Evaluation of Recurrent Neural Networks for Detecting Injections in API Requests

In online application HTTP, UDP API (application programming interface) are used to transfer data in a network and sometime some malicious programmers can intercept API call and modify their values to perform injection activities. To detect such malicious activity all existing algorithms were utilizing only SQL injection and propose paper author enhancing algorithm to predict different injections like XML, JSON and SQL.

SQL injection example is

Select \* from login where username=’1=1’

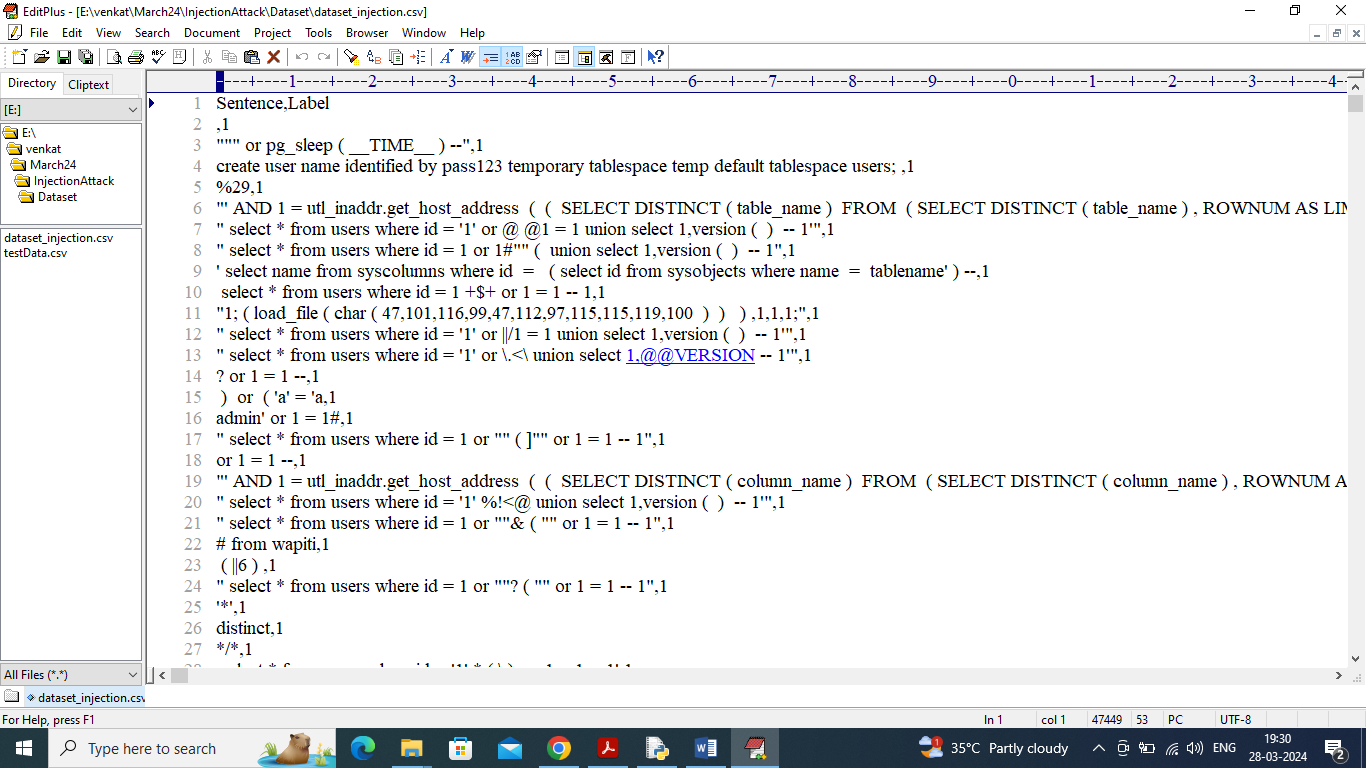
In above query username value will be modified as 1=1 and query executor will evaluate 1 = 1 which returns true and query get executed.

To detect above injections author utilizing various Recurrent Neural Networks (RNN) algorithms like Vanilla RNN, GRU RNN and LSTM RNN. RNN comes in two variants such as UNI variants and Bidirectional Variants. Among both variants Bidirectional RNN is giving best accuracy so we are implementing Bidirectional RNN algorithms.

In Bidirectional variants we are developing LSTM, GRU and Vanilla RNN and in all 3 algorithms GRU is giving best accuracy. Each algorithm performance is evaluated in terms of accuracy, precision, recall and FSCORE.

Univariants will optimize or filter features in forward direction and Bi-directional will filter features in both forward and backward direction so it will have more optimized features which result into best accuracy.

