What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

```
86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

```
604800
```

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see Overview of Colab. To create a new Colab notebook you can use the File menu above, or use the following link: create a new Colab notebook.

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see jupyter.org.

Data science

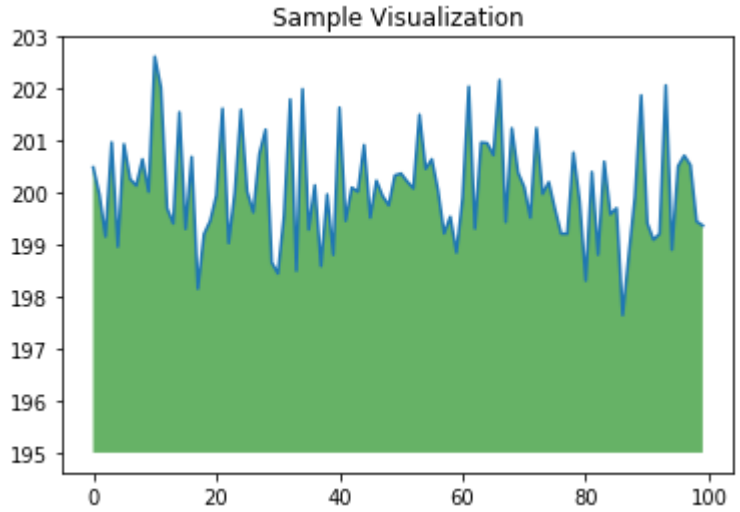
With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

```
import numpy as np
from matplotlib import pyplot as plt

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)

plt.title("Sample Visualization")
plt.show()
```



You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many other sources. To learn more about importing data, and how Colab can be used for data science, see the links below under Working with Data.

Machine learning

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just a few lines of code. Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including GPUs and TPUs, regardless of the power of your machine. All you need is a browser.

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the machine learning examples below.

More Resources

Working with Notebooks in Colab

- Overview of Colaboratory
- Guide to Markdown
- Importing libraries and installing dependencies
- Saving and loading notebooks in GitHub
- Interactive forms
- Interactive widgets
- NEW TensorFlow 2 in Colab

Working with Data

- Loading data: Drive, Sheets, and Google Cloud Storage
- Charts: visualizing data
- Getting started with BigQuery

Machine Learning Crash Course

These are a few of the notebooks from Google's online Machine Learning course. See the full course website for more.

- Intro to Pandas DataFrame
- Linear regression with tf.keras using synthetic data

Using Accelerated Hardware

- TensorFlow with GPUs

- TensorFlow with TPUs

Machine Learning Examples

To see end-to-end examples of the interactive machine learning analyses that Colaboratory makes possible, check out these tutorials using models from TensorFlow Hub.

A few featured examples:

- Retraining an Image Classifier: Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- Text Classification: Classify IMDB movie reviews as either *positive* or *negative*.
- Style Transfer: Use deep learning to transfer style between images.
- Multilingual Universal Sentence Encoder Q&A: Use a machine learning model to answer questions from the SQuAD dataset.
- Video Interpolation: Predict what happened in a video between the first and the last frame.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv("/CyberSecurity_Data.csv")
data.head()
```

