Time series prediction of COVID-19 by mutation rate analysis using recurrent neural network-based LSTM model

A PROJECT REPORT

Submitted by

CH.EN.U4AIE20021 HARISH B

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE

AMRITA SCHOOL OF ENGINEERING, CHENNAI

AMRITA VISHWA VIDYAPEETHAM

CHENNAI 601 103

ABSTRACT

The new coronavirus SARS-CoV-2, also known as COVID-19, has caused a global pandemic. This contagious RNA virus threatened the entire world. More than 7.9 million people have been infected as of June 15, 2021 with 432k deaths. This virus is capable of causing mutations in humans. Understanding the evolution of this virus and determining the risk of the emerging infectious disease requires accurate mutation rate determination. This research investigates the mutation rate of the entire genome sequence based on patient data. The obtained data is analysed independently to detect nucleotide and codon mutations. Furthermore, the mutation rate analysis shows that a large number of Thymine (T) and Adenine (A) nucleotides are mutated when compared to other nucleotides, although the mutation rate of codons is not frequent when compared to nucleotides. A recurrent neural network (RNN) based Long Short-Term Memory (LSTM) model was used to predict the future mutation rate. This model gives very less error it is around 1.03 in both the training and testing sets. The mutation of the 83rd patient has been predicted by this model.

1. Introduction

The illness caused by the SARS-CoV-2 is currently affecting the entire planet. In December 2019, there was an epidemic in Wuhan, China. On January 7th, the virus was discovered to be transferred through human-to-human transmission via droplets or direct touch. It has a 6.4-day incubation period and a 2.24-3.58 basic reproduction number. COVID-19 has spread rapidly around the globe since its discovery, prompting the World Health Organization to declare it as a worldwide pandemic on March 11, 2020. The SARS-CoV-2 is a human coronavirus that belongs to the Beta coronavirus genus. On January 10th, the full genomic sequence of (SARS-CoV-2) was found and entered into the NCBI GenBank.

The SARS-CoV-2 is a single-stranded positive-stranded RNA virus with a non-segmented nucleotide sequence. Although it is an RNA virus, the gene sequence has been presented as a DNA type for simplification. The SARS-CoV2 virus's genomic sequence is 79 percent and 50 percent identical to the SAR-CoV and MARS-CoV, respectively. During the replication of genetic information, SARS-CoV-2 performs mutation. The mutation develops as a result of mistakes in RNA copying to a new cell. There are three types of mutations: base substitution, insertion, and deletion. There are also some more categories in base substitutions: silent, nonsense, missense, and frameshift. The mutation rate is one of the most important factors for the assessment of the risk of emergent infectious diseases, like SARS-CoV-2.

An analysis of the Gene signature of SARS-CoV-2 virus has been performed and estimated the ancestry rate of the European genome from the reference population by applying a statistical tool. Different linear regression models were created by scientists. Finally, they used genome-wide association analysis (GWAS) to look at single-nucleotide polymorphisms (SNPs) linked to

SARS-CoV-2 viral infection in European and East Asian genomes. However, there are many studies on the genomic sequence in the current literature, but relatively few on the mutation rate. As a result, the current research aims to forecast SARS-CoV-2 mutation rates using a time series approach. Accurate data on virus mutation rates might be crucial in determining the best vaccination strategy. Using the NCBI GenBank dataset, we conducted a thorough investigation on this virus's mutation rate.

2. Literature Review

On the current pandemic situation due to COVID-19, many models were proposed within the span of a year. Most of the models include Machine learning and deep learning techniques for predictions, and mutation rates also. These models help in preparing vaccine and also being prepared for the upcoming dates. In this section some of the models were discussed.

A paper on predictions for covid-19 outbreak in India gave detailed explanation and prediction based on epidemiological models which is basically an exponential fit or logistic regression. Based on this model long and short-term predictions were made. The results showed much satisfactory for the Suspected-Infectious-Recovered model (SIR model).

Some papers proposed to show the effect of social distancing at the time of covid-19. Some mathematical models were proposed to establish relation between social distancing and spread of corona virus. Some models include Markov models for the representation of the dynamics of the epidemic. In this model the transition probabilities of a person being healthy to becoming ill and at last may lead to death based on Erlang distribution.

Time series analysis is indexing the data with respect to time. A method based on genetic programming was proposed for time series prediction. It is the improved version of the genetic algorithm. In this model new solutions are developed based on the computer algorithms rather than classical binary strings. The main advantage of this model is that we can get optimized linear regression models and most stable than when compared to ANN models. These algorithms work on the basis of two major data sets, confirmed cases and death cases.

Another model based on artificial intelligence was proposed for covid-19 genome analysis. In this method three models were used. First with the help of sequence pattern missing model the hidden patterns and their relationship in the genome were understood. Second with the help of sequence prediction model it was tested that whether the mutated virus patterns can be predicted from the previous ones. Third an algorithm was developed which basically calculates the mutation rate in the genome.

3. Dataset Analysis and Pre-processing

The full genomic sequence of SARS-CoV-2 can be found in the NCBI GenBank. The dataset was collected for 100 patients from India. Among the set of data available, we have filtered out the gene sequence and the date of collection. The date of collection varies from March 2020 to December 2020. Among the collected datasets few patients' gene sequences were duplicates. There are also few partial genes in the collected dataset. After filtering out such gene sequences, the final dataset consists of 82 patients' gene sequences. In this analysis we have only collected patient's gene sequences' whose country of origin is India.

4. Mutation Rate Analysis

Mutations are alterations in the DNA sequence that are a major source of biological diversity. These alterations can occur at a variety of levels and have a wide range of effects. A gene mutation occurs when one or more genes are altered. Some mutations can cause genetic disorders. Changes in single DNA bases, as well as minor intragenic deletions and rearrangements, are examples of gene mutations.

	T	C	A	G	
T	1. TTT	5. TCT	9. TAT	13. TGT	T
	2. TTC	6. TCC	10. TAC	14. TGC	C
	3. TTA	7. TCA	11. TAA	15. TGA	A
	4. TTG	8. TCG	12. TAG	16. TGG	G
C	17. CTT	21. CCT	25. CAT	29. CGT	T
	18. CTC	22. CCC	26. CAC	30. CGC	C
	19. CTA	23. CCA	27. CAA	31. CGA	A
	20. CTG	24. CCG	28. CAG	32. CGG	G
A	33. ATT	37. ACT	41. AAT	45. AGT	T
	34. ATC	38. ACC	42. AAC	46. AGC	C
	35. ATA	39. ACA	43. AAA	47. AGA	A
	36. ATG	40. ACG	44. AAG	48. AGG	G
G	49. GTT	53. GCT	57. GAT	61. GGT	T
	50. GTC	54. GCC	58. GAC	62. GGC	C
	51. GTA	55. GCA	59. GAA	63. GGA	A
	52. GTG	56. GCG	60. GAG	64. GGG	G

Fig.1.1 Codon Indexing

Original: **CTA AAA GAC TGC TAA CCG**

Converted: 19 43 58 14 11 24

Gene mutations and point mutations are commonly used interchangeably, despite the fact that they are genetically distinct events. Many factors can cause

a gene to mutate. When RNA attempts to duplicate genetic information from DNA, it may make an error, resulting in mutation.

Mutation can also be caused by errors in DNA replication, recombination, or chemical damage to DNA or RNA. There are generally three types of mutations: base substitutions, deletions, and insertions. We can determine the three types of substitution mutations from this dataset: silent, missense, and nonsense.

A silent mutation occurs when a codon is modified but the amino acid remains unchanged. A missense mutation occurs when the amino acid sequence changes as a result of the mutation. On the other hand, when a codon change produces a stop signal for gene translation, resulting in a non-functional protein is called nonsense mutation.

