Configuration Manual

MSc Data Analytics
Research Project

Msc. in Data Analytics
National College of Ireland

Supervisor: Mr. Noel Cosgrave

National College of Ireland

MSc Project Submission Sheet

Msc. Data analytics

Student Name:

Student ID:

Programme: MSc Data Analytics **Year:** 2024-2025

Module: Research Project

Supervisor: Mr. Noel Cosgrave

Submission Due Date: 12/12/2024

Project Title: **INTEGRATING SENTIMENT ANALYSIS AND FINANCIAL METRICS TO UNDERSTAND CONSUMER BEHAVIOR IN THE AUTOMOTIVE INDUSTRY**

Word Count: 786 **Page Count:** 11

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: Dale lobo

Date: 12/12/2024

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):	

Configuration Manual

Introduction:

This manual provides detailed instructions for configuring and deploying the phishing URL detection system developed in this research project. The system employs a hybrid model integrating machine learning and deep learning techniques to accurately identify phishing URLs.

1 System Requirements:

To guarantee efficient model processing and to minimize the duration required, it's crucial to be equipped with the necessary hardware and software resources.

1.1. Hardware Requirements:

The implementation is performed on an HP Pavilion; the configuration of the device is as follows.

1. Processor: AMD Ryzen 7 3700X, 3.0 GHz
2. RAM: 16.00 GB
3. Hard Disk: 1 TB HDD for data storage, 512 GB SSD
4. OS Windows 11 (64-bit)

1.2 Software Requirements:

Before beginning the model construction phase, the below mentioned software, libraries, and tools were set up and installed on the system.

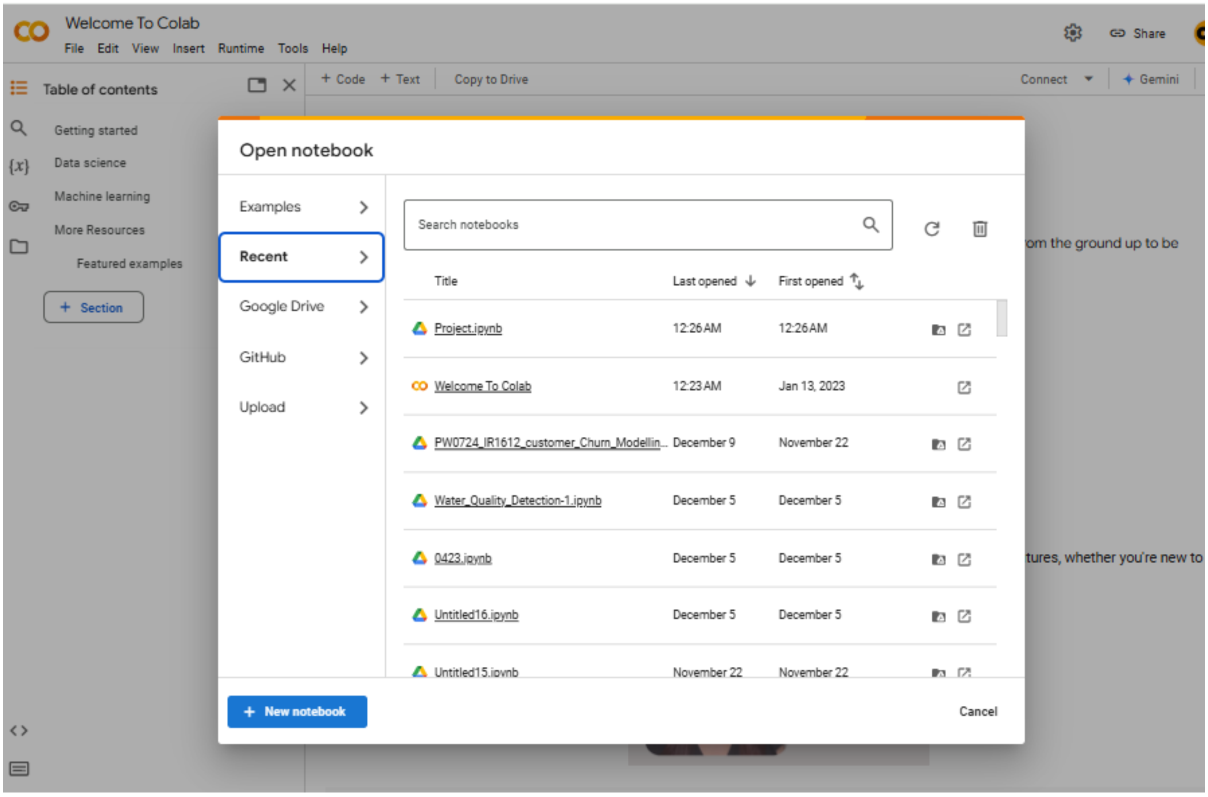
Software/Tools	Version	Information
Python		Python is a widely-used, high-level programming language known for its simplicity and versatility. It is commonly used for data analysis, machine learning, artificial intelligence, and web development. Python supports various libraries for scientific computing and data processing, making it ideal for tasks like sentiment analysis and financial modeling.
Google Colab		**Google Colab** (short for Colaboratory) is a cloud-based platform provided by Google that allows you to write and execute Python code in a Jupyter notebook

