National College of Ireland

Config Manual

MSc Research Project
Masters of Science in Data Analytics

Muhammad Hunzla Naqi Khan
Student ID: 23375060

School of Computing
National College of Ireland

Supervisor: Prof. Christian Horn

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Muhammad Hunzla Naqi Khan
Student ID:	23375060
Programme:	Masters of Science in Data Analytics **Year:** 2025-26
Module:	Research Practicum
Supervisor:	Prof. Christian Horn
Submission Due Date:	11-12-2025
Project Title:	A Comparative Study of Machine Learning Algorithms for Classifying Real and Fake COVID-19 Tweets

Word Count: 609 **Page Count** 06

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Muhammad Hunzla Naqi Khan

Date: 10-12-2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Config Manual

Muhammad Hunzla Naqi Khan
23375060

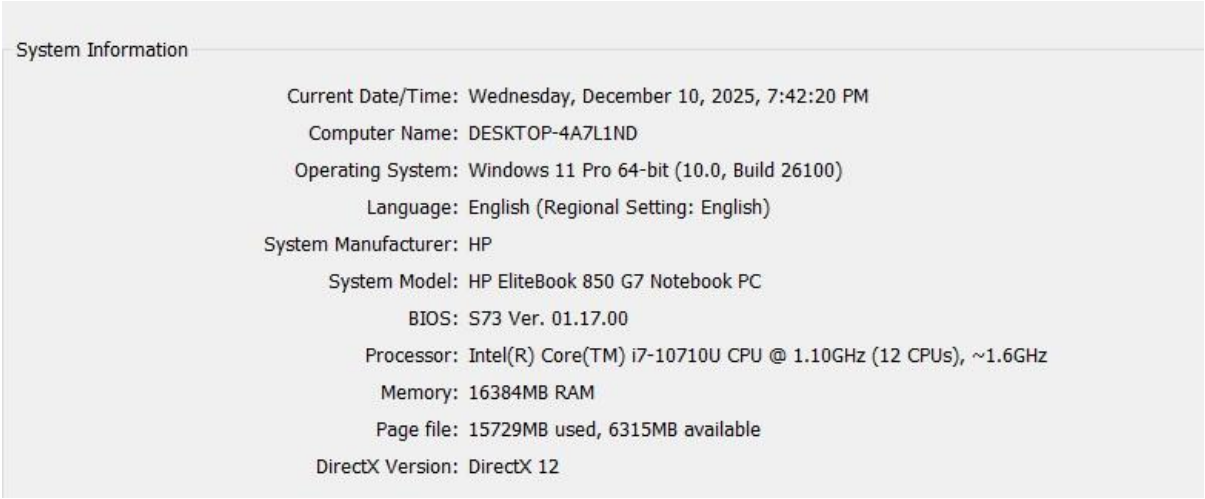
1 Introduction

This documentation explains how to prepare the environment, pre-process the dataset and run the machine learning pipeline for classification of COVID-19 tweets as real or fake based on the COVID Fake News Dataset. All code is executed in a Jupyter Notebook with in an Anaconda environment.

2 System Requirement

2.1 Hardware Information

Component	Specification
System Manufacturer	HP
System Model	HP EliteBook 850 G7 Notebook PC
Processor	Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz (12 CPUs) ~1.6GHz
RAM	16GB
Operating System	Windows 11 Pro 64-bit (Version 10.0, Build 26100)
Language	English
DirectX Version	DirectX 12
BIOS	S73 Ver. 01.17.00



2.2 Software Information

- **Python Version**: 3.12.7 (packaged by Anaconda, Inc., Oct 4 2024, 13:17:27)
- **Operating System**: Windows 11 Pro 64-bit
- **Software Environment**:
    - **Jupyter Notebook** for interactive analysis
    - **Anaconda** for environment and package management

3 Environment Setup and Libraries Used

This project relies on the following environment setup and Python libraries & packages for data preprocessing, machine learning, and visualization:

- pandas
- numpy
- re
- nltk
- sklearn.model_selection
- sklearn.neighbors
- sklearn.linear_model
- sklearn.svm
- sklearn.naive_bayes
- xgboost
- sklearn.tree
- sklearn.ensemble
- sklearn.metrics
- sklearn.feature_extraction.text
- scipy.sparse
- gensim.models
- matplotlib.pyplot
- seaborn
- wordcloud

Install all required libraries and packages.

