Phishing URL Detection Project

Libraries

In [ ]:
```
# Required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, classification_report, confusion_mat
from sklearn.model_selection import GridSearchCV
from sklearn.impute import SimpleImputer
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_selection import SelectKBest, chi2
```

Dataset loading, preprocessing and consolidation

In [ ]:
```
# load datasets
dataset_1 = pd.read_csv("medical_keywords_dset.csv")
dataset_2 = pd.read_csv("urls_medical_keywords.csv")

# ensure consistent labels
dataset_1["Label"] = dataset_1["Label"].map({"Benign": 0})
dataset_2["status"] = dataset_2["status"].map({"legitimate": 0, "phishing": 1})

# concatenate datasets
consolidated_dataset = pd.concat([dataset_1, dataset_2.rename(columns={"status":

# drop rows with missing target values
consolidated_dataset.dropna(subset=["Label", "url"], inplace=True)

# feature engineering
# extract length feature for URLs
consolidated_dataset["url_length"] = consolidated_dataset["url"].apply(len)

# split dataset into features (X) and target variable (y)
X = consolidated_dataset["url"]
y = consolidated_dataset["Label"]
```

Utilize 10-fold cross-validation and feature selection

In [ ]:
```
# initialize the tf-idf vectorizer
tfidf_vectorizer = TfidfVectorizer()

# transform the entire dataset
X_tfidf = tfidf_vectorizer.fit_transform(X)

# select top k features using chi-squared test
k_best = 10
selector = SelectKBest(chi2, k=k_best)
```

```
X_tfidf_selected = selector.fit_transform(X_tfidf, y)

# get indices of selected features
selected_feature_indices = selector.get_support(indices=True)

# get names of selected features
selected_feature_names = np.array(tfidf_vectorizer.get_feature_names_out() + ['u

# display selected features
print(f"Top {k_best} selected features:")
for feature_name in selected_feature_names:
    print("-", feature_name)

# initialize Random Forest classifier
rf_model = RandomForestClassifier(random_state=42)

# perform 10-fold cross-validation to measure accuracies
cv_scores = cross_val_score(rf_model, X_tfidf_selected, y, cv=10)

# output mean accuracy from cross-validation
print("Mean accuracy with 10-fold cross-validation:", cv_scores.mean())
```

```
Top 10 selected features:
- includesurl_length
- indexurl_length
- jsurl_length
- loginurl_length
- phpurl_length
- testurl_length
- uiurl_length
- v4url_length
- wpurl_length
- x4xurl_length
Mean accuracy with 10-fold cross-validation: 0.8658119658119657
```

Feature importance extraction

In [ ]:

```
# train the model on the entire dataset
rf_model.fit(X_tfidf_selected, y)

# make predictions on the entire dataset
y_pred = rf_model.predict(X_tfidf_selected)

# compute confusion matrix
conf_matrix = confusion_matrix(y, y_pred)

# display confusion matrix
print("Confusion Matrix:")
print(conf_matrix)

# extract feature importances
feature_importances = pd.DataFrame(rf_model.feature_importances_, index=selected

# display top 10 important features
print("\nTop 10 Important Features:")
print(feature_importances.sort_values(by="Importance", ascending=False).head(10)
```

```
Confusion Matrix:
[[193   0]
 [ 27  48]]

Top 10 Important Features:
                    Importance
testurl_length        0.311483
phpurl_length         0.205354
wpurl_length          0.153440
includesurl_length    0.108920
indexurl_length       0.088838
loginurl_length       0.086806
jsurl_length          0.030065
uiurl_length          0.006087
v4url_length          0.005775
x4xurl_length         0.003231
```

Using other methods like Decision Tree, KNN, Naive Bayes

In [ ]:
```
# initialize classifiers
dt_model = DecisionTreeClassifier()
knn_model = KNeighborsClassifier()
nb_model = MultinomialNB()

# train and evaluate Decision Tree model
dt_scores = cross_val_score(dt_model, X_tfidf_selected, y, cv=10)
print("Mean accuracy of Decision Tree:", dt_scores.mean())

# train and evaluate KNN model
knn_scores = cross_val_score(knn_model, X_tfidf_selected.toarray(), y, cv=10)
print("Mean accuracy of KNN:", knn_scores.mean())

# train and evaluate Naive Bayes model
nb_scores = cross_val_score(nb_model, X_tfidf_selected, y, cv=10)
print("Mean accuracy of Naive Bayes:", nb_scores.mean())
```

```
Mean accuracy of Decision Tree: 0.8508547008547008
Mean accuracy of KNN: 0.8656695156695156
Mean accuracy of Naive Bayes: 0.7202279202279203
```