365Project

April 27, 2024

#Fake News Detection Using Logistic Regression and Convolutional Neural Networks

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import re
from tqdm import tqdm
import gensim
from gensim.models import word2vec
from gensim.models.word2vec import Word2Vec
import numpy as np
import spacy
import string
import sklearn
from sklearn.model_selection import train_test_split
```

[ ]:
```
import gensim
from gensim.models import word2vec
from gensim.models.word2vec import Word2Vec
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import spacy
import string
```

[ ]:
```
!pip install keras-tuner
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv1D, MaxPooling1D,␣
  ↪Embedding
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from gensim.models import KeyedVectors
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow import keras
from kerastuner import HyperModel
from keras_tuner import RandomSearch
from keras_tuner.engine.hyperparameters import HyperParameters
import tensorflow as tf
from tensorflow.keras.utils import plot_model
```

```
Collecting keras-tuner
  Downloading keras_tuner-1.4.7-py3-none-any.whl (129 kB)
                           129.1/129.1

kB 765.9 kB/s eta 0:00:00
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-
packages (from keras-tuner) (2.15.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from keras-tuner) (24.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from keras-tuner) (2.31.0)
Collecting kt-legacy (from keras-tuner)
  Downloading kt_legacy-1.0.5-py3-none-any.whl (9.6 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->keras-tuner) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->keras-tuner) (2024.2.2)
Installing collected packages: kt-legacy, keras-tuner
Successfully installed keras-tuner-1.4.7 kt-legacy-1.0.5
```

```
<ipython-input-3-1ce6282d6374>:10: DeprecationWarning: `import kerastuner` is
deprecated, please use `import keras_tuner`.
  from kerastuner import HyperModel
```

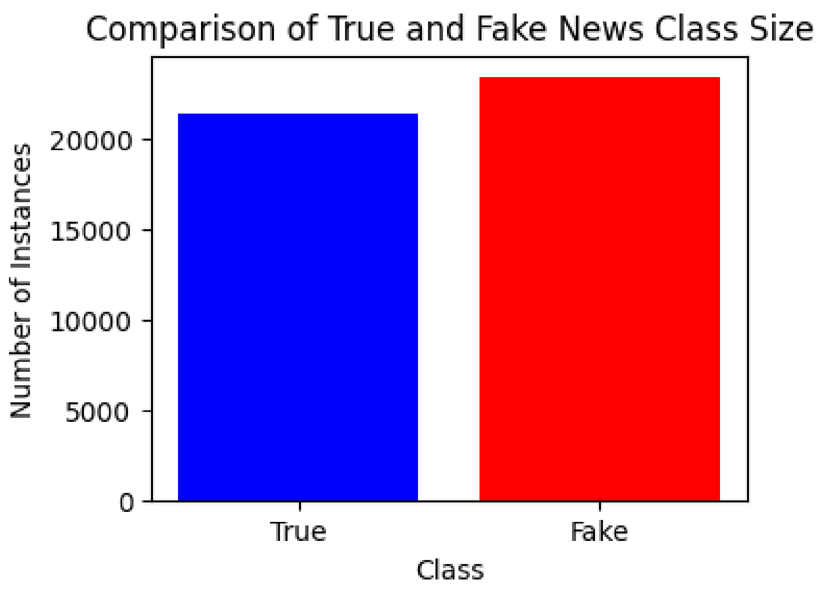
0.1 Exploratory Data Analysis

```
[ ]: from google.colab import drive
drive.mount('/content/drive')

true_csv_path = "/content/drive/My Drive/365Project/ISOT_dataset/True.csv"
fake_csv_path = "/content/drive/My Drive/365Project/ISOT_dataset/Fake.csv"

df_true = pd.read_csv(true_csv_path)
len_df_true = len(df_true)
df_fake = df_fake = pd.read_csv(fake_csv_path)
len_df_fake = len(df_fake)

data = {'Class': ['True', 'Fake'], 'Length': [len_df_true, len_df_fake]}
data = pd.DataFrame(data)
data.head()
```

```
Mounted at /content/drive
```

[ ]:
```
  Class  Length
0  True   21417
1  Fake   23481
```

[ ]:
```
plt.figure(figsize=(4, 3))
plt.bar(data['Class'], data['Length'], color=['blue', 'red'])
plt.title('Comparison of True and Fake News Class Size')
plt.xlabel('Class')
plt.ylabel('Number of Instances')
plt.show()
```



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

[ ]:
```
                                               title  \
0  As U.S. budget fight looms, Republicans flip t…
1  U.S. military to accept transgender recruits o…
2  Senior U.S. Republican senator: 'Let Mr. Muell…
3  FBI Russia probe helped by Australian diplomat…
4  Trump wants Postal Service to charge 'much mor…

                                                text       subject  \
0  WASHINGTON (Reuters) - The head of a conservat…  politicsNews
```

```
1  WASHINGTON (Reuters) - Transgender people will…  politicsNews
2  WASHINGTON (Reuters) - The special counsel inv…  politicsNews
3  WASHINGTON (Reuters) - Trump campaign adviser …  politicsNews
4  SEATTLE/WASHINGTON (Reuters) - President Donal…  politicsNews

                 date
0  December 31, 2017
1  December 29, 2017
2  December 31, 2017
3  December 30, 2017
4  December 29, 2017
```

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

[ ]:
```
                                              title  \
0   Donald Trump Sends Out Embarrassing New Year'…
1   Drunk Bragging Trump Staffer Started Russian …
2   Sheriff David Clarke Becomes An Internet Joke…
3   Trump Is So Obsessed He Even Has Obama's Name…
4   Pope Francis Just Called Out Donald Trump Dur…

                                              text subject  \
0  Donald Trump just couldn t wish all Americans …    News
1  House Intelligence Committee Chairman Devin Nu…    News
2  On Friday, it was revealed that former Milwauk…    News
3  On Christmas day, Donald Trump announced that …    News
4  Pope Francis used his annual Christmas Day mes…    News

                date
0  December 31, 2017
1  December 31, 2017
2  December 30, 2017
3  December 29, 2017
4  December 25, 2017
```

[ ]:
```
count_reuters_true = df_true['text'].str.contains('Reuters', case=False,␣
 ↪na=False).sum()

count_reuters_fake = df_fake['text'].str.contains('Reuters', case=False,␣
 ↪na=False).sum()

print(f'Number of times "Reuters" is found in True news dataset:␣
 ↪{count_reuters_true}')
print(f'Number of times "Reuters" is found in Fake news dataset:␣
 ↪{count_reuters_fake}')
```

```
Number of times "Reuters" is found in True news dataset: 21378
Number of times "Reuters" is found in Fake news dataset: 322
```

```
[ ]: twitter_occurrences = df_fake['text'].str.contains('twitter.com').sum()
     potus_occurrences = df_fake['text'].str.contains('@potus').sum()
     dt_occurrences = df_fake['text'].str.contains('@realdonaldtrump').sum()
     url_occurrences = df_fake['text'].str.count(r'http[s]?://(?:
       ↪[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+').sum()

     pattern1 = r'(?:\.\s*|\s)featured image via .*? images'
     occurrences1 = df_fake['text'].str.contains(pattern1).sum()

     pattern2 = r'(?:\.\s*| )featured image via .*? getty images(?: for.*?)?'
     occurrences2 = df_fake['text'].str.contains(pattern2).sum()

     pattern3 = r'featured image.*$'
     occurrences3 = df_fake['text'].str.contains(pattern3).sum()

     print(f"Occurrences of 'twitter.com': {twitter_occurrences}")
     print(f"Occurrences of '@potus': {potus_occurrences}")
     print(f"Occurrences of '@realdonaldtrump': {dt_occurrences}")
     print(f"Occurrences of pattern1: {occurrences1}")
     print(f"Occurrences of pattern2: {occurrences2}")
     print(f"Occurrences of pattern3: {occurrences3}")
     print(f"Occurrences of urls: {url_occurrences}")
```

```
Occurrences of 'twitter.com': 3615
Occurrences of '@potus': 8
Occurrences of '@realdonaldtrump': 57
Occurrences of pattern1: 0
Occurrences of pattern2: 0
Occurrences of pattern3: 36
Occurrences of urls: 4672
```

0.2 Data Pre-processing

0.2.1 Removing News Source From Text

```
[ ]: df_true['text'] = df_true['text'].str.replace(r'^.*?\(Reuters\)\s*-\s*', '',␣
       ↪regex=True)
```

0.2.2 Creating the main dataframe

```
[ ]: fake_df = df_fake[['title', 'text']]
     true_df = df_true[['title', 'text']]

     fake_df['class'] = 0
     true_df['class'] = 1
```

```
df = pd.concat([fake_df, true_df], ignore_index=True, sort=False)
df['title_text'] = df['title'] + ' ' + df['text']
df.drop(['title', 'text'], axis=1, inplace=True)

df.head()
```

```
<ipython-input-11-436b98bf7589>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  fake_df['class'] = 0
<ipython-input-11-436b98bf7589>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  true_df['class'] = 1
```

[ ]:
```
   class                                         title_text
0      0  Donald Trump Sends Out Embarrassing New Year'…
1      0  Drunk Bragging Trump Staffer Started Russian …
2      0  Sheriff David Clarke Becomes An Internet Joke…
3      0  Trump Is So Obsessed He Even Has Obama's Name…
4      0  Pope Francis Just Called Out Donald Trump Dur…
```

0.2.3 Change Text to Lowercase

[ ]:
```
def to_lowercase(text):
    return text.lower()

df['title_text'] = df['title_text'].apply(to_lowercase)
```

0.2.4 Removing non-textual elements

[ ]:
```
def remove_non_textual_elements(text):
    # Remove URLs
    text = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:
↪%[0-9a-fA-F][0-9a-fA-F]))+', '', text)
    # Remove hashtags
    text = re.sub(r'#\w+', '', text)
    # Remove mentions
    text = re.sub(r'@\w+', '', text)
    # Remove markdown links - (http://url.com)
    text = re.sub(r'\[.*?\]\(.*?\)', '', text)
    # Remove specific patterns
    text = re.sub(r'featured image.*$', '', text)
    # remove 'twitter.com'
    text = re.sub(r'twitter\.com', '', text, flags=re.IGNORECASE)
    return text

df['title_text'] = df['title_text'].apply(remove_non_textual_elements)
df = df[~df['title_text'].str.contains('twitter.com')]
```

0.2.5 Remove numbers & punctuation

[ ]:
```
def remove_numbers_punctuation(text):
    return re.sub(r'\d+|[^a-zA-Z\s]', '', text)

df['title_text'] = df['title_text'].apply(remove_numbers_punctuation)
```

0.2.6 Tokenize

[ ]:
```
nltk.download('punkt')
nltk.download('stopwords')
```

```
[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.
```

[ ]:
```
True
```

[ ]:
```
def tokenize(text):
    return word_tokenize(text)

df['tokens'] = df['title_text'].apply(tokenize)
```

0.2.7 Remove stopwords

[ ]:
```
stop_words = stopwords.words('english')

def remove_stopwords(tokens):
    return [word for word in tokens if word not in stop_words]

df['tokens'] = df['tokens'].apply(remove_stopwords)
```

0.2.8 Stemming

```
[ ]: stemmer = PorterStemmer()

     def stem(tokens):
         return [stemmer.stem(word) for word in tokens]

     df['stemmed_tokens'] = df['tokens'].apply(stem)
```

0.2.9 Join Stemmed Tokens

```
[ ]: def tokens_to_string(tokens):
         return ' '.join(tokens)

     df['clean_text'] = df['stemmed_tokens'].apply(tokens_to_string)
```

```
[ ]: df = df[['class','clean_text']]
     df.dropna(subset=['clean_text'], inplace=True)
     df.head()
```

```
<ipython-input-20-9f83ff90a13f>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.dropna(subset=['clean_text'], inplace=True)
```

```
[ ]:    class                                        clean_text
     0      0  donald trump send embarrass new year eve messa…
     1      0  drunk brag trump staffer start russian collus …
     2      0  sheriff david clark becom internet joke threat…
     3      0  trump obsess even obama name code websit imag …
     4      0  pope franci call donald trump christma speech …
```

```
[ ]: df.to_csv('cleanedISOT.csv', index=False)
```

0.3 Playing Around With Pre-trained Word Embedding Vectors

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

```
[ ]: import gensim.downloader as api
     print(list(gensim.downloader.info()['models'].keys())) #available pre-trained␣
      ↪word embedding models provided by the gensim api
```

```
['fasttext-wiki-news-subwords-300', 'conceptnet-numberbatch-17-06-300',
'word2vec-ruscorpora-300', 'word2vec-google-news-300', 'glove-wiki-gigaword-50',
'glove-wiki-gigaword-100', 'glove-wiki-gigaword-200', 'glove-wiki-gigaword-300',
'glove-twitter-25', 'glove-twitter-50', 'glove-twitter-100', 'glove-
twitter-200', '__testing_word2vec-matrix-synopsis']
```

[ ]:
```
from gensim.models import KeyedVectors
wv = KeyedVectors.load('/content/drive/MyDrive/365Project/wv_300_vectors.kv') #␣
 ↪Downloading the 300 dimensional Word2Vec Pre-trained Word Embedding Model
```

[ ]:
```
wv.similarity("beard", "mustache")
```

[ ]:
```
0.8025587
```

[ ]:
```
wv.similarity("beard", "pencil")
```

[ ]:
```
0.20437592
```

[ ]:
```
def cosine_similarity(vec1, vec2):
    dot_product = np.dot(vec1, vec2)
    norm_vec1 = np.linalg.norm(vec1)
    norm_vec2 = np.linalg.norm(vec2)
    similarity = dot_product / (norm_vec1 * norm_vec2)
    return similarity

vec_beard = wv['beard']
vec_mustache = wv['mustache']
vec_pencil = wv['pencil']

similarity_beard_mustache = cosine_similarity(vec_beard, vec_mustache)
similarity_beard_pencil = cosine_similarity(vec_beard, vec_pencil)

print("beard - mustache cosine similarity: " + str(similarity_beard_mustache))
print("beard - pencil cosine similarity: " + str(similarity_beard_pencil))
```

```
beard - mustache cosine similarity: 0.80255866
beard - pencil cosine similarity: 0.20437592
```

[ ]:
```
wv.most_similar(positive=['car', 'minivan'], topn=5)
```

[ ]:
```
[('SUV', 0.8532192707061768),
 ('vehicle', 0.8175783753395081),
 ('pickup_truck', 0.7763688564300537),
 ('Jeep', 0.7567334175109863),
 ('Ford_Explorer', 0.7565720081329346)]
```

[ ]:
```
wv.doesnt_match(['university', 'water', 'diploma', 'school', 'student',␣
 ↪'graduate'])
```

[ ]:
```
'water'
```

```
[ ]: wv.most_similar(positive=['woman', 'king'], negative=['man'], topn=3)
```

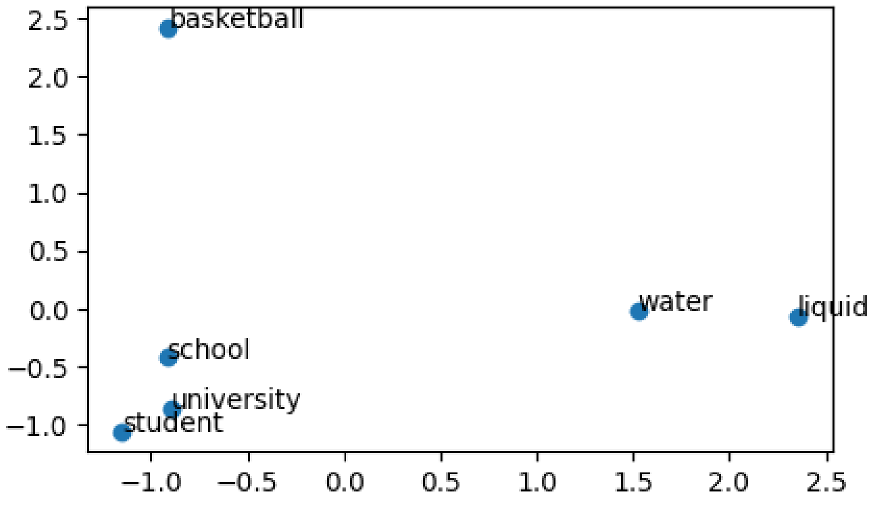
```
[ ]: [('queen', 0.7118193507194519),
      ('monarch', 0.6189674139022827),
      ('princess', 0.5902431011199951)]
```

0.3.1 Visualizing the vectors

```
[ ]: words = ['university', 'water', 'basketball', 'school', 'student', 'liquid']

     sample_vectors = np.array([wv[word] for word in words])
     from sklearn.decomposition import PCA
     pca = PCA(n_components=2)
     result = pca.fit_transform(sample_vectors)
     result
```

```
[ ]: array([[-0.89875144, -0.86489093],
            [ 1.5242077 , -0.0194262 ],
            [-0.91139513,  2.4191253 ],
            [-0.91719323, -0.4183276 ],
            [-1.1542435 , -1.0551665 ],
            [ 2.3573756 , -0.06131356]], dtype=float32)
```

```
[ ]: plt.figure(figsize=(5,3))
     plt.scatter(result[:,0], result[:,1])
     for i, word in enumerate(words):
         plt.annotate(word, xy=(result[i, 0], result[i, 1]))
     plt.show()
```



0.4 Applying Word Embedding on the Dataset for Logistic Regression

[ ]:
```
df = pd.read_csv('/content/drive/MyDrive/365Project/cleanedISOT.csv')
df.dropna(subset=['clean_text'], inplace=True)
```

[ ]:
```
df['tokens'] = df['clean_text'].apply(lambda x: x.split())
```

[ ]:
```
from scipy.sparse import csr_matrix

# sent_to_avg_vec: Takes the vectors for all the words in each text instance␣
 ↪and combines it all in one vector by taking the average
def avg_vec(sent, wv, dim):
    valid_embeddings = [wv[word] for word in sent if word in wv]

    if valid_embeddings:
        avg_embedding = np.mean(valid_embeddings, axis=0)
    else:
        avg_embedding = np.zeros(dim)

    return avg_embedding
```

[ ]:
```
# Here is an example case
dim = wv.vector_size
text = ["turkish", "football", "league"]
avg_vector = avg_vec(text, wv, dim)
avg_vector
```

[ ]:
```
array([-0.05354818, -0.00642904,  0.21223958,  0.06144206,  0.05013021,
       -0.1040802 ,  0.04701741, -0.17467754,  0.09830729,  0.20149739,
       -0.09488932, -0.17797852,  0.00264486,  0.14550781, -0.10994466,
        0.1648763 ,  0.03409831,  0.07230631,  0.1426595 , -0.00927734,
       -0.06005859,  0.08170573,  0.04695638, -0.13823445, -0.04353841,
       -0.01802317, -0.03792318,  0.27897134,  0.18286133,  0.19173177,
        0.04744466,  0.06453451,  0.1372884 , -0.06258138, -0.04817708,
        0.02244059,  0.15625   ,  0.00941976,  0.13232422,  0.18351237,
        0.1200358 , -0.10091146,  0.16194661,  0.16292317,  0.09699503,
       -0.13183594,  0.00716146, -0.14615886, -0.10913086,  0.15641277,
       -0.03441365, -0.02030436, -0.01794434,  0.07495117, -0.11067709,
       -0.2561442 ,  0.07942709, -0.02441406, -0.08854166, -0.0855306 ,
       -0.22916667,  0.0008138 , -0.06502279, -0.01554362,  0.05305989,
       -0.09680176,  0.04663086, -0.08119711,  0.1517334 ,  0.05631511,
        0.12923177,  0.15017192, -0.1763916 , -0.0809733 , -0.04659017,
       -0.04016113,  0.12288412, -0.05940755,  0.0242513 , -0.17317708,
       -0.13028972, -0.11165365,  0.00915527,  0.05745443,  0.1829427 ,
       -0.12556966, -0.18945312,  0.1414388 , -0.0160319 ,  0.03898112,
        0.03340657, -0.1538086 , -0.16194661, -0.1274058 , -0.01974996,
       -0.13444011, -0.00606283,  0.01245117,  0.2298177 , -0.02172852,
       -0.09726969,  0.18115234,  0.13785808, -0.03747559,  0.13704427,
        0.07173666,  0.01660156,  0.04219564, -0.1319987 , -0.09643555,
       -0.08626302,  0.18375652, -0.06982422, -0.11035156,  0.14672852,
        0.12174479,  0.02124023,  0.11989339, -0.05165609, -0.0925293 ,
       -0.12019857, -0.10880534,  0.03727214,  0.07104492, -0.1438802 ,
       -0.25683594, -0.2705078 , -0.04801432, -0.15022786, -0.14697266,
        0.03556315, -0.23274739, -0.07430013, -0.15043132, -0.02954102,
       -0.14242554, -0.19335938,  0.15820312, -0.08902995, -0.00425212,
       -0.02164713, -0.0978597 ,  0.1126709 ,  0.07942709,  0.04962158,
        0.13053386, -0.01253255, -0.01660156, -0.08382162, -0.28841147,
        0.09559059,  0.10017904, -0.33984375, -0.07745361, -0.11968485,
       -0.02372233,  0.03238932, -0.14420573,  0.12613933, -0.02442423,
       -0.01757812,  0.1410319 ,  0.05688477, -0.10599772,  0.01513672,
       -0.15413411,  0.07584635, -0.05110677, -0.07280477,  0.12125651,
       -0.17215984, -0.01688639,  0.14325969, -0.18180339,  0.05143229,
        0.01453654,  0.09427897, -0.06298828,  0.02115885,  0.00586273,
        0.16764323,  0.01692708, -0.03063965,  0.04618327, -0.01472982,
        0.25553384, -0.02986654, -0.02050781,  0.14253743,  0.01234436,
        0.10339355,  0.00728353, -0.01163737,  0.10520426,  0.03238932,
        0.08292643,  0.00528971,  0.15006511, -0.10465495, -0.07381185,
       -0.13313802, -0.03971354,  0.04239909, -0.05615234,  0.09895834,
       -0.27115884,  0.21386719, -0.16227214, -0.0555013 ,  0.04069011,
       -0.0137736 ,  0.08390299,  0.00429281,  0.05924479, -0.30045572,
        0.050354  ,  0.08805338,  0.00292969, -0.22688802, -0.01774088,
       -0.07513428, -0.21077473, -0.09815725,  0.06518555,  0.08056641,
        0.03697713,  0.15201823,  0.04056804,  0.07660929, -0.12158203,
        0.08805338, -0.14648438, -0.00684611,  0.11808268, -0.08317057,
        0.05192057,  0.04036458, -0.12239584,  0.05822754, -0.00911458,
        0.08072916, -0.06038411,  0.17490642, -0.08947754, -0.1167806 ,
        0.10953776, -0.08846029,  0.03194173, -0.2548828 , -0.07047526,
        0.06599935, -0.02864583,  0.18776448,  0.10099284,  0.02604167,
       -0.09859212, -0.10229492,  0.00265503,  0.01041667,  0.11073812,
        0.13274638,  0.04150391,  0.0874939 ,  0.04394531, -0.1155599 ,
        0.12239584, -0.09098307,  0.03873698, -0.13183594, -0.02284749,
       -0.02367655,  0.05777995, -0.17336655, -0.01595052,  0.12894695,
       -0.03973389,  0.07946777, -0.17138672, -0.10042318, -0.08561198,
       -0.16585286,  0.08138021,  0.0686849 , -0.03641891, -0.09330241,
       -0.07051595, -0.10595703,  0.14477539,  0.03450521,  0.02708944,
       -0.12070211,  0.01220703, -0.13167317, -0.08138021,  0.08935547,
        0.08284505, -0.07413737, -0.01015218, -0.12320963,  0.12402344],
      dtype=float32)
```

[ ]:
```
df['avg_vector'] = df['tokens'].apply(lambda x: avg_vec(x, wv, wv.vector_size))

vectors_df = pd.DataFrame(df['avg_vector'].tolist())
result_df = pd.concat([df, vectors_df], axis=1)
result_df = result_df.drop(['avg_vector'], axis=1)
```

```
columns_to_check = list(range(0, 300)) + ['class']
df = result_df.dropna(subset=columns_to_check)

df.head()
```

```
<ipython-input-20-5b98cde2e316>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['avg_vector'] = df['tokens'].apply(lambda x: avg_vec(x, wv,
wv.vector_size))
```

```
[ ]:    class                                        clean_text  \
     0    0.0  donald trump send embarrass new year eve messa…
     1    0.0  drunk brag trump staffer start russian collus …
     2    0.0  sheriff david clark becom internet joke threat…
     3    0.0  trump obsess even obama name code websit imag …
     4    0.0  pope franci call donald trump christma speech …

                                                  tokens         0         1  \
     0  [donald, trump, send, embarrass, new, year, ev…  0.008690  0.052507
     1  [drunk, brag, trump, staffer, start, russian, … -0.027989  0.026530
     2  [sheriff, david, clark, becom, internet, joke,…  0.003965  0.022524
     3  [trump, obsess, even, obama, name, code, websi…  0.004360  0.013982
     4  [pope, franci, call, donald, trump, christma, …  0.026814  0.045902

               2         3         4         5         6  …       290       291  \
     0  0.028267  0.102293 -0.074369  0.009419  0.028947  …  0.059832  0.068490
     1  0.016263  0.086592 -0.066898  0.005858  0.021110  …  0.041051  0.045733
     2  0.009441  0.068881 -0.057199  0.015096  0.012128  …  0.031391  0.053071
     3  0.036881  0.103634 -0.078380 -0.006526  0.041167  …  0.056276  0.049329
     4  0.031234  0.106746 -0.069552  0.001861  0.060818  … -0.015138  0.040970

             292       293       294       295       296       297       298  \
     0 -0.080953 -0.004112 -0.045639 -0.103846 -0.008104 -0.058081 -0.004457
     1 -0.038417  0.032777 -0.042354 -0.068702 -0.014460 -0.076555  0.002295
     2 -0.072740  0.052704 -0.060610 -0.105792 -0.011891 -0.083825 -0.018013
     3 -0.034890  0.030771 -0.008382 -0.081602  0.005466 -0.107022 -0.028669
     4 -0.092981 -0.012650 -0.034230 -0.018661 -0.033963 -0.042701  0.041217

             299
     0  0.018617
     1  0.029151
     2  0.038989
```

```
3   0.006137
4   0.024768

[5 rows x 603 columns]
```

0.5 PCA

[ ]:
```
def standardize_data(X):
    return (X - np.mean(X, axis=0)) / np.std(X, axis=0)
```

[ ]:
```
def compute_covariance_matrix(X):
    return np.cov(X, rowvar=False)
```

[ ]:
```
def pca(X, num_components):

    X_std = standardize_data(X)

    covariance_matrix = compute_covariance_matrix(X_std)

    eigenvalues, eigenvectors = np.linalg.eig(covariance_matrix)

    sorted_indices = np.argsort(eigenvalues)[::-1]
    sorted_eigenvalues = eigenvalues[sorted_indices]
    sorted_eigenvectors = eigenvectors[:, sorted_indices]

    selected_eigenvectors = sorted_eigenvectors[:, :num_components]

    X_pca = np.dot(X_std, selected_eigenvectors)

    return X_pca
```

[ ]:
```
feature_columns = [i for i in range(300)]
X = df[feature_columns].values

df_reduced = pca(X, num_components=2)
y_reduced = df['class'].values

df_reduced
```

[ ]:
```
array([[-6.9686691 , -3.35490097],
       [-2.457901  ,  2.47245576],
       [-8.32506832,  2.36309846],
       …,
       [ 6.70632216, -6.21416062],
       [ 2.28497819,  4.70727509],
       [ 6.4799931 ,  4.66618335]])
```

0.6 Logistic Regression

```
[ ]: def compute_gradient(X, y, b):
         intercept = np.ones((X.shape[0], 1))
         X_b = np.hstack((intercept, X))
         predictions = 1 / (1 + np.exp(-np.dot(X_b, b)))
         errors = y - predictions
         gradient = -np.dot(X_b.T, errors)
         return gradient
```

```
[ ]: def gradient_descent(X, y, initial_b, step_size, max_iteration):
         b = initial_b
         for iteration in range(max_iteration):
             gradient = compute_gradient(X, y, b)
             b -= step_size * gradient
         return b
```

```
[ ]: from sklearn.model_selection import train_test_split

     def logistic_regression(X, y, step_sizes):
         X = np.asarray(X, dtype=np.float64)

         accuracies = []
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
      ↪random_state=42)
         max_iterations = 1000

         for step_size in step_sizes:
             initial_b = np.zeros(X_train.shape[1] + 1)
             optimized_b = gradient_descent(X_train, y_train, initial_b, step_size,␣
      ↪max_iterations)

             intercept = np.ones((X_test.shape[0], 1))
             X_test_with_intercept = np.hstack((intercept, X_test))
             predicted_probabilities = 1 / (1 + np.exp(-np.
      ↪dot(X_test_with_intercept, optimized_b)))
             predicted_labels = (predicted_probabilities > 0.5).astype(int)
             accuracy = calculate_accuracy(y_test, predicted_labels)
             accuracies.append((step_size, accuracy))

         return accuracies
```

```
[ ]: def calculate_accuracy(y_true, y_pred):
         correct_predictions = np.sum(y_true == y_pred)
         total_predictions = len(y_true)
         return correct_predictions / total_predictions
```

```
[ ]: step_sizes = [0.001, 0.01, 0.03, 0.05, 0.07]

     # Original DataFrame
     X = df[[i for i in range(300)]].values
     y = df['class'].values
     results_df = logistic_regression(X, y, step_sizes)
```

```
<ipython-input-41-4242b51176dc>:4: RuntimeWarning: overflow encountered in exp
  predictions = 1 / (1 + np.exp(-np.dot(X_b, b)))
<ipython-input-43-f4fd32a04c7d>:16: RuntimeWarning: overflow encountered in exp
  predicted_probabilities = 1 / (1 + np.exp(-np.dot(X_test_with_intercept,
optimized_b)))
```

```
[ ]: # PCA-reduced DataFrame
     results_df_reduced = logistic_regression(df_reduced, y_reduced, step_sizes)
```

```
<ipython-input-41-4242b51176dc>:4: RuntimeWarning: overflow encountered in exp
  predictions = 1 / (1 + np.exp(-np.dot(X_b, b)))
<ipython-input-43-f4fd32a04c7d>:16: RuntimeWarning: overflow encountered in exp
  predicted_probabilities = 1 / (1 + np.exp(-np.dot(X_test_with_intercept,
optimized_b)))
```

```
[ ]: print("Results for Original DataFrame:")
     for step_size, accuracy in results_df:
         print(f"Step Size: {step_size}, Accuracy: {accuracy}")

     print("\nResults for PCA Reduced DataFrame:")
     for step_size, accuracy in results_df_reduced:
         print(f"Step Size: {step_size}, Accuracy: {accuracy}")
```

```
Results for Original DataFrame:
Step Size: 0.001, Accuracy: 0.9381408827463219
Step Size: 0.01, Accuracy: 0.9344627730717788
Step Size: 0.03, Accuracy: 0.9379179670084707
Step Size: 0.05, Accuracy: 0.9381408827463219
Step Size: 0.07, Accuracy: 0.9324565314311191

Results for PCA Reduced DataFrame:
Step Size: 0.001, Accuracy: 0.8041685242978154
Step Size: 0.01, Accuracy: 0.8264600980829246
Step Size: 0.03, Accuracy: 0.8293580026749888
Step Size: 0.05, Accuracy: 0.8242309407044137
Step Size: 0.07, Accuracy: 0.832255907267053
```

```
[ ]: from sklearn.linear_model import LogisticRegression
     from sklearn import metrics

     X = df[[i for i in range(300)]]
```

```
Y = df['class']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)

classifier = LogisticRegression()
classifier.fit(X_train,Y_train)

predicted = classifier.predict(X_test)
print("sklearn Logistic Regression on Original Dataframe Accuracy :",metrics.
 ↪accuracy_score(Y_test, predicted))
```

```
sklearn Logistic Regression on Original Dataframe Accuracy : 0.9400356665180561
```

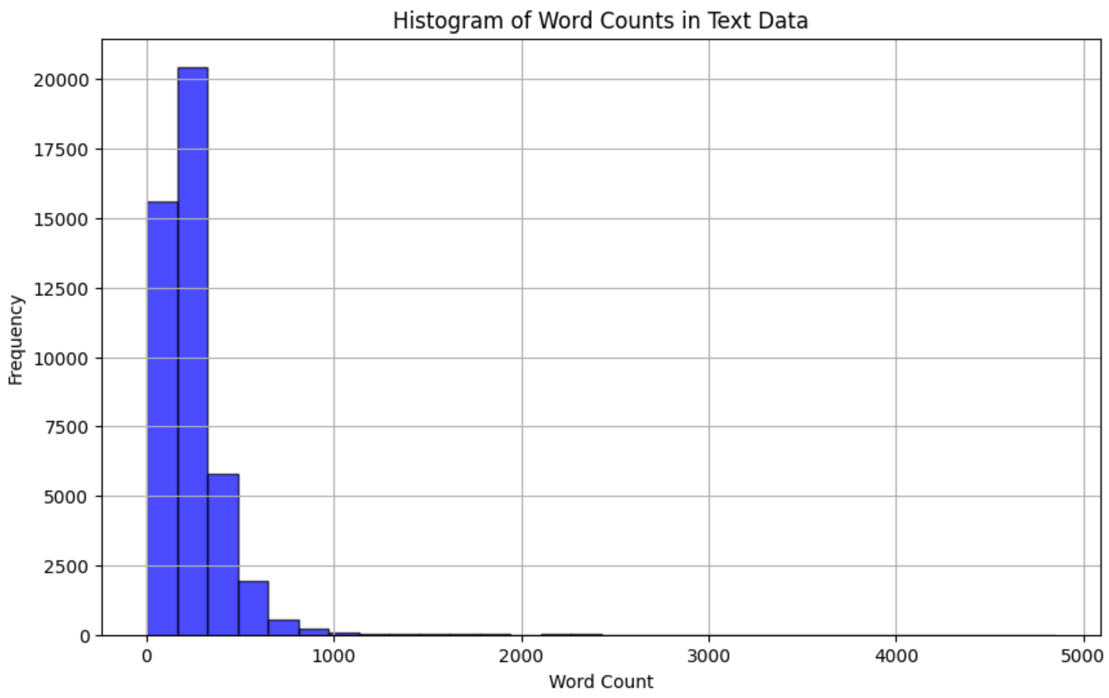
0.7 Data Analysis for CNN Modeling

[ ]:
```
df = pd.read_csv('/content/drive/MyDrive/365Project/cleanedISOT.csv')
df.dropna(subset=['clean_text'], inplace=True)
```

[ ]:
```
df['word_count'] = df['clean_text'].apply(lambda text: len(text.split()))

word_count_stats = df['word_count'].describe()
word_count_stats
```

[ ]:
```
count    44869.000000
mean       235.299093
std        198.915992
min          3.000000
25%        124.000000
50%        208.000000
75%        292.000000
max       4849.000000
Name: word_count, dtype: float64
```

[ ]:
```
plt.figure(figsize=(10, 6))
plt.hist(df['word_count'], bins=30, color='blue', edgecolor='black', alpha=0.7)
plt.title('Histogram of Word Counts in Text Data')
plt.xlabel('Word Count')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



```
[ ]: max_len_list = [df['word_count'].quantile(0.85), df['word_count'].quantile(0.
      ↪9), df['word_count'].quantile(0.95)]
     max_len_list
```

```
[ ]: [369.0, 430.0, 527.0]
```

```
[ ]: texts = df['clean_text'].tolist()

     tokenizer = Tokenizer()
     tokenizer.fit_on_texts(texts)

     num_unique_words = len(tokenizer.word_index)
     num_unique_words
```

```
[ ]: 168818
```

0.8 Introduction to CNN and its Parameters

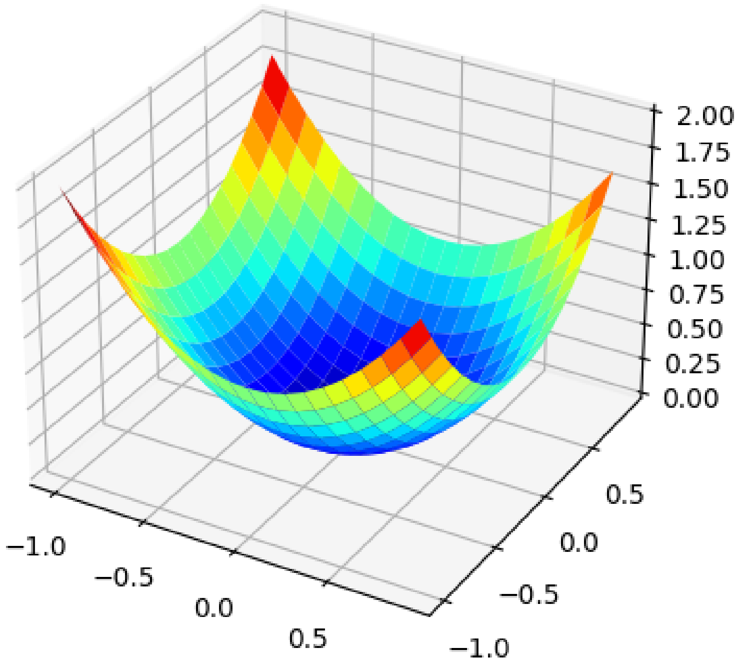
0.8.1 Adam Optimizer Algorithm

Gradient descent with learning rates (step sizes)

```
[ ]: def objective(x, y):
         return x**2 + y**2

     def derivatives(x, y):
```

```
    dx = 2 * x
    dy = 2 * y
    return np.array([dx, dy])
```

[ ]:
```
from numpy import arange
from numpy import meshgrid

r_min, r_max = -1.0, 1.0
xaxis = arange(r_min, r_max, 0.1)
yaxis = arange(r_min, r_max, 0.1)

x, y = meshgrid(xaxis, yaxis)

results = objective(x, y)

figure = plt.figure()
axis = figure.add_subplot(111, projection='3d')  # Change here
axis.plot_surface(x, y, results, cmap='jet')

plt.show()
```

