In [1]:
```
#Importing necessary libraries
import numpy as np
import pandas as pd
from scipy import stats
import matplotlib.ticker as mtick
import matplotlib.pyplot as plt
import seaborn as sns
import torch
import torch.nn as nn
from sklearn.neural_network import MLPRegressor
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
import time
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics
from sklearn.model_selection import train_test_split, StratifiedKFold
from joblib import dump
import pickle
```

Importing data

In [2]:
```
#TF-IDF
x_train_tf_idf = "Data/x_train_tf_idf.pickle"
with open(x_train_tf_idf, 'rb') as x:
    x_train_tf_idf = pickle.load(x)

x_test_tf_idf = "Data/x_test_tf_idf.pickle"
with open(x_test_tf_idf, 'rb') as x:
    x_test_tf_idf = pickle.load(x)

y_train_tf = "Data/y_train_tf.pickle"
with open(y_train_tf, 'rb') as x:
    y_train_tf = pickle.load(x)

y_test_tf = "Data/y_test_tf.pickle"
with open(y_test_tf, 'rb') as x:
    y_test_tf = pickle.load(x)
```

In [3]:
```
#TF-IDF with stop
x_train_tf_idf_stop = "Data/x_train_tf_idf_stop.pickle"
with open(x_train_tf_idf_stop, 'rb') as x:
    x_train_tf_idf_stop = pickle.load(x)

x_test_tf_idf_stop = "Data/x_test_tf_idf_stop.pickle"
with open(x_test_tf_idf_stop, 'rb') as x:
    x_test_tf_idf_stop = pickle.load(x)

y_train_tf_stop = "Data/y_train_tf_stop.pickle"
with open(y_train_tf_stop, 'rb') as x:
    y_train_tf_stop = pickle.load(x)

y_test_tf_stop = "Data/y_test_tf_stop.pickle"
with open(y_test_tf_stop, 'rb') as x:
    y_test_tf_stop = pickle.load(x)
```

In [4]:
```
#W2V
x_train_w2v = "Data/x_train_w2v.pickle"
with open(x_train_w2v, 'rb') as x:
    x_train_w2v = pickle.load(x)

x_test_w2v = "Data/x_test_w2v.pickle"
with open(x_test_w2v, 'rb') as x:
    x_test_w2v = pickle.load(x)

y_train_w2v = "Data/y_train_w2v.pickle"
with open(y_train_w2v, 'rb') as x:
    y_train_w2v = pickle.load(x)

y_test_w2v = "Data/y_test_w2v.pickle"
with open(y_test_w2v, 'rb') as x:
    y_test_w2v = pickle.load(x)
```

In [5]:
```
#W2V with stop
x_train_w2v_stop = "Data/x_train_w2v_stop.pickle"
with open(x_train_w2v_stop, 'rb') as x:
    x_train_w2v_stop = pickle.load(x)

x_test_w2v_stop = "Data/x_test_w2v_stop.pickle"
with open(x_test_w2v_stop, 'rb') as x:
    x_test_w2v_stop = pickle.load(x)

y_train_w2v_stop = "Data/y_train_w2v_stop.pickle"
with open(y_train_w2v_stop, 'rb') as x:
    y_train_w2v_stop = pickle.load(x)

y_test_w2v_stop = "Data/y_test_w2v_stop.pickle"
with open(y_test_w2v_stop, 'rb') as x:
    y_test_w2v_stop = pickle.load(x)
```

Baseline Model (Naive Bayes + TF-IDF)

In [6]:
```
#naive bayes without stopwords
model_naive = MultinomialNB()

start = time.time()
model_naive.fit(x_train_tf_idf, y_train_tf)
end = time.time()

print('Fitting time for Naive Bayes ', (end-start))
```

```
Fitting time for Naive Bayes  0.023221254348754883
```

In [7]:
```
#Exporting Naive bayes without stopword model
dump(model_naive, '01_NB.joblib')
```

Out[7]:
```
['01_NB.joblib']
```

In [8]:
```
#naive bayes with stopwords
model_naive_with_stop = MultinomialNB()

start = time.time()
model_naive_with_stop.fit(x_train_tf_idf_stop, y_train_tf_stop)
end = time.time()

print('Fitting time for Naive Bayes having stopwords', (end-start))
```

```
Fitting time for Naive Bayes having stopwords 0.023058414459228516
```

In [9]:
```
#Exporting Naive bayes with stopword model
dump(model_naive_with_stop, '01_NB_stop.joblib')
```

Out[9]:
```
['01_NB_stop.joblib']
```

MLP

Grid Search for MLP

In [10]:
```
#Defining 5-fold stratified cv
strat1 = StratifiedKFold(n_splits = 5, random_state = 40, shuffle = True)

#Defining sets of predefined hyperparameters
param = {'hidden_layer_sizes': [(5,5),(5,5,5)], "activation": ['logistic', "re

#Grid search
MLP = GridSearchCV(MLPClassifier(max_iter = 1000, random_state = 40), param, c
MLP.fit(x_train_w2v, y_train_w2v)

#Print best score and hyperparameters
print("Best accuracy score for base dataset: ", MLP.best_score_)
print("Hyperparameters used for best accuracy for base dataset: ", MLP.best_pa
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
Best accuracy score for base dataset:  0.5145369978546264
Hyperparameters used for best accuracy for base dataset:  {'activation': 'log
istic', 'hidden_layer_sizes': (5, 5, 5), 'learning_rate_init': 0.003}
```

