CHAPTER 1

INTRODUCTION

1.1 Overview

India is a country that contains billions of people, but around half of them are sufferers of major diseases and one such disease is diabetes. Diabetes is common disease that is present around every household. Our project deals with devising a small predictive model that gives the overall increase or decrease in the diabetes ratio of Indian patients using the basic concept of linear regression which comes under the discipline of machine learning.

Machine learning is about learning to make predictions from examples of desired behavior or past observations. Learning methods have found numerous applications in performance modeling and evaluation. Machine learning techniques are preferred in situations where engineering approaches like hand-crafted models simply cannot cope with the complexity of the problem. Machine learning explores the construction and study of algorithms that can learn from and make predictions on the data. They operate by building a model from input data sets in order to make data-driven predictions or decisions.

Linear regression is an approach for modelling the relationship between a scalar dependent variable *y* and one or more explanatory (or independent variable) denoted *X*. The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression. In linear regression, data are modelled using linear predictor functions, and unknown
model parameters are estimated from the data. Such models are called linear models.

1.2 Problem Statement

The proposed system is built to predict the overall increase or decrease in the diabetes ratio of Indian patients using the data sets which are monitored from a website named UC Irvine Machine Learning Repository. According to these data sets the scattering of the data takes place through which a line that best fits

our scattered data is passed and then can be used to make data-driven predictions.

1.3 Objective

Our objective is to devise a predictive model using the concept of linear regression which comes under the discipline of Machine learning. Our model is built around the data sets which are chosen from UC Irvine Machine Learning Repository. These data sets are then being imported into our model using inbuilt python packages. To these unordered and redundant data sets a property of feature extraction is applied which results into a data sets that are non-redundant and can be used for easy interpretations. Linear regression algorithm is then used in the final step for passing a line that best fits our given model. This linear model then can be used to make data-driven predictions.

1.4 Organization of the report

The main body of the document is preceded by detailed contents including project abstract, lists of tables, and lists of figures followed by abbreviations used in the document.

Chapter1 provides us with the detailed understanding of the predictive model which we are going to design. An overview of the problem is provided with and objective to resolve them.

Chapter 2 discusses the existing system for its problem and issues. It shows the potential of how different approaches can be taken in order to rectify the preceding problems and issues. It also shows an insight to a proposed system which is used to predict the overall increase or decrease in the diabetes ratio of Indian patients using the method of linear regression which comes under the discipline of machine learning.

Chapter 3 includes the modules which are used in the step by step development of our predictive model. The modules are described in detail with figurative explanation.

Chapter 4 deals with the actual implementation of all modules using Python language platform along with screenshots of their execution. An introduction to the platform is also included along with packages which are used in coding.

Chapter 5 provides a conclusion to the project and future work which may further improve the system proposed.

CHAPTER 2

LITERATURE SURVEY

2.1 Introduction

Over the past two decades Machine Learning has become one of the mainstays of information technology and with that, a rather central, albeit usually hidden, part of our life. With the ever increasing amounts of data becoming available there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress. The purpose is to provide the application which have at their heart a machine learning problem and to bring some degree of order to the zoo of problems. After that, we will discuss some basic tools from statistics and probability theory, since they form the language in which many machine learning problems must be phrased to become amenable to solving. Finally, we will outline a data set and algorithm to solve the problem. More sophisticated tools, a discussion of problem and a detailed analysis will done.

- Machine learning is a scientific discipline that explores the construction and study of algorithms that can learn from data. Such algorithms operate by building a model based on inputs and using that to make predictions or decisions.
- We will be using the concept of linear regression which comes under the discipline of Machine learning.
- A regression uses the historical relationship between an independent and a dependent variable to predict the future values of the dependent variable.
- Our aim from this project is to devise a small prediction system that is used to estimate the overall decrease or increase in diabetes ratio of Indian patients.
- Python programming would be used as a platform to implement the linear regression algorithm.

2.2 EXISTING SYSTEM

The main goal of the existing system was to study and apply as many Machine Learning Algorithms as possible on a data involving a particular domain, namely the Stock Market, as opposed to coming up with a newer (and/or better) algorithm that is more efficient in predicting the price of a stock.
In existing system, Machine Learning techniques which have been applied for stock trading to predict the rise and fall of stock prices before the actual event of an increase or decrease in the stock price occurs. In particular the application of Support Vector Machines, Prediction using Decision Stumps, Expert Weighting and Online Learning in detail along with the benefits and pitfalls of each method. The existing system introduces the parameters and variables that can be used in order to recognize the patterns in stock prices which can be helpful in the future prediction of stocks and Boosting is combined with learning algorithm to improve the accuracy of prediction systems.

2.3 ISSUES OF EXISTING SYSTEM

1. Recently, in the area of applying Machine Learning Algorithms for analyzing price patterns and predicting stock prices and index changes. Most stock traders nowadays depend on Intelligent Trading Systems which help them in predicting prices based on various situations and conditions, thereby helping them in making instantaneous investment decisions.

2. Stock Prices are considered to be very dynamic and susceptible to quick changes because of the underlying nature of the financial domain and in part because of the mix of known parameters (Previous Days Closing Price, P/E Ratio etc.) and unknown factors (like Election Results, Rumors etc.)

3. An intelligent trader would predict the stock price and buy a stock before the price rises, or sell it before its value declines. Though it is very hard to replace the expertise that an experienced trader has gained, an accurate prediction algorithm can directly result into high profits for investment firms, indicating a

direct relationship between the accuracy of the prediction algorithm and the profit made from using the algorithm.

