Documentation for Prediction Model

Name:- Ankit Biswas

Branch:- Electronics & Communication Technology

Year:- First

Enrollment number:- 19116008

Link:-

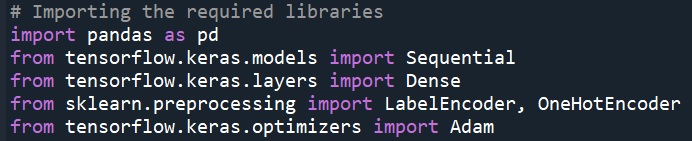
To solve the given problem of predicting the price of houses based on the various information provided, I used an artificial neural network.

The provided datasets had 13 independent features and 1 dependent feature (i.e., the price of the house).

**Note –**

**The images in the documentation are just a sample and may not match with the original code.**

At first, I imported the required libraries

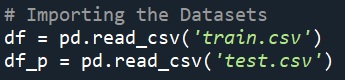


Pandas to handle the datasets

Keras to build the neural network

LabelEncoder & OneHotEncoder to encode categorical data

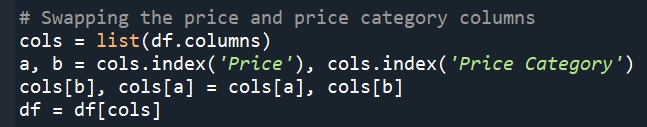
Then I imported the required datasets using pandas



In the dataframe “df”, the ‘Price’ and ‘Price Category’ columns had to be swapped so as to general structure of keeping the dependent variable at the end.

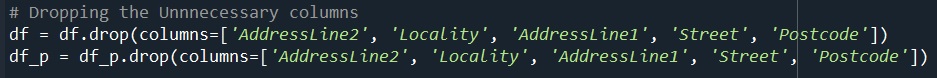
This swapping was achieved by the following code



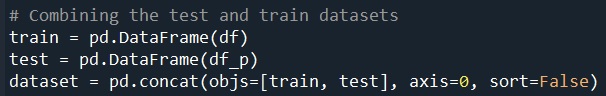
Now the datasets were checked for number of missing values and number of categorical variables per feature (column).

Based on that if a feature had too many missing values (more than 20% of total data) or too many categorical variables (more than 20000 around as then each type would have very less data to train on), it was dropped because of the fact that it won’t have any significant contribution to the model.

Also features like ID, Date and Postcode which doesn’t actually relate to how it can affect the price of a house were dropped by intuition.

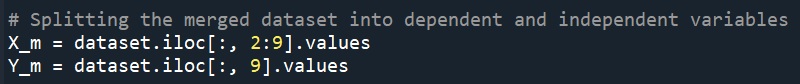


As a next step, I chose to merge both the Training and Test set into one dataset.



The reason for doing this, will be explained little later in the documentation.

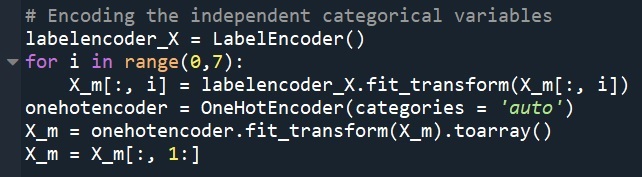
Now the merged dataset is split into the dependent and independent variables.



As most of the entries in the independent variable is categorical, so we need to encode the data so that our model can use it for calculations.

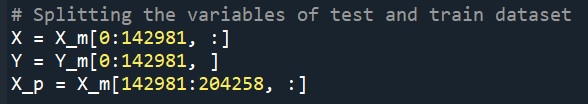
Here we use LabelEncoder to assign a different number to each categorical variable which gives us an idea of the number of categorical variables.

While the OneHotEncoder is used to convert the list of categorical variables into an array consisting of different categorical variables as features.



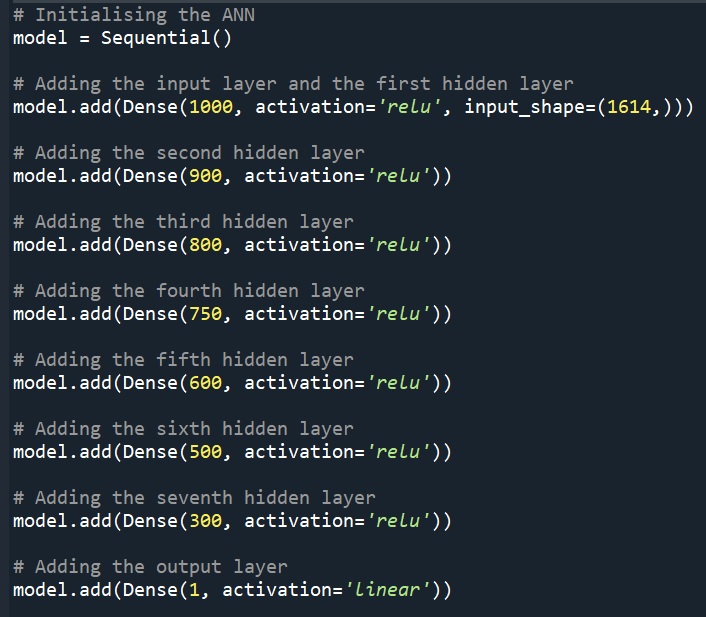
IMP –

The reason for merging the two datasets earlier was that each of them might have different number of categorical variables per feature and hence on applying OneHotEncoder each of the datasets (Train and Test) might end up having different number of features (columns). This will create a problem as the number of features in test set should be same as that on the training set on which the model was trained for the model to work. So, merging both before hand and then encoding will ensure uniform number of features after which they can be split apart.



Now comes the most important part i.e. building the artificial neural network which will be used to predict the prices.

I used Keras with TensorFlow backend to build my ANN.



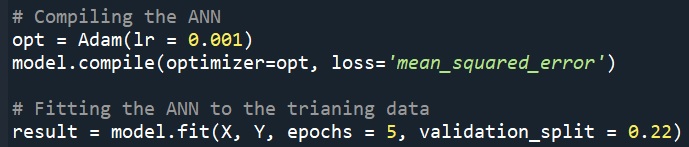
In the above iteration, I have added 7 hidden layers as the input dimension was very high (1614) after one hot encoding.

The activation function used in each hidden layer is ‘Relu’ (Rectified Linear Unit) with different hidden layers having different number of neurons.

For the output layer, with a single output node, I used the ‘linear’ activation function.

This formed the structure of my ANN

Now the model was ready to be compiled and fitted to the training set



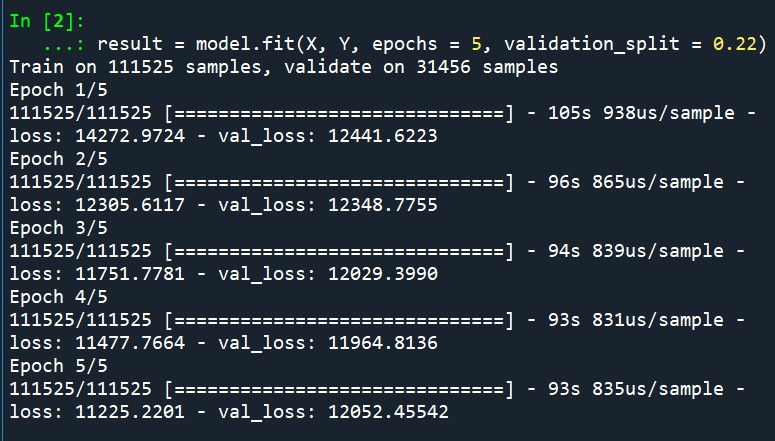
In compiling the ANN, I used ‘adam’ as optimizer and ‘mean squared error’ as the loss parameter. So, the model would try to reduce the loss in each iteration i.e. the mean squared error.

While fitting the model to the training set, I chose the number of epochs to be 5 as the loss starts to converge by the end of 4th epoch.

Also, I allowed a validation split by which the model would be trained on 78% of the training data and then tested on the rest 22% to get the test error.

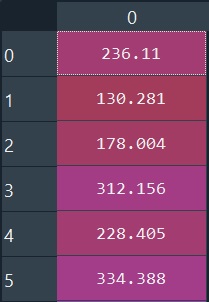
On changing the number of neurons in a particular hidden layer, adding new hidden layers, changing the split ratio or changing the number of epochs, the mean squared error is bound to vary. Hence different iterations of the same code with slight changes might have different orders of error.

A typical evaluation of the model would look like this



In this at every epoch, we can find the mean squared error on both the training and test part. We can see that the rate of reduction in error decreases as the successive epochs pass.



The final step is to apply our model on the test set to get the vector of predictions for our test set

This vector of predictions can then be transferred to an Excel sheet, sorted and made ready for final presentation.

So, this is how I created my model for predicting the price of houses.

In my models, the error used to vary between 11,800 and 12,200 on Cerebro for most of the cases on tweaking the model for the test set mostly due to some overfitting.

*Thank You*