If the mutation is missense, it can be assumed that the change in nucleotide has altered protein generation, potentially changing the virus behaviour. Furthermore, determining the gene sequence of the remedy is difficult. The programme calculated the rate of missense nucleotide mutations. After utilizing this procedure, the numbers were converted to percentages using the mutation rate equation.

$$\textbf{Mutation Rate} = \left(\frac{\textbf{mutation}}{\textbf{lg x gs}}\right) \textbf{x } \textbf{100}$$

Here, Mutation Rate is the final output array, mutation is the out-put array sized 4×4 containing raw values after applying the algorithm, lg is the length of the dataset we are working on, which is 82 and g sic the length of reference gene which is 29903 for this dataset. In this process, we have calculated the nucleotide mutation rate for the prepared dataset.

Algorithm:

Input: Dataset with gene sequences in rows.

Output: A 4 x 4 mutation matrix.

1. **let** mutation[1:4,1:4] = 0
2. **for** i = 1 to (len (dataset)) **do**
3. **for** j = 1 to (len (ReferenceGene)) **do**
4. **let** D1 = dataset[i][j]
5. **let** D2 = reference[j]
6. **if** D1! = D2 **then**
7. mutation[D1][D2] ← mutation[D1]D2] + 1
8. **end if**
9. **end for**
10. **end for**

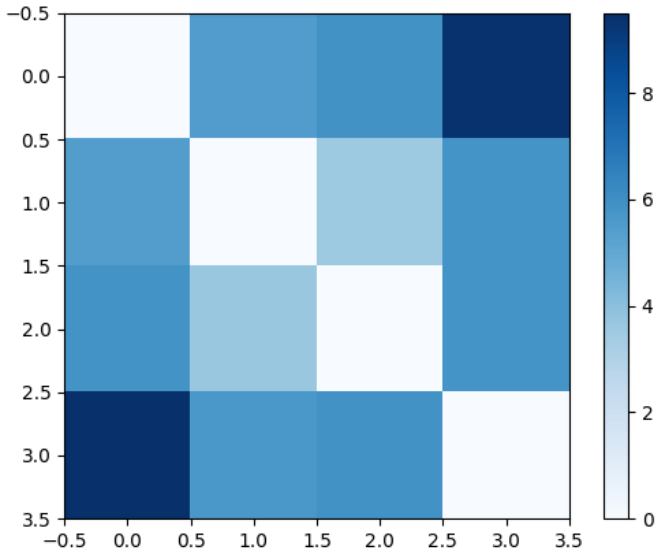

Fig 1.2. Mutation rate colormap for 4x4 mutation matrix

The above graph gives us the mutation rate of nucleotides in A, C, G, T. Denser colour represents more mutation rate. In the above graph the bottom left and

upper right boxes are in dark blue which represents that the mutation of A to T and vice versa is more when compared to mutation of remaining nucleotides.

