Phishing Detector using AI Report

Introduction

Phishing is a significant cyber threat where attackers deceive victims into revealing sensitive information and data by pretending to be some trustworthy source. This can occur via email, text, phone calls, or even social media platforms. Over 3.4 billion phishing emails are sent daily, which incurred a loss of over $52 million in 2023 alone. Between individuals and businesses, phishing remains a reoccurring issue with no end in sight. This project aims to mitigate the impact of phishing by utilizing artificial intelligence to detect phishing emails and prevent them from reaching the user.

Objectives

The objective of this project is to design an AI-powered phishing detector. More specifically, the goal is to reduce user exposure to phishing attacks by identifying malicious emails before they are seen by the user. Another goal is to enhance overall email security and protect sensitive information.

Methodology

I used a phishing email dataset from Kaggle to train the AI model. This dataset contains 18,650 emails, of which 61% of the dataset is safe emails and the other 39% are phishing emails. I then proceeded to split this data into two groups. The first group is the training group, made up of 11,322 emails, which the model used to train itself on distinguishing safe and phishing emails. The other group is the evaluation group, made up of 7,328 emails, which was used to test how effective the model is at distinguishing the two.

Getting into the project workflow, I began by importing the necessary libraries, as well as downloading the Natural Language Toolkit (NLTK) which is capable of identifying key words used in the detection of phishing emails. Then, I loaded the dataset into the

program.

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
from sklearn.preprocessing import label_binarize
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import nltk
import tempfile
import seaborn as sns
import matplotlib.pyplot as plt

# Ensure NLTK resources are downloaded
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')

# Load dataset
dataset_path = r"/content/drive/MyDrive/Phishing_Email.csv"  # Update with the correct path
df = pd.read_csv(dataset_path)
```

From there, I began preprocessing the data by filling blank values with empty strings, tokenizing text to facilitate analysis of relevant words, and filtered out stop words (words like “is”, “the”, etc which are not relevant).

```
# Replace missing or non-string values with an empty string
df['Email Text'] = df['Email Text'].fillna('').astype(str)

# Preprocessing function
def preprocess_text(text):
    stop_words = set(stopwords.words('english'))
    tokens = word_tokenize(text.lower())  # Tokenize and convert to lowercase
    filtered_tokens = [word for word in tokens if word.isalnum() and word not in stop_words]
    return ' '.join(filtered_tokens)

# Apply preprocessing
df['cleaned_text'] = df['Email Text'].apply(preprocess_text)
```

Then I distributed the emails into binary classes. 0 represents safe emails and 1 represents phishing emails. Another important aspect was converting the text into numerical representations using feature extraction so that the model can actually evaluate the text.

```
# Map labels to binary values (e.g., 1 for phishing, 0 for safe)
df['label'] = df['Email Type'].map({'Phishing Email': 1, 'Safe Email': 0})

# Class distribution
class_distribution = df['label'].value_counts()
print("Class Distribution:")
print(class_distribution)

# Feature extraction
vectorizer = TfidfVectorizer(max_features=5000)
X = vectorizer.fit_transform(df['cleaned_text']).toarray()
y = df['label']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Print training and evaluation dataset sizes
print(f"\nNumber of emails used for training: {len(X_train)}")
print(f"Number of emails used for evaluation: {len(X_test)}")
```

At this point the process of training the model is ready to begin. I utilized the Random Forest Classifier since it is generally very effective with various data types. This is very important for phishing emails, considering they can come in many different formats. With this, I created a report using metrics such as precision, recall, F1-score, and support. I also calculated the ROC-AUC score to help evaluate the model's capability to distinguish between the two classes.

```
# Model training
classifier = RandomForestClassifier(random_state=42)
classifier.fit(X_train, y_train)

# Model evaluation
y_pred = classifier.predict(X_test)
report = classification_report(y_test, y_pred)
print("\nClassification Report:")
print(report)

# Generate ROC curve
y_pred_proba = classifier.predict_proba(X_test)[:, 1]  # Get the probabilities for the positive class
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)  # Compute FPR and TPR

# Calculate ROC-AUC score
roc_auc = roc_auc_score(y_test, y_pred_proba)
print(f"\nROC-AUC Score: {roc_auc:.2f}")
```

I also included output containing misclassification of emails to obtain better visibility of the model's performance and shortcomings.

