Spam Checker Project Documentation

The following key steps are done:

1. **Importing Libraries:** Essential Python libraries like `pandas`, `numpy`, and `sklearn` are imported for data handling and model building.

2. **Importing CSV File:** The dataset is loaded from a file named `mail_data.csv`, which likely contains emails labeled as spam or not.

3. **Data Pre-processing:** The data is checked for null values, duplicates are removed, and the email categories (spam or ham) are encoded into binary values (0 and 1).

4. **Feature Extraction:** The email messages are processed to extract features that will be used for classification.

5. **Training and Testing:** The dataset is split into training and testing sets.

6. **Selection of Classification Algorithm:** Various algorithms may have been tested to classify the emails as spam or ham.

7. **Model Evaluation:** The performance of the model is evaluated using accuracy scores.

8. **Prediction:** The model is used to predict whether new emails are spam or not.

---

1. Introduction

The ***Spam Checker*** project aims to classify emails as either spam or ham (not spam) using machine learning techniques. This project involves several stages, including data preprocessing, feature extraction, model selection, and evaluation.

2. Importing Libraries

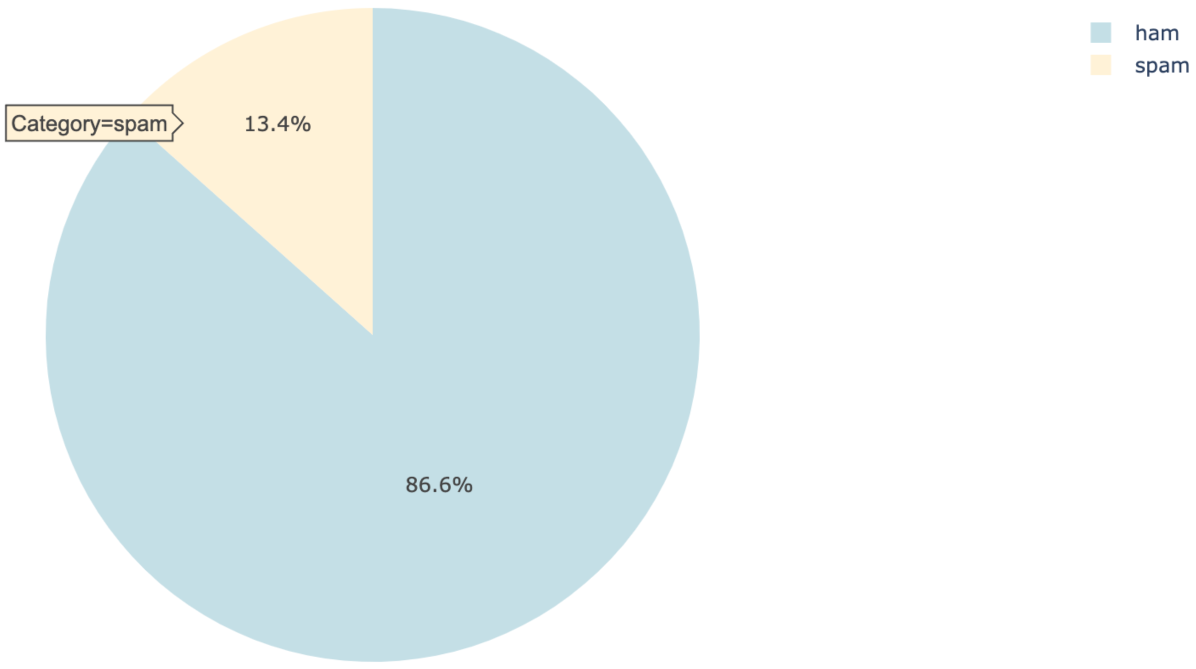
The following libraries were imported to handle data manipulation, visualization, and model training:

- **pandas:** For data handling and manipulation.
- **numpy:** For numerical operations.
- **sklearn:** For model selection, feature extraction, and evaluation.

- **plotly:** For data visualization.

3. Importing the Dataset

The dataset `mail_data.csv` is loaded using pandas. It contains email messages labeled as either 'spam' or 'ham'. The initial steps involved loading the dataset into a DataFrame and inspecting the first few rows to understand its structure.



4. Data Preprocessing

The data preprocessing involved several steps:

- **Checking for Missing Values:** The dataset was checked for any null values.
- **Removing Duplicates:** Duplicate entries were identified and removed to ensure data quality.
- **Encoding Labels:** The labels ('spam' and 'ham') were encoded into binary values, with 'spam' as 0 and 'ham' as 1.