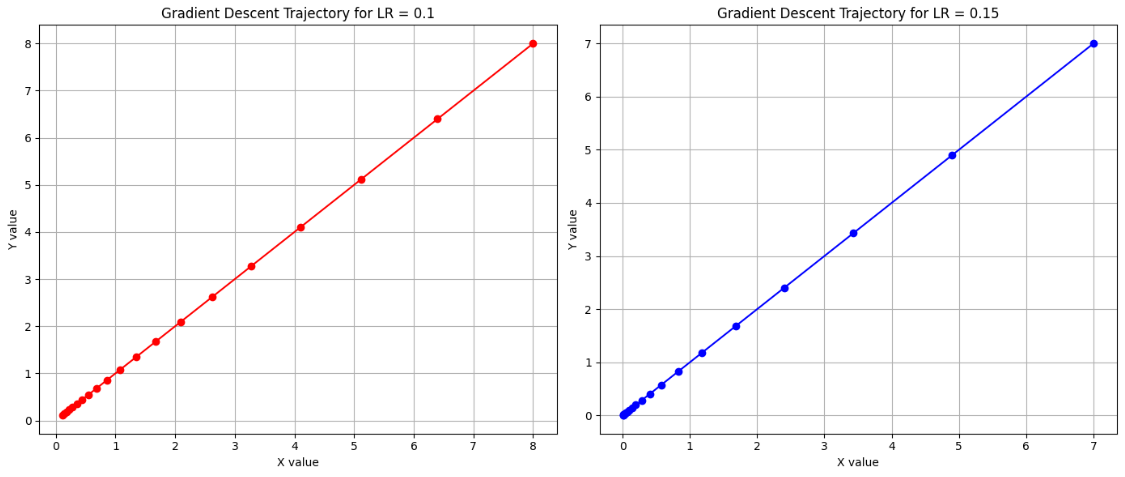


[ ]:
```
def gradient_descent(start_point, learning_rate, n_iter):
    x, y = start_point
    trajectory = np.zeros((n_iter, 2))

    for i in range(n_iter):
        grad = derivatives(x, y)
        x -= learning_rate * grad[0]
        y -= learning_rate * grad[1]
        trajectory[i] = [x, y]
        print(f"Iteration {i}: Point=({x:.4f}, {y:.4f}),␣
↪Objective={objective(x, y):.4f}")

    return x, y, trajectory
```

```
[ ]: start_point = (10, 10)  # Starting at (x=10, y=10)
     n_iter = 20  # Number of iterations

     # Experiment with learning rate 0.1
     print("Experiment with learning rate 0.1")
     final_x, final_y, trajectory_01 = gradient_descent(start_point, 0.1, n_iter)

     # Experiment with learning rate 0.15
     print("\nExperiment with learning rate 0.15")
     final_x, final_y, trajectory_015 = gradient_descent(start_point, 0.15, n_iter)
```

```
Experiment with learning rate 0.1
Iteration 0: Point=(8.0000, 8.0000), Objective=128.0000
Iteration 1: Point=(6.4000, 6.4000), Objective=81.9200
Iteration 2: Point=(5.1200, 5.1200), Objective=52.4288
Iteration 3: Point=(4.0960, 4.0960), Objective=33.5544
Iteration 4: Point=(3.2768, 3.2768), Objective=21.4748
Iteration 5: Point=(2.6214, 2.6214), Objective=13.7439
Iteration 6: Point=(2.0972, 2.0972), Objective=8.7961
Iteration 7: Point=(1.6777, 1.6777), Objective=5.6295
Iteration 8: Point=(1.3422, 1.3422), Objective=3.6029
Iteration 9: Point=(1.0737, 1.0737), Objective=2.3058
Iteration 10: Point=(0.8590, 0.8590), Objective=1.4757
Iteration 11: Point=(0.6872, 0.6872), Objective=0.9445
Iteration 12: Point=(0.5498, 0.5498), Objective=0.6045
Iteration 13: Point=(0.4398, 0.4398), Objective=0.3869
Iteration 14: Point=(0.3518, 0.3518), Objective=0.2476
Iteration 15: Point=(0.2815, 0.2815), Objective=0.1585
Iteration 16: Point=(0.2252, 0.2252), Objective=0.1014
Iteration 17: Point=(0.1801, 0.1801), Objective=0.0649
Iteration 18: Point=(0.1441, 0.1441), Objective=0.0415
Iteration 19: Point=(0.1153, 0.1153), Objective=0.0266

Experiment with learning rate 0.15
Iteration 0: Point=(7.0000, 7.0000), Objective=98.0000
Iteration 1: Point=(4.9000, 4.9000), Objective=48.0200
Iteration 2: Point=(3.4300, 3.4300), Objective=23.5298
Iteration 3: Point=(2.4010, 2.4010), Objective=11.5296
Iteration 4: Point=(1.6807, 1.6807), Objective=5.6495
Iteration 5: Point=(1.1765, 1.1765), Objective=2.7683
Iteration 6: Point=(0.8235, 0.8235), Objective=1.3564
Iteration 7: Point=(0.5765, 0.5765), Objective=0.6647
Iteration 8: Point=(0.4035, 0.4035), Objective=0.3257
Iteration 9: Point=(0.2825, 0.2825), Objective=0.1596
Iteration 10: Point=(0.1977, 0.1977), Objective=0.0782
Iteration 11: Point=(0.1384, 0.1384), Objective=0.0383
Iteration 12: Point=(0.0969, 0.0969), Objective=0.0188
Iteration 13: Point=(0.0678, 0.0678), Objective=0.0092
Iteration 14: Point=(0.0475, 0.0475), Objective=0.0045
Iteration 15: Point=(0.0332, 0.0332), Objective=0.0022
Iteration 16: Point=(0.0233, 0.0233), Objective=0.0011
Iteration 17: Point=(0.0163, 0.0163), Objective=0.0005
Iteration 18: Point=(0.0114, 0.0114), Objective=0.0003
Iteration 19: Point=(0.0080, 0.0080), Objective=0.0001
```

[ ]:
```python
fig, axes = plt.subplots(1, 2, figsize=(14, 6))

# plot for learning rate = 0.1
axes[0].plot(trajectory_01[:, 0], trajectory_01[:, 1], 'ro-')
axes[0].set_title('Gradient Descent Trajectory for LR = 0.1')
axes[0].set_xlabel('X value')
axes[0].set_ylabel('Y value')
axes[0].grid(True)

# plot for learning rate = 0.15
axes[1].plot(trajectory_015[:, 0], trajectory_015[:, 1], 'bo-')
axes[1].set_title('Gradient Descent Trajectory for LR = 0.15')
axes[1].set_xlabel('X value')
axes[1].set_ylabel('Y value')
axes[1].grid(True)

# display side by side
plt.tight_layout()
plt.show()
```



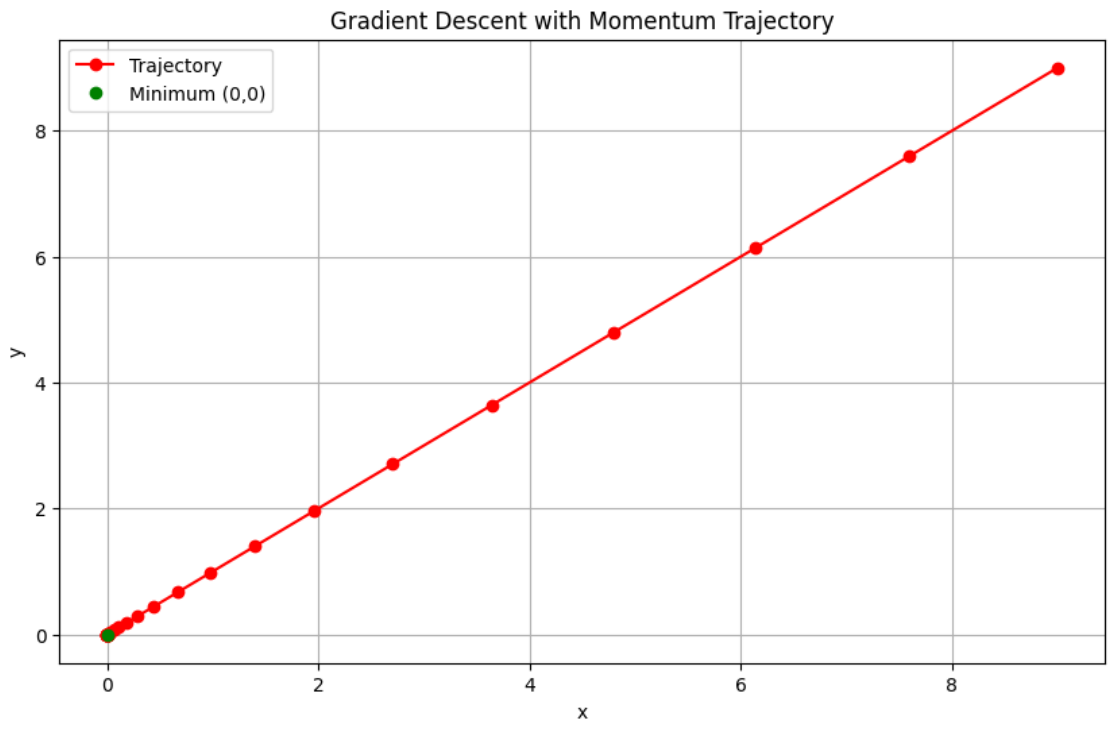
Gradient descent with momentum

```
[ ]: def gradient_descent_with_momentum(start_point, learning_rate, beta, n_iter):
         x, y = start_point
         v_x = v_y = 0  # Initialize velocity components
         trajectory = np.zeros((n_iter, 2))

         for i in range(n_iter):
             grad = derivatives(x, y)
             v_x = beta * v_x + (1 - beta) * grad[0]
             v_y = beta * v_y + (1 - beta) * grad[1]
             x -= learning_rate * v_x
             y -= learning_rate * v_y
             trajectory[i] = [x, y]
             print(f"Iteration {i}: Point=({x:.4f}, {y:.4f}),␣
      ↪Objective={objective(x, y):.4f}")

         return x, y, trajectory
```

```
[ ]: start_point = (10.0, 10.0)
     learning_rate = 0.1
     beta = 0.5
     n_iter = 50

     final_x, final_y, trajectory = gradient_descent_with_momentum(start_point,␣
      ↪learning_rate, beta, n_iter)
```

```
Iteration 0: Point=(9.0000, 9.0000), Objective=162.0000
Iteration 1: Point=(7.6000, 7.6000), Objective=115.5200
Iteration 2: Point=(6.1400, 6.1400), Objective=75.3992
Iteration 3: Point=(4.7960, 4.7960), Objective=46.0032
Iteration 4: Point=(3.6444, 3.6444), Objective=26.5633
Iteration 5: Point=(2.7042, 2.7042), Objective=14.6250
Iteration 6: Point=(1.9636, 1.9636), Objective=7.7116
Iteration 7: Point=(1.3970, 1.3970), Objective=3.9032
Iteration 8: Point=(0.9740, 0.9740), Objective=1.8973
Iteration 9: Point=(0.6651, 0.6651), Objective=0.8846
Iteration 10: Point=(0.4441, 0.4441), Objective=0.3945
Iteration 11: Point=(0.2892, 0.2892), Objective=0.1673
Iteration 12: Point=(0.1829, 0.1829), Objective=0.0669
Iteration 13: Point=(0.1114, 0.1114), Objective=0.0248
Iteration 14: Point=(0.0645, 0.0645), Objective=0.0083
Iteration 15: Point=(0.0346, 0.0346), Objective=0.0024
Iteration 16: Point=(0.0162, 0.0162), Objective=0.0005
Iteration 17: Point=(0.0054, 0.0054), Objective=0.0001
Iteration 18: Point=(-0.0006, -0.0006), Objective=0.0000
Iteration 19: Point=(-0.0035, -0.0035), Objective=0.0000
Iteration 20: Point=(-0.0046, -0.0046), Objective=0.0000
Iteration 21: Point=(-0.0047, -0.0047), Objective=0.0000
Iteration 22: Point=(-0.0043, -0.0043), Objective=0.0000
Iteration 23: Point=(-0.0036, -0.0036), Objective=0.0000
Iteration 24: Point=(-0.0029, -0.0029), Objective=0.0000
Iteration 25: Point=(-0.0023, -0.0023), Objective=0.0000
Iteration 26: Point=(-0.0018, -0.0018), Objective=0.0000
Iteration 27: Point=(-0.0013, -0.0013), Objective=0.0000
Iteration 28: Point=(-0.0010, -0.0010), Objective=0.0000
Iteration 29: Point=(-0.0007, -0.0007), Objective=0.0000
Iteration 30: Point=(-0.0005, -0.0005), Objective=0.0000
Iteration 31: Point=(-0.0003, -0.0003), Objective=0.0000
Iteration 32: Point=(-0.0002, -0.0002), Objective=0.0000
Iteration 33: Point=(-0.0001, -0.0001), Objective=0.0000
Iteration 34: Point=(-0.0001, -0.0001), Objective=0.0000
Iteration 35: Point=(-0.0001, -0.0001), Objective=0.0000
Iteration 36: Point=(-0.0000, -0.0000), Objective=0.0000
Iteration 37: Point=(-0.0000, -0.0000), Objective=0.0000
Iteration 38: Point=(-0.0000, -0.0000), Objective=0.0000
Iteration 39: Point=(-0.0000, -0.0000), Objective=0.0000
Iteration 40: Point=(-0.0000, -0.0000), Objective=0.0000
Iteration 41: Point=(0.0000, 0.0000), Objective=0.0000
Iteration 42: Point=(0.0000, 0.0000), Objective=0.0000
Iteration 43: Point=(0.0000, 0.0000), Objective=0.0000
Iteration 44: Point=(0.0000, 0.0000), Objective=0.0000
Iteration 45: Point=(0.0000, 0.0000), Objective=0.0000
Iteration 46: Point=(0.0000, 0.0000), Objective=0.0000
Iteration 47: Point=(0.0000, 0.0000), Objective=0.0000
Iteration 48: Point=(0.0000, 0.0000), Objective=0.0000
Iteration 49: Point=(0.0000, 0.0000), Objective=0.0000
```

[ ]:
```
x_coords, y_coords = trajectory[:, 0], trajectory[:, 1]

plt.figure(figsize=(10, 6))
plt.plot(x_coords, y_coords, 'ro-', label='Trajectory')
plt.plot(0, 0, 'go', label='Minimum (0,0)')
plt.title('Gradient Descent with Momentum Trajectory')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.legend()
plt.show()
```



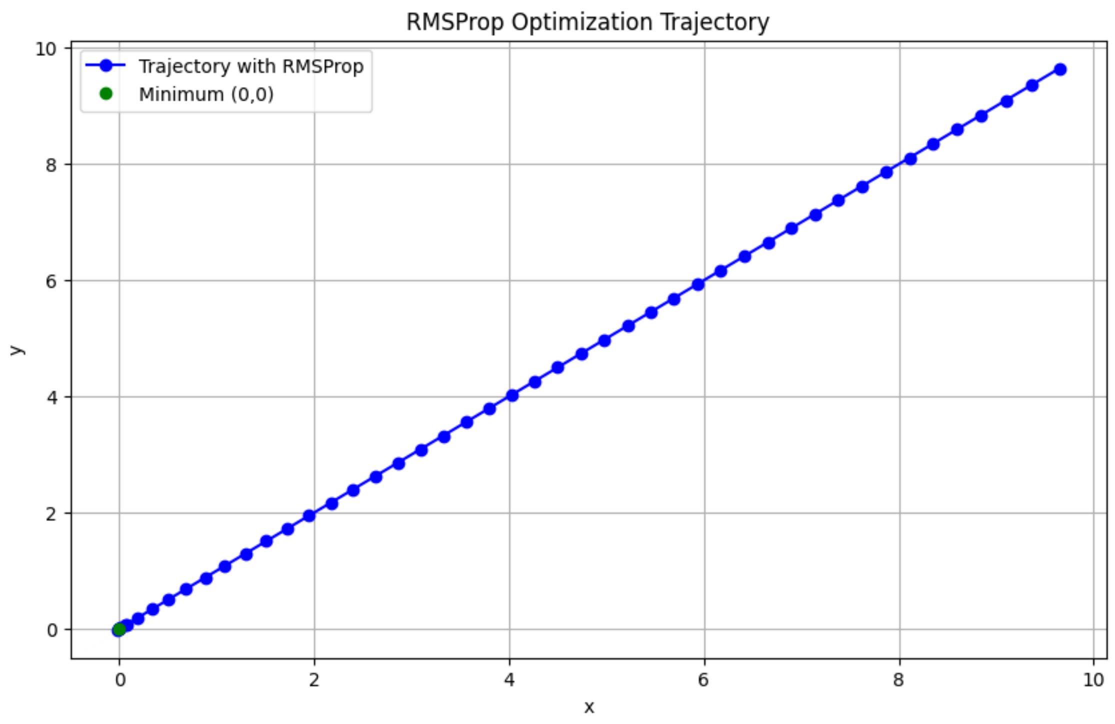
Root Mean Squared Propagation

[ ]:
```
def rmsprop(start_point, learning_rate, beta, epsilon, n_iter):
    x, y = start_point
    s_x = s_y = 0  # Initialize RMSProp accumulators
    trajectory = np.zeros((n_iter, 2))

    for i in range(n_iter):
        grad = derivatives(x, y)
        s_x = beta * s_x + (1 - beta) * grad[0]**2
        s_y = beta * s_y + (1 - beta) * grad[1]**2
        x -= (learning_rate / (np.sqrt(s_x) + epsilon)) * grad[0]
        y -= (learning_rate / (np.sqrt(s_y) + epsilon)) * grad[1]
        trajectory[i] = [x, y]
        print(f"Iteration {i}: Point=({x:.4f}, {y:.4f}),␣
↪Objective={objective(x, y):.4f}")

    return x, y, trajectory
```

[ ]:
```
start_point = (10.0, 10.0)
learning_rate = 0.25
beta = 0.5
epsilon = 1e-8
n_iter = 50

final_x, final_y, trajectory = rmsprop(start_point, learning_rate, beta,␣
↪epsilon, n_iter)
```

```
Iteration 0: Point=(9.6464, 9.6464), Objective=186.1079
Iteration 1: Point=(9.3613, 9.3613), Objective=175.2678
Iteration 2: Point=(9.0990, 9.0990), Objective=165.5819
Iteration 3: Point=(8.8465, 8.8465), Objective=156.5209
Iteration 4: Point=(8.5988, 8.5988), Objective=147.8786
Iteration 5: Point=(8.3535, 8.3535), Objective=139.5635
Iteration 6: Point=(8.1096, 8.1096), Objective=131.5322
Iteration 7: Point=(7.8665, 7.8665), Objective=123.7637
Iteration 8: Point=(7.6239, 7.6239), Objective=116.2475
Iteration 9: Point=(7.3817, 7.3817), Objective=108.9781
Iteration 10: Point=(7.1398, 7.1398), Objective=101.9528
Iteration 11: Point=(6.8982, 6.8982), Objective=95.1701
Iteration 12: Point=(6.6569, 6.6569), Objective=88.6290
Iteration 13: Point=(6.4160, 6.4160), Objective=82.3289
Iteration 14: Point=(6.1753, 6.1753), Objective=76.2692
Iteration 15: Point=(5.9350, 5.9350), Objective=70.4495
Iteration 16: Point=(5.6951, 5.6951), Objective=64.8694
Iteration 17: Point=(5.4557, 5.4557), Objective=59.5285
Iteration 18: Point=(5.2166, 5.2166), Objective=54.4261
Iteration 19: Point=(4.9780, 4.9780), Objective=49.5619
Iteration 20: Point=(4.7400, 4.7400), Objective=44.9352
Iteration 21: Point=(4.5025, 4.5025), Objective=40.5454
Iteration 22: Point=(4.2657, 4.2657), Objective=36.3920
Iteration 23: Point=(4.0295, 4.0295), Objective=32.4740
Iteration 24: Point=(3.7941, 3.7941), Objective=28.7907
Iteration 25: Point=(3.5596, 3.5596), Objective=25.3411
Iteration 26: Point=(3.3260, 3.3260), Objective=22.1242
Iteration 27: Point=(3.0935, 3.0935), Objective=19.1389
Iteration 28: Point=(2.8621, 2.8621), Objective=16.3837
Iteration 29: Point=(2.6322, 2.6322), Objective=13.8572
Iteration 30: Point=(2.4039, 2.4039), Objective=11.5575
Iteration 31: Point=(2.1774, 2.1774), Objective=9.4825
Iteration 32: Point=(1.9532, 1.9532), Objective=7.6298
Iteration 33: Point=(1.7315, 1.7315), Objective=5.9965
Iteration 34: Point=(1.5131, 1.5131), Objective=4.5787
Iteration 35: Point=(1.2985, 1.2985), Objective=3.3721
Iteration 36: Point=(1.0888, 1.0888), Objective=2.3710
Iteration 37: Point=(0.8854, 0.8854), Objective=1.5679
Iteration 38: Point=(0.6903, 0.6903), Objective=0.9531
Iteration 39: Point=(0.5066, 0.5066), Objective=0.5132
Iteration 40: Point=(0.3387, 0.3387), Objective=0.2295
Iteration 41: Point=(0.1939, 0.1939), Objective=0.0752
Iteration 42: Point=(0.0827, 0.0827), Objective=0.0137
Iteration 43: Point=(0.0168, 0.0168), Objective=0.0006
Iteration 44: Point=(-0.0021, -0.0021), Objective=0.0000
Iteration 45: Point=(0.0012, 0.0012), Objective=0.0000
Iteration 46: Point=(-0.0016, -0.0016), Objective=0.0000
Iteration 47: Point=(0.0034, 0.0034), Objective=0.0000
Iteration 48: Point=(-0.0119, -0.0119), Objective=0.0003
Iteration 49: Point=(0.0624, 0.0624), Objective=0.0078
```

