

DISEASE PREDICTION

A PROJECT REPORT

Submitted By
AMAN RAGHAVA
(2100290140020)
SHUBHANGI SINGH
(2100290140133)

**Submitted in partial fulfilment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

Under the Supervision of
DR. ARUN KUMAR TRIPATHI
HEAD OF DEPARTMENT



Submitted to

DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206

(DEC 2022)

CERTIFICATE

Certified that **AMAN RAGHAVA** (2100290140020), **SHUBHANGI SINGH** (2100290140133) has/ have carried out the project work having “**DISEASES PREDICATION USING MACHINE LEARNING**” for Master of Computer Applications from **Dr. A.P.J. Abdul Kalam Technical University (AKTU)** (formerly UPTU), Technical University, Lucknow under my supervision.

The project report embodies original work, and studies are carried out by the student himself /herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

AMANRAGHAVA
Roll No:(2100290140020)
SHUBHANGI SINGH
Roll No:(2100290140133)

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

DR. ARUN KUMAR TRIPATHI
Associate Professor
Department of Compute Applications
KIET Group of Institutions, Ghaziabad

Signature of Internal Examiner

Signature of External Examiner

DR. ARUN KUMAR TRIPATHI
Head, Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

Disease Prediction using Machine Learning is the system that is used to predict the diseases from the symptoms which are given by the patients or any user. The system processes the symptoms provided by the user as input and gives the output as the probability of disease. The disease is predicted using algorithms and comparison of the datasets with the symptoms provided by the user. The system processes the symptoms provided by the user as input and gives the output as the probability of disease.

The disease prediction system predicts diseases based on patient's symptoms and also some commonly prescribed medicines for a particular disease. Disease Prediction using Machine Learning is the system that is used to predict the diseases from the symptoms which are given by the patients or any user. The system processes the symptoms provided by the user as input and gives the output as the probability of disease.

ACKNOWLEDGEMENT

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Dr. Arun Kumar Tripathi** for his guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Aman Raghava

Shubhangi Singh

TABLE OF CONTENT

Chapter 1 – Introduction	6-10
1.1 Project description	
1.2 Project Scope	
1.3 Hardware / Software used in Project	
Chapter 2 - Feasibility Study	11-18
2.1 Technical feasibility	
2.2 Operational Feasibility	
2.3 Behavioral Feasibility	
2.4 Operational Feasibility	
Chapter 3 - Database Design	19-25
3.1 Database Tables	
3.2 Flow Chart	
3.3 Use Case Diagram	
3.4 Sequence Diagram	
3.5 Collaborative Diagram	
Chapter 4 - Form Design	26-45
4.1 Input / Output Form (Screenshot)	
Chapter 5 – Coding	46-55
Module wise code	
Chapter 6 - Testing and deployment	56-58
6.1 Test Case-1	
Chapter 7 – Conclusion	59
Chapter 8- Bibliography	62

LIST OF FIGURES

Figure No.	Name of Figure	Page No.
3.1	Cross Validation	19
6.1	Gantt Chart	32
6.2	Data Flow Diagram	34
6.3	Activity Diagram	35
6.4	Sequence Diagram	37
6.5	Use Case Diagram	39

Chapter 1

INTRODUCTION

With the advancement in technology, Machine Learning is becoming more popular and commonly use technology by industry experts for solving problems faced in real life. Machine Learning is the scientific study of algorithms and statistical models that computer to perform a specific task without using explicit instructions, relying on patterns and inference instead. Machine Learning is also used by the healthcare industry to bring advancement in their techniques so that they can provide better services to their patients. The disease prediction system predicts diseases based on patient's symptoms and also some commonly prescribed medicines for a particular disease.

1.1 Project Description

Disease Prediction using Machine Learning is the system that is used to predict the diseases from the symptoms which are given by the patients or any user. The system processes the symptoms provided by the user as input and gives the output as the probability of disease.

The disease is predicted using algorithms and comparison of the datasets with the symptoms provided by the user.

1.2 Project Scope

The user interacts with the Prediction Engine by filling a collection of symptoms that holds the parameter set provided as associate input to the trained models. The Prediction Engine makes use of 3 algorithms to predict the presence of a unwellness namely: call Tree, Random Forest and Naive Bayes.

1.3 HARDWARE AND SOFTWARE

. Python

. Flask

. Cross Validation (CV)

. Pandas

. Numpy

. HTML

. CSS

. Scikit-Learn

. Machine Learning



PYTHON

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.

Often, programmers fall in love with Python 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 very simple approach.

Flask

Flask is a web application framework written in Python. It was developed by Armin, who led a team of international Python enthusiasts called. Flask is based on the WSGI toolkit and the Jinja2 template engine. Both are projects.

Microframework

Flask is often referred to as a microframework. It is designed to keep the core of the application simple and scalable. Instead of an abstraction layer for database support, Flask supports extensions to add such capabilities to the application.

Unlike the Django framework, Flask is very Pythonic. It's easy to get started with Flask, because it doesn't have a huge learning curve.

On top of that it's very explicit, which increases readability. To create the "Hello World" app, you only need a few lines of code.

Cross-Validation

Suppose you train a model on a given dataset using any specific algorithm. You tried to find the accuracy of the trained model using the same training data and found the accuracy to be 95% or maybe even 100%. Is your model ready for prediction? The answer is no. Why?

Because your model has trained itself on the given data, it knows the data and it has generalized over it very well. But when you try and predict over a new set of data, it's most likely to give you very bad accuracy, because it has never seen the data before and thus it fails to generalize well over it. This is the problem of overfitting. To tackle such problem, Cross-validation comes into the picture. Cross-validation is a resampling technique with a basic idea of dividing the training dataset into two parts train and test. On one part (train) you try to train the model and on the second part (test) the data which is unseen for the model, you make the prediction and check how well your model works on it. If the model works with good accuracy on your test data, it means that the model has not overfitted the training data and can be trusted with the prediction, whereas if it performs with bad accuracy then our model is not to be trusted and we need to tweak our algorithm.

Let's see the different approaches of Cross-Validation:

Hold Out Method:

It is the most basic of the CV techniques. It simply divides the dataset into two sets an of training and test. The training dataset is used to train the model and then test data is fitted in the trained model to make predictions. We check the accuracy and assess our model on that basis.

Chapter - 3

k-fold Cross-Validation:

To tackle the high variance of Hold-out method, the k-fold method is used. The idea is simple, divide the whole dataset into 'k' sets preferably of equal sizes. Then the first set is selected as the test set and the rest 'k-1' sets are used to train the data. Error is calculated for this particular dataset. Then the steps are repeated, the second set is selected as the test data, and the remaining 'k-1' sets are used as the training data. Again, the error is calculated. Similarly, the process continues for 'k' times. In the end, the CV error is given as the mean of the total errors calculated individually, mathematically given as:

$$cv_{(k)} = \frac{1}{k} \sum_{i=1}^k MSE_i$$

Figure 3.1

The variance in error decreases with the increase in 'k'. The disadvantage of k-fold cv is that it is computationally expensive as the algorithm runs from scratch for 'k' times.



Figure 3.2



Library Highlights

- > A fast and efficient Data Frame object data manipulation with integrated indexing.
- > Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format.
- > Intelligent data alignment and integrated handling of missing data: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form.
- > Flexible reshaping and pivoting of data sets.
- > Intelligent label-based slicing, fancy indexing, and s of large data sets.
- > Columns can be inserted and deleted from data structures for size mutability.
- > Aggregating or transforming data with a powerful group by engine allowingsplit-apply- combine operations on data sets.
- > High performance merging and joining of data sets.



Figure 3.4

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation.

At the core of the NumPy package, is the object. This encapsulate n -dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically).

Changing the size of a will create a new array and delete the original.

The elements in a NumPy array are all required to be of the same data type, and an array thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.

NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.



Figure 3.5

HTML is an acronym which stands for **Hyper Text Markup Language** which is used for creating web pages and web applications. Let's see what is meant by Hypertext Markup Language, and Web page.

Hyper Text: Hyper Text simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. Hyper Text is a way to link two or more web pages (HTML documents).

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. With the help of HTML only, we can create static web pages.

BRIEF SYNTAX OF HTML:

<!DOCTYPE>

<html>

<head>

<title>Web page title</title>

</head>

<body>

<h1>Write Your First Heading</h1>

<p>Write Your First Paragraph.</p>

</body>

</html>

Figure 3.6

```
<!DOCTYPE html>
<html>

  <head>
    <title>My First Webpage</title>
  </head>

  <body>
    <h1>
      My First Webpage
    </h1>
    <p>This is a paragraph...</p>
  </body>

</html>
```



Figure 3.7

CSS stands for Cascading Style Sheets. It is a style sheet language which is used to describe the look and formatting of a document written in markup language. It provides an additional feature to HTML. It is generally used with HTML to change the style of web pages and user interfaces. It can also be used with any kind of XML documents including plain XML, SVG and XUL.

What does CSS do

- o You can add new looks to your old HTML documents.
- o You can completely change the look of your website with only a few changes in CSS Code.

Why use CSS

These are the three major benefits of CSS:

1) Solves a big problem

Before CSS, tags like font, background style, element alignments, border and size had to be repeated on every web page.

This was a very long process. For example: If you are developing a large website where fonts and information are added on every single page, it will become a long and expensive process. CSS was created to solve this problem. It was a W3 recommendation.

2) Saves a lot of time

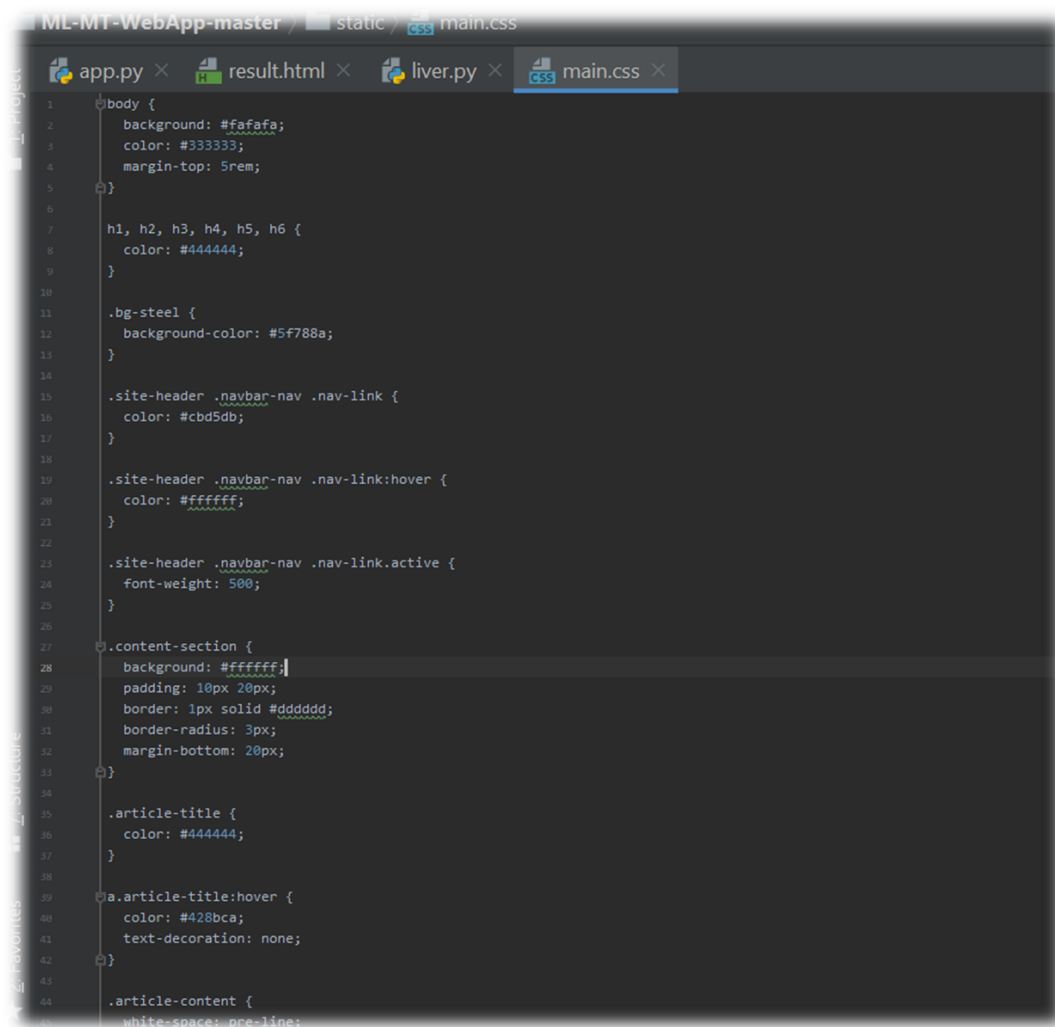
CSS style definitions are saved in external CSS files so it is possible to change the entire website by changing just one file.

3) Provide more attributes

CSS provides more detailed attributes than plain HTML to define the look and feel of the website.

BELOW IS THE CSS CODE USED IN THIS PROJECT

Figure 3.8



```
1 body {
2     background: #fafafa;
3     color: #333333;
4     margin-top: 5rem;
5 }
6
7 h1, h2, h3, h4, h5, h6 {
8     color: #444444;
9 }
10
11 .bg-steel {
12     background-color: #5f788a;
13 }
14
15 .site-header .navbar-nav .nav-link {
16     color: #cbd5db;
17 }
18
19 .site-header .navbar-nav .nav-link:hover {
20     color: #ffffff;
21 }
22
23 .site-header .navbar-nav .nav-link.active {
24     font-weight: 500;
25 }
26
27 .content-section {
28     background: #ffffff;
29     padding: 10px 20px;
30     border: 1px solid #dddddd;
31     border-radius: 3px;
32     margin-bottom: 20px;
33 }
34
35 .article-title {
36     color: #444444;
37 }
38
39 a.article-title:hover {
40     color: #428bca;
41     text-decoration: none;
42 }
43
44 .article-content {
45     white-space: pre-line;
```

Scikit-learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python.

It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use.

The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

NumPy: Base n-dimensional array package

SciPy: Fundamental library for scientific computing

Matplotlib: Comprehensive 2D/3D plotting

I Python: Enhanced interactive console.
Symbolic mathematics

Pandas: Data structures and analysis

Extensions or modules for SciPy are conventionally named [Sci Kits](#). As such, the module provides learning algorithms and is named scikit-learn.

The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

Although the interface is Python, c-libraries are leveraged for performance such as num for arrays and matrix operations, [LAPACK](#), [Lib SVM](#) and the careful use of python.

Features

The library is focused on modeling data. It is not focused on loading, manipulating and summarizing data. For these features, refer to NumPy and Pandas.

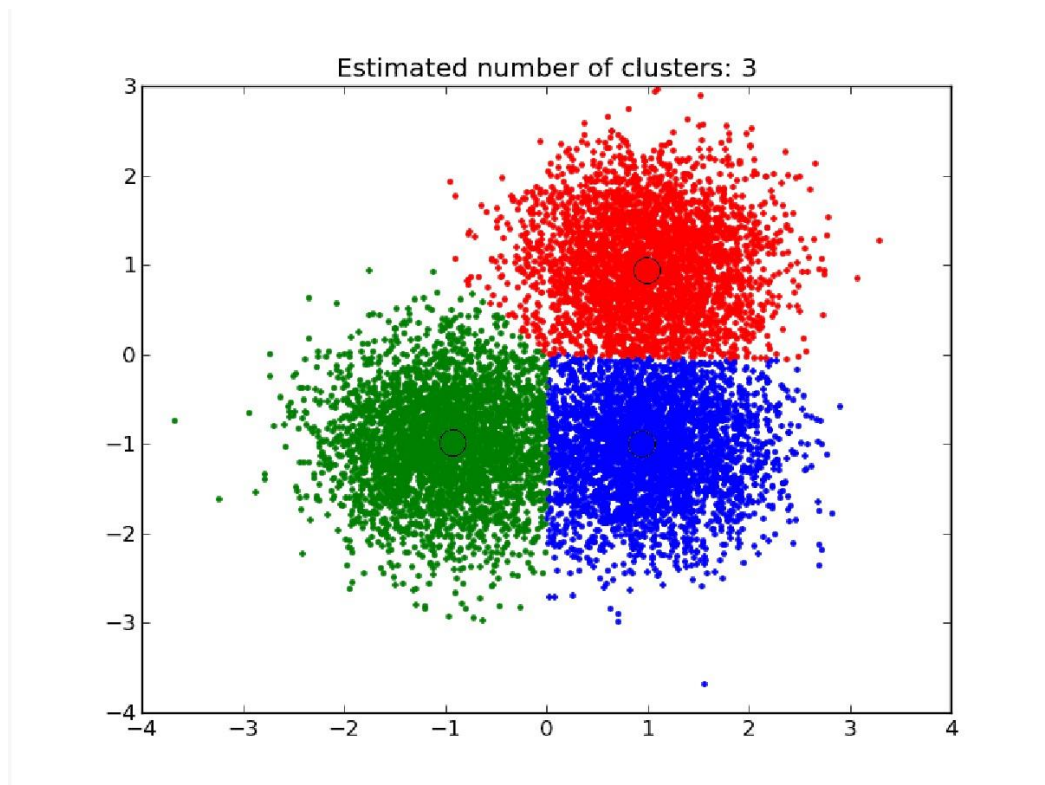


figure 3.9

Some popular groups of models provided by scikit-learn include:

Clustering: for grouping unlabeled data such as K Means.

Cross Validation: for estimating the performance of supervised models on unseen data.

Datasets: for test datasets and for generating datasets with specific properties

for investigating model behavior.

Dimensionality Reduction: for reducing the number of attributes in data for

summarization, visualization and feature selection such as Principal component analysis.

Ensemble methods: for combining the predictions of multiple supervised models.

Feature extraction: for defining attributes in image and text data.

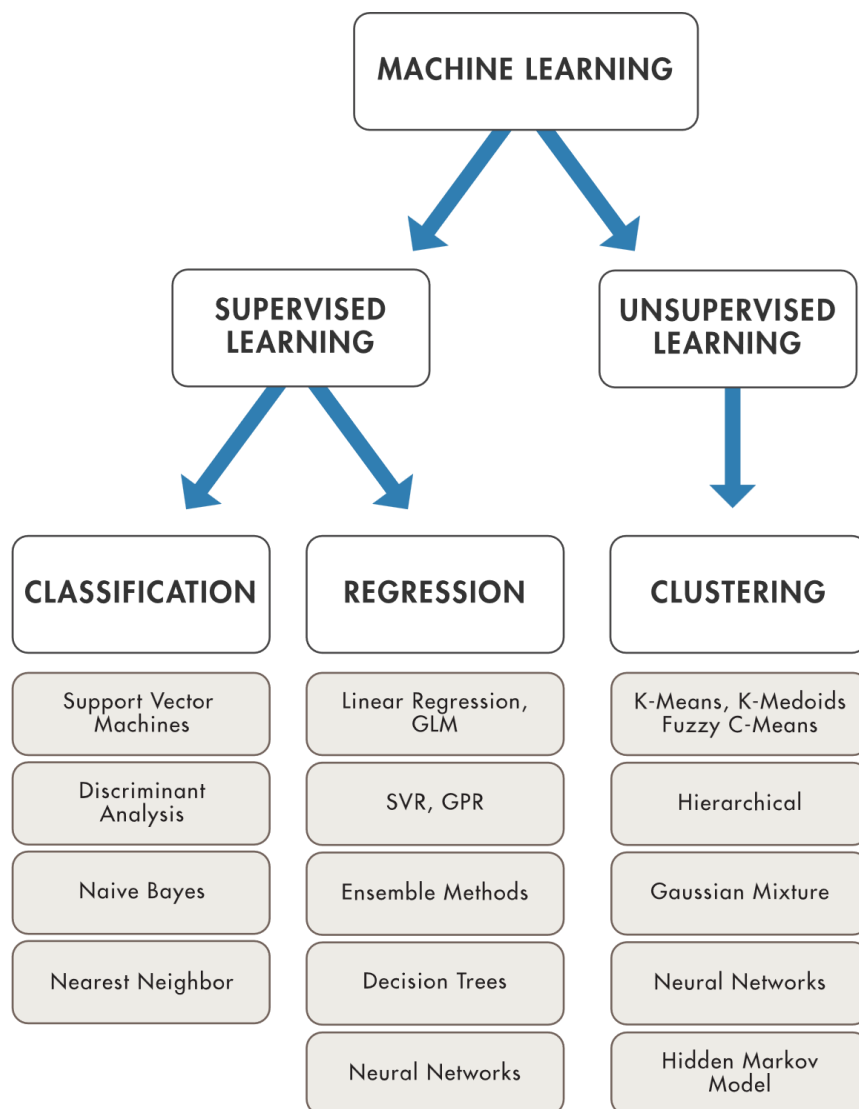
Feature selection: for identifying meaningful attributes from which to.

Parameter Tuning: for getting the most out of supervised models.

Manifold Learning: For summarizing and depicting complex multi-dimensional data.

Supervised Models: a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

Figure 3.10





Machine learning

Machine learning is a branch of artificial intelligence (AI).

Computer science which focuses on the use of data and algorithms.

Imitate the way that humans learn, gradually improving its accuracy.

How machine learning works?

Lets breaks down the learning system of a machine learning algorithm into three main parts.

A Decision Process: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabeled, your algorithm will produce an estimate about a pattern in the data.

An Error Function: An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.

An Model Optimization Process: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

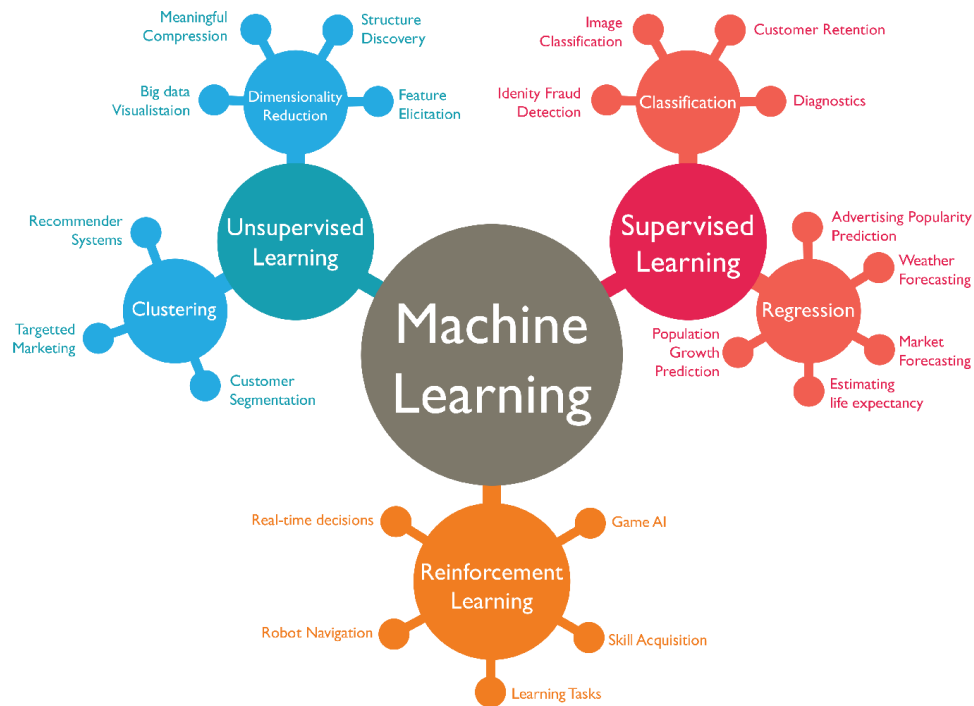


Figure 3.11

Machine learning classifiers fall into three primary categories:

Supervised machine learning

Supervised Learning, also known as supervised machine learning, is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately. This occurs as part of the cross validation process to ensure that the model avoids overfitting or underfitting. Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Some methods used in supervised learning include neural networks, naïve bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and more.

Unsupervised machine learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, image and pattern recognition. It's also used to reduce the number of features in a model through the process of dimensionality reduction; principal component analysis (PCA) and singular value decomposition (SVD) are two common approaches for this. Other algorithms used in unsupervised learning include neural networks, k-means clustering, probabilistic clustering methods, and more.

Semi-supervised learning

Semi-supervised learning offers a happy medium between supervised and unsupervised learning. During training, it uses a smaller labeled data set to guide classification and feature extraction from a larger, unlabeled data set. Semi-supervised learning can solve the problem of having not enough labeled data (or not being able to afford to label enough data) to train a supervised learning algorithm.

Reinforcement machine learning

Reinforcement machine learning is a behavioral machine learning model that is similar to supervised learning, but the algorithm isn't trained using sample data. This model learns as it goes by using trial and error. A sequence of successful outcomes will be reinforced to develop the best recommendation or policy.

Sigmoid function

We use the sigmoid function as the underlying function in Logistic regression. Mathematically and graphically, it is shown as:

Sigmoid Function

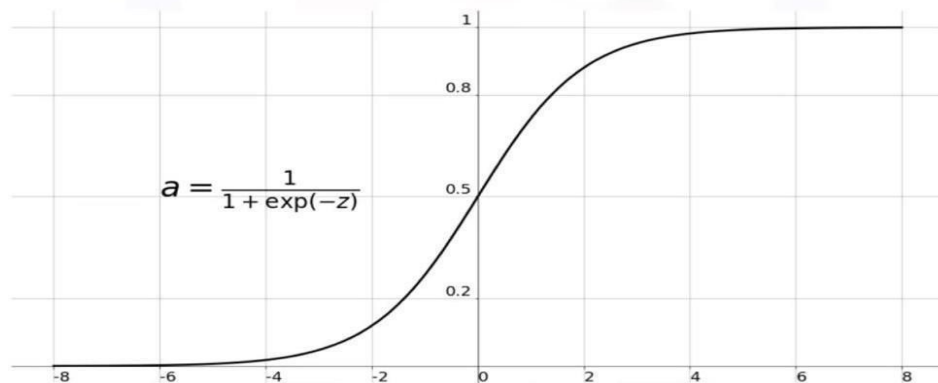


Figure 4.1

Why do we use the Sigmoid Function?

The sigmoid function's range is bounded between 0 and 1. Thus it's useful in calculating the probability for the Logistic function.

- 1) It's derivative is easy to calculate than other functions which is useful during gradient descent calculation.
- 2) It is a simple way of introducing non-linearity to the model.

Although there are other functions as well, which can be used, but sigmoid is the most common function used for logistic regression. We will talk about the rest of the functions in the neural network section.

The logistic function is given by:

$$p(X) = \frac{\exp(\beta_0 + \beta_1 X)}{1 + \exp(\beta_0 + \beta_1 X)}$$

Figure 4.2

The cost function for the whole training set is given as :

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))]$$

The values of parameters (θ) for which the cost function is minimum is calculated using the gradient descent (as discussed in the Linear Regression section) algorithm. The partial derivative for cost function is given as :

$$\begin{aligned} \frac{\partial(J(\theta))}{\partial(\theta_j)} &= -\frac{1}{m} * (\sum_{i=1}^m [y^{(i)} * (1 - h_{\theta}(x^{(i)})) * x_j^i - (1 - y^{(i)}) * h_{\theta}(x^{(i)}) * x_j^i]) \\ \frac{\partial(J(\theta))}{\partial(\theta_j)} &= -\frac{1}{m} * (\sum_{i=1}^m [y^{(i)} - y^{(i)} * h_{\theta}(x^{(i)}) - h_{\theta}(x^{(i)}) + y^{(i)} * h_{\theta}(x^{(i)})] * x_j^i) \\ \frac{\partial(J(\theta))}{\partial(\theta_j)} &= -\frac{1}{m} * (\sum_{i=1}^m [y^{(i)} - h_{\theta}(x^{(i)})] * x_j^i) \end{aligned}$$

Figure 4.3

Multiple Logistic Function

We can generalise the simple logistic function for multiple features as:

$$\hat{p} = \frac{\exp(b_0 + b_1 X_1 + b_2 X_2 + \dots + b_p X_p)}{1 + \exp(b_0 + b_1 X_1 + b_2 X_2 + \dots + b_p X_p)}$$

And the logit function can be written as:

$$\ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$$

The coefficients are calculated the same we did for simple logistic function, by passing the above equation in the cost function.

Just like we did in multilinear regression, we will check for correlation between different features for Multi logistic as well.

Multinomial Logistics Regression(Number of Labels >2)

Many times, there are classification problems where the number of classes is greater than 2. We can extend Logistic regression for multi-class classification. The logic is simple; we train our logistic model for each class and calculate the probability($h(\theta x)$) that a specific feature belongs to that class. Once we have trained the model for all the classes, we predict a new value's class by choosing that class for which the probability($h(\theta x)$) is maximum. Although we have libraries that we can use to perform multinomial logistic regression, we rarely use logistic regression for classification problems where the number of classes is more than 2.

There are many other classification models for such scenarios. We will see more of that in the coming lectures.

Learning Algorithm

The learning algorithm is how we search the set of possible hypotheses (hypothesis space H) for the best parameterization (in this case the weight vector w). This search is an optimization problem looking for the hypothesis that optimizes an error measure.

There is no sophisticated, closed-form solution like least-squares linear, so we will use gradient descent instead. Specifically we will use batch gradient descent which calculates the gradient from all data points in the data set.

Luckily, our "cross-entropy" error measure is convex so there is only one minimum. Thus the minimum we arrive at is the global minimum.

To learn we're going to minimize the following error measure using batch gradient descent.

$$e(h(\mathbf{x}_n), y_n) = -\ln(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n})$$

$$E(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^N e(h(\mathbf{x}_n), y_n) = \frac{1}{N} \sum_{n=1}^N -\ln(1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n})$$

$\text{Recall} = \frac{50}{50+200} = 0.2$ (The model was able to recall only 20% of the cancer patients)

Precision

Precision is a measure of amongst all the positive predictions, how many of them were actually positive. Mathematically,

$$\text{Precision} = \frac{TP}{TP+FP}$$

Let's suppose in the previous example, the model identified 50 people as cancer patients (TP) but also raised a false alarm for 100 patients (FP).

Hence,

$$\text{Precision} = \frac{50}{50+100} = 0.33$$
 (The model only has a precision of 33%)

F1 Score

From the previous examples, it is clear that we need a metric that considers both Precision and Recall for evaluating a model. One such metric is the F1 score.

F1 score is defined as the harmonic mean of Precision and Recall.

The mathematical formula is: F1

$$\text{score} = 2 * \left(\frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \right)$$

Specificity or True Negative Rate

This represents how specific is the model while predicting the True Negatives.

Mathematically, $\text{Specificity} = \frac{TN}{TN+FP}$ Or, it can be said that it quantifies the

total number of negatives predicted by the model with respect to the

total number of actual negative or non favorable outcomes.

ROC(Receiver Operator Characteristic)

We know that the classification algorithms work on the concept of probability of occurrence of the possible outcomes. A probability value lies between 0 and 1. Zero means that there is no probability of occurrence and one means that the occurrence is certain.

But while working with real-time data, it has been observed that we seldom get a perfect 0 or 1 value. Instead of that, we get different decimal values lying between 0 and 1. Now the question is if we are not getting binary probability values how are we actually determining the class in our classification problem?

There comes the concept of Threshold. A threshold is set, any probability value below the threshold is a negative outcome, and anything more than the threshold is a favourable or the positive outcome. For Example, if the threshold is 0.5, any probability value below 0.5 means a negative or an unfavourable outcome and any value above 0.5 indicates a positive or favourable outcome.

Now, the question is, what should

be an ideal threshold? A typical

ROC curve looks like the following

figure.



