12%, 95.93% and 93.90% of Naïve Bayes, J48 and multilayer perceptrons classifier respectively.

	Naïve Bayes	J48	Multilayer Perceptrons
DRC	70.73	80.89	95.52
HOLD	66.66	98.37	97.15
SETUP	86.17	96.34	99.18
ERC	95.12	95.93	93.90

Table 4.2: Accuracy of left subchip dataset in %age

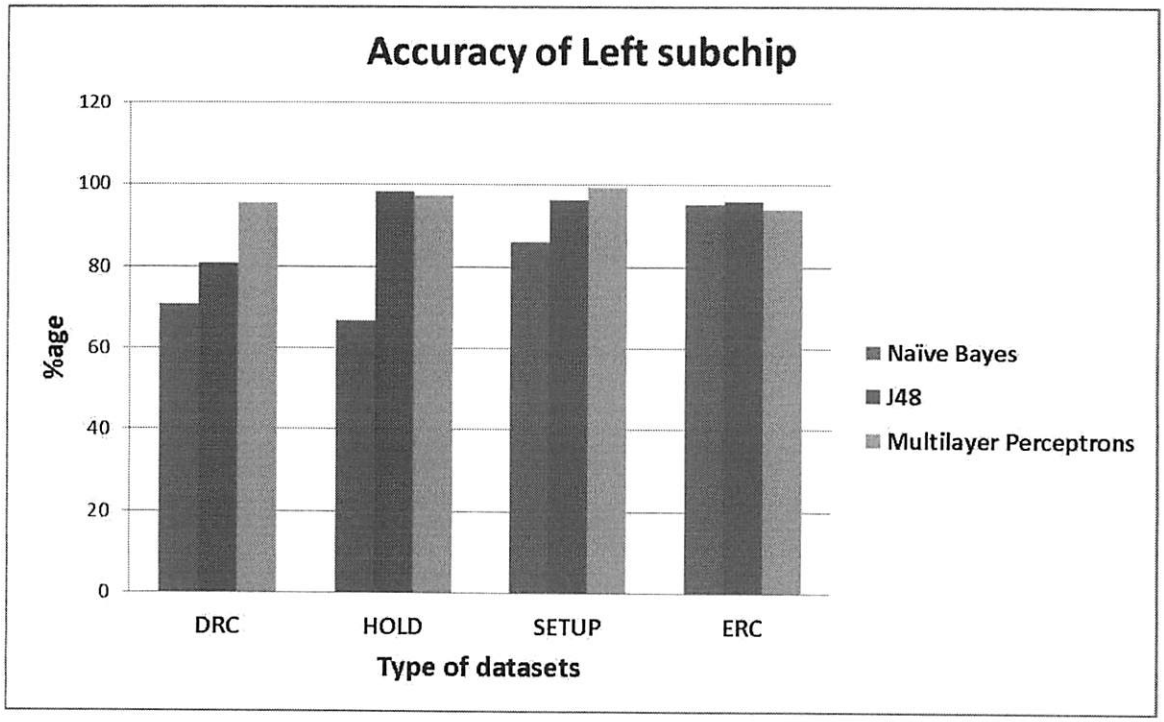

Figure 4.2: Comparison of left subchip dataset accuracy

Figure 4.2 that shows left subchip dataset classification algorithms comparison; neural network (multilayer perceptrons) has very-high accuracy as compared to other classification algorithms. Exclusively, for DRC and hold datasets, Naïve Bayes classifier has less classification accuracy where multilayer perceptrons is more than 95% accurate. We also examine that J48 (Decision Tree) classifier accuracy is equal to multilayer perceptrons accuracy. ERC datasets have comparatively very high accuracy in all classification algorithms.

