|  |  |
| --- | --- |
| DATE | 09/05/2023 |
| TEAM ID | NM2023TMID06916 |
| CODE | KAGGLE |

**Predicting House Prices (Keras - Artificial Neural Network)**

Table of Contents

1.Overview

2.Dataset

3.Working with Feature Data

4.Scaling and Train Test Split

5.Predicting on a Brand New House

Overview

One of the objectives of this notebook is to show step-by-step how to analyze and visualize the dataset to predict future home prices. Moreover, we are going to explain most of the concepts used so that you understand why we are using them. In base of features like sqft\_living, bathrooms, bedrooms, view, and others, we are going to build a deep learning model that can predict future price.



DATASET

* This dataset contains house sale prices for King County, which includes Seattle

***Feature Columns***

* **id:** Unique ID for each home sold
* **date:** Date of the home sale
* **price:** Price of each home sold
* **bedrooms:** Number of bedrooms
* **bathrooms:** Number of bathrooms, where .5 accounts for a room with a toilet but no shower
* **sqft\_living:** Square footage of the apartments interior living space
* **sqft\_lot:** Square footage of the land space
* **floors:** Number of floors
* **waterfront:** - A dummy variable for whether the apartment was overlooking the waterfront or not
* **view:** An index from 0 to 4 of how good the view of the property was
* **condition:** - An index from 1 to 5 on the condition of the apartment,
* **grade:** An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high quality level of construction and design.
* **sqft\_above:** The square footage of the interior housing space that is above ground level
* **sqft\_basement:** The square footage of the interior housing space that is below ground level
* **prebuilt:** The year the house was initially built
* **yr\_renovated:** The year of the house’s last renovation
* **zipcode:** What zipcode area the house is in
* **lat:** Lattitude
* **long:** Longitude
* **sqft\_living15:** The square footage of interior housing living space for the nearest 15 neighbors
* **sqft\_lot15:** The square footage of the land lots of the nearest 15 neighbors

### **Imports**

In [1]:

*# data analysis and wrangling*

import pandas as pd

import numpy as np

import random as rnd

*# visualization*

import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

*# scaling and train test split*

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import MinMaxScaler

*# creating a model*

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Activation

from tensorflow.keras.optimizers import Adam

*# evaluation on test data*

from sklearn.metrics import mean\_squared\_error,mean\_absolute\_error,explained\_variance\_score

from sklearn.metrics import classification\_report,confusion\_matrix

### **Acquire data**

The Python Pandas packages helps us work with our datasets. We start by acquiring the datasets into Pandas DataFrames.

In [2]:

df = pd.read\_csv('../input/housesalesprediction/kc\_house\_data.csv')

### **Analyze by describing data**

Pandas also helps describe the datasets answering following questions early in our project.

**dataset?**

In [3]:

print(df.columns.values)

['id' 'date' 'price' 'bedrooms' 'bathrooms' 'sqft\_living' 'sqft\_lot'

'floors' 'waterfront' 'view' 'condition' 'grade' 'sqft\_above'

'sqft\_basement' 'yr\_built' 'yr\_renovated' 'zipcode' 'lat' 'long'

'sqft\_living15' 'sqft\_lot15']

**categorical**

These values classify the samples into sets of similar samples. Within categorical features are the values nominal, ordinal, ratio, or interval based? Among other things this helps us select the appropriate plots for visualization.

* Categorical: id, waterfront, zipcode
* **numerical**

These values change from sample to sample. Within numerical features are the values discrete, continuous, or timeseries based? Among other things this helps us select the appropriate plots for visualization.

* Continous: price, bathrooms, floors, lat, long.
* Discrete: date, bedrooms, sqft\_living, sqft\_lot, view, condition, grade, sqft\_above, sqft\_basement, yr\_built, yr\_renovated, sqft\_living15, sqft\_lot15.