5. Feature Extraction

The email messages were processed to extract meaningful features that could be used for classification. This typically involves Vectorization techniques like Term Frequency-Inverse Document Frequency (TF-IDF)

6. Training and Testing

The dataset was split into training and testing sets using `train_test_split` from `sklearn` with 80% of training data and 20% of testing data. This is crucial for evaluating the model's performance on unseen data.

7. Classification Model Selection and Training

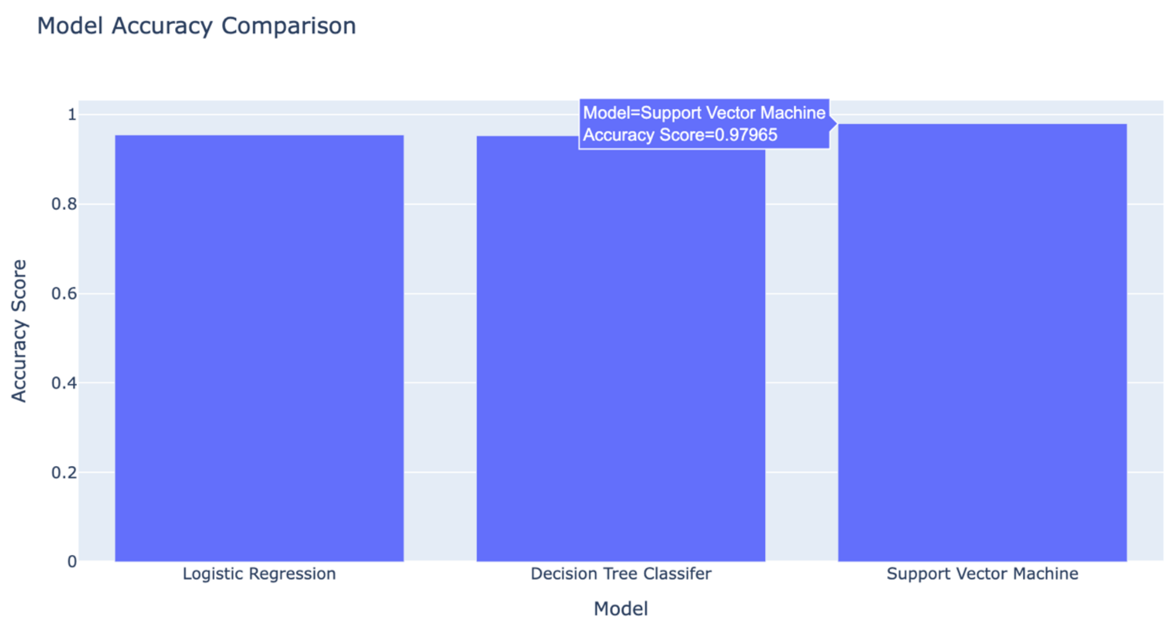
Various classification algorithms may have been evaluated, such as:

- Logistic Regression
- Support Vector Machines (SVM)
- Decision Tree Classifier

8. Model Evaluation

The ***(*Support Vector Machines*)*** classification model's was evaluated using the accuracy score, which measures the percentage of correctly classified emails.

- Logistic Regression - 0.95445 Accuracy Score
- Support Vector Machines (SVM) - 0.97965 Accuracy Score
- Decision Tree Classifier - 0.95251 Accuracy Score



9. Predictions

The trained model was used to predict whether new emails are spam or ham. The prediction process involves transforming the new data similarly to how the training data was processed.

Conclusion

The ***Spam Checker*** project successfully demonstrated the process of building a machine learning model to classify emails. Through careful data preprocessing, feature extraction, model training, and evaluation, the project provides a reliable solution for spam detection.

Phishing Link Checker Project Documentation

The following key steps are done:

1. Importing Libraries: Necessary libraries such as `pandas`, `plotly`, and `nltk` for data processing and visualization.

2. Importing CSV File: The dataset `phishing_site_urls.csv` is loaded, containing URLs labeled as either phishing or legitimate.

3. Data Pre-processing: The dataset is checked for missing values, and duplicates are removed.

4. Tokenization: URLs are tokenized into individual words.

5. Stemming: Tokenized words are stemmed to reduce them to their root forms.

6. One-Hot Encoding: Encodes categorical data into binary vectors.

7. Train-Test Split: The data is split into training and testing sets.

8. Feature Selection: Selection of relevant features for model training.

9. Model Selection and Training: Various classification algorithms are considered for detecting phishing URLs.

10. Model Evaluation: The model's performance is evaluated using accuracy scores.

11. Prediction: The model is used to predict whether new URLs are phishing or legitimate.

---

1. Introduction

The ***Phishing Checker*** project is designed to detect phishing websites by analyzing URLs. Using machine learning techniques, this project aims to classify URLs as either phishing or legitimate based on various features extracted from the URLs.

2. Importing Libraries

The following libraries were used in this project:

- **pandas:** For data manipulation and analysis.

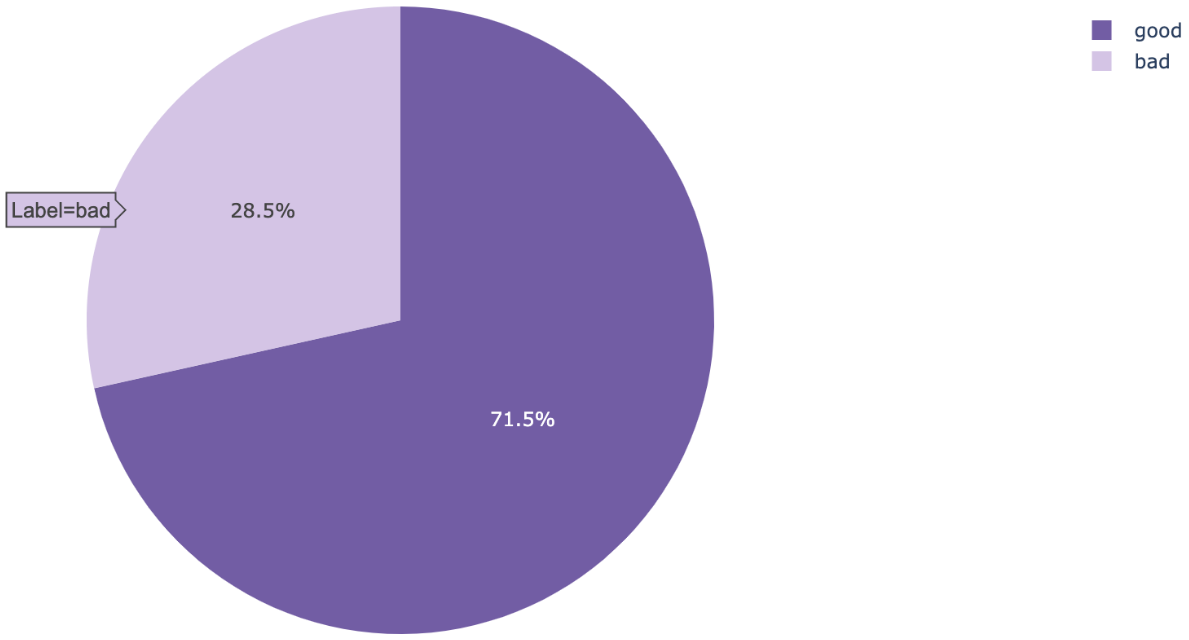
- **plotly:** For creating visualizations.
- **nltk:** For natural language processing tasks such as tokenization and stemming.

3. Importing the Dataset

The dataset `phishing_site_urls.csv` contains URLs labeled as 'phishing' or 'legitimate'. The data is loaded into a pandas DataFrame for analysis.

good – Not Phishing

bad - Phishing



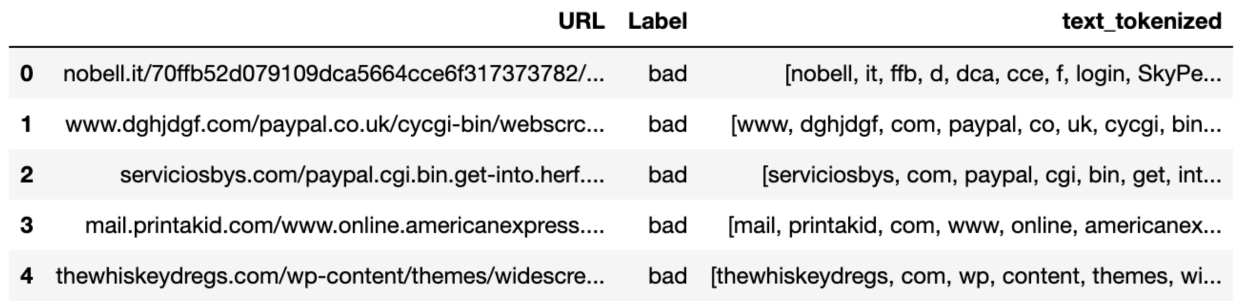
4. Data Pre-processing

The preprocessing steps included:

- **Checking for Missing Values:** The dataset was checked for any null values, ensuring the data is complete.
- **Removing Duplicates:** Duplicate URLs were identified and removed to maintain data integrity.