4.1.3 Accuracy of top subchip

Top subchip datasets have lower accuracy outcomes as compared to bottom and left subchip datasets. Especially, Naïve Bayes has 45% classification accuracy on primetime hold dataset. In fact, ERC dataset also has 64% accuracy which's lower than bottom and left subchip levels. Primetime-setup has pretty higher correctness rate for all three classification algorithms. Multilayer perceptrons and Decision tree have excellent accuracy outcomes in all types of dataset at this subchip level and, top (subchip) DRC datasets has exceptionally higher outcomes as compared to previous subchip levels.

	Naïve Bayes	J48	Multilayer Perceptrons
DRC	92.22	93.26	93.26
HOLD	45.59	92.22	83.41
SETUP	86.01	95.33	90.15
ERC	64.24	89.11	76.68

Table 4.3: Accuracy of top subchip dataset in %age

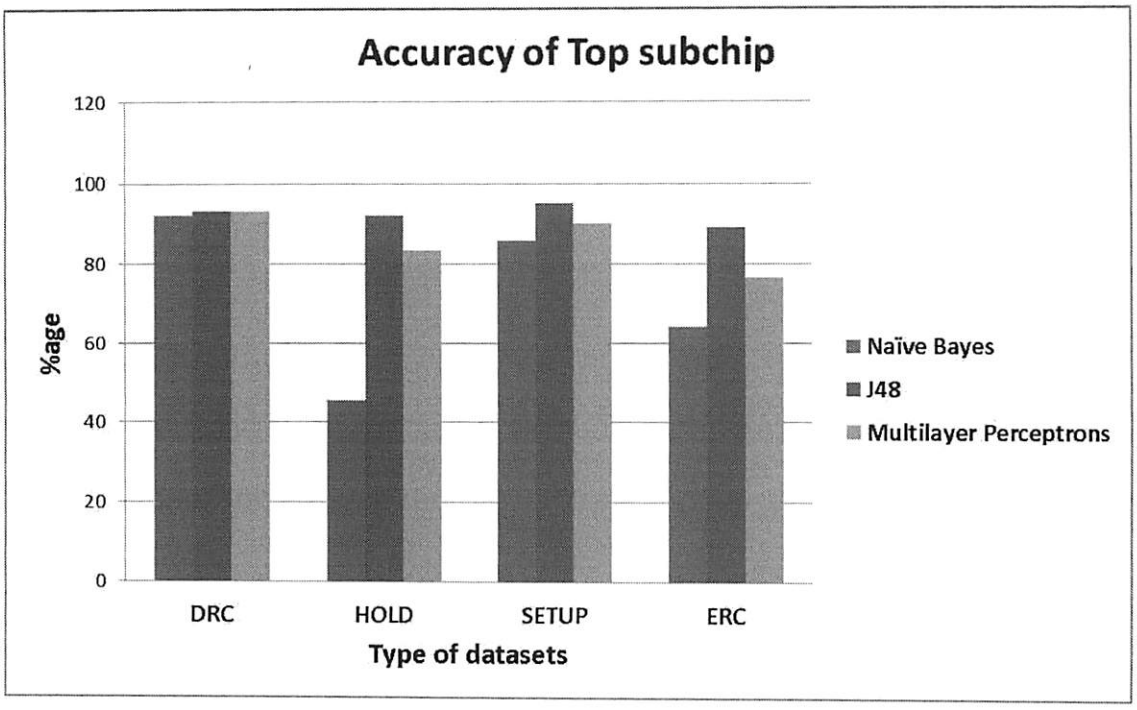

Figure 4.3: Comparison of top subchip dataset accuracy

Figure 4.3 shows that top subchip dataset classifier accuracy comparison; Decision Tree (J48) has very-high accuracy as compared to Naïve Bayes and Multilayer Perceptrons classifier. Design-rule-check dataset has comparatively high accuracy of all classification algorithms than other datasets. J48 classifier has averagely more than 92% correctness overall. We also examine that J48 (Decision Tree) classifier accuracy is slightly higher than multilayer perceptrons accuracy because multilayer perceptrons has 76.68% accuracy at ERC dataset which is lower accuracy as compared to other subchip levels.