To train and test all algorithm performance we have used below injection dataset



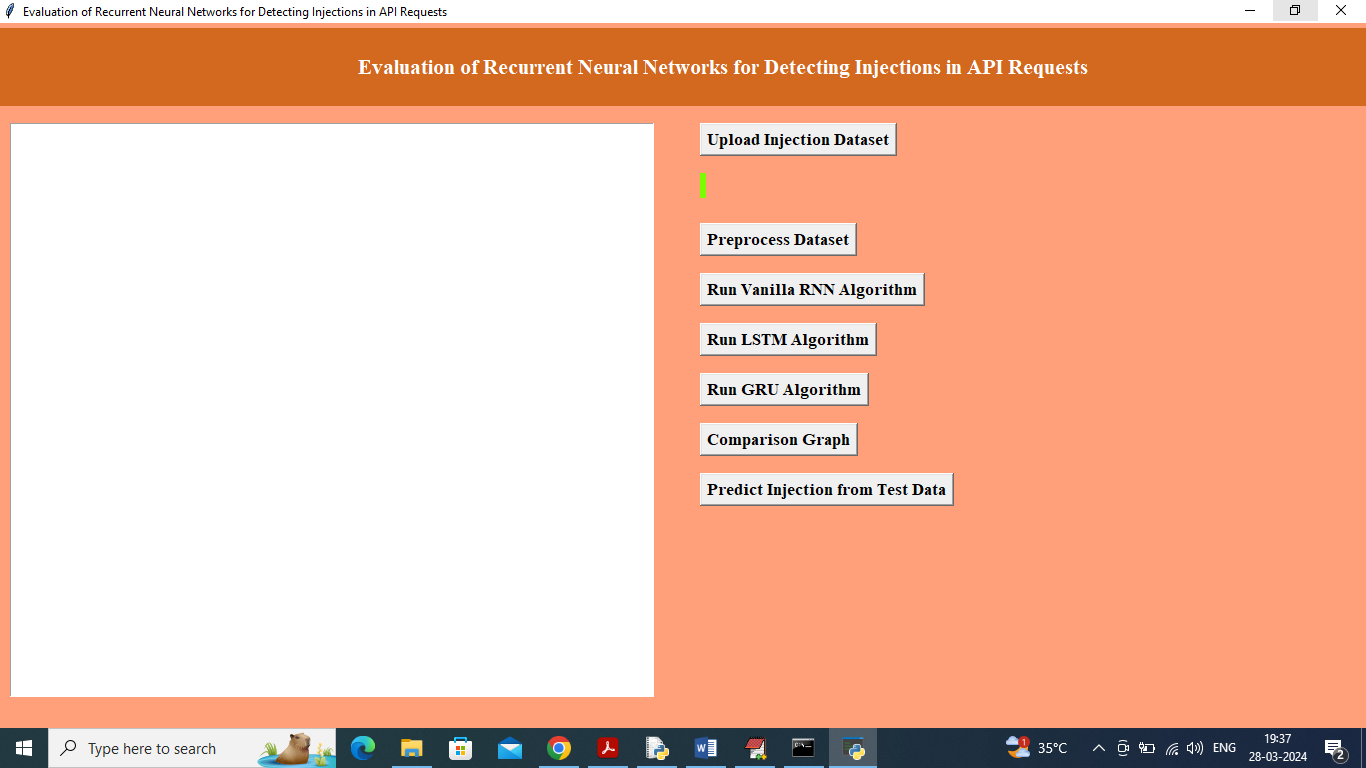
In above dataset first row contains dataset column names and remaining rows contains SQL injection, XML and JSON query statements and second column contains class label as 0 (Normal), 1 (SQL Injection) and 2 (XML/JSON Injections). So by using above dataset will train and test each algorithm performance.

