Yuvoh Assessment Task

Name: Vanam Sai Simha
Email: simha.1@iitj.ac.in

These are the steps I followed to deal with the problem statement

1. Data Exploration and Visualization
2.Data Cleansing and preparation
2. Feature Extraction
3. Multiple Model Training

Disclaimer: You may not find the code/program well structured as this is an assessment test and time constraint, you find some hard coded values in between, please excuse me for this.

Note: For data Exploration Graphs please go through 'data_exploration.ipynb' after reading this doc

Data Exploration and Visualization: (refer data_exploration.ipynb)
This phase is broadly divided into 3 steps:

- Understanding the problem.
- Univariable study (estimating the distribution of dependent variable).
- Multivariable study (how independent variables are relevant to the target or dependent variable (price in our case)).

This was plotted using matplotlib and seaborn tools (scatter, correlation matrix , skew, kurtosis and others)

After Step 1, I observed that there are many features which contain raw data are need to be parsed and missing values NaN which were to be handled.

Data Cleansing and preparation: (refer prep.ipynb and prep2.ipynb)
Here I initially Categorized column features in 2 classes

- Categorical Features (High variation and low variation)
- Continuous Features
- Textual Features

I had a simple flow for continuous features where I first parsed and made to int or float types and filled the missing values with mean or median or mode as per frequency requirements.
Alternative: (There is another efficient way of filling the missing values by learning them (prediction with missings feature as target variable) and KNN can be used))

Categorical Variables:
For low space variables (i.e 2-7 possible values) I went on with One-Hot Encoding having null as separate category
For high space variables I decided to go with Label Encoding and having null as seperate category

Textual Features:
This is explained in detail in **Feature Extraction** steps

Feature Extraction: (refer prep.ipynb and prep.ipynb)
This is a bit lengthy section and this included parsing and extracting the logic of very crude data E.g: host_since, first, review, last review, host_location, host_verifications, smart_location, amenities, price, security_deposit, cleaning_fee, extra_people etc

I went on to parse DATE stamps and extract new feature of duration between reviews, passed dates etc and I split the smart_location. Price ('$' tag) same for security_deposit, cleaning_fee, extra_people.

I also removed unnecessary or least significant features from the feedback of data exploration if they satisfy either of the condition

- Having null values more than 90% of dataset
- Having similar or duplicate columns
- Having no significance associated with it (*_url, zipcode,*_id etc) (**Note**: these variables can be considered when proper mapping to business logic is provided (e.g zipcode have info about location but only when its decoder or logic is given))

('scrape_id','neighborhood_overview','thumbnail_url','medium_url','xl_picture_url','picture_url','host_acceptance_rate','neighbourhood_group_cleansed','square_feet','weekly_price','monthly_price','license','jurisdiction_names','host_thumbnail_url','host_picture_url','market','zipcode' etc)

Till here we have parsed,extracted, selected and dropped features corresponding to **Categorical and Continuous features**

Now we deal with one of the important category: **Textual features**

All we need is numerical data to train any ML model (excluding few)

Here we need to understand that this feature have temporal aspect as well and this should be captured

texts['name','space','description','notes','transit','access','interaction','house_rules','host_about']

These are the features and here we first remove all the stop words and replace or remove few unwanted patterns and special characters with regular expression lib

Now comes 3 sub steps:

- **Tokenization**
- **Lemmatization**
- **One-hot encoding**
- **Word Embedding**

Here we perform tokenization which is nothing but flatmap and then lemmatize the words (i.e bringing its adjective or verb add-on's to its base word form)(e.g)

Now we encode for all unique words present in particular feature and perform one-hot encoding which is usually high dimensional (10k to 35K in our case)

Now we need to have a low dimensional vector which allows words with similar meaning to have a similar representation with lesser dimension.(100 D in our case)

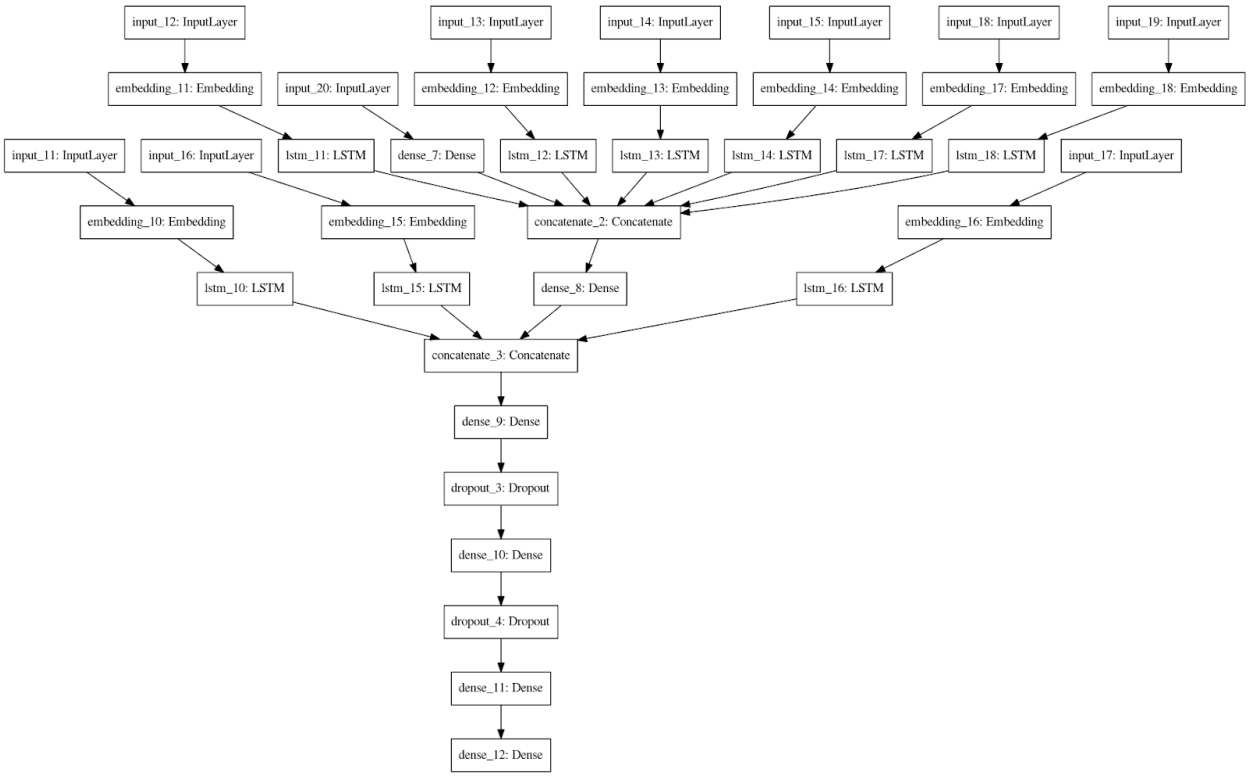
There are two well know open-source word embeddings

1. GloVe
2. word2vec

I decided to go with GloVe (100 D)

Now all the features are ready with their respective data
Data Normalization is performed as it is necessary to bring all the features to the same scale (MinMax Scaling)

ML Modelling and training: (refer NN_model.ipynb)



Here my main model is Artificial Neural Network Model:
(Please go through the jupyter notebook for error graph)
The below tensorflow graph model is a 4 layered dense network with dropout and early stopping as regularization factors.

The input to this model is structural processed data as explained in previous steps and another input is the output of a LSTM (RNN+) network for which textual data encodings are input and embedding layers are loaded with pre-trained GloVe word embeddings.

Optimizer: (Adam's Optimizer)
Loss Func: MAE (Mean Absolute error)

Alternative / Improvement: This particular problem can be modelled and trained differently by changing hyper-parameters like layers, slight change in architecture, loss function , optimizer etc (I didn't try to fine-tune the model much as It is a bit time taking process and felt that it is not much necessary for assessment test)

I, for time being tried 2 different architectures.

I also tried modelling with XGBoost Regressor and Random Forest Regressor but ignored the textual data as these models cannot capture sequential data.

Please go through the code/program and please let me know for any clarification of a particular action.

Thank You