Detection of Phishing Websites using Semi Supervised Learning

November 30, 2023

1 Detection of Phishing Websites using Semi-Supervised Learning

1.1 Author

Joel Ruetas

1.2 Abstract

This study focuses on enhancing the detection and reporting of phishing emails and URLs through semi-supervised learning, a machine learning approach that leverages both a small set of labeled data and a larger pool of unlabeled data. This method is particularly effective for scenarios where obtaining large amounts of labeled data is challenging or costly.

1.3 Background

Phishing, a method of deceiving internet users into divulging sensitive information, predominantly occurs via deceptive emails and malicious websites. Statistics reveal a troubling trend: 96% of phishing attacks are conducted through emails, and a smaller portion through harmful websites. The severity and frequency of phishing and ransomware attacks have been on the rise:

- In 2020, nearly 7 million new phishing and scam pages emerged. The same year saw a dramatic increase in the average ransom payment, up by 171% from 2019, reaching an average of $312,493.
- The peak of average ransom payments in September 2020 was $233,817. Phishing attempts notably surged by 510% from January to February 2020.
- Concurrently, the pandemic fueled a 20.7% increase in online transactions in 2020, which unfortunately provided opportunities for fraudsters to mask their activities. This period saw ransomware attacks grow by over 40% and email malware attacks increase by 600% compared to 2019.

The escalating success of social engineering attacks, which are the primary cause of security breaches in corporate networks, underscores the urgency of improving phishing detection. Effective detection can mitigate not just the direct costs of ransomware attacks but also the extensive legal and reputational damages arising from data breaches.

Figure 1: Monthly number of phishing attacks reported more than doubled since 1 May 2020.From "Phishing Attacks Rose 61% in 2022, New Study Finds" by GlobeNewswire (2022).

1.4 Objective

The aim of this research is to apply semi-supervised learning techniques to significantly enhance the detection and reporting of phishing attempts. By doing so, it seeks to address a critical need in cybersecurity, contributing to the reduction of successful social engineering attacks.

1.5 Data

The primary dataset for this project comprises of webpages sourced from PhishTank. These webpages have undergone extensive preprocessing to facilitate efficient analysis and model training. Due to constraints in time and computing resources, the model is trained on a selected subset of this extensive dataset.

1.5.1 Preprocessing Steps:

The preprocessing of the webpages involves several critical steps:

Retrieval of HTML Content: Using BeautifulSoup, the raw HTML content of each webpage is retrieved.

Cleaning HTML Content: The HTML content is then cleaned to remove unnecessary elements such as HTML tags, script and style tags. This is accomplished using regular expressions.

Tokenization: The cleaned HTML content is tokenized into words or phrases utilizing the Natural Language Toolkit (NLTK).

Normalization: The tokens are normalized by converting them to lowercase, removing punctuation, and excluding stop words to ensure consistency and relevance in the analysis.

1.5.2 Feature Extraction:

In addition to standard preprocessing, the following features are extracted to aid in the identification of phishing websites:

- `title_clean`. The contents of the `<title>` element.
- `is_english`. Uses the `langdetect` module to assess whether the webpage is in English.
- `img_count`. The number of `<img>` elements.
- `has_form`. Specify whether the web page contains one or more `<form>` objects.
- `has_login_form`. Specify whether the webpage contains one or more `<form>` objects containing an `input` of type `password`.
- `has_js`. Specify whether the webpage contains one or more `<javascript>` object.
- `js_include_b64`. Specify whether or not the `<javascript>` objects contain base64-encoded strings.
- `nb_tokens`. Number of tokens remaining after the initial cleaning up of the parsing phase.
- `classification`. The binary classification of the web site (maliciaous or benign).
- `nb_title_entities`. The number of words in the title.
- `nb_text_entities`. The number of words in the body.
- `jpmorgan_chase`. The number of references to JP Morgan Chase in the body.
- `bank_of_america`. The number of references to Bank of America in the body.
- `wells_fargo`. The number of references to Wells Fargo in the body.
- `hsbc`. The number of references to HSBC in the body.
- `deutsche_bank`. The number of references to Deutsche Bank in the body.

- `mitsubishi_ufj`. The number of references to Mitsubishi UFJ in the body.
- `citibank`. The number of references to Citibank in the body.
- `rbc`. The number of references to RBC in the body.
- `paypal`. The number of references to PayPal in the body.
- `scotiabank`. The number of references to Scotiabank in the body.
- `apple`. The number of references to Apple in the body.
- `microsoft`. The number of references to Microsoft in the body.
- `amazon`. The number of references to Amazon in the body.
- `google`. The number of references to Google in the body.
- `samsung`. The number of references to Samsung in the body.
- `facebook`. The number of references to Facebook in the body.
- `steam`. The number of references to Steam in the body.
- `netflix`. The number of references to Netflix in the body.
- `ups`. The number of references to UPS in the body.
- `fedex`. The number of references to Fedex in the body.
- `dhl`. The number of references to DHL in the body.
- `tnt`. The number of references to TNT in the body.
- `usps`. The number of references to USPS in the body.
- `royal_mail`. The number of references to Royal Mail in the body.
- `purolator`. The number of references to Purolator in the body.
- `canada_post`. The number of references to Canada Post in the body.
- `youtube`. The number of references to YouTube in the body.
- `whatsapp`. The number of references to WhatsApp in the body.
- `facebook_messenger`. The number of references to Facebook Messenger in the body.
- `wechat`. The number of references to WeChat in the body.
- `instagram`. The number of references to Instagram in the body.
- `tiktok`. The number of references to TikTok in the body.
- `qq`. The number of references to QQ in the body.
- `weibo`. The number of references to JWeibo in the body.
- `linkedin`. The number of references to LinkedIn in the body.
- `twitter`. The number of references to Twitter in the body.

These features are meticulously stored as columns in the resulting CSV file, along with the tokens derived from the text.

1.5.3 Limitations:

It is important to note that due to the complexities of HTML parsing, some HTML may remain in the tokens. This preprocessing step, crucial for data quality, is not included in this notebook.

This comprehensive data section provides a clear and detailed overview of the dataset, preprocessing steps, feature extraction, and compilation process, setting a solid foundation for the subsequent analysis and model training.

1.5.4 References

1. Canadian Centre for Cyber Security. (2021). *Cyber Threat Bulletin: Ransomware Threat 2021*. Retrieved from https://www.cyber.gc.ca/en/guidance/cyber-threat-bulletin-ransomware-threat-2021

2. GlobeNewswire. (2022). *Phishing Attacks Rose 61% in 2022, New Study Finds.* Retrieved from https://www.globenewswire.com/en/news-release/2022/07/26/2485743/0/en/Phishing-Attacks-Rose-61-in-2022-New-Study-Finds.html

3. Spiceworks. (2020). *Average Ransomware Payout Touched $312k in 2020, Up from $115k in 2019.* Retrieved from https://www.spiceworks.com/it-security/vulnerability-management/news/average-ransomware-payout-touched-312k-in-2020-up-from-115k-in-2019

1.6 Import necessary libraries

[ ]:
```
!pip install --upgrade pip
!pip install pycaret nltk
!pip install yellowbrick
!pip install ydata-profiling --upgrade

import matplotlib.pyplot as plt
import nltk
import numpy as np
import pandas as pd
import pickle
import re
import seaborn as sns
import six
import spacy
import sqlite3
import sys
import time
import xgboost as xgboost

from bs4 import BeautifulSoup
from collections import Counter
from html import unescape
from matplotlib.pyplot import figure
from mlxtend.classifier import EnsembleVoteClassifier
from mlxtend.plotting import plot_decision_regions
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
from sklearn import model_selection
from sklearn import tree
from sklearn.datasets import make_classification
from sklearn.ensemble import AdaBoostClassifier, BaggingClassifier,␣
 ↪ExtraTreesClassifier, ExtraTreesRegressor, GradientBoostingClassifier,␣
 ↪RandomForestClassifier, RandomForestRegressor, StackingClassifier,␣
 ↪VotingClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

```
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV,␣
 ↪RidgeClassifier, SGDClassifier
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report, ConfusionMatrixDisplay, f1_score, log_loss
from sklearn.model_selection import cross_val_score, GridSearchCV,␣
 ↪RandomizedSearchCV, RepeatedStratifiedKFold, StratifiedKFold,␣
 ↪train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, LabelEncoder
from sklearn.svm import LinearSVC, NuSVC, SVC
from sklearn.tree import DecisionTreeRegressor
from wordcloud import WordCloud
from xgboost import XGBRegressor
from ydata_profiling import ProfileReport
from yellowbrick.classifier import ClassificationReport, ClassPredictionError,␣
 ↪ConfusionMatrix, PrecisionRecallCurve, ROCAUC
from yellowbrick.features import RadViz
from yellowbrick.model_selection import CVScores, LearningCurve, RFECV,␣
 ↪ValidationCurve
from yellowbrick.text import TSNEVisualizer

nltk.download('punkt')
nltk.download('stopwords')
sp = spacy.load('en_core_web_sm')

sys.modules['sklearn.externals.six'] = six
```

```
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-packages
(23.1.2)
Collecting pip
  Downloading pip-23.3.1-py3-none-any.whl (2.1 MB)
                           2.1/2.1 MB
20.7 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 23.1.2
    Uninstalling pip-23.1.2:
      Successfully uninstalled pip-23.1.2
Successfully installed pip-23.3.1
Collecting pycaret
  Downloading pycaret-3.2.0-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Collecting category-encoders>=2.4.0 (from pycaret)
  Downloading category_encoders-2.6.3-py2.py3-none-any.whl.metadata (8.0 kB)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-
```

```
packages (from pycaret) (2.2.1)
Collecting deprecation>=2.1.0 (from pycaret)
  Downloading deprecation-2.1.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: imbalanced-learn>=0.8.1 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (0.10.1)
Requirement already satisfied: importlib-metadata>=4.12.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (6.8.0)
Requirement already satisfied: ipython>=5.5.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (7.34.0)
Requirement already satisfied: ipywidgets>=7.6.5 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (7.7.1)
Requirement already satisfied: jinja2>=1.2 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (3.1.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (1.3.2)
Collecting kaleido>=0.2.1 (from pycaret)
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
                           79.9/79.9 MB
26.6 MB/s eta 0:00:00
Requirement already satisfied: lightgbm>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (4.1.0)
Requirement already satisfied: markupsafe>=2.0.1 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (2.1.3)
Collecting matplotlib<=3.6,>=3.3.0 (from pycaret)
  Downloading
matplotlib-3.6.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(11.8 MB)
                           11.8/11.8 MB
81.8 MB/s eta 0:00:00
Requirement already satisfied: nbformat>=4.2.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (5.9.2)
Requirement already satisfied: numba>=0.55.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (0.58.1)
Requirement already satisfied: numpy<1.27,>=1.21 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (1.23.5)
Requirement already satisfied: pandas<2.0.0,>=1.3.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (1.5.3)
Collecting plotly-resampler>=0.8.3.1 (from pycaret)
  Downloading plotly_resampler-0.9.1-py3-none-any.whl.metadata (12 kB)
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (5.15.0)
Collecting pmdarima!=1.8.1,<3.0.0,>=1.8.0 (from pycaret)
  Downloading pmdarima-2.0.4-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata
(7.8 kB)
Requirement already satisfied: psutil>=5.9.0 in /usr/local/lib/python3.10/dist-
packages (from pycaret) (5.9.5)
Collecting pyod>=1.0.8 (from pycaret)
```

```
  Downloading pyod-1.1.2.tar.gz (160 kB)
                             160.5/160.5
kB 12.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: requests>=2.27.1 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (2.31.0)
Collecting schemdraw==0.15 (from pycaret)
  Downloading schemdraw-0.15-py3-none-any.whl (106 kB)
                             106.8/106.8
kB 6.7 MB/s eta 0:00:00
Requirement already satisfied: scikit-learn<1.3.0,>=1.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (1.2.2)
Collecting scikit-plot>=0.3.7 (from pycaret)
  Downloading scikit_plot-0.3.7-py3-none-any.whl (33 kB)
Collecting scipy~=1.10.1 (from pycaret)
  Downloading
scipy-1.10.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (34.4
MB)
                             34.4/34.4 MB
46.8 MB/s eta 0:00:00
Collecting sktime!=0.17.1,!=0.17.2,!=0.18.0,<0.22.0,>=0.16.1 (from
pycaret)
  Downloading sktime-0.21.1-py3-none-any.whl.metadata (25 kB)
Requirement already satisfied: statsmodels>=0.12.1 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (0.14.0)
Collecting tbats>=1.1.3 (from pycaret)
  Downloading tbats-1.1.3-py3-none-any.whl (44 kB)
                             44.0/44.0 kB
3.0 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.62.0 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (4.66.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages
(from pycaret) (3.4.1)
Requirement already satisfied: yellowbrick>=1.4 in
/usr/local/lib/python3.10/dist-packages (from pycaret) (1.5)
Collecting wurlitzer (from pycaret)
  Downloading wurlitzer-3.0.3-py3-none-any.whl (7.3 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk) (8.1.7)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.10/dist-
packages (from category-encoders>=2.4.0->pycaret) (0.5.3)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from deprecation>=2.1.0->pycaret) (23.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from imbalanced-learn>=0.8.1->pycaret)
(3.2.0)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-
packages (from importlib-metadata>=4.12.0->pycaret) (3.17.0)
Requirement already satisfied: setuptools>=18.5 in
/usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (67.7.2)
Collecting jedi>=0.16 (from ipython>=5.5.0->pycaret)
  Downloading jedi-0.19.1-py2.py3-none-any.whl.metadata (22 kB)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (0.7.5)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (5.7.1)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (3.0.41)
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (2.16.1)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (0.2.0)
Requirement already satisfied: matplotlib-inline in
/usr/local/lib/python3.10/dist-packages (from ipython>=5.5.0->pycaret) (0.1.6)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-
packages (from ipython>=5.5.0->pycaret) (4.8.0)
Requirement already satisfied: ipykernel>=4.5.1 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret)
(5.5.6)
Requirement already satisfied: ipython-genutils~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret)
(0.2.0)
Requirement already satisfied: widgetsnbextension~=3.6.0 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret)
(3.6.6)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from ipywidgets>=7.6.5->pycaret)
(3.0.9)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.6,>=3.3.0->pycaret)
(1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<=3.6,>=3.3.0->pycaret) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.6,>=3.3.0->pycaret)
(4.44.3)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.6,>=3.3.0->pycaret)
(1.4.5)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
```

```
packages (from matplotlib<=3.6,>=3.3.0->pycaret) (9.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.6,>=3.3.0->pycaret)
(3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.6,>=3.3.0->pycaret)
(2.8.2)
Requirement already satisfied: fastjsonschema in /usr/local/lib/python3.10/dist-
packages (from nbformat>=4.2.0->pycaret) (2.19.0)
Requirement already satisfied: jsonschema>=2.6 in
/usr/local/lib/python3.10/dist-packages (from nbformat>=4.2.0->pycaret) (4.19.2)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.10/dist-
packages (from nbformat>=4.2.0->pycaret) (5.5.0)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in
/usr/local/lib/python3.10/dist-packages (from numba>=0.55.0->pycaret) (0.41.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas<2.0.0,>=1.3.0->pycaret) (2023.3.post1)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly>=5.0.0->pycaret) (8.2.3)
Collecting dash<3.0.0,>=2.11.0 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading dash-2.14.1-py3-none-any.whl.metadata (11 kB)
Collecting orjson<4.0.0,>=3.8.0 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading orjson-3.9.10-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (49 kB)
                           49.3/49.3 kB
3.4 MB/s eta 0:00:00
Collecting trace-updater>=0.0.8 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading trace_updater-0.0.9.1-py3-none-any.whl (185 kB)
                           185.2/185.2

kB 13.6 MB/s eta 0:00:00
Collecting tsdownsample==0.1.2 (from plotly-resampler>=0.8.3.1->pycaret)
  Downloading
tsdownsample-0.1.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(2.3 MB)
                           2.3/2.3 MB
72.7 MB/s eta 0:00:00
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in
/usr/local/lib/python3.10/dist-packages (from
pmdarima!=1.8.1,<3.0.0,>=1.8.0->pycaret) (3.0.5)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-
packages (from pmdarima!=1.8.1,<3.0.0,>=1.8.0->pycaret) (2.0.7)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from pyod>=1.0.8->pycaret) (1.16.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.27.1->pycaret) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests>=2.27.1->pycaret) (3.4)
```

```
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests>=2.27.1->pycaret)
(2023.7.22)
Collecting deprecated>=1.2.13 (from
sktime!=0.17.1,!=0.17.2,!=0.18.0,<0.22.0,>=0.16.1->pycaret)
  Downloading Deprecated-1.2.14-py2.py3-none-any.whl.metadata (5.4 kB)
Collecting scikit-base<0.6.0 (from
sktime!=0.17.1,!=0.17.2,!=0.18.0,<0.22.0,>=0.16.1->pycaret)
  Downloading scikit_base-0.5.2-py3-none-any.whl.metadata (8.6 kB)
Requirement already satisfied: Flask<3.1,>=1.0.4 in
/usr/local/lib/python3.10/dist-packages (from dash<3.0.0,>=2.11.0->plotly-
resampler>=0.8.3.1->pycaret) (2.2.5)
Requirement already satisfied: Werkzeug<3.1 in /usr/local/lib/python3.10/dist-
packages (from dash<3.0.0,>=2.11.0->plotly-resampler>=0.8.3.1->pycaret) (3.0.1)
Collecting dash-html-components==2.0.0 (from dash<3.0.0,>=2.11.0->plotly-
resampler>=0.8.3.1->pycaret)
  Downloading dash_html_components-2.0.0-py3-none-any.whl (4.1 kB)
Collecting dash-core-components==2.0.0 (from dash<3.0.0,>=2.11.0->plotly-
resampler>=0.8.3.1->pycaret)
  Downloading dash_core_components-2.0.0-py3-none-any.whl (3.8 kB)
Collecting dash-table==5.0.0 (from dash<3.0.0,>=2.11.0->plotly-
resampler>=0.8.3.1->pycaret)
  Downloading dash_table-5.0.0-py3-none-any.whl (3.9 kB)
Requirement already satisfied: typing-extensions>=4.1.1 in
/usr/local/lib/python3.10/dist-packages (from dash<3.0.0,>=2.11.0->plotly-
resampler>=0.8.3.1->pycaret) (4.5.0)
Collecting retrying (from dash<3.0.0,>=2.11.0->plotly-
resampler>=0.8.3.1->pycaret)
  Downloading retrying-1.3.4-py3-none-any.whl (11 kB)
Collecting ansi2html (from dash<3.0.0,>=2.11.0->plotly-
resampler>=0.8.3.1->pycaret)
  Downloading ansi2html-1.8.0-py3-none-any.whl (16 kB)
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.10/dist-
packages (from dash<3.0.0,>=2.11.0->plotly-resampler>=0.8.3.1->pycaret) (1.5.8)
Requirement already satisfied: wrapt<2,>=1.10 in /usr/local/lib/python3.10/dist-
packages (from
deprecated>=1.2.13->sktime!=0.17.1,!=0.17.2,!=0.18.0,<0.22.0,>=0.16.1->pycaret)
(1.14.1)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.10/dist-
packages (from ipykernel>=4.5.1->ipywidgets>=7.6.5->pycaret) (6.1.12)
Requirement already satisfied: tornado>=4.2 in /usr/local/lib/python3.10/dist-
packages (from ipykernel>=4.5.1->ipywidgets>=7.6.5->pycaret) (6.3.2)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in
/usr/local/lib/python3.10/dist-packages (from
jedi>=0.16->ipython>=5.5.0->pycaret) (0.8.3)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=4.2.0->pycaret) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
```

```
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=2.6->nbformat>=4.2.0->pycaret) (2023.11.1)
Requirement already satisfied: referencing>=0.28.4 in
/usr/local/lib/python3.10/dist-packages (from
jsonschema>=2.6->nbformat>=4.2.0->pycaret) (0.31.0)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-
packages (from jsonschema>=2.6->nbformat>=4.2.0->pycaret) (0.13.0)
Requirement already satisfied: ptyprocess>=0.5 in
/usr/local/lib/python3.10/dist-packages (from
pexpect>4.3->ipython>=5.5.0->pycaret) (0.7.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-
packages (from prompt-
toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=5.5.0->pycaret) (0.2.10)
Requirement already satisfied: notebook>=4.4.1 in
/usr/local/lib/python3.10/dist-packages (from
widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (6.5.5)
Requirement already satisfied: platformdirs>=2.5 in
/usr/local/lib/python3.10/dist-packages (from jupyter-
core->nbformat>=4.2.0->pycaret) (4.0.0)
Requirement already satisfied: itsdangerous>=2.0 in
/usr/local/lib/python3.10/dist-packages (from
Flask<3.1,>=1.0.4->dash<3.0.0,>=2.11.0->plotly-resampler>=0.8.3.1->pycaret)
(2.1.2)
Requirement already satisfied: pyzmq<25,>=17 in /usr/local/lib/python3.10/dist-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (23.2.1)
Requirement already satisfied: argon2-cffi in /usr/local/lib/python3.10/dist-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (23.1.0)
Requirement already satisfied: nbconvert>=5 in /usr/local/lib/python3.10/dist-
packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (6.5.4)
Requirement already satisfied: Send2Trash>=1.8.0 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.8.2)
Requirement already satisfied: terminado>=0.8.3 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.18.0)
Requirement already satisfied: prometheus-client in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.18.0)
Requirement already satisfied: nbclassic>=0.4.7 in
/usr/local/lib/python3.10/dist-packages (from
notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.0.0)
Requirement already satisfied: jupyter-server>=1.8 in
/usr/local/lib/python3.10/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.24.0)
Requirement already satisfied: notebook-shim>=0.2.3 in
```

```
/usr/local/lib/python3.10/dist-packages (from nbclassic>=0.4.7->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.2.3)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages
(from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (4.9.3)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (4.11.2)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (6.1.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.10/dist-
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (0.7.1)
Requirement already satisfied: entrypoints>=0.2.2 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.4)
Requirement already satisfied: jupyterlab-pygments in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.2.2)
Requirement already satisfied: mistune<2,>=0.8.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.8.4)
Requirement already satisfied: nbclient>=0.5.0 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (0.9.0)
Requirement already satisfied: pandocfilters>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from nbconvert>=5->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.5.0)
Requirement already satisfied: tinycss2 in /usr/local/lib/python3.10/dist-
packages (from nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (1.2.1)
Requirement already satisfied: argon2-cffi-bindings in
/usr/local/lib/python3.10/dist-packages (from argon2-cffi->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (21.2.0)
Requirement already satisfied: anyio<4,>=3.1.0 in
/usr/local/lib/python3.10/dist-packages (from jupyter-server>=1.8-
>nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (3.7.1)
Requirement already satisfied: websocket-client in
/usr/local/lib/python3.10/dist-packages (from jupyter-server>=1.8-
>nbclassic>=0.4.7->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (1.6.4)
Requirement already satisfied: cffi>=1.0.1 in /usr/local/lib/python3.10/dist-
packages (from argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.16.0)
Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-
packages (from beautifulsoup4->nbconvert>=5->notebook>=4.4.1-
```

```
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (2.5)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-
packages (from bleach->nbconvert>=5->notebook>=4.4.1->widgetsnbextension~=3.6.0-
>ipywidgets>=7.6.5->pycaret) (0.5.1)
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.10/dist-
packages (from anyio<4,>=3.1.0->jupyter-server>=1.8->nbclassic>=0.4.7-
>notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.3.0)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-
packages (from anyio<4,>=3.1.0->jupyter-server>=1.8->nbclassic>=0.4.7-
>notebook>=4.4.1->widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (1.1.3)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-
packages (from cffi>=1.0.1->argon2-cffi-bindings->argon2-cffi->notebook>=4.4.1-
>widgetsnbextension~=3.6.0->ipywidgets>=7.6.5->pycaret) (2.21)
Downloading pycaret-3.2.0-py3-none-any.whl (484 kB)
                         484.7/484.7 kB
27.3 MB/s eta 0:00:00
Downloading category_encoders-2.6.3-py2.py3-none-any.whl (81 kB)
                         81.9/81.9 kB
7.0 MB/s eta 0:00:00
Downloading plotly_resampler-0.9.1-py3-none-any.whl (73 kB)
                         73.4/73.4 kB
5.8 MB/s eta 0:00:00
Downloading pmdarima-2.0.4-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
                         2.1/2.1 MB
69.8 MB/s eta 0:00:00
Downloading sktime-0.21.1-py3-none-any.whl (17.1 MB)
                         17.1/17.1 MB
66.9 MB/s eta 0:00:00
Downloading dash-2.14.1-py3-none-any.whl (10.4 MB)
                         10.4/10.4 MB
95.7 MB/s eta 0:00:00
Downloading Deprecated-1.2.14-py2.py3-none-any.whl (9.6 kB)
Downloading jedi-0.19.1-py2.py3-none-any.whl (1.6 MB)
                         1.6/1.6 MB
61.9 MB/s eta 0:00:00
Downloading
orjson-3.9.10-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (138
kB)
                         138.7/138.7 kB
11.0 MB/s eta 0:00:00
Downloading scikit_base-0.5.2-py3-none-any.whl (118 kB)
                         119.0/119.0 kB
9.2 MB/s eta 0:00:00
Building wheels for collected packages: pyod
  Building wheel for pyod (setup.py) … done
  Created wheel for pyod: filename=pyod-1.1.2-py3-none-any.whl size=190289
sha256=dca1fa09bc3fabdb0d5c001657e61190cd668af071498fabb7cb416b24d59acd
```

```
  Stored in directory: /root/.cache/pip/wheels/81/1b/61/aa85b78c3c0c8871f4231e3f
4a03bb23cecb7db829498380ee
Successfully built pyod
Installing collected packages: trace-updater, kaleido, dash-table, dash-html-
components, dash-core-components, wurlitzer, tsdownsample, scipy, scikit-base,
schemdraw, retrying, orjson, jedi, deprecation, deprecated, ansi2html,
matplotlib, sktime, scikit-plot, pyod, dash, pmdarima, plotly-resampler,
category-encoders, tbats, pycaret
  Attempting uninstall: scipy
    Found existing installation: scipy 1.11.3
    Uninstalling scipy-1.11.3:
      Successfully uninstalled scipy-1.11.3
  Attempting uninstall: matplotlib
    Found existing installation: matplotlib 3.7.1
    Uninstalling matplotlib-3.7.1:
      Successfully uninstalled matplotlib-3.7.1
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
lida 0.0.10 requires uvicorn, which is not installed.
Successfully installed ansi2html-1.8.0 category-encoders-2.6.3 dash-2.14.1
dash-core-components-2.0.0 dash-html-components-2.0.0 dash-table-5.0.0
deprecated-1.2.14 deprecation-2.1.0 jedi-0.19.1 kaleido-0.2.1 matplotlib-3.6.0
orjson-3.9.10 plotly-resampler-0.9.1 pmdarima-2.0.4 pycaret-3.2.0 pyod-1.1.2
retrying-1.3.4 schemdraw-0.15 scikit-base-0.5.2 scikit-plot-0.3.7 scipy-1.10.1
sktime-0.21.1 tbats-1.1.3 trace-updater-0.0.9.1 tsdownsample-0.1.2
wurlitzer-3.0.3
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommended to
use a virtual environment instead: https://pip.pypa.io/warnings/venv


Requirement already satisfied: yellowbrick in /usr/local/lib/python3.10/dist-
packages (1.5)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.2 in
/usr/local/lib/python3.10/dist-packages (from yellowbrick) (3.6.0)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-
packages (from yellowbrick) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from yellowbrick) (1.2.2)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-
```

```
packages (from yellowbrick) (1.23.5)
Requirement already satisfied: cycler>=0.10.0 in /usr/local/lib/python3.10/dist-
packages (from yellowbrick) (0.12.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.2.0)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (4.44.3)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (23.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.0.0,>=2.0.2->yellowbrick) (9.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from
matplotlib!=3.0.0,>=2.0.2->yellowbrick) (2.8.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-
packages (from scikit-learn>=1.0.0->yellowbrick) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->yellowbrick)
(3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib!=3.0.0,>=2.0.2->yellowbrick)
(1.16.0)
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommended to
use a virtual environment instead: https://pip.pypa.io/warnings/venv

Collecting ydata-profiling
  Downloading ydata_profiling-4.6.2-py2.py3-none-any.whl.metadata (20 kB)
Requirement already satisfied: scipy<1.12,>=1.4.1 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (1.10.1)
Requirement already satisfied: pandas!=1.4.0,<2.1,>1.1 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (1.5.3)
Requirement already satisfied: matplotlib<=3.7.3,>=3.2 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (3.6.0)
Collecting pydantic>=2 (from ydata-profiling)
  Downloading pydantic-2.5.2-py3-none-any.whl.metadata (65 kB)
                           65.2/65.2 kB
2.2 MB/s eta 0:00:00
```

```
Requirement already satisfied: PyYAML<6.1,>=5.0.0 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (6.0.1)
Requirement already satisfied: jinja2<3.2,>=2.11.1 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (3.1.2)
Collecting visions==0.7.5 (from visions[type_image_path]==0.7.5->ydata-
profiling)
  Downloading visions-0.7.5-py3-none-any.whl (102 kB)
                             102.7/102.7
kB 4.7 MB/s eta 0:00:00
Requirement already satisfied: numpy<1.26,>=1.16.0 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (1.23.5)
Collecting htmlmin==0.1.12 (from ydata-profiling)
  Downloading htmlmin-0.1.12.tar.gz (19 kB)
  Preparing metadata (setup.py) … done
Collecting phik<0.13,>=0.11.1 (from ydata-profiling)
  Downloading
phik-0.12.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (679 kB)
                             679.5/679.5
kB 18.4 MB/s eta 0:00:00
Requirement already satisfied: requests<3,>=2.24.0 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (2.31.0)
Requirement already satisfied: tqdm<5,>=4.48.2 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (4.66.1)
Requirement already satisfied: seaborn<0.13,>=0.10.1 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (0.12.2)
Collecting multimethod<2,>=1.4 (from ydata-profiling)
  Downloading multimethod-1.10-py3-none-any.whl.metadata (8.2 kB)
Requirement already satisfied: statsmodels<1,>=0.13.2 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (0.14.0)
Collecting typeguard<5,>=4.1.2 (from ydata-profiling)
  Downloading typeguard-4.1.5-py3-none-any.whl.metadata (3.7 kB)
Collecting imagehash==4.3.1 (from ydata-profiling)
  Downloading ImageHash-4.3.1-py2.py3-none-any.whl (296 kB)
                             296.5/296.5
kB 20.7 MB/s eta 0:00:00
Requirement already satisfied: wordcloud>=1.9.1 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (1.9.2)
Collecting dacite>=1.8 (from ydata-profiling)
  Downloading dacite-1.8.1-py3-none-any.whl.metadata (15 kB)
Requirement already satisfied: numba<0.59.0,>=0.56.0 in
/usr/local/lib/python3.10/dist-packages (from ydata-profiling) (0.58.1)
Requirement already satisfied: PyWavelets in /usr/local/lib/python3.10/dist-
packages (from imagehash==4.3.1->ydata-profiling) (1.4.1)
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages
(from imagehash==4.3.1->ydata-profiling) (9.4.0)
Requirement already satisfied: attrs>=19.3.0 in /usr/local/lib/python3.10/dist-
```

```
packages (from visions==0.7.5->visions[type_image_path]==0.7.5->ydata-profiling)
(23.1.0)
Requirement already satisfied: networkx>=2.4 in /usr/local/lib/python3.10/dist-
packages (from visions==0.7.5->visions[type_image_path]==0.7.5->ydata-profiling)
(3.2.1)
Collecting tangled-up-in-unicode>=0.0.4 (from
visions==0.7.5->visions[type_image_path]==0.7.5->ydata-profiling)
  Downloading tangled_up_in_unicode-0.2.0-py3-none-any.whl (4.7 MB)
                           4.7/4.7 MB
51.3 MB/s eta 0:00:00
WARNING: visions 0.7.5 does not provide the extra 'type-image-
path'
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2<3.2,>=2.11.1->ydata-
profiling) (2.1.3)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.7.3,>=3.2->ydata-
profiling) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib<=3.7.3,>=3.2->ydata-profiling) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.7.3,>=3.2->ydata-
profiling) (4.44.3)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.7.3,>=3.2->ydata-
profiling) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.7.3,>=3.2->ydata-
profiling) (23.2)
Requirement already satisfied: pyparsing>=2.2.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.7.3,>=3.2->ydata-
profiling) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib<=3.7.3,>=3.2->ydata-
profiling) (2.8.2)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in
/usr/local/lib/python3.10/dist-packages (from numba<0.59.0,>=0.56.0->ydata-
profiling) (0.41.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas!=1.4.0,<2.1,>1.1->ydata-profiling) (2023.3.post1)
Requirement already satisfied: joblib>=0.14.1 in /usr/local/lib/python3.10/dist-
packages (from phik<0.13,>=0.11.1->ydata-profiling) (1.3.2)
Collecting annotated-types>=0.4.0 (from pydantic>=2->ydata-profiling)
  Downloading annotated_types-0.6.0-py3-none-any.whl.metadata (12 kB)
Collecting pydantic-core==2.14.5 (from pydantic>=2->ydata-profiling)
  Downloading pydantic_core-2.14.5-cp310-cp310-
```

```
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.5 kB)
Collecting typing-extensions>=4.6.1 (from pydantic>=2->ydata-profiling)
  Downloading typing_extensions-4.8.0-py3-none-any.whl.metadata (3.0 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-
profiling) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests<3,>=2.24.0->ydata-profiling) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-
profiling) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.24.0->ydata-
profiling) (2023.7.22)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-
packages (from statsmodels<1,>=0.13.2->ydata-profiling) (0.5.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from patsy>=0.5.2->statsmodels<1,>=0.13.2->ydata-profiling) (1.16.0)
Downloading ydata_profiling-4.6.2-py2.py3-none-any.whl (357 kB)
                         357.5/357.5 kB
25.2 MB/s eta 0:00:00
Downloading dacite-1.8.1-py3-none-any.whl (14 kB)
Downloading multimethod-1.10-py3-none-any.whl (9.9 kB)
Downloading pydantic-2.5.2-py3-none-any.whl (381 kB)
                         381.9/381.9 kB
27.1 MB/s eta 0:00:00
Downloading
pydantic_core-2.14.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(2.1 MB)
                         2.1/2.1 MB
78.2 MB/s eta 0:00:00
Downloading typeguard-4.1.5-py3-none-any.whl (34 kB)
Downloading annotated_types-0.6.0-py3-none-any.whl (12 kB)
Downloading typing_extensions-4.8.0-py3-none-any.whl (31 kB)
Building wheels for collected packages: htmlmin
  Building wheel for htmlmin (setup.py) … done
  Created wheel for htmlmin: filename=htmlmin-0.1.12-py3-none-any.whl size=27082
sha256=d52bc2467874fbe68c9693528d0b99f30a480dfee55f8bb732c1ee4462b90ec6
  Stored in directory: /root/.cache/pip/wheels/dd/91/29/a79cecb328d01739e64017b6
fb9a1ab9d8cb1853098ec5966d
Successfully built htmlmin
Installing collected packages: htmlmin, typing-extensions, tangled-up-in-
unicode, multimethod, dacite, annotated-types, typeguard, pydantic-core,
imagehash, visions, pydantic, phik, ydata-profiling
  Attempting uninstall: typing-extensions
    Found existing installation: typing_extensions 4.5.0
    Uninstalling typing_extensions-4.5.0:
      Successfully uninstalled typing_extensions-4.5.0
```

```
  Attempting uninstall: pydantic
    Found existing installation: pydantic 1.10.13
    Uninstalling pydantic-1.10.13:
      Successfully uninstalled pydantic-1.10.13
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
lida 0.0.10 requires fastapi, which is not installed.
lida 0.0.10 requires python-multipart, which is not installed.
lida 0.0.10 requires uvicorn, which is not installed.
llmx 0.0.15a0 requires cohere, which is not installed.
llmx 0.0.15a0 requires openai, which is not installed.
llmx 0.0.15a0 requires tiktoken, which is not installed.
tensorflow-probability 0.22.0 requires typing-extensions<4.6.0, but you have
typing-extensions 4.8.0 which is incompatible.
Successfully installed annotated-types-0.6.0 dacite-1.8.1 htmlmin-0.1.12
imagehash-4.3.1 multimethod-1.10 phik-0.12.3 pydantic-2.5.2 pydantic-core-2.14.5
tangled-up-in-unicode-0.2.0 typeguard-4.1.5 typing-extensions-4.8.0
visions-0.7.5 ydata-profiling-4.6.2
WARNING: Running pip as the 'root' user can result in broken permissions
and conflicting behaviour with the system package manager. It is recommended to
use a virtual environment instead: https://pip.pypa.io/warnings/venv
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Functions

[ ]:
```python
# Remove column from df.
def drop_column(label):
  if label in df:
    # 'labels' specifies the column name or a list of column names to be␣
↪dropped.
    # 'axis = 1' indicates that the operation should be performed on columns␣
↪(axis = 0 is for rows).
    # 'inplace = True' means that the change will be applied directly to 'df',␣
↪modifying the original DataFrame without returning a new one.
    df.drop(label, axis = 1, inplace = True)
```

[ ]:
```
# Replace occurrences of a substring with a new substring in a specified column.
def replace_string(label, old, new):
  # The function takes three parameters:
  # label: The name of the column in the DataFrame where the replacement should␣
 ↪occur.
  # old: The substring that needs to be replaced.
  # new: The substring that will replace the 'old' substring.
  df[label] = df[label].str.replace(old, new, regex = True)
```

[ ]:
```
# This function is designed to display a bar chart of value counts for a␣
 ↪specified column.
def display_value_counts(label, title, x_label, y_label):
  value_counts = df[label].value_counts()

  # Plot the histogram
  value_counts.plot(kind = 'bar')
  plt.title(title, fontsize = 18)
  plt.xlabel(x_label)
  plt.ylabel(y_label)
```

[ ]:
```
# Remove URL strings from a given text.
def remove_urls(x):
    cleaned_string = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '',␣
 ↪str(x), flags = re.MULTILINE)
    return cleaned_string
```

[ ]:
```
# Convert HTML entities in a string to their corresponding characters
# and then extract the plain text content from any HTML or XML markup.
def unescape_stuff(x):
    soup = BeautifulSoup(unescape(x), 'lxml')
    return soup.text
```

[ ]:
```
# Clean the text by removing unnecessary punctuation, special characters, and␣
 ↪symbols.
def remove_symbols(x):
    cleaned_string = re.sub(r"[^a-zA-Z0-9]+", ' ', x)
    return cleaned_string
```

[ ]:
```
# Remove multiple spaces
def unify_whitespaces(x):
    cleaned_string = re.sub(' +', ' ', x)
    return cleaned_string
```

[ ]:
```
# Add custom stop words here:
custom_stop_words = ["com", "ca", "go", "td", "tr", "px", "co", "uv", "ru",
                     "mx", "also", "use", "abc", "wo", "may", "oo",␣
 ↪"javascript",
```

```
                    "www", "html", "id", "class", "http", "https"]

cachedStopWords = sp.Defaults.stop_words
cachedStopWords = [x.lower() for x in cachedStopWords]
cachedStopWords.extend(list(stopwords.words('english')))
cachedStopWords.extend(list(custom_stop_words))
cachedStopWords = list(set(cachedStopWords))

def remove_stopwords(x):

    meaningful_words = []
    my_list = x

    tokenized_my_list = word_tokenize(my_list)
    meaningful_words = [w for w in tokenized_my_list if not w in␣
 ↪cachedStopWords]

    return " ".join(meaningful_words)
```

[ ]:
```
def get_ngrams(text, n = 2):
    text = str(text)
    n_grams = ngrams(text.split(), n)
    returnVal = []

    try:
        for grams in n_grams:
            returnVal.append('_'.join(grams))
    except(RuntimeError):
        pass

    return ' '.join(returnVal).strip()
```

[ ]:
```
def my_tokenizer(text):
    return text.split() if text != None else []
```

[ ]:
```
def display_most_common_ngrams(title, counter):
  print(title)

  # convert list of tuples into data frame
  most_common_words_df = pd.DataFrame.from_records(counter.most_common(20),␣
 ↪columns = ['token', 'count'])

  # create bar plot
  most_common_words_df.plot(kind = 'barh', x = 'token');
```

[ ]:
```
def get_ngram_string(label):
  ngram_string_list = df_text_clean[label].tolist()
  ngram_string = ' '.join(ngram_string_list)
  return ngram_string
```

[ ]:
```
def get_wordcloud(ngram_string):
  return WordCloud(width = 2000,
                   height = 1334,
                   random_state = 1,
                   background_color = 'black',
                   colormap = 'Pastel1',
                   max_words = 200,
                   collocations = False,
                   normalize_plurals = False).generate(ngram_string)
```

[ ]:
```
# Define a function to plot word cloud
def plot_wordcloud(title, wordcloud):
    fig = plt.figure(figsize = (12, 8), dpi = 80)
    plt.title(title, fontsize = 18)
    plt.tight_layout(pad=0)
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.box(False)
    plt.show()
    plt.close()
```

[ ]:
```
def handle_non_numerical_data(new_df):
    columns = new_df.columns.values
    for column in columns:
        text_digit_vals = {}
        def convert_to_int(val):
            return text_digit_vals[val]

        if new_df[column].dtype != np.int64 and new_df[column].dtype != np.
 ↪float64:
            column_contents = new_df[column].values.tolist()
            unique_elements = set(column_contents)

            x = 0

            for unique in unique_elements:
                if unique not in text_digit_vals:
                    text_digit_vals[unique] = x
                    x += 1

            new_df[column] = list(map(convert_to_int, new_df[column]))
```

```
    return new_df
```

[ ]:
```
def display_accuracy(model):
  # Train the model
  t0 = time.time()
  model.fit(X_train, y_train)
  print("Training time:", time.time() - t0)

  y_pred = model.predict(X_test)

  score = accuracy_score(y_pred, y_test)
  print('Test accuracy : {:.2f}%'.format(score * 100))
```

[ ]:
```
def display_classification_report(model):
  # Create the visualizer, score and show it
  visualizer = ClassificationReport(model)
  visualizer.score(X_test, y_test)       # Evaluate the model on the test data
  visualizer.size = (1080, 720)
  visualizer.show()                      # Finalize and render the figure
```

[ ]:
```
def display_confusion_matrix(model):
  # Create the visualizer, fit, score, and show it
  visualizer = ConfusionMatrix(
      model,
      classes = model.classes_
  )

  visualizer.fit(X_train, y_train)     # Fit the training data to the visualizer
  visualizer.score(X_test, y_test)     # Evaluate the model on the test data
  visualizer.size = (1080, 720)
  visualizer.show()                    # Finalize and render the figure
```

[ ]:
```
def display_roc_curve(model):
  # Create the visualizer, fit, score, and show it
  visualizer = ROCAUC(
      model,
      classes = model.classes_
  )

  visualizer.fit(X_train, y_train)      # Fit the training data to the␣
 ↪visualizer
  visualizer.score(X_test, y_test)      # Evaluate the model on the test data
  visualizer.size = (1080, 720)
  visualizer.show()                       # Finalize and render the figure
```

[ ]:
```
def display_precision_recall_curve(model):
  # Create the visualizer, fit, score, and show it
```

```
  visualizer = PrecisionRecallCurve(
      model,
      per_class=True,
      cmap="Set1"
  )
  visualizer.fit(X_train, y_train)    # Fit the training data to the visualizer
  visualizer.score(X_test, y_test)    # Evaluate the model on the test data
  visualizer.size = (1080, 720)
  visualizer.show()                   # Finalize and render the figure
```

[ ]:
```
def display_learning_curve(model):
  # Create a cross-validation strategy
  cv = StratifiedKFold(n_splits = 12)

  # Set the sizes
  sizes = np.linspace(0.3, 1.0, 10)

  # Create the visualizer, fit and show it
  visualizer = LearningCurve(
      model,
      cv = cv,
      scoring = 'f1_weighted',
      train_sizes = sizes,
      n_jobs = 4
  )

  visualizer.fit(X_train, y_train)      # Fit the data to the visualizer
  visualizer.size = (1080, 720)
  visualizer.show()                     # Finalize and render the figure
```

[ ]:
```
def display_validation_curve(model):
  visualizer = ValidationCurve(
      DecisionTreeRegressor(), param_name = "max_depth",
      param_range = np.arange(1, 11), cv = 10, scoring = "r2"
  )

  visualizer.fit(X_train, y_train)      # Fit the data to the visualizer
  visualizer.size = (1080, 720)
  visualizer.show()                     # Finalize and render the figure
```

[ ]:
```
def display_feature_importance(model):
  # get importance
  importance = model.feature_importances_

  # summarize feature importance
  print('Summary of Feature Importance:')
  for i, v in enumerate(importance):
```

```
    print(model.feature_names_in_[i] + ': Score: %.5f' % (v))

  # plot feature importance
  plt.bar([x for x in range(len(importance))], importance)
  plt.show()
```

[ ]:
```
#@title Data Source
datafile = "./phishing-dataset.csv" #@param {type:"string"}

# Load the pre-processed data into the `df dataframe
df = pd.read_csv(datafile, sep = ',')
print(f"Dataset '{datafile}' contains {df.shape[0]} row(s) and {df.shape[1]}␣
 ↪column(s).")
```

```
Dataset './phishing-dataset.csv' contains 19997 row(s) and 48 column(s).
```

[ ]:
```
# View the first 5 rows
df.head()
```

[ ]:
```
                                        title_clean  is_english  img_count  \
0                                         one drive           1          1
1        american express : online service : log in           1          3
2  alibabamanufacturerdirectory-suppliers , manuf…           1          6
3                                                 …           1          2
4                                               NaN           1          6

   has_form  has_login_form  has_js  js_include_b64  nb_tokens  \
0         0               0       1               0         31
1         1               1       1               0          5
2         1               1       1               0        168
3         0               1       1               0          6
4         1               1       0               0          7

                                      text_clean classification  …  \
0  one drive read document please choose email pr…      malicious  …
1               american express online service log      malicious  …
2  alibabamanufacturerdirectory-suppliers manufac…      malicious  …
3          … session expired password keep logged       malicious  …
4  gjin.jung samsung.com 3c 95338 2017 icp0803424…      malicious  …

   youtube  whatsapp  facebook_messenger  wechat  instagram  tiktok   qq  \
0      0.0       0.0                 0.0     0.0        0.0     0.0  0.0
1      0.0       0.0                 0.0     0.0        0.0     0.0  0.0
2      0.0       0.0                 0.0     0.0        0.0     0.0  0.0
3      0.0       0.0                 0.0     0.0        0.0     0.0  0.0
4      0.0       0.0                 0.0     0.0        0.0     0.0  0.0
```

```
   weibo  linkedin  twitter
0    0.0       0.0      0.0
1    0.0       0.0      0.0
2    0.0       0.0      0.0
3    0.0       0.0      0.0
4    0.0       0.0      0.0

[5 rows x 48 columns]
```

[ ]:
```
# Print the full summary
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19997 entries, 0 to 19996
Data columns (total 48 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   title_clean         17066 non-null  object
 1   is_english          19997 non-null  int64
 2   img_count           19997 non-null  int64
 3   has_form            19997 non-null  int64
 4   has_login_form      19997 non-null  int64
 5   has_js              19997 non-null  int64
 6   js_include_b64      19997 non-null  int64
 7   nb_tokens           19997 non-null  int64
 8   text_clean          19461 non-null  object
 9   classification      19997 non-null  object
 10  nb_title_entities   19997 non-null  float64
 11  nb_text_entities    19997 non-null  float64
 12  jpmorgan_chase      19997 non-null  float64
 13  bank_of_america     19997 non-null  float64
 14  wells_fargo         19997 non-null  float64
 15  hsbc                19997 non-null  float64
 16  deutsche_bank       19997 non-null  float64
 17  mitsubishi_ufj      19997 non-null  float64
 18  citibank            19997 non-null  float64
 19  rbc                 19997 non-null  float64
 20  paypal              19997 non-null  float64
 21  scotiabank          19997 non-null  float64
 22  apple               19997 non-null  float64
 23  microsoft           19997 non-null  float64
 24  amazon              19997 non-null  float64
 25  google              19997 non-null  float64
 26  samsung             19997 non-null  float64
 27  facebook            19997 non-null  float64
 28  steam               19997 non-null  float64
 29  netflix             19997 non-null  float64
```

```
 30  ups                 19997 non-null  float64
 31  fedex               19997 non-null  float64
 32  dhl                 19997 non-null  float64
 33  tnt                 19997 non-null  float64
 34  usps                19997 non-null  float64
 35  royal_mail          19997 non-null  float64
 36  purolator           19997 non-null  float64
 37  canada_post         19997 non-null  float64
 38  youtube             19997 non-null  float64
 39  whatsapp            19997 non-null  float64
 40  facebook_messenger  19997 non-null  float64
 41  wechat              19997 non-null  float64
 42  instagram           19997 non-null  float64
 43  tiktok              19997 non-null  float64
 44  qq                  19997 non-null  float64
 45  weibo               19997 non-null  float64
 46  linkedin            19997 non-null  float64
 47  twitter             19997 non-null  float64
dtypes: float64(38), int64(7), object(3)
memory usage: 7.3+ MB
```

1.7 Exploratory Data Analysis

The code below automatically creates an exploratory data analysis report. The report is output as an html file in the local files (see the files pane on the left).