		environment.
Pandas		Pandas is a powerful Python library used for data manipulation and analysis. It provides two main data structures, DataFrame and Series, which are used for handling and analyzing structured data. It supports operations like data cleaning, transformation, and aggregation, making it an essential tool for working with large datasets.
Transformers		Transformers is a Python library by Hugging Face that provides pre-trained deep learning models, particularly for Natural Language Processing (NLP) tasks. It includes models like BERT, GPT, and T5, which can be used for tasks such as sentiment analysis, text classification, and language generation. It simplifies working with state-of-the-art models and integrates easily with frameworks like PyTorch and TensorFlow.
Sci-kit Learn		Scikit-learn is a machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It offers a wide range of algorithms for classification, regression, clustering, and dimensionality reduction. Scikit-learn is widely used for building machine learning models due to its easy-to-understand API and robust documentation.

2. Implementation:

In this section there is a complete guide to run the project in any windows system.

1. Opening a web browser and going to Google Colab.



1. After opening jupyter notebook click on the File, New Notebook or Open Notebook.
2. In notebook, Import all the required libraries.

```
# Sentiment Analysis
from nrclex import NRCLex
```

```
import nltk
```

```
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import CountVectorizer
```

```
nltk.download('punkt_tab')
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, Conv1D, GlobalMaxPooling1D, LSTM, Dense, Dropout, Co
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
```

3. Import the Provided Dataset.

```
import pandas as pd

df = pd.read_csv("/content/car_5_brands.csv")
```

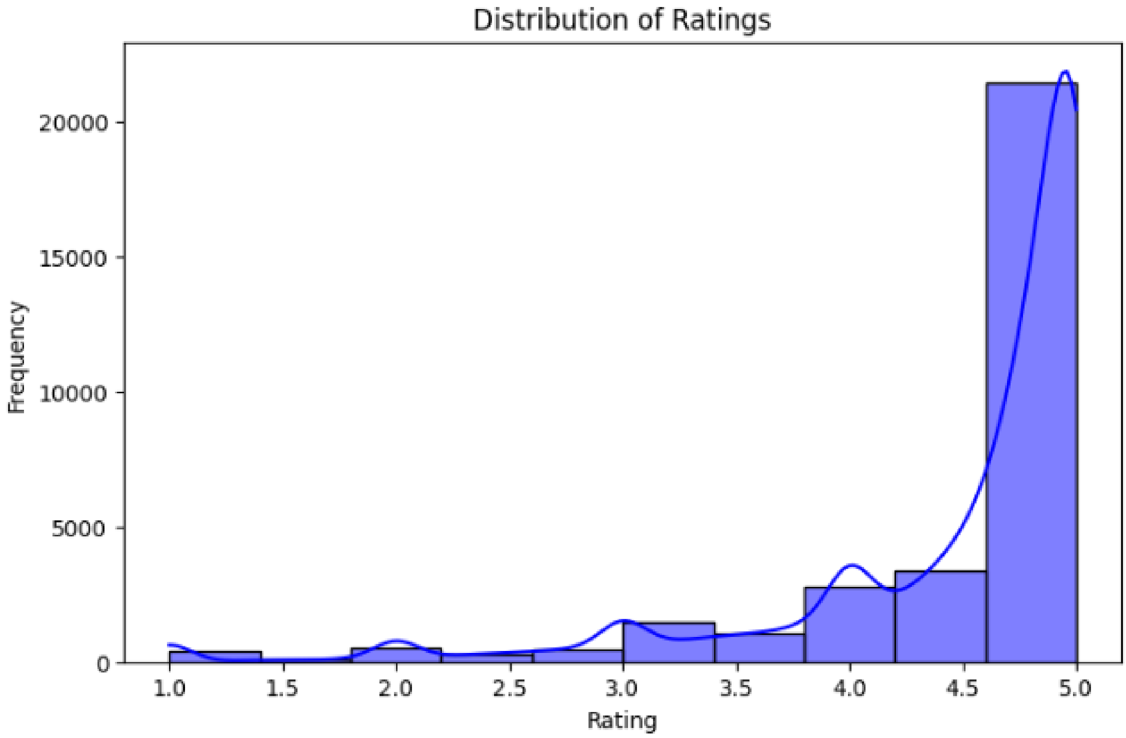
6. Next Step will be Pre Processing Step will be performed using following Code.