2.4 SUMMARY OF LITERATURE SURVEY

The issues which we faced in the existing system were acting as hindrances to their proper development. The developers used different technology and concepts to carry over their research. All have attained their maximum to complete their work successfully. Somehow they all struck in a point, which does not allow them to come out magnificent.

CHAPTER 3

SYSTEM DESIGN

3.1 INTRODUCTION

Our system is based on the concept of Linear regression which will be useful in predicting the increase or decrease in the diabetes ratio by creating a linear model through which a line passes which best-fits our given data. These lines then can be useful in making appropriate data-driven predictions.

3.2 SYSTEM ARCHITECTURE AND FLOW DIAGRAM

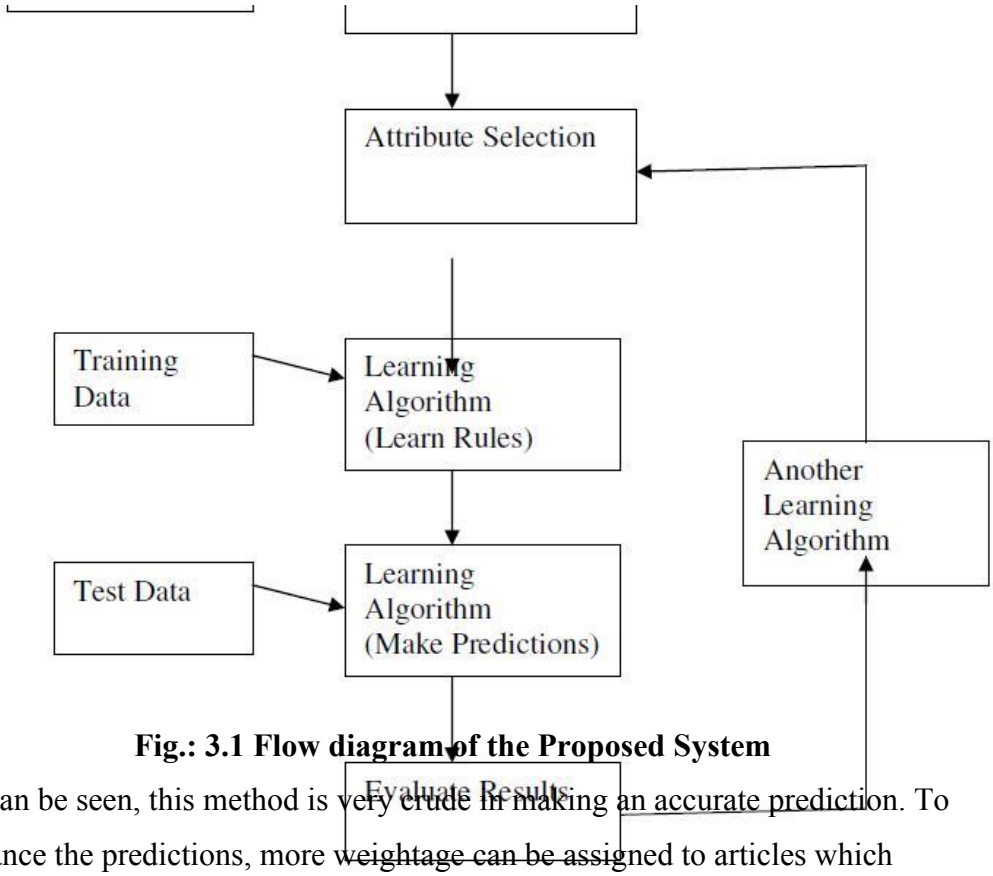

Fig.: 3.1 Flow diagram of the Proposed System

As can be seen, this method is very crude in making an accurate prediction. To enhance the predictions, more weightage can be assigned to articles which come from credible sources. Also, more weightage can be assigned to a news Headline which contains a Positive Prediction Term or a Negative Prediction Term. Boosting can then be applied to the base learners to see if the accuracy can be boosted further.

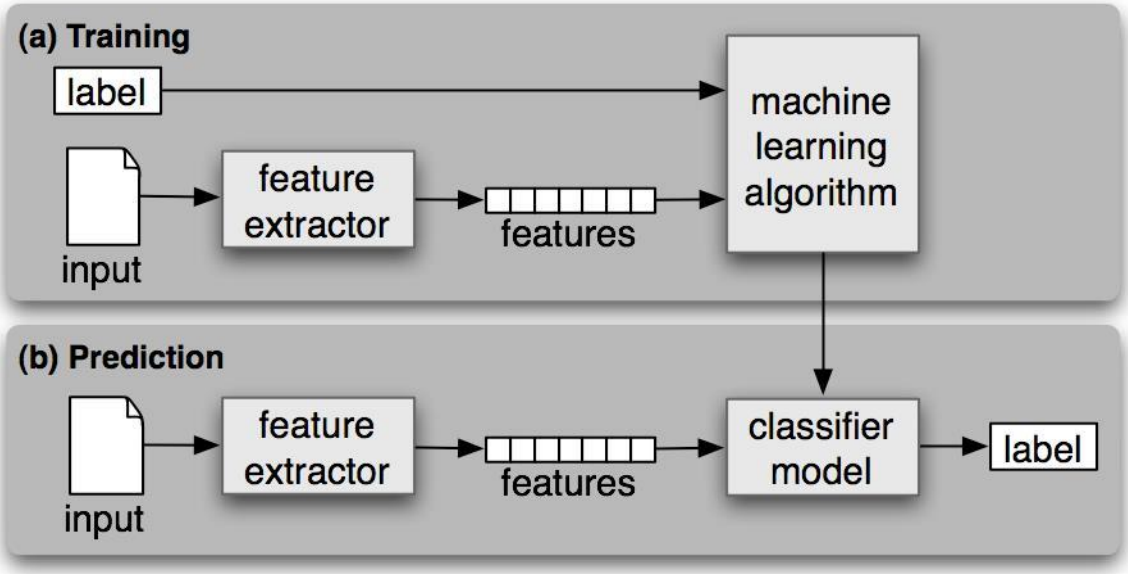

Fig.: 3.2 Activity Diagram of the Proposed System

3.2.1 DESCRIPTION OF PROPOSED SYSTEM

- A predictive model using the concept of linear regression which comes under the discipline of Machine learning.
- A regression uses the historical relationship between an independent and a dependent variable to predict the future values of the dependent variable.
- The estimation of overall increase or decrease in the ratio of diabetes for Indian patients can be given from our predictive model.
- Python would be used as a platform to implement this logic and give a graph to implement our logic.

3.3 SYSTEM REQUIREMENTS

3.3.1 HARDWARE SPECIFICATION

- CPU : Pentium 4 or above
- Clock Speed: 1.2 Ghz or above
- Ram: 128MB or above
- Storage: 10GB

3.3.2 SOFTWARE SPECIFICATION

- Windows XP/Vista/7/8/8.1 or above
- Python Compiler: To operate and perform various heuristic Algorithms.
- Pygame package for Python
- Text Editor or IDLE

CHAPTER 4

MODULE DESCRIPTION

4.1 INTRODUCTION

The basic structure of our predictive model can be given as follows:-

1. Import all the necessary packages which are used for the implementation of our program.
2. The appropriate datasets are monitored and are collected from the website **https://archive.ics.uci.edu/ml.**
3. Feature extraction is performed on the input datasets as these datasets may be too large to process and maybe redundant.
4. Linear regression model is being implemented which gives us the relationship between a scalar dependent variable and one or more explanatory variables.
5. Using the above given relationship we describe the equation for a line which best fits our data which in turn can be used for prediction in our model.
6. Graphs are being implemented to show the scattering of the given datasets and fit a line which best fits the scattered data and then can be used to make predictions.
7. The values may be numbers, such as real numbers or integers, for example representing a person's height in centimeters, but may also be nominal data , for example representing a person's ethnicity. More generally, values may be of any of the kinds described as a level of measurement.

LIST OF MODULES

1. Training data sets and interpretation
2. Feature vectors and extraction
3. Linear regression

4.2 Training Data sets and Interpretation

4.2.1 ARCHITECTURE AND FLOW DIAGRAM

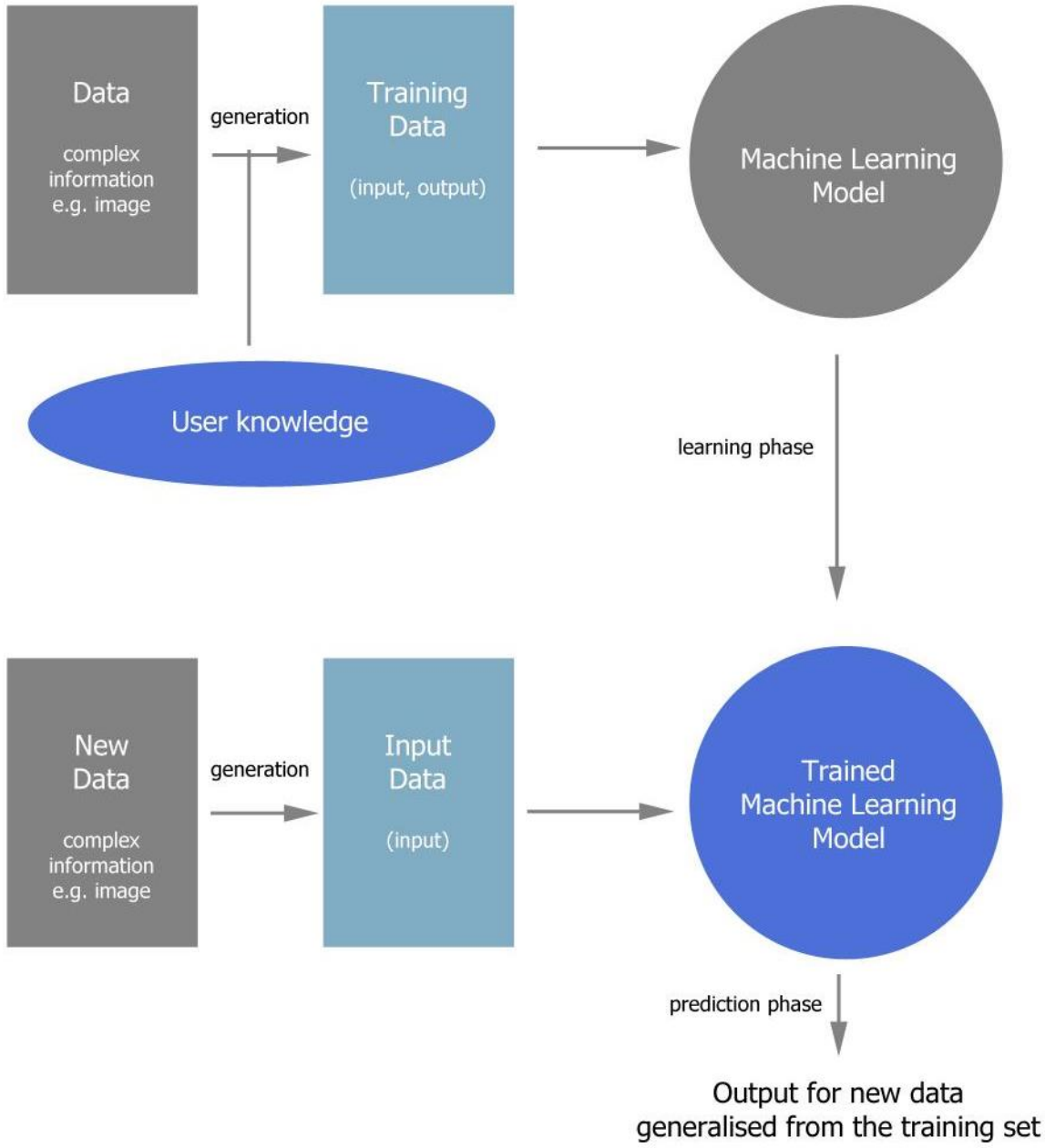

Fig.: 4.1 Architecture diagram representing Training Sets Generation

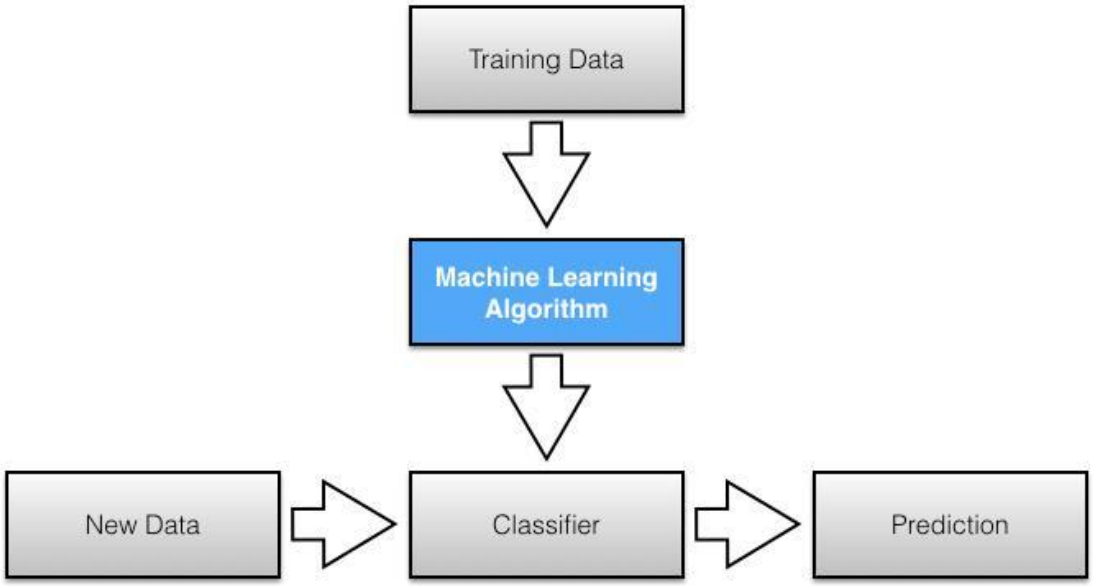

Fig.: 4.2 Flow Diagram Representing training data

4.2.2 DESCRIPTION

A dataset is a dictionary-like object that holds all the data and some metadata about the data. Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set in question. The data is stored in the form which is a n_samples , n_features array.

Several characteristics define a data set's structure and properties. These include the number and types of the attributes or variables, and various statistical measures applicable to them, such as standard deviation .

The values may be numbers, such as real numbers or integers, for example representing a person's height in centimeters, but may also be nominal data , for example representing a person's ethnicity. More generally, values may be of any of the kinds described as a level of measurement.

In statistics, datasets usually come from actual observations obtained by sampling a statistical population, and each row corresponds to the

observations on one element of that population. Datasets may further be generated by algorithms for the purpose of testing certain kinds of software.

Initially the datasets are complex, unordered, redundant and may take a long amount of time in its processing. To these data sets a property named feature extraction is applied which makes our data non-redundant and informative.

Here in our model we have monitored all the appropriate data sets and one such data set is taken from the website using the urllib2 package which is an inbuilt package present in our python platform.

The input dataset are in the form of a CSV(comma-separated value) file which is present in on your local workstation or on a remote server. This CSV file is then downloaded using the urllib2 package and then implemented in our code.

4.3 Feature Extraction

4.3.1 FLOW DIAGRAM

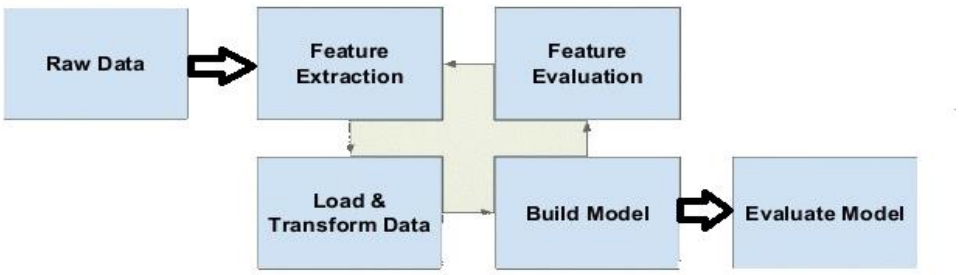

Fig.: 4.3 Flow diagram representing feature extraction process

4.3.2 DESCRIPTION

In machine learning feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative, non redundant, facilitating the subsequent learning and generalization steps, in

some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

When the raw input data is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it is transformed into a reduced set of features (also named features vector). This process is called feature extraction. The extracted features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. To this reduced representation further features are being added and their step by step evaluation takes place using the property of feature evaluation which results in the datasets which can give better interpretations.

4.4 LINEAR REGRESSION

4.4.1 ARCHITECTURE AND FLOW DIAGRAM

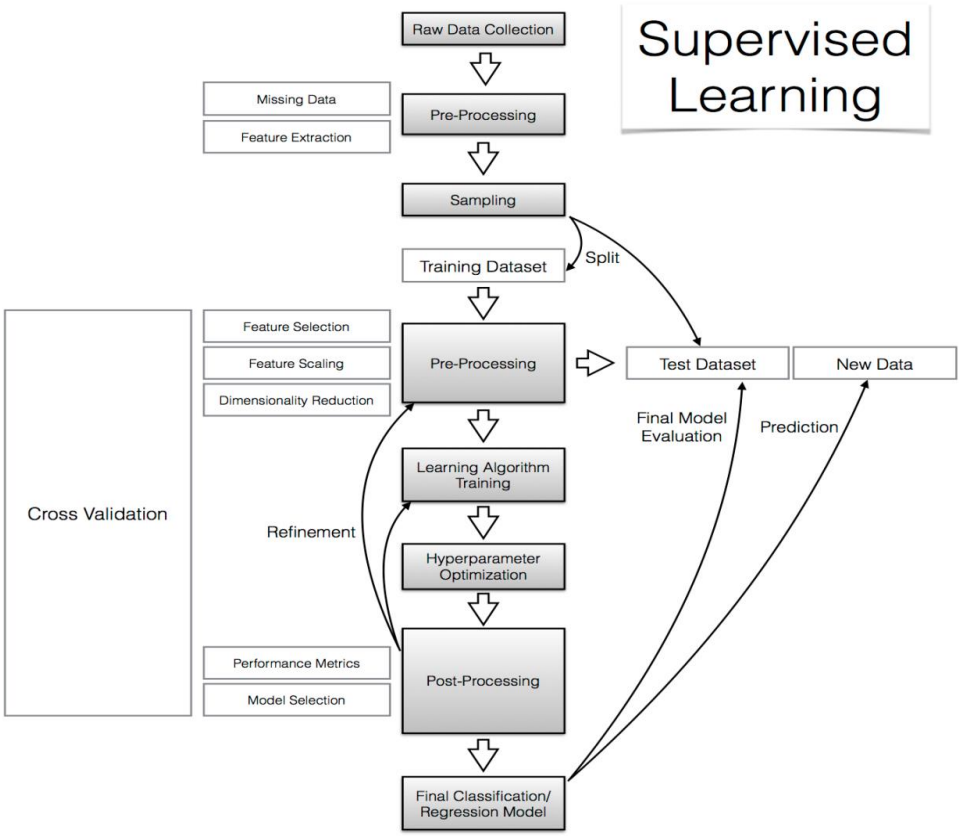

Fig.: 4.4 Flow diagram representing a Linear Model

4.4.2 DESCRIPTION

Regression is a statistical technique to determine the linear relationship between two or more variables. Regression is primarily used for prediction and causal inference. In its simplest (bivariate) form, regression shows the relationship between one independent variable (X) and a dependent variable (Y), as in the formula below:

$$Y= \beta0 + \beta1X + u$$

The magnitude and direction of that relation are given by the slope parameter (β1), and the status of the dependent variable when the independent variable is absent is given by the intercept parameter (β0). An error term (u) captures the amount of variation not predicted by the slope and intercept terms. The regression coefficient (R2) shows how well the values fit the data. Regression thus shows us how variation in one variable co-occurs with variation in another. What regression cannot show is causation; causation is only demonstrated analytically, through substantive theory. For example, a regression with shoe size as an independent variable and foot size as a dependent variable would show a very high regression coefficient and highly significant parameter estimates, but we should not conclude that higher shoe size causes higher foot size. All that the mathematics can tell us is whether or not they are correlated, and if so, by how much. It is important to recognize that regression analysis is fundamentally different from ascertaining the correlations among different variables. Correlation determines the strength of the relationship between variables, while regression attempts to describe that relationship between these variables in more detail.

Using lin_reg.fit(X,Y) creates a linear model of the responses Y to the data matrix X which was created by the module feature extraction. This linear model can be defined as a best fit line that passes through the scattered points on the graph. If a 'best fit' line is found, it can be used as the basis for estimating the future values of the function by extending it while maintaining its slope.

4.5 SUMMARY

In the module description above we mentioned about 3 key modules that were used during the implementation and simulation of this project. Each of this module contributes significantly towards the step by step progress of our model. All these modules have been tested and successfully executed.

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 Introduction

Python is been chosen as a platform because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

5.2 Overview of the Platform

5.2.1 Python programming language

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.

Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

5.2.2 Python packages

The Python Package Index or PyPI is the official third-party software repository for the Python programming language. Python developers intend it to be a comprehensive catalog of all open source Python packages.

While the PyPI website is maintained by the Python Software Foundation, its contents are uploaded by individual package maintainers. Python package managers such as pipdefault to downloading packages from PyPI.

5.2.3 NumPy

NumPy is an extension to the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers.

Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open source and has many contributors.

Array Creation

```
import numpy as np
>>> x = np.array([1, 2, 3])
>>> x
array([1, 2, 3])
>>> y = np.arange(10)  # like Python's range, but returns an array
>>> y
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Basic Operations

```
>>> a = np.array([1, 2, 3, 6])
>>> b = np.linspace(0, 2, 4)  # create an array with 4 equally spaced points
starting with 0 and ending with 2.
>>> c = a - b
>>> c
array([ 1. ,  1.33333333 ,  1.66666667,  4.        ])
>>> a**2
array([ 1,  4,  9, 36])
```

5.2.4 Urllib2

The urllib2 module defines functions and classes which help in opening URLs (mostly HTTP) in a complex world — basic and digest authentication, redirections, cookies and more.

The urllib2 module defines the following functions:

urllib2.urlopen(url*[*, data*[*, timeout*[*, cafile*[*, capath*[*, cadefault*[*, context*]]]]]*)
Open the URL url, which can be either a string or a Request object.

data may be a string specifying additional data to send to the server, or None if no such data is needed. Currently HTTP requests are the only ones that use data; the HTTP request will be a POST instead of a GET when the data parameter is provided.

Data should be a buffer in the standard application/x-www-form-url encoded format. The urllib.urlencode () function takes a mapping or sequence of 2-tuples and returns a string in this format. Urllib2 module sends HTTP/1.1 requests with Connection: close header included.

The optional timeout parameter specifies a timeout in seconds for blocking operations like the connection attempt (if not specified, the global default timeout setting will be used). This actually only works for HTTP, HTTPS and FTP connections.

If context is specified, it must be a ssl.SSLContext instance describing the various SSL options. See HTTPSConnection for more details.

The optional cafile and capath parameters specify a set of trusted CA certificates for HTTPS requests. cafile should point to a single file containing a bundle of CA certificates, whereas capath should point to a directory of hashed certificate files. More information can be found inssl.SSLContext.load_verify_locations().

The cadefault parameter is ignored.

This function returns a file-like object with three additional methods:

- Raises URLError on errors.

- Note that none may be returned if no handler handles the request (though the default installed global Opener Director uses Unknown Handler to ensure this never happens).

- In addition, if proxy settings are detected (for example, when a *_proxy environment variable like http_proxy is set), Proxy Handler is default installed and makes sure the requests are handled through the proxy.

This example gets the python.org main page and displays the first 100 bytes of it:

>>> import urllib2

>>> f = urllib2.urlopen('http://www.python.org/')

>>> print f.read(100)

Adding HTTP headers:

Use the headers argument to the Request constructor:

Import urllib2

req = urllib2.Request('http://www.example.com/')

req.add_header ('Referrer', 'http: //www.python.org/')

r = urllib2.urlopen (req)

5.2.5 Matplotlib

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like wxPython, Qt, orGTK+.

There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB. SciPy makes use of matplotlib.

matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, the python and I python shell (ala MATLAB®* or Mathematical®†), web application servers, and six graphical user interface toolkits.

matplotlib tries to make easy things easy and hard things possible.

You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code.

For simple plotting the pyplot interface provides a MATLAB-like interface, particularly when combined with IPython.

For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Example Code:

Line plot

```
>>> import matplotlib.pyplot as plt
>>> import numpy as np
>>> a = np.linspace(0,10,100)
>>> b = np.exp(-a)
>>> plt.plot(a,b)
>>> plt.show()
```

5.2.6 Sklearn

Scikit-learn (formerly scikits.learn) is an open source machine learning library for the Python programming language.

It features various classification, regression and clustering algorithms including support vector machines, logistic regression, naive Bayes, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

However, coefficient estimates for Ordinary Least Squares rely on the independence of the model terms. When terms are correlated and the columns of the design matrix X have an approximate linear dependence, the design matrix becomes close to singular and as a result, the least-squares estimate becomes highly sensitive to random errors in the observed response, producing a large variance. This situation of multi-collinearity can arise, for example, when data are collected without an experimental design.

5.3 Implementation Details

5.3.1 Simulation Parameters

