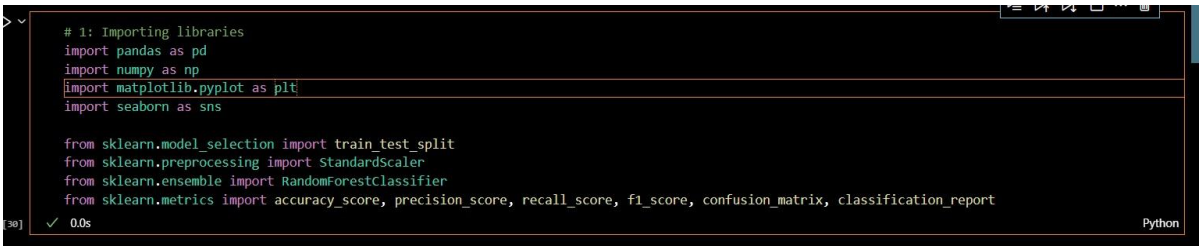
WEEK 1: TASK (PREDICTING WHETHER THE URL IS PHISING OR NOT)

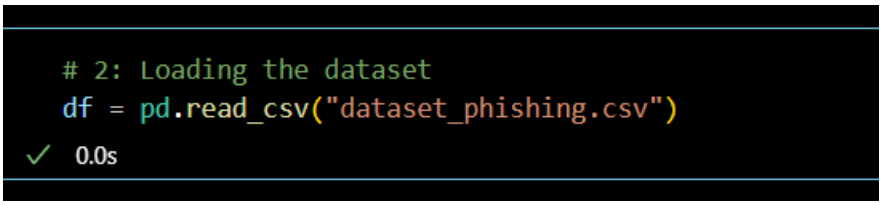
1 STEP: IMPORTING ALL THE LIBRARIES



All the basic library like pandas, numpy, matplotlib, seaborn are imported first.

Then other feature from sklearn are called like train_test_split, standardScaler, randomforest classifier, accuracy, precision, recall, f1 score and confusion matrix. These are called for calculation, training and testing are used for prediction and know aout the dataset.

2 STEP: LOADING THE DATASET



Name of dataset is been changed to dataset_phishing.csv. pandas is used to read the dataset. The 0: 'Legitimate', 1: 'Phishing' is directly done in csv file.

3 STEP: OVERVIEW OF THE DATASET

```
# 3: Overview of the dataset
print(df.shape)
print(df.info())
print(df.describe())
print(df['status'].value_counts())
print(df.columns.tolist())

sns.countplot(x='status', data=df, color= 'green')
plt.title("Website Distribution")
plt.xlabel("Status")
plt.ylabel("No. of Websites")
plt.xticks([0, 1], ['Legitimate (0)', 'Phishing (1)'])
plt.show()
```

✓ 0.2s

OUTPUT: (11430, 89)

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 11430 entries, 0 to 11429

Data columns (total 89 columns):

#	Column	Non-Null Count	Dtype
0	url	11430 non-null	object
1	length_url	11430 non-null	int64
2	length_hostname	11430 non-null	int64
3	ip	11430 non-null	int64
4	nb_dots	11430 non-null	int64
5	nb_hyphens	11430 non-null	int64
6	nb_at	11430 non-null	int64
7	nb_qm	11430 non-null	int64
8	nb_and	11430 non-null	int64
9	nb_or	11430 non-null	int64