5. Tokenization

Tokenization involves breaking down the URLs into individual components (words). This is crucial for understanding the structure of the URL and identifying patterns associated with phishing.



	URL	Label	text_tokenized
0	nobell.it/70ffb52d079109dca5664cce6f317373782/...	bad	[nobell, it, ffb, d, dca, cce, f, login, SkyPe...
1	www.dghjdgf.com/paypal.co.uk/cycgi-bin/webscrc...	bad	[www, dghjdgf, com, paypal, co, uk, cycgi, bin...
2	serviciosbys.com/paypal.cgi.bin.get-into.herf....	bad	[serviciosbys, com, paypal, cgi, bin, get, int...
3	mail.printakid.com/www.online.americanexpress....	bad	[mail, printakid, com, www, online, americanex...
4	thewhiskeydregs.com/wp-content/themes/widescre...	bad	[thewhiskeydregs, com, wp, content, themes, wi...

6. Stemming

Stemming reduces words to their root forms, helping in normalizing the data and reducing dimensionality. This step is important in handling variations of the same word.

7. One-Hot Encoding

One-Hot Encoding converts categorical variables into a binary (0 and 1) matrix. This is useful for machine learning models that require numerical input.

8. Train-Test Split

The dataset was split into training and testing (80% training and 20% testing) sets to evaluate the model's performance on unseen data.

9. Feature Selection

Feature selection was performed to choose the most relevant features that contribute to the classification task. This step helps in improving model accuracy and reducing computational cost.

10. Model Selection and Training

Several classification algorithms were tested to find the best model for phishing detection. Common algorithms that might be used include:

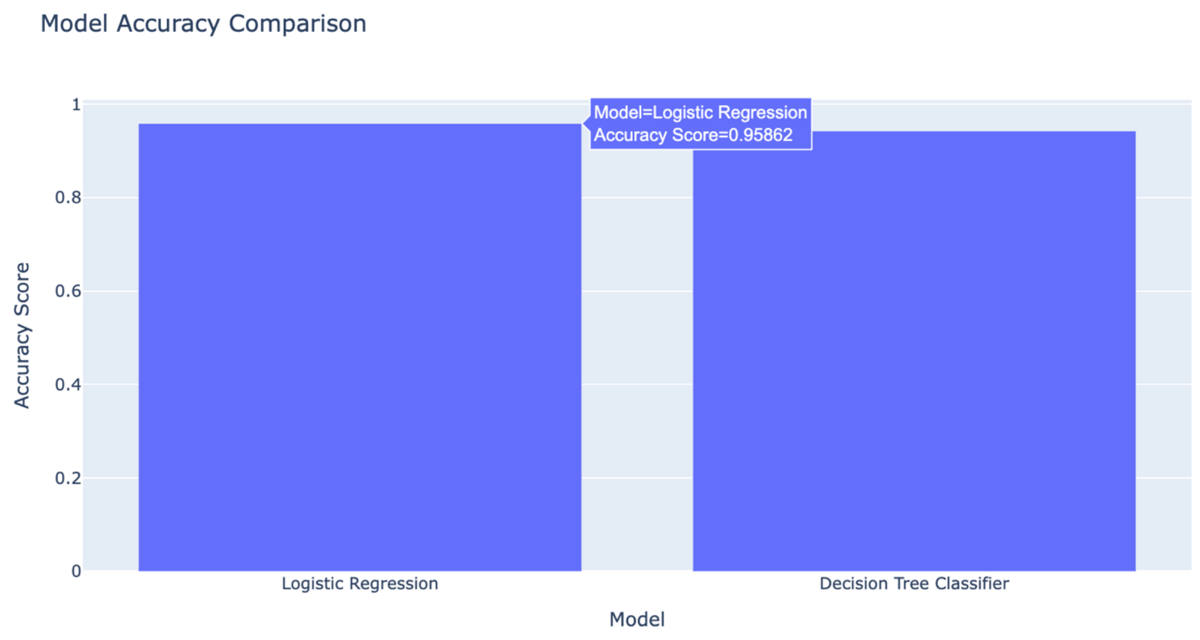
- Logistic Regression

- Decision Tree Classifier

11. Model Evaluation

The model's performance was evaluated using the accuracy score, which measures the proportion of correctly classified URLs.