4.1.4 Accuracy of right subchip

Right subchip data extremely fluctuate as compared to bottom and left subchip datasets because some violations are in very high numbers. However, we have used same set of rules for labeling the class attribute such as clean, below-low, low, below-high, high and very-high level. In the implementation section, we simply apply three classification algorithms which give us following results;

In DRC dataset, Naïve Bayes, J48 and Multilayer Perceptron have 62.92%, 85.39% and 89.88% accuracy respectively. Similarly, for hold-time dataset we obtained 64.60%, 91.01% and 85.39% of accuracy and for setup-time dataset 54.49, 89.88% and 93.25% of accuracy. And, we find accuracy from electrical-rule-check dataset which is 84.83%, 96.06% and 86.51% of classification accuracy of Naïve Bayes, J48 and Multilayer perceptrons classifier respectively.

	Naïve Bayes	J48	Multilayer Perceptrons
DRC	62.92	85.39	89.88
HOLD	64.60	91.01	85.39
SETUP	54.49	89.88	93.25
ERC	84.83	96.06	86.51

4.4: Accuracy of right subchip dataset in %age

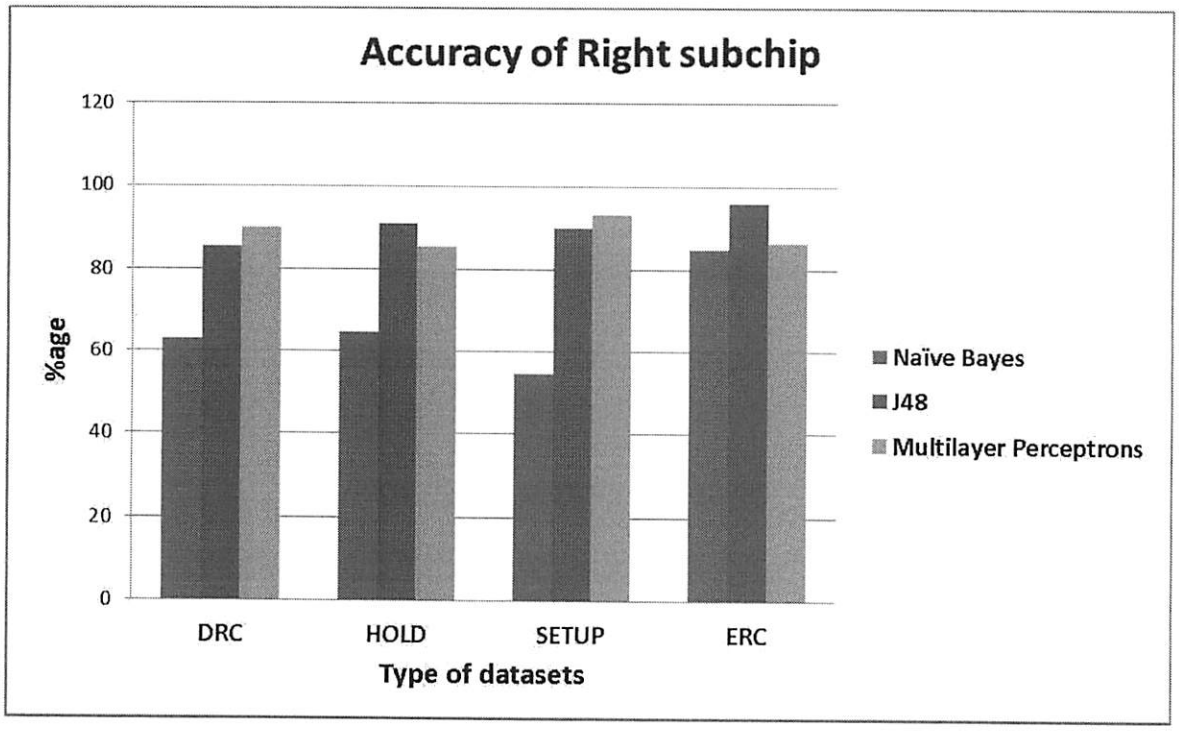

Figure 4.4: Comparison of right subchip dataset accuracy

Right subchip dataset has enormous lack of classification accuracy on Naïve Bayes classifier which is generally below than 70%.

Figure 4.4 clearly shows that right subchip dataset classifiers comparison; Decision Tree (J48) classifier has very-high accuracy as compared to other classifiers. Exclusively, in ERC dataset J48 has 96.06% of classification accuracy which is comparatively higher than naïve Bayes and multilayer perceptrons algorithms. We also examine that multilayer perceptrons classifier accuracy is approximately equal to Decision Tree accuracy.

4.2 Precision of all subchip levels

Table 4.5 shows the confusion matrix for the naïve Bayes classifier trained on the bottom DRC dataset. This table is now used to demonstrate evaluation techniques for precision and recall. The table cells represent the number of counts in the test dataset. The columns represent the predicted class and the rows represent the actual class in the dataset. The *ATotal* (actual total) column indicates the number of test samples whose actual label is specified by the row. The *PTotal* (predict total) row indicates the number of predicted records in each column [25].

For example, subclass BELOW-LOW from the table:

134 = samples were actual total (*ATotal*) in this class

114 = is the number of true positives (TP).

158 = represent predict total (*PTotal*)

44 = (mean 158-114) represent false positives (FP)

The precision for each class are calculated. The overall precision is computed as a weighted average.

$\text{Precision}_{\text{Below-Low}} = 114/158 \Rightarrow 0.722$ $\text{Precision}_{\text{Low}} = 7/18 \Rightarrow 0.389$ and so on.

Evaluate precision individually by classes and determine the weighted average by taking sum of all individually precision and divide by number of classes:

$\text{Precision}_{\text{Weighted Avg}} = (0.722+0.389+0.839+0.593+0.455+0.934)/6 \Rightarrow 0.793$

CLASS	BELOW-LOW	LOW	HIGH	BELOW-HIGH	VERY-HIGH	CLEAN	*ATotal*
BELOW-LOW	114	7	1	5	0	7	*134*
LOW	0	7	0	5	0	0	*12*
HIGH	0	0	52	0	0	0	*52*
BELOW-HIGH	0	0	4	35	0	0	*39*
VERY-HIGH	0	0	0	0	20	0	*20*
CLEAN	44	4	5	14	24	99	*190*
PTotal	*158*	*18*	*62*	*59*	*44*	*106*	**447**

Table 4.5: Confusion matrix for Naïve Bayes classifier trained on the bottom DRC dataset

Weka tool are estimated weighted average precision of each datasets and precision of all classifier are given in Table 4.6.

Datasets	Precision of three classifiers		
	Naïve Bayes	J48 (Decision Tree)	Multilayer Perceptrons
Bottom DRC	79.3	83.8	93.3
Bottom ERC	80.7	93.6	91.2
Bottom Hold	90.7	98.8	93.2
Bottom Setup	97.2	97.5	98.7
Left DRC	80.2	84.9	84.9
Left ERC	95.9	96.8	94.3
Left Hold	91	96.4	99.2
Left Setup	83.9	98	96.8
Right DRC	76	86.8	90.4
Right ERC	84.2	96.6	87.5
Right Hold	77.5	91.3	86.6
Right Setup	90	91	93
Top DRC	93.3	93	93
Top ERC	58.3	88.6	77.3
Top Hold	51.3	92.4	81
Top Setup	86.3	95.8	89.9

Table 4.6: Precision outcomes of all PD subchip dataset in %age

Precision is commonly used to evaluate measures of classifier performance for given datasets. As far as we know, accuracy only tells us the correctness of classifiers prediction. However, accuracy does not tell how close the measured values are to each other. In order to determine the classifier consistency we need to determine the precision which gives confidence level to classification algorithms.