10 nb_eq 11430 non-null int64
11 nb_underscore 11430 non-null int64
12 nb_tilde 11430 non-null int64
13 nb_percent 11430 non-null int64
14 nb_slash 11430 non-null int64
15 nb_star 11430 non-null int64
16 nb_colon 11430 non-null int64
17 nb_comma 11430 non-null int64
18 nb_semicolumn 11430 non-null int64
19 nb_dollar 11430 non-null int64
20 nb_space 11430 non-null int64
21 nb_www 11430 non-null int64
22 nb_com 11430 non-null int64
23 nb_dslash 11430 non-null int64
24 http_in_path 11430 non-null int64
25 https_token 11430 non-null int64
26 ratio_digits_url 11430 non-null float64
27 ratio_digits_host 11430 non-null float64
28 punycode 11430 non-null int64
29 port 11430 non-null int64
30 tld_in_path 11430 non-null int64
31 tld_in_subdomain 11430 non-null int64
32 abnormal_subdomain 11430 non-null int64
33 nb_subdomains 11430 non-null int64
34 prefix_suffix 11430 non-null int64
35 random_domain 11430 non-null int64
36 shortening_service 11430 non-null int64
37 path_extension 11430 non-null int64
38 nb_redirection 11430 non-null int64
39 nb_external_redirection 11430 non-null int64

40 length_words_raw 11430 non-null int64
41 char_repeat 11430 non-null int64
42 shortest_words_raw 11430 non-null int64
43 shortest_word_host 11430 non-null int64
44 shortest_word_path 11430 non-null int64
45 longest_words_raw 11430 non-null int64
46 longest_word_host 11430 non-null int64
47 longest_word_path 11430 non-null int64
48 avg_words_raw 11430 non-null float64
49 avg_word_host 11430 non-null float64
50 avg_word_path 11430 non-null float64
51 phish_hints 11430 non-null int64
52 domain_in_brand 11430 non-null int64
53 brand_in_subdomain 11430 non-null int64
54 brand_in_path 11430 non-null int64
55 suspecious_tld 11430 non-null int64
56 statistical_report 11430 non-null int64
57 nb_hyperlinks 11430 non-null int64
58 ratio_intHyperlinks 11430 non-null float64
59 ratio_extHyperlinks 11430 non-null float64
60 ratio_nullHyperlinks 11430 non-null int64
61 nb_extCSS 11430 non-null int64
62 ratio_intRedirection 11430 non-null int64
63 ratio_extRedirection 11430 non-null float64
64 ratio_intErrors 11430 non-null int64
65 ratio_extErrors 11430 non-null float64
66 login_form 11430 non-null int64
67 external_favicon 11430 non-null int64
68 links_in_tags 11430 non-null float64
69 submit_email 11430 non-null int64

```
 70  ratio_intMedia              11430 non-null  float64
 71  ratio_extMedia              11430 non-null  float64
 72  sfh                         11430 non-null  int64
 73  iframe                      11430 non-null  int64
 74  popup_window                11430 non-null  int64
 75  safe_anchor                 11430 non-null  float64
 76  onmouseover                 11430 non-null  int64
 77  right_clic                  11430 non-null  int64
 78  empty_title                 11430 non-null  int64
 79  domain_in_title             11430 non-null  int64
 80  domain_with_copyright       11430 non-null  int64
 81  whois_registered_domain     11430 non-null  int64
 82  domain_registration_length  11430 non-null  int64
 83  domain_age                  11430 non-null  int64
 84  web_traffic                 11430 non-null  int64
 85  dns_record                  11430 non-null  int64
 86  google_index                11430 non-null  int64
 87  page_rank                   11430 non-null  int64
 88  status                      11430 non-null  int64
dtypes: float64(13), int64(75), object(1)
memory usage: 7.8+ MB
None
        length_url  length_hostname            ip       nb_dots  \
count  11430.000000     11430.000000  11430.000000  11430.000000
mean      61.126684        21.090289      0.150569      2.480752
std       55.297318        10.777171      0.357644      1.369686
min       12.000000         4.000000      0.000000      1.000000
25%       33.000000        15.000000      0.000000      2.000000
50%       47.000000        19.000000      0.000000      2.000000
75%       71.000000        24.000000      0.000000      3.000000
```

max	1641.000000	214.000000	1.000000	24.000000

	nb_hyphens	nb_at	nb_qm	nb_and	nb_or \
count	11430.000000	11430.000000	11430.000000	11430.000000	11430.0
mean	0.997550	0.022222	0.141207	0.162292	0.0
std	2.087087	0.155500	0.364456	0.821337	0.0
min	0.000000	0.000000	0.000000	0.000000	0.0
25%	0.000000	0.000000	0.000000	0.000000	0.0
50%	0.000000	0.000000	0.000000	0.000000	0.0
75%	1.000000	0.000000	0.000000	0.000000	0.0
max	43.000000	4.000000	3.000000	19.000000	0.0

	nb_eq	...	domain_in_title	domain_with_copyright \
count	11430.000000	...	11430.000000	11430.000000
mean	0.293176	...	0.775853	0.439545
std	0.998317	...	0.417038	0.496353
min	0.000000	...	0.000000	0.000000
25%	0.000000	...	1.000000	0.000000
50%	0.000000	...	1.000000	0.000000
75%	0.000000	...	1.000000	1.000000
max	19.000000	...	1.000000	1.000000

	whois_registered_domain	domain_registration_length	domain_age \
count	11430.000000	11430.000000	11430.000000
mean	0.072878	492.532196	4062.543745
std	0.259948	814.769415	3107.784600
min	0.000000	-1.000000	-12.000000
25%	0.000000	84.000000	972.250000
50%	0.000000	242.000000	3993.000000
75%	0.000000	449.000000	7026.750000

max 1.000000 29829.000000 12874.000000

	web_traffic	dns_record	google_index	page_rank	status
count	1.143000e+04	11430.000000	11430.000000	11430.000000	11430.000000
mean	8.567566e+05	0.020122	0.533946	3.185739	0.500000
std	1.995606e+06	0.140425	0.498868	2.536955	0.500022
min	0.000000e+00	0.000000	0.000000	0.000000	0.000000
25%	0.000000e+00	0.000000	0.000000	1.000000	0.000000
50%	1.651000e+03	0.000000	1.000000	3.000000	0.500000
75%	3.738455e+05	0.000000	1.000000	5.000000	1.000000
max	1.076799e+07	1.000000	1.000000	10.000000	1.000000

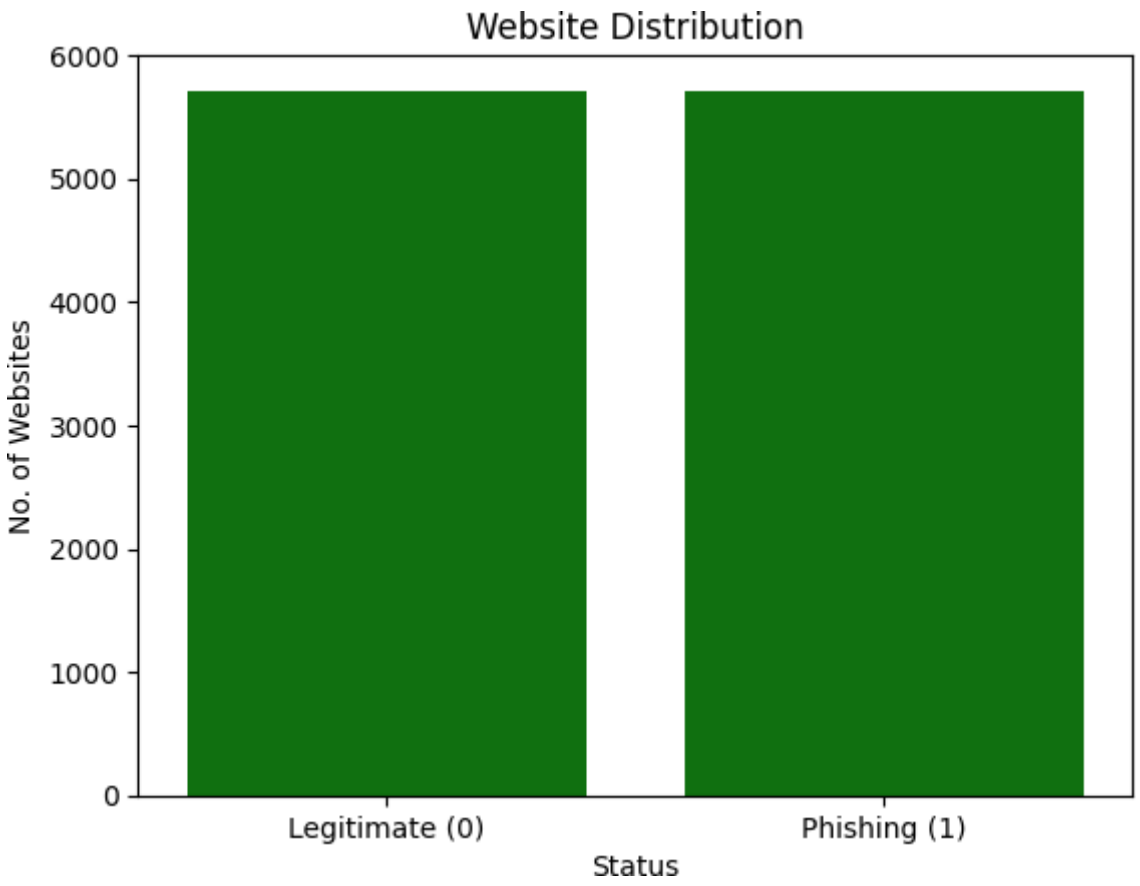
[8 rows x 88 columns]

0 5715

1 5715

Name: status, dtype: int64

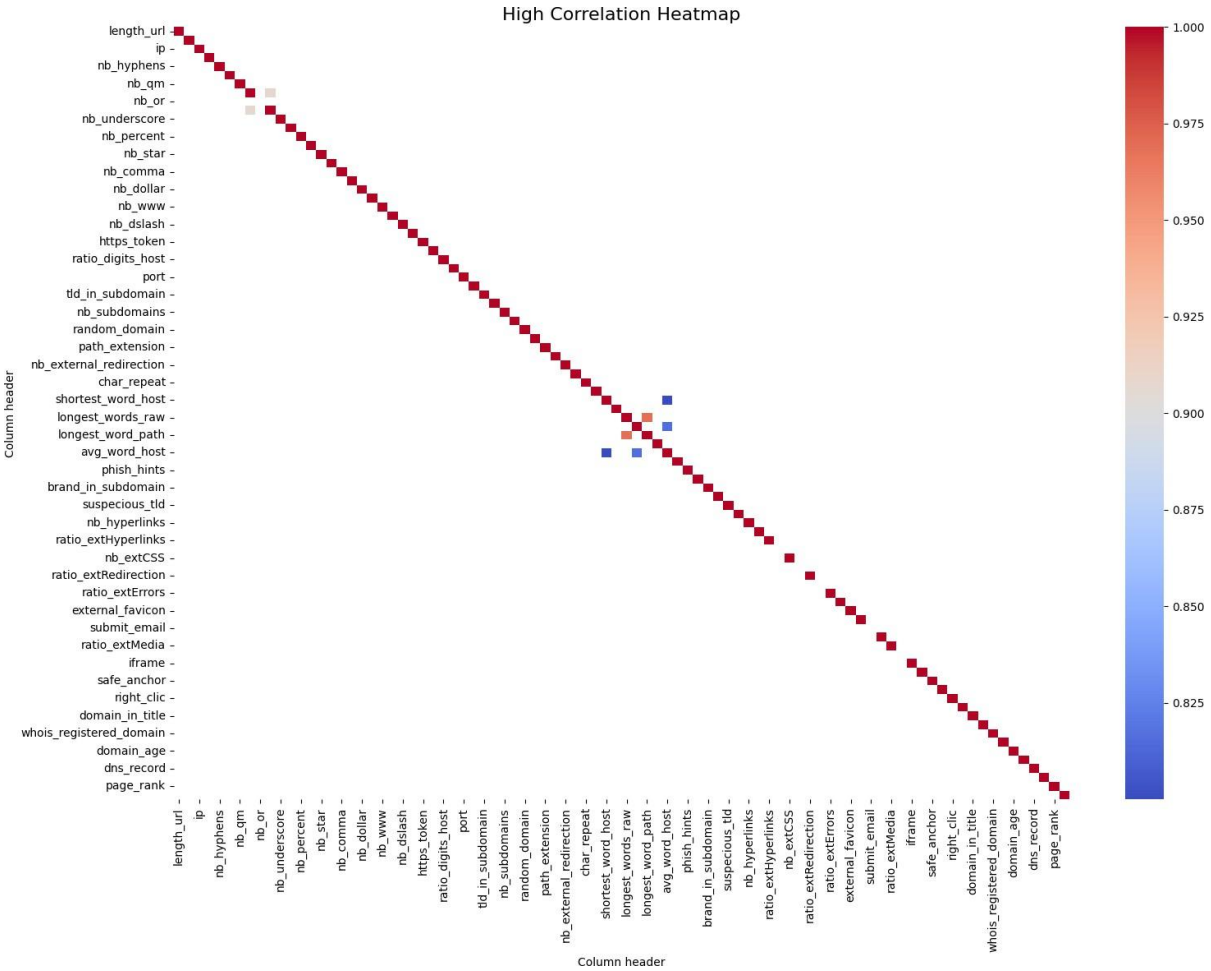
['url', 'length_url', 'length_hostname', 'ip', 'nb_dots', 'nb_hyphens', 'nb_at', 'nb_qm', 'nb_and', 'nb_or', 'nb_eq', 'nb_underscore', 'nb_tilde', 'nb_percent', 'nb_slash', 'nb_star', 'nb_colon', 'nb_comma', 'nb_semicolumn', 'nb_dollar', 'nb_space', 'nb_www', 'nb_com', 'nb_dslash', 'http_in_path', 'https_token', 'ratio_digits_url', 'ratio_digits_host', 'punycode', 'port', 'tld_in_path', 'tld_in_subdomain', 'abnormal_subdomain', 'nb_subdomains', 'prefix_suffix', 'random_domain', 'shortening_service', 'path_extension', 'nb_redirection', 'nb_external_redirection', 'length_words_raw', 'char_repeat', 'shortest_words_raw', 'shortest_word_host', 'shortest_word_path', 'longest_words_raw', 'longest_word_host', 'longest_word_path', 'avg_words_raw', 'avg_word_host', 'avg_word_path', 'phish_hints', 'domain_in_brand', 'brand_in_subdomain', 'brand_in_path', 'suspecious_tld', 'statistical_report', 'nb_hyperlinks', 'ratio_intHyperlinks', 'ratio_extHyperlinks', 'ratio_nullHyperlinks', 'nb_extCSS', 'ratio_intRedirection', 'ratio_extRedirection', 'ratio_intErrors', 'ratio_extErrors', 'login_form', 'external_favicon', 'links_in_tags', 'submit_email', 'ratio_intMedia', 'ratio_extMedia', 'sfh', 'iframe', 'popup_window', 'safe_anchor', 'onmouseover', 