In [3]:
```
import numpy as np
import pandas as pd
```

AI/ML for Cybersecurity project (Phase 1: Data Preprocessing and Feature Engineering)

Group 6

This project aims at using the Phishing Corpus dataset from UCI to build classification models that can detect phishing websites. In phase 1 of the project, the dataset acquired from the UCI ML repository will be evaluated and preprocessed with the following objectives:

- understanding the dataset
- Cleaning the dataset by handling missing or erroneous values
- Engineer features from URL and webcontent
- Creating additional features based on heuristics to improve detection

In [4]:
```
#importing dataset
data = pd.read_csv(r"C:\Users\PC\Downloads\phiusiil+phishing+url+dataset\Phi
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
Input In [4], in <cell line: 2>()
      1 #importing dataset
----> 2 data = pd.read_csv(r"C:\Users\PC\Downloads\phiusiil+phishing+url+dataset\PhiUSIIL_Phishing_URL_Dataset.csv")

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/util/_decorators.py:311, in deprecate_nonkeyword_arguments.<locals>.decorate.<locals>.wrapper(*args, **kwargs)
    305 if len(args) > num_allow_args:
    306     warnings.warn(
    307         msg.format(arguments=arguments),
    308         FutureWarning,
    309         stacklevel=stacklevel,
    310     )
--> 311 return func(*args, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/io/parsers/readers.py:680, in read_csv(filepath_or_buffer, sep, delimiter, header, names, index_col, usecols, squeeze, prefix, mangle_dupe_cols, dtype, engine, converters, true_values, false_values, skipinitialspace, skiprows, skipfooter, nrows, na_values, keep_default_na, na_filter, verbose, skip_blank_lines, parse_dates, infer_datetime_format, keep_date_col, date_parser, dayfirst, cache_dates, iterator, chunksize, compression, thousands, decimal, lineterminator, quotechar, quoting, doublequote, escapechar, comment, encoding, encoding_errors, dialect, error_bad_lines, warn_bad_lines, on_bad_lines, delim_whitespace, low_memory, memory_map, float_precision, storage_options)
    665 kwds_defaults = _refine_defaults_read(
    666     dialect,
    667     delimiter,
   (...)
    676     defaults={"delimiter": ","},
    677 )
    678 kwds.update(kwds_defaults)
--> 680 return _read(filepath_or_buffer, kwds)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/io/parsers/readers.py:575, in _read(filepath_or_buffer, kwds)
    572 _validate_names(kwds.get("names", None))
    574 # Create the parser.
--> 575 parser = TextFileReader(filepath_or_buffer, **kwds)
    577 if chunksize or iterator:
    578     return parser

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/io/parsers/readers.py:934, in TextFileReader.__init__(self, f, engine, **kwds)
    931     self.options["has_index_names"] = kwds["has_index_names"]
    933 self.handles: IOHandles | None = None
--> 934 self._engine = self._make_engine(f, self.engine)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/io/parsers/readers.py:1218, in TextFileReader._make_engine(self, f, engine)
   1214     mode = "rb"
   1215 # error: No overload variant of "get_handle" matches argument types
   1216 # "Union[str, PathLike[str], ReadCsvBuffer[bytes], ReadCsvBuffer[str]]"
   1217 # , "str", "bool", "Any", "Any", "Any", "Any", "Any"
-> 1218 self.handles = get_handle(  # type: ignore[call-overload]
   1219     f,
```

```
   1220     mode,
   1221     encoding=self.options.get("encoding", None),
   1222     compression=self.options.get("compression", None),
   1223     memory_map=self.options.get("memory_map", False),
   1224     is_text=is_text,
   1225     errors=self.options.get("encoding_errors", "strict"),
   1226     storage_options=self.options.get("storage_options", None),
   1227 )
   1228 assert self.handles is not None
   1229 f = self.handles.handle

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/pandas/io/common.py:786, in get_handle(path_or_buf, mode, encoding, compression, memory_map, is_text, errors, storage_options)
    781 elif isinstance(handle, str):
    782     # Check whether the filename is to be opened in binary mode.
    783     # Binary mode does not support 'encoding' and 'newline'.
    784     if ioargs.encoding and "b" not in ioargs.mode:
    785         # Encoding
--> 786         handle = open(
    787             handle,
    788             ioargs.mode,
    789             encoding=ioargs.encoding,
    790             errors=errors,
    791             newline="",
    792         )
    793     else:
    794         # Binary mode
    795         handle = open(handle, ioargs.mode)

FileNotFoundError: [Errno 2] No such file or directory: 'C:\\Users\\PC\\Downloads\\phiusiil+phishing+url+dataset\\PhiUSIIL_Phishing_URL_Dataset.csv'
```

In [4]:
```
#first look at the data
data.head()
```

Out[4]:

	FILENAME	URL	URLLength	Domain	Dom
0	521848.txt	https://www.southbankmosaics.com	31	www.southbankmosaics.com	
1	31372.txt	https://www.uni-mainz.de	23	www.uni-mainz.de	
2	597387.txt	https://www.voicefmradio.co.uk	29	www.voicefmradio.co.uk	
3	554095.txt	https://www.sfnmjournal.com	26	www.sfnmjournal.com	
4	151578.txt	https://www.rewildingargentina.org	33	www.rewildingargentina.org	

5 rows × 56 columns

In [5]:
```
data.columns
```

Out[5]:
```
Index(['FILENAME', 'URL', 'URLLength', 'Domain', 'DomainLength', 'IsDomainIP',
       'TLD', 'URLSimilarityIndex', 'CharContinuationRate',
       'TLDLegitimateProb', 'URLCharProb', 'TLDLength', 'NoOfSubDomain',
       'HasObfuscation', 'NoOfObfuscatedChar', 'ObfuscationRatio',
       'NoOfLettersInURL', 'LetterRatioInURL', 'NoOfDegitsInURL',
       'DegitRatioInURL', 'NoOfEqualsInURL', 'NoOfQMarkInURL',
       'NoOfAmpersandInURL', 'NoOfOtherSpecialCharsInURL',
       'SpacialCharRatioInURL', 'IsHTTPS', 'LineOfCode', 'LargestLineLength',
       'HasTitle', 'Title', 'DomainTitleMatchScore', 'URLTitleMatchScore',
       'HasFavicon', 'Robots', 'IsResponsive', 'NoOfURLRedirect',
       'NoOfSelfRedirect', 'HasDescription', 'NoOfPopup', 'NoOfiFrame',
       'HasExternalFormSubmit', 'HasSocialNet', 'HasSubmitButton',
       'HasHiddenFields', 'HasPasswordField', 'Bank', 'Pay', 'Crypto',
       'HasCopyrightInfo', 'NoOfImage', 'NoOfCSS', 'NoOfJS', 'NoOfSelfRef',
       'NoOfEmptyRef', 'NoOfExternalRef', 'label'],
      dtype='object')
```

In [6]:
```
#evaluating a few instances
pd.set_option('display.max_rows', None)
data[0:10].T
```

Out[6]:

	0	1	
FILENAME	521848.txt	31372.txt	
URL	https://www.southbankmosaics.com	https://www.uni-mainz.de	https://www.v
URLLength	31	23	
Domain	www.southbankmosaics.com	www.uni-mainz.de	www.v
DomainLength	24	16	
IsDomainIP	0	0	
TLD	com	de	
URLSimilarityIndex	100.0	100.0	
CharContinuationRate	1.0	0.666667	
TLDLegitimateProb	0.522907	0.03265	
URLCharProb	0.061933	0.050207	
TLDLength	3	2	
NoOfSubDomain	1	1	
HasObfuscation	0	0	
NoOfObfuscatedChar	0	0	
ObfuscationRatio	0.0	0.0	
NoOfLettersInURL	18	9	
LetterRatioInURL	0.581	0.391	
NoOfDegitsInURL	0	0	
DegitRatioInURL	0.0	0.0	
NoOfEqualsInURL	0	0	
NoOfQMarkInURL	0	0	
NoOfAmpersandInURL	0	0	
NoOfOtherSpecialCharsInURL	1	2	
SpacialCharRatioInURL	0.032	0.087	
IsHTTPS	1	1	
LineOfCode	558	618	
LargestLineLength	9381	9381	
HasTitle	1	1	
Title	à¸‚à¹^à¸²à¸§à¸ª à¸‚à¹^à¸²à¸§à¸§à¸±à¸™à¸µ...	johannes gutenberg-universitÃ¤t mainz	voice
DomainTitleMatchScore	0.0	55.555556	
URLTitleMatchScore	0.0	55.555556	
HasFavicon	0	1	
Robots	1	1	
IsResponsive	1	0	

	0	1
NoOfURLRedirect	0	0
NoOfSelfRedirect	0	0
HasDescription	0	0
NoOfPopup	0	0
NoOfiFrame	1	0
HasExternalFormSubmit	0	0
HasSocialNet	0	1
HasSubmitButton	1	1
HasHiddenFields	1	0
HasPasswordField	0	0
Bank	1	0
Pay	0	0
Crypto	0	0
HasCopyrightInfo	1	1
NoOfImage	34	50
NoOfCSS	20	9
NoOfJS	28	8
NoOfSelfRef	119	39
NoOfEmptyRef	0	0
NoOfExternalRef	124	217
label	1	1

In [7]:
```
data.describe(include='all').T
```

Out[7]:

	count	unique	top	freq
FILENAME	235795	235795	521848.txt	1
URL	235795	235370	https://disclosepack.myportfolio.com/	2
URLLength	235795.0	NaN	NaN	NaN
Domain	235795	220086	ipfs.io	1197
DomainLength	235795.0	NaN	NaN	NaN
IsDomainIP	235795.0	NaN	NaN	NaN
TLD	235795	695	com	112554
URLSimilarityIndex	235795.0	NaN	NaN	NaN
CharContinuationRate	235795.0	NaN	NaN	NaN
TLDLegitimateProb	235795.0	NaN	NaN	NaN
URLCharProb	235795.0	NaN	NaN	NaN
TLDLength	235795.0	NaN	NaN	NaN
NoOfSubDomain	235795.0	NaN	NaN	NaN
HasObfuscation	235795.0	NaN	NaN	NaN
NoOfObfuscatedChar	235795.0	NaN	NaN	NaN
ObfuscationRatio	235795.0	NaN	NaN	NaN
NoOfLettersInURL	235795.0	NaN	NaN	NaN
LetterRatioInURL	235795.0	NaN	NaN	NaN
NoOfDegitsInURL	235795.0	NaN	NaN	NaN
DegitRatioInURL	235795.0	NaN	NaN	NaN
NoOfEqualsInURL	235795.0	NaN	NaN	NaN
NoOfQMarkInURL	235795.0	NaN	NaN	NaN
NoOfAmpersandInURL	235795.0	NaN	NaN	NaN
NoOfOtherSpecialCharsInURL	235795.0	NaN	NaN	NaN
SpacialCharRatioInURL	235795.0	NaN	NaN	NaN
IsHTTPS	235795.0	NaN	NaN	NaN
LineOfCode	235795.0	NaN	NaN	NaN
LargestLineLength	235795.0	NaN	NaN	NaN
HasTitle	235795.0	NaN	NaN	NaN
Title	235795	197874	0	32719
DomainTitleMatchScore	235795.0	NaN	NaN	NaN
URLTitleMatchScore	235795.0	NaN	NaN	NaN
HasFavicon	235795.0	NaN	NaN	NaN
Robots	235795.0	NaN	NaN	NaN
IsResponsive	235795.0	NaN	NaN	NaN
NoOfURLRedirect	235795.0	NaN	NaN	NaN
NoOfSelfRedirect	235795.0	NaN	NaN	NaN

	count	unique	top	freq
HasDescription	235795.0	NaN	NaN	NaN
NoOfPopup	235795.0	NaN	NaN	NaN
NoOfiFrame	235795.0	NaN	NaN	NaN
HasExternalFormSubmit	235795.0	NaN	NaN	NaN
HasSocialNet	235795.0	NaN	NaN	NaN
HasSubmitButton	235795.0	NaN	NaN	NaN
HasHiddenFields	235795.0	NaN	NaN	NaN
HasPasswordField	235795.0	NaN	NaN	NaN
Bank	235795.0	NaN	NaN	NaN
Pay	235795.0	NaN	NaN	NaN
Crypto	235795.0	NaN	NaN	NaN
HasCopyrightInfo	235795.0	NaN	NaN	NaN
NoOfImage	235795.0	NaN	NaN	NaN
NoOfCSS	235795.0	NaN	NaN	NaN
NoOfJS	235795.0	NaN	NaN	NaN
NoOfSelfRef	235795.0	NaN	NaN	NaN
NoOfEmptyRef	235795.0	NaN	NaN	NaN
NoOfExternalRef	235795.0	NaN	NaN	NaN
label	235795.0	NaN	NaN	NaN