Figure 4.4

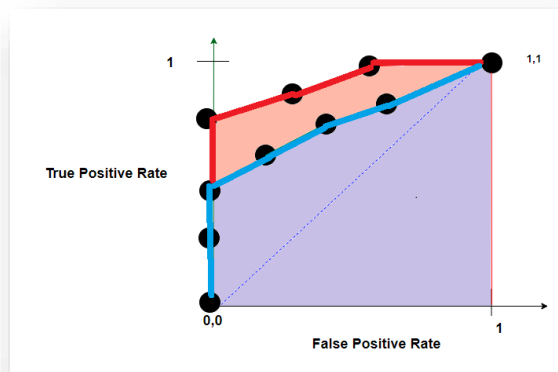
- > Mathematically, it represents the various confusion matrices for various thresholds.
- > Each black dot is one confusion matrix.
- > The green dotted line represents the scenario when the true positive rate equals the false positive rate.
- > As evident from the curve, as we move from the rightmost dot towards left, after a certain threshold, the false positive rate decreases.
- > After some time, the false positive rate becomes zero.
- > But that is not a rule of thumb. Based on the requirement, we need to select the point of a threshold.
- > The ROC curve answers our question of which threshold to choose.

But we have a confusion!!

Let's suppose that we used different classification algorithms, and different ROCs for the corresponding algorithms have been plotted. The question is: which algorithm to choose now? The answer is to calculate the area under each ROC curve.

AUC(Area Under Curve)

Figure 4.5



- > It helps us to choose the best model amongst the models for which we have plotted the ROC curve.
- > The best model is the one which encompasses the maximum area under it.
- > In the adjacent diagram, amongst the two curves, the model that resulted in the red one should be chosen as it clearly covers more area than the blue one

What is the significance of Roc curve and AUC?

In real life, we create various models using different algorithms that we can use for classification purpose. We use AUC to determine which model is the best one to use for a given dataset. Suppose we have created Logistic regression, SVM as well as a clustering model for classification purpose. We will calculate AUC for all the models separately. The model with highest AUC value will be the best model to use.

Advantages of Logistic Regression

- > It is very simple and easy to implement.
- > The output is more informative than other classification algorithms.
- > It expresses the relationship between independent and dependent variables.
- > Very effective with linearly separable data.

Disadvantages of Logistic Regression

- > Not effective with data which are not linearly separable.
- > Not as powerful as other classification models.
- > Multiclass classifications are much easier to do with .

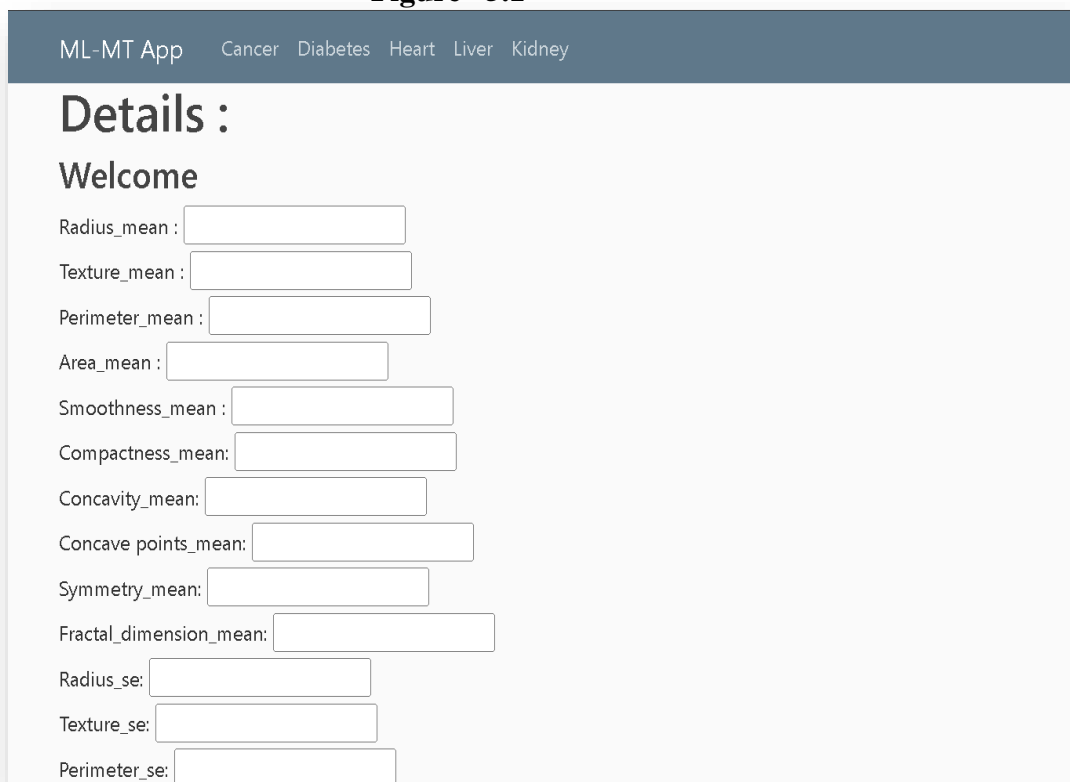
Chapter-5

THIS IS THE FIRST PAGE YOU WILL SEE WHEN YOU RUN THE PROJECT

1. This page will help you to check if you have cancer or not using amachine learning model after you fill the details accordingly.

The page is mentioned

Figure 5.1



The screenshot displays the 'ML-MT App' interface. At the top, there is a navigation bar with the app name and several category tabs: 'Cancer', 'Diabetes', 'Heart', 'Liver', and 'Kidney'. Below the navigation bar, the main heading is 'Details :'. Underneath this, there is a 'Welcome' message. The form contains two sections of input fields. The first section lists ten parameters with their respective mean values, each followed by a text input box: 'Radius_mean', 'Texture_mean', 'Perimeter_mean', 'Area_mean', 'Smoothness_mean', 'Compactness_mean', 'Concavity_mean', 'Concave points_mean', 'Symmetry_mean', and 'Fractal_dimension_mean'. The second section lists three parameters with their respective standard error values, each followed by a text input box: 'Radius_se', 'Texture_se', and 'Perimeter_se'.

ML-MT App Cancer Diabetes Heart Liver Kidney

Details :

Welcome

Radius_mean :

Texture_mean :

Perimeter_mean :

Area_mean :

Smoothness_mean :

Compactness_mean:

Concavity_mean:

Concave points_mean:

Symmetry_mean:

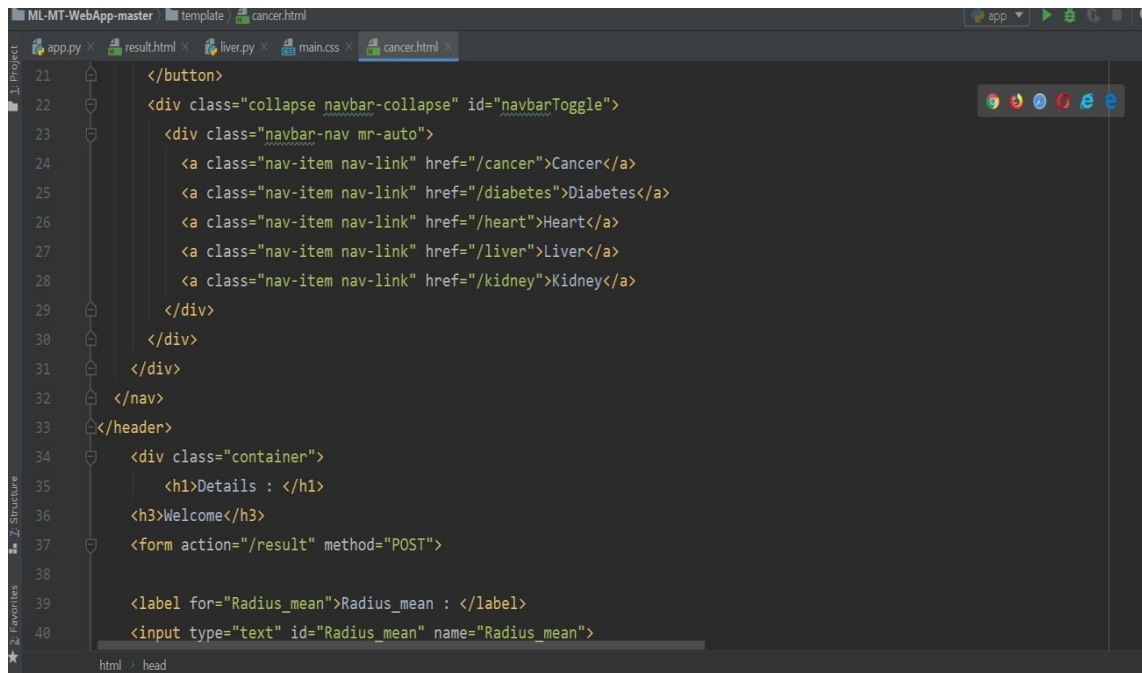
Fractal_dimension_mean:

Radius_se:

Texture_se:

Perimeter_se:

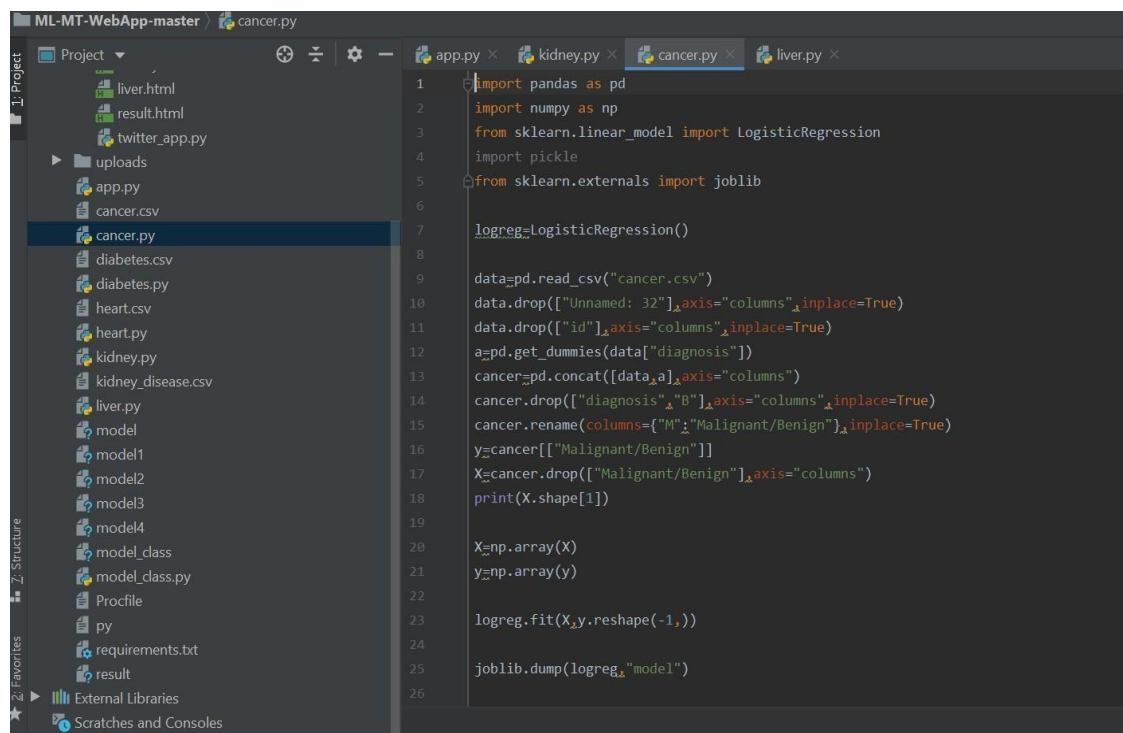
**THE ABOVE PAGE IS BEEN MADE WITH THE HELP OF
HTML LANGUAGE.THE CODE FOR THAT PAGE IS
MENTIONED BELOW.**



```
21 </button>
22 <div class="collapse navbar-collapse" id="navbarToggle">
23   <div class="navbar-nav mr-auto">
24     <a class="nav-item nav-link" href="/cancer">Cancer</a>
25     <a class="nav-item nav-link" href="/diabetes">Diabetes</a>
26     <a class="nav-item nav-link" href="/heart">Heart</a>
27     <a class="nav-item nav-link" href="/liver">Liver</a>
28     <a class="nav-item nav-link" href="/kidney">Kidney</a>
29   </div>
30 </div>
31 </div>
32 </nav>
33 </header>
34 <div class="container">
35   <h1>Details : </h1>
36   <h3>Welcome</h3>
37   <form action="/result" method="POST">
38
39     <label for="Radius_mean">Radius_mean : </label>
40     <input type="text" id="Radius_mean" name="Radius_mean">
```

Figure 5.2

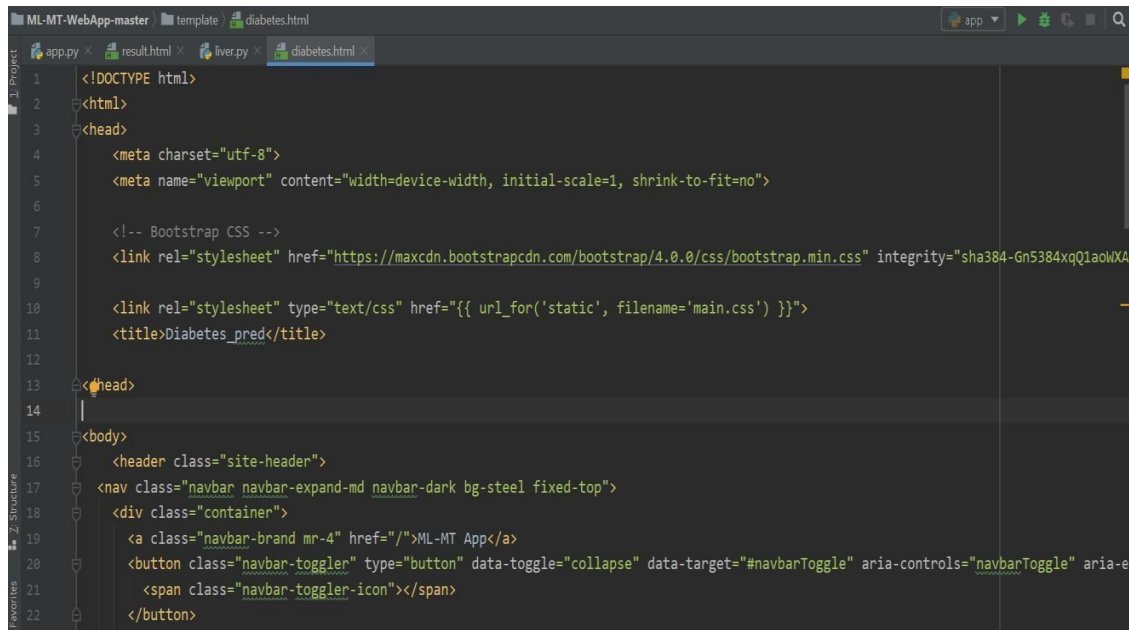
NOW YOU KNOW WHICH DATASET WAS USED TO PREDICT THE OUTPUT. NOW LET'S SEE THE PYTHON CODE I USED TO BUILT MY MACHINE LEARNING MODEL. FOR THIS I USED PYCHAR IDE (INTIGRATED DEVELOPMENT ENVIRONMENT). IN BELOW CODE I USED LOGISTIC REGRESSION MODEL AND DID SOME DATA CLEANING AND HANDLING CATEGORICAL FEATURES USING ONE HOT ENCODING BEFORE GIVING DATA TO THE MODEL SO THAT THE ACCURACY OF THE MODEL CAN BE MAXIMUM DURING TRAINING. THEN I SPLITTED THE DATA INTO X AND Y AND FIT THESE INTO MODEL AND AT THE END SAVED THE MODEL FILE FOR FURTHER PREDICTION ON USER DEIFINED DATASET.



```
1 import pandas as pd
2 import numpy as np
3 from sklearn.linear_model import LogisticRegression
4 import pickle
5 from sklearn.externals import joblib
6
7 logreg=LogisticRegression()
8
9 data=pd.read_csv("cancer.csv")
10 data.drop(["Unnamed: 32"],axis="columns",inplace=True)
11 data.drop(["id"],axis="columns",inplace=True)
12 a=pd.get_dummies(data["diagnosis"])
13 cancer=pd.concat([data,a],axis="columns")
14 cancer.drop(["diagnosis","B"],axis="columns",inplace=True)
15 cancer.rename(columns={"M":"Malignant/Benign"},axis="columns",inplace=True)
16 y=cancer[["Malignant/Benign"]]
17 X=cancer.drop(["Malignant/Benign"],axis="columns")
18 print(X.shape[1])
19
20 X=np.array(X)
21 y=np.array(y)
22
23 logreg.fit(X,y.reshape(-1,))
24
25 joblib.dump(logreg,"model")
26
```