To implement this project we have designed following modules

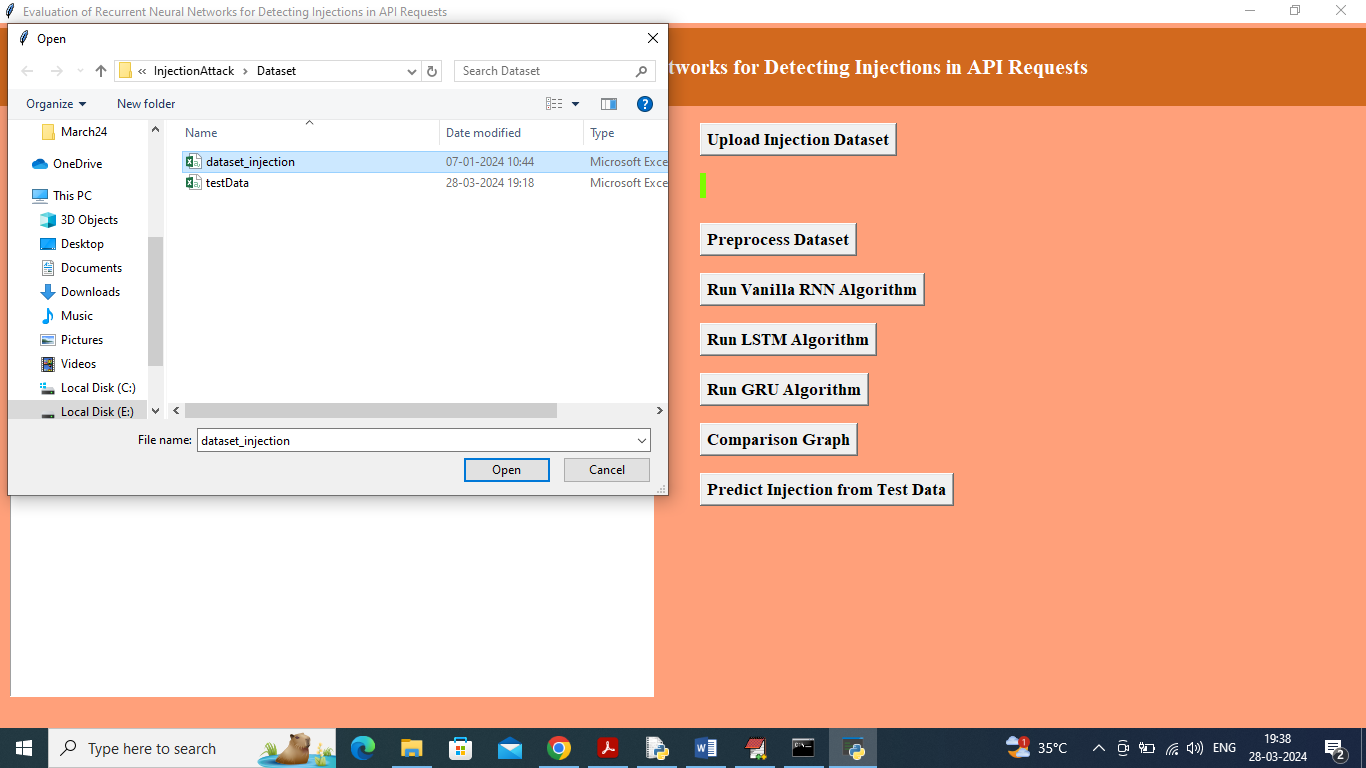
1. Upload Injection Dataset: using this module will upload and display dataset values and then visualize different injection graph available in dataset
2. Pre-process Dataset: in this module we will convert all sentences into numeric vector which will replace each words with its average frequency and then split dataset into train and test where application using 80% dataset for training and 20% for testing.
3. Run Vanilla RNN Algorithm: 80% training data will be input to this algorithm to train a model and 20% test data will be applied on this model to calculate prediction accuracy
4. Run LSTM RNN Algorithm: 80% training data will be input to this algorithm to train a model and 20% test data will be applied on this model to calculate prediction accuracy
5. Run GRU RNN Algorithm: 80% training data will be input to this algorithm to train a model and 20% test data will be applied on this model to calculate prediction accuracy
6. Comparison Graph: will plot comparison graph between all algorithms
7. Predict Injection from Test Data: will upload test data and then GRU will predict type of injection found in test data

SCREEN SHOTS

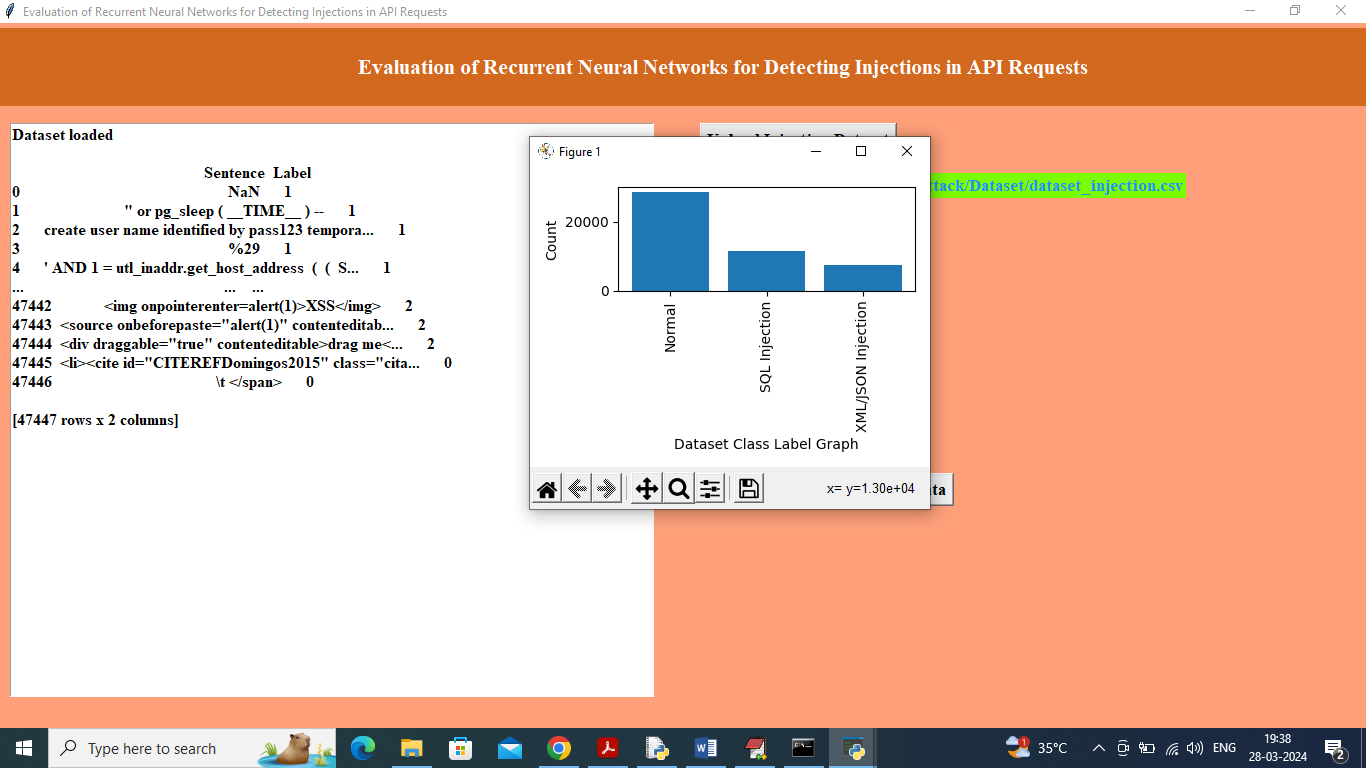
To run project double click on run.bat file to get below screen



