National College of Ireland

Configuration Manual

MSc Research Project
DATA ANALYTICS (MSCDAD_JAN23C)

SHREEDHAR SHARMA
Student ID: 22142835

School of Computing
National College of Ireland

Supervisor: DR. ARGHIR NICOLAE MOLDOVAN

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	SHREEDHAR SHARMA
Student ID:	22142835
Programme:	MSc DATA ANALYTICS **Year:** 2023-24
Module:	MSc RESEARCH PROJECT
Lecturer:	DR. ARGHIR NICOLAE MOLDOVAN
Submission Due Date:	31.01.2024
Project Title:	DISASTER TWEETS ENSEMBLE: ENSEMBLE FOR DISASTER TWEETS CLASSIFICATION
Word Count:	1055 **Page Count**: 14

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: SHREEDHAR SHARMA

Date: 30.01.2024

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

SHREEDHAR SHARMA
22142835

1 Introduction

This document discusses the system requirements; hardware and software required for the implementation of the project. It also showcases the steps performed in the successful implementation of the research project," Disaster Tweets Ensemble: Ensemble for Disaster Tweets Classification".

2 System Configuration

The minimum system requirements for running the discussed project are specified as follows:

2.1 Hardware Requirement

- Processor: i5 or equivalent
- RAM: Minimum 8GB
- Storage: Minimum 5GB

2.2 Software Requirement

- Jupyter Notebook: Open source for interactive documents with code, widely used in data analysis and machine learning.
- Google Colab: cloud-based Python platform for collaborative machine learning tasks, eliminating the need for extensive local resources.

3 Environment Setup

In this step, we will install the necessary libraries using 'pip install' and import the necessary modules in Python for the successful execution of our project. In our project, we worked on three datasets Queensland, Nepal and Crisis and implemented four models Logistic Regression, Naïve Bayes, XGBoost and LSTM with three techniques CountVectorizer, Word2Vec and TF-IDF on all these datasets to check the best accuracy of these models on the dataset so that we can use the embedded approach to combine the models and enhance the accuracy. We are going ahead with the Queensland Dataset, considering the common steps in all three datasets (Queensland, Nepal and Crisis).

```
In [1]: pip install textblob

Requirement already satisfied: textblob in c:\users\ashts\anaconda3\lib\site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in c:\users\ashts\anaconda3\lib\site-packages (from textblob) (3.7)
Requirement already satisfied: regex>=2021.8.3 in c:\users\ashts\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (2022.7.9)
Requirement already satisfied: joblib in c:\users\ashts\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (1.1.0)
Requirement already satisfied: click in c:\users\ashts\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (8.0.4)
Requirement already satisfied: tqdm in c:\users\ashts\anaconda3\lib\site-packages (from nltk>=3.1->textblob) (4.64.1)
Requirement already satisfied: colorama in c:\users\ashts\anaconda3\lib\site-packages (from click->nltk>=3.1->textblob) (0.4.5)
Note: you may need to restart the kernel to use updated packages.

In [2]: import numpy as np# is for mathematical calculations
import pandas as pd # to deal with tabulear data
import requests # use to send html requests
import csv
import missingno as msno # used to find missing values
import time
##########################
# visualizations
import plotly
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
##########################
```

Fig 1. Installation of libraries and modules

4 Exploratory Data Analysis

4.1 Data Loading

In this step, we will be using the Pandas library to read the CSV (Comma-separated values). As we execute this command the data from queensland.csv can be accessed, analyzed and manipulated using the variable queensland. The same method we will be following for loading the CSV files of Nepal and Crisis in the variable Nepal and crisis for loading and working with respective datasets.

```
In [3]: # Record the start time
start_time = time.time()
queensland = pd.read_csv('queensland.csv')
```

Fig 2. Data Loading

4.2 Data Understanding

This step will used to view the first 5 rows of the dataset, this command is particularly used for getting the column names and the type of data stored in these columns. We are using the same command on the other datasets.

```
In [4]: queensland.head()
```

Out[4]:

	tweet_id	text	label
0	295541465013182465	I just though about the night I went clubbing ...	not_relevant
1	295485717465935873	Looks like its going to be another long night ...	not_relevant
2	297292225811972097	@LaniiBanani hahahaha I just told him id have ...	not_relevant
3	297199686392094720	Off to meeting.... with so called... Baaps of ...	not_relevant
4	295527618449666049	Doubt I'll be getting much sleep tonight	not_relevant

Fig 3. Data Understanding

4.3 Finding Missing and Duplicate values

In this step, we will be analysing the data for the missing values so that we have properly balanced data for further computations. We are using the `msno.bar(queensland)` to visualise the missing values in the queensland dataset and then we will check for duplicate values and delete them.

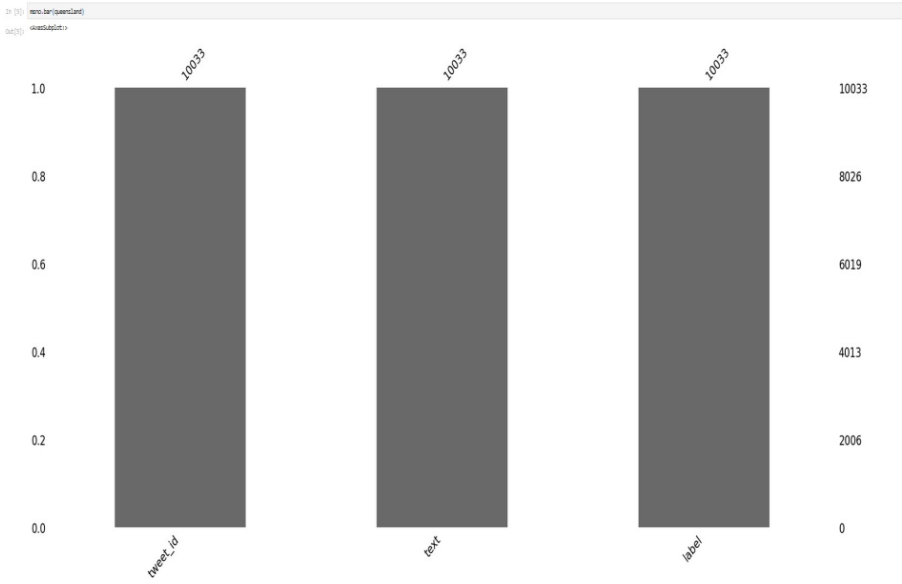

Fig 4. Visualization of missing values

```
In [9]: queensland.duplicated().sum()
Out[9]: 989

In [10]: queensland.drop_duplicates(inplace=True)

In [11]: queensland.isna().sum()
Out[11]: text     0
         label    0
         dtype: int64
```

Fig 5. Checking and deleting the duplicate values

4.4 Data Subsetting

Data subsetting is the process of choosing the specific data from the dataset. In the queensland dataset, we are choosing columns ‘text’ and ‘label’ for further analysis. We are dealing with the labelled data having tweets and the information whether they are relevant or not relevant to the disaster.

