image

Melbourne House Price Prediction Using ANN

I. Outline

- Demand forecasting, house price prediction is always an important requirement. Assuming you are looking for a dream home, you will be very concerned about whether the price offered by the seller is already the lowest? Will the house price be suitable for the market after negotiating? Or you are lucky to have the opportunity to own the best price house in the area because the owner has an urgent need to sell,…
- Home price forecasting will always be helpful to buyers, real estate agents, and sellers also. Because no one wants to buy a house that's 50% or more expensive than the market price. Raising house prices too high only accelerates the freezing of the real estate market, not really helpful in increasing the market's real estate volume.
- In this project, we will build a model that predicts house prices in Melbourne based on house parameters. This can give buyers an idea of a suitable price to negotiate with the seller ## II. Business Objective/ Problem
- Assume that you work in the Data Science department of a real estate company. Your task is to support the buyer and advise the seller of the most suitable price for both parties so that the transaction can be done as soon as possible.
- Your company is expanding to Melbourne, so they urgently need a model to forecast house prices in this area
- This project is built based on that request. ## III. Project implementation ### 1. Business Understanding Based on the above description => identify the problem:
- Find solutions to attract customers with the most accurate advice, thereby expanding business in this area

- Objectives/problems: build a model to predict house prices based on the parameters of the house, thereby giving suggestions to customers.
- Applied method: ANN ### 2. Data Understanding/ Acquire
- This data was scraped from publicly available results posted every week from Domain.com.au and be cleaned by Tony Pino.
- You can download the dataset at: https://www.kaggle.com/datasets/anthonypino/melbourne-housing-market
- Some Key Details
  - Suburb: Suburb
  - Address: Address
  - Rooms: Number of rooms
  - Price: Price in Australian dollars

  Method:
  - S - property sold;
  - SP - property sold prior;
  - PI - property passed in;
  - PN - sold prior not disclosed;
  - SN - sold not disclosed;
  - NB - no bid;
  - VB - vendor bid;
  - W - withdrawn prior to auction;
  - SA - sold after auction;
  - SS - sold after auction price not disclosed.
  - N/A - price or highest bid not available.

  Type:
  - br - bedroom(s);
  - h - house,cottage,villa, semi,terrace;
  - u - unit, duplex;
  - t - townhouse;
  - dev site - development site;
  - o res - other residential.
  - SellerG: Real Estate Agent
  - Date: Date sold
  - Distance: Distance from CBD in Kilometres
  - Regionname: General Region (West, North West, North, North east …etc)
  - Propertycount: Number of properties that exist in the suburb.
  - Bedroom2 : Scraped # of Bedrooms (from different source)
  - Bathroom: Number of Bathrooms
  - Car: Number of carspots
  - Landsize: Land Size in Metres
  - BuildingArea: Building Size in Metres

- YearBuilt: Year the house was built
- CouncilArea: Governing council for the area
- Lattitude: Self explanitory
- Longtitude: Self explanitory

image

3. Build model

3.1. Understand the dataset:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Postcode	Regionname	Propertycount	Distance	CouncilArea
0	Abbotsford	49 Lithgow St	3	h	1490000.0	S	Jellis	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
1	Abbotsford	59A Turner St	3	h	1220000.0	S	Marshall	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
2	Abbotsford	119B Yarra St	3	h	1420000.0	S	Nelson	1/04/2017	3067	Northern Metropolitan	4019	3.0	Yarra City Council
3	Aberfeldie	68 Vida St	3	h	1515000.0	S	Barry	1/04/2017	3040	Western Metropolitan	1543	7.5	Moonee Valley City Council
4	Airport West	92 Clydesdale Rd	2	h	670000.0	S	Nelson	1/04/2017	3042	Western Metropolitan	3464	10.4	Moonee Valley City Council

image

```
df.describe()
```

	Rooms	Price	Postcode	Propertycount	Distance
count	63023.000000	4.843300e+04	63023.000000	63023.000000	63023.000000
mean	3.110595	9.978982e+05	3125.673897	7617.728131	12.684829
std	0.957551	5.934989e+05	125.626877	4424.423167	7.592015
min	1.000000	8.500000e+04	3000.000000	39.000000	0.000000
25%	3.000000	6.200000e+05	3056.000000	4380.000000	7.000000
50%	3.000000	8.300000e+05	3107.000000	6795.000000	11.400000
75%	4.000000	1.220000e+06	3163.000000	10412.000000	16.700000
max	31.000000	1.120000e+07	3980.000000	21650.000000	64.100000

image

3.2. Pre-processing data:

```
#1. Xóa các cột không liên quan
#Xóa cột Address, SellerF, Propertycount, Date
df = df.drop(['Address', 'SellerG', 'Propertycount', 'Date'], axis=1)
df.head()
```

