CANCER-PRONE-PREDICTOR

PROJECT PROGRESS REPORT

OF PROJECT-1 (IT795)

BACHELOR OF TECHNOLOGY
in
Information Technology

SUBMITTED BY

Saumya Ranjan (13000217036)
Rishab Kumar (13000217051)
Saquib Alam (13000217038)
Pallavi Saumya (13000217068)

Under the Supervision of
Dr TAPASI BHATTACHARIEE

Department of Information Technology
Techno India, Salt Lake.
Kolkata -700091

PROJECT PROGRESS REPORT

1. **Introduction**

2. **Literature Survey**

3. **Detail description of the project**

4. **Design/Architecture**

5. **Coding**

6. **Implementations**

7. **Conclusion**

Submitted by,

(Signature of students)

Signature of the mentor

INTRODUCTION

Cancer is a critical disease from many years. This leads to death if it is not diagnosed at early stage. Cancer has been characterized as a heterogeneous disease consisting of many different subtypes. The early diagnosis and prognosis of a cancer type have become a necessity in cancer research, as it can facilitate the subsequent clinical management of patients.

Manual detection of a cancer cell is a tiresome task and involves human error, and hence computer-aided mechanisms are applied to obtain better results as compared with manual pathological detection systems

Our model will predict whether the person is cancer prone or not when various inputs affecting the parameters is provided.

We have used Machine Learning for making the predication and deployed the same model on a web host using a Python Flak file and implemented the model.

Literature Survey

Various researches have been carried out using the data mining techniques for the diagnosis and prognosis of Cancer. The goal of this studies was to identify the most well performing algorithms used on medical databases.

At times some algorithms perform better than others, but there are cases when a combination of the best properties of some of the aforementioned algorithms together results more effective. An extensive search was conducted relevant to the use of ML techniques in cancer susceptibility, recurrence and survivability prediction. The majority of these studies use different types of input data: genomic, clinical, histological, imaging, demographic, epidemiological data or combination of these. With the advent of new technologies in the field of medicine, large amounts of cancer data have been collected and are available to the medical research community. However, the accurate prediction of a disease outcome is one of the most interesting and challenging tasks for physicians. As a result, ML methods have become a popular tool for medical researchers.

An obvious trend in the proposed works includes the integration of mixed data, such as clinical and genomic. However, a common problem that we noticed in several works is the lack of external validation or testing regarding the predictive performance of their models. It is clear that the application of ML methods could improve the accuracy of cancer susceptibility, recurrence and survival prediction. Based on [3], the accuracy of cancer prediction outcome has significantly improved by 15%–20% the last years, with the application of ML techniques.

These techniques can discover and identify patterns and relationships between them, from complex datasets, while they are able to effectively predict future outcomes of a cancer type. In the present work only studies that employed ML techniques for modeling cancer diagnosis and prognosis are presented.

DESCRIPTION OF THE PROJECT

Cancer prediction starts with the Registration/Login Page of the user. At first the medical staff will provide the details of patients like clump thickness, uniformity of cell size, Bare nuclei and various parameters Which we have been analyzed from dataset. Now, our model will start predicting whether the person is cancer prone or not.

We have taken the dataset from UCI Machine Learning Repository.
In the predicting algorithm, first we have analyzed the dataset.
Here we came to know that there is no missing value in the dataset.
Then we have checked using heatmap whether the any two independent variable is highly correlated or not.

We split the data set into training and testing sets and use the training set to train the model and testing set to test the model. We then evaluate the model performance based on an error metric to determine the accuracy of the model.
Then we started applying various models and came to conclusion that In Decision Tree Classifier we are getting the highest accuracy 95.90.

This method however, is not very reliable as the accuracy obtained for one test set can be very different to the accuracy obtained for a different test set. **K-fold Cross Validation(CV)** provides a solution to this problem by dividing the data into folds and ensuring that each fold is used as a testing set at some point.

After applying k-fold Cross Validation we came to know that its score is matching with the accuracy score.

Now by using this technique our machine is getting trained for prediction. Thus, Decision Tree Classifier finally enables us to predict whether the person is cancer prone or not.

❖ About dataset:

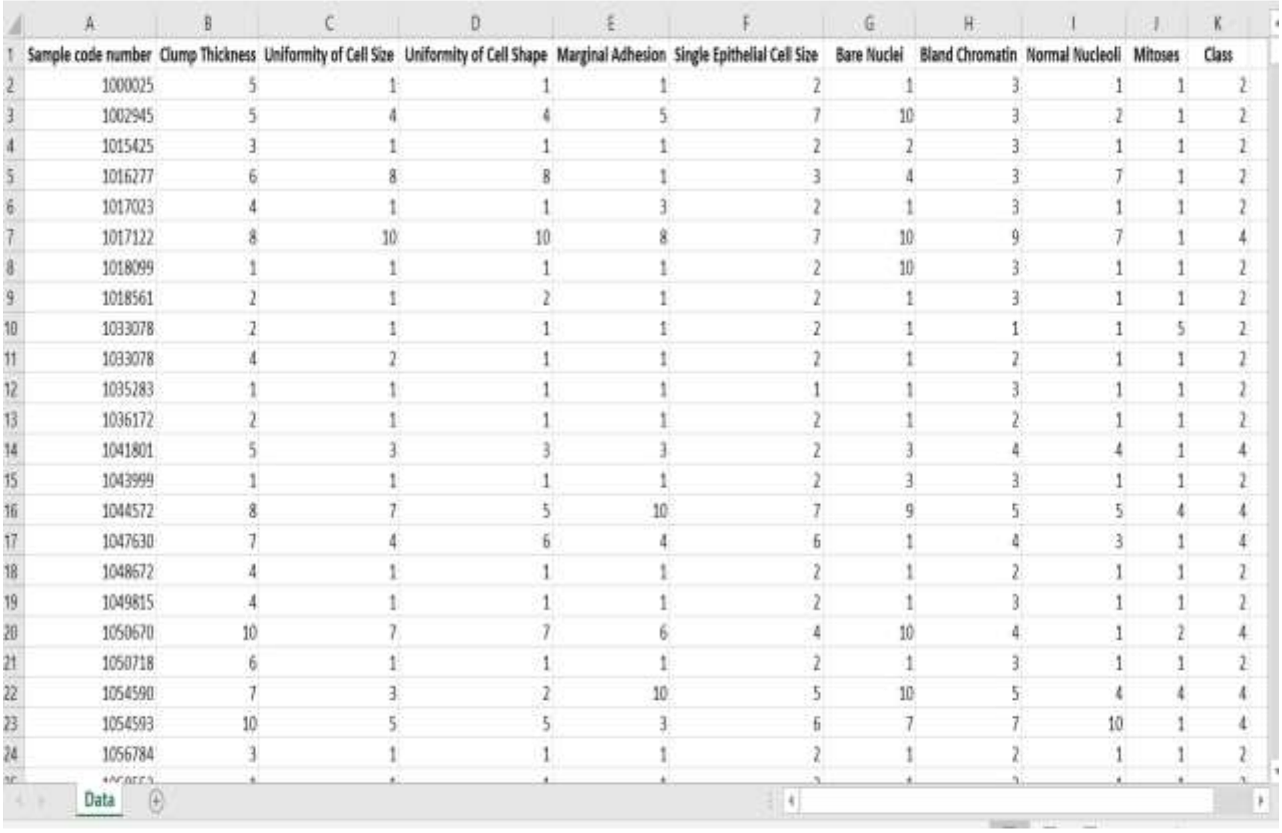
Source-

Creator:

Dr. WIlliam H. Wolberg (physician)
University of Wisconsin Hospitals
Madison, Wisconsin, USA

Attribute Information:

1.Sample code number: id number
2. Clump Thickness: 1 – 10
3. Uniformity of Cell Size: 1 – 10
4. Uniformity of Cell Shape: 1 – 10
5. Marginal Adhesion: 1 – 10
6. Single Epithelial Cell Size: 1 – 10
7. Bare Nuclei: 1 – 10
8. Bland Chromatin: 1 – 10
9. Normal Nucleoli: 1 – 10
10. Mitoses: 1 – 10
11. Class: (2 for benign, 4 for malignant)



	A	B	C	D	E	F	G	H	I	J	K
1	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
2	1000025	5	1	1	1	2	1	3	1	1	2
3	1002945	5	4	4	5	7	10	3	2	1	2
4	1015425	3	1	1	1	2	2	3	1	1	2
5	1016277	6	8	8	1	3	4	3	7	1	2
6	1017023	4	1	1	3	2	1	3	1	1	2
7	1017122	8	10	10	8	7	10	9	7	1	4
8	1018099	1	1	1	1	2	10	3	1	1	2
9	1018561	2	1	2	1	2	1	3	1	1	2
10	1033078	2	1	1	1	2	1	1	1	5	2
11	1033078	4	2	1	1	2	1	2	1	1	2
12	1035283	1	1	1	1	1	1	3	1	1	2
13	1036172	2	1	1	1	2	1	2	1	1	2
14	1041801	5	3	3	3	2	3	4	4	1	4
15	1043999	1	1	1	1	2	3	3	1	1	2
16	1044572	8	7	5	10	7	9	5	5	4	4
17	1047630	7	4	6	4	6	1	4	3	1	4
18	1048672	4	1	1	1	2	1	2	1	1	2
19	1049815	4	1	1	1	2	1	3	1	1	2
20	1050670	10	7	7	6	4	10	4	1	2	4
21	1050718	6	1	1	1	2	1	3	1	1	2
22	1054590	7	3	2	10	5	10	5	4	4	4
23	1054593	10	5	5	3	6	7	7	10	1	4
24	1056784	3	1	1	1	2	1	2	1	1	2