'right_clic', 'empty_title', 'domain_in_title', 'domain_with_copyright', 'whois_registered_domain', 'domain_registration_length', 'domain_age', 'web_traffic', 'dns_record', 'google_index', 'page_rank', 'status']


Website Distribution
6000
5000
4000
3000
2000
1000
0
No. of Websites
Legitimate (0)
Phishing (1)
Status

4 STEP: Ploting the correlation map for better knowledge about the dataset

```
# 4: ploting the correlation map for better knowledge about the dataset
corr = df.corr()
high_corr = corr[(corr > 0.8) | (corr < -0.8)]

plt.figure(figsize=(16, 12))
sns.heatmap(high_corr, cmap='coolwarm', annot=False)
plt.title("High Correlation Heatmap", fontsize=16)
plt.xlabel("Column header")
plt.ylabel("Column header")
plt.xticks(rotation=90)
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```



5 STEP: Preparing the x & y column

```
# 5: Preparing the x & y cloumn
X = df.drop(['url', 'status'], axis=1)
y = df['status']
```

6 STEP: Spliting dataset into two parts one for training and other for testing

```
# 6: Spliting the dataset (80:20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✓ 0.0s

7 STEP: StandardScaler is used for scaling the dataset and perform feature selection

```
# 7: Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

✓ 0.0s

8 STEP: Training the dataset with random forest.