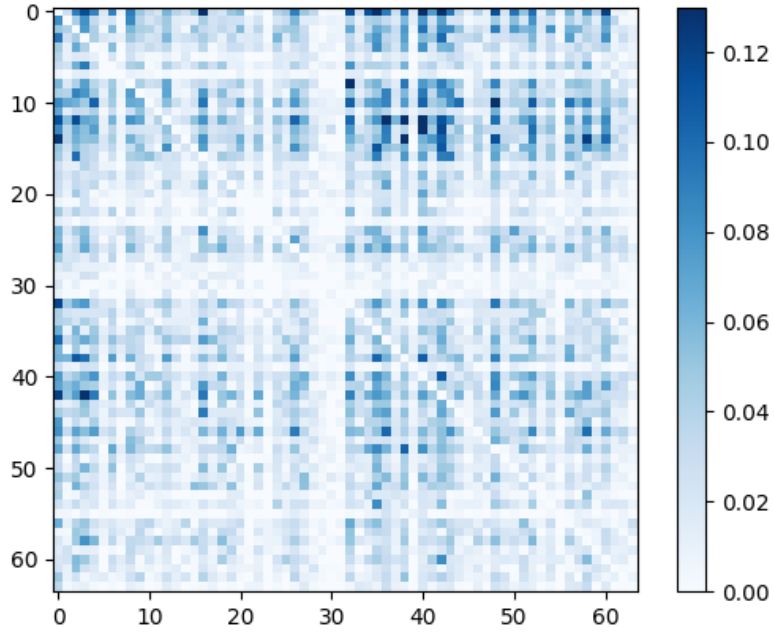

Fig 1.3. Codon mutation rate colormap for 64x64 mutation matrix

Furthermore, we have taken a reference sequence of length 29903, and a set of each patients' gene sequences, and we calculated mutation of these sequences with respect to the reference and plotted it as shown in above colormap. Denser colour represents more mutation rate.

5. Long Short-Term Memory

The figure 1.4 is an LSTM neuron in which there are two functions, sigmoid and tangent functions, and gates, which makes this method intelligent by storing the words in long sentences. Notice that in this method, all the methods are not stored. Only those words are stored, which will have an impact in the future. The neuron will be able to detect them from the training phase.

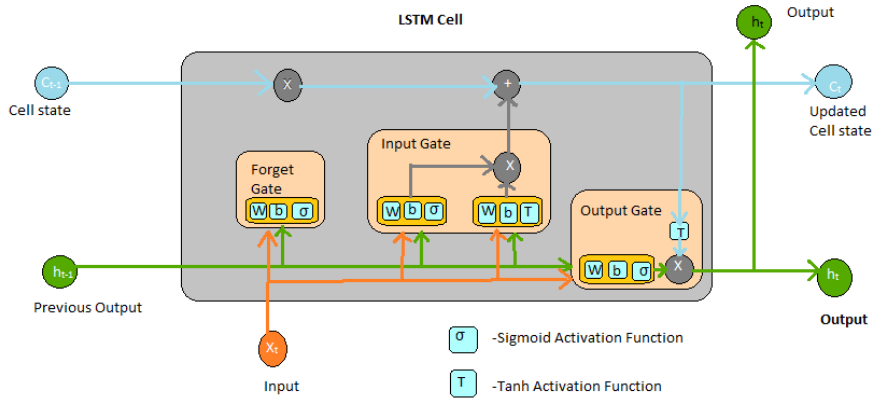

Fig 1.4. Long short-term memory

In an LSTM, there will be three gates, memory cell, and candidate memory. Candidate memory records the information of the present state, and in the memory cell, the basic information was updated by adding necessary data and removing unnecessary data.

A. Forget Gate:

This gate ensures how much information needed to be erased. This gate will perform element-wise multiplication between the cell state and the previous state. This gate will have a sigmoid function that ensures that output values are continuous and lie between 0 and 1.

B. Input Gate:

This gate uses two functional units. The first functional unit uses a tangent activation that outputs values between -1 and 1 and decides the change of the cell state. The latter uses a sigmoid activation function, and it is responsible for the magnitude of the change. After multiplying the results of these two functional units, the input gate adds the result to the cell state, and this completes the update of the cell state.

C. Output Gate:

Cell memory is squashed to lie between -1 and 1 using the function to compute the current hidden state. Output gate controls what to output in the hidden state at this point.

6. LSTM model:

Deep learning has made use of the Long Short-Term Memory network, which is a form of recurrent neural network (RNN). All of the data has been separated into two categories: training and testing, each accounting for 80 percent of the total. As a result, we have 64 training data and 18 testing data.

To train the dataset, an LSTM model was developed with keras, a Python deep learning API, and the structure is illustrated below. The input layer was given a 500-neuron set of training data to begin with. Then it went via a 250-neuron thick layer with a relu activation layer. Following that, a 0.25 dropout was used.