- Logistic Regression - 0.95862 Accuracy Score
- Decision Tree Classifier - 0.94281 Accuracy Score



12. Predictions

The trained model can now be used to predict whether new URLs are phishing or legitimate. This involves processing new data similarly and applying the trained model to make predictions.

Conclusion

The Phishing Checker project effectively demonstrates the use of machine learning techniques to detect phishing websites. By carefully preprocessing data, selecting relevant features, and training a robust model, the project offers a reliable solution for identifying potentially harmful URLs.

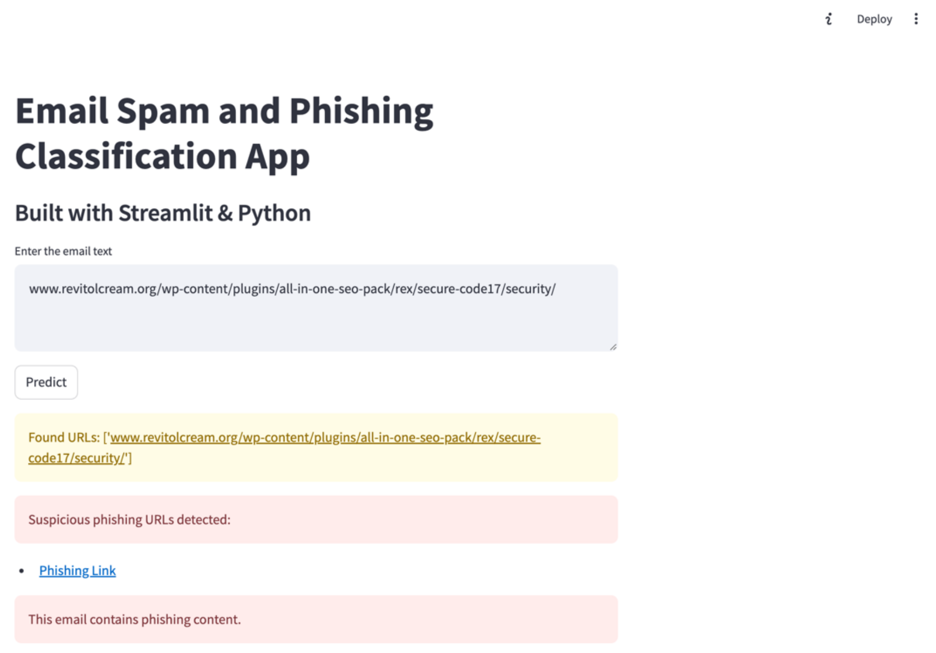
Final Output and Implementation

In the final stage of both the **Spam Checker** and **Phishing Checker** projects, the trained models and their corresponding feature extraction methods were saved as pickle files for easy reuse and deployment.

For the Spam Checker, the TF-IDF vectorizer and the **Support Vector Machine** classifier were saved as `feature_extraction_spam.pkl` and `svm_spam.pkl`, respectively. Similarly, for the **Phishing Checker**, the TF-IDF vectorizer and the logistic regression classifier were saved as `tfidf_vectorize_phishing.pkl` and `lr_classifier_phishing.pkl`.

To implement these models in a practical application, they were loaded from their respective pickle files and integrated into a **Streamlit** application. This application allows users to input an email message or a URL, which is then processed by the appropriate model. The TF-IDF vectorizer converts the input into the necessary format, and the classifier determines whether the email is spam or not, or whether the URL is phishing or legitimate.

This setup enables a seamless and interactive user experience, where the complex machine learning models operate behind the scenes to provide real-time predictions based on user input.



Email Spam and Phishing Classification App

Built with Streamlit & Python

Enter the email text

www.revitolcream.org/wp-content/plugins/all-in-one-seo-pack/rex/secure-code17/security/

Found URLs: ['www.revitolcream.org/wp-content/plugins/all-in-one-seo-pack/rex/secure-code17/security/']

Suspicious phishing URLs detected:

- Phishing Link

This email contains phishing content.