**Hospital Readmission Predictor Using IBM Watson**

**Submitted by:**

Priyal Thomas

Arjun k

TABLE OF CONTENTS

**1.INTRODUCTION………………………………………………………………………………...…3**

[1.1 OVERVIEW 3](#_Toc119231855)

[1.2 MOTIVATION 3](#_Toc119231856)

[1.2. PURPOSE 3](#_Toc119231857)

[2. LITERATURE SURVEY 5](#_Toc119231858)

[2.1 EXISTING PROBLEM 5](#_Toc119231859)

[2.2 PURPOSED SOLUTION 5](#_Toc119231860)

[3. THEORETICAL ANALYSIS 6](#_Toc119231861)

[3.1. Flow-Chart: 6](#_Toc119231862)

[3.2. Hardware/Software designs: 6](#_Toc119231863)

[4. EXPERIMENTAL INVESTIGATION 7](#_Toc119231864)

[5. FLOWCHART 8](#_Toc119231865)

[PROPOSED SYSTEM 9](#_Toc119231866)

[**6. EXPERIMENTAL ANALYSIS** 12](#_Toc119231867)

[7. RESULT 16](#_Toc119231868)

[Building Html Pages 16](#_Toc119231869)

[8. ADVANTAGES AND DISADVANTAGES 19](#_Toc119231870)

[9. APPLICATIONS: 19](#_Toc119231871)

[10. CONCLUSION 20](#_Toc119231872)

[11. FUTURESCOPE 20](#_Toc119231873)

[12. BIBILOGRAPHY 20](#_Toc119231874)

[**APPENDIX** 21](#_Toc119231875)

[A. Source Code 21](#_Toc119231876)

d

**1. INTRODUCTION**

## 1.1 OVERVIEW

As the healthcare system moves toward value-based care, CMS has created many programs to improve the quality of care of patients. One of these programs is called the Hospital Readmission Reduction Program ([HRRP](https://www.cms.gov/Medicare/Quality-Initiatives-Patient-Assessment-Instruments/Value-Based-Programs/HRRP/Hospital-Readmission-Reduction-Program.html)), which reduces reimbursement to hospitals with above average readmissions. For those hospitals which are currently penalized under this program, one solution is to create interventions to provide additional assistance to patients with increased risk of readmission. But how do we identify these patients? We can use predictive modeling from data science to help prioritize patients.

## 

## 1.2. PURPOSE

If a hospital has multiple readmissions, it means that the hospital needs to work on the quality of services it is providing with respect to the health and wellness of its patients.  Being able to predict whether a person will be readmitted to the hospital within 30 days or not, will be of great help to the hospital in developing an idea of the incoming number of repeated patients which in turn helps to provide better services for patients with increased risk of disease. One patient population that is at increased risk of hospitalization and readmission is diabetes. Diabetes is a medical condition that affects approximately 1 in 10 patients in the United States.  So in this project, we will be focusing on hospital readmission prediction for patients who are having diabetes.

This study used the Health Facts database (Cerner Corporation, Kansas City, MO), a national data warehouse that collects comprehensive clinical records across hospitals throughout the United States.  The Health Facts data we used was an extract representing 10 years (1999–2008) of clinical care at 130 hospitals and integrated delivery networks throughout the United States.The main purpose of this project is to predict whether a person who is suffering from diabetes and consulting a specific hospital will be readmitted or not, based on multiple factors.We will be using classification algorithms such as Logistic Regression, KNN, Decision tree, Random Forest, AdaBoost, and GradientBoost. We will train and test the data with these algorithms. From this, the best model is selected and saved in pkl format. We will also be deploying our model locally using Flask.

# 

# **2. LITERATURE SURVEY**

## 

## 2.1 EXISTING PROBLEM

**To complete this project, you should have the following software’s and packages Anaconda Navigator:**

Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS.Conda is an open-source, cross-platform, package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder.

## 2.2 PURPOSED SOLUTION

The method or solution is Jupiter notebook and spyder we used to complete this project.

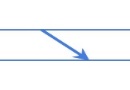
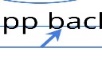
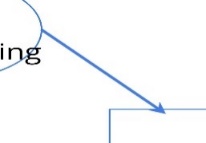
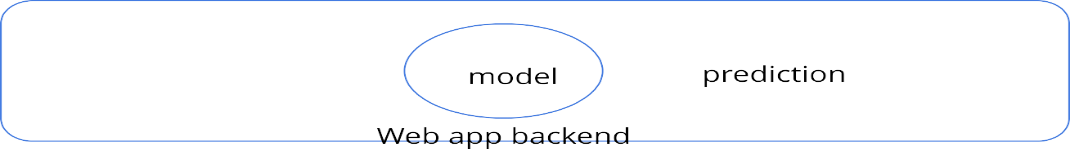
and you will use this jupiter notebook for you recommended.

**To build Machine learning models you must require the following packages**

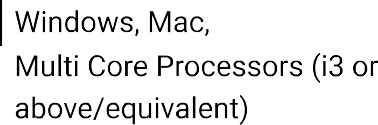
* **Sklearn:** Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms.
* **NumPy:** NumPy is a Python package that stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object
* **Pandas:** pandas is a fast, powerful, flexible, and easy to use open source data analysis and manipulation tool,built on top of the Python programming language.

# **3. THEORETICAL ANALYSIS**

## 3.1. Flow-Chart:



## 3.2. Hardware/Software designs:



software requirements:



# 

# **4. EXPERIMENTAL INVESTIGATION**

**Importing the Libraries**

First step is usually importing the libraries that will be needed in the program.

**Pandas:** It is a python library mainly used for data manipulation.

**NumPy:** This python library is used for numerical analysis.

**Matplotlib and Seaborn:** Both are the data visualization library used for plotting graph which will help us for understanding the data.

**csr\_matrix() :**A dense matrix stored in a NumPy array can be converted into a sparse matrix using the CSR representation by calling the csr\_matrix() function.

**Train\_test\_split:** used for splitting data arrays into training data and for testing data.

**Pickle:** to serialize your machine learning algorithms and save the serialized format to a file.

**Reading the Dataset:**

. The next step is to read the dataset into a data structure that’s compatible with pandas. Let’s load a .csv data file into pandas. There is a function for it, called **read\_csv().**We will need to locate the directory of the CSV file at first (it’s more efficient to keep the dataset in the same directory as your program).If the dataset in same directory of your program, you can directly read it, without any path. After the next Steps we made following bellow:

1.Data visualization

2.Collabrative analysis

3.Creating the Model

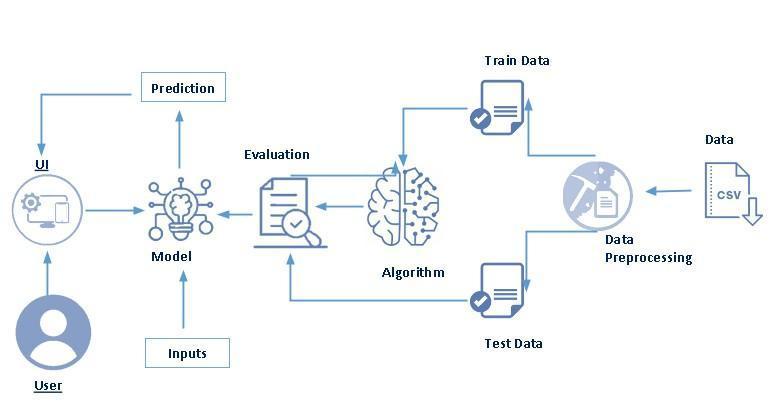
4.Test and save the model

5.Build Python Code

6.Build HTML Code

7.Run the Application

# **5. FLOWCHART**



**Project Flow:**

* User interacts with the UI (User Interface) to upload the input features.
* Uploaded features/input is analysed by the model which is integrated
* Once model analyses the uploaded inputs, the prediction is showcased on the UI.

1. **Data Collection.** 
   * Collect the dataset or create the dataset
2. **Data Pre- processing.** 
   * Import the Libraries.
   * Importing the dataset.
   * Exploratory Data Analysis
   * Data Visualization
3. **Collaborating Filtering** 
   * Merging datasets
   * Creating the Model
   * Predicting the results
   * Saving our model and dataset
4. **Application Building** 
   * Create an HTML file
   * Build a Python Code

PROPOSED SYSTEM:

Here we are building a model by applying various machine learning algorithms find the best accurate model.

In this section, we will first compare the performance of the following 7 machine learning models using default hyperparameters:

* K-nearest neighbours
* Logistic regression
* Stochastic gradient descent
* Naive Bayes
* Decision tree
* Random forest
* Gradient boosting classifier

K nearest neighbours (KNN):

KNN is one the simplest machine learning models. For a given sample point, the model looks at the k closest datapoints and determines the probability by counting the number of positive labels divided by K. This model is easy to implement and understand, but comes at the disadvantage of being sensitivity to K and takes a long time to evaluate if the number of trained samples is large. We can fit KNN using the following code from scikit-learn.

* The K-NN working can be explained on the basis of the below algorithm:

**Step-1:** Select the number K of the neighbors

**Step-2:** Calculate the Euclidean distance of **K number of neighbors**

**Step-3:** Take the K nearest neighbors as per the calculated Euclidean distance.

**Step-4:** Among these k neighbors, count the number of the data points in each category.

**Step-5:** Assign the new data points to that category for which the number of the neighbor is maximum.

**Step-6:** Our model is ready.

Logistic Regression:

Logistic regression is a traditional machine learning model that fits a linear decision boundary between the positive and negative samples. This linear function is then passed through a sigmoid function to calculate the probability of the positive class. Logistic regression is an excellent model to use when the features are linearly separable. One advantage of logistic regression is the model is interpretable — i.e. we know which features are important for predicting positive or negative.

Stochastic gradient descent

Stochastic gradient descent is similar to logistic regression. Both methods use gradient descent to optimize the coefficients of a linear function. In logistic regression, all data samples are used at each iteration whereas in stochastic gradient descent only a small batch of samples is used. This allows stochastic gradient descent to speed up training. We can fit stochastic gradient descent using the following code from scikit-learn.

Naive Bayes

Naive Bayes is another model occasionally used in machine learning. In Naive Bayes, we utilize [Bayes Rule](https://en.wikipedia.org/wiki/Bayes%27_theorem) to calculate the probabilities. The “naive” part of this model is that it assumes all the features are independent (which is generally not the case). This works well for natural language processing models, but let’s try it out here anyways. We can fit Naive Bayes with the following code.

Decision tree

Another class of popular machine learning models is tree-based methods. The simplest tree-based method is known as a decision tree. Essentially, in tree methods you utilize the methodology behind the game 20 questions to continue to divide your samples.