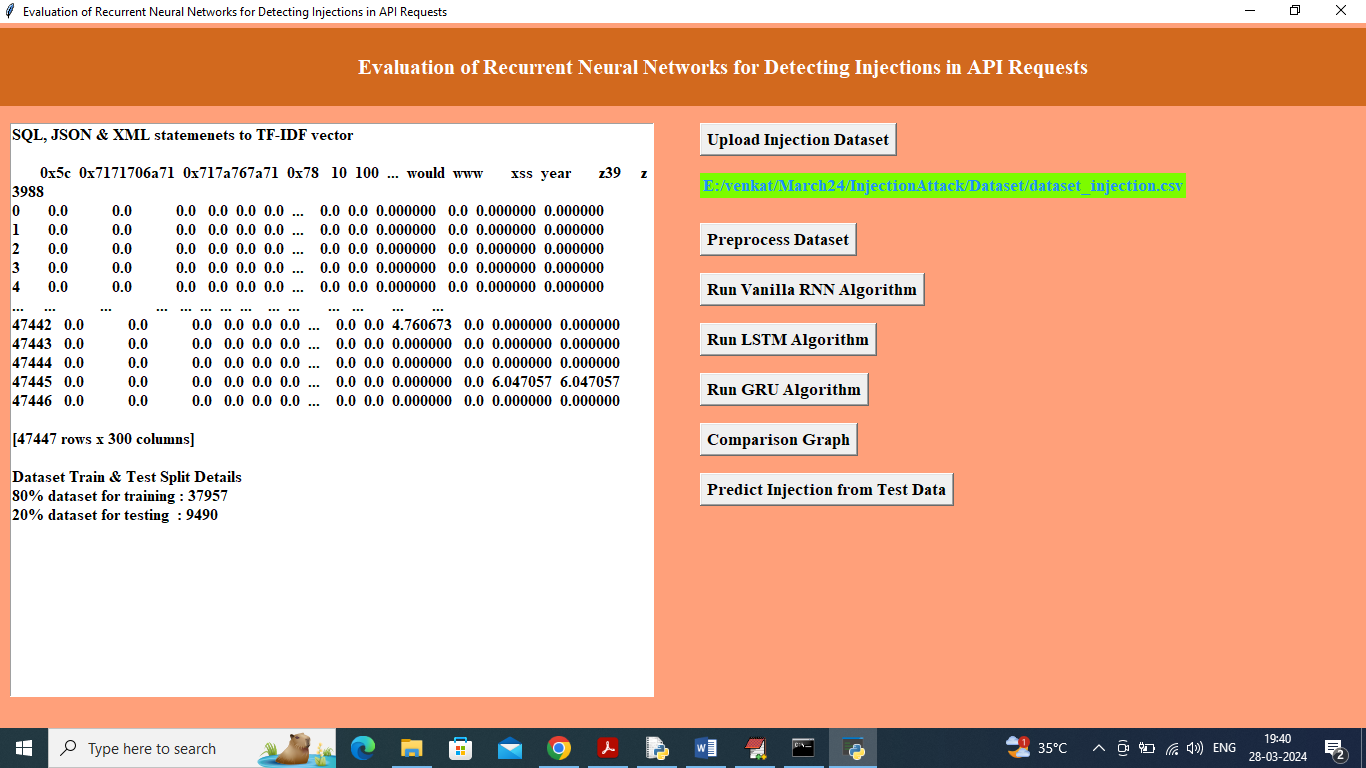
In above screen click on ‘Upload Injection Dataset’ button to upload dataset and get below page