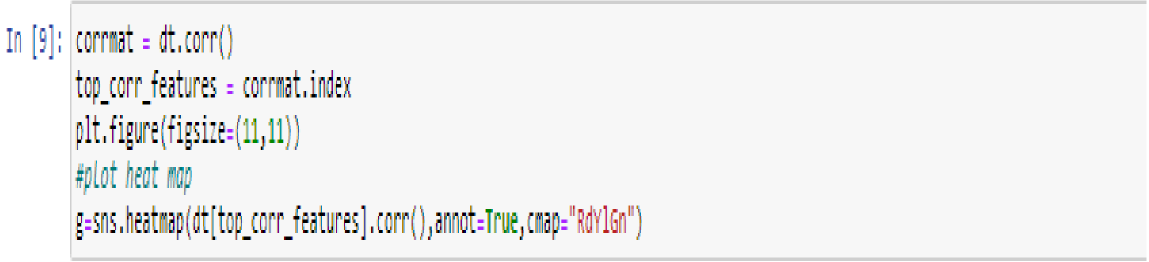
❖ Data Analysis and Model Analysis

I. Heat map:

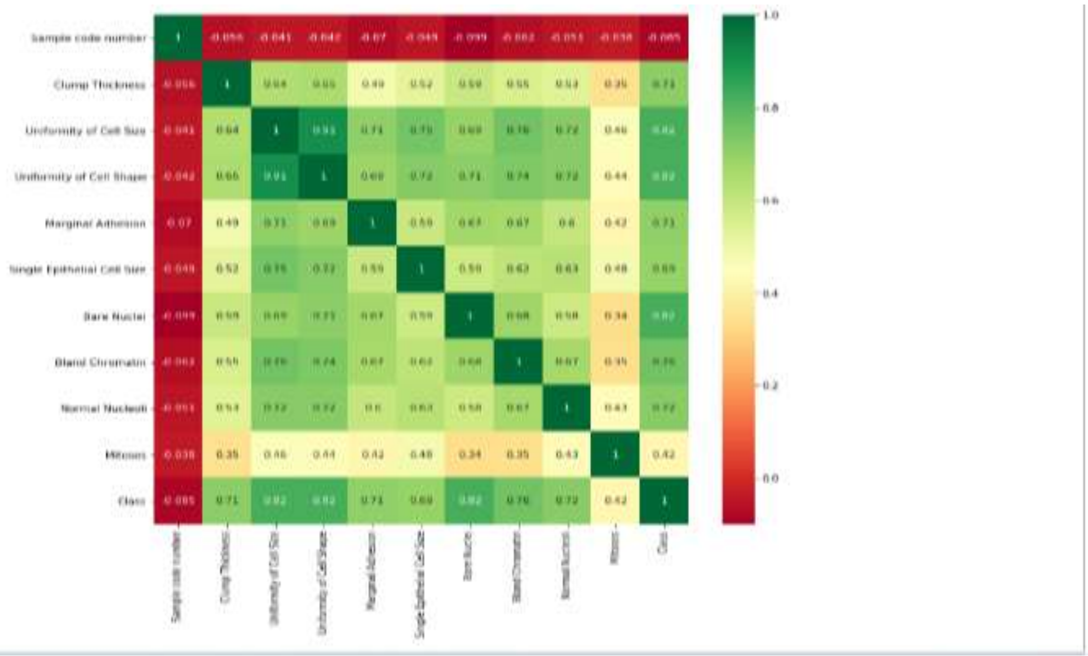
Correlation states how the features are related to each other or the target variable.

Correlation can be positive (increase in one value of feature increases the value of the target variable) or negative (increase in one value of feature decreases the value of the target variable)

Heatmap makes it easy to identify which features are most related to the target variable, we will plot heatmap of correlated features using the seaborn library.