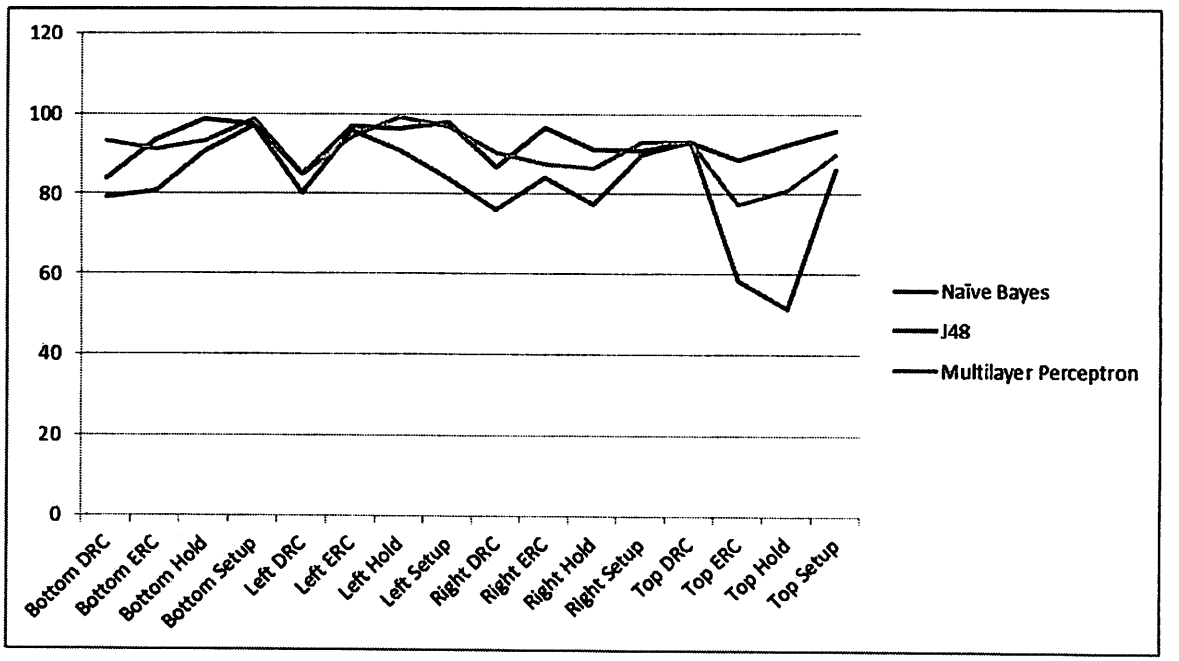

Figure 4.5: Precision graph for all classifiers at all subchip levels

Figure 4.5 shows that all subchip levels datasets precision comparison; Decision Tree (J48) classifier has very-high precision as compared to other classifiers. Decision tree and multilayer perceptrons are overlap on left subchip datasets.

4.3 Experiment discussion

Figure 4.6 shows the estimated classification accuracy (on y-axis) of the three classification algorithms (represented by colors) on each physical design's dataset (on x-axis). As we can see from Figure 4.6, the classification accuracy of the decision tree (J48) algorithm is better in fourteen out of the sixteen datasets. The graph also shows that Naïve Bayes accuracy relatively

low classification accuracy in all the datasets. However, Naïve Bayes classifier is the best classification results on the right-subchip setup and top-subchip setup datasets. For the all categories of top-subchip datasets, multilayer perceptrons (neural network) come out with high accuracy. Based on the outcomes, one can argue that the decision tree (J48) and multilayer perceptrons classifier are the equally best classifiers because of theirs classification accuracy and precision above 85% in fourteen out of the sixteen datasets; as shown in graph and Table 4.6, Naïve Bayes accuracy is better in three out of the sixteen datasets (bottom setup, left ERC and top DRC).

