Requirements :

```
pip install keras
pip install tensorflow
pip install sci-kit learn
pip install nltk
pip install wordcloud
```

Dataset description :

generic_id	created_at	complaint_status_id	voteup_count	city_id	ward_id	category_id	sub_category_id	civic_agency_id	title	description	location
W01730C147452	We are facing multiple issues related to road ...	4	28	2	946	15.0	66	22.0	08-05-2016 09:45	Potholes, Illegal Parking and Movement of Heav...	NSS Palkar Road, Asalfa Village, Ghatkopar Wes...
W0960C153580	This is a really bad road. we need it cleaned ...	4	2	2	943	6.0	155	22.0	05-06-2016 17:11	Bad road now with garbage	21st Rd, MIDC Industrial Estate, Pandit Dinday...
W01760C153649	This needs to be fixed asap.	2	40	2	946	21.0	84	25.0	05-06-2016 21:16	Traffic issue that needs immediate attention	5, JVLR, IIT Area, Powai, Mumbai, Maharashtra ...
W01590C157072	We observed frequently heavy traffic at dahisa...	2	7	2	933	21.0	84	25.0	16-06-2016 19:42	Traffic jam at Dahisar check naka	Western Express Hwy, Diamond Industrial Estate...
W02080C164464	A pothole in front of Chand Shahwli Baba Darga...	3	19079	2	945	15.0	66	22.0	13-07-2016 12:57	Pothole on Pipeline Road	Pipe Line Rd, Powai, IIT Area, Powai, Mumbai, ...

More than 9000 entries, we chose the two relevant rows **: category_id** and **description**.

Files :

- m.h5 : contains the saved model structure and weights which have been trained on the corpus of complaint texts
- tokenizer.pickle : pickle file containing tokens mapping words in the corpus to numbers

Approach :

Text - processing using NLTK and tensorflow

Preprocessing steps :

1) Drop nan values where no category specified
2) Drop non existent category values
3) Remove stop words; eg and,the,a
4) Lemmatize all Words to reduce corpus

```
0       potholes  illegal parking and movement of heav...
1                              bad road now with garbage
2           traffic issue that needs immediate attention
3                     traffic jam at dahisar check naka
4                               pothole on pipeline road
                              ...
9293                                 garbage on foot path
9294                               garbage near foot path
9295    please add sound barrier along all the suburba...
9296                                 garbage on foot path
9297                                 garbage on foot path
Name: description, Length: 8434, dtype: object
```

Tokenization and padding :

1. Split into train test dataset
2. Fit train sentences to tokenizer
3. Convert words to tokens and select 800 most frequent tokens
4. Add padding to make all sequences to same length

Machine Learning

1. Build Tensorflow model with word embeddings
2. Train model for 10 epochs
3. Evaluate and observe accuracy

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, embedding_dim, input_length=len(X_train[0])),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(6, activation='relu'),
    tf.keras.layers.Dense(23, activation='sigmoid')
])
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 14, 100)           150000
_________________________________________________________________
flatten_1 (Flatten)          (None, 1400)              0
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 8406
_________________________________________________________________
dense_3 (Dense)              (None, 23)                161
=================================================================
Total params: 158,567
Trainable params: 158,567
Non-trainable params: 0
_________________________________________________________________
```

Result :

```
num_epochs = 10
model.fit(X_train, y_train, epochs=num_epochs,validation_data=(X_test, y_test))
```

```
Train on 6747 samples, validate on 1687 samples
Epoch 1/10
6747/6747 [==============================] - 1s 114us/sample - loss: 2.3337 - acc: 0.0566 - val_loss: 1.6771 - val_acc: 0.0913
Epoch 2/10
6747/6747 [==============================] - 1s 75us/sample - loss: 1.3487 - acc: 0.0963 - val_loss: 1.3187 - val_acc: 0.0913
Epoch 3/10
6747/6747 [==============================] - 1s 91us/sample - loss: 1.0318 - acc: 0.0963 - val_loss: 0.9635 - val_acc: 0.0913
Epoch 4/10
6747/6747 [==============================] - 1s 76us/sample - loss: 0.8473 - acc: 0.0963 - val_loss: 0.9426 - val_acc: 0.0913
Epoch 5/10
6747/6747 [==============================] - 1s 94us/sample - loss: 0.6953 - acc: 0.3308 - val_loss: 0.4026 - val_acc: 0.8619
Epoch 6/10
6747/6747 [==============================] - 1s 77us/sample - loss: 0.2224 - acc: 0.8871 - val_loss: 0.3724 - val_acc: 0.8755
Epoch 7/10
6747/6747 [==============================] - 1s 93us/sample - loss: 0.1839 - acc: 0.9046 - val_loss: 0.3778 - val_acc: 0.8785
Epoch 8/10
6747/6747 [==============================] - 1s 77us/sample - loss: 0.1592 - acc: 0.9109 - val_loss: 0.3845 - val_acc: 0.8809
Epoch 9/10
6747/6747 [==============================] - 1s 96us/sample - loss: 0.1406 - acc: 0.9231 - val_loss: 0.3987 - val_acc: 0.9004
Epoch 10/10
6747/6747 [==============================] - 1s 77us/sample - loss: 0.1249 - acc: 0.9591 - val_loss: 0.3928 - val_acc: 0.9158
```

Overall Accuracy : 93.06%

F1 - Score : 91%

Can be further improved with more data.