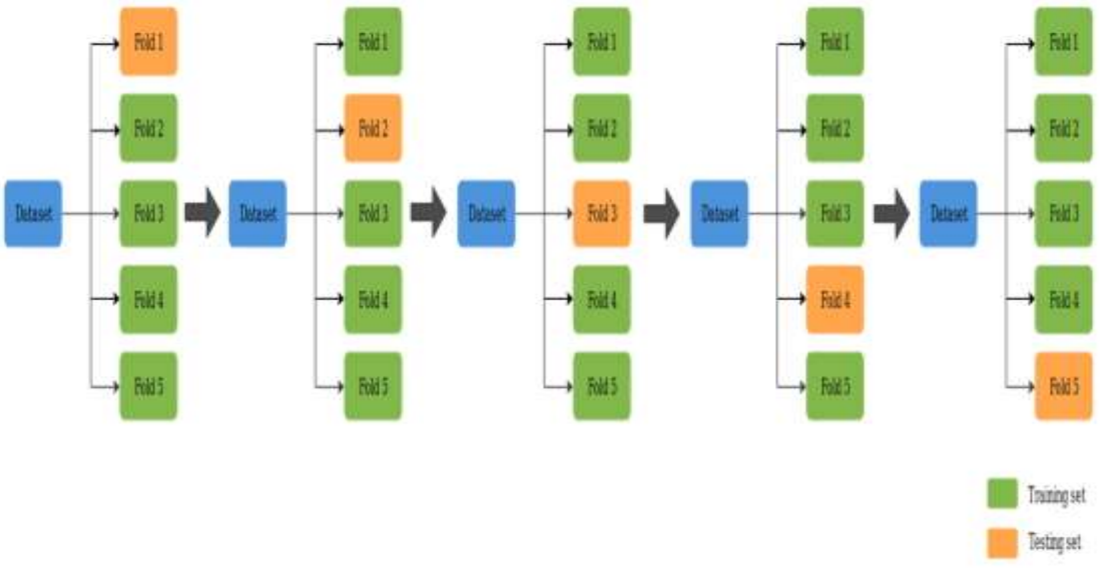


```
In [9]: corrmat = dt.corr()
        top_corr_features = corrmat.index
        plt.figure(figsize=(11,11))
        #plot heat map
        g=sns.heatmap(dt[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



II. K-Fold Cross Validation

K-Fold CV is where a given data set is split into a ***K*** number of sections/folds where each fold is used as a testing set at some point. Let’s take the scenario of 5-Fold cross validation(K=5). Here, the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration,2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds have been used as the testing set.



5-Fold Cross Validation

❖ Decision Tree Classifier

A decision tree is a tree-like graph with nodes representing the place where we pick an attribute and ask a question; edges represent the answers the to the question; and the leaves represent the actual output or class label. They are used in non-linear decision making with simple linear decision surface.

Decision trees classify the examples by sorting them down the tree from the root to some leaf node, with the leaf node providing the classification to the example. Each node in the tree acts as a test case for some attribute, and each edge descending from that node corresponds to one of the possible answers to the test case. This process is recursive in nature and is repeated for every subtree rooted at the new nodes.

Attribute Selection Measures

While implementing a Decision tree, the main issue arises that how to select the best attribute for the root node and for sub-nodes. So, to solve such problems there is a technique which is called as **Attribute selection measure or ASM.** By this measurement, we can easily select the best attribute for the nodes of the tree. There are two popular techniques for ASM, which are:

- **Information Gain**
- **Gini Index**

I. <u>Information Gain:</u>

- Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.
- It calculates how much information a feature provides us about a class.

- According to the value of information gain, we split the node and build the decision tree.
- A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first. It can be calculated using the below formula:

Information Gain= Entropy(S)-
[(Weighted Avg) *Entropy(each feature)

Entropy: Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

- *S= Total number of samples*
- *P(yes)= probability of yes*
- *P(no)= probability of no*

II. Gini Index:

- Gini index is a measure of impurity or purity used while creating a decision tree in the CART (Classification and Regression Tree) algorithm.
- An attribute with the low Gini index should be preferred as compared to the high Gini index.

- It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.
- Gini index can be calculated using the below formula:

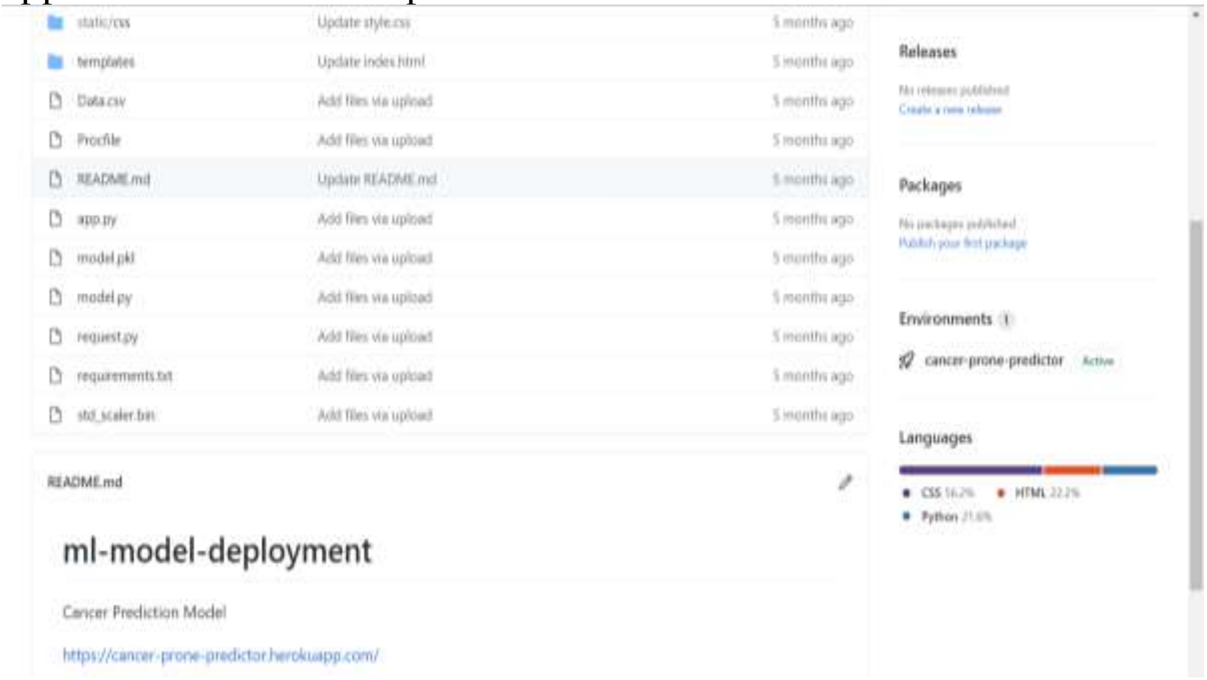
Gini Index= 1- $\sum_j P_j^2$

❖ Deployment

When our Machine Learning algorithm was ready, we have to then deploy this system to a useful and presentable system. To achieve this, we have designed a web application and hosted the same application on a web host.

For this, we have created a web page using **Html & CSS** taking the values of the dataset needed for the **Machine Learning algorithm**. Then we have made a web application on a **Flask framework using python**. In that python file we have called our machine learning model to make predictions.

We have used **GitHub as a Version Control** and deployed our Web application on **Heroku** platform.



static/css	Update style.css	5 months ago
templates	Update index.html	5 months ago
Data.csv	Add files via upload	5 months ago
Procfile	Add files via upload	5 months ago
README.md	Update README.md	5 months ago
app.py	Add files via upload	5 months ago
model.pkl	Add files via upload	5 months ago
model.py	Add files via upload	5 months ago
request.py	Add files via upload	5 months ago
requirements.txt	Add files via upload	5 months ago
std_scaler.bin	Add files via upload	5 months ago

README.md

ml-model-deployment

Cancer Prediction Model

https://cancer-prone-predictor.herokuapp.com/

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

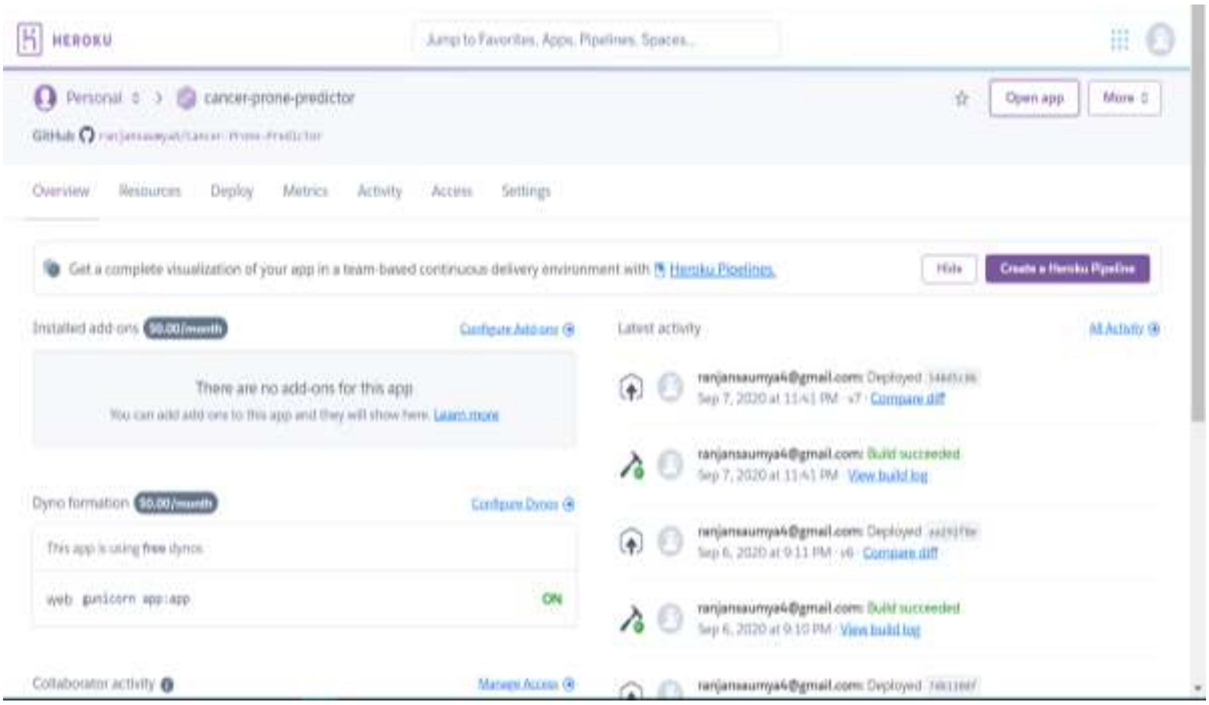
Environments 1

cancer-prone-predictor Active

Languages

● CSS 56.2% ● HTML 22.2%
● Python 21.6%

Heroku is a cloud platform as a service (PaaS) supporting several programming languages. The Heroku network runs the customer's apps in virtual containers which execute on a reliable runtime environment. Heroku calls these containers "Dynos". These Dynos can run code written in Node, Ruby, PHP, Go, Scala, Python, Java, or Clojure. Heroku also provides custom build packs with which the developer can deploy apps in any other language. Heroku lets the developer scale the app instantly just by either increasing the number of dynos or by changing the type of dyno the app runs in.



GitHub ranjansaumya4/Cancer-Prone-Predictor

Get a complete visualization of your app in a team-based continuous delivery environment with Heroku Pipelines.

Installed add-ons $0.00/month

There are no add-ons for this app

You can add add-ons to this app and they will show here. Learn more

Dyno formation $0.00/month

This app is using free dynos

web gunicorn app:app ON

Collaborator activity

Latest activity

ranjansaumya4@gmail.com: Deployed
Sep 7, 2020 at 11:41 PM · v7 · Compare diff

ranjansaumya4@gmail.com: Build succeeded
Sep 7, 2020 at 11:41 PM · View build log

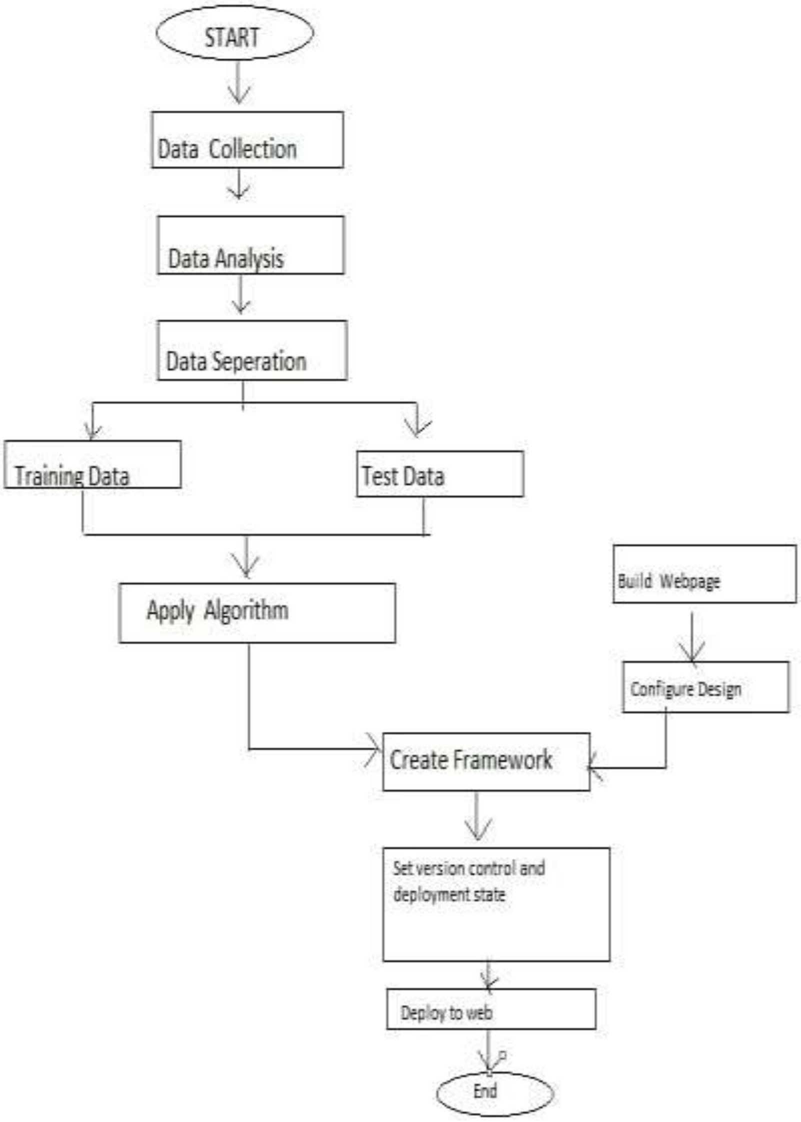
ranjansaumya4@gmail.com: Deployed
Sep 6, 2020 at 9:11 PM · v6 · Compare diff

ranjansaumya4@gmail.com: Build succeeded
Sep 6, 2020 at 9:10 PM · View build log

ranjansaumya4@gmail.com: Deployed

After successful deployment of our model we can access that on a url provided by heroku from any remote loaction and that application will implement our Machine Leaning model and make the predictions and will display the predictions on the web page.
The url assigned to our application by heroku is –
https://cancer-prone-predictor.herokuapp.com/

DESIGN/ARCHITECTURE

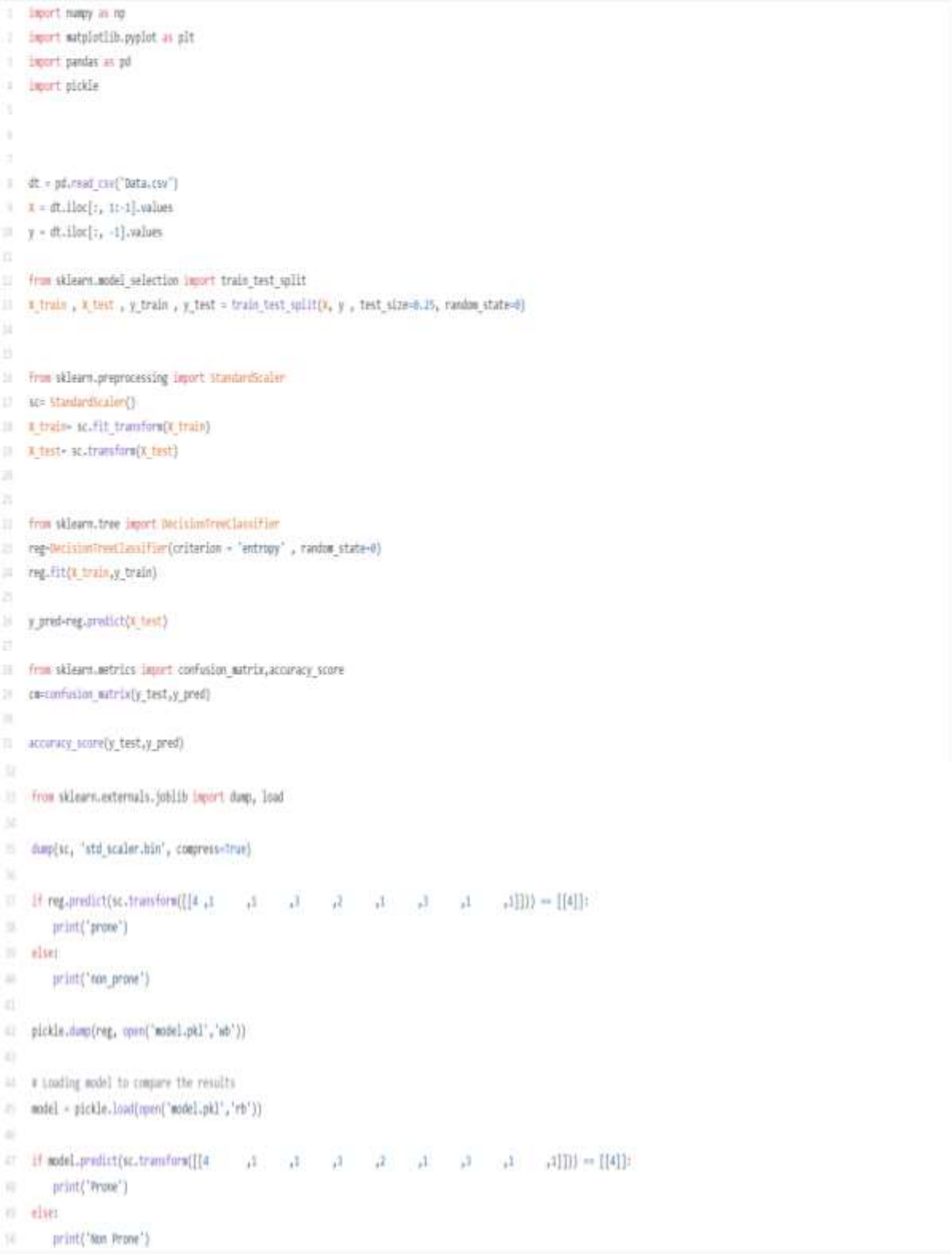


Flowchart for engaged development

CODING

I. MACHINE LEARNING CODE:

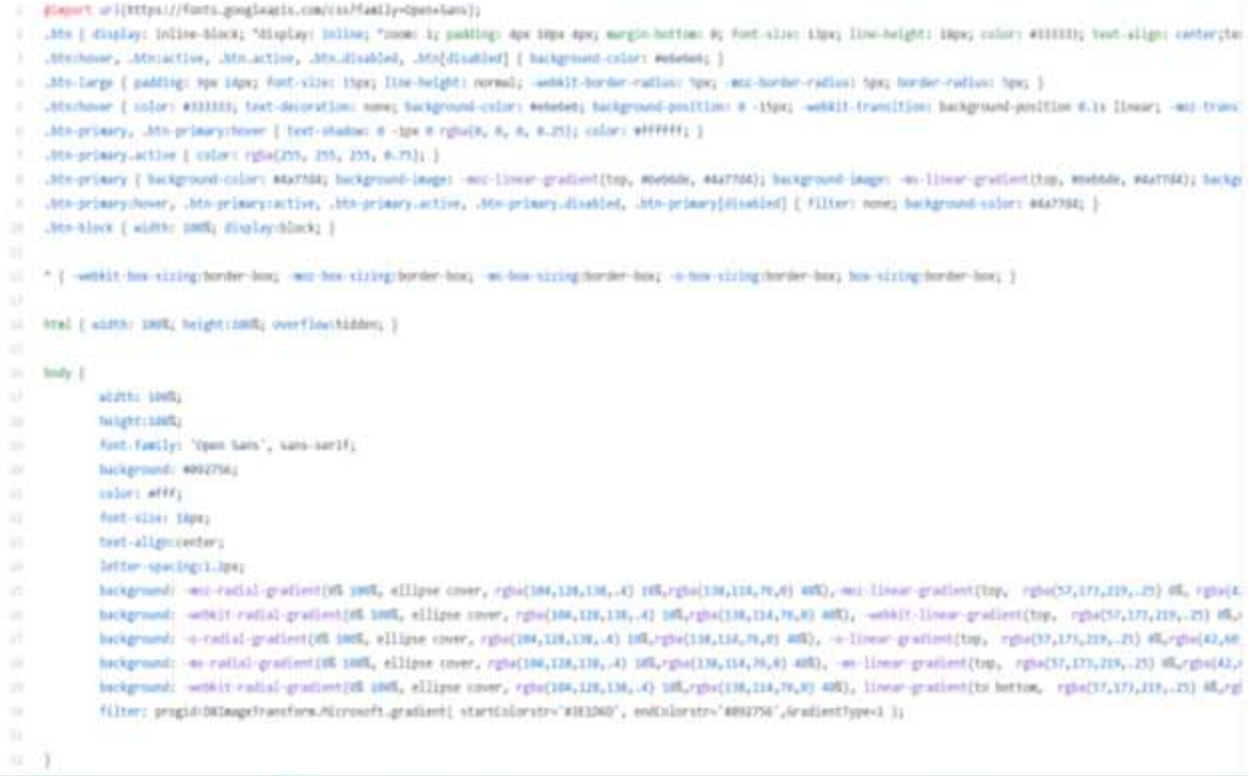
A. model.py



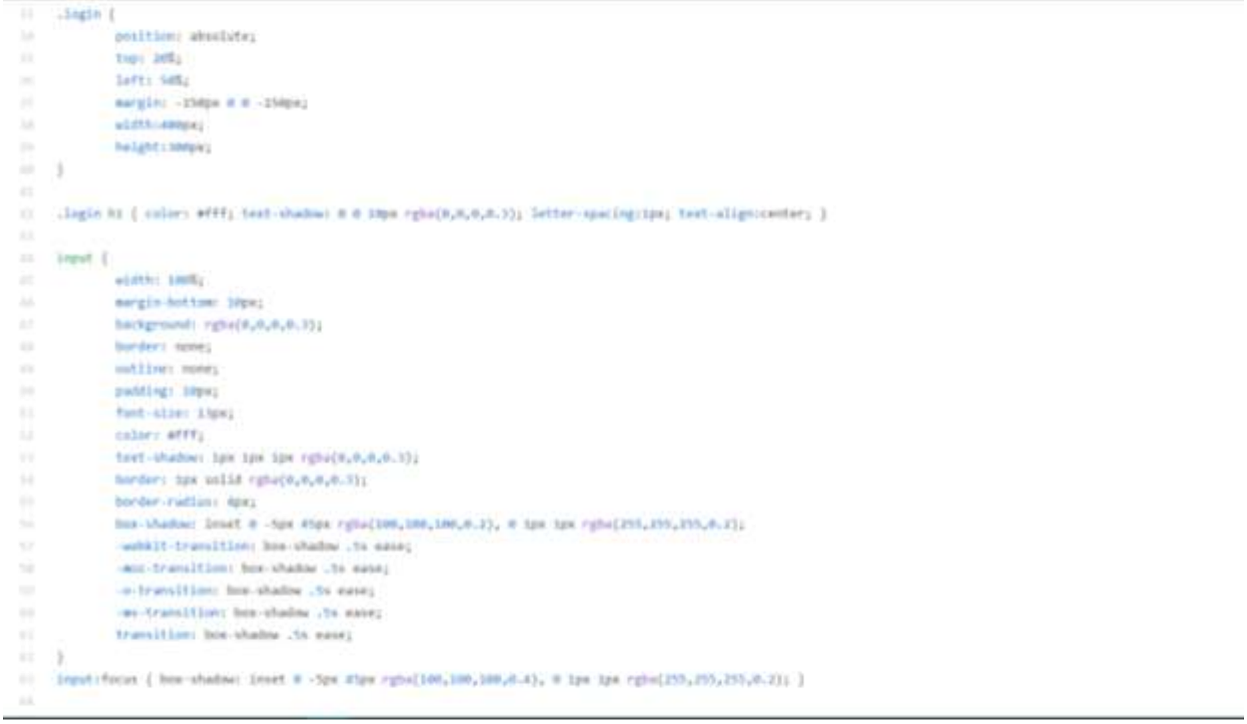
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import pickle



dt = pd.read_csv("Data.csv")
X = dt.iloc[:, 1:-1].values
y = dt.iloc[:, -1].values

from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(X, y , test_size=0.25, random_state=0)


from sklearn.preprocessing import StandardScaler
sc= StandardScaler()
X_train= sc.fit_transform(X_train)
X_test= sc.transform(X_test)


from sklearn.tree import DecisionTreeClassifier
reg=DecisionTreeClassifier(criterion = 'entropy' , random_state=0)
reg.fit(X_train,y_train)

y_pred=reg.predict(X_test)

from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)

accuracy_score(y_test,y_pred)

from sklearn.externals.joblib import dump, load

dump(sc, 'std_scaler.bin', compress=True)

if reg.predict(sc.transform([[4 ,1     ,1     ,3     ,2     ,1     ,3     ,1     ,1]])) == [[4]]:
    print('prone')
else:
    print('non_prone')

pickle.dump(reg, open('model.pkl','wb'))

# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))

if model.predict(sc.transform([[4     ,1     ,1     ,3     ,2     ,1     ,3     ,1     ,1]])) == [[4]]:
    print('Prone')
else:
    print('Non Prone')
```