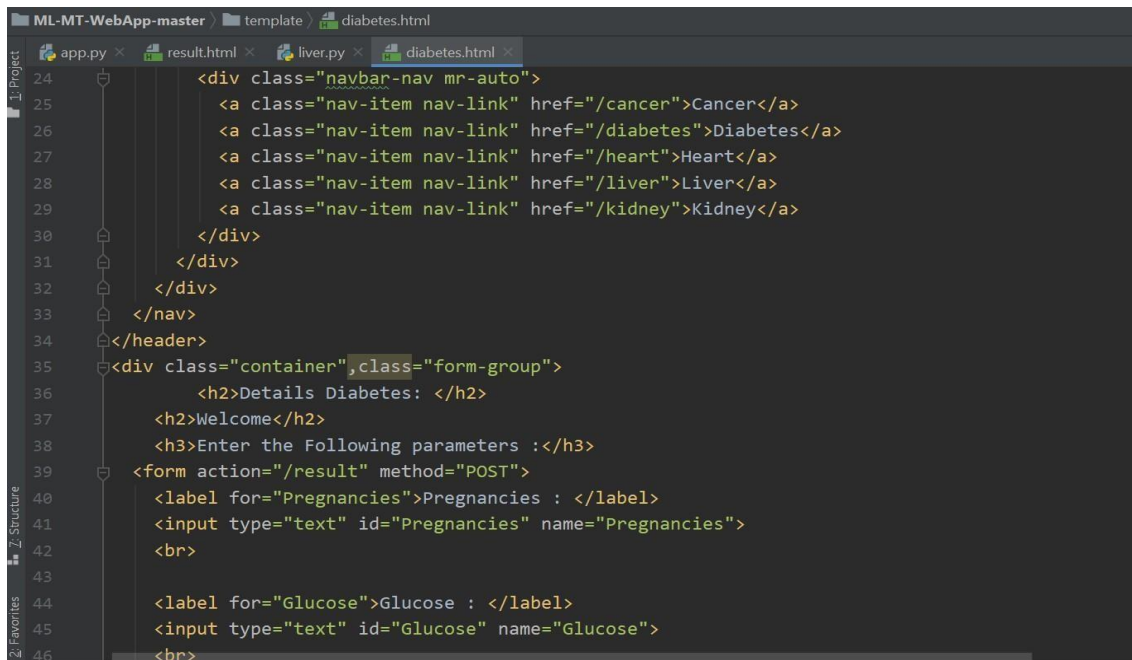
Figure 5.3

THE ABOVE PAGE IS BEEN MADE WITH THE HELP OF
HTML LANGUAGE.THE CODE FOR THAT PAGE IS
MENTIONED BELOW.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
6
7   <!-- Bootstrap CSS -->
8   <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA4
9
10   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='main.css') }}">
11   <title>Diabetes_pred</title>
12
13 </head>
14
15 <body>
16   <header class="site-header">
17     <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
18       <div class="container">
19         <a class="navbar-brand mr-4" href="/">ML-MT App</a>
20         <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle" aria-controls="navbarToggle" aria-e
21         <span class="navbar-toggler-icon"></span>
22       </button>
```

Figure 5.4



```
24     <div class="navbar-nav mr-auto">
25       <a class="nav-item nav-link" href="/cancer">Cancer</a>
26       <a class="nav-item nav-link" href="/diabetes">Diabetes</a>
27       <a class="nav-item nav-link" href="/heart">Heart</a>
28       <a class="nav-item nav-link" href="/liver">Liver</a>
29       <a class="nav-item nav-link" href="/kidney">Kidney</a>
30     </div>
31   </div>
32 </div>
33 </nav>
34 </header>
35 <div class="container",class="form-group">
36   <h2>Details Diabetes: </h2>
37   <h2>Welcome</h2>
38   <h3>Enter the Following parameters :</h3>
39   <form action="/result" method="POST">
40     <label for="Pregnancies">Pregnancies : </label>
41     <input type="text" id="Pregnancies" name="Pregnancies">
42     <br>
43
44     <label for="Glucose">Glucose : </label>
45     <input type="text" id="Glucose" name="Glucose">
46     <br>
```

Figure 5.5

```
ML-MT-WebApp-master / template / diabetes.html
app.py x result.html x liver.py x diabetes.html x
48 <label for="BloodPressure">BloodPressure : </label>
49 <input type="text" id="BloodPressure" name="BloodPressure">
50 <br>
51
52 <label for="SkinThickness">SkinThickness : </label>
53 <input type="text" id="SkinThickness" name="SkinThickness">
54 <br>
55
56 <label for="Insulin">Insulin : </label>
57 <input type="text" id="Insulin" name="Insulin">
58 <br>
59
60 <label for="BMI">BMI : </label>
61 <input type="text" id="BMI" name="BMI">
62 <br>
63
64 <label for="DiabetesPedigreeFunction">DiabetesPedigreeFunction : </label>
65 <input type="text" id="DiabetesPedigreeFunction" name="DiabetesPedigreeFunction">
66 <br>
67
68 <label for="Age">Age : </label>
69 <input type="text" id="Age" name="Age">
70 <br>
```

Figure 5.6

THE DATASET THAT I USED TO TRAIN MY MACHINE LEARNING MODEL WAS IN .CSV FORMAT THEREFORE I USED PANDAS TO READ MY DATA TO FIT THAT INTO MODEL. HERE I USED JUPYTER NOTEBOOK TO SHOW THE DATA BELOW.

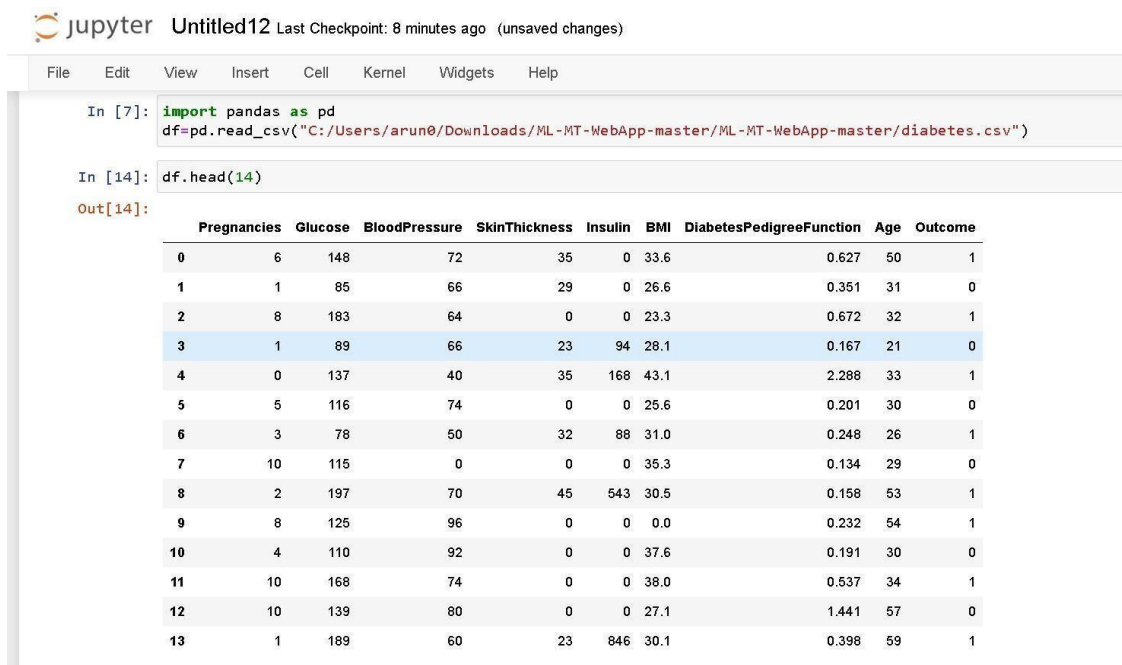


Figure 5.7

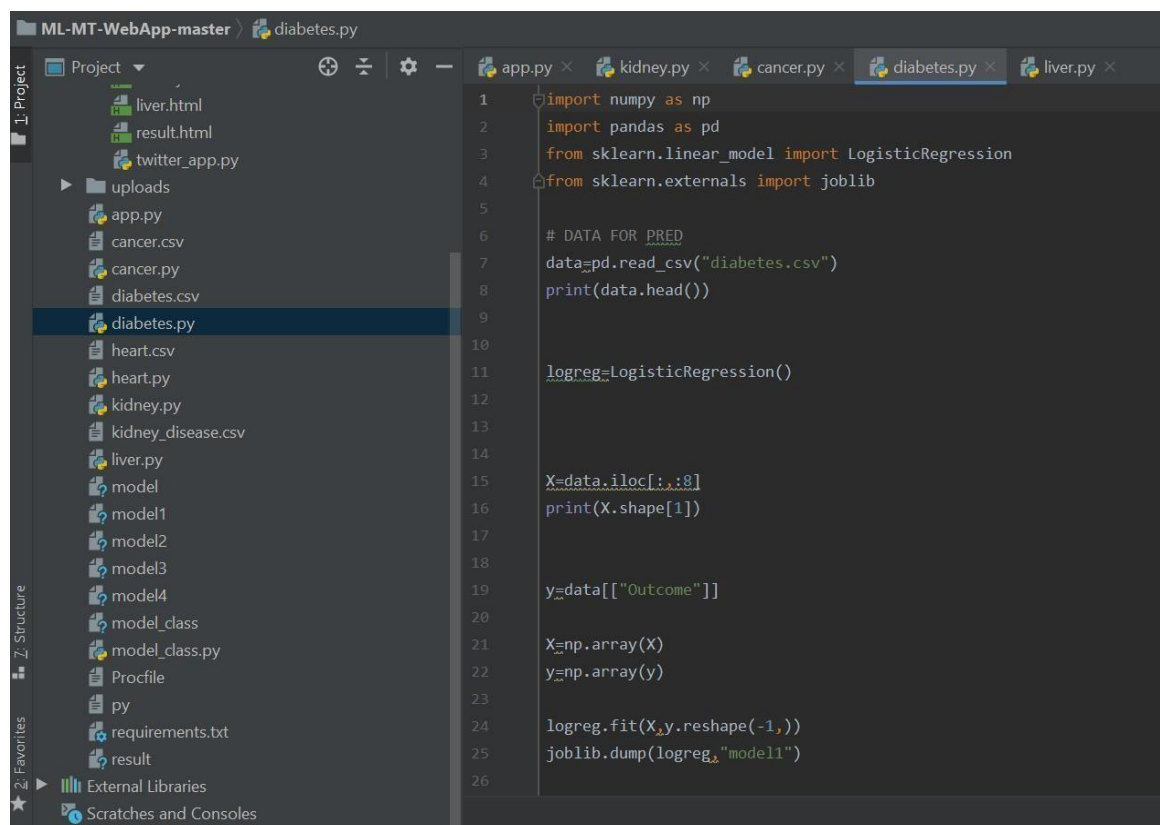
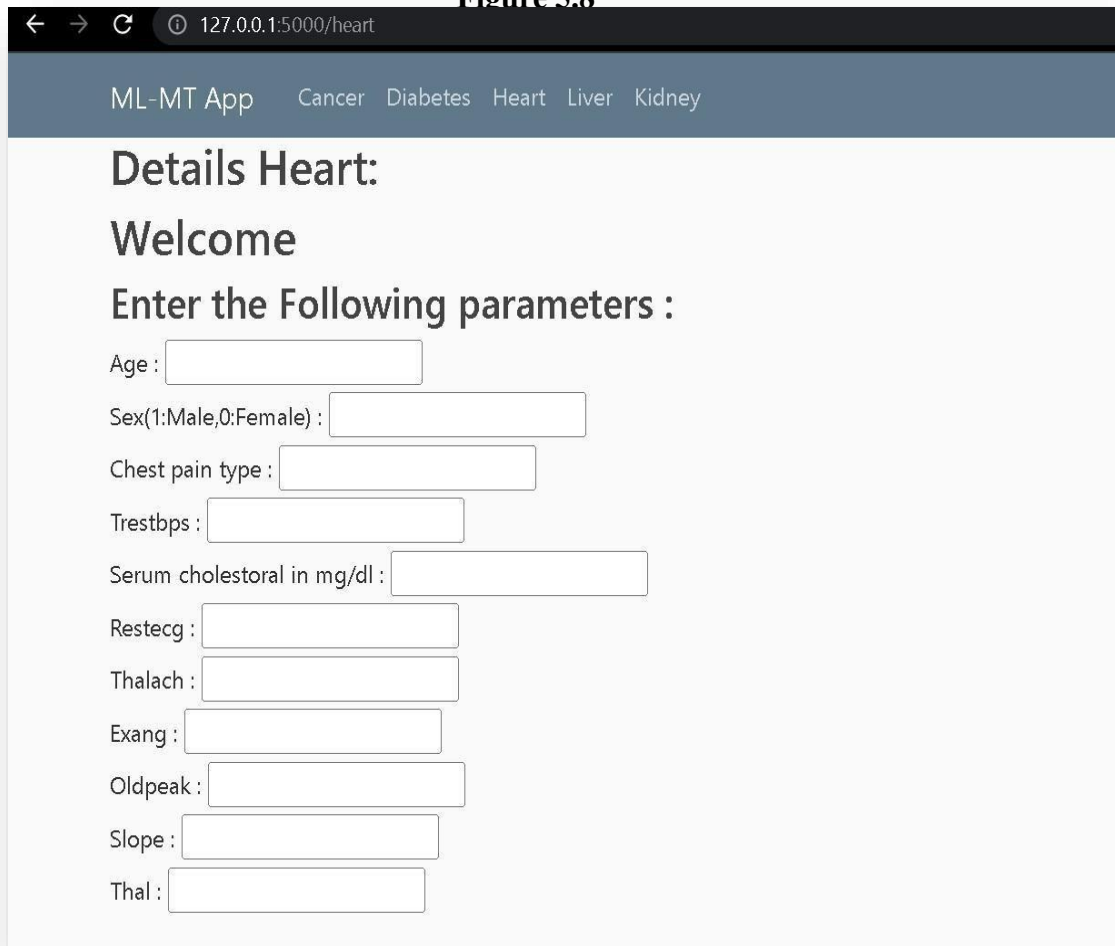


Figure 5.7

3. The next option available is to check if you have diabetes or not using the Machine Learning model, after filling the detailed required for analysis in the page.

This is the page you will see below:

Figure 5.8



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/heart". The page title is "ML-MT App" and the navigation bar includes links for "Cancer", "Diabetes", "Heart", "Liver", and "Kidney". The main content area is titled "Details Heart:" and "Welcome". It prompts the user to "Enter the Following parameters :" and lists ten input fields for the following parameters: Age, Sex (1:Male,0:Female), Chest pain type, Trestbps, Serum cholestoral in mg/dl, Restecg, Thalach, Exang, Oldpeak, Slope, and Thal.

← → ↻ ⓘ 127.0.0.1:5000/heart

ML-MT App Cancer Diabetes Heart Liver Kidney

Details Heart:

Welcome

Enter the Following parameters :

Age :

Sex(1:Male,0:Female) :

Chest pain type :

Trestbps :

Serum cholestoral in mg/dl :

Restecg :

Thalach :

Exang :

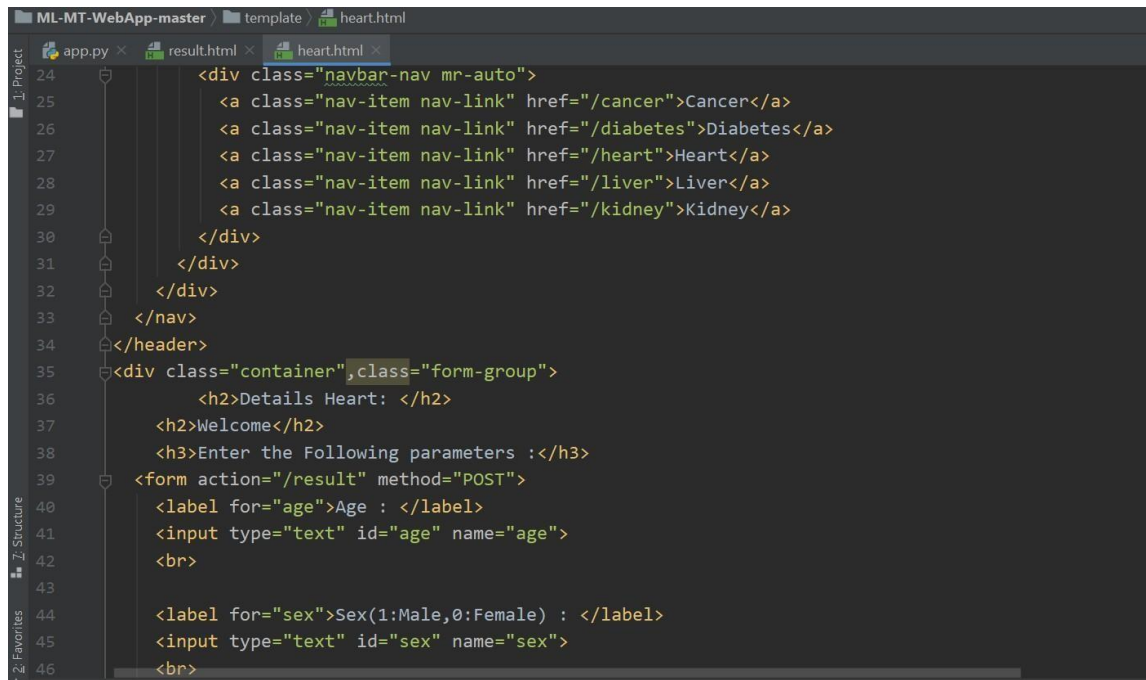
Oldpeak :

Slope :

Thal :

THE ABOVE PAGE IS BEEN MADE WITH THE HELP OF HTML LANGUAGE. THE CODE FOR THAT PAGE IS MENTIONED BELOW.

figure 5.8