In [4]:

*# preview the data*

df.head()

Out[4]:

|  | id |  | date | price | bedrooms | bathrooms | sqft\_living | sqft\_lot | floors | waterfront | view | ... | grade | sqft\_above | sqft\_basement | yr\_built | yr\_renovated | zipcode | lat | long | sqft\_living15 | sqft\_lot15 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 7129300520 |  | 20141013T000000 | 221900.0 | 3 | 1.00 | 1180 | 5650 | 1.0 | 0 | 0 | ... | 7 | 1180 | 0 | 1955 | 0 | 98178 | 47.5112 | -122.257 | 1340 | 5650 |
| 1 | | 6414100192 |  | 20141209T000000 | 538000.0 | 3 | 2.25 | 2570 | 7242 | 2.0 | 0 | 0 | ... | 7 | 2170 | 400 | 1951 | 1991 | 98125 | 47.7210 | -122.319 | 1690 | 7639 |
| 2 | 5631500400 |  | 20150225T000000 | 180000.0 | 2 | 1.00 | 770 | 10000 | 1.0 | 0 | 0 | ... | 6 | 770 | 0 | 1933 | 0 | 98028 | 47.7379 | -122.233 | 2720 | 8062 |
| 3 | 2487200875 |  | 20141209T000000 | 604000.0 | 4 | 3.00 | 1960 | 5000 | 1.0 | 0 | 0 | ... | 7 | 1050 | 910 | 1965 | 0 | 98136 | 47.5208 | -122.393 | 1360 | 5000 |
| 4 | 1954400510 |  | 20150218T000000 | 510000.0 | 3 | 2.00 | 1680 | 8080 | 1.0 | 0 | 0 | ... | 8 | 1680 | 0 | 1987 | 0 | 98074 | 47.6168 | -122.045 | 1800 | 7503 |

| 1 | id | date | price | bedrooms | bathrooms | sqft\_living | sqft\_lot | floors | waterfront | view | ... | grade | sqft\_above | sqft\_basement | yr\_built | yr\_renovated | zipcode | lat | long | sqft\_living15 | sqft\_lot15 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 21608 | 263000018 | 20140521T000000 | 360000.0 | 3 | 2.50 | 1530 | 1131 | 3.0 | 0 | 0 | ... | 8 | 1530 | 0 | 2009 | 0 | 98103 | 47.6993 | -122.346 | 1530 | 1509 |
| 21609 | 6600060120 | 20150223T000000 | 400000.0 | 4 | 2.50 | 2310 | 5813 | 2.0 | 0 | 0 | ... | 8 | 2310 | 0 | 2014 | 0 | 98146 | 47.5107 | -122.362 | 1830 | 7200 |
| 21610 | 1523300141 | ­­­­­00 | 402101.0 | 2 | 0.75 | 1020 | 1350 | 2.0 | 0 | 0 | ... | 7 | 1020 | 0 | 2009 | 0 | 98144 | 47.5944 | -122.299 | 1020 | 2007 |
| 21611 | 291310100 | 20150116T000000 | 400000.0 | 3 | 2.50 | 1600 | 2388 | 2.0 | 0 | 0 | ... | 8 | 1600 | 0 | 2004 | 0 | 98027 | 47.5345 | -122.069 | 1410 | 1287 |
| 21612 | 1523300157 | 20141015T000000 | 325000.0 | 2 | 0.75 | 1020 | 1076 | 2.0 | 0 | 0 | ... | 7 | 1020 | 0 | 2008 | 0 | 98144 | 47.5941 | -122.299 | 1020 | 1357 |

5 rows × 21 columns

In [5]:

*# preview the data*

df.tail()

Out[5]:

5 rows × 21 columns

**blank, null or empty values**

We can check for missing values with pandas isnull(). This indicates whether values are missing or not. Then we can sum all the values to check every column.

In [6]:

*# No missing values*

df.isnull().sum()

Out[6]:

id 0

date 0

price 0

bedrooms 0

bathrooms 0

sqft\_living 0

sqft\_lot 0

floors 0

waterfront 0

view 0

condition 0

grade 0

sqft\_above 0

sqft\_basement 0

yr\_built 0

yr\_renovated 0

zipcode 0

lat 0

long 0

sqft\_living15 0

sqft\_lot15 0

dtype: int64

**data types**

Five features are floats, fifteen are integers and one is an object.