Data Subsetting

```
In [6]: queensland = queensland[['text','label']]

In [7]: queensland.label.value_counts()

Out[7]: relevant        5414
        not_relevant    4619
        Name: label, dtype: int64

In [8]: fig = px.pie(queensland, values=queensland.label.value_counts(),names = ['Relevant','Not_Relevant'],
                     title='Composition of variables in Queensland Dataset')
        fig.show()
```

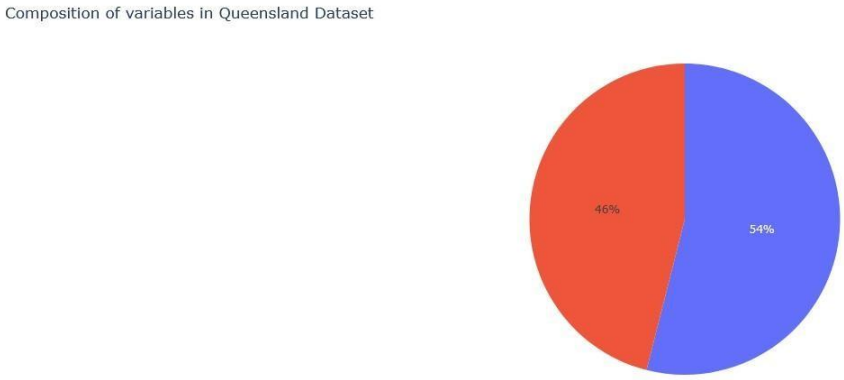

Fig 6. Data Subsetting on Queensland Dataset

5 Natural language Understanding (NLU)

The following steps are performed on the datasets to do the case standardization, removal of punctuation and other unnecessary values, additional text analysis to find the composition of stop words and then removal of stopwords and finally lemmatization (reducing the word to base form) and stemming (trimming word to root form). These are the vital steps to prepare the data and to make it uniform and understandable for processing and analysis.

```
In [12]: queensland['text'] = queensland['text'].astype(str)
         queensland['text'] = queensland['text'].apply(lambda x:x.lower())
```

Fig 7. Case Standardization on Queensland Dataset

```
In [13]: import re
         def pre_process_text(x):
             x = x.strip()
             x = re.sub(r' +', ' ', x)
             x = re.sub(r"[-()\"#/@;:{}`+=~|.!?,'0-9]", "", x)
             return(x)

In [14]: queensland['text'] = queensland['text'].apply(pre_process_text)

In [15]: def clean_text(x):
             for i in x.split(" "):
                 return "".join(re.sub(r"[^a-zA-Z ]","",x))
         queensland['text'] = queensland['text'].apply(clean_text)

In [16]: queensland['label'] = queensland['label'].map({'relevant':0,'not_relevant':1})
         queensland
```

Fig 8. Removal of Punctuation and other values from Queensland Dataset

In [17]:
```
# Additional Text Analysis
queensland['numeric'] = queensland['text'].apply(lambda x: len([x for x in x.split() if x.isdigit()]))
queensland['alphanumeric'] = queensland['text'].apply(lambda x: len([x for x in x.split() if x.isalnum()]))
queensland['alphabetics'] = queensland['text'].apply(lambda x: len([x for x in x.split() if x.isalpha()]))
queensland["Stopwords"] = queensland['text'].apply(lambda x: len([w for w in str(x).lower().split() if w in eng_stopwords]))
queensland
```

Out[17]:

	text	label	numeric	alphanumeric	alphabetics	Stopwords
0	i just though about the night i went clubbing ...	1	0	14	14	8
1	looks like its going to be another long night ...	1	0	14	14	4
2	laniibanani hahahaha i just told him id have t...	1	0	30	30	18
3	off to meeting with so called baaps of mineral...	1	0	10	10	5
4	doubt ill be getting much sleep tonight	1	0	7	7	1
...	...	...	...	...	...	...
10022	is flood seriously that trending rt channelnew...	0	0	13	13	2
10028	grafton queensland flood peaks at meters torr...	0	0	15	15	3
10029	helicopters deployed to rescue flood victims i...	0	0	16	16	3
10031	skeletonunicorn the waters upto the door my ol...	0	0	20	20	11
10032	queenslands flood crisis deepens as death toll...	0	0	16	16	3

9044 rows × 6 columns

Fig 9. Additional text analysis performed on the Queensland Dataset

In [18]:
```
plt.figure(figsize=(12, 6))
sns.kdeplot(queensland[queensland['label'] == 0]['Stopwords'], label='Relevant Tweet')
sns.kdeplot(queensland[queensland['label'] == 1]['Stopwords'], label='Not Relevant Tweet', alpha=0.5, shade=True, color='#ea4335')
plt.title('Stopwords Distribution')
plt.legend(title='Tweet')
plt.show()
```