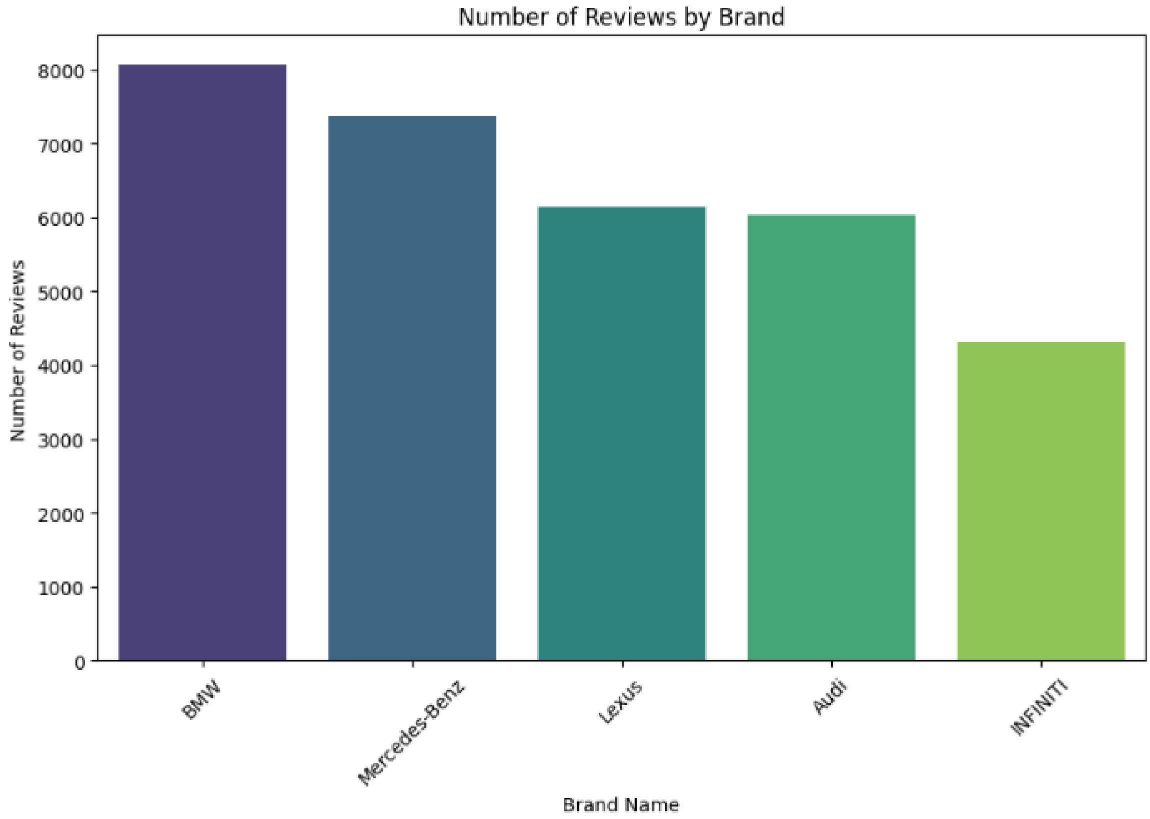
```
# Drop redundant columns
df = df.drop(columns=["Unnamed: 0"], errors='ignore')

# Handle missing values
# Check for missing values
missing_values = df.isnull().sum()
print("Missing values:\n", missing_values)
```