All three classification algorithms show above 85% accuracy outcomes on six out of sixteen datasets but show slightly fluctuate on rest of the datasets. For example Naïve Bayes classifier shows excellent (> 85%) accuracy results on the bottom hold, bottom setup, left setup, left ERC, top DRC and top setup datasets and comparatively good (> 75% and < 85%) accuracy on the bottom DRC, bottom ERC, left DRC, right ERC datasets but they show relatively poor (< 70) results on the left hold, top hold, top ERC, right DRC, right hold and right setup datasets (Figure 4.5).

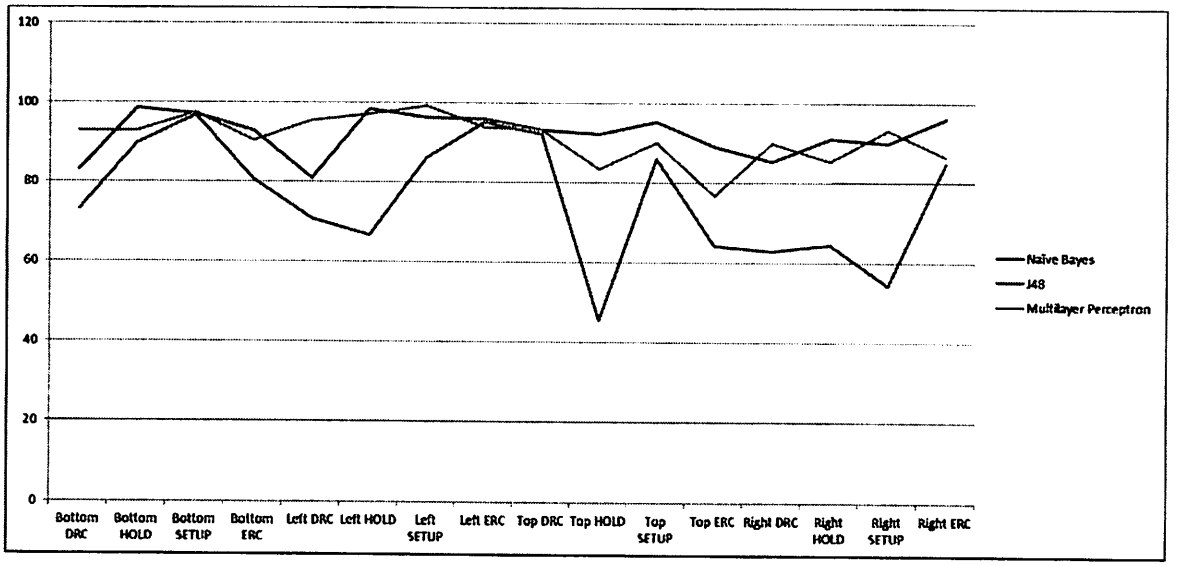

Figure 4.6: Overall comparison of classification algorithm (X-axis shows accuracy in %age, Y-axis shows name of datasets and green, red, blue colors show line of naïve Bayes, decision tree and multilayer perceptrons classifier respectively)

According to graph above graph, naïve Bayes classifier produces clearly lower correctness as compared to decision tree and multilayer perceptrons classifiers except some datasets. On the other hand, multilayer perceptrons is apparently high in accuracy similar to decision tree (J48) classifier which is more than 91%.

In Figure 4.7 linear trend lines in between these algorithms accuracy lines show that naïve Bayes classifier accuracy rapidly declines. Similarly, multilayer perceptrons accuracy slightly declines. Whereas, J48 (decision tree) classifier has straight line towards all datasets.