In [7]:

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 21613 entries, 0 to 21612

Data columns (total 21 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 id 21613 non-null int64

1 date 21613 non-null object

2 price 21613 non-null float64

3 bedrooms 21613 non-null int64

4 bathrooms 21613 non-null float64

5 sqft\_living 21613 non-null int64

6 sqft\_lot 21613 non-null int64

7 floors 21613 non-null float64

8 waterfront 21613 non-null int64

9 view 21613 non-null int64

10 condition 21613 non-null int64

11 grade 21613 non-null int64

12 sqft\_above 21613 non-null int64

13 sqft\_basement 21613 non-null int64

14 yr\_built 21613 non-null int64

15 yr\_renovated 21613 non-null int64

16 zipcode 21613 non-null int64

17 lat 21613 non-null float64

18 long 21613 non-null float64

19 sqft\_living15 21613 non-null int64

20 sqft\_lot15 21613 non-null int64

dtypes: float64(5), int64(15), object(1)

memory usage: 3.5+ MB

**distribution of numerical feature values across the samples**

In [8]:

df.describe().transpose()

Out[8]:

|  | count | mean | std | min | 25% | 50% | 75% | max |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | 21613.0 | 4.580302e+09 | 2.876566e+09 | 1.000102e+06 | 2.123049e+09 | 3.904930e+09 | 7.308900e+09 | 9.900000e+09 |
| price | 21613.0 | 5.400881e+05 | 3.671272e+05 | 7.500000e+04 | 3.219500e+05 | 4.500000e+05 | 6.450000e+05 | 7.700000e+06 |
| bedrooms | 21613.0 | 3.370842e+00 | 9.300618e-01 | 0.000000e+00 | 3.000000e+00 | 3.000000e+00 | 4.000000e+00 | 3.300000e+01 |
| bathrooms | 21613.0 | 2.114757e+00 | 7.701632e-01 | 0.000000e+00 | 1.750000e+00 | 2.250000e+00 | 2.500000e+00 | 8.000000e+00 |
| sqft\_living | 21613.0 | 2.079900e+03 | 9.184409e+02 | 2.900000e+02 | 1.427000e+03 | 1.910000e+03 | 2.550000e+03 | 1.354000e+04 |
| sqft\_lot | 21613.0 | 1.510697e+04 | 4.142051e+04 | 5.200000e+02 | 5.040000e+03 | 7.618000e+03 | 1.068800e+04 | 1.651359e+06 |
| floors | 21613.0 | 1.494309e+00 | 5.399889e-01 | 1.000000e+00 | 1.000000e+00 | 1.500000e+00 | 2.000000e+00 | 3.500000e+00 |
| waterfront | 21613.0 | 7.541757e-03 | 8.651720e-02 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 1.000000e+00 |
| view | 21613.0 | 2.343034e-01 | 7.663176e-01 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 4.000000e+00 |
| condition | 21613.0 | 3.409430e+00 | 6.507430e-01 | 1.000000e+00 | 3.000000e+00 | 3.000000e+00 | 4.000000e+00 | 5.000000e+00 |
| grade | 21613.0 | 7.656873e+00 | 1.175459e+00 | 1.000000e+00 | 7.000000e+00 | 7.000000e+00 | 8.000000e+00 | 1.300000e+01 |
| sqft\_above | 21613.0 | 1.788391e+03 | 8.280910e+02 | 2.900000e+02 | 1.190000e+03 | 1.560000e+03 | 2.210000e+03 | 9.410000e+03 |
| sqft\_basement | 21613.0 | 2.915090e+02 | 4.425750e+02 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 5.600000e+02 | 4.820000e+03 |
| yr\_built | 21613.0 | 1.971005e+03 | 2.937341e+01 | 1.900000e+03 | 1.951000e+03 | 1.975000e+03 | 1.997000e+03 | 2.015000e+03 |
| yr\_renovated | 21613.0 | 8.440226e+01 | 4.016792e+02 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | 2.015000e+03 |
| zipcode | 21613.0 | 9.807794e+04 | 5.350503e+01 | 9.800100e+04 | 9.803300e+04 | 9.806500e+04 | 9.811800e+04 | 9.819900e+04 |
| lat | 21613.0 | 4.756005e+01 | 1.385637e-01 | 4.715590e+01 | 4.747100e+01 | 4.757180e+01 | 4.767800e+01 | 4.777760e+01 |
| long | 21613.0 | -1.222139e+02 | 1.408283e-01 | -1.225190e+02 | -1.223280e+02 | -1.222300e+02 | -1.221250e+02 | -1.213150e+02 |
| sqft\_living15 | 21613.0 | 1.986552e+03 | 6.853913e+02 | 3.990000e+02 | 1.490000e+03 | 1.840000e+03 | 2.360000e+03 | 6.210000e+03 |
| sqft\_lot15 | 21613.0 | 1.276846e+04 | 2.730418e+04 | 6.510000e+02 | 5.100000e+03 | 7.620000e+03 | 1.008300e+04 | 8.712000e+05 |

### **Assumtions based on data analysis**

We arrive at following assumptions based on data analysis done so far. We may validate these assumptions further before taking appropriate actions.