For the final report/project we will want to highlight specific aspects from the EDA document that justify our decisions below and code these explicitly.

[ ]:
```
# Create a ProfileReport object 'pr' from the DataFrame 'df'.
# This generates a comprehensive exploratory data analysis (EDA) report
# for the DataFrame, including statistics, correlations, missing value analysis,
# and other relevant details about the data.
pr = ProfileReport(df)

# Export the generated EDA report to an HTML file.
# The report is saved with the name "EDA.html" in the current working directory.
# This HTML file will contain all the visualizations and insights generated by
# ProfileReport and can be viewed in a web browser.
pr.to_file(output_file = "EDA.html")
```

```
Summarize dataset:   0%|          | 0/5 [00:00<?, ?it/s]

Generate report structure:   0%|          | 0/1 [00:00<?, ?it/s]

Render HTML:   0%|          | 0/1 [00:00<?, ?it/s]

Export report to file:   0%|          | 0/1 [00:00<?, ?it/s]
```

1.7.1 EDA Overview

####Dataset statistics: * 48 variables (3 numeric predictors and 45 categorical) * 19,997 observations * 3,467 missing cells

####Constants: * mitsubishi_ufj has constant value "" * royal_mail has constant value "" * purolator has constant value "" * canada_post has constant value "" * facebook_messenger has constant value "" * wechat has constant value "" * tiktok has constant value "" * qq has constant value "" * weibo has constant value ""

####High Cardinality: * title_clean has 4,719 distinct values * text_clean has 8,109 distinct values

####Correlations: * nb_tokens is highly overall correlated with nb_text_entities * nb_text_entities is highly overall correlated with nb_tokens * has_form is highly overall correlated with has_login_form * has_login_form is highly overall correlated with has_form and classification * classification is highly overall correlated with has_login_form

####Duplicate rows: * There are 1,545 duplicate rows. It is hard to say if these are genuinely duplicates and should be removed, or if they are coincidental duplicates.

For now, leaving, but will revisit for fine tuning if we are getting poor performance.

1.8 Initial Filtering Decision

The first filtering decision we make is removing null values.

[ ]:
```
df.isna().sum()
```

[ ]:
```
title_clean          2931
is_english              0
img_count               0
has_form                0
has_login_form          0
has_js                  0
js_include_b64          0
nb_tokens               0
text_clean            536
classification          0
nb_title_entities       0
nb_text_entities        0
jpmorgan_chase          0
bank_of_america         0
wells_fargo             0
hsbc                    0
deutsche_bank           0
mitsubishi_ufj          0
citibank                0
rbc                     0
paypal                  0
scotiabank              0
```

```
apple                    0
microsoft                0
amazon                   0
google                   0
samsung                  0
facebook                 0
steam                    0
netflix                  0
ups                      0
fedex                    0
dhl                      0
tnt                      0
usps                     0
royal_mail               0
purolator                0
canada_post              0
youtube                  0
whatsapp                 0
facebook_messenger       0
wechat                   0
instagram                0
tiktok                   0
qq                       0
weibo                    0
linkedin                 0
twitter                  0
dtype: int64
```

[ ]:
```
# filling null values in title_clean with "no title"
df['title_clean'].fillna('no title', inplace = True)
```

[ ]:
```
# filling null values in text_clean with "no text"
df['text_clean'].fillna('no text', inplace = True)
```

[ ]:
```
df.isna().sum()
```

[ ]:
```
title_clean          0
is_english           0
img_count            0
has_form             0
has_login_form       0
has_js               0
js_include_b64       0
nb_tokens            0
text_clean           0
classification       0
nb_title_entities    0
```

```
nb_text_entities      0
jpmorgan_chase        0
bank_of_america       0
wells_fargo           0
hsbc                  0
deutsche_bank         0
mitsubishi_ufj        0
citibank              0
rbc                   0
paypal                0
scotiabank            0
apple                 0
microsoft             0
amazon                0
google                0
samsung               0
facebook              0
steam                 0
netflix               0
ups                   0
fedex                 0
dhl                   0
tnt                   0
usps                  0
royal_mail            0
purolator             0
canada_post           0
youtube               0
whatsapp              0
facebook_messenger    0
wechat                0
instagram             0
tiktok                0
qq                    0
weibo                 0
linkedin              0
twitter               0
dtype: int64
```

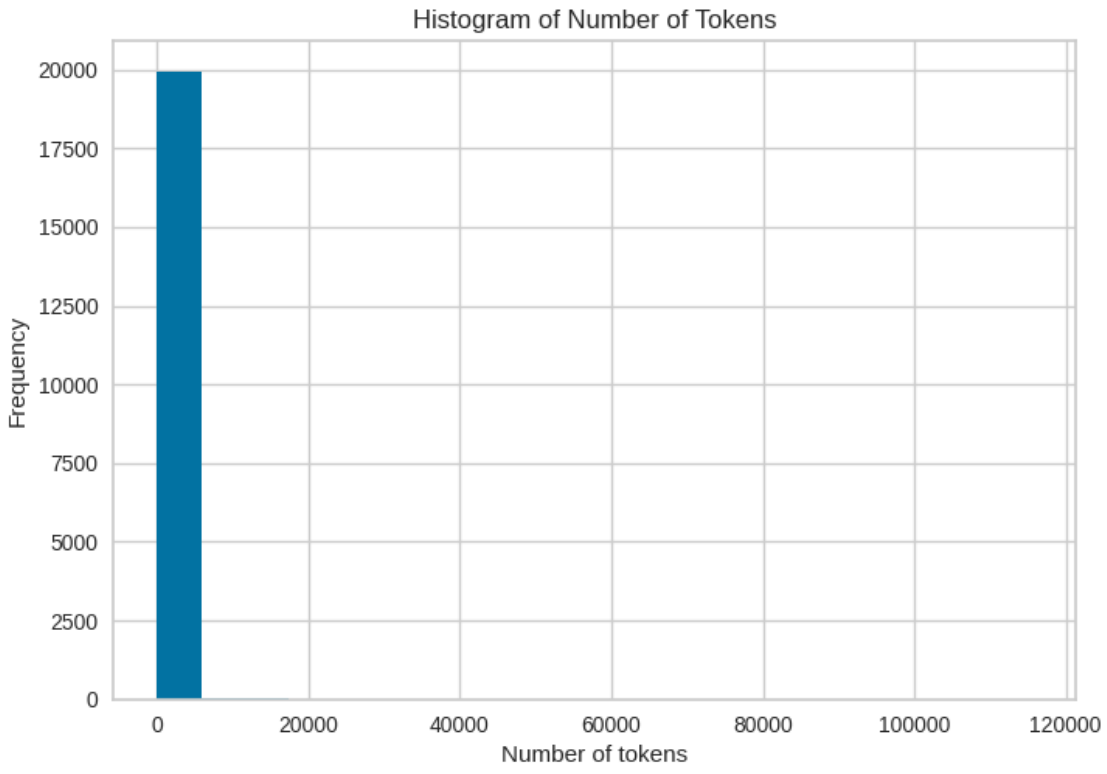
The second filtering decision we will make is to control the number of tokens to analyze by eliminating outliers in terms of token size. As such, let's review the nb_tokens column.

[ ]:
```
# Get the minimum, maximum, mean, and median values of the nb_tokens column
nb_tokens = df['nb_tokens']
min_tokens = nb_tokens.min()
max_tokens = nb_tokens.max()
mean_tokens = nb_tokens.mean()
```

```
median_tokens = nb_tokens.median()

print(f"Minimum number of tokens: {min_tokens}")
print(f"Maximum number of tokens: {max_tokens}")
print(f"Mean number of tokens: {mean_tokens}")
print(f"Median number of tokens: {median_tokens}")
```

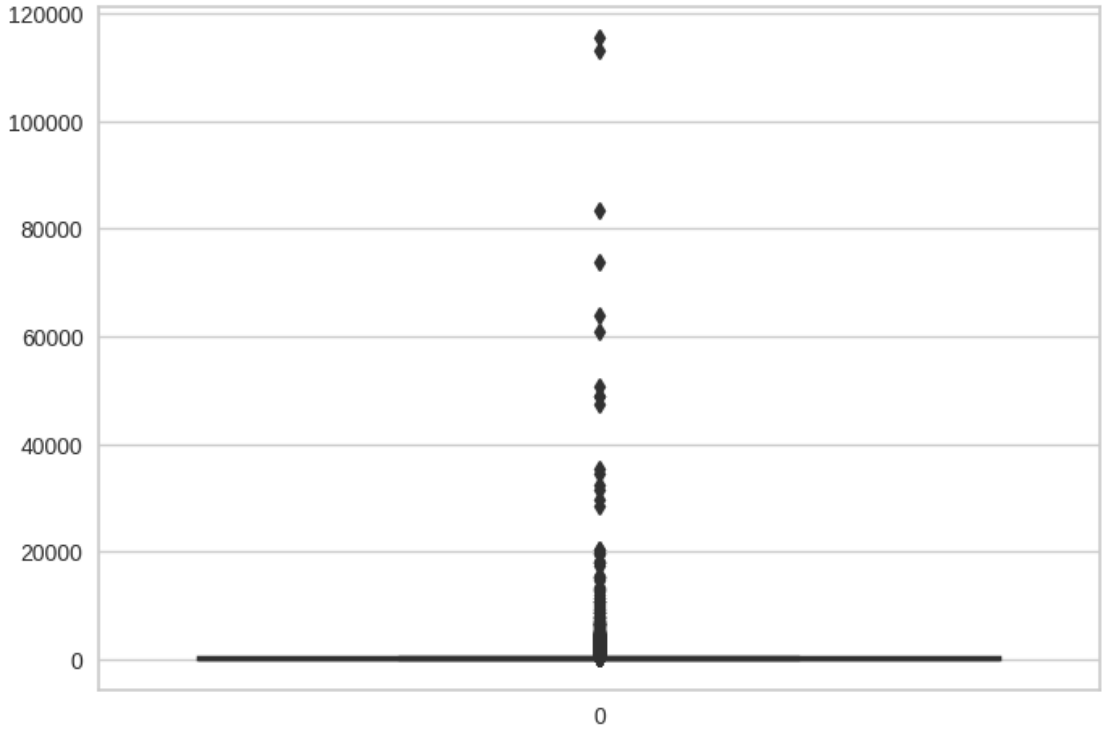
```
Minimum number of tokens: 0
Maximum number of tokens: 115550
Mean number of tokens: 363.7171075661349
Median number of tokens: 28.0
```

[ ]:
```
# Display Histogram
plt.hist(df['nb_tokens'], bins = 20) # 20 is the number of bins you want to use
plt.xlabel('Number of tokens')
plt.ylabel('Frequency')
plt.title('Histogram of Number of Tokens')
plt.show()
```



[ ]:
```
# Display plot
plt.figure()
sns.boxplot(df['nb_tokens'])
```

```
plt.show()
```



As we can see, there are a relatively small number of very large documents which will need to be removed as most have around 21 tokens.

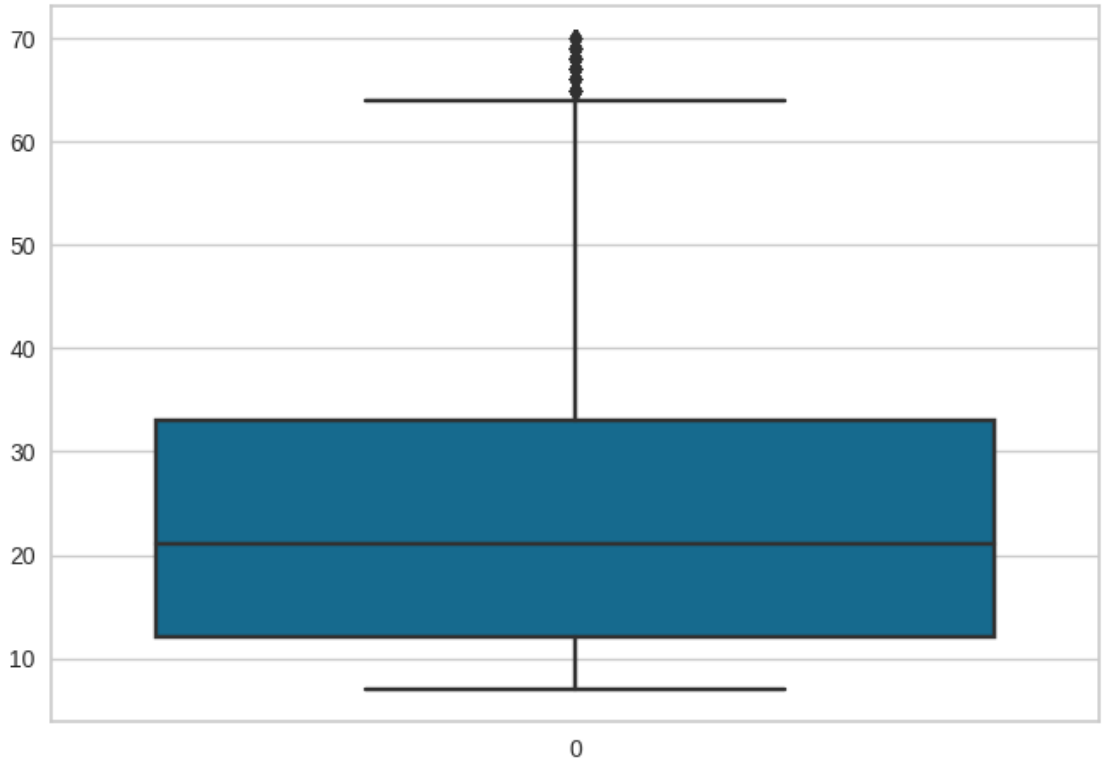
[ ]:
```
#@title Filtering Options
min_tokens = 7 #@param { type:"integer" }
max_tokens = 70 #@param { type:"integer" }
```

[ ]:
```
# Find out a range for `nb-tokens`
df_token = df[(df['nb_tokens'] >= min_tokens) & (df['nb_tokens'] <= max_tokens)]

nb_tokens = df_token['nb_tokens']
min_tokens = nb_tokens.min()
max_tokens = nb_tokens.max()
mean_tokens = nb_tokens.mean()
median_tokens = nb_tokens.median()

print(f"Minimum number of tokens: {min_tokens}")
print(f"Maximum number of tokens: {max_tokens}")
print(f"Mean number of tokens: {mean_tokens}")
print(f"Median number of tokens: {median_tokens}")
```

```
Minimum number of tokens: 7
```

```
Maximum number of tokens: 70
Mean number of tokens: 24.809953161592507
Median number of tokens: 21.0
```

[ ]:
```
# Display the plot
plt.figure()
sns.boxplot(df_token['nb_tokens'].values)
plt.show()
```



After multiple adjustments, the optimal range of tokens appears to be between 7 and 65 tokens.

1.8.1 Initial Filtering

After specifying the file containing our dataset, we load its contents into a `DataFrame` object. Some additional filtering is done on the dataset to eliminate rows:

- **Minimum Tokens**: We filter out webpages that to not have the required minimum of tokens.
- **Maximum Tokens**: We also filter out webpages that have an outsized number of tokens,
- **Maximum Number of N-Grams**: The number of n-grams to include in our `CountVectorizer` object,
- **Non-English Webpages**: We only consider webpages written in English.

During the same filtering step, we are also going to drop all columns that have a constant value of 0: * mitsubishi_ufj * royal_mail * purolator * canada_post * facebook_messenger * tiktok * qq

* wechat * weibo

[ ]:
```
#@title Filtering Options
min_tokens = 7 #@param { type:"integer" }
max_tokens = 70 #@param { type:"integer" }
max_words = 250 #@param { type:"integer" }
english_only = True #@param { type:"boolean" }
```

[ ]:
```
%%capture
# Drop unneeded columns
drop_column('is_english')
drop_column('title_raw')

# Remove columns that have a constant value 0

# mitsubishi_ufj
# royal_mail
# purolator
# canada_post
# facebook_messenger
# wechat
# tiktok
# qq
# weibo

drop_column('mitsubishi_ufj')
drop_column('royal_mail')
drop_column('purolator')
drop_column('canada_post')
drop_column('facebook_messenger')
drop_column('wechat')
drop_column('tiktok')
drop_column('qq')
drop_column('weibo')

# Keep rows with at the required amount of tokens
df = df[(df['nb_tokens'] >= min_tokens) & (df['nb_tokens'] <= max_tokens)]

# Remove misclassified rows
df = df[(df['classification'] == 'benign') | (df['classification'] ==␣
 ↪'malicious')]

# Remove strings containing special characters or misparsed HTML tags and code.
replace_string('text_clean', '_', ' ')
replace_string('text_clean', '//', ' ')
replace_string('text_clean', 'javascript', '')
replace_string('text_clean', 'https', '')
```

```
replace_string('text_clean', 'http', '')
```

1.8.2 Dataset Information

We can now verify the remaining number of rows in our dataset. Using the sample of 20,000 websites results in a small dataset of 8,205 rows, which is relatively small. In a production environment, we would be using the full dataset of 380,000 samples, but for the sake of demonstrating the process, we will keep this small set for now.

Size

[ ]:
```
print(f"The features dataset contains {df.shape[0]} rows and {df.shape[1]}␣
↪columns.")
```

```
The features dataset contains 8540 rows and 38 columns.
```

Empty Columns

[ ]:
```
df.isnull().sum()
```

[ ]:
```
title_clean          0
img_count            0
has_form             0
has_login_form       0
has_js               0
js_include_b64       0
nb_tokens            0
text_clean           0
classification       0
nb_title_entities    0
nb_text_entities     0
jpmorgan_chase       0
bank_of_america      0
wells_fargo          0
hsbc                 0
deutsche_bank        0
citibank             0
rbc                  0
paypal               0
scotiabank           0
apple                0
microsoft            0
amazon               0
google               0
samsung              0
facebook             0
steam                0
netflix              0
ups                  0
```

```
fedex                0
dhl                  0
tnt                  0
usps                 0
youtube              0
whatsapp             0
instagram            0
linkedin             0
twitter              0
dtype: int64
```

1.8.3 Labels

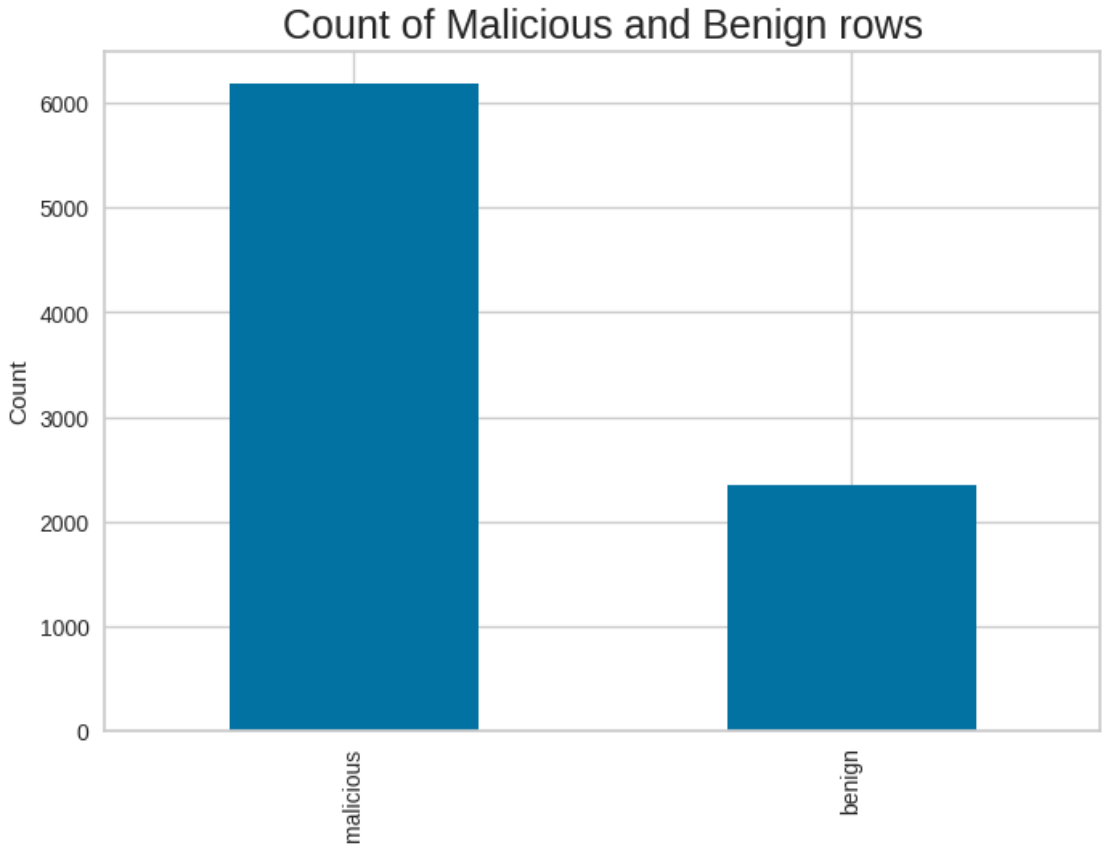
[ ]:
```
df.head()
```

[ ]:
```
                       title_clean  img_count  has_form  has_login_form  \
0                        one drive          1         0               0
4                         no title          6         1               1
5   email security : : user account          0         1               1
8           sign in to your account          4         1               1
10                       worldclient          2         1               1

    has_js  js_include_b64  nb_tokens  \
0        1               0         31
4        0               0          7
5        1               0         19
8        0               0         14
10       1               0         13

                                          text_clean classification  \
0   one drive read document please choose email pr…      malicious
4   gjin.jung samsung.com 3c 95338 2017 icp0803424…      malicious
5   email security user account verification neede…      malicious
8   sign account nobody mycraftmail.com enter pass…      malicious
10  worldclient microsoft office verification port…      malicious

    nb_title_entities  …  ups  fedex  dhl  tnt  usps  youtube  whatsapp  \
0                 0.0  …  0.0    0.0  0.0  0.0   0.0      0.0       0.0
4                 0.0  …  0.0    0.0  0.0  0.0   0.0      0.0       0.0
5                 0.0  …  0.0    0.0  0.0  0.0   0.0      0.0       0.0
8                 0.0  …  0.0    0.0  0.0  0.0   0.0      0.0       0.0
10                0.0  …  0.0    0.0  0.0  0.0   0.0      0.0       0.0

    instagram  linkedin  twitter
0         0.0       0.0      0.0
4         0.0       0.0      0.0
5         0.0       0.0      0.0
```

```
8          0.0       0.0       0.0
10         0.0       0.0       0.0

[5 rows x 38 columns]
```

Value Counts Calculate and visualize the frequency distribution of values in each column.

[ ]:
```
label = "classification"
title = "Count of Malicious and Benign rows"
y_label = "Count"
x_label = ""
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```

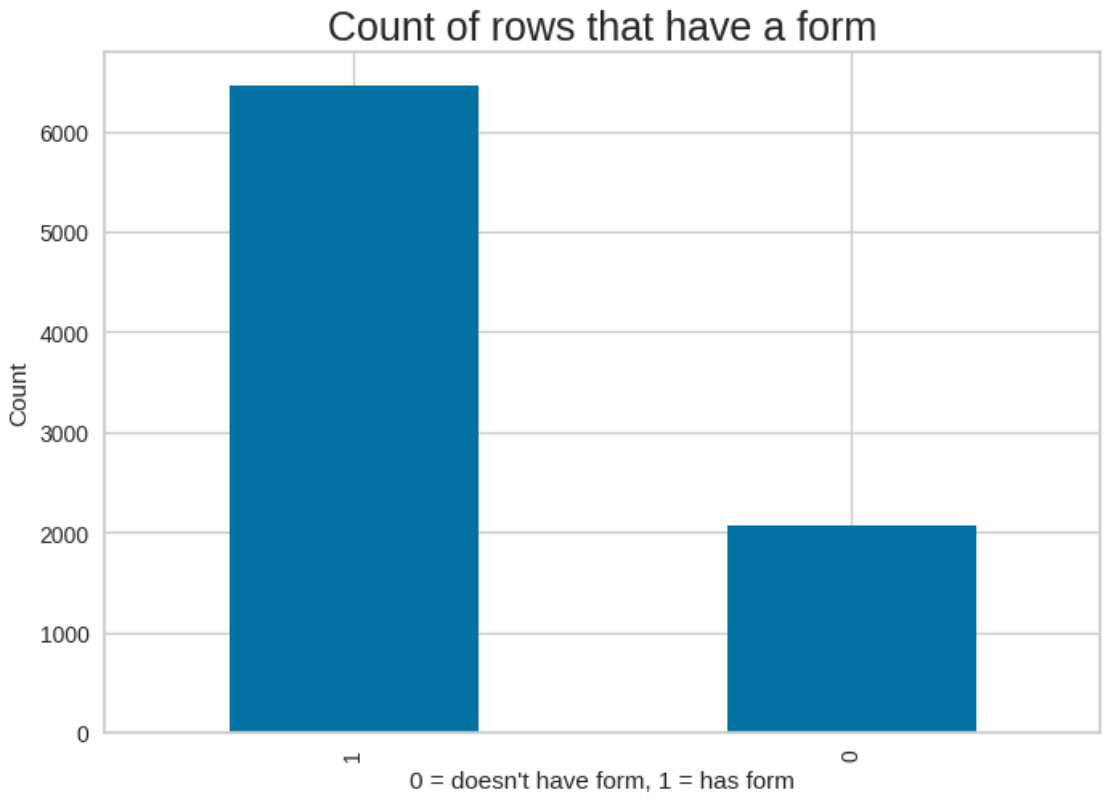


[ ]:
```
# Display counts of Malicious and Benign rows
classification_counts = df['classification'].value_counts()
classification_counts
```

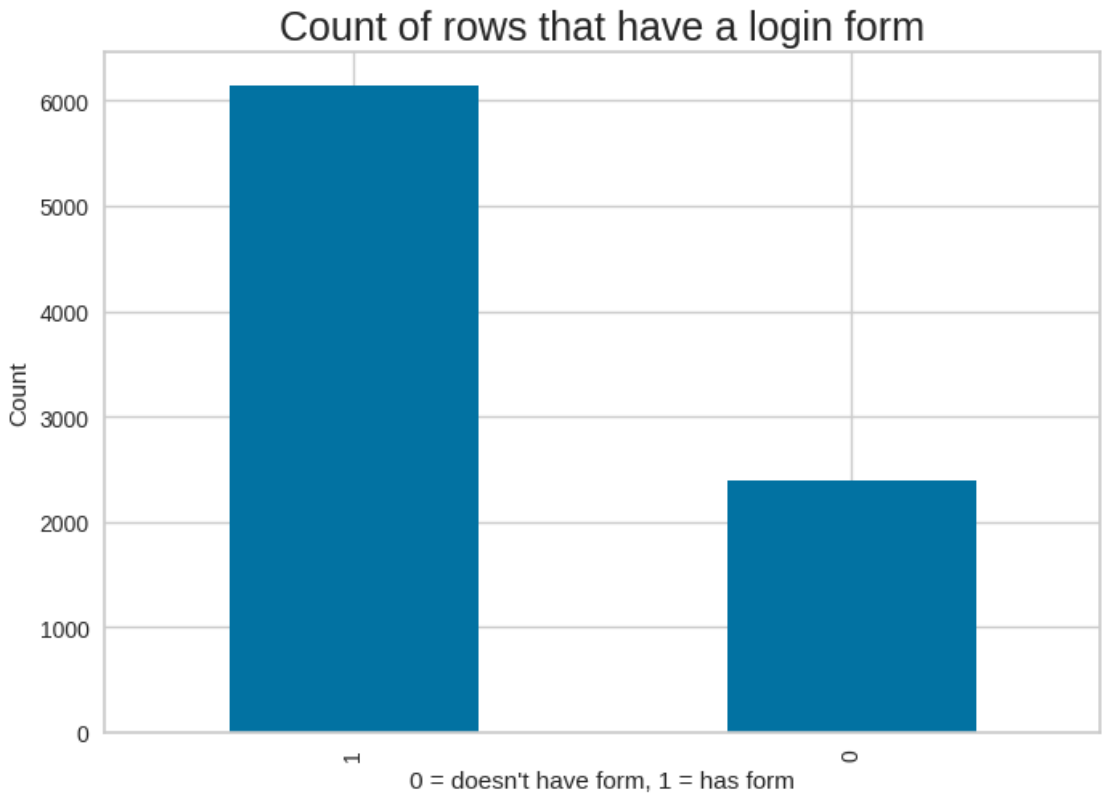
[ ]:
```
malicious    6187
benign       2353
```

```
Name: classification, dtype: int64
```

[ ]:
```
label = "has_form"
title = "Count of rows that have a form"
y_label = "Count"
x_label = "0 = doesn't have form, 1 = has form"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



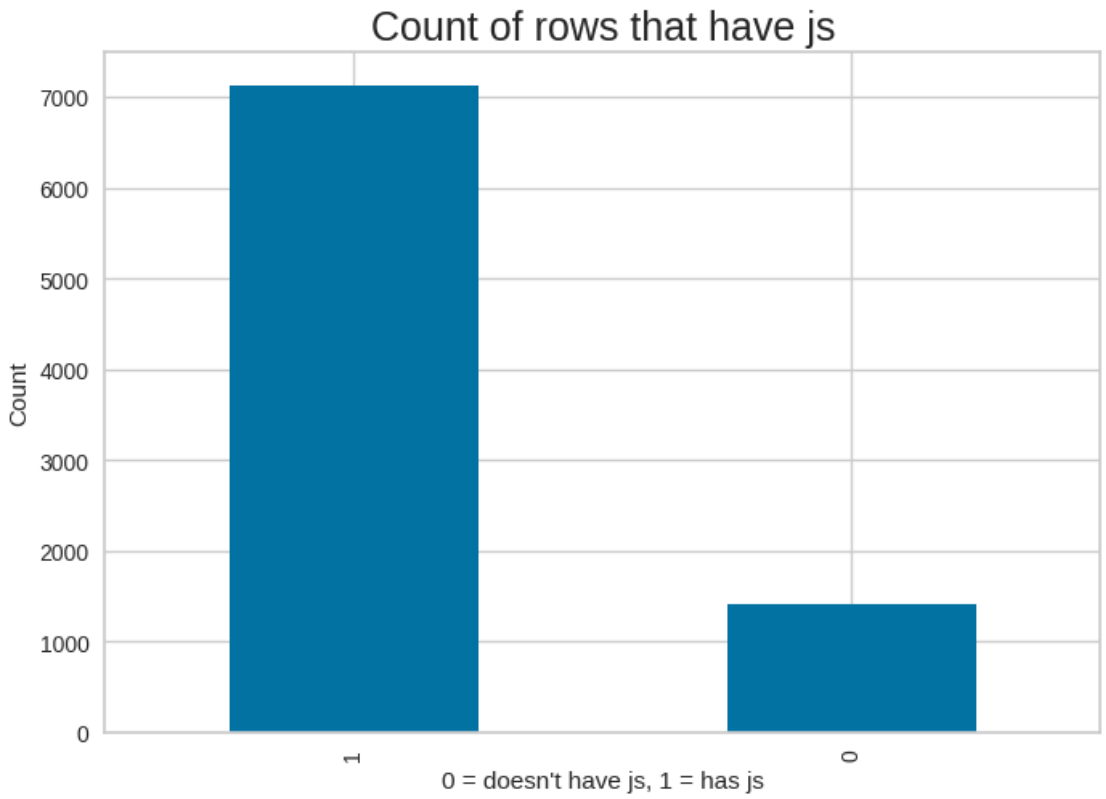
[ ]:
```
label = "has_login_form"
title = "Count of rows that have a login form"
y_label = "Count"
x_label = "0 = doesn't have form, 1 = has form"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



JavaScript Data

The presence of *JavaScript* from webpages is collected. *JavaScript* is widely used on many webpages to provide dynamic content. It can also be used for cross-site scripting and inject remote malicious code in the page.

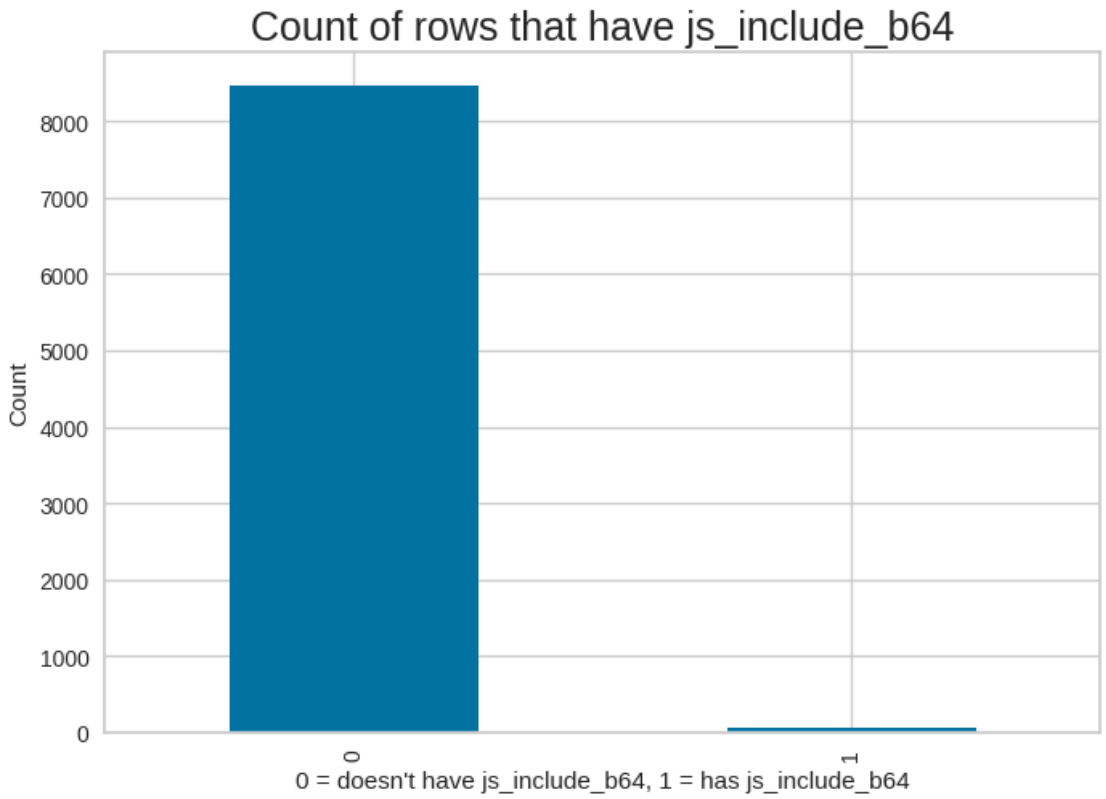
[ ]:
```
label = "has_js"
title = "Count of rows that have js"
y_label = "Count"
x_label = "0 = doesn't have js, 1 = has js"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



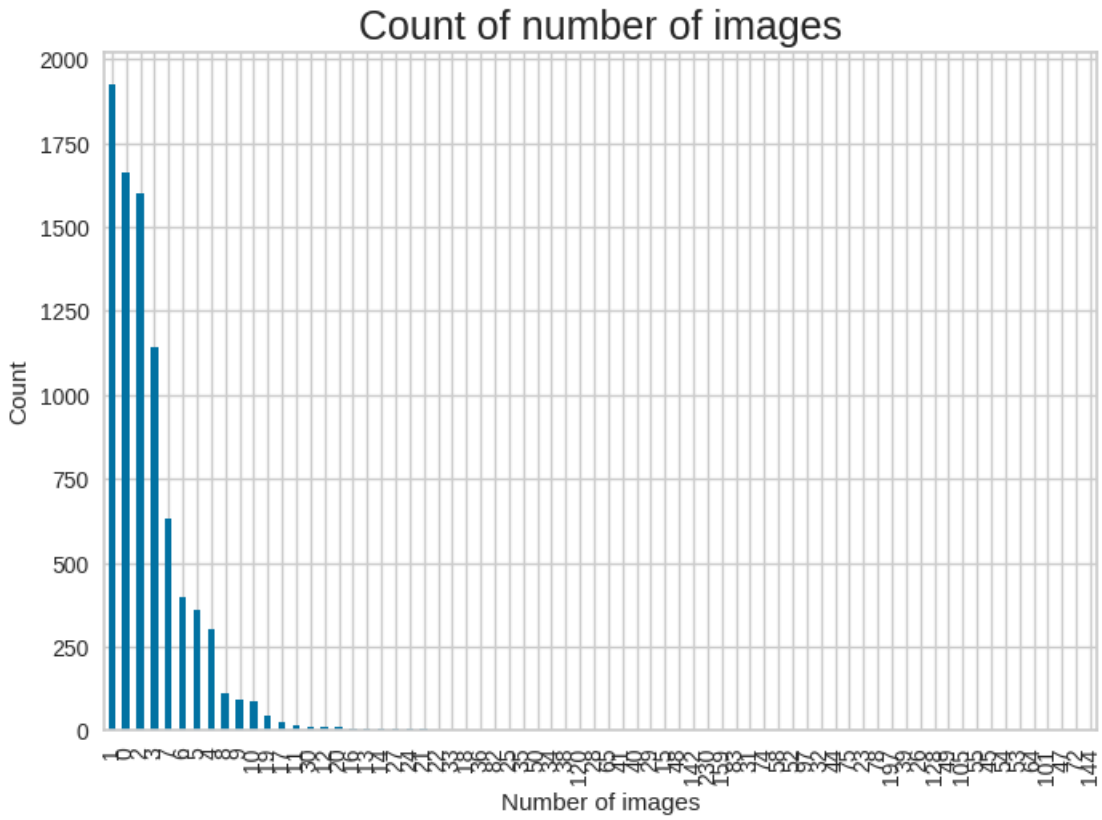
Base64

We collect information about the presence of Base64 encoded-string in webpages. Base64 is often used to obfuscated malicious JavaScript code. While this is not common, when this feature is present, there may be a high correlation between this feature the malicious webpages.