[ ]:
```
x_coords, y_coords = trajectory[:, 0], trajectory[:, 1]

plt.figure(figsize=(10, 6))
plt.plot(x_coords, y_coords, 'bo-', label='Trajectory with RMSProp')
plt.plot(0, 0, 'go', label='Minimum (0,0)')
plt.title('RMSProp Optimization Trajectory')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.legend()
plt.show()
```



Root Mean Squared Propagation with Momentum

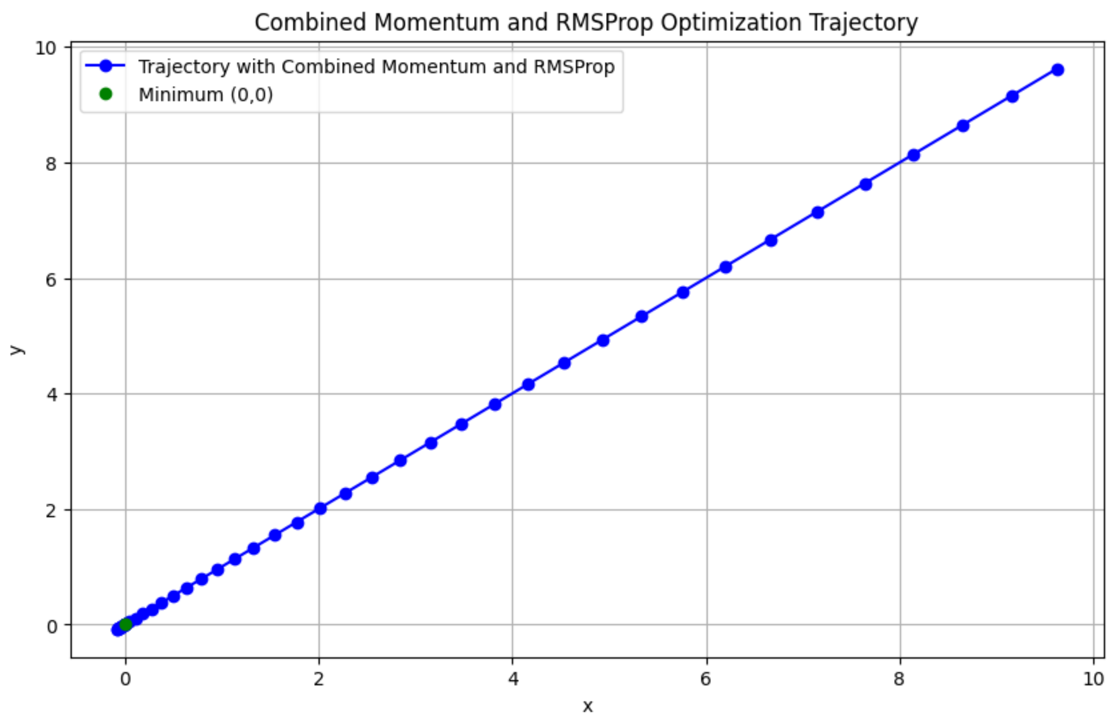
```
[ ]: def momentum_rmsprop(start_point, learning_rate, beta1, beta2, epsilon, n_iter):
         x, y = start_point
         v_x = v_y = 0  # momentum
         s_x = s_y = 0  # RMSProp
         trajectory = np.zeros((n_iter, 2))

         for i in range(n_iter):
             grad = derivatives(x, y)
             v_x = beta1 * v_x + (1 - beta1) * grad[0]
             v_y = beta1 * v_y + (1 - beta1) * grad[1]
             s_x = beta2 * s_x + (1 - beta2) * grad[0]**2
             s_y = beta2 * s_y + (1 - beta2) * grad[1]**2
             x -= (learning_rate / (np.sqrt(s_x) + epsilon)) * v_x
             y -= (learning_rate / (np.sqrt(s_y) + epsilon)) * v_y
             trajectory[i] = [x, y]
             print(f"Iteration {i}: Point=({x:.4f}, {y:.4f}),␣
      ↪Objective={objective(x, y):.4f}")

         return x, y, trajectory
```

```
[ ]: start_point = (10.0, 10.0)
     learning_rate = 0.4
```

```
beta1 = 0.7
beta2 = 0.9
epsilon = 1e-8
n_iter = 50

final_x, final_y, trajectory = momentum_rmsprop(start_point, learning_rate,␣
 ↪beta1, beta2, epsilon, n_iter)
```

```
Iteration 0: Point=(9.6205, 9.6205), Objective=185.1091
Iteration 1: Point=(9.1537, 9.1537), Objective=167.5815
Iteration 2: Point=(8.6529, 8.6529), Objective=149.7452
Iteration 3: Point=(8.1429, 8.1429), Objective=132.6149
Iteration 4: Point=(7.6371, 7.6371), Objective=116.6499
Iteration 5: Point=(7.1426, 7.1426), Objective=102.0332
Iteration 6: Point=(6.6634, 6.6634), Objective=88.8028
Iteration 7: Point=(6.2016, 6.2016), Objective=76.9206
Iteration 8: Point=(5.7581, 5.7581), Objective=66.3109
Iteration 9: Point=(5.3330, 5.3330), Objective=56.8807
Iteration 10: Point=(4.9261, 4.9261), Objective=48.5327
Iteration 11: Point=(4.5371, 4.5371), Objective=41.1709
Iteration 12: Point=(4.1656, 4.1656), Objective=34.7041
Iteration 13: Point=(3.8110, 3.8110), Objective=29.0474
Iteration 14: Point=(3.4729, 3.4729), Objective=24.1226
Iteration 15: Point=(3.1510, 3.1510), Objective=19.8578
Iteration 16: Point=(2.8449, 2.8449), Objective=16.1873
Iteration 17: Point=(2.5544, 2.5544), Objective=13.0504
Iteration 18: Point=(2.2794, 2.2794), Objective=10.3913
Iteration 19: Point=(2.0197, 2.0197), Objective=8.1584
Iteration 20: Point=(1.7754, 1.7754), Objective=6.3037
Iteration 21: Point=(1.5464, 1.5464), Objective=4.7825
Iteration 22: Point=(1.3328, 1.3328), Objective=3.5528
Iteration 23: Point=(1.1348, 1.1348), Objective=2.5756
Iteration 24: Point=(0.9524, 0.9524), Objective=1.8143
Iteration 25: Point=(0.7858, 0.7858), Objective=1.2349
Iteration 26: Point=(0.6349, 0.6349), Objective=0.8062
Iteration 27: Point=(0.4998, 0.4998), Objective=0.4996
Iteration 28: Point=(0.3803, 0.3803), Objective=0.2893
Iteration 29: Point=(0.2763, 0.2763), Objective=0.1527
Iteration 30: Point=(0.1874, 0.1874), Objective=0.0702
Iteration 31: Point=(0.1131, 0.1131), Objective=0.0256
Iteration 32: Point=(0.0527, 0.0527), Objective=0.0056
Iteration 33: Point=(0.0054, 0.0054), Objective=0.0001
Iteration 34: Point=(-0.0298, -0.0298), Objective=0.0018
Iteration 35: Point=(-0.0540, -0.0540), Objective=0.0058
Iteration 36: Point=(-0.0686, -0.0686), Objective=0.0094
Iteration 37: Point=(-0.0750, -0.0750), Objective=0.0113
Iteration 38: Point=(-0.0747, -0.0747), Objective=0.0112
Iteration 39: Point=(-0.0692, -0.0692), Objective=0.0096
Iteration 40: Point=(-0.0599, -0.0599), Objective=0.0072
Iteration 41: Point=(-0.0484, -0.0484), Objective=0.0047
Iteration 42: Point=(-0.0358, -0.0358), Objective=0.0026
Iteration 43: Point=(-0.0234, -0.0234), Objective=0.0011
Iteration 44: Point=(-0.0121, -0.0121), Objective=0.0003
Iteration 45: Point=(-0.0026, -0.0026), Objective=0.0000
Iteration 46: Point=(0.0047, 0.0047), Objective=0.0000
Iteration 47: Point=(0.0096, 0.0096), Objective=0.0002
Iteration 48: Point=(0.0121, 0.0121), Objective=0.0003
Iteration 49: Point=(0.0125, 0.0125), Objective=0.0003
```

[ ]:
```
x_coords, y_coords = trajectory[:, 0], trajectory[:, 1]

plt.figure(figsize=(10, 6))
plt.plot(x_coords, y_coords, 'bo-', label='Trajectory with Combined Momentum␣
  ↪and RMSProp')
plt.plot(0, 0, 'go', label='Minimum (0,0)')  # Mark the minimum
plt.title('Combined Momentum and RMSProp Optimization Trajectory')
plt.xlabel('x')
plt.ylabel('y')
plt.grid(True)
plt.legend()
plt.show()
```



Adam Algorithm

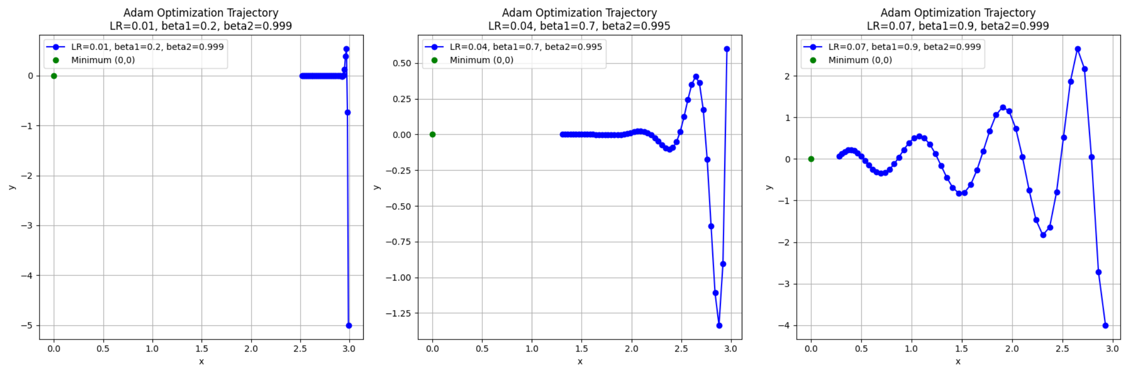
```
[ ]: def adam(start_point, learning_rate, beta1, beta2, epsilon, n_iter):
         x, y = start_point
         m_x = m_y = 0  # First moment vector
         v_x = v_y = 0  # Second moment vector
         trajectory = np.zeros((n_iter, 2))

         for i in range(1, n_iter + 1):
             grad = derivatives(x, y)
             m_x = beta1 * m_x + (1 - beta1) * grad[0]
             m_y = beta1 * m_y + (1 - beta1) * grad[1]
             v_x = beta2 * v_x + (1 - beta2) * grad[0]**2
             v_y = beta2 * v_y + (1 - beta2) * grad[1]**2
             m_hat_x = m_x / (1 - beta1**i)
             m_hat_y = m_y / (1 - beta2**i)
             v_hat_x = v_x / (1 - beta2**i)
             v_hat_y = v_y / (1 - beta2**i)
             x -= (learning_rate / (np.sqrt(v_hat_x) + epsilon)) * m_hat_x
             y -= (learning_rate / (np.sqrt(v_hat_y) + epsilon)) * m_hat_y
             trajectory[i - 1] = [x, y]
             print(f"Iteration {i}: Point=({x:.4f}, {y:.4f}),␣
      ↪Objective={objective(x, y):.4f}")

         return x, y, trajectory
```

```
[ ]: params = [
         {'lr': 0.01, 'beta1': 0.2, 'beta2': 0.999},
         {'lr': 0.04, 'beta1': 0.7, 'beta2': 0.995},
         {'lr': 0.07, 'beta1': 0.9, 'beta2': 0.999}
     ]

     start_point = (3.0, 3.0)
     epsilon = 1e-8
     n_iter = 50
```

```
[ ]: fig, axes = plt.subplots(1, len(params), figsize=(18, 6))
     for i, param in enumerate(params):
         _, _, trajectory = adam(start_point, param['lr'], param['beta1'],␣
      ↪param['beta2'], epsilon, n_iter)
         axes[i].plot(trajectory[:, 0], trajectory[:, 1], 'bo-',␣
      ↪label=f"LR={param['lr']}, beta1={param['beta1']}, beta2={param['beta2']}")
         axes[i].plot(0, 0, 'go', label='Minimum (0,0)')
         axes[i].set_title(f"Adam Optimization Trajectory\nLR={param['lr']},␣
      ↪beta1={param['beta1']}, beta2={param['beta2']}")
         axes[i].set_xlabel('x')
         axes[i].set_ylabel('y')
         axes[i].grid(True)
```

```
    axes[i].legend()

plt.tight_layout()
plt.show()
```

```
Iteration 1: Point=(2.9900, -5.0000), Objective=33.9401
Iteration 2: Point=(2.9800, -0.7297), Objective=9.4130
Iteration 3: Point=(2.9700, 0.5369), Objective=9.1094
Iteration 4: Point=(2.9601, 0.3909), Objective=8.9149
Iteration 5: Point=(2.9501, 0.1277), Objective=8.7196
Iteration 6: Point=(2.9402, 0.0089), Objective=8.6449
Iteration 7: Point=(2.9303, -0.0177), Objective=8.5870
Iteration 8: Point=(2.9204, -0.0142), Objective=8.5290
Iteration 9: Point=(2.9105, -0.0071), Objective=8.4713
Iteration 10: Point=(2.9007, -0.0027), Objective=8.4140
Iteration 11: Point=(2.8908, -0.0007), Objective=8.3570
Iteration 12: Point=(2.8810, -0.0001), Objective=8.3003
Iteration 13: Point=(2.8712, 0.0001), Objective=8.2439
Iteration 14: Point=(2.8614, 0.0001), Objective=8.1877
Iteration 15: Point=(2.8517, 0.0001), Objective=8.1319
Iteration 16: Point=(2.8419, 0.0000), Objective=8.0764
Iteration 17: Point=(2.8322, 0.0000), Objective=8.0211
Iteration 18: Point=(2.8224, 0.0000), Objective=7.9662
Iteration 19: Point=(2.8127, 0.0000), Objective=7.9115
Iteration 20: Point=(2.8030, 0.0000), Objective=7.8571
Iteration 21: Point=(2.7934, 0.0000), Objective=7.8029
Iteration 22: Point=(2.7837, 0.0000), Objective=7.7491
Iteration 23: Point=(2.7741, 0.0000), Objective=7.6956
Iteration 24: Point=(2.7645, 0.0000), Objective=7.6423
Iteration 25: Point=(2.7549, 0.0000), Objective=7.5893
Iteration 26: Point=(2.7453, 0.0000), Objective=7.5366
Iteration 27: Point=(2.7357, 0.0000), Objective=7.4842
Iteration 28: Point=(2.7262, 0.0000), Objective=7.4320
Iteration 29: Point=(2.7166, 0.0000), Objective=7.3801
Iteration 30: Point=(2.7071, 0.0000), Objective=7.3285
Iteration 31: Point=(2.6976, 0.0000), Objective=7.2772
Iteration 32: Point=(2.6881, 0.0000), Objective=7.2261
Iteration 33: Point=(2.6787, 0.0000), Objective=7.1753
Iteration 34: Point=(2.6692, 0.0000), Objective=7.1248
Iteration 35: Point=(2.6598, 0.0000), Objective=7.0746
Iteration 36: Point=(2.6504, 0.0000), Objective=7.0246
Iteration 37: Point=(2.6410, 0.0000), Objective=6.9749
Iteration 38: Point=(2.6316, 0.0000), Objective=6.9255
Iteration 39: Point=(2.6223, 0.0000), Objective=6.8763
Iteration 40: Point=(2.6129, 0.0000), Objective=6.8274
Iteration 41: Point=(2.6036, 0.0000), Objective=6.7788
Iteration 42: Point=(2.5943, 0.0000), Objective=6.7304
Iteration 43: Point=(2.5850, 0.0000), Objective=6.6823
Iteration 44: Point=(2.5757, 0.0000), Objective=6.6345
Iteration 45: Point=(2.5665, 0.0000), Objective=6.5869
Iteration 46: Point=(2.5573, 0.0000), Objective=6.5396
Iteration 47: Point=(2.5480, 0.0000), Objective=6.4925
Iteration 48: Point=(2.5388, 0.0000), Objective=6.4457
Iteration 49: Point=(2.5297, 0.0000), Objective=6.3992
Iteration 50: Point=(2.5205, 0.0000), Objective=6.3529
Iteration 1: Point=(2.9600, 0.6000), Objective=9.1216
Iteration 2: Point=(2.9200, -0.9032), Objective=9.3424
Iteration 3: Point=(2.8802, -1.3349), Objective=10.0772
Iteration 4: Point=(2.8404, -1.1092), Objective=9.2982
Iteration 5: Point=(2.8008, -0.6422), Objective=8.2568
Iteration 6: Point=(2.7613, -0.1740), Objective=7.6551
Iteration 7: Point=(2.7220, 0.1739), Objective=7.4397
Iteration 8: Point=(2.6830, 0.3614), Objective=7.3289
Iteration 9: Point=(2.6441, 0.4050), Objective=7.1553
Iteration 10: Point=(2.6055, 0.3490), Objective=6.9104
Iteration 11: Point=(2.5671, 0.2421), Objective=6.6487
Iteration 12: Point=(2.5290, 0.1239), Objective=6.4113
Iteration 13: Point=(2.4912, 0.0214), Objective=6.2065
Iteration 14: Point=(2.4536, -0.0519), Objective=6.0230
Iteration 15: Point=(2.4164, -0.0926), Objective=5.8473
Iteration 16: Point=(2.3794, -0.1048), Objective=5.6724
Iteration 17: Point=(2.3427, -0.0959), Objective=5.4973
Iteration 18: Point=(2.3063, -0.0745), Objective=5.3244
Iteration 19: Point=(2.2702, -0.0483), Objective=5.1560
Iteration 20: Point=(2.2344, -0.0230), Objective=4.9929
Iteration 21: Point=(2.1989, -0.0022), Objective=4.8350
Iteration 22: Point=(2.1637, 0.0124), Objective=4.6817
Iteration 23: Point=(2.1288, 0.0206), Objective=4.5322
Iteration 24: Point=(2.0942, 0.0234), Objective=4.3863
Iteration 25: Point=(2.0599, 0.0220), Objective=4.2439
Iteration 26: Point=(2.0260, 0.0181), Objective=4.1050
Iteration 27: Point=(1.9923, 0.0129), Objective=3.9696
Iteration 28: Point=(1.9590, 0.0077), Objective=3.8378
Iteration 29: Point=(1.9260, 0.0030), Objective=3.7095
Iteration 30: Point=(1.8933, -0.0006), Objective=3.5846
Iteration 31: Point=(1.8609, -0.0030), Objective=3.4631
Iteration 32: Point=(1.8289, -0.0043), Objective=3.3448
Iteration 33: Point=(1.7971, -0.0046), Objective=3.2297
Iteration 34: Point=(1.7657, -0.0043), Objective=3.1177
Iteration 35: Point=(1.7346, -0.0036), Objective=3.0088
Iteration 36: Point=(1.7038, -0.0026), Objective=2.9030
Iteration 37: Point=(1.6733, -0.0017), Objective=2.8001
Iteration 38: Point=(1.6432, -0.0008), Objective=2.7001
Iteration 39: Point=(1.6134, -0.0001), Objective=2.6031
Iteration 40: Point=(1.5839, 0.0004), Objective=2.5088
Iteration 41: Point=(1.5547, 0.0007), Objective=2.4172
Iteration 42: Point=(1.5259, 0.0008), Objective=2.3284
Iteration 43: Point=(1.4974, 0.0008), Objective=2.2422
Iteration 44: Point=(1.4692, 0.0007), Objective=2.1586
Iteration 45: Point=(1.4413, 0.0006), Objective=2.0775
Iteration 46: Point=(1.4138, 0.0004), Objective=1.9989
Iteration 47: Point=(1.3866, 0.0002), Objective=1.9227
Iteration 48: Point=(1.3597, 0.0001), Objective=1.8488
Iteration 49: Point=(1.3332, 0.0000), Objective=1.7773
Iteration 50: Point=(1.3069, -0.0001), Objective=1.7081
Iteration 1: Point=(2.9300, -4.0000), Objective=24.5849
Iteration 2: Point=(2.8600, -2.7125), Objective=15.5376
Iteration 3: Point=(2.7902, 0.0487), Objective=7.7875
Iteration 4: Point=(2.7204, 2.1724), Objective=12.1201
Iteration 5: Point=(2.6508, 2.6509), Objective=14.0541
Iteration 6: Point=(2.5814, 1.8676), Objective=10.1517
Iteration 7: Point=(2.5122, 0.5203), Objective=6.5819
Iteration 8: Point=(2.4433, -0.7985), Objective=6.6072
Iteration 9: Point=(2.3747, -1.6421), Objective=8.3356
Iteration 10: Point=(2.3064, -1.8328), Objective=8.6787
Iteration 11: Point=(2.2385, -1.4646), Objective=7.1559
Iteration 12: Point=(2.1710, -0.7576), Objective=5.2873
Iteration 13: Point=(2.1040, 0.0469), Objective=4.4290
Iteration 14: Point=(2.0375, 0.7334), Objective=4.6892
Iteration 15: Point=(1.9715, 1.1521), Objective=5.2142
Iteration 16: Point=(1.9062, 1.2485), Objective=5.1921
Iteration 17: Point=(1.8414, 1.0568), Objective=4.5077
Iteration 18: Point=(1.7774, 0.6665), Objective=3.6032
Iteration 19: Point=(1.7140, 0.1884), Objective=2.9733
Iteration 20: Point=(1.6514, -0.2690), Objective=2.7995
Iteration 21: Point=(1.5896, -0.6181), Objective=2.9089
Iteration 22: Point=(1.5286, -0.8064), Objective=2.9869
Iteration 23: Point=(1.4685, -0.8214), Objective=2.8312
Iteration 24: Point=(1.4093, -0.6858), Objective=2.4565
Iteration 25: Point=(1.3511, -0.4452), Objective=2.0237
Iteration 26: Point=(1.2938, -0.1559), Objective=1.6983
Iteration 27: Point=(1.2376, 0.1264), Objective=1.5477
Iteration 28: Point=(1.1824, 0.3548), Objective=1.5240
Iteration 29: Point=(1.1283, 0.4984), Objective=1.5216
Iteration 30: Point=(1.0754, 0.5450), Objective=1.4534
Iteration 31: Point=(1.0235, 0.4997), Objective=1.2974
Iteration 32: Point=(0.9729, 0.3819), Objective=1.0923
Iteration 33: Point=(0.9234, 0.2189), Objective=0.9006
Iteration 34: Point=(0.8752, 0.0412), Objective=0.7677
Iteration 35: Point=(0.8282, -0.1227), Objective=0.7010
Iteration 36: Point=(0.7825, -0.2501), Objective=0.6749
Iteration 37: Point=(0.7381, -0.3270), Objective=0.6517
Iteration 38: Point=(0.6950, -0.3484), Objective=0.6044
Iteration 39: Point=(0.6532, -0.3182), Objective=0.5279
Iteration 40: Point=(0.6127, -0.2470), Objective=0.4364
Iteration 41: Point=(0.5735, -0.1496), Objective=0.3513
Iteration 42: Point=(0.5357, -0.0425), Objective=0.2888
Iteration 43: Point=(0.4993, 0.0585), Objective=0.2527
Iteration 44: Point=(0.4642, 0.1406), Objective=0.2352
Iteration 45: Point=(0.4304, 0.1952), Objective=0.2233
Iteration 46: Point=(0.3980, 0.2184), Objective=0.2060
Iteration 47: Point=(0.3669, 0.2108), Objective=0.1790
Iteration 48: Point=(0.3371, 0.1770), Objective=0.1450
Iteration 49: Point=(0.3087, 0.1245), Objective=0.1108
Iteration 50: Point=(0.2815, 0.0621), Objective=0.0831
```