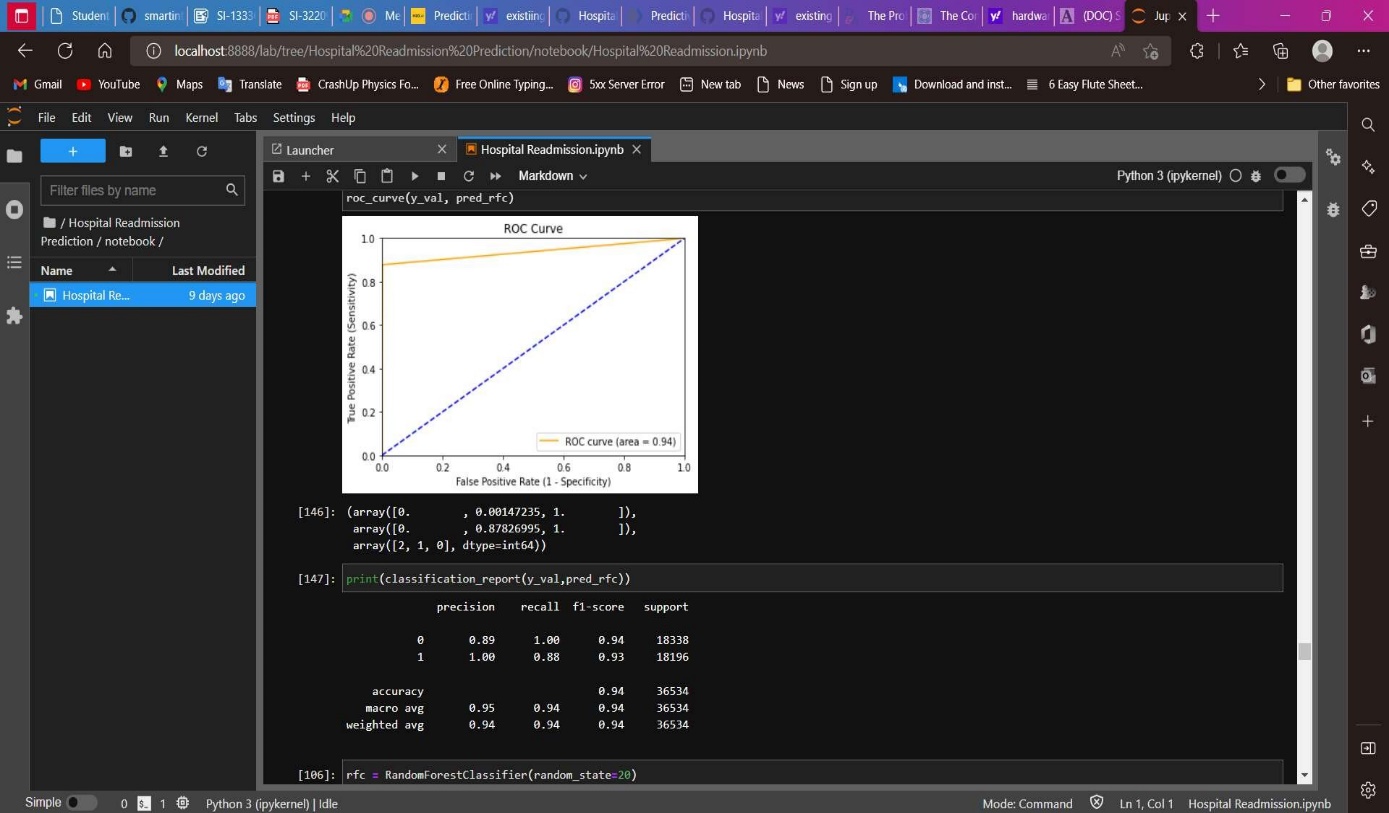
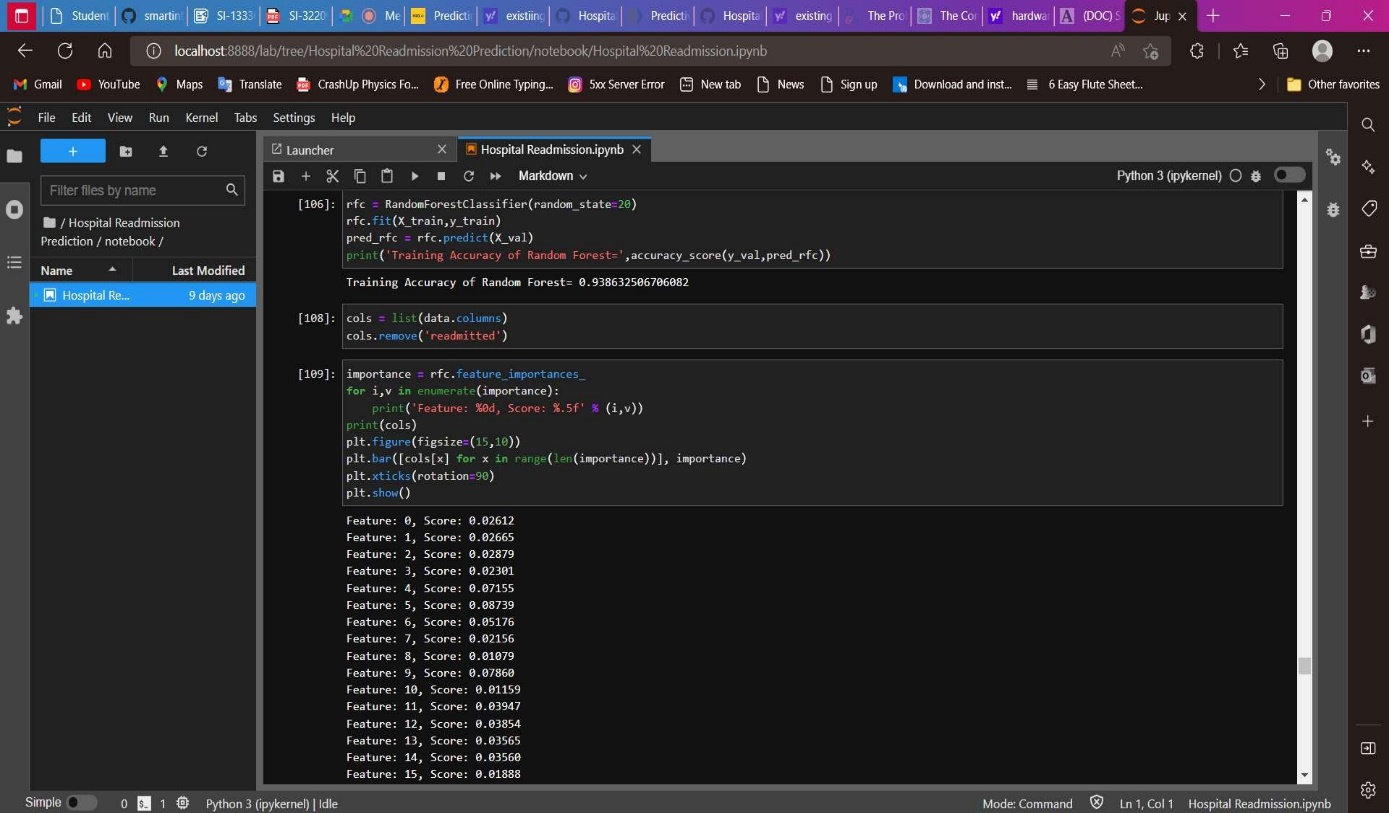
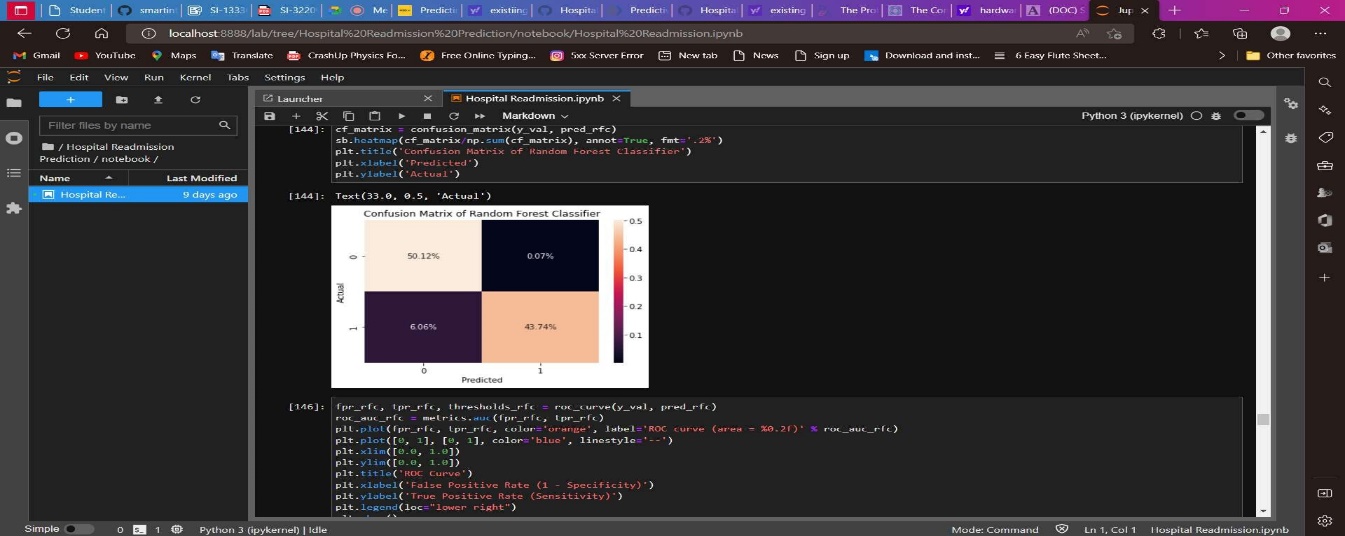
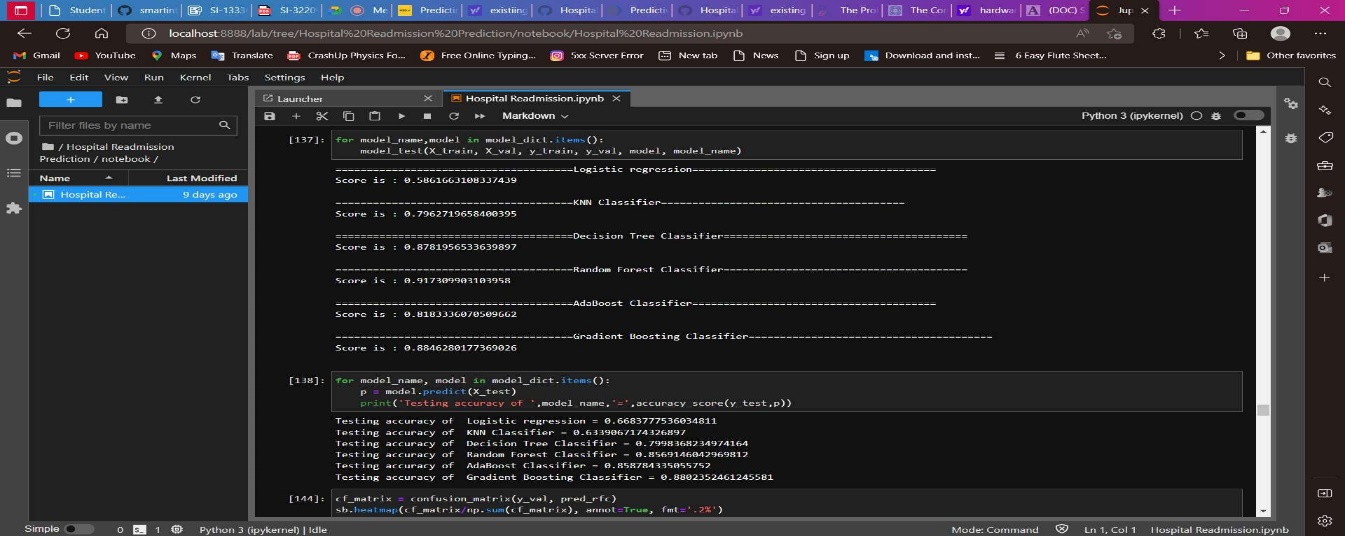
Random forest

One disadvantage of decision trees is that they tend overfit very easily by memorizing the training data. As a result, random forests were created to reduce the overfitting. In random forest models, multiple trees are created and the results are aggregated.

Gradient boosting classifier

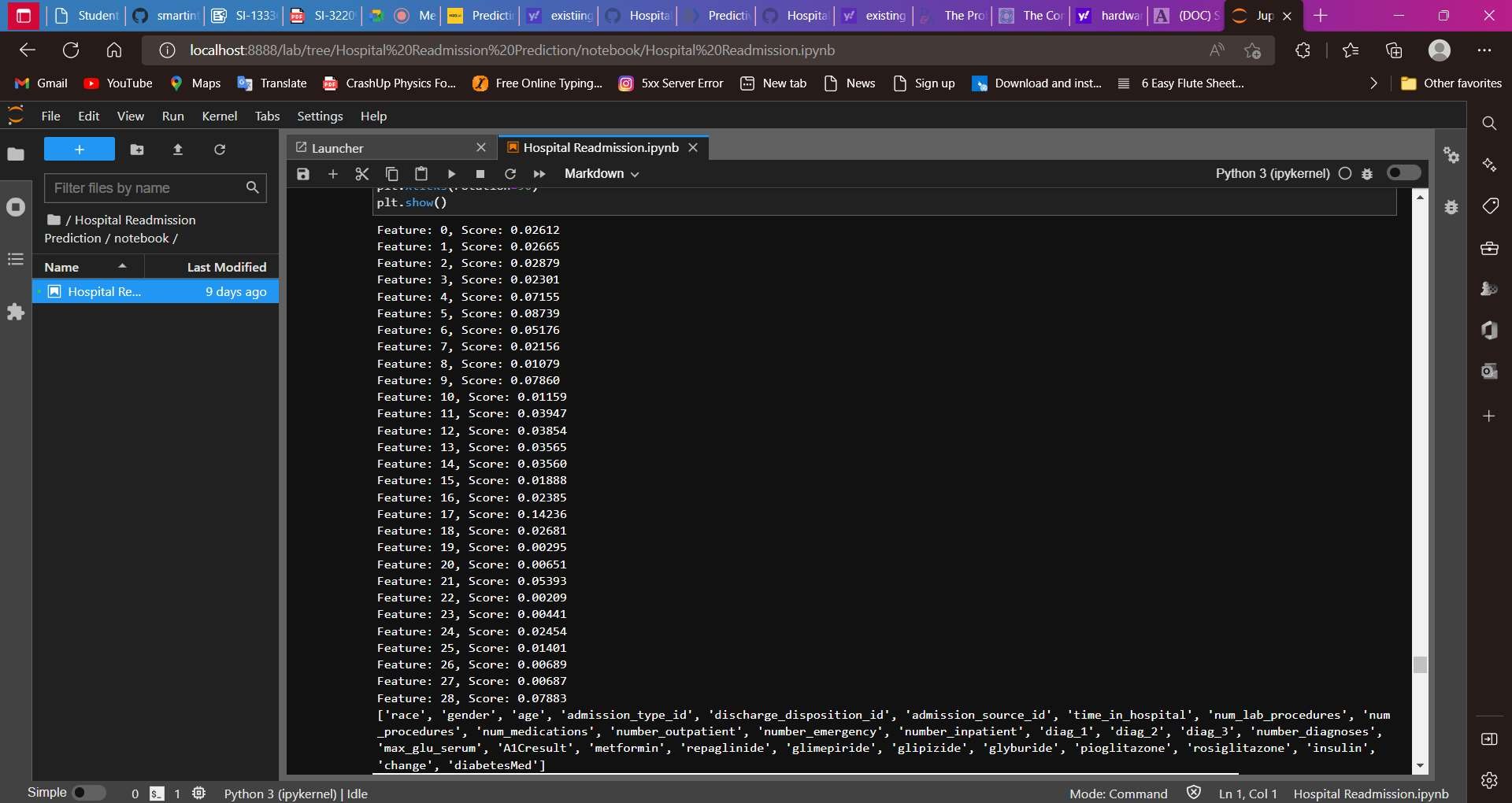
Another approach to improving decision trees is using a technique called boosting. In this method, you create a bunch of shallow trees that try to improve on the errors of the previously trained trees.