In [8]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 235795 entries, 0 to 235794
Data columns (total 56 columns):
 #   Column                       Non-Null Count   Dtype  
---  ------                       --------------   -----  
 0   FILENAME                     235795 non-null  object 
 1   URL                          235795 non-null  object 
 2   URLLength                    235795 non-null  int64  
 3   Domain                       235795 non-null  object 
 4   DomainLength                 235795 non-null  int64  
 5   IsDomainIP                   235795 non-null  int64  
 6   TLD                          235795 non-null  object 
 7   URLSimilarityIndex           235795 non-null  float64
 8   CharContinuationRate         235795 non-null  float64
 9   TLDLegitimateProb            235795 non-null  float64
 10  URLCharProb                  235795 non-null  float64
 11  TLDLength                    235795 non-null  int64  
 12  NoOfSubDomain                235795 non-null  int64  
 13  HasObfuscation               235795 non-null  int64  
 14  NoOfObfuscatedChar           235795 non-null  int64  
 15  ObfuscationRatio             235795 non-null  float64
 16  NoOfLettersInURL             235795 non-null  int64  
 17  LetterRatioInURL             235795 non-null  float64
 18  NoOfDegitsInURL              235795 non-null  int64  
 19  DegitRatioInURL              235795 non-null  float64
 20  NoOfEqualsInURL              235795 non-null  int64  
 21  NoOfQMarkInURL               235795 non-null  int64  
 22  NoOfAmpersandInURL           235795 non-null  int64  
 23  NoOfOtherSpecialCharsInURL   235795 non-null  int64  
 24  SpacialCharRatioInURL        235795 non-null  float64
 25  IsHTTPS                      235795 non-null  int64  
 26  LineOfCode                   235795 non-null  int64  
 27  LargestLineLength            235795 non-null  int64  
 28  HasTitle                     235795 non-null  int64  
 29  Title                        235795 non-null  object 
 30  DomainTitleMatchScore        235795 non-null  float64
 31  URLTitleMatchScore           235795 non-null  float64
 32  HasFavicon                   235795 non-null  int64  
 33  Robots                       235795 non-null  int64  
 34  IsResponsive                 235795 non-null  int64  
 35  NoOfURLRedirect              235795 non-null  int64  
 36  NoOfSelfRedirect             235795 non-null  int64  
 37  HasDescription               235795 non-null  int64  
 38  NoOfPopup                    235795 non-null  int64  
 39  NoOfiFrame                   235795 non-null  int64  
 40  HasExternalFormSubmit        235795 non-null  int64  
 41  HasSocialNet                 235795 non-null  int64  
 42  HasSubmitButton              235795 non-null  int64  
 43  HasHiddenFields              235795 non-null  int64  
 44  HasPasswordField             235795 non-null  int64  
 45  Bank                         235795 non-null  int64  
 46  Pay                          235795 non-null  int64  
 47  Crypto                       235795 non-null  int64  
 48  HasCopyrightInfo             235795 non-null  int64  
 49  NoOfImage                    235795 non-null  int64  
 50  NoOfCSS                      235795 non-null  int64  
 51  NoOfJS                       235795 non-null  int64  
 52  NoOfSelfRef                  235795 non-null  int64  
 53  NoOfEmptyRef                 235795 non-null  int64  
 54  NoOfExternalRef              235795 non-null  int64  
 55  label                        235795 non-null  int64  
dtypes: float64(10), int64(41), object(5)
memory usage: 100.7+ MB
```

In [9]:
```
data.shape
```

Out[9]:
```
(235795, 56)
```

After exploring the dataset, the following key insights have been acquired:

- the dataset has 235795 rows and 56 columns
- Only 5 features are Non-numeric with the remaining being binary or continuous values
- The mean of the labels is about 0.57 (indicating 57% for class 1 and 43% for class 0), showing reasonable class balance.
- The dataset does not seem to have any missing values, however there does seem to be some missing titles that are not properly labeled as having titles in the 'HasTitle' feature.
- Some features need to be adjusted for outliers becuase of very high maximum values compared to their average, such as 'LargestLineLength' and 'LineofCode'.
- A healthy humber of features exist describing the URL of a phishing website, such as URL length, number of special characters in URL, number of equal signs, and many more. However, not many features are present describing the Domain and Title of the website.

In [10]:
```
#handling erroneous values at 'HasTitle' column
data.loc[data['Title'] == '0', 'HasTitle'] = 0
data.loc[data['Title'] != '0', 'HasTitle'] = 1
```

Feature engineering and creating additional features

Feature engineering is the process of transforming raw data into meaningful features that enhance the performance of machine learning models. Furthermore, more features could be created using heuristics and additional calculations.

In [11]:
```
#New features for domain beggining, special characters ratio, and numbers
special_chars = ['-', '_', '.', '@']
data['Domain_starts_with_www.'] = data['Domain'].str.startswith('www.').asty
data['special_char_in_domain_ratio'] = data['Domain'].apply(lambda x: sum(1
data['Domain_contains_number'] = data['Domain'].str.contains(r'\d').astype(
```

In [12]:
```
#New features for titles containing number and special characters
data['Title_contains_number'] = data['Title'].str.contains(r'\d').astype(int
data['Title_contains_specialcharacter'] = data['Title'].str.contains(r'[^a-z
```

In [13]:
```
import re
data['Title_nonalphanumeric_count'] = data['Title'].apply(lambda x: len(re.
```

In [14]:
```
#New feature for uppercase to lowercase ratio
data['url_uppercase_ratio'] = data['URL'].apply(lambda x: sum(1 for c in x :
data['title_uppercase_ratio'] = data['Title'].apply(lambda x: sum(1 for c in
```

In [15]:
```
#New feature for external to internal references ratio
data['external_to_internal_ref_ratio'] = data['NoOfExternalRef'] / (data['No
```

In [16]:
```
#New feature for image to text ratio
data['image_to_text_ratio'] = data['NoOfImage'] / (data['LineOfCode'] + 1)
```

In [17]:
```
#New features to check for shortened URLs
shortening_services = ['bit.ly', 'tinyurl', 'goo.gl', 't.co', 'ow.ly', 'is.g
data['is_shortened_url'] = data['Domain'].apply(lambda x: 1 if any(service :
```

In [18]:
```
#New features for HTML tag to code ratio
data['tag_to_code_ratio'] = (data['NoOfImage'] + data['NoOfCSS'] + data['NoO
```

Newly Added Features and Justifications

New features have been engineered to help in detecting phishing websites. These features were chosen based on the characteristics of phishing websites commonly identified in academic research.

- **Domain_starts_with_www.**: This feature identifies if the domain starts with "www." Legitimate websites typically adhere to conventional URL structures, and deviations from this format may indicate phishing behavior. Phishing websites might skip this prefix to create ambiguity or seem unconventional [1].
- **special_char_in_domain_ratio**: The ratio of special characters (such as `-`, `_`, `.`, `@`) present in the domain name is calculated. These characters are often used by phishing websites to mimic legitimate domains or to obscure the actual intent of the website. Phishing sites that heavily rely on special characters can be flagged as suspicious [2].
- **Domain_contains_number**: This feature checks if numeric characters are present in the domain. Phishing websites often include numbers in their domain names to impersonate legitimate websites or bypass filters. Numbers in domain names are a common feature in phishing attempts [3].
- **Title_contains_number**: Phishing sites may include numbers in their page titles, especially when attempting to mimic institutions like banks or ecommerce sites. Since legitimate titles rarely contain numbers, this feature can help distinguish phishing sites from legitimate ones [4].
- **Title_contains_specialcharacter**: This feature flags special characters in the page title. Phishing websites often use special characters to draw user attention or obfuscate their true intentions. Legitimate websites seldom include such characters in their titles, making their presence suspicious [5].
- **Title_nonalphanumeric_count**: The total number of non-alphanumeric characters in the title is calculated. Non-alphanumeric characters, including symbols and punctuation, are commonly used in phishing attacks as part of their obfuscation strategies. This feature helps to detect such patterns [6].
- **is_shortened_url**: This feature identifies whether the URL has been shortened using popular URL shortening services (e.g., bit.ly, tinyurl). Phishing websites frequently use shortened URLs to hide the actual destination, making it difficult for

users to detect malicious intent. Shortened URLs are a known indicator of phishing behavior [7].

- **tag_to_code_ratio**: This feature calculates the ratio of HTML tags to the total amount of code in the page source. Phishing websites often have a higher number of tags compared to legitimate websites, as they use HTML tricks to manipulate the appearance of the page and deceive users. A higher tag-to-code ratio can therefore signal potential phishing activity [8].

- **url_uppercase_ratio** and **title_uppercase_ratio**: The ratios of uppercase letters in the URL and title are calculated. Phishing websites may use uppercase letters to emphasize certain parts of the URL or title, making them appear important or legitimate. A high uppercase ratio is often a sign of malicious intent [5][6].

- **external_to_internal_ref_ratio**: This feature measures the ratio of external to internal references. Phishing websites frequently link to external malicious sites or use excessive redirects. A high external reference ratio can be a strong signal of phishing [9].

- **image_to_text_ratio**: This feature computes the ratio of images to text content. Phishing websites may use more images than text to mask their malicious intent or create the illusion of legitimacy. A high image-to-text ratio often correlates with phishing attempts [10].

References

1. Abbasi, H., Abbasi, M., & Javidan, R. (2019). A Comparative Study on Phishing Detection Techniques. IEEE Access, 7, 52946–52963. https://doi.org/10.1109/ACCESS.2019.2910954
2. Zhang, P., Wang, W., Zhang, X., & Li, Y. (2018). A Machine Learning-Based Phishing Detection Framework Using URL Features. IEEE Transactions on Network and Service Management, 15(4), 1332–1345. https://doi.org/10.1109/TNSM.2018.2877796
3. Marchal, S., Francois, J., State, R., & Engel, T. (2017). PhishStorm: Detecting Phishing with Streaming Analytics. IEEE Transactions on Network and Service Management, 14(3), 626-640. https://doi.org/10.1109/TNSM.2017.2718619
4. Basnet, M., Sung, A. H., & Liu, Q. (2011). Rule-based Phishing Attack Detection. Proceedings of the 8th International Conference on Information Technology: New Generations, 249-254. https://doi.org/10.1109/ITNG.2011.48
5. Jain, A. K., & Gupta, B. B. (2014). A Machine Learning Based Approach for Phishing Detection Using URLs. Proceedings of the 7th International Conference on Contemporary Computing, 265-270. https://doi.org/10.1109/IC3.2014.6897205
6. Dash, T. K., Sahoo, A. K., & Giri, M. S. (2018). Phishing Website Detection Based on URL Features. Proceedings of the 2018 International Conference on Smart Computing and Communication, 23-29. https://doi.org/10.1109/SMARTCOM.2018.8707851
7. Song, J., Lee, J., & Kim, S. (2015). Suspicious URL Detection Based on Word Embedding and N-Gram Features. Proceedings of the 2015 IEEE International

Conference on Computer and Information Technology, 106–113. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.33

8. Mohammad, M. A., Thabtah, F., & McCluskey, L. (2014). Predicting Phishing Websites Based on Self-Structuring Neural Networks. Neural Computing and Applications, 25(2), 443-458. https://doi.org/10.1007/s00521-013-1510-7
9. Canova, G., Volkamer, M., Bergmann, C., & Borza, R. (2014). NoPhish: An Anti-Phishing Education App. Proceedings of the 9th International Conference on Availability, Reliability, and Security (ARES), 261-269. https://doi.org/10.1109/ARES.2014.38
10. Mohammad, M. A., Thabtah, F., & McCluskey, L. (2014). Predicting Phishing Websites Based on Self-Structuring Neural Networks. Neural Computing and Applications, 25(2), 443-458. https://doi.org/10.1007/s00521-013-1510-7
ttps://doi.org/10.1007/s00521-013-1510-7

Multicollinearity

Multicollinearity occurs when two or more independent features are highly correlated, meaning they can predict one another better than they predict the dependent variable. Given the large number of numerical features in the dataset, **Variance Inflation Factor (VIF)** is an excellent method for evaluating multicollinearity. VIFmeasuress how much the variance of a feature is inflated due to its correlation with other features. Features with VIF values above 5 indicate moderate multicollinearity, while values above 10 suggest severe multicollinearity, which could negatively impact model performance by distorting the coefficients and interpretations of features.

In [19]:
```
!pip install statsmodels --upgrade
```

```
Requirement already satisfied: statsmodels in c:\users\pc\anaconda3\lib\site-packages (0.14.4)
Requirement already satisfied: numpy<3,>=1.22.3 in c:\users\pc\anaconda3\lib\site-packages (from statsmodels) (1.26.4)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in c:\users\pc\anaconda3\lib\site-packages (from statsmodels) (1.13.1)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in c:\users\pc\anaconda3\lib\site-packages (from statsmodels) (2.2.2)
Requirement already satisfied: patsy>=0.5.6 in c:\users\pc\anaconda3\lib\site-packages (from statsmodels) (0.5.6)
Requirement already satisfied: packaging>=21.3 in c:\users\pc\anaconda3\lib\site-packages (from statsmodels) (23.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\pc\anaconda3\lib\site-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\pc\anaconda3\lib\site-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\pc\anaconda3\lib\site-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2023.3)
Requirement already satisfied: six in c:\users\pc\anaconda3\lib\site-packages (from patsy>=0.5.6->statsmodels) (1.16.0)
[notice] A new release of pip is available: 24.2 -> 24.3.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [20]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.preprocessing import StandardScaler
numeric_data = data.select_dtypes(include=['float64', 'int64'])
```

```
scaler = StandardScaler()
numeric_data_scaled = scaler.fit_transform(numeric_data)

vif_data = pd.DataFrame()
vif_data['Feature'] = numeric_data.columns
vif_data['VIF'] = [variance_inflation_factor(numeric_data_scaled, i) for i :

print(vif_data)
```

	Feature	VIF
0	URLLength	2686.356585
1	DomainLength	4.620159
2	IsDomainIP	1.503856
3	URLSimilarityIndex	12.100384
4	CharContinuationRate	5.299405
5	TLDLegitimateProb	1.543365
6	URLCharProb	2.544787
7	TLDLength	1.336924
8	NoOfSubDomain	2.848905
9	HasObfuscation	2.871798
10	NoOfObfuscatedChar	7.511137
11	ObfuscationRatio	3.268967
12	NoOfLettersInURL	1334.523125
13	LetterRatioInURL	7.787741
14	NoOfDegitsInURL	225.794620
15	DegitRatioInURL	5.867963
16	NoOfEqualsInURL	11.319363
17	NoOfQMarkInURL	2.488912
18	NoOfAmpersandInURL	4.408761
19	NoOfOtherSpecialCharsInURL	38.713110
20	SpacialCharRatioInURL	6.862070
21	IsHTTPS	2.789535
22	LineOfCode	1.454770
23	LargestLineLength	1.020653
24	HasTitle	1.817035
25	DomainTitleMatchScore	17.249141
26	URLTitleMatchScore	16.398050
27	HasFavicon	1.423054
28	Robots	1.324184
29	IsResponsive	1.849486
30	NoOfURLRedirect	1.447802
31	NoOfSelfRedirect	1.482691
32	HasDescription	2.020243
33	NoOfPopup	1.006272
34	NoOfiFrame	1.227548
35	HasExternalFormSubmit	1.081355
36	HasSocialNet	2.750877
37	HasSubmitButton	1.942746
38	HasHiddenFields	1.688706
39	HasPasswordField	1.293296
40	Bank	1.255963
41	Pay	1.379860
42	Crypto	1.034932
43	HasCopyrightInfo	2.388814
44	NoOfImage	1.521246
45	NoOfCSS	1.086109
46	NoOfJS	1.255274
47	NoOfSelfRef	2.508847
48	NoOfEmptyRef	1.038200
49	NoOfExternalRef	2.451559
50	label	14.096734
51	special_char_in_domain_ratio	6.875081
52	Title_nonalphanumeric_count	9.836452
53	url_uppercase_ratio	1.001201
54	title_uppercase_ratio	1.001393
55	external_to_internal_ref_ratio	1.238128
56	image_to_text_ratio	1.954439
57	is_shortened_url	1.032982
58	tag_to_code_ratio	1.667590

Analysis

Although several features indicate to have a VIF score severly exceeding 10, most features are related to the URL metrics (such as URL length, URL letters, etc.). No features will be removed at this phase of the project however this insight can be used later on to improve the performance of the ML models.

Outlier handling

Outliers can have a negative impact on models' perfomrmance by adding an element of bias to the training process. By removing outliers, it can be ensured that the models focus on the core patterns in the data, leading to more reliable and accurate predictions. **Interquartile range (IQR) method will be used to remove the outliers.

In [21]:
```
# Step 1: Extract non-binary numeric feature
non_binary_numeric_features = data.loc[:, data.apply(lambda x: x.nunique() >