```
24 <div class="navbar-nav mr-auto">
25   <a class="nav-item nav-link" href="/cancer">Cancer</a>
26   <a class="nav-item nav-link" href="/diabetes">Diabetes</a>
27   <a class="nav-item nav-link" href="/heart">Heart</a>
28   <a class="nav-item nav-link" href="/liver">Liver</a>
29   <a class="nav-item nav-link" href="/kidney">Kidney</a>
30 </div>
31 </div>
32 </div>
33 </nav>
34 </header>
35 <div class="container", class="form-group">
36   <h2>Details Heart: </h2>
37   <h2>Welcome</h2>
38   <h3>Enter the Following parameters :</h3>
39   <form action="/result" method="POST">
40     <label for="age">Age : </label>
41     <input type="text" id="age" name="age">
42     <br>
43
44     <label for="sex">Sex(1:Male,0:Female) : </label>
45     <input type="text" id="sex" name="sex">
46     <br>
```

```
ML-MT-WebApp-master > template > heart.html
app.py x result.html x heart.html x
48 <label for="chest pain type">Chest pain type : </label>
49 <input type="text" id="chest pain type" name="chest pain type">
50 <br>
51
52 <label for="trestbps">Trestbps : </label>
53 <input type="text" id="trestbps" name="trestbps">
54 <br>
55
56 <label for="serum_cholesterol in mg/dl">Serum_cholesterol in mg/dl : </label>
57 <input type="text" id="serum_cholesterol in mg/dl" name="serum_cholesterol in mg/dl">
58 <br>
59
60 <label for="restecg">Restecg : </label>
61 <input type="text" id="restecg" name="restecg">
62 <br>
63
64 <label for="thalach">Thalach : </label>
65 <input type="text" id="thalach" name="thalach">
66 <br>
67
68 <label for="exang">Exang : </label>
69 <input type="text" id="exang" name="exang">
70 <br>
```

Figure 5.9

jupyter Untitled12 Last Checkpoint: 10 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

```
In [17]: import pandas as pd
df=pd.read_csv("C:/Users/arun0/Downloads/ML-MT-WebApp-master/ML-MT-WebApp-master/heart.csv")

In [18]: df.head(14)

Out[18]:
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
5	57	1	0	140	192	0	1	148	0	0.4	1	0	1	1
6	56	0	1	140	294	0	0	153	0	1.3	1	0	2	1
7	44	1	1	120	263	0	1	173	0	0.0	2	0	3	1
8	52	1	2	172	199	1	1	162	0	0.5	2	0	3	1
9	57	1	2	150	168	0	1	174	0	1.6	2	0	2	1
10	54	1	0	140	239	0	1	160	0	1.2	2	0	2	1
11	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1
12	49	1	1	130	266	0	1	171	0	0.6	2	0	2	1
13	64	1	3	110	211	0	0	144	1	1.8	1	0	2	1

Figure 5.10

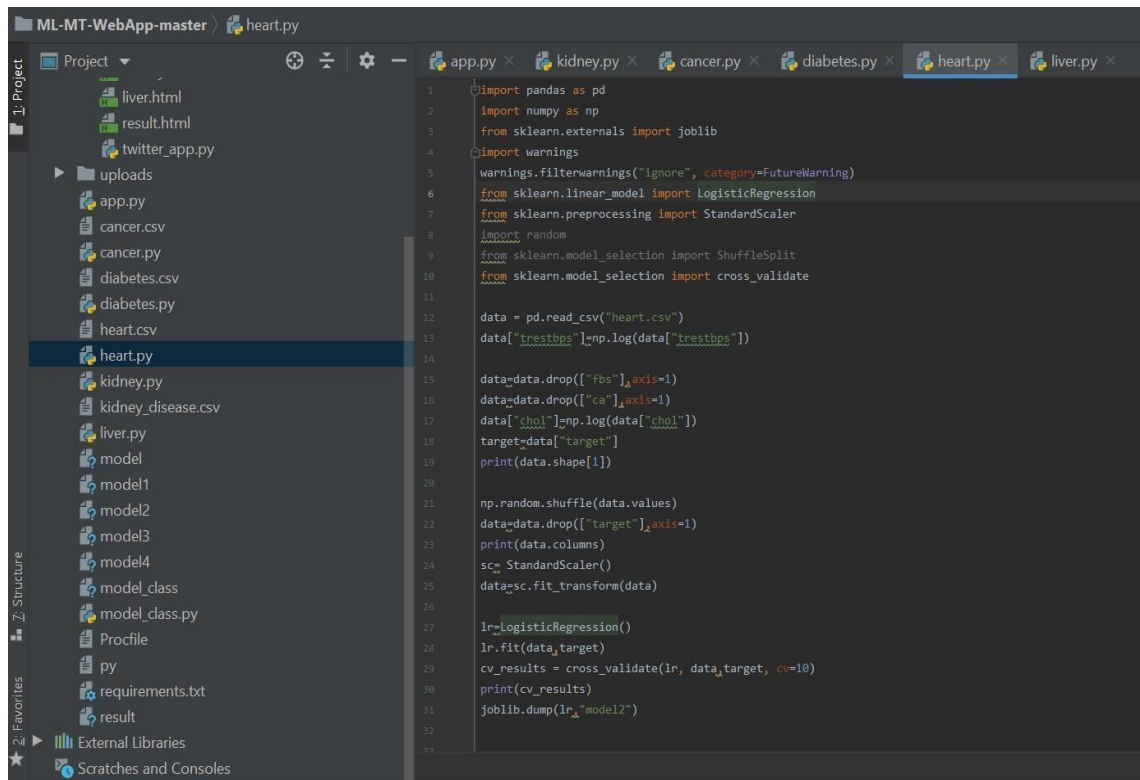
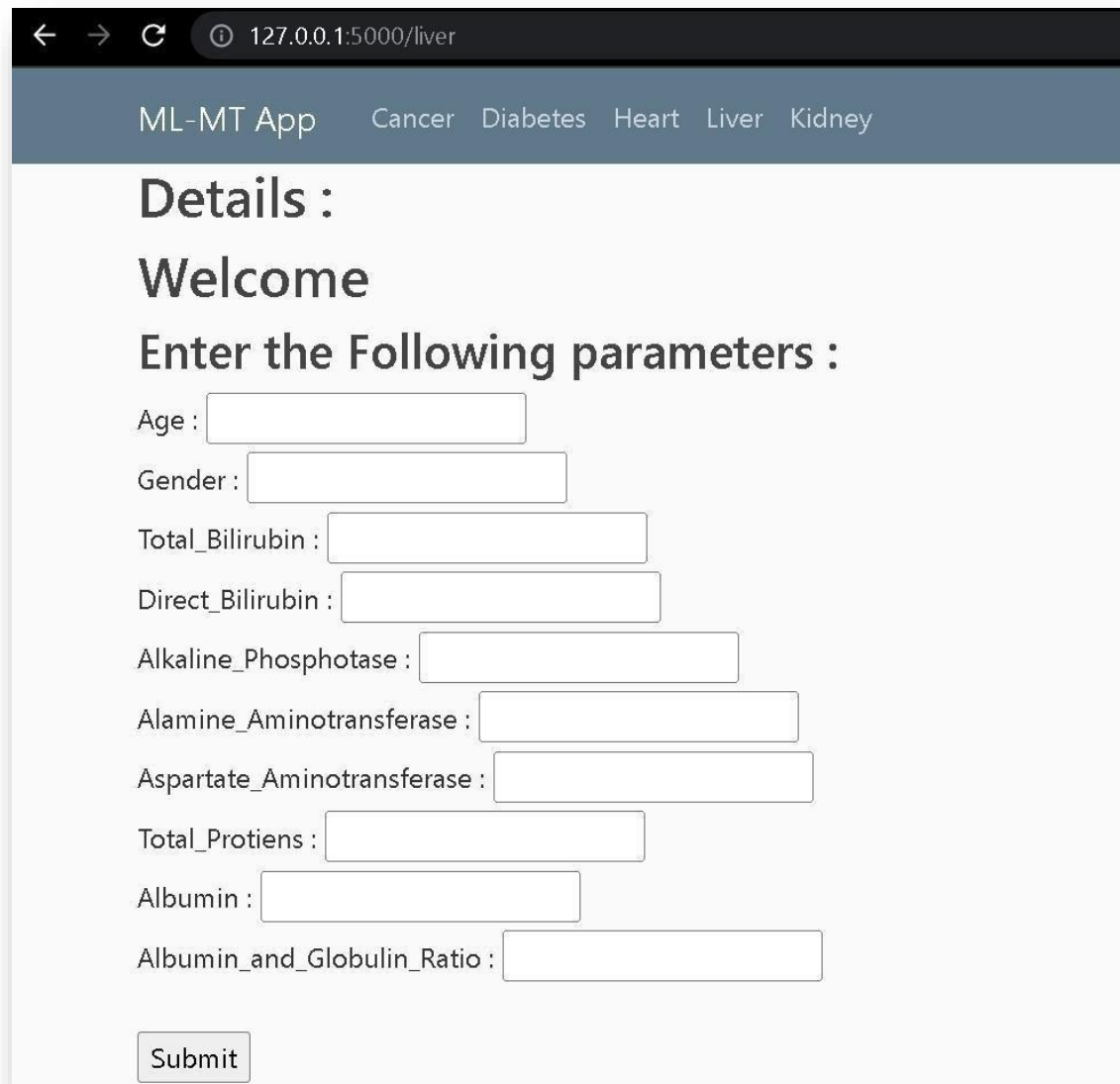


Figure 5.10

4. The next option available is to check if you have diabetes or not using the Machine Learning model, after filling the detailed required for analysis in the page.

This is the page you will see below: figure.5.11



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/liver'. The page has a dark blue header with the text 'ML-MT App' and navigation links for 'Cancer', 'Diabetes', 'Heart', 'Liver', and 'Kidney'. The main content area is white and contains the following text and form elements:

Details :

Welcome

Enter the Following parameters :

Age :

Gender :

Total_Bilirubin :

Direct_Bilirubin :

Alkaline_Phosphotase :

Alamine_Aminotransferase :

Aspartate_Aminotransferase :

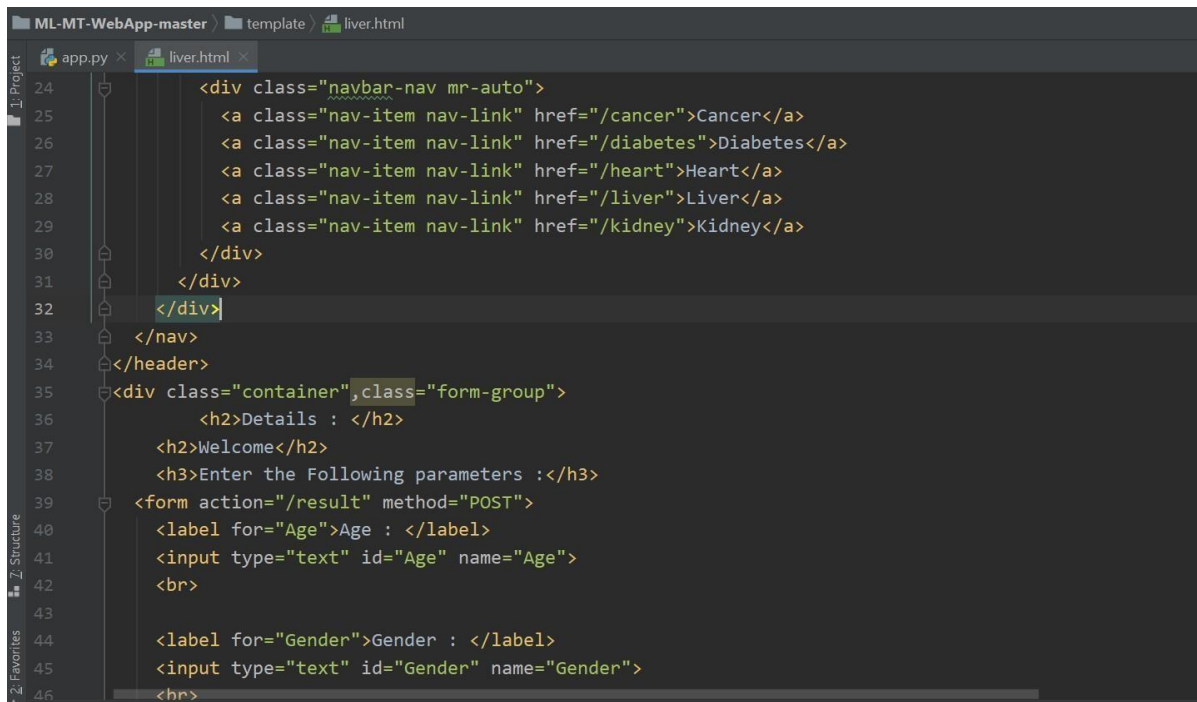
Total_Protiens :

Albumin :

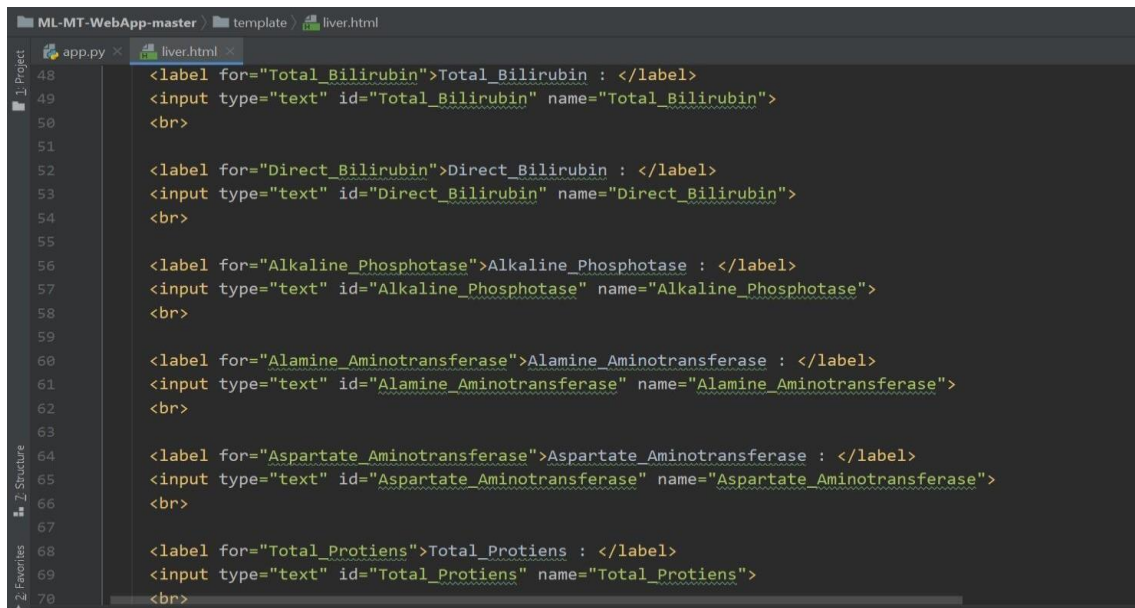
Albumin_and_Globulin_Ratio :

THE ABOVE PAGE IS BEEN MADE WITH THE HELP OF
HTML LANGUAGE.THE CODE FOR THAT PAGE IS
MENTIONED BELOW.