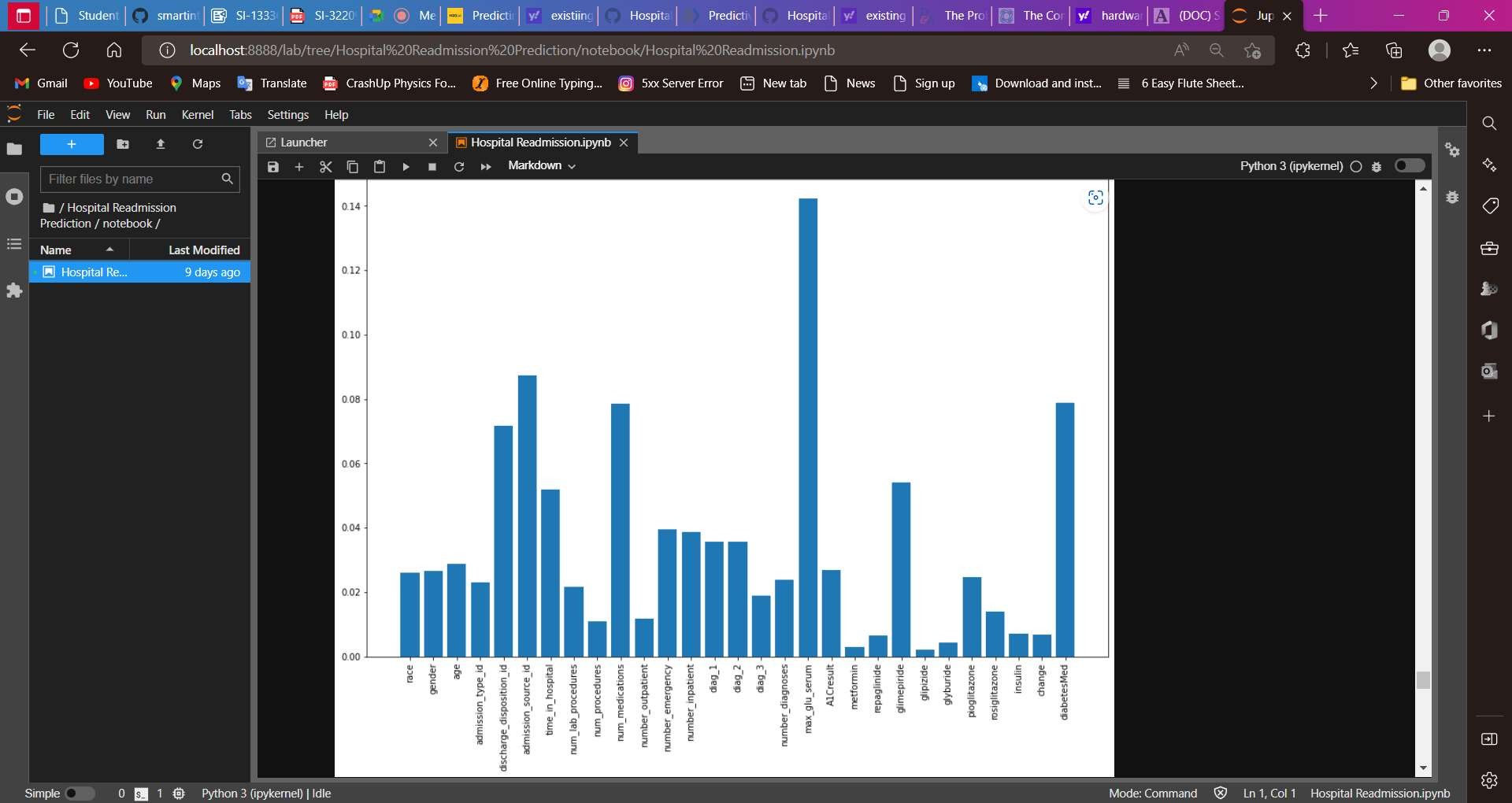
## **6. EXPERIMENTAL ANALYSIS**



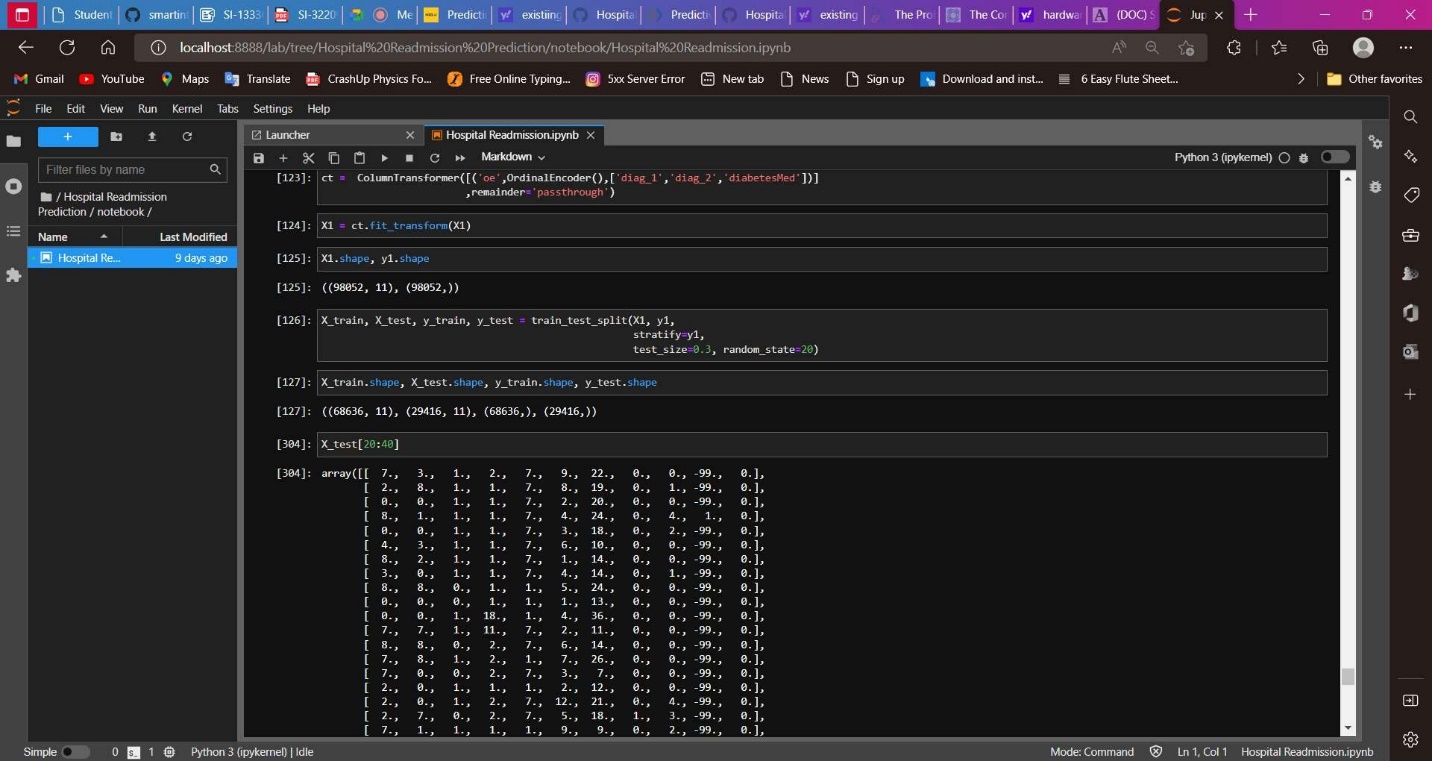
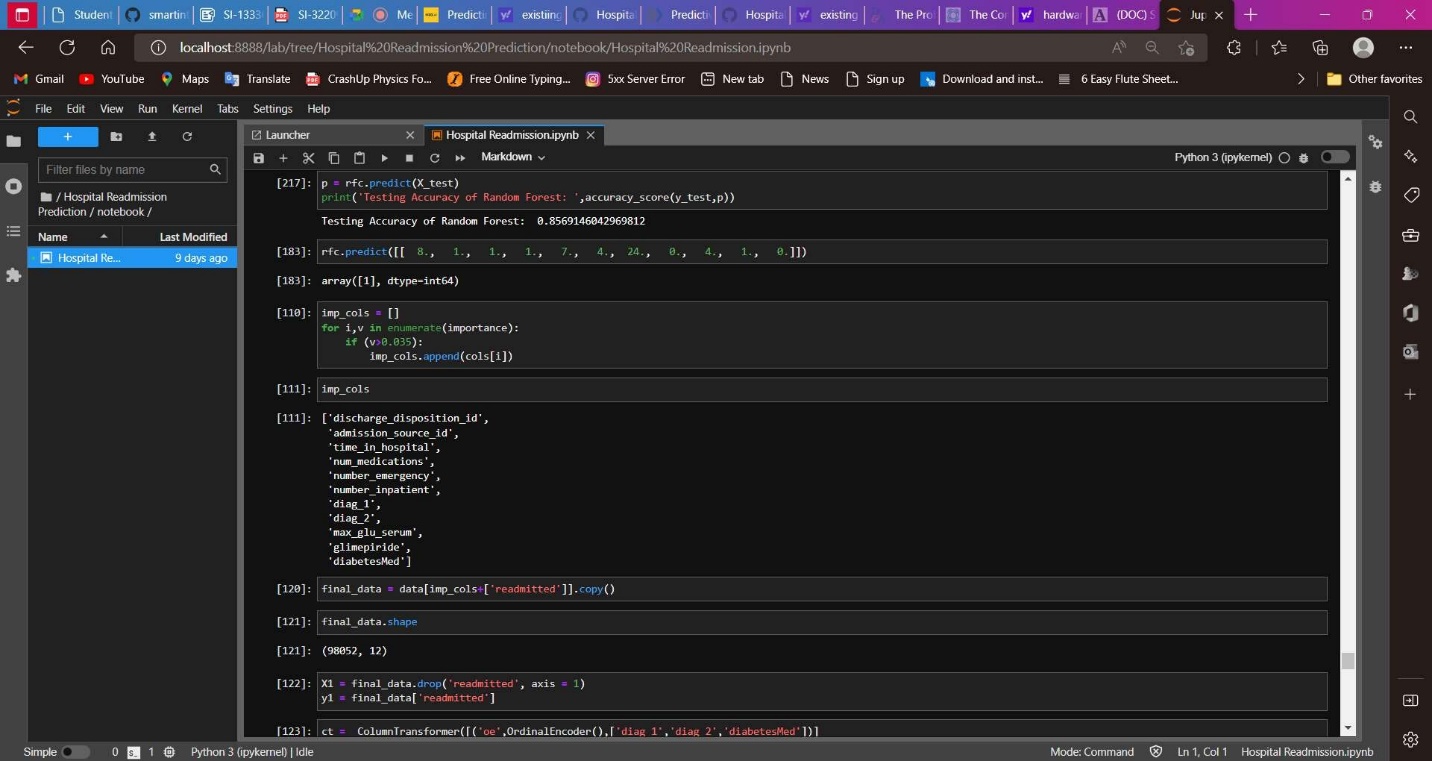
We will compare the confusion matrix, ROC curve and classification report for both models.

In order to obtain these, we will be using the confusion\_matrix(),roc\_curve() and classification\_report() functions from sklearn.metrics.





We have trained our model with 29 features.  But all these features may not be important for prediction.  Hence, we will select the features that contribute significantly to the model performance.