	id	having_IP_Address	URL_Length	Shortining_Service	having_At_Symbol	double_slas
0	1	-1	1	1	1	
1	2	1	1	1	1	
2	3	1	0	1	1	
3	4	1	0	1	1	
4	5	1	0	-1	1	

```
data.drop(["id"],axis=1,inplace=True)
data.columns
```

```
Index(['having_IP_Address', 'URL_Length', 'Shortining_Service',
       'having_At_Symbol', 'double_slash_redirecting', 'Prefix_Suffix',
       'having_Sub_Domain', 'SSLfinal_State', 'Domain_registeration_length',
       'Favicon', 'port', 'HTTPS_token', 'Request_URL', 'URL_of_Anchor',
       'Links_in_tags', 'SFH', 'Submitting_to_email', 'Abnormal_URL',
```

```
       'Redirect', 'on_mouseover', 'RightClick', 'popUpWidnow', 'Iframe',
       'age_of_domain', 'DNSRecord', 'web_traffic', 'Page_Rank',
       'Google_Index', 'Links_pointing_to_page', 'Statistical_report',
       'Result'],
      dtype='object')
```

```
data.shape
```

```
(11055, 31)
```

```
data.isnull().values.any()
```

```
False
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_IP_Address',axis=1)
y=data['having_IP_Address']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  87.44911804613297
Test set accuracy :  87.01944821347807
              precision    recall  f1-score   support

          -1       0.83      0.78      0.81       762
           1       0.89      0.92      0.90      1449

    accuracy                           0.87      2211
   macro avg       0.86      0.85      0.85      2211
weighted avg       0.87      0.87      0.87      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('URL_Length',axis=1)
y=data['URL_Length']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
```

```
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  93.31750339213026
Test set accuracy :  92.8991406603347
              precision    recall  f1-score   support

          -1       0.93      0.98      0.96      1800
           0       1.00      0.50      0.67        30
           1       0.89      0.71      0.79       381

    accuracy                           0.93      2211
   macro avg       0.94      0.73      0.81      2211
weighted avg       0.93      0.93      0.93      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Shortining_Service',axis=1)
y=data['Shortining_Service']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  98.60922659430122
Test set accuracy :  98.23609226594301
              precision    recall  f1-score   support

          -1       0.97      0.89      0.93       287
           1       0.98      1.00      0.99      1924

    accuracy                           0.98      2211
   macro avg       0.98      0.94      0.96      2211
weighted avg       0.98      0.98      0.98      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('having_At_Symbol',axis=1)
y=data['having_At_Symbol']
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  92.34509271822705
Test set accuracy :  92.08502939846224
              precision    recall  f1-score   support

          -1       0.87      0.57      0.69       342
           1       0.93      0.99      0.95      1869

    accuracy                           0.92      2211
   macro avg       0.90      0.78      0.82      2211
weighted avg       0.92      0.92      0.91      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Prefix_Suffix',axis=1)
y=data['Prefix_Suffix']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  89.15649027589326
Test set accuracy :  89.37132519222072
              precision    recall  f1-score   support

          -1       0.90      0.99      0.94      1930
           1       0.76      0.24      0.36       281

    accuracy                           0.89      2211
   macro avg       0.83      0.61      0.65      2211
weighted avg       0.88      0.89      0.87      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

```
x=data.drop('having_Sub_Domain',axis=1)
y=data['having_Sub_Domain']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  64.54093170511081
Test set accuracy :  61.55585707824513
              precision    recall  f1-score   support

          -1       0.56      0.47      0.51       650
           0       0.61      0.63      0.62       712
           1       0.66      0.71      0.69       849

    accuracy                           0.62      2211
   macro avg       0.61      0.61      0.60      2211
weighted avg       0.61      0.62      0.61      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('SSLfinal_State',axis=1)
y=data['SSLfinal_State']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  84.96155585707824
Test set accuracy :  83.67254635911353
              precision    recall  f1-score   support

          -1       0.74      0.79      0.77       710
           0       0.73      0.62      0.67       247
           1       0.91      0.90      0.91      1254

    accuracy                           0.84      2211
   macro avg       0.80      0.77      0.78      2211
weighted avg       0.84      0.84      0.84      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Domain_registeration_length',axis=1)
y=data['Domain_registeration_length']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  83.87607417458163
Test set accuracy :  82.76797829036636
              precision    recall  f1-score   support

          -1       0.89      0.83      0.86      1436
           1       0.73      0.81      0.77       775

    accuracy                           0.83      2211
   macro avg       0.81      0.82      0.82      2211
weighted avg       0.83      0.83      0.83      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Favicon',axis=1)
y=data['Favicon']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  99.36680235187698
Test set accuracy :  99.32157394843962
              precision    recall  f1-score   support

          -1       0.97      1.00      0.98       426
           1       1.00      0.99      1.00      1785

    accuracy                           0.99      2211
   macro avg       0.98      0.99      0.99      2211
weighted avg       0.99      0.99      0.99      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('port',axis=1)
y=data['port']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  99.36680235187698
Test set accuracy :  99.05020352781547
              precision    recall  f1-score   support

          -1       0.96      0.97      0.96       295
           1       1.00      0.99      0.99      1916

    accuracy                           0.99      2211
   macro avg       0.98      0.98      0.98      2211
weighted avg       0.99      0.99      0.99      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('HTTPS_token',axis=1)
y=data['HTTPS_token']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  96.5513342379014
Test set accuracy :  95.70330167345092
              precision    recall  f1-score   support