Figure 5.12



```
24 <div class="navbar-nav mr-auto">
25   <a class="nav-item nav-link" href="/cancer">Cancer</a>
26   <a class="nav-item nav-link" href="/diabetes">Diabetes</a>
27   <a class="nav-item nav-link" href="/heart">Heart</a>
28   <a class="nav-item nav-link" href="/liver">Liver</a>
29   <a class="nav-item nav-link" href="/kidney">Kidney</a>
30 </div>
31 </div>
32 </div>
33 </nav>
34 </header>
35 <div class="container", class="form-group">
36   <h2>Details : </h2>
37   <h2>Welcome</h2>
38   <h3>Enter the Following parameters :</h3>
39   <form action="/result" method="POST">
40     <label for="Age">Age : </label>
41     <input type="text" id="Age" name="Age">
42     <br>
43
44     <label for="Gender">Gender : </label>
45     <input type="text" id="Gender" name="Gender">
46     <hr>
```

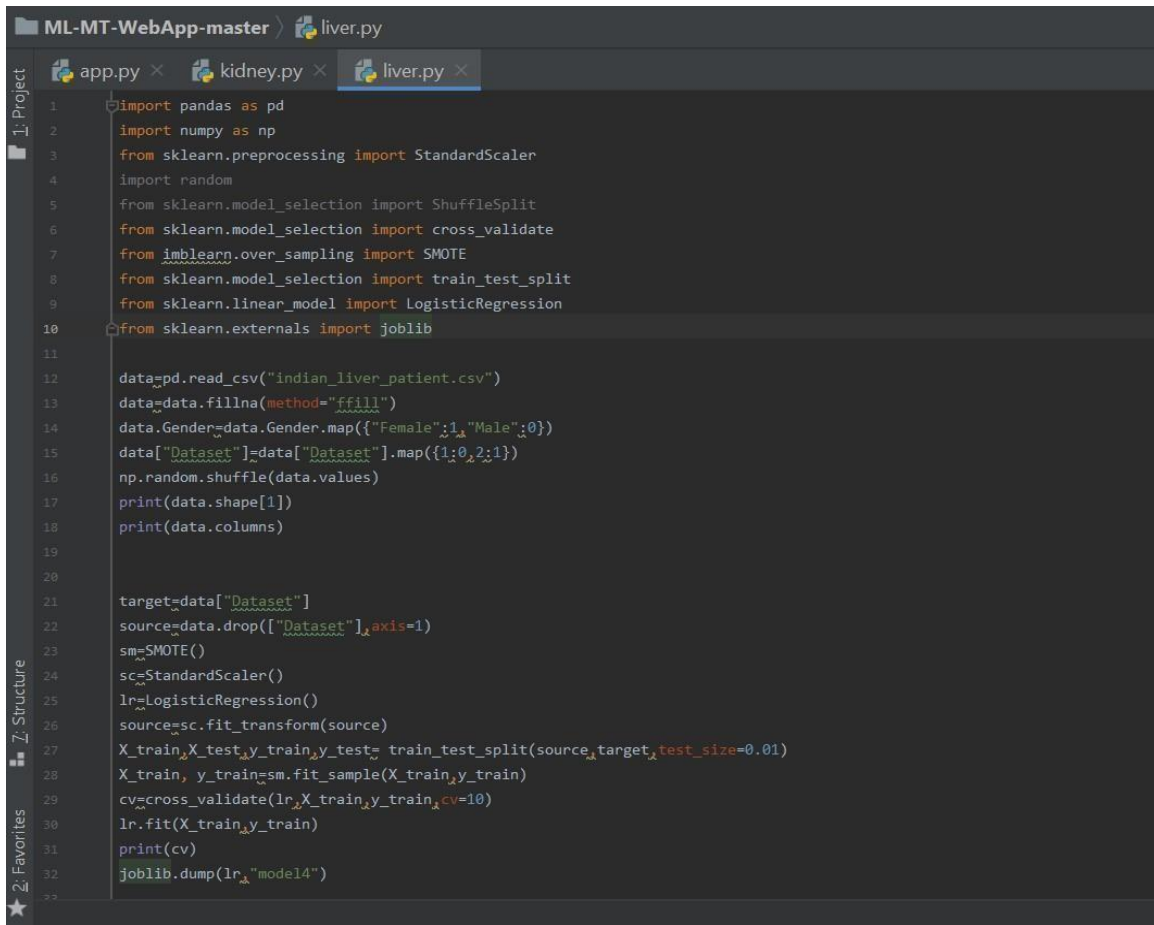
A screenshot of a code editor window titled 'ML-MT-WebApp-master' with a sub-folder 'template' and a file 'liver.html'. The editor shows HTML code for a form with five input fields. The left sidebar shows a project structure with 'app.py' and 'liver.html' files. The code is as follows:

```
48 <label for="Total_Bilirubin">Total_Bilirubin : </label>
49 <input type="text" id="Total_Bilirubin" name="Total_Bilirubin">
50 <br>
51
52 <label for="Direct_Bilirubin">Direct_Bilirubin : </label>
53 <input type="text" id="Direct_Bilirubin" name="Direct_Bilirubin">
54 <br>
55
56 <label for="Alkaline_Phosphotase">Alkaline_Phosphotase : </label>
57 <input type="text" id="Alkaline_Phosphotase" name="Alkaline_Phosphotase">
58 <br>
59
60 <label for="Alamine_Aminotransferase">Alamine_Aminotransferase : </label>
61 <input type="text" id="Alamine_Aminotransferase" name="Alamine_Aminotransferase">
62 <br>
63
64 <label for="Aspartate_Aminotransferase">Aspartate_Aminotransferase : </label>
65 <input type="text" id="Aspartate_Aminotransferase" name="Aspartate_Aminotransferase">
66 <br>
67
68 <label for="Total_Protiens">Total_Protiens : </label>
69 <input type="text" id="Total_Protiens" name="Total_Protiens">
70 <br>
```

Figure 5.13

THE DATASET THAT I USED TO TRAIN MY MACHINE LEARNING MODEL WAS IN .CSV FORMAT THEREFORE I USED PANDAS TO READ MY DATA TO FIT THAT INTO MODEL. HERE I USED JUPYTER NOTEBOOK TO SHOW THE DATA BELOW.

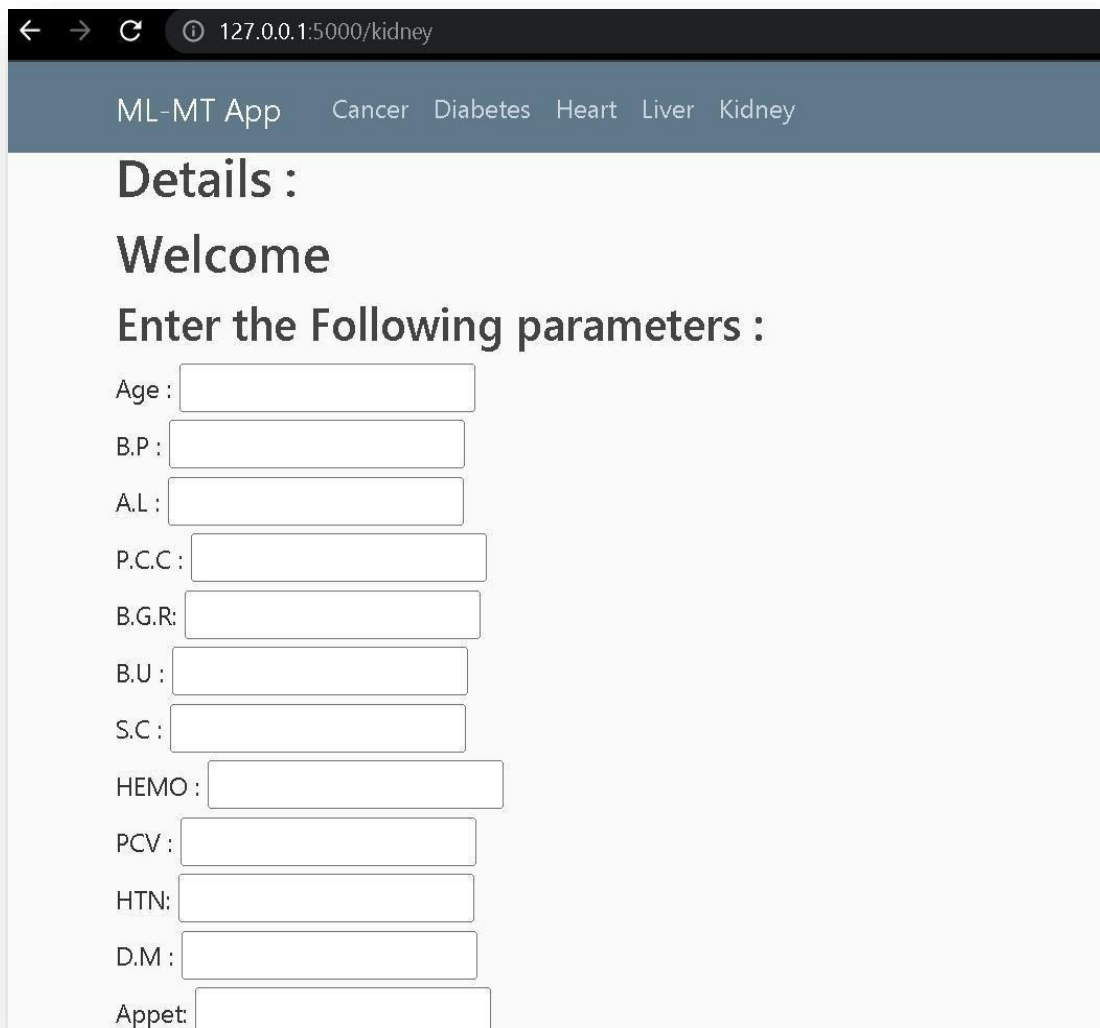
figure 5.14



```
1 import pandas as pd
2 import numpy as np
3 from sklearn.preprocessing import StandardScaler
4 import random
5 from sklearn.model_selection import ShuffleSplit
6 from sklearn.model_selection import cross_validate
7 from imblearn.over_sampling import SMOTE
8 from sklearn.model_selection import train_test_split
9 from sklearn.linear_model import LogisticRegression
10 from sklearn.externals import joblib
11
12 data=pd.read_csv("indian_liver_patient.csv")
13 data=data.fillna(method="ffill")
14 data.Gender=data.Gender.map({"Female":1,"Male":0})
15 data["Dataset"]=data["Dataset"].map({1:0,2:1})
16 np.random.shuffle(data.values)
17 print(data.shape[1])
18 print(data.columns)
19
20
21 target=data["Dataset"]
22 source=data.drop(["Dataset"],axis=1)
23 sm=SMOTE()
24 sc=StandardScaler()
25 lr=LogisticRegression()
26 source=sc.fit_transform(source)
27 X_train,X_test,y_train,y_test=train_test_split(source,target,test_size=0.01)
28 X_train,y_train=sm.fit_sample(X_train,y_train)
29 cv=cross_validate(lr,X_train,y_train,cv=10)
30 lr.fit(X_train,y_train)
31 print(cv)
32 joblib.dump(lr,"model4")
```


5. The last option available is to check if you have diabetes or not using the Machine Learning model, after filling the detailed required for analysis in the page.

This is the page you will see below:



The screenshot shows a web browser window with the address bar displaying "127.0.0.1:5000/kidney". The page has a dark blue header with the text "ML-MT App" and navigation links for "Cancer", "Diabetes", "Heart", "Liver", and "Kidney". The main content area is white and contains the following text:

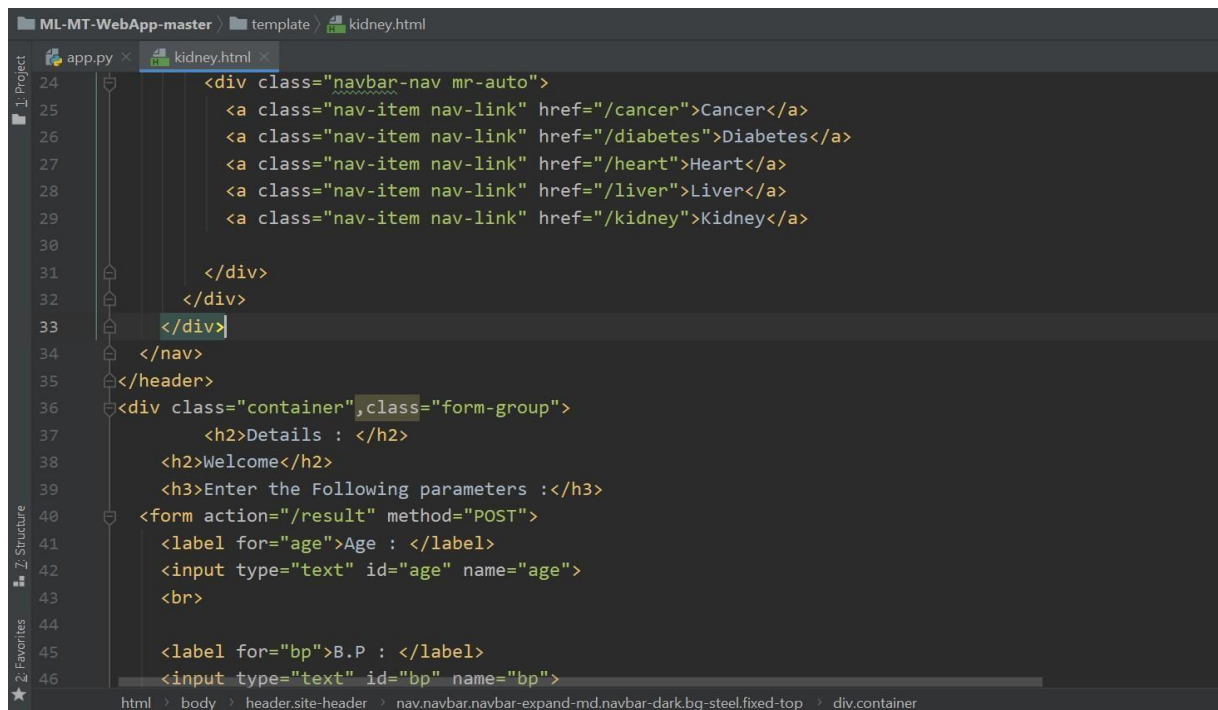
Details :
Welcome
Enter the Following parameters :

Below the text, there is a list of parameters, each followed by a text input field:

- Age :
- B.P :
- A.L :
- P.C.C :
- B.G.R:
- B.U :
- S.C :
- HEMO :
- PCV :
- HTN:
- D.M :
- Appet:

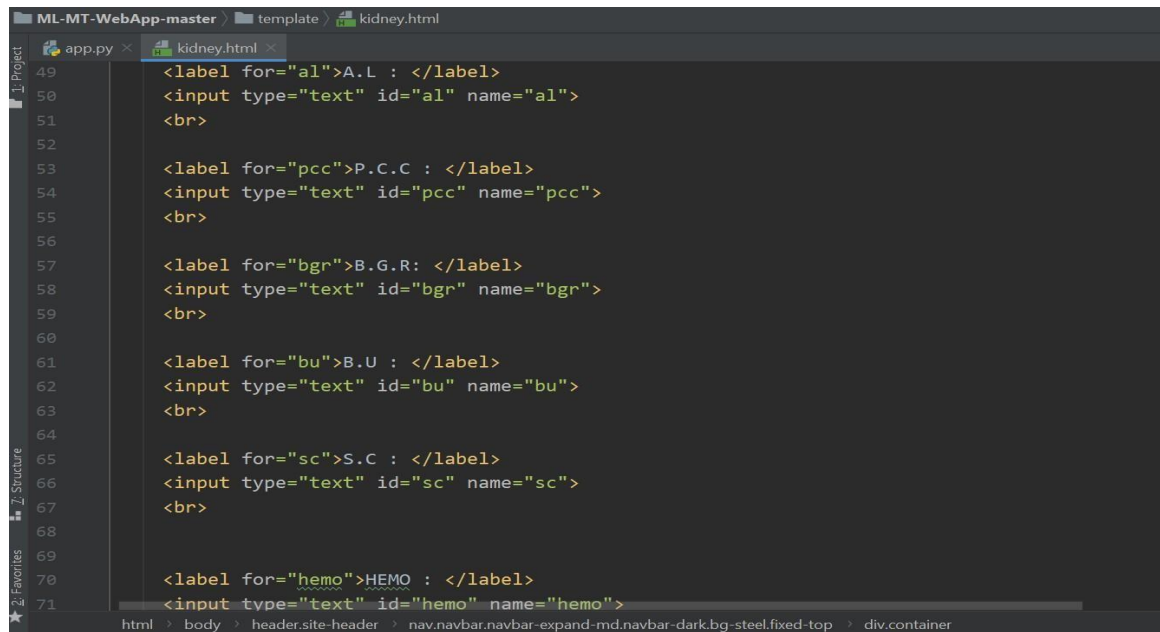
THE ABOVE PAGE IS BEEN MADE WITH THE HELP OF HTML LANGUAGE.THE CODE FOR THAT PAGE IS MENTIONED BELOW.