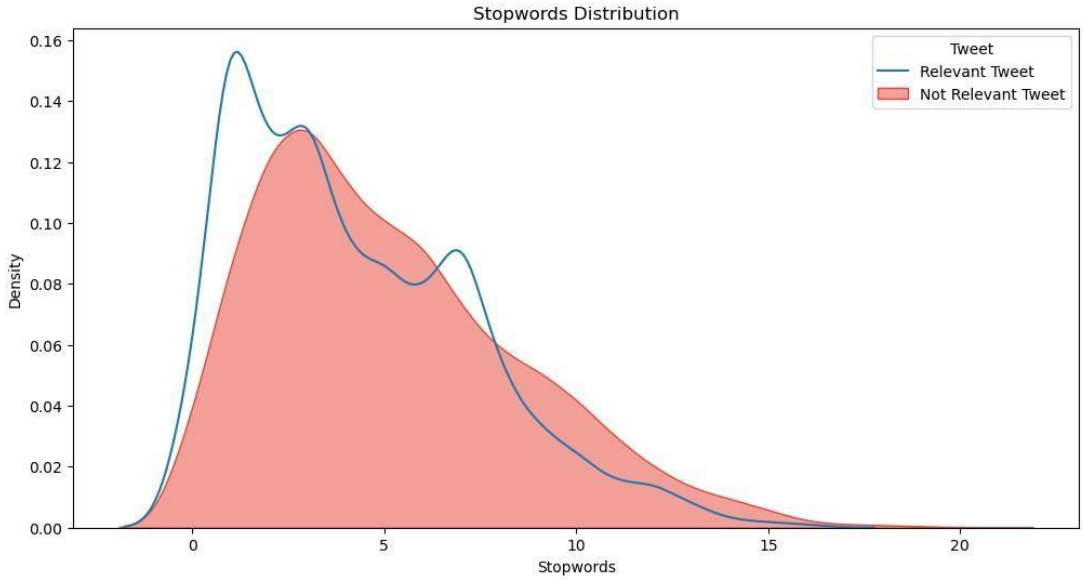

Fig 10. Visualization of Stpwords on the Queensland Dataset

In [19]:
```
stop = set(nltk.corpus.stopwords.words('english'))
queensland['text'] = queensland['text'].apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

Fig 11. Removal of Stopwords from the Queensland Dataset

In [20]:
```
def lemmatize_stemming(text):
    stemmer=SnowballStemmer('english')
    return stemmer.stem(WordNetLemmatizer().lemmatize(text, pos='v'))
```

In [21]:
```
import nltk
nltk.download('omw-1.4')
queensland['text'] = queensland['text'].apply(lambda x:lemmatize_stemming(x))
```

Fig 12. Lemmatization and stemming performed on the Queensland Dataset

6 Data Preparation

The process of organizing and structuring the data suitable for feeding to the machine learning model. The train-test split is useful to evaluate the model performance as the model was trained on the training set of data and then evaluated on the test set of data to understand how well the model behaves to new and unseen data. On the other hand data representation converts the data into the format that the machine learning algorithm can learn from.

Train-Test Split

```
In [24]: x_queens = queensland['text']
         y_queens = queensland['label']

In [25]: train_x_queens, test_x_queens, train_y_queens, test_y_queens = train_test_split(x_queens,y_queens,random_state=101,test_size=0.3)
```

Fig 13. Train-Test split of the Queensland Dataset

Representations Queensland Dataset

```
In [26]: count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}', ngram_range=(1, 2))
         xtrain_count_queens = count_vect.fit_transform(train_x_queens)

         tfidf_transformer = TfidfTransformer()
         X_train_tfidf_queens = tfidf_transformer.fit_transform(xtrain_count_queens)
         xtest_count_queens = count_vect.transform(test_x_queens)


         X_test_tfidf_queens = tfidf_transformer.transform(xtest_count_queens)

In [27]: import gensim
         queensland_model = gensim.models.Word2Vec(sentences=queensland['text'], vector_size=100, window=5, min_count=5, workers=4)
         queensland_word_vectors = queensland_model.wv

         def queensland_word2vec(text):
             vecs = []
             for word in text.split():
                 if word in queensland_word_vectors:
                     vecs.append(queensland_word_vectors[word])
             if len(vecs) > 0:
                 return np.mean(vecs, axis=0)
             else:
                 return np.zeros(100)

         queensland['text_vecs'] = queensland['text'].apply(queensland_word2vec)

In [28]: queensland_features = np.array(list(map(np.array, queensland['text_vecs'])))

In [29]: queensland_train_x, queensland_test_x, queensland_train_y, queensland_test_y = train_test_split(queensland_features, queensland['label'], test_size=0.3, random_state=42)
```

Fig 14. Representation of the Queensland Dataset

7 Implemented Dataset Models

The following machine learning models (Logistic Regression, XGBoost, Naïve Bayes and LSTM) with vectors (CountVectorizer, Word2Vec and TF-IDF) were implemented on all three datasets (Nepal, Crisis and Queensland) to find out the best accuracy so that the same can be used in developing the ensemble for the project which can perform better. The following is the implementation code for the Queensland dataset models. Further, we will be comparing the accuracy and ROC-AUC Score of all the models on all the datasets.

1.1 Logistic Regression With CountVectorizer

```
In [30]: logreg = LogisticRegression()
         logreg.fit(xtrain_count_queens, train_y_queens)

Out[30]: LogisticRegression()

In [31]: pred_lg_cnt_queens = logreg.predict(xtest_count_queens)
         queens_logit_cnt = accuracy_score(test_y_queens, pred_lg_cnt_queens).round(3)
         print("Accuracy:",accuracy_score(test_y_queens, pred_lg_cnt_queens).round(3))
         queens_logit_cnt_auc = roc_auc_score(test_y_queens, pred_lg_cnt_queens).round(3)
         print("ROC-AUC Score: ",queens_logit_cnt_auc)
         print(classification_report(test_y_queens, pred_lg_cnt_queens,digits=3))
```

Fig 15. Logistic Regression with CountVectorizer on Queensland Dataset

1.2 Logistic Regression With TF-IDF

In [33]:
```
logreg = LogisticRegression()
logreg.fit(X_train_tfidf_queens, train_y_queens)
```

Out[33]:
```
LogisticRegression()
```

In [34]:
```
pred_lg_tfidf_queens = logreg.predict(X_test_tfidf_queens)
queens_logit_tfidf = accuracy_score(test_y_queens, pred_lg_tfidf_queens).round(3)
print("Accuracy:",accuracy_score(test_y_queens, pred_lg_tfidf_queens).round(3))
queens_logit_tfidf_auc = roc_auc_score(test_y_queens, pred_lg_tfidf_queens).round(3)
print("ROC-AUC Score: ",queens_logit_tfidf_auc)
print(classification_report(test_y_queens, pred_lg_tfidf_queens,digits=3))
```

Fig 16. Logistic Regression with TF-IDF on Queensland Dataset

1.3 Logistic Regression with Word2Vec

In [36]:
```
queens_lr_model = LogisticRegression(max_iter=1000)
queens_lr_model.fit(queensland_train_x, queensland_train_y)
queens_lr_pred_y = queens_lr_model.predict(queensland_test_x)

