AI-BASED THREAT INTELLIGENCE AND PREDICTION SYSTEM

Anshika Tyagi

CONTENTS

1. INTRODUCTION

The spread of cyberthreats in the modern era presents serious difficulties for businesses all around the world. In many cases, sophisticated attacks are difficult to detect and mitigate using conventional cybersecurity techniques. The goal of this project is to create an AI-Based Threat Intelligence and Prediction System that will improve cyber threat detection and prediction by utilizing machine learning techniques. By analyzing phishing site URLs, this system will categorize them as benign or malicious, taking a proactive approach to cybersecurity.

2. BACKGROUND

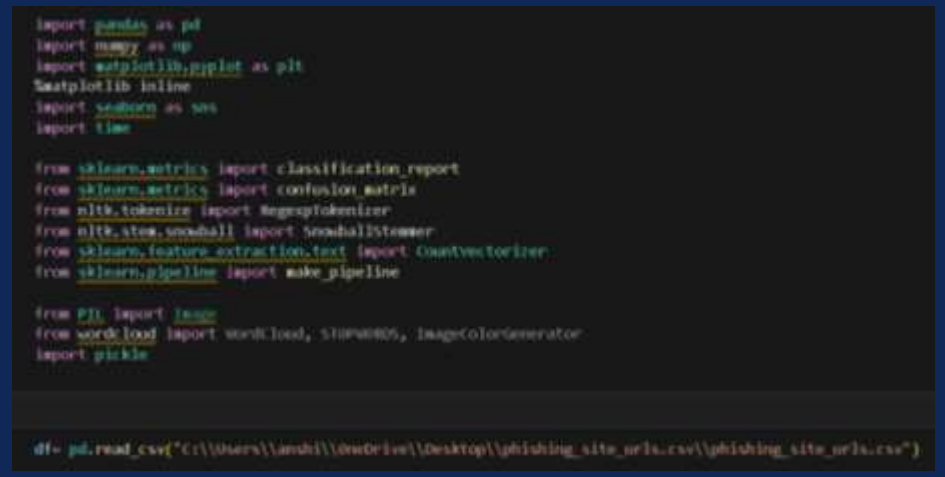
The process of obtaining, evaluating, and interpreting data regarding present and future risks to an organization is known as threat intelligence. It seeks to offer practical insights to improve security posture. Conventional approaches frequently depend on manual procedures and static rules, which are inadequate to counter threats that are dynamic and ever-changing.

Advanced threat intelligence capabilities are provided by machine learning (ML) and artificial intelligence (AI). Artificial intelligence (AI)-based systems can improve threat detection speed and accuracy by analyzing large datasets, finding patterns, and making predictions. The goal of this project is to categorize phishing URLs—which are frequently used in cyberattacks to trick users into divulging sensitive information—using artificial intelligence techniques.

3. IMPLEMENTATION DETAILS

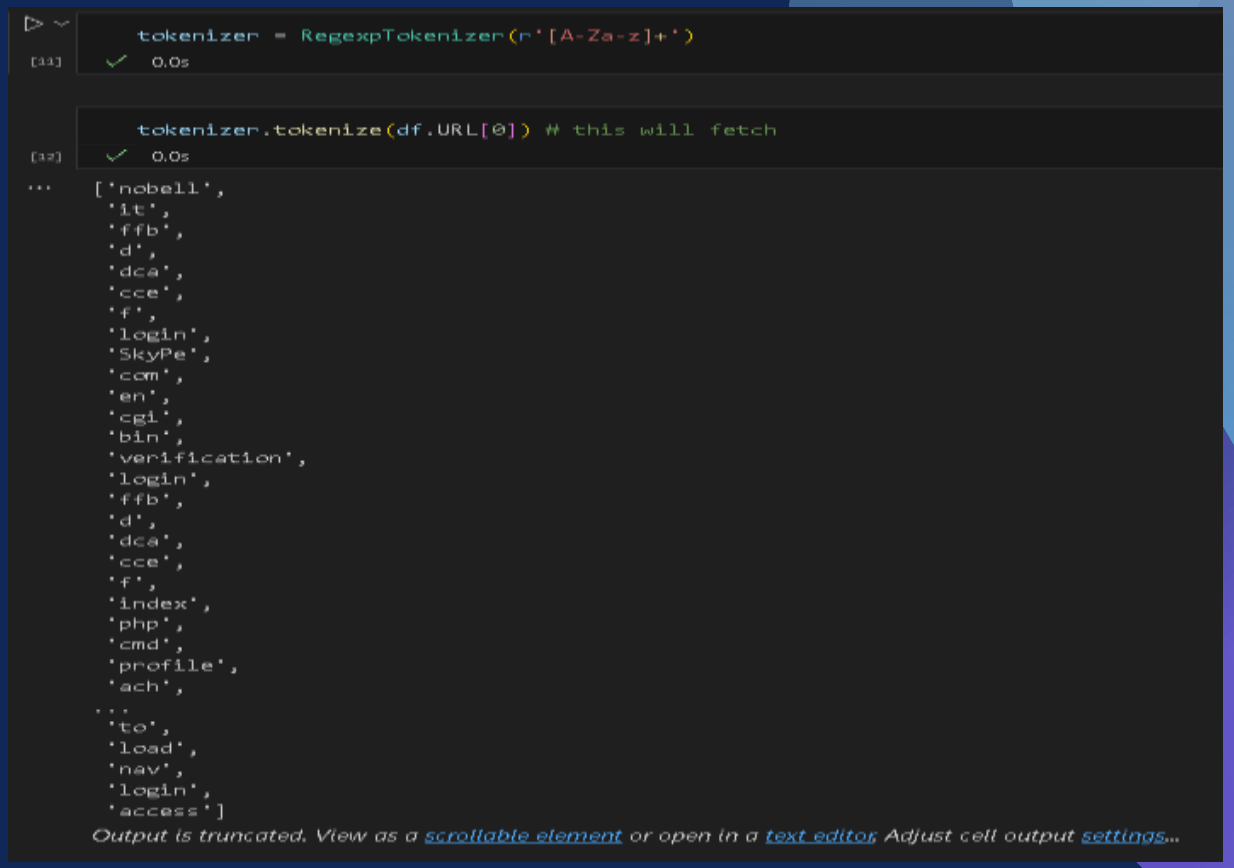
3.1 Data Collection and Preprocessing: The dataset used in this project consists of phishing site URLs, sourced from Kaggle. The implementation involves several key steps:

- Data Loading: The dataset is loaded using Pandas.
- Exploratory Data Analysis (EDA): Understanding the dataset through basic statistics and visualizations.
- Tokenization: Splitting URLs into individual tokens.
- Stemming: Reducing words to their base or root form.
- Joining Tokens: Reconstructing URLs from tokens for further processing.



3.2 Data PreprocessingTokenization:

- **Using RegexpTokenizer to split URLs into tokens.**
- **Stemming: Using SnowballStemmer to reduce tokens to their root forms.Joining**
- **Tokens: Combining tokens back into strings.**



```
print('Getting words tokenized ...')
t0= time.perf_counter()
df['text_tokenized'] = df.URL.map(lambda t: tokenizer.tokenize(t))
t1 = time.perf_counter() - t0
print('Time taken',t1 ,'sec')
```

✓ 12s

```
Getting words tokenized ...
Time taken 1.2950465000003533 sec
```

```
df.sample(10)
```

✓ 00s

	URL	Label	text_tokenized
126965	royalgateenergy.com/wp-admin/js/images/httpdoc...	bad	[royalgateenergy, com, wp, admin, js, images, ...
175665	en.wikipedia.org/wiki/Hy_Sandham	good	[en, wikipedia, org, wiki, Hy, Sandham]
301121	cduniverse.com/productinfo.asp?pid=1168879	good	[cduniverse, com, productinfo, asp, pid]
248876	torontobeerweek.com/tbw-store/	good	[torontobeerweek, com, tbw, store]
480909	loot.co.za/index/html/index2144.html	good	[loot, co, za, index, html, index, html]
275979	amazon.com/Locus-1-Patrick-Fillion/dp/1897102577	good	[amazon, com, Locus, Patrick, Fillion, dp]
432577	smhc.qc.ca/en/research	good	[smhc, qc, ca, en, research]
534815	42roza.ru/	bad	[roza, ru]
338883	flickr.com/photos/12150532@N04/4700249211/	good	[flickr, com, photos, N]
490338	dontskid.me/netboot/cp.php?m=login	bad	[dontskid, me, netboot, cp, php, m, login]

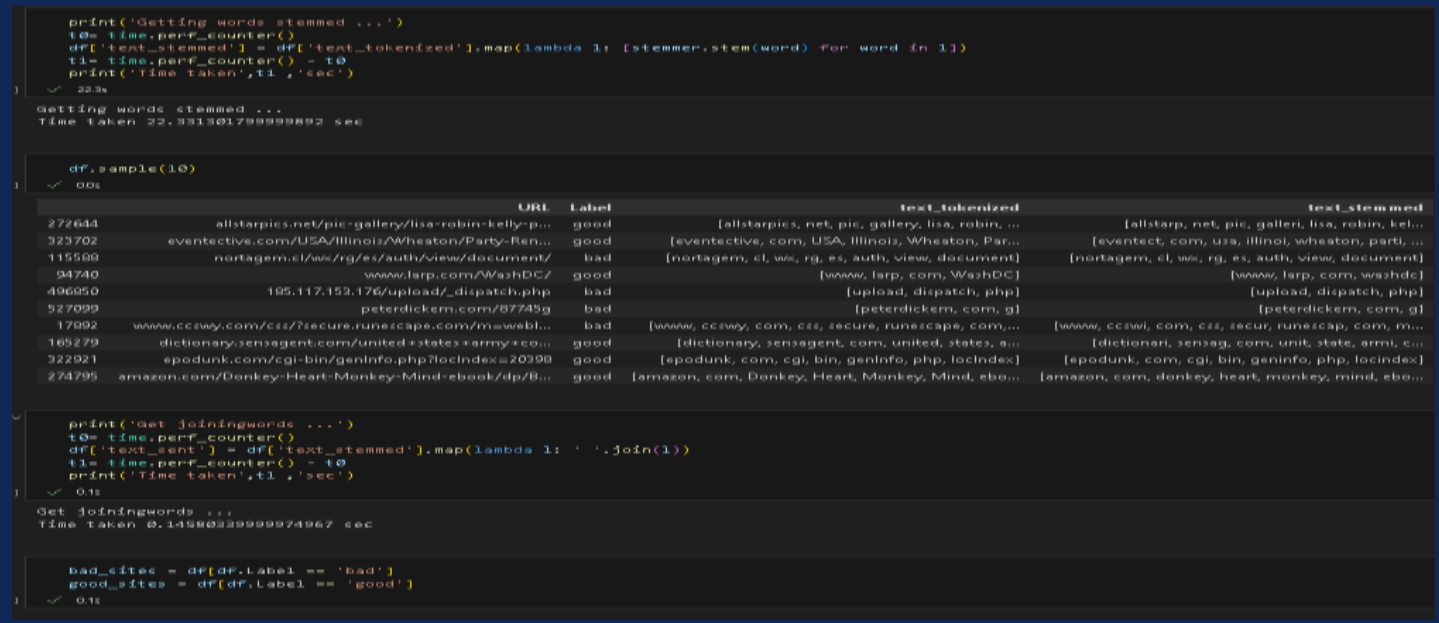
```
stemmer = SnowballStemmer("english")
```

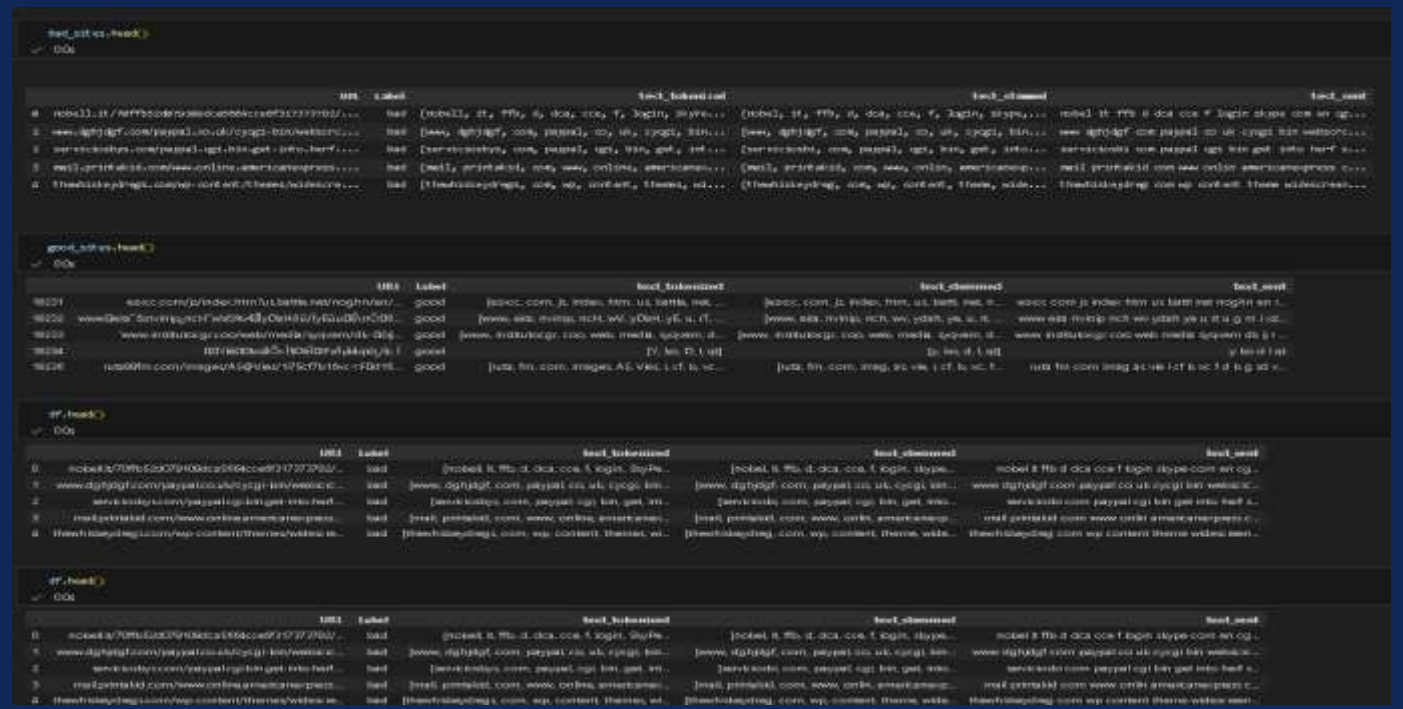
✓ 00s

```
print('Getting words stemmed ...')
t0= time.perf_counter()
df['text_stemmed'] = df['text_tokenized'].map(lambda l: [stemmer.stem(word) for word in l])
t1= time.perf_counter() - t0
print('Time taken',t1 ,'sec')
```

✓ 223s

```
Getting words stemmed ...
Time taken 22.331301799999892 sec
```





AI-Based Threat Intelligence and Prediction System

3.3 Feature Extraction Count Vectorizer :

Converting text data into numerical features using Count Vectorizer.c

```
cv = CountVectorizer()
✓ 0.0s
```

```
feature = cv.fit_transform(df.text_sent)
✓ 3.7s
```

```
feature[:5].toarray()
✓ 0.0s
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

3.4 Model Training and Evaluation:

- **Logistic Regression: Training and evaluating a logistic regression model.**
- **Multinomial Naive Bayes: Training and evaluating a multinomial Naive Bayes model.**

```
from sklearn.model_selection import train_test_split
✓ 0.0s
```

```
trainX, testX, trainY, testY = train_test_split(feature, df.Label)
✓ 0.0s
```

```
from sklearn.linear_model import LogisticRegression
✓ 0.0s
```

```
lr = LogisticRegression()
lr.fit(trainX,trainY)
✓ 7.3s
```

```
e:\Python\Lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
LogisticRegression()
```

3.5 Model Comparison

- **Accuracy Comparison: Comparing the accuracy of different models using a bar plot.**

```
from sklearn.naive_bayes import MultinomialNB
```
✓ 0.0s

```
mnb = MultinomialNB()
```
✓ 0.0s

```
mnb.fit(trainX,trainY)
```
✓ 0.9s

MultinomialNB
MultinomialNB()

```
mnb.score(testX,testY)
```
✓ 0.2s

```
0.9586200368436765
```

```
Scores_ml['MultinomialNB'] = np.round(mnb.score(testX,testY),2)
```
✓ 0.2s

```
from sklearn.linear_model import LogisticRegression
```
✓ 0.0s

```
lr = LogisticRegression()
lr.fit(trainX,trainY)
```
✓ 7.3s

```
e:\Python\lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

LogisticRegression
LogisticRegression()

```
lr.score(testX,testY)
```
✓ 0.2s

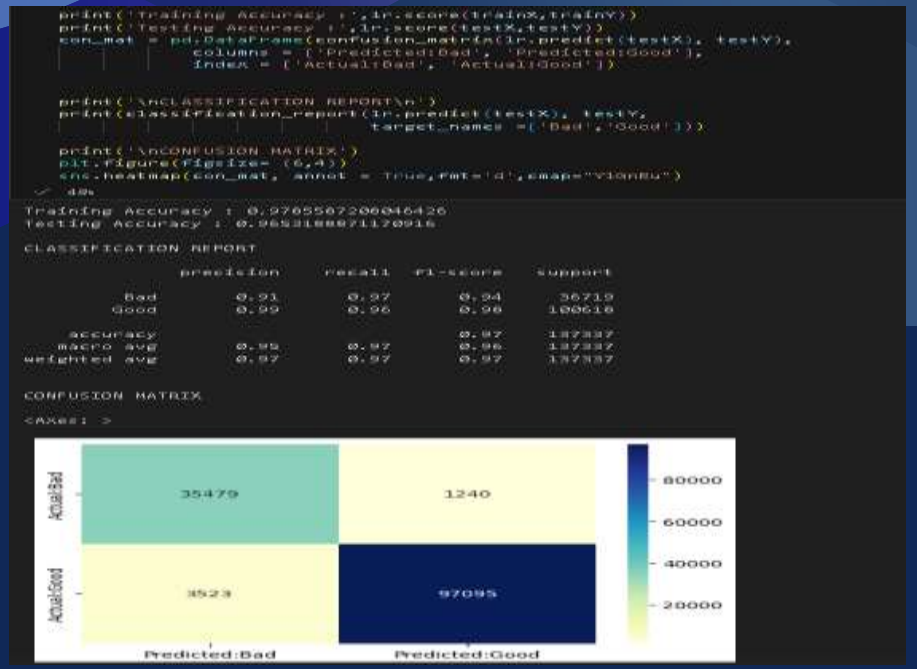
```
0.9653188871170916
```

```
Scores_ml = {}
Scores_ml['Logistic Regression'] = np.round(lr.score(testX,testY),2)
```
✓ 0.1s

4. RESULTS AND ANALYSIS

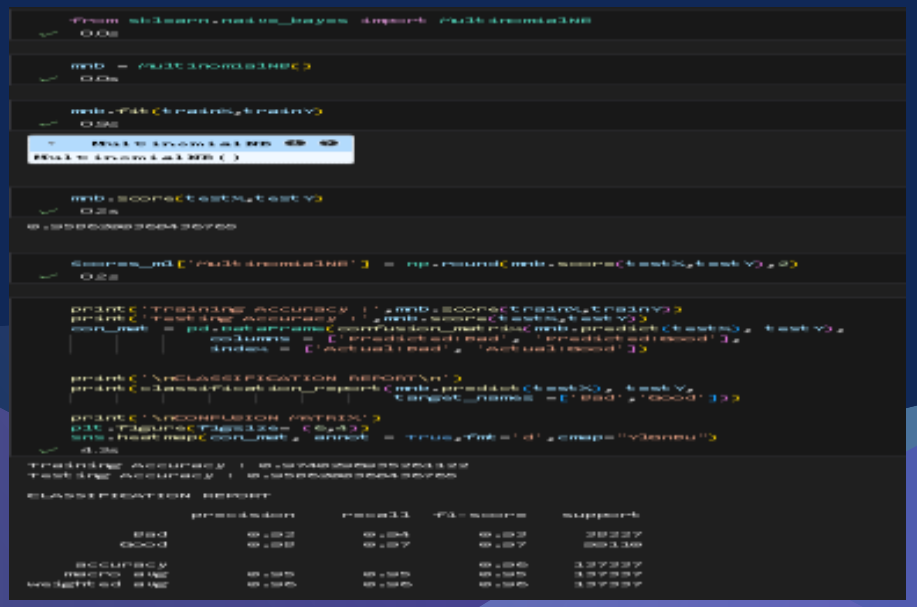
4.1 Logistic Regression

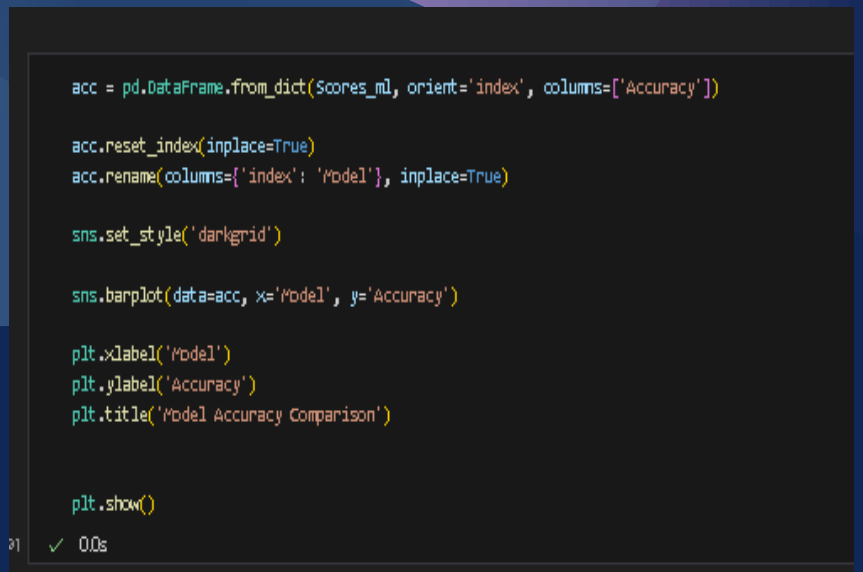
- Training Accuracy: 98%
- Testing Accuracy: 97%

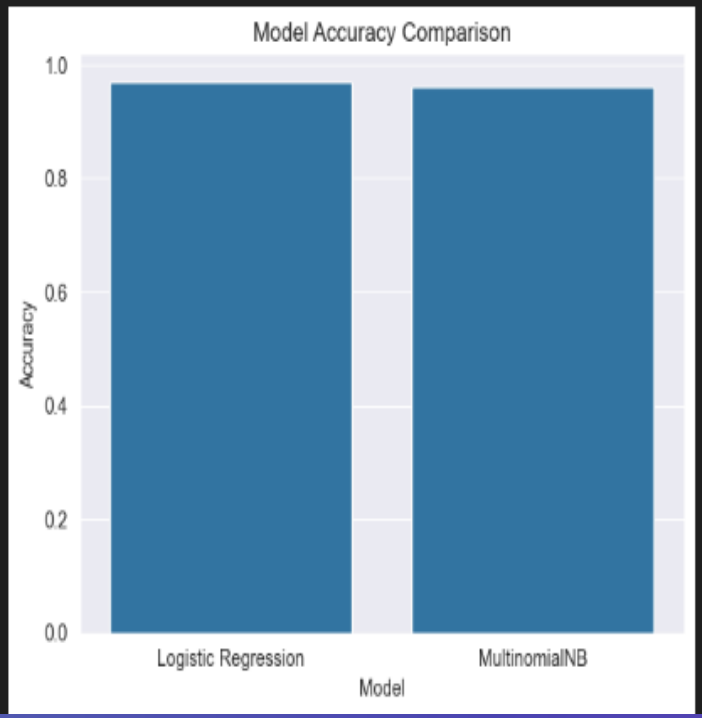


4.2 Multinomial Naive Bayes

- Training Accuracy: 96%
- Testing Accuracy: 95%







5. CONCLUSION

The AI-Based Threat Intelligence and Prediction System demonstrated effective performance in classifying phishing URLs using machine learning techniques. Both Logistic Regression and Multinomial Naive Bayes models showed high accuracy, with Logistic Regression slightly outperforming in this context. The system can be further enhanced by incorporating additional features and advanced algorithms to improve prediction accuracy and robustness.

REFERENCES

- Books:

"Artificial Intelligence in Cybersecurity" by Leslie F. Sikos.

"Machine Learning and Security" by Clarence Chio and David Freeman.

- Articles:

"The Role of Artificial Intelligence in Cyber Security" by John A. Clark.

"Predictive Threat Intelligence Using Machine Learning" by Karen Scarfone.

- Websites:
  - Darktrace
  - QRadar