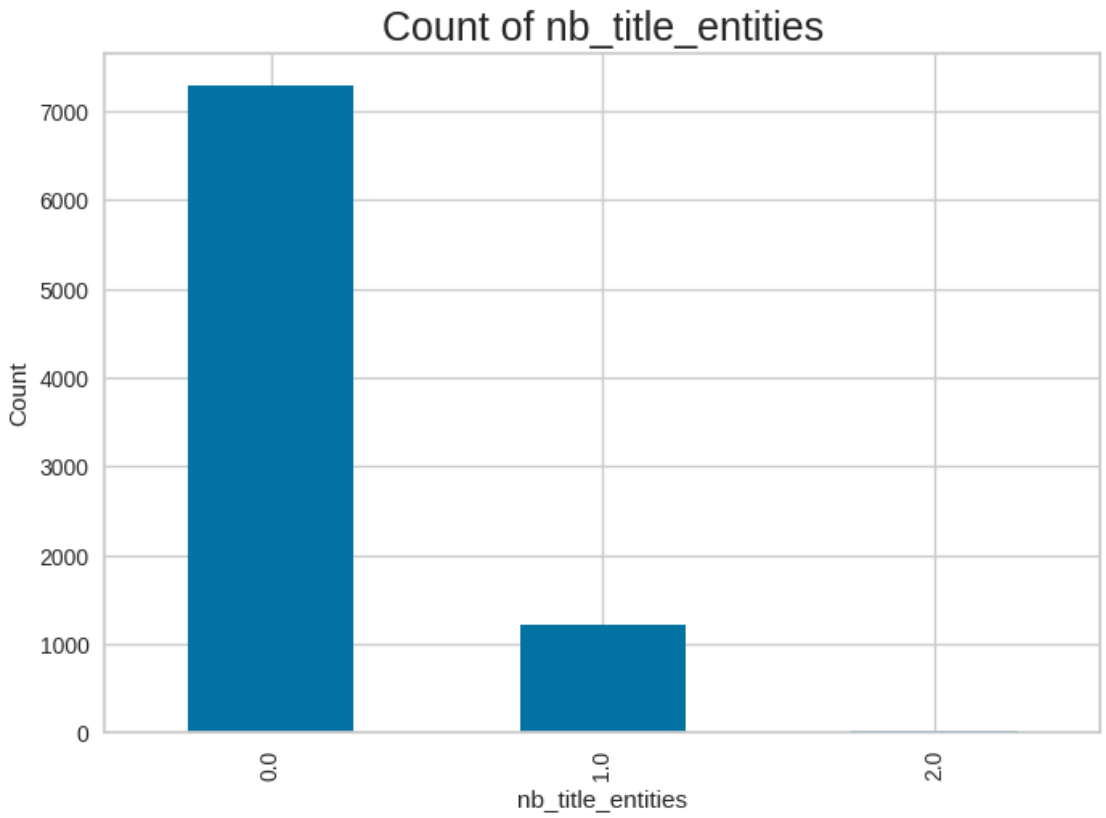
[ ]:
```
label = "js_include_b64"
title = "Count of rows that have js_include_b64"
y_label = "Count"
x_label = "0 = doesn't have js_include_b64, 1 = has js_include_b64"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



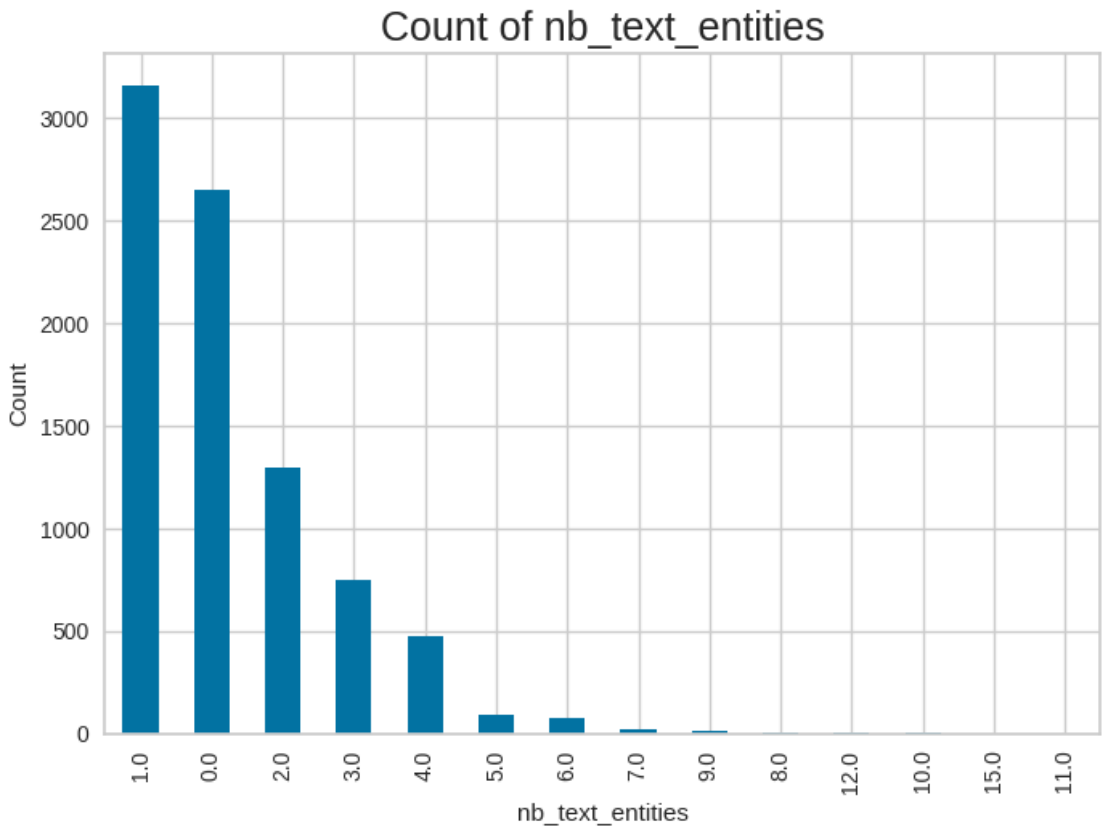
[ ]:
```
label = "img_count"
title = "Count of number of images"
y_label = "Count"
x_label = "Number of images"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



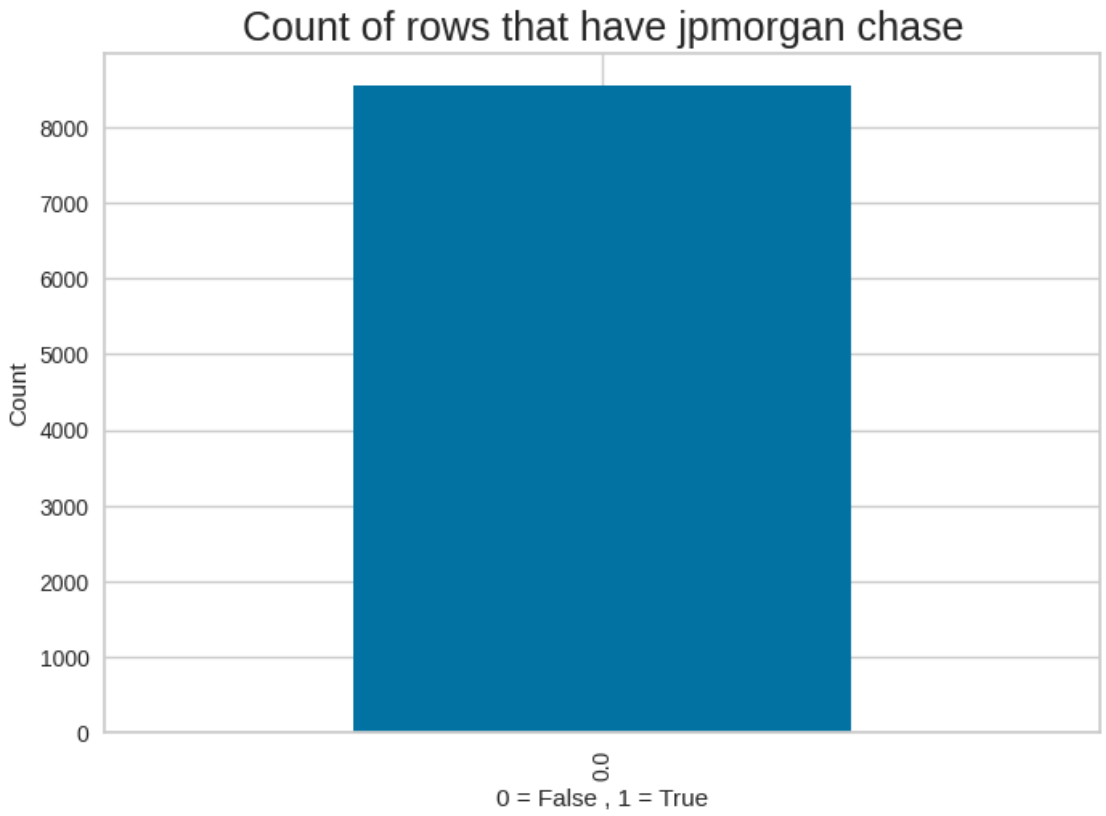
[ ]:
```
label = "nb_title_entities"
title = "Count of nb_title_entities"
y_label = "Count"
x_label = "nb_title_entities"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



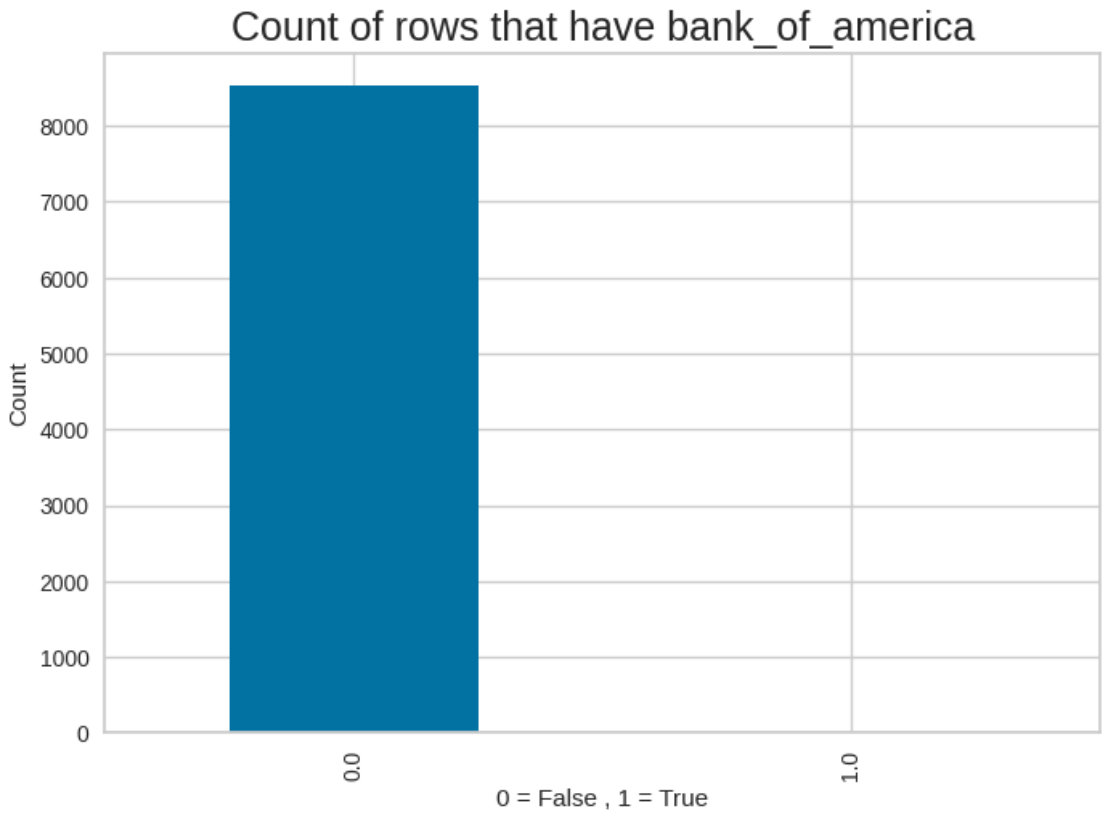
[ ]:
```
label = "nb_text_entities"
title = "Count of nb_text_entities"
y_label = "Count"
x_label = "nb_text_entities"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



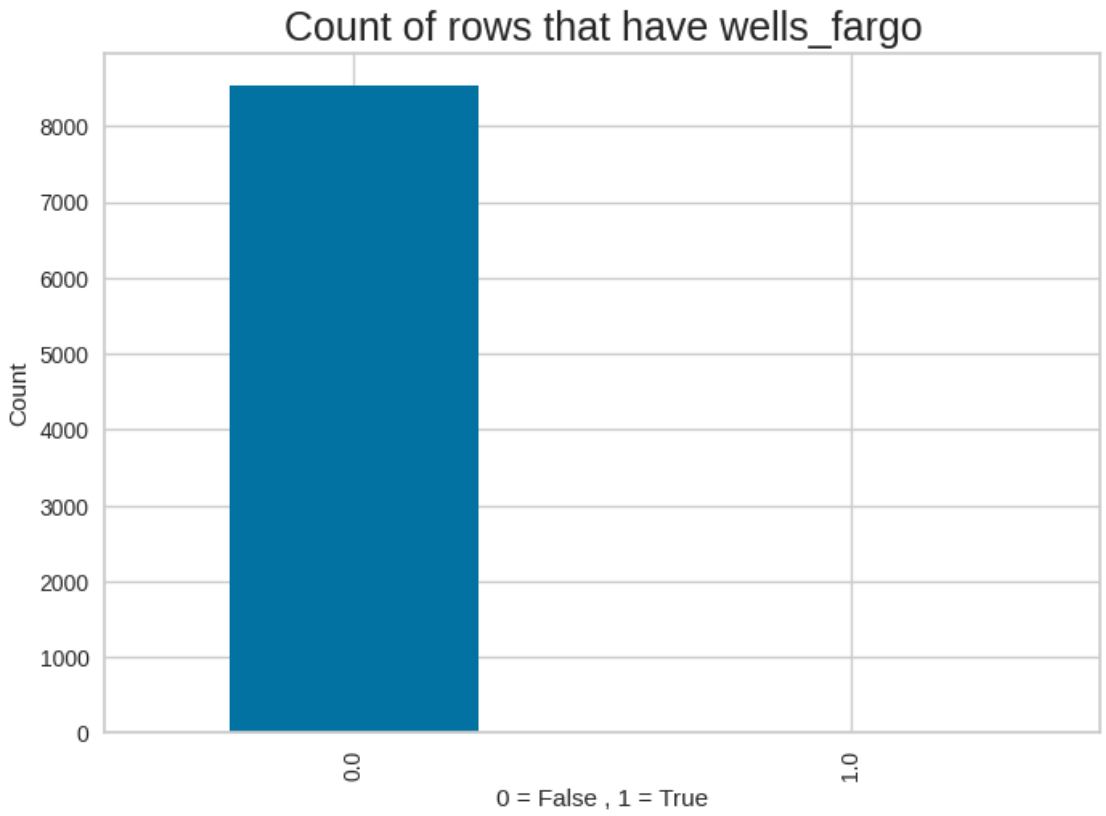
[ ]:
```
label = "jpmorgan_chase"
title = "Count of rows that have jpmorgan chase"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



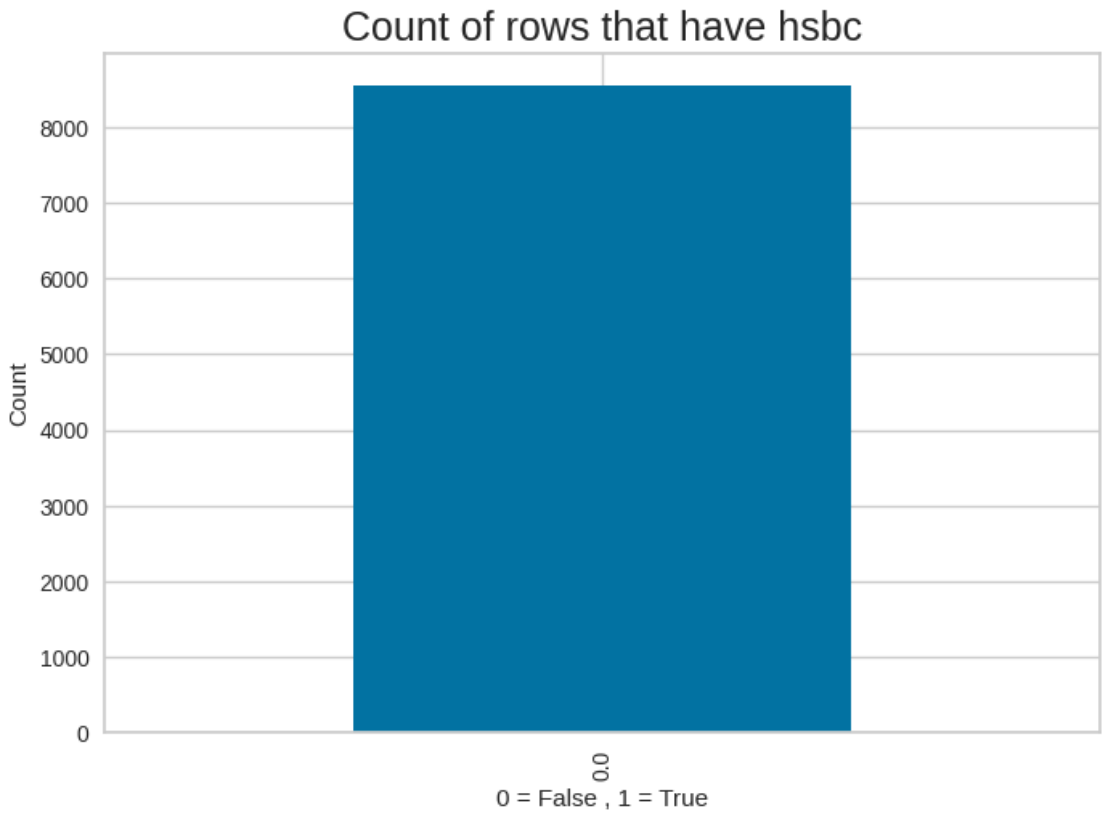
[ ]:
```
label = "bank_of_america"
title = "Count of rows that have bank_of_america"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



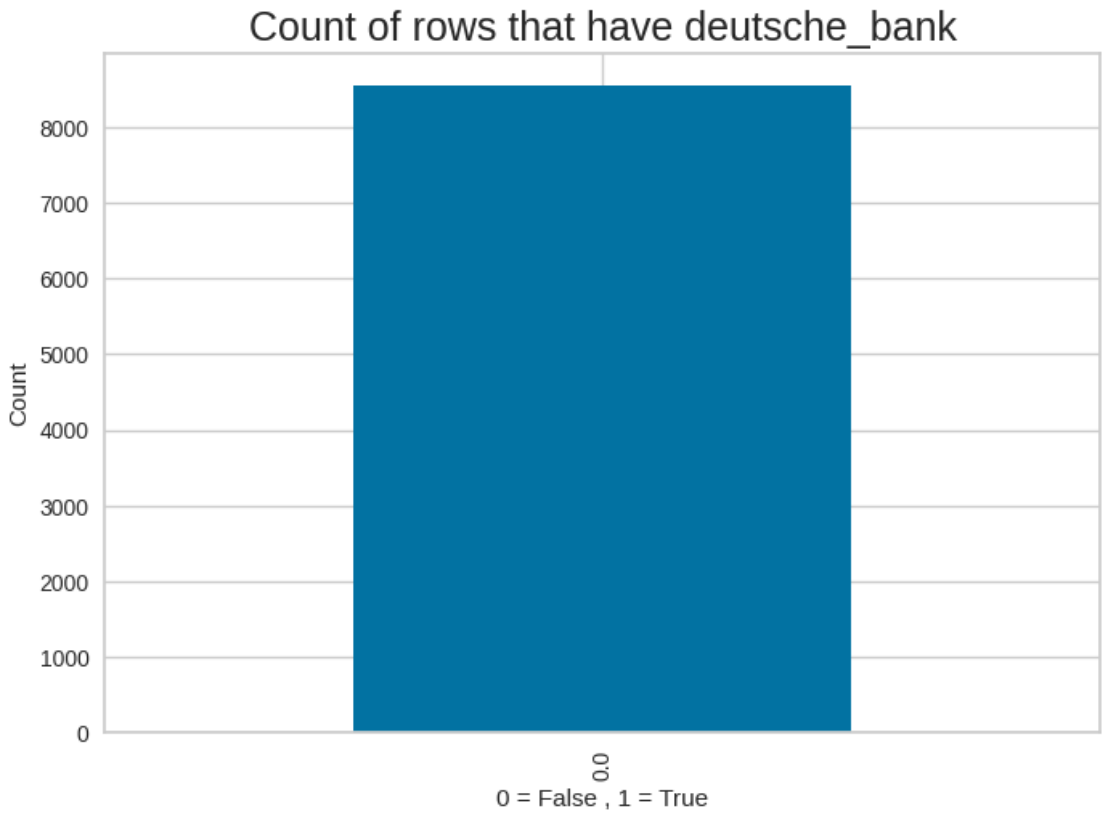
[ ]:
```
label = "wells_fargo"
title = "Count of rows that have wells_fargo"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



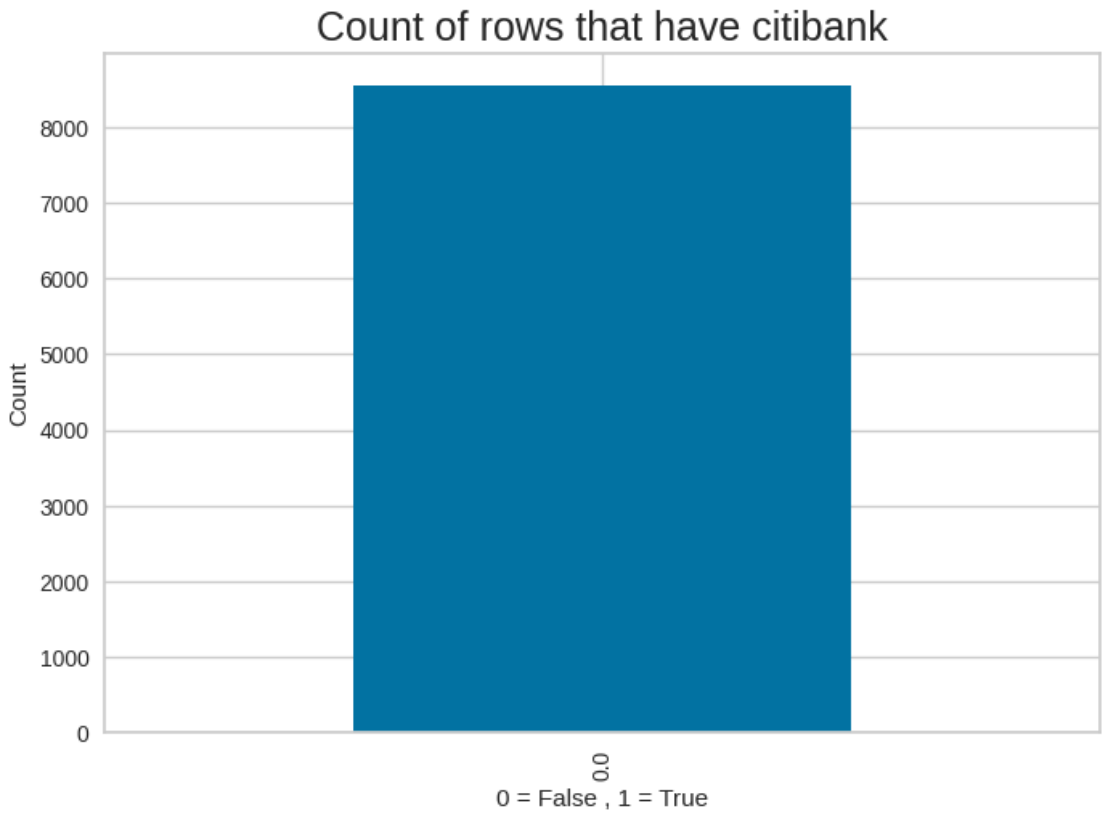
[ ]:
```
label = "hsbc"
title = "Count of rows that have hsbc"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



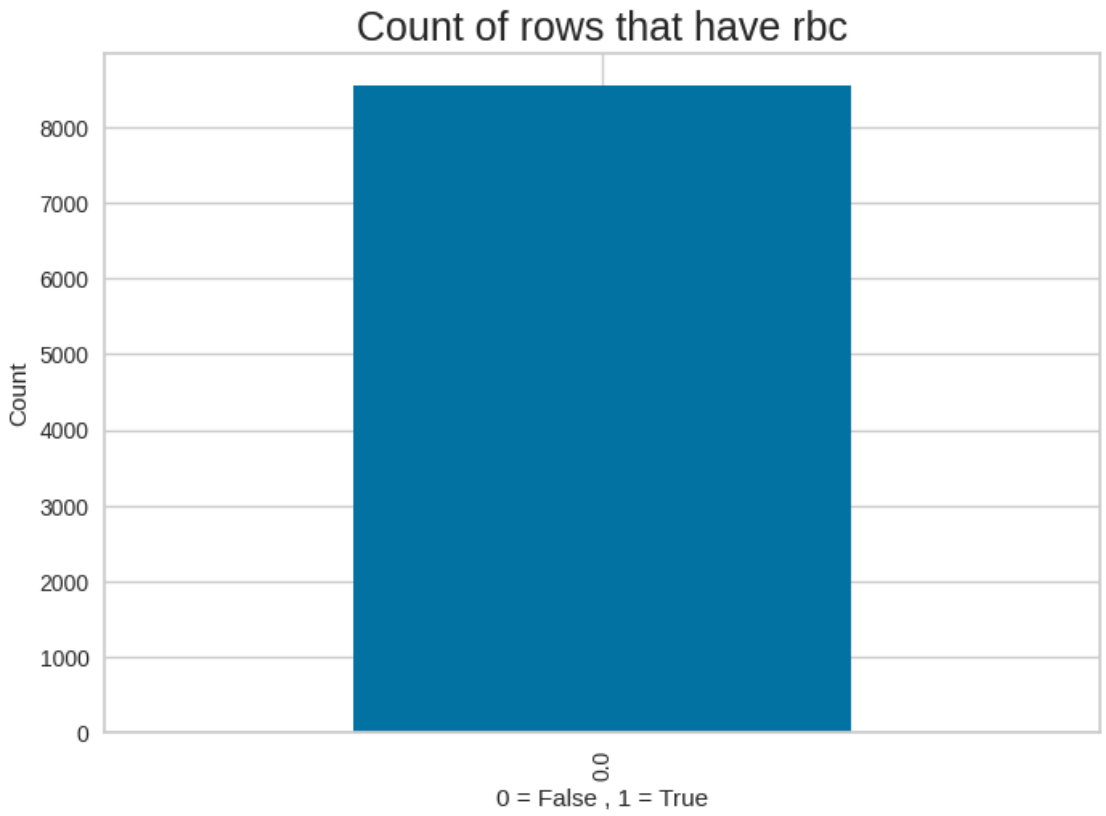
[ ]:
```
label = "deutsche_bank"
title = "Count of rows that have deutsche_bank"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



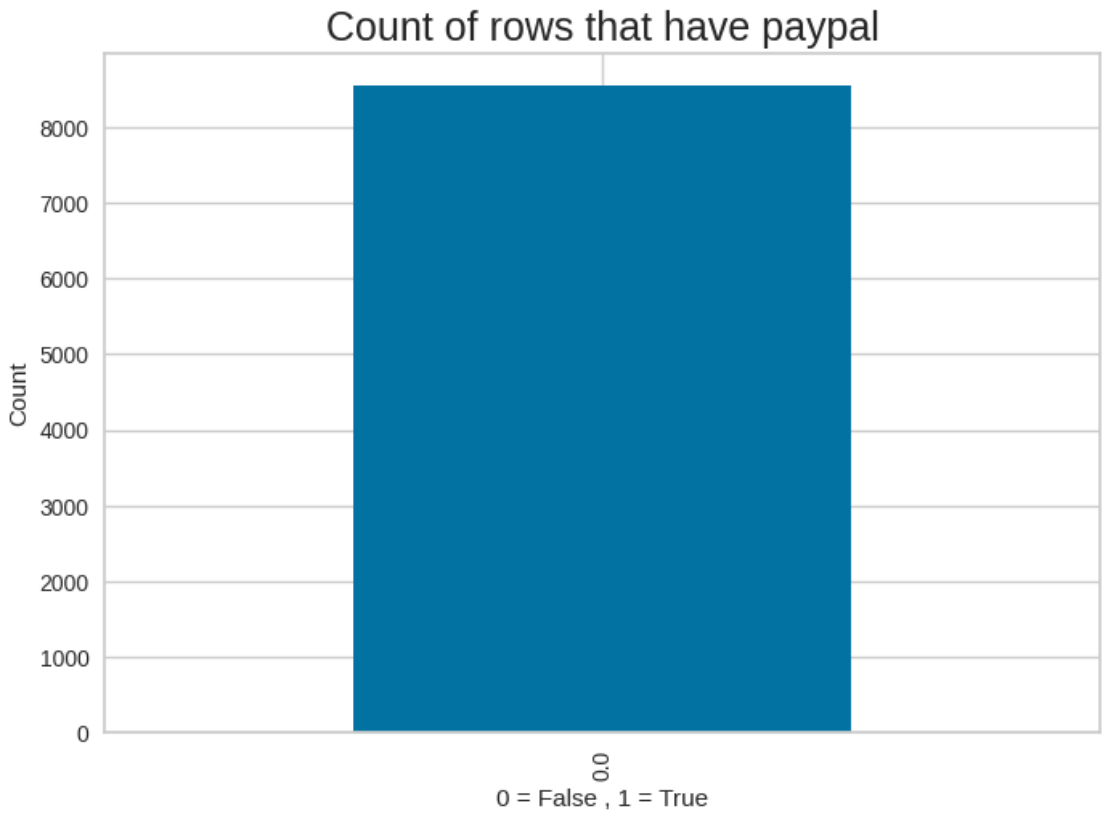
[ ]:
```
label = "citibank"
title = "Count of rows that have citibank"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



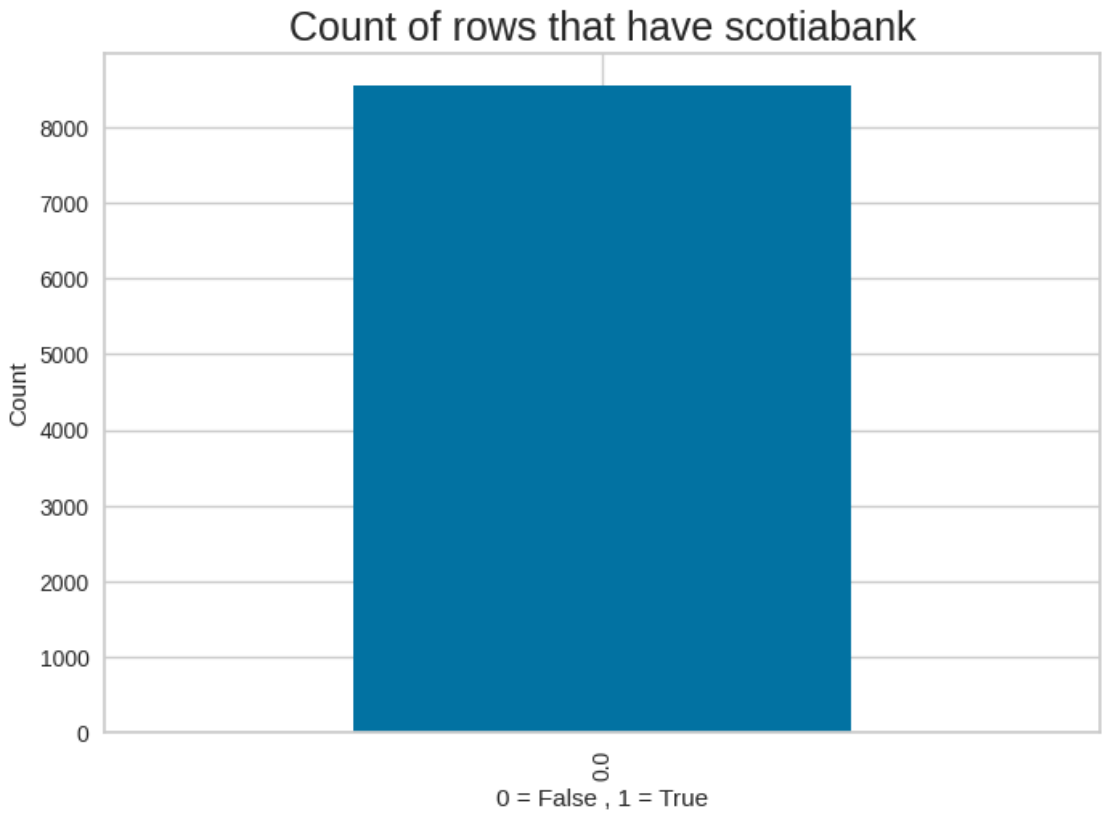
[ ]:
```
label = "rbc"
title = "Count of rows that have rbc"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



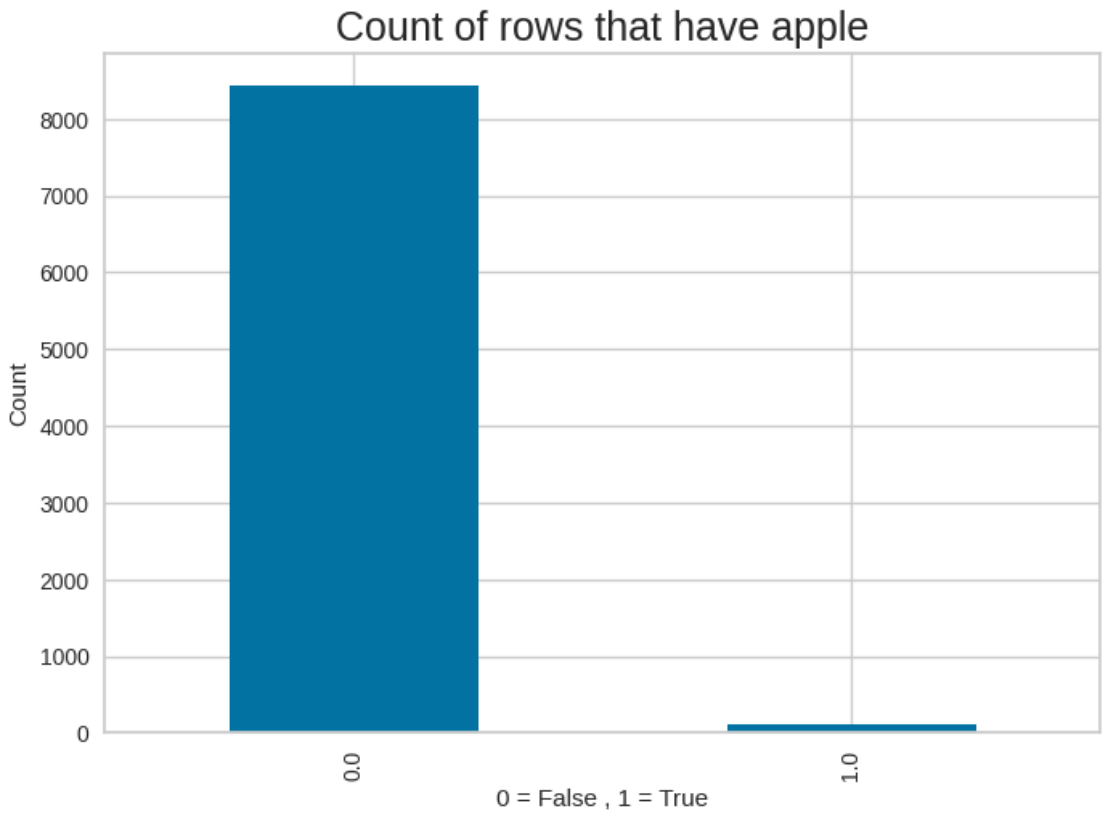
[ ]:
```
label = "paypal"
title = "Count of rows that have paypal"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



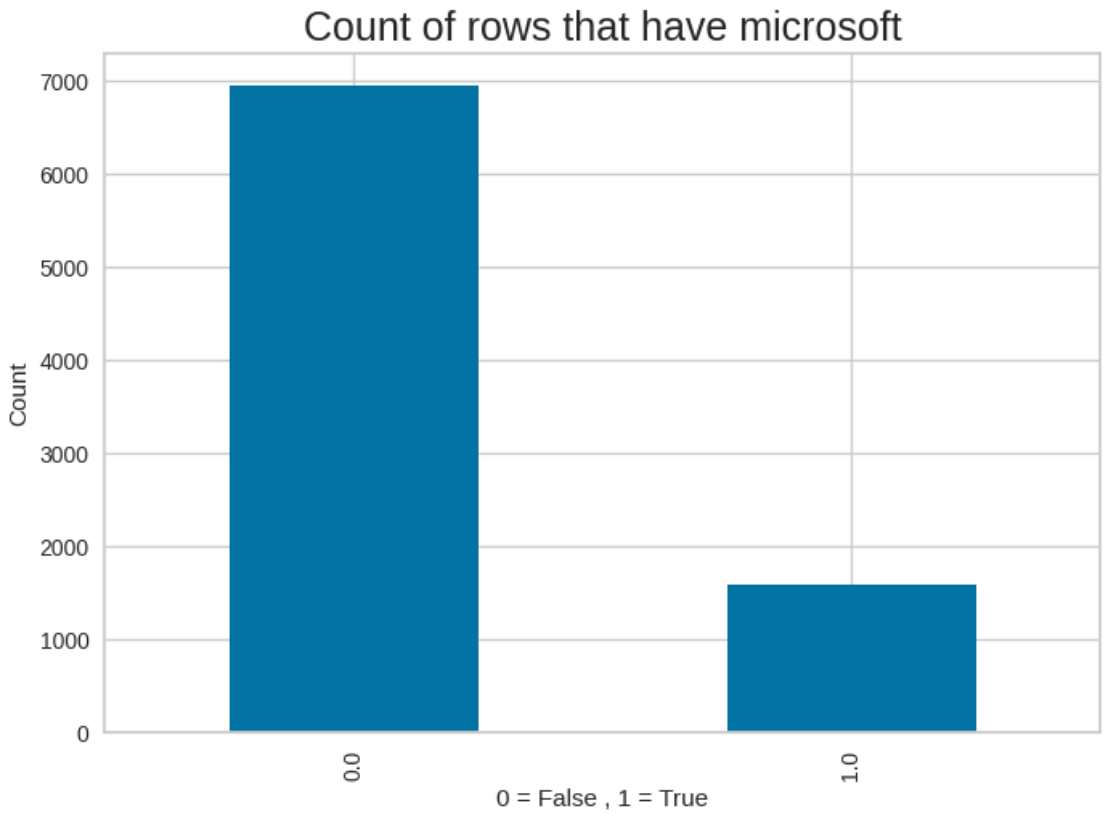
[ ]:
```
label = "scotiabank"
title = "Count of rows that have scotiabank"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



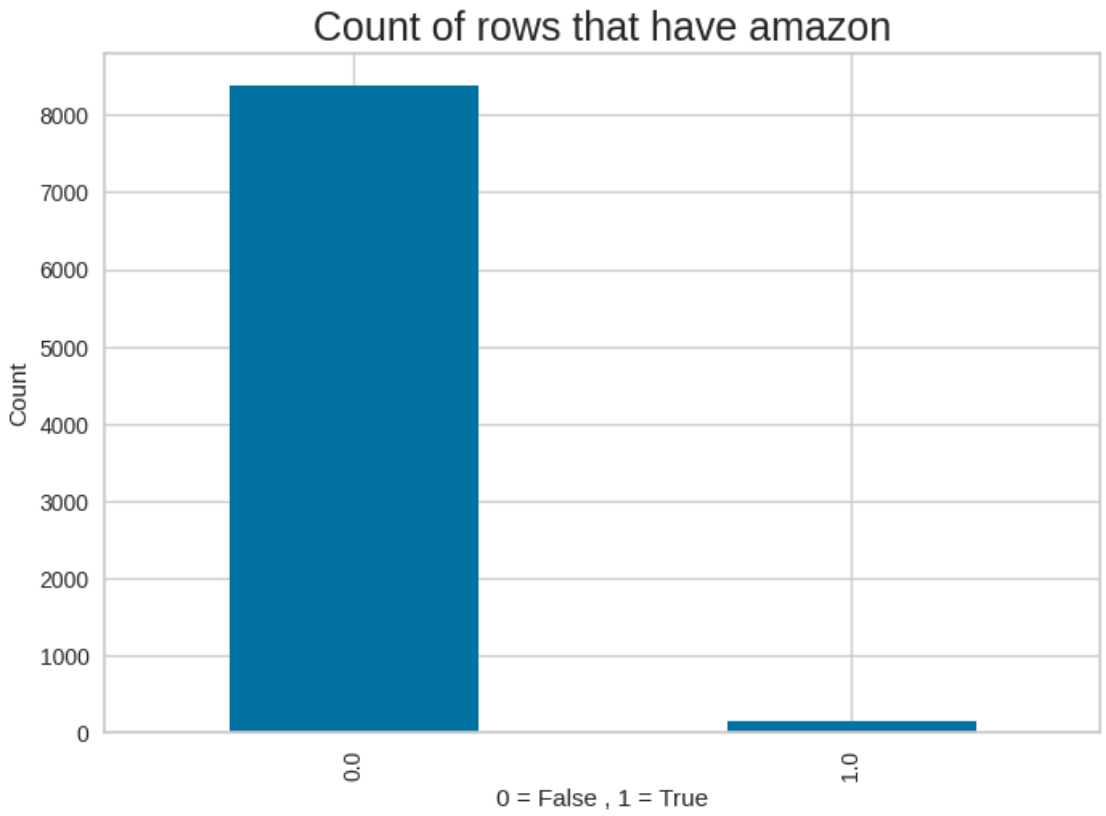
[ ]:
```
label = "apple"
title = "Count of rows that have apple"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



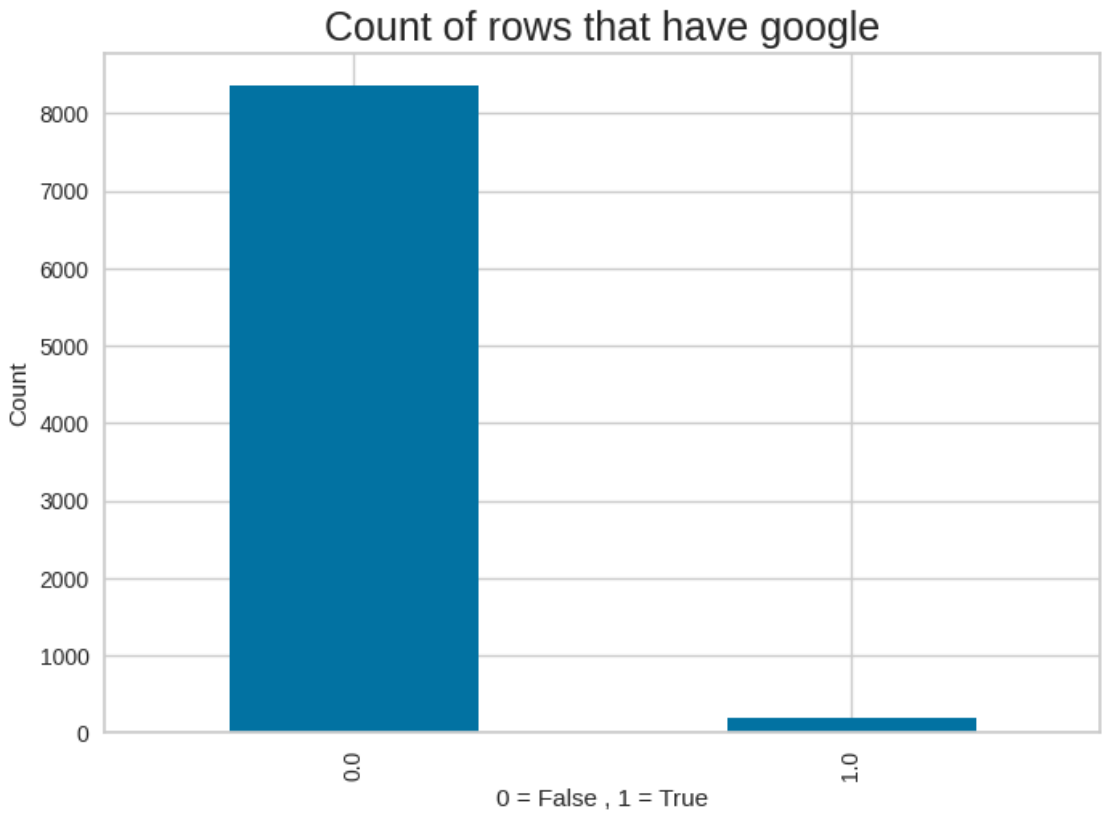
[ ]:
```
label = "microsoft"
title = "Count of rows that have microsoft"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



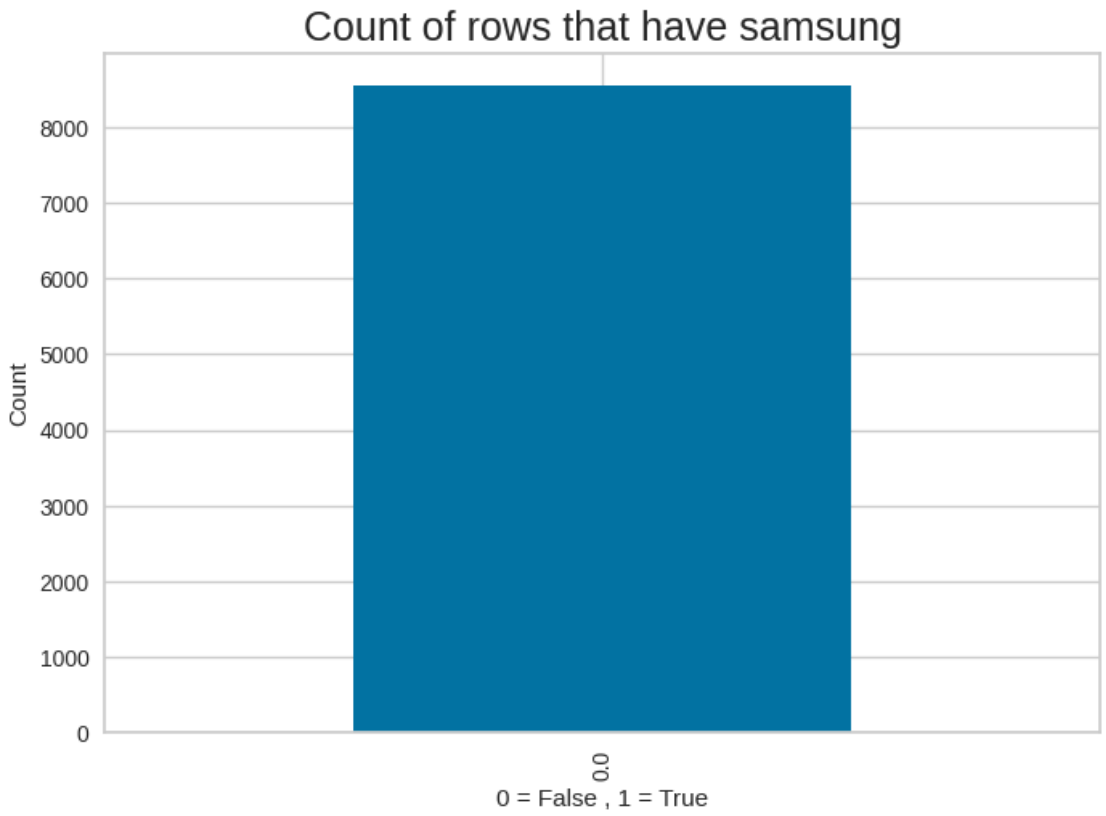
[ ]:
```
label = "amazon"
title = "Count of rows that have amazon"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