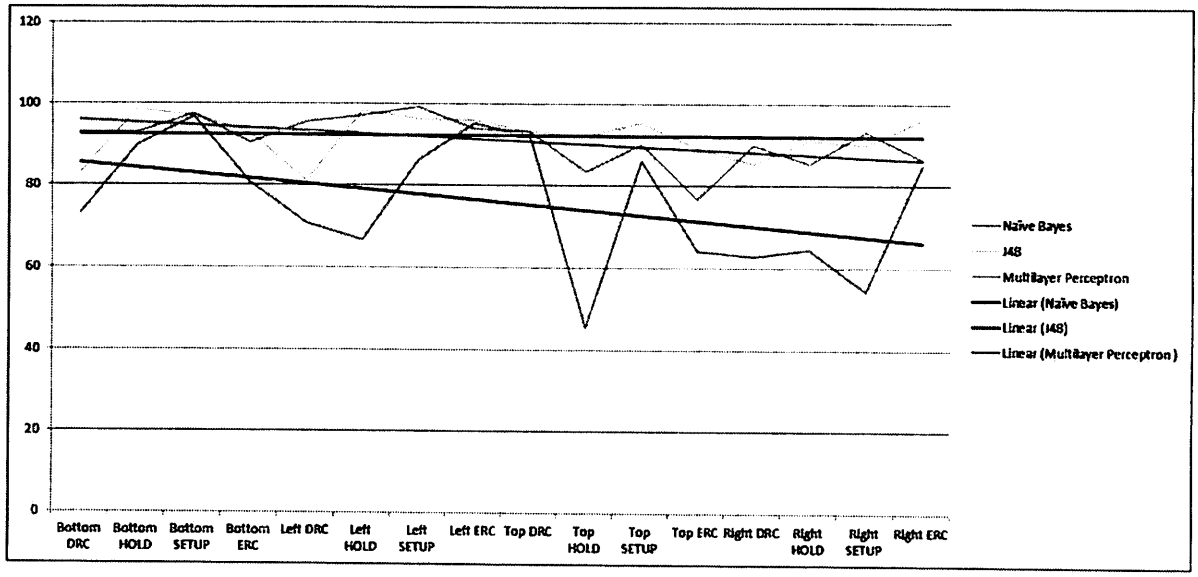

Figure 4.7: Linear trend line comparison of classification algorithm

From the above Figure, it's clear that J48 has comparatively superior classifier for physical design datasets. In fact, multilayer perceptrons has been proved more efficient for bottom subchip datasets and it also takes more time than decision tree algorithm. In conclusion, J48 (decision tree) algorithm is more appropriate to classify physical design datasets.

5 Conclusion

The first part of report discussed PD (Physical Design) tool development process which is broadly used by a PD engineer for quality tracking, design estimation, time forecasting and model optimization. The tool is based on 3 main elements namely, Extract, Transform & Load (ETL) methodology. It gathers meaningful data from unstructured files; cleans inappropriate and inconsistent data and finally loads into operational database. It has a daily history archive mechanism which takes snapshot of operational database on daily basis.

Frontend of this tool provides user friendly interface to user to generate data reports called data visualization. Data visualization techniques show information using graph and tabular representation. It helps to show interrelation and correlation between blocks and theirs parameters. The second part of report focused on three classification methods which are used in PD datasets and furthermore discussed theirs results. Weka tool analysis gives us clear picture that J48 (decision tree) classifier provided better model of classification on PD datasets. J48 classifier classifies all datasets more than 91% correctness and more than 92% consistent. This time, it ensures that PD datasets completely have rule-based records. This is the particular reason Naïve Bayes did not perform well throughout the datasets except the bottom subchip datasets.

Another reason we observed that physical design process was on peak stage of work, data was critical which caused large number of fluctuations. Although, we have used one corner attribute to predict total violations of DRC, ERC, setup-time and hold-time, these total violations are aggregate of thirteen corners. If we opt to use more than one corner, we might get optimum performance from the classifiers.

6 References

[1] King, R. D., Feng, C., & Sutherland, A. (1995). StatLog: Comparison of Classification Algorithms on Large Real-World Problems.

[2] Hand, D. J., & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: a review. Journal of the Royal Statistical Society, Series A, 160, 523-541.

[3] Berardi, V. L., Patuwo, B. E., & Hu, M. Y. (2004). A principled Approach for Building and Evaluating Neural Network Classification Models. Decision Support Systems, 233-246.

