Configuration Manual

MSc Research Project

Cloud Computing

Student Name: Basanti Pun

Student ID: 21130931

School of Computing

National College of Ireland

Supervisor: Dr. Shivani Jaswal

National College of Ireland

MSc Project Submission Sheet
School of Computing

Student Name:	Basanti Pun		
Student ID:	21130931		
Programme:	MSc in Cloud Computing	**Year:**	2023
Module:	Research Project		
Supervisor:	Dr. Shivani Jaswal		
Submission Due Date:	25/04/2023		
Project Title:	Sentiment Analysis of Ireland Housing Problem using Ensemble Learning and XAI Techniques		
Word Count:	1361		
Page Count	15		

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:	Basanti Pun
Date:	25/04/2023

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

Basanti Pun

21130931

1 Introduction

The setup guide elucidates the process of executing the developed code relevant to the current study. By adhering to these instructions, the smooth and error-free operation of the code is guaranteed. Additionally, it encompasses details concerning the hardware requirements for the system on which the code will be executed, including the recommended minimum specifications. Adhering to these guidelines will facilitate the duplication of the project's results, enabling further analysis and seamless continuation of research.

2 System Configuration

2.1 Hardware and Software Configuration

As the project is implemented on AWS by using the service Amazon SageMaker there is no specific requirement for the local system. The main required tool is a browser and reliable internet connection.

For creating the instance of Jupyter notebook on Amazon SageMaker follow the following steps:

1. Login to AWS console

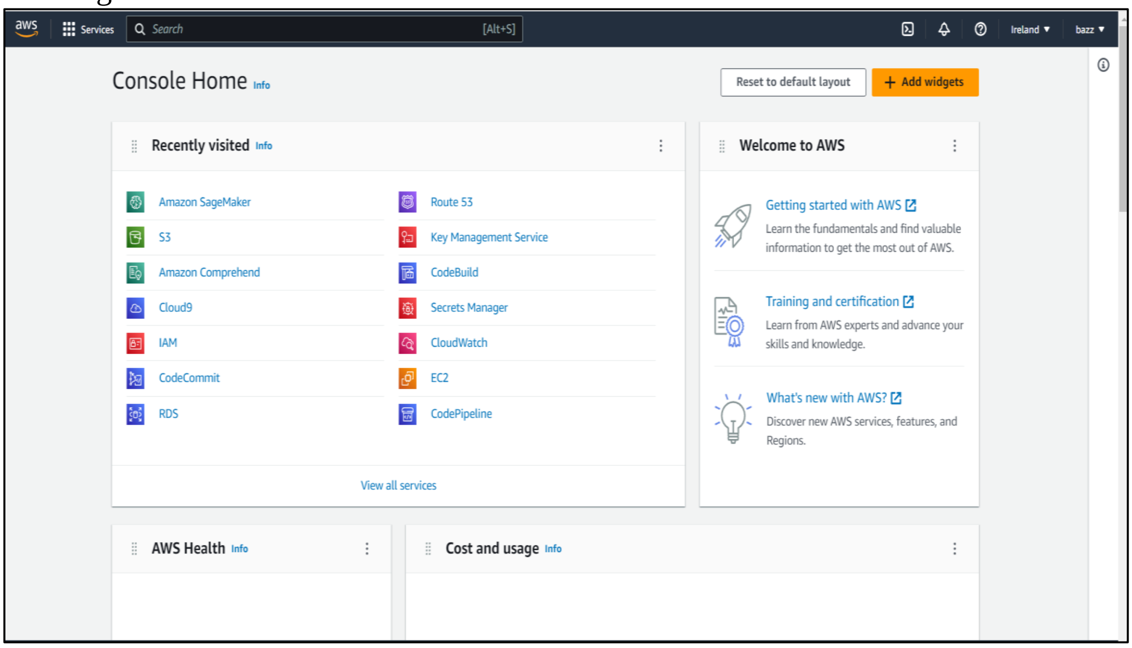

Figure 1

2. From service section select Amazon SageMaker.

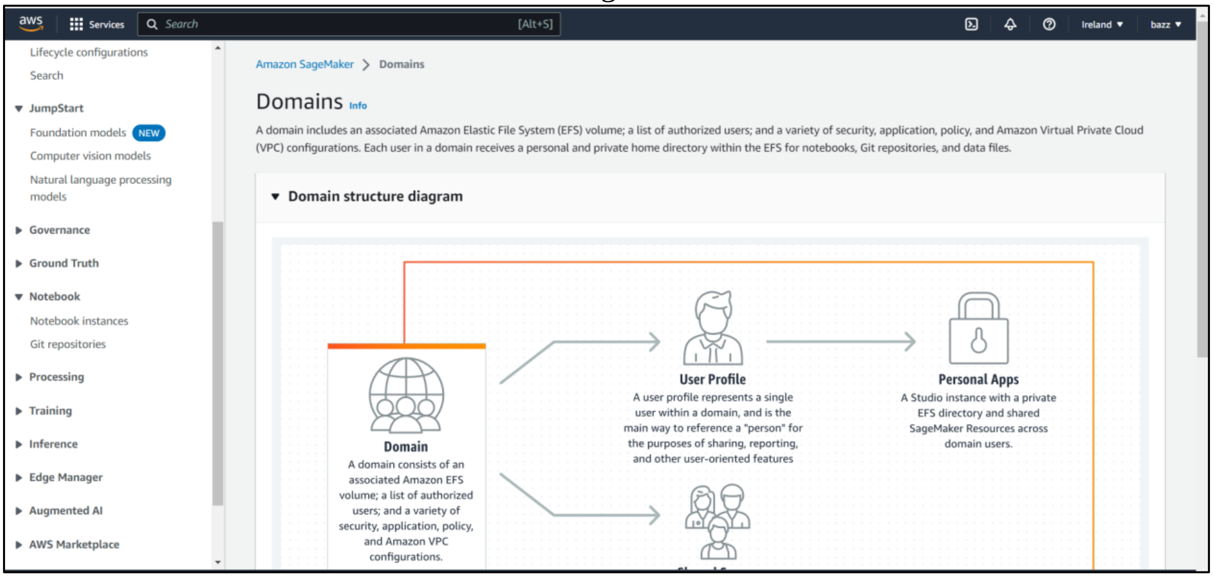

Figure 2

3. From the left panel click on Notebook -> Notebook Instances -> create notebook instance.

Figure 3

4. Enter the instance name and then choose the instance type "ml.t3.medium", platform identifier -> Amazon Linux 2, Jupyter Lab 3. Other configurations are optional and can be done as per the requirement.

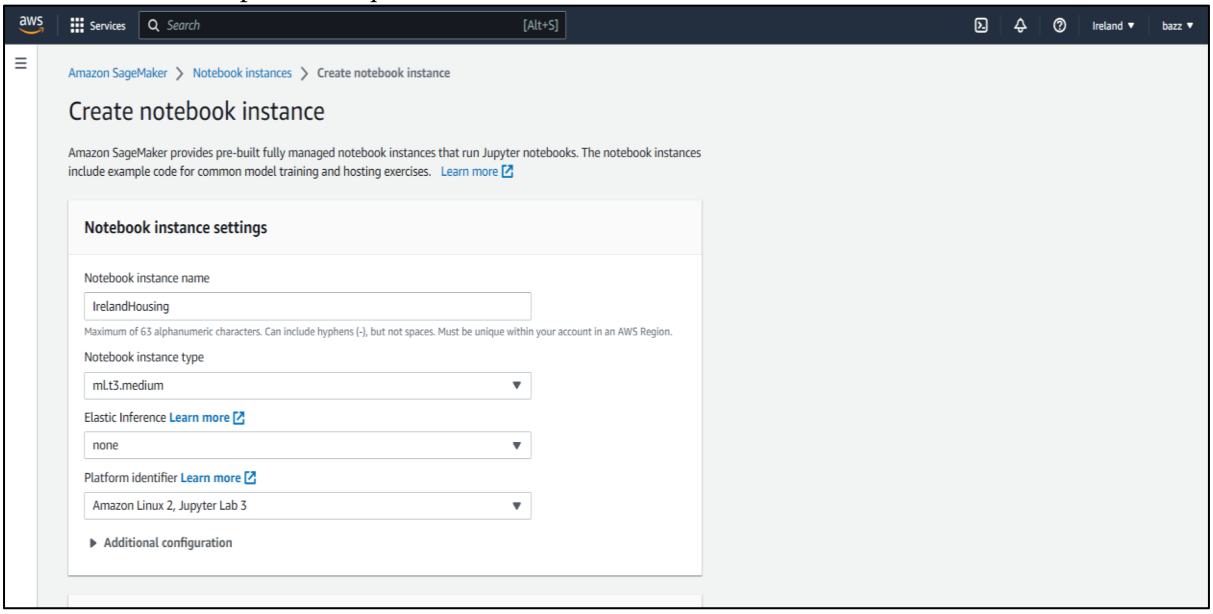

Figure 4

5. Click on the create notebook instance.

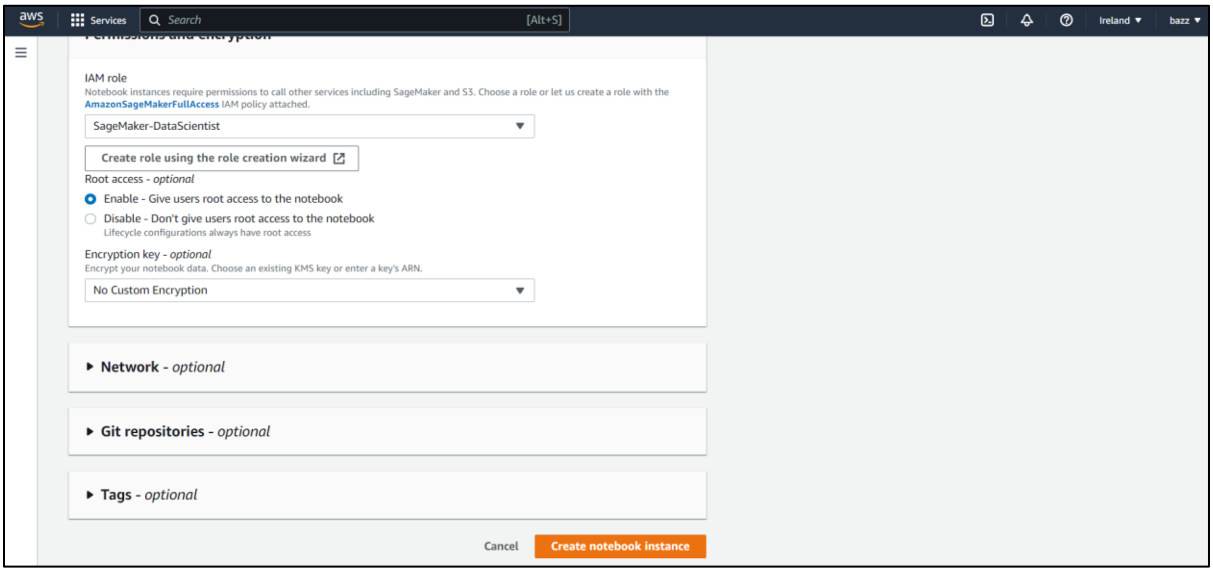

Figure 5

6. After creating the notebook instance, it will come in ready state.

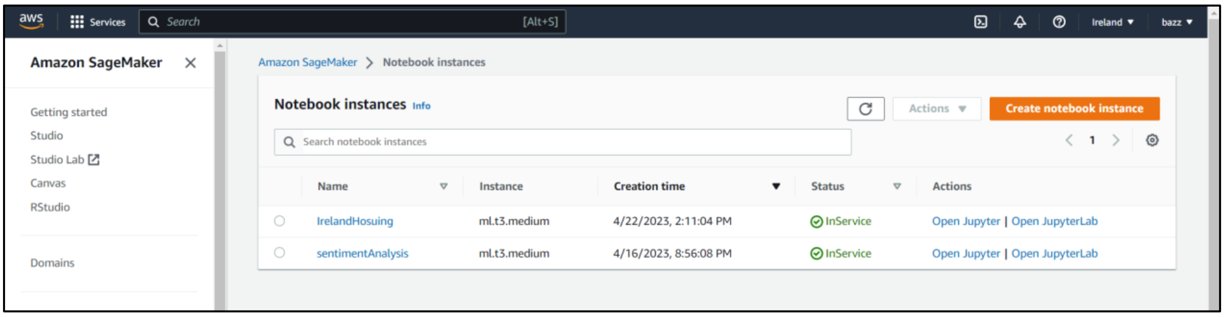

Figure 6

7. Click on the open Jupyter button. Following interface will appear

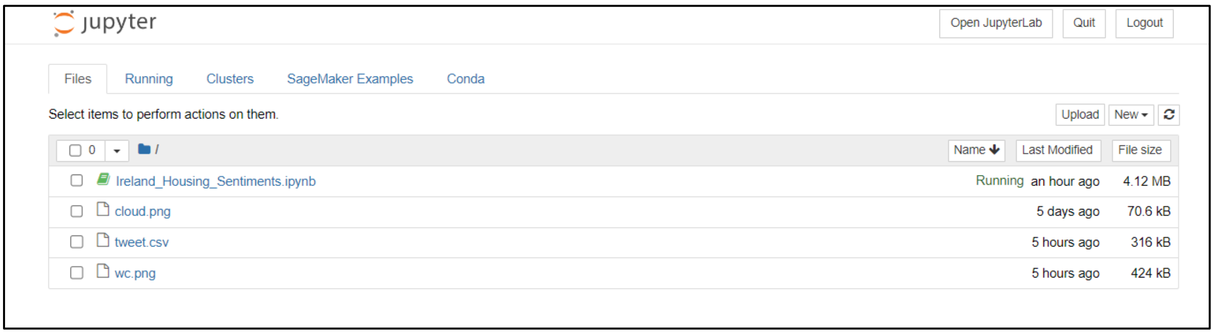

Figure 7

Click on the new button and choose the conda_python3 environment to start writing the python code.

3 Project Development

For the implementation of the Machine Learning model some libraries are required to be installed which can be done by using pip command. In the Jupyter notebook, the library can be installed by using this command “!!pip install <library name>”.

Following libraries are needed to be install in Jupyter notebook

- **matplotlib** - Matplotlib is a data visualization library and is commonly used for data analysis. It provides a variety of tools for creating static, animated, and interactive visualizations in Python.
- **seaborn –** Seaborn is a well-known Python library for data visualization that builds upon Matplotlib and offers a user-friendly interface for generating visually appealing and informative statistical graphics.
- **numPy –** NumPy is a software library for Python that is utilized in scientific calculations and the analysis of data. It provides a robust object for N-dimensional arrays, as well as an assortment of functions to manipulate these arrays.
- **pandas -** Pandas is a library in Python that is designed for the purpose of analysing and manipulating data. It offers a robust collection of tools that are specifically meant to handle structured data like time series data, series, and data frames.
- **nltk** - NLTK, which stands for Natural Language Toolkit, is a Python library that is utilized in the field of natural language processing. It presents a collection of techniques and resources for examining textual data, encompassing tokenization, stemming, identifying the grammatical category of words, recognizing named entities, gauging the sentiment of text, and additional functionalities.
- **textblob** - TextBlob enables the analysis of a text's sentiment, which can reveal whether the text conveys positive, negative, or neutral emotions. The resulting sentiment score is a floating-point number ranging between -1 and 1, where a score of -1 indicates very negative sentiment, and a score of 1 indicates very positive sentiment.
- **pycountry** - The pycountry library is a software module written in Python that offers a user-friendly approach to retrieve the ISO databases associated with various categories such as countries, subdivisions, languages, currencies, and scripts.
- **langdetect** - The langdetect is a Python library that utilizes n-gram frequency analysis and machine learning methods to detect the language of a particular text in an automated manner.
- **tweepy** - Tweepy is a library for Python that offers a user-friendly approach to connect with and engage with the Twitter API. It streamlines the authentication process and presents a range of classes and methods to help access different features of the Twitter API.
- **wordcloud** - The wordcloud library is a Python package that is commonly utilized to generate word clouds. These clouds are graphic depictions of text information, wherein more commonly appearing words are depicted in bigger font sizes, while less frequently occurring words are displayed in smaller font sizes.
- **lime** - LIME (Local Interpretable Model-Agnostic Explanations) is a library in Python used for explaining the predictions of machine learning models. It helps in understanding the reasoning behind the predictions made by a model, especially for complex models like deep neural networks. LIME is designed to work with any machine learning model, whether it is a classification or a regression model.

After installing the libraries, the code can be run block by block.

3.1 Data Extraction and Pre-Processing

```
import tweepy
import pandas as pd

# function to display data of each tweet
def printtweetdata(n, ith_tweet):
        print()
        print(f"Tweet {n}:")
        print(f"Tweet Text:{ith_tweet[0]}")
        print(f"Location:{ith_tweet[1]}")
        print(f"Following Count:{ith_tweet[2]}")
        print(f"Follower Count:{ith_tweet[3]}")
        print(f"Total Tweets:{ith_tweet[5]}")


# function to perform data extraction
def scrape(words, date_since, numtweet):

        # Creating DataFrame using pandas
        df = pd.DataFrame(columns=['text',
                                   'location',
                                   'following',
                                   'followers',
                                    'totaltweets'])

        # We are using .Cursor() to search
        # through twitter for the required tweets.
        # The number of tweets can be
        # restricted using .items(number of tweets)
        tweets = tweepy.Cursor(api.search_tweets,
                               words, lang="en",
                               since_id=date_since,
                               tweet_mode='extended').items(numtweet)


        # .Cursor() returns an iterable object. Each item in
        # the iterator has various attributes
        # that you can access to
        # get information about each tweet
        list_tweets = [tweet for tweet in tweets]

        # Counter to maintain Tweet Count
        i = 1

        # we will iterate over each tweet in the
        # list for extracting information about each tweet
        for tweet in list_tweets:
                text = tweet.user.text
                location = tweet.user.location
                following = tweet.user.friends_count
                followers = tweet.user.followers_count
                hashtags = tweet.entities['hashtags']
```

```
                # Retweets can be distinguished by
                # a retweeted_status attribute,
                # in case it is an invalid reference,
                # except block will be executed
                try:
                        text = tweet.retweeted_status.full_text
                except AttributeError:
                        text = tweet.full_text
                hashtext = list()
                for j in range(0, len(hashtags)):
                        hashtext.append(hashtags[j]['text'])

                # Here we are appending all the
                # extracted information in the DataFrame
                ith_tweet = [text, location, following,
                             followers, totaltweets]
                db.loc[len(db)] = ith_tweet

                # Function call to print tweet data on screen
                printtweetdata(i, ith_tweet)
                i = i+1
        filename = 'twitter.csv'

        # we will save our database as a CSV file.
        db.to_csv(filename)
```

```python
if __name__ == '__main__':

    # Enter your own credentials obtained
    # from your developer account
    consumer_key = "g2JxHTaaiM8AS4fK3uDwlXc9w"
    consumer_secret = "hnEUuhcxBFrBEnGnam0oq8D8kPZqZoPsvX5rd82JIYpa3O08g8"
    access_key = "2243994841-WrjmErWEdfNZXBsBDDwBqR1SNe8fpWt65xq7X8e"
    access_secret = "fsmeNbAakCNYqRDKiUmlJPupW5TGsArhCeEqjmjldVhm2"

    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    api = tweepy.API(auth)

    # Enter Hashtags and initial date
    print("Enter Twitter HashTags separated by commas to search for")
    hashtags_input = input()
    hashtags_list = [hashtag.strip() for hashtag in hashtags_input.split(',')]
    words = ' OR '.join([f'#{hashtag}' for hashtag in hashtags_list])

    print("Enter Date since The Tweets are required in yyyy-mm-dd")
    date_since = input()

    # number of tweets you want to extract in one run
    numtweet = 10000
    scrape(words, date_since, numtweet)
    print('Scraping has completed!')
```

Twitter data is extracted using the twitter API and the below hashtags are used to retrieve relevant data from twitter.

```
Enter Twitter HashTags separated by commas to search for
#IrelandHousingCrisis, #HousingShortage, #AffordableHousingIreland, #RentCrisisIreland, #HomelessnessIreland, #IrishPropertyMar
ket, #HousingForAll, #RisingRentIreland, #IrishHousingIssues, #HousingPolicyIreland, #HousingProtest, #HousingFirstIreland, #Ho
usingSupplyIreland, #HousingBubbleIreland, #IrelandIsFull, #HouseTheIrish
Enter Date since The Tweets are required in yyyy-mm-dd

2009-01-01
```

Dataframe head below shows the data on the top 5 tweets -

```python
In [13]: import pandas as pd
         df = pd.read_csv('tweet.csv',encoding='MacRoman')
         df.head()
```

Out[13]:

	Unnamed: 0	location	text	hashtags	following	followers	totaltweets
0	0	Ireland	@SligoLeitLabour @labour An alternative to Sin...	['HousingForAll']	1010	1806	7522
1	1	Dublin, Ireland	The @LDA_Ireland has identified the potential...	['HousingForAll']	2518	210	29360
2	2	Dublin, Ireland	@KitMurray @paulmurphy_TD Have you an issue wi...	['housingforall']	1286	1254	8804
3	3	Limerick, Ireland	Want to give your vacant property a new lease ...	['oldhousenewhome', 'housingforall', 'Limerick']	183	282	14910
4	4	West of Ireland	Calling on all political figures to stop evect...	['evections', 'HousingCrisis', 'housingforall']	706	300	1021

Use of Lambda function to remove special characters, ‘RT@’ and mentions from the dataset

```python
df["text"] =tweet_list
#tweet_list.drop_duplicates(inplace = True)
tw_list = pd.DataFrame(tweet_list)
tw_list['text'] = tw_list[0]
#Removing RT, Punctuation etc

#Removing RT, Punctuation etc
remove_rt = lambda x: re.sub('RT @\w+: '," ",x)
rt = lambda x: re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)"," ",x)
tw_list["text"] = tw_list.text.map(remove_rt).map(rt)
tw_list["text"] = tw_list.text.str.lower()
tw_list.head(20)
```

	0	text
0	The situation with affordable rent and housing...	the situation with affordable rent and housing...
1	Dublin's Housing market continue to burn red h...	dublin s housing market continue to burn red h...
2	@KitMurray @paulmurphy_TD Have you an issue wi...	td have you an issue with peaceful protest...
3	Want to give your vacant property a new lease ...	want to give your vacant property a new lease ...
4	Calling on all political figures to stop evect...	calling on all political figures to stop evect...
5	A first-of-its-kind report on the potential of...	a first of its kind report on the potential of...

3.2 Data Transformation

```
#Determining positive,negative and neutral tweets

def percentage(part,whole):
    return 100 * float(part)/float(whole)
positive = 0
negative = 0
neutral = 0
polarity = 0
tweet_list = []
neutral_list = []
negative_list = []
positive_list = []
for tweet in df["text"]:
    tweet_list.append(tweet)
    analysis = TextBlob(tweet)
    score = SentimentIntensityAnalyzer().polarity_scores(tweet)
    neg = score['neg']
    neu = score['neu']
    pos = score['pos']
    comp = score['compound']
    polarity += analysis.sentiment.polarity

    if neg > pos:
        negative_list.append(tweet)
        negative += 1
    elif pos > neg:
        positive_list.append(tweet)
        positive += 1

    elif pos == neg:
        neutral_list.append(tweet)
        neutral += 1
positive = percentage(positive, 327)
negative = percentage(negative, 327)
neutral = percentage(neutral, 327)
polarity = percentage(polarity, 327)
positive = format(positive, '.1f')
negative = format(negative, '.1f')
neutral = format(neutral, '.1f')
```

```
tw_list[['polarity', 'subjectivity']] = tw_list['text'].apply(lambda Text: pd.Series(TextBlob(Text).sentiment))
for index, row in tw_list['text'].iteritems():
    score = SentimentIntensityAnalyzer().polarity_scores(row)
    neg = score['neg']
    neu = score['neu']
    pos = score['pos']
    comp = score['compound']
    if neg > pos:
        tw_list.loc[index, 'sentiment'] = "negative"
    elif pos > neg:
        tw_list.loc[index, 'sentiment'] = "positive"
    else:
        tw_list.loc[index, 'sentiment'] = "neutral"
    tw_list.loc[index, 'neg'] = neg
    tw_list.loc[index, 'neu'] = neu
    tw_list.loc[index, 'pos'] = pos
    tw_list.loc[index, 'compound'] = comp

tw_list.head(10)
```

	0	text	polarity	subjectivity	sentiment	neg	neu	pos	compound
0	The situation with affordable rent and housing...	the situation with affordable rent and housing...	-0.333333	0.678571	positive	0.000	0.977	0.023	0.0258
1	Dublin's Housing market continue to burn red h...	dublin s housing market continue to burn red h...	0.262500	0.362500	positive	0.000	0.936	0.064	0.2263
2	@KitMurray @paulmurphy_TD Have you an issue wi...	td have you an issue with peaceful protest...	0.250000	0.500000	positive	0.092	0.645	0.263	0.5719
3	Want to give your vacant property a new lease ...	want to give your vacant property a new lease ...	0.259091	0.338636	positive	0.000	0.877	0.123	0.4215
4	Calling on all political figures to stop evect...	calling on all political figures to stop evect...	0.000000	0.100000	negative	0.167	0.833	0.000	-0.2960

Pie chart for the twitter sentiment:

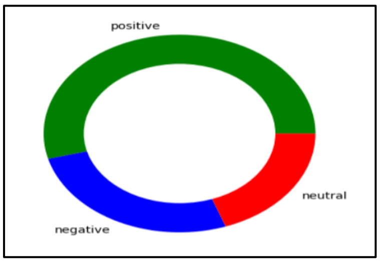

Figure 8

Count values for sentiment:

```
#Count_values for sentiment
count_values_in_column(tw_list,"sentiment")
```

	Total	Percentage
positive	650	54.12
negative	317	26.39
neutral	234	19.48

Figure 9

WordCloud:

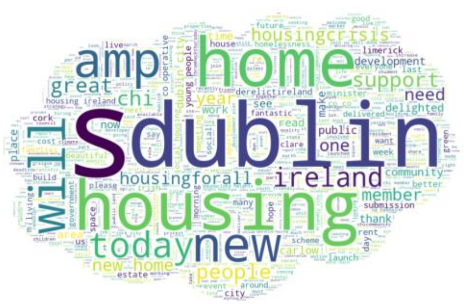

Figure 10

Word Cloud for Positive Sentiment

Figure 11

Word Cloud for Negative Sentiment

Top 5 locations in Ireland with the most negative sentiment tweets -

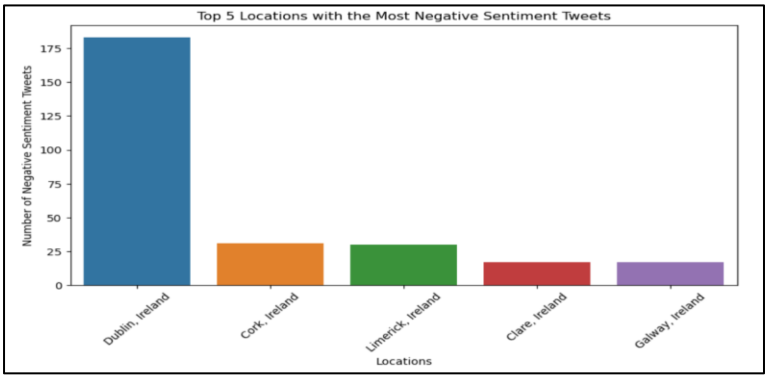

Figure 12

3.3 Data Mining

Logistic Regression

```python
from sklearn.linear_model import LogisticRegression

# Load the dataset
data = tw_list

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['sentiment'], test_size=0.3, random_state=42)

# Create feature vectors using CountVectorizer and TfidfTransformer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

# Train the model using logistic regression
clf = LogisticRegression(random_state=42)
clf.fit(X_train_tfidf, y_train)

# Test the model on the test set
X_test_counts = count_vect.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
y_pred = clf.predict(X_test_tfidf)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.6398891966759003
```

```
Classification Report:
              precision    recall  f1-score   support

    negative       0.79      0.37      0.51        99
     neutral       0.73      0.16      0.26        69
    positive       0.61      0.95      0.74       193

    accuracy                           0.64       361
   macro avg       0.71      0.49      0.50       361
weighted avg       0.68      0.64      0.59       361
```

Decision Tree

```python
from sklearn.tree import DecisionTreeClassifier

# Load data from CSV file
data = tw_list

# Split the dataset into train and test sets
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['sentiment'], test_size=0.3, random_state=42)

# Create feature vectors using CountVectorizer and TfidfTransformer
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(X_train)
tfidf_transformer = TfidfTransformer()
X_train_tfidf = tfidf_transformer.fit_transform(X_train_counts)

# Train a Decision Tree classifier on the training data
clf = DecisionTreeClassifier()
clf.fit(X_train_tfidf, y_train)

# Test the classifier on the test data
X_test_counts = count_vect.transform(X_test)
X_test_tfidf = tfidf_transformer.transform(X_test_counts)
y_pred = clf.predict(X_test_tfidf)

# Evaluate the performance of the classifier
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

```
Accuracy: 0.631578947368421
```

```
Classification Report:
              precision    recall  f1-score   support

    negative       0.57      0.56      0.56        99
     neutral       0.48      0.57      0.52        69
    positive       0.69      0.65      0.67       193

    accuracy                           0.61       361
   macro avg       0.58      0.59      0.58       361
weighted avg       0.62      0.61      0.61       361
```

Ensemble Model

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score, classification_report
import lime
from lime.lime_text import LimeTextExplainer
import shap

X = tw_list['text']
y = tw_list['sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

logistic_regression = LogisticRegression(random_state=42)
decision_tree = DecisionTreeClassifier(random_state=42)
support_vector_machine = SVC(random_state=42, probability=True)

ensemble_classifier = VotingClassifier(
    estimators=[
        ('lr', logistic_regression),
        ('dt', decision_tree),
        ('svm', support_vector_machine)
    ],
    voting='soft'
)
ensemble_classifier.fit(X_train_tfidf, y_train)
y_pred = ensemble_classifier.predict(X_test_tfidf)

print("Accuracy: ", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

Evaluation:

```
Accuracy:  0.6680497925311203

Classification Report:
               precision    recall  f1-score   support

    negative       0.60      0.50      0.55        64
     neutral       0.64      0.56      0.60        45
    positive       0.70      0.79      0.74       132

    accuracy                           0.67       241
   macro avg       0.65      0.61      0.63       241
weighted avg       0.66      0.67      0.66       241
```

Ensemble model with hyperparameter tuning

```
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import accuracy_score, classification_report
import lime
from lime.lime_text import LimeTextExplainer
import shap

X = tw_list['text']
y = tw_list['sentiment']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

logistic_regression = LogisticRegression(random_state=42)
decision_tree = DecisionTreeClassifier(random_state=42)
support_vector_machine = SVC(random_state=42, probability=True)

ensemble_classifier = VotingClassifier(
    estimators=[
        ('lr', logistic_regression),
        ('dt', decision_tree),
        ('svm', support_vector_machine)
    ],
    voting='soft'
)

# define parameter grid for hyperparameter tuning
param_grid = {
    'lr__C': [0.1, 1.0, 10.0],
    'dt__max_depth': [None, 5, 10],
    'svm__C': [0.1, 1.0, 10.0]
}

# create GridSearchCV object with the defined parameter grid
grid_search = GridSearchCV(ensemble_classifier, param_grid=param_grid, cv=5)
grid_search.fit(X_train_tfidf, y_train)

# get the best hyperparameters and evaluate the model
best_params = grid_search.best_params_
ensemble_classifier.set_params(**best_params)
ensemble_classifier.fit(X_train_tfidf, y_train)
y_pred = ensemble_classifier.predict(X_test_tfidf)

print("Best Parameters: ", best_params)
print("Accuracy: ", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

- This code is designed to analyse the sentiment of a dataset of tweets by combining three machine learning algorithms: Logistic Regression, Decision Tree, and Support Vector Machine (SVM), into an ensemble classifier using a "soft" voting strategy.
- The dataset is loaded into variables X and y, where X contains the tweet text and y contains the corresponding sentiment labels.
- The dataset is then split into training and testing sets using the train_test_split() function from scikit-learn.
- The tweet text is transformed into a numerical format using the TfidfVectorizer() function to create a matrix of TF-IDF features.

- A parameter grid is defined for hyperparameter tuning using the GridSearchCV() function, which fits the VotingClassifier to the training set with the defined parameter grid and uses cross-validation to evaluate the model's performance.
- The best hyperparameters are extracted using the best_params_ attribute of the GridSearchCV object.
- The VotingClassifier is then retrained using the best hyperparameters and evaluated on the test set using the accuracy_score() and classification_report() functions.
- Finally, the best hyperparameters and evaluation metrics are displayed in the console.

Evaluation:

```
Best Parameters:  {'dt__max_depth': 10, 'lr__C': 10.0, 'svm__C': 10.0}
Accuracy:  0.6970954356846473

Classification Report:
              precision    recall  f1-score   support

    negative       0.64      0.50      0.56        64
     neutral       0.74      0.58      0.65        45
    positive       0.71      0.83      0.76       132

    accuracy                           0.70       241
   macro avg       0.70      0.64      0.66       241
weighted avg       0.69      0.70      0.69       241
```

Comparison of the implemented models

Model	accuracy	precision	recall	f1-score
Logistic Regression	0.64	0.68	0.64	0.59
Decission Tree	0.61	0.62	0.61	0.61
Ensemble Model(SVM,Logistic Regression and Decision Tree)	0.67	0.66	0.67	0.66
Ensemble Model(SVM,Logistic Regression and Decision Tree) with hyperparameter tuning	0.70	0.69	0.70	0.69

Data Interpretation with LIME

```
# Initialize the explainer
explainer = LimeTextExplainer(class_names=['Negative', 'Positive'])

# Choose a sample from the test set for explanation
sample_idx = 20
sample_text = X_test.iloc[sample_idx]

# Define a custom predict_proba function to handle the input format for LIME
def custom_predict_proba(texts):
    transformed_texts = vectorizer.transform(texts)
    return ensemble_classifier.predict_proba(transformed_texts)

# Explain the prediction for the chosen sample
explanation = explainer.explain_instance(sample_text, custom_predict_proba, num_features=10, top_labels=2)
explanation.show_in_notebook(text=True)
```

The Individual tweets below are interpreted by using LIME.