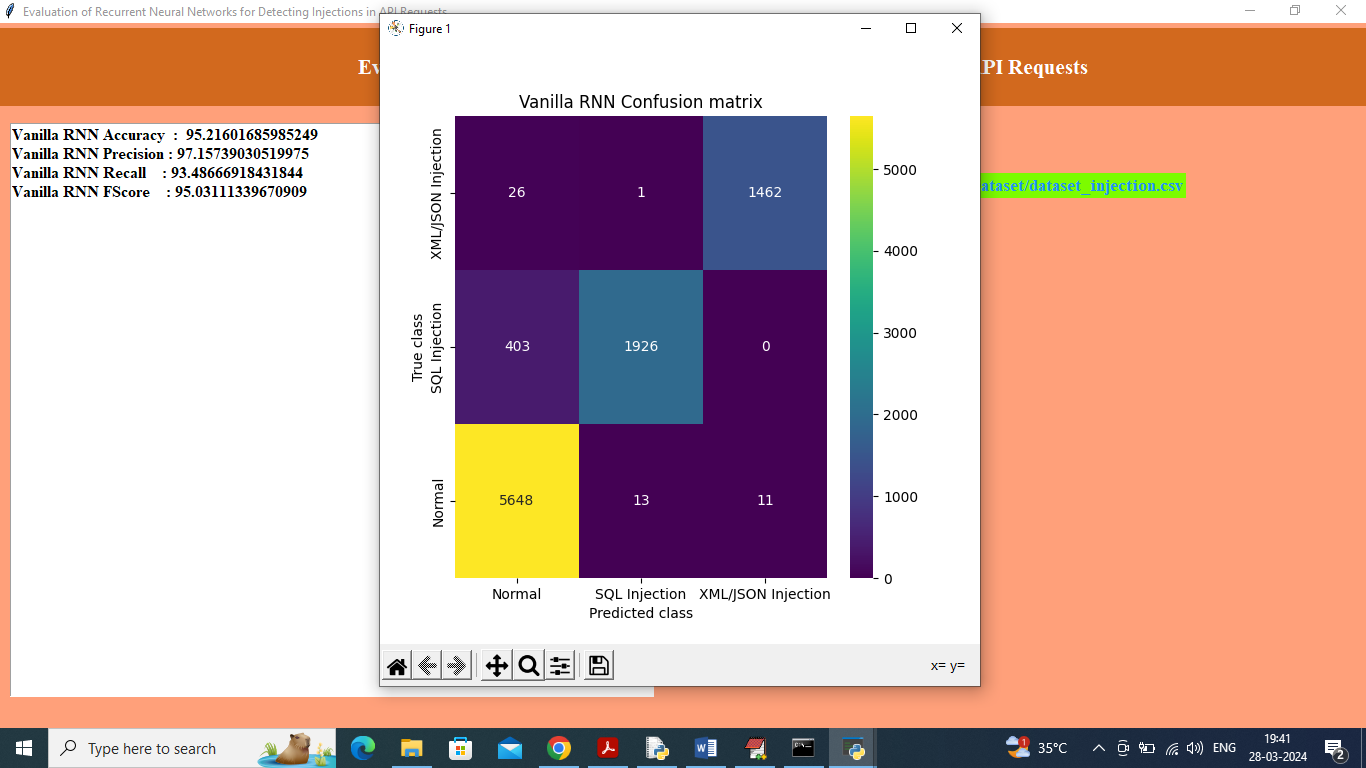
In above screen selecting and uploading dataset file and then click on ‘Open’ button to load dataset and get below page



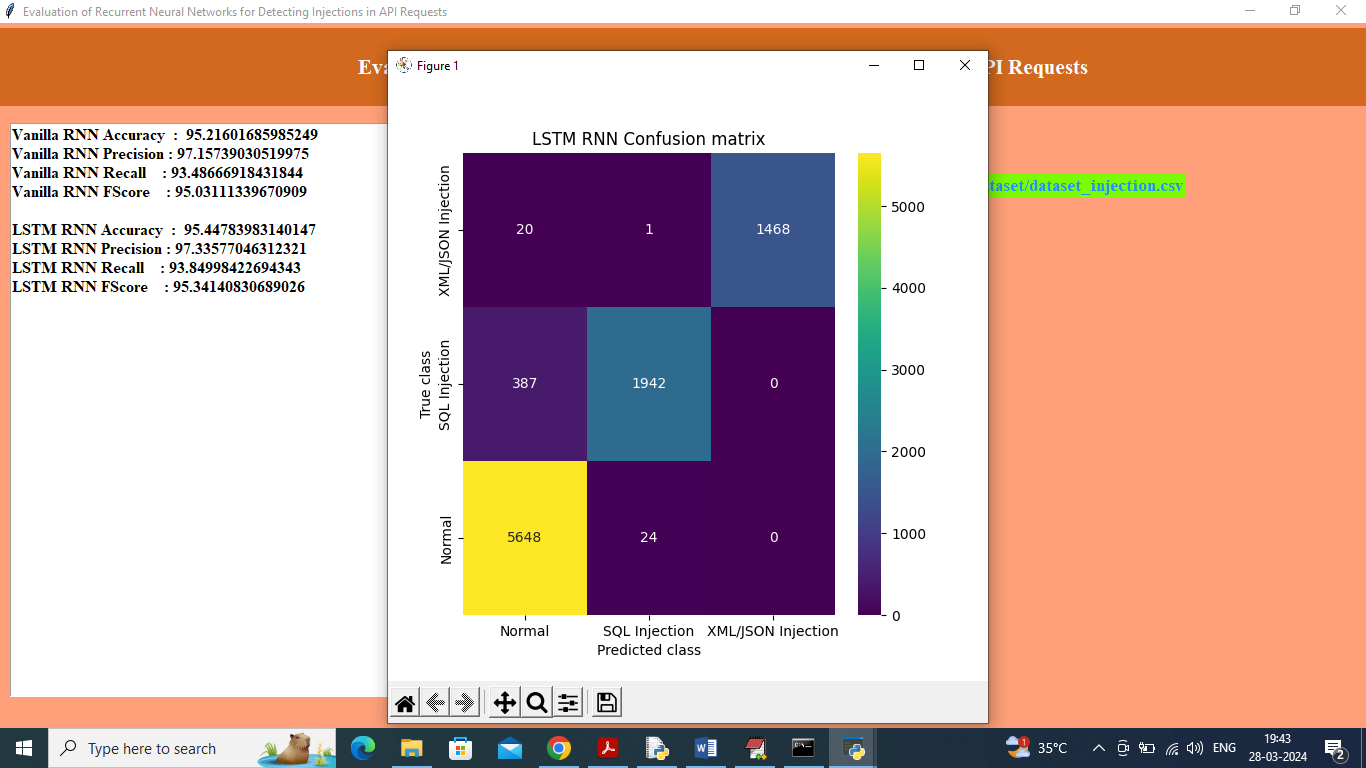
In above screen can see dataset loaded and in graph x-axis represents injection type and y-axis represents number of injection found in that type and now click on ‘Pre-process Dataset’ button to convert dataset into numeric vector and split into train and test and then will get below output