acc_queens_word2vec_logit = accuracy_score(queensland_test_y, queens_lr_pred_y).round(3)
print("Accuracy:", accuracy_score(queensland_test_y, queens_lr_pred_y).round(3))
queens_logit_word2vec_auc = roc_auc_score(queensland_test_y, queens_lr_pred_y).round(3)
print("ROC-AUC Score: ",queens_logit_word2vec_auc)
print(classification_report(queensland_test_y, queens_lr_pred_y,digits=3))
```

Fig 17. Logistic Regression with Word2Vec on Queensland Dataset

2.1 Naive Bayes With CountVectorizer

In [38]:
```
nb = MultinomialNB()
# Fit the data
nb.fit(xtrain_count_queens, train_y_queens)
```

Out[38]:
```
MultinomialNB()
```

In [39]:
```
pred_nb_cnt_queens = nb.predict(xtest_count_queens)
queens_nb_cnt = accuracy_score(test_y_queens, pred_nb_cnt_queens).round(3)
print("Accuracy:",accuracy_score(test_y_queens, pred_nb_cnt_queens).round(3))
queens_nb_cnt_auc = roc_auc_score(test_y_queens, pred_nb_cnt_queens).round(3)
print("ROC-AUC Score: ",queens_logit_cnt_auc)
print(classification_report(test_y_queens, pred_nb_cnt_queens,digits=3))
```

Fig 18. Naïve Bayes with CountVectorizer on Queensland Dataset

2.2 Naive Bayes With TF-IDF

In [41]:
```
nb = MultinomialNB()
# Fit the data
nb.fit(X_train_tfidf_queens, train_y_queens)
```

Out[41]:
```
MultinomialNB()
```

In [42]:
```
pred_nb_tfidf_queens = nb.predict(X_test_tfidf_queens)
queens_nb_tfidf = accuracy_score(test_y_queens, pred_nb_tfidf_queens).round(3)
print("Accuracy:",accuracy_score(test_y_queens, pred_nb_tfidf_queens).round(3))
queens_nb_tfidf_auc = roc_auc_score(test_y_queens, pred_nb_tfidf_queens).round(3)
print("ROC-AUC Score: ",queens_logit_cnt_auc)
print(classification_report(test_y_queens, pred_nb_tfidf_queens,digits=3))
```

Fig 19. Naïve Bayes with TF-IDF on Queensland Dataset

2.3 Naive Bayes with Word2Vec

In [44]:
```
# Train and Evaluate Naive Bayes with Word2Vec features for Queensland Dataset
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report, accuracy_score

queens_nb_model = GaussianNB()
queens_nb_model.fit(queensland_train_x, queensland_train_y)
queens_nb_pred_y = queens_nb_model.predict(queensland_test_x)
acc_queens_nb_word2vec =  accuracy_score(queensland_test_y, queens_nb_pred_y).round(3)
print("Accuracy:", accuracy_score(queensland_test_y, queens_nb_pred_y).round(3))
queens_nb_word2vec_auc = roc_auc_score(queensland_test_y, queens_nb_pred_y).round(3)
print("ROC-AUC Score: ",queens_nb_word2vec_auc)
print(classification_report(queensland_test_y, queens_nb_pred_y,digits=3))
```

Fig 20. Naïve Bayes with Word2Vec on Queensland Dataset

3.1 XgBoost With Count-Vectorizer

In [46]:
```
xgb_estimator = XGBClassifier(n_estimators=200,
                      random_state = 42,
                     n_jobs=-1,
                     verbose=1)
xgb_estimator.fit(xtrain_count_queens, train_y_queens)
```

```
[16:55:05] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07593ffd91cd9da33-1\xgboost\xgboost-ci-windows\src\learner.cc:767:
Parameters: { "verbose" } are not used.
```

Out[46]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=200, n_jobs=-1, num_parallel_tree=None,
              predictor=None, random_state=42, ...)
```

In [47]:
```
pred_xgb_cnt_queens = xgb_estimator.predict(xtest_count_queens)
queens_xgb_cnt = accuracy_score(test_y_queens, pred_xgb_cnt_queens).round(3)
print("Accuracy:",accuracy_score(test_y_queens, pred_xgb_cnt_queens).round(3))
queens_xgb_cnt_auc = roc_auc_score(test_y_queens, pred_xgb_cnt_queens).round(3)
print("ROC-AUC Score: ",queens_logit_cnt_auc)

print(classification_report(test_y_queens, pred_xgb_cnt_queens,digits=3))
```

Fig 21. XGBoost with CountVectorizer on Queensland Dataset

3.2 XgBoost With TF-IDF

In [49]:
```
xgb_estimator_queens_tfidf = XGBClassifier(n_estimators=200,
                         random_state = 42,
                        n_jobs=-1,
                        verbose=1)
xgb_estimator_queens_tfidf.fit(X_train_tfidf_queens, train_y_queens)
```

```
[16:55:06] WARNING: C:\buildkite-agent\builds\buildkite-windows-cpu-autoscaling-group-i-07593ffd91cd9da33-1\xgboost\xgboost-ci-windows\src\learner.cc:767:
Parameters: { "verbose" } are not used.
```

Out[49]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              n_estimators=200, n_jobs=-1, num_parallel_tree=None,
              predictor=None, random_state=42, ...)
```

In [50]:
```
pred_xgb_tfidf_queens = xgb_estimator_queens_tfidf.predict(X_test_tfidf_queens)
queens_xgb_tfidf = accuracy_score(test_y_queens, pred_xgb_tfidf_queens).round(3)
print("Accuracy:",accuracy_score(test_y_queens, pred_xgb_tfidf_queens).round(3))
queens_xgb_tfidf_auc = roc_auc_score(test_y_queens, pred_xgb_tfidf_queens).round(3)
print("ROC-AUC Score: ",queens_logit_cnt_auc)
print(classification_report(test_y_queens, pred_xgb_tfidf_queens,digits=3))
```

Fig 22. XGBoost with TF-IDF on Queensland Dataset

3.3 XgBoost With word2vec

In [52]:
```
# Train and Evaluate XGBoost with Word2Vec features for Queensland Dataset
from xgboost import XGBClassifier

queens_xgb_model = XGBClassifier(use_label_encoder=False, eval_metric='logloss')
queens_xgb_model.fit(queensland_train_x, queensland_train_y)
queens_xgb_pred_y = queens_xgb_model.predict(queensland_test_x)

print("Accuracy:", accuracy_score(queensland_test_y, queens_xgb_pred_y).round(3))
acc_queens_xgb_word2vec = accuracy_score(queensland_test_y, queens_xgb_pred_y).round(3)
queens_xgb_word2vec_auc = roc_auc_score(queensland_test_y, queens_xgb_pred_y).round(3)
print("ROC-AUC Score: ",queens_xgb_word2vec_auc)
print(classification_report(queensland_test_y, queens_xgb_pred_y,digits=3))
```

Fig 23. XGBoost with Word2Vec on Queensland Dataset

4.1 LSTM with Word2vec

In [54]:
```
queens_word2vec_lstm = Sequential()
queens_word2vec_lstm.add(LSTM(64, input_shape=(queensland_train_x.shape[1], 1)))
queens_word2vec_lstm.add(Dropout(0.2))
queens_word2vec_lstm.add(Dense(1, activation='sigmoid'))
queens_word2vec_lstm.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
queens_word2vec_lstm_history = queens_word2vec_lstm.fit(queensland_train_x, queensland_train_y, epochs=5, batch_size=64, validation_data=(queensland_test_x, queensland_test_y))
plt.plot(queens_word2vec_lstm_history.history['accuracy'], label='Train Accuracy')
plt.plot(queens_word2vec_lstm_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Queensland Word2Vec LSTM Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