### **Correlating**

We want to know how well does each feature correlate with Price. We want to do this early in our project and match these quick correlations with modelled correlations later in the project.

### **Completing**

Since there are no missing values we do not need to complete any values.

### **Correcting**

Id feature may be dropped from our analysis since it does not add value. Date feature may be dropped since we are going to do feature engineering and make a year and month column. Zipcode feature is a special case, we could use it, but since we do not know exactly the zones of King County we are just going to drop it.

### **Creating**

We may want to create a new feature called Year based on Date to analyze the price change throughout the years. We may want to create a new feature called Month based on Date to analyze the price change throughout the months.

### **Price correlation**

* This allow us to explore labels that are highly correlated to the price.
* sqft\_living looks like a highly correlated label to the price, as well as grade, sqft\_above, sqft\_living15 and bathrooms.

price 1.000000

sqft\_living 0.702035

grade 0.667434

sqft\_above 0.605567

sqft\_living15 0.585379

bathrooms 0.525138

view 0.397293

sqft\_basement 0.323816

bedrooms 0.308350

lat 0.307003

waterfront 0.266369

floors 0.256794

yr\_renovated 0.126434

sqft\_lot 0.089661

sqft\_lot15 0.082447

yr\_built 0.054012

condition 0.036362

long 0.021626

id -0.016762

zipcode -0.053203

Name: price, dtype: float64

### **Price feature**

* Most of the house prices are between $0 and \$1,500,000.
* The average house price is $540,000.
* Keep in mind that it may be a good idea to drop extreme values. For instance, we could focus on house from $0 to \$3,000,000 and drop the other ones.
* It seems that there is a positive linear relationship between the price and sqft\_living.
* An increase in living space generally corresponds to an increase in house price.

### **Bedrooms and floors box plots**

Box plot is a method for graphically depicting groups of numerical data through their quartiles. Box plots may also have lines extending from the boxes (whiskers) indicating variability outside the upper and lower quartiles, hence the terms box-and-whisker plot. Outliers may be plotted as individual points. The spacings between the different parts of the box indicate the degree of dispersion (spread).

* We can see outliers plotted as individual points; this probably are the more expensive houses.
* We can see that the price tends to go up when the house has more bedrooms.

Out[12]:

[Text(0, 0.5, 'Price'),

Text(0.5, 0, 'Floors'),

Text(0.5, 1.0, 'Floors vs Price Box Plot')].

### **Predicting on brand new data**

In this part we are giving the model the test set to get a list of predictions. Then we compare the correct values with the list of predictions. We use different metrics to compare the predictions, in this case we use MAE, MSE, RMSE and Variance Regression Score.

Let us start by analyzing the MAE, which is $103,500. This means that our model is off on average about \$100,000.

## Predicting on a brand new house

We are going to use the model to predict the price on a brand-new house. We are going to choose the first house of the data set and drop the price. single\_house is going to have all the features that we need to predict the price. After that we need to reshape the variable and scale the features.

The original price is $221,900 and the model prediction is \$280,000.

Features of new house:

bedrooms 3.0000

bathrooms 1.0000

sqft\_living 1180.0000

sqft\_lot 5650.0000

floors 1.0000

waterfront 0.0000

view 0.0000

condition 3.0000

grade 7.0000

sqft\_above 1180.0000

sqft\_basement 0.0000

yr\_built 1955.0000

yr\_renovated 0.0000

lat 47.5112

long -122.2570

sqft\_living15 1340.0000

sqft\_lot15 5650.0000

month 10.0000

year 2014.0000

Name: 0, dtype: float64

Prediction Price: 285858.25

Original Price: 221900.0.