II. Html and CSS code:

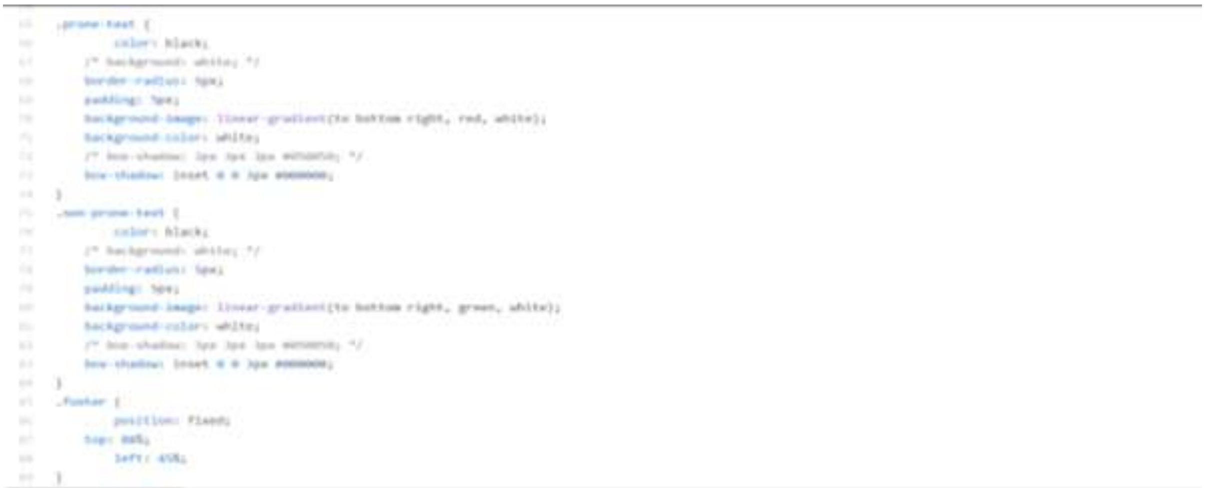
A. style.css



```css
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn { display: inline-block; *display: inline; *zoom: 1; padding: 4px 10px 4px; margin-bottom: 0; font-size: 13px; line-height: 18px; color: #333333; text-align: center;te
.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color: #e6e6e6; }
.btn-large { padding: 9px 14px; font-size: 15px; line-height: normal; -webkit-border-radius: 5px; -moz-border-radius: 5px; border-radius: 5px; }
.btn:hover { color: #333333; text-decoration: none; background-color: #e6e6e6; background-position: 0 -15px; -webkit-transition: background-position 0.1s linear; -moz-trans
.btn-primary, .btn-primary:hover { text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25); color: #ffffff; }
.btn-primary.active { color: rgba(255, 255, 255, 0.75); }
.btn-primary { background-color: #4a77d4; background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4); background-image: -ms-linear-gradient(top, #6eb6de, #4a77d4); backg
.btn-primary:hover, .btn-primary:active, .btn-primary.active, .btn-primary.disabled, .btn-primary[disabled] { filter: none; background-color: #4a77d4; }
.btn-block { width: 100%; display:block; }

* { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-sizing:border-box; -o-box-sizing:border-box; box-sizing:border-box; }

html { width: 100%; height:100%; overflow:hidden; }

body {
	width: 100%;
	height:100%;
	font-family: 'Open Sans', sans-serif;
	background: #092756;
	color: #fff;
	font-size: 18px;
	text-align:center;
	letter-spacing:1.2px;
	background: -moz-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%),-moz-linear-gradient(top,  rgba(57,173,219,.25) 0%, rgba(4
	background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -webkit-linear-gradient(top,  rgba(57,173,219,.25) 0%,r
	background: -o-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -o-linear-gradient(top,  rgba(57,173,219,.25) 0%,rgba(42,60
	background: -ms-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), -ms-linear-gradient(top,  rgba(57,173,219,.25) 0%,rgba(42,
	background: -webkit-radial-gradient(0% 100%, ellipse cover, rgba(104,128,138,.4) 10%,rgba(138,114,76,0) 40%), linear-gradient(to bottom,  rgba(57,173,219,.25) 0%,rg
	filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#3E1D6D', endColorstr='#092756',GradientType=1 );

}
```

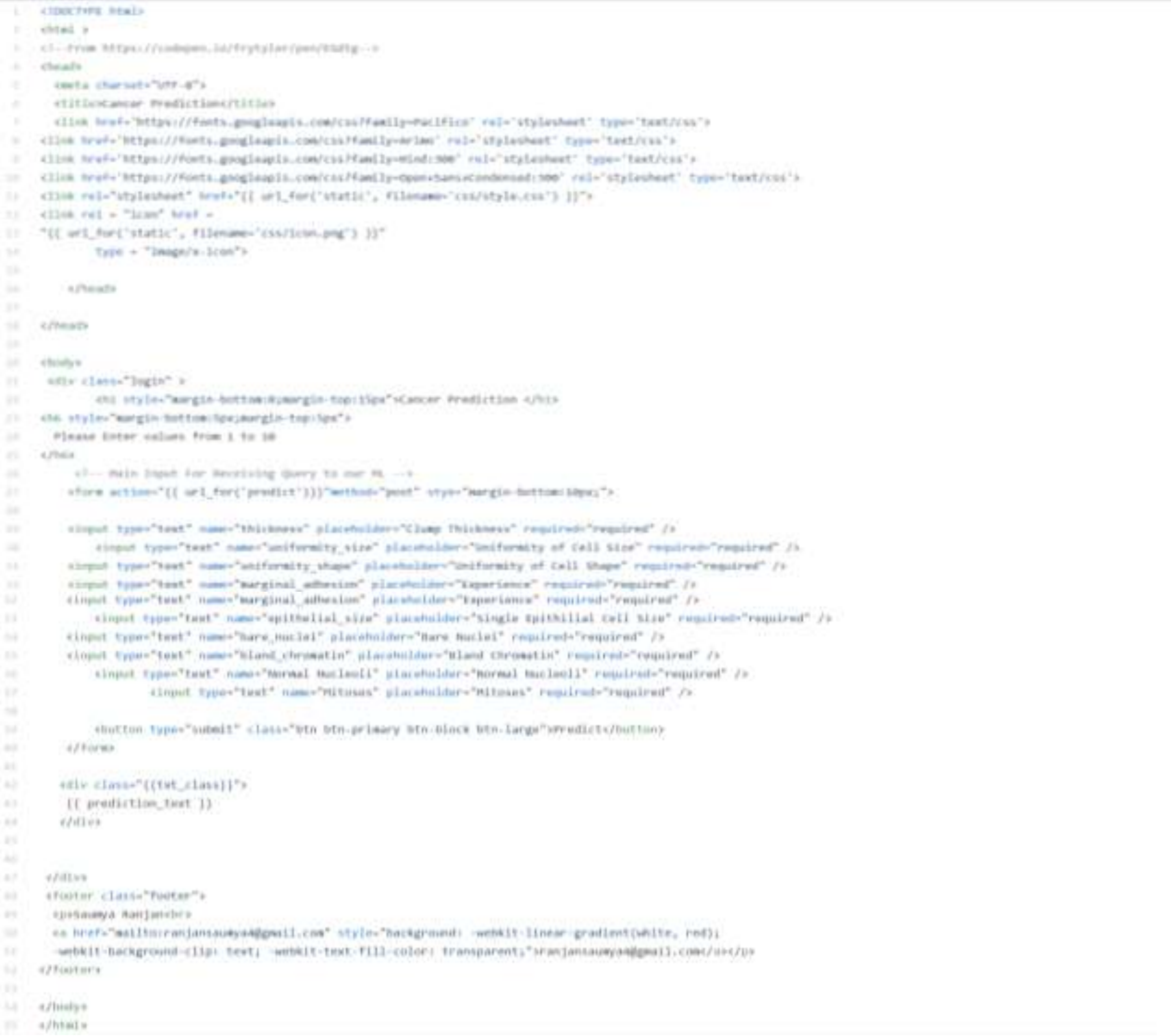


```css
.login {
	position: absolute;
	top: 20%;
	left: 50%;
	margin: -150px 0 0 -150px;
	width:400px;
	height:300px;
}

.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px; text-align:center; }

input {
	width: 100%;
	margin-bottom: 10px;
	background: rgba(0,0,0,0.3);
	border: none;
	outline: none;
	padding: 10px;
	font-size: 13px;
	color: #fff;
	text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
	border: 1px solid rgba(0,0,0,0.3);
	border-radius: 4px;
	box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
	-webkit-transition: box-shadow .5s ease;
	-moz-transition: box-shadow .5s ease;
	-o-transition: box-shadow .5s ease;
	-ms-transition: box-shadow .5s ease;
	transition: box-shadow .5s ease;
}
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px rgba(255,255,255,0.2); }
```



```css
.prone-text {
        color: black;
    /* background: white; */
    border-radius: 5px;
    padding: 5px;
    background-image: linear-gradient(to bottom right, red, white);
    background-color: white;
    /* box-shadow: 3px 3px 3px #474747; */
    box-shadow: inset 0 0 3px #000000;
}
.non-prone-text {
        color: black;
    /* background: white; */
    border-radius: 5px;
    padding: 5px;
    background-image: linear-gradient(to bottom right, green, white);
    background-color: white;
    /* box-shadow: 3px 3px 3px #474747; */
    box-shadow: inset 0 0 3px #000000;
}
.footer {
        position: fixed;
    top: 88%;
        left: 45%;
}
```

