**A Project Report on**

**ML + Web combination**

Submitted to

**Mr. Sanjay tewari**

(HOD of computer science department)

Submitted by

**Mr. *Sagar joshi***

Enrollment Number**: *BSCCS008***

Roll Number**: *20171119208***

Course Name**: *BSC. Cyber security 6th semester***

**Uttarakhand Residential University, Almora**



ACKNOWLEDGEMENT

It is my great pleasure to present this report on the project named “Mushroom’s edibility detection” undertaken by me as part of my Bachelor of Science’s final year project.

I would like to express my special thanks to my teacher’s, who provide me insightful to implement this idea.

Thank you

CONTENTS

1. INTRODUCTION

2. SYSTEM ANALYSIS

2.1 life-cycle

3. SYSTEM IMPLEMENTATION & RESULTS

3.2 System Configuration

4. CODEING & SCREENSHOTS

4.1 Source Code

4.2 Screens

5. IMPLEMENTATION

6. CONCLUSION

7. Reference

1. INTRODUCTION

Mushroom edibility detection model is helpful for every new machine learner

because it provide the scikit-learn algorithm’s online processing, and this web

application will allows users to choose what classification algorithm they want to

use and let them interactively set hyper-parameter values, all without them

knowing to code.

***“This project gives opportunity to user, use Machine-learning model with the help of the web console with this project anyone can use SVM, logistic regression, Random forest.”***

• The system provides train Logistic Regression, Random Forest, and Support Vector Classifiers using scikit-learn.

• The system contains different Plot evaluation metrics for binary classification algorithms.

• The System gave opportunity to a new user to find his data and model’s accuracy, and other aspects without knowing machine- learning coding.

• All the console is easy to use, and anyone can use it without any difficulties.

• The system is user friendly, fully customizable. Not any special requirement to install the system only scikit-learn is sufficient.

2. SYSTEM ANALYSIS

2.1 INTRODUCTION TO SYSTEM ANALYSIS

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minutest detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews,

questionnaires etc. The data collected by these sources must be scrutinized to arrive to a

conclusion. The conclusion is an understanding of how the system functions. This system is

called the Existing system. Now the existing system is subjected to close study and problem

areas are identified. The designer now functions as a problem solver and tries to sort out the

difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then

weighed with the existing system analytically and the best one is selected. The proposal is

presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal. Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies a rough figure of the system activities can be obtained, from 6 which the decision about the strategies to be followed for effective system study and analysis can be taken.

**SVM** is a supervised machine learning algorithm which can be **used for** classification or regression problems. It uses a technique called the kernel trick to transform your data and then based on these transformations it finds an optimal boundary between the possible outputs.

**Logistic regression** is the appropriate **regression** analysis to conduct when the dependent variable is dichotomous (binary). ... **Logistic regression** is **used to** describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

The **random forest** is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated **forest** of trees whose prediction by committee is more accurate than that of any individual tree.

# 2.2 Confusion Matrix and ROC in Machine Learning-

In the field of machine learning and specifically the problem of statistical classification, a

confusion matrix, also known as an error matrix. A confusion matrix is a table that is often used

to describe the performance of a classification model (or “classifier”) on a set of test data for

which the true values are known. It allows the visualization of the performance of an algorithm.

It allows easy identification of confusion between classes e.g. one class is commonly mislabeled

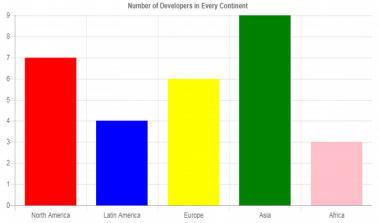
as the other. Most performance measures are computed from the confusion matrix.