# Step 2: Create a copy of the original dataframe
filtered_data = data.copy()

# Step 3: Remove outliers using 3 IQR method
Q1 = filtered_data[non_binary_numeric_features].quantile(0.25)
Q3 = filtered_data[non_binary_numeric_features].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 3 * IQR
upper_bound = Q3 + 3 * IQR
filtered_data = filtered_data[~((filtered_data[non_binary_numeric_features]
                                (filtered_data[non_binary_numeric_features]
filtered_data.shape
```

Out[21]:
```
(75650, 68)
```

With **75,000 rows**, this dataset remains large enough to train robust machine learning models. Unfiltered data can be kept for any further exploratory data analysis (EDA) or to re-valuate/re-perform the outlier handling process.

In [22]:
```
#Saving data to excel files
data.to_csv('unfiltered_phishing_data.csv', index=False)
filtered_data.to_csv('filtered_phishing_data.csv', index=False)
```

In [23]:
```
#Downloading excel files
from IPython.display import FileLink
display(FileLink('unfiltered_phishing_data.csv'))
display(FileLink('filtered_phishing_data.csv'))
```

unfiltered_phishing_data.csv
filtered_phishing_data.csv

Phase 2: Implementation of Basic Machine Learning Models

Tasks:

- Implement foundational classification models, including **Logistic Regression** and **k-Nearest Neighbors (k-NN)**.

- Train the models on the preprocessed dataset.
- Evaluate model performance using standard metrics such as **accuracy**, **precision**, **recall**, **F1-score**, and **confusion matrix**.
- Apply **cross-validation** to measure model generalizability and robustness.

Deliverable:

A comprehensive report that includes:

- Detailed descriptions of model implementation,
- Performance evaluation using key metrics,
- Cross-validation results,
- The associated code provided in a **Jupyter Notebook**.

Model #1: kNN

In [24]:
```
# Import necessary libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
```

In [25]:
```
data = pd.read_csv("filtered_phishing_data.csv")
```

In [26]:
```
# display the first 5 rows
data.head()
```

Out[26]:

	FILENAME	URL	URLLength	Domain	DomainLength
0	mw42508.txt	http://www.teramill.com	22	www.teramill.com	16
1	7972389.txt	https://service-mitld.firebaseapp.com/	37	service-mitld.firebaseapp.com	29
2	62523.txt	https://www.vysor.io	19	www.vysor.io	12
3	386777.txt	https://www.ainewsletter.com	27	www.ainewsletter.com	20
4	527611.txt	https://www.sweatdrop.com	24	www.sweatdrop.com	17

5 rows × 68 columns

In [27]:
```
# check the shape of the data (rows, columns)
print(data.shape)
```

```
(75650, 68)
```

In [28]:
```
# Setting features and target data

data_df = pd.DataFrame(data=data, columns=[
    "URLLength",
    "DomainLength",
    "IsDomainIP",
    "TLD",
    "URLSimilarityIndex",
```

```
    "CharContinuationRate",
    "TLDLegitimateProb",
    "URLCharProb",
    "TLDLength",
    "NoOfSubDomain",
    "HasObfuscation",
    "NoOfObfuscatedChar",
    "ObfuscationRatio",
    "NoOfLettersInURL",
    "LetterRatioInURL",
    "NoOfDegitsInURL",
    "DegitRatioInURL",
    "NoOfEqualsInURL",
    "NoOfQMarkInURL",
    "NoOfAmpersandInURL",
    "NoOfOtherSpecialCharsInURL",
    "SpacialCharRatioInURL",
    "IsHTTPS",
    "LineOfCode",
    "LargestLineLength",
    "HasTitle",
    "DomainTitleMatchScore",
    "URLTitleMatchScore",
    "HasFavicon",
    "Robots",
    "IsResponsive",
    "NoOfURLRedirect",
    "NoOfSelfRedirect",
    "HasDescription",
    "NoOfPopup",
    "NoOfiFrame",
    "HasExternalFormSubmit",
    "HasSocialNet",
    "HasSubmitButton",
    "HasHiddenFields",
    "HasPasswordField",
    "Bank",
    "Pay",
    "Crypto",
    "HasCopyrightInfo",
    "NoOfImage",
    "NoOfCSS",
    "NoOfJS",
    "NoOfSelfRef",
    "NoOfEmptyRef",
    "NoOfExternalRef",
    "Domain_contains_number",
    "Title_contains_number",
    "Title_contains_specialcharacter",
    "Title_nonalphanumeric_count",
    "url_uppercase_ratio",
    "title_uppercase_ratio",
    "external_to_internal_ref_ratio",
    "image_to_text_ratio",
    "is_shortened_url",
    "tag_to_code_ratio"
]
)

print(data_df.head())
```

```
   URLLength  DomainLength  IsDomainIP  TLD  URLSimilarityIndex  \
0         22            16           0  com           82.644628
1         37            29           0  com           64.645264
2         19            12           0   io          100.000000
3         27            20           0  com          100.000000
4         24            17           0  com          100.000000

   CharContinuationRate  TLDLegitimateProb  URLCharProb  TLDLength  \
0                  1.00           0.522907     0.067418          3
1                  0.48           0.522907     0.059401          3
2                  1.00           0.012927     0.051312          2
3                  1.00           0.522907     0.066271          3
4                  1.00           0.522907     0.062493          3

   NoOfSubDomain  ...  Domain_contains_number  Title_contains_number  \
0              1  ...                       0                      1
1              1  ...                       0                      1
2              1  ...                       0                      0
3              1  ...                       0                      0
4              1  ...                       0                      0

   Title_contains_specialcharacter  Title_nonalphanumeric_count  \
0                                0                            0
1                                0                            0
2                                0                            0
3                                0                            0
4                                0                            0

   url_uppercase_ratio  title_uppercase_ratio  external_to_internal_ref_rat
io  \
0                  0.0                    0.0                        1.0000
00
1                  0.0                    0.0                        0.0000
00
2                  0.0                    0.0                        0.5000
00
3                  0.0                    0.0                        0.9545
45
4                  0.0                    0.0                        0.3000
00

   image_to_text_ratio  is_shortened_url  tag_to_code_ratio
0             0.000000                 0           0.000000
1             0.000000                 0           0.000000
2             0.013043                 0           0.065217
3             0.003067                 0           0.009202
4             0.076159                 0           0.096026

[5 rows x 61 columns]
```

In [29]:

```python
# Convert non-numeric values to numeric
# TLD: Convert to categorical numeric
data['TLD'] = data['TLD'].astype('category').cat.codes

# IsDomainIP: Convert to 0, 1
data['IsDomainIP'] = data['IsDomainIP'].replace({'Yes': 1, 'No': 0}).astype(

# HasObfuscation: Convert to 0, 1
data['HasObfuscation'] = data['HasObfuscation'].replace({'Yes': 1, 'No': 0})

# Title, Bank, Pay, Crypto: Encoding via a simple method or keep as is for T
# Example: You can convert them to category codes if needed
for column in ['Title', 'Bank', 'Pay', 'Crypto']:
    data[column] = data[column].astype('category').cat.codes
```

```
# HasCopyrightInfo: Convert to 0, 1
data['HasCopyrightInfo'] = data['HasCopyrightInfo'].replace({'Yes': 1, 'No'

# Domain_starts_with_www.: Convert to 0, 1
data['Domain_starts_with_www.'] = data['Domain_starts_with_www.'].replace({

# Domain_contains_number: Convert to 0, 1
data['Domain_contains_number'] = data['Domain_contains_number'].replace({'Ye

# Title_contains_number: Convert to 0, 1
data['Title_contains_number'] = data['Title_contains_number'].replace({'Yes

# Title_contains_specialcharacter: Convert to 0, 1
data['Title_contains_specialcharacter'] = data['Title_contains_specialcharac

# Title_nonalphanumeric_count: Keep as is (already numeric)

# IsHTTPS: Convert to 0, 1
data['IsHTTPS'] = data['IsHTTPS'].replace({'Yes': 1, 'No': 0}).astype(int)

# url_uppercase_ratio: Map to numbers
data['url_uppercase_ratio'] = data['url_uppercase_ratio'].replace({'High': 2

# title_uppercase_ratio: Map to numbers
data['title_uppercase_ratio'] = data['title_uppercase_ratio'].replace({'High

# external_to_internal_ref_ratio: Map to numbers
data['external_to_internal_ref_ratio'] = data['external_to_internal_ref_rati

# is_shortened_url: Convert to 0, 1
data['is_shortened_url'] = data['is_shortened_url'].replace({'Yes': 1, 'No'

# special_char_in_domain_ratio: Map to numbers
data['special_char_in_domain_ratio'] = data['special_char_in_domain_ratio']
```

In [30]:
```
# Split the dataset into training and testing sets
X = data.drop(['label', 'URL', 'FILENAME', 'Domain', 'Title'], axis=1)
y = data['label']

SEED = 42
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, ra
```

In [31]:
```
# Standardize the dataset. Scale both X_train and X_test

# use the function scaler.transform
scaler = StandardScaler()

# Fit only on X_train
scaler.fit(X_train)

# Scale both X_train and X_test
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Train a KNN model with K=5
# use the function KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
```

Out[31]:

KNeighborsClassifier

KNeighborsClassifier()

In [32]:
```
# Make predictions
y_pred = knn.predict(X_test)
```

In [33]:
```
# Evaluate the model (calculate accuracy and confusion matrix)
accuracy = accuracy_score(y_test, y_pred)
confusion_matrix = confusion_matrix(y_test, y_pred)
```

In [34]:
```
# Print the result of the accuracy and confusion matrix
print("Accuracy:", accuracy)
print("Confusion Matrix:", confusion_matrix)
```

```
Accuracy: 0.9987310315655898
Confusion Matrix: [[ 6386    18]
 [    6 12503]]
```

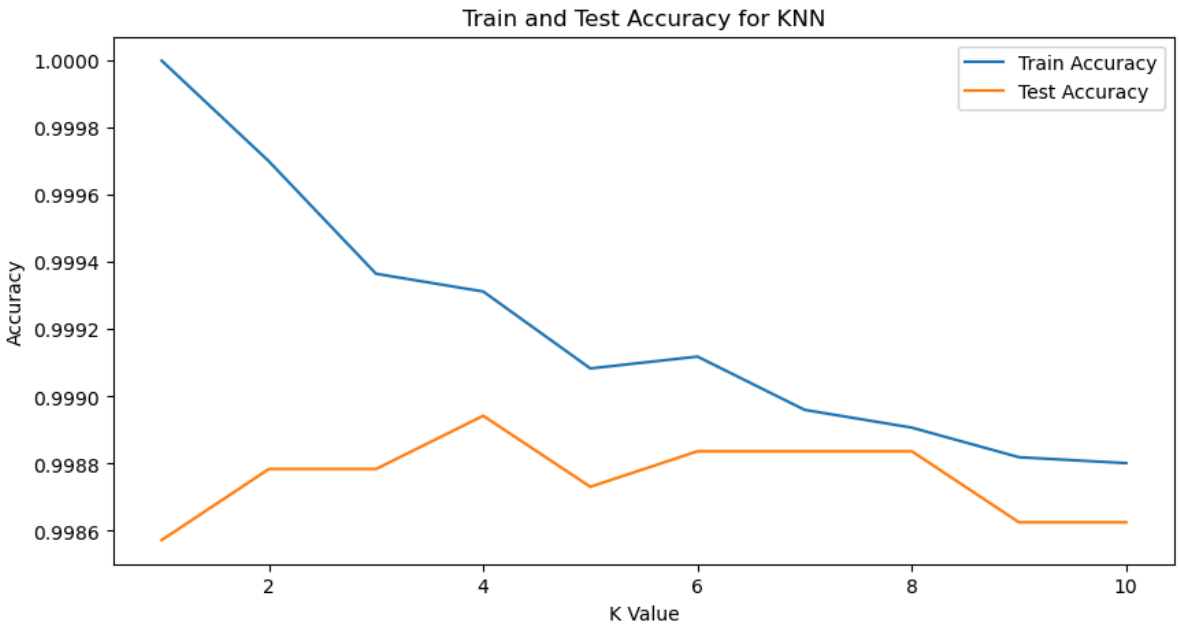
In [35]:
```
# Perform 5-fold cross-validation using KNN with K=5
# use the function cross_val_score
cv_scores = cross_val_score(knn, X, y, cv=5)
```

In [36]:
```
# Displaying the cross-validation scores and mean accuracy
print("Cross-validation scores:", cv_scores)
print("Mean accuracy:", cv_scores.mean())
```