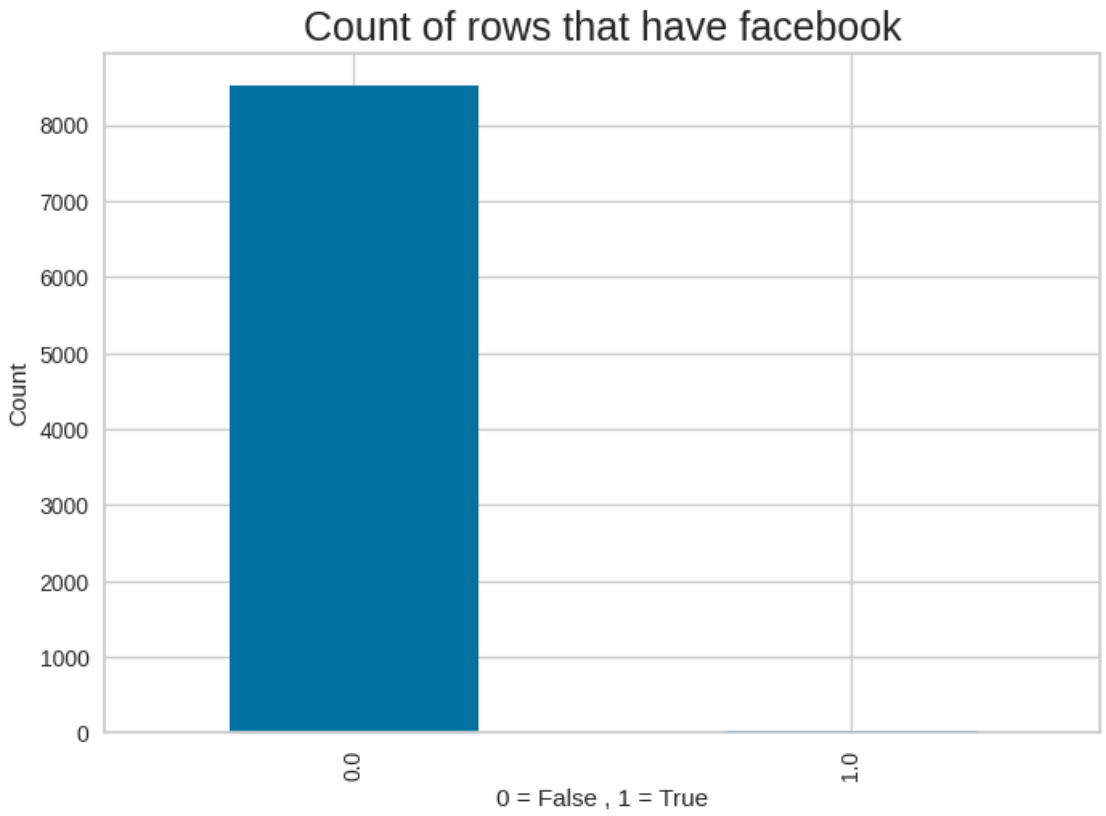
[ ]:
```
label = "google"
title = "Count of rows that have google"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



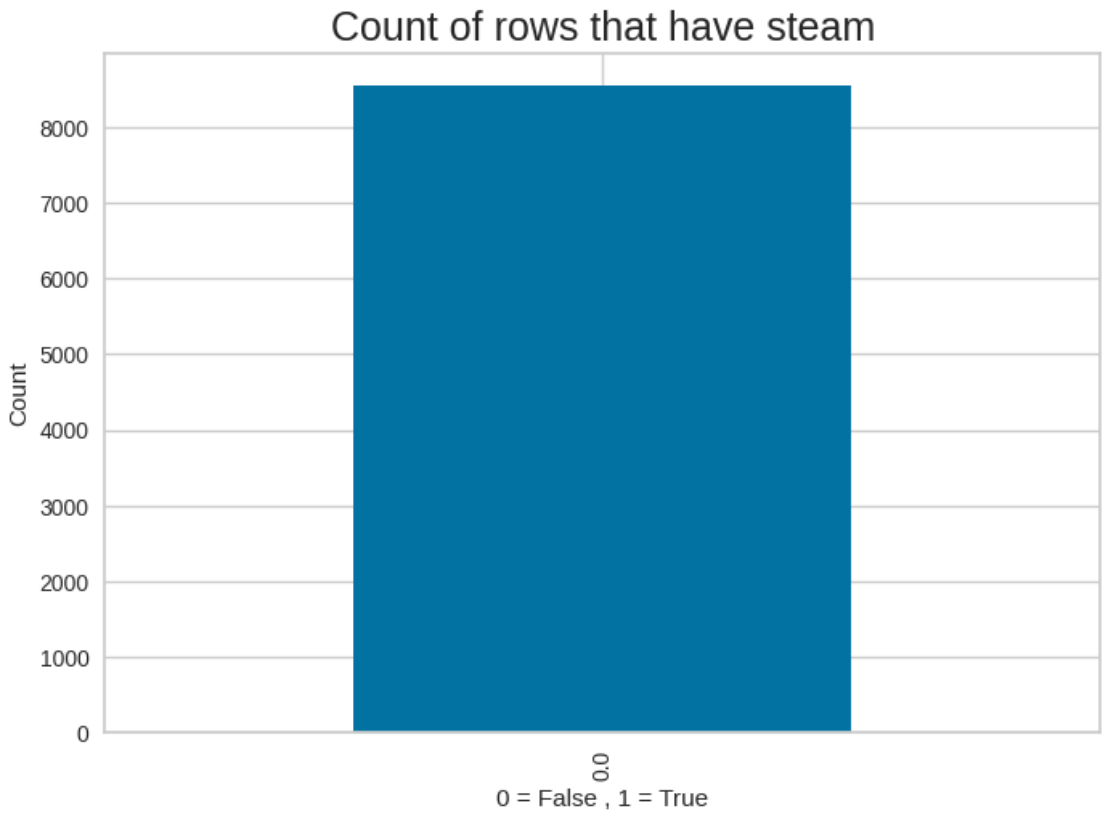
[ ]:
```
label = "samsung"
title = "Count of rows that have samsung"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



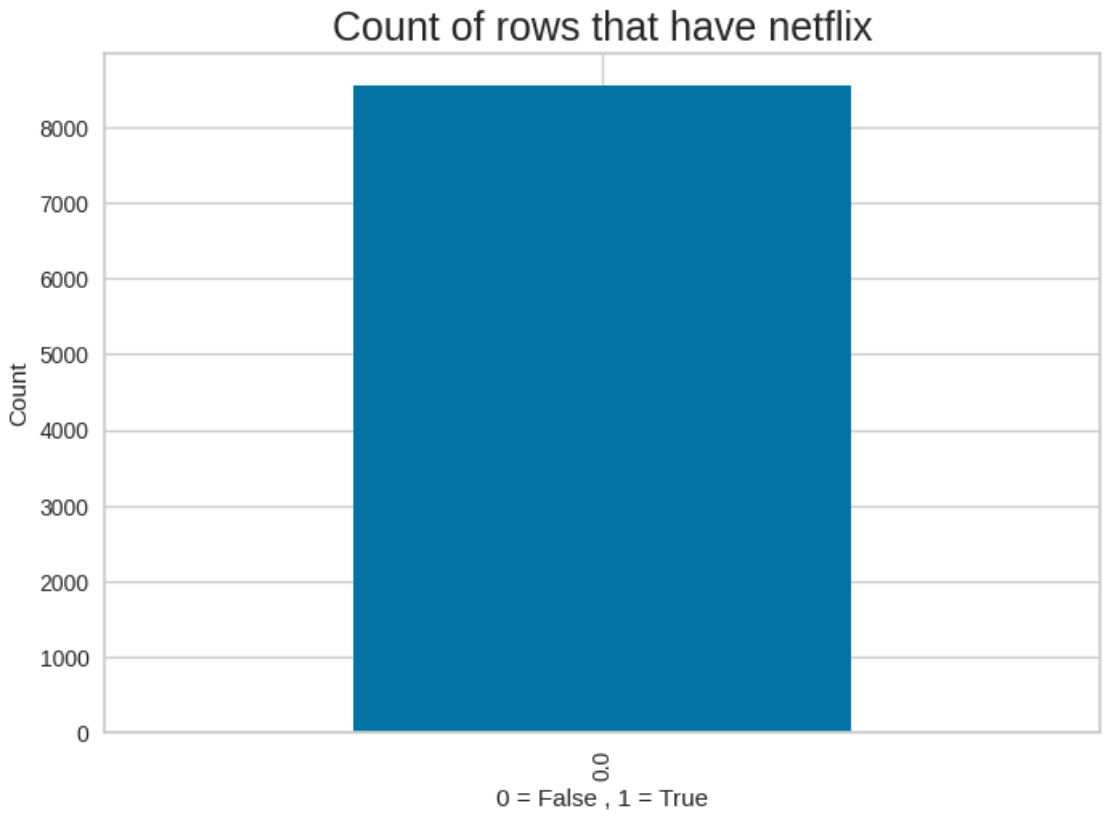
[ ]:
```
label = "facebook"
title = "Count of rows that have facebook"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



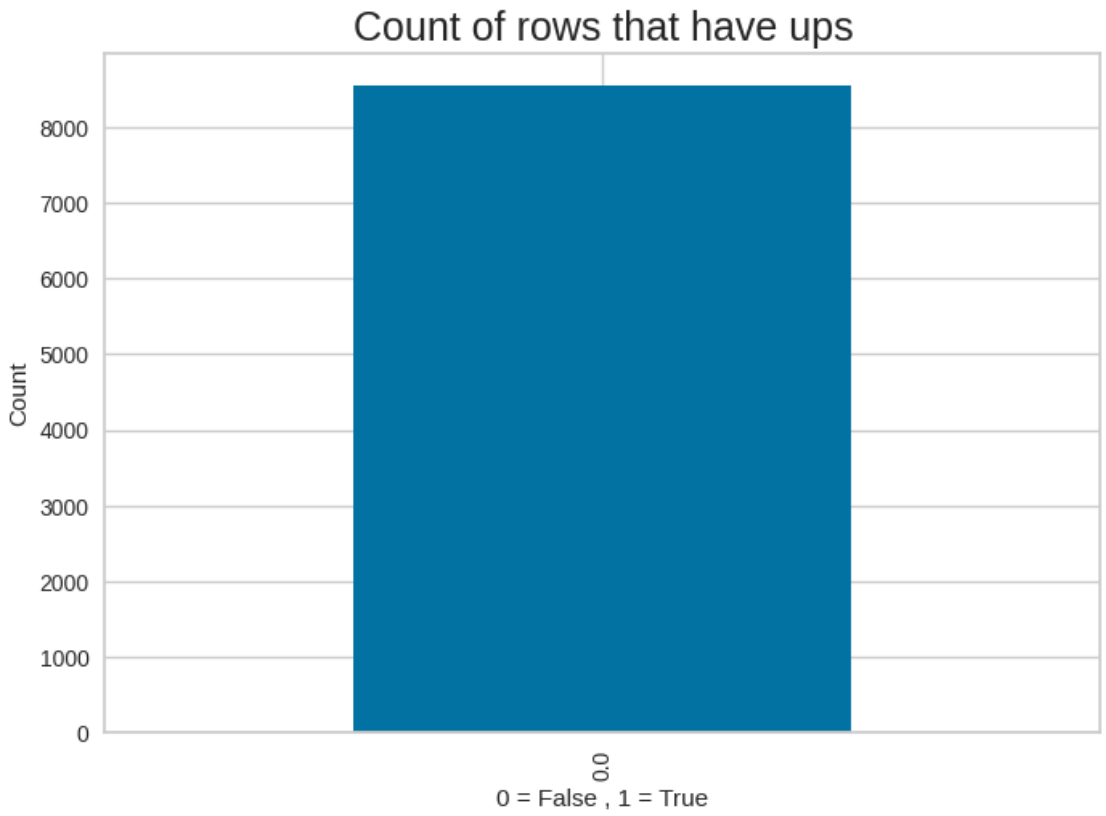
[ ]:
```
label = "steam"
title = "Count of rows that have steam"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



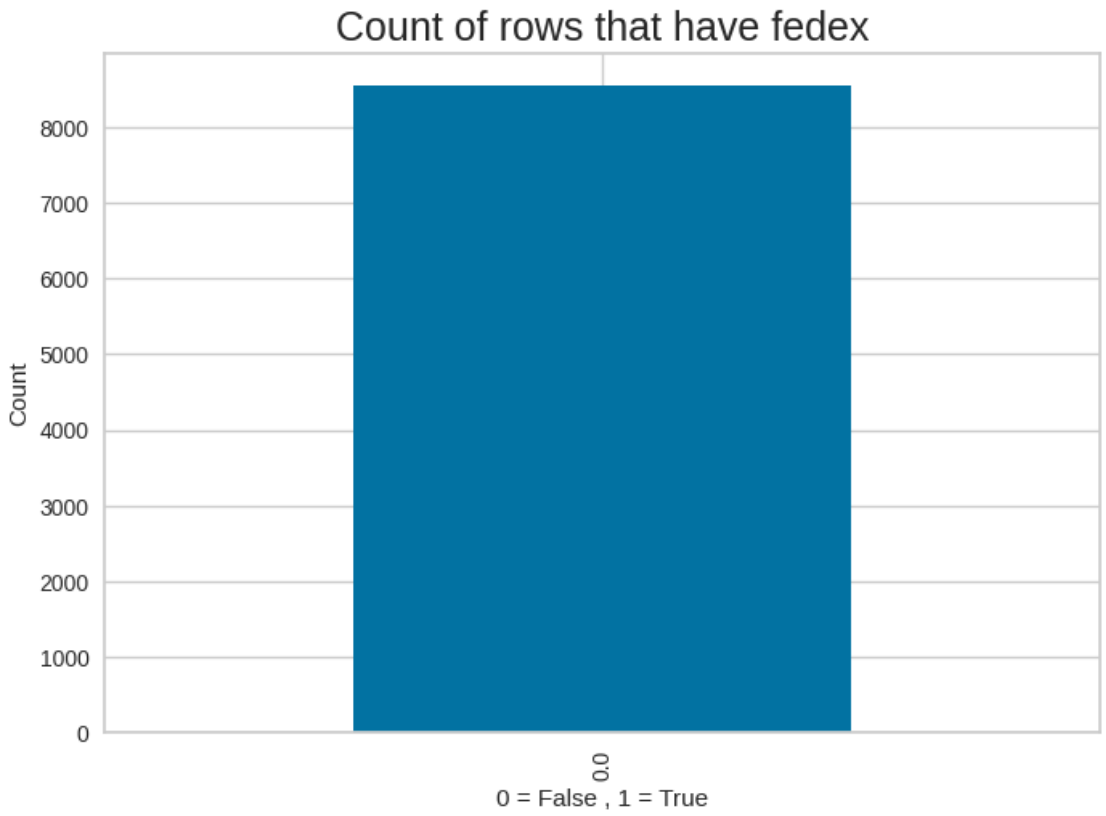
[ ]:
```
label = "netflix"
title = "Count of rows that have netflix"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



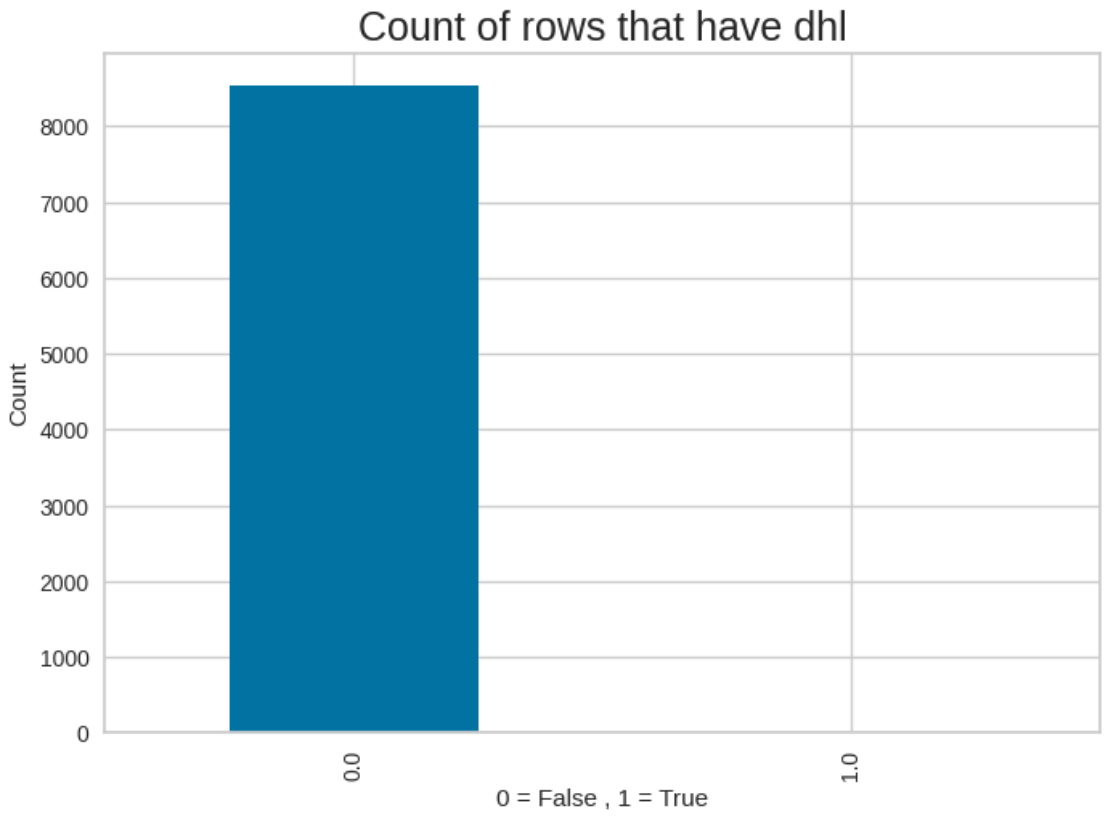
[ ]:
```
label = "ups"
title = "Count of rows that have ups"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



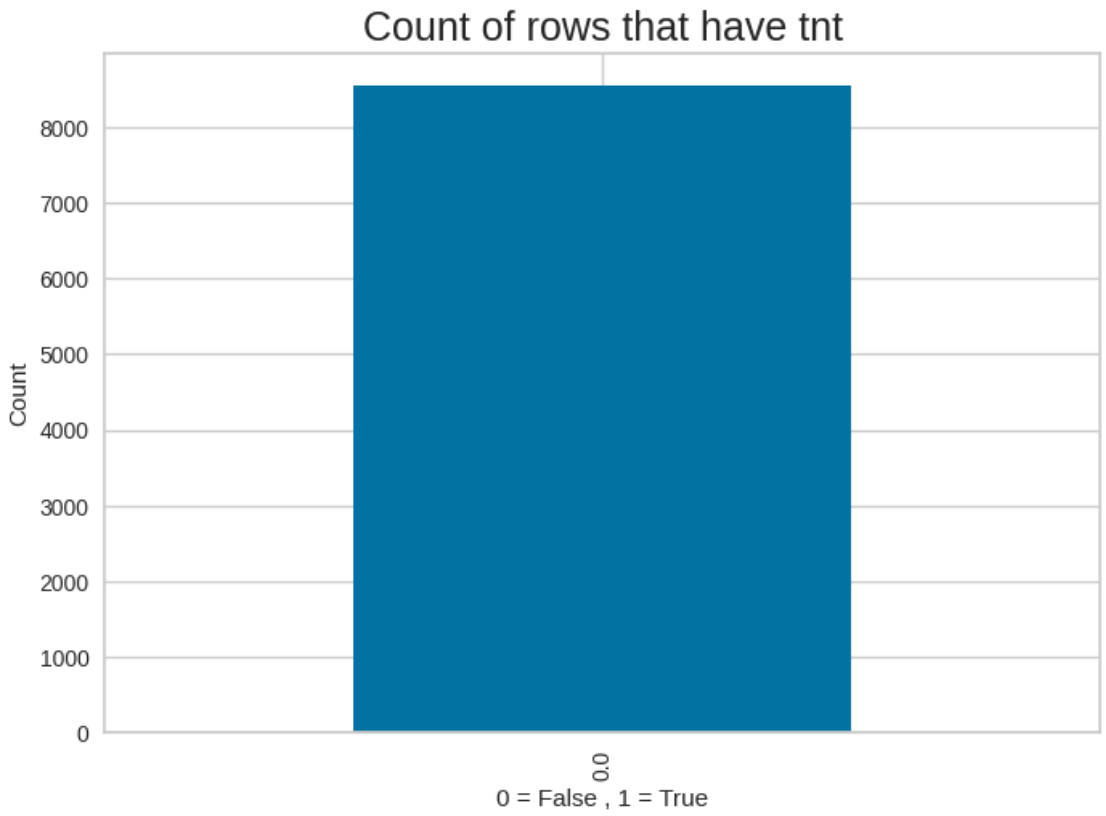
[ ]:
```
label = "fedex"
title = "Count of rows that have fedex"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



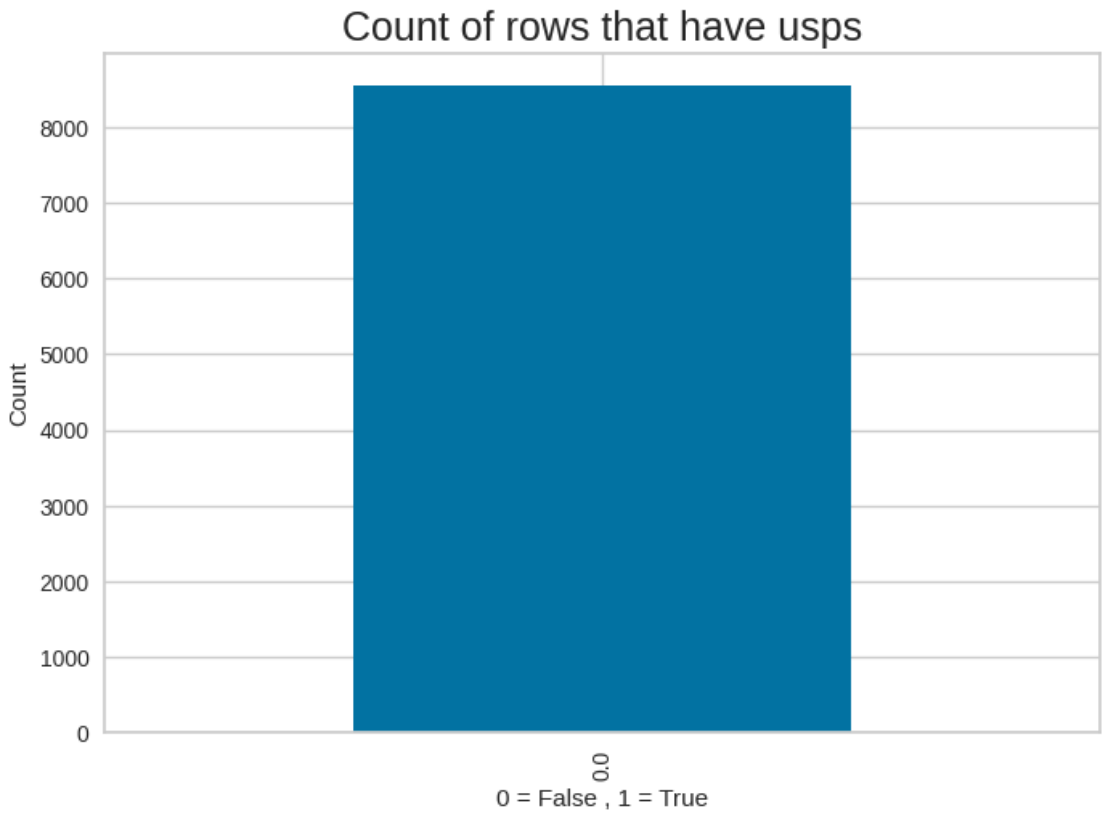
[ ]:
```
label = "dhl"
title = "Count of rows that have dhl"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



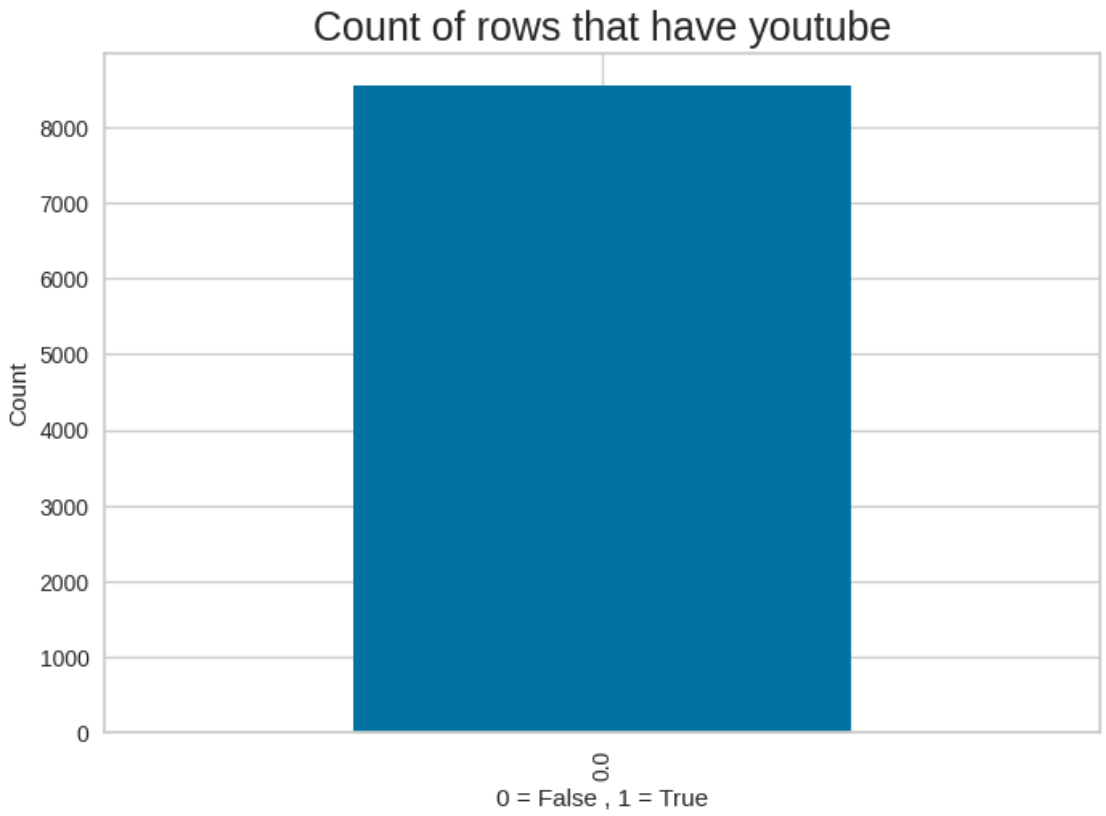
[ ]:
```
label = "tnt"
title = "Count of rows that have tnt"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



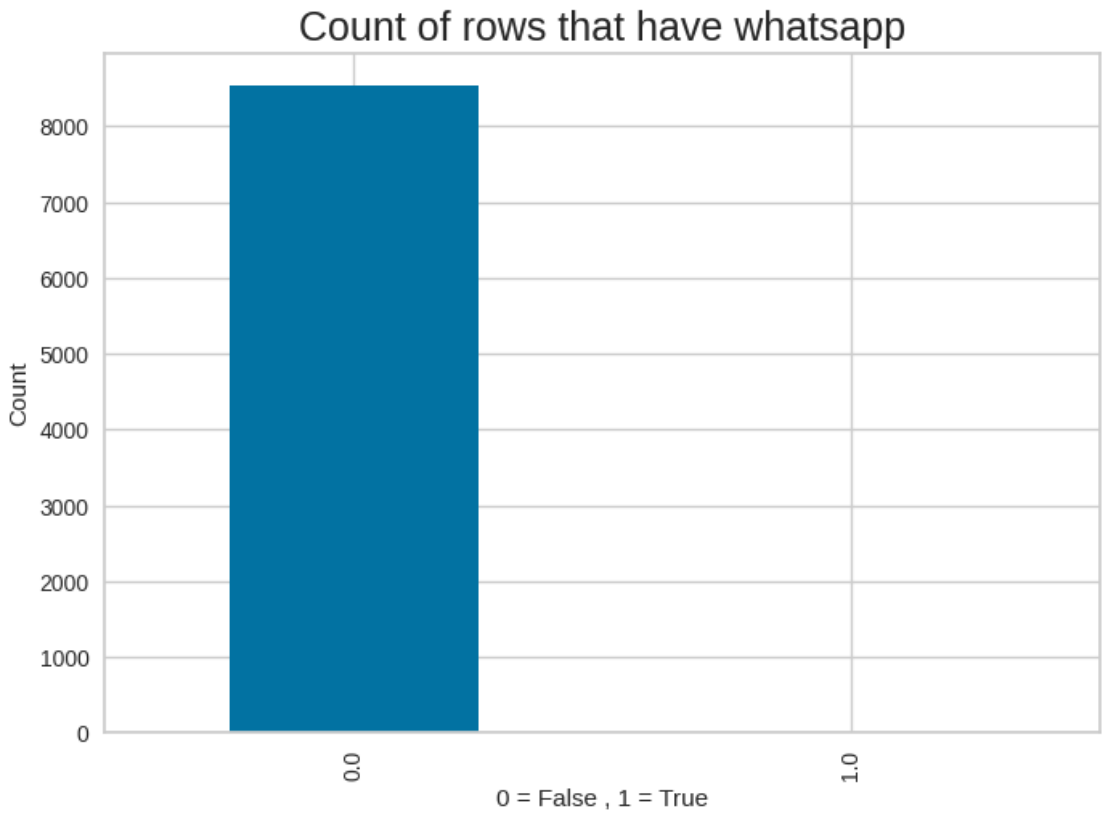
[ ]:
```
label = "usps"
title = "Count of rows that have usps"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



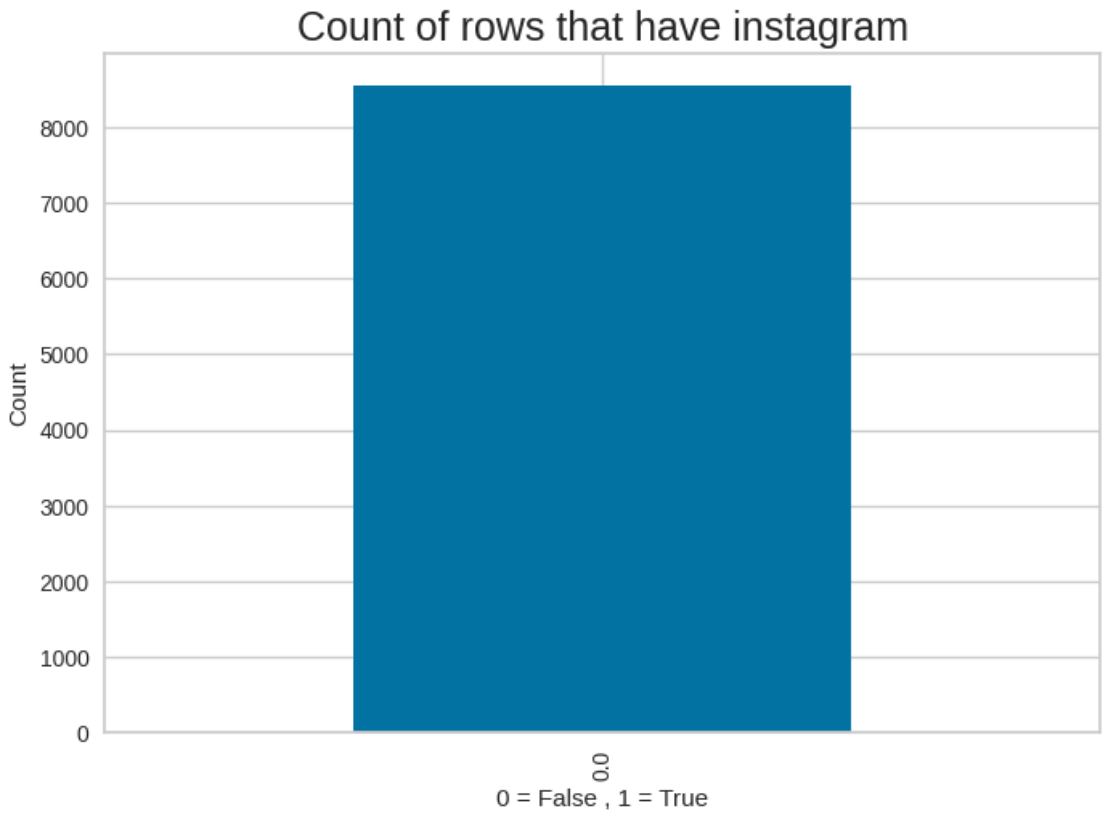
[ ]:
```
label = "youtube"
title = "Count of rows that have youtube"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



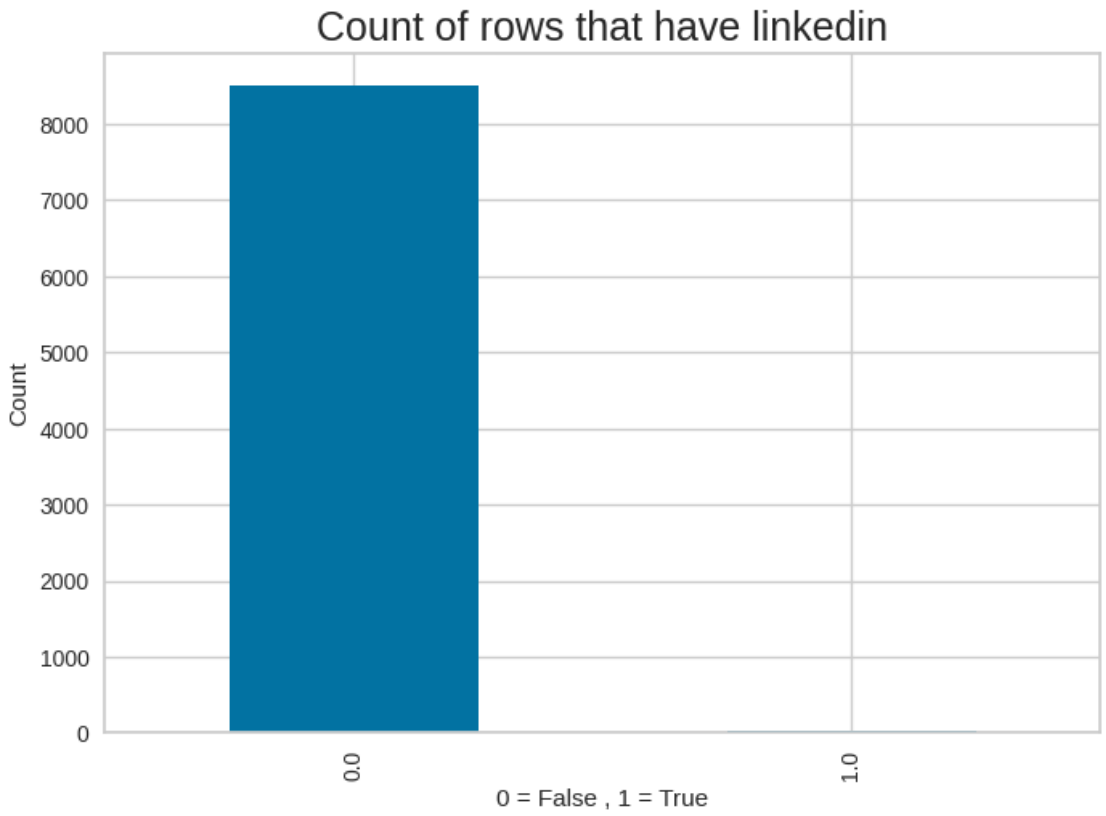
[ ]:
```
label = "whatsapp"
title = "Count of rows that have whatsapp"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



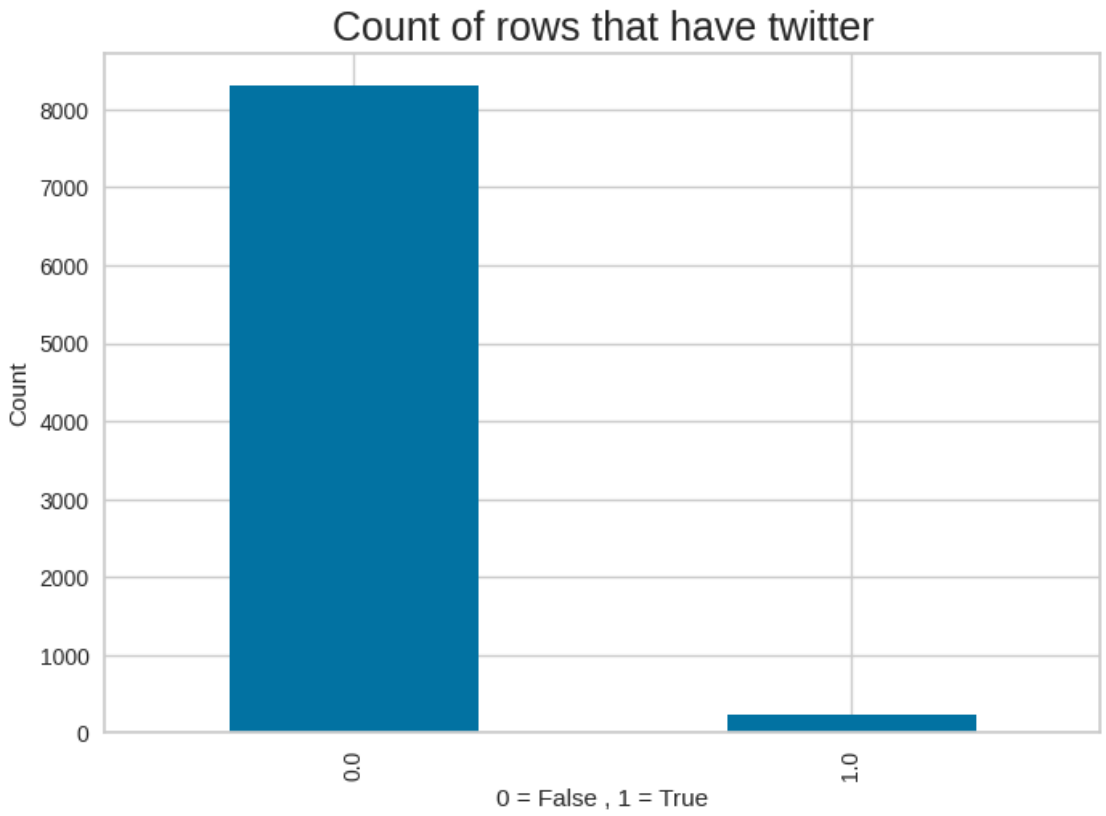
[ ]:
```
label = "instagram"
title = "Count of rows that have instagram"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



[ ]:
```
label = "linkedin"
title = "Count of rows that have linkedin"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



[ ]:
```
label = "twitter"
title = "Count of rows that have twitter"
y_label = "Count"
x_label = "0 = False , 1 = True"
display_value_counts(label = label, title = title, x_label = x_label, y_label =␣
↪y_label)
```



Summary of Value Counts:

By reviewing the value counts, we notice that there is a larger number of malicious samples than benign ones, which may require us to consider when generating our classifier to have a more balanced distribution.

We also discovered that after the initial filtering step where we limited the number of tokens, the following features now have a constant 0: * jpmorgan_chase * hsbc * deutsche_bank * citibank * paypal * scotiabank * samsung * steam * netflix * ups * fedex * tnt * usps * instagram

We are simply going to remove all these columns.

[ ]:
```
# Remove columns that have a constant value 0 (after initial filtering)

# jpmorgan_chase
# hsbc
# deutsche_bank
# citibank
# paypal
# scotiabank
# samsung
# steam
# netflix
```

```
# ups
# fedex
# tnt
# usps
# instagram

drop_column('jpmorgan_chase')
drop_column('hsbc')
drop_column('deutsche_bank')
drop_column('rbc')
drop_column('paypal')
drop_column('scotiabank')
drop_column('samsung')
drop_column('steam')
drop_column('netflix')
drop_column('ups')
drop_column('fedex')
drop_column('tnt')
drop_column('usps')
drop_column('instagram')
```

Filter for Malicious

[ ]:
```
df_text_clean = df[(df['classification'] == 'malicious')][['text_clean']].copy()
df_text_clean['text_clean'] = df_text_clean['text_clean'].apply(remove_urls)
df_text_clean['text_clean'] = df_text_clean['text_clean'].apply(unescape_stuff)
df_text_clean['text_clean'] = df_text_clean['text_clean'].apply(remove_symbols)
df_text_clean['text_clean'] = df_text_clean['text_clean'].
  ↪apply(unify_whitespaces)
```

```
<ipython-input-6-db31889a9805>:4: MarkupResemblesLocatorWarning: The input looks
more like a filename than markup. You may want to open this file and pass the
filehandle into Beautiful Soup.
  soup = BeautifulSoup(unescape(x), 'lxml')
```

Remove english stopwords

[ ]:
```
df_text_clean['text_clean'] = df_text_clean['text_clean'].
  ↪apply(remove_stopwords)
```

Extract the Malicious unigrams

[ ]:
```
df_text_clean["unigram_text"] = df_text_clean["text_clean"].apply(get_ngrams, n␣
  ↪= 1)
```

Extract the Malicious bigrams

[ ]:
```
df_text_clean["bigram_text"] = df_text_clean["text_clean"].apply(get_ngrams, n␣
  ↪= 2)
```

Extract the Malicious trigrams

```
[ ]: df_text_clean["trigram_text"] = df_text_clean["text_clean"].apply(get_ngrams, n␣
      ↪= 3)
```

1.8.4 Creating a List of Tokens from a List of Documents

```
[ ]: # transform list of documents into a single list of tokens
     unigram_tokens = df_text_clean.unigram_text.map(my_tokenizer).sum()
     bigram_tokens = df_text_clean.bigram_text.map(my_tokenizer).sum()
     trigram_tokens = df_text_clean.trigram_text.map(my_tokenizer).sum()
```

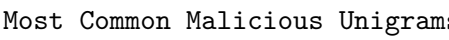
1.8.5 Most Common Malicious Words

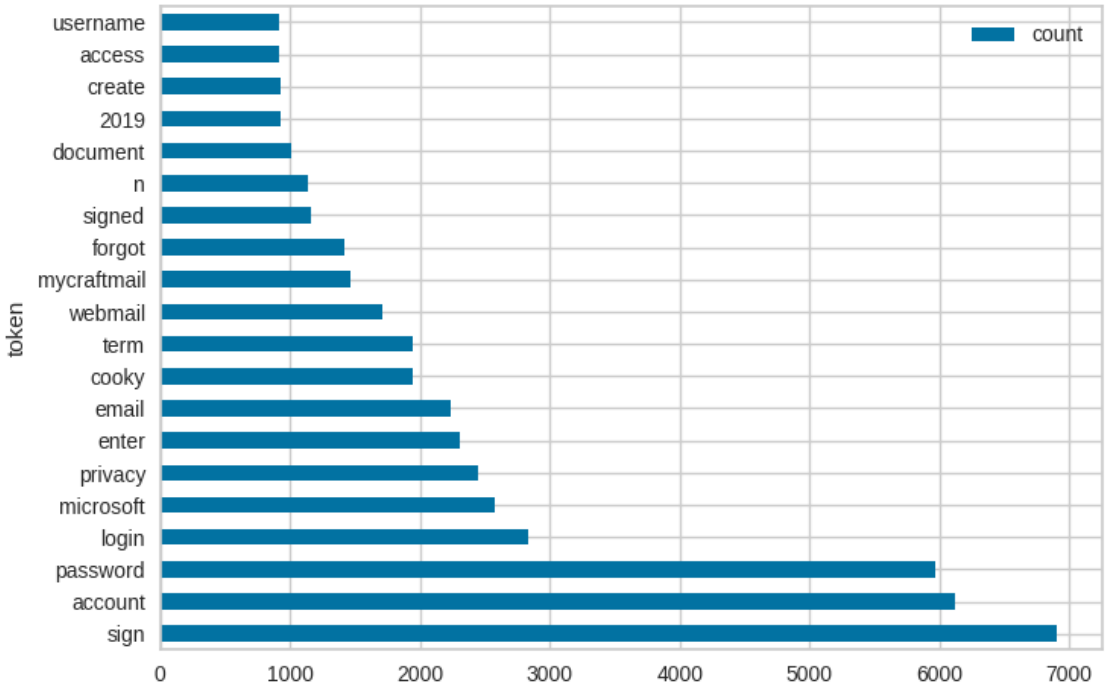
Most Common Malicious Unigrams

```
[ ]: unigram_counter = Counter(unigram_tokens)
     unigram_counter.most_common(20)
```

```
[ ]: [('sign', 6907),
      ('account', 6118),
      ('password', 5965),
      ('login', 2833),
      ('microsoft', 2571),
      ('privacy', 2447),
      ('enter', 2312),
      ('email', 2232),
      ('cooky', 1947),
      ('term', 1940),
      ('webmail', 1714),
      ('mycraftmail', 1465),
      ('forgot', 1417),
      ('signed', 1155),
      ('n', 1143),
      ('document', 1010),
      ('2019', 927),
      ('create', 923),
      ('access', 917),
      ('username', 913)]
```

```
[ ]: title = 'Most Common Malicious Unigrams'
     counter = unigram_counter
     display_most_common_ngrams(title, counter)
```





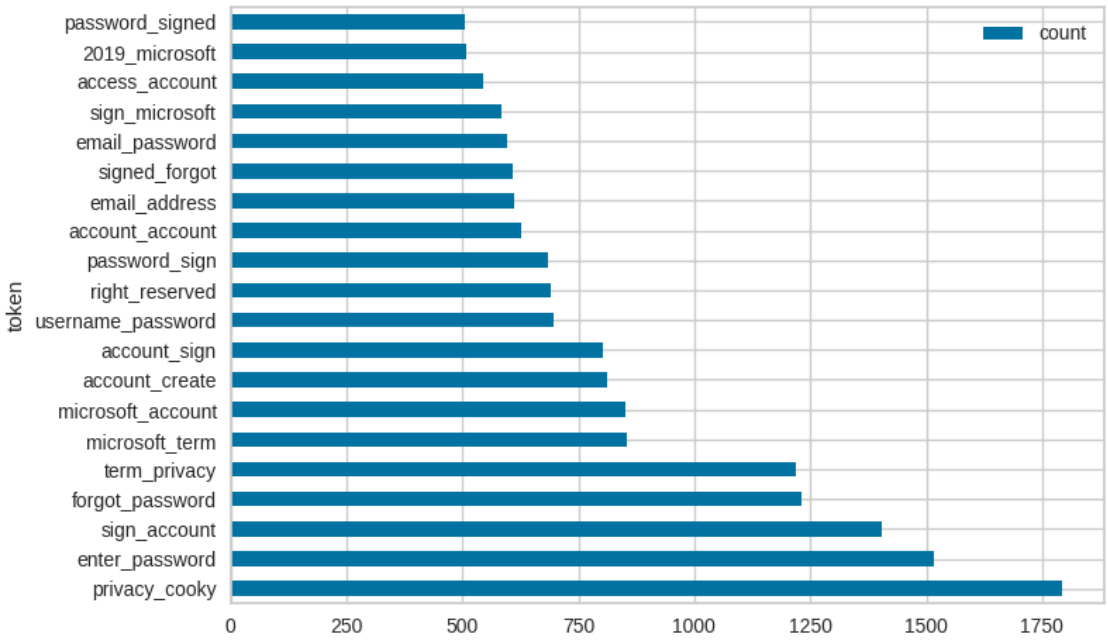
Most Common Malicious bigrams

[ ]:
```
bigram_counter = Counter(bigram_tokens)
bigram_counter.most_common(20)
```

[ ]:
```
[('privacy_cooky', 1793),
 ('enter_password', 1518),
 ('sign_account', 1404),
 ('forgot_password', 1231),
 ('term_privacy', 1218),
 ('microsoft_term', 855),
 ('microsoft_account', 852),
 ('account_create', 812),
 ('account_sign', 803),
 ('username_password', 696),
 ('right_reserved', 691),
 ('password_sign', 686),
 ('account_account', 627),
 ('email_address', 611),
 ('signed_forgot', 609),
 ('email_password', 597),
 ('sign_microsoft', 584),
 ('access_account', 544),
 ('2019_microsoft', 510),
 ('password_signed', 505)]
```

```
[ ]: title = 'Most Common Malicious Bigrams'
     counter = bigram_counter
     display_most_common_ngrams(title, counter)
```