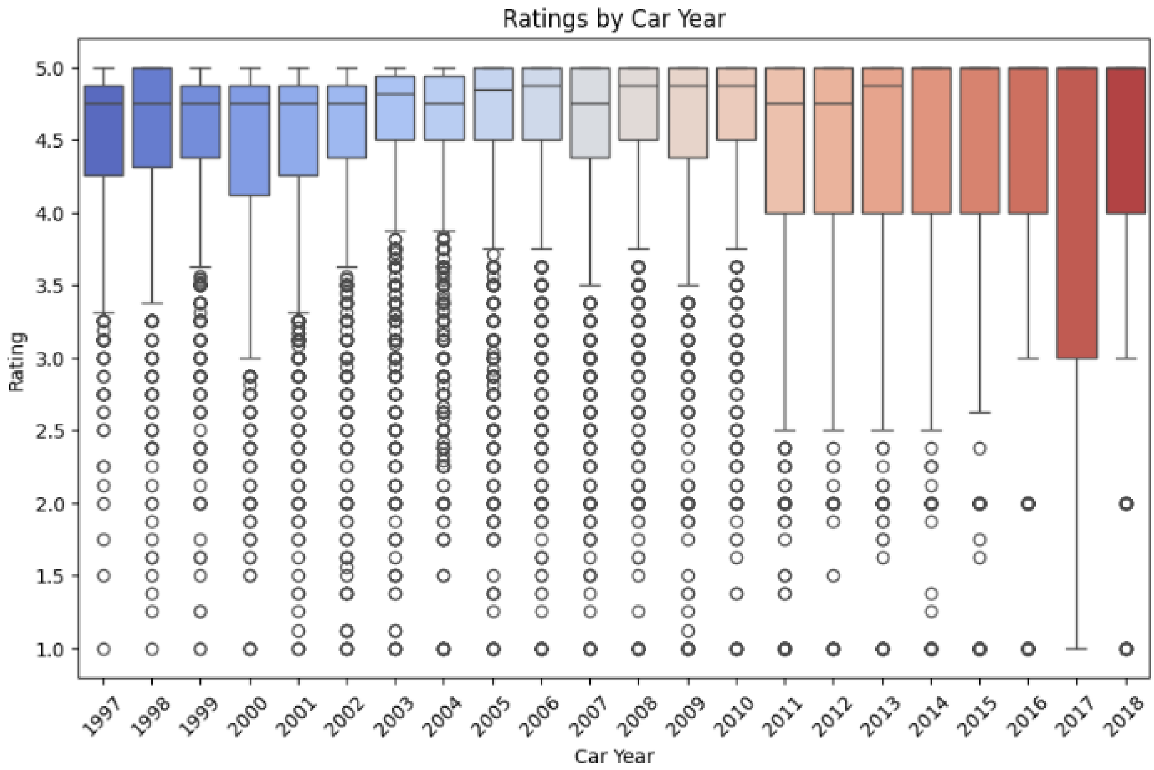
```
Missing values:
 Rating        0
car_year      0
brand_name    0
date          0
review        0
dtype: int64
```

```
Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31938 entries, 0 to 31937
Data columns (total 8 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Rating            31938 non-null  float64
 1   car_year          31938 non-null  int64
 2   brand_name        31938 non-null  object
 3   date              31938 non-null  object
 4   review            31938 non-null  object
 5   cleaned_review    31938 non-null  object
 6   emotions          31938 non-null  object
 7   dominant_emotion  31938 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 1.9+ MB
None

First 5 Rows:
   Rating  car_year brand_name        date  \
0     5.0      2018       Audi  2018-07-11
1     5.0      2018       Audi  2018-06-24
2     5.0      2018       Audi  2018-05-02
3     5.0      2018       Audi  2017-12-07
4     5.0      2018       Audi  2017-10-25
```

4. Exploratory Data Analysis has been Performed and Visualisation has been done using following Code







5. After Data Pre Processing the Data Splitting is Performed before Building a Model

```
# Train-test split
# Split into train and test first
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Now verify the shapes again
print("Shape of X_train:", X_train.shape)
print("Shape of y_train:", y_train.shape)
```

```
Shape of X_train: (25550, 6)
Shape of y_train: (25550,)
```