	Suburb	Rooms	Type	Price	Method	Postcode	Regionname	Distance	CouncilArea
0	Abbotsford	3	h	1490000.0	S	3067	Northern Metropolitan	3.0	Yarra City Council
1	Abbotsford	3	h	1220000.0	S	3067	Northern Metropolitan	3.0	Yarra City Council
2	Abbotsford	3	h	1420000.0	S	3067	Northern Metropolitan	3.0	Yarra City Council
3	Aberfeldie	3	h	1515000.0	S	3040	Western Metropolitan	7.5	Moonee Valley City Council
4	Airport West	2	h	670000.0	S	3042	Western Metropolitan	10.4	Moonee Valley City Council

image

```
#2. Kiểm tra NaN --> chỉ có Price có NaN
df.isna().sum()
```

```
Suburb              0
Rooms               0
Type                0
Price           14590
Method              0
Postcode            0
Regionname          0
Distance            0
CouncilArea         0
dtype: int64
```

```
#3. Loại bỏ những dòng bị trùng (all-columns, nếu có):
df = df.drop_duplicates()
df = df.reset_index(drop=True)
```

image

Phát hiện và xử lý ngoại lệ

```
#Tao dataframe kiem tra co can xoa outlier hay khong (df_now):
df_now = df[['Distance', 'Postcode']]
```

```
#lấy những giá trị không phải outlier:
df_now = df_now[(df_now['Distance'] <= (Q3_Distance + 1.5*iqr_Distance))] #upper outlier
df_now = df_now[(df_now['Postcode'] <= (Q3_Postcode + 1.5*iqr_Postcode))] #upper outlier
```

image

```
print('Distance:')
print('mean (before) = ',df.Distance.mean(), ', mean (after) = ', df_now.Distance.mean() )
print('chênh lệch mean (before/after - 100%) = ', (df.Distance.mean() / df_now.Distance.mean() - 1 )* 100, '%')
print('Phải loại bỏ outlier vì chênh lệch mean trước và sau khi loại bỏ là lớn')
```

```
Distance:
mean (before) =  12.712249067146132 , mean (after) =  11.663038658267645
chênh lệch mean (before/after - 100%) =  8.996029590751009 %
Phải loại bỏ outlier vì chênh lệch mean trước và sau khi loại bỏ là lớn
```

```
print('Postcode:')
print('mean (before) = ',df.Postcode.mean(), ', mean (after) = ', df_now.Postcode.mean() )
print('chênh lệch mean (before/after - 100%) = ', (df.Postcode.mean() / df_now.Postcode.mean() - 1 )* 100, '%')
print('Không cần loại bỏ outlier vì chênh lệch mean trước và sau khi loại bỏ là không đáng kể')
```

```
Postcode:
mean (before) =  3125.3366200412747 , mean (after) =  3103.265668561895
chênh lệch mean (before/after - 100%) =  0.7112169513223687 %
Không cần loại bỏ outlier vì chênh lệch mean trước và sau khi loại bỏ là không đáng kể
```

image

Loại bỏ outlier cột Distance

```
df = df[(df['Distance'] <= (Q3_Distance + 1.5*iqr_Distance))] #upper outlier
df = df.reset_index(drop=True)
```

```
df.head()
```

	Suburb	Rooms	Type	Price	Method	Postcode	Regionname	Distance	CouncilArea
0	Abbotsford	3	h	1490000.0	S	3067	Northern Metropolitan	3.0	Yarra City Council
1	Abbotsford	3	h	1220000.0	S	3067	Northern Metropolitan	3.0	Yarra City Council
2	Abbotsford	3	h	1420000.0	S	3067	Northern Metropolitan	3.0	Yarra City Council
3	Aberfeldie	3	h	1515000.0	S	3040	Western Metropolitan	7.5	Moonee Valley City Council
4	Airport West	2	h	670000.0	S	3042	Western Metropolitan	10.4	Moonee Valley City Council

image

Chuyển các cột có dạng text sang dạng nhị phân (dummies)

```
df = pd.get_dummies(data=df, columns=['Suburb', 'Type', 'Method', 'Regionname', 'CouncilArea'], drop_first=True)
```

```
df.head()
```

	Rooms	Price	Postcode	Distance	Suburb_Aberfeldie	Suburb_Airport West	Suburb_Albanvale	Suburb_Albert Park	Subur
0	3	1490000.0	3067	3.0	0	0	0	0	
1	3	1220000.0	3067	3.0	0	0	0	0	
2	3	1420000.0	3067	3.0	0	0	0	0	
3	3	1515000.0	3040	7.5	1	0	0	0	

image

Comment:

Data has preprocessed (df) and Price column still has missing value -> split df into 2 parts: - df_final (df has dropped missing value), used to build prediction model - df_new (df filter to get missing value data), used to predict using the model above

Loại bỏ NaN df_final

```
df['Price'].isna().sum()
```

```
5190
```

```
#df_final đã drop NaN
df_final = df.dropna()
```

image

Lọc lấy data có NaN để dự đoán

```
#df_new lọc lấy data bị missing
df_new = df[df['Price'].isna()]
```

image

image

3.3 Build model:

- Split training/testing data

```
train_X = df_final.drop(columns=['Price'])
train_y = df_final[['Price']]
```

- Rescale train_X due to the difference in range between the inputs

```
scaler_x = MinMaxScaler()
train_X = scaler_x.fit_transform(train_X)
```

- Create and add layers to ANN model

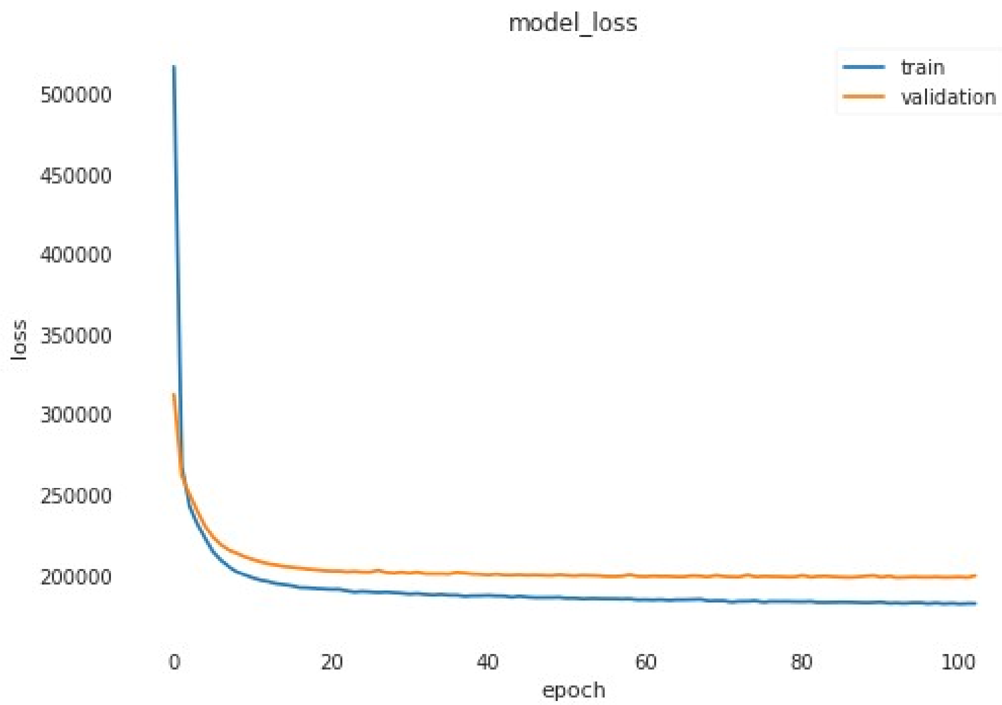
```
#create model
model = Sequential()
#add model layers
model.add(Dense(182, activation='relu', input_shape=(n_cols, )))
#(362+1)/2
model.add(Dropout(rate=0.1))
model.add(Dense(128, activation='relu'))
model.add(Dropout(rate=0.1))
model.add(Dense(128, activation='relu')) #cải tiến bằng cách cho học sâu thêm
model.add(Dense(1, activation='linear')) #output

#compile model
model.compile(optimizer='adam', loss='mae')
```

- Fit model

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping_minitor = EarlyStopping(patience=10)
#train model
history = model.fit(train_X, train_y,
                    epochs=300,
                    batch_size=32,
                    validation_split=0.2,
                    callbacks=[early_stopping_minitor])
```

- Plot loss of train and test set

```
print(history.history.keys())
#Loss in train and test
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train', 'validation'], loc='upper right')
plt.show()
```



image

- Evaluate result

```
#evaluate the result
print('Evaluation on test data')
results = model.evaluate(train_X, train_y)
print('mae: ', results)
```

```
Evaluation on test data
1289/1289 [==============================] - 1s 1ms/step - loss: 180736.1406
mae:  180736.140625
```

image

Comment: mean(Price) = 9.9e5, mae ~ 180000 –> mae/mean ratio ~ 18% –> acceptable 18% difference from mean to predict house price

3.4. Make prediction on new data

- Use the dataframe df_new that we have filtered with NaN rows. This data uses for new price prediction

- Remove output column (Price), which contains NaN. The remain columns for inputs

```
train_X_1 = df_new.drop(columns=['Price'])
```

- Transform inputs

```
train_X_1 = scaler_x.transform(train_X_1)
```

- Make prediction on inputs

```
test_y_predictions = model.predict(train_X_1)
```

Print some rows of prediction

```
test_y_predictions[:10]
```

```
array([[ 780776.7 ],
       [ 692471.94],
       [1948649.8 ],
       [ 724873.7 ],
       [1474910.4 ],
       [1775041.2 ],
       [ 934622.4 ],
       [1084697.5 ],
       [ 728664.6 ],
       [1302552.8 ]], dtype=float32)
```

image

image

4. Conclusion

- House price forecasting is a useful tool to help buyers understand the value of the home they are planning to buy
- With a difference of 18% compared to the mean, the model can be used to suggest buying/selling prices to customers

Thank you for your experience with my project. Hope you enjoy it!