Most Common Malicious Bigrams



Most Common Malicious trigrams

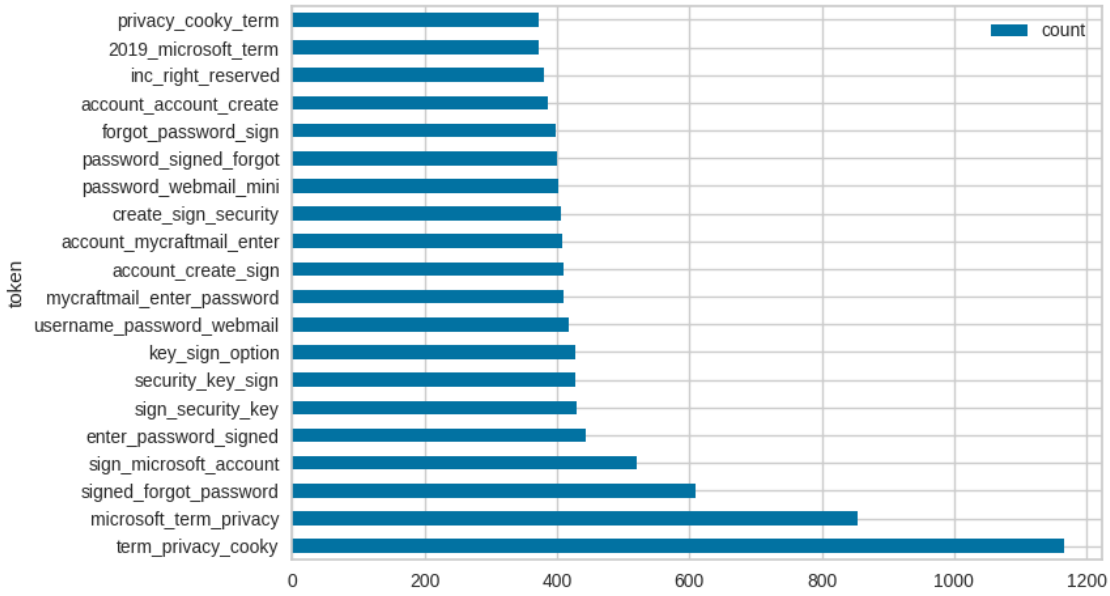
```
[ ]: trigram_counter = Counter(trigram_tokens)
     trigram_counter.most_common(20)
```

```
[ ]: [('term_privacy_cooky', 1166),
      ('microsoft_term_privacy', 854),
      ('signed_forgot_password', 609),
      ('sign_microsoft_account', 521),
      ('enter_password_signed', 443),
      ('sign_security_key', 429),
      ('security_key_sign', 428),
      ('key_sign_option', 428),
      ('username_password_webmail', 418),
      ('mycraftmail_enter_password', 410),
      ('account_create_sign', 409),
      ('account_mycraftmail_enter', 407),
      ('create_sign_security', 406),
      ('password_webmail_mini', 403),
      ('password_signed_forgot', 401),
```

```
 ('forgot_password_sign', 399),
 ('account_account_create', 386),
 ('inc_right_reserved', 381),
 ('2019_microsoft_term', 373),
 ('privacy_cooky_term', 372)]
```

[ ]:
```
title = 'Most Common Malicious Trigrams'
counter = trigram_counter
display_most_common_ngrams(title, counter)
```

```
Most Common Malicious Trigrams
```



Display Malicious Unigrams WordCloud

[ ]:
```
title = "Malicious unigrams"
unigram_wordcloud = get_wordcloud(get_ngram_string('unigram_text'))
plot_wordcloud(title, unigram_wordcloud)
```

Malicious unigrams

Display Malicious Bigrams WordCloud

```
[ ]: title = "Malicious bigrams"
bigram_wordcloud = get_wordcloud(get_ngram_string('bigram_text'))
plot_wordcloud(title, bigram_wordcloud)
```

Display Malicious Trigrams WordCloud

```
[ ]: title = "Malicious trigrams"
trigram_wordcloud = get_wordcloud(get_ngram_string('trigram_text'))
plot_wordcloud(title, trigram_wordcloud)
```

Filter for Benign

```
[ ]: df_text_clean = df[(df['classification'] == 'benign')][['text_clean']].copy()
     df_text_clean['text_clean'] = df_text_clean['text_clean'].apply(remove_urls)
     df_text_clean['text_clean'] = df_text_clean['text_clean'].apply(unescape_stuff)
     df_text_clean['text_clean'] = df_text_clean['text_clean'].apply(remove_symbols)
     df_text_clean['text_clean'] = df_text_clean['text_clean'].
       ↪apply(unify_whitespaces)
```

```
<ipython-input-6-db31889a9805>:4: MarkupResemblesLocatorWarning: The input looks
more like a filename than markup. You may want to open this file and pass the
filehandle into Beautiful Soup.
  soup = BeautifulSoup(unescape(x), 'lxml')
```

Remove english stopwords

```
[ ]: df_text_clean['text_clean'] = df_text_clean['text_clean'].
       ↪apply(remove_stopwords)
```

Extract the Benign Unigrams

```
[ ]: df_text_clean["unigram_text"] = df_text_clean["text_clean"].apply(get_ngrams, n␣
       ↪= 1)
```

Extract the Benign bigrams

```
[ ]: df_text_clean["bigram_text"] = df_text_clean["text_clean"].apply(get_ngrams, n␣
      ↪= 2)
```

Extract the Benign trigrams

```
[ ]: df_text_clean["trigram_text"] = df_text_clean["text_clean"].apply(get_ngrams, n␣
      ↪= 3)
```

1.8.6 Creating a List of Tokens from a List of Documents

```
[ ]: # transform list of documents into a single list of tokens
     unigram_tokens = df_text_clean.unigram_text.map(my_tokenizer).sum()
     bigram_tokens = df_text_clean.bigram_text.map(my_tokenizer).sum()
     trigram_tokens = df_text_clean.trigram_text.map(my_tokenizer).sum()
```

1.8.7 Most Common Benign Words

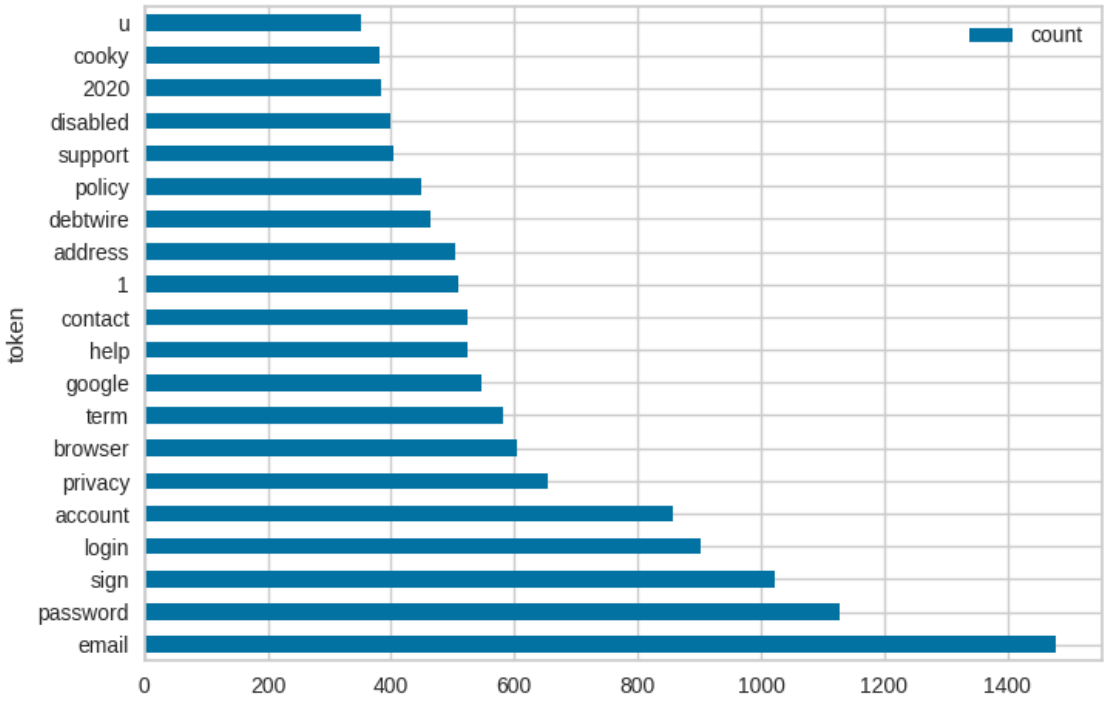
Most Common Benign Unigrams

```
[ ]: unigram_counter = Counter(unigram_tokens)
     unigram_counter.most_common(20)
```

```
[ ]: [('email', 1479),
      ('password', 1127),
      ('sign', 1023),
      ('login', 902),
      ('account', 858),
      ('privacy', 655),
      ('browser', 604),
      ('term', 581),
      ('google', 547),
      ('help', 524),
      ('contact', 523),
      ('1', 510),
      ('address', 503),
      ('debtwire', 465),
      ('policy', 448),
      ('support', 405),
      ('disabled', 400),
      ('2020', 384),
      ('cooky', 382),
      ('u', 350)]
```

```
[ ]: title = 'Most Common Benign Unigrams'
     counter = unigram_counter
     display_most_common_ngrams(title, counter)
```

Most Common Benign Unigrams



Most Common Benign bigrams

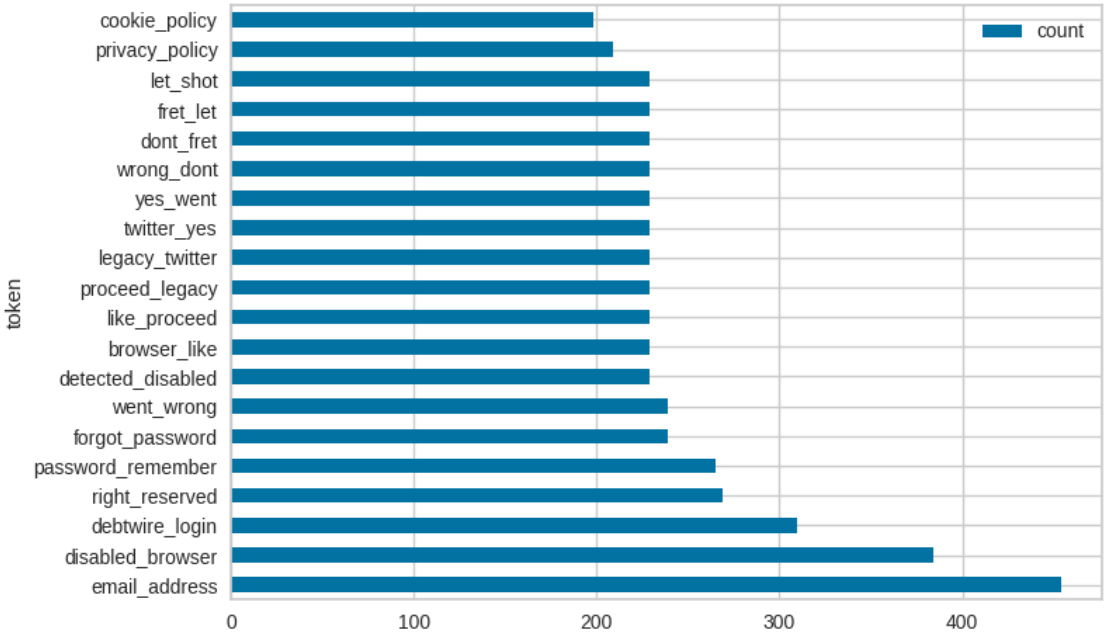
[ ]:
```
bigram_counter = Counter(bigram_tokens)
bigram_counter.most_common(20)
```

[ ]:
```
[('email_address', 454),
 ('disabled_browser', 384),
 ('debtwire_login', 310),
 ('right_reserved', 269),
 ('password_remember', 265),
 ('forgot_password', 239),
 ('went_wrong', 239),
 ('detected_disabled', 229),
 ('browser_like', 229),
 ('like_proceed', 229),
 ('proceed_legacy', 229),
 ('legacy_twitter', 229),
 ('twitter_yes', 229),
 ('yes_went', 229),
 ('wrong_dont', 229),
 ('dont_fret', 229),
 ('fret_let', 229),
 ('let_shot', 229),
```

```
 ('privacy_policy', 209),
 ('cookie_policy', 198)]
```

[ ]:
```
title = 'Most Common Benign Bigrams'
counter = bigram_counter
display_most_common_ngrams(title, counter)
```

Most Common Benign Bigrams



Most Common Benign trigrams

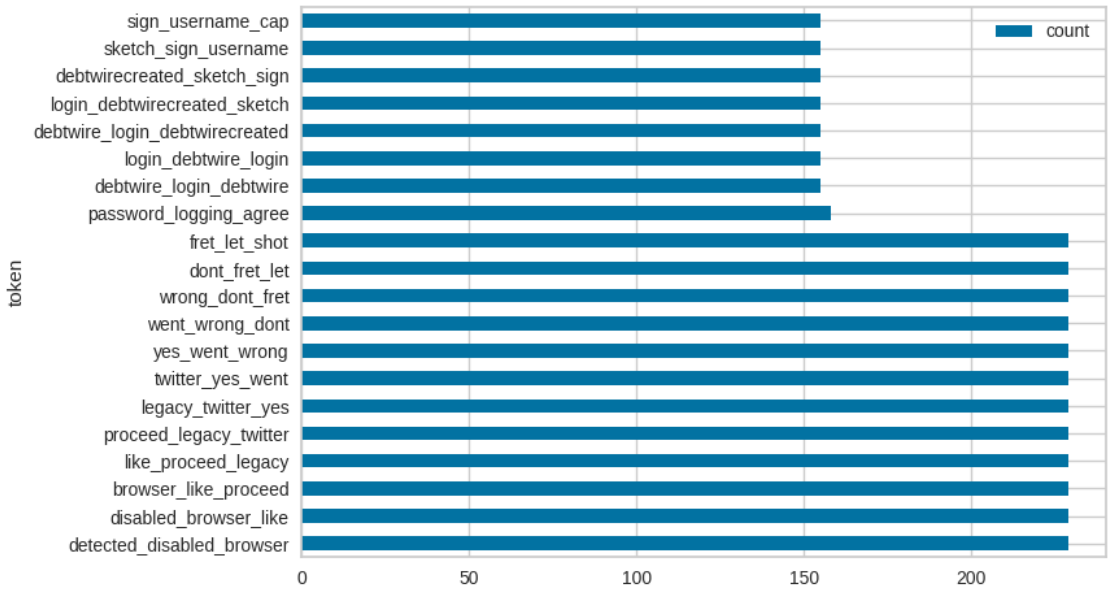
[ ]:
```
trigram_counter = Counter(trigram_tokens)
trigram_counter.most_common(20)
```

[ ]:
```
[('detected_disabled_browser', 229),
 ('disabled_browser_like', 229),
 ('browser_like_proceed', 229),
 ('like_proceed_legacy', 229),
 ('proceed_legacy_twitter', 229),
 ('legacy_twitter_yes', 229),
 ('twitter_yes_went', 229),
 ('yes_went_wrong', 229),
 ('went_wrong_dont', 229),
 ('wrong_dont_fret', 229),
 ('dont_fret_let', 229),
 ('fret_let_shot', 229),
```

```
 ('password_logging_agree', 158),
 ('debtwire_login_debtwire', 155),
 ('login_debtwire_login', 155),
 ('debtwire_login_debtwirecreated', 155),
 ('login_debtwirecreated_sketch', 155),
 ('debtwirecreated_sketch_sign', 155),
 ('sketch_sign_username', 155),
 ('sign_username_cap', 155)]
```

[ ]:
```
title = 'Most Common Benign Trigrams'
counter = trigram_counter
display_most_common_ngrams(title, counter)
```

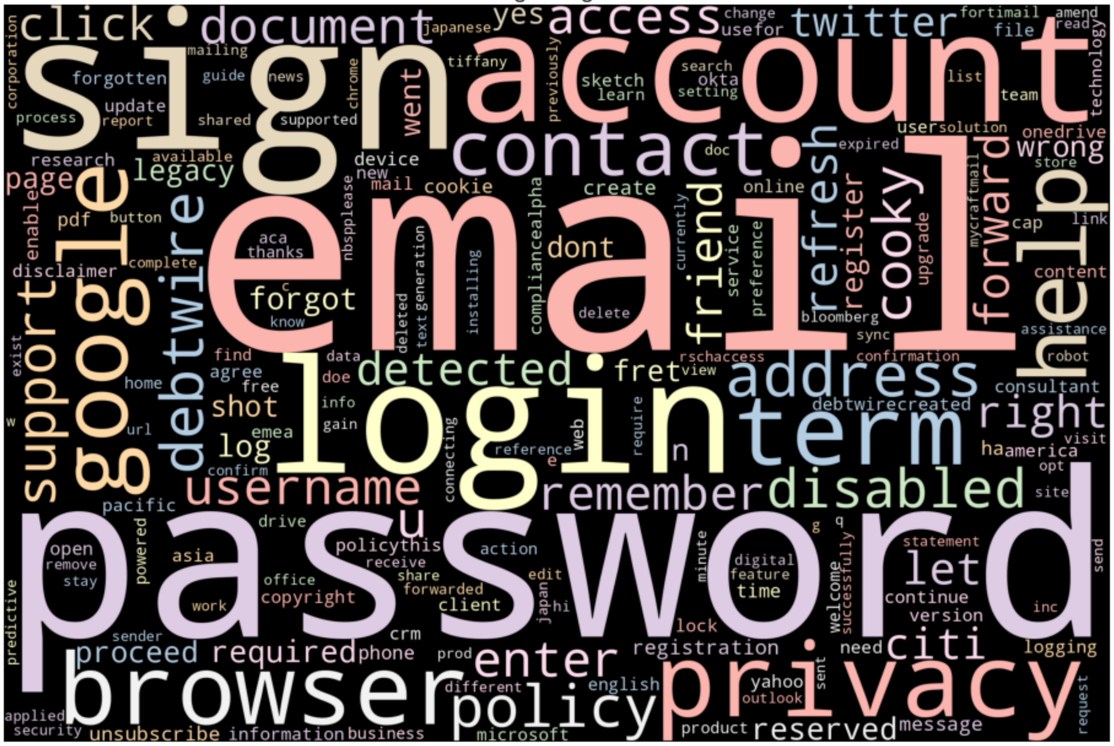
Most Common Benign Trigrams



Display Benign Unigrams WordCloud

[ ]:
```
title = "Benign unigrams"
trigram_wordcloud = get_wordcloud(get_ngram_string('unigram_text'))
plot_wordcloud(title, trigram_wordcloud)
```

Benign unigrams

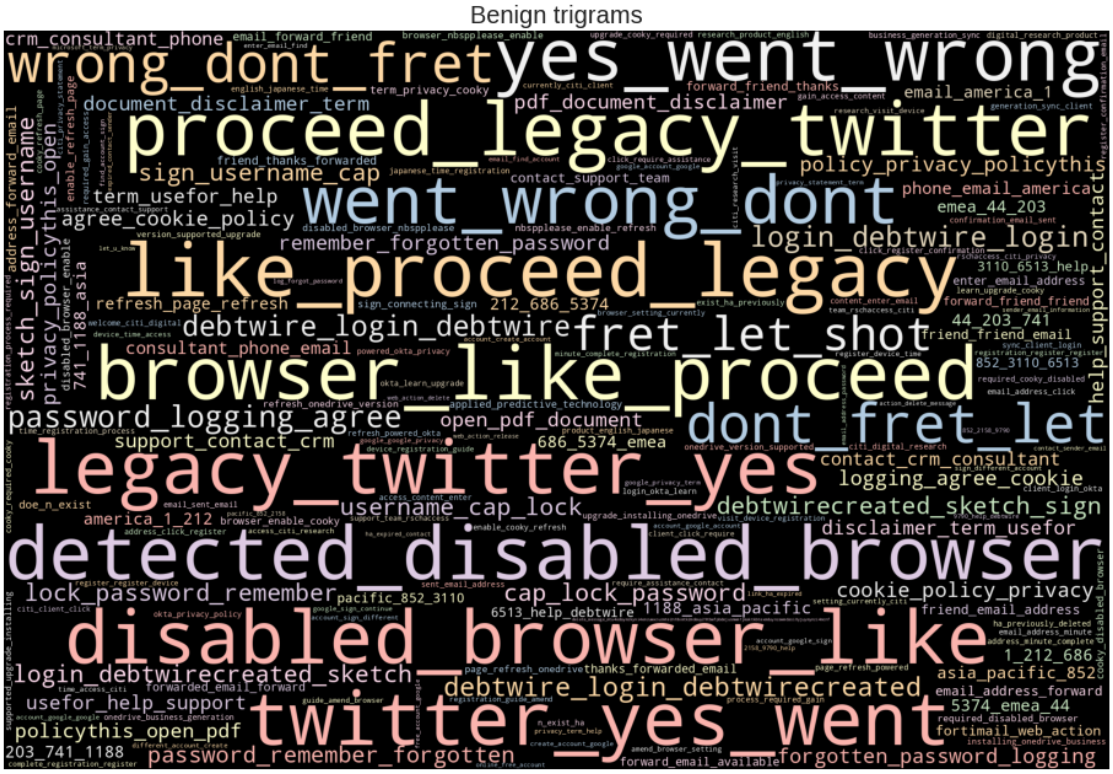


Display Benign Bigrams WordCloud

[ ]:
```
title = "Benign bigrams"
trigram_wordcloud = get_wordcloud(get_ngram_string('bigram_text'))
plot_wordcloud(title, trigram_wordcloud)
```

Display Benign Trigrams WordCloud

```
[ ]: title = "Benign trigrams"
trigram_wordcloud = get_wordcloud(get_ngram_string('trigram_text'))
plot_wordcloud(title, trigram_wordcloud)
```



1.8.8 Text Analysis Summary

When inspecting the top 20 Unigrams the following 8 words appear in both Malicious and Benign lists: * email * password * sign * login * account * privacy * term * cooky

When inspecting the top 20 bigrams the following 3 words appear in both Malicious and Benign lists: * email_address * right_reserved * forgot_password

When inspecting the top 20 trigrams none appear in both lists.

Since it is very difficult to classify an email as Malicious or Benign based on Unigrams, we will perform our classification based on bigrams and trigrams.

1.9 Feature Engineering

1.9.1 Stop Words

After multiple iterations, custom stop words were included from the default English stop words from the `ntlk` framework. These includes:

- **ccTLDs**. Country-Coded Top-Level Domains associated with URLs embedded in some web-pages. While the entire URL is useful, the country code, by itself, is not;
- **HTML tags**. Despite the parser's best efforts, some HTML tags remain embedded in the code. As such, we try to reduce some of the HTML code left behind;
- **Adverbs**. Some adverbs, such as *also* or *may* are not useful to the analysis.

[ ]:
```
# Load the default list of English stop words
default_stop_words = stopwords.words('english')

# Create the classifier
# Pass the complete dataset as data and the featured to be predicted as target
# Add custom stop words here:
custom_stop_words = ["com", "ca", "go", "td", "tr", "px", "co", "uv",
                     "ru", "mx", "also", "use", "abc", "wo", "may", "oo",
                     "javascript", "www", "html", "id", "class", "http",␣
 ↪"https"]

# Append the custom list to the default list of stop words
stop_words = default_stop_words + custom_stop_words
```

1.10 Extracting features from text

[ ]:
```
# Define a custom token pattern that matches only alphabetic characters
pattern = r'\b[A-Za-z]+\b'

# Create a CountVectorizer
# We keep only bigrams and trigrams
# We remove words not within the [0.015, 0.8] frequency
count_vectorizer = CountVectorizer(min_df = 0.015,
                                   max_df = 0.8,
                                   stop_words = stop_words,
                                   max_features = max_words,
                                   ngram_range = (2, 3),
                                   token_pattern = pattern)

# Fit the vectorizer to the text data and transform the data
X = count_vectorizer.fit_transform(df['text_clean'])

df_words = pd.DataFrame(data = X.toarray(), columns = count_vectorizer.
 ↪get_feature_names_out())
df_words.head()
```

[ ]:

	access account	account account	account account create	account another	\
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	

	account another account	account create	account create one	account doe	\
0	0	0	0	0	
1	0	0	0	0	

```
2                       0              0                    0            0
3                       0              0                    0            0
4                       0              0                    0            0

   account doe n  account enter  …  webmail mini  webmail welcome  \
0              0              0  …             0                0
1              0              0  …             0                0
2              0              0  …             0                0
3              0              0  …             0                0
4              0              0  …             0                0

   went wrong  went wrong dont  would like  would like proceed  wrong dont  \
0           0                0           0                    0           0
1           0                0           0                    0           0
2           0                0           0                    0           0
3           0                0           0                    0           0
4           0                0           0                    0           0

   wrong dont fret  yes something  yes something went
0                0              0                   0
1                0              0                   0
2                0              0                   0
3                0              0                   0
4                0              0                   0

[5 rows x 250 columns]
```

1.11 Iterate over all rows and perform NLP

[ ]:
```
# Reset the index of the two DataFrames
df.reset_index(drop = True, inplace = True)
df_words.reset_index(drop = True, inplace = True)

# Concatenate the 2 `DataFrame` to generate the dataset
df = pd.concat([df, df_words], axis = 1)

print(f"The features data frame contains {df.shape[0]} row(s) and {df.shape[1]}␣
  ↪column(s).")
print(f"The word vector contains {df_words.shape[0]} row(s) and {df_words.
  ↪shape[1]} column(s).")
print(f"The resulting dataframe contains {df.shape[0]} row(s) and {df.shape[1]}␣
  ↪column(s).")
```

```
The features data frame contains 8540 row(s) and 274 column(s).
The word vector contains 8540 row(s) and 250 column(s).
The resulting dataframe contains 8540 row(s) and 274 column(s).
```

```
[ ]: df.head()
```

```
[ ]:                       title_clean  img_count  has_form  has_login_form  \
0                        one drive          1         0               0
1                         no title          6         1               1
2  email security : : user account          0         1               1
3          sign in to your account          4         1               1
4                       worldclient          2         1               1

   has_js  js_include_b64  nb_tokens  \
0       1               0         31
1       0               0          7
2       1               0         19
3       0               0         14
4       1               0         13

                                          text_clean classification  \
0  one drive read document please choose email pr…      malicious
1  gjin.jung samsung.com 3c 95338 2017 icp0803424…      malicious
2  email security user account verification neede…      malicious
3  sign account nobody mycraftmail.com enter pass…      malicious
4  worldclient microsoft office verification port…      malicious

   nb_title_entities  …  webmail mini  webmail welcome  went wrong  \
0                0.0  …             0                0           0
1                0.0  …             0                0           0
2                0.0  …             0                0           0
3                0.0  …             0                0           0
4                0.0  …             0                0           0

   went wrong dont  would like  would like proceed  wrong dont  \
0                0           0                   0           0
1                0           0                   0           0
2                0           0                   0           0
3                0           0                   0           0
4                0           0                   0           0

   wrong dont fret  yes something  yes something went
0                0              0                   0
1                0              0                   0
2                0              0                   0
3                0              0                   0
4                0              0                   0

[5 rows x 274 columns]
```

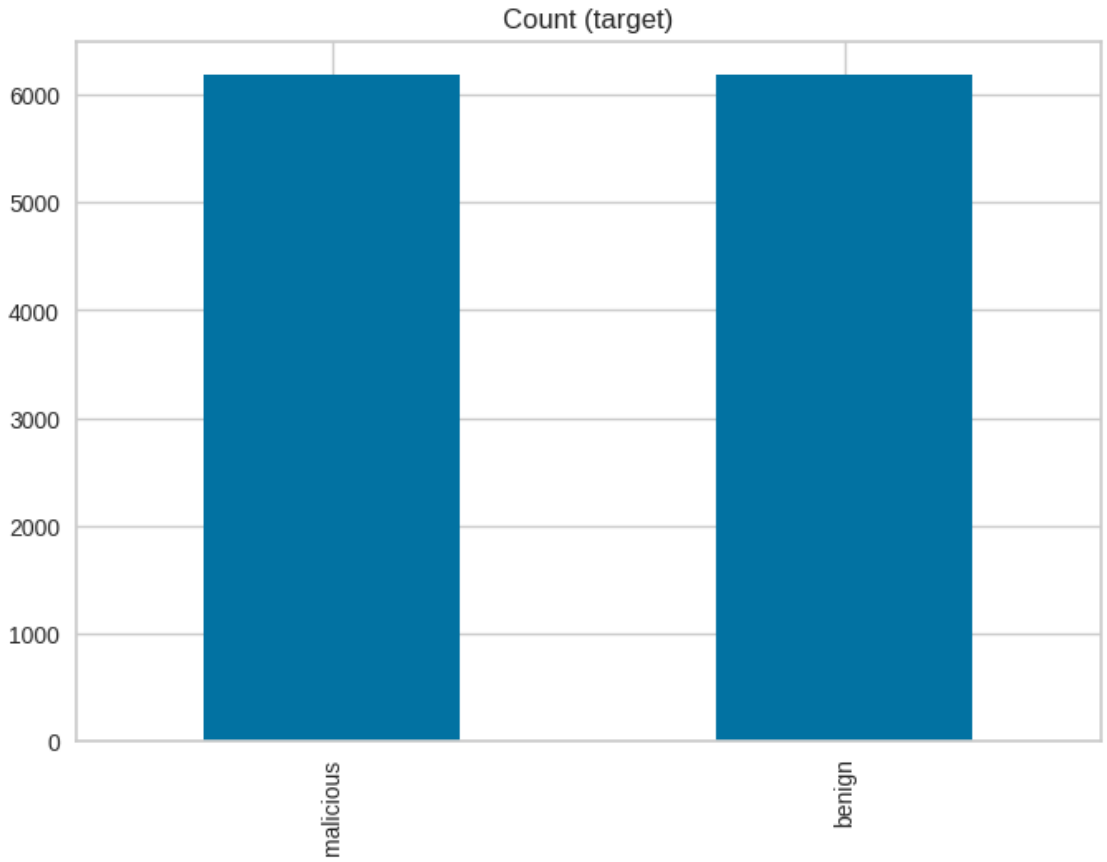
1.12 Modeling

[ ]:
```
# Class count
count_class_0, count_class_1 = df.classification.value_counts()

# Divide by class
df_class_0 = df[df['classification'] == 'malicious']
df_class_1 = df[df['classification'] == 'benign']
```

Random over-sampling

[ ]:
```
df_class_1_over = df_class_1.sample(count_class_0, replace = True)
df_test_over = pd.concat([df_class_0, df_class_1_over], axis = 0)

print('Random over-sampling:')
print(df_test_over.classification.value_counts())

df_test_over.classification.value_counts().plot(kind='bar', title='Count␣
 ↪(target)');
```

```
Random over-sampling:
malicious    6187
benign       6187
Name: classification, dtype: int64
```



1.12.1 Convert non-numeric features to numeric

[ ]:
```
df = handle_non_numerical_data(df)
df.head()
```

[ ]:

	title_clean	img_count	has_form	has_login_form	has_js	js_include_b64	\
0	93	1	0	0	1	0	
1	488	6	1	1	0	0	
2	882	0	1	1	1	0	
3	1183	4	1	1	0	0	
4	634	2	1	1	1	0	

	nb_tokens	text_clean	classification	nb_title_entities	…	\
0	31	166	0	0.0	…	
1	7	2963	0	0.0	…	
2	19	2380	0	0.0	…	
3	14	2992	0	0.0	…	
4	13	2964	0	0.0	…	

```
   webmail mini  webmail welcome  went wrong  went wrong dont  would like  \
0             0                0           0                0           0
1             0                0           0                0           0
2             0                0           0                0           0
3             0                0           0                0           0
4             0                0           0                0           0

   would like proceed  wrong dont  wrong dont fret  yes something  \
0                   0           0                0              0
1                   0           0                0              0
2                   0           0                0              0
3                   0           0                0              0
4                   0           0                0              0

   yes something went
0                   0
1                   0
2                   0
3                   0
4                   0

[5 rows x 274 columns]
```

[ ]:
```
# Split the dataframe into feature and target variables
X = df.drop('classification', axis = 1)
y = df['classification']

# 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,␣
 ↪random_state = 0)
```

1.13 Create classifiers

[ ]:
```
seed = 42
np.random.seed(seed)
```

[ ]:
```
ab = AdaBoostClassifier(random_state = seed)
et = ExtraTreesClassifier(random_state = seed)
gb = GradientBoostingClassifier(random_state = seed)
knn = KNeighborsClassifier()
lr = LogisticRegression(random_state = seed)
rf = RandomForestClassifier(random_state = seed)
rg = RidgeClassifier(random_state = seed)
svc = LinearSVC(random_state = seed)
xgb = xgboost.XGBClassifier()
```

1.13.1 AdaBoost Classifier

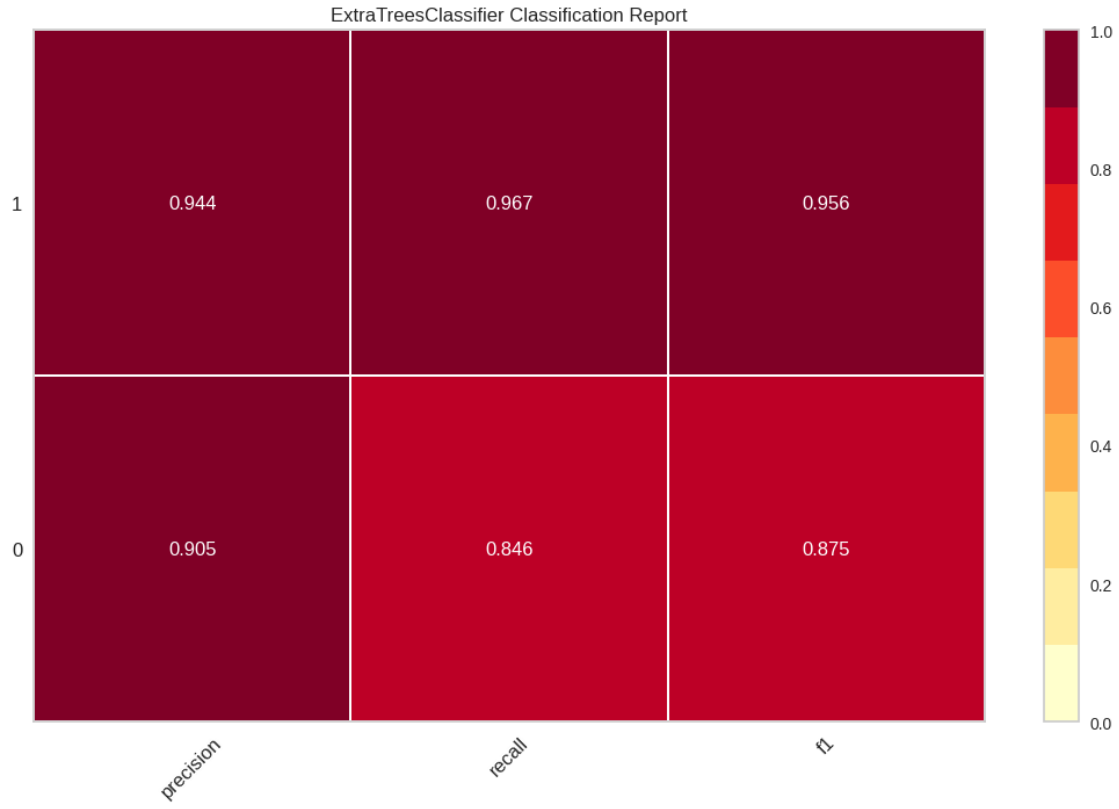
[ ]:
```
model = ab
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 1.076869010925293
Test accuracy : 90.81%
```

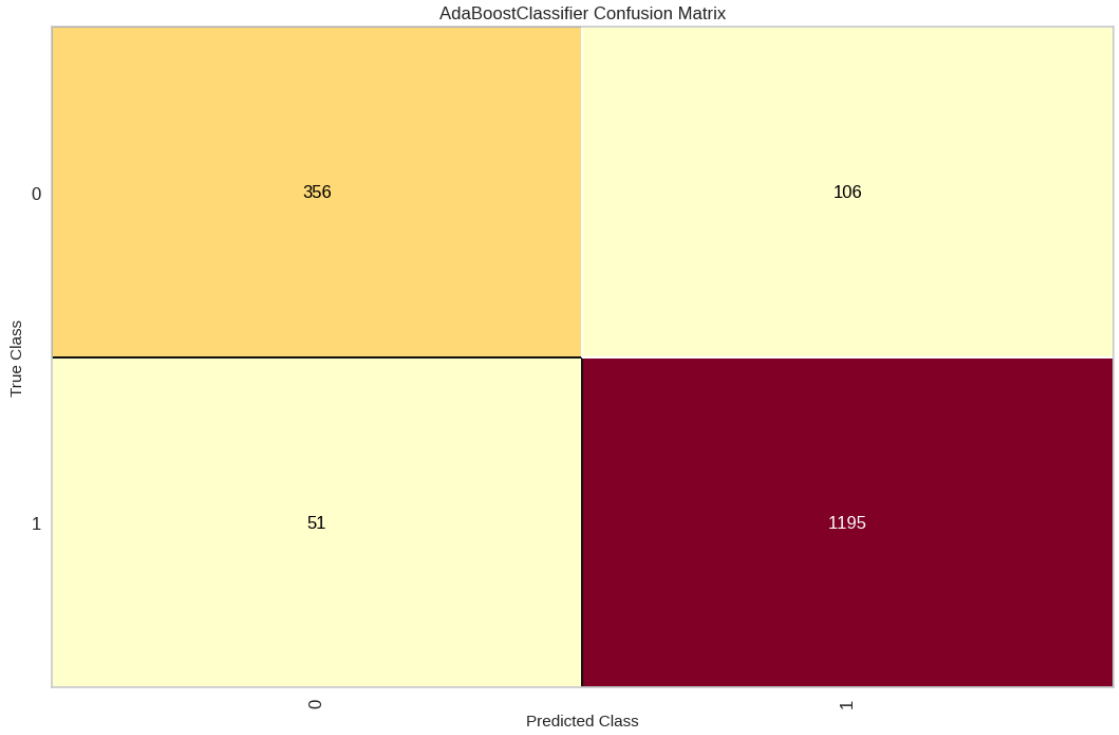
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but AdaBoostClassifier was fitted with feature
names
  warnings.warn(
```



1.13.2 ExtraTrees Classifier

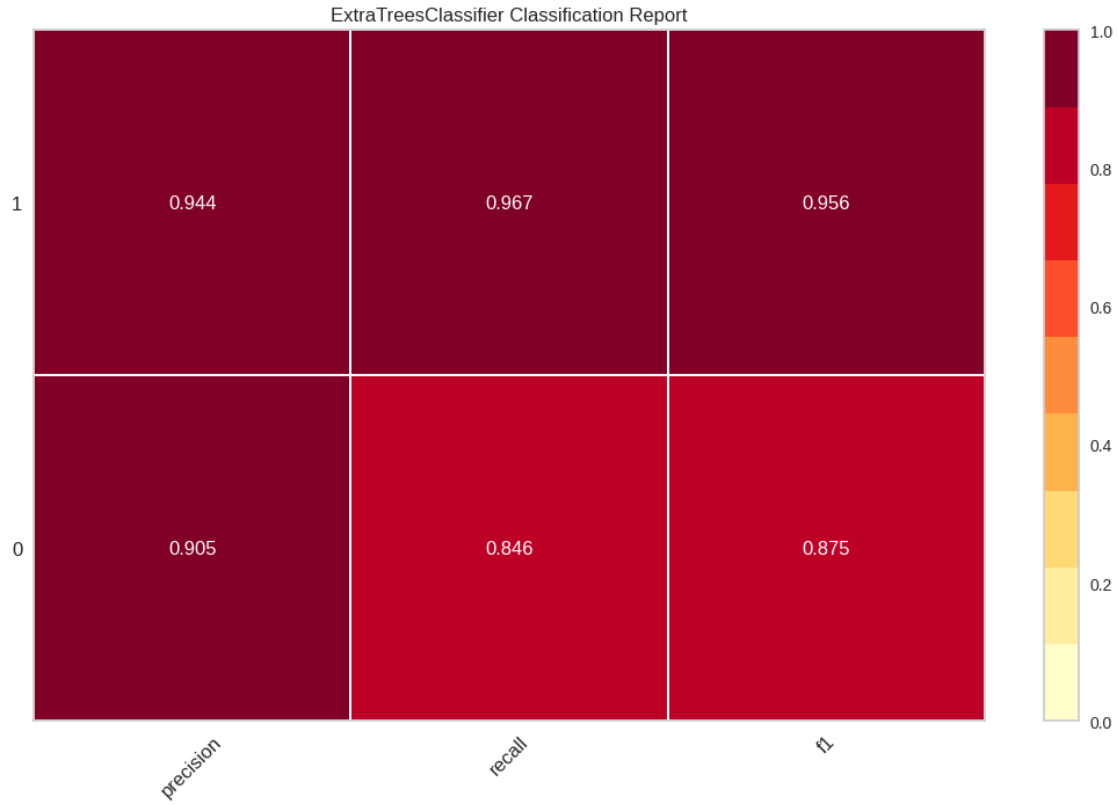
[ ]:
```
model = et
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 1.3714451789855957
Test accuracy : 93.44%
```

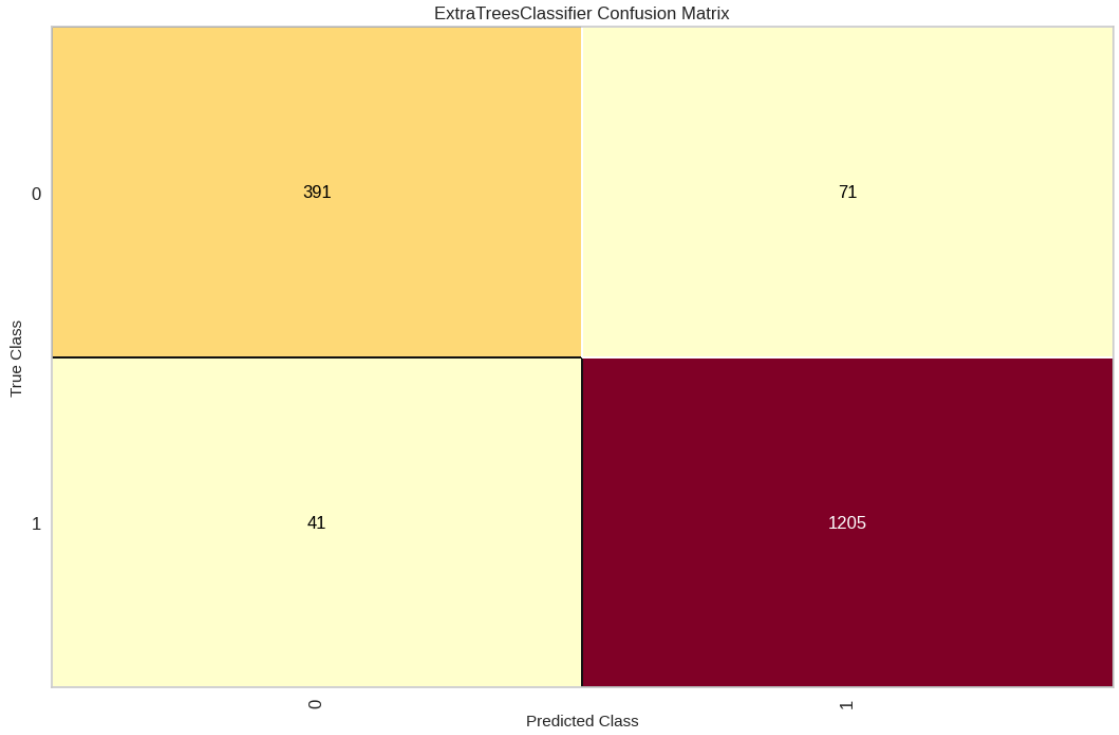
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



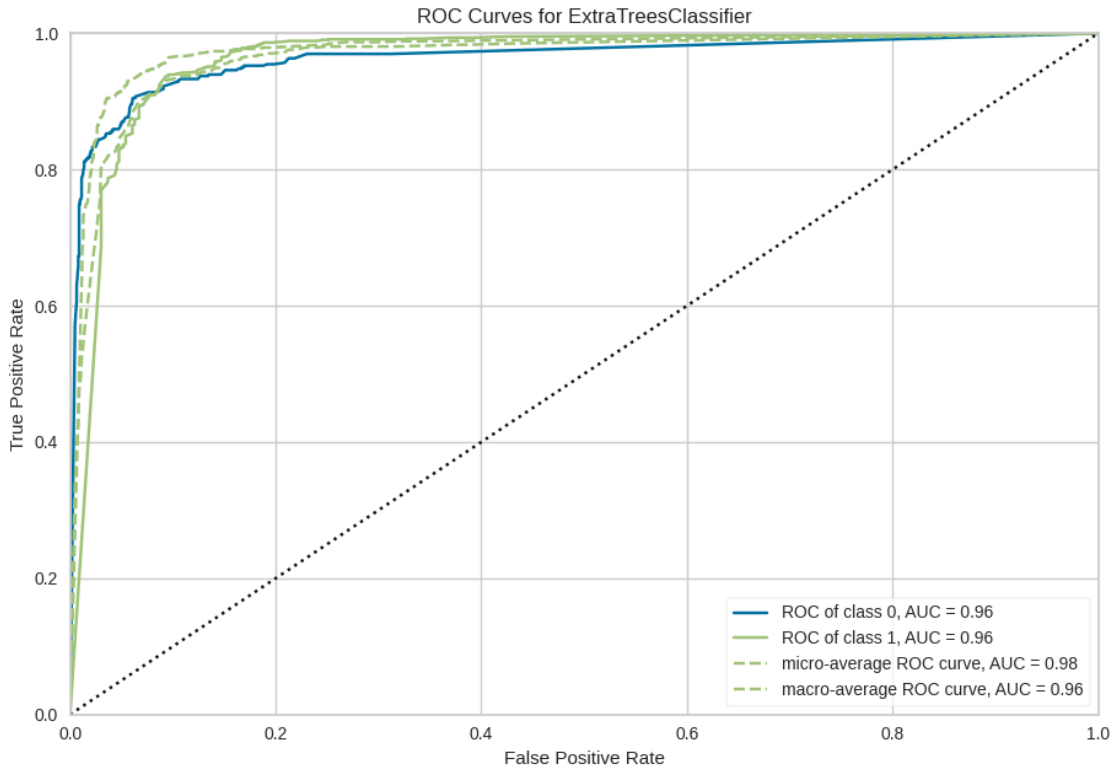
[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but ExtraTreesClassifier was fitted with feature
names
  warnings.warn(
```