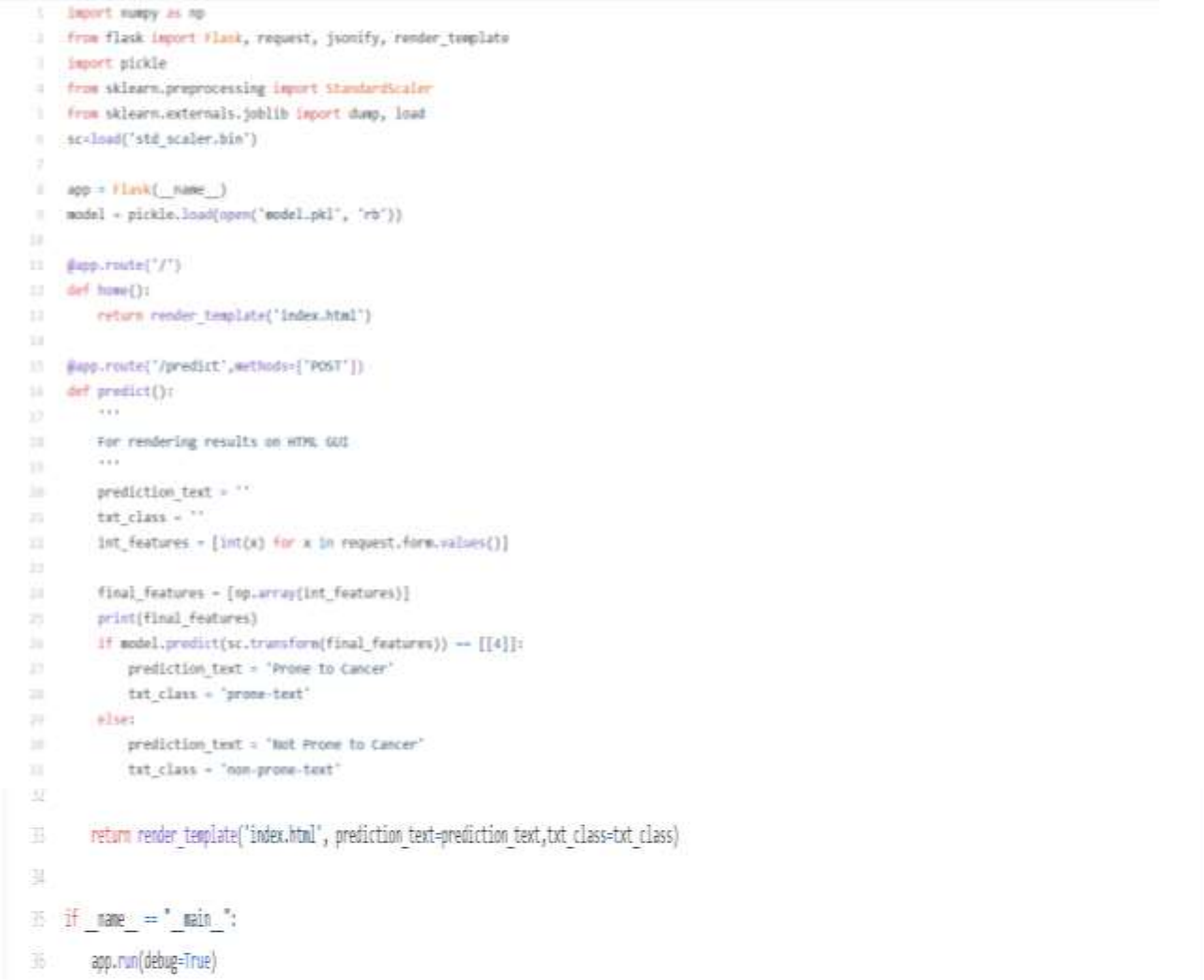
B. index.html



```html
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>Cancer Prediction</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
<link rel = "icon" href =
"{{ url_for('static', filename='css/icon.png') }}"
        type = "image/x-icon">

    </head>

</head>

<body>
 <div class="login" >
        <h1 style="margin-bottom:0;margin-top:15px">Cancer Prediction </h1>
<h6 style="margin-bottom:5px;margin-top:5px">
  Please Enter values from 1 to 10
</h6>
     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post" stye="margin-bottom:10px;">

    <input type="text" name="thickness" placeholder="Clump Thickness" required="required" />
        <input type="text" name="uniformity_size" placeholder="Uniformity of Cell Size" required="required" />
    <input type="text" name="uniformity_shape" placeholder="Uniformity of Cell Shape" required="required" />
    <input type="text" name="marginal_adhesion" placeholder="Experience" required="required" />
    <input type="text" name="marginal_adhesion" placeholder="Experience" required="required" />
        <input type="text" name="epithelial_size" placeholder="Single Epithilial Cell Size" required="required" />
    <input type="text" name="bare_nuclei" placeholder="Bare Nuclei" required="required" />
    <input type="text" name="bland_chromatin" placeholder="Bland Chromatin" required="required" />
        <input type="text" name="Normal Nucleoli" placeholder="Normal Nucleoli" required="required" />
                <input type="text" name="Mitosas" placeholder="Mitoses" required="required" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <div class="{{txt_class}}">
    {{ prediction_text }}
   </div>


 </div>
 <footer class="footer">
  <p>Saumya Ranjan<br>
  <a href="mailto:ranjansaumya4@gmail.com" style="background: -webkit-linear-gradient(white, red);
  -webkit-background-clip: text; -webkit-text-fill-color: transparent;">ranjansaumya4@gmail.com</a></p>
</footer>

</body>
</html>
```

III. Flask Framework:

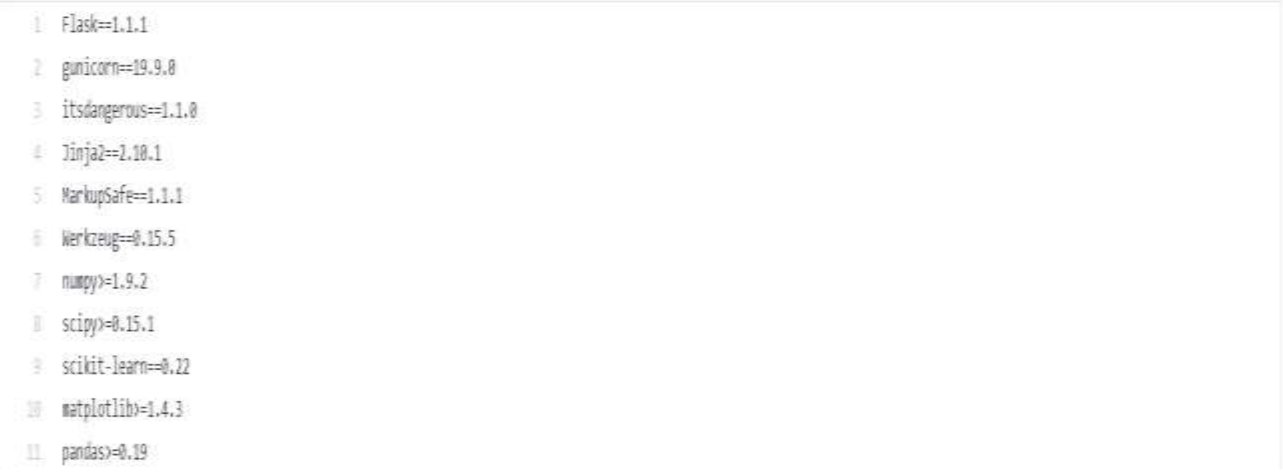
A. app.py



```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
from sklearn.preprocessing import StandardScaler
from sklearn.externals.joblib import dump, load
sc=load('std_scaler.bin')

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    prediction_text = ''
    txt_class = ''
    int_features = [int(x) for x in request.form.values()]

    final_features = [np.array(int_features)]
    print(final_features)
    if model.predict(sc.transform(final_features)) == [[4]]:
        prediction_text = 'Prone to Cancer'
        txt_class = 'prone-text'
    else:
        prediction_text = 'Not Prone to Cancer'
        txt_class = 'non-prone-text'

    return render_template('index.html', prediction_text=prediction_text,txt_class=txt_class)

if __name__ == "__main__":
    app.run(debug=True)
```