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

A **receiver operating characteristic curve**, or **ROC curve**, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true-positive rate is also known as sensitivity, recall or *probability of detection*in machine learning. The false-positive rate is also known as *probability of false alarm* and can be calculated as (1 − specificity). It can also be thought of as a plot of the power as a function of the Type I Error of the decision rule (when the performance is calculated from just a sample of the population, it can be thought of as estimators of these quantities). The ROC curve is thus the sensitivity or recall as a function of fall-out. In general, if the probability distributions for both detection and false alarm are known, the ROC curve can be generated by plotting the cumulative distribution function (area under the probability distribution from {\displaystyle -\infty }to the discrimination threshold) of the detection probability in the y-axis versus the cumulative distribution function of the false-alarm probability on the x-axis.

2.1 Life-cycle



Data

csv, excel

Matrices selection

Data preprocessing prediction

Select which matrices you need to plot.

Model selection ex:

SVM, Logical regression, Random forest

Upload data, data preprocessing, train and test data split

Project life cycle

3. SYSTEM IMPLEMENTATION & RESULTS

Streamlit - Web development framework for creating machine learning models in websites.

Pandas- Python library for working with machine learning.

Numpy- Python library for mathematics calculation.

scikit-learn- Python library for making machine-learning models and prediction.

Dataset – for making a prediction model, with the help of scikit- learn library.

3.2 SYSTEM CONFIGURATION

Hardware Requirements

Processor : X86 Compatible processor (With 1.7 ghz Clock speed)

RAM : 4 GB or more

Hard disk : 20 GB or more

Mouse : 2 buttons/ 3 buttons

Software Requirements

Operating System : Windows 10

Front end : Streamlit

Back end : scikit-learn

4. CODEING & SCREENSHOTS

4.1 Source Code-

1. Libraries include -

import streamlit as st

import pandas as pd

import numpy as np

from sklearn.svm import SVC

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import plot\_confusion\_matrix, plot\_roc\_curve, plot\_precision\_recall\_curve

from sklearn.metrics import precision\_score, recall\_score

2. data preprocessing –

st.title("Binary Classification WEb App.")

st.sidebar.title("Binary Classification Web App.")

st.markdown("Are your mushroom edible or poisionous? 🍄")

st.sidebar.markdown("Are your mushroom edible or poisionous? 🍄")

@st.cache(persist= True)

def load\_data():

data = pd.read\_csv('mushrooms.csv')

label = LabelEncoder()

for col in data.columns:

data[col] = label.fit\_transform(data[col])

return data

@st.cache(persist= True)

def split(df):

y= df.type

x = df.drop(columns=['type'])

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y , test\_size=0.3, random\_state =0)

return x\_train, x\_test, y\_train, y\_test

3. plot matrics

def plot\_metrics(metrics):

if 'Confusion Martix' in metrics:

st.subheader("Confusion Matrix")

plot\_confusion\_matrix(model, x\_test, y\_test, display\_labels= class\_names)

st.pyplot()

if 'ROC Curve' in metrics:

st.subheader("ROC curve")

plot\_roc\_curve(model, x\_test, y\_test)

st.pyplot()

if 'Precision-Recall' in metrics:

st.subheader("Precision-Recall Curve")

plot\_precision\_recall\_curve(model, x\_test, y\_test)

st.pyplot()

4. Data loading and model selection-

df = load\_data()

x\_train, x\_test, y\_train, y\_test = split(df)

class\_names = ['edible','poisonous']

st.sidebar.subheader("Choose Classifier")

classifier = st.sidebar.selectbox("Classifier", ("Support Vector Machine (SVM)", "Logistic Regression", "Random Forest"))

5. For SVM ( support vector machine) –

if classifier == 'Support Vector Machine (SVM)':

st.sidebar.subheader("Model Hyperparameters")

C = st.sidebar.number\_input("C (Regularization parameters)", 0.01, 10.0, step =0.01, key ='C')

Kernel = st.sidebar.radio("Kernel", ("rbf", "linear"), key = 'Kernel')

gamma = st.sidebar.radio("Gamma(Kernel coefficient)",("scale", "auto"), key = 'gamma')

metrics = st.sidebar.multiselect("What metrics to Plot ?", ('Confusion Martix', 'ROC Curve', 'Precision-Recall'))

if st.sidebar.button("Classify", key = 'classify'):

st.subheader("Support Vector Machine (SVM) Results")

model = SVC(C=C, kernel=Kernel, gamma = gamma)

model.fit(x\_train, y\_train)

accuracy = model.score(x\_test, y\_test)

y\_pred = model.predict(x\_test)

st.write("Accuracy : ", accuracy.round(2))

st.write("Pricision : ", precision\_score(y\_test, y\_pred, labels = class\_names).round(2))

st.write("Recall : ", recall\_score(y\_test, y\_pred, labels = class\_names).round(2))

plot\_metrics(metrics)

6. for Logistic Regression –

if classifier == 'Logistic Regression':

st.sidebar.subheader("Model Hyperparameters")

C = st.sidebar.number\_input("C (Regularization parameters)", 0.01, 10.0, step =0.01, key ='C\_LR')

max\_iter = st.sidebar.slider("maximum number of interations", 100, 500, key = 'max\_iter')

metrics = st.sidebar.multiselect("What metrics to Plot ?", ('Confusion Martix', 'ROC Curve', 'Precision-Recall'))

if st.sidebar.button("Classify", key = 'classify'):

st.subheader("Logistic Regression Results")

model = LogisticRegression(C=C, max\_iter= max\_iter)

model.fit(x\_train, y\_train)

accuracy = model.score(x\_test, y\_test)

y\_pred = model.predict(x\_test)

st.write("Accuracy : ", accuracy.round(2))

st.write("Pricision : ", precision\_score(y\_test, y\_pred, labels = class\_names).round(2))

st.write("Recall : ", recall\_score(y\_test, y\_pred, labels = class\_names).round(2))

plot\_metrics(metrics)

7. For Random Forest -

if classifier == 'Random Forest':

st.sidebar.subheader("Model Hyperparameters")

n\_estimators = st.sidebar.number\_input ("The number of trees in the forest",100, 5000, step = 10 ,key = 'n\_estimators')

max\_depth = st.sidebar.number\_input("The maximum depth of the tree", 1 , 20, step= 1, key = 'max\_depth')

bootstrap = st.sidebar.radio ("Bootstrap samples when building trees", ('True', 'False'), key = 'bootstrap')

metrics = st.sidebar.multiselect("What metrics to Plot ?", ('Confusion Martix', 'ROC Curve', 'Precision-Recall'))

if st.sidebar.button("Classify", key = 'classify'):

st.subheader("Random Forest Rsults")

model = RandomForestClassifier(n\_estimators=n\_estimators, max\_depth= max\_depth, bootstrap= bootstrap, n\_jobs=-1)

model.fit(x\_train, y\_train)

accuracy = model.score(x\_test, y\_test)

y\_pred = model.predict(x\_test)

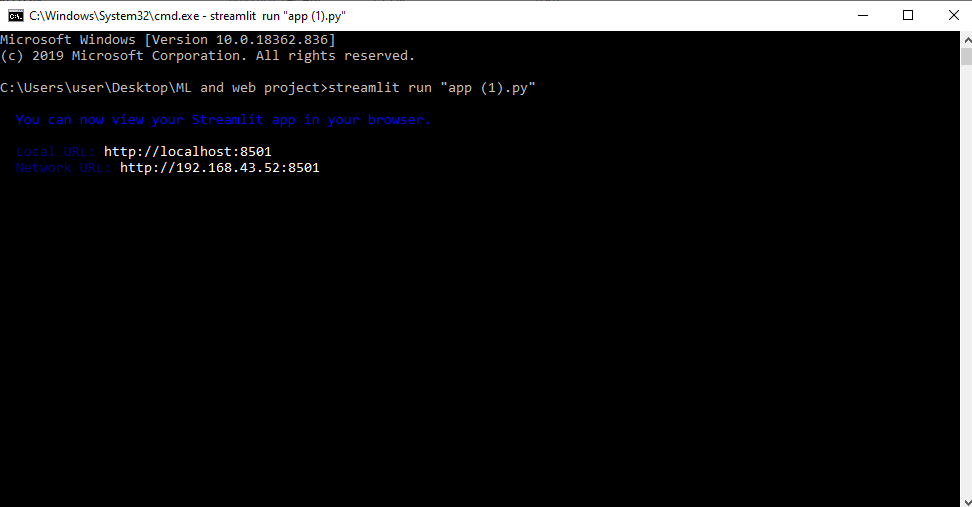
st.write("Accuracy : ", accuracy.round(2))

st.write("Pricision : ", precision\_score(y\_test, y\_pred, labels = class\_names).round(2))

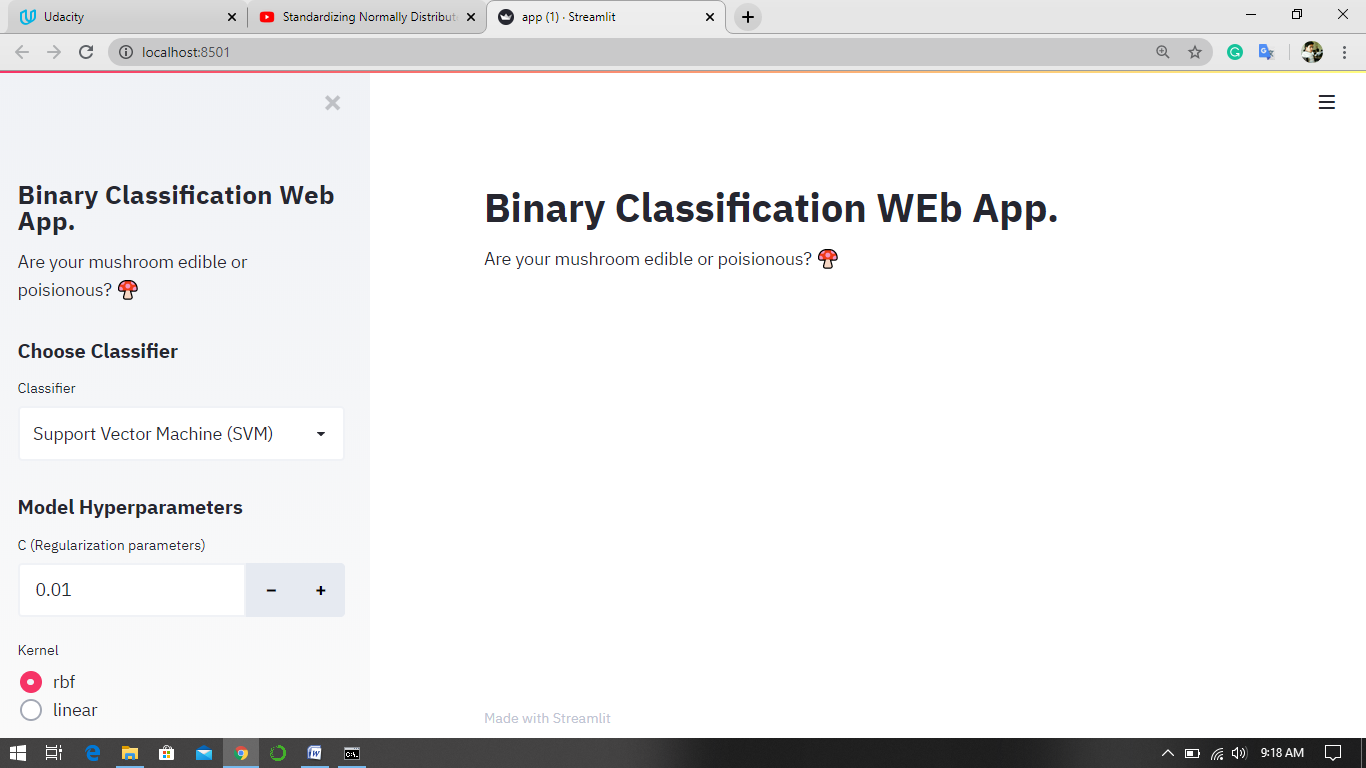
st.write("Recall : ", recall\_score(y\_test, y\_pred, labels = class\_names).round(2))

plot\_metrics(metrics)

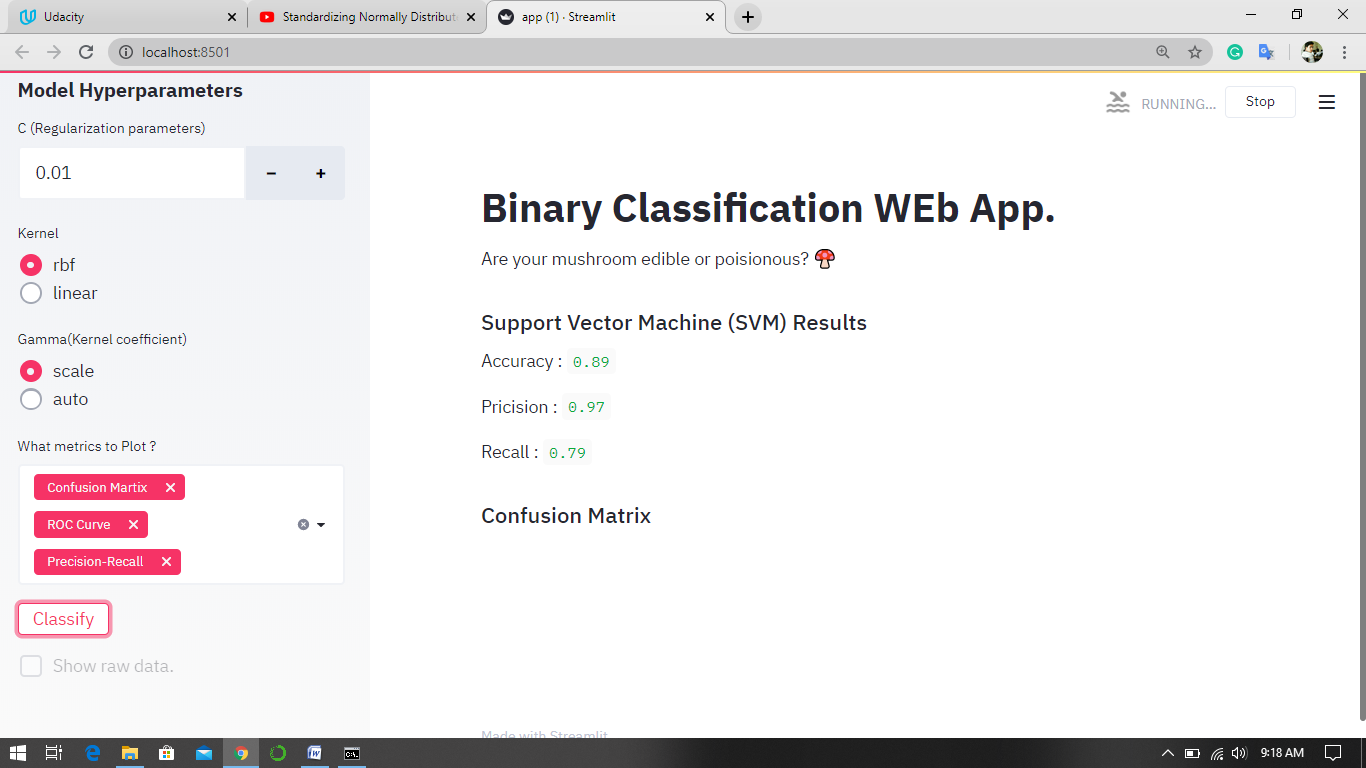
4.2 Screens –



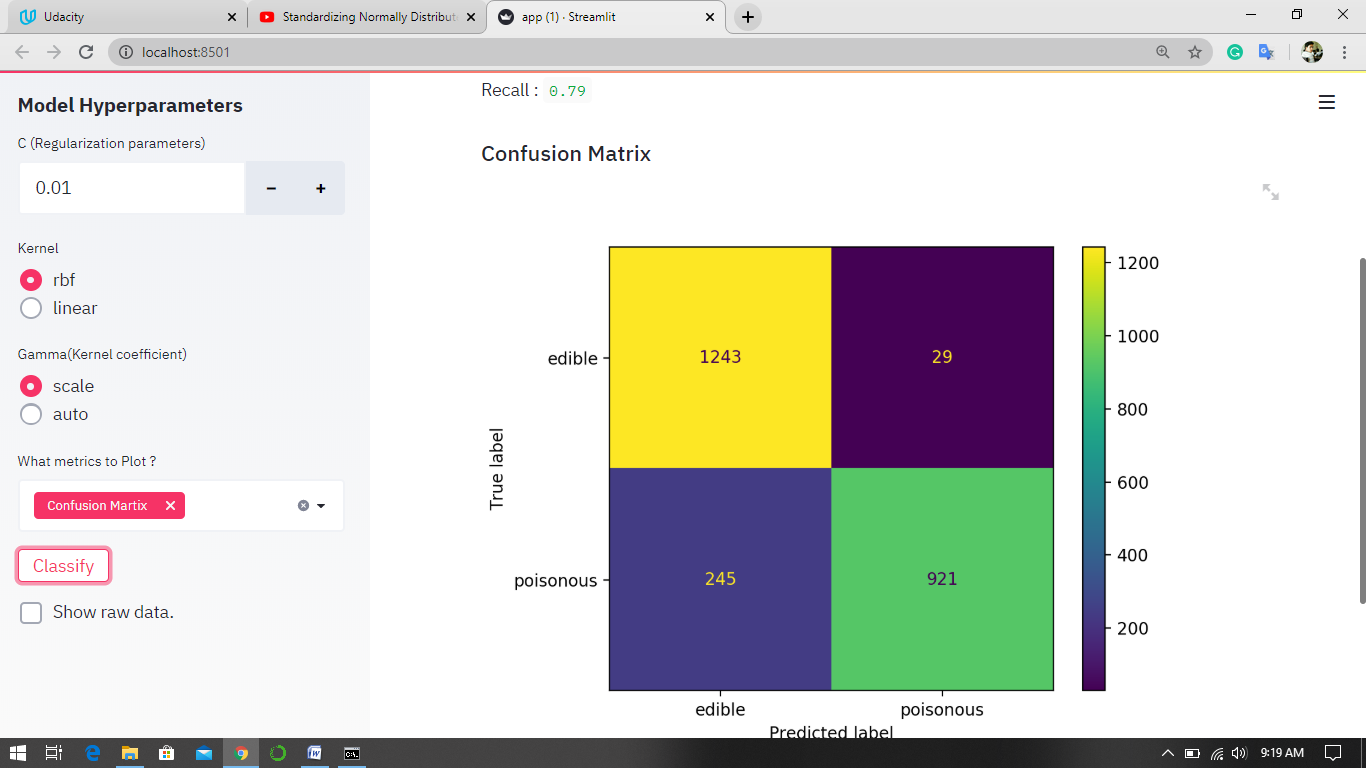
Starting the server in local host. Port 8501