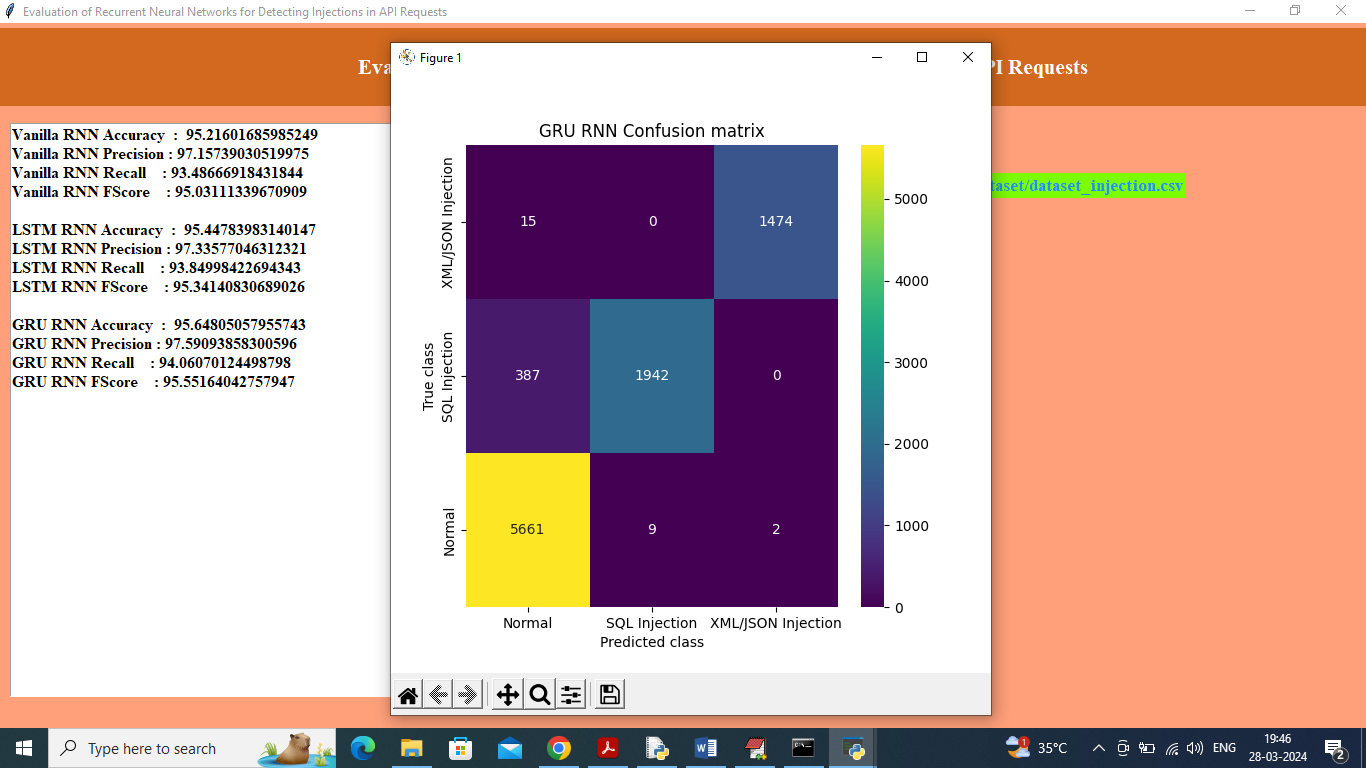
In above screen can see each word is replaced with average frequency and in last lines can see train and test split size and now click on ‘Run Vanilla Algorithm’ button to train Vanilla algorithm and get below output



