```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
import warnings
warnings.filterwarnings("ignore")


# Step 1: Load dataset
df = pd.read_csv('/content/PhiUSIIL_Phishing_URL_Dataset.csv')
df.head()
```

	FILENAME	URL	URLLength	Domain	DomainLength	IsDomainIP	TLD	URLSimilarityIndex	Char
0	521848.txt	https://www.southbankmosaics.com	31	www.southbankmosaics.com	24	0	com	100.0	
1	31372.txt	https://www.uni-mainz.de	23	www.uni-mainz.de	16	0	de	100.0	
2	597387.txt	https://www.voicefmradio.co.uk	29	www.voicefmradio.co.uk	22	0	uk	100.0	
3	554095.txt	https://www.sfnmjournal.com	26	www.sfnmjournal.com	19	0	com	100.0	
4	151578.txt	https://www.rewildingargentina.org	33	www.rewildingargentina.org	26	0	org	100.0	

5 rows × 56 columns

```
# Step 2: Preprocess the data
# Check column names
print("Columns in dataset:", df.columns.tolist())

# Assuming 'URL' and 'label' are in the dataset
# Drop columns that are not numerical features
columns_to_drop = []
if 'URL' in df.columns:
    columns_to_drop.append('URL')
if 'FILENAME' in df.columns: # Add 'FILENAME' to the list of columns to drop
    columns_to_drop.append('FILENAME')

df = df.drop(columns_to_drop, axis=1)


# Encode labels if not already numeric
# Changed 'Label' to 'label'
if df['label'].dtype == 'object':
    le = LabelEncoder()
    df['label'] = le.fit_transform(df['label'])
```

```
Columns in dataset: ['FILENAME', 'URL', 'URLLength', 'Domain', 'DomainLength', 'IsDomainIP', 'TLD', 'URLSimilarityIndex', 'CharContinuat
```

```
# Step 3: Train/test split
# Changed 'Label' to 'label'
X = df.drop('label', axis=1)
y = df['label']

# Ensure all feature columns are numeric
for col in X.columns:
    if X[col].dtype == 'object':
        print(f"Warning: Column '{col}' is still an object type and will be dropped.")
        X = X.drop(col, axis=1)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
Warning: Column 'Domain' is still an object type and will be dropped.
Warning: Column 'TLD' is still an object type and will be dropped.
Warning: Column 'Title' is still an object type and will be dropped.
```

```
# Step 4: Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

```
▾   RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
# Step 5: Evaluate model
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 1.0

Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00     20124
           1       1.00      1.00      1.00     27035

    accuracy                           1.00     47159
   macro avg       1.00      1.00      1.00     47159
weighted avg       1.00      1.00      1.00     47159
```

```
# Step 6: Prediction from user input
print("\n--- URL CLASSIFICATION ---")
input_url = input("Enter a URL to check if it is Phishing or Legit: ")

# Extract simple features manually from the input URL
import re
def extract_features(url):
    return pd.DataFrame([{
        'Length': len(url),
        'NumDots': url.count('.'),
        'HasAt': int("@" in url),
        'HasHttps': int("https" in url),
        'HasHttp': int("http" in url),
        'NumDigits': len(re.findall(r'\d', url)),
        'NumSpecialChar': len(re.findall(r'[^A-Za-z0-9]', url)),
    }])

# Example Feature Matching (simplified for live input prediction)
features = extract_features(input_url)
# Align with training features if feature names differ
features = features.reindex(columns=X.columns, fill_value=0)

# Predict
prediction = model.predict(features)[0]
print("\nResult: The URL is", "🔴 PHISHING (Bad)" if prediction == 1 else "🟢 LEGITIMATE (Good)")
```

```
--- URL CLASSIFICATION ---
Enter a URL to check if it is Phishing or Legit: https://docs.github.com/en/search-github/github-code-search/understanding-github-code-s

Result: The URL is 🟢 LEGITIMATE (Good)
```