```
# 8: Training the model with random forest as it is one of the best model used for regression
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

✓ 1.2s

9 STEP: Calculation part (confusion matrix, accuracy, etc) help us to know about the dataset. So, that we can perform any modification if needed.

```python
# 9: calculating the accuracy, precision, recall, f1 score and confusion matrix
print(f"Accuracy: {accuracy_score(y_test, y_pred):.2f}")
print(f"Precision: {precision_score(y_test, y_pred, pos_label=1):.2f}")
print(f"Recall: {recall_score(y_test, y_pred, pos_label=1):.2f}")
print(f"F1 Score: {f1_score(y_test, y_pred, pos_label=1):.2f}")
print("\nConfusion Matrix: ", confusion_matrix(y_test, y_pred))
print("\nClassification Report: ", classification_report(y_test, y_pred))
```

✓ 0.0s

```
Accuracy: 0.97
Precision: 0.98
Recall: 0.96
F1 Score: 0.97

Confusion Matrix:  [[1130   27]
 [  43 1086]]

Classification Report:               precision    recall  f1-score   support

           0       0.96      0.98      0.97      1157
           1       0.98      0.96      0.97      1129

    accuracy                           0.97      2286
   macro avg       0.97      0.97      0.97      2286
weighted avg       0.97      0.97      0.97      2286
```