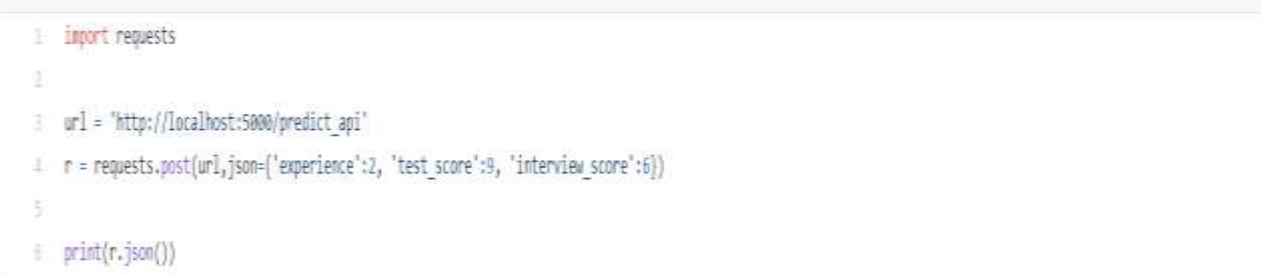
IV. Deployment Code:

A. requirement.txt



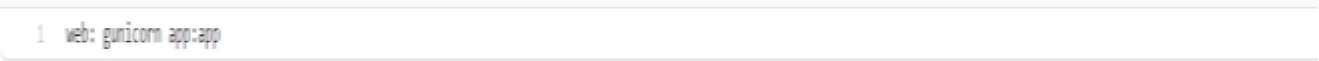
```
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy>=1.9.2
scipy>=0.15.1
scikit-learn==0.22
matplotlib>=1.4.3
pandas>=0.19
```

B. request.py



```python
import requests

url = 'http://localhost:5000/predict_api'
r = requests.post(url,json={'experience':2, 'test_score':9, 'interview_score':6})

print(r.json())
```

C. ProcFile



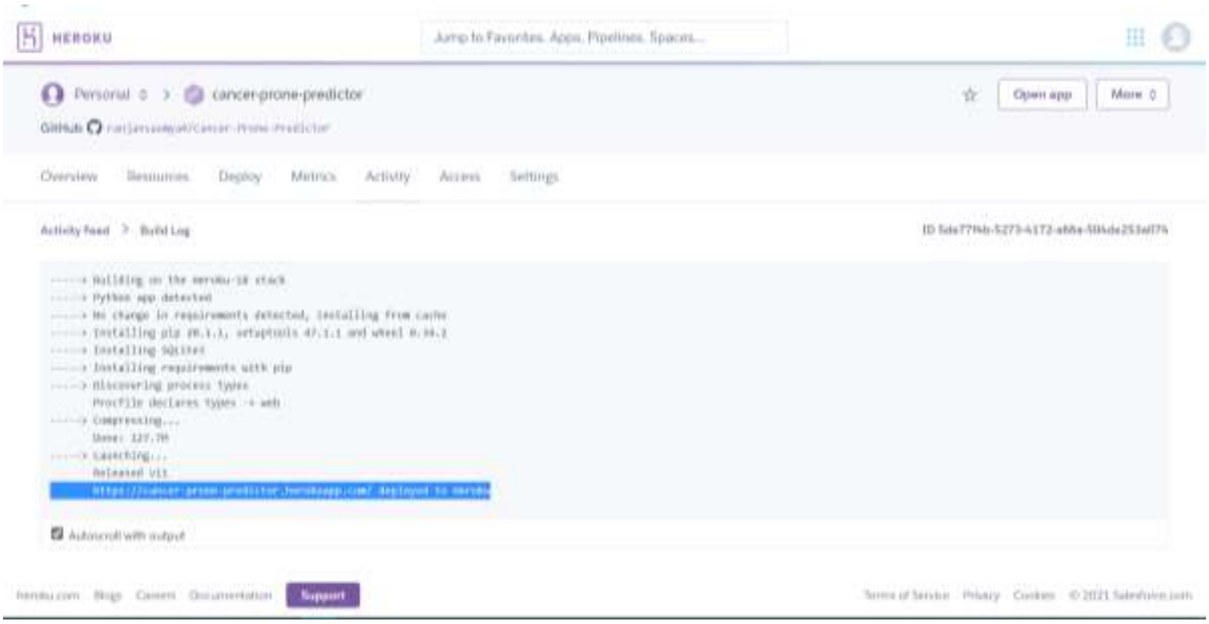
```
web: gunicorn app:app
```

IMPLEMENTATION

After building these codes and deploying the master branch in the Heroku app we have successfully implemented our model and now can use this application.

We can browse our application on the url provided by Heroku

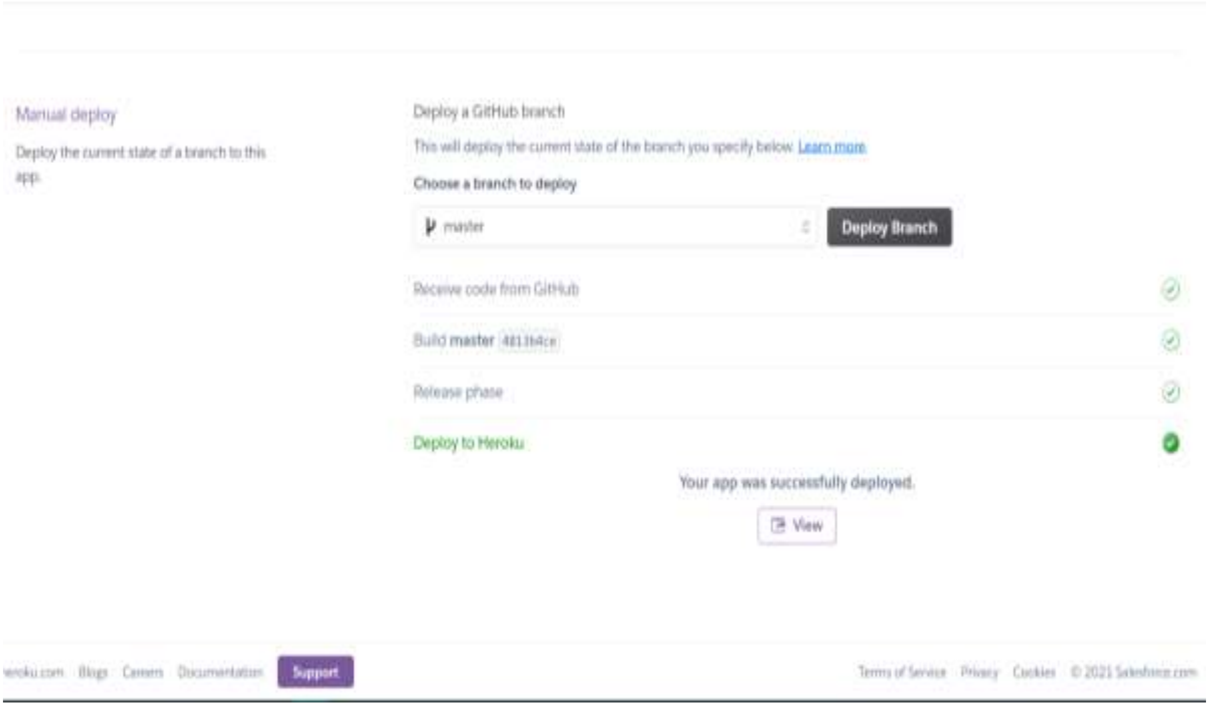
https://cancer-prone-predictor.herokuapp.com/



ID 5de77f4b-5273-4172-a66a-5f4de253af74

```
-----> Building on the heroku-18 stack
-----> Python app detected
-----> No change in requirements detected, installing from cache
-----> Installing pip 20.1.1, setuptools 47.1.1 and wheel 0.34.2
-----> Installing SQLite3
-----> Installing requirements with pip
-----> Discovering process types
       Procfile declares types -> web
-----> Compressing...
       Done: 127.7M
-----> Launching...
       Released v11
       https://cancer-prone-predictor.herokuapp.com/ deployed to Heroku
```

☑ Autoscroll with output


Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

This will deploy the current state of the branch you specify below. Learn more

Choose a branch to deploy

master

Receive code from GitHub ✓

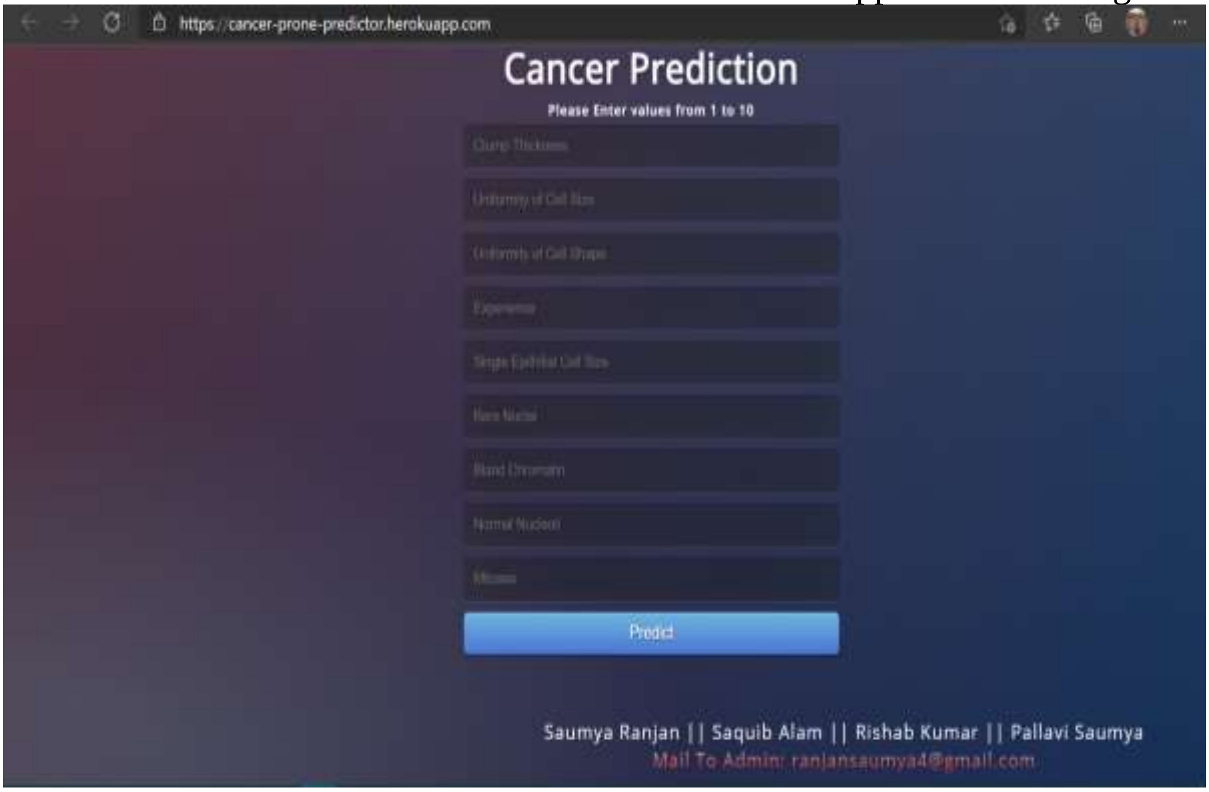
Build master 481364ce ✓

Release phase ✓

Deploy to Heroku ●

Your app was successfully deployed.

Now when we browse our url then we can see our Web application running.



https://cancer-prone-predictor.herokuapp.com

Cancer Prediction

Please Enter values from 1 to 10

Clump Thickness

Uniformity of Cell Size

Uniformity of Cell Shape

Experience

Single Epithelial Cell Size

Bare Nuclei

Bland Chromatin

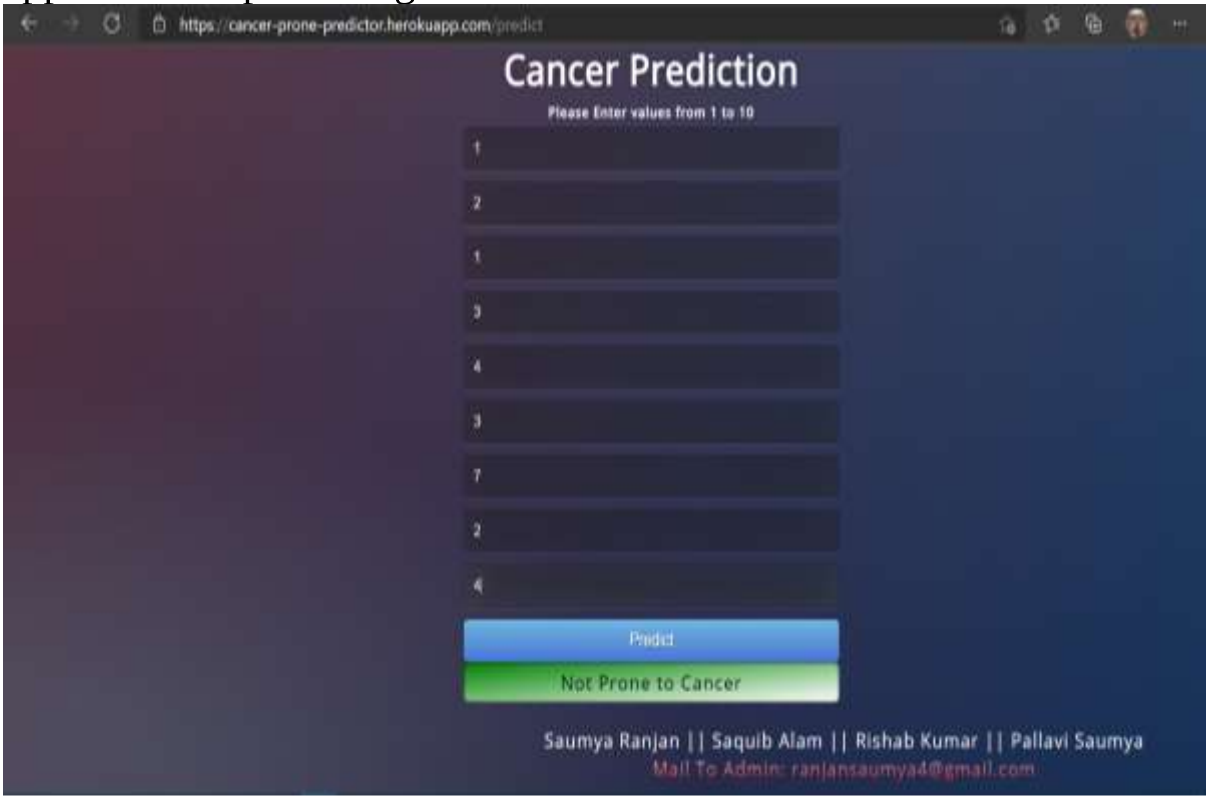
Normal Nucleoli

Mitoses

Saumya Ranjan || Saquib Alam || Rishab Kumar || Pallavi Saumya

Mail To Admin: ranjansaumya4@gmail.com

When we enter the values for every field then we can see that our application is predicting the result.



https://cancer-prone-predictor.herokuapp.com/predict

Cancer Prediction

Please Enter values from 1 to 10

1

2

1

3

4

3

7

2

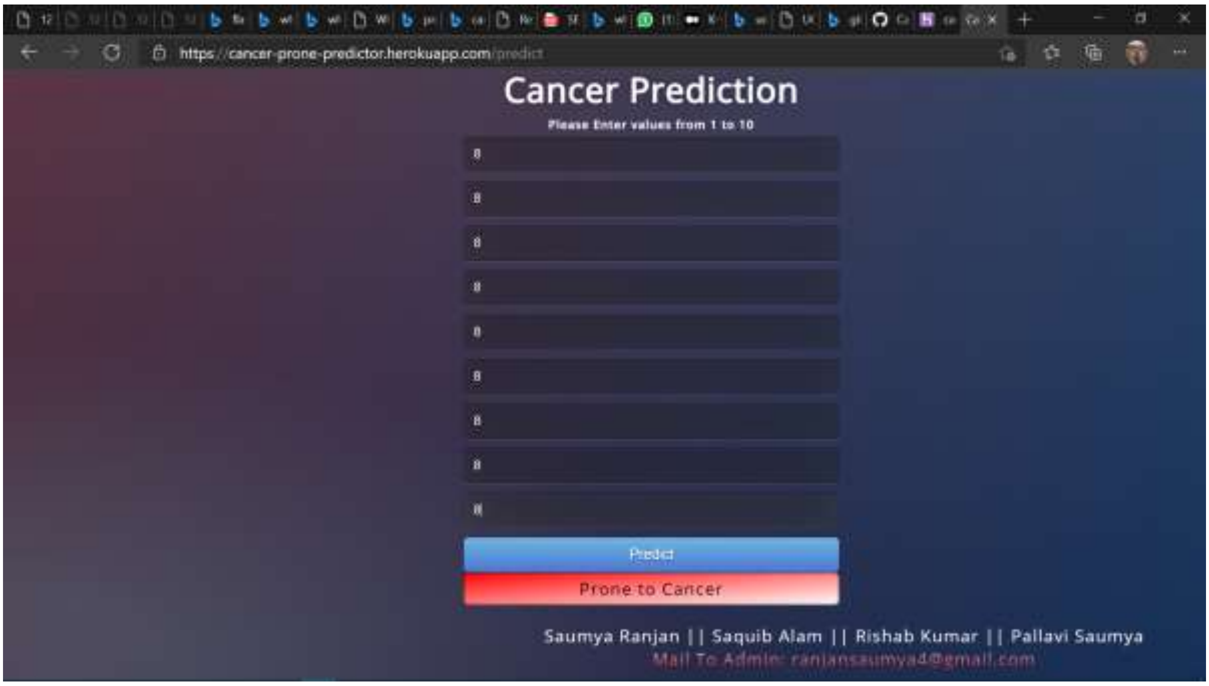
4

Not Prone to Cancer

Saumya Ranjan || Saquib Alam || Rishab Kumar || Pallavi Saumya

Mail To Admin: ranjansaumya4@gmail.com

Result showing “PRONE TO CANCER”


https://cancer-prone-predictor.herokuapp.com/predict

Cancer Prediction

Please Enter values from 1 to 10

8

8

8

8

8

8

8

8

8

Prone to Cancer

Saumya Ranjan || Saquib Alam || Rishab Kumar || Pallavi Saumya

Mail To Admin: ranjansaumya4@gmail.com

Result Showing “ NOT PRONE TO CANCER ”

CONCLUSION

Cancer is a critical disease which leads to death if not diagnosed at early stage. So, this project can be useful in reducing the risk factor by detecting cancer in patients at early stage so that their lives can be saved after treatment.

Specifically, we identified a number of trends with respect to the types of machine learning methods being used, the types of training data being integrated, the kinds of endpoint predictions being made, and the overall performance of these methods in predicting cancer susceptibility or outcomes. Given, the growing trend on the application of ML methods in cancer research, we have tried to present here a project as an aim to model cancer risk of patients.

We have used decision tree algorithm. Also, we have deployed our model and integrated it in a webpage using Flask framework. So, the end-product of out project is a webpage where a user can give input factors predicting cancer and eventually get the prediction result.

Overall, we believe that if the quality of studies continues to improve, it is likely that the use of machine learning classifier will become much more common-place in many clinical and hospital settings.

References:

- https://github.com/ranjansaumya4/Cancer-Prone-Predictor
- https://www.researchgate.net/publication/348862857_Cancer_prediction_using_machine_learning
- https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29