[1]:
```
import pandas as pd
import numpy as np
import re, string
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import SnowballStemmer
from nltk.corpus import wordnet
from nltk.stem import WordNetLemmatizer
#for model-building
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from xgboost import XGBClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report, f1_score, accuracy_score, confusion_matrix
# bag of words
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from scipy.sparse import hstack
#for word embedding
import gensim
from gensim.models import Word2Vec
import os
import csv
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
from sklearn import metrics
import pandas as pd
from pandas import read_excel
from wordcloud import WordCloud
```

4 Dataset Setup

Dataset Source: [COVID Fake News Dataset at Kaggle]

Download dataset, and keep it on your local drive as CSV file. Example file: dataset/dataset. csv

Make sure the file has the following columns: ID, Tweet, Label.

Copy the dataset into your project folder and use a dataset/directory.

5 Preprocessing Steps

There are several phases of preprocessing the dataset to obtain clean and structured data that is friendly towards machine learning. Below are the key steps:

5.1 Load Dataset

Now let's import the dataset into a **Pandas Dataframe** for easier handling.

```
[2]: dataframe=pd.read_csv('dataset/dataset.csv')
```

```
[3]: dataframe.head(10)
```

[3]:

	id	tweet	label
0	1	The CDC currently reports 99031 deaths. In gen...	real
1	2	States reported 1121 deaths a small rise from ...	real
2	3	Politically Correct Woman (Almost) Uses Pandem...	fake
3	4	#IndiaFightsCorona: We have 1524 #COVID testin...	real
4	5	Populous states can generate large case counts...	real
5	6	Covid Act Now found "on average each person in...	real
6	7	If you tested positive for #COVID19 and have n...	real
7	8	Obama Calls Trump's Coronavirus Response A Cha...	fake
8	9	???Clearly, the Obama administration did not l...	fake
9	10	Retraction—Hydroxychloroquine or chloroquine w...	fake

```
[4]: dataframe.describe()
```

[4]:

	id
count	6420.000000
mean	3210.500000
std	1853.438696
min	1.000000
25%	1605.750000
50%	3210.500000
75%	4815.250000
max	6420.000000

```
[5]: dataframe['label'].value_counts()
```

```
[5]: label
real    3360
fake    3060
Name: count, dtype: int64
```

5.2 Data Cleaning

The steps for cleaning the dataset is as follows:

Address missing values: Impute or remove depending on the percentage of missing observation.

Removal of unwanted attributes: Here we remove unnecessary characters such as punctuation, special characters and numbers including URL from the tweet text.

5.3 Text Processing

After cleaning the dataset, apply these text processing actions:

Lowercase: Convert to lowercase letters.

Tokenize: Split the text into tokens with NLTK, or a similar library.

Remove stopwords: Remove the stop words like "the", "is", "and" who won't give a much sense here.

```
processed_features = []

for sentence in range(0, len(features)):
    # Remove URLs
    processed_feature = re.sub(r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+', '', str(features[sentence]))

    # Remove 'http' and 'https' tags
    processed_feature = re.sub(r'http|https', '', processed_feature)

    # Remove numbers
    processed_feature = re.sub(r'\d+', '', processed_feature)

    # Remove all the special characters
    processed_feature = re.sub(r'\W', ' ', processed_feature)

    # Remove all single characters
    processed_feature = re.sub(r'\s+[a-zA-Z]\s+', ' ', processed_feature)

    # Remove single characters from the start
    processed_feature = re.sub(r'\^[a-zA-Z]\s+', ' ', processed_feature)

    # Substituting multiple spaces with a single space
    processed_feature = re.sub(r'\s+', ' ', processed_feature, flags=re.I)

    # Removing prefixed 'b'
    processed_feature = re.sub(r'^b\s+', '', processed_feature)

    processed_features.append(processed_feature)
```

5.4 TF-IDF Vectorization

The tweets text is being converted into numerical value through the use of TF-IDF (Term Frequency-Inverse Document Frequency). This reflects the relevance of words in the tweets compared to all documents.

5.5 Word-Level and Character-Level TF-IDF

The other word-level and character-level TF-IDF features are also generated to encode patterns in the tweets.