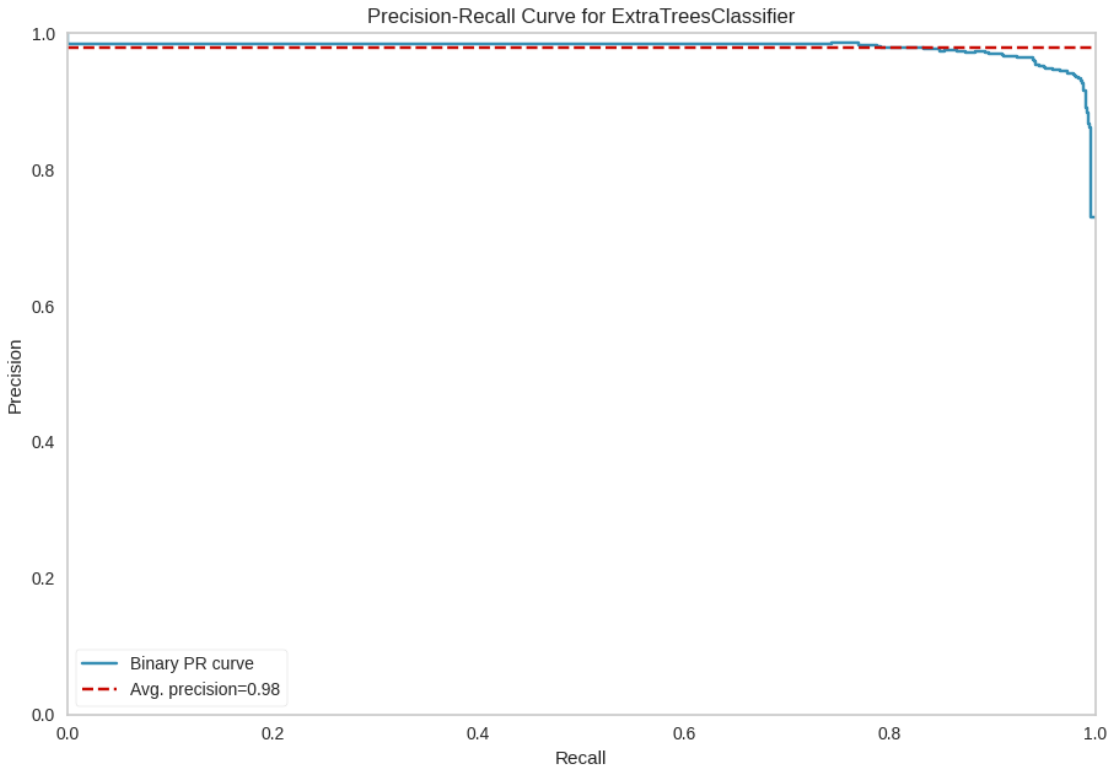
[ ]:
```
display_roc_curve(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but ExtraTreesClassifier was fitted with feature
names
  warnings.warn(
```

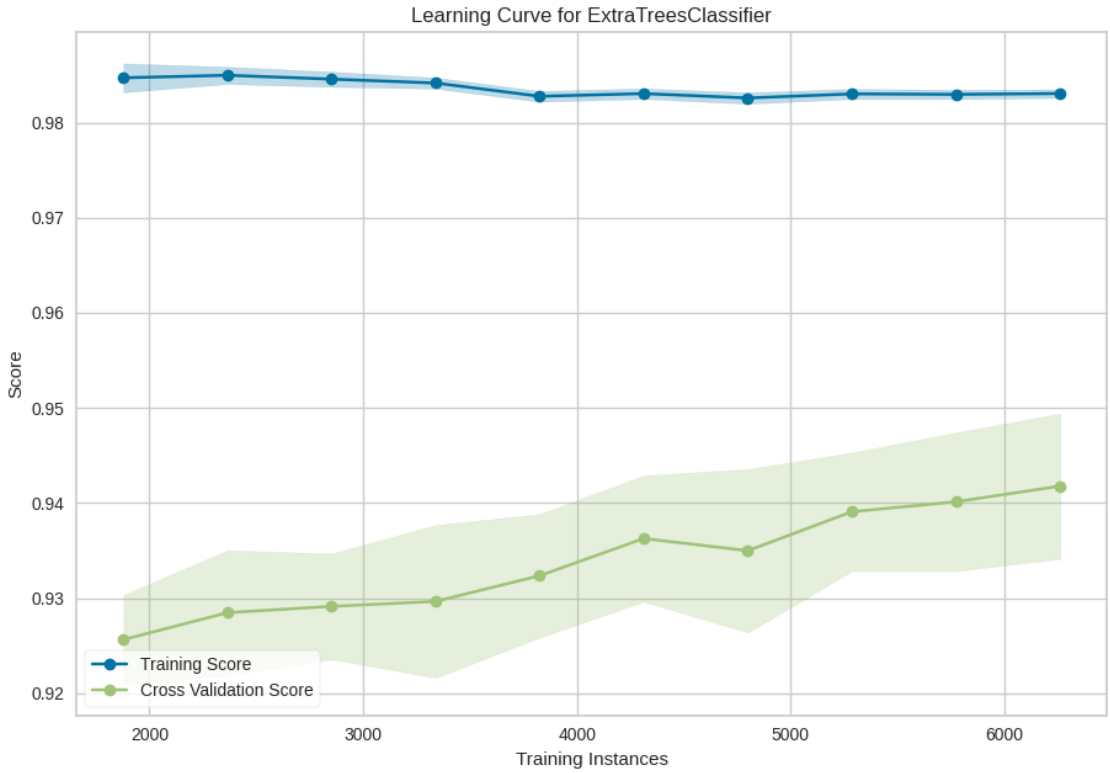


[ ]:
```
display_precision_recall_curve(model)
```

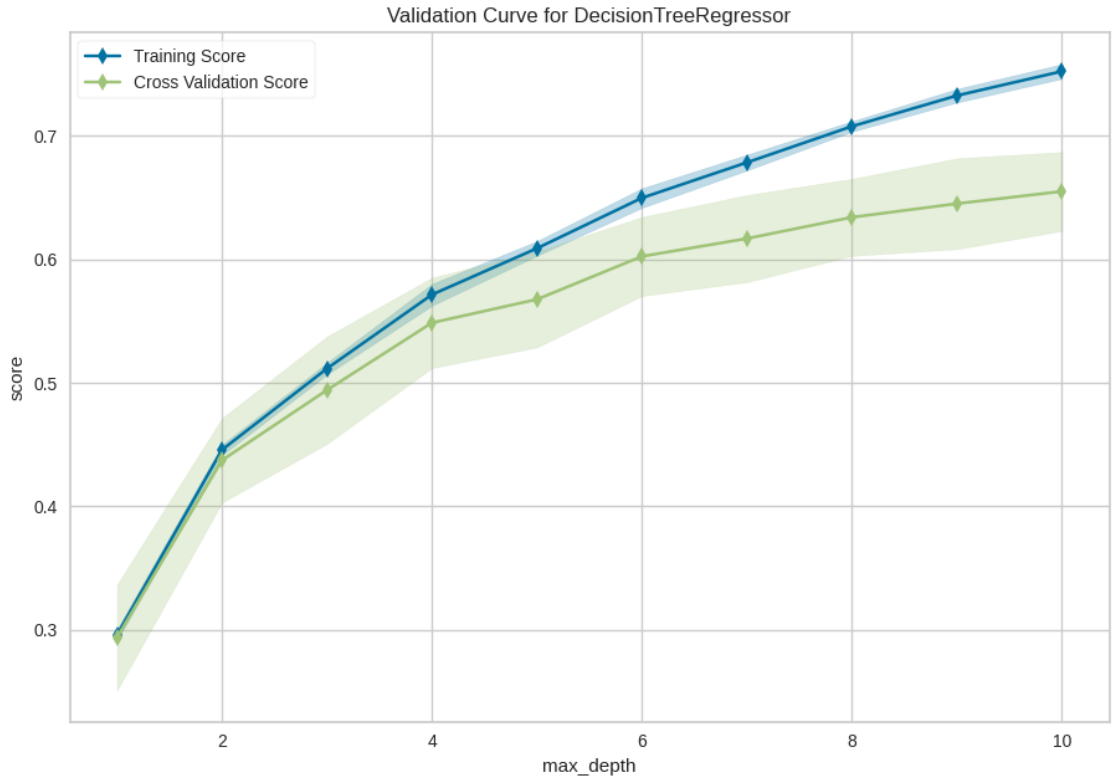
```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/prcurve.py:254:
YellowbrickWarning: micro=True is ignored;specify per_class=False to draw a PR
curve after micro-averaging
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but ExtraTreesClassifier was fitted with feature
names
  warnings.warn(
```



[ ]:
```
display_learning_curve(model)
```



[ ]:
```
display_validation_curve(model)
```



[ ]:
```
display_feature_importance(model)
```

```
Summary of Feature Importance:
title_clean: Score: 0.04789
img_count: Score: 0.03640
has_form: Score: 0.04284
has_login_form: Score: 0.16852
has_js: Score: 0.00961
js_include_b64: Score: 0.00146
nb_tokens: Score: 0.05232
text_clean: Score: 0.05871
nb_title_entities: Score: 0.01115
nb_text_entities: Score: 0.02469
bank_of_america: Score: 0.00035
wells_fargo: Score: 0.00016
citibank: Score: 0.00000
apple: Score: 0.00370
microsoft: Score: 0.01127
amazon: Score: 0.00267
google: Score: 0.00414
facebook: Score: 0.00065
dhl: Score: 0.00000
```

youtube: Score: 0.00000
whatsapp: Score: 0.00001
linkedin: Score: 0.00040
twitter: Score: 0.00568
access account: Score: 0.00410
account account: Score: 0.00102
account account create: Score: 0.00059
account another: Score: 0.00035
account another account: Score: 0.00015
account create: Score: 0.00455
account create one: Score: 0.00135
account doe: Score: 0.00130
account doe n: Score: 0.00146
account enter: Score: 0.00095
account enter password: Score: 0.00009
account microsoft: Score: 0.00082
account microsoft term: Score: 0.00022
account nobody: Score: 0.00080
account nobody mycraftmail: Score: 0.00079
account sign: Score: 0.00405
account sign microsoft: Score: 0.00006
address password: Score: 0.00275
another account: Score: 0.00301
another account microsoft: Score: 0.00008
another shot: Score: 0.00372
anyone one: Score: 0.00030
anyone one drive: Score: 0.00027
anywhere device: Score: 0.00212
anywhere device share: Score: 0.00001
browser setting: Score: 0.00209
browser would: Score: 0.00298
browser would like: Score: 0.00017
cant access: Score: 0.00067
cant access account: Score: 0.00046
cap lock: Score: 0.01469
choose email: Score: 0.00016
choose email provider: Score: 0.00010
client mycraftmail: Score: 0.00057
client mycraftmail username: Score: 0.00018
contact u: Score: 0.00460
cookie policy: Score: 0.00417
cooky term: Score: 0.00082
cooky term microsoft: Score: 0.00042
cooky trm: Score: 0.00089
cooky trm f: Score: 0.00072
copyright rakuten: Score: 0.00166
copyright rakuten inc: Score: 0.00146
create one: Score: 0.00153

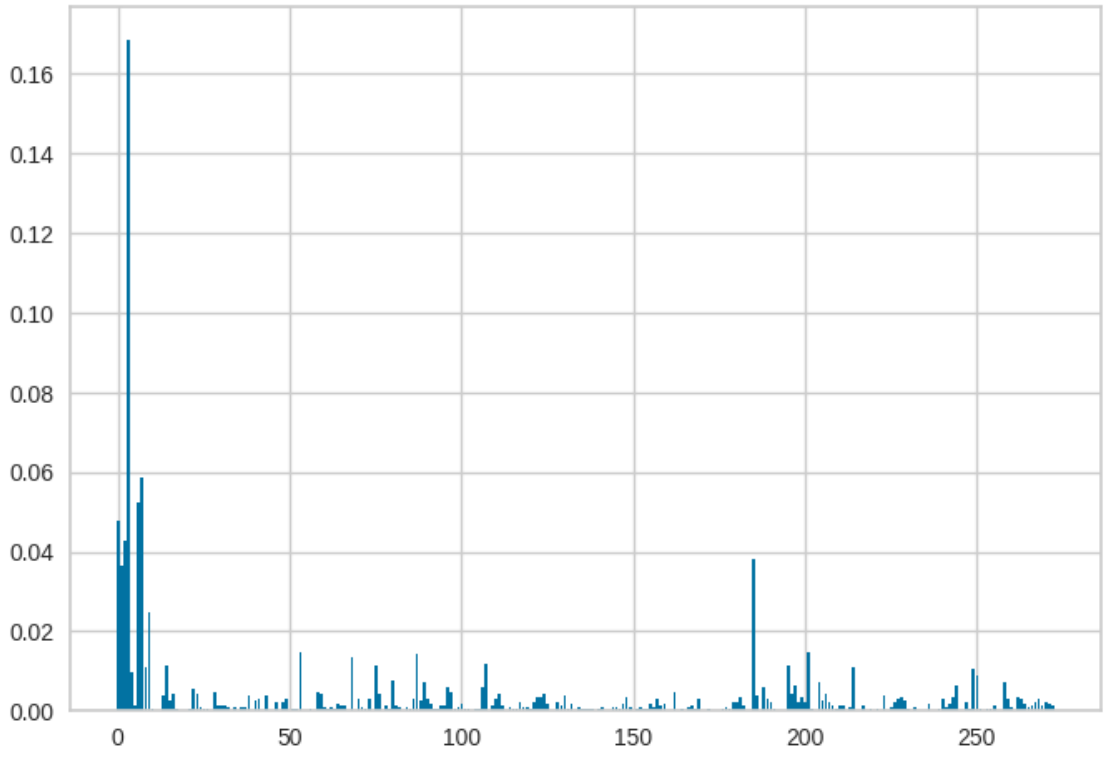
create one sign: Score: 0.00005
debtwire login: Score: 0.01362
detected disabled: Score: 0.00004
detected disabled browser: Score: 0.00300
device share: Score: 0.00076
device share anyone: Score: 0.00072
different account: Score: 0.00304
different account account: Score: 0.00016
disabled browser: Score: 0.01146
disabled browser would: Score: 0.00445
document one: Score: 0.00004
document please: Score: 0.00146
document please choose: Score: 0.00059
doe n: Score: 0.00775
doe n exist: Score: 0.00119
dont fret: Score: 0.00075
dont fret let: Score: 0.00000
drive shared: Score: 0.00098
drive shared document: Score: 0.00063
e mail: Score: 0.00293
email address: Score: 0.01439
email address password: Score: 0.00266
email password: Score: 0.00719
email provider: Score: 0.00292
email provider login: Score: 0.00188
english idcapslock: Score: 0.00061
english idcapslock copyright: Score: 0.00054
enter different: Score: 0.00141
enter different account: Score: 0.00129
enter email: Score: 0.00601
enter password: Score: 0.00477
enter password forgot: Score: 0.00040
enter password keep: Score: 0.00101
exist enter: Score: 0.00177
exist enter different: Score: 0.00036
f u: Score: 0.00067
file anywhere: Score: 0.00030
file anywhere device: Score: 0.00061
file login: Score: 0.00033
find account: Score: 0.00598
forgot password: Score: 0.01169
forgot password privacy: Score: 0.00026
forgot password sign: Score: 0.00127
forgotten password: Score: 0.00322
fret let: Score: 0.00449
fret let give: Score: 0.00151
frgt sswrd: Score: 0.00000
frgt sswrd sign: Score: 0.00084

get file: Score: 0.00066
get file anywhere: Score: 0.00039
give another: Score: 0.00221
give another shot: Score: 0.00088
idcapslock copyright: Score: 0.00109
idcapslock copyright rakuten: Score: 0.00067
inc affiliate: Score: 0.00201
inc right: Score: 0.00366
inc right reserved: Score: 0.00347
keep signed: Score: 0.00423
keep signed forgot: Score: 0.00168
key sign: Score: 0.00006
key sign option: Score: 0.00032
legacy twitter: Score: 0.00223
legacy twitter yes: Score: 0.00150
let give: Score: 0.00370
let give another: Score: 0.00070
like proceed: Score: 0.00163
like proceed legacy: Score: 0.00000
login mail: Score: 0.00104
login mail get: Score: 0.00038
login office: Score: 0.00041
login office login: Score: 0.00046
login outlook: Score: 0.00055
login outlook login: Score: 0.00021
login view: Score: 0.00046
login view shared: Score: 0.00086
mail get: Score: 0.00010
mail get file: Score: 0.00006
make sure: Score: 0.00099
microsoft account: Score: 0.00115
microsoft account doe: Score: 0.00043
microsoft term: Score: 0.00186
microsoft term privacy: Score: 0.00326
mycraftmail client: Score: 0.00111
mycraftmail client mycraftmail: Score: 0.00026
mycraftmail enter: Score: 0.00047
mycraftmail enter password: Score: 0.00085
mycraftmail username: Score: 0.00054
mycraftmail username password: Score: 0.00074
n access: Score: 0.00166
n access account: Score: 0.00099
n exist: Score: 0.00290
n exist enter: Score: 0.00141
need help: Score: 0.00197
next password: Score: 0.00027
next password account: Score: 0.00001
nobody mycraftmail: Score: 0.00481

nobody mycraftmail enter: Score: 0.00024
nthr unt: Score: 0.00039
nthr unt sign: Score: 0.00029
ntr sswrd: Score: 0.00075
office login: Score: 0.00120
office login outlook: Score: 0.00050
one drive: Score: 0.00298
one drive shared: Score: 0.00013
one sign: Score: 0.00024
one sign security: Score: 0.00038
option login: Score: 0.00001
option next: Score: 0.00004
option next password: Score: 0.00011
outlook login: Score: 0.00044
password account: Score: 0.00082
password account create: Score: 0.00051
password forgot: Score: 0.00223
password forgot password: Score: 0.00214
password keep: Score: 0.00341
password keep signed: Score: 0.00148
password privacy: Score: 0.00017
password privacy cooky: Score: 0.00003
password remember: Score: 0.03828
password sign: Score: 0.00404
password sign another: Score: 0.00022
password webmail: Score: 0.00614
password webmail mini: Score: 0.00295
phone number: Score: 0.00232
pick account: Score: 0.00049
pick account another: Score: 0.00013
please choose: Score: 0.00014
please choose email: Score: 0.00011
please contact: Score: 0.01131
please enable: Score: 0.00430
please enter: Score: 0.00653
please enter password: Score: 0.00219
please sign: Score: 0.00330
policy term: Score: 0.00241
privacy cooky: Score: 0.01488
privacy cooky term: Score: 0.00064
privacy cooky trm: Score: 0.00001
privacy policy: Score: 0.00728
privacy policy term: Score: 0.00251
proceed legacy: Score: 0.00446
proceed legacy twitter: Score: 0.00228
provider login: Score: 0.00137
provider login view: Score: 0.00047
rakuten inc: Score: 0.00133

rakuten inc right: Score: 0.00152
read document: Score: 0.00020
read document please: Score: 0.00085
right reserved: Score: 0.01106
security key: Score: 0.00013
security key sign: Score: 0.00004
share anyone: Score: 0.00142
share anyone one: Score: 0.00006
shared document: Score: 0.00053
shared document one: Score: 0.00007
shared file: Score: 0.00058
shared file login: Score: 0.00004
sign account: Score: 0.00383
sign account nobody: Score: 0.00018
sign account sign: Score: 0.00080
sign continue: Score: 0.00201
sign different: Score: 0.00293
sign email: Score: 0.00332
sign microsoft: Score: 0.00243
sign microsoft account: Score: 0.00071
sign nthr: Score: 0.00023
sign nthr unt: Score: 0.00078
sign option: Score: 0.00023
sign option login: Score: 0.00001
sign option next: Score: 0.00005
sign privacy: Score: 0.00176
sign privacy cooky: Score: 0.00031
sign security: Score: 0.00003
sign security key: Score: 0.00048
sign sign: Score: 0.00300
signed forgot: Score: 0.00110
signed forgot password: Score: 0.00175
something went: Score: 0.00329
something went wrong: Score: 0.00633
sswrd sign: Score: 0.00030
sswrd sign nthr: Score: 0.00015
stay signed: Score: 0.00212
term microsoft: Score: 0.00074
term privacy: Score: 0.01050
term privacy cooky: Score: 0.00875
timed please: Score: 0.00054
trm f: Score: 0.00021
trm f u: Score: 0.00043
twitter yes: Score: 0.00072
twitter yes something: Score: 0.00150
unt sign: Score: 0.00023
unt sign privacy: Score: 0.00014
username password: Score: 0.00712

```
username password webmail: Score: 0.00295
view shared: Score: 0.00080
view shared file: Score: 0.00026
webmail login: Score: 0.00366
webmail mini: Score: 0.00285
webmail welcome: Score: 0.00176
went wrong: Score: 0.00115
went wrong dont: Score: 0.00150
would like: Score: 0.00203
would like proceed: Score: 0.00301
wrong dont: Score: 0.00150
wrong dont fret: Score: 0.00215
yes something: Score: 0.00160
yes something went: Score: 0.00153
```



Hyperparameter Tuning

[ ]:
```python
# Number of trees
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]

# Number of features to consider at every split
max_features = ['auto', 'sqrt']
```

```
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)

# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]

# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]

# Method of selecting samples for training each tree
bootstrap = [True, False]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000],
'max_features': ['auto', 'sqrt'], 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80,
90, 100, 110, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2,
4], 'bootstrap': [True, False]}
```

[ ]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
etr = ExtraTreesRegressor()

# Random search of parameters, using 3 fold cross validation,
# search across 10 different combinations, and use all available cores
et_random = RandomizedSearchCV(
    estimator = etr,
    param_distributions = random_grid,
    n_iter = 100,
    cv = 3,
    verbose = 2,
    random_state = 1,
    n_jobs = -1
)

# Fit the random search model
et_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

```
[ ]: RandomizedSearchCV(cv=3, estimator=ExtraTreesRegressor(), n_iter=100, n_jobs=-1,
                        param_distributions={'bootstrap': [True, False],
                                             'max_depth': [10, 20, 30, 40, 50, 60,
                                                           70, 80, 90, 100, 110,
                                                           None],
                                             'max_features': ['auto', 'sqrt'],
                                             'min_samples_leaf': [1, 2, 4],
                                             'min_samples_split': [2, 5, 10],
                                             'n_estimators': [200, 400, 600, 800,
                                                              1000, 1200, 1400, 1600,
                                                              1800, 2000]},
                        random_state=1, verbose=2)
```

```
[ ]: et_random.best_params_
```

```
[ ]: {'n_estimators': 2000,
      'min_samples_split': 2,
      'min_samples_leaf': 1,
      'max_features': 'sqrt',
      'max_depth': 70,
      'bootstrap': True}
```

```
[ ]: et_tr = ExtraTreesClassifier(
        n_estimators = 2000,
        min_samples_split = 2,
        min_samples_leaf = 1,
        max_features = 'sqrt',
        max_depth = 70,
        bootstrap = True
    )
```

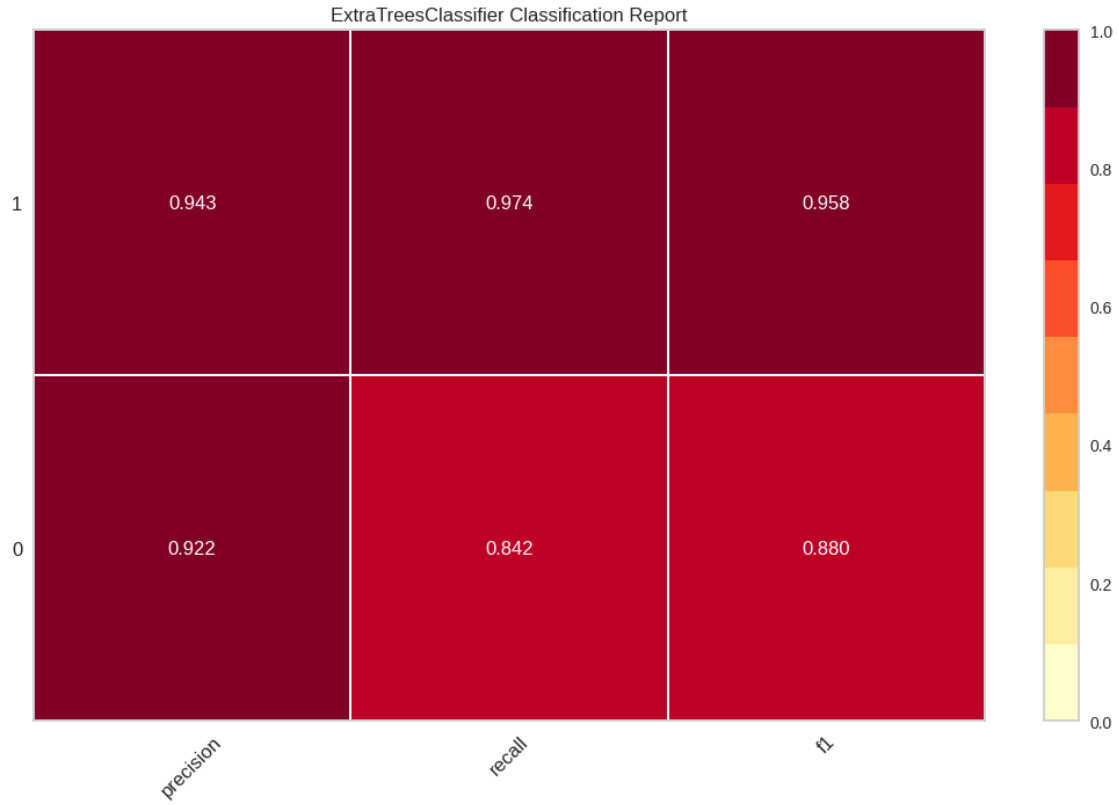
```
[ ]: model = et_tr
```

```
[ ]: display_accuracy(model)
```

```
Training time: 20.540549993515015
Test accuracy : 93.79%
```

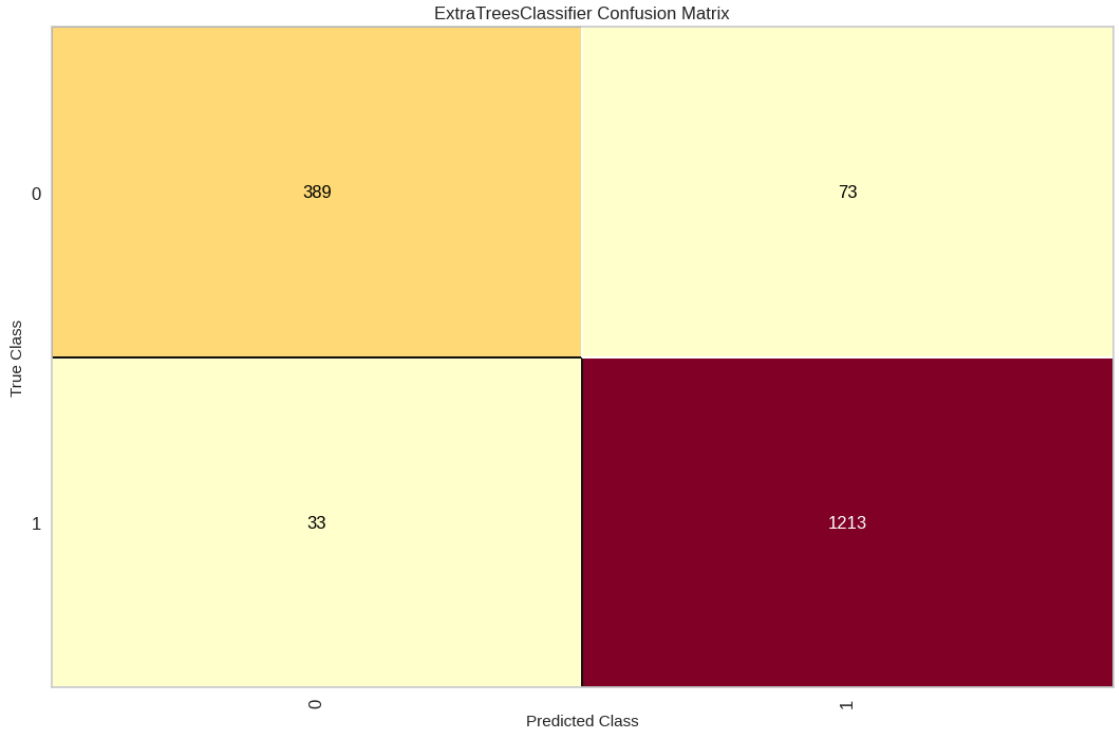
```
[ ]: display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but ExtraTreesClassifier was fitted with feature
names
  warnings.warn(
```



1.13.3 GradientBoosting Classifier

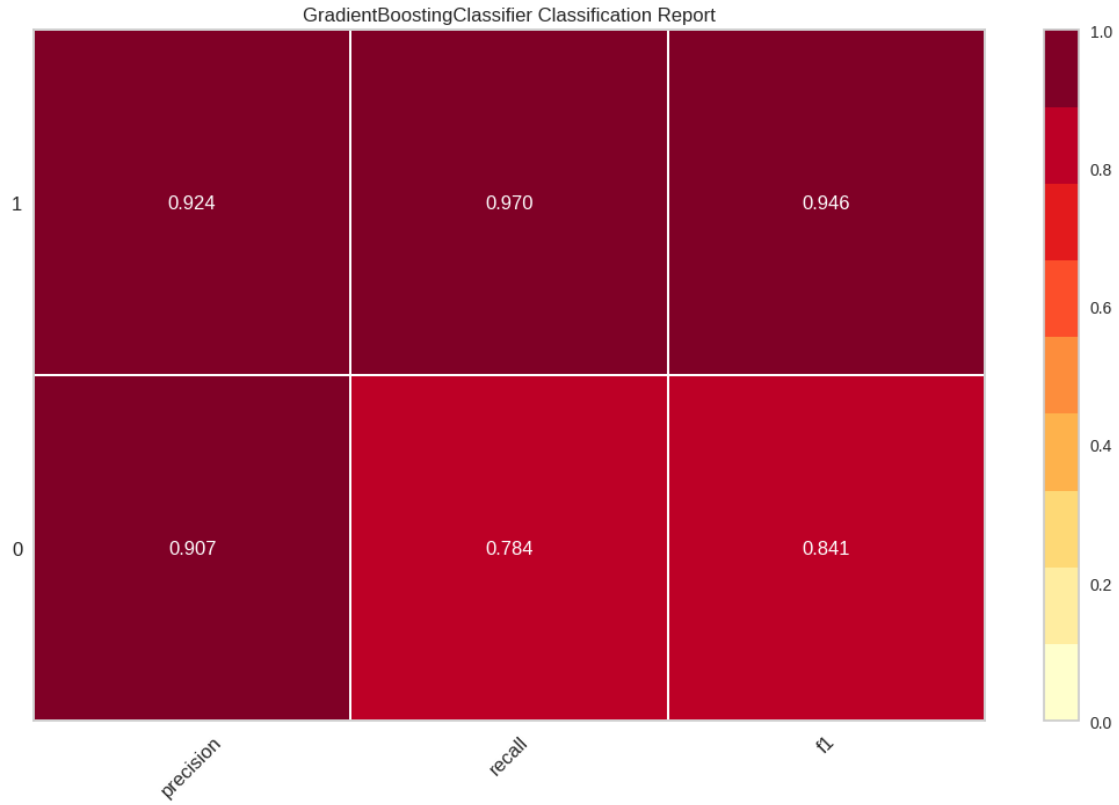
[ ]:
```
model = gb
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 3.425570487976074
Test accuracy : 91.98%
```

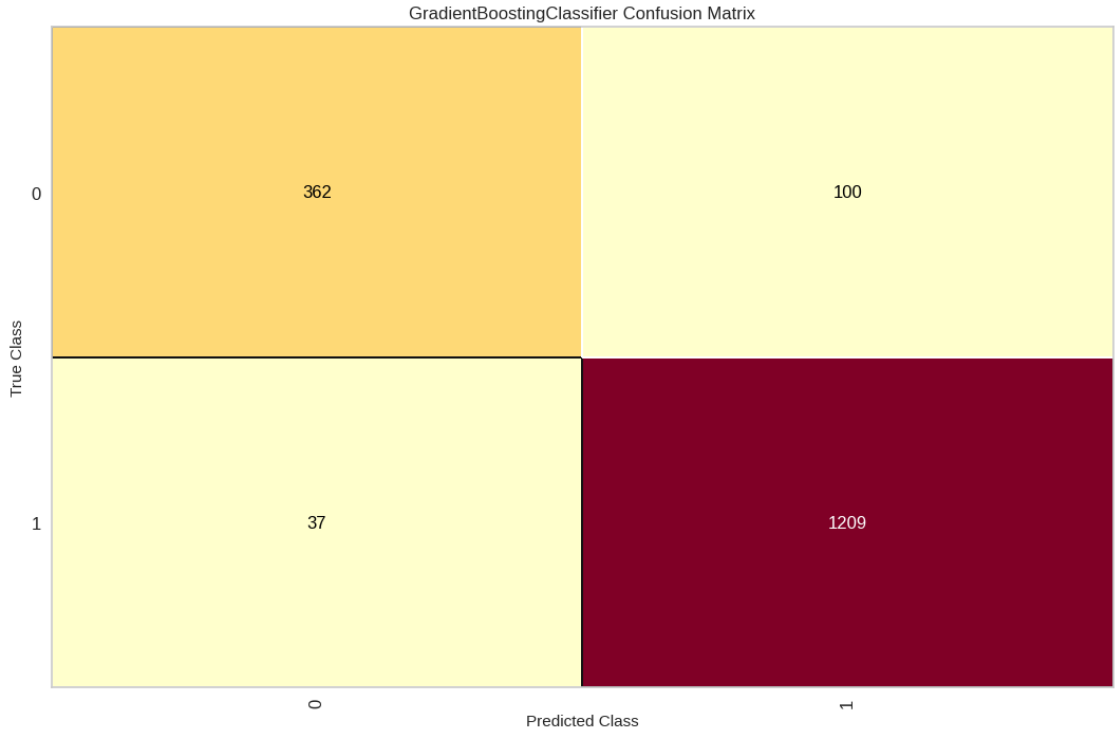
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but GradientBoostingClassifier was fitted with
feature names
  warnings.warn(
```



1.13.4 K-Nearest Neighbors Classifier

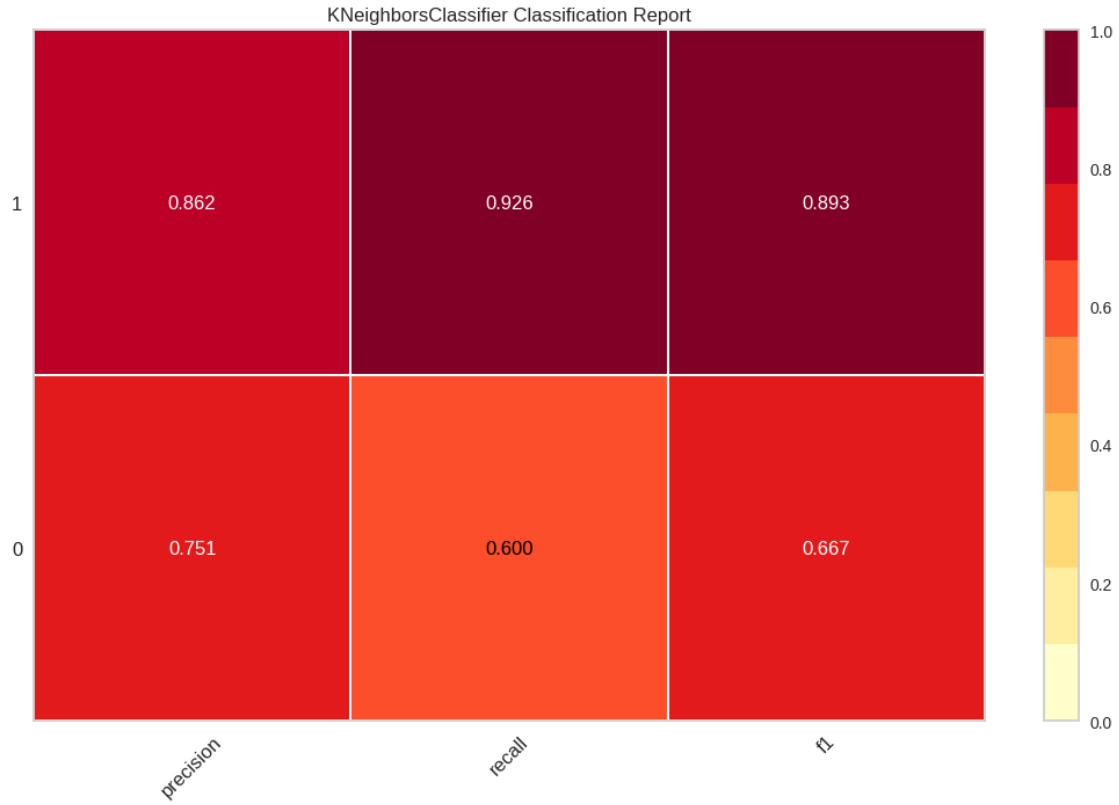
[ ]:
```
model = knn
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 0.016538381576538086
Test accuracy : 83.78%
```

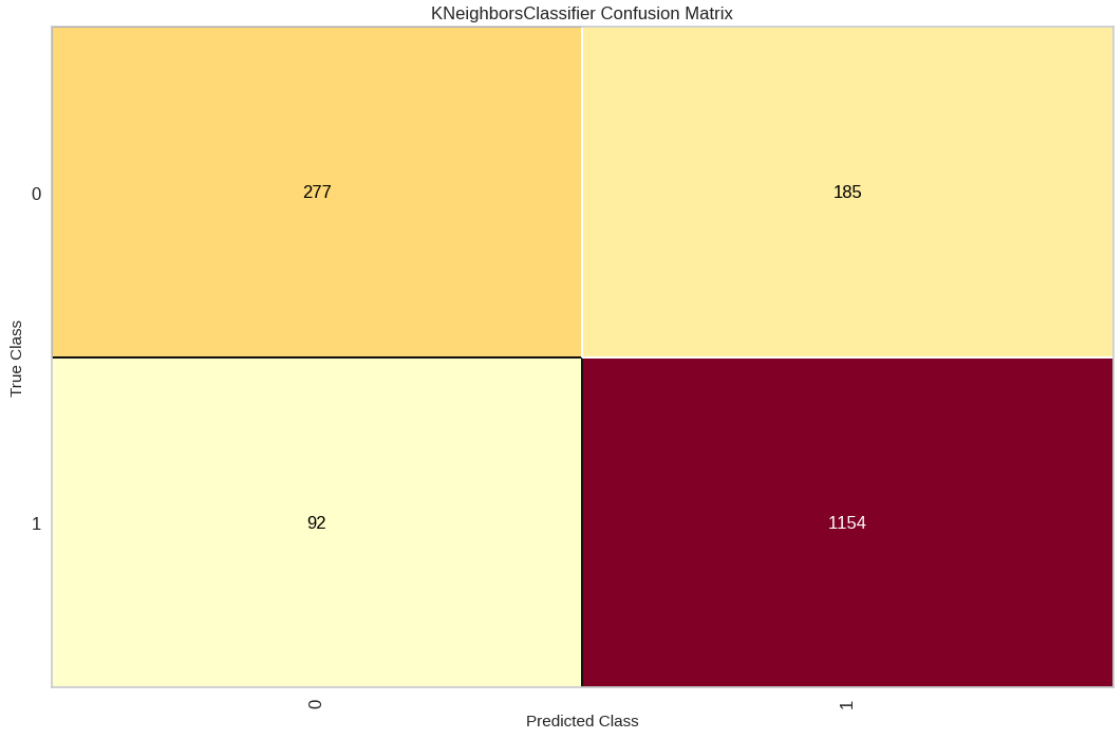
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with feature
names
  warnings.warn(
```



1.13.5 Logistic Regression Classifier

[ ]:
```
model = lr
```

[ ]:
```
display_accuracy(model)
```

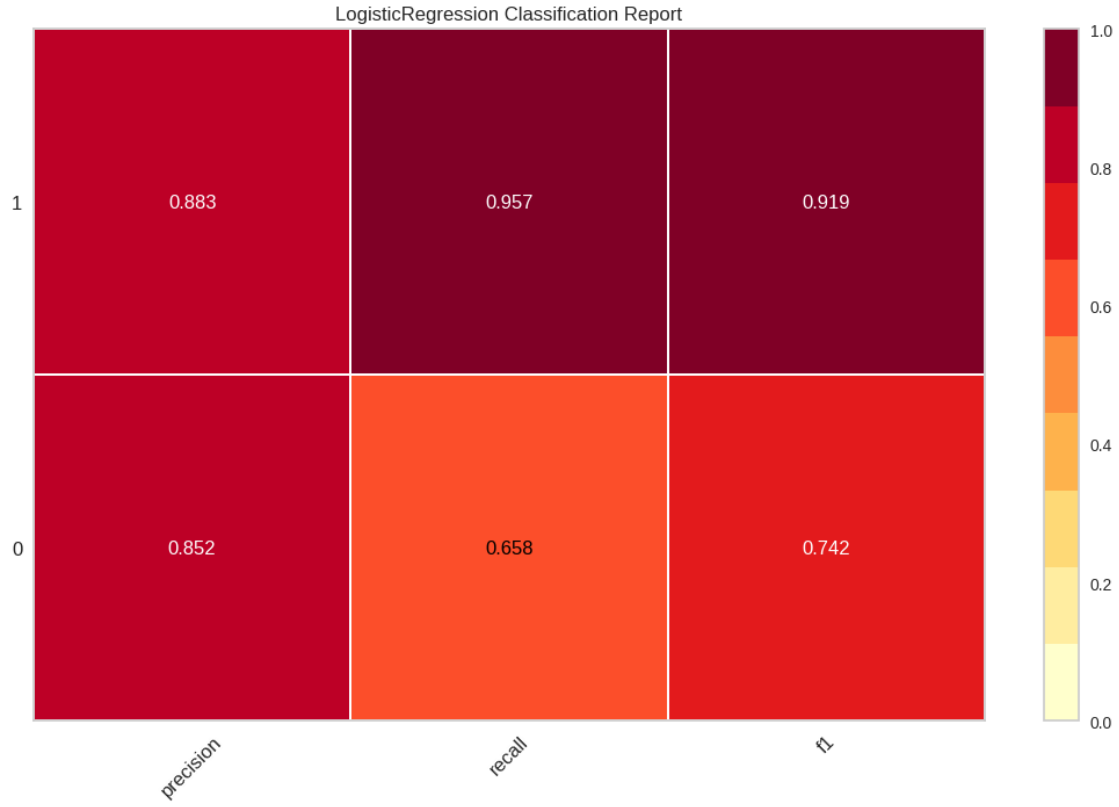
```
Training time: 0.39572787284851074
Test accuracy : 87.65%
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

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

[ ]:
```
display_classification_report(model)
```

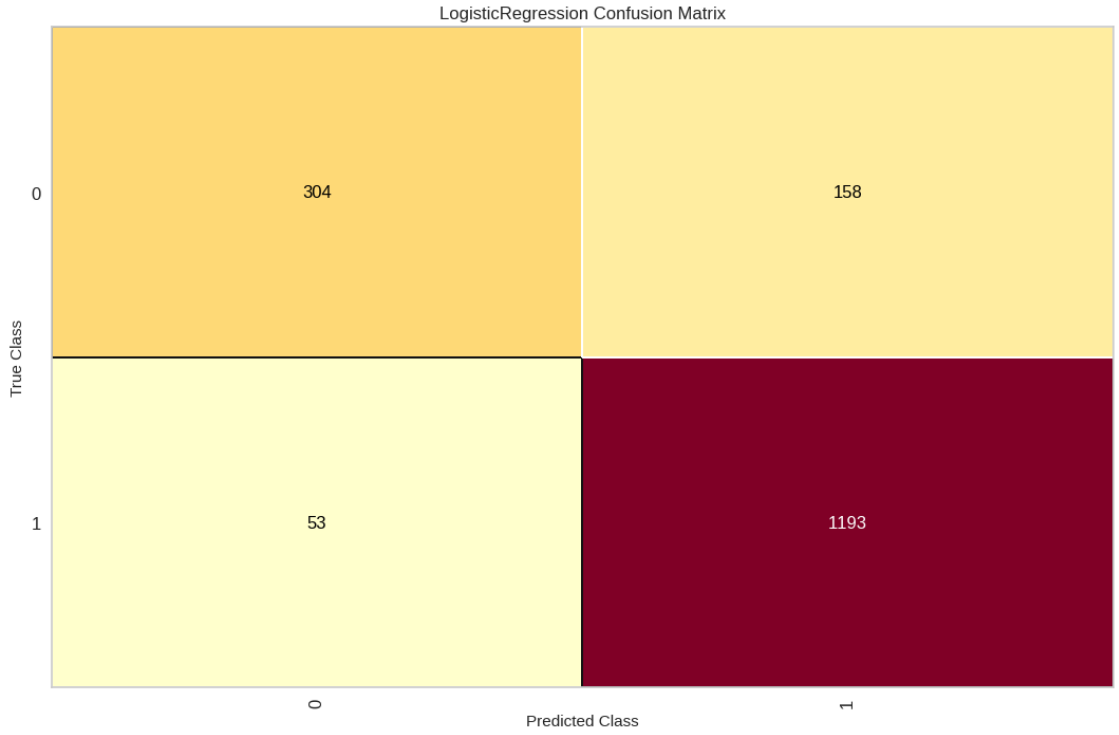
```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
```

