In [ ]:
```
#DO NOT RUN THIS CELL.

#Please run the following command to install all the required packages for this pro
ject.
#pip install -r requirements.txt

#The following cell mentions all the features with its label which are included in
this machine learning project of
#detecting phishing website.
@attribute having_IP_Address  { -1,1 }
@attribute URL_Length   { 1,0,-1 }
@attribute Shortining_Service { 1,-1 }
@attribute having_At_Symbol   { 1,-1 }
@attribute double_slash_redirecting { -1,1 }
@attribute Prefix_Suffix  { -1,1 }
@attribute having_Sub_Domain  { -1,0,1 }
@attribute SSLfinal_State  { -1,1,0 }
@attribute Domain_registeration_length { -1,1 }
@attribute Favicon { 1,-1 }
@attribute port { 1,-1 }
@attribute HTTPS_token { -1,1 }
@attribute Request_URL  { 1,-1 }
@attribute URL_of_Anchor { -1,0,1 }
@attribute Links_in_tags { 1,-1,0 }
@attribute SFH  { -1,1,0 }
@attribute Submitting_to_email { -1,1 }
@attribute Abnormal_URL { -1,1 }
@attribute Redirect  { 0,1 }
@attribute on_mouseover  { 1,-1 }
@attribute RightClick  { 1,-1 }
@attribute popUpWidnow  { 1,-1 }
@attribute Iframe { 1,-1 }
@attribute age_of_domain  { -1,1 }
@attribute DNSRecord   { -1,1 }
@attribute web_traffic  { -1,0,1 }
@attribute Page_Rank { -1,1 }
@attribute Google_Index { 1,-1 }
@attribute Links_pointing_to_page { 1,0,-1 }
@attribute Statistical_report { -1,1 }
@attribute Result  { -1,1 }
```

In [ ]:
```
#DO NOT RUN THIS CELL.
#This cell includes the code required to extract the particular feature from the UR
L and to convert it to its corresponding
#heuristics.

import ipaddress
import re
import urllib.request
from bs4 import BeautifulSoup
import socket
import requests
from googlesearch import search
import whois
from datetime import datetime
import time
from dateutil.parser import parse as date_parse

# Calculates number of months
def diff_month(d1, d2):
    return (d1.year - d2.year) * 12 + d1.month - d2.month

# Generate data set by extracting the features from the URL
def generate_data_set(url):

    data_set = []

    # Converts the given URL into standard format
    if not re.match(r"^https?", url):
        url = "http://" + url
    # Stores the response of the given URL
    try:
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')
    except:
        response = ""
        soup = -999
    # Extracts domain from the given URL
    domain = re.findall(r"://([^/]+)/?", url)[0]
    if re.match(r"^www.",domain):
                domain = domain.replace("www.","")
    # Requests all the information about the domain
    whois_response = whois.whois(domain)
    rank_checker_response = requests.post("https://www.checkpagerank.net/index.ph
p", {
        "name": domain
    })
    # Extracts global rank of the website
    try:
        global_rank = int(re.findall(r"Global Rank: ([0-9]+)", rank_checker_respons
e.text)[0])
    except:
        global_rank = -1
    # 1.having_IP_Address
    try:
        ipaddress.ip_address(url)
        data_set.append(-1)
    except:
        data_set.append(1)

    # 2.URL_Length
    if len(url) < 54:
        data_set.append(1)
    elif len(url) >= 54 and len(url) <= 75:
        data_set.append(0)
```

In [1]:
```
#This Cell helps us to create a New Dataset which consist of the required 30 features.
#Run this cell by placing New URLs in urls.txt file located in this folder.
#The format of the New URL should be as follows:
#URL,(1,-1)
#i.e 1 = Legitimate , -1 = Phishing
#Example : http://www.plu.sh/c3p7g/,-1
#The New Dataset created is saved as new_dataset.csv under the local directory.
#We can go on adding new URLs to the urls.txt file and dataset will be appended to new_dataset.csv

import pandas as pd
import feature_extraction as fe

urls = open("urls.txt", 'r')
new_dataset = open("new_dataset.csv", 'a')

features = []

for url in urls.readlines():

    label = int(url.strip().split(',')[1])
    feat = fe.generate_data_set(url.split(',')[0])
    feat += [label]
    print(str(feat))
    f = str(feat)[1:-1]
    new_dataset.write(f)
    new_dataset.write("\n")
    #features.append(feat)
new_dataset.close()
urls.close()
#features_df = pd.DataFrame(features)
#features_df.to_csv("new_dataset.csv")
#print(features)
```

```
[1, 1, 1, 1, 1, 1, 0, 1, -1, 1, 1, -1, 1, -1, 0, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 0, -1, 1, -1, 1]
[1, 1, 1, 1, 1, 1, 0, 1, -1, 1, 1, -1, 1, -1, 0, -1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 0, -1, 1, -1, -1]
[1, 1, 1, 1, 1, 1, 0, 1, -1, 1, 1, 1, 0, -1, -1, 1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, 1, 1, 1, 1, 1, 0, 1, -1, 1, 1, 1, 0, -1, -1, 1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1]
[1, -1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, -1, 1]
[1, -1, 1, 1, 1, 1, -1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, -1, -1, -1, -1, -1, 1, 1, 1, 1, 1, 1, -1, 1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, 1, -1, 0, 1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1]
[1, 1, 1, 1, 1, 1, 1, 1, -1, -1, 1, 1, 1, -1, 0, 1, 1, -1, -1, -1, -1, -1, 1, 1, -1, 1, 1, 1, -1, 1, 1]
```

In [8]:
```
#This Cell help us to generate a model and save it in pickle format.
#Pickle is the standard way of serializing objects in Python.
#The pickle operation is used to serialize the machine learning algorithms, later w
e can load the file to deserialize the model and use it to make new predictions.

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
import pickle

save_model_folder = "model/"
directory = os.path.dirname(save_model_folder)
if not os.path.exists(directory):
        os.makedirs(directory)

input_file = "dataset.csv"
 #Importing dataset
data = np.loadtxt("dataset.csv", delimiter = ",")
#Seperating features and labels
X = data[: , :-1]
y = data[: , -1]
#Seperating training features, testing features, training labels & testing labels
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
h = .02  # step size in the mesh
names = [
                "Nearest Neighbors",
                "RBF SVM",
        "Decision Tree",
                "Random Forest"
                ]
classifiers = [
           KNeighborsClassifier(3),
           SVC(kernel='rbf', C=1, gamma='auto'),
           DecisionTreeClassifier(max_depth=5),
           RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1)
          ]
# iterate over classifiers
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    pickle.dump(clf, open(save_model_folder+name+".sav", 'wb'))
    print(name," -- ",100*score,"%")
```

```
Nearest Neighbors  --  95.56761646313885 %
RBF SVM  --  94.70827679782904 %
Decision Tree  --  92.22071460877432 %
Random Forest  --  89.64269561284488 %
```

In [10]:
```
#In this cell we should specify a particular model which has obtained the highest a
ccuracy during training/testing phase.
#Then when we run the cell the user needs to input the URL in the text field.
#According to the model trained with the dataset the system will try to predict tha
t the given URL is Phishing or Legitimate
#For example of Phishing website one can look into spam mailbox his/her email clien
t.
import numpy as np
import feature_extraction
from sklearn.ensemble import RandomForestClassifier as rfc
from sklearn.linear_model import LogisticRegression as lr
from flask import jsonify
import pickle

def getResult(url):
    clf = pickle.load(open("model/Nearest Neighbors.sav", 'rb'))
    X_new = []
    X_input = url
    X_new=feature_extraction.generate_data_set(X_input)
    X_new = np.array(X_new).reshape(1,-1)

    try:
        prediction = clf.predict(X_new)
        if prediction == -1:
            return "Phishing Url"
        else:
            return "Legitimate Url"
    except:
        return "Phishing Url"

URL = input()
print(getResult(URL))
```

```
http://email.email.vast.gg/c/eJydUU1vozAU_DXmgoJsAyEcOJAAbUkTbXfzofSCDBjqFDA1Dg3
769dk2y69rmTL896M3huNM1K3hJUNsJeCZqxltJFJzmvCVCvwSgUqI-O1lv0T0rGZdJf0TDM5qhDUr_r
GD4Jwq2Oor3hO9buHQ-gf_ZMOIgjcCLgW8G0t96ysICacjuuJYEQyflu44o0UvNKYhyFy4QIitDAtaBv
IWC4DFIZ45ZvQslerBbDgzYnRk04aZam9eCRdFNgklKA8c6zCdNM0RXPo4gxaNHfM6dr_8l95L1K2HTB
9gCN1yoqS2mBcwbDcL6_qHd3MStZT8k6GGbmUF1UjdSX_rHCTAzPqqeAJUyhIVQBNR0njOA7AtgW_Ygd
4fpNlvOmVEDvR-rk9vq59cxmLu-tTvf7trDfXOHn70bvnX9vi8Ib7p0eBRPE4SHlPytfEPtZps9_X3c_
nRSvO9WZQke9Ou7Df3Z-KgcUsnu3WvHjYxprwvnmZOJlG1wo-Rpc0pKZjfoePD5hI5NDeqC197yoqJRX
fWFq3FZH0a8KU_IRjNoqaO64JbU16YxCzv9XMwS6C8KP5B1EB5G0
[1, -1, 1, 1, 1, 1, -1, 1, -1, -1, 1, -1, 1, -1, 1, 1, -1, -1, -1, -1, -1, 1, 1,
-1, 0, -1, 1, -1, 1]
Phishing Url
```

In [ ]: