# NEURAL NETWORK AND FUZZY LOGIC

## IA-1 REPORT

## TOPIC: HOUSEPRICEPREDICTION


GROUP MEMBERS

1813001 – ABHIRAM ANILKUMAR

1813009–PRATHAMESHDAPHAL

1813031 – SHUBH MODY

# INTRODUCTION

In this project, we will develop and evaluate the performance and the predictive power of a model trained and tested on data collected from houses in the state of Tamil Nadu.

The data is collected from the 7 major cities in Tamil Nadu which has houses of different build type like housing, commercial and others and several other parameters which affect the sales price of these houses.

Once we get a good fit, we will use this model to predict the sales price of a house located in these regions.

A model like this would be very valuable for a real estate agent who could make use of the information provided in a daily basis.

You can find the complete project, documentation and dataset on:

# □ DATASET

The dataset is the Chennai_house_price_prediction.csv file which is provided in the github repository. To import the dataset pandas and other libraries are imported for the entire preprocessing and graphical analysis, which is done using:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
import streamlit as st
from sklearn.linear_model import LinearRegression

```

The dataset has got 18 different features and 7109 different rows with the SALES_PRICE being the target variable. This is checked using the df.shape command

The overview of the dataset is given below

| AREA | INT_SQFT | T_MAINRC | _BEDROOI | BATHROC | N_ROOM | SALE_COND | ARK_FACI | BUILDTYPE | ITILITY_AVA | STREET | MZZONE | )S_ROOM | _BATHRO( | ; BEDROO | S_OVERAL | COMMIS | ALES_PRICI |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Karapakkam | 1004 | 131 | 1 | 1 | 3 | AbNormal | Yes | Commercial | AllPub | Paved | A | 4 | 3.9 | 4.9 | 4.33 | 144400 | 7600000 |
| Anna Nagar | 1986 | 26 | 2 | 1 | 5 | AbNormal | No | Commercial | AllPub | Gravel | RH | 4.9 | 4.2 | 2.5 | 3.765 | 304049 | 21717770 |
| Adyar | 909 | 70 | 1 | 1 | 3 | AbNormal | Yes | Commercial | ELO | Gravel | RL | 4.1 | 3.8 | 2.2 | 3.09 | 92114 | 13159200 |
| Velachery | 1855 | 14 | 3 | 2 | 5 | Family | No | Others | NoSewr | Paved | I | 4.7 | 3.9 | 3.6 | 4.01 | 77042 | 9630290 |
| Karapakkam | 1226 | 84 | 1 | 1 | 3 | AbNormal | Yes | Others | AllPub | Gravel | C | 3 | 2.5 | 4.1 | 3.29 | 74063 | 7406250 |
| Chrompet | 1220 | 36 | 2 | 1 | 4 | Partial | No | Commercial | NoSeWa | No Access | RH | 4.5 | 2.6 | 3.1 | 3.32 | 198316 | 12394750 |
| Chrompet | 1167 | 137 | 1 | 1 | 3 | Partial | No | Other | AllPub | No Access | RL | 3.6 | 2.1 | 2.5 | 2.67 | 33955 | 8488790 |

This also has a unique id for each row which can be dropped since it doesn't have any significance while predicting the model.

The features can be summarized as follows:

1. INT_SQFT: It is the total square feet area of a particular house.
2. DIST_MAINROAD: It is distance of a house from the main road.
3. N_BEDROOM: Number of bedrooms in a house
4. N_BATHROOM: Number of bathrooms in a house
5. N_ROOM: Total number of rooms in a house which is the cumulative sum of all the different type of rooms in a house
6. SALE_COND: The current condition in which the house is sold.
7. PARK_FACIL: Used to identify whether a house has a parking facility or not
8. BUILDTYPE: The purpose for which the property is used, like housing or commercial or office etc.
9. UTIL_AVAIL: The utilities available in the particular house or property.
10. STREET: The type of road leading to a particular property.
11. QS: Quality scores are given for bedrooms, bathrooms and for the total rooms in a house which can affect its sales price (QS_ROOMS, QS_BEDROOMS, QS_BATHROOMS)
12. COMMISSION: The commission for each house which highly depends on the sales price.

## PREPROCESSING

Inpreprocessingwecleanthedatathatiswefillorremovethemissingvalues, changethedatatypesand correcttheerrorsinspelling.Alsoduplicateitemsshould be removed since it would be redundant.

Usingthiscommandwegetthetotalmissingvalues inthedatasetandthedatatypeof each feature.

```
df.isnull().sum()
df.dtypes
```

These commands will give us output as:

```
In [8]: df.dtypes
Out[8]:
PRT_ID            object
AREA              object
INT_SQFT           int64
DIST_MAINROAD      int64
N_BEDROOM        float64
N_BATHROOM       float64
N_ROOM             int64
SALE_COND         object
PARK_FACIL        object
BUILDTYPE         object
UTILITY_AVAIL     object
STREET            object
MZZONE            object
QS_ROOMS         float64
QS_BATHROOM      float64
QS_BEDROOM       float64
QS_OVERALL       float64
COMMIS             int64
SALES_PRICE        int64
dtype: object
```

```
In [7]: df.isnull().sum()
Out[7]:
PRT_ID            0
AREA              0
INT_SQFT          0
DIST_MAINROAD     0
N_BEDROOM         1
N_BATHROOM        5
N_ROOM            0
SALE_COND         0
PARK_FACIL        0
BUILDTYPE         0
UTILITY_AVAIL     0
STREET            0
MZZONE            0
QS_ROOMS          0
QS_BATHROOM       0
QS_BEDROOM        0
QS_OVERALL       48
COMMIS            0
SALES_PRICE       0
dtype: int64
```

Thisshowsthatwehavesomemissingvalueswhichisneededtobetakencareof.

Filling out the missing values:

```
df['N_BEDROOM'].mode()
df['N_BEDROOM'].fillna(value=(df['N_BEDROOM'].mode()[0]),inplace=True)
```

Themissingvaluesofthebedroomisfilledwiththemodeofthebedroomfeature

```
for i in range(0,len(df)):
    if pd.isnull(df['N_BATHROOM'][i])==True:
        if (df['N_BEDROOM'][i]==1.0):
            df['N_BATHROOM'][i] = 1.0
        else:
            df['N_BATHROOM'][i] = 2.0
```

If there is only one bedroom then only one bathroom is required otherwise atleast two bathrooms are filled in the missing value space.

```
for i in range(0,len(df)):
    if pd.isnull(df['QS_OVERALL'][i])==True:
        df['QS_OVERALL'][i] = (df['QS_ROOMS'][i] + df['QS_BEDROOM'][i]+df['QS_BATHROOM'][i])/3
```

QS-OVERALL value is approximated as the average of the QS_ROOMS, QS_BEDROOM, QS_BATHROOM.

Therefore now the total missing values after preprocessing is:

```
In [13]: df.isnull().sum()
Out[13]:
AREA             0
INT_SQFT         0
DIST_MAINROAD    0
N_BEDROOM        0
N_BATHROOM       0
N_ROOM           0
SALE_COND        0
PARK_FACIL       0
BUILDTYPE        0
UTILITY_AVAIL    0
STREET           0
MZZONE           0
QS_ROOMS         0
QS_BATHROOM      0
QS_BEDROOM       0
QS_OVERALL       0
COMMIS           0
SALES_PRICE      0
dtype: int64
```

Duplicates in the dataset is removed by the command df.drop_duplicates( )

The spelling error correction is done below using the following code. Spelling errors can increase the unique values of each feature and create imbalance of data due to which the accuracy of the model is very much affected.

```
CorrectNames = ['AREA','SALE_COND','PARK_FACIL','BUILDTYPE','UTILITY_AVAIL','STREET','MZZONE']

for i in CorrectNames:
    print('*********Value Counts*********')
    print(df[i].value_counts())
    print('Unique values are:',df[i].nunique())

df['AREA'].replace({'Chrompt':'Chrompet','Chormpet':'Chrompet','Chrmpet':'Chrompet',
                    'TNagar':'T Nagar','Ana Nagar':'Anna Nagar','Ann Nagar':'Anna Nagar',
                    'Karapakam':'Karapakkam','Velchery':'Velachery','Adyr':'Adyar',
                    'KKNagar':'KK Nagar'},inplace=True)
df['AREA'].value_counts()

df['BUILDTYPE'].replace({'Comercial':'Commercial','Other':'Others'},inplace=True)

df['SALE_COND'].replace({'Adj Land':'AdjLand','Ab Normal':'AbNormal','Partiall':'Partial',
                         'PartiaLl':'Partial'},inplace=True)
df['SALE_COND'].value_counts()

df['PARK_FACIL'].replace({'Noo':'No'},inplace=True)
df['PARK_FACIL'].value_counts()

df['UTILITY_AVAIL'].replace({'All Pub':'AllPub'},inplace=True)
df['UTILITY_AVAIL'].value_counts()

df['STREET'].replace({'Pavd':'Paved','NoAccess':'No Access'},inplace=True)
df['STREET'].value_counts()
```

.replacefunctionisusedwhichconsistsofadictionarywhichreplacestheincorrect word with correct word.

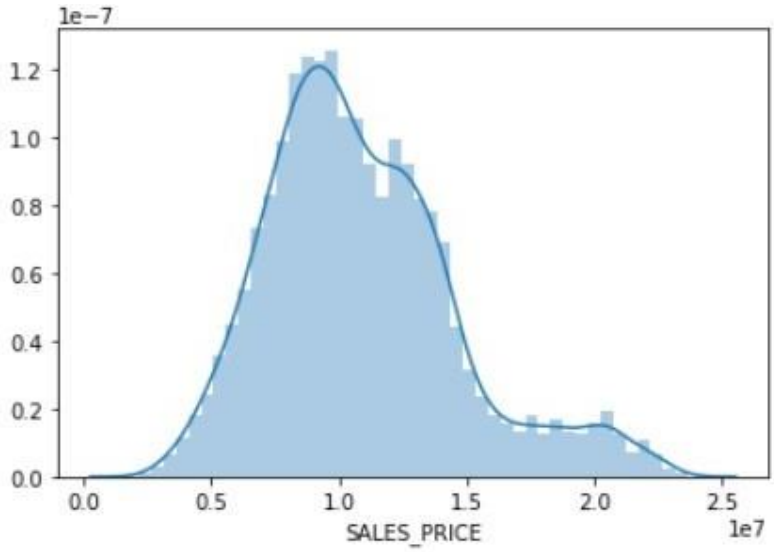
The data types changes using the .astype function which is implemented below:

```
df = df.astype({'N_BATHROOM':'int64','N_BEDROOM':'int64'})
```

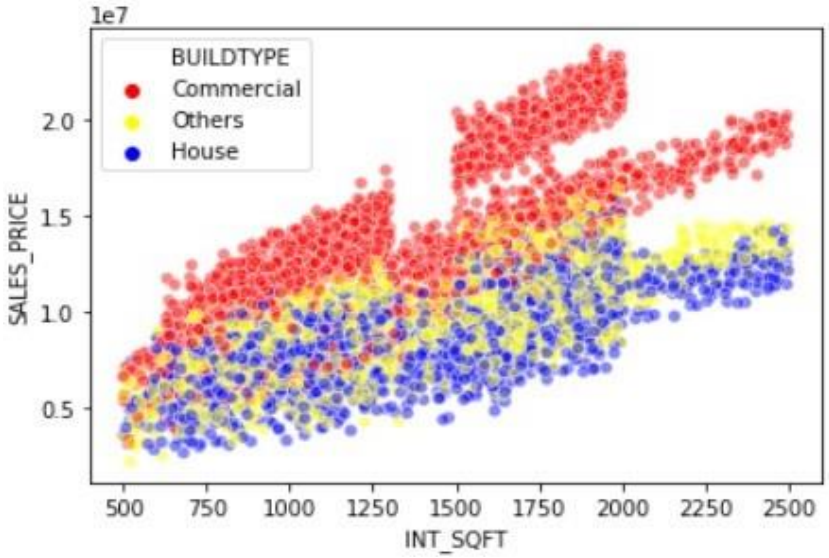
ThedatatypesofN_BEDROOMandN_BATHROOMarechangedfromfloattointegers.

## Graphical analysis:

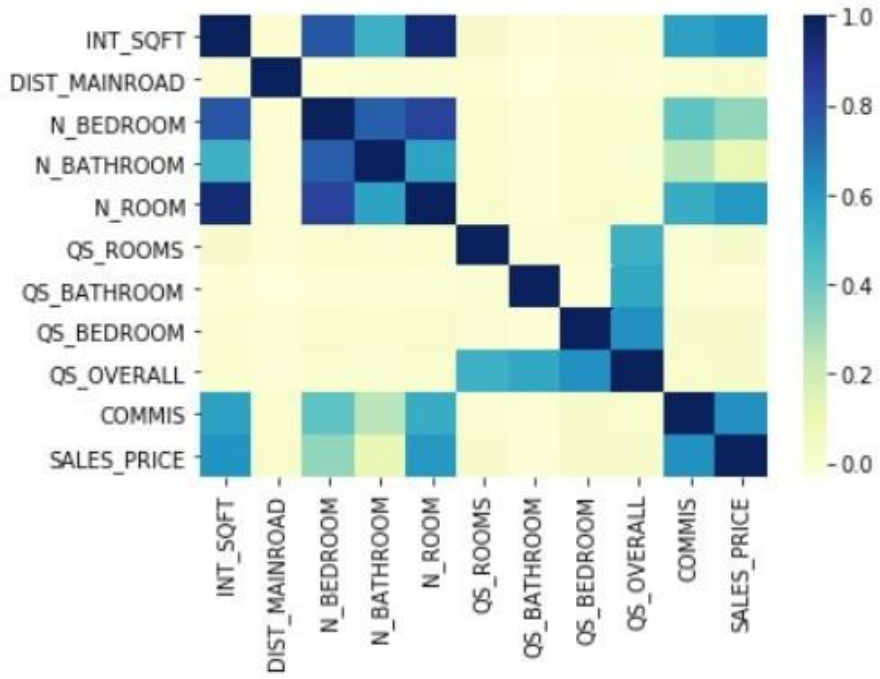
Thefollowingaresomeofthegraphicalanalysismadefromthedatasetusingthe seaborn and matplotlib libraries.



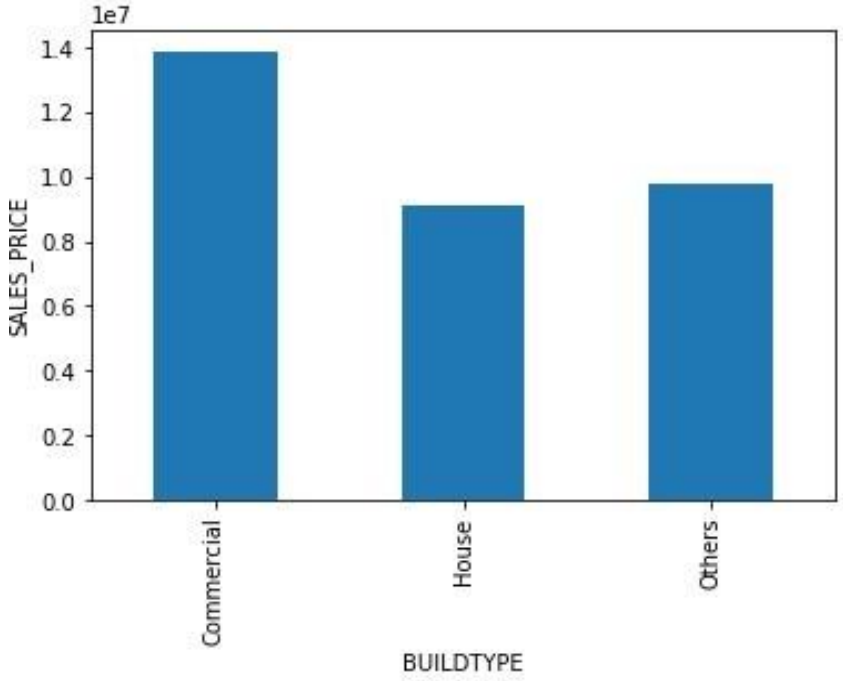

Thisfollowingisahistogramwithdensityplotwhichshowsthatthesalespriceof most of the house falls in the 1 crore mark.

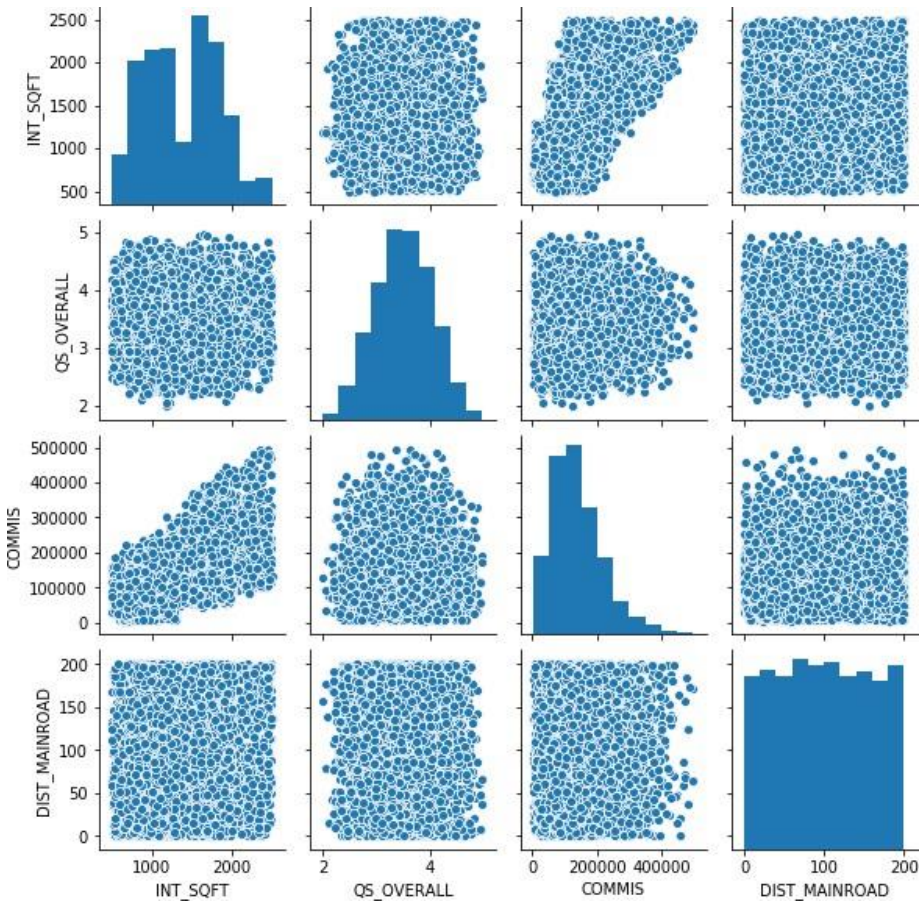



Thisscatterplotshowstherelationbetweensquarefeetandsalespriceofthehouse with the hue as buildtype.




Heatmapshowthecorrelationbetweenvariousfeaturesofthedataset.






Following is a pair plot which shows relationship between various continuous variables

## Label encoding:

Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning

Code :

