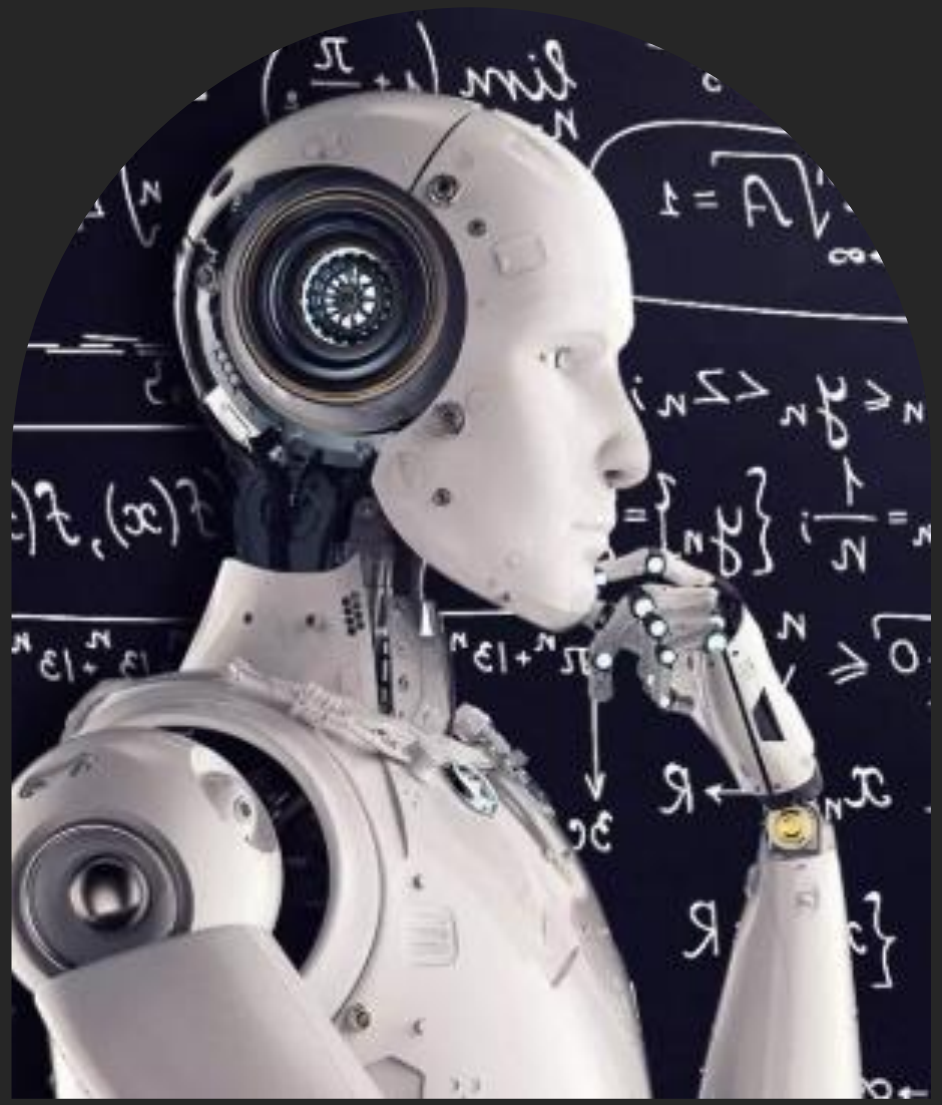


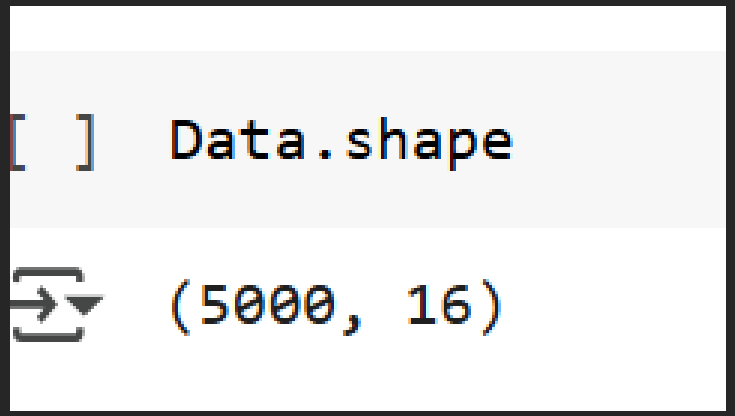
Prediction of price and sqft_area using Raw house data

AN INVESTOR APPROACH

Introduction

We have a dataset with 5000 rows and 16 columns
Our attributes are:

MLS, sold_price, zipcode, longitude, latitude, lot_acres, taxes, year_built, bedrooms, bathrooms, sqrt_ft, garage, kitchen_features, fireplaces, floor_covering, HOA



Columns: [MLS, sold_price, zipcode, longitude, latitude, lot_acres, taxes, year_built, bedrooms, bathrooms, sqrt_ft, garage, kitchen_features, fireplaces, floor_covering, HOA]

Data Cleaning

I have cleaned the data and exported into csv.

I am using the cleaned data from previous Data Cleaning and EDA as my input in this model.