```
Cross-validation scores: [0.99775281 0.99636484 0.99576999 0.99576999 0.996
10046]
Mean accuracy: 0.9963516192994053
```

In [37]:
```
# Calculate accuracy for K values between 1 and 10. Calculate both the trai
train_accuracies = []
test_accuracies = []
for k in range(1, 11):
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(X_train, y_train)
        train_accuracy = knn.score(X_train, y_train)
        test_accuracy = knn.score(X_test, y_test)
        train_accuracies.append(train_accuracy)
        test_accuracies.append(test_accuracy)
```

In [38]:
```
# Visualize the results of step 10, including the train and test accuracies.
plt.figure(figsize=(10, 5))
plt.plot(range(1, 11), train_accuracies, label='Train Accuracy')
plt.plot(range(1, 11), test_accuracies, label='Test Accuracy')
plt.xlabel('K Value')
plt.ylabel('Accuracy')
plt.title('Train and Test Accuracy for KNN')
plt.legend()
plt.show()
```



Logistic Regression

1. Import Libraries

In [39]:
```
import pandas as pd
import joblib
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, precision_score, recall_score, 
from sklearn.model_selection import KFold, cross_val_score
```

In [40]:
```
df = data

# Drop any columns that are not features or target (e.g., 'FILENAME', which
X = df.drop(columns=['FILENAME', 'label'])  # Assuming 'label' is the targe
y = df['label']  # Define the target variable
```

In [41]:
```
# Identify categorical, text, and numerical columns
#categorical_columns = ['IsDomainIP', 'IsHTTPS', 'Bank', 'Pay', 'Crypto']
text_columns = ['URL', 'Title']  # Text data columns
numerical_columns = X.select_dtypes(include=['int64', 'float64']).columns.to
```

In [42]:
```
# Define the preprocessing pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_columns),  # Scale numerical co
        #('cat', OneHotEncoder(handle_unknown='ignore'), categorical_columns
    ], remainder='drop')  # We will drop text columns for simplicity
```

2. Split the Data into Training and Testing Sets

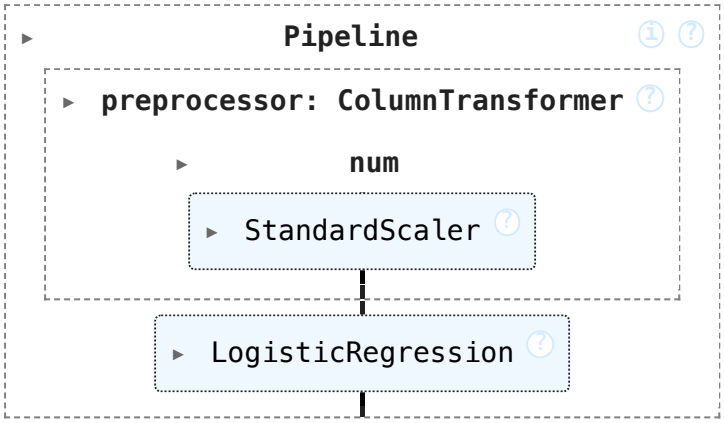
In [43]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

3. Define the Logistic Regression Model Using a Pipeline

In [44]:
```
# Define a Logistic Regression model
log_reg_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', LogisticRegression(max_iter=1000))  # Increased max_iter
])
```

In [45]:
```
# Train the Logistic Regression model
log_reg_pipeline.fit(X_train, y_train)
```

Out[45]:



4. Train the Logistic Regression Model

In [46]:
```
y_pred = log_reg_pipeline.predict(X_test)
y_pred
```

Out[46]:
```
array([1, 0, 1, ..., 0, 1, 0], dtype=int64)
```

5. Evaluate the Model

In [47]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
conf_matrix = confusion_matrix(y_test, y_pred)
# Print evaluation metrics
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
F1 Score: 1.0000
Confusion Matrix:
[[ 5100     0]
 [    0 10030]]
```

6. Perform K-Fold Cross-Validation

In [48]:
```
# Perform cross-validation
cv = KFold(n_splits=5, random_state=42, shuffle=True)
cv_scores = cross_val_score(log_reg_pipeline, X_train, y_train, cv=cv, scor
```

7. Results Summary

In [49]:
```
# Results
{
    "accuracy": accuracy,
    "precision": precision,
    "recall": recall,
    "f1_score": f1,
    "confusion_matrix": conf_matrix,
    "cross_val_scores": cv_scores,
    "mean_cross_val_score": cv_scores.mean()
}
```

Out[49]:
```
{'accuracy': 1.0,
 'precision': 1.0,
 'recall': 1.0,
 'f1_score': 1.0,
 'confusion_matrix': array([[ 5100,     0],
        [    0, 10030]], dtype=int64),
 'cross_val_scores': array([0.99958691, 0.99983477, 0.9995043 , 0.99975215,
0.99991738]),
 'mean_cross_val_score': 0.9997191011235955}
```

8. Save the Trained Model and Preprocessing Pipeline

In [50]:
```
# Save the trained pipeline (preprocessor + logistic regression model)
joblib.dump(log_reg_pipeline, 'logistic_regression_pipeline.pkl')

# Save the cross-validation scores for reference
np.save('cv_scores.npy', cv_scores)
# joblib.dump(preprocessor, 'preprocessor.pkl')  # Already included in the p
```

Explanation:

1. Import Libraries: We first import the necessary libraries for model training, evaluation, and cross-validation.
2. Data Splitting: The dataset is split into training and testing sets with an 80-20 ratio.
3. Model Definition: A pipeline is defined where preprocessing is followed by logistic regression. The logistic regression model is set with a higher iteration limit (max_iter=1000) to ensure convergence.
4. Model Training: The model is trained on the training set, and predictions are made on the test set.

5. Model Evaluation: Key metrics such as accuracy, precision, recall, F1 score, and the confusion matrix are calculated based on predictions on the test set.
6. Cross-Validation: K-fold cross-validation with 5 splits is performed on the training set to evaluate the model's performance across different data subsets. The mean cross-validation score gives an estimate of the model's generalization performance.
7. Results Summary: The results, including cross-validation scores, are summarized and presented in a dictionary for easy reference.
8. Saving Model: Model is saved for deployment.

Phase 2 summary

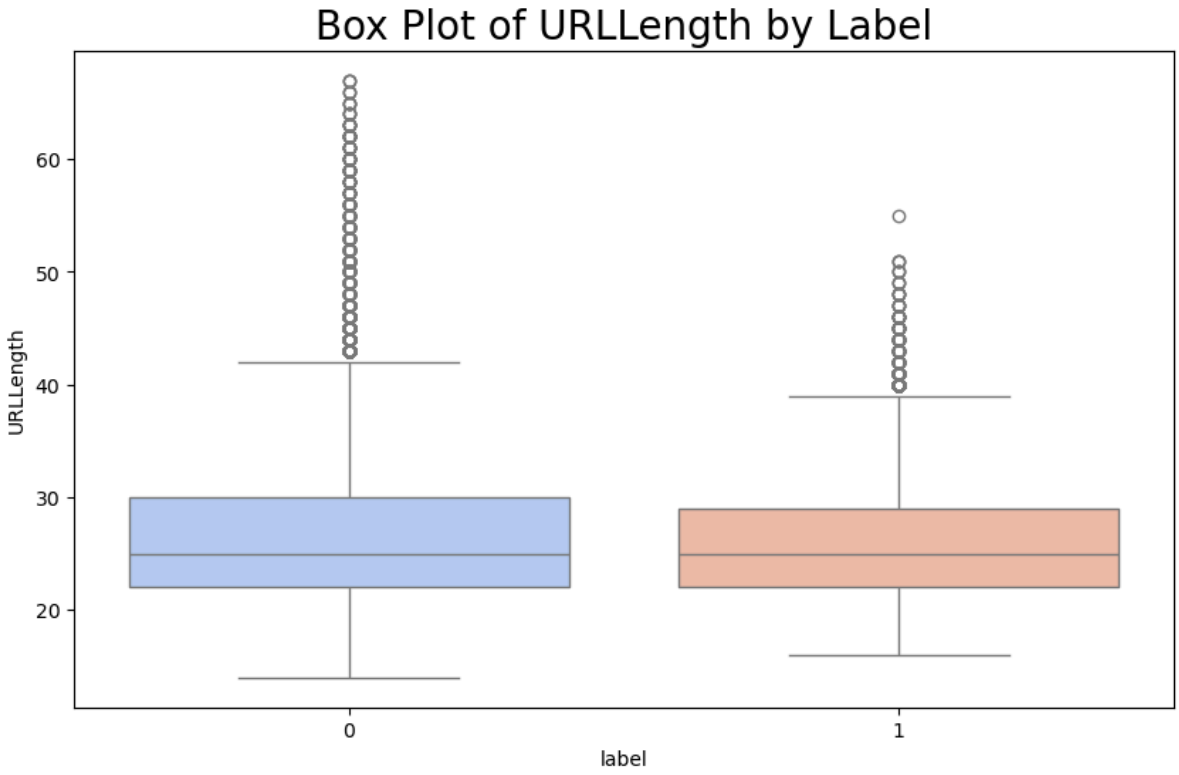
Phase 3: Building advanced models

- Tasks:
  - Implement advanced machine learning models, including:
    - XGBoost: A powerful ensemble method using gradient boosting.
    - Support Vector Machine (SVM): For finding the optimal boundary between classes.
    - Neural Networks: A simple feed-forward neural network architecture for classification.
    - Choice of Advanced Neural Network (e.g., CNN, RNN, Transformer Based, etc.)
  - Compare the performance of advanced models with basic models.
  - Optimize hyperparameters using grid search or random search.
  - Use additional evaluation metrics like AUC-ROC curves for model performance.
- Deliverable: A comprehensive report comparing the performance of basic vs. advanced models, optimization steps, and final results in terms of model accuracy and robustness, along with the Jupyter Notebook containing the code

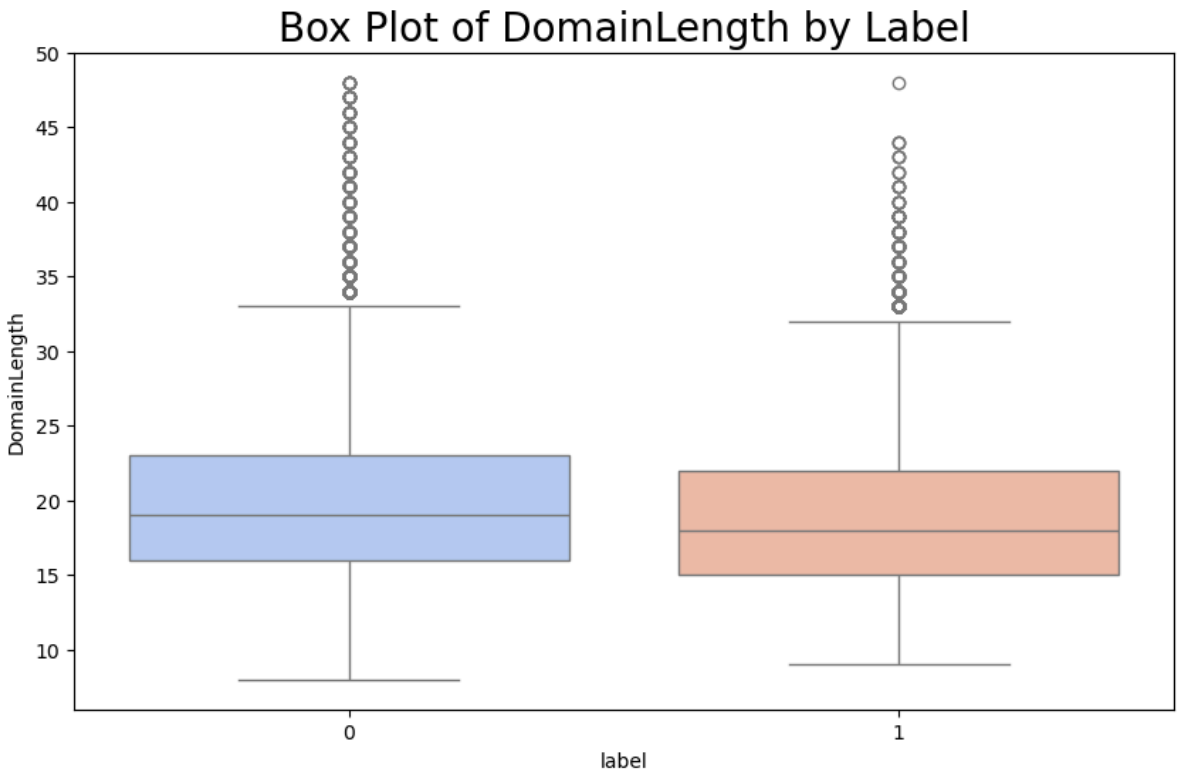
Simple NN model

In [51]:
```
import pandas as pd
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = pd.read_csv("filtered_phishing_data.csv")

df2 = pd.read_csv("filtered_phishing_data.csv")

# Visualize the distribution of different features by label using box plots
features = ['URLLength', 'DomainLength', 'NoOfLettersInURL', 'NoOfDegitsInUR
for feature in features:
    plt.figure(figsize=(10, 6))
    sns.boxplot(x='label', y=feature, data=df, palette='coolwarm')
    plt.title(f'Box Plot of {feature} by Label', size=20)
    plt.show()
```

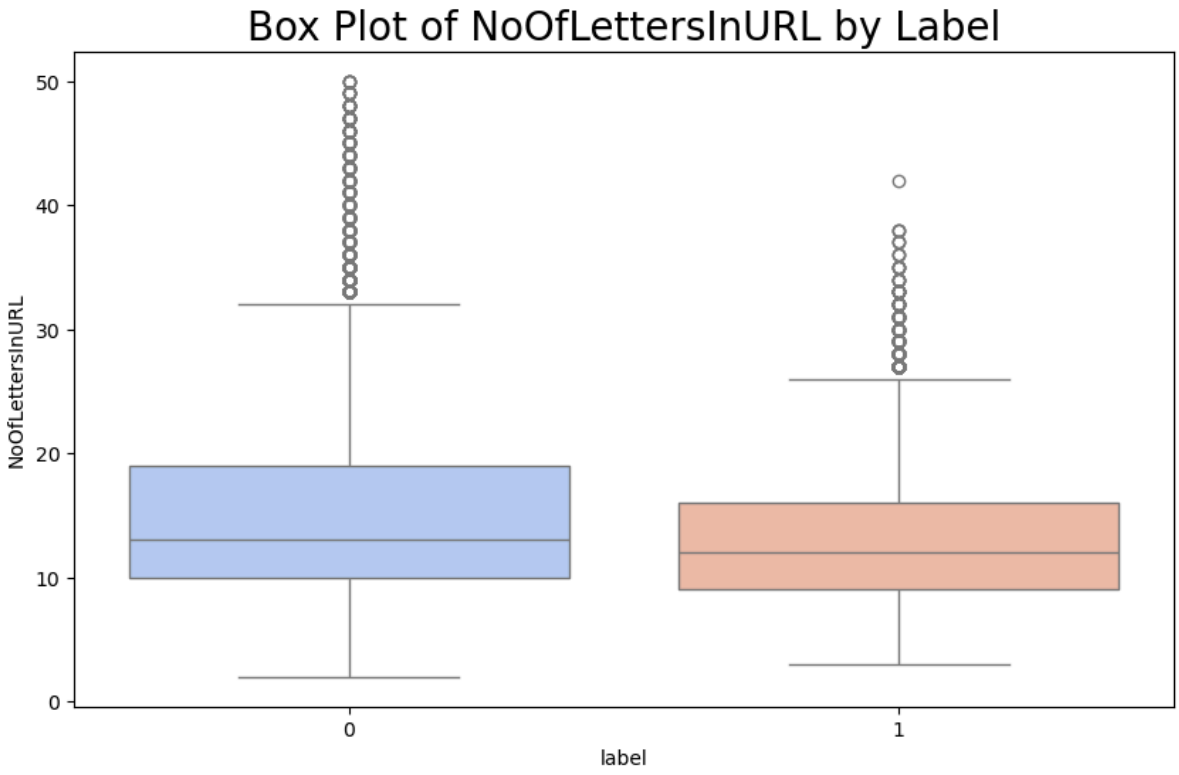
```
C:\Users\PC\AppData\Local\Temp\ipykernel_9768\3253874369.py:14: FutureWarni
ng:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.boxplot(x='label', y=feature, data=df, palette='coolwarm')
```



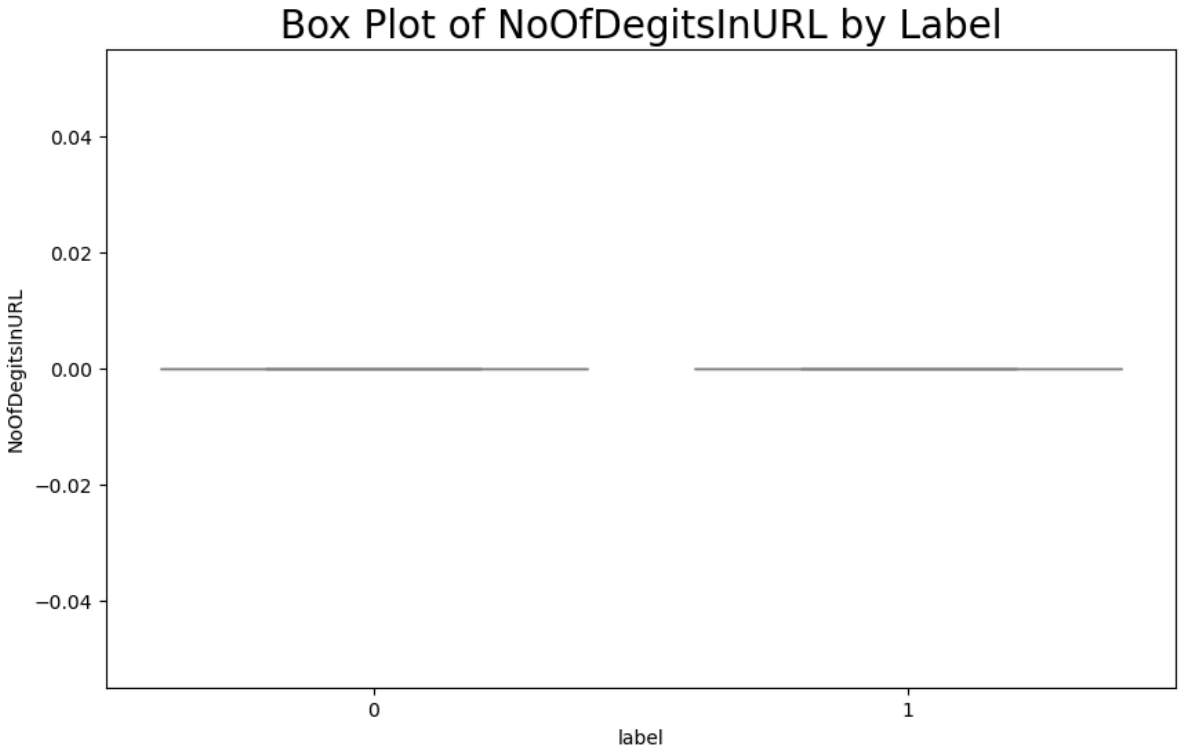

```
C:\Users\PC\AppData\Local\Temp\ipykernel_9768\3253874369.py:14: FutureWarni
ng:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.boxplot(x='label', y=feature, data=df, palette='coolwarm')
```




```
C:\Users\PC\AppData\Local\Temp\ipykernel_9768\3253874369.py:14: FutureWarni
ng:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.boxplot(x='label', y=feature, data=df, palette='coolwarm')
```



```
C:\Users\PC\AppData\Local\Temp\ipykernel_9768\3253874369.py:14: FutureWarni
ng:

Passing `palette` without assigning `hue` is deprecated and will be removed
in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the
same effect.

  sns.boxplot(x='label', y=feature, data=df, palette='coolwarm')
```



In [52]:
```
# split df1 into feature1 and target1
feature1 = df.drop(columns=['FILENAME', 'URL', 'Domain', 'TLD', 'Title', 'la
target1 = df['label']
```

K-fold cross validation

In [53]:
```
# create an instance of stratified K-fold cross validation
from sklearn.model_selection import StratifiedKFold

K = 5

skf = StratifiedKFold(n_splits=K, shuffle=True, random_state=0) # 5-fold cro
```

MLP neural network

In [54]:
```
from sklearn.neural_network import MLPClassifier

# create an instance of an MLP neural network

# 1000 maxinum number of epoches
# 1 layer of 30 hidden neurons
mlp = MLPClassifier(random_state=1, max_iter=1000, hidden_layer_sizes=(30))

total_train_score = 0
total_validate_score = 0

fold = 1
for train_index, validate_index in skf.split(feature1, target1):
    X_train, X_validate = feature1.iloc[train_index], feature1.iloc[validate
```

```
    y_train, y_validate = target1.iloc[train_index], target1.iloc[validate_i

    mlp.fit(X_train, y_train)
    train_score = mlp.score(X_train, y_train)
    validate_score = mlp.score(X_validate, y_validate)

    print(f"Fold {fold}: Train Score = {train_score}, Validation Score = {va

    total_train_score += train_score
    total_validate_score += validate_score

    fold += 1

avg_train_score = total_train_score / K
avg_validate_score = total_validate_score / K

print(f"Average Train Score: {avg_train_score}")
print(f"Average Validation Score: {avg_validate_score}")
```

```
Fold 1: Train Score = 0.9993721083939193, Validation Score = 0.999140779907
4686
Fold 2: Train Score = 0.9987937871777924, Validation Score = 0.998215465961
6655
Fold 3: Train Score = 0.9986450760079313, Validation Score = 0.998612029081
2954
Fold 4: Train Score = 0.9989755452742894, Validation Score = 0.998810310641
1104
Fold 5: Train Score = 0.9995703899537343, Validation Score = 0.999206873760
7402
Average Train Score: 0.9990713813615333
Average Validation Score: 0.9987970918704562
```

In [55]:

```
# create another instance of an MLP neural network with different hyperparam

# 1000 maxinum number of epoches
# 2 layer of 10 hidden neurons each
mlp = MLPClassifier(random_state=1, max_iter=1000, hidden_layer_sizes=(10, 1

# ==================================

# Run K-fold cross validation again

total_train_score = 0
total_validate_score = 0

fold = 1
for train_index, validate_index in skf.split(feature1, target1):
    print("Fold", fold)
    print("=======")
    X_train  = feature1.loc[train_index, :]
    X_validate = feature1.loc[validate_index, :]
    y_train = target1.loc[train_index]
    y_validate = target1.loc[validate_index]

    print("TRAIN:", X_train.shape, y_train.shape)
    # number of samples for each class
    # print(pd.Index(y_train).value_counts())

    print("VALIDATION:", X_validate.shape, y_validate.shape)
    # number of samples for each class
    # print(pd.Index(y_validate).value_counts())

    # fit mlp with the training data for this fold
    mlp.fit(X_train, y_train)
```

```
    # get the score (accuracy) for the training set
    train_score = mlp.score(X_train, y_train)
    total_train_score += train_score

    # get the score (accuracy) for the validation set
    validate_score = mlp.score(X_validate, y_validate)
    total_validate_score += validate_score

    print("Training set score: {:.2f}".format(train_score))
    print("Validation set score: {:.2f}".format(validate_score))
    print("==================================")

    fold += 1
    print()

print("##################################")
print("Average training set score: {:.2f}".format(total_train_score/K))
print("Average validation set score: {:.2f}".format(total_validate_score/K))
```

```
Fold 1
=======
TRAIN: (60520, 62) (60520,)
VALIDATION: (15130, 62) (15130,)
Training set score: 1.00
Validation set score: 1.00
==================================

Fold 2
=======
TRAIN: (60520, 62) (60520,)
VALIDATION: (15130, 62) (15130,)
Training set score: 1.00
Validation set score: 1.00
==================================

Fold 3
=======
TRAIN: (60520, 62) (60520,)
VALIDATION: (15130, 62) (15130,)
Training set score: 1.00
Validation set score: 1.00
==================================

Fold 4
=======
TRAIN: (60520, 62) (60520,)
VALIDATION: (15130, 62) (15130,)
Training set score: 1.00
Validation set score: 1.00
==================================

Fold 5
=======
TRAIN: (60520, 62) (60520,)
VALIDATION: (15130, 62) (15130,)
Training set score: 1.00
Validation set score: 1.00
==================================

##################################
Average training set score: 1.00
Average validation set score: 1.00
```

In [56]:
```
# Choose the model that gives you the best average validation set score (i.e
# (In case of the same aveage validation set score, choose the one with the

mlp = MLPClassifier(random_state=1, max_iter=1000, hidden_layer_sizes=(10, 1

# Fit the chosen model again using the whole training set (Why is this neede
mlp.fit(feature1, target1)

# Apply the chosen model to the (final) testing set
X_test = df2.drop(columns=['FILENAME', 'URL', 'Domain', 'TLD', 'Title', 'lab
y_test = df2['label']

test_score = mlp.score(X_test, y_test)
print("Testing set score: {:.2f}".format(test_score))
```

```
Testing set score: 1.00
```

Comparing Simple NN and kNN (should be a comparison between all models at once, let's do it once all models are ready)

In [57]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

# Define the kNN model
knn = KNeighborsClassifier(n_neighbors=5)

# Fit the kNN model using the whole training set
knn.fit(feature1, target1)

# Predictions for the testing set
y_pred_knn = knn.predict(X_test)

# Calculate the accuracy of the kNN model on the testing set
test_accuracy_knn = accuracy_score(y_test, y_pred_knn)
print("Testing set accuracy (kNN): {:.2f}".format(test_accuracy_knn))

# Compare with MLP performance
print("Testing set accuracy (MLP): {:.2f}".format(test_score))
```

```
Testing set accuracy (kNN): 1.00
Testing set accuracy (MLP): 1.00
```

We find no difference in prediction accuracy between a simple NN and kNN model.

Optimizing hyperparameters using grid search

In [58]:
```
from sklearn.model_selection import GridSearchCV

# Define the parameter grid for MLP
parameter_space = {
    'hidden_layer_sizes': [(10, 10), (50,50), (100,)],
    'activation': ['tanh', 'relu'],
    'solver': ['sgd', 'adam'],
    'alpha': [0.0001, 0.05],
    'learning_rate': ['constant','adaptive'],
}

# Create a GridSearchCV object with MLPClassifier
mlp_gs = GridSearchCV(mlp, parameter_space, n_jobs=-1, cv=3)
mlp_gs.fit(feature1, target1)
```

```
# Print the best parameters
print("Best parameters found:", mlp_gs.best_params_)

# Apply the best model to the (final) testing set
best_mlp = mlp_gs.best_estimator_
test_score_gs = best_mlp.score(X_test, y_test)
print("Testing set score after GridSearch: {:.2f}".format(test_score_gs))
```

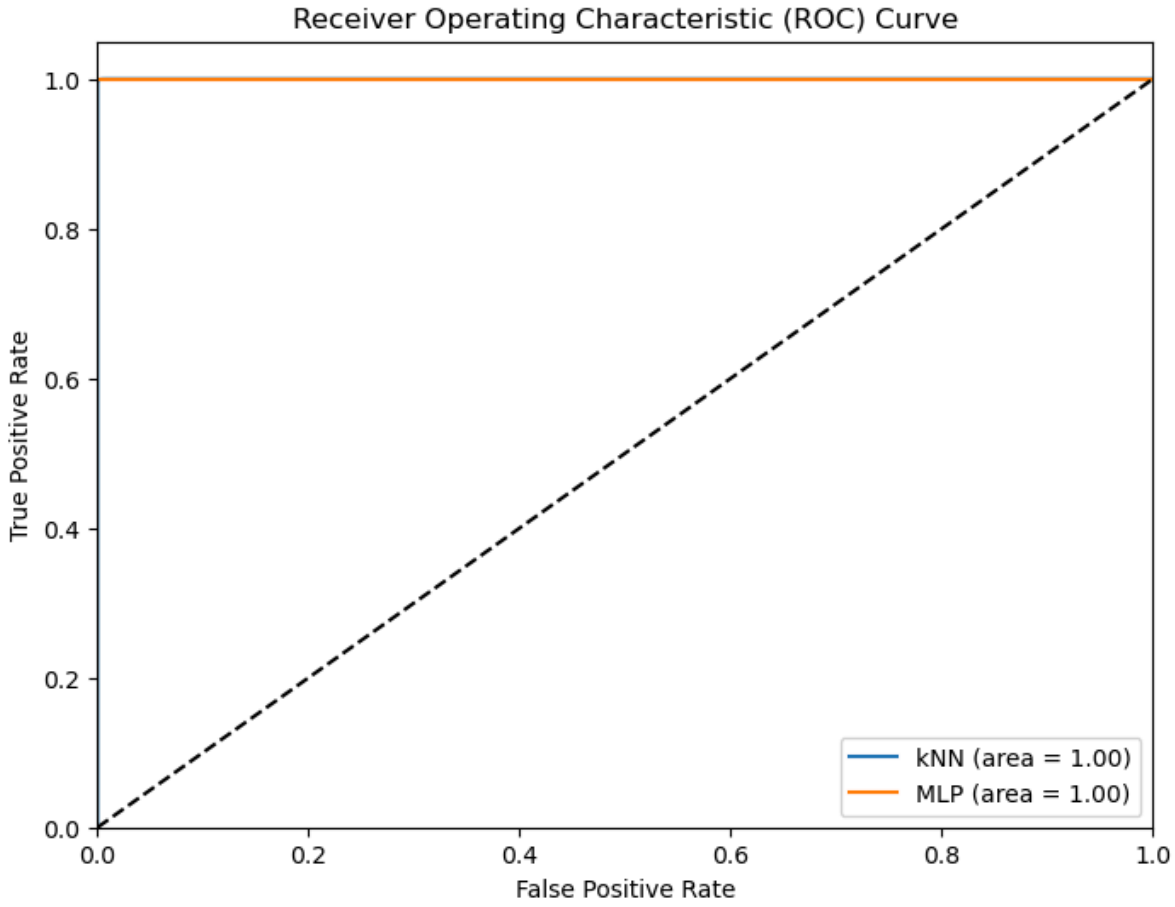
```
Best parameters found: {'activation': 'relu', 'alpha': 0.0001, 'hidden_laye
r_sizes': (10, 10), 'learning_rate': 'constant', 'solver': 'adam'}
Testing set score after GridSearch: 1.00
```

AUC-ROC curves for model performance

In [59]:

```
from sklearn.metrics import roc_auc_score, roc_curve, auc
import matplotlib.pyplot as plt

# Calculate the probabilities of each class for the testing set
y_prob_knn = knn.predict_proba(X_test)[:, 1]
y_prob_mlp = best_mlp.predict_proba(X_test)[:, 1]

# Calculate AUC-ROC scores
roc_auc_knn = roc_auc_score(y_test, y_prob_knn)
roc_auc_mlp = roc_auc_score(y_test, y_prob_mlp)

print("AUC-ROC (kNN): {:.2f}".format(roc_auc_knn))
print("AUC-ROC (MLP): {:.2f}".format(roc_auc_mlp))

# Calculate ROC curve for kNN
fpr_knn, tpr_knn, _ = roc_curve(y_test, y_prob_knn)
# Calculate ROC curve for MLP
fpr_mlp, tpr_mlp, _ = roc_curve(y_test, y_prob_mlp)

# Plot the ROC curve
plt.figure(figsize=(8, 6))
plt.plot(fpr_knn, tpr_knn, label='kNN (area = %0.2f)' % roc_auc_knn)
plt.plot(fpr_mlp, tpr_mlp, label='MLP (area = %0.2f)' % roc_auc_mlp)
plt.plot([0, 1], [0, 1], 'k--')  # random predictions curve
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```

```
AUC-ROC (kNN): 1.00
AUC-ROC (MLP): 1.00
```



In [60]:
```
data.shape
```

Out[60]:
```
(75650, 68)
```

Support vector machine

In [61]:
```
#import libraries
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score,confusion_
```

In [68]:
```
#split data
y = data['label']
x = data.drop(['FILENAME', 'URL', 'Domain', 'TLD', 'Title', 'label'], axis =
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.5, random_s
```

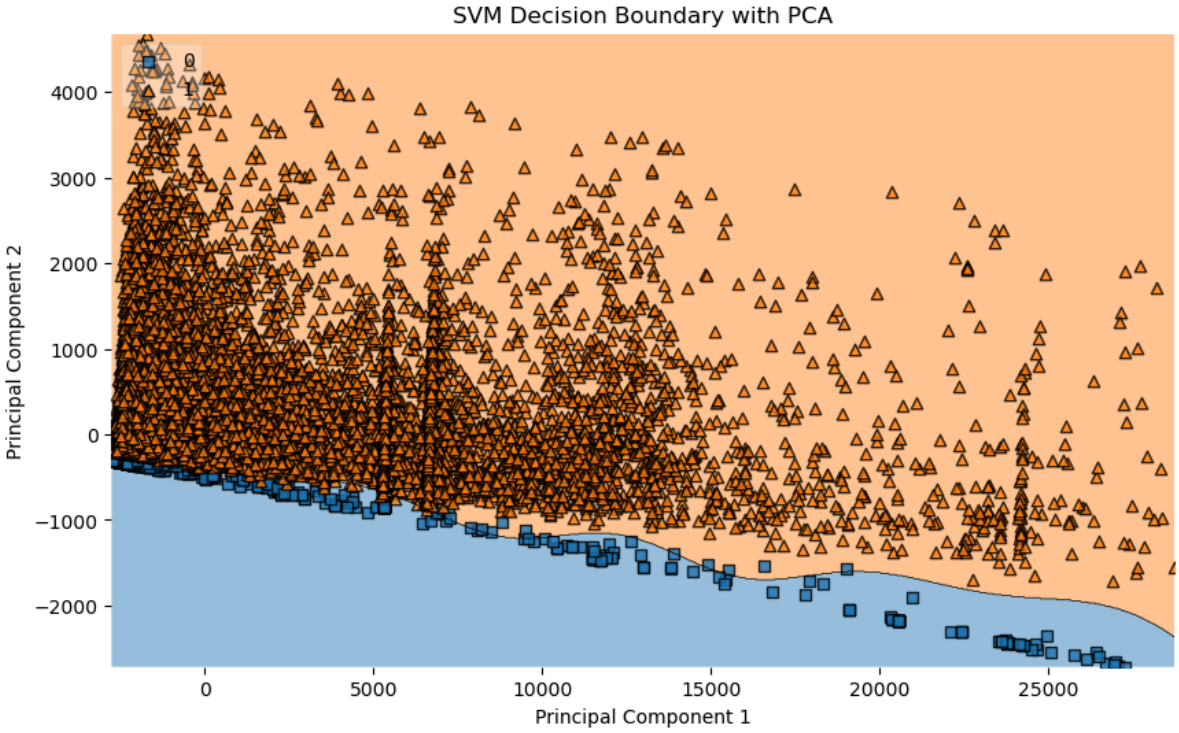
In [70]:
```
#train model using rbf kernel
svm_model = SVC(kernel='rbf')
svm_model.fit(x_train,y_train)
y_pred = svm_model.predict(x_test)
```

In [71]:
```
# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy * 100:.2f}%")

# Detailed classification report
print(classification_report(y_test, y_pred))

print(confusion_matrix(y_test,y_pred))
```

```
Accuracy: 98.82%
              precision    recall  f1-score   support

           0       0.97      0.99      0.98     12765
           1       1.00      0.99      0.99     25060

    accuracy                           0.99     37825
   macro avg       0.98      0.99      0.99     37825
weighted avg       0.99      0.99      0.99     37825

[[12655   110]
 [  335 24725]]
```

In [72]:

```python
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from mlxtend.plotting import plot_decision_regions

# Apply PCA to reduce the features to 2 dimensions
pca = PCA(n_components=2)
x_train_pca = pca.fit_transform(x_train)

# Refit the SVM model on the reduced data
svm_model.fit(x_train_pca, y_train)

# Plot the decision boundary
plt.figure(figsize=(10, 6))
plot_decision_regions(x_train_pca, y_train.values, clf=svm_model, legend=2)
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('SVM Decision Boundary with PCA')
plt.show()
```



XGboost

In [67]:

```python
#import libraries
import xgboost as xgb
```

In [73]:
```
#intialize model
xgb_model = xgb.XGBClassifier(objective = 'binary:logistic', eval_metric =

#Train the model
xgb_model.fit(x_train,y_train)
```

```
c:\Users\PC\anaconda3\Lib\site-packages\xgboost\core.py:158: UserWarning:
[09:21:33] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autosca
ling-group-i-0015a694724fa8361-1\xgboost\xgboost-ci-windows\src\learner.cc:
740:
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

Out[73]:

XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_
rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold
=None,
              max_cat_to_onehot=None, max_delta_step=None, max_de
pth=None,
```

In [74]:
```
#model evaluation
y_pred = xgb_model.predict(x_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy * 100:.2f}%')

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

print("\nConfusion Matrix: ")
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 100.00%

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     12765
           1       1.00      1.00      1.00     25060

    accuracy                           1.00     37825
   macro avg       1.00      1.00      1.00     37825
weighted avg       1.00      1.00      1.00     37825


Confusion Matrix:
[[12764     1]
 [    0 25060]]
```

In [76]:
```
#Model evaluation on large sum, unfiltered data

unfiltered_data = pd.read_csv('unfiltered_phishing_data.csv')
x_unfiltered = unfiltered_data.drop(['FILENAME', 'URL', 'Domain', 'TLD', 'T
```

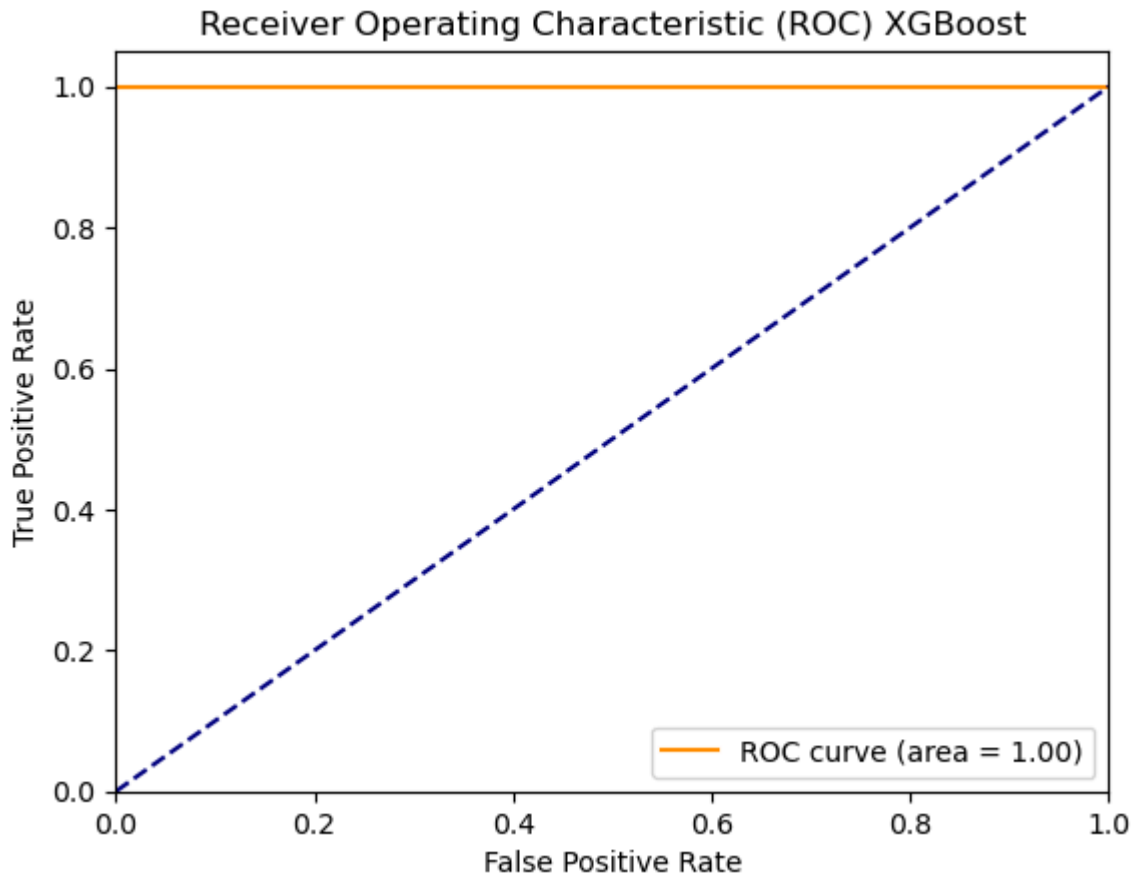
```
y_unfiltered = unfiltered_data['label']

y_pred_unfiltered = xgb_model.predict(x_unfiltered)

unfiltered_accuracy = accuracy_score(y_unfiltered,y_pred_unfiltered)
print(f'Unfiltered Accuracy : {unfiltered_accuracy * 100:.2f} %')

print("\nClassification Report")
print(classification_report(y_unfiltered,y_pred_unfiltered))

print("\nConfusion Matrix")
print(confusion_matrix(y_unfiltered,y_pred_unfiltered))
```

```
Unfiltered Accuracy : 100.00 %

Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    100945
           1       1.00      1.00      1.00    134850

    accuracy                           1.00    235795
   macro avg       1.00      1.00      1.00    235795
weighted avg       1.00      1.00      1.00    235795


Confusion Matrix
[[100942      3]
 [     0 134850]]
```

In [81]:
```
#hyperparameter tuning usng grid search

from sklearn.model_selection import GridSearchCV

param_grid = {'max_depth' : [3,5,7], 'learning_rate' : [0.01,0.1,0.2], 'n_es

grid_search = GridSearchCV(estimator = xgb.XGBClassifier(eval_metric = 'log

grid_search.fit(x_train, y_train)

print(f"Best Parameters: {grid_search.best_params_}")
print(f"Bets Accuracy: {grid_search.best_score_ * 100:.2f}%")
```

```
Best Parameters: {'learning_rate': 0.01, 'max_depth': 3, 'n_estimators': 10
0}
Bets Accuracy: 100.00%
```

In [83]:
```
#Plotting AUC-ROC curve

y_pred_proba = grid_search.predict_proba(x_test)[:,1]

fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', label=f'ROC curve (area = {roc_auc:.2
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) XGBoost')
plt.legend(loc="lower right")
plt.show()
```



Advanced model (Decision Trees)

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion

# Load your dataset
data = pd.read_csv('filtered_phishing_data.csv')  # Replace with your actual
```

```
Accuracy: 1.00

Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2439
           1       1.00      1.00      1.00      7466

    accuracy                           1.00      9905
   macro avg       1.00      1.00      1.00      9905
weighted avg       1.00      1.00      1.00      9905