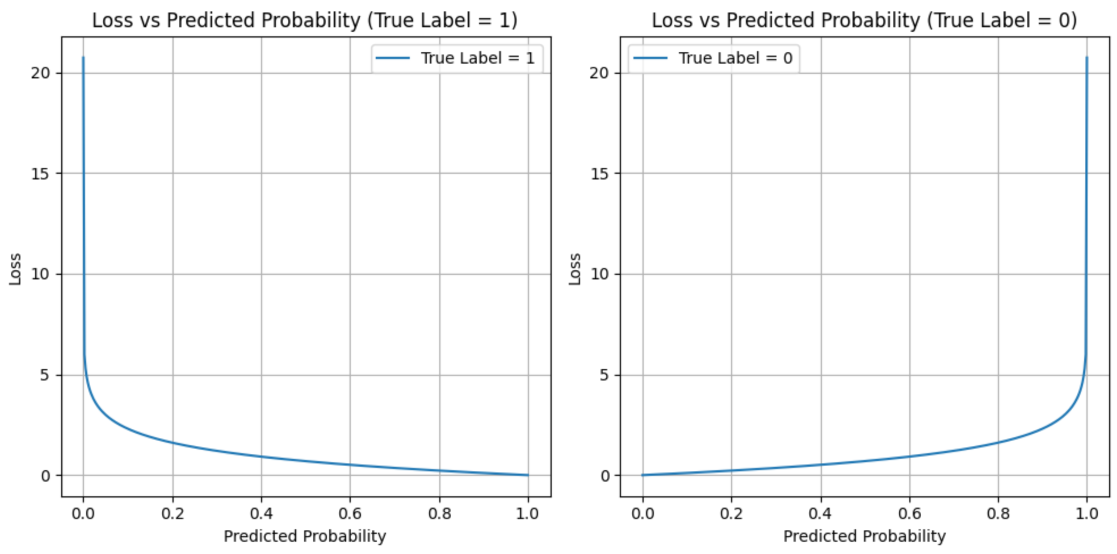


0.8.2 Binary Cross Entropy

[ ]:
```
def binary_cross_entropy(y_true, y_pred):
    # avoid division by zero
    y_pred = np.clip(y_pred, 1e-9, 1 - 1e-9)

    return -y_true * np.log(y_pred) - (1 - y_true) * np.log(1 - y_pred) #␣
 ↪formula for binary cross entropy
```

[ ]:
```
predictions = np.linspace(0, 1, 400)

# True label = 1
loss_when_true = binary_cross_entropy(1, predictions)

# True label = 0
loss_when_false = binary_cross_entropy(0, predictions)
```

[ ]:
```
plt.figure(figsize=(10, 5))

# Plot for true label = 1
plt.subplot(1, 2, 1)
plt.plot(predictions, loss_when_true, label='True Label = 1')
plt.title('Loss vs Predicted Probability (True Label = 1)')
plt.xlabel('Predicted Probability')
plt.ylabel('Loss')
plt.grid(True)
plt.legend()

# Plot for true label = 0
plt.subplot(1, 2, 2)
plt.plot(predictions, loss_when_false, label='True Label = 0')
plt.title('Loss vs Predicted Probability (True Label = 0)')
plt.xlabel('Predicted Probability')
plt.ylabel('Loss')
plt.grid(True)
plt.legend()

plt.tight_layout()
plt.show()
```



0.8.3 Rectified Linear Unit, Relu Activation Function

[ ]:
```
from ipywidgets import interact

def relu(x):
    return np.maximum(0, x)

def plot_relu(x_range):
    x = np.linspace(-x_range, x_range, 400)
    y = relu(x)

    plt.figure(figsize=(10, 5))
    plt.plot(x, y, label='ReLU: max(0, x)', linewidth=2)
    plt.title('ReLU Activation Function')
    plt.xlabel('Input (x)')
    plt.ylabel('Output (ReLU(x))')
    plt.grid(True)
    plt.axhline(0, color='gray', lw=0.5)
    plt.axvline(0, color='gray', lw=0.5)
    plt.legend()
    plt.show()

interact(plot_relu, x_range=(1, 10));
```

```
interactive(children=(IntSlider(value=5, description='x_range', max=10, min=1),␣
↪Output()), _dom_classes=('widg…
```

[ ]:
```
from sklearn.datasets import make_circles
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# generate synthetic data
X, y = make_circles(n_samples=100, factor=0.5, noise=0.1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)

# train a model using ReLU
model_relu = MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000,␣
↪activation='relu', random_state=1)
model_relu.fit(X_train, y_train)

accuracy_relu = accuracy_score(y_test, model_relu.predict(X_test))

print(f"Accuracy with ReLU: {accuracy_relu:.2f}")
```

```
Accuracy with ReLU: 1.00
```

```
/usr/local/lib/python3.10/dist-
packages/sklearn/neural_network/_multilayer_perceptron.py:686:
ConvergenceWarning: Stochastic Optimizer: Maximum iterations (1000) reached and
the optimization hasn't converged yet.
  warnings.warn(
```

0.8.4 Sigmoid Activation Function

```
[ ]: def sigmoid(x):
         return 1 / (1 + np.exp(-x))

     def plot_sigmoid(x_range):
         x = np.linspace(-x_range, x_range, 400)
         y = sigmoid(x)

         plt.figure(figsize=(10, 5))
         plt.plot(x, y, label='Sigmoid: 1 / (1 + exp(-x))', linewidth=2)
         plt.title('Sigmoid Activation Function')
         plt.xlabel('Input (x)')
         plt.ylabel('Output (Sigmoid(x))')
         plt.grid(True)
         plt.axhline(0, color='gray', lw=0.5)
         plt.axhline(1, color='gray', lw=0.5)
         plt.axvline(0, color='gray', lw=0.5)
         plt.legend()
         plt.show()

     interact(plot_sigmoid, x_range=(1, 10));
```

```
interactive(children=(IntSlider(value=5, description='x_range', max=10, min=1),␣
 ↪Output()), _dom_classes=('widg…
```

```
[ ]: X, y = make_circles(n_samples=100, factor=0.5, noise=0.1)

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
      ↪random_state=42)

     # train a model using Sigmoid
     model_sigmoid = MLPClassifier(hidden_layer_sizes=(10,), max_iter=1000,␣
      ↪activation='logistic', random_state=1)
     model_sigmoid.fit(X_train, y_train)

     accuracy_sigmoid = accuracy_score(y_test, model_sigmoid.predict(X_test))

     print(f"Accuracy with Sigmoid: {accuracy_sigmoid:.2f}")
```

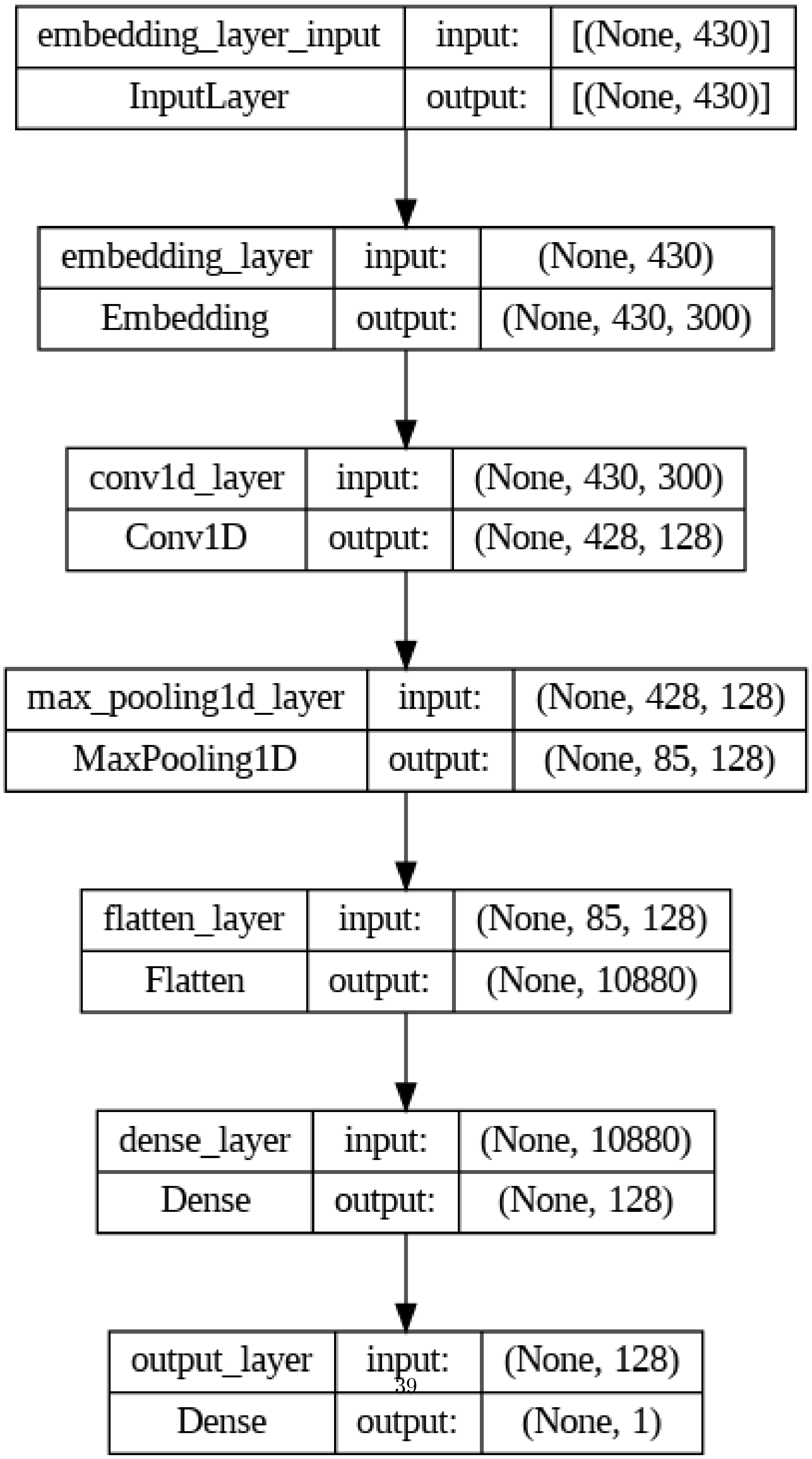
```
Accuracy with Sigmoid: 0.43
```

0.8.5 Layers in Neural Networks

```
[ ]: model = Sequential([
         Embedding(input_dim=10000, output_dim=300, input_length=430,␣
      ↪name='embedding_layer'),
         Conv1D(filters=128, kernel_size=3, activation='relu', name='conv1d_layer'),
         MaxPooling1D(pool_size=5, name='max_pooling1d_layer'),
         Flatten(name='flatten_layer'),
         Dense(128, activation='relu', name='dense_layer'),
         Dense(1, activation='sigmoid', name='output_layer')
     ])

     model.summary()

     plot_model(model, to_file='model_structure.png', show_shapes=True,␣
      ↪show_layer_names=True)
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_layer (Embedding  (None, 430, 300)          3000000
 )

 conv1d_layer (Conv1D)       (None, 428, 128)          115328

 max_pooling1d_layer (MaxPo  (None, 85, 128)           0
 oling1D)

 flatten_layer (Flatten)     (None, 10880)             0

 dense_layer (Dense)         (None, 128)               1392768

 output_layer (Dense)        (None, 1)                 129

=================================================================
Total params: 4508225 (17.20 MB)
Trainable params: 4508225 (17.20 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

[ ]:

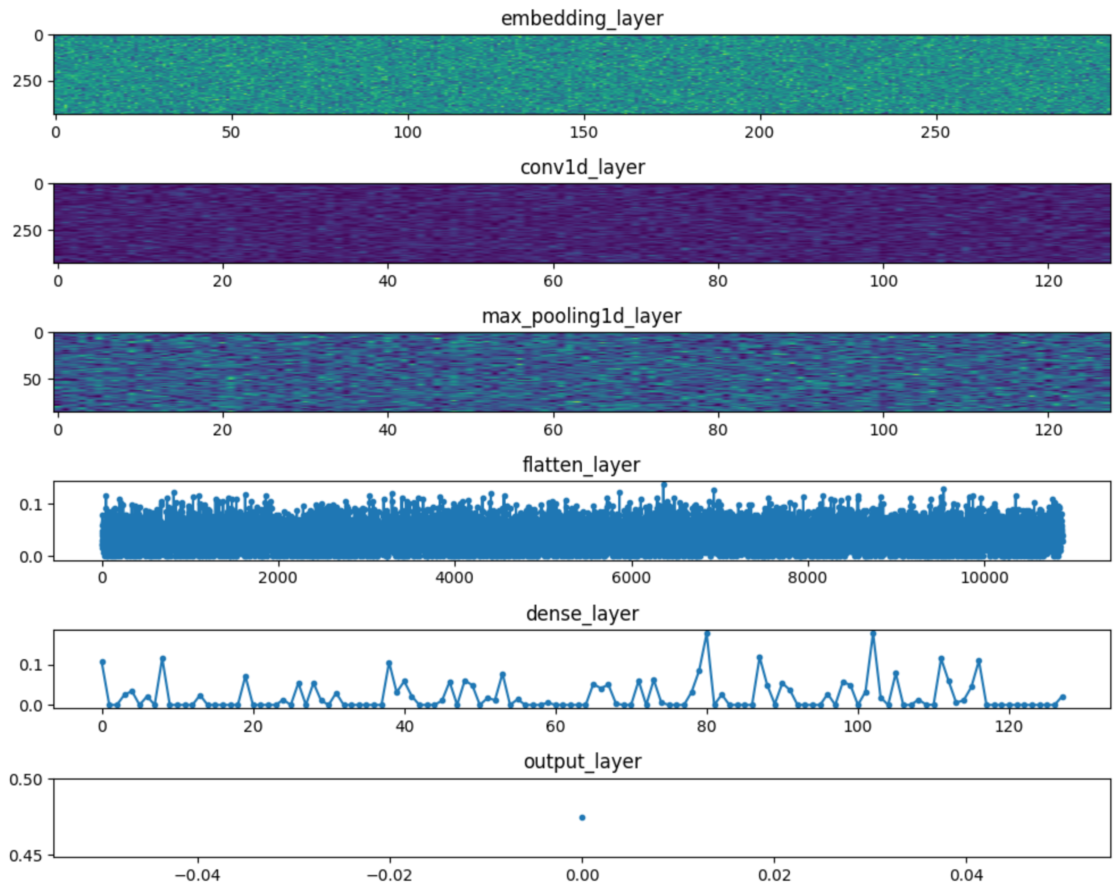

embedding_layer_input
InputLayer
input:
output:
[(None, 430)]
[(None, 430)]
embedding_layer
Embedding
input:
output:
(None, 430)
(None, 430, 300)
conv1d_layer
Conv1D
input:
output:
(None, 430, 300)
(None, 428, 128)
max_pooling1d_layer
MaxPooling1D
input:
output:
(None, 428, 128)
(None, 85, 128)
flatten_layer
Flatten
input:
output:
(None, 85, 128)
(None, 10880)
dense_layer
Dense
input:
output:
(None, 10880)
(None, 128)
output_layer
Dense
input:
output:
(None, 128)
(None, 1)