New Attributes

Added new features :

pps(Price Per Sqft)

Binned the pps values in intervals of 100 each ranging from 1 to 7

Bin range 1 refers to the records having price per sqft ranging from 50$ to 150$ and,
Bin range 7 refers to the records having price per sqft ranging from 650$ to 750$.

```
selected_coloumns = ["sold_price",
                     "longitude",
                     "latitude",
                     "lot_acres",
                     "taxes",
                     "year_built",
                     "bedrooms",
                     "bathrooms",
                     "sqrt_ft",
                     "garage",
                     "fireplaces",
                     "HOA",
                     "sold_price_normal",
                     "sqrt_ft_normal",
                     "pps",
                     "bin_label"]

Data_subset = Data_cleaned[selected_coloumns]
```

Splitting Train and Test sets

I have splited the data in 80:20 ratio, with the first 80% of Data as Train data and last 20% of Data as Test data

I have taken the target variable as Bin_range.

```
X_train = X[:int(0.8 * len(X))]

y_train =y[:int(0.8* len(y))]

X_test = X[int(0.8 * len(X)):]
y_test = y[int(0.8 * len(y)):]

X1_train = X_train[:, (0,1)]
```

KNN

Developed a KNN algorithm, to predict the bin labels based on the latitude and longitude.

Achieved a Training accuracy of 81.38% and testing accuracy of 70.51%

```
accuracy(y_train, y_hat_train)
```

```
np.float64(0.8138881797293848)
```

```
accuracy(y_test, y_hat_test)
```

```
np.float64(0.7051020408163265)
```

Investors POV

Investors like to buy a house in a busy place, more land/sq.ft and less taxes

Most of the investors, just invest in the place and then, use it for revenue generation

Investors tend to pay nothing/bare minimum HOA, as they don't live there.

My Approach

Investors give me the coordinates of the location, they are looking to purchase home.

My KNN model will find the bin range for that coordinates, resulting in finding the price per sqft in that area.

But, not only coordinates decide the expected sq.ft for the the price investor wants to invest.

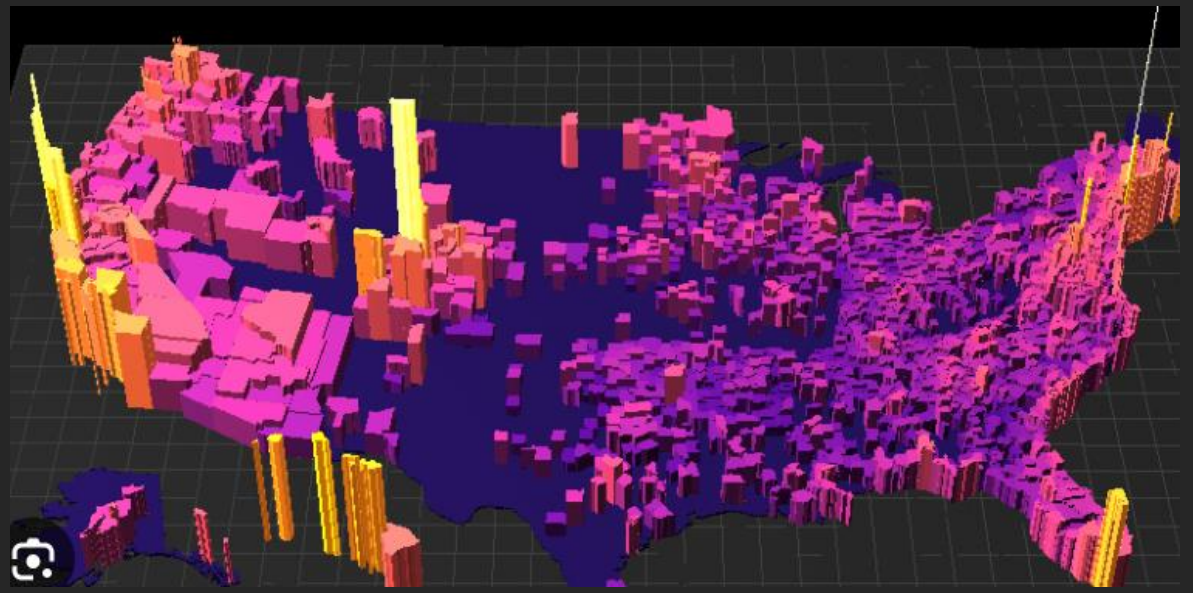
There are also other factors such as number of bedrooms, number of bathrooms and fireplaces/garages, he wants.

My Approach

So, my model needs to predict the sqft which investor can buy with his savings/ investment, number of bedrooms, bathrooms, garages, and fireplaces he wants in the home.

So, I took Price per sq.ft as my target variable.

Since, PPS is a continuous variable, I used MultiVariate Linear Regression giving the inputs bedrooms, bathrooms, garage, HOA, Fireplaces.



Why only those features?

Bedrooms, Bathrooms and garages/Fireplaces are crucial in any household and are the main factors to decide the rental income.

These are the basic features any family looks while renting a home. So, Investor is looking specifically on these features.

And, densely populated area has high probability of people to take home for rent. So, longitude, latitude and zipcode too matters.

Result

I have implemented the model in my approach.

Investor gives Latitude, longitude, sold_price, lot_acres, year_built, bedrooms, bathrooms, garages, fireplaces and HOA.

My model predicts the sqft, he can get with the money in a busiest area.

```
sqft_predict_MV_KNN(-110.378200,31.356362,5300000.0,2154.00,1941,13,10.0,0.0,6.0,0.0)
```

```
array([28067.97974889])
```