Confusion Matrix:
[[2439    0]
 [   0 7466]]
```

Preparing the data for decision tree

In [ ]:
```
# Prepare features and target
X = data.drop(['label', 'FILENAME', 'URL', 'Domain', 'TLD', 'Title'], axis=1
y = data['label']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

Train model using `DecisionTreeClassifier`

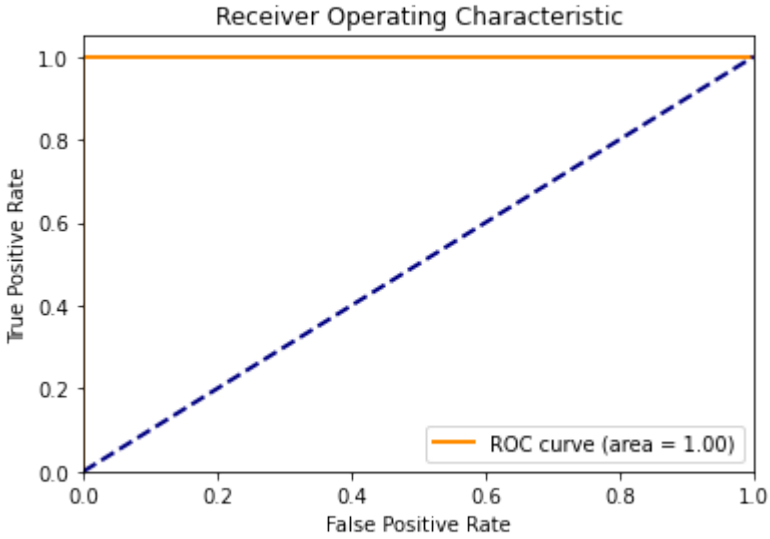
In [ ]:
```
# Initialize the Decision Tree Classifier
dt_classifier = DecisionTreeClassifier(random_state=42)

# Train the model
dt_classifier.fit(X_train, y_train)

# Make predictions
y_pred = dt_classifier.predict(X_test)
```

Calcuting Accuracy for Decision Trees

In [ ]:
```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

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

print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

Decision Tree AUC-ROC Analysis

In [2]:
```
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Compute ROC curve and ROC area for each class
y_score = dt_classifier.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



Advanced NN (Transformer)

In [9]:
```
!pip install transformers
```

```
Requirement already satisfied: transformers in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (4.45.2)
Requirement already satisfied: filelock in /Library/Frameworks/Python.frame
work/Versions/3.9/lib/python3.9/site-packages (from transformers) (3.3.2)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /Library/Fra
meworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from tra
nsformers) (0.26.1)
Requirement already satisfied: numpy>=1.17 in /Library/Frameworks/Python.fr
amework/Versions/3.9/lib/python3.9/site-packages (from transformers) (1.26.
4)
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.9/lib/python3.9/site-packages (from transformers) (2
4.1)
Requirement already satisfied: pyyaml>=5.1 in /Library/Frameworks/Python.fr
amework/Versions/3.9/lib/python3.9/site-packages (from transformers) (6.0.
2)
Requirement already satisfied: regex!=2019.12.17 in /Library/Frameworks/Pyt
hon.framework/Versions/3.9/lib/python3.9/site-packages (from transformers)
(2024.9.11)
Requirement already satisfied: requests in /Library/Frameworks/Python.frame
work/Versions/3.9/lib/python3.9/site-packages (from transformers) (2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from transformers)
(0.4.5)
Requirement already satisfied: tokenizers<0.21,>=0.20 in /Library/Framework
s/Python.framework/Versions/3.9/lib/python3.9/site-packages (from transform
ers) (0.20.1)
Requirement already satisfied: tqdm>=4.27 in /Library/Frameworks/Python.fra
mework/Versions/3.9/lib/python3.9/site-packages (from transformers) (4.66.
5)
Requirement already satisfied: fsspec>=2023.5.0 in /Library/Frameworks/Pyth
on.framework/Versions/3.9/lib/python3.9/site-packages (from huggingface-hub
<1.0,>=0.23.2->transformers) (2024.9.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /Library/Frame
works/Python.framework/Versions/3.9/lib/python3.9/site-packages (from huggi
ngface-hub<1.0,>=0.23.2->transformers) (4.12.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /Library/Framewo
rks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from request
s->transformers) (2.0.9)
Requirement already satisfied: idna<4,>=2.5 in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (from requests->transform
ers) (3.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from requests->tra
nsformers) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from requests->tra
nsformers) (2022.6.15)

[notice] A new release of pip is available: 23.1.2 -> 24.3.1
[notice] To update, run: pip install --upgrade pip
```

Prepare Your Data

In [10]:

```python
import pandas as pd
from sklearn.model_selection import train_test_split

# Load your dataset
data = pd.read_csv('filtered_phishing_data.csv')  # Replace with your actua

# Prepare features and target
X = data['URL']  # Use the URL column as input
y = data['label']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

Tokenize the URLs for Transformer

In [11]:
```
from transformers import BertTokenizer

# Load pre-trained tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenize the URLs
train_encodings = tokenizer(list(X_train), truncation=True, padding=True, ma
test_encodings = tokenizer(list(X_test), truncation=True, padding=True, max_
```

Create a Dataset Class

In [12]:
```
import torch

class URLDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: torch.tensor(val[idx]) for key, val in self.encodings.i
        item['labels'] = torch.tensor(self.labels[idx])
        return item

    def __len__(self):
        return len(self.labels)

train_dataset = URLDataset(train_encodings, y_train.tolist())
test_dataset = URLDataset(test_encodings, y_test.tolist())
```

Load the Transformer Model

In [13]:
```
from transformers import BertForSequenceClassification

# Load pre-trained model
model = BertForSequenceClassification.from_pretrained('bert-base-uncased', n
```

```
Some weights of BertForSequenceClassification were not initialized from the
model checkpoint at bert-base-uncased and are newly initialized: ['classifi
er.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to us
e it for predictions and inference.
```

In [14]:
```
!pip3 install 'transformers[torch]'
!pip install --upgrade 'accelerate>=0.26.0'
!pip3 show accelerate
```

```
Requirement already satisfied: transformers[torch] in /Library/Frameworks/P
ython.framework/Versions/3.9/lib/python3.9/site-packages (4.45.2)
Requirement already satisfied: filelock in /Library/Frameworks/Python.frame
work/Versions/3.9/lib/python3.9/site-packages (from transformers[torch])
(3.3.2)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /Library/Fra
meworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from tra
nsformers[torch]) (0.26.1)
Requirement already satisfied: numpy>=1.17 in /Library/Frameworks/Python.fr
amework/Versions/3.9/lib/python3.9/site-packages (from transformers[torch])
(1.26.4)
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.9/lib/python3.9/site-packages (from transformers[tor
ch]) (24.1)
Requirement already satisfied: pyyaml>=5.1 in /Library/Frameworks/Python.fr
amework/Versions/3.9/lib/python3.9/site-packages (from transformers[torch])
(6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /Library/Frameworks/Pyt
hon.framework/Versions/3.9/lib/python3.9/site-packages (from transformers[t
orch]) (2024.9.11)
Requirement already satisfied: requests in /Library/Frameworks/Python.frame
work/Versions/3.9/lib/python3.9/site-packages (from transformers[torch])
(2.32.3)
Requirement already satisfied: safetensors>=0.4.1 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from transformers
[torch]) (0.4.5)
Requirement already satisfied: tokenizers<0.21,>=0.20 in /Library/Framework
s/Python.framework/Versions/3.9/lib/python3.9/site-packages (from transform
ers[torch]) (0.20.1)
Requirement already satisfied: tqdm>=4.27 in /Library/Frameworks/Python.fra
mework/Versions/3.9/lib/python3.9/site-packages (from transformers[torch])
(4.66.5)
Requirement already satisfied: accelerate>=0.26.0 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from transformers
[torch]) (1.1.0)
Requirement already satisfied: torch in /Library/Frameworks/Python.framewor
k/Versions/3.9/lib/python3.9/site-packages (from transformers[torch]) (2.2.
2)
Requirement already satisfied: psutil in /Library/Frameworks/Python.framewo
rk/Versions/3.9/lib/python3.9/site-packages (from accelerate>=0.26.0->trans
formers[torch]) (5.9.0)
Requirement already satisfied: fsspec>=2023.5.0 in /Library/Frameworks/Pyth
on.framework/Versions/3.9/lib/python3.9/site-packages (from huggingface-hub
<1.0,>=0.23.2->transformers[torch]) (2024.9.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /Library/Frame
works/Python.framework/Versions/3.9/lib/python3.9/site-packages (from huggi
ngface-hub<1.0,>=0.23.2->transformers[torch]) (4.12.2)
Requirement already satisfied: sympy in /Library/Frameworks/Python.framewor
k/Versions/3.9/lib/python3.9/site-packages (from torch->transformers[torc
h]) (1.13.3)
Requirement already satisfied: networkx in /Library/Frameworks/Python.frame
work/Versions/3.9/lib/python3.9/site-packages (from torch->transformers[tor
ch]) (3.2.1)
Requirement already satisfied: jinja2 in /Library/Frameworks/Python.framewo
rk/Versions/3.9/lib/python3.9/site-packages (from torch->transformers[torc
h]) (3.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /Library/Framewo
rks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from request
s->transformers[torch]) (2.0.9)
Requirement already satisfied: idna<4,>=2.5 in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (from requests->transform
ers[torch]) (3.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from requests->tra
```

```
nsformers[torch]) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from requests->tra
nsformers[torch]) (2022.6.15)
Requirement already satisfied: MarkupSafe>=2.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.9/lib/python3.9/site-packages (from jinja2->torch->t
ransformers[torch]) (2.1.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from sympy->torch-
>transformers[torch]) (1.3.0)

[notice] A new release of pip is available: 23.1.2 -> 24.3.1
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: accelerate>=0.26.0 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (1.1.0)
Requirement already satisfied: numpy<3.0.0,>=1.17 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from accelerate>=
0.26.0) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.9/lib/python3.9/site-packages (from accelerate>=0.2
6.0) (24.1)
Requirement already satisfied: psutil in /Library/Frameworks/Python.framewo
rk/Versions/3.9/lib/python3.9/site-packages (from accelerate>=0.26.0) (5.9.
0)
Requirement already satisfied: pyyaml in /Library/Frameworks/Python.framewo
rk/Versions/3.9/lib/python3.9/site-packages (from accelerate>=0.26.0) (6.0.
2)
Requirement already satisfied: torch>=1.10.0 in /Library/Frameworks/Python.
framework/Versions/3.9/lib/python3.9/site-packages (from accelerate>=0.26.
0) (2.2.2)
Requirement already satisfied: huggingface-hub>=0.21.0 in /Library/Framewor
ks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from accelera
te>=0.26.0) (0.26.1)
Requirement already satisfied: safetensors>=0.4.3 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from accelerate>=
0.26.0) (0.4.5)
Requirement already satisfied: filelock in /Library/Frameworks/Python.frame
work/Versions/3.9/lib/python3.9/site-packages (from huggingface-hub>=0.21.0
->accelerate>=0.26.0) (3.3.2)
Requirement already satisfied: fsspec>=2023.5.0 in /Library/Frameworks/Pyth
on.framework/Versions/3.9/lib/python3.9/site-packages (from huggingface-hub
>=0.21.0->accelerate>=0.26.0) (2024.9.0)
Requirement already satisfied: requests in /Library/Frameworks/Python.frame
work/Versions/3.9/lib/python3.9/site-packages (from huggingface-hub>=0.21.0
->accelerate>=0.26.0) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (from huggingface-hub>=0.
21.0->accelerate>=0.26.0) (4.66.5)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /Library/Frame
works/Python.framework/Versions/3.9/lib/python3.9/site-packages (from huggi
ngface-hub>=0.21.0->accelerate>=0.26.0) (4.12.2)
Requirement already satisfied: sympy in /Library/Frameworks/Python.framewor
k/Versions/3.9/lib/python3.9/site-packages (from torch>=1.10.0->accelerate>
=0.26.0) (1.13.3)
Requirement already satisfied: networkx in /Library/Frameworks/Python.frame
work/Versions/3.9/lib/python3.9/site-packages (from torch>=1.10.0->accelera
te>=0.26.0) (3.2.1)
Requirement already satisfied: jinja2 in /Library/Frameworks/Python.framewo
rk/Versions/3.9/lib/python3.9/site-packages (from torch>=1.10.0->accelerate
>=0.26.0) (3.1.2)
Requirement already satisfied: MarkupSafe>=2.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.9/lib/python3.9/site-packages (from jinja2->torch>=
1.10.0->accelerate>=0.26.0) (2.1.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /Library/Framewo
```

```
rks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from request
s->huggingface-hub>=0.21.0->accelerate>=0.26.0) (2.0.9)
Requirement already satisfied: idna<4,>=2.5 in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (from requests->huggingfa
ce-hub>=0.21.0->accelerate>=0.26.0) (3.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from requests->hug
gingface-hub>=0.21.0->accelerate>=0.26.0) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from requests->hug
gingface-hub>=0.21.0->accelerate>=0.26.0) (2022.6.15)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from sympy->torch>
=1.10.0->accelerate>=0.26.0) (1.3.0)

[notice] A new release of pip is available: 23.1.2 -> 24.3.1
[notice] To update, run: pip install --upgrade pip
Name: accelerate
Version: 1.1.0
Summary: Accelerate
Home-page: https://github.com/huggingface/accelerate
Author: The HuggingFace team
Author-email: zach.mueller@huggingface.co
License: Apache
Location: /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/s
ite-packages
Requires: huggingface-hub, numpy, packaging, psutil, pyyaml, safetensors, t
orch
Required-by:
```

Define Training Arguments

In [17]:
```
!which python3
!pip install tf-keras
```

