Microsoft Elevate

CAPSTONE PROJECT

Phishing Website Detection

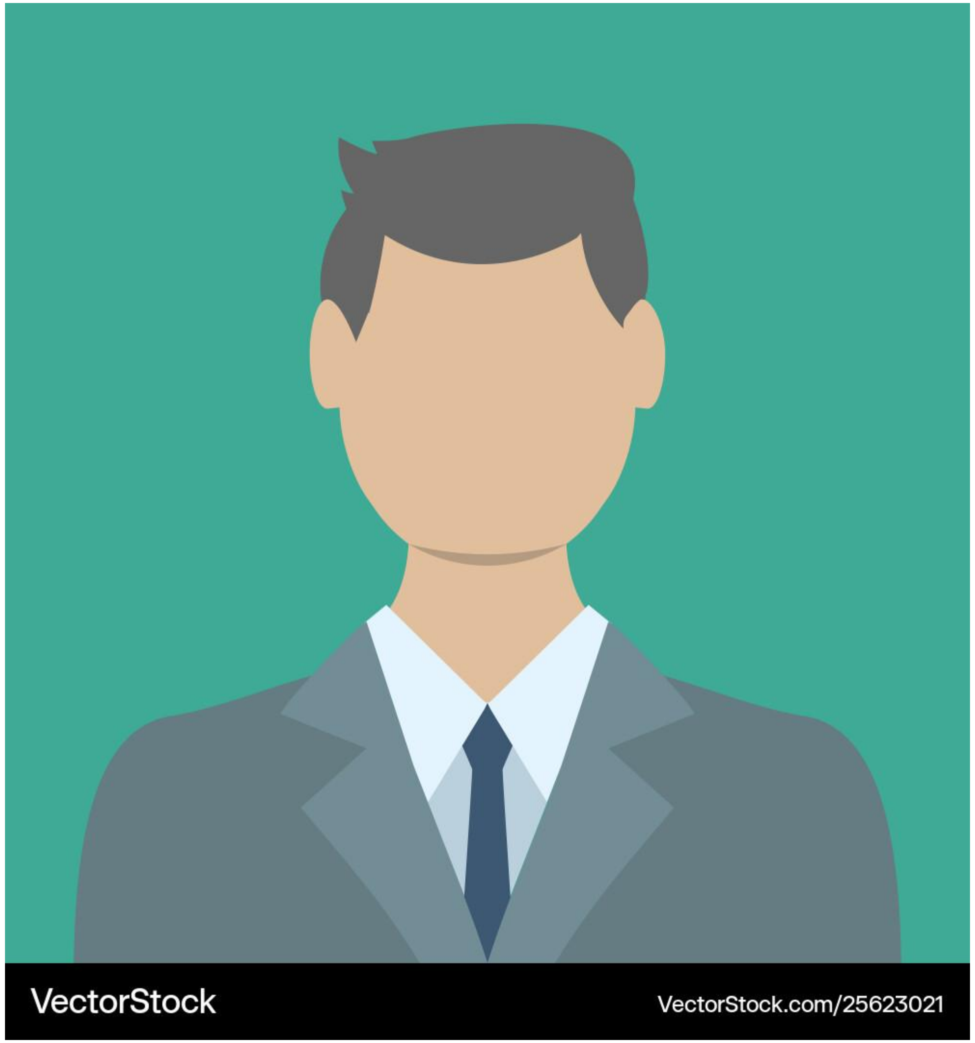
PRESENTED BY

STUDENT NAME:MEDARI BHARATH KUMAR

COLLEGE NAME: INDIAN INSTITUTE OF INFORMATION TECHNOLOGY KOTTAYAM

DEPARTMENT:BTECH-CSE-CYBER SECURITY

EMAIL ID: MEDARIBHARATHKUMAR@GMAIL.COM



OUTLINE:

- **Problem Statement** (Should not include solution)
- **Proposed System/Solution**
- **System Development Approach** (Technology Used)
- **Algorithm & Deployment**
- **Result (Output Image)**
- **Conclusion**
- **Future Scope**
- **References**

PROBLEM STATEMENT:

- Currently, phishing attacks are one of the most common cybersecurity threats. Fraudulent websites are designed to mimic legitimate websites and steal sensitive information such as passwords, credit card details, and personal data.

- Users often cannot differentiate between legitimate and phishing websites.

- The challenge is to automatically detect whether a website is phishing or legitimate using data-driven techniques.

PROPOSED SOLUTION:

- The system uses **Machine Learning** to classify websites as:
- **Legitimate**
- **Phishing**

◆ **Workflow**

- **Data Collection:** 10,000 labeled websites
- **Preprocessing:** Removed Domain column, cleaned data
- **Features Used:** URL length, IP presence, HTTPS, redirection, DNS, domain age, iFrame, etc.
- **Model:** Random Forest (80% train / 20% test)
- **Accuracy Achieved:** 86%
- **Real-Time Testing:** User enters URL → model predicts result
- **Deployment:** Model saved for integration into apps or browser tools

SYSTEM APPROACH:

- **System Requirements (Local System)**
- Operating System: Windows / Linux / macOS
- Processor: Intel i3 or higher
- RAM: Minimum 4GB (8GB recommended)
- Storage: 1GB free space
- Python Version: Python 3.x

Development Environment

- Google Colab (Cloud-based)
  OR
- VS Code / Jupyter Notebook

Libraries Used

Pandas,numpy,scikit-learn,Matplotlib,Seaborn,Joblib,re (Regex),urllib

ALGORITHM & DEPLOYMENT:

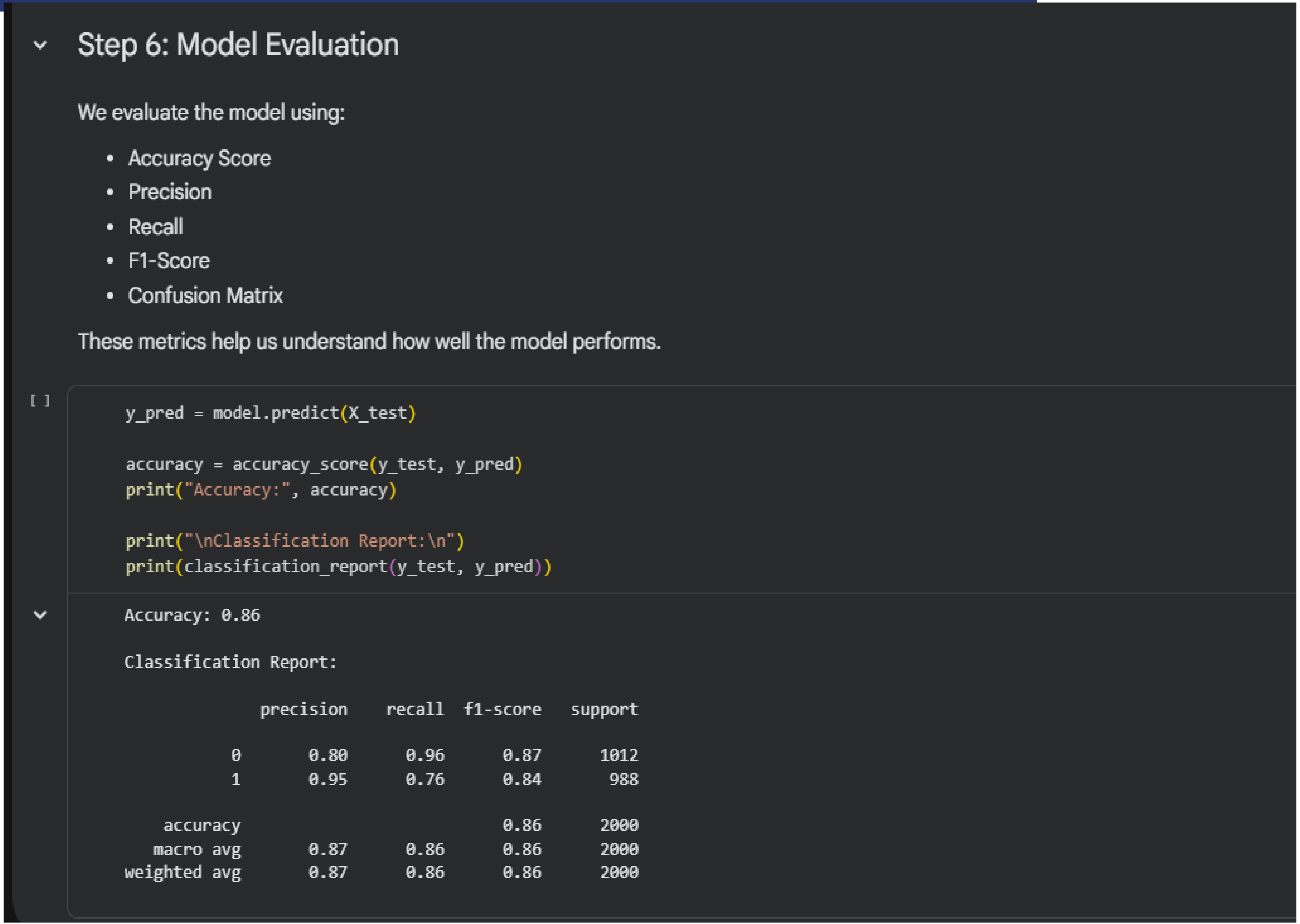
: Algorithm Selection:

- **Algorithm Selection:**
  - Random Forest Classifier
  - Ensemble of decision trees
  - Reduces overfitting
  - Provides feature importance
- **Data Input:**
  - URL Length & Depth
  - IP Address & HTTPS
  - Redirection & Prefix/Suffix
  - DNS Record & Domain Age
  - iFrame & Mouse behavior
  - Output:
    0 → Legitimate | 1 → Phishing

- **Training Process:**
- Dataset: 10,000 labeled websites
- Train-Test Split: 80% / 20%
- Model Parameters:
  - n_estimators = 300
  - max_depth = 20
- **Prediction Process:**
  - User enters website URL
  - System extracts numerical features
  - Features passed to trained model
  - Model predicts phishing or legitimate

RESULT:

- **Model Performance**
- Accuracy: 86%
- Precision (Phishing): 0.95
- Recall (Phishing): 0.76
- F1-Score: 0.84



Step 6: Model Evaluation

We evaluate the model using:

- Accuracy Score
- Precision
- Recall
- F1-Score
- Confusion Matrix

These metrics help us understand how well the model performs.

```
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.86

Classification Report:

              precision    recall  f1-score   support

           0       0.80      0.96      0.87      1012
           1       0.95      0.76      0.84       988

    accuracy                           0.86      2000
   macro avg       0.87      0.86      0.86      2000
weighted avg       0.87      0.86      0.86      2000
```