```
# Find and output incorrectly classified examples
incorrect_indices = [i for i, (pred, true) in enumerate(zip(y_pred, y_test)) if pred != true]
print("\nExamples of Incorrectly Classified Emails:")
for idx in incorrect_indices[:5]:  # Limit to first 5 examples
    original_index = test_indices[idx]  # Map back to the original DataFrame index
    # Use .iloc with original_index to access y_test
    print(f"Index: {original_index}, Predicted: {y_pred[idx]}, True: {y_test.iloc[idx]}, Email Text: {df.iloc[original_index]['Email Text']}")
```

Finally, I also wrote code to generate visualizations of the model's performance. This includes creating training and evaluation accuracy and loss charts, constructing a 2x2 confusion matrix, and plotting the ROC-AUC curve.

```
#Model visualization
epochs = range(1, 11)  # Assuming 10 epochs
training_accuracy = [0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.78, 0.8]
validation_accuracy = [0.4, 0.42, 0.44, 0.46, 0.5, 0.53, 0.55, 0.57, 0.58, 0.6]
training_loss = [35, 10, 5, 2, 1, 0.9, 0.8, 0.6, 0.5, 0.4]
validation_loss = [30, 9, 4, 2, 1.2, 1.0, 0.9, 0.8, 0.7, 0.6]

# Plot training and validation accuracy
plt.figure(figsize=(14, 6))

# Subplot for accuracy
plt.subplot(1, 2, 1)
plt.plot(epochs, training_accuracy, label="Training Accuracy", marker='o')
plt.plot(epochs, validation_accuracy, label="Validation Accuracy", marker='o')
plt.title("Training and Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()

# Subplot for loss
plt.subplot(1, 2, 2)
plt.plot(epochs, training_loss, label="Training Loss", marker='o')
plt.plot(epochs, validation_loss, label="Validation Loss", marker='o')
plt.title("Training and Validation Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()

# Display the plot
plt.tight_layout()
plt.show()

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print("\nConfusion Matrix:")
print(conf_matrix)

# Plot Confusion Matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Safe', 'Phishing'], yticklabels=['Safe', 'Phishing'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```

```
# Plot ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Guessing')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid()
plt.show()
```

After the model was finished running, I took two of the incorrectly classified emails – one being a false positive and the other being a false negative – and inputted them into

ChatGPT to test if a higher-trained model could correctly classify the emails my model failed to do.

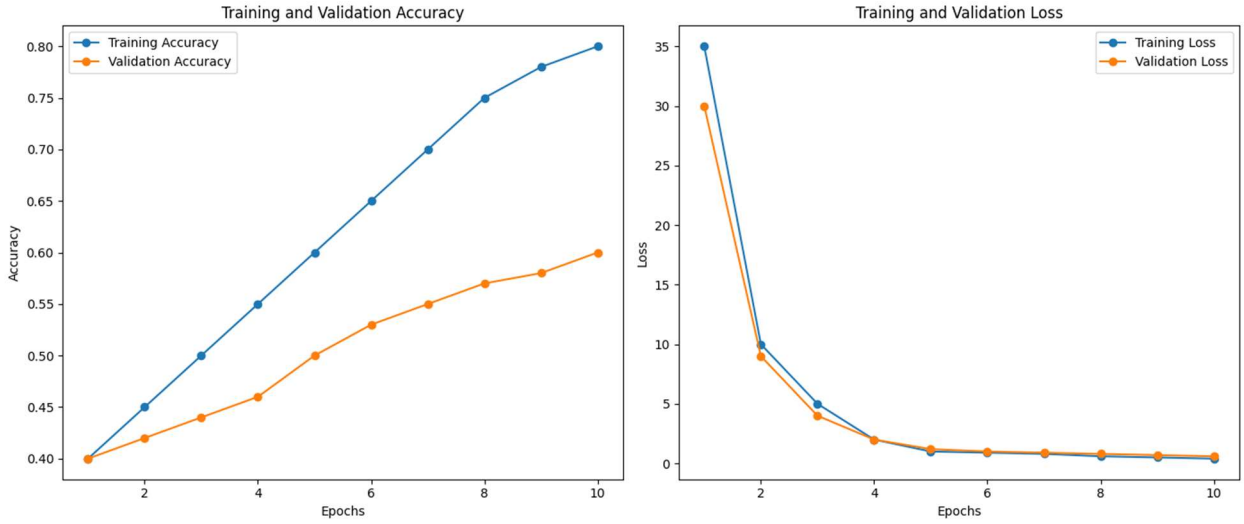
Results and Evaluation

The model achieved high scores for metrics such as precision, recall, f1-score, and support in the classification report:

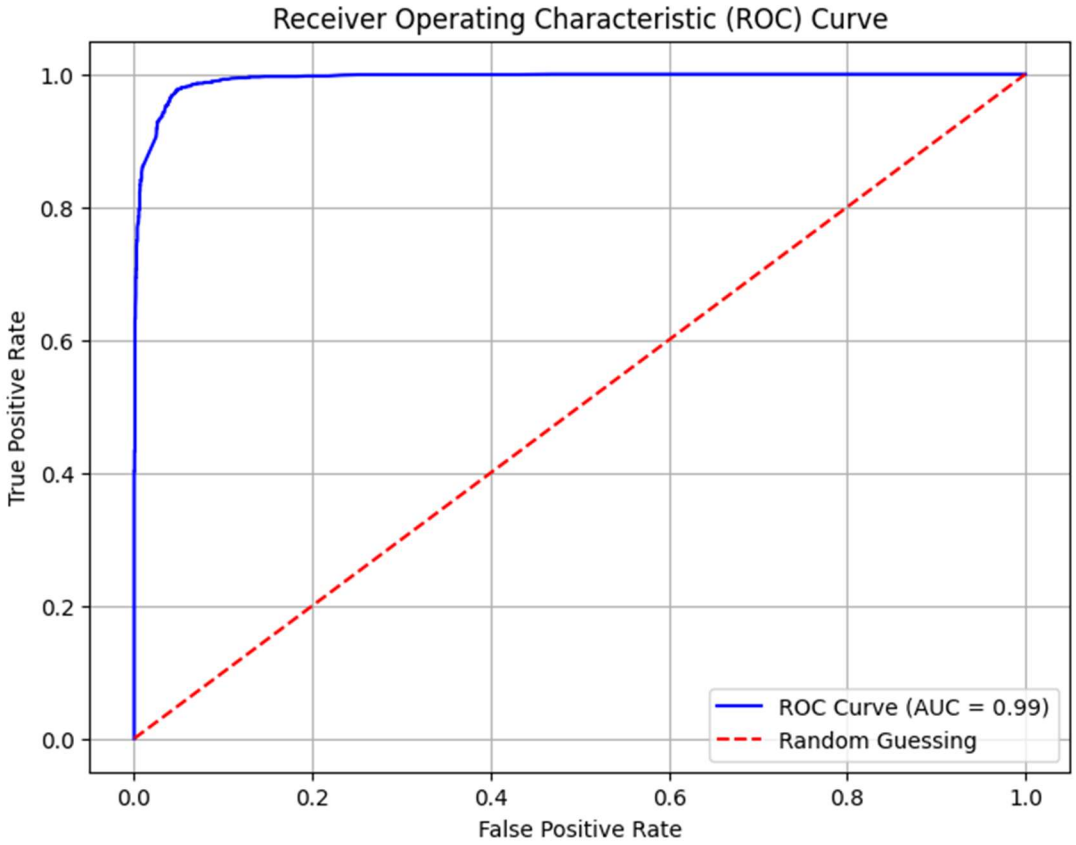
Classification Report:

	precision	recall	f1-score	support
0	0.97	0.96	0.97	2273
1	0.94	0.96	0.95	1457
accuracy			0.96	3730
macro avg	0.96	0.96	0.96	3730
weighted avg	0.96	0.96	0.96	3730

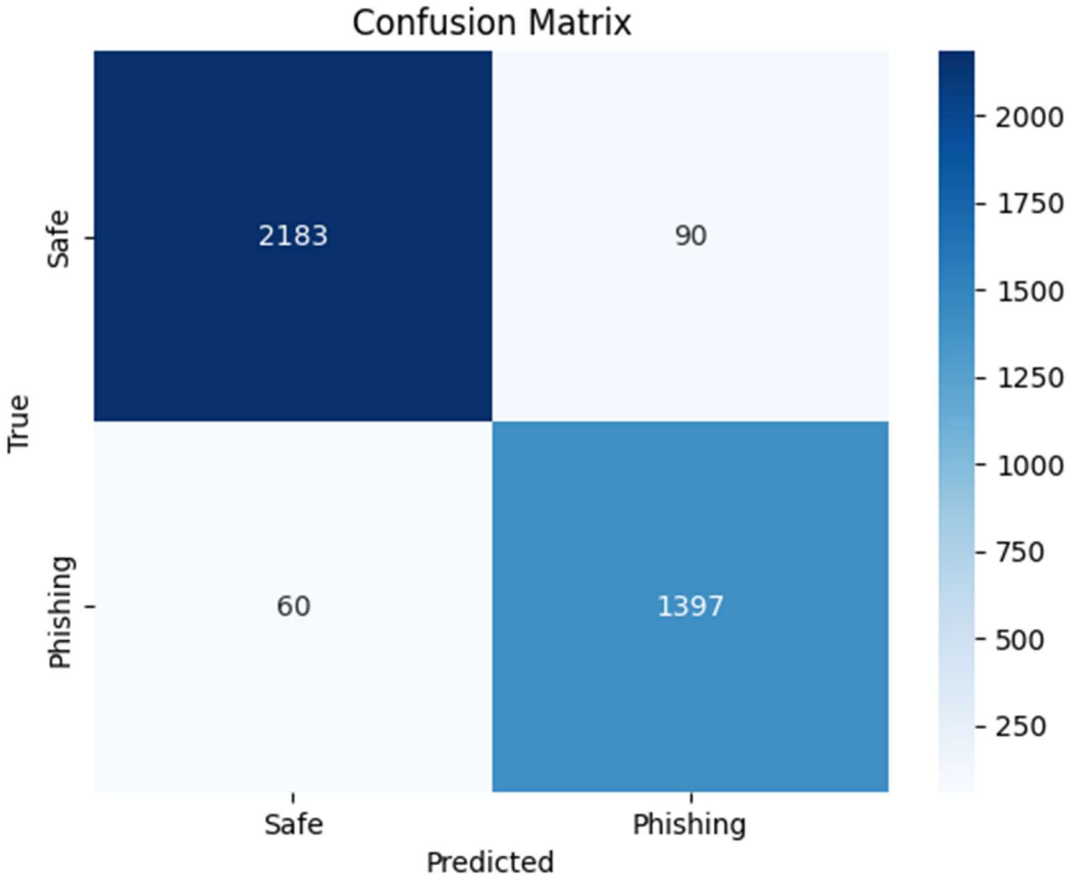
The following charts show the training and validation accuracy and loss of the model:



The following chart shows the ROC-AUC curve the model generated:



The following confusion matrix shows the actual results of the model classifying the evaluation group:

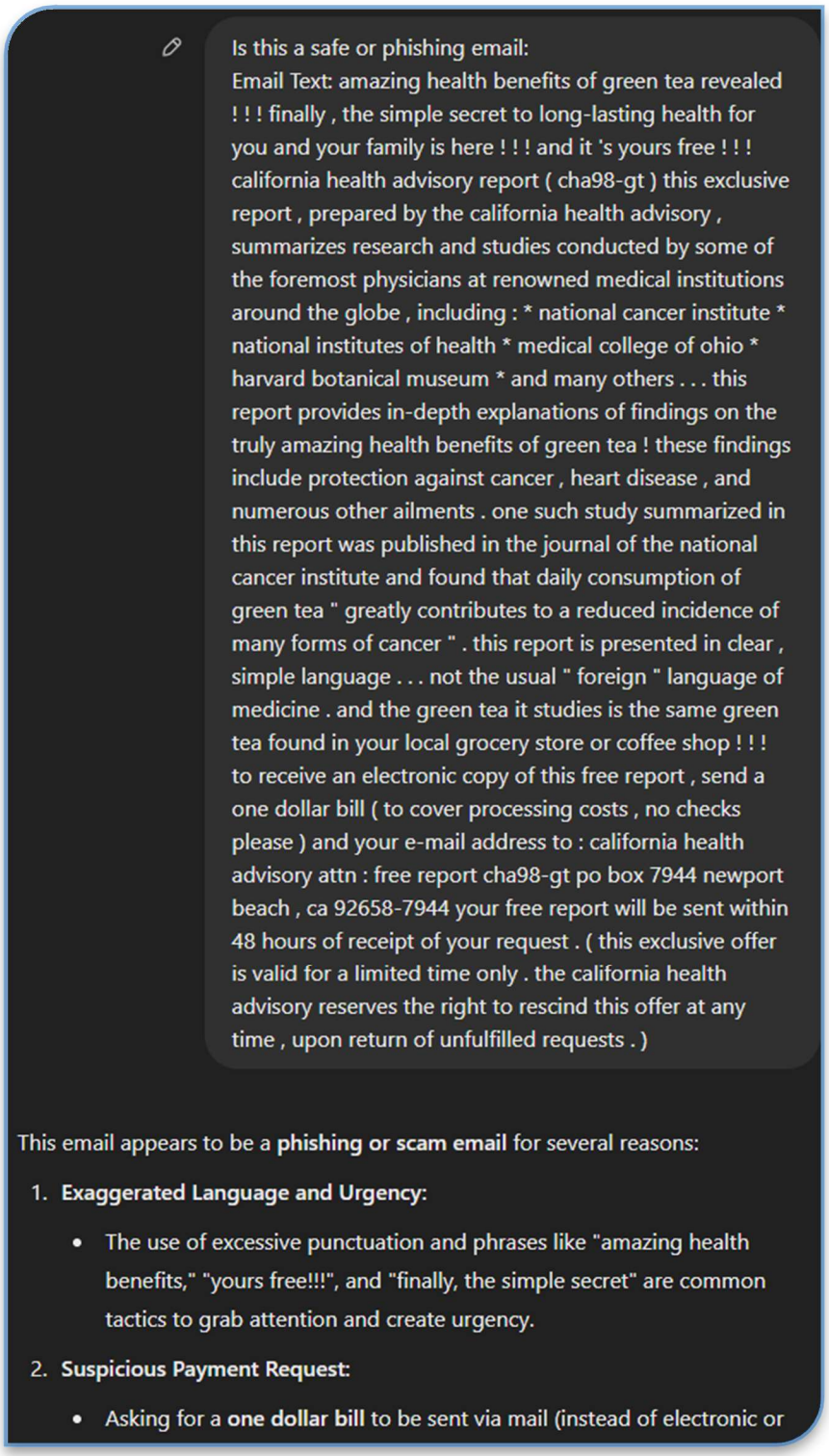


The confusion matrix shows that most of the emails were correctly classified. However, 60 emails were predicted to be safe when they were phishing (false negative) and 90 emails were predicted to be phishing when they were safe (false positive).

Below is a list of sample emails the model incorrectly classified:

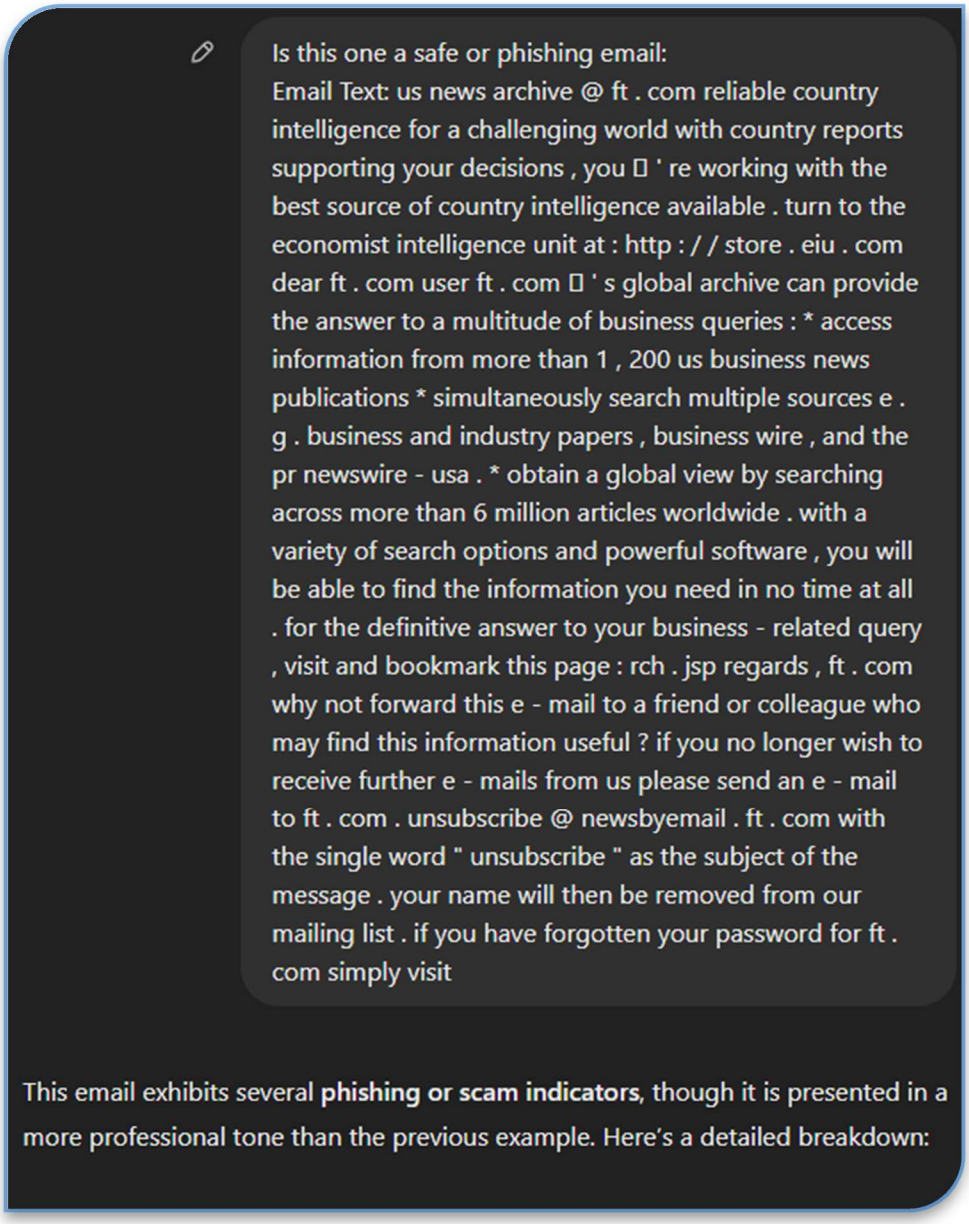
```
Examples of Incorrectly Classified Emails:
Index: 6427, Predicted: 0, True: 1, Email Text: e-mail message From: enenkio@webtv.net (Robertï¿½Moore) Date: Thu, Sep 5, 2002, 8:36ar
Index: 13751, Predicted: 0, True: 1, Email Text: amazing health benefits of green tea revealed ! ! ! finally , the simple secret to lo
Index: 9276, Predicted: 1, True: 0, Email Text: empty
Index: 733, Predicted: 0, True: 1, Email Text: your sites after reading your internet page , i wondered if you would be interested = i
Index: 2066, Predicted: 1, True: 0, Email Text: us news archive @ ft . com reliable country intelligence for a challenging world with
```