          -1       0.97      0.77      0.86       368
           1       0.96      1.00      0.97      1843

    accuracy                           0.96      2211
   macro avg       0.96      0.88      0.92      2211
```

```
weighted avg       0.96      0.96      0.95      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Request_URL',axis=1)
y=data['Request_URL']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  82.41745816372682
Test set accuracy :  81.32066938037087
              precision    recall  f1-score   support

          -1       0.83      0.72      0.77       950
           1       0.81      0.89      0.84      1261

    accuracy                           0.81      2211
   macro avg       0.82      0.80      0.81      2211
weighted avg       0.81      0.81      0.81      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('URL_of_Anchor',axis=1)
y=data['URL_of_Anchor']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  74.90954319312529
Test set accuracy :  74.31026684758028
              precision    recall  f1-score   support

          -1       0.82      0.87      0.84       694
           0       0.71      0.81      0.76      1043
```

```
           1       0.68      0.40      0.50       474

    accuracy                           0.74      2211
   macro avg       0.73      0.69      0.70      2211
weighted avg       0.74      0.74      0.73      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Links_in_tags',axis=1)
y=data['Links_in_tags']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  60.36861148801448
Test set accuracy :  56.490275893260964
              precision    recall  f1-score   support

          -1       0.57      0.59      0.58       785
           0       0.54      0.67      0.60       869
           1       0.64      0.37      0.47       557

    accuracy                           0.56      2211
   macro avg       0.58      0.54      0.55      2211
weighted avg       0.57      0.56      0.56      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('SFH',axis=1)
y=data['SFH']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  84.36227951153325
Test set accuracy :  84.48665762098598
```

```
              precision    recall  f1-score   support

          -1       0.86      0.98      0.91      1712
           0       0.58      0.05      0.09       151
           1       0.77      0.54      0.64       348

    accuracy                           0.84      2211
   macro avg       0.74      0.52      0.54      2211
weighted avg       0.82      0.84      0.81      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Submitting_to_email',axis=1)
y=data['Submitting_to_email']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  96.82270465852555
Test set accuracy :  94.9796472184532
              precision    recall  f1-score   support

          -1       0.95      0.78      0.85       420
           1       0.95      0.99      0.97      1791

    accuracy                           0.95      2211
   macro avg       0.95      0.88      0.91      2211
weighted avg       0.95      0.95      0.95      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Abnormal_URL',axis=1)
y=data['Abnormal_URL']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  98.24739936680236
Test set accuracy :  97.51243781094527
              precision    recall  f1-score   support

          -1       0.93      0.90      0.91       327
           1       0.98      0.99      0.99      1884

    accuracy                           0.98      2211
   macro avg       0.96      0.94      0.95      2211
weighted avg       0.97      0.98      0.97      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Redirect',axis=1)
y=data['Redirect']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  94.92311171415649
Test set accuracy :  93.89416553595657
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      1951
           1       0.87      0.57      0.69       260

    accuracy                           0.94      2211
   macro avg       0.91      0.78      0.83      2211
weighted avg       0.94      0.94      0.93      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('on_mouseover',axis=1)
y=data['on_mouseover']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
```

```
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  97.72727272727273
Test set accuracy :  96.65309814563547
              precision    recall  f1-score   support

          -1       0.87      0.84      0.85       260
           1       0.98      0.98      0.98      1951

    accuracy                           0.97      2211
   macro avg       0.93      0.91      0.92      2211
weighted avg       0.97      0.97      0.97      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('RightClick',axis=1)
y=data['RightClick']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  98.23609226594301
Test set accuracy :  97.78380823156942
              precision    recall  f1-score   support

          -1       0.76      0.71      0.74        96
           1       0.99      0.99      0.99      2115

    accuracy                           0.98      2211
   macro avg       0.88      0.85      0.86      2211
weighted avg       0.98      0.98      0.98      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('popUpWidnow',axis=1)
y=data['popUpWidnow']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
```

```
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  98.9710538218001
Test set accuracy :  98.64314789687924
              precision    recall  f1-score   support

          -1       0.97      0.97      0.97       439
           1       0.99      0.99      0.99      1772

    accuracy                           0.99      2211
   macro avg       0.98      0.98      0.98      2211
weighted avg       0.99      0.99      0.99      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Iframe',axis=1)
y=data['Iframe']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  98.11171415649027
Test set accuracy :  97.64812302125735
              precision    recall  f1-score   support

          -1       0.89      0.83      0.86       194
           1       0.98      0.99      0.99      2017

    accuracy                           0.98      2211
   macro avg       0.94      0.91      0.92      2211
weighted avg       0.98      0.98      0.98      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('age_of_domain',axis=1)
y=data['age_of_domain']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
```

```
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  77.71370420624152
Test set accuracy :  77.74762550881954
              precision    recall  f1-score   support

          -1       0.77      0.73      0.75      1010
           1       0.78      0.82      0.80      1201

    accuracy                           0.78      2211
   macro avg       0.78      0.77      0.77      2211
weighted avg       0.78      0.78      0.78      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('DNSRecord',axis=1)
y=data['DNSRecord']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  88.7946630483944
Test set accuracy :  89.59746720940751
              precision    recall  f1-score   support

          -1       0.86      0.79      0.82       674
           1       0.91      0.94      0.93      1537

    accuracy                           0.90      2211
   macro avg       0.88      0.87      0.87      2211
weighted avg       0.89      0.90      0.89      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('web_traffic',axis=1)
y=data['web_traffic']
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  68.09136137494346
Test set accuracy :  66.25961103573044
              precision    recall  f1-score   support

          -1       0.62      0.51      0.56       553
           0       0.55      0.34      0.42       501
           1       0.70      0.88      0.78      1157

    accuracy                           0.66      2211
   macro avg       0.62      0.57      0.58      2211
weighted avg       0.65      0.66      0.64      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Page_Rank',axis=1)
y=data['Page_Rank']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  79.01402080506557
Test set accuracy :  78.47127996381728
              precision    recall  f1-score   support

          -1       0.81      0.93      0.87      1636
           1       0.66      0.36      0.47       575

    accuracy                           0.78      2211
   macro avg       0.73      0.65      0.67      2211
weighted avg       0.77      0.78      0.76      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
```

```
from sklearn.model_selection import train_test_split
x=data.drop('Google_Index',axis=1)
y=data['Google_Index']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  88.37630031659882
Test set accuracy :  87.4265038444143
              precision    recall  f1-score   support

          -1       0.71      0.17      0.28       310
           1       0.88      0.99      0.93      1901

    accuracy                           0.87      2211
   macro avg       0.80      0.58      0.61      2211
weighted avg       0.86      0.87      0.84      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Links_pointing_to_page',axis=1)
y=data['Links_pointing_to_page']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  81.6146540027137
Test set accuracy :  80.82315694255993
              precision    recall  f1-score   support

          -1       0.74      0.36      0.49       107
           0       0.83      0.86      0.84      1238
           1       0.78      0.79      0.78       866

    accuracy                           0.81      2211
   macro avg       0.78      0.67      0.71      2211
weighted avg       0.81      0.81      0.80      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Statistical_report',axis=1)
y=data['Statistical_report']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  92.28855721393035
Test set accuracy :  92.35639981908639
              precision    recall  f1-score   support

          -1       0.86      0.52      0.65       300
           1       0.93      0.99      0.96      1911

    accuracy                           0.92      2211
   macro avg       0.90      0.75      0.80      2211
weighted avg       0.92      0.92      0.92      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('Result',axis=1)
y=data['Result']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  95.33016734509272
Test set accuracy :  94.61781999095432
              precision    recall  f1-score   support

          -1       0.95      0.93      0.94      1014
           1       0.94      0.96      0.95      1197

    accuracy                           0.95      2211
   macro avg       0.95      0.95      0.95      2211
weighted avg       0.95      0.95      0.95      2211
```

```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
from sklearn import metrics
from sklearn.model_selection import train_test_split
x=data.drop('double_slash_redirecting',axis=1)
y=data['double_slash_redirecting']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
rb=GradientBoostingClassifier(random_state=0)
rb.fit(x_train,y_train)
y_predict=rb.predict(x_test)
print("Train set accuracy: ",100*rb.score(x_train,y_train))
print("Test set accuracy : ",100*rb.score(x_test,y_test))
print(metrics.classification_report(y_test,y_predict))
```

```
Train set accuracy:  98.92582541836273
Test set accuracy :  98.37177747625509
              precision    recall  f1-score   support

          -1       0.97      0.90      0.93       278
           1       0.99      1.00      0.99      1933

    accuracy                           0.98      2211
   macro avg       0.98      0.95      0.96      2211
weighted avg       0.98      0.98      0.98      2211
```

✓ 0s completed at 10:29 AM