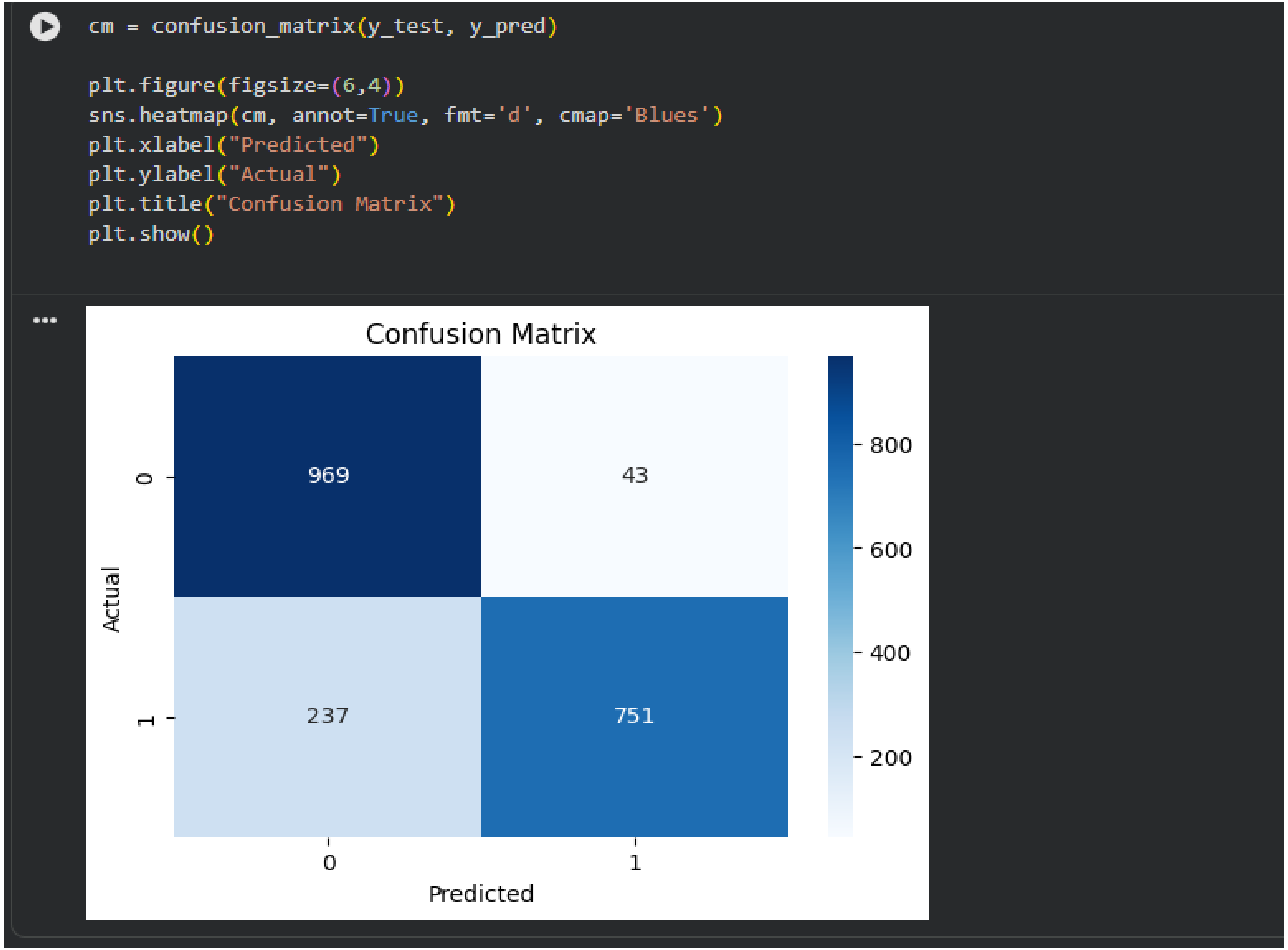
Confusion Matrix Analysis

•True Legitimate: 969

•True Phishing: 751

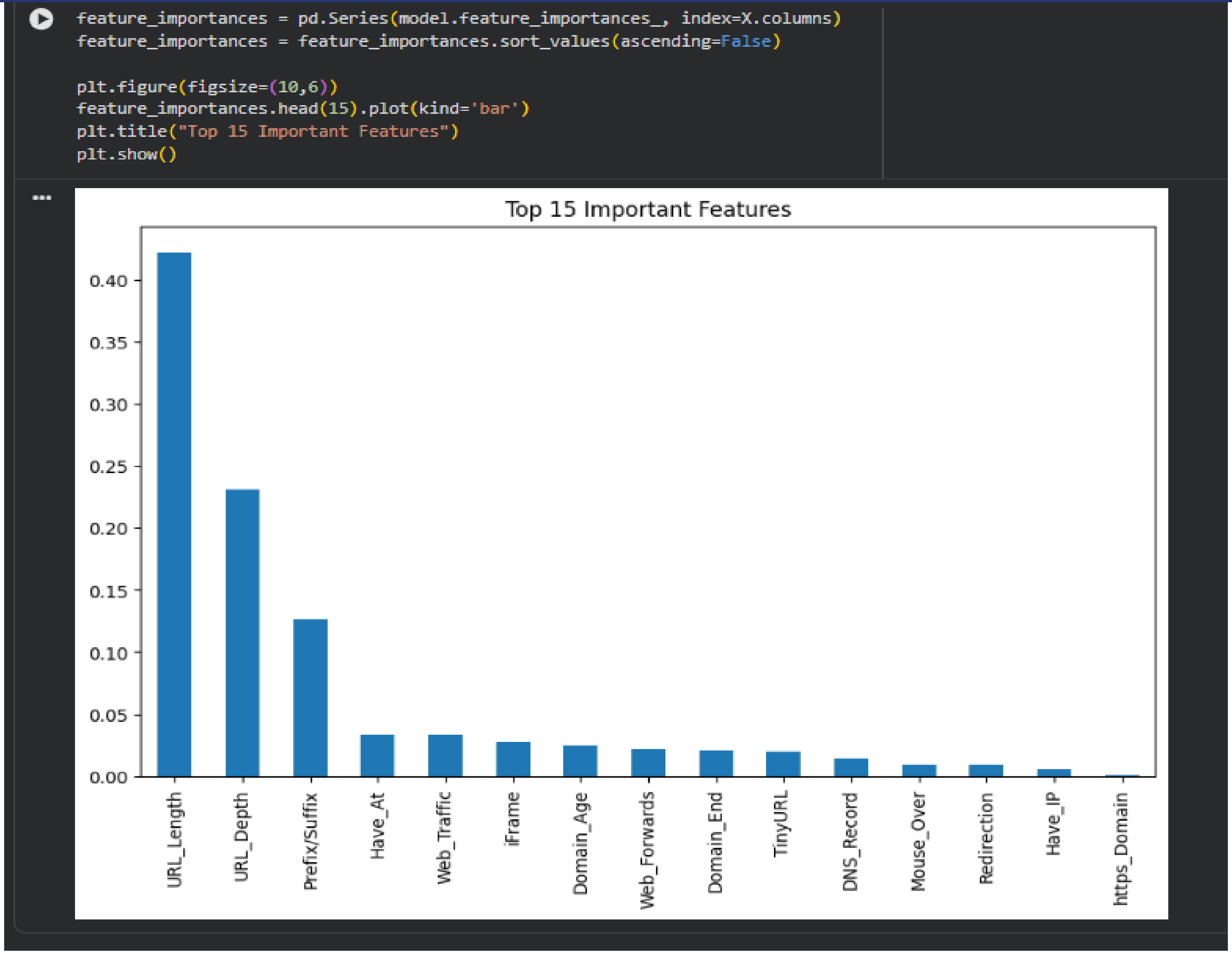
•False Positives: 43

•False Negatives: 237



```
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



```
feature_importances = pd.Series(model.feature_importances_, index=X.columns)
feature_importances = feature_importances.sort_values(ascending=False)

plt.figure(figsize=(10,6))
feature_importances.head(15).plot(kind='bar')
plt.title("Top 15 Important Features")
plt.show()
```

CONCLUSION:

- Successfully developed a phishing website detection system using Machine Learning.
- Implemented Random Forest classifier with 86% accuracy.
- Identified key features such as URL length and URL depth for detecting phishing patterns.
- Demonstrated real-time URL prediction capability.
- The system shows how AI can enhance cybersecurity and prevent phishing attacks.

FUTURE SCOPE:

- Improve model accuracy using advanced algorithms (XGBoost, Gradient Boosting).
- Apply hyperparameter tuning and cross-validation.
- Integrate WHOIS lookup for real-time domain age detection.
- Develop a web application or browser extension.
- Expand system for real-time enterprise cybersecurity monitoring.

REFERENCES:

- Scikit-learn Documentation – https://scikit-learn.org
- Pandas Documentation – https://pandas.pydata.org
- Phishing Dataset (GitHub Repository)
- Research articles on Phishing Website Detection using Machine Learning
- Python Official Documentation – https://docs.python.org

- GitHub Link:Link
- Collab Link: Link

Thank You