**Implementation of App**

#importing required libraries

from flask import Flask, request, render\_template import numpy as np

import pandas as pd

from sklearn import metrics import warnings

import pickle warnings.filterwarnings('ignore') from feature import FeatureExtraction

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

file.close()

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

@app.route("/", methods=["GET", "POST"]) def index():

if request.method == "POST":

url = request.form["url"] obj = FeatureExtraction(url)

x = np.array(obj.getFeaturesList()).reshape(1,30)

y\_pred =gbc.predict(x)[0] #1 is safe

#-1 is unsafe

y\_pro\_phishing = gbc.predict\_proba(x)[0,0] y\_pro\_non\_phishing = gbc.predict\_proba(x)[0,1] # if(y\_pred ==1 ):

pred = "It is {0:.2f} % safe to go ".format(y\_pro\_phishing\*100)

return render\_template('index.html',xx =round(y\_pro\_non\_phishing,2),url=url ) return render\_template("index.html", xx =-1)

if \_\_name\_\_ == "\_\_main\_\_": app.run(debug=True)

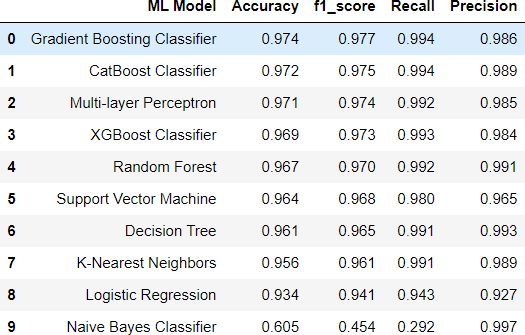
### Comparison of Models

result = pd.DataFrame({ 'ML Model' : ML\_Model, 'Accuracy' : accuracy,

'f1\_score' : f1\_score, 'Recall' : recall, 'Precision': precision,

})

result



### Fig 5.1: Comparison of Models

**Implementation of best ML model**

# XGBoost Classifier Model

from xgboost import XGBClassifier

# instantiate the model

gbc = GradientBoostingClassifier(max\_depth=4,learning\_rate=0.7)

# fit the model gbc.fit(X\_train,y\_train)

import pickle

# dump information to that file pickle.dump(gbc, open('pickle/model.pkl', 'wb'))

#checking the feature improtance in the model plt.figure(figsize=(9,7))

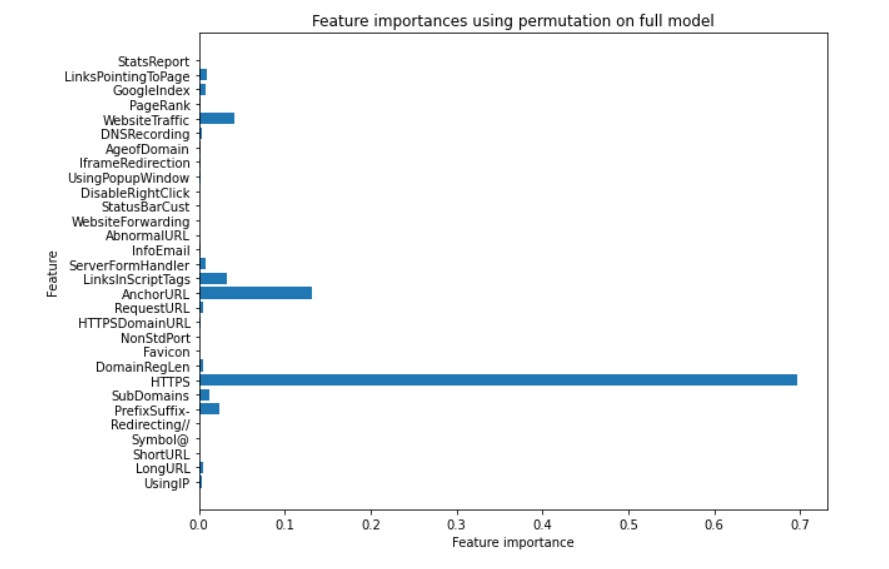
n\_features = X\_train.shape[1]

plt.barh(range(n\_features), gbc.feature\_importances\_, align='center') plt.yticks(np.arange(n\_features), X\_train.columns)

plt.title("Feature importances using permutation on full model") plt.xlabel("Feature importance")

plt.ylabel("Feature") plt.show()

.



**Fig 5.2: Feature Importance using Permutation on Model**