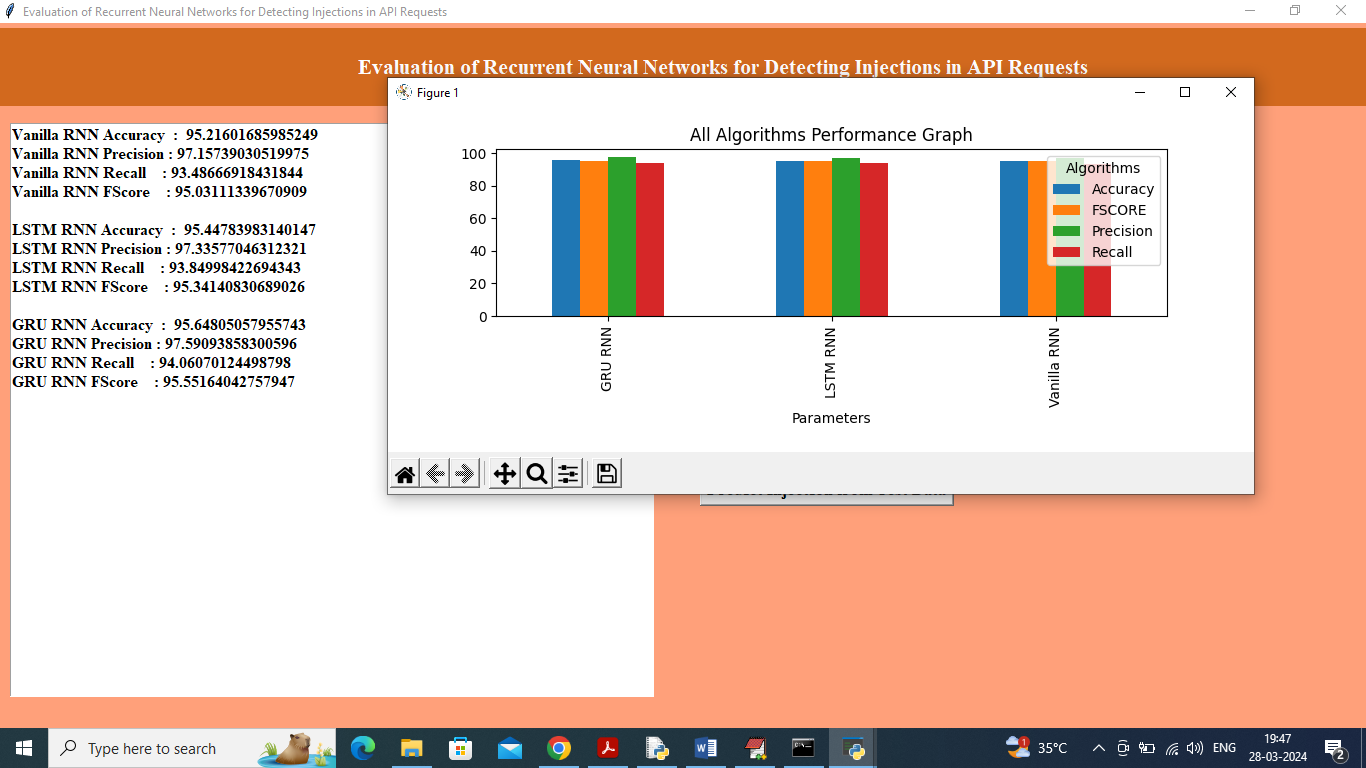
In above screen Vanilla RNN got 95.21% accuracy and can see other metrics like precision, recall and FSCORE and in confusion matrix graph x-axis represents ‘Predicted Injection Type’ and y-axis represents True Injection Type and then all different colour boxes in diagnol represents correct prediction count and remaining all blue boxes contains incorrect prediction count which are very few. Now click on ‘Run LSTM Algorithm’ button to get below output



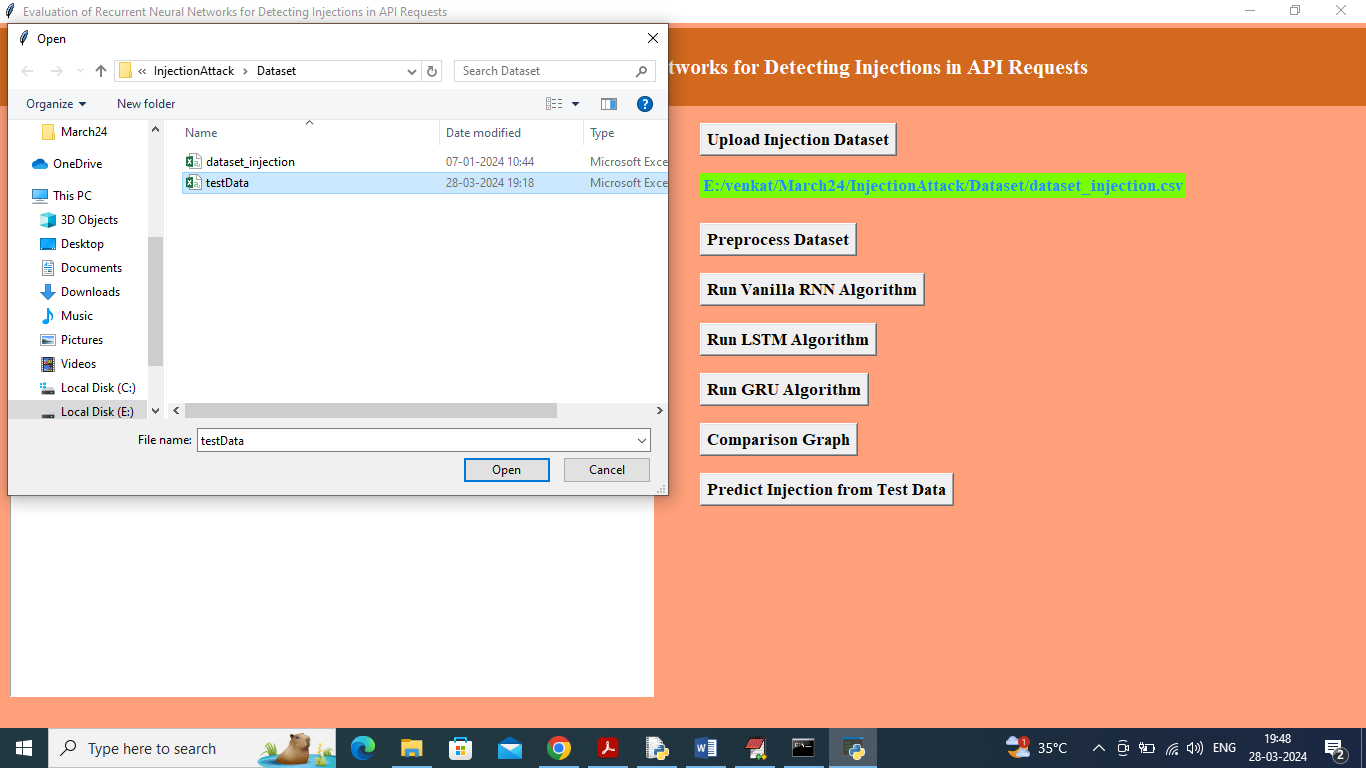
In above screen LSTM got 95.44% accuracy and can see other metrics also and now click on ‘Run GRU Algorithm’ button to train GRU and get below output