In [55]:
```
pred_lstm_word2vec = queens_word2vec_lstm.predict(queensland_test_x)
print("Accuracy:", accuracy_score(queensland_test_y, pred_lstm_word2vec.round(),normalize=True).round(3))
word2vec_lstm_acc = accuracy_score(queensland_test_y, pred_lstm_word2vec.round(),normalize=True).round(3)
```

In [56]:
```
lstm_word2vec_auc = roc_auc_score(queensland_test_y, pred_lstm_word2vec).round(3)
print("ROC-AUC Score: ",lstm_word2vec_auc)
print(classification_report(queensland_test_y, pred_lstm_word2vec.round(),digits=3))
```

Fig 24. LSTM with Word2Vec on Queensland Dataset

4.2 LSTM with TF-IDF

```
In [57]: X_train_tfidf_reshaped_queens = np.reshape(X_train_tfidf_queens.toarray(), (X_train_tfidf_queens.shape[0], 1, X_train_tfidf_queens.shape[1]))
X_test_tfidf_reshaped_queens = np.reshape(X_test_tfidf_queens.toarray(), (X_test_tfidf_queens.shape[0], 1, X_test_tfidf_queens.shape[1]))

queensland_tfidf_lstm = Sequential()
queensland_tfidf_lstm.add(LSTM(64, input_shape=(1, X_train_tfidf_queens.shape[1])))
queensland_tfidf_lstm.add(Dropout(0.2))
queensland_tfidf_lstm.add(Dense(1, activation='sigmoid'))
queensland_tfidf_lstm.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
queensland_tfidf_lstm_history = queensland_tfidf_lstm.fit(X_train_tfidf_reshaped_queens,queensland_train_y , epochs=5, batch_size=64, validation_data=(X_test_tfidf_reshaped_queens, queensland_test_y))
plt.plot(queensland_tfidf_lstm_history.history['accuracy'], label='Train Accuracy')
plt.plot(queensland_tfidf_lstm_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Queensland TF-IDF LSTM Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
In [58]: pred_lstm_tfidf = queensland_tfidf_lstm.predict(X_test_tfidf_reshaped_queens)
print("Accuracy:", accuracy_score(queensland_test_y, pred_lstm_tfidf.round(),normalize=True).round(3))
tfidf_lstm_acc = accuracy_score(queensland_test_y, pred_lstm_tfidf.round(),normalize=True).round(3)
```

```
In [59]: lstm_tfidf_auc = roc_auc_score(queensland_test_y, pred_lstm_tfidf).round(3)
print("ROC-AUC Score: ",lstm_tfidf_auc)
print(classification_report(queensland_test_y, pred_lstm_tfidf.round(),digits=3))
```

Fig 25. LSTM with TF-IDF on Queensland Dataset

4.3 LSTM with CountVectorizer

```
In [60]: xtrain_count_reshaped_queens = np.reshape(xtrain_count_queens.toarray(), (xtrain_count_queens.shape[0], 1, xtrain_count_queens.shape[1]))
xtest_count_reshaped_queens = np.reshape(xtest_count_queens.toarray(), (xtest_count_queens.shape[0], 1, xtest_count_queens.shape[1]))

queens_count_lstm = Sequential()
queens_count_lstm.add(LSTM(64, input_shape=(1, xtrain_count_queens.shape[1])))
queens_count_lstm.add(Dropout(0.2))
queens_count_lstm.add(Dense(1, activation='sigmoid'))
queens_count_lstm.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
queens_count_lstm_history = queens_count_lstm.fit(xtrain_count_reshaped_queens, train_y_queens, epochs=5, batch_size=64, validation_data=(xtest_count_reshaped_queens, test_y_queens))
plt.plot(queens_count_lstm_history.history['accuracy'], label='Train Accuracy')
plt.plot(queens_count_lstm_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Queensland CountVectorizer LSTM Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
In [61]: queens_count_lstm = queens_count_lstm.predict(xtest_count_reshaped_queens)
```

```
85/85 [==============================] - 1s 13ms/step
```

```
In [62]: print("Accuracy:", accuracy_score(queensland_test_y,queens_count_lstm.round(),normalize=True).round(3))
queens_lstm_cnt = accuracy_score(queensland_test_y,queens_count_lstm.round(),normalize=True).round(3)
```

```
In [63]: lstm_cnt_auc = roc_auc_score(queensland_test_y, queens_count_lstm).round(3)
print("ROC-AUC Score: ",lstm_cnt_auc)
print(classification_report(queensland_test_y, queens_count_lstm.round(),digits=3))
```

Fig 26. LSTM with CountVectorizer on Queensland Dataset

Comparison

```
In [64]: Queens_df = {'Classifiers':['Logistic_Regression_CountVectorizer','Logistic_Regression_TF-IDF','Logistic_Regression_Word2Vec',
                      'NaiveBayes_CountVectorizer','NaiveBayes_TF-IDF','NiveBayes_Word2Vec','XGBoost_CountVectorizer','XGBoost_TF-IDF',
                           'XgBoost_Word2Vec','LSTM_CountVectorizer','LSTM_TF-IDF','LSTM_Word2Vec'],
                        'Accuracy': [queens_logit_cnt,queens_logit_tfidf,acc_queens_word2vec_logit,
                                     queens_nb_cnt,queens_nb_tfidf,acc_queens_nb_word2vec,queens_xgb_cnt,queens_xgb_tfidf,
acc_queens_xgb_word2vec,queens_lstm_cnt,tfidf_lstm_acc,word2vec_lstm_acc],
                }
```

```
In [65]: Queens_df = pd.DataFrame(Queens_df).round(3)
Queens_df
```

Out[65]:

	Classifiers	Accuracy
0	Logistic_Regression_CountVectorizer	0.957
1	Logistic_Regression_TF-IDF	0.953
2	Logistic_Regression_Word2Vec	0.508
3	NaiveBayes_CountVectorizer	0.941
4	NaiveBayes_TF-IDF	0.934
5	NiveBayes_Word2Vec	0.510
6	XGBoost_CountVectorizer	0.958
7	XGBoost_TF-IDF	0.955
8	XgBoost_Word2Vec	0.509
9	LSTM_CountVectorizer	0.497
10	LSTM_TF-IDF	0.503
11	LSTM_Word2Vec	0.516