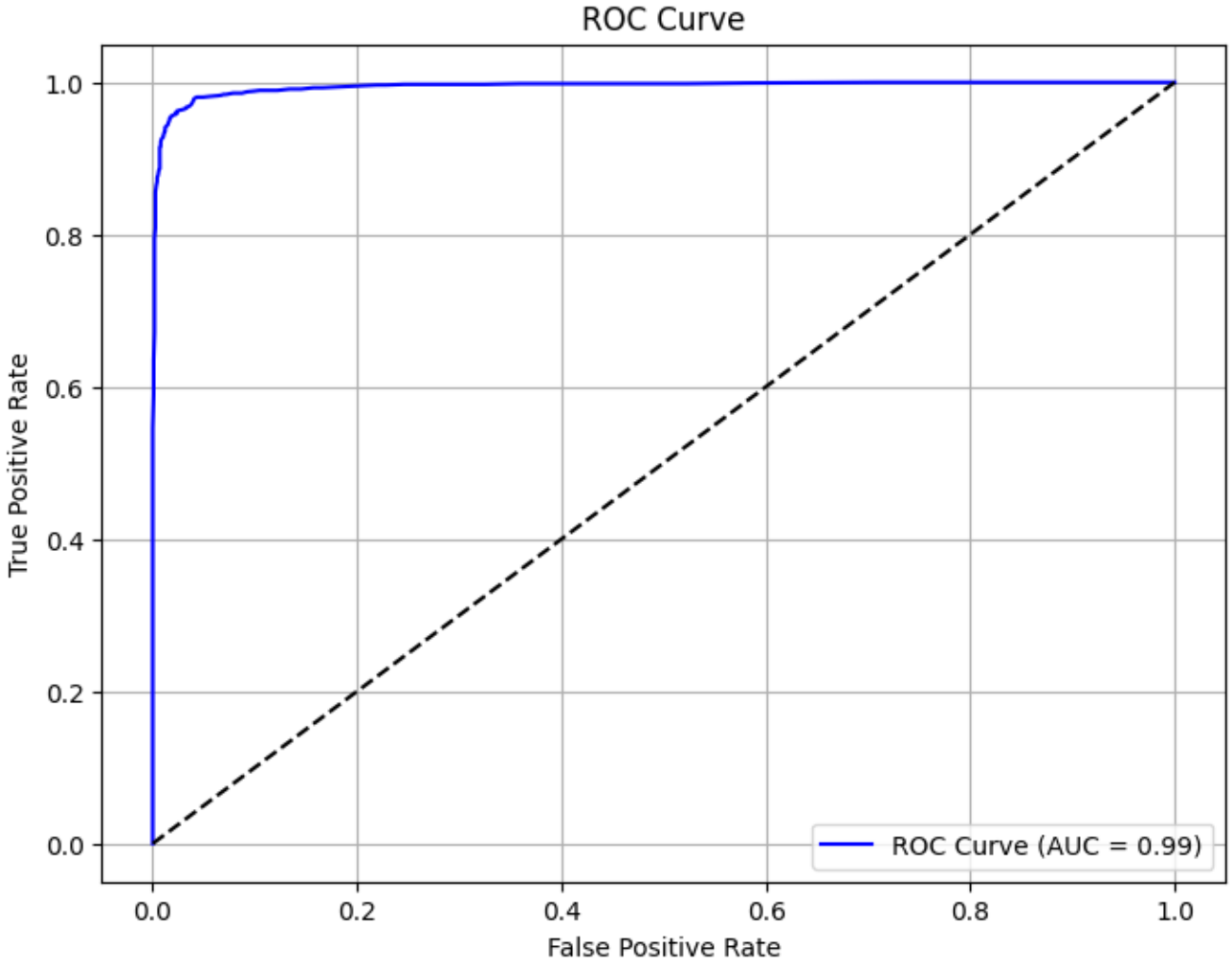
10 STEP :

```
y_pred = model.predict_proba(X_test)[:, 1]

# ROC-AUC
roc_auc = roc_auc_score(y_test, y_pred)
print("ROC AUC Score:", roc_auc)

# Plot ROC Curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.figure(figsize=(8,6))
plt.plot(fpr, tpr, label=f"ROC Curve (AUC = {roc_auc:.2f})", color="blue")
plt.plot([0, 1], [0, 1], "k--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("ROC Curve")
plt.legend()
plt.grid(True)
plt.show()
```

ROC AUC Score: 0.9944742710638751



11 STEP: Predicting the value with 5 random rows data

```python
# 10: predicting from random rows between actual and predicted

# 5 random rows
sample_indices = np.random.choice(len(X_test), size=5, replace=False)

print("Random sample indices from test set (position within X_test):", sample_indices)

(variable) y_sample_actual: Any
y_sample_actual = y_test.iloc[sample_indices]
y_sample_pred = model.predict(X_sample)

label_map = {0: 'Legitimate', 1: 'Phishing'}

for i, idx in enumerate(sample_indices):
    print(f"Sample {i+1} (Test set position: {idx}):")
    print(f"  Actual:   {label_map[y_sample_actual.iloc[i]]}")
    print(f"  Predicted:{label_map[y_sample_pred[i]]}")
    print()
```

✓ 0.0s

```
Random sample indices from test set (position within X_test): [1913  475 1398  427 1845]
Sample 1 (Test set position: 1913):
  Actual:   Phishing
  Predicted:Phishing

Sample 2 (Test set position: 475):
  Actual:   Legitimate
  Predicted:Legitimate

Sample 3 (Test set position: 1398):
  Actual:   Phishing
  Predicted:Phishing

Sample 4 (Test set position: 427):
  Actual:   Phishing
  Predicted:Phishing

Sample 5 (Test set position: 1845):
  Actual:   Phishing
  Predicted:Phishing
```

11 STEP : SAVING IN PICKLE FORMAT

```
import joblib

# Save the model
joblib.dump(model, "phishing_model.pkl")
print("Model saved as phishing_model.pkl")
```

OUTPUT: Model saved as phishing_model.pkl

Github link: https://github.com/SHREY275/codeb