The summary method which is present in the keras library gives the composition of each layer. This involves the output shape parameters involved and how many layers involved int it. Following is the summary of our model.

Model: "sequential"

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 500)	1026000
dense (Dense)	(None, 250)	125250
dropout (Dropout)	(None, 250)	0
dense_1 (Dense)	(None, 150)	37650
dropout_1 (Dropout)	(None, 150)	0

```
dense_2 (Dense)              (None, 12)                1812
=================================================================
Total params: 1,190,712
Trainable params: 1,190,712
Non-trainable params: 0
_________________________________________________________________
```

With relu activation, another dense layer of 150 neurons was utilized. Then there's a 0.25 dropout. Finally, with the Adam optimizer, a density of 12 neurons was employed as an output layer. The Root Mean Square Error (RMSE) of this model is 0.09 in testing and 1.03 in training. The model was found to be working at the expected level after the training and testing phase.

So, we utilised the mutation rates of the previous 12 patients to estimate one future patient's mutation rate. If more continuous data from other locations and dates can be collected, this method can be used to calculate the mutation rate for a specific date in the future.

```
Epoch 70/70
2/2 [==============================] - 0s 13ms/step - loss: 1.0328
```

a) Loss of trained LSTM model

```
[5.6241508, 5.755442 , 9.502816 , 5.449258 , 3.4954212, 5.8204675,
 6.160357 , 3.7389784, 5.987761 , 9.541414 , 5.710477 , 6.073836 ]
```

b) Predicted Mutation rate

```
[5.52787346 5.87933318 9.33016754 5.42420493 3.5347624  5.81212587
 5.89238538 3.61167776 5.66498345 9.5375046  5.65829515 5.95160686]
```

c) Actual Mutation rate

5. Conclusion

As we can see in the output that this model is working accurately in calculating mutation rates with very less error. Based on the mutation rate, scientists can develop vaccines, medication at a faster rate than classical techniques. They can warn the governments in prior so that there will be time for taking necessary steps to prevent spreading of the virus. The accuracy of the LSTM model can be improved by adjusting the dense layers and neurons in it, or by increasing the epochs value. Thus, the proposed model works accurately and can be used for real time purpose.

6. Appendix

Python code for writing xml dataset from NCBI GenBank to excel sheet

```
import xml.etree.ElementTree as ET
import pandas as pd
import os

root_node = ET.parse('sequence.gbc.xml').getroot()

tags = root_node.findall('INSDSeq/INSDSeq_feature-
table/INSDFeature/INSDFeature_quals/INSDQualifier')
date = [ tag.find('INSDQualifier_value').text for tag in tags if tag.find('INS
DQualifier_name').text=="collection_date"]

tags = root_node.findall('INSDSeq')
sequence = [ tag.find('INSDSeq_sequence').text for tag in tags ]

df = pd.DataFrame({'Sequence':sequence, 'Date':date})
writer = pd.ExcelWriter('sequence.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='Sheet1', index=False)
writer.save()
```

In this code the xml dataset at NCBI GenBank need to be converted into excel sheet for further calculations and predictions to happen easily. All the necessary

libraries are imported and next the xml file is parsed with the help of command parser(). After that we are getting all the tags with help of findall() command. In further steps we separating the sequence and the date from the XML file and with the help of pandas library functions we are appending into a excel file.