[20]:
```
# Word-level TF-IDF
tfidf_word = TfidfVectorizer(max_features=15000, ngram_range=(1,2), min_df=3, max_df=0.85)
X_word = tfidf_word.fit_transform(processed_features)

# Character-level TF-IDF
tfidf_char = TfidfVectorizer(analyzer='char', ngram_range=(3,5), min_df=3)
X_char = tfidf_char.fit_transform(processed_features)

# Combine features
processed_features = hstack([X_word, X_char])
```

6 Model Training & Evaluation

In our work, we evaluate ten supervised machine learning algorithms effectiveness in classifying COVID-19-related tweets as either real or fake. The following algorithms were used:

1. Logistic Regression
2. Decision Tree Classifier
3. Random Forest Classifier
4. Gradient Boosting Classifier
5. AdaBoost Classifier
6. Passive Aggressive Classifier
7. Multinomial Naïve Bayes
8. K-Nearest Neighbors (KNN)
9. XGBoost Classifier
10. Linear Support Vector Machine

Every algorithm was trained with the TF-IDF features of clean tweet text. The models were also assessed with various metrics such as accuracy, precision, recall and F1-score. The assessment also involved conducting confusion matrix analysis to further investigate how the classifications performed.
Below are the performances for all the algorithms used:

Logistic Regression

```
[55]: model =LogisticRegression()

[56]: model.fit(X_train, y_train)

[56]: LogisticRegression()

[57]: predictions3 = model.predict(X_test)

[58]: print(confusion_matrix(y_test,predictions3))
      print(classification_report(y_test,predictions3))
      print(accuracy_score(y_test, predictions3))
```

```
[[578  49]
 [ 64 593]]
              precision    recall  f1-score   support

           0       0.90      0.92      0.91       627
           1       0.92      0.90      0.91       657

    accuracy                           0.91      1284
   macro avg       0.91      0.91      0.91      1284
weighted avg       0.91      0.91      0.91      1284

0.911993769470405
```

Decision Tree

```
[61]: model = DecisionTreeClassifier()
      model.fit(X_train, y_train)

      y_pred = model.predict(X_test)

[62]: print(confusion_matrix(y_test,y_pred))
      print(classification_report(y_test,y_pred))
      print(accuracy_score(y_test, y_pred))
```

```
[[500 127]
 [118 539]]
              precision    recall  f1-score   support

           0       0.81      0.80      0.80       627
           1       0.81      0.82      0.81       657

    accuracy                           0.81      1284
   macro avg       0.81      0.81      0.81      1284
weighted avg       0.81      0.81      0.81      1284

0.809190031152648
```

Random Forest

```
[30]: text_classifier = RandomForestClassifier(n_estimators=200, random_state=0)
      text_classifier.fit(X_train, y_train)

[30]: RandomForestClassifier(n_estimators=200, random_state=0)

[31]: predictions = text_classifier.predict(X_test)

[32]: predictions

[32]: array([0, 1, 0, ..., 1, 0, 1], dtype=int64)

[33]: print(confusion_matrix(y_test,predictions))
      print(classification_report(y_test,predictions))
      print(accuracy_score(y_test, predictions))
```

```
[[546  81]
 [ 76 581]]
              precision    recall  f1-score   support

           0       0.88      0.87      0.87       627
           1       0.88      0.88      0.88       657

    accuracy                           0.88      1284
   macro avg       0.88      0.88      0.88      1284
weighted avg       0.88      0.88      0.88      1284

0.8777258566978193
```

Gradient Boosting Classifier

```
[73]: gb_model = GradientBoostingClassifier()
      gb_model.fit(X_train, y_train)
      y_pred_gb = gb_model.predict(X_test)

[74]: print(confusion_matrix(y_test,y_pred_gb))
      print(classification_report(y_test,y_pred_gb))
      print(accuracy_score(y_test, y_pred_gb))
```

```
[[547  80]
 [ 84 573]]
              precision    recall  f1-score   support

           0       0.87      0.87      0.87       627
           1       0.88      0.87      0.87       657

    accuracy                           0.87      1284
   macro avg       0.87      0.87      0.87      1284
weighted avg       0.87      0.87      0.87      1284

0.8722741433021807
```

AdaBoost Classifier

```
[122]: ada_model = AdaBoostClassifier(n_estimators=100, random_state=42, algorithm='SAMME')
       ada_model.fit(X_train, y_train)
       y_pred_ada = ada_model.predict(X_test)

[123]: print(confusion_matrix(y_test,y_pred_ada))
       print(classification_report(y_test,y_pred_ada))
       print(accuracy_score(y_test, y_pred_ada))
```

```
[[523 104]
 [ 96 561]]
              precision    recall  f1-score   support

           0       0.84      0.83      0.84       627
           1       0.84      0.85      0.85       657

    accuracy                           0.84      1284
   macro avg       0.84      0.84      0.84      1284
weighted avg       0.84      0.84      0.84      1284

0.8442367601246106
```

