Airbnb Price Classification

Jahnavi Chavali

Baseline accuracy: 39.8%
Accuracy achieved: 65.04%

In [28]:
```
model.fit(train_data, train_labels, epochs=250)
```

```
7907/7907 [==============================] - 0s 39us/sample - loss: 0.9031 - acc: 0.6028
Epoch 145/250
7907/7907 [==============================] - 0s 38us/sample - loss: 0.9045 - acc: 0.5996
Epoch 146/250
7907/7907 [==============================] - 0s 38us/sample - loss: 0.9039 - acc: 0.6030
Epoch 147/250
7907/7907 [==============================] - 0s 39us/sample - loss: 0.9110 - acc: 0.6006
Epoch 148/250
7907/7907 [==============================] - 0s 40us/sample - loss: 0.9057 - acc: 0.5997
Epoch 149/250
7907/7907 [==============================] - 0s 39us/sample - loss: 0.8990 - acc: 0.6038
Epoch 150/250
7907/7907 [==============================] - 0s 38us/sample - loss: 0.9015 - acc: 0.6105
Epoch 151/250
7907/7907 [==============================] - 0s 38us/sample - loss: 0.8998 - acc: 0.5997
Epoch 152/250
7907/7907 [==============================] - 0s 38us/sample - loss: 0.8965 - acc: 0.6028
Epoch 153/250
7907/7907 [==============================] - 0s 38us/sample - loss: 0.8960 - acc: 0.6033
Epoch 154/250
```

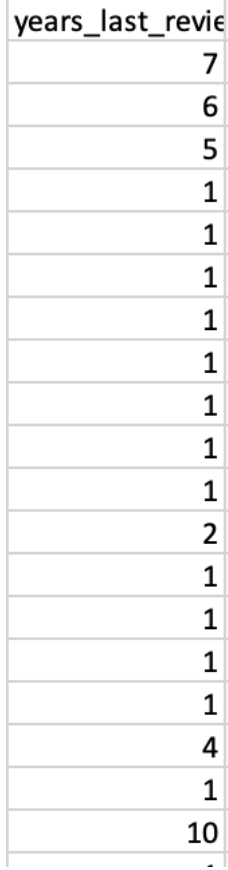
In [29]:
```
train_loss, train_acc = model.evaluate(train_data, train_labels)
print('Train accuracy:', train_acc)
```

```
7907/7907 [==============================] - 0s 21us/sample - loss: 0.8024 - acc: 0.6504
Train accuracy: 0.65043634
```

Things changed to improve accuracy

- **Cleaned** the data (in Excel) to replace missing values with mean data, or in some cases the minimum value.

- As shows in the figure to the left, replaced the “last_review” column whose values were in DateTime format to “years_last_review”, which represents the number of years since the last review was left, making this value easier to read.



years_last_revie
7
6
5
1
1
1
1
1
1
1
1
2
1
1
1
1
4
1
10

Things changed to improve accuracy

- Converted numeric dependent variable to categorical variables and ensured that the data distribution wasn't too skewed.

In [5]:
```
price_ranges = []
for i in range(0, len(data['price'])):
    if data['price'][i] < 50:
        price_ranges.append("4")
    elif data['price'][i] < 100:
        price_ranges.append("3")
    elif data['price'][i] < 150:
        price_ranges.append("2")
    elif data['price'][i] < 230:
        price_ranges.append("1")
    else:
        price_ranges.append("0")

data['price_range'] = price_ranges
convert_to_categorical('price_range')
data.head()
```

In [165]:
```
data_final['price_range'].value_counts()
```

Out[165]:
```
3    2148
2    1683
1    1570
0    1491
4    1015
Name: price_range, dtype: int64
```

Things changed to improve accuracy

- Further manipulation of dataset to normalize values and create “dummy” variables for categorical variables.

In [6]:
```
def normalise(col_name):
    col = data[col_name]
    deno = 1/(col.max()-col.min())
    #print(deno)
    data[col_name] = (col-col.min())*deno

normalise('minimum_nights')
normalise('number_of_reviews')
normalise('years_last_review')
normalise('calculated_host_listings_count')
normalise('availability_365')


data.head()
```

In [10]:
```
new_cols = pd.get_dummies(data_final['neighbourhood_group'],prefix='neighbourhood_group')
data_final = pd.concat([data_final,new_cols.round().astype(int)], axis=1)

new_cols = pd.get_dummies(data_final['room_type'],prefix='room_type')
data_final = pd.concat([data_final,new_cols.round().astype(int)], axis=1)

data_final.head()
```

Out[10]:

neighbourhood_group_North Region	neighbourhood_group_North-East Region	neighbourhood_group_West Region	room_type_Entire home/apt	room_type_Priv room
1	0	0	0	1
0	0	0	0	1
1	0	0	0	1
0	0	0	0	1
0	0	0	0	1

Things changed to improve accuracy

- Added 2 more hidden layers. The first hidden layer improved my accuracy by 5-6% and the second hidden layer further improved the accuracy by 6-7%. Adding a third hidden layer did not improve the accuracy at all, therefore I removed it.

- Changed the activation function for the output nodes to Softmax, however this did not significantly affect the accuracy.

In [121]:
```
model = keras.Sequential([
#    keras.layers.Flatten(input_shape=(1, 6)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(56, activation=tf.nn.relu),
    keras.layers.Dense(28, activation=tf.nn.relu),
    keras.layers.Dense(5, activation=tf.nn.softmax)
])
```

Things changed to improve accuracy

- Tried different learning rates with Adam optimiser. Increasing it from the default rate to 0.00146 increased the accuracy by 5%.

In [27]:
```
optimiser = keras.optimizers.Adam(lr=0.00146)
model.compile(
              #optimizer='adam',  #update weights
              optimizer = optimiser,
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Things changed to improve accuracy

- Changed the number of epochs to 250. Increasing it to 300 did not improve the accuracy, however there was a 2% difference between 200 and 250.

```
In [123]: model.fit(train_data, train_labels, epochs=250)
          Epoch 242/250
          7907/7907 [==============================] - 0s 40us/sample - loss: 0.9082 - acc: 0.5966
          Epoch 243/250
          7907/7907 [==============================] - 0s 40us/sample - loss: 0.9073 - acc: 0.6007
          Epoch 244/250
          7907/7907 [==============================] - 0s 40us/sample - loss: 0.9065 - acc: 0.5992
          Epoch 245/250
          7907/7907 [==============================] - 0s 41us/sample - loss: 0.9037 - acc: 0.6048
          Epoch 246/250
          7907/7907 [==============================] - 0s 40us/sample - loss: 0.9069 - acc: 0.5974
          Epoch 247/250
          7907/7907 [==============================] - 0s 40us/sample - loss: 0.9080 - acc: 0.5983
          Epoch 248/250
          7907/7907 [==============================] - 0s 40us/sample - loss: 0.9051 - acc: 0.6016
          Epoch 249/250
          7907/7907 [==============================] - 0s 40us/sample - loss: 0.9030 - acc: 0.6012
          Epoch 250/250
          7907/7907 [==============================] - 0s 41us/sample - loss: 0.9043 - acc: 0.6047
Out[123]: <tensorflow.python.keras.callbacks.History at 0x7f798ab0af98>
```