6. Models Implementation has been Performed with the following Code

```
from keras.models import Model
from keras.layers import Input, Embedding, LSTM, Dense, Dropout, Concatenate, Flatten
from keras.optimizers import Adam
```

```
# Define the text input layer
text_input = Input(shape=(50,), name='text_input')  # 50 tokens max length for padded sequences
embedding_layer = Embedding(input_dim=5000, output_dim=128, input_length=50)(text_input)  # Embedding laye
lstm_layer = LSTM(64)(embedding_layer)  # LSTM layer to capture sequential patterns

# Define the numerical input layer
numerical_input = Input(shape=(X_train_numerical.shape[1],), name='numerical_input')
numerical_dense = Dense(64, activation='relu')(numerical_input)

# Define the categorical input layer
categorical_input = Input(shape=(X_train_categorical.shape[1],), name='categorical_input')
categorical_dense = Dense(64, activation='relu')(categorical_input)

# Concatenate all the input layers
merged = Concatenate()([lstm_layer, numerical_dense, categorical_dense])

# Add some dense layers for further processing
merged_dense = Dense(128, activation='relu')(merged)
dropout_layer = Dropout(0.5)(merged_dense)
output_layer = Dense(1)(dropout_layer)  # Output layer for regression (since you're predicting a rating)

# Define the model
model = Model(inputs=[text_input, numerical_input, categorical_input], outputs=output_layer)

# Compile the model
model.compile(optimizer=Adam(), loss='mean_squared_error', metrics=['mae'])

# Summarize the model architecture
model.summary()
```

```
from keras.models import Model
from keras.layers import Input, Embedding, Conv1D, MaxPooling1D, GlobalMaxPooling1D, Dense, Dropout, Conca
from keras.optimizers import Adam

# Define the text input layer
text_input = Input(shape=(50,), name='text_input')  # 50 tokens max length for padded sequences
embedding_layer = Embedding(input_dim=5000, output_dim=128, input_length=50)(text_input)  # Embedding Laye

# Convolutional Layer for Text Feature Extraction
conv_layer = Conv1D(128, 5, activation='relu')(embedding_layer)  # 1D convolution with 128 filters and ker
max_pool_layer = MaxPooling1D(pool_size=2)(conv_layer)  # Max pooling with pool size 2
global_pooling_layer = GlobalMaxPooling1D()(max_pool_layer)  # Global max pooling to get the best feature

# Define the numerical input layer
numerical_input = Input(shape=(X_train_numerical.shape[1],), name='numerical_input')
numerical_dense = Dense(64, activation='relu')(numerical_input)

# Define the categorical input layer
categorical_input = Input(shape=(X_train_categorical.shape[1],), name='categorical_input')
categorical_dense = Dense(64, activation='relu')(categorical_input)

# Concatenate all the input layers
merged = Concatenate()([global_pooling_layer, numerical_dense, categorical_dense])

# Add some dense layers for further processing
merged_dense = Dense(128, activation='relu')(merged)
dropout_layer = Dropout(0.5)(merged_dense)
output_layer = Dense(1)(dropout_layer)  # Output layer for regression (since you're predicting a rating)

# Define the model
model_cnn = Model(inputs=[text_input, numerical_input, categorical_input], outputs=output_layer)

# Compile the model
model_cnn.compile(optimizer=Adam(), loss='mean_squared_error', metrics=['mae'])

# Summarize the model architecture
model_cnn.summary()
```

```
# Train the CNN model
history = model_cnn.fit(
    [X_train_text, X_train_numerical, X_train_categorical],  # Input data
    y_train,  # Target labels (ratings)
    epochs=10,  # Number of epochs
    batch_size=32,  # Batch size
    validation_split=0.1,  # Validation split (10% of the training data for validation)
    verbose=1  # Show training progress
)

# Evaluate the model on the test data
loss, mae = model_cnn.evaluate(
    [X_test_text, X_test_numerical, X_test_categorical],  # Test data
    y_test  # Test target labels
)

print(f"Test Mean Absolute Error (MAE): {mae:.2f}")
```

```
Epoch 1/10
```

7. The Accuracy is considered as evaluation factor after Model Implementation

```
200/200 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step
LSTM Model Evaluation Metrics:
Mean Absolute Error (MAE): 0.3959
Mean Squared Error (MSE): 0.4108
Root Mean Squared Error (RMSE): 0.6410
R-squared (R²): 0.3242
Explained Variance Score: 0.3263
```

8.

```
200/200 ━━━━━━━━━━━━━━━━━━━━ 3s 14ms/step
CNN Model Evaluation Metrics:
Mean Absolute Error (MAE): 0.4685
Mean Squared Error (MSE): 0.4149
Root Mean Squared Error (RMSE): 0.6441
R-squared (R²): 0.3175
Explained Variance Score: 0.3323
```

```
Linear Regression Metrics:
R-squared: 0.3307
Mean Squared Error (MSE): 0.3770
Root Mean Squared Error (RMSE): 0.6140

Random Forest Regressor Metrics:
R-squared: 0.3768
Mean Squared Error (MSE): 0.3510
Root Mean Squared Error (RMSE): 0.5925
```

The concluding code files are include the ipynb file and the csv dataset

References

Numpy.org. 2021. NumPy. [online] Available at: <https://numpy.org/>.