Fig 27. Comparison of Accuracies of all models on the Queensland Dataset

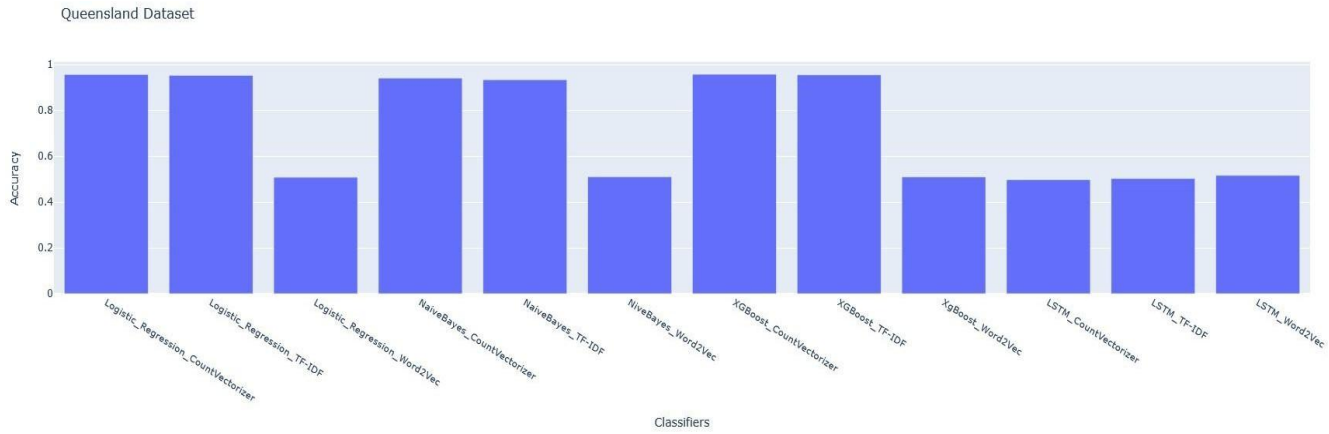

Fig 28. Graphical representation of the accuracy of all models on the Queensland Dataset

AUC Comparison

```
In [67]: queens_df_auc = {'Classifiers':['Logistic_Regression_CountVectorizer','Logistic_Regression_TF-IDF','Logistic_Regression_Word2Vec',
                        'NaiveBayes_CountVectorizer','NaiveBayes_TF-IDF','NiveBayes_Word2Vec','XGBoost_CountVectorizer','XGBoost_TF-IDF',
                            'XgBoost_Word2Vec','LSTM_CountVectorizer','LSTM_TF-IDF','LSTM_Word2Vec'],
                         'AUC': [queens_logit_cnt_auc,queens_logit_tfidf_auc,queens_logit_word2vec_auc,queens_nb_cnt_auc,queens_nb_tfidf_auc,queens_nb_word2vec_auc,queens_xgb_cnt_auc,queens_xgb_tfidf_auc,
queens_xgb_word2vec_auc,lstm_cnt_auc,lstm_tfidf_auc,lstm_word2vec_auc]}

In [68]: queens_df_auc = pd.DataFrame(queens_df_auc)
         queens_df_auc
```

Out[68]:

	Classifiers	AUC
0	Logistic_Regression_CountVectorizer	0.957
1	Logistic_Regression_TF-IDF	0.952
2	Logistic_Regression_Word2Vec	0.521
3	NaiveBayes_CountVectorizer	0.942
4	NaiveBayes_TF-IDF	0.934
5	NiveBayes_Word2Vec	0.523
6	XGBoost_CountVectorizer	0.958
7	XGBoost_TF-IDF	0.955
8	XgBoost_Word2Vec	0.523
9	LSTM_CountVectorizer	0.494
10	LSTM_TF-IDF	0.493
11	LSTM_Word2Vec	0.525

Fig 29. AUC-ROC comparison of all models on the Queensland Dataset

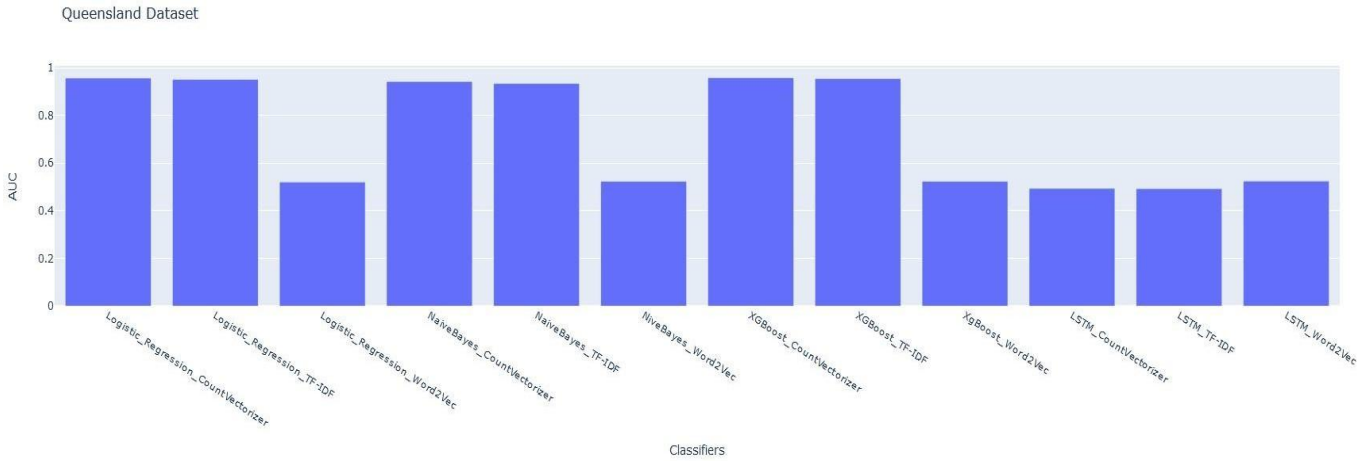

Fig 30. Graphical representation of the AUC-ROC of all models on the Queensland Dataset