Passive Aggressive Classifier

```
[81]: pa_model = PassiveAggressiveClassifier(max_iter=1000, random_state=42)
      pa_model.fit(X_train, y_train)
      y_pred_pa = pa_model.predict(X_test)

[82]: print(confusion_matrix(y_test,y_pred_pa))
      print(classification_report(y_test,y_pred_pa))
      print(accuracy_score(y_test, y_pred_pa))
```

```
[[574  53]
 [ 51 606]]
              precision    recall  f1-score   support

           0       0.92      0.92      0.92       627
           1       0.92      0.92      0.92       657

    accuracy                           0.92      1284
   macro avg       0.92      0.92      0.92      1284
weighted avg       0.92      0.92      0.92      1284

0.9190031152647975
```

Multinomial Naive Bayes

```
[65]: nb_model = MultinomialNB()
      nb_model.fit(X_train, y_train)
      y_pred_nb = nb_model.predict(X_test)

[66]: print(confusion_matrix(y_test,y_pred_nb))
      print(classification_report(y_test,y_pred_nb))
      print(accuracy_score(y_test, y_pred_nb))
```

```
[[537  90]
 [ 52 605]]
              precision    recall  f1-score   support

           0       0.91      0.86      0.88       627
           1       0.87      0.92      0.89       657

    accuracy                           0.89      1284
   macro avg       0.89      0.89      0.89      1284
weighted avg       0.89      0.89      0.89      1284

0.8894080996884736
```

KNN

```
[50]: text_classifier2 = KNeighborsClassifier(n_neighbors = 5)#no of neighbors is hpyer parameter
      text_classifier2.fit(X_train, y_train)

[50]: KNeighborsClassifier()

[51]: predictions2 = text_classifier2.predict(X_test)

[52]: print(confusion_matrix(y_test,predictions2))
      print(classification_report(y_test,predictions2))
      print(accuracy_score(y_test, predictions2))
```

```
[[591  36]
 [ 94 563]]
              precision    recall  f1-score   support

           0       0.86      0.94      0.90       627
           1       0.94      0.86      0.90       657

    accuracy                           0.90      1284
   macro avg       0.90      0.90      0.90      1284
weighted avg       0.90      0.90      0.90      1284

0.8987538940809969
```

XGBoost Classifier

```
[128]: xgb_model = XGBClassifier(eval_metric='logloss')
xgb_model.fit(X_train, y_train)
y_pred_xgb = xgb_model.predict(X_test)
```

```
[129]: print(confusion_matrix(y_test,y_pred_xgb))
print(classification_report(y_test,y_pred_xgb))
print(accuracy_score(y_test, y_pred_xgb))
```

```
[[563  64]
 [ 81 576]]
              precision    recall  f1-score   support

           0       0.87      0.90      0.89       627
           1       0.90      0.88      0.89       657

    accuracy                           0.89      1284
   macro avg       0.89      0.89      0.89      1284
weighted avg       0.89      0.89      0.89      1284

0.8870716510903427
```

Linear SVM

```
[85]: svm_clf = LinearSVC(class_weight='balanced')
svm_clf.fit(X_train, y_train)
ypredsvm = svm_clf.predict(X_test)
```

```
[86]: print(confusion_matrix(y_test, ypredsvm))
print(classification_report(y_test, ypredsvm))
print(accuracy_score(y_test, ypredsvm))
```

```
[[585  42]
 [ 57 600]]
              precision    recall  f1-score   support

           0       0.91      0.93      0.92       627
           1       0.93      0.91      0.92       657

    accuracy                           0.92      1284
   macro avg       0.92      0.92      0.92      1284
weighted avg       0.92      0.92      0.92      1284

0.9228971962616822
```

7 Troubleshooting

Memory error when TF-IDF vectorizing: Decrease TFIDF_MAX_FEATURES, or try on a machine with higher RAM.
Missing dependencies: Both Anaconda environment must be activated and dependent libraries must be installed.
Wrong path: Make sure DATA_PATH is the right CSV.

8 Conclusion

This config manual discusses how to setup the environment for preprocessing data, and deploying machine learning models in the context of detecting COVID-19 tweets. A manual provides with a reproducible environment to find the best algorithm for such a classification task.