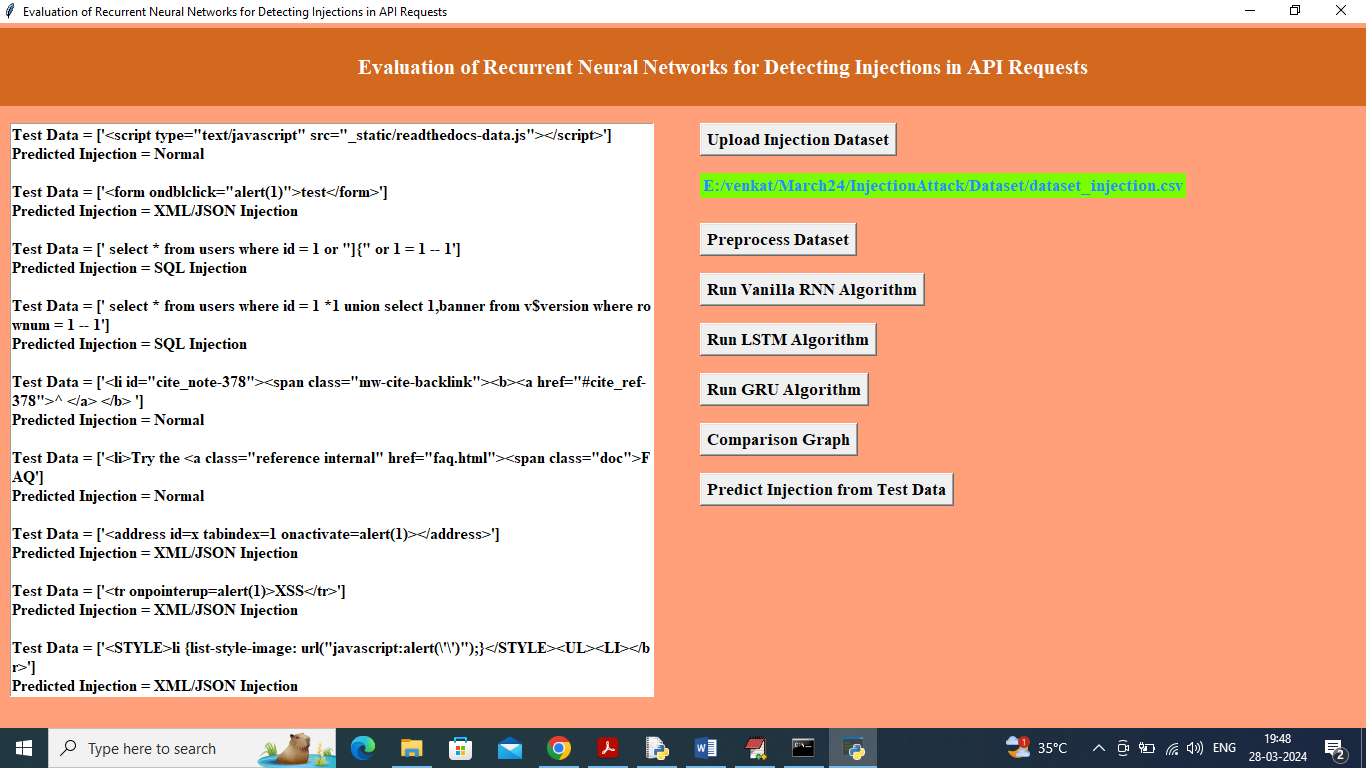
In above screen GRU got 95.64% accuracy which is higher in all 3 algorithms and now click on ‘Comparison Graph’ button to get below graph



In above graph x-axis represents algorithm names and y-axis represents accuracy and other metrics in different colour bars and in all algorithms GRU got high accuracy and now click on ‘Predict Injection from Test Data’ button to upload test data and get below output



In above screen selecting and uploading ‘test data’ file and then click on ‘Open’ button to get below prediction output



In above screen in each line we can see SQL/XML/JSON statement and in next line can see predicted injected type.

Similarly by using above application we can predict different injection type