Python code for converting Nucleotides to Codon

```
import xml.etree.ElementTree as ET
import pandas as pd

df = pd.read_excel("reference.xlsx", usecols="A")
reference = df.iloc[0][0]

df1 = pd.read_excel("sequence.xlsx", usecols="A")
dataset = [df1.iloc[i][0] for i in range(0,82)]
df2 = pd.read_excel("sequence.xlsx", usecols="B")
date = [df2.iloc[i][0] for i in range(0,82)]

CodonIndex = {'TTT':1,'TTC':2,'TTA':3,'TTG':4,'TCT':5,'TCC':6,'TCA':7,'TCG':8,
'TAT':9,'TAC':10,'TAA':11,'TAG':12,'TGT':13,'TGC':14,'TGA':15,'TGG':16,
            'CTT':17,'CTC':18,'CTA':19,'CTG':20,'CCT':21,'CCC':22,'CCA':23,'CC
G':24,'CAT':25,'CAC':26,'CAA':27,'CAG':28,'CGT':29,'CGC':30,'CGA':31,'CGG':32,
            'ATT':33,'ATC':34,'ATA':35,'ATG':36,'ACT':37,'ACC':38,'ACA':39,'AC
G':40,'AAT':41,'AAC':42,'AAA':43,'AAG':44,'AGT':45,'AGC':46,'AGA':47,'AGG':48,
            'GTT':49,'GTC':50,'GTA':51,'GTG':52,'GCT':53,'GCC':54,'GCA':55,'GC
G':56,'GAT':57,'GAC':58,'GAA':59,'GAG':60,'GGT':61,'GGC':62,'GGA':63,'GGG':64}

codon=[]

for seq in dataset:
    seq = seq.upper()
    n = len(seq)

    cd = []
    for i in range(0,n-3+1,3):
        cd.append(str(CodonIndex[seq[i:i+3]]))
    cd = "+".join(cd)
    codon.append(cd)

df = pd.DataFrame({'Sequence':codon, 'Date':date})
writer = pd.ExcelWriter('sequenceCodon.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='Sheet1', index=False)
writer.save()

seq = reference.upper()
n = len(seq)
```

```
rd = []
rcodon=[]
for i in range(0,n-3+1,3):
    rd.append(str(CodonIndex[seq[i:i+3]]))
rd = "+".join(rd)
rcodon.append(rd)

df = pd.DataFrame({'Sequence':codon})
writer = pd.ExcelWriter('referenceCodon.xlsx', engine='xlsxwriter')
df.to_excel(writer, sheet_name='Sheet1', index=False)
writer.save()
```

In this code conversion of Nucleotides to codons occurs. We are reading the reference file and the sequence file and getting the sequences from it and storing it data frames df1 and df2. A dictionary is defined in which is codon is connected to a number which is given in the above sections of this report. In next step two excel files are created with the respective codon number separated by '+' along with the date. One excel file is for reference and another is for other patients' sequence.

Python code for Mutation rate calculation

```
import numpy as np
import pandas as pd

df = pd.read_excel("reference.xlsx", usecols="A")
reference = df.iloc[0][0]

df1 = pd.read_excel("sequence.xlsx", usecols="A")
dataset = [df1.iloc[i][0] for i in range(0,82)]

mutation = np.zeros((4,4))
nuc = ["a", "c", "g", "t"]

for i in range(0,len(dataset)):
    for j in range(0,min(len(reference),len(dataset[i]))):
        D1 = dataset[i][j]
        D2 = reference[j]
        if D1!=D2:
            mutation[nuc.index(D1)][nuc.index(D2)] += 1

MutationRate = (mutation/(len(dataset)*len(reference)))*100
```

```
print(MutationRate)

import matplotlib.pyplot as p

cmap = p.imshow(MutationRate,interpolation='none',cmap="Blues",origin='upper')
p.colorbar()
p.show()
```

In this code the mutation rate of nucleotides is calculated. The reference and sequence files are read by the python and the sequences were stored in the data frames df and df1. In next step the mutation matrix is calculate with the algorithm discussed and next the mutation rate with the help of the formula given. A colormap is plotted for this mutation rate.