[4] Eftekhar, B., Mohammad, K., Ardebili, H. E., Ghodsi, M., & Ketabchi, E. (2005). Comparison of Artificial Neural Network and Logistic Regression Models for Prediction of Mortality in Head Trauma based on Initial Clinical Data. BMC Medical Informatics and Decision Making, 5(3) Doi: 10.1186/1472-6947-5-3

[5] Eng, J. (2002). Predicting the Presence of Acute Pulmonary Embolism: A Comparative Analysis of the Artificial Neural, Logistic Regression, and Threshold Models. AJR American Journal of Roentgenology, 179(4).

[6] El-Sappagh, S.H.A., Hendawi, A.M.A., El Bastawissy, A.H.: A proposed model for data warehouse ETL processes. Journal of King Saud University – Computer and Information Sciences 23 (2011)

[7] Nadzornik računskih virov -- Load Sharing Facility – LSF (http://hpc.fs.uni-lj.si/lsf)

[8] "Utopia: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems". John Wiley & Sons. Retrieved 2007-12-29.

[9] Definition of data extraction. (http://www.extractingdata.com/)

[10] Rahm, E., Do, H. H.: Data Cleaning: Problems and Current Approaches. IEEE Bulletin on Data Engineering 23:4, 2000.

[11] M. S. Farhan, M. E. Marie, L. M. El-Fangary and Y. K. Helmy. "Transforming Conceptual Model into Logical Model for Temporal Data Warehouse Security: A Case Study", International Journal of Advanced Computer Science and Applications, 3 (3): pp. 115-122, 2012.

[12] Introduction to Table Partitioning by Jan Leers on 07 March 2013

[13] Gao Cong , Wenfei Fan , Floris Geerts , Xibei Jia , Shuai Ma, Improving data quality: consistency and accuracy, Proceedings of the 33rd international conference on Very large data bases, September 23-27, 2007, Vienna, Austria

[14] Stigler, Stephen M. (1989). "Francis Galton's Account of the Invention of Correlation". Statistical Science 4 (2): 73–79. doi:10.1214/ss/1177012580. JSTOR 2245329.

[15] Daniel T. Larose (2006). Naïve Bayes estimation: Data Mining Methods and Models. John Wiley & Sons, Feb 6, 2006.

[16] Chapter 5: Classification Method (http://www.d.umn.edu/~padhy005/Chapter5.html)

[17] Kanchan A.K., Mr. L.M.R.J.Lobo." Data Mining: You've missed it If Not Used" IOSR Journal of Computer Engineering (IOSR-JCE) ISSN: 2278-0661, ISBN: 2278-8727, PP: 06-14

[18] Mehmed Kantardzic, "Data Mining: Concepts, Models, Methods and Algorithms", © 2003 by John and Wiley Sons

[19] Chapter 5: Understanding Back Propagation – "Introduction to Neural Networks with Java" published by Jeff Heaton

[20] Haykin, Simon (1998). Neural Networks: A Comprehensive Foundation (2 ed.). Prentice Hall. ISBN 0-13-273350-1.

[21] John G. Carney and P´adraig Cunningham. "The Epoch Interpretation of Learning". Department of Computer Science University of Dublin Trinity College Ireland.1999

[22] John Robert Taylor (1999). An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements. University Science Books. pp. 128–129. ISBN 0-935702-75-X.

[23] http://www.vlsi-expert.com/2011/04/static-timing-analysis-sta-basic-part3a.html

[24] Static Timing Analysis for Nanometer Designs, by R. Chadha and J. Bhasker, ISBN 978-0-387-93819-6, Springer, 2009.

[25] EDA for IC Implementation, Circuit Design, and Process Technology edited by Luciano Lavagno, Louis Scheffer, Grant Martin

[26] Electrical Overstress (EOS): Devices, Circuits and Systems by Steven H. Voldman