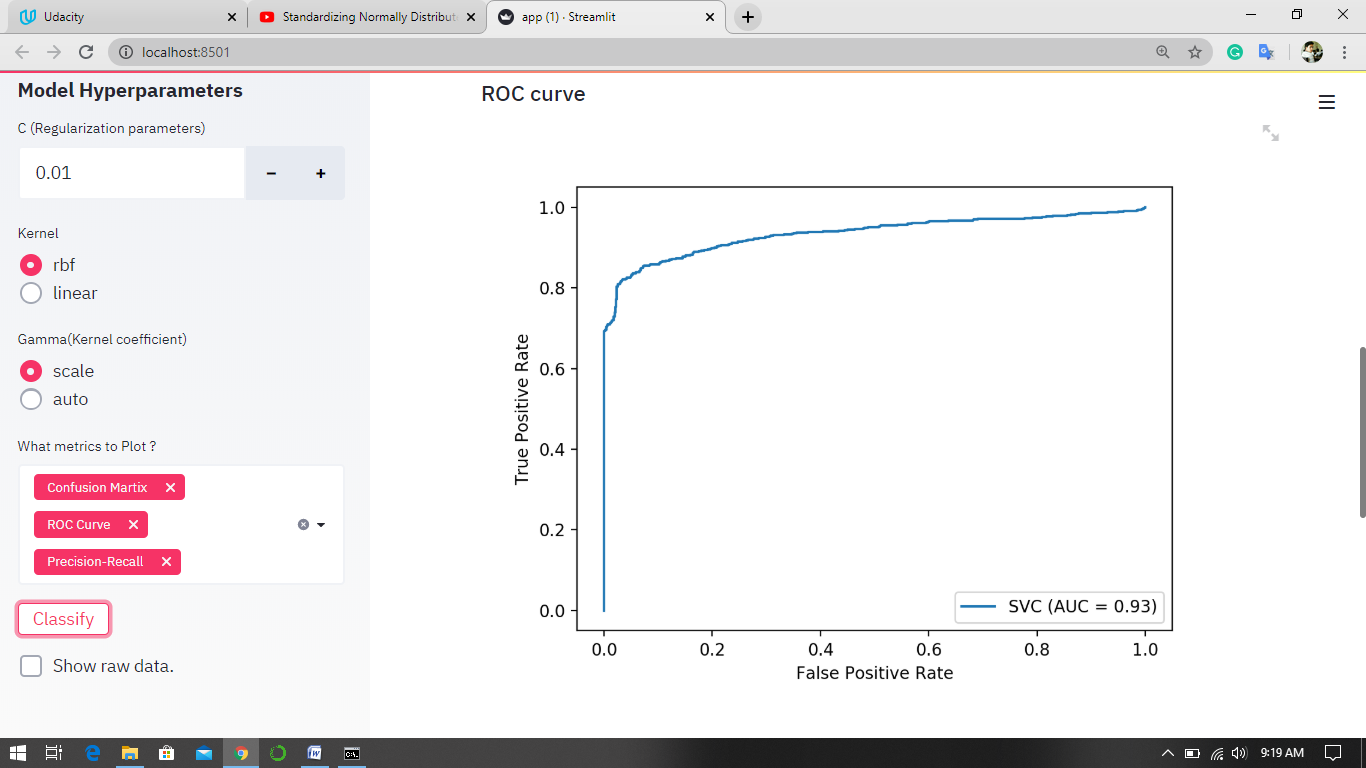


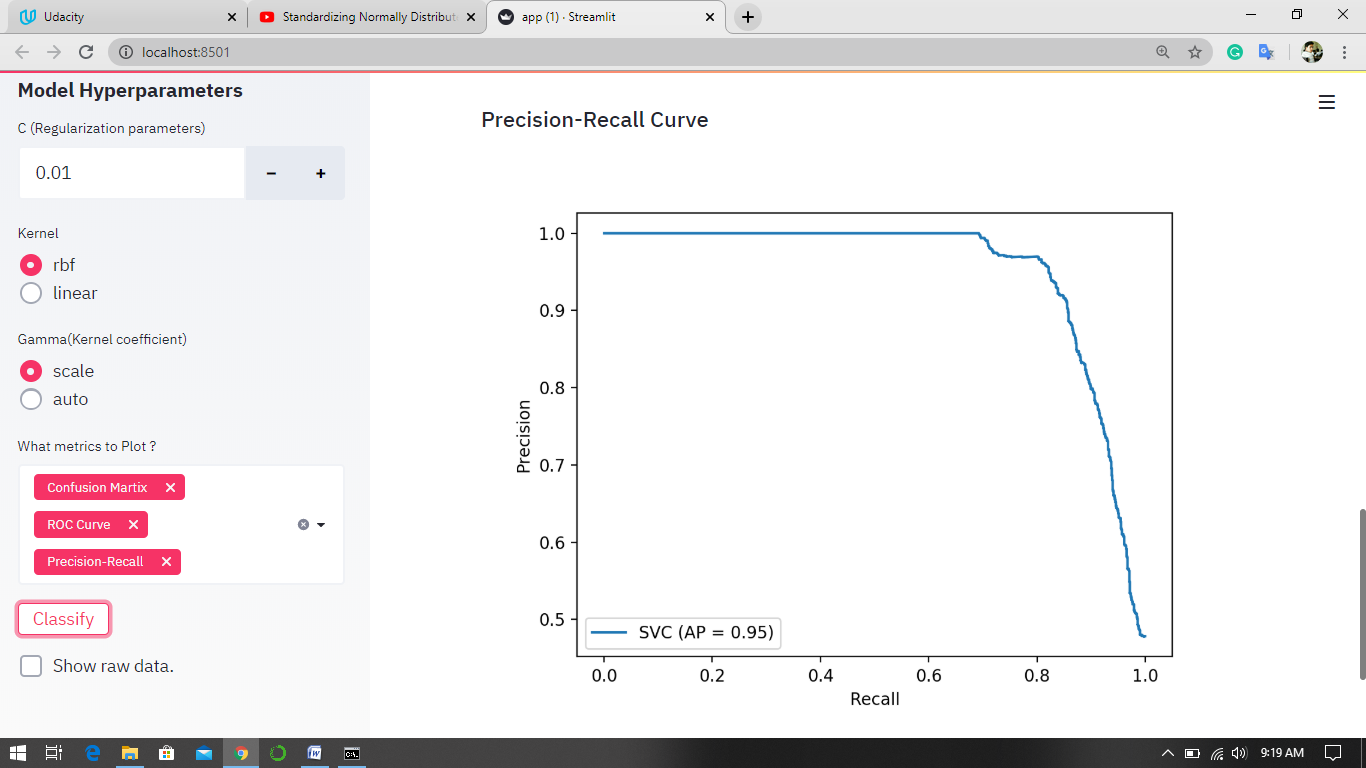
Web console screen



Model and parameters selection







Confusion, ROC, precision-recall curve

5. Implementation-

This project is basically about using machine-learning models with the web technology. But this project can be used in anywhere, where we need to include web and Artificial intelligence. At this time many organizations are using machine- learning on their web products, which gave and provide them a huge edge to their competitors.

And also we can implement same concept with predictive analytics. Where we can get an idea what will happen in future, how to take actions and decisions. In older days organization’s were using ‘***HIPPO***’ strategy for taking a decision, which gives more power and votes to the peoples which were sitting on the higher designation, but this decision not taken by consideration the data.

6. Conclusion -

*So at the end of this documentation we can say with the help of this project model we can implement any machine learning model at any web portal, and any user can use it whether he/she have knowledge about programming or not. This project only needs data, and other things it can do by itself. You only need to select which model you need for prediction. You can get your predictions and analytics*.

References –

<https://www.geeksforgeeks.org/confusion-matrix-machine-learning/>

<https://en.wikipedia.org/wiki/Random_forest>

<https://scikit-learn.org/>

<https://www.coursera.org/>

<https://www.udacity.com/>