Python code for Codon mutation rate calculation

```
import numpy as np
import pandas as pd

df = pd.read_excel("referenceCodon.xlsx", usecols="A")
reference = df.iloc[0][0]
reference = reference.split("+")

df1 = pd.read_excel("sequenceCodon.xlsx", usecols="A")
dataset = [df1.iloc[i][0] for i in range(0,82)]
dataset = [codon.split("+") for codon in dataset]
mutation = np.zeros((64,64))

for i in range(0,len(dataset)):
    for j in range(0,min(len(reference),len(dataset[i]))):
        D1 = dataset[i][j]
        D2 = reference[j]
        if D1!=D2:
            mutation[int(D1)-1][int(D2)-1] += 1

MutationRate = (mutation/(len(dataset)*len(reference)))*100
print(MutationRate)

import matplotlib.pyplot as p

cmap = p.imshow(MutationRate,interpolation='None',cmap="Blues",origin='upper',
vmin=0,vmax=0.13)
p.colorbar()
p.show()
```

This is same as the code for mutation rate calculation but the only difference is that here we will be reading the codons which we created in second step and the output will be a 64x64 matrix.

Python code for LSTM prediction model

```
import numpy as np
import pandas as pd

df = pd.read_excel("reference.xlsx", usecols="A")
reference = df.iloc[0][0]

df1 = pd.read_excel("sequence.xlsx", usecols="A")
dataset = [df1.iloc[i][0] for i in range(0,82)]

nuc = ["a", "c", "g", "t"]

mt = []
for i in range(0,len(dataset)):
    mutation = np.zeros((4,4))
    for j in range(0,min(len(reference),len(dataset[i]))):
        D1 = dataset[i][j]
        D2 = reference[j]
        if D1!=D2:
            mutation[nuc.index(D1)][nuc.index(D2)] += 1
    MutationRate = (mutation/(1*len(reference)))*100
    mt.append(MutationRate)


mt = [m[~np.eye(m.shape[0],dtype=bool)].reshape(m.shape[0],-1) for m in mt]
mt = [m.flatten() for m in mt]

train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
train, test = mt[0:train_size], mt[train_size:len(dataset)]

features = list(train[i:i+12] for i in range(0,train_size-12))
labels = list(train[i] for i in range(12,train_size))

# establishing lstm
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout
from sklearn.metrics import mean_squared_error
```

```
model = Sequential()
model.add(LSTM(500))
model.add(Dense(250, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(12, activation='relu'))
model.compile(optimizer='adam',loss='mse')

hist = model.fit(np.array(features),np.array(labels),epochs=70)

t_features = list(test[i:i+11] for i in range(0,test_size-11))
t_labels = list(test[i] for i in range(11,test_size))
model.predict(np.array(t_features))
print(np.array(t_labels))
```

In this code we are getting all the mutation rates into a matrix and it is flattened. This matrix is broken down into training set and testing set. From the training set we are extracting features and labels each of size 12 x 12 Next a LSTM model is generated with the help of commands in keras with input features are 500 and output labels as250 which means that a dense layer of 250 neurons were created and next 0.25% of it get dropped out leaving 150 neurons dense layers and at last the number of neurons present will be 12 which means that the will be a 12 x 12 matrix. After that, taking Adam optimizer and loss function as rmse we are finally creating a model with epoch 70 and loss value around 1.3 which is satisfactory when compared to other machine learning models.

7. References

1. Pathan, Refat & Biswas, Munmun & Khandaker, Mayeen. (2020). Time Series Prediction of COVID-19 by Mutation Rate Analysis using Recurrent Neural Network-based LSTM Model. Chaos, Solitons & Fractals. 138. 110018. 10.1016/j.chaos.2020.110018.

2. Coronaviridae Study Group of the International Committee on Taxonomy of Viruses The species Severe acute respiratory syndrome-related coronavirus: classifying 2019-nCoV and naming it SARS-CoV-2. *Nat Microbiol.* 2020;5(4):536-544. doi: 10.1038/s41564-020-0695-z.

3. Graham R.L., Baric R.S. Recombination, reservoirs, and the modular spike: mechanisms of coronavirus cross-species transmission. *J Virol.* 2010;84(7):3134–3146.