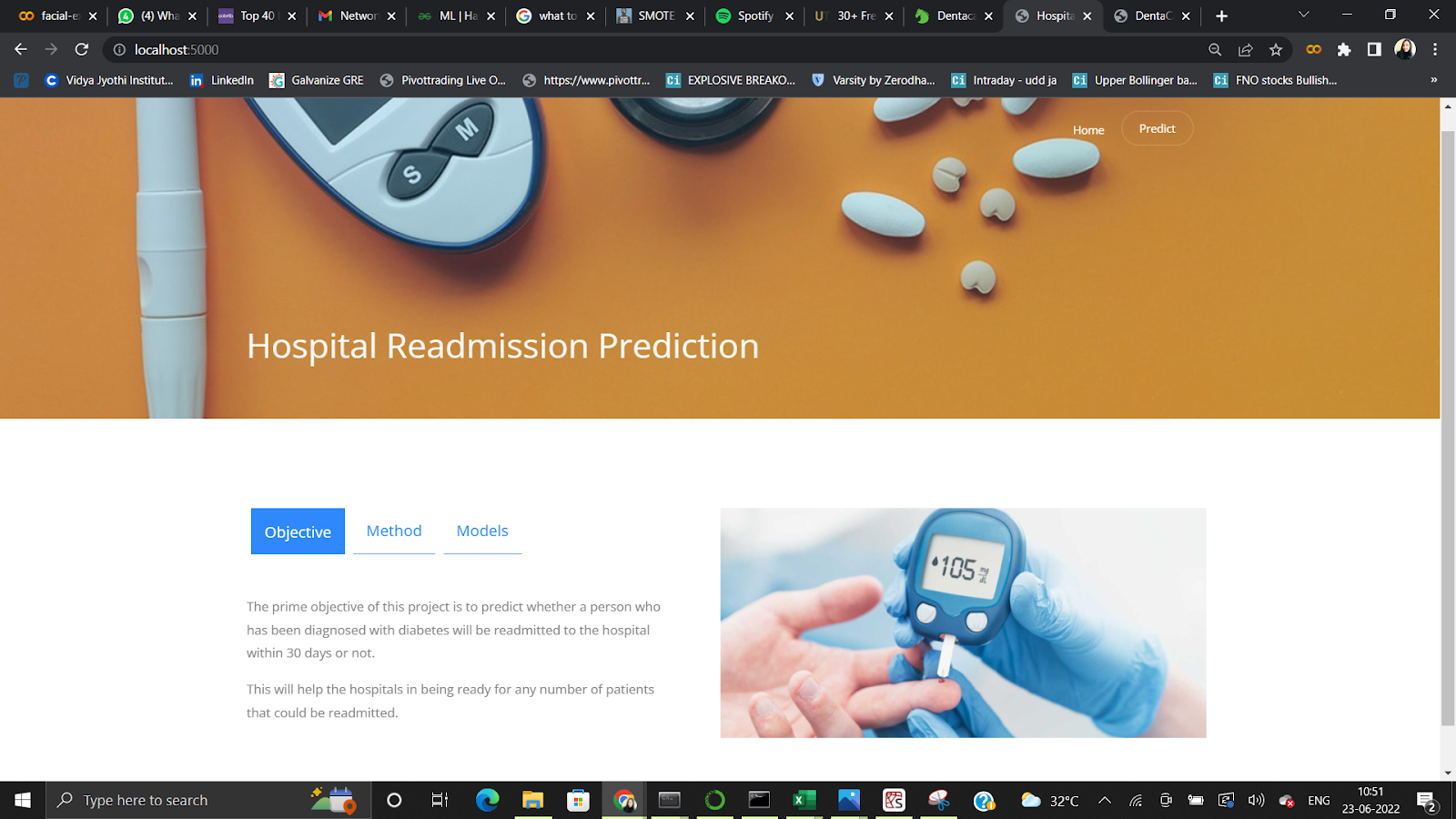
# **7. RESULT**

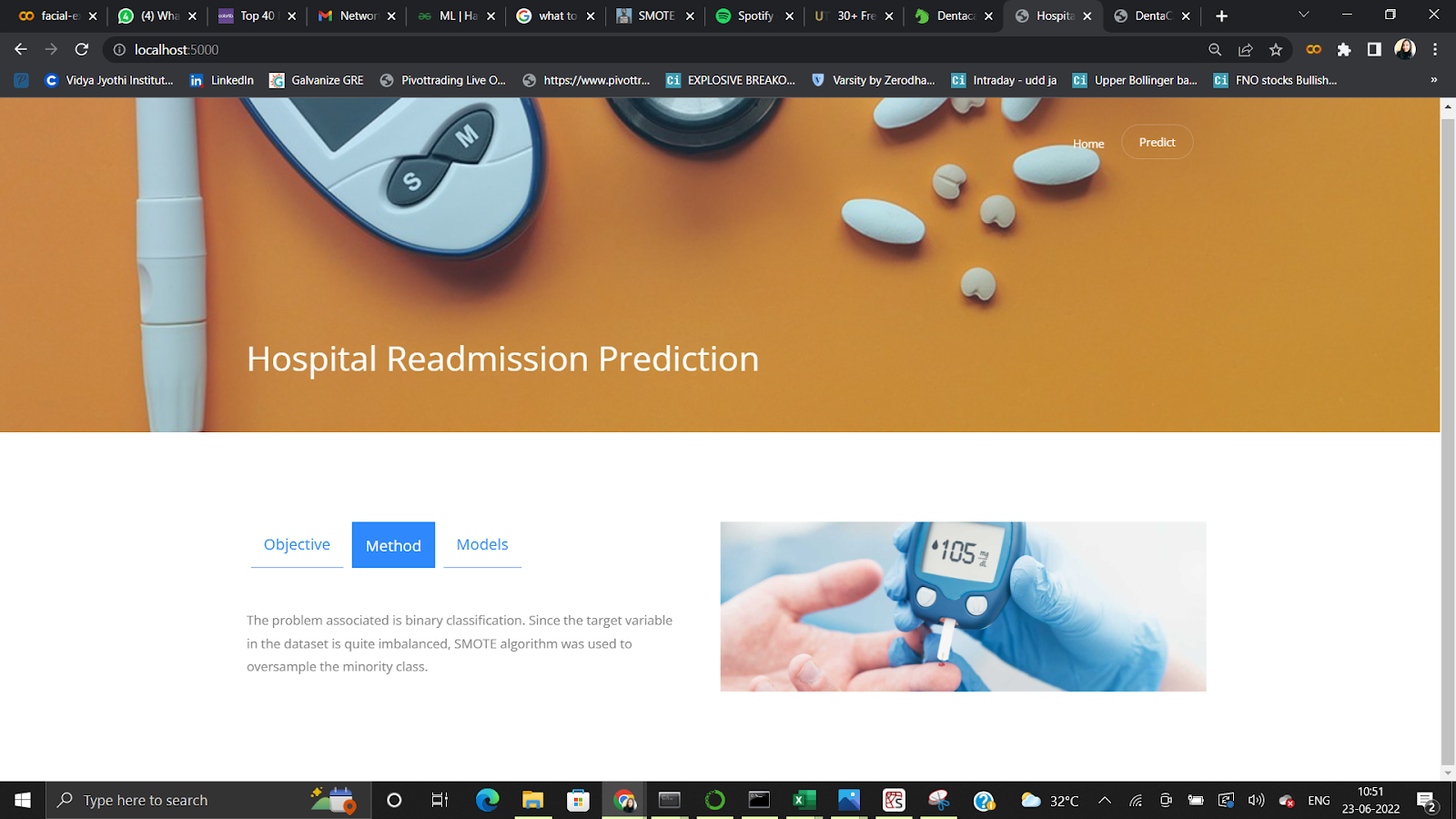
## Building Html Pages

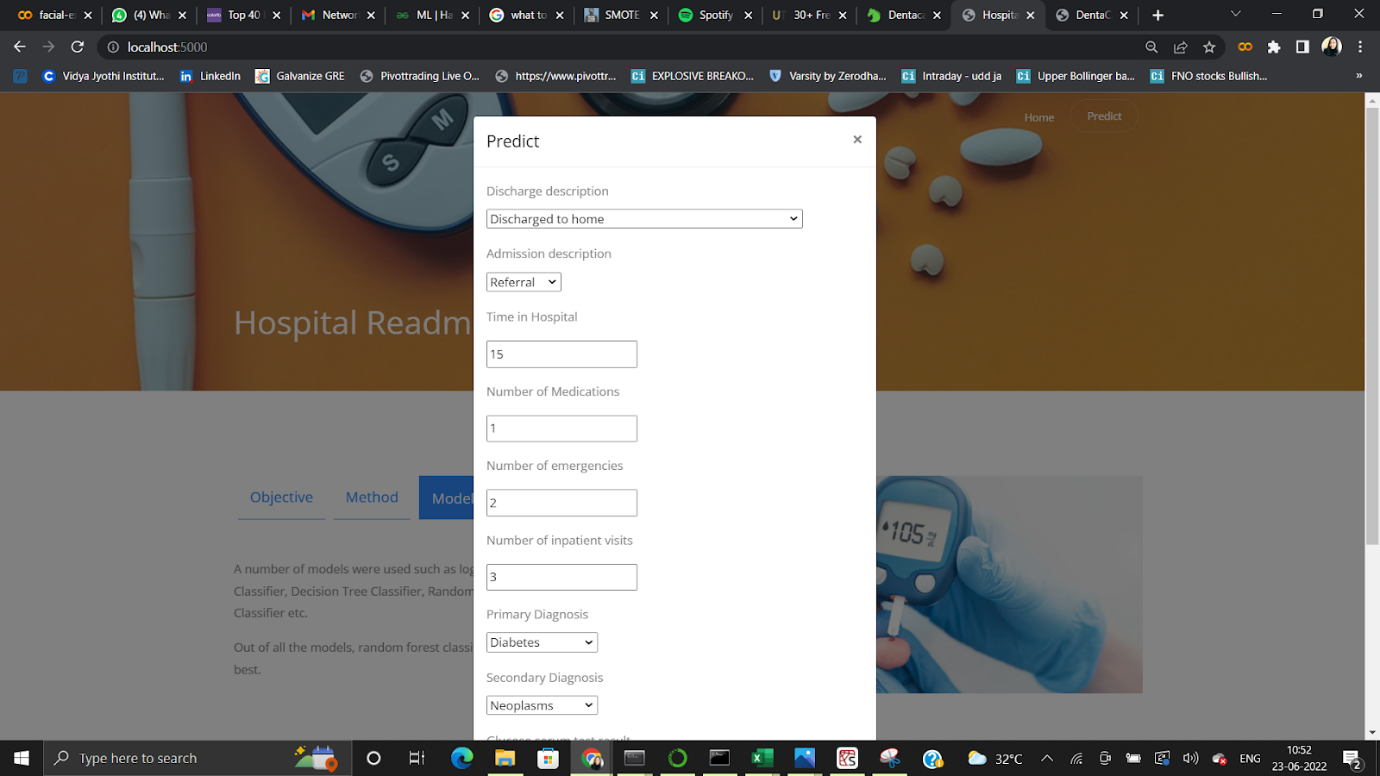
For this project create three HTML files namely