```
/Library/Frameworks/Python.framework/Versions/3.9/bin/python3
Collecting tf-keras
  Downloading tf_keras-2.18.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 1.5 MB/s eta 0:00:
0000:0100:01
INFO: pip is looking at multiple versions of tf-keras to determine which ve
rsion is compatible with other requirements. This could take a while.
  Downloading tf_keras-2.17.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 3.8 MB/s eta 0:00:
0000:0100:01
  Downloading tf_keras-2.16.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 3.6 MB/s eta 0:00:
0000:0100:01
Requirement already satisfied: tensorflow<2.17,>=2.16 in /Library/Framework
s/Python.framework/Versions/3.9/lib/python3.9/site-packages (from tf-keras)
(2.16.2)
Requirement already satisfied: absl-py>=1.0.0 in /Library/Frameworks/Pytho
n.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.17,
>=2.16->tf-keras) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in /Library/Frameworks/Pyt
hon.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.1
7,>=2.16->tf-keras) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /Library/Frameworks/
Python.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<
2.17,>=2.16->tf-keras) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /Libr
ary/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages (f
rom tensorflow<2.17,>=2.16->tf-keras) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /Library/Frameworks/P
ython.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<
2.17,>=2.16->tf-keras) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.17,>=
2.16->tf-keras) (3.12.1)
Requirement already satisfied: libclang>=13.0.0 in /Library/Frameworks/Pyth
on.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.1
7,>=2.16->tf-keras) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.3.1 in /Library/Frameworks/Pyth
on.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.1
7,>=2.16->tf-keras) (0.3.2)
Requirement already satisfied: opt-einsum>=2.3.2 in /Library/Frameworks/Pyt
hon.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.1
7,>=2.16->tf-keras) (3.4.0)
Requirement already satisfied: packaging in /Library/Frameworks/Python.fram
ework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.17,>=2.16
->tf-keras) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.
3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /Library/Frameworks/Python.framew
ork/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.17,>=2.16->
tf-keras) (4.25.5)
Requirement already satisfied: requests<3,>=2.21.0 in /Library/Frameworks/P
ython.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<
2.17,>=2.16->tf-keras) (2.32.3)
Requirement already satisfied: setuptools in /Library/Frameworks/Python.fra
mework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.17,>=2.1
6->tf-keras) (57.4.0)
Requirement already satisfied: six>=1.12.0 in /Library/Frameworks/Python.fr
amework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.17,>=2.
16->tf-keras) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /Library/Frameworks/Pyth
on.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.1
7,>=2.16->tf-keras) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /Library/Framewo
rks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from tensorf
```

```
low<2.17,>=2.16->tf-keras) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /Library/Frameworks/Python.
framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.17,>=
2.16->tf-keras) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /Library/Frameworks/P
ython.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<
2.17,>=2.16->tf-keras) (1.66.2)
Requirement already satisfied: tensorboard<2.17,>=2.16 in /Library/Framewor
ks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from tensorfl
ow<2.17,>=2.16->tf-keras) (2.16.2)
Requirement already satisfied: keras>=3.0.0 in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<2.17,>=
2.16->tf-keras) (3.5.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /Lib
rary/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages
(from tensorflow<2.17,>=2.16->tf-keras) (0.37.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /Library/Frameworks/
Python.framework/Versions/3.9/lib/python3.9/site-packages (from tensorflow<
2.17,>=2.16->tf-keras) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from astunparse>=
1.6.0->tensorflow<2.17,>=2.16->tf-keras) (0.44.0)
Requirement already satisfied: rich in /Library/Frameworks/Python.framewor
k/Versions/3.9/lib/python3.9/site-packages (from keras>=3.0.0->tensorflow<
2.17,>=2.16->tf-keras) (13.8.1)
Requirement already satisfied: namex in /Library/Frameworks/Python.framewor
k/Versions/3.9/lib/python3.9/site-packages (from keras>=3.0.0->tensorflow<
2.17,>=2.16->tf-keras) (0.0.8)
Requirement already satisfied: optree in /Library/Frameworks/Python.framewo
rk/Versions/3.9/lib/python3.9/site-packages (from keras>=3.0.0->tensorflow<
2.17,>=2.16->tf-keras) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /Library/Framewo
rks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from request
s<3,>=2.21.0->tensorflow<2.17,>=2.16->tf-keras) (2.0.9)
Requirement already satisfied: idna<4,>=2.5 in /Library/Frameworks/Python.f
ramework/Versions/3.9/lib/python3.9/site-packages (from requests<3,>=2.21.0
->tensorflow<2.17,>=2.16->tf-keras) (3.3)
Requirement already satisfied: urllib3<3,>=1.21.1 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from requests<3,>=
2.21.0->tensorflow<2.17,>=2.16->tf-keras) (1.26.7)
Requirement already satisfied: certifi>=2017.4.17 in /Library/Frameworks/Py
thon.framework/Versions/3.9/lib/python3.9/site-packages (from requests<3,>=
2.21.0->tensorflow<2.17,>=2.16->tf-keras) (2022.6.15)
Requirement already satisfied: markdown>=2.6.8 in /Library/Frameworks/Pytho
n.framework/Versions/3.9/lib/python3.9/site-packages (from tensorboard<2.1
7,>=2.16->tensorflow<2.17,>=2.16->tf-keras) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /Li
brary/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages
(from tensorboard<2.17,>=2.16->tensorflow<2.17,>=2.16->tf-keras) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /Library/Frameworks/Pytho
n.framework/Versions/3.9/lib/python3.9/site-packages (from tensorboard<2.1
7,>=2.16->tensorflow<2.17,>=2.16->tf-keras) (2.3.6)
Requirement already satisfied: importlib-metadata>=4.4 in /Library/Framewor
ks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from markdown
>=2.6.8->tensorboard<2.17,>=2.16->tensorflow<2.17,>=2.16->tf-keras) (4.12.
0)
Requirement already satisfied: MarkupSafe>=2.1.1 in /Library/Frameworks/Pyt
hon.framework/Versions/3.9/lib/python3.9/site-packages (from werkzeug>=1.0.
1->tensorboard<2.17,>=2.16->tensorflow<2.17,>=2.16->tf-keras) (2.1.1)
Requirement already satisfied: markdown-it-py>=2.2.0 in /Library/Framework
s/Python.framework/Versions/3.9/lib/python3.9/site-packages (from rich->ker
as>=3.0.0->tensorflow<2.17,>=2.16->tf-keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /Library/Framewor
ks/Python.framework/Versions/3.9/lib/python3.9/site-packages (from rich->ke
```

```
ras>=3.0.0->tensorflow<2.17,>=2.16->tf-keras) (2.18.0)
Requirement already satisfied: zipp>=0.5 in /Library/Frameworks/Python.fram
ework/Versions/3.9/lib/python3.9/site-packages (from importlib-metadata>=4.
4->markdown>=2.6.8->tensorboard<2.17,>=2.16->tensorflow<2.17,>=2.16->tf-ker
as) (3.8.1)
Requirement already satisfied: mdurl~=0.1 in /Library/Frameworks/Python.fra
mework/Versions/3.9/lib/python3.9/site-packages (from markdown-it-py>=2.2.0
->rich->keras>=3.0.0->tensorflow<2.17,>=2.16->tf-keras) (0.1.2)
Installing collected packages: tf-keras
Successfully installed tf-keras-2.16.0

[notice] A new release of pip is available: 23.1.2 -> 24.3.1
[notice] To update, run: pip install --upgrade pip
```

In [18]:

```python
from transformers import TrainingArguments

# Define training arguments
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=3,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=64,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    evaluation_strategy="epoch"
)
```

Train and Evaluate the Model

In [19]:

```python
from transformers import Trainer
from sklearn.metrics import accuracy_score, precision_recall_fscore_support

# Define a compute metrics function
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    precision, recall, f1, _ = precision_recall_fscore_support(labels, preds
    acc = accuracy_score(labels, preds)
    return {
        'accuracy': acc,
        'f1': f1,
        'precision': precision,
        'recall': recall
    }

# Initialize Trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    compute_metrics=compute_metrics
)

# Train the model
trainer.train()

# Evaluate the model
trainer.evaluate()
```

```
  0%|          | 10/7428 [00:42<7:17:47,  3.54s/it]
```

```
{'loss': 0.7727, 'grad_norm': 9.1824369430542, 'learning_rate': 1.0000000000000002e-06, 'epoch': 0.0}
  0%|          | 20/7428 [01:32<15:03:31,  7.32s/it]
{'loss': 0.7424, 'grad_norm': 11.944734573364258, 'learning_rate': 2.0000000000000003e-06, 'epoch': 0.01}
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Input In [19], in <cell line: 27>()
     18 trainer = Trainer(
     19     model=model,
     20     args=training_args,
   (...)
     23     compute_metrics=compute_metrics
     24 )
     26 # Train the model
---> 27 trainer.train()
     29 # Evaluate the model
     30 trainer.evaluate()

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/transformers/trainer.py:2052, in Trainer.train(self, resume_from_checkpoint, trial, ignore_keys_for_eval, **kwargs)
   2050         hf_hub_utils.enable_progress_bars()
   2051 else:
-> 2052     return inner_training_loop(
   2053         args=args,
   2054         resume_from_checkpoint=resume_from_checkpoint,
   2055         trial=trial,
   2056         ignore_keys_for_eval=ignore_keys_for_eval,
   2057     )

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/transformers/trainer.py:2452, in Trainer._inner_training_loop(self, batch_size, args, resume_from_checkpoint, trial, ignore_keys_for_eval)
   2448         grad_norm = _grad_norm
   2450 self.control = self.callback_handler.on_pre_optimizer_step(args, self.state, self.control)
-> 2452 self.optimizer.step()
   2454 self.control = self.callback_handler.on_optimizer_step(args, self.state, self.control)
   2456 optimizer_was_run = not self.accelerator.optimizer_step_was_skipped

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/accelerate/optimizer.py:171, in AcceleratedOptimizer.step(self, closure)
    169         self._accelerate_step_called = False
    170     else:
--> 171         self.optimizer.step(closure)
    172 if self.accelerator_state.distributed_type == DistributedType.XLA:
    173     self.gradient_state.is_xla_gradients_synced = False

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/torch/optim/lr_scheduler.py:75, in LRScheduler.__init__.<locals>.with_counter.<locals>.wrapper(*args, **kwargs)
     73 instance._step_count += 1
     74 wrapped = func.__get__(instance, cls)
---> 75 return wrapped(*args, **kwargs)

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/torch/optim/optimizer.py:385, in Optimizer.profile_hook_step.<locals>.wrapper(*args, **kwargs)
    380         else:
    381             raise RuntimeError(
    382                 f"{func} must return None or a tuple of (new_args, new_kwargs), but got {result}."
    383             )
--> 385 out = func(*args, **kwargs)
    386 self._optimizer_step_code()
    388 # call optimizer step post hooks
```

```
File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/torch/optim/optimizer.py:76, in _use_grad_for_differentiable.<locals>._use_grad(self, *args, **kwargs)
     74     torch.set_grad_enabled(self.defaults['differentiable'])
     75     torch._dynamo.graph_break()
---> 76     ret = func(self, *args, **kwargs)
     77 finally:
     78     torch._dynamo.graph_break()

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/torch/optim/adamw.py:187, in AdamW.step(self, closure)
    174     beta1, beta2 = group["betas"]
    176     has_complex = self._init_group(
    177         group,
    178         params_with_grad,
   (...)
    184         state_steps,
    185     )
--> 187     adamw(
    188         params_with_grad,
    189         grads,
    190         exp_avgs,
    191         exp_avg_sqs,
    192         max_exp_avg_sqs,
    193         state_steps,
    194         amsgrad=amsgrad,
    195         beta1=beta1,
    196         beta2=beta2,
    197         lr=group["lr"],
    198         weight_decay=group["weight_decay"],
    199         eps=group["eps"],
    200         maximize=group["maximize"],
    201         foreach=group["foreach"],
    202         capturable=group["capturable"],
    203         differentiable=group["differentiable"],
    204         fused=group["fused"],
    205         grad_scale=getattr(self, "grad_scale", None),
    206         found_inf=getattr(self, "found_inf", None),
    207         has_complex=has_complex,
    208     )
    210 return loss

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-packages/torch/optim/adamw.py:339, in adamw(params, grads, exp_avgs, exp_avg_sqs, max_exp_avg_sqs, state_steps, foreach, capturable, differentiable, fused, grad_scale, found_inf, has_complex, amsgrad, beta1, beta2, lr, weight_decay, eps, maximize)
    336 else:
    337     func = _single_tensor_adamw
--> 339 func(
    340     params,
    341     grads,
    342     exp_avgs,
    343     exp_avg_sqs,
    344     max_exp_avg_sqs,
    345     state_steps,
    346     amsgrad=amsgrad,
    347     beta1=beta1,
    348     beta2=beta2,
    349     lr=lr,
    350     weight_decay=weight_decay,
    351     eps=eps,
    352     maximize=maximize,
```

```
    353     capturable=capturable,
    354     differentiable=differentiable,
    355     grad_scale=grad_scale,
    356     found_inf=found_inf,
    357     has_complex=has_complex,
    358 )

File /Library/Frameworks/Python.framework/Versions/3.9/lib/python3.9/site-p
ackages/torch/optim/adamw.py:415, in _single_tensor_adamw(params, grads, ex
p_avgs, exp_avg_sqs, max_exp_avg_sqs, state_steps, grad_scale, found_inf, a
msgrad, beta1, beta2, lr, weight_decay, eps, maximize, capturable, differen
tiable, has_complex)
    412 step_t += 1
    414 # Perform stepweight decay
--> 415 param.mul_(1 - lr * weight_decay)
    417 # Decay the first and second moment running average coefficient
    418 exp_avg.lerp_(grad, 1 - beta1)

KeyboardInterrupt:
```

Accuracy Testing

In [ ]:
```python
# Make predictions
predictions = trainer.predict(test_dataset)
y_pred = predictions.predictions.argmax(-1)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Test Accuracy: {accuracy:.2f}")
```

AUC-ROC Analysis

In [ ]:
```python
from sklearn.metrics import roc_auc_score, roc_curve, auc
import matplotlib.pyplot as plt

# Calculate probabilities for the positive class
y_prob = predictions.predictions[:, 1]

# Calculate AUC-ROC score
roc_auc = roc_auc_score(y_test, y_prob)
print(f"AUC-ROC: {roc_auc:.2f}")

# Calculate ROC curve
fpr, tpr, _ = roc_curve(y_test, y_prob)

# Plot the ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', label=f'ROC curve (area = {roc_auc:.2
plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```