```
[ ]: from tensorflow.keras.models import Model

     sample_input = np.random.randint(10000, size=(1, 430))

     # a model that outputs from each layer
     layer_outputs = [layer.output for layer in model.layers]
     activation_model = Model(inputs=model.input, outputs=layer_outputs)

     # the outputs for each layer
     activations = activation_model.predict(sample_input)
```

```
1/1 [==============================] - 0s 407ms/step
```

```
[ ]: def plot_layer_activations(layer_activations, layer_names):
         plt.figure(figsize=(10, 8))
         for i, activation in enumerate(layer_activations):
             plt.subplot(len(layer_activations), 1, i+1)
             plt.title(layer_names[i])
             if activation.ndim > 2:  # Not flatten or dense layer
                 plt.imshow(activation[0, :, :], aspect='auto', cmap='viridis')
             else:  # Flatten or dense layer
                 plt.plot(activation[0, :], '.-')
             plt.grid(False)
         plt.tight_layout()
         plt.show()
```

```
[ ]: layer_names = [layer.name for layer in model.layers]
     plot_layer_activations(activations, layer_names)
```



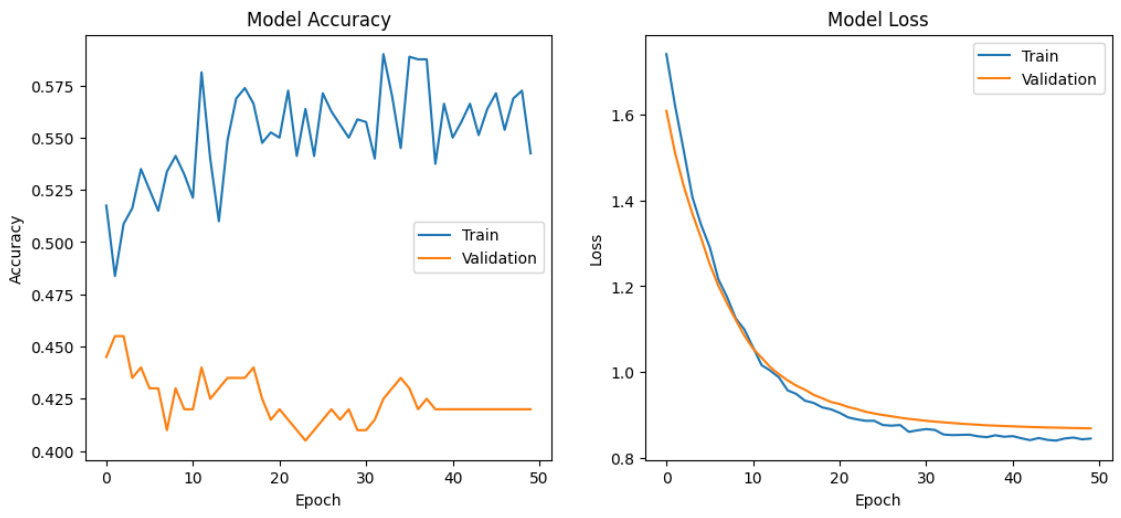
0.8.6 Training and Validation Loss & Accuracy Values

```
[ ]: from tensorflow.keras.utils import to_categorical
     from sklearn.model_selection import train_test_split

     # generate synthetic binary classification data
     np.random.seed(0)
     data = np.random.randn(1000, 20)
     labels = np.random.randint(2, size=(1000, 1))

     X_train, X_val, y_train, y_val = train_test_split(data, labels, test_size=0.2,␣
       ↪random_state=42)

     # one-hot encoding
     y_train = to_categorical(y_train, num_classes=2)
     y_val = to_categorical(y_val, num_classes=2)
```

[ ]:
```python
from tensorflow.keras.layers import Dropout
from tensorflow.keras.regularizers import l2

model = Sequential([
    Dense(64, activation='relu', input_shape=(20,), kernel_regularizer=l2(0.
 ↪01)),
    Dropout(0.5),
    Dense(64, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense(2, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[ ]:
```python
def scheduler(epoch, lr):
    if epoch < 10:
        return lr
    else:
        return lr * tf.math.exp(-0.1)

callback = tf.keras.callbacks.LearningRateScheduler(scheduler)

history = model.fit(X_train, y_train,
                    epochs=50,
                    validation_data=(X_val, y_val),
                    callbacks=[callback],
                    verbose=0)
```

[ ]:
```python
# training & validation accuracy values
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train')
plt.plot(history.history['val_accuracy'], label='Validation')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

# training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train')
plt.plot(history.history['val_loss'], label='Validation')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



0.9 Create Word Embedding Matrix for CNN

[ ]:
```
texts = df['clean_text'].tolist()
labels_cnn = df['class'].values
```

[ ]:
```
num_words_list = [20000, 25000, 30000, 35000, 40000]
tokenizer = Tokenizer(num_words=35000)  # Only the top 35000 words will be kept
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts) # list of lists where each␣
 ↪integer represents one of the top 35000 words

sequences[:1] # The first text instance represented as a series of integers
```

[ ]:
```
[[21,
  1,
  678,
  1290,
  12,
  11,
  3531,
  458,
  1915,
  21,
  1,
  1430,
  36,
  1450,
  12,
  11,
  289,
  541,
  164,
  2098,
  1392,
  4626,
  3635,
  587,
  44,
  68,
  58,
  879,
  63,
  858,
  10,
  177,
  27,
  3568,
  832,
  2422,
  7433,
  47,
  1430,
  531,
  25,
  1392,
  4626,
  54,
  3635,
  587,
  44,
  68,
  1450,
  3267,
  12,
  11,
  5,
  1414,
  4320,
  220,
  325,
  11,
  90,
  27,
  3568,
  832,
  2422,
  7433,
  47,
  1430,
  531,
  25,
  1392,
  4626,
  54,
  3635,
  587,
  44,
  68,
  1450,
  3267,
  12,
  11,
  325,
  11,
  90,
  21,
  811,
  1,
  610,
  1,
  220,
  364,
  627,
  5,
  678,
  12,
  11,
  3459,
  17,
  4963,
  4381,
  16633,
  19455,
  1,
  957,
  5481,
  54,
  165,
  817,
  9649,
  279,
  455,
  1430,
  36,
  423,
  1450,
  12,
  11,
  3866,
  13935,
  610,
  10,
  17,
  14502,
  610,
  1624,
  6,
  34,
  325,
  11,
  90,
  13,
  588,
  4287,
  246,
  131,
  9239,
  610,
  355,
  98,
  48,
  66,
  7,
  465,
  1096,
  465,
  4288,
  11575,
  610,
  43,
  385,
  4626,
  12,
  11,
  1430,
  24768,
  610,
  8,
  1450,
  12,
  11,
  610,
  1,
  12,
  11,
  3531,
  220,
  1450,
  12,
  11,
  48,
  66,
  1392,
  1940,
  648,
  2799,
  77,
  567,
  21,
  811,
  1,
  610,
  395,
  12,
  1,
  31696,
  548,
  458,
  1392,
  4626,
  12,
  11,
  5607,
  5120,
  3250,
  2530,
  9326,
  610,
  1,
  2177,
  220,
  813,
  279,
  49,
  7002,
  207,
  5,
  2154,
  31697,
  610,
  715,
  17,
  359,
  42,
  11,
  6590,
  566,
  2185,
  3902,
  610,
  1461,
  2527,
  43,
  438,
  4626,
  7397,
  610,
  2913,
  11,
  696,
  77,
  610,
  7,
  37,
  2908,
  75,
  6,
  114,
  301,
  117,
  770,
  546,
  436,
  114,
  11,
  1671,
  665]]
```

[ ]:
```
max_len = 430
X_padded = pad_sequences(sequences, maxlen=max_len)
```

```
# embedding matrix
vocab_size = min(len(tokenizer.word_index) + 1, 35001)
embedding_dim = wv.vector_size
embedding_matrix = np.zeros((vocab_size, embedding_dim))

embedding_matrix = np.zeros((vocab_size, embedding_dim))
for word, i in tokenizer.word_index.items():
    if i < vocab_size:
        try:
            embedding_vector = wv[word]
            if embedding_vector is not None:
                embedding_matrix[i] = embedding_vector
        except KeyError:
            continue
```

```
[ ]: print("Shape of embedding matrix:", embedding_matrix.shape)
     print("Sample values from the embedding matrix:\n", embedding_matrix[1:10])
```

```
Shape of embedding matrix: (35001, 300)
Sample values from the embedding matrix:
 [[-0.07910156  0.12158203 -0.00842285 … -0.39257812  0.07763672
   0.27148438]
 [-0.00909424 -0.04418945  0.09960938 …  0.14453125  0.18066406
  -0.08691406]
 [ 0.02990723  0.05639648  0.0037384  … -0.02416992  0.01086426
  -0.14746094]
 …
 [ 0.04052734 -0.07324219  0.06201172 … -0.04614258 -0.09570312
  -0.02050781]
 [-0.03613281 -0.12109375  0.13378906 … -0.08642578  0.14355469
   0.02734375]
 [-0.03393555 -0.17871094  0.09033203 … -0.05078125  0.17285156
   0.29492188]]
```

0.10 Building the CNN Model and HyperParameter Tuning for the Dataset

```
[ ]: texts = df['clean_text'].tolist()
     labels_cnn = df['class'].values

     X_train, X_test, y_train, y_test = train_test_split(X_padded, labels_cnn,␣
      ↪test_size=0.2, random_state=42)
```

```
[ ]: print(X_train[0])
     print(y_train[0])

     print('Shape of training data: ')
     print(X_train.shape)
```

```python
print(y_train.shape)

print('Shape of test data: ')
print(X_test.shape)
print(y_test.shape)
```

```
[    0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0
     0     0   454   109  2786  1727     1    65  2252   858   154   146
   927   648   709   109  1539  1484     9    85   332    21     1    65
   154  5490 16040  9272 13840   709    53  3104  4166 23880   626  1539
 16912   484   252   173   922  2630    50 26772  2435 13840   479   884
  9272  1846    63   325  2965   302   772    19   800   425  1846 28745
   287   496  1389    63  3340  2965  2562   302   899  1033    21     1
  1396  1108     8   287  1164   126    21     1   280  2106  9410   922
  1396  2756   154  2756    65  2252   858   154    65   242  2853   242
     1    65  1846   154   182  1395    44    13  1021   159  1186  1186
   709   127    79  3377   119   464     1   552   307  2748   119  1458
   356   109   482 13840    13   515   144   436   139   109    34  3377
   219    60   109   154    34  3377   219    60    21     1  5613   514
   237   329   329   177   645   514    54   364  4966   645   514   645
     7   247    13   144   957   659  1002     9    35    73    17    29
   128    73  1794  1034   816    14   589   503  1119   302    41    65
  2756  2252   858   154   404    32  1119   302   589   503   302  1794
  1034   302  2942  2010   302  3103    56    18   924  1091  1846   154
    48    95   508  3967  1846   154   162   521    90  5359]
0
Shape of training data:
(35895, 430)
(35895,)
Shape of test data:
```

```
(8974, 430)
(8974,)
```

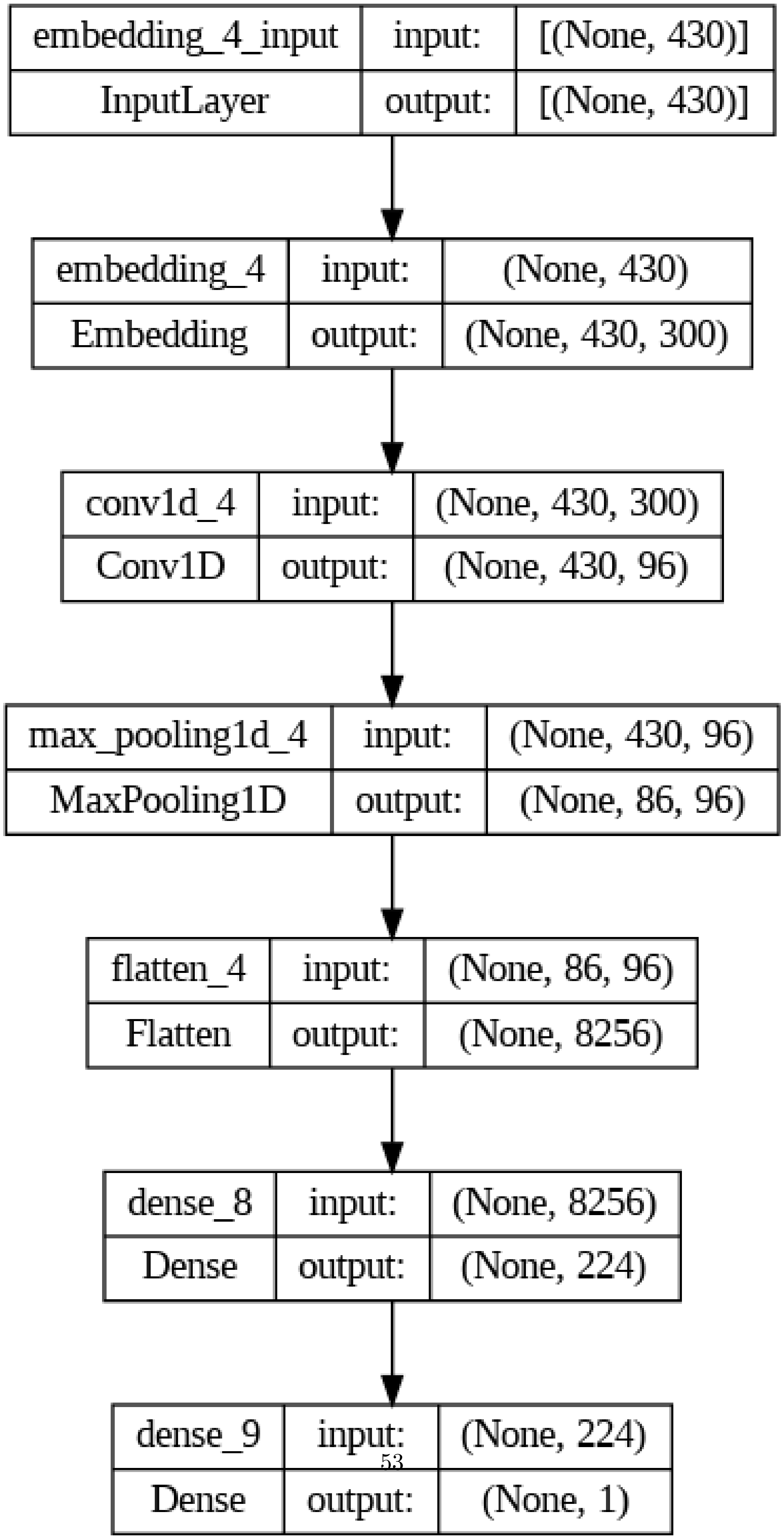
[ ]:
```
from tensorflow.keras.metrics import Precision, Recall

class CNNHyperModel(HyperModel):
    def __init__(self, vocab_size, embedding_dim, max_len, embedding_matrix):
        self.vocab_size = vocab_size
        self.embedding_dim = embedding_dim
        self.max_len = max_len
        self.embedding_matrix = embedding_matrix

    def build(self, hp):
        model = Sequential()
        model.add(Embedding(self.vocab_size, self.embedding_dim,
                            weights=[self.embedding_matrix],
                            input_length=self.max_len,
                            trainable=False))
        model.add(Conv1D(
            filters=hp.Int('conv_1_filters', min_value=32, max_value=128,␣
 ↪step=32),
            kernel_size=hp.Choice('conv_1_kernel_size', values=[3, 5, 7]),
            padding='same',
            activation='relu'))
        model.add(MaxPooling1D(
            pool_size=hp.Choice('max_pool_1_size', values=[2, 5, 7])))
        model.add(Flatten())
        model.add(Dense(
            units=hp.Int('dense_1_units', min_value=64, max_value=256, step=32),
            activation='relu'))
        model.add(Dense(1, activation='sigmoid'))
        model.compile(
            optimizer='adam',
            loss='binary_crossentropy',
            metrics=['accuracy', Precision(), Recall()])
        return model
```

[ ]:
```
tuner = RandomSearch(
    hypermodel=CNNHyperModel(vocab_size, embedding_dim, max_len,␣
 ↪embedding_matrix),
    objective='val_accuracy',
    max_trials=4,
    executions_per_trial=1,
    directory='my_dir',
    project_name='keras_tuner_cnn')

tuner.search(x=X_train, y=y_train,
              epochs=7,
              batch_size=128,
              validation_data=(X_test, y_test),
              verbose=2)
```

```
Reloading Tuner from my_dir/keras_tuner_cnn/tuner0.json
```

```
[ ]: from IPython.display import Image

     best_model = CNNHyperModel(vocab_size, embedding_dim, max_len,␣
      ↪embedding_matrix).build(best_hps)

     best_model.compile(optimizer='adam',
                        loss='binary_crossentropy',
                        metrics=['accuracy', Precision(), Recall()])

     plot_model(best_model, to_file='best_model_architecture.png', show_shapes=True,␣
      ↪show_layer_names=True)

     Image(filename='best_model_architecture.png')
```

```
[ ]:
```


embedding_4_input
InputLayer
input:
output:
[(None, 430)]
[(None, 430)]
embedding_4
Embedding
input:
output:
(None, 430)
(None, 430, 300)
conv1d_4
Conv1D
input:
output:
(None, 430, 300)
(None, 430, 96)
max_pooling1d_4
MaxPooling1D
input:
output:
(None, 430, 96)
(None, 86, 96)
flatten_4
Flatten
input:
output:
(None, 86, 96)
(None, 8256)
dense_8
Dense
input:
output:
(None, 8256)
(None, 224)
dense_9
Dense
input:
output:
(None, 224)
(None, 1)

[ ]:
```
# get the optimal hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]

print('Best hyperparameters found:\n')
print(f"Best number of filters in the first Conv1D layer: {best_hps.
  ↪get('conv_1_filters')}")
print(f"Best kernel size for the first Conv1D layer: {best_hps.
  ↪get('conv_1_kernel_size')}")
print(f"Best pooling size for the first MaxPooling1D layer: {best_hps.
  ↪get('max_pool_1_size')}")
print(f"Best number of units in the first Dense layer: {best_hps.
  ↪get('dense_1_units')}")
```

```
Best hyperparameters found:

Best number of filters in the first Conv1D layer: 96
Best kernel size for the first Conv1D layer: 7
Best pooling size for the first MaxPooling1D layer: 5
Best number of units in the first Dense layer: 224
```

[ ]:
```
best_model = tuner.get_best_models(num_models=1)[0]

best_model.compile(optimizer='adam',
                   loss='binary_crossentropy',
                   metrics=['accuracy', Precision(), Recall()])

results = best_model.evaluate(X_test, y_test, verbose=0)

print(f"Accuracy: {results[1]*100:.2f}%")
print(f"Precision: {results[2]*100:.2f}%")
print(f"Recall: {results[3]*100:.2f}%")

f1_score = 2 * (results[2] * results[3]) / (results[2] + results[3])
print(f"F1 Score: {f1_score:.2f}%")
```

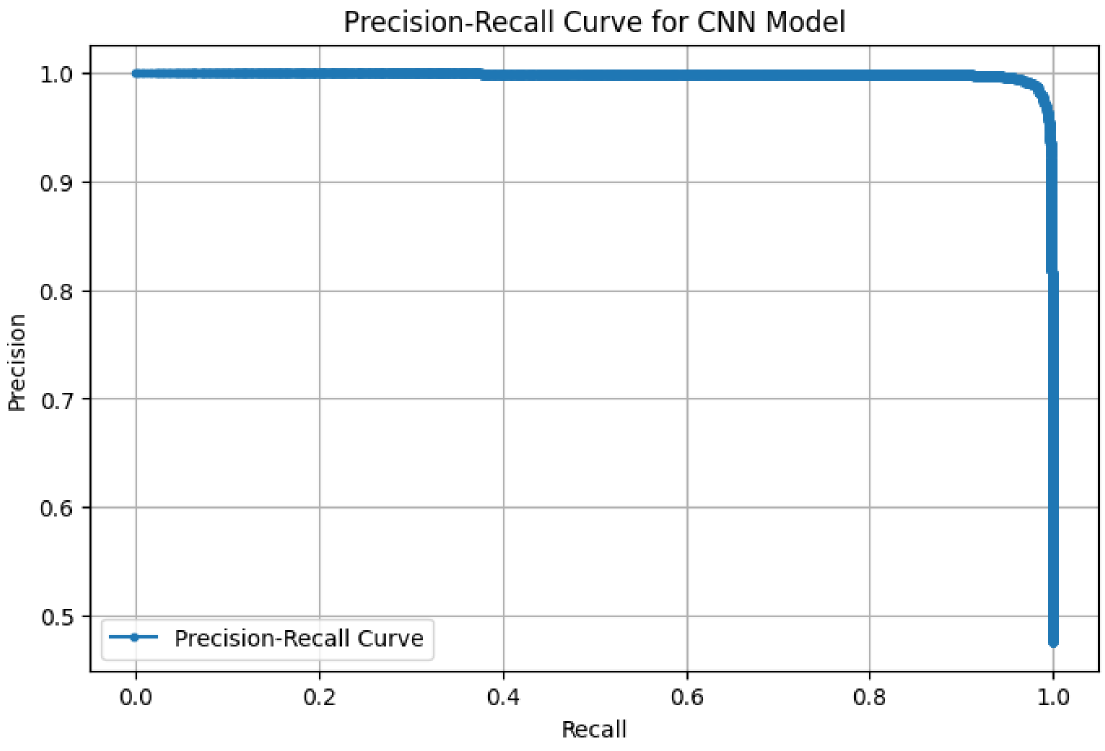
```
WARNING:tensorflow:Detecting that an object or model or tf.train.Checkpoint is
being deleted with unrestored values. See the following logs for the specific
values in question. To silence these warnings, use `status.expect_partial()`.
See https://www.tensorflow.org/api_docs/python/tf/train/Checkpoint#restorefor
details about the status object returned by the restore function.
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.1
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.2
WARNING:tensorflow:Value in checkpoint could not be found in the restored
```

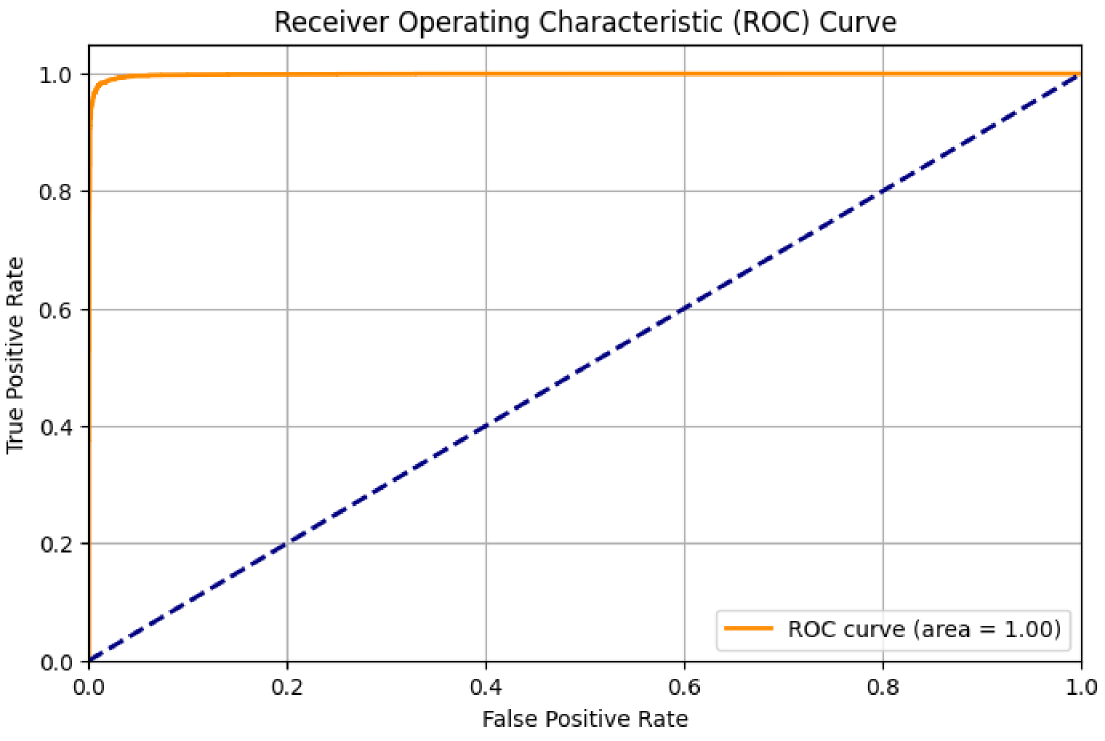
```
object: (root).optimizer._variables.3
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.4
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.5
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.6
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.7
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.8
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.9
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.10
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.11
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.12
WARNING:tensorflow:Detecting that an object or model or tf.train.Checkpoint is
being deleted with unrestored values. See the following logs for the specific
values in question. To silence these warnings, use `status.expect_partial()`.
See https://www.tensorflow.org/api_docs/python/tf/train/Checkpoint#restorefor
details about the status object returned by the restore function.
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.1
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.2
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.3
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.4
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.5
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.6
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.7
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.8
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.9
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.10
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.11
WARNING:tensorflow:Value in checkpoint could not be found in the restored
object: (root).optimizer._variables.12
```

```
Accuracy: 98.57%
Precision: 98.73%
Recall: 98.27%
F1 Score: 0.98%
```

```
[ ]: from sklearn.metrics import precision_recall_curve, roc_curve, auc

     y_pred_prob = best_model.predict(X_test)

     precision, recall, thresholds = precision_recall_curve(y_test, y_pred_prob)
     plt.figure(figsize=(8, 5))
     plt.plot(recall, precision, marker='.', label='Precision-Recall Curve')
     plt.xlabel('Recall')
     plt.ylabel('Precision')
     plt.title('Precision-Recall Curve for CNN Model')
     plt.legend()
     plt.grid(True)
     plt.show()

     # ROC curve
     fpr, tpr, _ = roc_curve(y_test, y_pred_prob)
     roc_auc = auc(fpr, tpr)
     plt.figure(figsize=(8, 5))
     plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:
       ↪.2f})')
     plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
     plt.xlim([0.0, 1.0])
     plt.ylim([0.0, 1.05])
     plt.xlabel('False Positive Rate')
     plt.ylabel('True Positive Rate')
     plt.title('Receiver Operating Characteristic (ROC) Curve')
     plt.legend(loc="lower right")
     plt.grid(True)
     plt.show()
```