```
Python 2.7.9 (default, Dec 10 2014, 12:24:55) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ================================ RESTART ================================
>>>
(768, 9)
[[   6.     148.      72.     ...,   33.6      0.627   50.   ]
 [   1.      85.      66.     ...,   26.6      0.351   31.   ]
 [   8.     183.      64.     ...,   23.3      0.672   32.   ]
 ...,
 [   5.     121.      72.     ...,   26.2      0.245   30.   ]
 [   1.     126.      60.     ...,   30.1      0.349   47.   ]
 [   1.      93.      70.     ...,   30.4      0.315   23.   ]] [[   6.     148.
     72.     ...,   33.6      0.627   50.   ]
 [   1.      85.      66.     ...,   26.6      0.351   31.   ]
 [   8.     183.      64.     ...,   23.3      0.672   32.   ]
 ...,
 [   5.     121.      72.     ...,   26.2      0.245   30.   ]
 [   1.     126.      60.     ...,   30.1      0.349   47.   ]
 [   1.      93.      70.     ...,   30.4      0.315   23.   ]]
[[  1.75855810e-02   9.66424666e-02   3.59190374e-02 ...,   9.24160351e-02
    1.79333494e-02   2.97476437e-01]
 [  2.67400442e-02   2.58861630e-01   1.88107496e-02 ...,   3.16799053e-03
    1.03937947e-01   1.81270778e-02]
 [  9.41569390e-03   5.30741876e-02   4.33007810e-02 ...,   6.17558462e-02
    4.46224171e-02   4.10489328e-01]
 ...,
 [  1.13538450e-04   2.00423274e-03   3.96779177e-01 ...,   6.65928345e-02
    7.46427980e-02   1.16570019e-01]
 [  3.43812254e-02   1.78400675e-01   2.79644203e-03 ...,   2.07884981e-02
    3.18119702e-02   3.40446670e-01]
 [  1.73550703e-01   3.87683093e-02   2.21452608e-01 ...,   1.16699842e-03
    1.19868795e-01   1.86879509e-02]]
```

Fig.: 5.1 Parameters used in the form of a matrix

5.4 Sample coding

```
Import numpy as np
Import urllib2
Import numpy
Import matplotlib.pyplot as plt
from sklearn import datasets,linear_model
from sklearn.cross_validation import train_test_split
url="https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"
raw_data=urllib2.urlopen(url)
dataset=np.loadtxt(raw_data,delimiter=",")
print(dataset.shape)
x=dataset[:,0:8]
y=dataset[:,0:8]
lin_reg= linear_model.LinearRegression()
print x,y
lin_reg.fit(x,y)
predicted_y=lin_reg.predict(x)
mse=np.random.normal(0.0,0.5,size=(1000,10))**2
mse=mse/np.sum(mse,axis=1)[:,None]
plt.pcolor(mse)
maxvi = np.argsort(mse,axis=1
ii = np.argsort(maxvi[:,-1])
plt.pcolor(mse[ii,:])
print mse
plt.figure()
plt.scatter(x,y,color='red',label='actual points')
plt.plot(x,predicted_y,color='blue',label='fitted line')
plt.legend(loc='upper right')
 plt.show ()
```

```
def gradient_descent(alpha, x, y, ep=0.0001, max_iter=10000):
converged = False
iter = 0
m = x.shape[0] # number of samples

# initial theta
t0 = np.random.random(x.shape[1])
t1 = np.random.random(x.shape[1])

# total error, J(theta)
J = sum([(t0 + t1*x[i] - y[i])**2 for i in range(m)])

# Iterate Loop
while not converged:
# for each training sample, compute the gradient (d/d_theta j(theta))
grad0 = 1.0/m * sum([(t0 + t1*x[i] - y[i]) for i in range(m)])
grad1 = 1.0/m * sum([(t0 + t1*x[i] - y[i])*x[i] for i in range(m)])

# update the theta_temp
temp0 = t0 - alpha * grad0
temp1 = t1 - alpha * grad1

# update theta
t0 = temp0
t1 = temp1

# mean squared error
e = sum( [ (t0 + t1*x[i] - y[i])**2 for i in range(m)] )

if abs(J-e) <= ep:
print 'Converged, iterations: ', iter, '!!!'
```

```
converged = True

J = e   # update error
iter += 1  # update iter

if iter == max_iter:
print 'Max interactions exceeded!'
converged = True

return t0,t1

if __name__ == '__main__':

x, y = make_regression(n_samples=100, n_features=1, n_informative=1,
random_state=0, noise=35)
print 'x.shape = %s y.shape = %s' %(x.shape, y.shape)

alpha = 0.01 # learning rate
ep = 0.01 # convergence criteria

# call gredient decent, and get intercept(=theta0) and slope(=theta1)
theta0, theta1 = gradient_descent(alpha, x, y, ep, max_iter=1000)
print ('theta0 = %s theta1 = %s') %(theta0, theta1)

# check with scipy linear regression
slope, intercept, r_value, p_value, slope_std_error = stats.linregress(x[:,0], y)
print ('intercept = %s slope = %s') %(intercept, slope)