```
enc=LabelEncoder()
df.iloc[:,0]=enc.fit_transform(df.iloc[:,0])
df.iloc[:,6]=enc.fit_transform(df.iloc[:,6])
df.iloc[:,7]=enc.fit_transform(df.iloc[:,7])
df.iloc[:,8]=enc.fit_transform(df.iloc[:,8])
df.iloc[:,9]=enc.fit_transform(df.iloc[:,9])
df.iloc[:,10]=enc.fit_transform(df.iloc[:,10])
df.iloc[:,11]=enc.fit_transform(df.iloc[:,11])
```

**An overview off the dataset after one-hot encoding and normalization**

| Index | AREA | INT_SQFT | T_MAINRC | _BEDROO | BATHROC | N_ROOM | ALE_CONI | ARK_FACI | BUILDTYPI | ILITY_AVA | STREET | MZZONE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 1004 | 131 | 1 | 1 | 3 | 0 | 1 | 0 | 0 | 2 | 0 |
| 1 | 1 | 1986 | 26 | 2 | 1 | 5 | 0 | 0 | 0 | 0 | 0 | 3 |
| 2 | 0 | 909 | 70 | 1 | 1 | 3 | 0 | 1 | 0 | 1 | 0 | 4 |
| 3 | 6 | 1855 | 14 | 3 | 2 | 5 | 2 | 0 | 2 | 3 | 2 | 2 |
| 4 | 4 | 1226 | 84 | 1 | 1 | 3 | 0 | 1 | 2 | 0 | 0 | 1 |
| 5 | 2 | 1220 | 36 | 2 | 1 | 4 | 4 | 0 | 0 | 2 | 1 | 3 |
| 6 | 2 | 1167 | 137 | 1 | 1 | 3 | 4 | 0 | 2 | 0 | 1 | 4 |
| 7 | 6 | 1847 | 176 | 3 | 2 | 5 | 2 | 0 | 0 | 0 | 0 | 5 |
| 8 | 2 | 771 | 175 | 1 | 1 | 2 | 1 | 0 | 2 | 3 | 2 | 5 |
| 9 | 6 | 1635 | 74 | 2 | 1 | 4 | 0 | 0 | 2 | 1 | 1 | 2 |
| 10 | 2 | 1203 | 78 | 2 | 1 | 4 | 1 | 1 | 0 | 0 | 1 | 5 |
| 11 | 2 | 1054 | 143 | 1 | 1 | 3 | 4 | 0 | 2 | 3 | 0 | 5 |

# DEVELOPING THEMODEL

Splitting and Normalizing the data:

```
X = df.drop('SALES_PRICE',axis=1).values
y = df['SALES_PRICE'].values
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=101)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test=scaler.fit_transform(X_test)
```

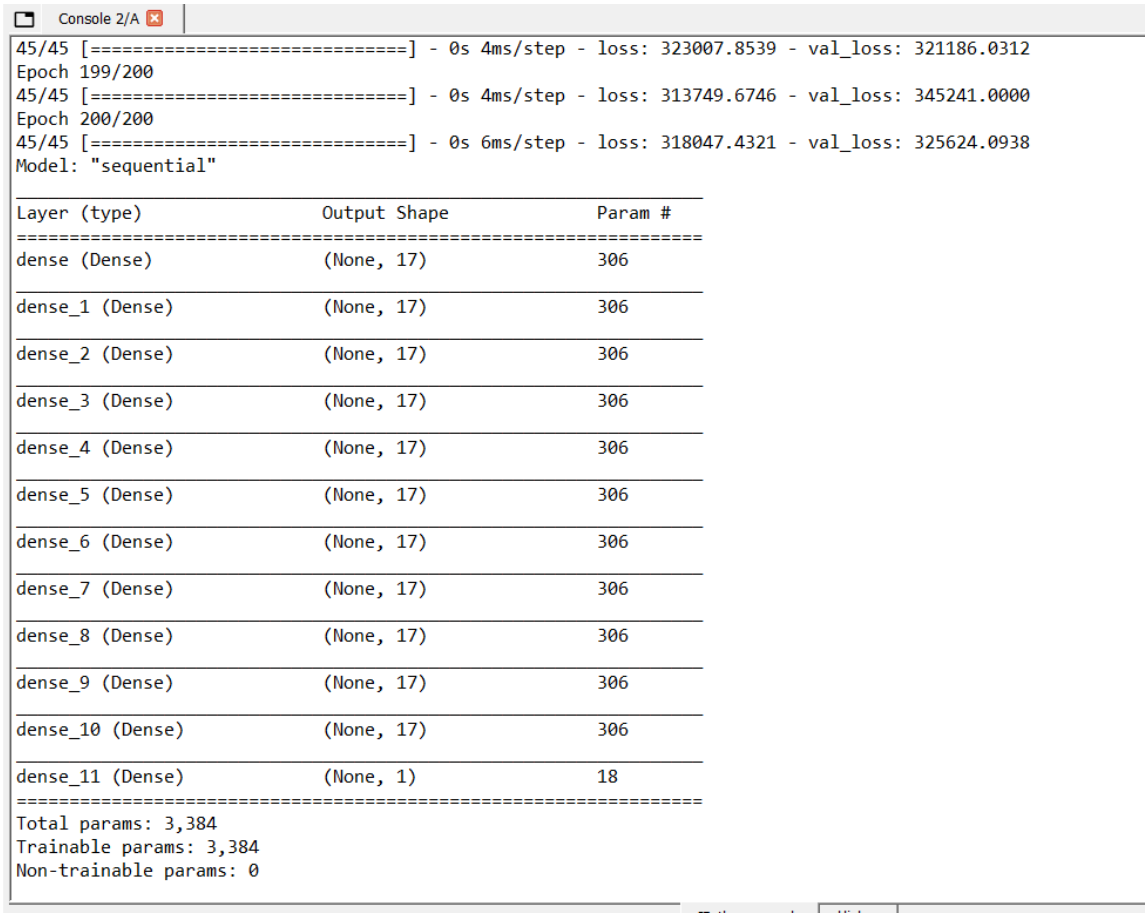
Data after Normalizing:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0.666667 | 0.437719 | 0.19 | 0.333333 | 1 | 0.5 | 0.25 |
| 1 | 0.333333 | 0.206103 | 0.355 | 0 | 0 | 0.25 | 0.75 |
| 2 | 0.166667 | 0.743372 | 0.545 | 0.333333 | 0 | 0.75 | 1 |
| 3 | 1 | 0.715358 | 0.7 | 0.666667 | 1 | 0.75 | 0.25 |
| 4 | 0 | 0.3997 | 0.31 | 0.333333 | 1 | 0.5 | 0.75 |
| 5 | 0.333333 | 0.312156 | 0.12 | 0 | 0 | 0.25 | 0.25 |
| 6 | 0.333333 | 0.247624 | 0.205 | 0 | 0 | 0.25 | 0.5 |
| 7 | 0.333333 | 0.376688 | 0.675 | 0.333333 | 0 | 0.5 | 0.5 |
| 8 | 0.166667 | 0.71986 | 0.65 | 0.333333 | 0 | 0.75 | 0.75 |
| 9 | 0.833333 | 0.63932 | 0.625 | 0 | 0 | 0.5 | 0.25 |

Our Deep Learning Model:

```
model=Sequential()  #initialising ANN
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))
model.add(Dense(17,activation='relu'))

model.add(Dense(1))



model.compile(optimizer='adam',
              loss='mae',

              )
model.fit(x=X_train,y=y_train,
          validation_data=(X_test,y_test),
          batch_size=128,epochs=200,
          )
model.summary()
```

Output:



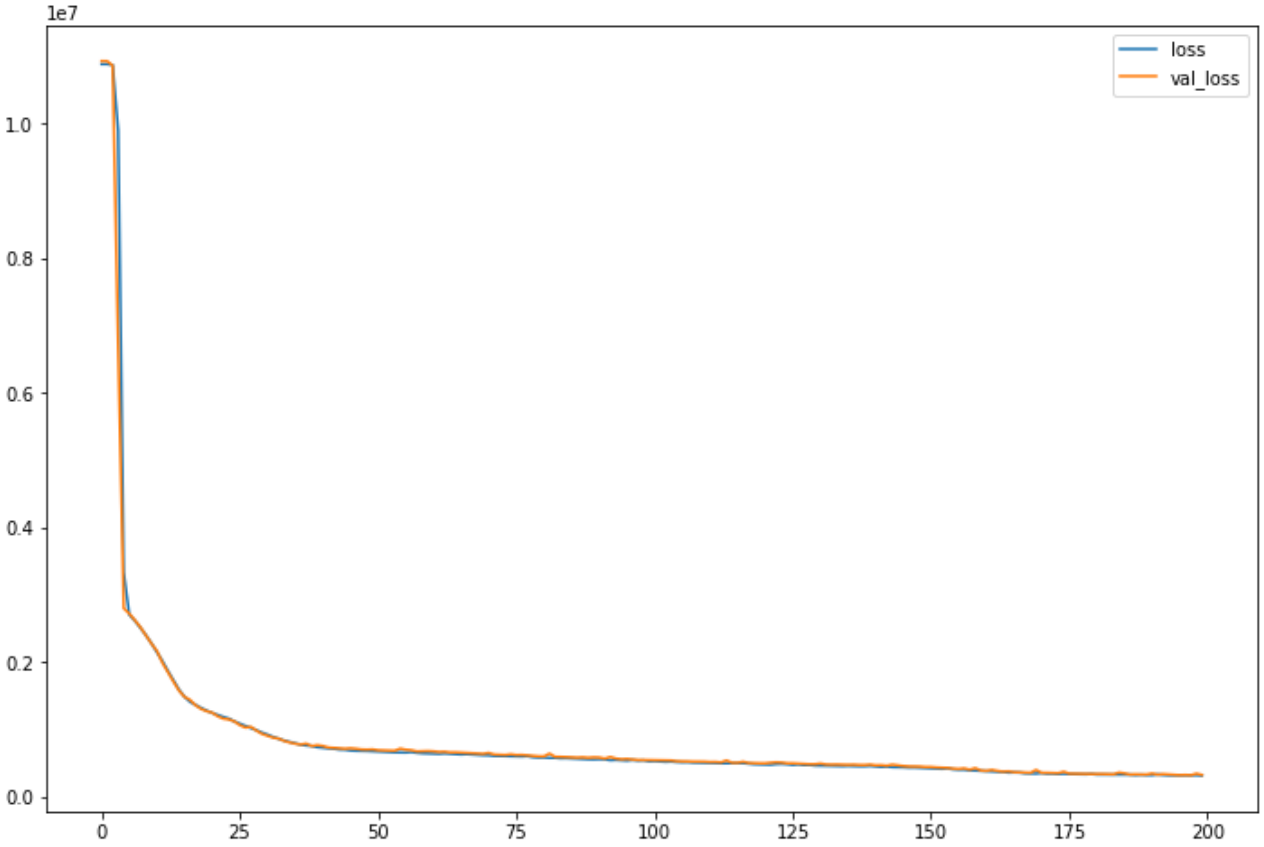
```
Console 2/A
45/45 [==============================] - 0s 4ms/step - loss: 323007.8539 - val_loss: 321186.0312
Epoch 199/200
45/45 [==============================] - 0s 4ms/step - loss: 313749.6746 - val_loss: 345241.0000
Epoch 200/200
45/45 [==============================] - 0s 6ms/step - loss: 318047.4321 - val_loss: 325624.0938
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 17)                306
_________________________________________________________________
dense_1 (Dense)              (None, 17)                306
_________________________________________________________________
dense_2 (Dense)              (None, 17)                306
_________________________________________________________________
dense_3 (Dense)              (None, 17)                306
_________________________________________________________________
dense_4 (Dense)              (None, 17)                306
_________________________________________________________________
dense_5 (Dense)              (None, 17)                306
_________________________________________________________________
dense_6 (Dense)              (None, 17)                306
_________________________________________________________________
dense_7 (Dense)              (None, 17)                306
_________________________________________________________________
dense_8 (Dense)              (None, 17)                306
_________________________________________________________________
dense_9 (Dense)              (None, 17)                306
_________________________________________________________________
dense_10 (Dense)             (None, 17)                306
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 18
=================================================================
Total params: 3,384
Trainable params: 3,384
Non-trainable params: 0
```

MAE on the test data set and the Accuracy of the model:

```
Trainable params: 3,384
Non-trainable params: 0
_________________________________________________________________
MAE: 325624.11251758086
MSE: 167282056707.2385
RMSE: 409001.2918160999
VarScore: 0.9892304917510264
```

So we have got almost 99% accuracy on or model.

Plot of training and Validation loss:




Interpretation of Results:

```
print(df['SALES_PRICE'].describe())
```

```
In [3]: runcell(8, 'C:/Users/Prathamesh/Desktop/NNFL_IA/NNFL_final.py')
count    7.109000e+03
mean     1.089491e+07
std      3.768603e+06
min      2.156875e+06
25%      8.272100e+06
50%      1.033505e+07
75%      1.299390e+07
max      2.366734e+07
Name: SALES_PRICE, dtype: float64

In [4]:
```

From statistics, the mean price of a house in these regions of TamilNadu is around 1 crore. Therefore mean absolute error of about 3 lakhs means that we are bound to overshoot or undershoot by around 0 – 4 lakhs. This means we have an error of about 0% - 4% when compared to the target variable.

## DEPLOYING model aa WEB APPLICATION:

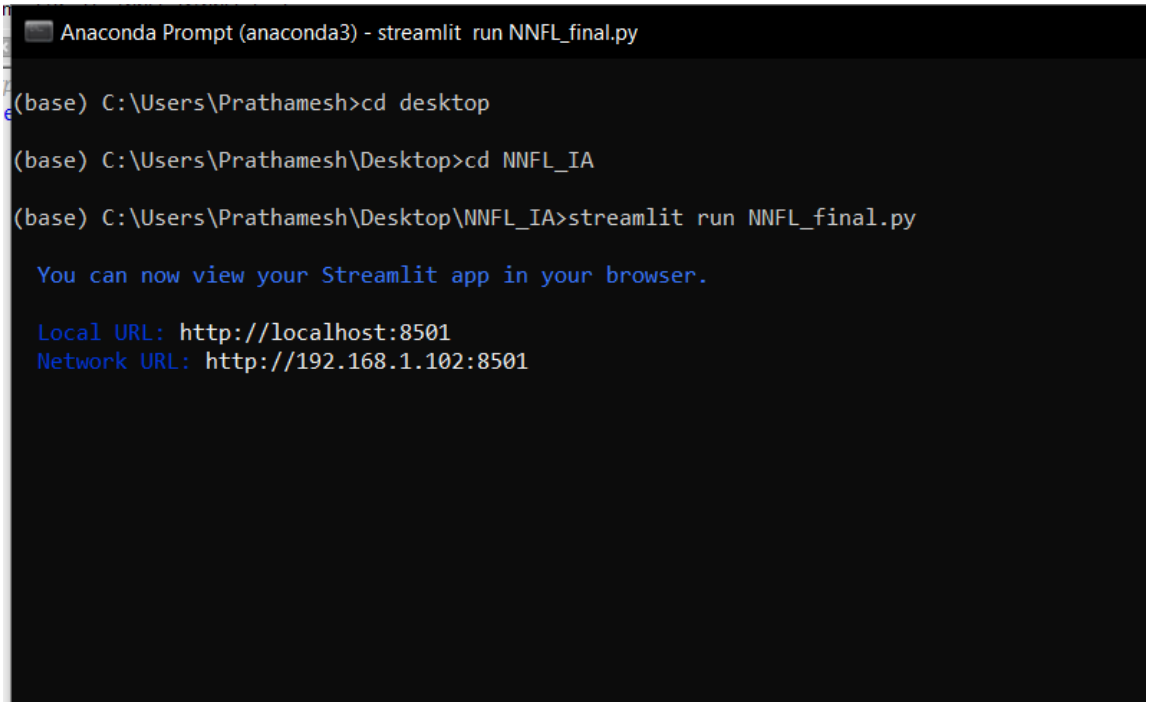
We have used Streamlit library of python for deploying model on web.

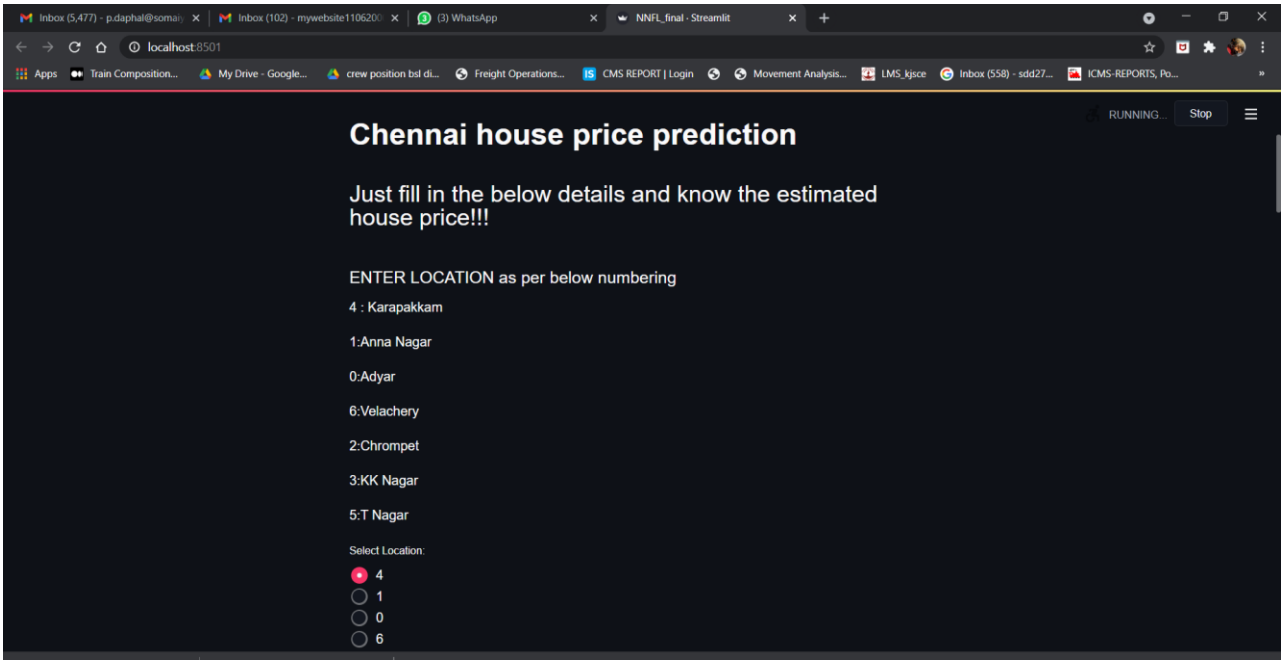
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,Activation
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
import streamlit as st
from sklearn.linear_model import LinearRegression
```

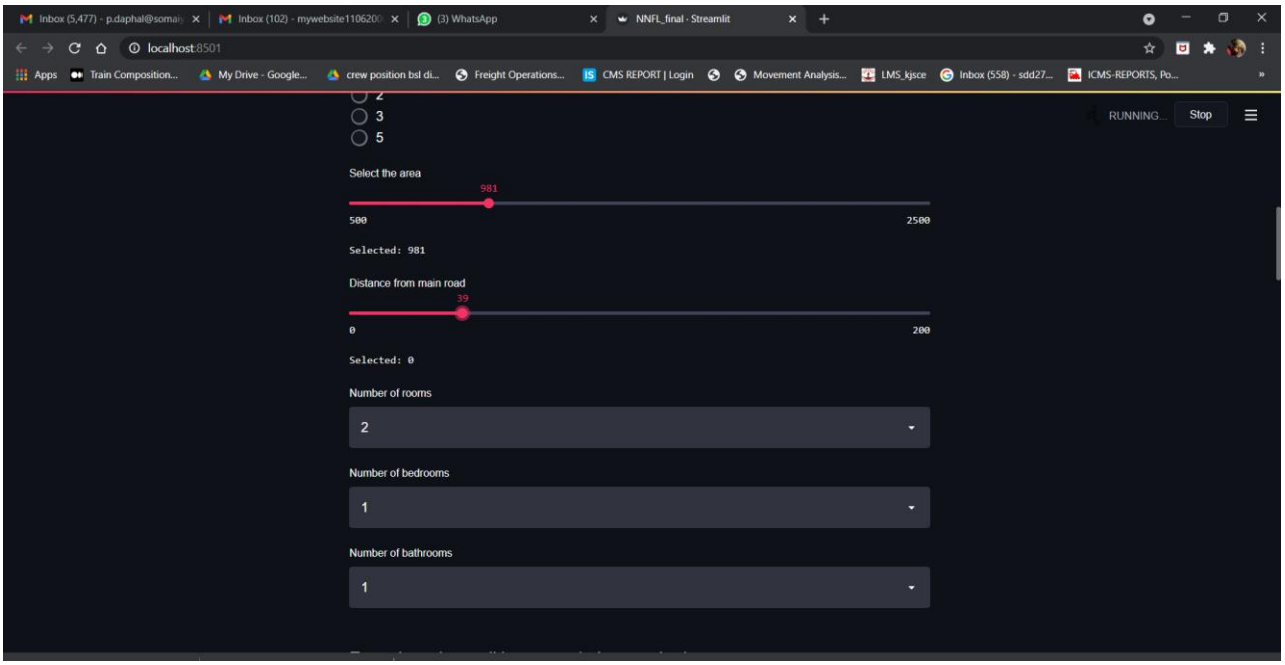
Code for the same can be referred through the uploaded python file on github.

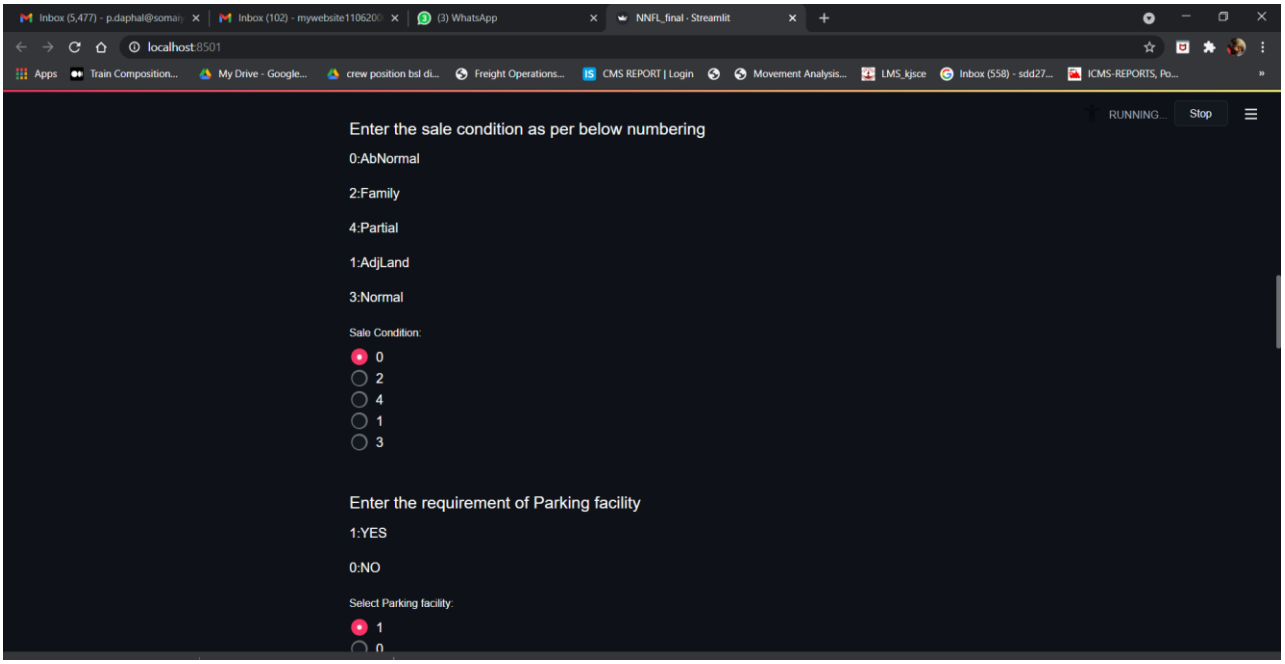
HOW THE INTERFACE LOOKS LIKE??

Command for running code:



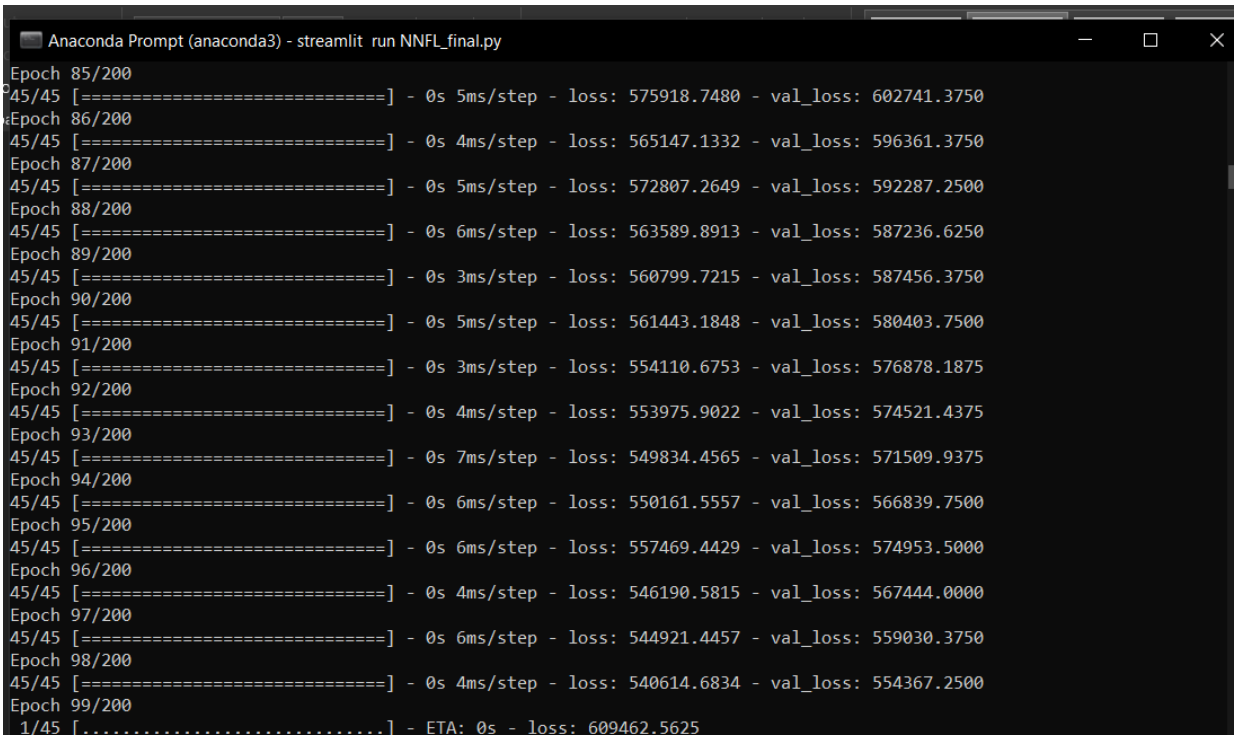





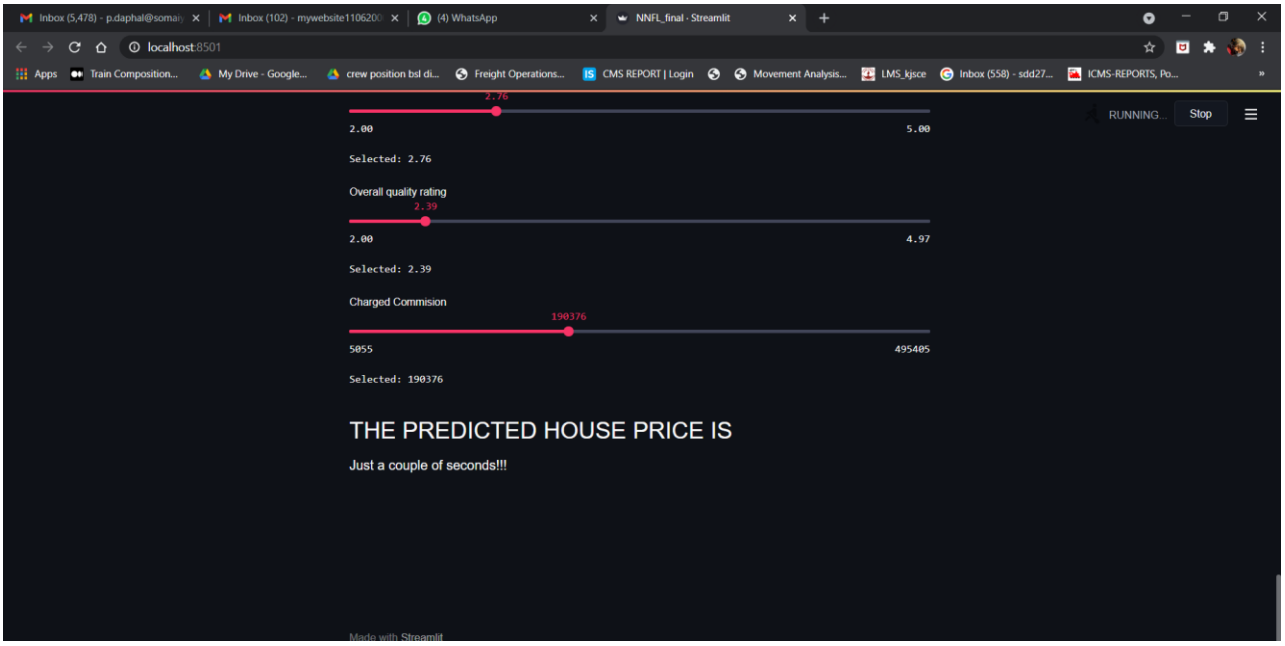




In this way we take inputs from user…

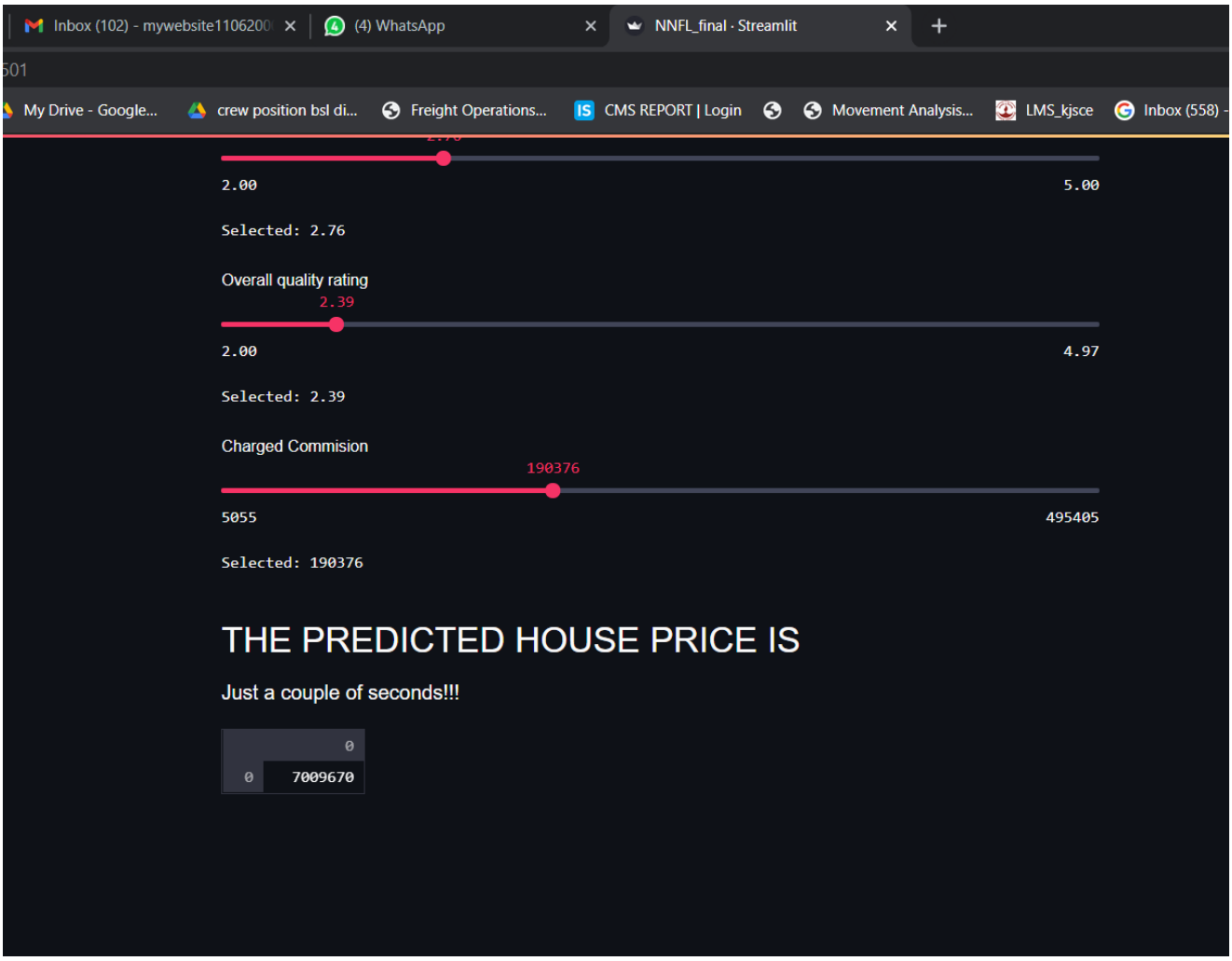
And this is the DL model running at backend:




It takes some time for results to appear because model is running at the backend.




Predicted OUTPUT value:


Inbox (102) - mywebsite1106200
(4) WhatsApp
NNFL_final · Streamlit
My Drive - Google...
crew position bsl di...
Freight Operations...
CMS REPORT | Login
Movement Analysis...
LMS_kjsce
Inbox (558)
2.00
5.00
Selected: 2.76
Overall quality rating
2.39
2.00
4.97
Selected: 2.39
Charged Commision
190376
5055
495405
Selected: 190376
THE PREDICTED HOUSE PRICE IS
Just a couple of seconds!!!
0
0
7009670

Thankyou!!!