```
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LogisticRegression was fitted with feature
names
  warnings.warn(
```



1.13.6 LinearSVC Classifier

[ ]:
```
model = svc
```

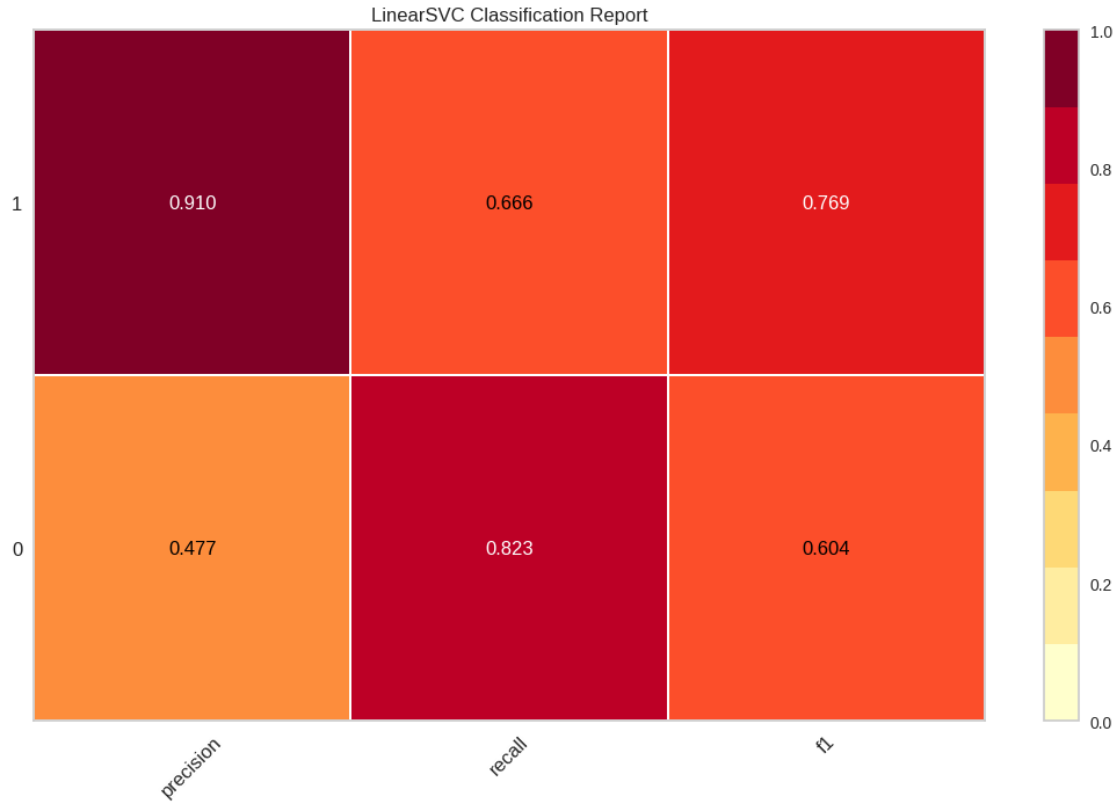
[ ]:
```
display_accuracy(model)
```

```
Training time: 0.738835334777832
Test accuracy : 70.84%
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py:1244:
ConvergenceWarning: Liblinear failed to converge, increase the number of
iterations.
  warnings.warn(
```

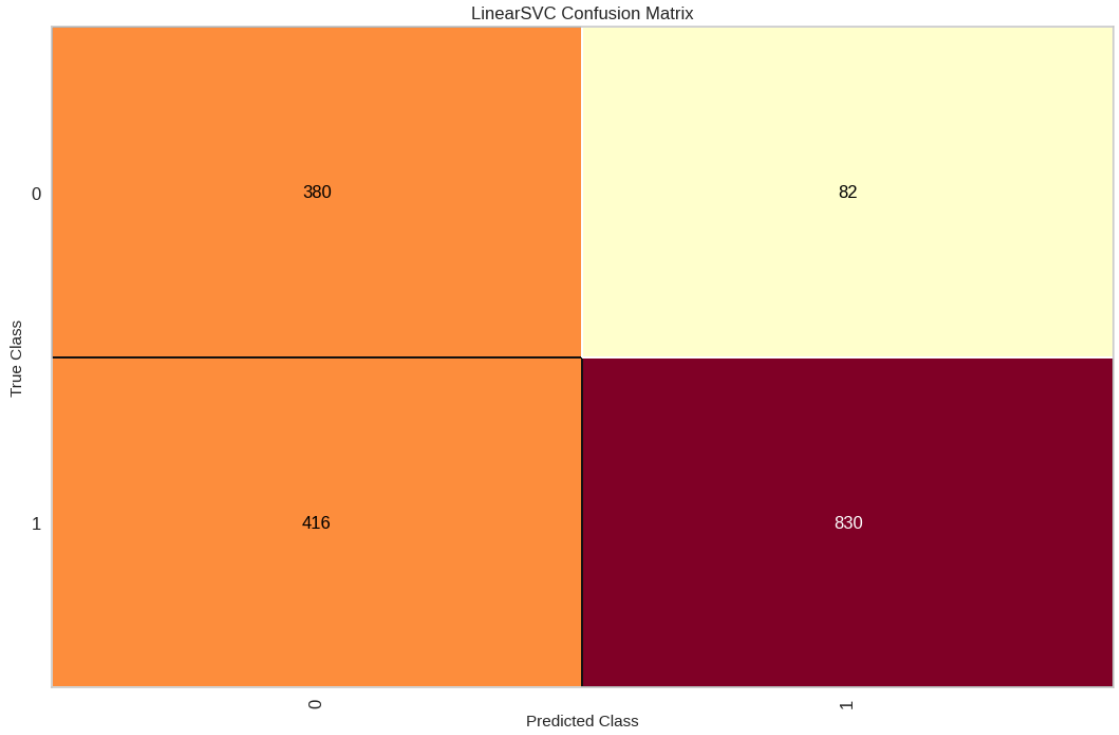
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LinearSVC was fitted with feature names
  warnings.warn(
```



1.13.7 Random Forest Classifier

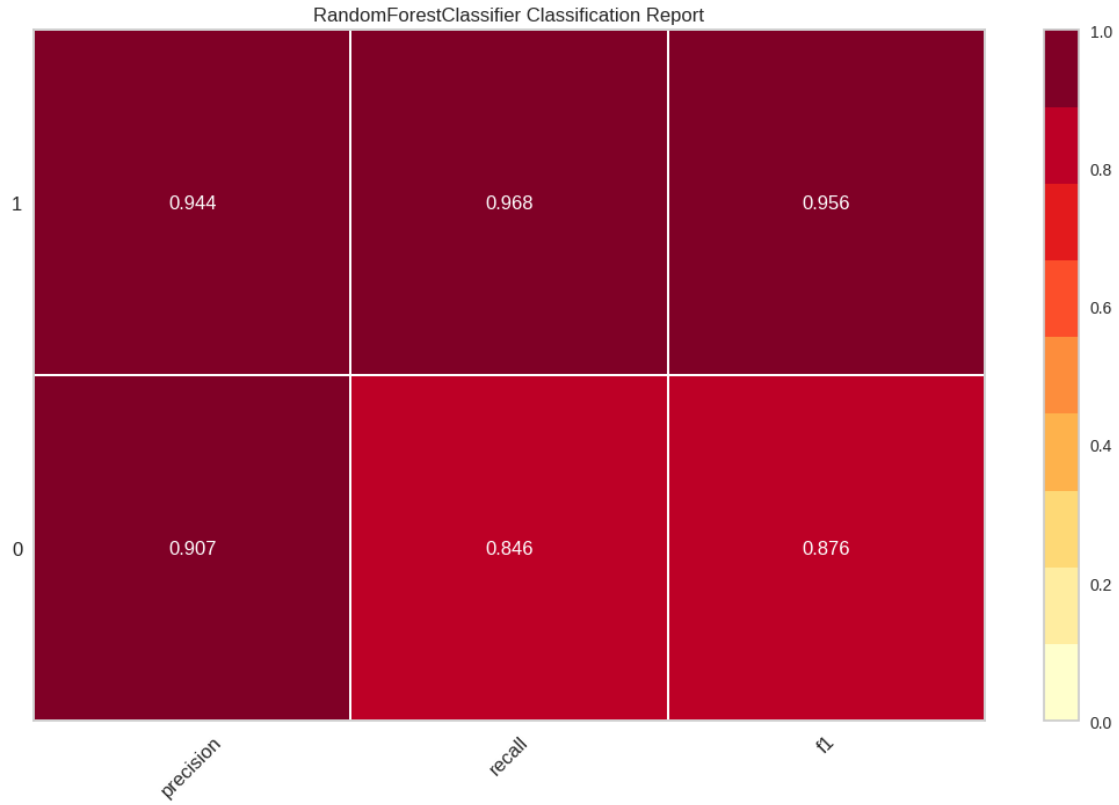
[ ]:
```
model = rf
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 1.2332193851470947
Test accuracy : 93.50%
```

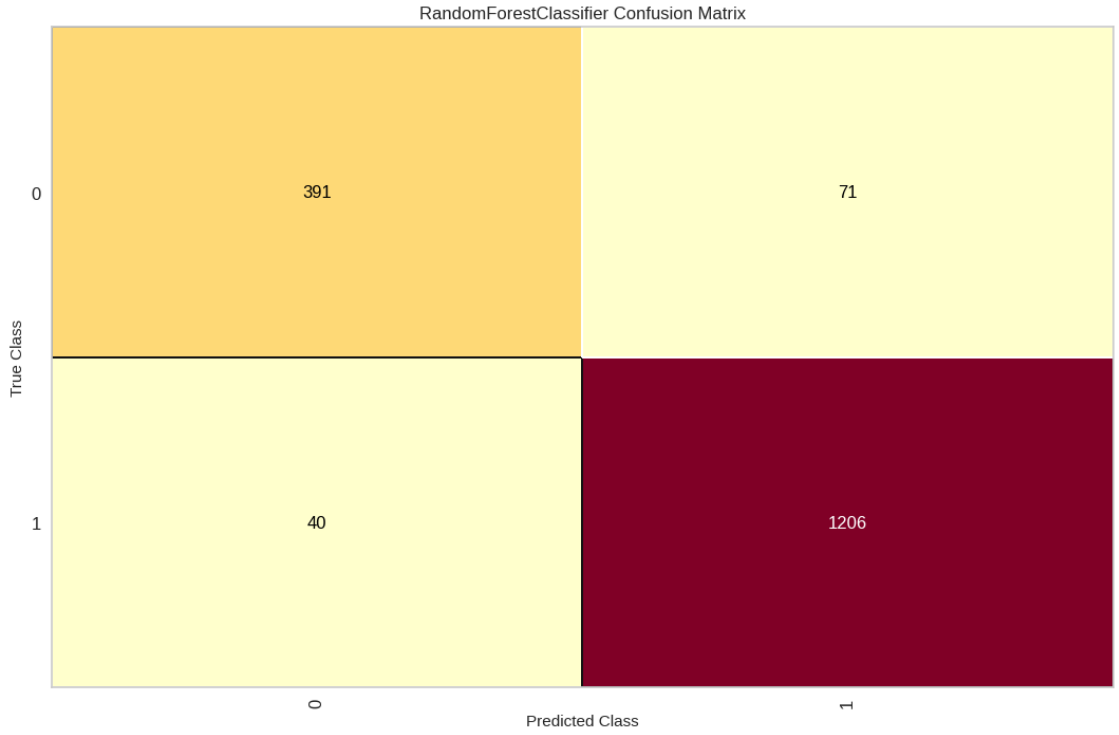
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



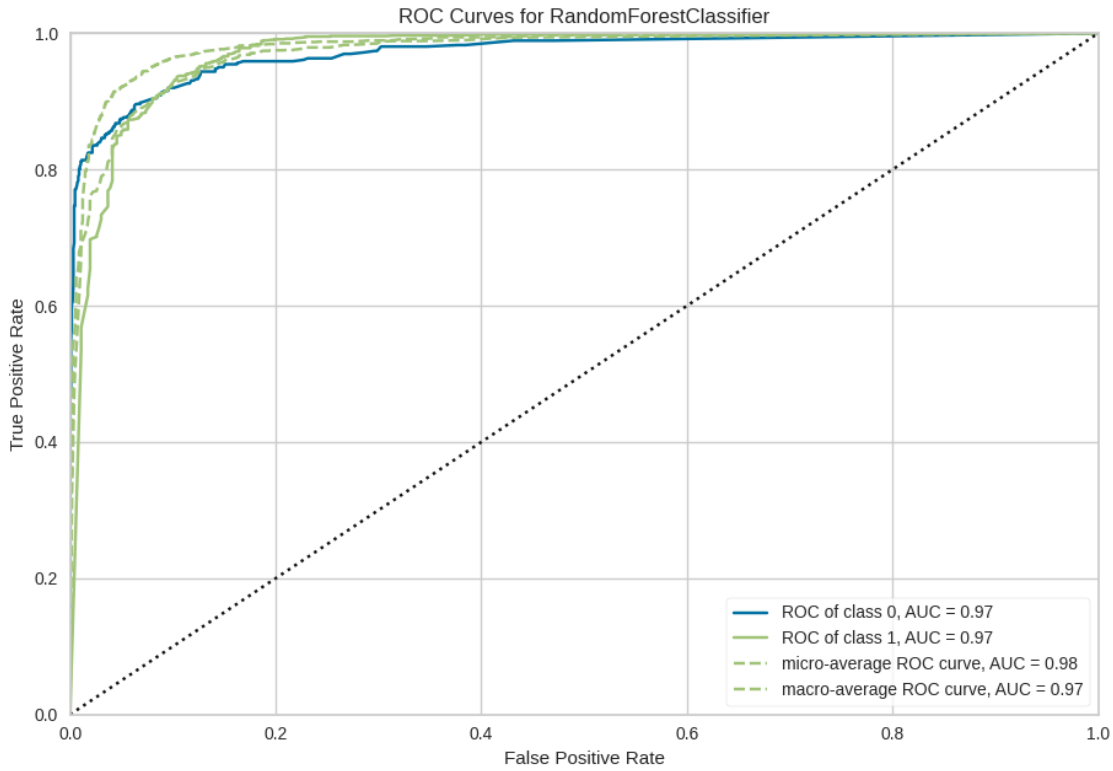
[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```



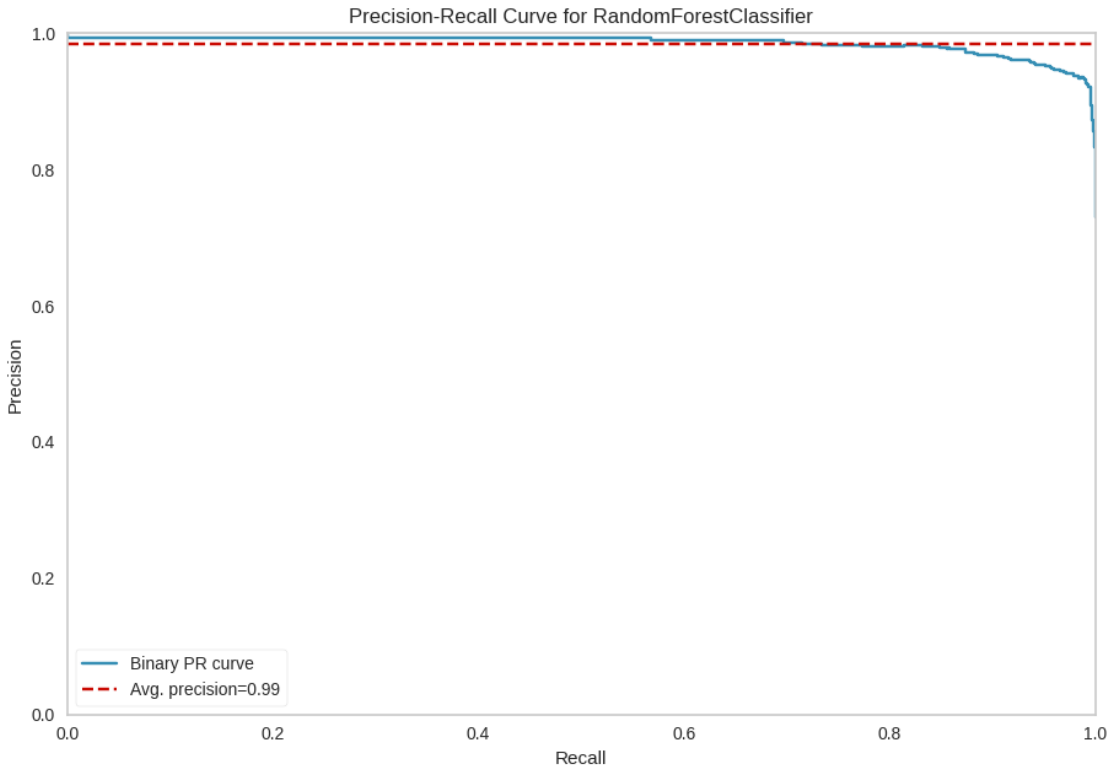
[ ]:
```
display_roc_curve(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```

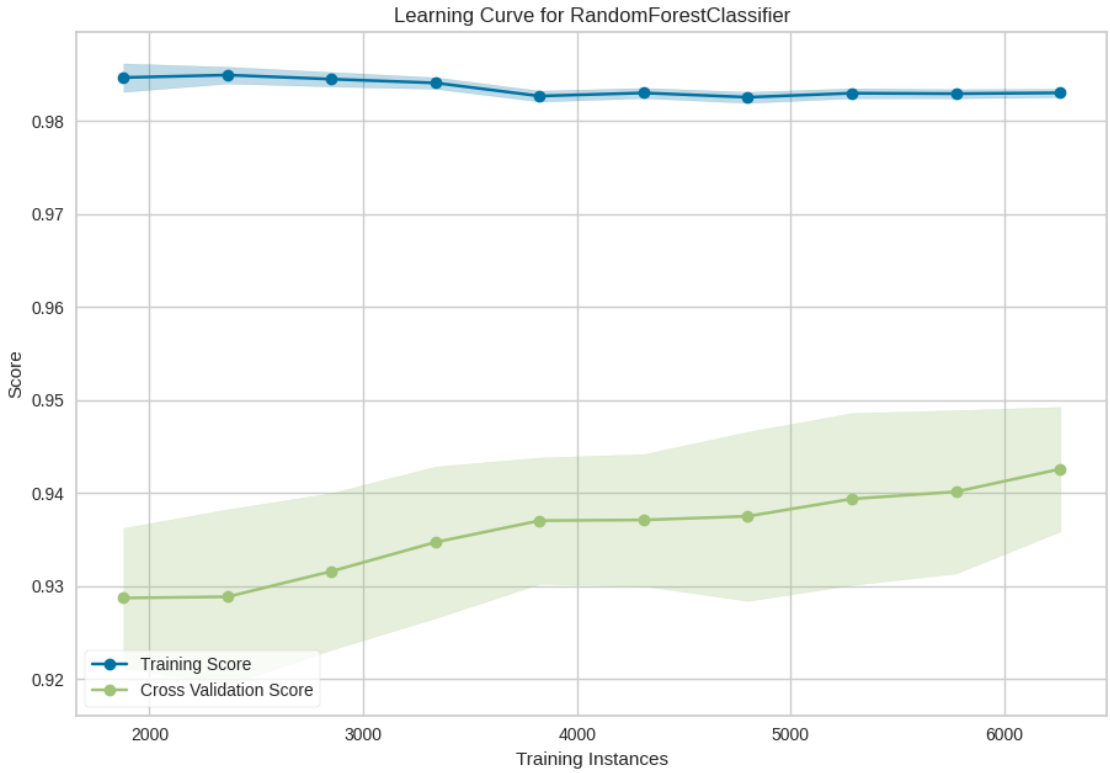


[ ]:
```
display_precision_recall_curve(model)
```

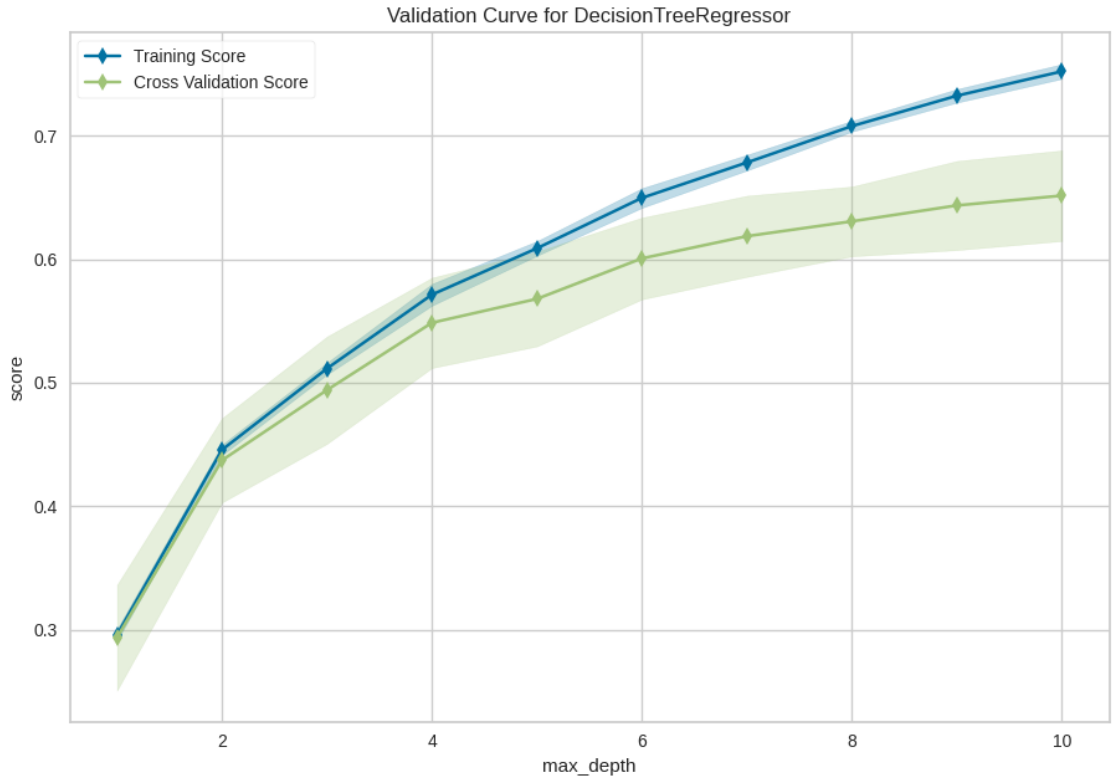
```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/prcurve.py:254:
YellowbrickWarning: micro=True is ignored;specify per_class=False to draw a PR
curve after micro-averaging
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```



[ ]:
```
display_learning_curve(model)
```



[ ]:
```
display_validation_curve(model)
```



[ ]:
```
display_feature_importance(model)
```

```
Summary of Feature Importance:
title_clean: Score: 0.07037
img_count: Score: 0.07146
has_form: Score: 0.03281
has_login_form: Score: 0.14673
has_js: Score: 0.00713
js_include_b64: Score: 0.00138
nb_tokens: Score: 0.08440
text_clean: Score: 0.07437
nb_title_entities: Score: 0.00946
nb_text_entities: Score: 0.03107
bank_of_america: Score: 0.00030
wells_fargo: Score: 0.00011
citibank: Score: 0.00000
apple: Score: 0.00301
microsoft: Score: 0.01066
amazon: Score: 0.00192
google: Score: 0.00354
facebook: Score: 0.00055
dhl: Score: 0.00000
```

youtube: Score: 0.00000
whatsapp: Score: 0.00003
linkedin: Score: 0.00035
twitter: Score: 0.00656
access account: Score: 0.00221
account account: Score: 0.00080
account account create: Score: 0.00031
account another: Score: 0.00004
account another account: Score: 0.00012
account create: Score: 0.00297
account create one: Score: 0.00101
account doe: Score: 0.00043
account doe n: Score: 0.00103
account enter: Score: 0.00072
account enter password: Score: 0.00008
account microsoft: Score: 0.00042
account microsoft term: Score: 0.00029
account nobody: Score: 0.00067
account nobody mycraftmail: Score: 0.00067
account sign: Score: 0.00370
account sign microsoft: Score: 0.00006
address password: Score: 0.00280
another account: Score: 0.00179
another account microsoft: Score: 0.00002
another shot: Score: 0.00432
anyone one: Score: 0.00037
anyone one drive: Score: 0.00023
anywhere device: Score: 0.00066
anywhere device share: Score: 0.00001
browser setting: Score: 0.00208
browser would: Score: 0.00075
browser would like: Score: 0.00395
cant access: Score: 0.00044
cant access account: Score: 0.00059
cap lock: Score: 0.01033
choose email: Score: 0.00042
choose email provider: Score: 0.00041
client mycraftmail: Score: 0.00033
client mycraftmail username: Score: 0.00056
contact u: Score: 0.00362
cookie policy: Score: 0.00533
cooky term: Score: 0.00057
cooky term microsoft: Score: 0.00070
cooky trm: Score: 0.00036
cooky trm f: Score: 0.00000
copyright rakuten: Score: 0.00067
copyright rakuten inc: Score: 0.00131
create one: Score: 0.00137

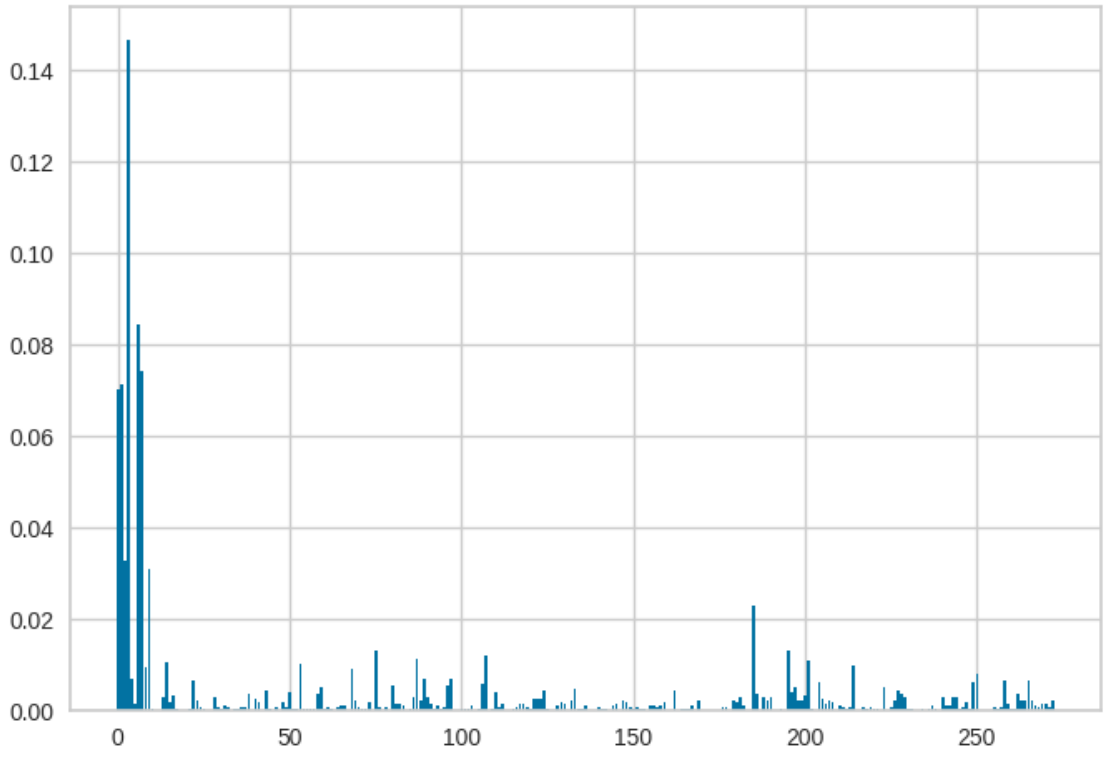
create one sign: Score: 0.00006
debtwire login: Score: 0.00924
detected disabled: Score: 0.00225
detected disabled browser: Score: 0.00078
device share: Score: 0.00054
device share anyone: Score: 0.00060
different account: Score: 0.00183
different account account: Score: 0.00015
disabled browser: Score: 0.01332
disabled browser would: Score: 0.00075
document one: Score: 0.00026
document please: Score: 0.00094
document please choose: Score: 0.00015
doe n: Score: 0.00549
doe n exist: Score: 0.00145
dont fret: Score: 0.00162
dont fret let: Score: 0.00123
drive shared: Score: 0.00013
drive shared document: Score: 0.00032
e mail: Score: 0.00285
email address: Score: 0.01154
email address password: Score: 0.00241
email password: Score: 0.00687
email provider: Score: 0.00307
email provider login: Score: 0.00142
english idcapslock: Score: 0.00044
english idcapslock copyright: Score: 0.00107
enter different: Score: 0.00056
enter different account: Score: 0.00068
enter email: Score: 0.00573
enter password: Score: 0.00709
enter password forgot: Score: 0.00024
enter password keep: Score: 0.00009
exist enter: Score: 0.00051
exist enter different: Score: 0.00042
f u: Score: 0.00030
file anywhere: Score: 0.00118
file anywhere device: Score: 0.00021
file login: Score: 0.00031
find account: Score: 0.00579
forgot password: Score: 0.01211
forgot password privacy: Score: 0.00012
forgot password sign: Score: 0.00052
forgotten password: Score: 0.00425
fret let: Score: 0.00081
fret let give: Score: 0.00149
frgt sswrd: Score: 0.00016
frgt sswrd sign: Score: 0.00000

get file: Score: 0.00054
get file anywhere: Score: 0.00078
give another: Score: 0.00154
give another shot: Score: 0.00154
idcapslock copyright: Score: 0.00079
idcapslock copyright rakuten: Score: 0.00035
inc affiliate: Score: 0.00248
inc right: Score: 0.00254
inc right reserved: Score: 0.00269
keep signed: Score: 0.00434
keep signed forgot: Score: 0.00051
key sign: Score: 0.00006
key sign option: Score: 0.00002
legacy twitter: Score: 0.00129
legacy twitter yes: Score: 0.00184
let give: Score: 0.00148
let give another: Score: 0.00000
like proceed: Score: 0.00242
like proceed legacy: Score: 0.00467
login mail: Score: 0.00047
login mail get: Score: 0.00034
login office: Score: 0.00132
login office login: Score: 0.00021
login outlook: Score: 0.00005
login outlook login: Score: 0.00026
login view: Score: 0.00097
login view shared: Score: 0.00031
mail get: Score: 0.00034
mail get file: Score: 0.00005
make sure: Score: 0.00131
microsoft account: Score: 0.00145
microsoft account doe: Score: 0.00000
microsoft term: Score: 0.00217
microsoft term privacy: Score: 0.00197
mycraftmail client: Score: 0.00065
mycraftmail client mycraftmail: Score: 0.00060
mycraftmail enter: Score: 0.00093
mycraftmail enter password: Score: 0.00063
mycraftmail username: Score: 0.00051
mycraftmail username password: Score: 0.00027
n access: Score: 0.00107
n access account: Score: 0.00117
n exist: Score: 0.00097
n exist enter: Score: 0.00113
need help: Score: 0.00184
next password: Score: 0.00021
next password account: Score: 0.00003
nobody mycraftmail: Score: 0.00446

nobody mycraftmail enter: Score: 0.00021
nthr unt: Score: 0.00030
nthr unt sign: Score: 0.00029
ntr sswrd: Score: 0.00051
office login: Score: 0.00121
office login outlook: Score: 0.00024
one drive: Score: 0.00240
one drive shared: Score: 0.00026
one sign: Score: 0.00004
one sign security: Score: 0.00024
option login: Score: 0.00002
option next: Score: 0.00009
option next password: Score: 0.00005
outlook login: Score: 0.00087
password account: Score: 0.00066
password account create: Score: 0.00005
password forgot: Score: 0.00217
password forgot password: Score: 0.00200
password keep: Score: 0.00302
password keep signed: Score: 0.00112
password privacy: Score: 0.00003
password privacy cooky: Score: 0.00001
password remember: Score: 0.02319
password sign: Score: 0.00386
password sign another: Score: 0.00009
password webmail: Score: 0.00305
password webmail mini: Score: 0.00240
phone number: Score: 0.00313
pick account: Score: 0.00024
pick account another: Score: 0.00015
please choose: Score: 0.00049
please choose email: Score: 0.00011
please contact: Score: 0.01307
please enable: Score: 0.00393
please enter: Score: 0.00507
please enter password: Score: 0.00228
please sign: Score: 0.00214
policy term: Score: 0.00323
privacy cooky: Score: 0.01085
privacy cooky term: Score: 0.00058
privacy cooky trm: Score: 0.00010
privacy policy: Score: 0.00637
privacy policy term: Score: 0.00257
proceed legacy: Score: 0.00146
proceed legacy twitter: Score: 0.00238
provider login: Score: 0.00175
provider login view: Score: 0.00008
rakuten inc: Score: 0.00124

rakuten inc right: Score: 0.00089
read document: Score: 0.00040
read document please: Score: 0.00065
right reserved: Score: 0.01005
security key: Score: 0.00003
security key sign: Score: 0.00005
share anyone: Score: 0.00084
share anyone one: Score: 0.00036
shared document: Score: 0.00088
shared document one: Score: 0.00039
shared file: Score: 0.00042
shared file login: Score: 0.00006
sign account: Score: 0.00525
sign account nobody: Score: 0.00015
sign account sign: Score: 0.00082
sign continue: Score: 0.00229
sign different: Score: 0.00447
sign email: Score: 0.00386
sign microsoft: Score: 0.00291
sign microsoft account: Score: 0.00050
sign nthr: Score: 0.00029
sign nthr unt: Score: 0.00000
sign option: Score: 0.00023
sign option login: Score: 0.00030
sign option next: Score: 0.00013
sign privacy: Score: 0.00056
sign privacy cooky: Score: 0.00127
sign security: Score: 0.00040
sign security key: Score: 0.00006
sign sign: Score: 0.00319
signed forgot: Score: 0.00128
signed forgot password: Score: 0.00131
something went: Score: 0.00309
something went wrong: Score: 0.00314
sswrd sign: Score: 0.00056
sswrd sign nthr: Score: 0.00081
stay signed: Score: 0.00203
term microsoft: Score: 0.00064
term privacy: Score: 0.00615
term privacy cooky: Score: 0.00795
timed please: Score: 0.00003
trm f: Score: 0.00013
trm f u: Score: 0.00017
twitter yes: Score: 0.00000
twitter yes something: Score: 0.00080
unt sign: Score: 0.00048
unt sign privacy: Score: 0.00074
username password: Score: 0.00662

```
username password webmail: Score: 0.00171
view shared: Score: 0.00030
view shared file: Score: 0.00037
webmail login: Score: 0.00375
webmail mini: Score: 0.00236
webmail welcome: Score: 0.00227
went wrong: Score: 0.00678
went wrong dont: Score: 0.00221
would like: Score: 0.00128
would like proceed: Score: 0.00075
wrong dont: Score: 0.00162
wrong dont fret: Score: 0.00169
yes something: Score: 0.00073
yes something went: Score: 0.00228
```



Hyperparameter Tuning

[ ]:
```python
# Number of trees
n_estimators = [int(x) for x in np.linspace(start = 200, stop = 2000, num = 10)]

# Number of features to consider at every split
max_features = ['auto', 'sqrt']
```

```
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(10, 110, num = 11)]
max_depth.append(None)

# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10]

# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 4]

# Method of selecting samples for training each tree
bootstrap = [True, False]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf,
               'bootstrap': bootstrap}
print(random_grid)
```

```
{'n_estimators': [200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000],
'max_features': ['auto', 'sqrt'], 'max_depth': [10, 20, 30, 40, 50, 60, 70, 80,
90, 100, 110, None], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2,
4], 'bootstrap': [True, False]}
```

[ ]:
```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rfr = RandomForestRegressor()

# Random search of parameters, using 3 fold cross validation,
# search across 10 different combinations, and use all available cores
rf_random = RandomizedSearchCV(
    estimator = rfr,
    param_distributions = random_grid,
    n_iter = 100,
    cv = 3,
    verbose = 2,
    random_state = 1,
    n_jobs = -1
)

# Fit the random search model
rf_random.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 100 candidates, totalling 300 fits
```

```
[ ]: RandomizedSearchCV(cv=3, estimator=RandomForestRegressor(), n_iter=100,
                        n_jobs=-1,
                        param_distributions={'bootstrap': [True, False],
                                             'max_depth': [10, 20, 30, 40, 50, 60,
                                                           70, 80, 90, 100, 110,
                                                           None],
                                             'max_features': ['auto', 'sqrt'],
                                             'min_samples_leaf': [1, 2, 4],
                                             'min_samples_split': [2, 5, 10],
                                             'n_estimators': [200, 400, 600, 800,
                                                              1000, 1200, 1400, 1600,
                                                              1800, 2000]},
                        random_state=1, verbose=2)
```

```
[ ]: rf_random.best_params_
```

```
[ ]: {'n_estimators': 1400,
      'min_samples_split': 2,
      'min_samples_leaf': 1,
      'max_features': 'sqrt',
      'max_depth': 60,
      'bootstrap': True}
```

```
[ ]: rf_rr = RandomForestClassifier(
        n_estimators = 1400,
        min_samples_split = 2,
        min_samples_leaf = 1,
        max_features = 'sqrt',
        max_depth = 60,
        bootstrap = True
    )
```

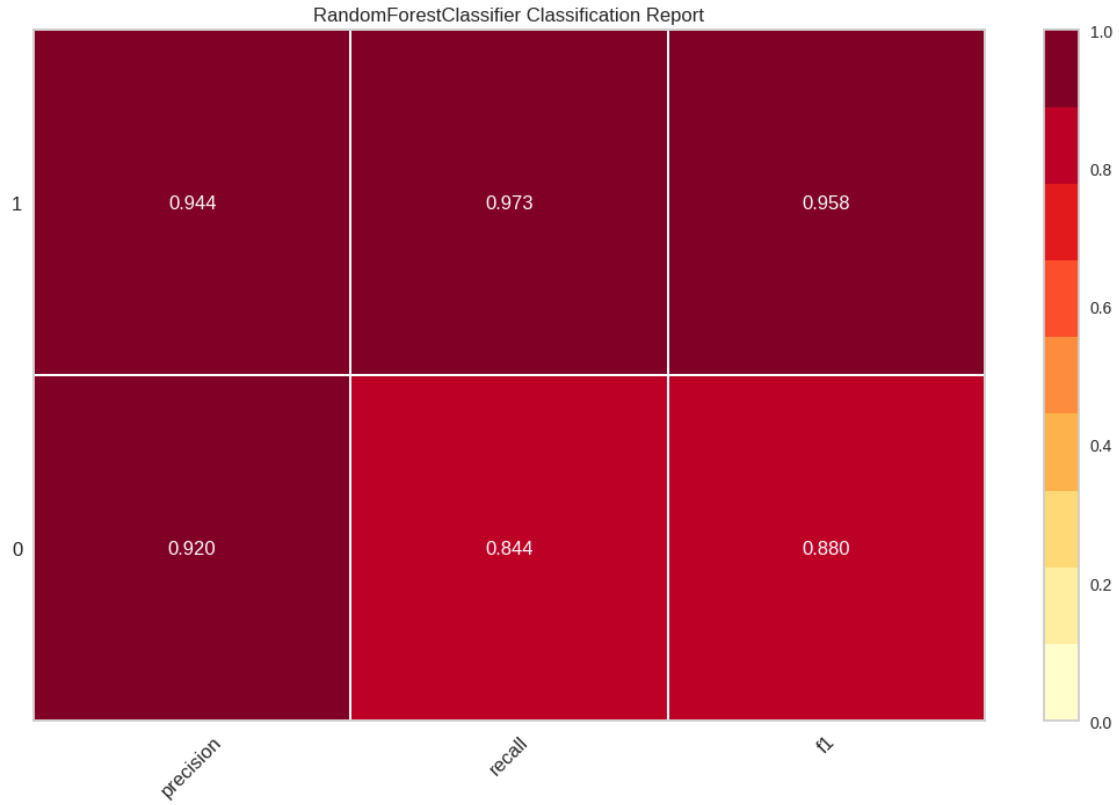
```
[ ]: model = rf_rr
```

```
[ ]: display_accuracy(model)
```

```
Training time: 16.37372660636902
Test accuracy : 93.79%
```

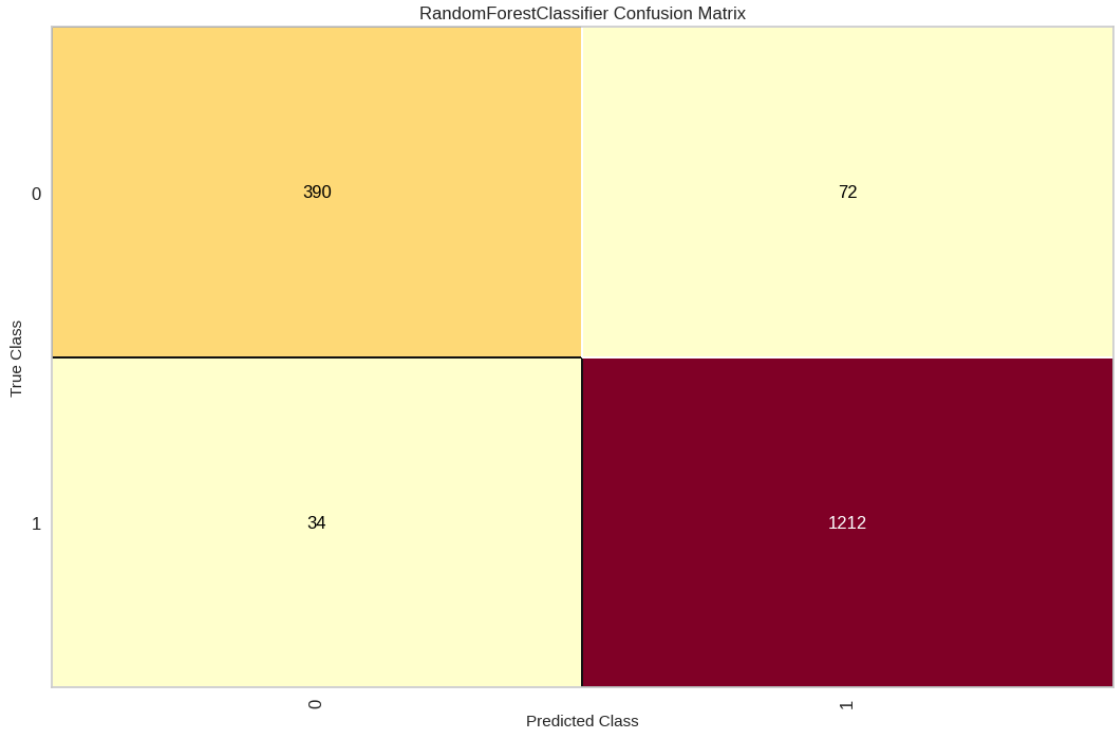
```
[ ]: display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
```



1.13.8 Ridge Regression Classifier

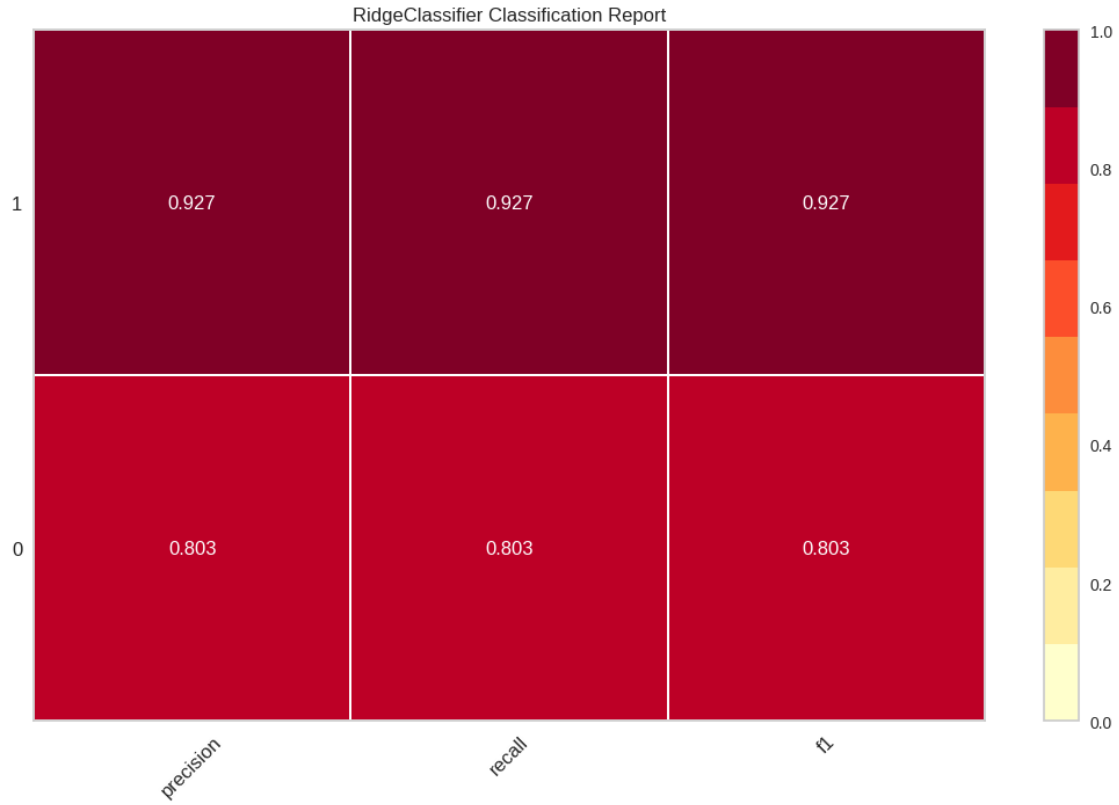
[ ]:
```
model = rg
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 0.08659052848815918
Test accuracy : 89.34%
```

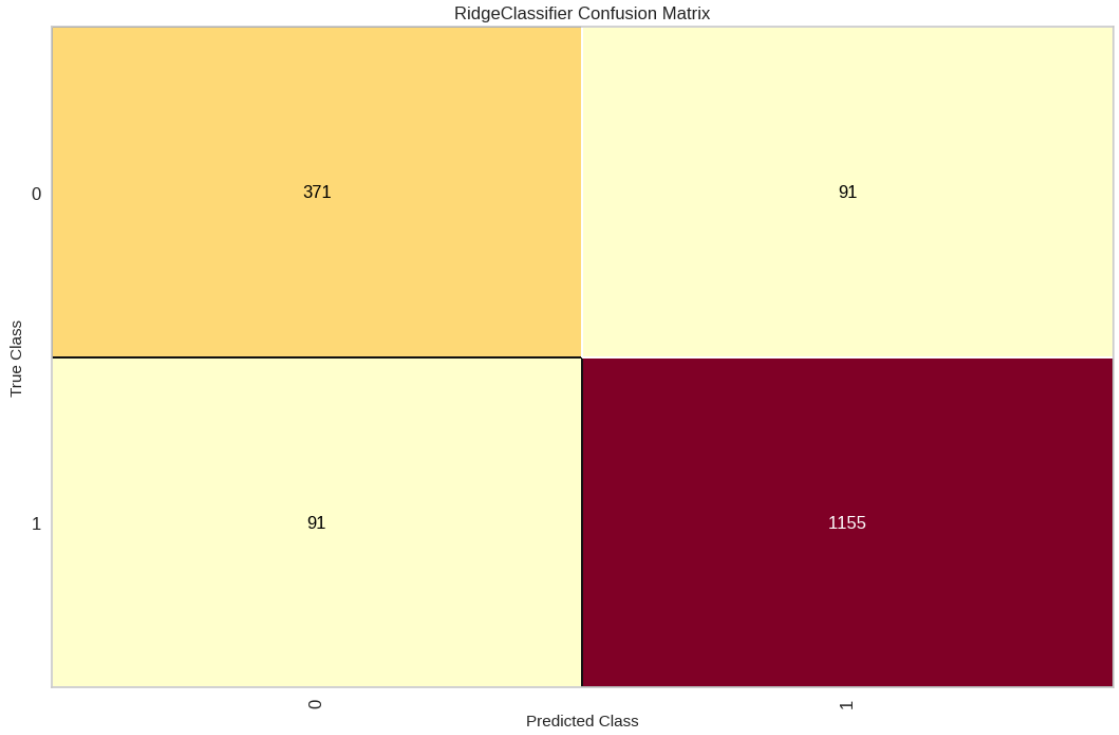
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RidgeClassifier was fitted with feature names
  warnings.warn(
```



1.13.9 XGBoost Classifier

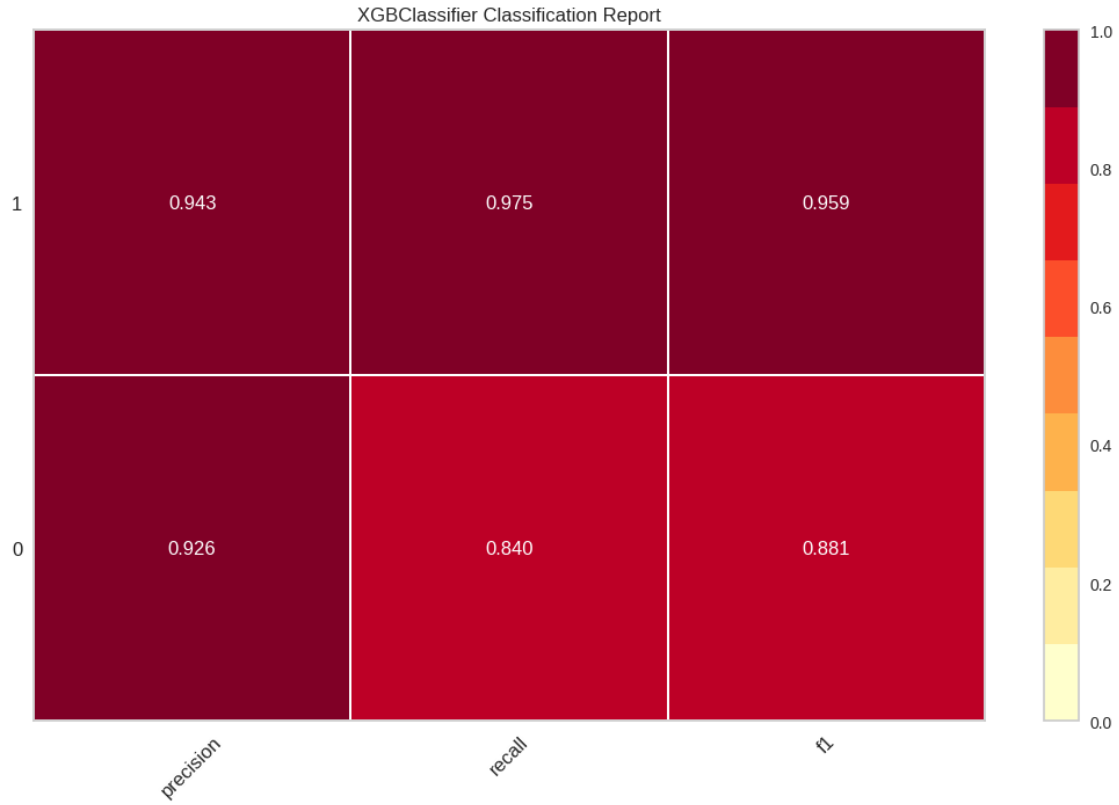
[ ]:
```
model = xgb
```

[ ]:
```
display_accuracy(model)
```

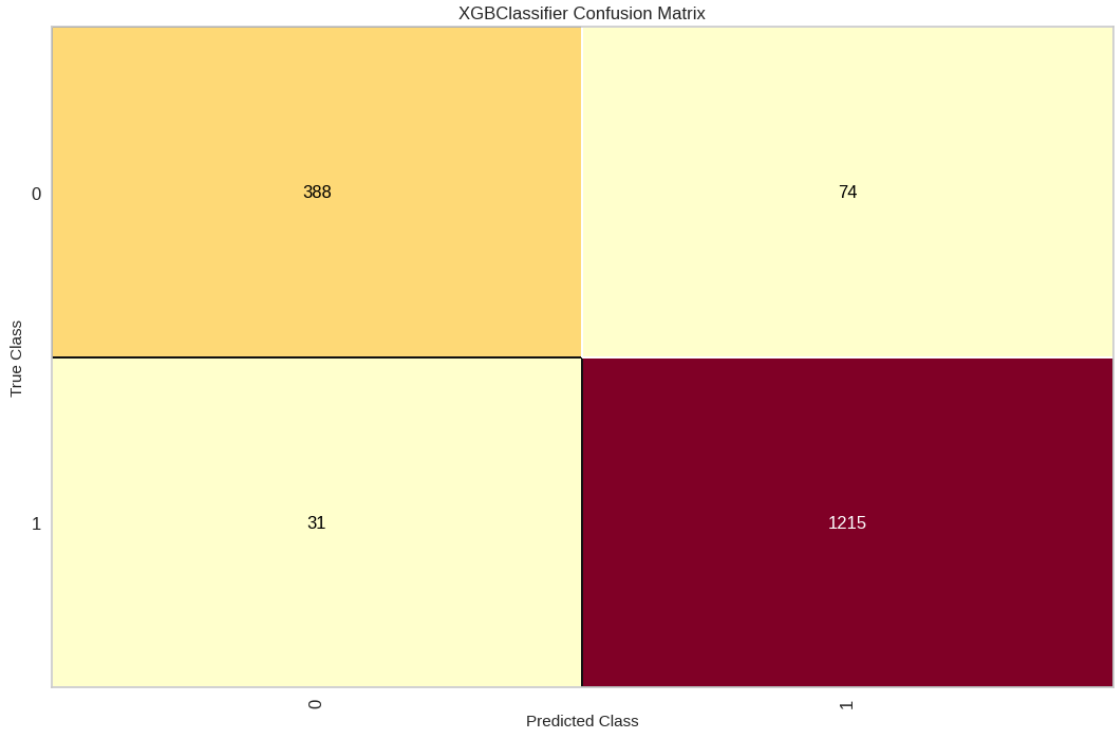
```
Training time: 0.482086181640625
Test accuracy : 93.85%
```

[ ]:
```
display_classification_report(model)
```

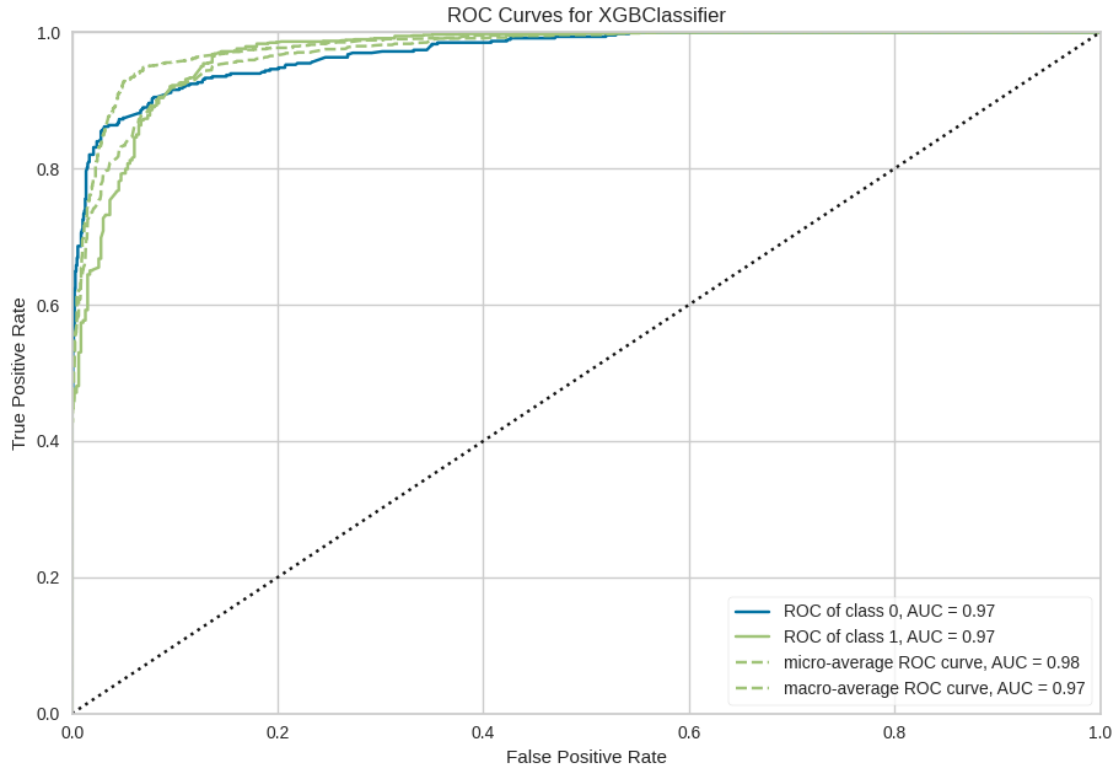
```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

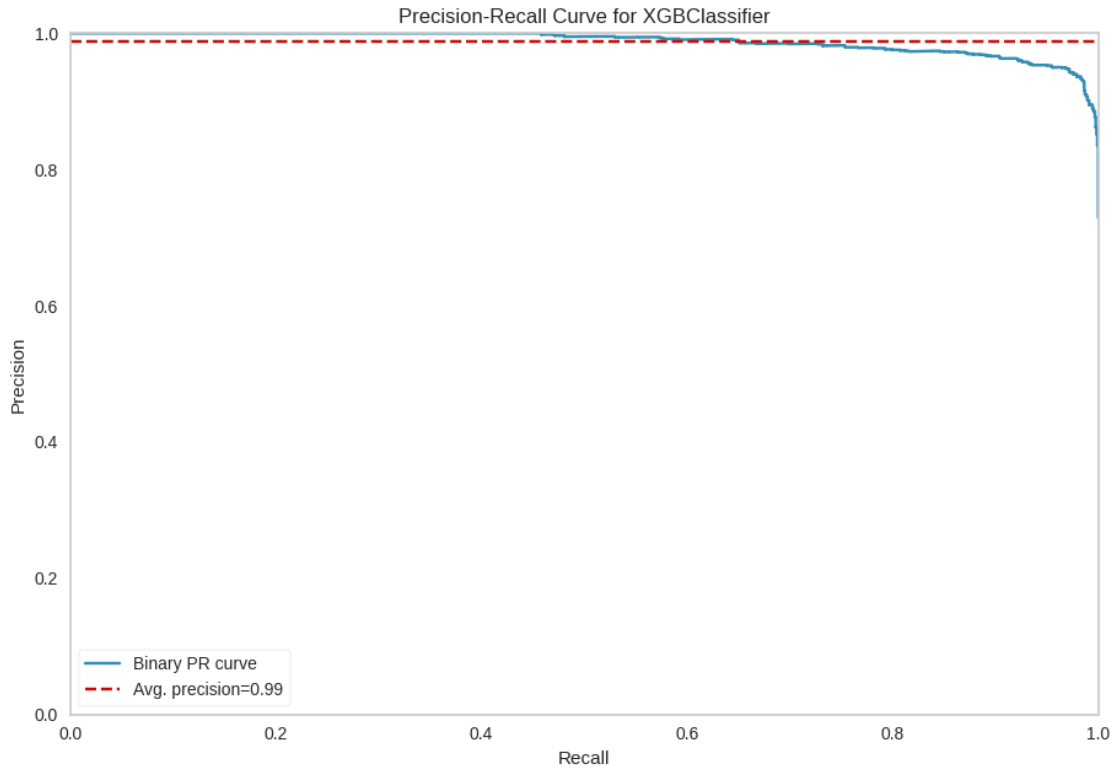


[ ]:
```
display_roc_curve(model)
```

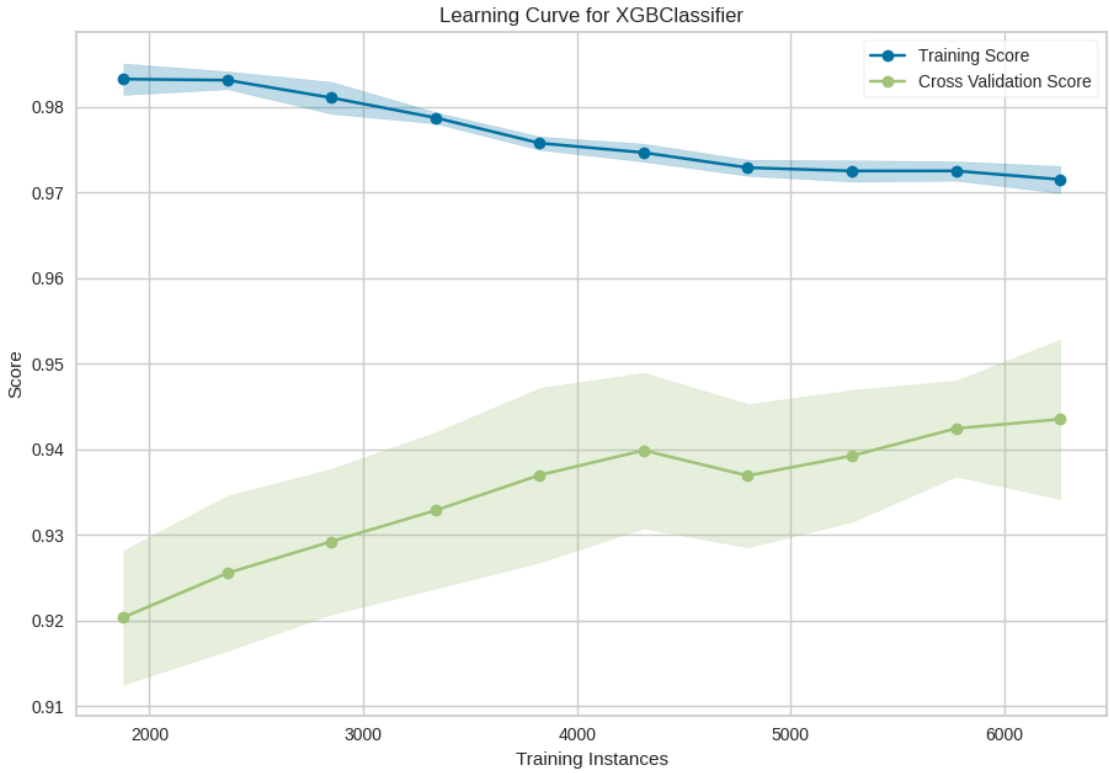


[ ]:
```
display_precision_recall_curve(model)
```

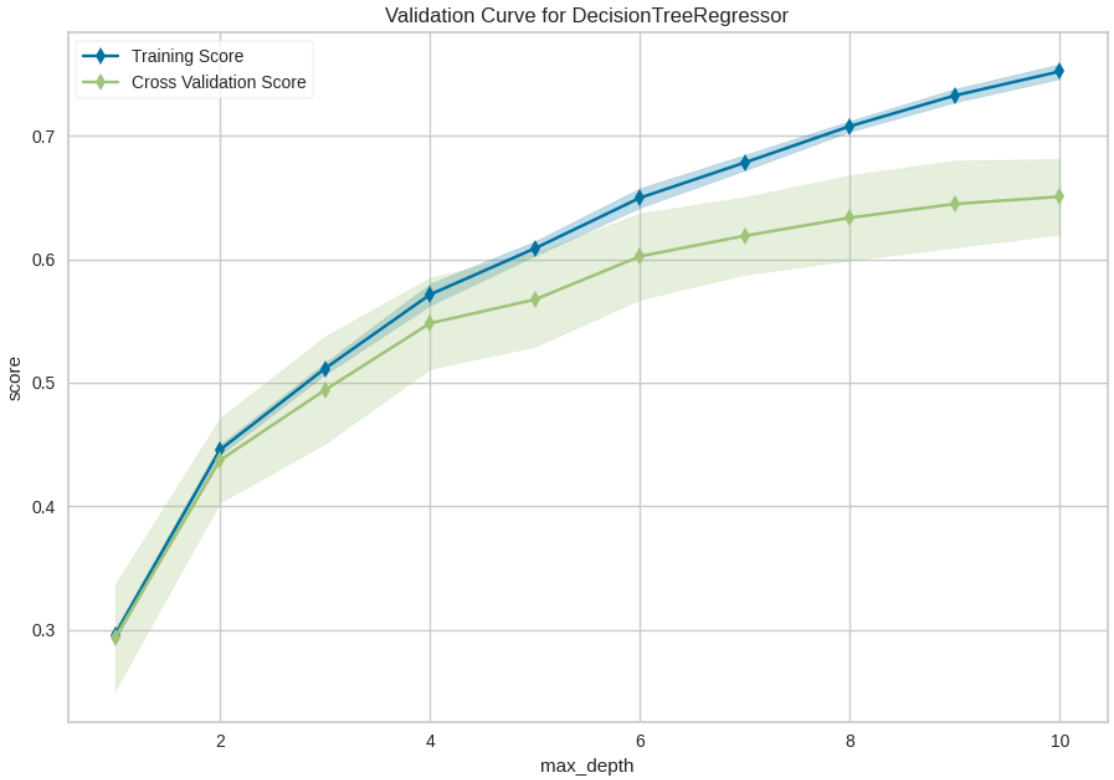
```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/prcurve.py:254:
YellowbrickWarning: micro=True is ignored;specify per_class=False to draw a PR
curve after micro-averaging
  warnings.warn(
```



[ ]:
```
display_learning_curve(model)
```



[ ]:
```
display_validation_curve(model)
```



1.14 Ensemble Learning

1.14.1 Bagging

[ ]:
```
clf_array = [rf, et, knn, svc, rg]

for clf in clf_array:
    vanilla_scores = cross_val_score(clf, X, y, cv = 10, n_jobs = -1)
    bagging_clf = BaggingClassifier(clf, max_samples = 0.4, max_features = 10,␣
 ↪random_state = seed)
    bagging_scores = cross_val_score(bagging_clf, X, y, cv = 10, n_jobs = -1)

    print("Mean of: {1:.3f}, std: (+/-) {2:.3f} [{0}]".format(clf.__class__.
 ↪__name__, vanilla_scores.mean(), vanilla_scores.std()))
    print("Mean of: {1:.3f}, std: (+/-) {2:.3f} [Bagging {0}]\n".format(clf.
 ↪__class__.__name__, bagging_scores.mean(), bagging_scores.std()))
```

```
Mean of: 0.944, std: (+/-) 0.007 [RandomForestClassifier]
Mean of: 0.751, std: (+/-) 0.004 [Bagging RandomForestClassifier]

Mean of: 0.942, std: (+/-) 0.007 [ExtraTreesClassifier]
Mean of: 0.751, std: (+/-) 0.004 [Bagging ExtraTreesClassifier]
```

```
Mean of: 0.855, std: (+/-) 0.014 [KNeighborsClassifier]
Mean of: 0.751, std: (+/-) 0.004 [Bagging KNeighborsClassifier]

Mean of: 0.796, std: (+/-) 0.046 [LinearSVC]
Mean of: 0.751, std: (+/-) 0.004 [Bagging LinearSVC]

Mean of: 0.896, std: (+/-) 0.009 [RidgeClassifier]
Mean of: 0.751, std: (+/-) 0.004 [Bagging RidgeClassifier]
```

1.14.2 Set up voting

[ ]:
```
eclf = VotingClassifier(estimators=[('Random Forests', rf),
                                    ('Extra Trees', et),
                                    ('Ridge Classifier', rg)],
                        voting = 'hard')

for clf, label in zip([rf, et, rg, eclf], ['Random Forest', 'Extra Trees',␣
 ↪'Ridge Classifier', 'Ensemble']):
    scores = cross_val_score(clf, X, y, cv = 10, scoring = 'accuracy')
    print("Mean: {0:.3f}, std: (+/-) {1:.3f} [{2}]".format(scores.mean(),␣
 ↪scores.std(), label))
```

```
Mean: 0.944, std: (+/-) 0.007 [Random Forest]
Mean: 0.942, std: (+/-) 0.007 [Extra Trees]
Mean: 0.896, std: (+/-) 0.009 [Ridge Classifier]
Mean: 0.942, std: (+/-) 0.008 [Ensemble]
```

1.14.3 Set up ensemble voting for bagging

[ ]:
```
ebclf_array = []

for clf in clf_array:
    ebclf_array.append(BaggingClassifier(clf, max_samples = 0.25, max_features␣
 ↪= 10, random_state = seed))

v_eclf = VotingClassifier(estimators=[('Bagging Random Forest',␣
 ↪BaggingClassifier(rf, max_samples = 0.25, max_features = 10, random_state =␣
 ↪seed)),
                                      ('Bagging Extra Trees',␣
 ↪BaggingClassifier(et, max_samples = 0.25, max_features = 10, random_state =␣
 ↪seed)),
                                      ('Bagging Ridge Classifier',␣
 ↪BaggingClassifier(rg, max_samples = 0.25, max_features = 10, random_state =␣
 ↪seed))],
                        voting='hard')
```

```
ebclf_array.append(v_eclf)

for clf, label in zip(ebclf_array, ['Bagging Random Forest',
                                    'Bagging Extra Trees',
                                    'Bagging Ridge Classifier',
                                    'Bagging Ensemble']):
    scores = cross_val_score(clf, X, y, cv = 10, scoring = 'accuracy')
    print("Mean: {0:.3f}, std: (+/-) {1:.3f} [{2}]".format(scores.mean(),␣
 ↪scores.std(), label))
```

```
Mean: 0.751, std: (+/-) 0.004 [Bagging Random Forest]
Mean: 0.751, std: (+/-) 0.004 [Bagging Extra Trees]
Mean: 0.763, std: (+/-) 0.016 [Bagging Ridge Classifier]
Mean: 0.751, std: (+/-) 0.004 [Bagging Ensemble]
```

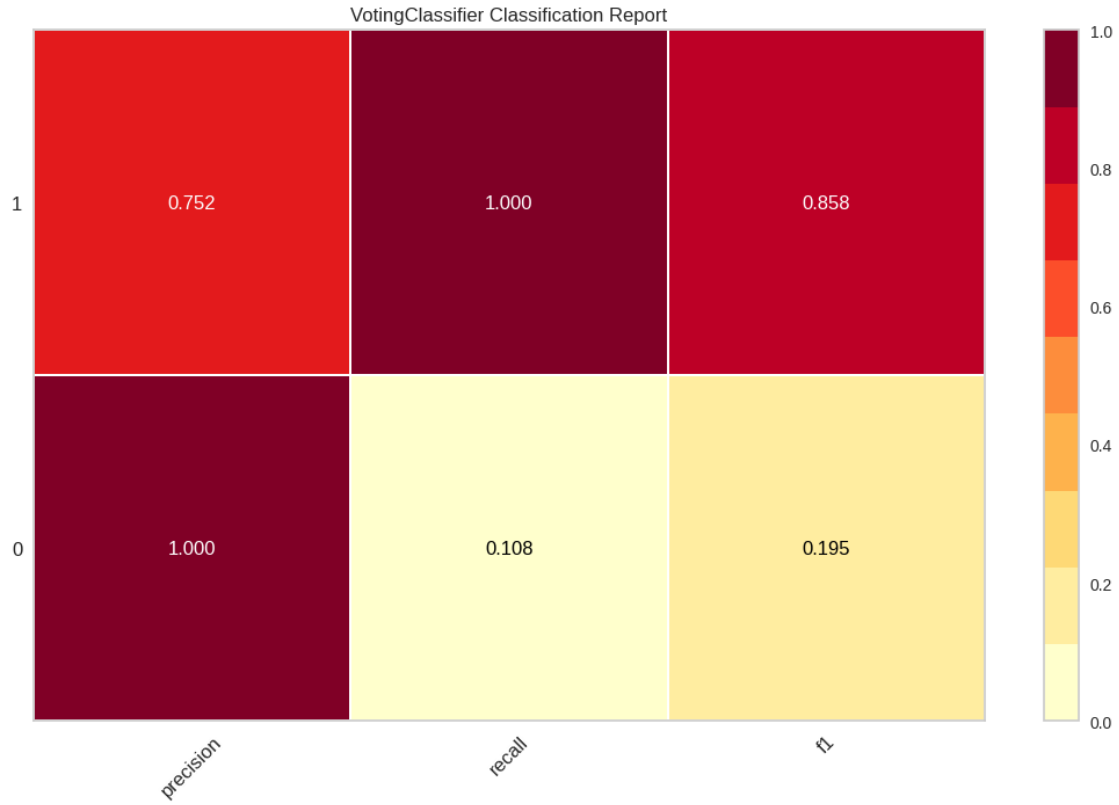
[ ]:
```
model = v_eclf
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 4.152900457382202
Test accuracy : 75.88%
```

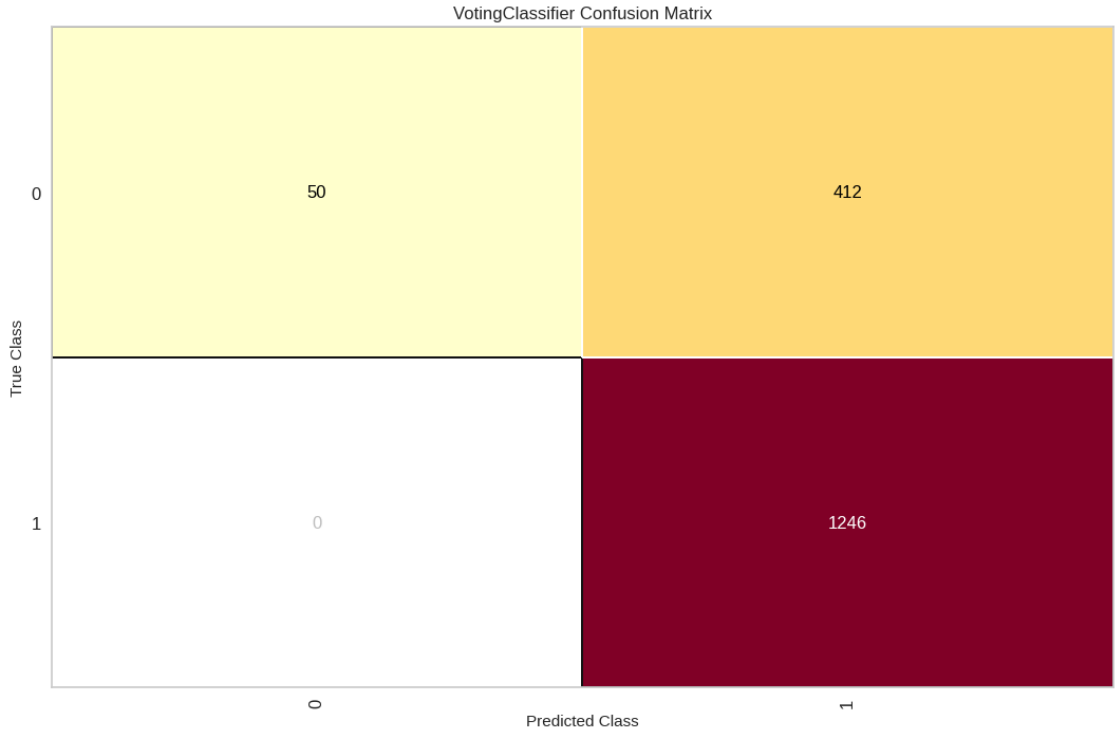
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but BaggingClassifier was fitted with feature
names
  warnings.warn(
```



1.14.4 Boosting

[ ]:
```
boost_array = [ab, gb, xgb]

eclf = EnsembleVoteClassifier(clfs = [ab, gb, xgb], voting = 'hard')

labels = ['Ada Boost', 'Grad Boost', 'XG Boost', 'Ensemble']

for clf, label in zip([ab, gb, xgb, eclf], labels):
    scores = cross_val_score(clf, X, y, cv = 10, scoring = 'accuracy')
    print("Mean: {0:.3f}, std: (+/-) {1:.3f} [{2}]".format(scores.mean(),␣
 ↪scores.std(), label))
```

```
Mean: 0.904, std: (+/-) 0.005 [Ada Boost]
Mean: 0.916, std: (+/-) 0.008 [Grad Boost]
Mean: 0.942, std: (+/-) 0.005 [XG Boost]
Mean: 0.920, std: (+/-) 0.007 [Ensemble]
```

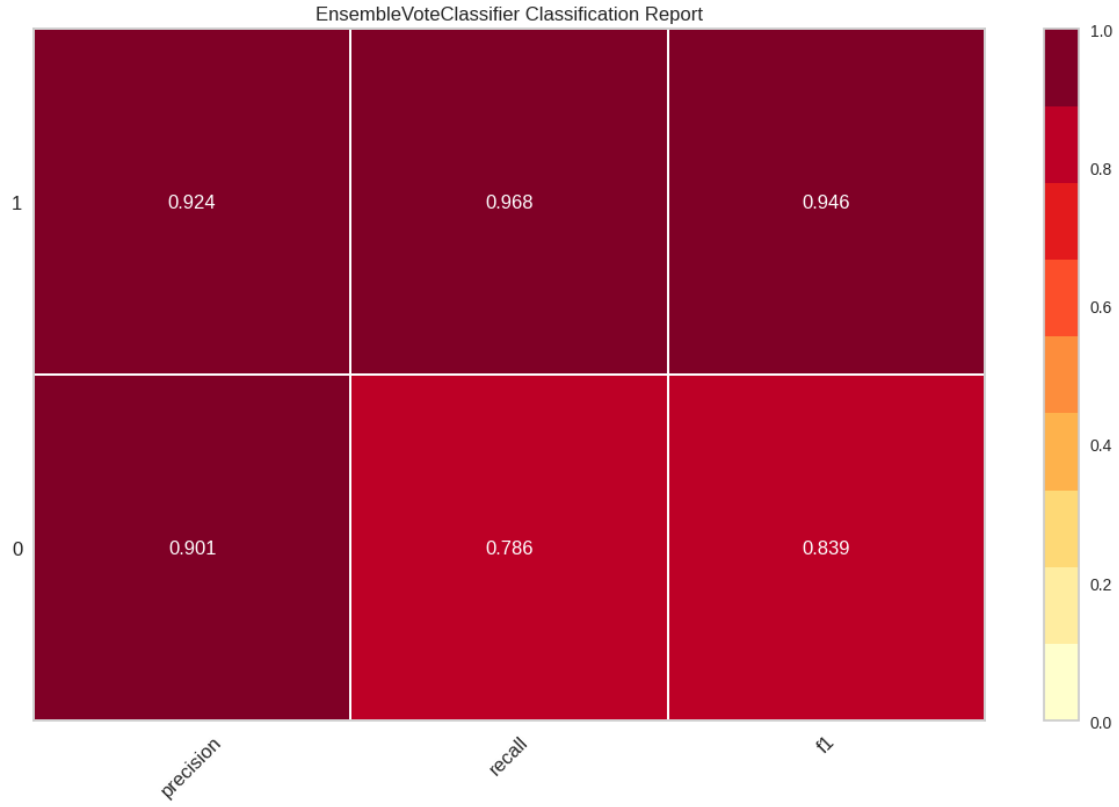
[ ]:
```
model = eclf
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 4.291194915771484
Test accuracy : 91.86%
```

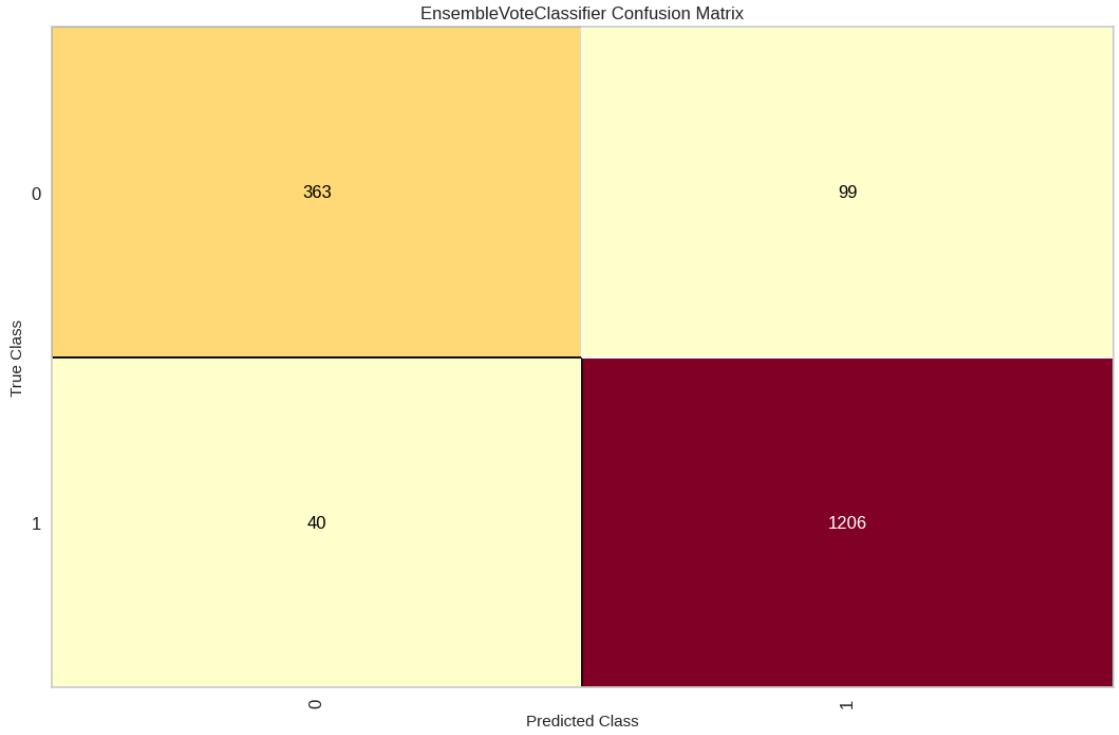
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but AdaBoostClassifier was fitted with feature
names
  warnings.warn(
```



1.14.5 Stacking

[ ]:
```
estimators = [
  ('et', et),
  ('rf', rf),
  ('xgb', xgb)
]

s_eclf = StackingClassifier(
  estimators = estimators,
  final_estimator = et
)

s_eclf.fit(X_train, y_train)
```

[ ]:
```
StackingClassifier(estimators=[('et', ExtraTreesClassifier(random_state=42)),
                               ('rf', RandomForestClassifier(random_state=42)),
                               ('xgb',
                                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=None,
                                                feature_types=…
                                                interaction_constraints=None,
                                                learning_rate=None, max_bin=None,
                                                max_cat_threshold=None,
                                                max_cat_to_onehot=None,
                                                max_delta_step=None,
                                                max_depth=None, max_leaves=None,
                                                min_child_weight=None,
                                                missing=nan,
                                                monotone_constraints=None,
                                                multi_strategy=None,
                                                n_estimators=None, n_jobs=None,
                                                num_parallel_tree=None,
                                                random_state=None, …))],
                    final_estimator=ExtraTreesClassifier(random_state=42))
```

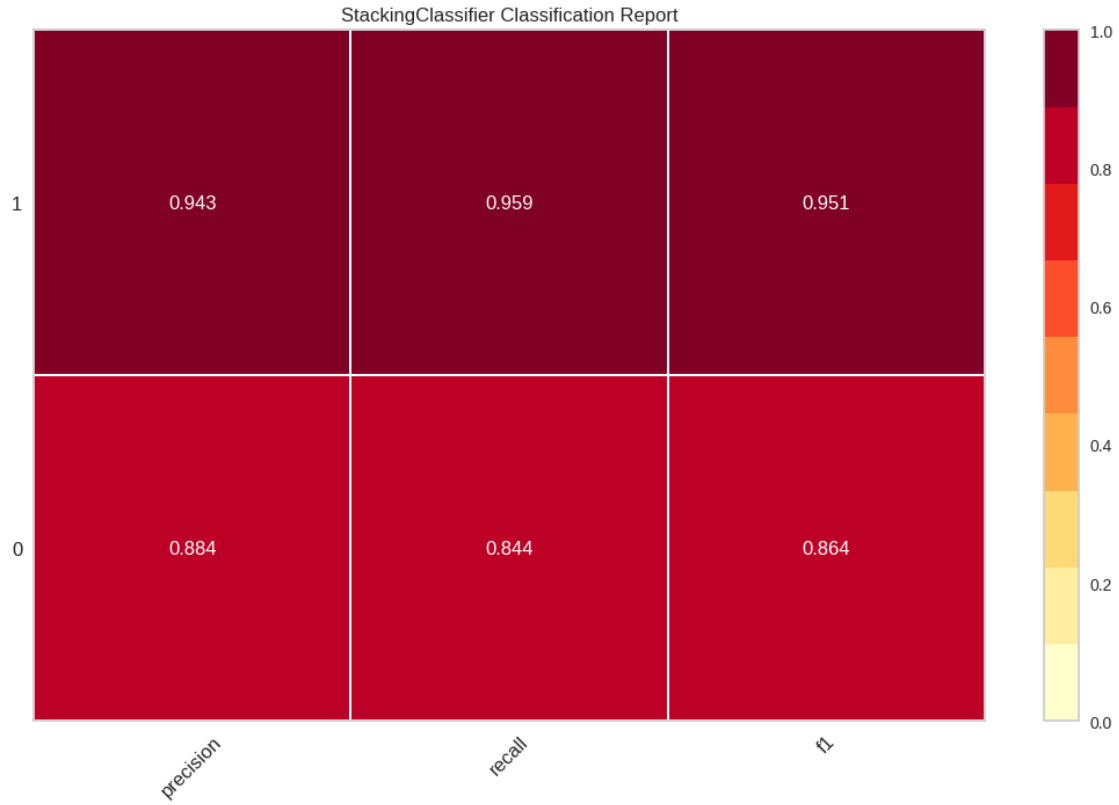
[ ]:
```
model = s_eclf
```

[ ]:
```
display_accuracy(model)
```

```
Training time: 15.832176685333252
Test accuracy : 92.80%
```

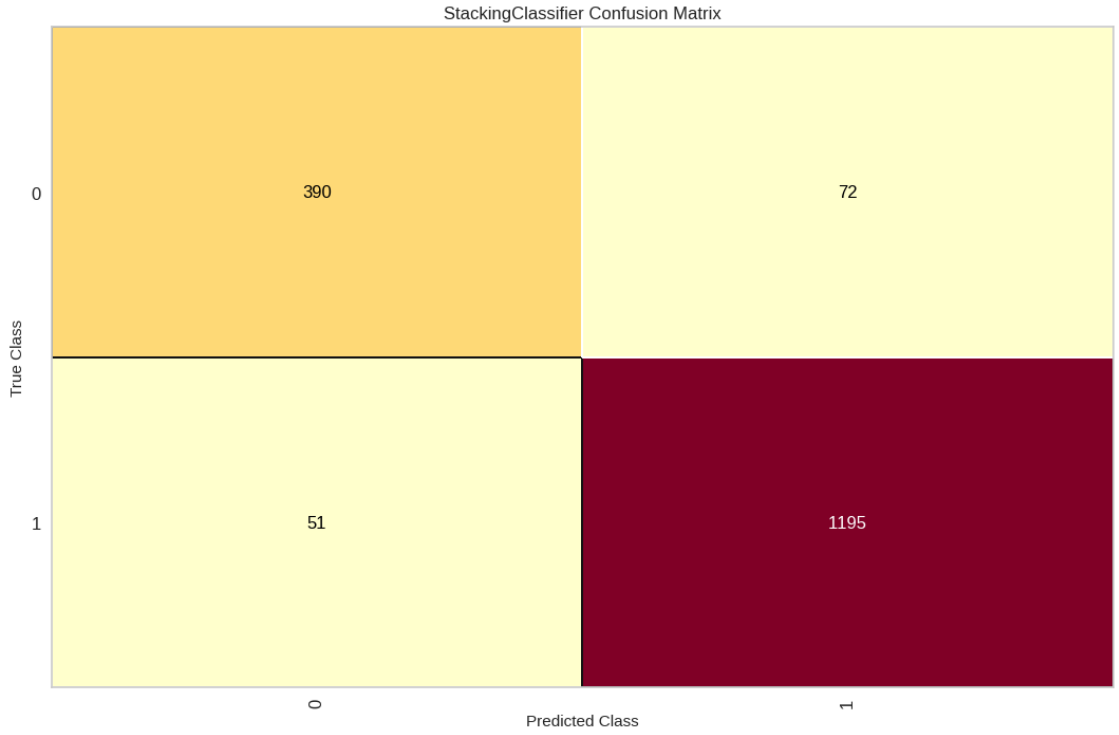
[ ]:
```
display_classification_report(model)
```

```
/usr/local/lib/python3.10/dist-packages/yellowbrick/classifier/base.py:232:
YellowbrickWarning: could not determine class_counts_ from previously fitted
classifier
  warnings.warn(
```



[ ]:
```
display_confusion_matrix(model)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but ExtraTreesClassifier was fitted with feature
names
  warnings.warn(
```



1.15 Save Model

[ ]:
```
# Save the model to file
filename = 'finalized_model.pkl'

with open(filename, 'wb') as file:
  pickle.dump(xgb, file)
```

1.16 Load the Model

[ ]:
```
with open(filename, 'rb') as file:
  loaded_model = pickle.load(file)

loaded_model
```

[ ]:
```
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=None, 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_depth=None, max_leaves=None,
```

```
min_child_weight=None, missing=nan, monotone_constraints=None,
multi_strategy=None, n_estimators=None, n_jobs=None,
num_parallel_tree=None, random_state=None, …)
```

1.16.1 Conclusions

This experiment focused on enhancing phishing website detection using semi-supervised learning, the implementation of the XGBoost model has led to significant achievements. The model demonstrated a high level of accuracy with a test score of 93.85% and an AUC of 0.97, effectively distinguishing between phishing and legitimate websites.

A variety of models were explored, including AdaBoost, ExtraTrees, Gradient Boosting, K-Nearest Neighbors, Logistic Regression, RandomForest, Ridge Classifier, Linear SVC, and XGBoost. The outstanding performance of the XGBoost model can be attributed to its proficiency in handling sparse data, its built-in regularization features, efficient tree pruning capabilities, and the ability to manage missing data effectively.

The learning curve observed during the experiment offers additional insights. The training score started high at just over 0.98 and slightly decreased to 0.99, indicating a very high level of accuracy achieved by the model on the training data. This decrease suggests that as more data was introduced, the model became slightly less precise on the training set, which is a typical behavior as the model starts to generalize from the training data to unseen data.

Simultaneously, the cross-validation score, which began at 0.92, increased to 0.945. This improvement in the cross-validation score as more data was added indicates that the model was successfully learning and generalizing from the training data to the validation data. It's a positive sign that the model is not overfitting and is capable of maintaining its performance on unseen data.

These learning curve trends are essential as they demonstrate the model's ability to learn effectively from the data and its robustness in handling new, unseen data, which is critical in the ever-changing landscape of cybersecurity threats like phishing.

While the current results are promising, the dynamic nature of cyber threats, particularly phishing attacks, necessitates ongoing model adaptation and improvement. Future work should focus on continuous experimentation, refinement of features, and model optimization to further enhance the phishing detection capabilities.

To conclude, this individual project has not only proved the feasibility of using semi-supervised learning for effective phishing detection but has also laid a strong groundwork for future research in this crucial area of cybersecurity. The high performance of the XGBoost model, in particular, is a promising basis for developing more advanced, accurate, and efficient anti-phishing tools.