# plot
for i in range(x.shape[0]):
y_predict = theta0 + theta1*x
```

```
pylab.plot(x,y,'o')
pylab.plot(x,y_predict,'k-')
pylab.show()
def gradient_descent_2(alpha, x, y, numIterations):
m = x.shape[0] # number of samples
theta = np.ones(2)
x_transpose = x.transpose()
for iter in range(0, numIterations):
hypothesis = np.dot(x, theta)
loss = hypothesis - y
J = np.sum(loss ** 2) / (2 * m)  # cost
print "iter %s | J: %.3f" % (iter, J)
gradient = np.dot(x_transpose, loss) / m
theta = theta - alpha * gradient  # update
return theta
if __name__ == '__main__':
x, y = make_regression(n_samples=100, n_features=1, n_informative=1,
random_state=0, noise=35)
m, n = np.shape(x)
x = np.c_[ np.ones(m), x] # insert column
alpha = 0.01 # learning rate
theta = gradient_descent_2(alpha, x, y, 1000)
# plot
for i in range(x.shape[1]):
y_predict = theta[0] + theta[1]*x
pylab.plot(x[:,1],y,'o')
pylab.plot(x,y_predict,'k-')
pylab.show()
```

5.4.1 Screen Shots

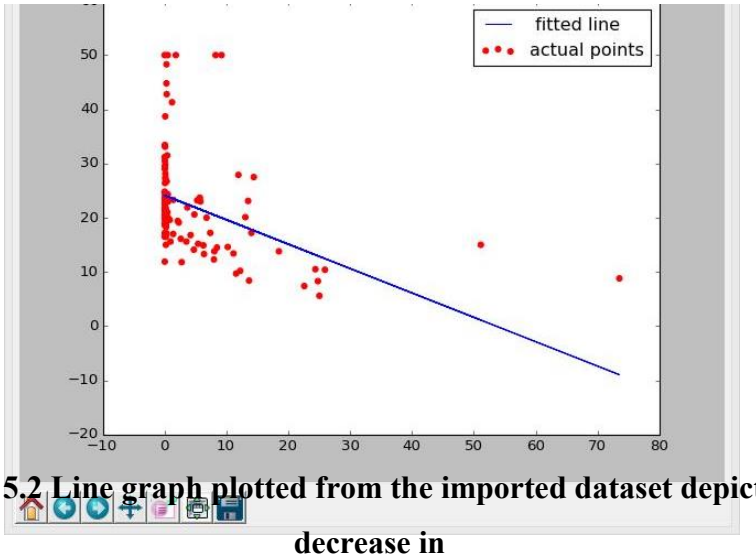

Fig.: 5.2 Line graph plotted from the imported dataset depicting decrease in diabetes ratio

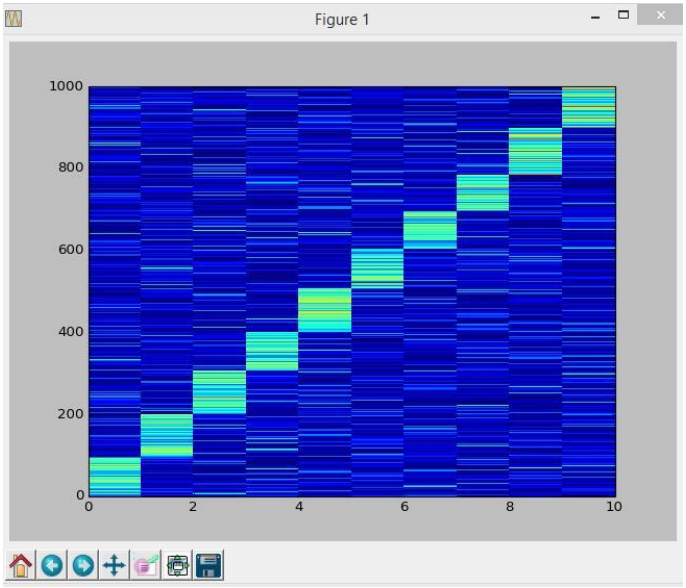

Fig.: 5.3 Pcolor graph plotted for Indian diabetes ratio after sorting by the probability that they belong to a cluster.

5.5 Summary

We have developed a program to implement a prediction system to predict the future trends.
This model has been implemented for predicting Indians-diabetes rate from a chosen data set obtained from the UC Irvine Machine Learning Repository. Parameters given as a dataset have been imported from the URL link" https://archive.ics.uci.edu/ml/machine-learning-databases/pima-indians-diabetes/pima-indians-diabetes.data"

Python programming language is use as a platform for the development of the prediction system using machine learning. Pre-Required packages for the implementation of the prediction system have been installed from Python Package Index or PyPI is the official third-party software repository for the language. Packages used in the project are: - "NumPy

Urllib2, Matplotlib and Sklearn".The detailed description of the packages are given in their respective section.

The sample code shows the use of the dataset imported from the given URL and the graph is generated by the sample code which plots the matrix function (dataset format) using the linear regression algorithm. The screenshot of the output graph hence generated is shown.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

Linear regression has been widely used for years applying to almost all the areas in our lives. In this paper, we presented a process of building a Linear regression model for a simplified problem of estimating the increase or decrease in the diabetes ratio of the Indian patients. This process involves three steps: a) Data sets and interpretation to import the chosen data sets which supports our predictive model; b) Feature Extraction process which makes the imported datasets non-redundant and can be used for easy interpretations; c) Linear regression algorithm to pass a line that best-fits our given data which in turn can be used to make data-driven predictions. This is a typical process of building regression models that may apply to many applications where predictive modeling is the goal.

6.2 FUTURE WORK

Some issues that can be faced during the algorithm are that the data sets which we monitored may not be accurate and they may still be redundant even after applying the process of feature extraction.

These problems need to get rectified and our future work will mainly be focusing on resolving the issues we faced in our model. More efforts would be put in to maximize the predictions which we have obtained from our model so that it can be used as a high scale predictive system in the near future.

REFRENCES

- D.zeller, M.olson, o.Blume, A.Ferling, W.Tomaselli and I.Godor,"Predictive System For stock Marketing-the Future project", The Future Internet,ser. Lecture Notes in computer.
- On accelerating content delivery in machine learning tao Han, IEEE, Nirwan Ansari, Mingquan Wu, Heather Yu.
- Joint Downlink Base Station Association and Power Control using machine learning.
- "Artificial Intelligence" By Elaine Rich and Kevin Knight.
- http://www.bogotobogo.com/python/python_numpy_batch_gradient_descent_algorithm.php
- https://archive.ics.uci.edu/ml For machine learning data sets.
- http://www.personal.umich.edu/~copyrght/image/solstice/sum07/Solstice_GoutamiED.pdf
- http://www.acisinternational.org/Springer/SamplePaper.pdf