Comparison

```
In [63]: nepal_df = {'Classifiers':['Logistic_Regression_CountVectorizer','Logistic_Regression_TF-IDF','Logistic_Regression_Word2Vec',
                                    'NaiveBayes_CountVectorizer','NaiveBayes_TF-IDF','NiveBayes_Word2Vec','XGBoost_CountVectorizer','XGBoost_TF-IDF',
                                        'XgBoost_Word2Vec','LSTM_CountVectorizer','LSTM_TF-IDF','LSTM_Word2Vec'],
                                     'Accuracy': [nepal_logit_cnt,nepal_logit_tfidf,acc_nepal_word2vec_logit,
                                                  nepal_nb_cnt,nepal_nb_tfidf,acc_nepal_nb_word2vec,
                                                  nepal_xgb_cnt,nepal_xgb_tfidf,acc_nepal_xgb_word2vec,
                                                 nepal_lstm_cnt,tfidf_lstm_acc,word2vec_lstm_acc],
                       }
```

```
In [71]: nepal_df = pd.DataFrame(nepal_df).round(3)
         nepal_df
```

Out[71]:

	Classifiers	Accuracy
0	Logistic_Regression_CountVectorizer	0.772
1	Logistic_Regression_TF-IDF	0.773
2	Logistic_Regression_Word2Vec	0.526
3	NaiveBayes_CountVectorizer	0.770
4	NaiveBayes_TF-IDF	0.761
5	NiveBayes_Word2Vec	0.495
6	XGBoost_CountVectorizer	0.760
7	XGBoost_TF-IDF	0.745
8	XgBoost_Word2Vec	0.525
9	LSTM_CountVectorizer	0.497
10	LSTM_TF-IDF	0.773
11	LSTM_Word2Vec	0.524

Fig 31. Comparison of Accuracy of all models on the Nepal Dataset

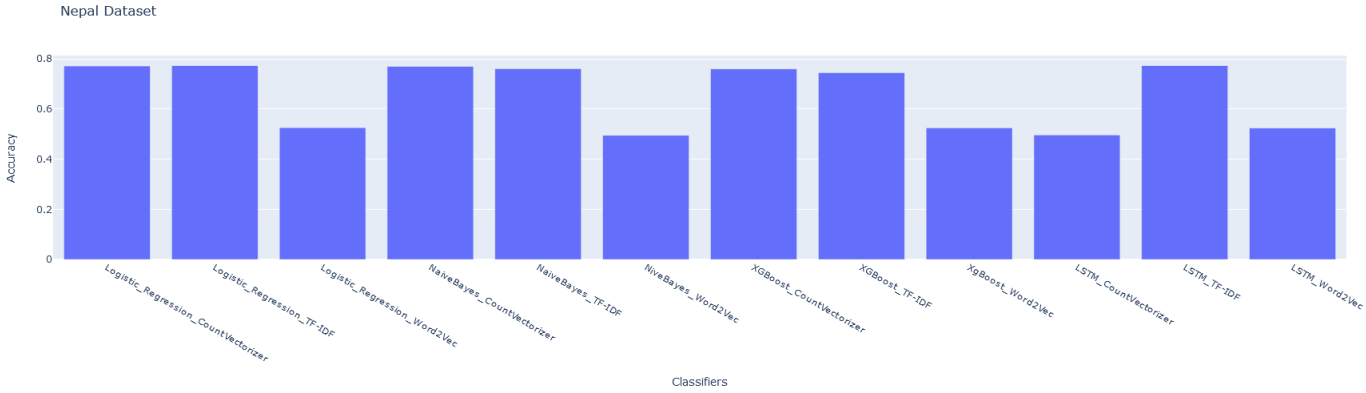

Fig 32. Graphical representation of the accuracy of all models on the Nepal Dataset

AUC Comparison

```
In [66]: nepal_df_auc = {'Classifiers':['Logistic_Regression_CountVectorizer','Logistic_Regression_TF-IDF','Logistic_Regression_Word2Vec',
                                       'NaiveBayes_CountVectorizer','NaiveBayes_TF-IDF','NiveBayes_Word2Vec','XGBoost_CountVectorizer','XGBoost_TF-IDF',
                                           'XgBoost_Word2Vec','LSTM_CountVectorizer','LSTM_TF-IDF','LSTM_Word2Vec'],
                                         'AUC': [nepal_logit_cnt_auc,nepal_logit_tfidf_auc,nepal_logit_word2vec_auc,
                                                 nepal_nb_cnt_auc,nepal_nb_tfidf_auc,nepal_nb_word2vec_auc,
                                                 nepal_xgb_cnt_auc,nepal_xgb_tfidf_auc,nepal_xgb_word2vec_auc,
                                                lstm_cnt_auc,lstm_tfidf_auc,lstm_word2vec_auc,]}
```

```
In [67]: nepal_df_auc = pd.DataFrame(nepal_df_auc)
         nepal_df_auc
```

Out[67]:

	Classifiers	AUC
0	Logistic_Regression_CountVectorizer	0.769
1	Logistic_Regression_TF-IDF	0.772
2	Logistic_Regression_Word2Vec	0.503
3	NaiveBayes_CountVectorizer	0.770
4	NaiveBayes_TF-IDF	0.760
5	NiveBayes_Word2Vec	0.516
6	XGBoost_CountVectorizer	0.757
7	XGBoost_TF-IDF	0.743
8	XgBoost_Word2Vec	0.502
9	LSTM_CountVectorizer	0.490
10	LSTM_TF-IDF	0.846
11	LSTM_Word2Vec	0.519