Figure 5.15



```
24 <div class="navbar-nav mr-auto">
25   <a class="nav-item nav-link" href="/cancer">Cancer</a>
26   <a class="nav-item nav-link" href="/diabetes">Diabetes</a>
27   <a class="nav-item nav-link" href="/heart">Heart</a>
28   <a class="nav-item nav-link" href="/liver">Liver</a>
29   <a class="nav-item nav-link" href="/kidney">Kidney</a>
30
31 </div>
32 </div>
33 </div>
34 </nav>
35 </header>
36 <div class="container",class="form-group">
37   <h2>Details : </h2>
38   <h2>Welcome</h2>
39   <h3>Enter the Following parameters :</h3>
40   <form action="/result" method="POST">
41     <label for="age">Age : </label>
42     <input type="text" id="age" name="age">
43     <br>
44
45     <label for="bp">B.P : </label>
46     <input type="text" id="bp" name="bp">
```

html > body > header.site-header > nav.navbar.navbar-expand-md.navbar-dark.bg-steel.fixed-top > div.container

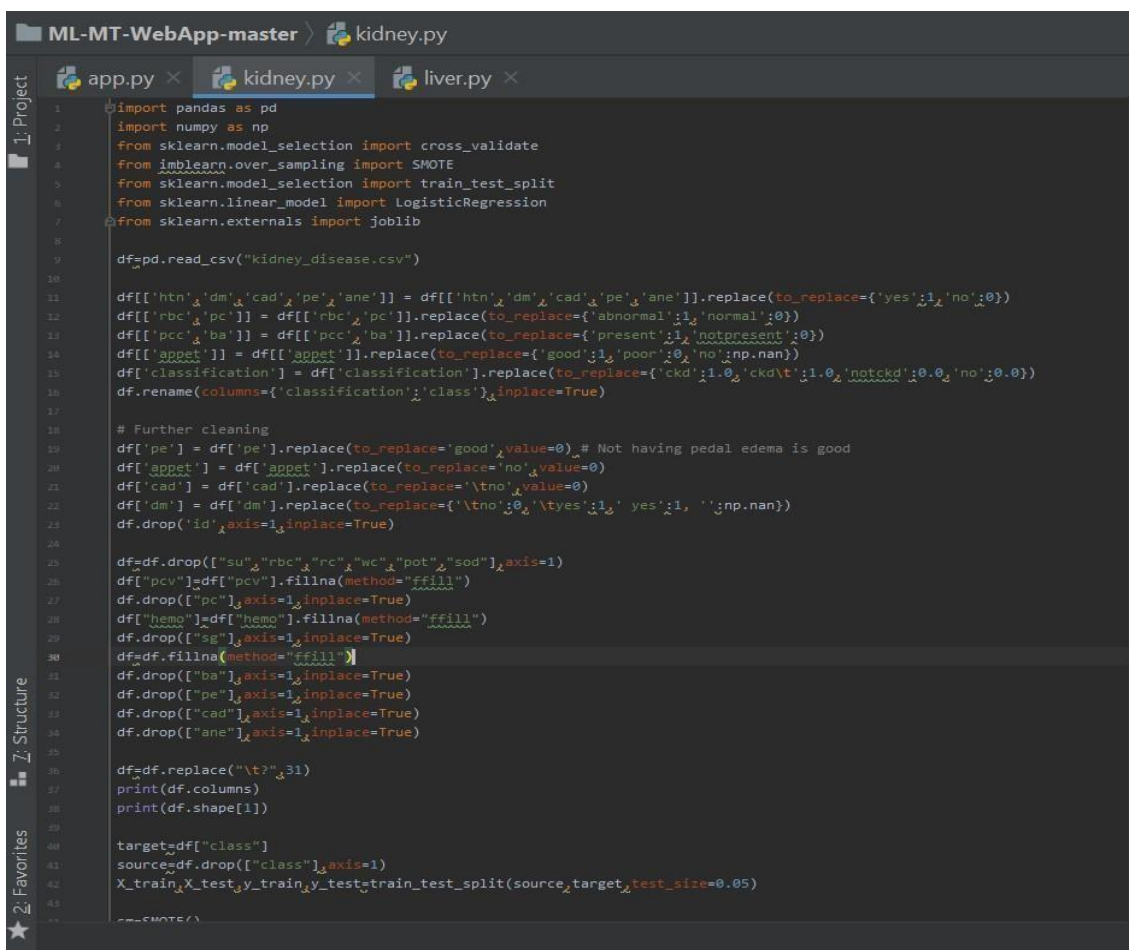


```
ML-MT-WebApp-master > template > kidney.html
app.py x kidney.html x
49 <label for="al">A.L : </label>
50 <input type="text" id="al" name="al">
51 <br>
52
53 <label for="pcc">P.C.C : </label>
54 <input type="text" id="pcc" name="pcc">
55 <br>
56
57 <label for="bgr">B.G.R: </label>
58 <input type="text" id="bgr" name="bgr">
59 <br>
60
61 <label for="bu">B.U : </label>
62 <input type="text" id="bu" name="bu">
63 <br>
64
65 <label for="sc">S.C : </label>
66 <input type="text" id="sc" name="sc">
67 <br>
68
69
70 <label for="hemo">HEMO : </label>
71 <input type="text" id="hemo" name="hemo">
html > body > header.site-header > nav.navbar.navbar-expand-md.navbar-dark.bg-steel.fixed-top > div.container
```

Figure 5.16

THE DATASET THAT I USED TO TRAIN MY MACHINE LEARNING MODEL WAS IN .CSV FORMAT THEREFORE I USED PANDAS TO READ MY DATA TO FIT THAT INTO MODEL. HERE I USED JUPYTER NOTEBOOK TO SHOW THE DATA BELOW.

NOW YOU KNOW WHICH DATASET WAS USED TO PREDICT THE OUTPUT. NOW LET'S SEE THE PYTHON CODE I USED TO BUILT MY MACHINE LEARNING MODEL. FOR THIS I USED PYCHAR IDE (INTIGRATED DEVELOPMENT ENVIRONMENT). IN BELOW CODE I USED LOGISTIC REGRESSION MODEL AND DID SOME DATA CLEANING AND HANDLING CATEGORICAL FEATURES USING ONE HOT ENCODING BEFORE GIVING DATA TO THE MODEL SO THAT THE ACCURACY OF THE MODEL CAN BE MAXIMUM DURING TRAINING. THEN I SPLITTED THE DATA INTO X AND Y AND FIT THESE INTO MODEL AND AT THE END SAVED THE MODEL FILE USING SKLEARN JOBLIB LIBRARY FOR FURTHER PREDICTION ON USER DEIFINED DATASET.



```

1  import pandas as pd
2  import numpy as np
3  from sklearn.model_selection import cross_validate
4  from imblearn.over_sampling import SMOTE
5  from sklearn.model_selection import train_test_split
6  from sklearn.linear_model import LogisticRegression
7  from sklearn.externals import joblib
8
9  df=pd.read_csv("kidney_disease.csv")
10
11  df[["htn","dm","cad","pe","ane"]] = df[["htn","dm","cad","pe","ane"]].replace(to_replace={'yes':1,'no':0})
12  df[["rbc","pc"]] = df[["rbc","pc"]].replace(to_replace={'abnormal':1,'normal':0})
13  df[["pcc","ba"]] = df[["pcc","ba"]].replace(to_replace={'present':1,'notpresent':0})
14  df[["appet"]] = df[["appet"]].replace(to_replace={'good':1,'poor':0,'no':np.nan})
15  df[["classification"]] = df[["classification"]].replace(to_replace={'ckd':1.0,'ckd\':1.0,'notckd':0.0,'no':0.0})
16  df.rename(columns={'classification':'class'},inplace=True)
17
18  # Further cleaning
19  df['pe'] = df['pe'].replace(to_replace='good',value=0)# Not having pedal edema is good
20  df['appet'] = df['appet'].replace(to_replace='no',value=0)
21  df['cad'] = df['cad'].replace(to_replace='tno',value=0)
22  df['dm'] = df['dm'].replace(to_replace={'\tno':0,'\tyes':1,' yes':1, '' :np.nan})
23  df.drop('id',axis=1,inplace=True)
24
25  df=df.drop(["su","rbc","rc","wc","pot","sod"],axis=1)
26  df["pcv"]=df["pcv"].fillna(method="ffill")
27  df.drop(["pc"],axis=1,inplace=True)
28  df["hemo"]=df["hemo"].fillna(method="ffill")
29  df.drop(["sg"],axis=1,inplace=True)
30  df=df.fillna(method="ffill")
31  df.drop(["ba"],axis=1,inplace=True)
32  df.drop(["pe"],axis=1,inplace=True)
33  df.drop(["cad"],axis=1,inplace=True)
34  df.drop(["ane"],axis=1,inplace=True)
35
36  df=df.replace("\t?",' ')
37  print(df.columns)
38  print(df.shape[1])
39
40  targets=df[["class"]]
41  source=df.drop(["class"],axis=1)
42  X_train,X_test,y_train,y_test=train_test_split(source,target,test_size=0.05)
43
44  sm=SMOTE()

```

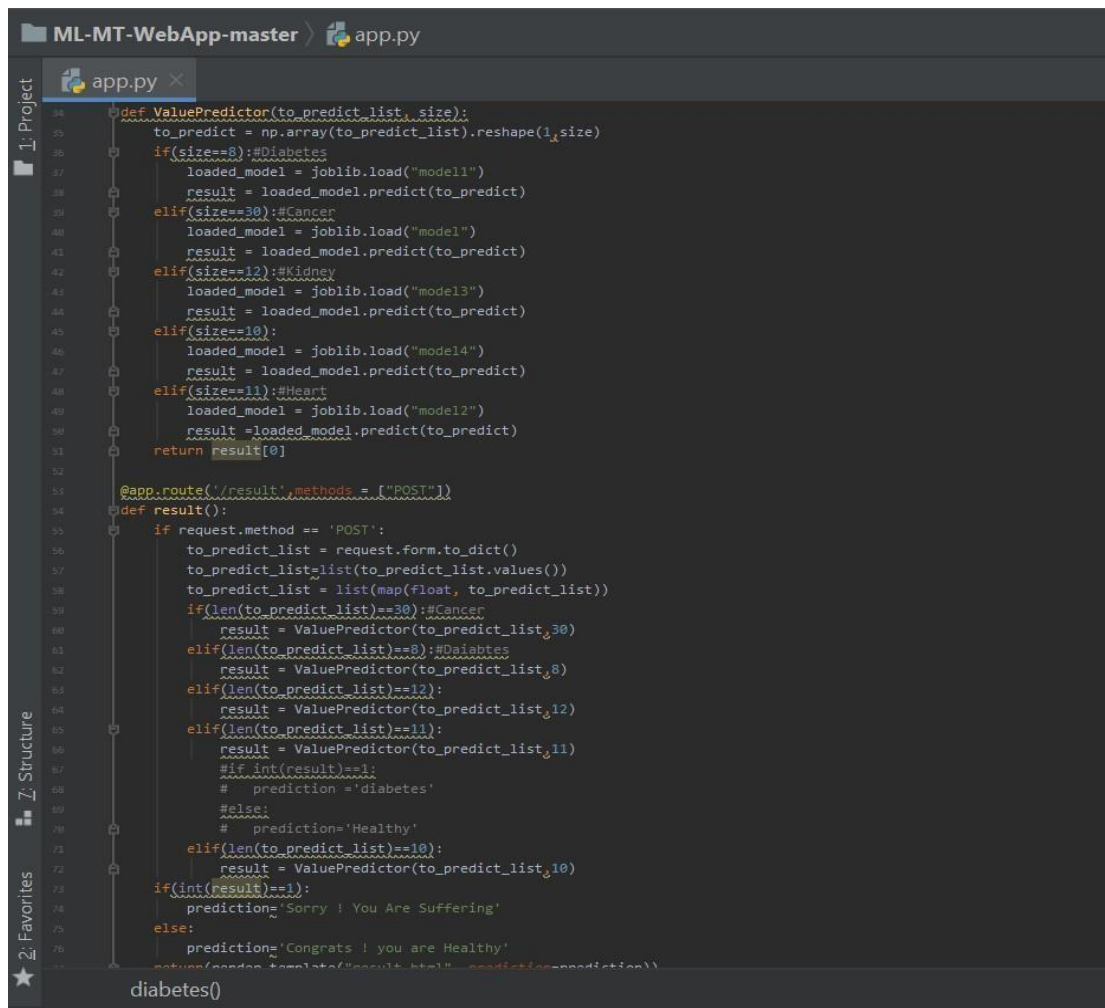
figure 5.17

FLASK IS USED TO RUN THIS ENVIRONMENT TOGETHER

Now we know everything about the project like different models used for different diseases and their respective datasets which were used to train those models. Html pages used to get the entries required to predict the possibilities of disease.

Now it's time to put everything together to build a working Machine learning based web application. For this I used Flask web framework to build API (application programming interface) which will help our front end to connect with our backend and share data accordingly. I build this flask app in PyCharm IDE (Integrated Development Environment) Below is the code I write for the same.

I USED **@app.route** METHOD TO CONNECT BETWEEN
OTHER PAGES LIKE CANCER , DIABETES , HEART ,



```
def ValuePredictor(to_predict_list, size):
    to_predict = np.array(to_predict_list).reshape(1, size)
    if(size==8):#Diabetes
        loaded_model = joblib.load("model1")
        result = loaded_model.predict(to_predict)
    elif(size==30):#Cancer
        loaded_model = joblib.load("model")
        result = loaded_model.predict(to_predict)
    elif(size==12):#Kidney
        loaded_model = joblib.load("model3")
        result = loaded_model.predict(to_predict)
    elif(size==10):
        loaded_model = joblib.load("model4")
        result = loaded_model.predict(to_predict)
    elif(size==11):#Heart
        loaded_model = joblib.load("model2")
        result = loaded_model.predict(to_predict)
    return result[0]

@app.route('/result', methods = ['POST'])
def result():
    if request.method == 'POST':
        to_predict_list = request.form.to_dict()
        to_predict_list=list(to_predict_list.values())
        to_predict_list = list(map(float, to_predict_list))
        if(len(to_predict_list)==30):#Cancer
            result = ValuePredictor(to_predict_list,30)
        elif(len(to_predict_list)==8):#Diabetes
            result = ValuePredictor(to_predict_list,8)
        elif(len(to_predict_list)==12):
            result = ValuePredictor(to_predict_list,12)
        elif(len(to_predict_list)==11):
            result = ValuePredictor(to_predict_list,11)
        #if int(result)==1:
        #    prediction='diabetes'
        #else:
        #    prediction='Healthy'
        elif(len(to_predict_list)==10):
            result = ValuePredictor(to_predict_list,10)
        if(int(result)==1):
            prediction='Sorry ! You Are Suffering'
        else:
            prediction='Congrats ! you are Healthy'
        return(render_template("result.html", prediction=prediction))
```

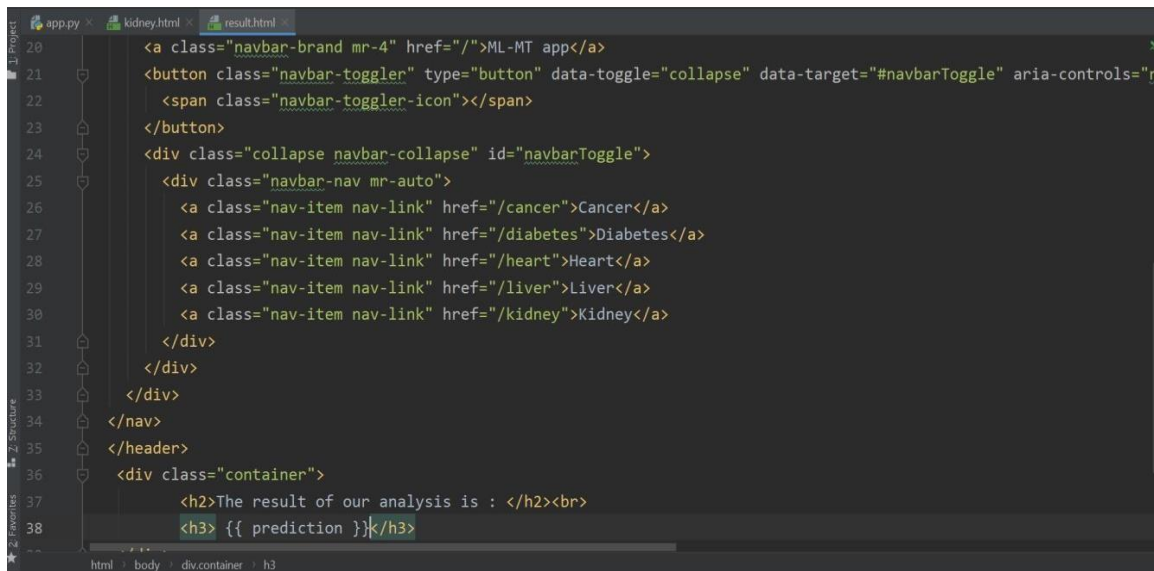
KIDNEY , LIVER . figure 5.18

In above code I used **@app.route** method to route result page. Inside that I defined a function which is matching the length of input data with the trained column length. And then pushes that data (if matching is proper) to another function (Value Predictor) which is defined to predict the final output in the form of 0 and 1. In Value Predictor function I have loaded all the machine learning models that I have built before using python and scikit-learn libraries. The data is being provided to the suitable model using a simple math trick i.e. whichever model total column count matches with the length of the data values entered. Finally the output

patient is suffering from that disease else if it matches with then there is noworry the patient is healthy from that disease. Finally, we will be able to see the output of whether suffering or not suffering onanother page which we have routed at the start called the [result page](#). The resultfunction will render the result.html page and throw the output to this page.

The [html code](#) for the result page is below:

Figure 5.19



```
20 <a class="navbar-brand mr-4" href="/">ML-MT app</a>
21 <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarToggle" aria-controls="n
22 <span class="navbar-toggler-icon"></span>
23 </button>
24 <div class="collapse navbar-collapse" id="navbarToggle">
25 <div class="navbar-nav mr-auto">
26 <a class="nav-item nav-link" href="/cancer">Cancer</a>
27 <a class="nav-item nav-link" href="/diabetes">Diabetes</a>
28 <a class="nav-item nav-link" href="/heart">Heart</a>
29 <a class="nav-item nav-link" href="/liver">Liver</a>
30 <a class="nav-item nav-link" href="/kidney">Kidney</a>
31 </div>
32 </div>
33 </div>
34 </nav>
35 </header>
36 <div class="container">
37 <h2>The result of our analysis is : </h2><br>
38 <h3>{{ prediction }}</h3>
```

Chapter- 6

TESTING & DEPLOYMENT PART

For the deployment part I used **HEROKU** a cloud platform run by **SALESEFORCE**.

HEROKU

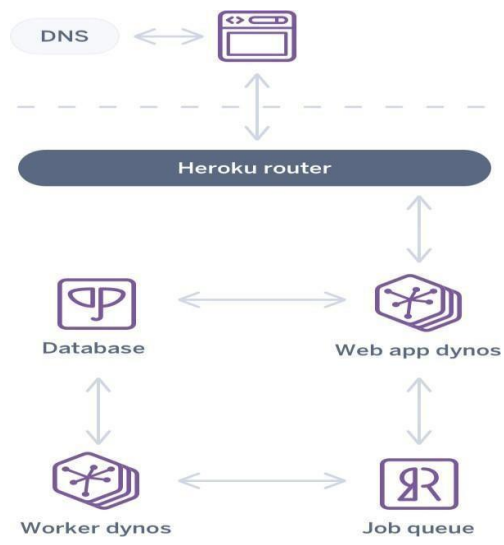
Heroku is a **cloud platform as a service** (PaaS) supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported only the Ruby programming language.

Supports **Java, Node.js, Scala, Clojure, Python, PHP, and Go**. For this reason, Heroku is said to be a polyglot platform as it has features for a developer to build, run and scale applications in a similar manner across most languages. Heroku was acquired by a **Salesforce** in 2010 for \$212 million.

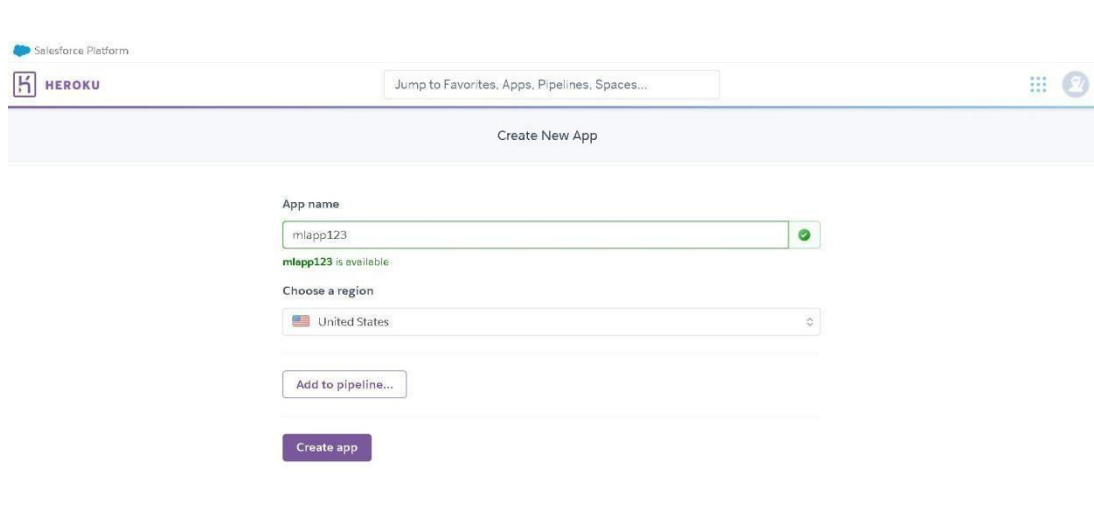
Applications that are run on Heroku typically have a unique domain used to route HTTP requests to the correct application container or *dyno*. Each of the dynos are spread across a "dyno grid" which consists of several servers. Heroku's **Git** server handles application repository pushes from permitted users.

All Heroku services are hosted on Amazon's **EC2** cloud-computing platform.

Figure 6.1

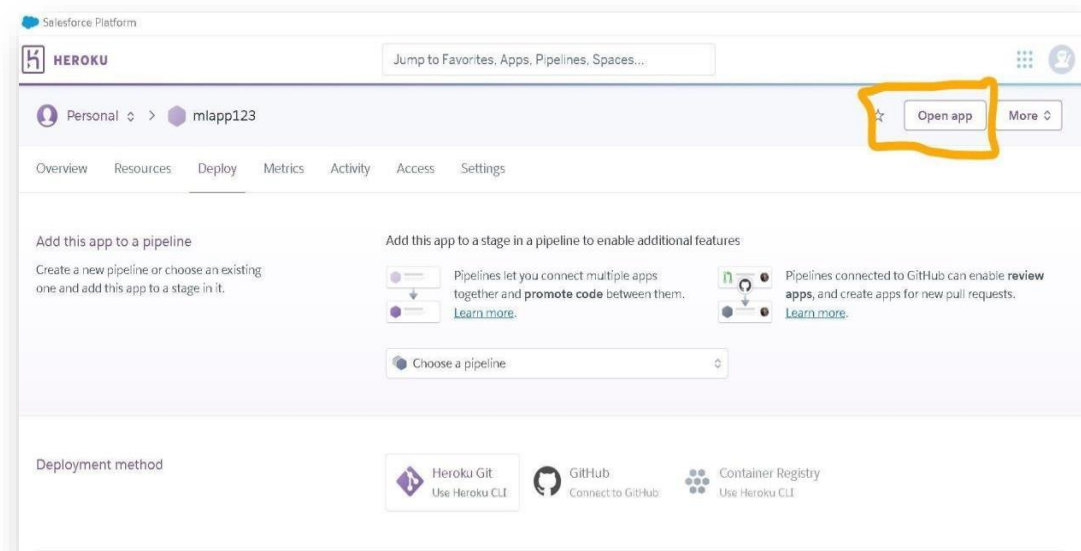


THE FIRST PAGE OF HEROKU LOOKS LIKE THIS.



The screenshot shows the 'Create New App' page on the Heroku dashboard. At the top, there's a navigation bar with the Heroku logo and a search bar. Below the navigation bar, the page title 'Create New App' is centered. The main form contains an 'App name' field with the text 'mlapp123' and a green checkmark icon to its right. Below the name field, a message states 'mlapp123 is available'. Underneath, there's a 'Choose a region' dropdown menu currently set to 'United States'. At the bottom of the form, there are two buttons: 'Add to pipeline...' and 'Create app'.

IN ABOVE PAGE YOU HAVE TO GIVE YOUR APP NAME AND THEN CREATE APP TO GET STARTED.



The screenshot shows the Heroku app page for 'mlapp123'. The top navigation bar includes the Heroku logo, a search bar, and a user profile icon. Below the navigation bar, the breadcrumb 'Personal > mlapp123' is visible. To the right of the breadcrumb, there's a yellow box highlighting the 'Open app' button and a 'More' dropdown menu. Below the breadcrumb, there's a tabbed interface with 'Overview', 'Resources', 'Deploy', 'Metrics', 'Activity', 'Access', and 'Settings'. The 'Overview' tab is selected. The main content area has two sections: 'Add this app to a pipeline' and 'Add this app to a stage in a pipeline to enable additional features'. The 'Add this app to a pipeline' section includes a dropdown menu labeled 'Choose a pipeline'. The 'Add this app to a stage in a pipeline to enable additional features' section includes two cards: 'Pipelines let you connect multiple apps together and promote code between them.' and 'Pipelines connected to GitHub can enable review apps, and create apps for new pull requests.' At the bottom, there's a 'Deployment method' section with three options: 'Heroku Git Use Heroku CLI', 'GitHub Connect to GitHub', and 'Container Registry Use Heroku CLI'.

Figure 6.2

```
We use c
You can g

remote: Downloading scipy-1.7.1-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.whl (28.5 MB)
remote: Downloading scipy-1.7.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.whl (28.5 MB)
remote: Downloading scipy-1.6.3-cp37-cp37m-manylinux1_x86_64.whl (27.4 MB)
remote: Downloading scipy-1.6.2-cp37-cp37m-manylinux1_x86_64.whl (27.4 MB)
remote: Downloading scipy-1.6.1-cp37-cp37m-manylinux1_x86_64.whl (27.4 MB)
remote: Downloading scipy-1.6.0-cp37-cp37m-manylinux1_x86_64.whl (27.4 MB)
remote: Downloading scipy-1.5.4-cp37-cp37m-manylinux1_x86_64.whl (25.9 MB)
remote: Collecting zipp>=0.5
remote: Downloading zipp-3.8.0-py3-none-any.whl (5.4 kB)
remote: Collecting typing-extensions>=3.6.4
remote: Downloading typing_extensions-4.2.0-py3-none-any.whl (24 kB)
remote: Installing collected packages: pytz, zipp, Werkzeug, typing-extensions, six, Pillow, numpy, MarkupSafe, j
oblib, itsdangerous, gunicorn, scipy, python-dateutil, Jinja2, importlib-metadata, scikit-learn, pandas, click, Flask
remote: Successfully installed Flask-1.0.2 Jinja2-2.10.1 MarkupSafe-1.1.1 Pillow-8.3.2 Werkzeug-0.15.5 click-8.1.
3 gunicorn-19.9.0 importlib-metadata-4.11.3 itsdangerous-1.1.0 joblib-1.1.0 numpy-1.15.1 pandas-0.25.1 python-dateutil-2
.8.2 pytz-2022.1 scikit-learn-0.22.1 scipy-1.5.4 six-1.16.0 typing-extensions-4.2.0 zipp-3.8.0
remote: -----> Discovering process types
remote: Procfile declares types -> web
remote: -----> Compressing...
remote: Done: 135.4M
remote: -----> Launching...
remote: Released v3
remote: https://mlapp123.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/mlapp123.git
 * [new branch] master -> master

C:\Users\arun0\Downloads\ML-MT-WebApp-master\ML-MT-WebApp-master>
```

Figure 6.3

Chapter 7

CONCLUSION

The conclusion of a decision prediction report will depend on the specific analysis and findings presented in the report. In general, the conclusion should summarize the key insights gained from the analysis and provide recommendations based on those insights.

If the report involves predicting a decision, the conclusion should discuss the accuracy and reliability of the prediction model and the potential implications of the predicted decision. It may also be useful to discuss any limitations or uncertainties in the analysis and potential areas for future research.

In addition, the conclusion should emphasize the importance of making data-driven decisions and the value of predictive analytics in improving decision-making processes. Overall, the conclusion should provide a clear and concise summary of the key findings and recommendations presented in the report.

If the report involves predicting a decision, the conclusion should discuss the accuracy and reliability of the prediction model and the potential implications of the predicted decision. It may also be useful to discuss any limitations or uncertainties in the analysis and potential areas for future research.

Chapter - 8

BIBLIOGRAPHY

Here are some references for bibliography in disease prediction using machine learning:

Raj komar, A., Dean, J., & Kohane, I. (2019). Machine learning in medicine. *New England Journal of Medicine*, 380(14), 1347-1358.

Choi, E., Baha, M. T., Kulas, J. A., Schuetz, A., Stewart, W. F., & Sun, J. (2016). Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in neural information processing systems* (pp. 3504-3512).

T. A., Denny, J. C., & Levy, M. A. (2013). Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data. *One*, 8(6), e66341.

Kumar, A., & Kim, J. H. (2017). Applications of machine learning in cancer prediction and prognosis. *Frontiers in artificial intelligence and applications*, 295, 55-70.

R., Li, L., Kidd, B. A., & Dudley, J. T. (2016). Deep patient: An unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6, 26094.

Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way, G. P., ... & Xiao, C. (2018). Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141), 20170387.

Jung, K. J., & Jeon, H. (2020). Machine learning-based prediction models for Alzheimer's disease using genetic data. *International Journal of Medical Informatics*, 136, 104063.

Yang, Y., Zhao, Z., Li, L., Li, Y., & Wang, Y. (2020). A review of deep learning models for disease prediction in clinical notes. *BMC Medical Informatics and Decision Making*, 20(1), 33.

THIS IS THE LINK :- <https://mlapp123.herokuapp.com/>

-----**END-----OF-----PROJECT**-----