Hyperparameter table

In [11]:
```
#Initializing pandas table
table = pd.concat([pd.DataFrame(MLP.cv_results_["params"]),pd.DataFrame(MLP.cv
                                                                        colu
#Sort table based on validation accuracy
table = table.sort_values(by = ['val_acc'], ascending = False)

#Clean the table and display it
table.reset_index(inplace = True)
table = table.drop(columns = ['index'])
table
```

Out[11]:

	activation	hidden_layer_sizes	learning_rate_init	val_acc
0	logistic	(5, 5, 5)	0.003	0.514537
1	logistic	(5, 5)	0.003	0.514084
2	logistic	(5, 5, 5)	0.030	0.513993
3	relu	(5, 5)	0.030	0.513993
4	relu	(5, 5, 5)	0.003	0.513993
5	relu	(5, 5, 5)	0.030	0.513993
6	logistic	(5, 5)	0.030	0.513087
7	relu	(5, 5)	0.003	0.511729

Train MLP + W2V

In [12]:
```
#Define model
model = MLPClassifier(max_iter= 1000, random_state= 42, activation= "logistic"
                      learning_rate_init = 0.030)

start_time = time.time() #Measure time
model.fit(x_train_w2v, y_train_w2v) #Fitting model
training_time = time.time() - start_time #Training time

#Exporting Optimised MLP model
dump(model, '02_MLP_Optimised_w2v.joblib')

#Print training time
print("Training time: ", training_time)
```

```
Training time:  3.68182110786438
```

Train MLP + TF-IDF

In [13]:
```
#Define model
model = MLPClassifier(max_iter= 1000, random_state= 42, activation= "logistic"
                      learning_rate_init = 0.030)

start_time = time.time() #Measure time
model.fit(x_train_tf_idf, y_train_tf) #Fitting model
training_time = time.time() - start_time #Training time

#Exporting Optimised MLP model
dump(model, '02_MLP_Optimised_tf_idf.joblib')

#Print training time
print("Training time: ", training_time)
```

```
Training time:  13.107040405273438
```

Decision Tree

Grid Search for Decision Tree

In [14]:
```
#Defining 5-fold stratified cv
strat2 = StratifiedKFold(n_splits = 5, random_state = 40, shuffle = True)

#Defining sets of predefined hyperparameters
param2 = {'max_depth': [2, 4, 8],
              'min_samples_leaf': [2, 8, 16]}

#Grid search
DT= DecisionTreeClassifier()
gri = GridSearchCV(DT, param2, cv = strat2, verbose = 1)
gri.fit(x_train_w2v, y_train_w2v)
#Print best score and hyperparameters
print("Best accuracy score: ", gri.best_score_)
print("Hyperparameters used for best accuracy: ", gri.best_params_)
```

```
Fitting 5 folds for each of 9 candidates, totalling 45 fits
Best accuracy score:  0.5133592565985003
Hyperparameters used for best accuracy:  {'max_depth': 2, 'min_samples_leaf':
2}
```

Hyperparameter table

In [15]:
```
#Initializing pandas table
table = pd.concat([pd.DataFrame(gri.cv_results_["params"]),pd.DataFrame(gri.cv
                                                                        colu
#Sort table based on validation accuracy
table = table.sort_values(by = ['val_acc'], ascending = False)

#Clean the table and display it
table.reset_index(inplace = True)
table = table.drop(columns = ['index'])
table
```

Out[15]:

	max_depth	min_samples_leaf	val_acc
0	2	2	0.513359
1	2	8	0.513359
2	2	16	0.513359
3	4	8	0.512635
4	4	16	0.512635
5	4	2	0.512363
6	8	2	0.483018
7	8	8	0.481117
8	8	16	0.480844

Train Decision Tree + W2V

In [16]:
```
#Define model
model = DecisionTreeClassifier(max_depth = 2, min_samples_leaf = 2)

start_time = time.time() #Measure time
model.fit(x_train_w2v, y_train_w2v) #Fitting model
training_time = time.time() - start_time #Training time

#Exporting Optimised MLP model
dump(model, '02_DT_Optimised_w2v.joblib')

#Print training time
print("Training time: ", training_time)
```

```
Training time:  0.37710070610046387
```

Train Decision Tree + TF-IDF

In [17]:
```
#Define model
model = DecisionTreeClassifier(max_depth = 2, min_samples_leaf = 2)

start_time = time.time() #Measure time
model.fit(x_train_tf_idf, y_train_tf) #Fitting model
training_time = time.time() - start_time #Training time

#Exporting Optimised MLP model
dump(model, '02_DT_Optimised_tf_idf.joblib')

#Print training time
print("Training time: ", training_time)
```

```
Training time:  0.11645627021789551
```

In [ ]:

In [ ]: