PROJECT:- HOUSE PRICE PREDICTION MODEL

MACHINE LEARNING BASED PYTHON PROJECT

TECHNO INDIA
UNIVERSITY

FINAL YEAR PROJECT

Teacher name- AB Choudhury(HOD), Soumyadeep Das

IMCA-5TH YEAR

DEPARTMENT OF COMPUTER APPLICATION

TECHNO INDIA UNIVERSITY ,WEST BENGAL

ACKNOWLEDGEMENTS

I would like to express our deep gratitude to Professor AB Choudhury(HOD) and Professor Tapas Kumar Mitra, our project supervisors Lecturer Soumyadeep Das, for their patient guidance, enthusiastic encouragement and useful critiques of this project work. I would also like to thank Lec. Soumyadeep Das, for his advice and assistance in keeping my progress on schedule. My grateful thanks are also extended to other professor for their help in doing the meteorological data analysis, helped us in this AI&ML project.

I would also like to extend our thanks to laboratory assistance of the computer application department for their help in this project work.

Finally, I wish to thank our parents for their support and encouragement throughout our study.

CERTIFICATE

This is to certify that this project **. "HOUSE PRICE PREDICTION MODEL".** in **"AI&ML BASED PYTHON PROJECT".**in the year **2022** is a bonafide record of work done at **"TECHNO INDIA UNIVERSITY, WESTBENGAL"** by the following Students

Id	Name	Sign
171001091010	**SASWATA DHAR**	
171001091001	**SAYANI SENGUPTA**	
171001091004	**SUBHAJIT DAS**	
171001091002	**ADITI DAS**	
171001102259	**ANAITA BANERJEE**	

Under our guidance and supervision and submitted in partial fulfillment of then requirements of the **FINAL YEAR PROJECT-2022**.

(Signature of the Project Mentor)
Mr. Soumyadeep Das

(Signature of the Head Of Department)
Prof.Anil Bikash Chowdhury
TECHNO INDIAUNIVERSITY
WEST BENGAL

(Signature of theAcademic Coordinator)
Mr. Ishan Ghosh
Techno India University, West Bengal

ABSTRACT

The development of a housing prices prediction model can assist a house seller or a real estate agent to make better-informed decisions based on house price valuation. Only a few works report the use of machine learning (ML) algorithms to predict the values of properties . This study analyses a dataset composed of 502 housing , collected from different websites from 2015 to 2018. Each instance comprises fourteen features of types: integer. To predict the property prices, we ensemble different ML architectures, based on Random Forest (RF) ,Decision tree, linear regression . This study demonstrates that enriching the dataset and combining different ML approaches can be a better alternative for prediction of housing prices.

CONTENT

INTRODUCTION

A housing market can be understood as any market for properties which are negotiated either directly from their owners to buyers, or through the services of real state brokers. People and companies are drawn to this market, which presents many profit opportunities that come from housing demands worldwide. These demands are influenced by several factors, such as demography, economy, and politics. For this reason, the analysis of such markets has been challenging data scientists and ML engineers around the world, as they must take into account a wide range of scientific fields, each one addressing different kinds of data , to come up with accurate results to customers and stakeholders .

The prediction of housing prices as they vary through time and place is a complex task, particularly when the available data is massive and divided into many different data types. As such, it was an appropriate choice for the Data Science Challenge at Engineering Education for the Future .The data for training and testing the models was composed of 502 housing data.

In this research we used supervised learning as the feature and level is included with our working.so this research is easy to built. And also we use regression technique as the dataset we use is numeric and our object is only predict house price so that we can tally with the level and find correct price data, we use several regression method with rmse factor and from them we use the best regression method to built the model. And lastly we use batch learning as we build this model with the static data . so if any ambiguity of model found in future then we can easily train our model with future data.

RELATED WORK

By the decade of 1990, the increasing popularity of the Internet made it suitable for hosting advertisements which previously were published in newspapers and magazines. As of today, there is a huge number of property advertisements in the Web and exploiting knowledge from them is a topic which is observed in the recent ML literature [Wu and Brynjolfsson 2015, Sirignano et al. 2016]. Due to the large amounts of data from these advertisements, deep learning approaches seem to perform feature extraction for housing prices prediction with good performance [Poursaeed et al. 2018]. Statistical models have been a common approach to analyze and predict property prices for a long time. In [Fik et al. 2003] a study to explain the housing prices variation was conducted by analyzing the influence of location features on the property prices. [Goodman and Thibodeau 2003] employed hierarchical linear models.

[Raudenbush and Bryk 2002] for estimating housing prices in Dallas County, USA, using data extracted from 28,000 single-family transactions in the period between 1995 and 1997. Among the contributions of this study, [Goodman and Thibodeau 2003] argue that housing market is segmented by the quality of public education, which is not properly a property attribute. Linear Regression (LR) and Artificial Neural Networks (ANN) algorithms were successfully applied to housing prices prediction by [Wu and Brynjolfsson 2015] and [Selim 2009], respectively. In the former, the authors collected data from Google searches to predict property prices based on advertisement frequency. On the other hand, the latter used an ANN architecture composed of input, hidden, and output layers that yielded better price prediction results than the statistical method of Hedonic Regression. Furthermore, [Selim 2009] compared the influence of many variables on housing prices variation. The most significant variables were found to be: water system, pool, type of house, type of building, number of rooms, house size, characteristic of the location. Decision Trees, Random Forest (RF), and Support Vector Regression (SVR) have also been successfully employed on this task. In [Park and Bae 2015], the C4.5 decision tree, Repeated Incremental Pruning to Produce Error Reduction (RIPPER), Na¨ıve Bayes, and AdaBoost were used to predict housing prices on data from Multiple Listing Service, historical mortgage rates, and public school ratings. As a result, RIPPER outperformed all the other models, due to its strategy of selecting more recurrent values as default outcomes at the same time as learning rules for the lowest values. Another multi-model comparison for housing prices prediction was performed in [Fan et al. 2018], using RF, SVD, Ridge Linear Regression, Lasso Linear Regression, and XGB algorithms. Deep learning architectures are a rising research topic because of their high performances on massive amounts of data.

IMPORTANCE OF ML & DEEP LEARNING

In this section we are looking for importance of ML and deep learing for building model for project. We are dicused about ML aand Deep Learning concept for the building of Project Model

What is ML?

➔ **Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.**

Why ML important for this project?

➔ **Previously this type of research done by manually so there may be a chance of error. A research said due to manually doing this type of jobs 25-30% error occurs in predicting correct price . For this reason there might be occurring big loss off customer . Not only that as we have a gigantic data house but due to manually solving it may consume huge amount of time. So Machine learning and Data Science important for this job.**

How do we implicate machine learning in this project?

➔ **We can implicate ML by using R-Programming, Python programming, java programming etc. but off them R & Python are the best as both of them is easy to use open source.
In this project we use Python as Python work bench is better than R-Programming, Not only that Python have Jupyter Notebook which give a good frame work to develop this project .**

What is Deep Learning?

➔ **Deep learning algorithms run data through several "layers" of neural network algorithms, each of which passes a simplified representation of the data to the next layer.**

Most machine learning algorithms work well on datasets that have up to a few hundred features, or columns. However, an unstructured dataset, like one from an image, has such a large number of features that this process becomes cumbersome or completely unfeasible. A single 800-by-1000-pixel image in RGB color has 2.4 million features – far too many for traditional machine learning algorithms to handle.

Why is Deep Learning Important?

➔ The ability to process large numbers of features makes deep learning very powerful when dealing with unstructured data. However, deep learning algorithms can be overkill for less complex problems because they require access to a vast amount of data to be effective. For instance, ImageNet, the common benchmark for training deep learning models for comprehensive image recognition, has access to over 14 million images.
If the data is too simple or incomplete, it is very easy for a deep learning model to become overfitted and fail to generalize well to new data. As a result, deep learning models are not as effective as other techniques (such as boosted decision trees or linear models) for most practical business problems such as understanding customer churn, detecting fraudulent transactions and other cases with smaller datasets and fewer features. In certain cases like multiclass classification, deep learning can work for smaller, structured datasets.

TOOLS AND TECHNIQUE USED

In this project we used we used Python programming language .

For creating this model we use several pip library which is used for machine learning algorithm.

This model is building in supervised learning technique. And also we here used batch learning technique also

Some pip install for constructing this model are:

- **Jupyter notebook as IDE for developing the model.**
- **Pandas for working with data to build the model.**
- **Matplotlib for graphical plotting of data to showing histogram.**
- **Numpy for making mathematical and logical operation with data.**
- **Scikit learn for building and training the model.**

Apart from that we use UCI Machine learning Repository . From where we make our survey for building this model and getting working Data for building the Model.

What is Supervised Learning?

- **Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately, which occurs as part of the cross validation process. Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox.**

What is batch learning?

- **Batch learning represents the training of machine learning models in a batch manner. In other words, batch learning represents the training of the models at regular intervals** such as weekly, bi-weekly, monthly, quarterly, etc. The data gets accumulated over a period of time. The models then get trained with the accumulated data from time to time at periodic intervals. Batch learning is also called **offline learning**. The models trained using batch or offline learning are moved into production only at regular intervals based on the performance of models trained with new data.

Why Python?

➔ A Python library is a group of pre-built coded modules that help you complete common tasks with fewer lines of code. Instead of coding from scratch, you can load tools to help you with data visualization, analysis, cleaning, and machine learning.
Not only that python is open source, easy to used and have an enormous set of library for any ML project.

What is Jupyter NoteBook?

➔ The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

What is Pandas?

➔ Pandas is an open-source library that is made mainly for working with relational or labeled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

What is Scikit Learn?

➔ Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

What is Matplotlib?

➔ Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

What is NumPY?

➔ Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

MATERIAL AND METHOD

This section presents the details concerning the data and tools used in the study. The choice of the algorithms to perform housing prices prediction has taken into account the nature of the dataset, since real-world data are not always balanced, complete, scaled, or easy to handle

Dataset for model:-

- ➔ **Dataset in Python is mostly used for manipulation of Gifs and other custom data which frames the entire dataset as per requirement. It helps in maintaining the order and simplifying the complex data structure for further manipulation or enhancement. Dataset in any format is mostly used for many other necessities that streamline the process.**
- ➔ **In this project we use dataset of 506 house of them we use 404 house data for training our model and 102 as testing data. Ratio we use for division is 75%-25%.Discuss later.**

Missing Attribute :-

- **The real-world data often has a lot of missing values. The cause of missing values can be data corruption or failure to record data. The handling of missing data is very important during the preprocessing of the dataset as many machine learning algorithms do not support missing values.**

  This article covers 7 ways to handle missing values in the dataset:

    - **Deleting Rows with missing values**
    - **Impute missing values for continuous variable**
    - **Impute missing values for categorical variable**
    - **Other Imputation Methods**
    - **Using Algorithms that support missing values**
    - **Prediction of missing values**
    - **Imputation using Deep Learning Library — Datawig**
- **In this project we use Impute mean /median for missing value.**

- What is Impute mean/median for missing value?

  ➔ Columns in the dataset which are having numeric continuous values can be replaced with the mean, median, or mode of remaining values in the column. This method can prevent the loss of data compared to the earlier method. Replacing the above two approximations (mean, median) is a statistical approach to handle the missing values.

Method used for model:-

- **Mainly For supervised learning there is two method used**
  A. **Regression**
  B. **Classification.**

  Here we use Regression method as we only want to predict the price from the model.

- What is Regression?

  ➔ Regression is a method to determine the statistical relationship between a dependent variable and one or more independent variables. The change independent variable is associated with the change in the independent variables. This can be broadly classified into two major types.

  - **Linear Regression**
  - **Logistic Regression**

Performance measure:-

- A typical measure for regression model is **Root-mean-square-error(RMSE).**
- We can also use Mean absolute error, Manhattan norm etc.

- What is Root –mean –square-error(RMSE)?

  ➔ Statistically, the **root mean square (RMS)** is the square root of the mean square, which is the arithmetic mean of the squares of a group of values. RMS is also called a quadratic mean and is a special case of the generalized mean whose exponent is 2. Root mean square is also defined as a varying function based on an integral of the squares of the values which are instantaneous in a cycle.

  In other words, the RMS of a group of numbers is the square of the arithmetic mean or the function's square which defines the continuous waveform.

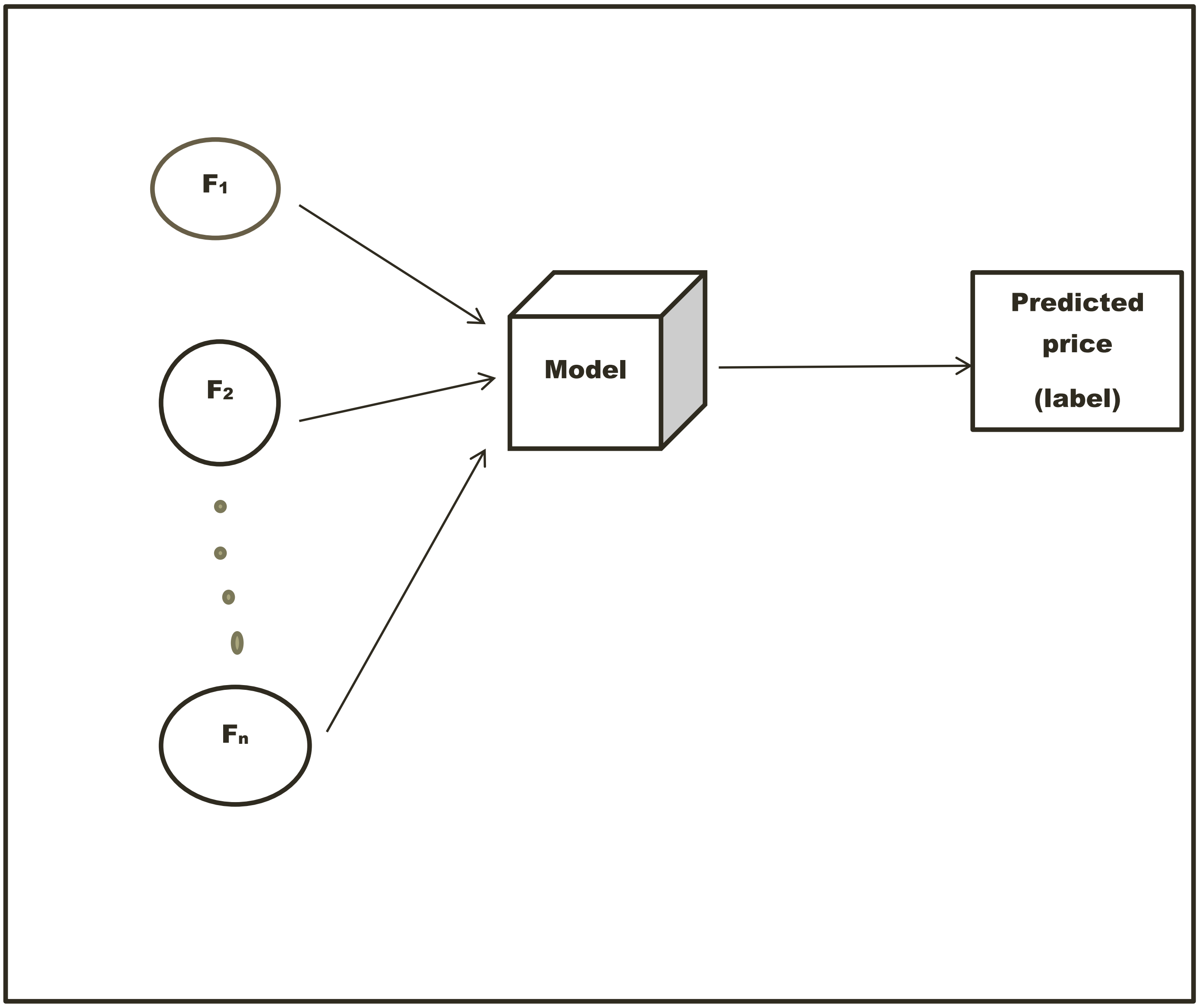
  Formula:-

  The RMSE of a predicted model with respect to the estimated variable x_{model} is defined as the square root of the mean squared error.

$$\boxed{y = \frac{1}{n}\sum_{i=1}^{n}\left(\sqrt{\mathrm{xobs} - \mathrm{xmodel}}\right)^2}$$

  Where, x_{obs} is observed values, x_{model} is modelled values at time i.

ARCHITECTURE OF MODEL



Here F_1 ,F_2 ……. F_n are the Feature by which we train our model we find the label I,e predicted price.

As this process build in regression technique so we can't again classify that predicted price into cheap or expensive.

PROBLEM STATEMENT

A house value is simply more than location and square footage. Like the features that make up a person, an educated party would want to know all aspects that give a house its value. For example, you want to sell a house and you don't know the price which you can take — it can't be too low or too high.

To find house price you usually try to find similar properties in your neighbourhood and based on gathered data you will try to assess your house price.

Now our main object is to find only price. But if the price is re-categorised into expensive or cheap. Then its not under supervised learning.

If latter is the case ,formulating the problem as regression task will be big problem.

So we have to first clear that the model build only to predict the price value.

In this project we don't use any type of image data we only use numerical data. Here we use 14 attribute for training our model. And as we use batch learning technique so we build our model one time . And train our model with the current data only not we creating pipeline where data come from outside and the model train with the data from outside. This system is under Online learning Technique.

So now we train data from current attribute if any changes come after some year we again train our model.

SURVEY OF DATA FOR PROJECT

We make survey on boston housing complex data and build our project.

Details :-

Sources:
(a) Origin: This dataset was taken from the StatLib library which is
maintained at Carnegie Mellon University.
(b) Creator: Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the
demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.
(c) Date: July 7, 1993

Past Usage:
- Used in Belsley, Kuh & Welsch, 'Regression diagnostics ...', Wiley,
1980. N.B. Various transformations are used in the table on
pages 244-261.
- Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning.
In Proceedings on the Tenth International Conference of Machine
Learning, 236-243, University of Massachusetts, Amherst. Morgan
Kaufmann.

Relevant Information:

Concerns housing values in suburbs of Boston.

Number of Instances: 506

Number of Attributes:
13 continuous attributes (including "class" attribute "MEDV"), 1 binary-valued attribute.

Attribute Information:

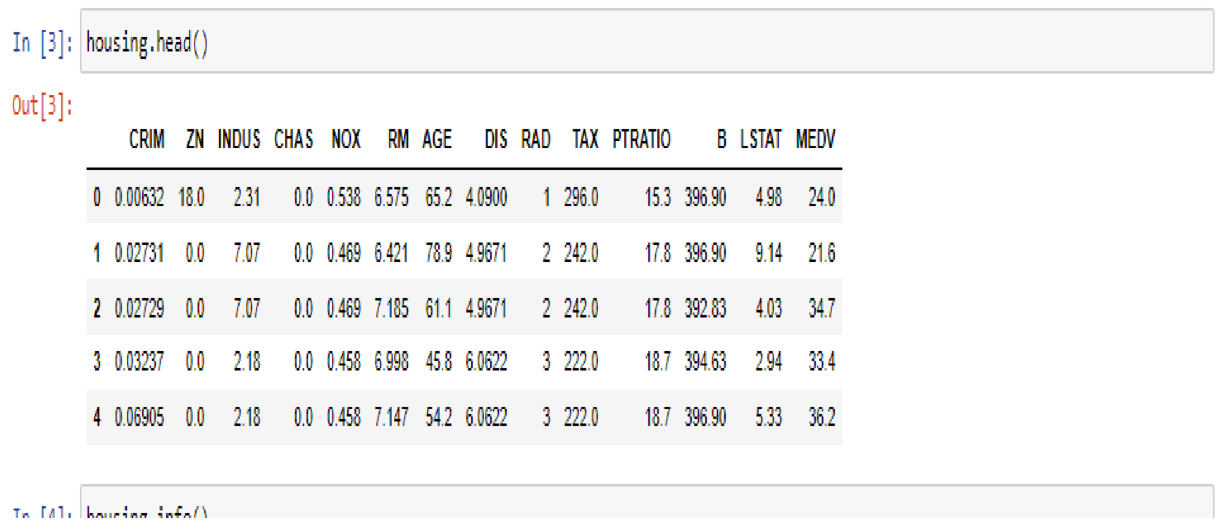
1. **CRIM-** per capita crime rate by town
2. **ZN -** proportion of residential land zoned for lots over 25,000 sq.ft.
3. **INDUS -** proportion of non-retail business acres per town
4. **CHAS -** Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
5. **NOX-** nitric oxides concentration (parts per 10 million)
6. **RM -** average number of rooms per dwelling
7. **AGE -** proportion of owner-occupied units built prior to 1940
8. **DIS -** weighted distances to five Boston employment centres
9. **RAD-** index of accessibility to radial highways
10. **TAX -** full-value property-tax rate per $10,000
11. **PTRATIO** - pupil-teacher ratio by town
12. **B -** 1000(Bk - 0.63)^2 where Bk is the proportion of blacks
by town
13. **LSTAT-** % lower status of the population
14. **MEDV-** Median value of owner-occupied homes in $1000's

8. Missing Attribute Values: YES RM AND MEDV.

STEP TO BUILD MODEL

Data :-

We give first four data from our current dataset.



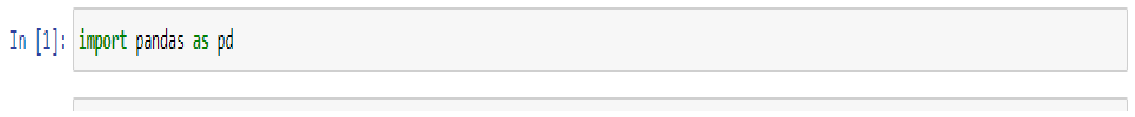
```
In [3]: housing.head()

Out[3]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2

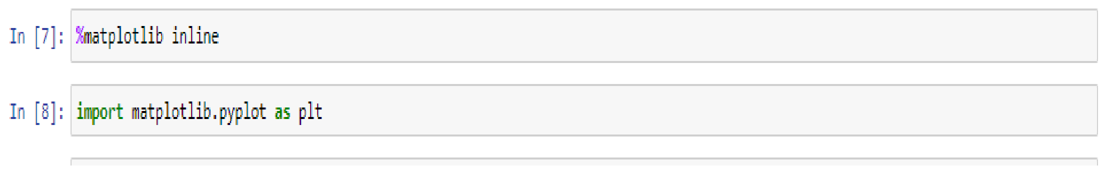
```
In [4]: housing.info()
```

Step1:

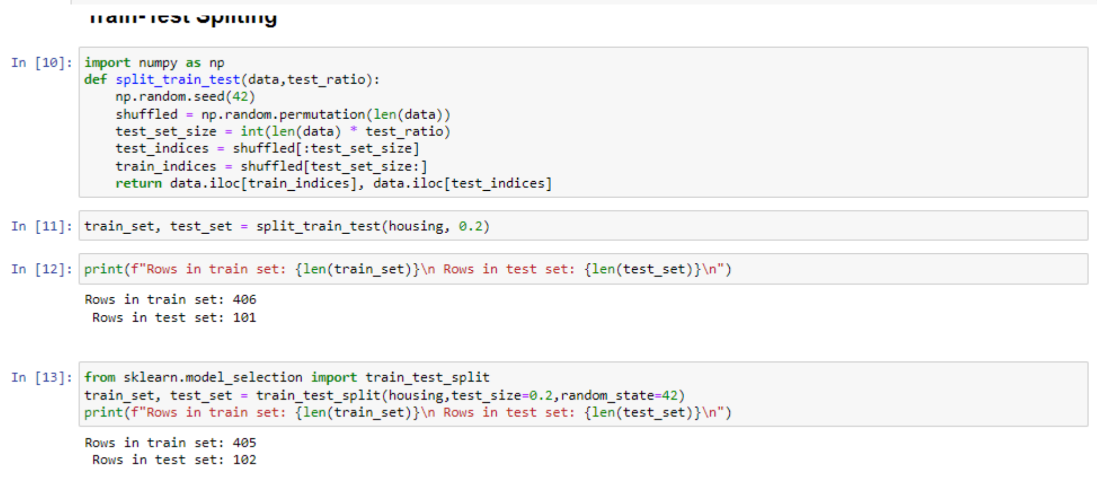
Importing all library for Project



```
In [1]: import pandas as pd
```



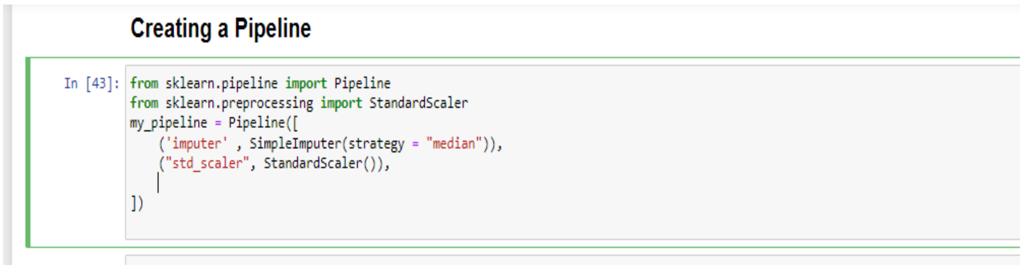
```
In [7]: %matplotlib inline

In [8]: import matplotlib.pyplot as plt
```



Train-Test Splitting

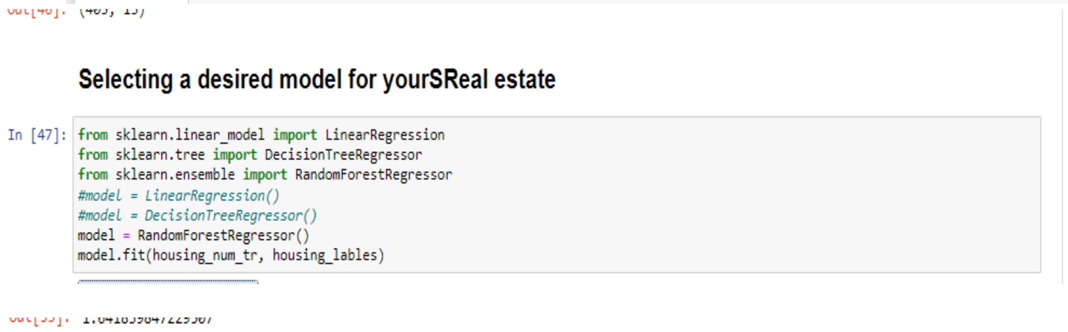
```
In [10]: import numpy as np
         def split_train_test(data,test_ratio):
             np.random.seed(42)
             shuffled = np.random.permutation(len(data))
             test_set_size = int(len(data) * test_ratio)
             test_indices = shuffled[:test_set_size]
             train_indices = shuffled[test_set_size:]
             return data.iloc[train_indices], data.iloc[test_indices]

In [11]: train_set, test_set = split_train_test(housing, 0.2)

In [12]: print(f"Rows in train set: {len(train_set)}\n Rows in test set: {len(test_set)}\n")

         Rows in train set: 406
          Rows in test set: 101

In [13]: from sklearn.model_selection import train_test_split
         train_set, test_set = train_test_split(housing,test_size=0.2,random_state=42)
         print(f"Rows in train set: {len(train_set)}\n Rows in test set: {len(test_set)}\n")

         Rows in train set: 405
          Rows in test set: 102
```

from pandas.plotting import scatter_matrix



Creating a Pipeline

```
In [43]: from sklearn.pipeline import Pipeline
         from sklearn.preprocessing import StandardScaler
         my_pipeline = Pipeline([
             ('imputer' , SimpleImputer(strategy = "median")),
             ("std_scaler", StandardScaler()),
             |
         ])
```

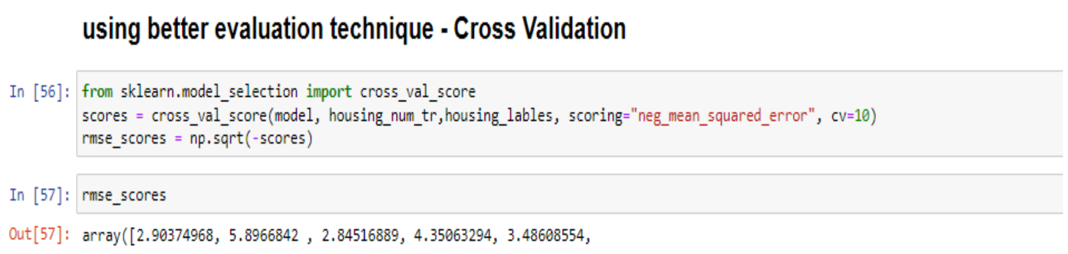


```
Out[46]: (405, 13)
```

Selecting a desired model for yourSReal estate

```
In [47]: from sklearn.linear_model import LinearRegression
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.ensemble import RandomForestRegressor
         #model = LinearRegression()
         #model = DecisionTreeRegressor()
         model = RandomForestRegressor()
         model.fit(housing_num_tr, housing_lables)
```

```
Out[55]: [illegible]
```

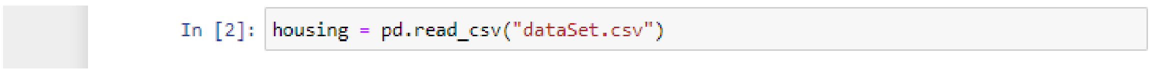


using better evaluation technique - Cross Validation

```
In [56]: from sklearn.model_selection import cross_val_score
         scores = cross_val_score(model, housing_num_tr,housing_lables, scoring="neg_mean_squared_error", cv=10)
         rmse_scores = np.sqrt(-scores)

In [57]: rmse_scores

Out[57]: array([2.90374968, 5.8966842 , 2.84516889, 4.35063294, 3.48608554,
```

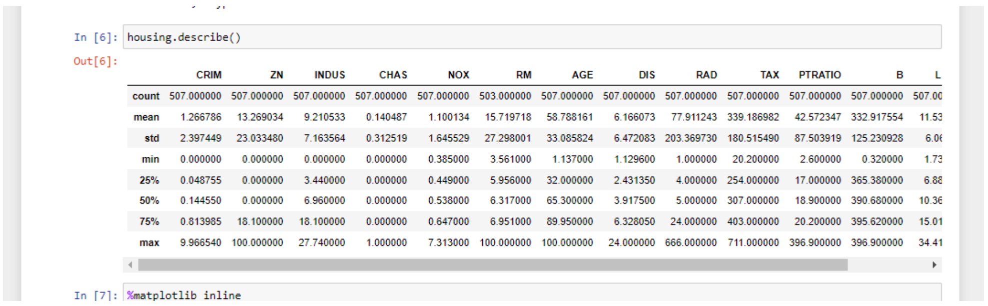
Step2:

We take the total data into a data set to manipulate the data. For crating data set we use pandas libarary.



```
In [2]: housing = pd.read_csv("dataSet.csv")
```

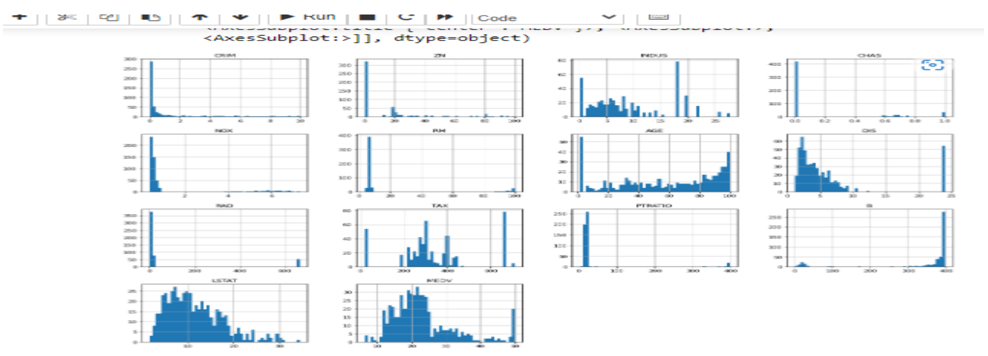
No we get the histogram of Data of the attribute in the data bank. Graphical represent the data.



```
In [6]: housing.describe()
Out[6]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
count	507.000000	507.000000	507.000000	507.000000	507.000000	503.000000	507.000000	507.000000	507.000000	507.000000	507.000000	507.000000	507.00
mean	1.266786	13.269034	9.210533	0.140487	1.100134	15.719718	58.788161	6.166073	77.911243	339.186982	42.572347	332.917554	11.53
std	2.397449	23.033480	7.163564	0.312519	1.645529	27.298001	33.085824	6.472083	203.369730	180.515490	87.503919	125.230928	6.0
min	0.000000	0.000000	0.000000	0.000000	0.385000	3.561000	1.137000	1.129600	1.000000	20.200000	2.600000	0.320000	1.73
25%	0.048755	0.000000	3.440000	0.000000	0.449000	5.956000	32.000000	2.431350	4.000000	254.000000	17.000000	365.380000	6.88
50%	0.144550	0.000000	6.960000	0.000000	0.538000	6.317000	65.300000	3.917500	5.000000	307.000000	18.900000	390.680000	10.36
75%	0.813985	18.100000	18.100000	0.000000	0.647000	6.951000	89.950000	6.328050	24.000000	403.000000	20.200000	395.620000	15.01
max	9.966540	100.000000	27.740000	1.000000	7.313000	100.000000	100.000000	24.000000	666.000000	711.000000	396.900000	396.900000	34.41

```
In [7]: %matplotlib inline
```



```
<AxesSubplot:>]], dtype=object)
```

```
array([[<AxesSubplot:title={'center':'CRIM'}>,

        <AxesSubplot:title={'center':'ZN'}>,
        <AxesSubplot:title={'center':'INDUS'}>,
        <AxesSubplot:title={'center':'CHAS'}>],
       [<AxesSubplot:title={'center':'NOX'}>,
        <AxesSubplot:title={'center':'RM'}>,
        <AxesSubplot:title={'center':'AGE'}>,
        <AxesSubplot:title={'center':'DIS'}>],
       [<AxesSubplot:title={'center':'RAD'}>,
        <AxesSubplot:title={'center':'TAX'}>,
        <AxesSubplot:title={'center':'PTRATIO'}>,
        <AxesSubplot:title={'center':'B'}>],
       [<AxesSubplot:title={'center':'LSTAT'}>,
        <AxesSubplot:title={'center':'MEDV' AxesSubplot:>]], dtype=object)}>, <AxesSubplot : >,
              <
```

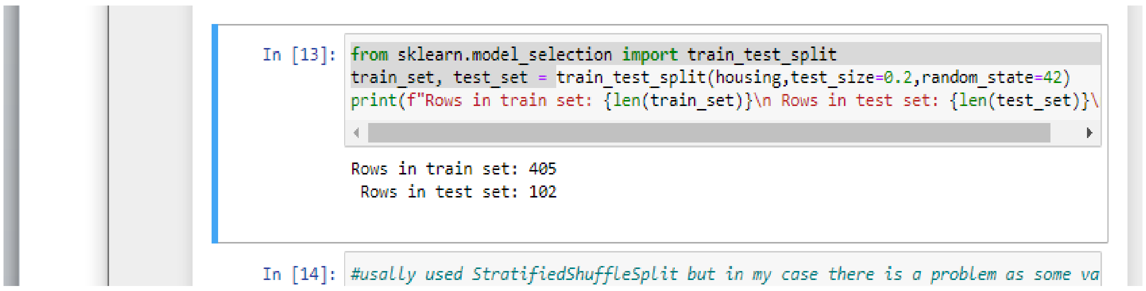
Histogram:

In statistics, a **histogram** is a graphical representation of the distribution of data. The histogram is represented by a set of rectangles, adjacent to each other, where each bar represent a kind of data. Statistics is a stream of mathematics that is applied in various fields. When numerals are repeated in statistical data, this repetition is known as Frequency and which can be written in the form of a table, called a frequency distribution.

STEP3:-

-In this step we divide our data in 75%-25% ratio. That we kept 75% of data to a dataset by which we train our model and rest 25% of data by which we use to test our model. So that we find whether our model gives our correct prediction or not.

We can either use our own method for splitting data or we use method form scikit learn.
Here we use scikit learn for splitting our data.

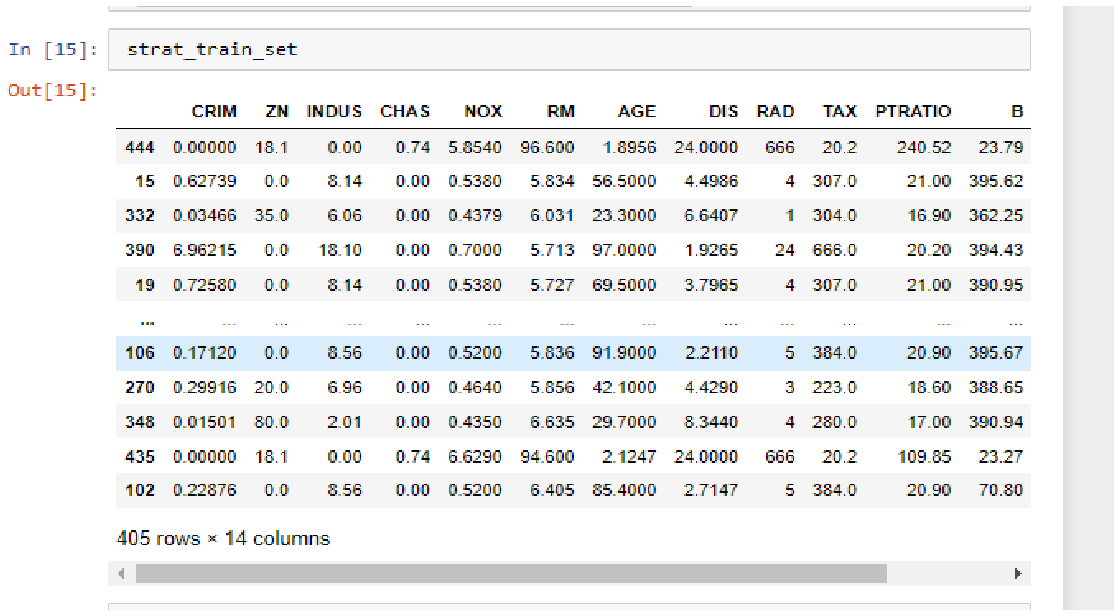


```
In [13]: from sklearn.model_selection import train_test_split
         train_set, test_set = train_test_split(housing,test_size=0.2,random_state=42)
         print(f"Rows in train set: {len(train_set)}\n Rows in test set: {len(test_set)}\

         Rows in train set: 405
          Rows in test set: 102

In [14]: #usally used StratifiedShuffleSplit but in my case there is a problem as some va
```

from sklearn.model_selection import ShuffleSplit
split = ShuffleSplit(n_splits =1, test_size = 0.2, random_state=42)
for train_index, test_index in split.split(housing, housing['CHAS']):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]

usally using StratifiedShuffleSplit for best case but my model split data by using Shuffled split method only.

Train set:

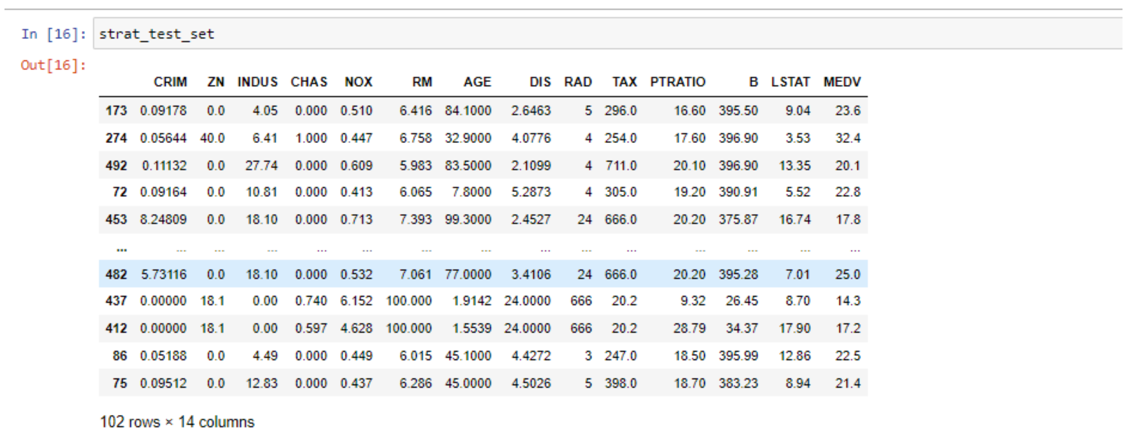


```
In [15]: strat_train_set
Out[15]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
444	0.00000	18.1	0.00	0.74	5.8540	96.600	1.8956	24.0000	666	20.2	240.52	23.79
15	0.62739	0.0	8.14	0.00	0.5380	5.834	56.5000	4.4986	4	307.0	21.00	395.62
332	0.03466	35.0	6.06	0.00	0.4379	6.031	23.3000	6.6407	1	304.0	16.90	362.25
390	6.96215	0.0	18.10	0.00	0.7000	5.713	97.0000	1.9265	24	666.0	20.20	394.43
19	0.72580	0.0	8.14	0.00	0.5380	5.727	69.5000	3.7965	4	307.0	21.00	390.95
...	...	...	...	...	...	...	...	...	...	...	...	...
106	0.17120	0.0	8.56	0.00	0.5200	5.836	91.9000	2.2110	5	384.0	20.90	395.67
270	0.29916	20.0	6.96	0.00	0.4640	5.856	42.1000	4.4290	3	223.0	18.60	388.65
348	0.01501	80.0	2.01	0.00	0.4350	6.635	29.7000	8.3440	4	280.0	17.00	390.94
435	0.00000	18.1	0.00	0.74	6.6290	94.600	2.1247	24.0000	666	20.2	109.85	23.27
102	0.22876	0.0	8.56	0.00	0.5200	6.405	85.4000	2.7147	5	384.0	20.90	70.80

405 rows × 14 columns

__Test set:__



```
In [16]: strat_test_set
Out[16]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
173	0.09178	0.0	4.05	0.000	0.510	6.416	84.1000	2.6463	5	296.0	16.60	395.50	9.04	23.6
274	0.05644	40.0	6.41	1.000	0.447	6.758	32.9000	4.0776	4	254.0	17.60	396.90	3.53	32.4
492	0.11132	0.0	27.74	0.000	0.609	5.983	83.5000	2.1099	4	711.0	20.10	396.90	13.35	20.1
72	0.09164	0.0	10.81	0.000	0.413	6.065	7.8000	5.2873	4	305.0	19.20	390.91	5.52	22.8
453	8.24809	0.0	18.10	0.000	0.713	7.393	99.3000	2.4527	24	666.0	20.20	375.87	16.74	17.8
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
482	5.73116	0.0	18.10	0.000	0.532	7.061	77.0000	3.4106	24	666.0	20.20	395.28	7.01	25.0
437	0.00000	18.1	0.00	0.740	6.152	100.000	1.9142	24.0000	666	20.2	9.32	26.45	8.70	14.3
412	0.00000	18.1	0.00	0.597	4.628	100.000	1.5539	24.0000	666	20.2	28.79	34.37	17.90	17.2
86	0.05188	0.0	4.49	0.000	0.449	6.015	45.1000	4.4272	3	247.0	18.50	395.99	12.86	22.5
75	0.09512	0.0	12.83	0.000	0.437	6.286	45.0000	4.5026	5	398.0	18.70	383.23	8.94	21.4

102 rows × 14 columns

Step 4:

Here we looking for correlation of attribute.

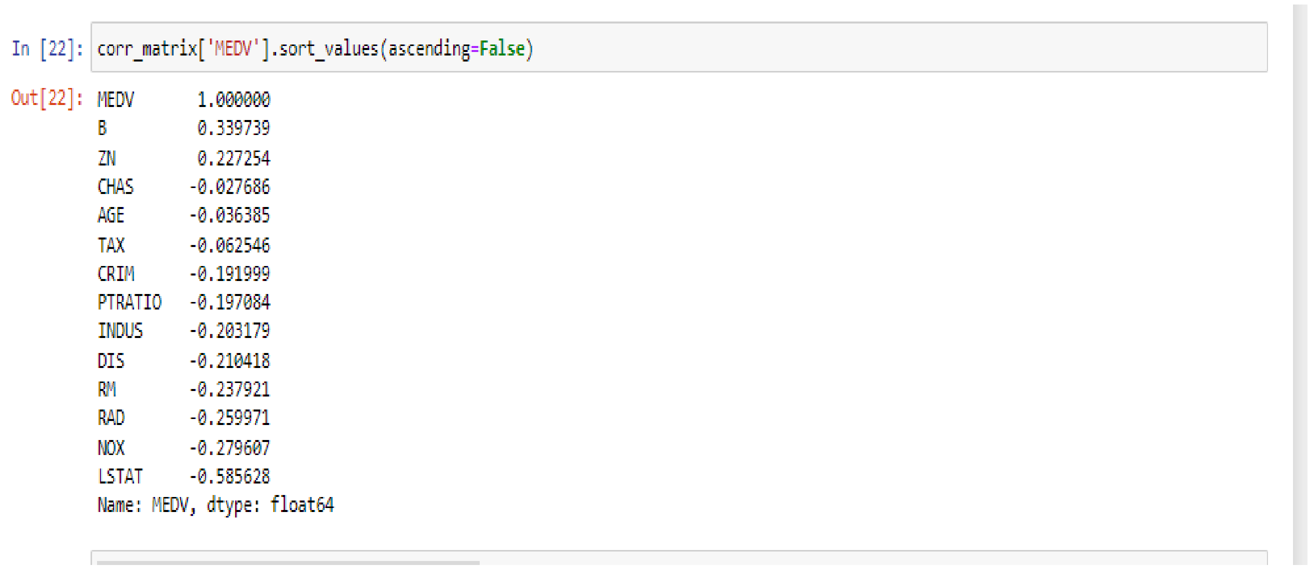
What is correlation coefficient?

➔ Correlation Coefficient is a statistical concept, which helps in establishing a relation between predicted and actual values obtained in a statistical experiment. The calculated value of the correlation coefficient explains the exactness between the predicted and actual values.

Correlation Coefficient value always lies between -1 to +1. If correlation coefficient value is positive, then there is a similar and identical relation between the two variables. Else it indicates the dissimilarity between the two variables.

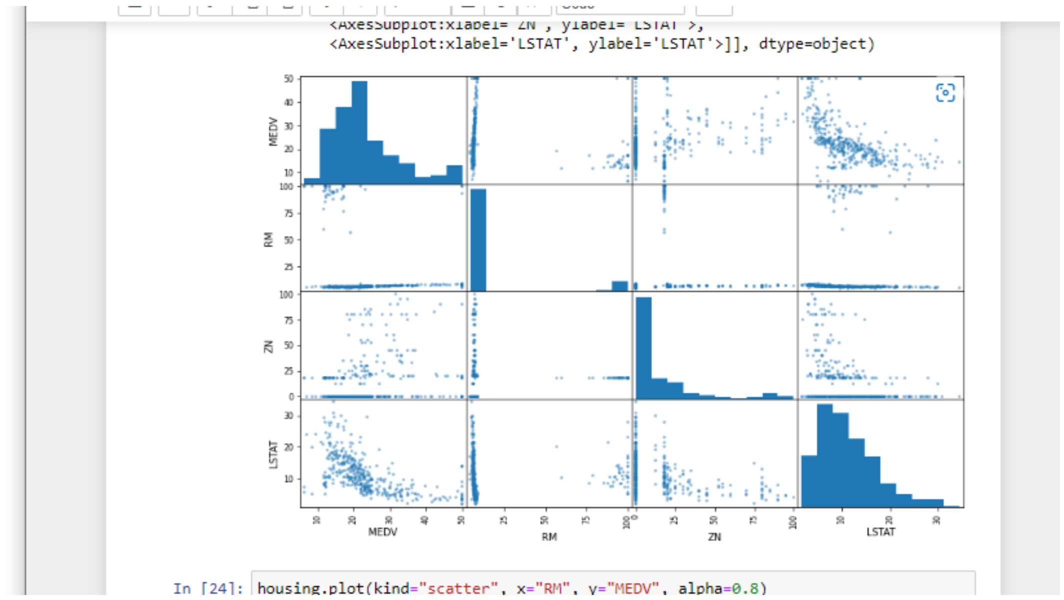
$$\boxed{\rho(X,Y)=E\,\frac{(X-\mu_X)(Y-\mu_Y)}{\sigma_X*\sigma_Y}}$$

μ_x and μ_y are mean of x and mean of y respectively. E is the expectation.



```
In [22]: corr_matrix['MEDV'].sort_values(ascending=False)

Out[22]: MEDV       1.000000
         B          0.339739
         ZN         0.227254
         CHAS      -0.027686
         AGE       -0.036385
         TAX       -0.062546
         CRIM      -0.191999
         PTRATIO   -0.197084
         INDUS     -0.203179
         DIS       -0.210418
         RM        -0.237921
         RAD       -0.259971
         NOX       -0.279607
         LSTAT     -0.585628
         Name: MEDV, dtype: float64
```

Here As MEDV is dependent variable so we find B, zn, Chas is correlated with MEDV. As the give positive value. Other all give value in negative manner .
We can clarify my determination if we see histogram of them with this correlation.

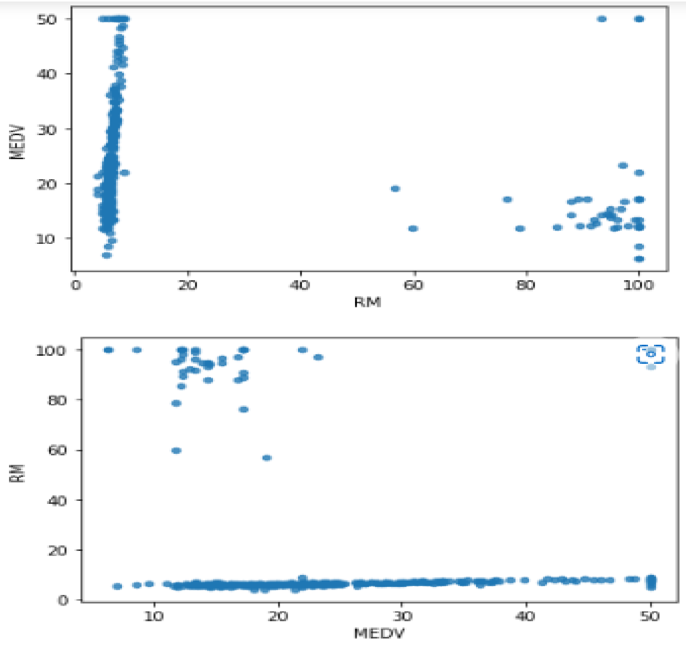


```
<AxesSubplot:xlabel='ZN', ylabel='LSTAT'>,
<AxesSubplot:xlabel='LSTAT', ylabel='LSTAT'>]], dtype=object)
```

```
In [24]: housing.plot(kind="scatter", x="RM", y="MEDV", alpha=0.8)
```

Though Rm is not effecting But we find the price according to number of room in house

Histogram plot:

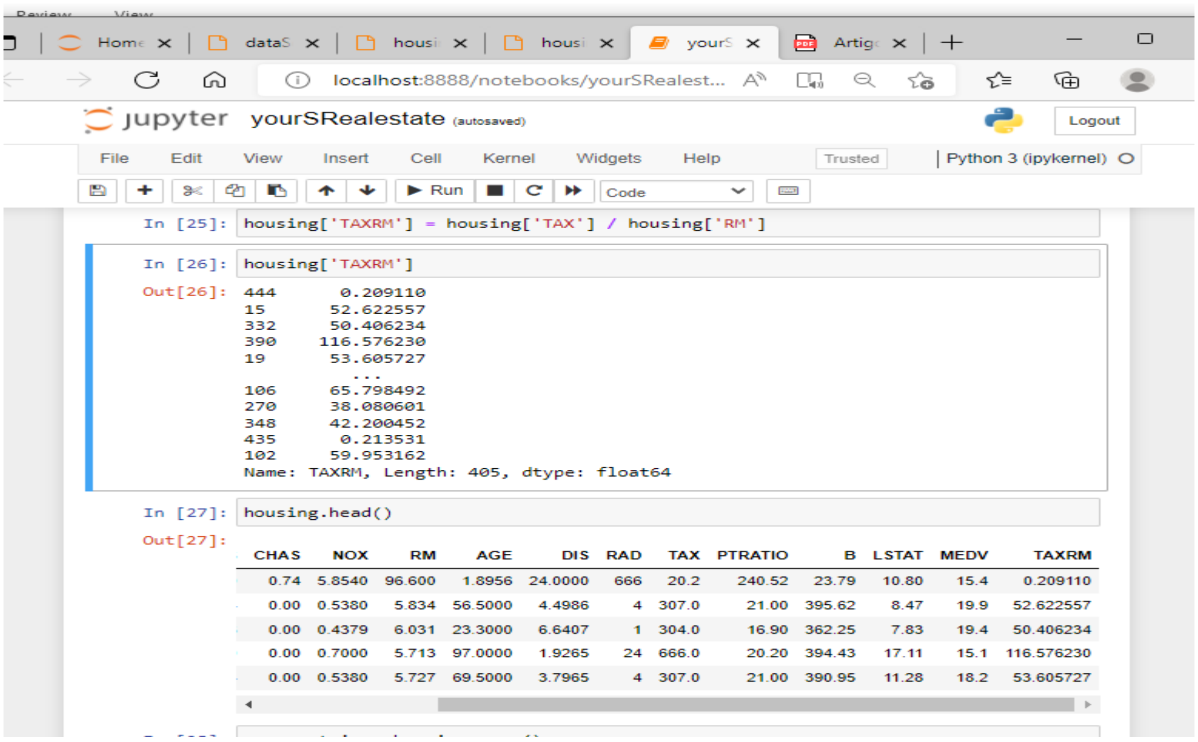


MEDV$\propto RM$

Step 5:

IN this step we see several attribute combination and its correlation with lebel.

Here we use TAX,RM and make new attribute TAXRM and find its correlation with Label.


Home
dataS
housi
housi
yourS
Artig
localhost:8888/notebooks/yourSRealest...
jupyter yourSRealestate (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [25]: housing['TAXRM'] = housing['TAX'] / housing['RM']
```

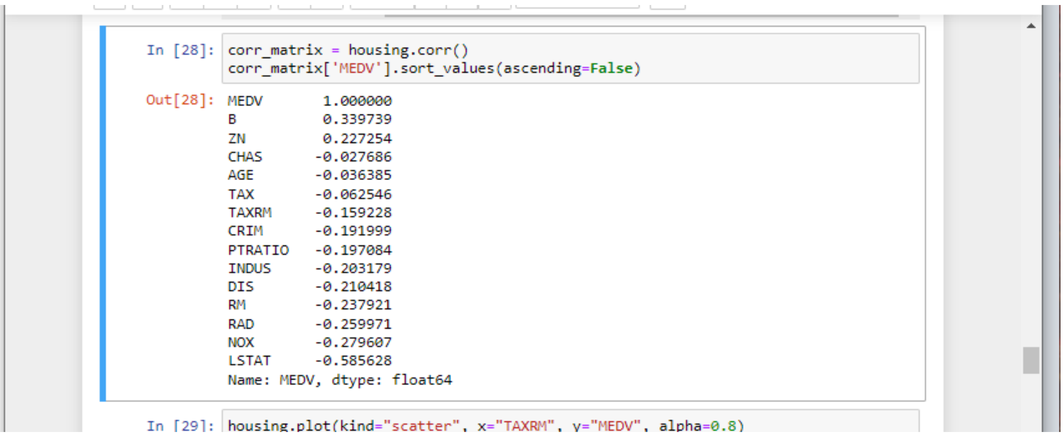
```
In [26]: housing['TAXRM']
Out[26]: 444      0.209110
         15      52.622557
         332     50.406234
         390    116.576230
         19      53.605727
                   ...
         106     65.798492
         270     38.080601
         348     42.200452
         435      0.213531
         102     59.953162
         Name: TAXRM, Length: 405, dtype: float64
```

```
In [27]: housing.head()
Out[27]:
```

CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	TAXRM
0.74	5.8540	96.600	1.8956	24.0000	666	20.2	240.52	23.79	10.80	15.4	0.209110
0.00	0.5380	5.834	56.5000	4.4986	4	307.0	21.00	395.62	8.47	19.9	52.622557
0.00	0.4379	6.031	23.3000	6.6407	1	304.0	16.90	362.25	7.83	19.4	50.406234
0.00	0.7000	5.713	97.0000	1.9265	24	666.0	20.20	394.43	17.11	15.1	116.576230
0.00	0.5380	5.727	69.5000	3.7965	4	307.0	21.00	390.95	11.28	18.2	53.605727

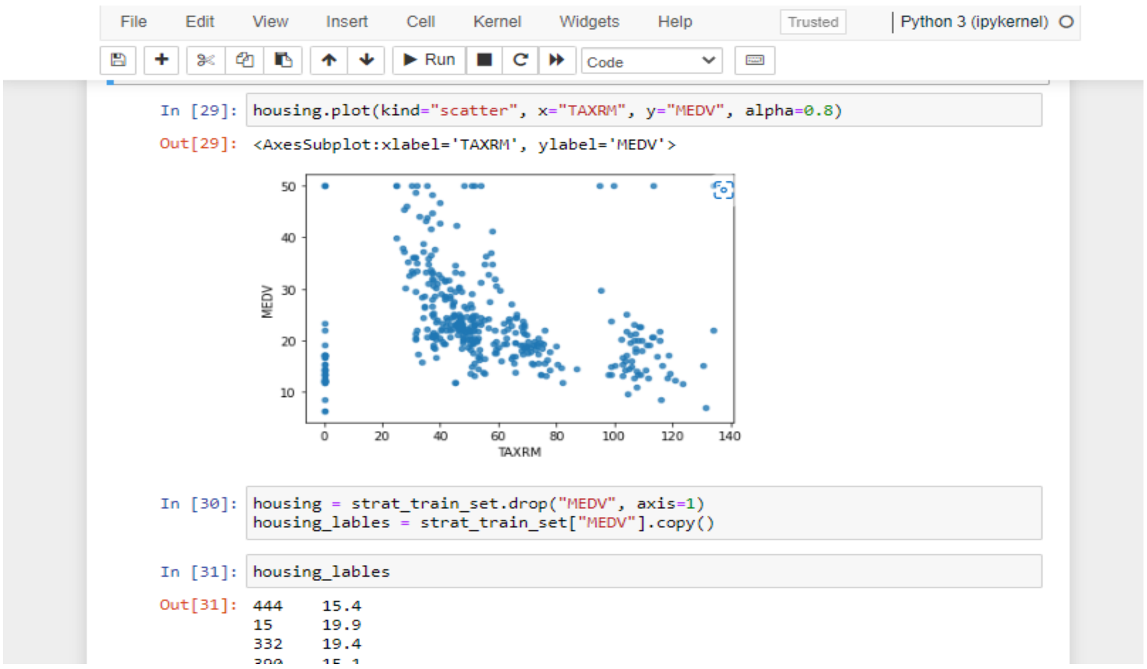
New attribute **TAXRM created .**

Correlation with MEDV:



```
In [28]: corr_matrix = housing.corr()
         corr_matrix['MEDV'].sort_values(ascending=False)

Out[28]: MEDV       1.000000
         B          0.339739
         ZN         0.227254
         CHAS      -0.027686
         AGE       -0.036385
         TAX       -0.062546
         TAXRM     -0.159228
         CRIM      -0.191999
         PTRATIO   -0.197084
         INDUS     -0.203179
         DIS       -0.210418
         RM        -0.237921
         RAD       -0.259971
         NOX       -0.279607
         LSTAT     -0.585628
         Name: MEDV, dtype: float64

In [29]: housing.plot(kind="scatter", x="TAXRM", y="MEDV", alpha=0.8)
```

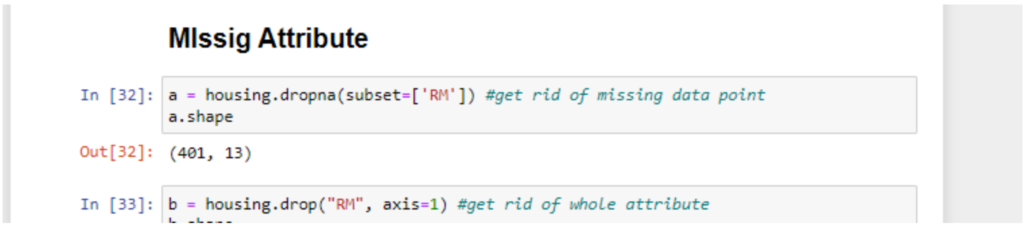
Histogram:



```
In [29]: housing.plot(kind="scatter", x="TAXRM", y="MEDV", alpha=0.8)
Out[29]: <AxesSubplot:xlabel='TAXRM', ylabel='MEDV'>

In [30]: housing = strat_train_set.drop("MEDV", axis=1)
         housing_lables = strat_train_set["MEDV"].copy()

In [31]: housing_lables
Out[31]: 444    15.4
         15     19.9
         332    19.4
```

Step 6:

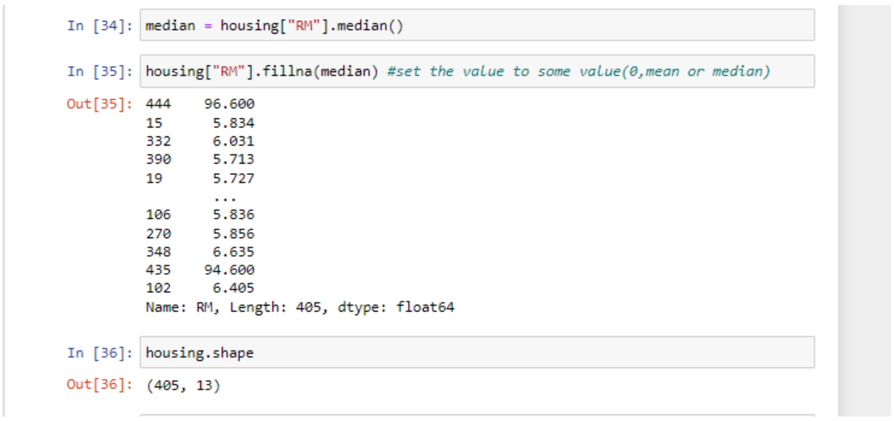
Here we deal with the missing attribute present in our dataset As we use RM to build our model so we find whether missing value present in that attribute or not.



MIssig Attribute

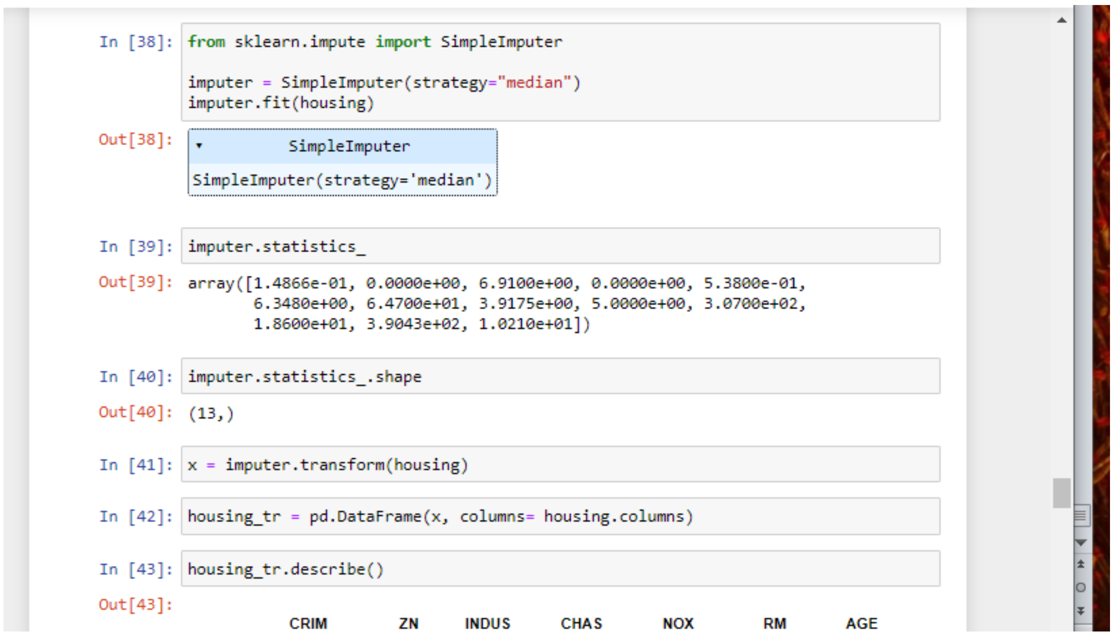
```
In [32]: a = housing.dropna(subset=['RM']) #get rid of missing data point
         a.shape

Out[32]: (401, 13)

In [33]: b = housing.drop("RM", axis=1) #get rid of whole attribute
```

Here we find RM have 401 value so 4 value is missing from RM attribute.
So to overcome this problem we use Impute mean/median for missing value.



```
In [34]: median = housing["RM"].median()

In [35]: housing["RM"].fillna(median) #set the value to some value(0,mean or median)

Out[35]: 444    96.600
         15      5.834
         332     6.031
         390     5.713
         19      5.727
                 ...
         106     5.836
         270     5.856
         348     6.635
         435    94.600
         102     6.405
         Name: RM, Length: 405, dtype: float64

In [36]: housing.shape

Out[36]: (405, 13)
```

Here we find the missing value is replace with median value of the attribute.



```
In [38]: from sklearn.impute import SimpleImputer

         imputer = SimpleImputer(strategy="median")
         imputer.fit(housing)

Out[38]: ▾        SimpleImputer
         SimpleImputer(strategy='median')

In [39]: imputer.statistics_

Out[39]: array([1.4866e-01, 0.0000e+00, 6.9100e+00, 0.0000e+00, 5.3800e-01,
                6.3480e+00, 6.4700e+01, 3.9175e+00, 5.0000e+00, 3.0700e+02,
                1.8600e+01, 3.9043e+02, 1.0210e+01])

In [40]: imputer.statistics_.shape

Out[40]: (13,)

In [41]: x = imputer.transform(housing)

In [42]: housing_tr = pd.DataFrame(x, columns= housing.columns)

In [43]: housing_tr.describe()

Out[43]:
               CRIM        ZN     INDUS      CHAS       NOX        RM       AGE
```

Step7:

Here we design skleran and creating pipeline so that if missing data come then it replace it with median value.

Scikit-learn Design

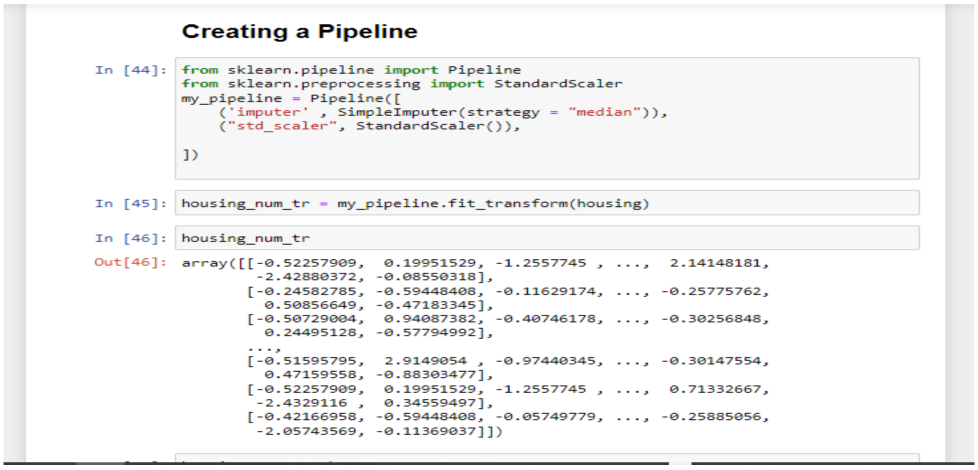
primary , three type of object

1. Estimators - it estimetes some parameter based on a dataset eg- imputer its has a fit method and transform method fit method - fits the dataset calculate internal parameters
2. transformers
3. predictor

Feature Scaling

primarly, two types of feature scaling methods:

1. Min-max scalinf(Normalization) (value-min)/(max-min)
2. Standardization (value - min)/std



Creating a Pipeline

```
In [44]: from sklearn.pipeline import Pipeline
         from sklearn.preprocessing import StandardScaler
         my_pipeline = Pipeline([
             ('imputer' , SimpleImputer(strategy = "median")),
             ("std_scaler", StandardScaler()),

         ])

In [45]: housing_num_tr = my_pipeline.fit_transform(housing)

In [46]: housing_num_tr

Out[46]: array([[-0.52257909,  0.19951529, -1.2557745 , ...,  2.14148181,
                 -2.42880372, -0.08550318],
                [-0.24582785, -0.59448408, -0.11629174, ..., -0.25775762,
                  0.50856649, -0.47183345],
                [-0.50729004,  0.94087382, -0.40746178, ..., -0.30256848,
                  0.24495128, -0.57794992],
                ...,
                [-0.51595795,  2.9149054 , -0.97440345, ..., -0.30147554,
                  0.47159558, -0.88303477],
                [-0.52257909,  0.19951529, -1.2557745 , ...,  0.71332667,
                 -2.4329116 ,  0.34559497],
                [-0.42166958, -0.59448408, -0.05749779, ..., -0.25885056,
                 -2.05743569, -0.11369037]])
```

Step 8:

Now we train our model with the training data set.
As earlier we discuss we use regression technique as we only predict the value so here we use Three regression technique

- Linear Regression.
- Decission tree Regreesion.
- Random forest Regression.

Linear Regression:

Linear regression is used to predict the relationship between two variables by applying a linear equation to observed data. There are two types of variable, one variable is called an independent variable, and the other is a dependent variable. Linear regression is commonly used for predictive analysis. The main idea of regression is to examine two things. First, does a set of predictor variables do a good job in predicting an outcome (dependent) variable? The second thing is which variables are significant predictors of the outcome variable?

$$\boxed{Y = a + bX}$$

Model output from linear Regression:
Mean: 6.680459559065733
Standrad_deviation: 1.6227697600705984

Decision tree Regression:

The decision trees is used to fit a sine curve with addition noisy observation. As a result, it learns local linear regressions approximating the sine curve.

We can see that if the maximum depth of the tree (controlled by the `max_depth` parameter) is set too high, the decision trees learn too fine details of the training data and learn from the noise, i.e. they overfit.

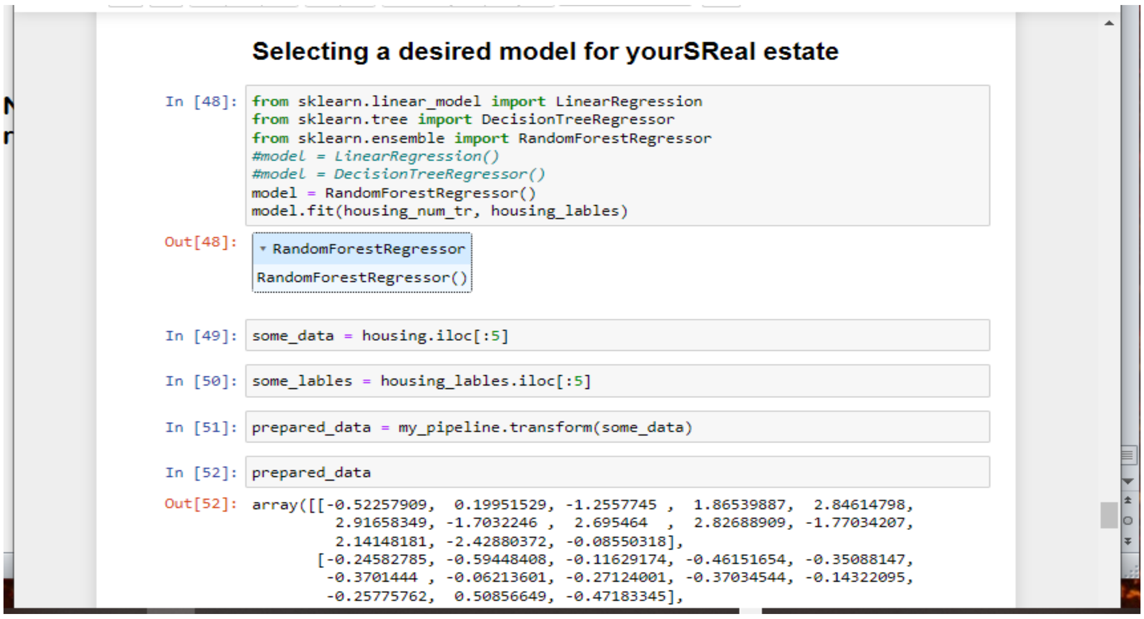
Model output from Decision tree Regression:
Mean: 5.40672778982749
Standrad_deviation: 1.327555325887593

Random Forest Regression:-

We will use the sklearn module for training our random forest regression model, specifically the RandomForestRegressor function. The RandomForestRegressor documentation shows many different parameters we can select for our model.

Model output from Random Forest Regression:
Mean: 4.0756729872979145
Standrad_deviation: 1.397065834059177

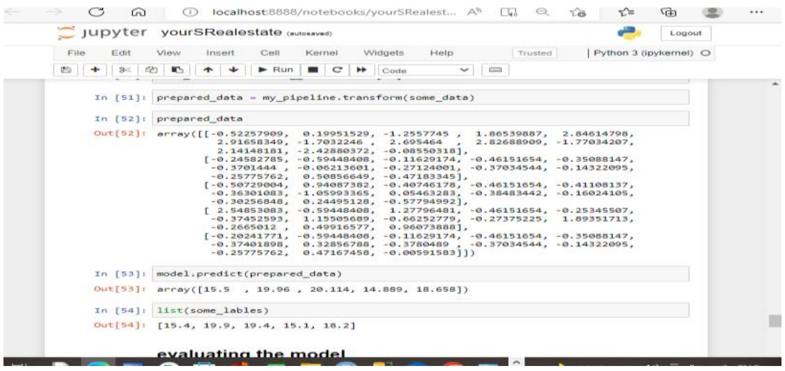
Now we find of the best result come from Random forest regression. So we use Random Forest to build our model.



Selecting a desired model for yourSReal estate

```
In [48]: from sklearn.linear_model import LinearRegression
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.ensemble import RandomForestRegressor
         #model = LinearRegression()
         #model = DecisionTreeRegressor()
         model = RandomForestRegressor()
         model.fit(housing_num_tr, housing_lables)

Out[48]: ▾ RandomForestRegressor
         RandomForestRegressor()

In [49]: some_data = housing.iloc[:5]

In [50]: some_lables = housing_lables.iloc[:5]

In [51]: prepared_data = my_pipeline.transform(some_data)

In [52]: prepared_data

Out[52]: array([[-0.52257909,  0.19951529, -1.2557745 ,  1.86539887,  2.84614798,
                  2.91658349, -1.7032246 ,  2.695464  ,  2.82688909, -1.77034207,
                  2.14148181, -2.42880372, -0.08550318],
                [-0.24582785, -0.59448408, -0.11629174, -0.46151654, -0.35088147,
                 -0.3701444 , -0.06213601, -0.27124001, -0.37034544, -0.14322095,
                 -0.25775762,  0.50856649, -0.47183345],
```

Prepared data for building model:

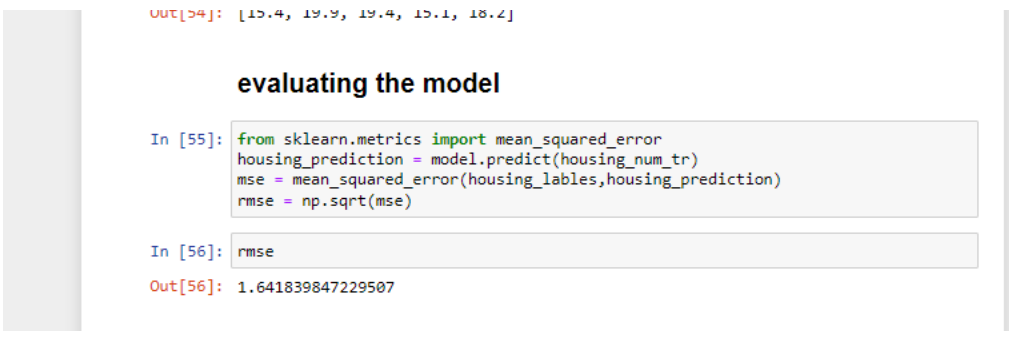


```
In [51]: prepared_data = my_pipeline.transform(some_data)

In [52]: prepared_data
Out[52]: array([[-0.52257909,  0.19951529, -1.2557745 ,  1.86539887,  2.84614798,
                  2.91658349, -1.7032246 ,  2.695464  ,  2.82688909, -1.77034207,
                  2.14148181, -2.42880372, -0.08550318],
                [-0.24582785, -0.59448408, -0.11629174, -0.46151654, -0.35088147,
                 -0.3701444 , -0.06213601, -0.27124001, -0.37034544, -0.14322095,
                 -0.25775762,  0.50856649, -0.47183345],
                [-0.50729004,  0.94087382, -0.40746178, -0.46151654, -0.41108137,
                 -0.36301083, -1.05993365,  0.05463283, -0.38483442, -0.16024105,
                 -0.30256848,  0.24495128, -0.57794992],
                [ 2.54853083, -0.59448408,  1.27796481, -0.46151654, -0.25345507,
                 -0.37452593,  1.15505689, -0.66252779, -0.27375225,  1.89351713,
                 -0.2665012 ,  0.49916577,  0.96073888],
                [-0.20241771, -0.59448408, -0.11629174, -0.46151654, -0.35088147,
                 -0.37401898,  0.32856788, -0.3780489 , -0.37034544, -0.14322095,
                 -0.25775762,  0.47167458, -0.00591583]])