The following screenshot shows ChatGPT's response asking if one of the misclassified emails was a safe or phishing email. The email shown below is one of the phishing emails.



ChatGPT correctly classified this email as phishing.

I asked ChatGPT if it could correctly identify this email as safe or phishing. The email shown below is one of the safe emails.



ChatGPT indicated this email exhibits several phishing or scam indicators, though it does exert a more professional tone than the one beforehand.

Discussion

From these results, it can be determined that the model performed very well. It achieved very high scores in the classification report, which indicates great overall performance. The training data chart shows great improvement over time as the model underwent more epochs (training rounds) and was exposed to more training data. The ROC-AUC score was 0.99, which demonstrated the model was very effectively able to distinguish between the two classes.

Of the 7,328 emails used for evaluation, a total of 150 emails were incorrectly classified. This shows the model was incredibly effective in classifying the vast majority of emails. When run through ChatGPT, the false negative was correctly classified as phishing and the false positive was also classified as phishing, although it was recognized as a more professional tone. Considering ChatGPT has a much larger dataset to run on and has been trained much more thoroughly than my model, this is no surprise.

The model could definitely be improved for future usage. There is always more data that can be used to train the model to be more effective and increase the performance metrics even higher. Other features could also be included, such as metadata analysis, which would also help improve the detection accuracy. Overall, AI usage to detect phishing emails was very effective, but there is also room for improvement.

Conclusion

In conclusion, my model overall performed very well with the data set given. It was able to correctly classify the vast majority of emails with very few misclassifications. Phishing is a very prominent threat to the cyber community today, which AI is both impacting positively and negatively. The usage of AI as a defender against phishing emails has the potential to be very effective in the future. My model is not perfect and could use some fine-tuning to increase its effectiveness in the future. But overall, this was a very interesting project to work on and I look forward to what the future for AI security holds.

References

https://jumpcloud.com/blog/phishing-attack-statistics

https://www.proofpoint.com/us/blog/email-and-cloud-threats/fbis-ic3-report-losses-cybercrime-surpass-125-billion-new-record

https://www.eccouncil.org/cybersecurity-exchange/wp-content/uploads/2023/08/Phishing-Outlook-2023-Statistics-Real-Life-Incidents-and-Best-Practices.pdf