```
281/281 [==============================] - 64s 226ms/step
```

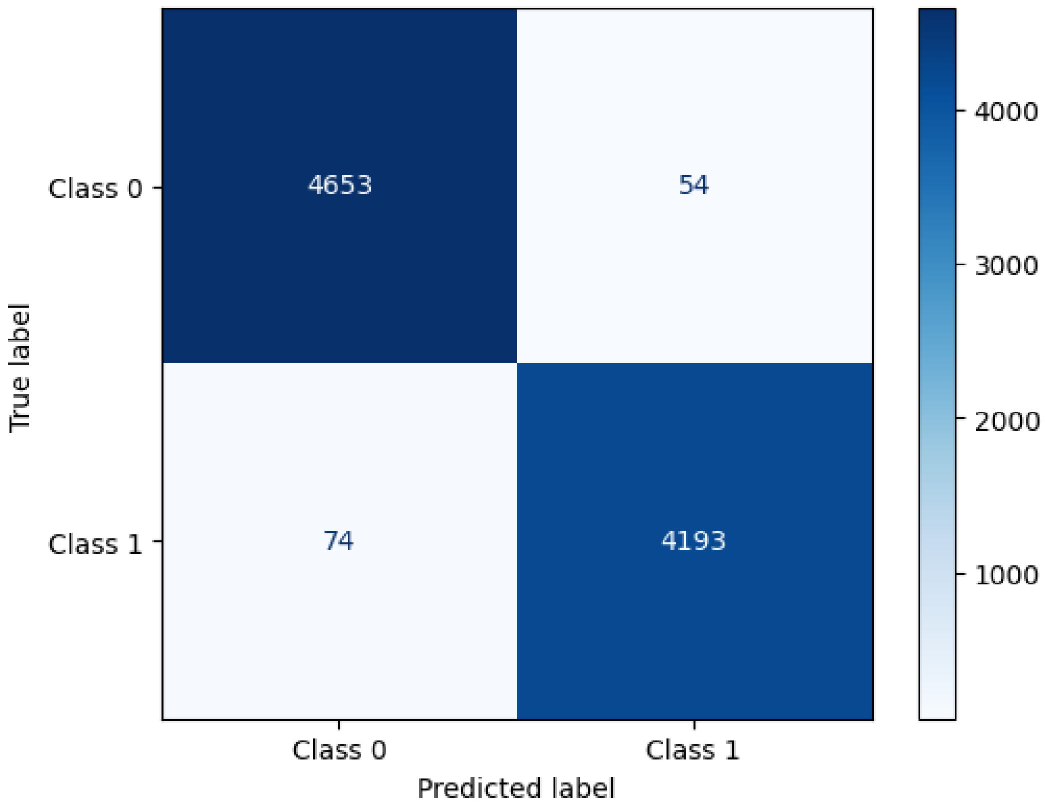

Precision-Recall Curve for CNN Model
Precision
Recall
Precision-Recall Curve
0.0
0.2
0.4
0.6
0.8
1.0
0.5
0.6
0.7
0.8
0.9
1.0


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC curve (area = 1.00)
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]: from sklearn.metrics import precision_recall_curve, roc_curve, auc,␣
      ↪classification_report, confusion_matrix
     from sklearn.metrics import ConfusionMatrixDisplay

     y_pred = (y_pred_prob > 0.5).astype(int)

     print(classification_report(y_test, y_pred, target_names=['Class 0', 'Class␣
      ↪1']))
```

```
              precision    recall  f1-score   support

     Class 0       0.98      0.99      0.99      4707
     Class 1       0.99      0.98      0.98      4267

    accuracy                           0.99      8974
   macro avg       0.99      0.99      0.99      8974
weighted avg       0.99      0.99      0.99      8974
```

```
[ ]: cm = confusion_matrix(y_test, y_pred)
     disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=['Class 0',␣
      ↪'Class 1'])
     disp.plot(cmap='Blues')
     plt.show()
```



```
[ ]: misclassified_indices = np.where(y_pred.flatten() != y_test)[0]
     misclassified_texts = [texts[i] for i in misclassified_indices]
     misclassified_probs = y_pred_prob[misclassified_indices]
     actual_labels = y_test[misclassified_indices]

     for i in range(min(20, len(misclassified_texts))):
         print(f"Text: {misclassified_texts[i]}")
         print(f"Predicted Prob: {misclassified_probs[i][0]}, Actual Label:␣
      ↪{actual_labels[i]}\n")
```

```
Text: ohio fireman deep sht horribl remark save ns firefight ohio recent came
fire express econom uncertainti synonym media concoct racism attempt explain
peopl vote trump rescu black peopleif came choos rescu singl dog million ngger
franklin township firefight tyler roysdon say would save dog import
africanamerican hatesaccord roysdon facebook like fan donald trump duck dynasti
farright facebook group uncl sam misguid children know sort person deal
withroysdon suspend posit indefinit soon township offici learn post went viral
social media happen terribl peopl say terribl thing also like lose job accord
statement offici recent franklin township volunt firefight post unaccept remark
social media upon gain knowledg inform fire chief steve bishop immedi contact
firefight direct comment remov firefight suspend without pay board township
```

```
truste could meet determin cours action chief bishop author termin employe
termin township employe requir vote board truste unclear roysdon racism ever
affect work caus anyon die absolut someth investig
Predicted Prob: 0.5541610717773438, Actual Label: 0

Text: florida gop offici tri murder teen claw hammer refus resign back may
republican broward counti elect rupert tarsey secretari gop affili littl know
actual elect rupert ditsworth former student la harvardwestlak school per year
preparatori academi california decad ago ditsworth attack fellow student hw
elizabeth barcay hit head least time claw hammer date jamba juic gone awri dump
car fleeingaft attempt murder ditsworth parent enlist psychiatrist tri suss son
may beaten girl within inch life word whether shrink use rupert still claim
defend went work trump presidenti campaign run elect broward gopso nonow florida
republican understand freak fort lauderdal area larg democrat parti secretari
split open skull high school girl tend cast organ dim lighttarsey rupert known
see problem sat exclus interview dailymailcom million beachfront condo resign
noth wrong elect parti polit tarsey insist name chang noth horrif attack
defenseless young woman deadli weapon estrang dad tarsey say perhap broken skull
nose leg girl suffer hand foot son explain dear old dad may creat distanc themth
chair broward gop bob sutton apoplect blindsid member knight columbu christ sake
came highli recommend former chairw idea background want refus resign deceiv us
look like even use reput manag firm make sure find realli shame republican
better figur member parti horribl monster perhap start racist sexist homophob
might get better spot guy claw hammer closet
Predicted Prob: 0.13779878616333008, Actual Label: 1

Text: victim terrorist attack question lack presidenti support minnesota
governor someth current occup white hous quickli denounc act terror makeshift
bomb tore mosqu earli saturday morn terribl dastardli cowardli terribl act
commit yesterday gov mark dayton said someon said meet role revers would call
terrorist attack act terror author bloomington said one injur blast citi dar
alfarooq islam centerif attack happen white evangel church morn prayer muslim
donald trump would use incid promot agenda ban anyon religion enter usaft lash
media democrat senat militari record trump still address mosqu attack unnerv
muslim communityresid state still wait trump condemn attack instead amateur
presid busi ragetweet critic silenc part public offici nation level serv empow
islamophob ibrahim hooper spokesman council americanislam relat said statement
call trump condemn attack accord lo angel time wonder presid trump tweet asad
zaman director muslim american societi minnesota told buzzfe news seem want
tweet secur terror issu new york time white hous correspond glenn thrush tweet
statement white hous monday trump still address bomb us soilemail wh minn mosqu
bomb presid continu updat monitor situat glenn thrush august word compass white
hous offer muslim commun terribl timeth fbi tweet statement updat statement
special agent charg explos dar al farooq commun center picmtjveguc fbi
minneapoli august fortun live lost physic damag repar fbi recogn pain anger
commun anytim place worship attack work hard hold respons account statement
readloc offici condemn attack america leader seem knowledg lead go go limb
suggest faith someth trump silenc mattera gofundm donat page rais two day time
```

```
help repair mosqu comment request donat read pleas feel welcom countri welcom
sorri happen white hous may care plenti us countri wait trump address issu least
pretend care want interrupt undeserv day vacat luxuri golf cours american
citizen attackedread morephoto ian macnicolgetti imag
Predicted Prob: 0.9106874465942383, Actual Label: 0

Text: gop senat set huge trap trump hell screw matter gop congress mostli fallen
right line behind dear orang leader sign veneer support begin crack thursday
congress sent donald trump near unanim bill test trump patriot allegi russiath
bill would increas sanction iran north korea russia would also prevent trump eas
current sanction russia senat berni sander rand paul vote bill sander said could
interfer iran nuclear agreement hous bill pass vote word even trump veto bill
congress easili overrid itund bill presid requir notifi congress make alter
russia sanction polici lawmak day block presid implement chang procedur known
congression review sweep author congress given check presid sanction polici
decadessuch matter tradit left execut branch congress author sanction administr
dispos even case mandatori sanction congress usual steer clear presid matter
nation securityif trump veto bill tacit admiss ye putin puppet signal russia
trump longer live bargain win onewhil white hous press secretari sarah huckabe
sander said know whether trump sign bill trump new commun director anthoni
scaramucci hint trump might veto bill negoti even tougher deal russian believ
trump univers diploma salethi interest trap show congress least trust trump
nation secur admiss guilt part republican parti know would publicli admit worri
futur countri
Predicted Prob: 0.4108050763607025, Actual Label: 1

Text: trump get wreck confirm secret cia program twitter tweet trump continu
attack fake news media monday night level yet anoth attack amazon washington
post stori publish regard one mani thing donald support russia appear confirm
exist secret cia programth stori question claim trump end secret cia program
train syrian rebel fight dictat bashar alassad accord post vladimir putin long
want see program shut reflect trump interest find way work russia saw antiassad
program assault interest amazon washington post fabric fact end massiv danger
wast payment syrian rebel fight assad trump tweetedth amazon washington post
fabric fact end massiv danger wast payment syrian rebel fight assad donald j
trump juli trump cours disput exist program fact surround decis end right back
attack fake news appar unawar reveal classifi informationso mani stori fake news
bad rate challeng lobbyist amazon tax donald j trump juli fake news washington
post use lobbyist weapon congress keep politician look amazon notax monopoli
donald j trump juli got topic politico report hada gold note fox news sean
hanniti discuss secret program show monday night discuss post stori juli tonight
program hada gold juli natur american littl bit concern alleg presid twitter
blab nation secret againisn tweet classifi program kelli pranghof juli misspel
directli support bashar al assad russia support jule suzdaltsev juli screw bone
spur trump ignorantli declassifi covert oper via twitter unfit serv april juli
serious fuck manag make freakin boy scout look like fascist today shut pleas
michaelmarshallsmith juli trump rant wapo stori us aid syria sound like putin
word key juli potu trump discuss covert cia oper threaten major us corpor due
```

```
ceo own newspap dislik picvfpumgb david rothschild juli disclos secret cia oper
endang valuabl hard find syrian asset week donovan rozier juli god realli stupid
enough reveal formerli covert oper twitter cat liter smarter buffoon carter
gaddi juli leak classifi info leaker traitor give away state secret j f juli
puti want puti get right hippiechick juli clear happen donald trump like media
outlet confirm exist classifi cia program vladimir putin want gonethi normal
Predicted Prob: 0.43466877937316895, Actual Label: 1

Text: congression black caucu jeff session got go congression black caucu
influenti voic hous repres republican hesit critic group sinc alreadi enough
problem public percept regard attitud issu race therefor pretti power thing
congression black caucu come major posit mondaythi group lawmak band togeth take
controversi stand call resign racist keebler elf like call america attorney
gener jeff session said statement attorney gener session unfit serv top law
enforc offici nation resign posit immedi old say goe fool shame fool twice shame
attorney gener session treat congress american peopl like fool caucu vote
whether publicli call session step even washington post reveal friday session
truth oath post stori alleg unit state intellig listen phone call held russian
ambassador sergey kislyak explicitli said convers presidenti campaign session
met repeatedli session forc deni oathth congression black caucu went say revel
everi day depart justic prosecut peopl lie oath yet man lead depart lie oath one
occas exactli forget resign jeff session need prosecut us done prison year man
definit kind law enforc offic much less top one disgrac constitut depart justic
hope congression black caucu influenc move other respond session grave
transgress lawbreak kind
Predicted Prob: 0.936808168888092, Actual Label: 0

Text: trump america white supremacist blogger given job feder judg controversi
judici nomine presid trump list confirm thursday us senat vote catastrophemeet
john bush lawyer kentucki outspoken polit blogger bash gay right believ roe v
wade decis equal deplor suprem court proslaveri decisionthi rightw nutjob newest
judg th circuit us court appealsjohn bush publish articl career rightw blogger
fake pen name regularli cite conspiraci theori fake news stori altright media
report includ ridicul stori presid obama born outsid usdur confirm hear bush
attempt downplay destruct altright view polit activ openli made pledg separ
person polit work judg courtroom benchth confirm hear final vote came without
republican john mccainthi fourth judici nomine trump win confirm twentytwo pend
nomin far judici vacanc feder bench usual trump activ seek fill slot presid look
judg conserv also young abl serv long time effort perman shift judiciari conserv
activist rate trump quickli eras judici nomin gain made obama administr entir
two termsfeatur imag photo mark wilsongetti imag
Predicted Prob: 0.00757093308493495, Actual Label: 1

Text: break gop chairman grassley enough demand trump jr testimoni donald trump
white hous chao tri cover russia problem mount hour refus acknowledg problem
surround fake news hoax howev fact bear thing differ seem crack congression
public leadershipchuck grassley riowa head senat judiciari committe fed demand
donald trump jr former trump campaign manag paul manafort testifi committe
```

```
regard infam shadi meet donald trump shadi russian lawyer promis dirt democrat
presidenti nomine hillari clinton fact inform due well demand send signal team
trump notabl fire special counsel robert mueller circumst despit fact seem seem
trump white hous lay groundwork speak speakher tweet regard grassley warningalso
anyon think senat grassley rest senat seriou need look warn alreadi given trump
jr manafort either follow order serv subpoena forc compli refus held contempt
congress carri seriou jail timeeven cruel craven creatur within gop sick donald
trump corrupt scandalridden white hous angri stage hostil takeov parti first
birther give perman racist label decim effort made pretend republican parti hotb
racism turn world upsid nation seem oldtim like grassley clearli sick trump
bullshit might one could save republ need bit courag
Predicted Prob: 0.67667156457901, Actual Label: 0

Text: trump panic deutsch bank plan turn financi inform investig one mani reason
american suspici donald trump begin variou busi conflict shadi financ obvious
becom major problem presidencywhil trump russia scandal erupt also kept busi
defend sever conflict interest struggl continu hide tax return reveal long
histori debt trail shadi busi deal trump want america know much money owe owe
money tonot deutsch bank one lender want work trump despit horribl reput su
lender go back contract bank tie russia loan trump hundr million dollar thank
trump russia investig investig go find whyaccord new york time bank regul review
hundr million dollar loan made mr trump busi deutsch bank privat wealth manag
unit see loan might expos bank heighten risk trump relationship deutsch bank
problemat decad go bite deutsch given trump billion last year return trump su
bank fell behind payment million load avoid pay bank trump blame global crisi
said bank pay instead leverag extraordinari event claus contract trump said
deutsch bank one bank primarili respons econom dysfunct current face respons
deutsch countersu classic trump move trump abl pay bank back got anoth loan
deutsch wealthmanag unitthi come shock american trump shadi famili also involv
ivanka trump deutsch bank client husband jare kushner also neck russian scandal
kushner got russian mess hide meet sergey gorkov happen chief execut russian
stateown develop bank vnesheconombank bank deutsch bank cooper agreement
withdeutsch bank far innoc exactli investig right along trump check may feder
prosecutor settl case cypru invest vehicl own russian businessman close famili
connect kremlin firm prevezon hold repres natalia veselnitskaya russian lawyer
among peopl met presidenti campaign donald trump jr hillari clinton feder
prosecutor unit state claim prevezon admit wrongdo launder proce alleg russian
tax fraud real estat prevezon partner reli part million financ big european
financi institut court record show deutsch banktrump want anyon investig financi
tie made clear interview time wednesday night trump clearli state want anyon
examin famili financ beyond relationship russia think violat thank shadi deal
deutsch bank trump nightmar come true
Predicted Prob: 0.15864278376102448, Actual Label: 1

Text: texa suprem court prove support marriag equal ever sinc histor rule made
samesex marriag legal state june oppon cri war realli seem like war samesex
marriag even courthous rather live let live peopl enjoy make other miserablewhat
defend uphold justic let samesex spous texa texa suprem court rule constitut
```

```
clearli requir state extend spousal benefit samesex coupl decis came friday rule
unanim make clear believ justic marriag equalityth texa suprem court interpret
rule obergefel v hodg narrowli question whether compel state treat samesex coupl
equal oppositesex coupl context outsid marriag licens guess expect anyth better
texasfriday rule pidgeon v turner revolv around spousal benefit govern worker
texa law prohibit samesex coupl receiv benefit even possibl beyond meslat report
accord court obergefel address resolv specif issu state spousal benefit therefor
state appeal court er order trial court resolv case consist obergefel de leon
instead texa suprem court insist trial court must settl issu keep mind obergefel
hold state must provid publicli fund benefit marri person flabbergast judg meant
serv justic deni justic interpret rule law way bend rule person level probabl
like samesex marriagemi heart goe samesex spous receiv fair equal spousal
benefit
Predicted Prob: 0.753642737865448, Actual Label: 0

Text: hillari break silenc gop health care bill brilliantli shut tweet senat
republican unveil disastr health care bill thursday immedi shot everyon
republican democrat former presid barack obama former democrat presidenti nomine
hillari clinton broken silenc weigh bill gop go hate iton friday clinton urg
support speak obamacar replac plan encourag choos peopl polit republican health
care plan strip ten million american lifesav health coverag messag sure import
oneclinton took twitter take messag public solidar word former presid barack
obama clinton slam gop bill right critic moment choos peopl polit speak bill
senat republican creat bill absolut secreci wonder health care bill monstros
assault live american thank gop propos cut would made medicaid feder fund plan
parenthood would prohibit made even conserv voic concernsyesterday obama call
senat pathet attempt replac health care bill massiv transfer wealth middleclass
poor famili richest peopl america also said health care bill clinton call gop
even vicious follow first tweet forget death panel republican pass bill death
parti could said better gop health care bill prove give sht live american measur
must stop track
Predicted Prob: 0.6762102842330933, Actual Label: 0

Text: sen cotton intern caught tape call brit ft declar paul ryan cuck audio hey
justin anoth titl told one goe far hope come someth worksrepublican senat tom
cotton intern think british faggot hous speaker paul ryan cuck appar afraid say
actual made repugn remark reporterth intern mediait refer first name nate want
keep name clean futur employ googl search insert angri eye roll record say paul
ryan cuck cuck get paul ryan cuck first yanke second case know cuck mediait
happi explain even refus fulli loudmouth internth term cuck origin polit space
white nationalist farright began call republican deem moder cuckserv word racial
charg cuck joan walsh describ pornograph genr white husband either shame lust
watch wife taken black man altright otherwis known white supremacist neonazi use
term refer ryan even name cuck year intern also told report militari record show
american superior race everyon world superior peopl continu rant said british
faggot deem benedict arnold homosexu nate big fan donald trump stupid wall
xenophobia behind bigot immigr parti told report say need lax immigr system let
peopl fuck bigot toward muslim guess damn nate addedn intern cotton six month
```

```
also weigh health care debat insist argument health care human right garbag
fundament wrong die street idiot believ social darwin idiot get fuck saidth news
nate rage bigot come surpris sen cotton assum vet even littl mediait report
facebook page full derogatori term faggot fag tranni made habit shout offens
rhetor hall congressmediait contact sen cotton offic comment certainli must
someth say right well much spokesperson said nate longer intern senat cotton
offic beyond comment personnel matter listen nate disgust remark
Predicted Prob: 0.061239346861839294, Actual Label: 1

Text: montana dem hilari troll reporterslam goper suggest go jail instead
congress rememb eve montana special elect thenrepublican candid greg gianfort
made headlin guardian ben jacob ask simpl question congression budget offic
score disastr gop healthcar bill instead answer like normal person even say
comment time gianfort suddenli becam enrag without caus bodi slam jacob break
glass injur arm gianfort later charg misdemeanor assault despit disturb set
event still racenow gianfort sworn congress montana democrat surpris wait first
day bit troll sent gianfort orang jumpsuit clearli suggest jail ben jacob head
washington fill seat congress offici montana democrat parti liber activist sens
humor post offici websit everyon know need plenti suit work capitol hill
millionair gianfort certainli afford know lot legal fee pay thought help get
start mail new suit offic longworth hous offic build washington orang jumpsuit
roy loewenstein spokesman montana democrat went say gianfort greg gianfort
previous wellknown lose governor race wave year wellknown plead guilti big sky
bodi slam convict crimin hidden leadership given posit influenc washington got
mr gianfort welcom gift help new colleagu identifi cours gop happi bit fun
montana democrat gianfort expens nation republican congression committe livid
way democrat humili gianfort say mouthpiec jack pandol swearingin ceremoni held
today montanan regain voic hous repres partisan polit instead montana democrat
cri spill milk simpli lost elect lose elect fact clearli violent man convict
assault sit unit state hous repres republican parti absolut problem alway open
arm violent crimin nobodi watch nonrespons rightw violenc horrif scene trump
ralli surpris gianfort right home paul ryan gop hous confer
Predicted Prob: 0.3040706217288971, Actual Label: 1

Text: depart justic say donald trump accept foreign money depart justic argu
presid take money foreign govern doj argu trump citizen respons ethic washington
file lawsuit trump state violat constitut accept foreign moneyaccord report
trump take money foreign govern hotel room fee golf club fee travel differ
countri affair countri howev crew citizen respons ethic washington still say
trump violat constitut accept money differ countri travel say stop took
officealthough crew file lawsuit trump administr trump administr think crew lack
legal stand even abl file lawsuit first placeit matter trump team think enough
legal stand crew alreadi file lawsuit trump team first week offic presidentin
fact crew say lawsuit file trump administr want get point hope presid trump
would take necessari step avoid violat constitut took offic howev constitut
violat immedi seriou forc take legal action sinc lawsuit crew file first week
offic sever plaintiff come onboard trump new plaintiff ad lawsuit trump
administr includ restaur associ restaur worker even woman book banquet hall
```

```
hotel washington dc
Predicted Prob: 0.5233359932899475, Actual Label: 0

Text: realli wise presid give person cellphon number us tend quit select give
phone number would instantli assum presid unit state would guard person cell
number great deal secreci well appar casein yet anoth break convent presid
donald trump hand person cellphon number world leader tell call directli rais
secur concern white hous even convers world leader still littl hypocrit presid
trump use person phone spent great deal time energi berat hillari clinton use
privat email server secretari state claim practic left vulner send confidenti
inform lead chant lock trump support presidenti campaignaccord former current
unnam us offici presid trump urg leader mexico canada call person number howev
canadian prime minist justin trudeau taken offer furthermor accord unnam french
offici trump also swap phone number emmanuel macron presid franc two spoke
immedi macron victori earlier mayalthough seem crazi think world leader might
hit cell leadertolead call gener follow standard protocol case us presid call
usual place one sever secur phone line includ white hous situat room oval offic
presidenti limousin accord nation secur expert trump use cellphon call put
extrem high risk listen particularli foreign govern speak open line open line
mean abil monitor convers said derek chollet former pentagon advis nation secur
council offici someon tri spi everyth say presum other listen perfect exampl
practic edward snowden leak us monitor german chancellor angela merkel cell back
despit germani american alli macron leader countri get cellphon number presid
unit state reason assum hand right intel servic said ashley deek law professor
univers virginia former assist legal advis politicalmilitari affair us state
departmentgiv person number new york citi real estat mogul obviouslyli common
practic trump probabl find anoth way make deal presid unit state
Predicted Prob: 0.023200305178761482, Actual Label: 1

Text: confirm trump tri everi corrupt trick book obstruct fbi russia investig
appear trump tri everyth could think discredit fbi investig whether campaign
collud russia prior nov elect jame comey testifi congress fbi inde look possibl
collus trump campaign russia trump began tri pressur intellig chief help push
back investigationthi monument develop mean trump activ work obstruct independ
investig tri enlist help member intellig commun daniel coat director nation
intellig time admir michael roger director nsa pressur trump administr deni
collus campaign russiathey refus felt request inappropri allegedli document memo
could provid congress doj special counsel evid trump tri interfer fbi worktrump
ask comey near end februari drop investig michael flynn russia later grew irat
comey told congress fbi investig fire comey may claim first doj recommend later
comey fire noth russia probehowev recent came told russian offici comey dismiss
took pressur believ would take pressur investig much us includ congress
flabbergast trump would make obviou move signal worri investig would turn
upbooki start put odd whether trump even abl finish term accord
fivethirtyeightcom betfair trump odd fail serv four year term percent put odd
offic end year percentnow say due impeach trump could well resign pressur
investig possibl impeach proceed nixon shoe keep drop trump revel add grow pile
evid activ tri obstruct justic interf investigationswhat hidesilli question
```

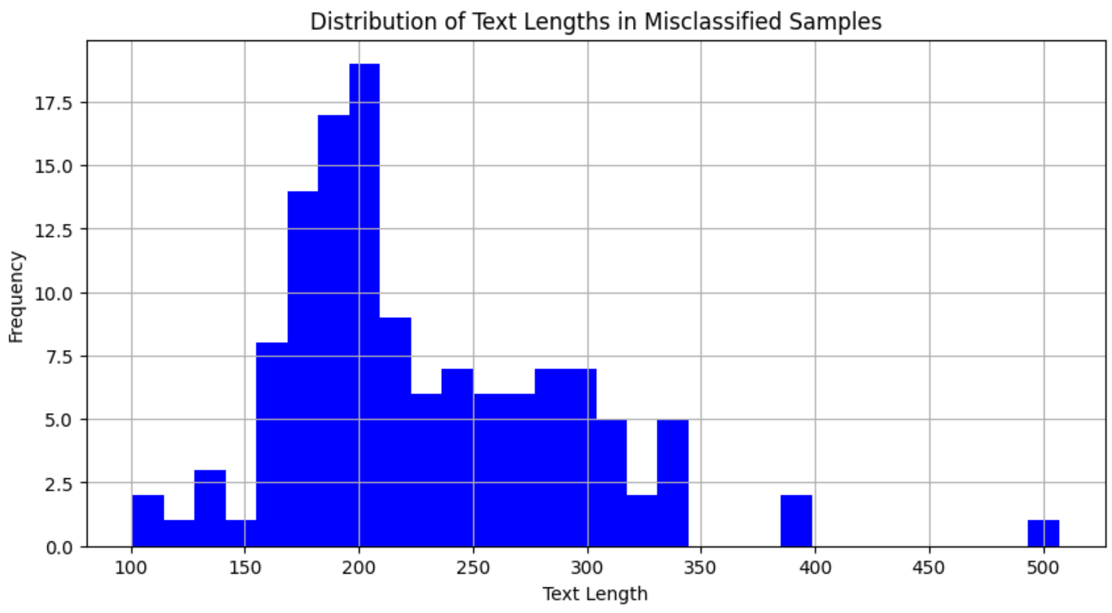
```
Predicted Prob: 9.02159299585037e-05, Actual Label: 1

Text: network join boycott new trump ad readi give thought anyth polit luck
campaign alreadi begun trumppenc team alreadi put first ad decri fake news caus
sever main news network say air new adcnn abc nbc cite graphic show word fake
news face number anchor rest ad devot list presid donald trump accomplish first
day white hous ad goe flame fake news cover everyth presid done sinc assum
officeal network run ad say inaccur abc news report ad person attack network
report target trump ad rachel maddow wolf blitzer andrea mitchel scott pelley
georg stephanopoulostrump daughter law weigh lara trump eric wife said appar
mainstream media champion first amend serv polit view work presid reelect
effortsth campaign make mislead ad howev also activ work donat court high level
donorsth ad also list problem campaign seen press coverag presid team exampl
sean hanniti press interview minut beef way edit piec anoth exampl fake news
coverag segment rachel maddow show copi presid tax returnth ad run fox news fox
busi network
Predicted Prob: 0.05944615975022316, Actual Label: 1

Text: watch sean spicer debut bold babbl everyth expect presid donald trump made
one thing clear care rate anyth els world week washington post report ask plan
fire sean spicer white hous press secretari trump said fire sean spicer guy get
great rate everyon tune presid went compar daili white hous brief soap opera
note spicer get better rate soap today trump love televis well known aid use way
reach foreign leader even start find way show show think presid like
watchstephen colbert took advantag comment connect spicer soap opera creat one
clip show preview spicer soap bold babbl would look like
Predicted Prob: 0.3106158375740051, Actual Label: 1

Text: china warn trump provok north korea pay price thursday donald trump tweet
china north korea sole major alli deal north korea properli unit state alli
china neither alli enemi unit state major power around nuclear warhead largest
armi world fact mani alli would consid back us war might involv trump would well
interest avert world war iii trump say love war call parti refrain provok
threaten whether word action let situat get irrevers unmanag stage chines
foreign minist wang yi told report beij hope appeal trump better natur forget
one war occur result situat everybodi lose winner yi ad friday joint press
confer french counterpart late tension risen us south korea one side north korea
one feel conflict could break moment yi said ad side provok conflict trump case
correctli taxpay repres must assum histor respons pay correspond price world
worri trump satisfi massiv ego especi drop mother bomb recent seem nonpluss
spend weekend golf instead worri crisi help exacerbatewil trump risk world war
gigant hit unit state treasuri think know answer soon done golf
Predicted Prob: 0.2900969386100769, Actual Label: 1

Text: break comey shut trump tweet lie real time hear video monday fbi director
jame comey nsa director mike roger sat gruel hourslong hear donald trump busili
thumb way oblivion twitter tweet got factcheck real timeth donald leap opportun
speak friend vladimir putin individu explain make assess whether russian
```

```
interfer influenc elector process nsa fbi tell congress russia influenc elector
process picdhqkxybt presid trump march unfortun trump claim two member intellig
commun confirm russia influenc elect lie got fact check littl later hear ask
tweet accur comey explain offer opinion view inform potenti impact never someth
look cours fbi nsa posit begin chang shame donald trump would attempt misrepres
testimoni way watch
Predicted Prob: 0.955234706401825, Actual Label: 0
```

[ ]:
```python
from collections import Counter

misclassified_words = []
for text in misclassified_texts:
    misclassified_words.extend(text.split())

word_freq = Counter(misclassified_words)
print(word_freq.most_common(20))
```

```
[('trump', 816), ('peopl', 183), ('donald', 178), ('presid', 164), ('would',
154), ('one', 149), ('say', 148), ('like', 146), ('said', 146), ('go', 137),
('republican', 136), ('state', 130), ('even', 121), ('make', 111), ('get', 107),
('time', 105), ('know', 103), ('want', 99), ('right', 98), ('us', 93)]
```

[ ]:
```python
misclassified_lengths = [len(text.split()) for text in misclassified_texts]
plt.figure(figsize=(10, 5))
plt.hist(misclassified_lengths, bins=30, color='blue')
plt.title('Distribution of Text Lengths in Misclassified Samples')
plt.xlabel('Text Length')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



```
[ ]: word_text_count = {}

     for text in misclassified_texts:
         unique_words = set(text.split())
         for word in unique_words:
             if word in word_text_count:
                 word_text_count[word] += 1
             else:
                 word_text_count[word] = 1

     # sort the words based on the number of texts they appear in
     sorted_word_text_count = sorted(word_text_count.items(), key=lambda item:␣
      ↪item[1], reverse=True)

     # the top 20 words that appear in the most texts
     print("Top 20 words that appear in the most misclassified texts:")
     for word, count in sorted_word_text_count[:20]:
         print(f"{word}: {count}")
```

```
Top 20 words that appear in the most misclassified texts:
trump: 99
donald: 85
one: 82
say: 76
said: 75
peopl: 74
go: 74
```

```
like: 71
make: 71
would: 70
time: 68
even: 67
presid: 64
state: 62
get: 61
right: 60
us: 58
think: 57
want: 56
new: 56
```