Fig 33. Comparison of AUC-ROC of all models on the Nepal Dataset

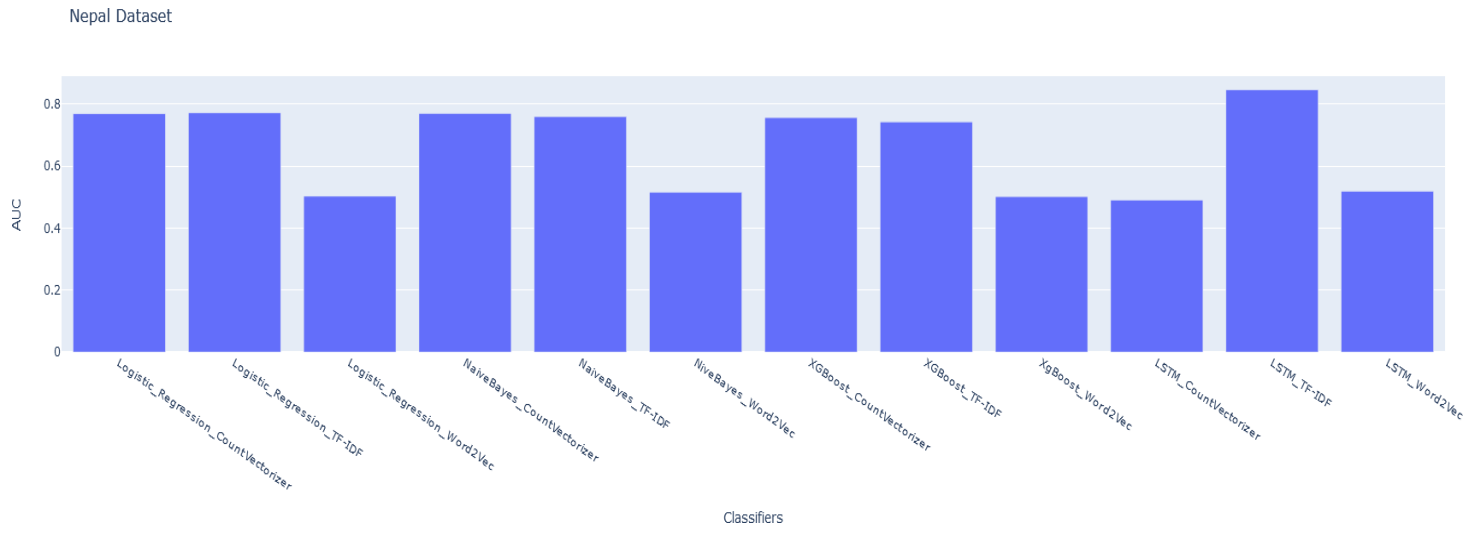

Fig 34. Graphical representation of AUC-ROC of all models on the Nepal Dataset

Comaprison

In [72]:
```
crisis_df = {'Classifiers':['Logistic_Regression_CountVectorizer','Logistic_Regression_TF-IDF','Logistic_Regression_Word2Vec',
                      'NaiveBayes_CountVectorizer','NaiveBayes_TF-IDF','NiveBayes_Word2Vec','XGBoost_CountVectorizer','XGBoost_TF-IDF',
                          'XgBoost_Word2Vec','LSTM_CountVectorizer','LSTM_TF-IDF','LSTM_Word2Vec'],
                       'Accuracy': [crisis_logit_cnt,crisis_logit_tfidf,acc_crisis_word2vec_logit,
                                    crisis_nb_cnt,crisis_nb_tfidf,acc_crisis_nb_word2vec,
                                    crisis_xgb_cnt,crisis_xgb_tfidf,acc_crisis_xgb_word2vec,
                                   crisis_lstm_acc,tfidf_lstm_acc,word2vec_lstm_acc],
             }
```

In [73]:
```
crisis_df = pd.DataFrame(crisis_df)
crisis_df
```

Out[73]:

	Classifiers	Accuracy
0	Logistic_Regression_CountVectorizer	0.861
1	Logistic_Regression_TF-IDF	0.852
2	Logistic_Regression_Word2Vec	0.605
3	NaiveBayes_CountVectorizer	0.828
4	NaiveBayes_TF-IDF	0.774
5	NiveBayes_Word2Vec	0.600
6	XGBoost_CountVectorizer	0.846
7	XGBoost_TF-IDF	0.839
8	XgBoost_Word2Vec	0.605
9	LSTM_CountVectorizer	0.815
10	LSTM_TF-IDF	0.598
11	LSTM_Word2Vec	0.604

Fig 35. Comparison of Accuracy of all models on the Crisis Dataset

AUC Comparison

```
In [75]: crisis_df_auc = {'Classifiers':['Logistic_Regression_CountVectorizer','Logistic_Regression_TF-IDF','Logistic_Regression_Word2Vec',
                        'NaiveBayes_CountVectorizer','NaiveBayes_TF-IDF','NiveBayes_Word2Vec','XGBoost_CountVectorizer','XGBoost_TF-IDF',
                            'XgBoost_Word2Vec','LSTM_CountVectorizer','LSTM_TF-IDF','LSTM_Word2Vec'],
                         'AUC': [crisis_logit_cnt_auc,crisis_logit_tfidf_auc,crisis_logit_word2vec_auc,crisis_nb_cnt_auc,
crisis_nb_tfidf_auc,crisis_nb_word2vec_auc,crisis_xgb_cnt_auc,crisis_xgb_tfidf_auc,crisis_xgb_word2vec_auc,
lstm_word2vec_auc,lstm_tfidf_auc,lstm_cnt_auc]}

In [76]: crisis_df_auc = pd.DataFrame(crisis_df_auc)
         crisis_df_auc
```

Out[76]:

	Classifiers	AUC
0	Logistic_Regression_CountVectorizer	0.857
1	Logistic_Regression_TF-IDF	0.837
2	Logistic_Regression_Word2Vec	0.510
3	NaiveBayes_CountVectorizer	0.802
4	NaiveBayes_TF-IDF	0.725
5	NiveBayes_Word2Vec	0.511
6	XGBoost_CountVectorizer	0.845
7	XGBoost_TF-IDF	0.840
8	XgBoost_Word2Vec	0.509
9	LSTM_CountVectorizer	0.513
10	LSTM_TF-IDF	0.493
11	LSTM_Word2Vec	0.889

Fig 36. Comparison of AUC-ROC of all models on the Crisis Dataset

Embedded Model

```
In [70]: from sklearn.ensemble import VotingClassifier
         model1 = LogisticRegression()
         model2 = MultinomialNB()
         model3 = XGBClassifier()
         model = VotingClassifier(estimators=[('lr', model1), ('nb', model2),('xgb',model3)], voting='hard')
         model.fit(xtrain_count_queens, train_y_queens)
         model.score(xtest_count_queens,test_y_queens).round(3)

Out[70]: 0.959
```

Fig 37. Embedded model with accuracy on Queensland Dataset

Embedded Model

```
In [69]: from sklearn.ensemble import VotingClassifier
         model1 = LogisticRegression()
         model2 = MultinomialNB()
         model3 = XGBClassifier()
         model = VotingClassifier(estimators=[('lr', model1), ('nb', model2),('xgb',model3)], voting='hard')
         model.fit(xtrain_count_nepal, train_y_nepal)
         model.score(xtest_count_nepal,test_y_nepal).round(3)

Out[69]: 0.779
```

Fig 38. Embedded model with accuracy on Nepal Dataset

Embedded Model

```
In [78]: from sklearn.ensemble import VotingClassifier
         model1 = LogisticRegression()
         model2 = MultinomialNB()
         model3 = XGBClassifier()
         model = VotingClassifier(estimators=[('lr', model1), ('nb', model2), ('xgb', model3)], voting='hard')
         model.fit(xtrain_count_crisis, train_y_crisis)
         model_score = model.score(xtest_count_crisis, test_y_crisis).round(3)
         model_score

Out[78]: 0.862
```

Fig 39. Embedded model with accuracy on Crisis Dataset