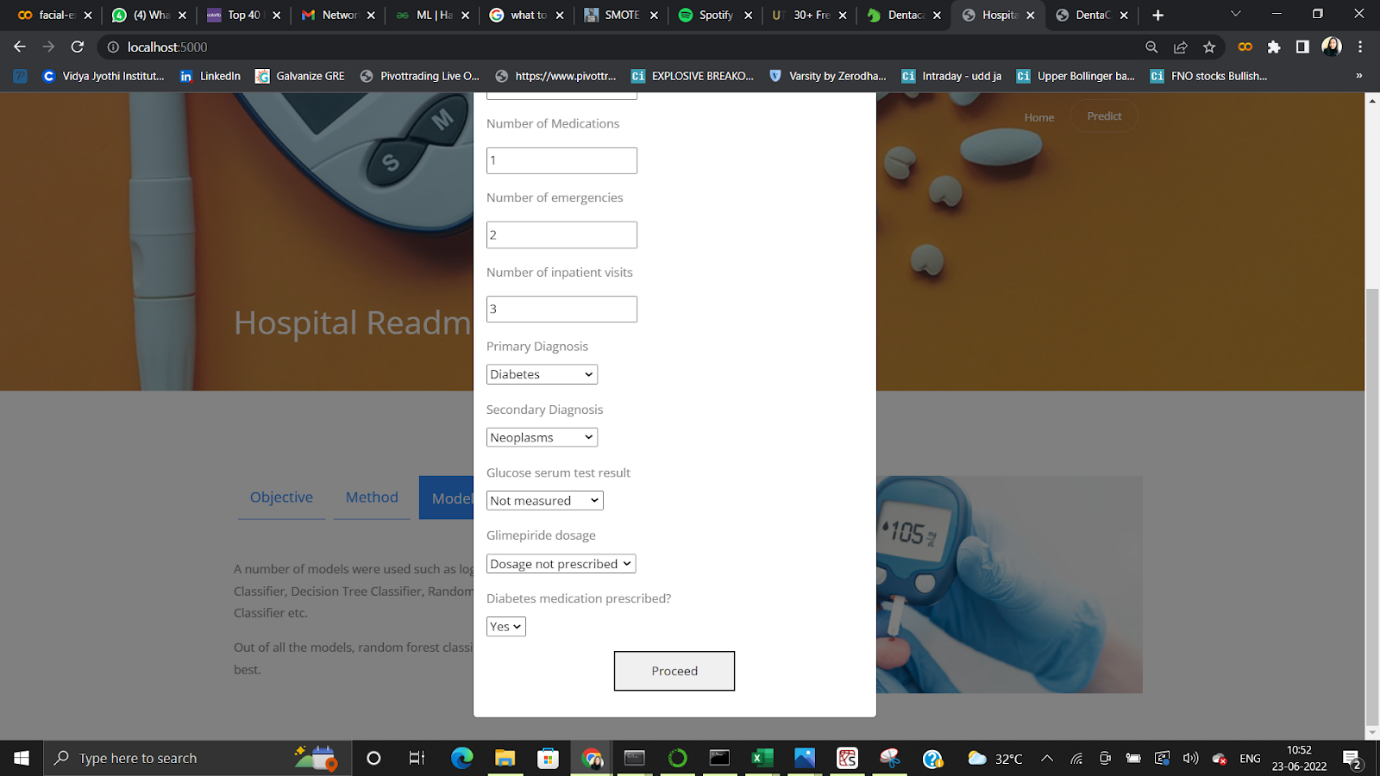
* home.html
* index.html
* output.html

Let’s see how our home.html page looks like:

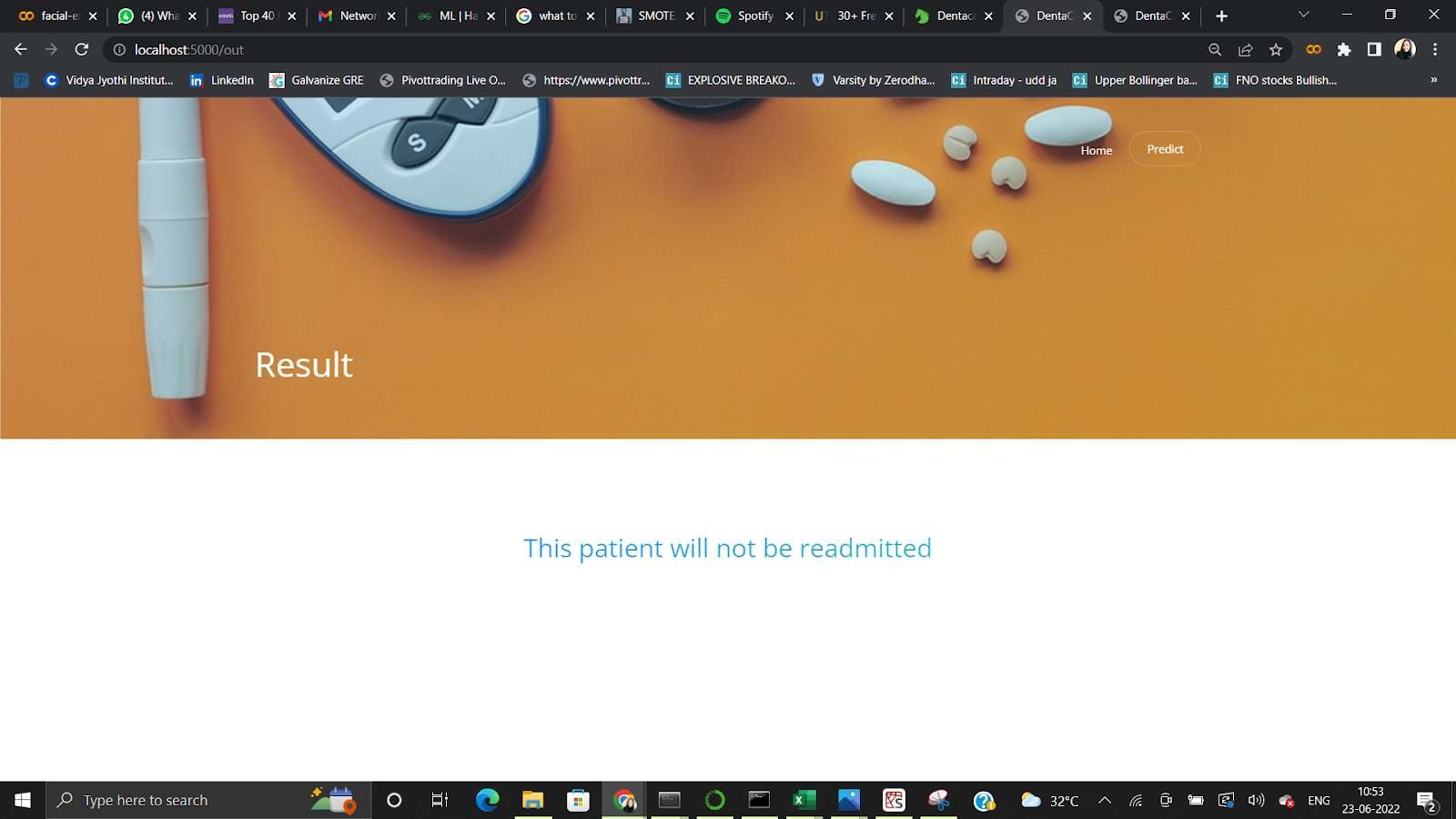
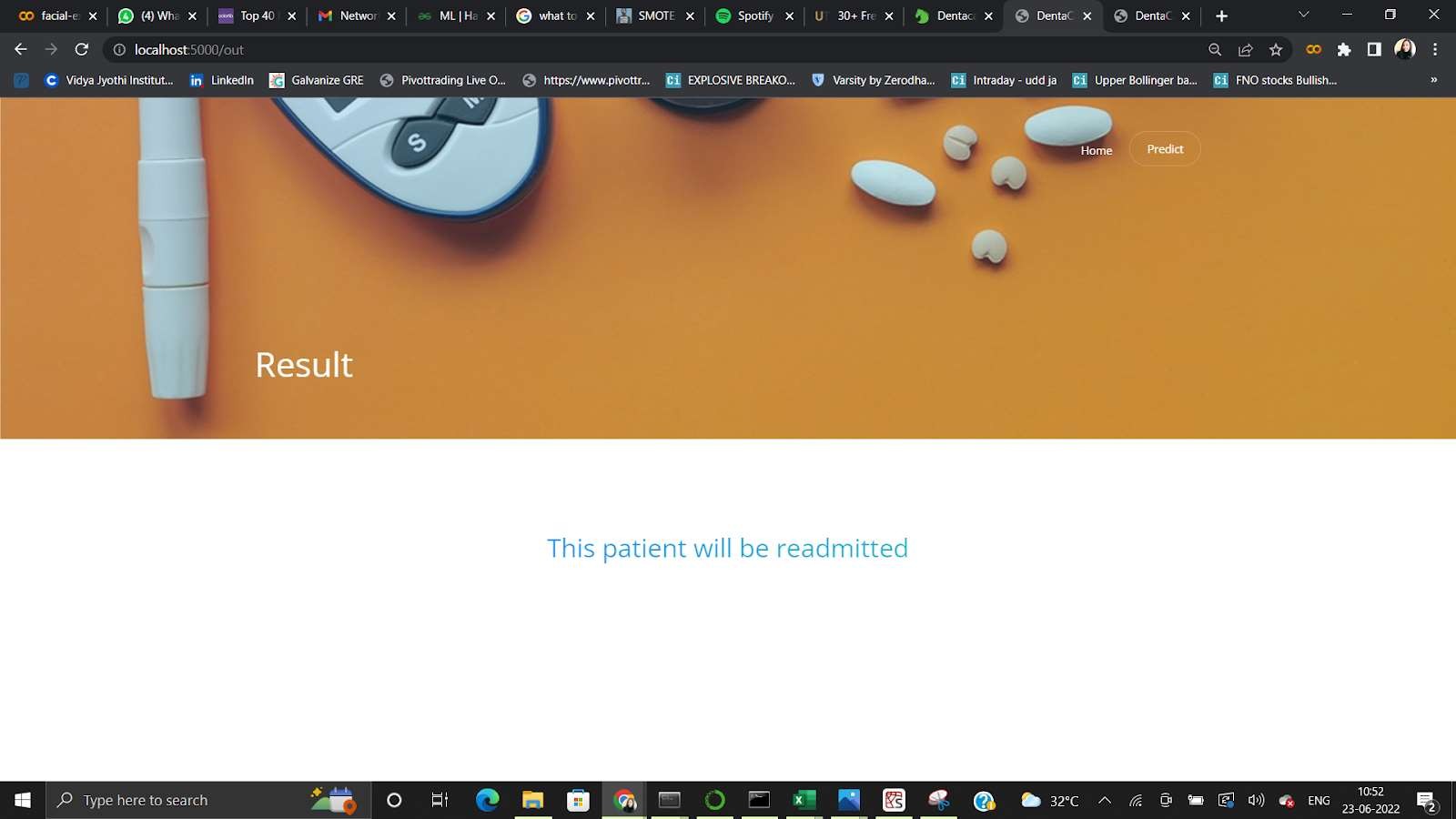








Now when you click on proceed button you will get redirected to output.html



# **8. ADVANTAGES AND DISADVANTAGES**

**ADVANTAGES:**

1. The Hospital will be able to ready for any no of patients that will be Readmitted in the feature.

**DISADVANTAGES:**

1. The accuracy is only 91.73 percentage.
2. It will not be able to predict 100% accurately.

# **9. APPLICATIONS:**

1. It can be used by clinics related to hospitals.
2. It can be used by diabetes hospitals.
3. It can be used by diabetic patients.

# **10. CONCLUSION**

* Through this project, we created a machine learning model that is able to predict the patients with diabetes with highest risk of being readmitted within 30 days.
* The best model was a gradient boosting classifier with optimized hyperparameters. The model was able to catch 58% of the readmissions and is about 1.5 times better than just randomly picking patients.

# **11. FUTURESCOPE**

One path for improving your models to understand what features are important to your models. This can usually only be investigated for simpler models such as Logistic Regression or Random Forests. This analysis can help in a few areas:  
  
— inspire new feature ideas → helps with both high bias and high variance  
— obtain a list of the top features to be used for feature reduction → helps with high variance  
— point out errors in your pipeline → helps with robustness of model

# **12. BIBILOGRAPHY**

**1) Mayo clinic staff: Hyperglycemia in diabetes causes - Mayo clinic** (2015)

1. Umpierrez G.E., Isaacs S.D., Bazargan N., You X., Thaler L.M., Kitabchi A.E.

**Hyperglycemia: An independent marker of in-hospital mortality in patients with undiagnosed diabetes**

1. J. Clin. Endocrinol. Metab., 87 (2002), pp. 978-982

Strack B., Deshazo J.P., Gennings C., Olmo J.L., Ventura S., Cios K.J., Clore J.N.

**Impact of HbA1c measurement on hospital readmission rates: Analysis of 70,000 clinical database patient records**

1. Wexler J., Pushkarna M., Bolukbasi T., Wattenberg M., Viégas F., Wilson J.

**The what-if tool: Interactive probing of machine learning models**

IEEE Trans. Vis. Comput. Graphics, 26 (2019), pp. 56-65

## **APPENDIX**

### A. Source Code

from flask import Flask, render\_template, request

import pickle, joblib

import pandas as pd

app = Flask(\_\_name\_\_)

model = pickle.load(open("model.pkl","rb"))

ct = joblib.load('feature\_values')

print('Model loaded. Check http://127.0.0.1:5000/')

@app.route('/')

def home():

return render\_template("home.html")

@app.route('/pred')

def predict():

return render\_template("index.html")

@app.route('/out', methods =["POST"])

def output():

discharge\_disposition\_id = request.form["discharge\_disposition\_id"]

admission\_source\_id = request.form["admission\_source\_id"]

time\_in\_hospital = request.form["time\_in\_hospital"]

num\_medications = request.form["num\_medications"]

number\_emergency = request.form["number\_emergency"]

number\_inpatient = request.form["number\_inpatient"]

diag\_1 = request.form["diag\_1"]

diag\_2 = request.form["diag\_2"]

max\_glu\_serum = request.form["max\_glu\_serum"]

glimepiride = request.form["glimepiride"]

diabetesMed = request.form["diabetesMed"]

if max\_glu\_serum == '>200' or max\_glu\_serum=='>300':

max\_glu\_serum=1

elif max\_glu\_serum=='Norm':

max\_glu\_serum=0

else:

max\_glu\_serum=-99

if glimepiride == 'No':

glimepiride = 0

else:

glimepiride=1

data = [[discharge\_disposition\_id,admission\_source\_id,time\_in\_hospital,

num\_medications, number\_emergency, number\_inpatient,diag\_1, diag\_2,

max\_glu\_serum, glimepiride, diabetesMed]]

feature\_cols = ['discharge\_disposition\_id', 'admission\_source\_id', 'time\_in\_hospital',

'num\_medications','number\_emergency', 'number\_inpatient',

'diag\_1', 'diag\_2','max\_glu\_serum', 'glimepiride', 'diabetesMed']

pred = model.predict(ct.transform(pd.DataFrame(data,columns=feature\_cols)))

pred = pred[0]

if pred:

return render\_template("output.html",y="This patient will be readmitted ")

else:

return render\_template("output.html",y="This patient will not be readmitted")

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug = True)