In [53]: model.predict(prepared_data)
Out[53]: array([15.5  , 19.96 , 20.114, 14.889, 18.658])

In [54]: list(some_lables)
Out[54]: [15.4, 19.9, 19.4, 15.1, 18.2]
```

evaluating the model

Step 9:

In this step we evaluating our training data
With the model that with build .
We use both normal and cross validation and find exact error of the model



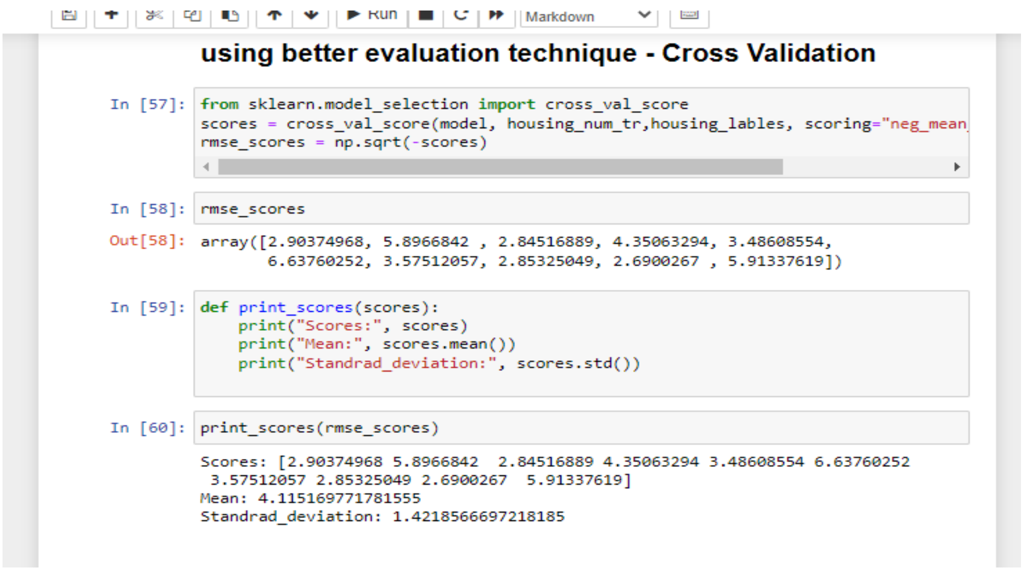
```
Out[54]: [15.4, 19.9, 19.4, 15.1, 18.2]
```

evaluating the model

```
In [55]: from sklearn.metrics import mean_squared_error
         housing_prediction = model.predict(housing_num_tr)
         mse = mean_squared_error(housing_lables,housing_prediction)
         rmse = np.sqrt(mse)

In [56]: rmse

Out[56]: 1.641839847229507
```

Here we find **root-mean-square** error from **Random Forest** gives ***1.64183.....*** appx ***2*** . Which is fine form 25% error list.

If we go on cross validation to get better result we find this solution.



using better evaluation technique - Cross Validation

```
In [57]: from sklearn.model_selection import cross_val_score
         scores = cross_val_score(model, housing_num_tr,housing_lables, scoring="neg_mean
         rmse_scores = np.sqrt(-scores)

In [58]: rmse_scores

Out[58]: array([2.90374968, 5.8966842 , 2.84516889, 4.35063294, 3.48608554,
                6.63760252, 3.57512057, 2.85325049, 2.6900267 , 5.91337619])

In [59]: def print_scores(scores):
             print("Scores:", scores)
             print("Mean:", scores.mean())
             print("Standrad_deviation:", scores.std())

In [60]: print_scores(rmse_scores)

         Scores: [2.90374968 5.8966842  2.84516889 4.35063294 3.48608554 6.63760252
          3.57512057 2.85325049 2.6900267  5.91337619]
         Mean: 4.115169771781555
         Standrad_deviation: 1.4218566697218185
```

Rmse score we find from this model is:

```
Scores: [2.90374968 5.8966842  2.84516889 4.35063294 3.48608554 6.63760252
 3.57512057 2.85325049 2.6900267  5.91337619]
Mean: 4.115169771781555
Standrad_deviation: 1.4218566697218185
```

What is mean?

➔ **Mean is average of a given set of data.**

$$\boxed{\mu = \frac{\sum_{i=1}^{n} x_i}{N}}$$

Where μ is mean and x_1, x_2, x_3...., x_i are elements. Also note that mean is sometimes denoted by.

What is Variance?

➔**Variance is the sum of squares of differences between all numbers and means.**

$$\sigma^2 = \sum_{i=1}^{N}(x_i - \mu) / N$$

What is Standard Deviation ?

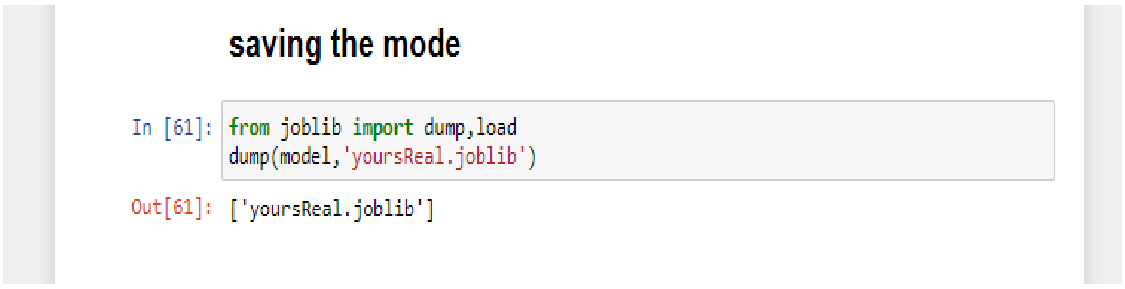
➔ Standard Deviation is square root of variance. It is a measure of the extent to which data varies from the mean.

So the coefficient of variation we determined is

$$\text{Coefficient of variation} = \frac{\text{Standard deviation}}{Mean} * 100$$

Step 10:

Here we make a joblib of our model so that we use our model externally.
I.e. we save our model.
So for that we have to use joblib and dump our model.

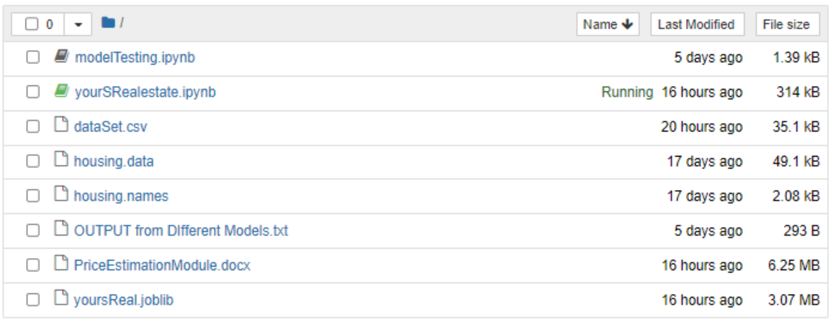


saving the mode

```
In [61]: from joblib import dump,load
         dump(model,'yoursReal.joblib')

Out[61]: ['yoursReal.joblib']
```

We dump our model with name **'yoursReal.joblib'** for External use

File directory:



Name		Last Modified	File size
modelTesting.ipynb		5 days ago	1.39 kB
yourSRealestate.ipynb	Running	16 hours ago	314 kB
dataSet.csv		20 hours ago	35.1 kB
housing.data		17 days ago	49.1 kB
housing.names		17 days ago	2.08 kB
OUTPUT from DIfferent Models.txt		5 days ago	293 B
PriceEstimationModule.docx		16 hours ago	6.25 MB
yoursReal.joblib		16 hours ago	3.07 MB

What is joblib?

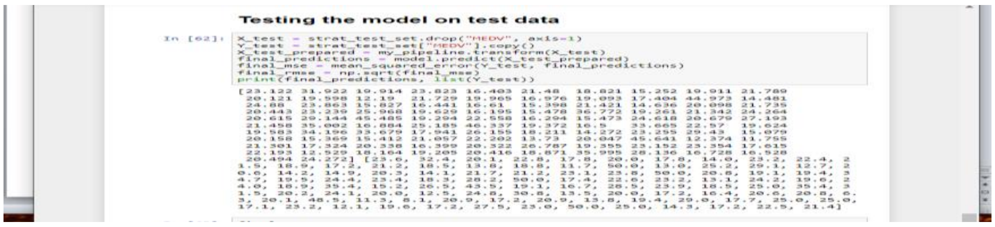
➔ joblib is a set of tools to provide **lightweight pipelining in Python**. In particular:

1. transparent disk-caching of functions and lazy re-evaluation (memoize pattern)
2. easy simple parallel computing

Joblib is optimized to be **fast** and **robust** on large data in particular and has specific optimizations for *numpy* arrays. It is **BSD-licensed**.

Step 11:

Now we test our model with the testing data set so that we predict the price

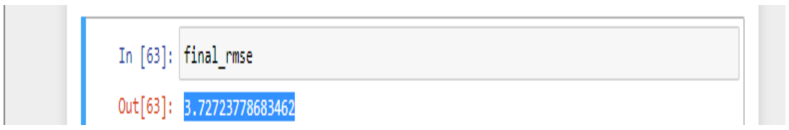


Testing the model on test data

```
In [62]: X_test = strat_test_set.drop("MEDV", axis=1)
         Y_test = strat_test_set["MEDV"].copy()
         X_test_prepared = my_pipeline.transform(X_test)
         final_predictions = model.predict(X_test_prepared)
         final_mse = mean_squared_error(Y_test, final_predictions)
         final_rmse = np.sqrt(final_mse)
         print(final_predictions, list(Y_test))

[23.122 31.922 19.914 23.823 16.403 21.48  18.821 15.252 19.911 21.789
 20.121 19.598 12.19  21.729 19.965 16.976 19.093 17.404 44.973 14.481
 24.88  23.863 15.827 16.441 16.61  15.398 21.421 14.636 20.098 21.735
 20.443 23.239 25.968 19.629 16.195 15.478 36.772 19.261 21.342 24.264
 20.615 29.144 45.485 19.294 22.558 16.294 15.473 24.618 20.679 27.193
 21.458 35.002 16.884 25.185 46.337 19.372 16.5   33.665 22.57  19.624
 19.583 34.196 33.679 17.941 26.155 18.211 14.272 23.255 29.43  15.079
 20.158 15.369 15.412 21.057 22.202 13.73  20.047 45.641 12.374 11.755
 21.301 17.324 20.338 16.399 20.322 26.787 19.355 23.152 23.354 17.615
 22.193 12.529 18.164 19.205 20.416 18.871 35.995 28.136 16.728 16.528
 20.494 24.272] [23.6, 32.4, 20.1, 22.8, 17.8, 20.0, 17.8, 14.0, 23.2, 22.4, 2
1.5, 18.9, 17.2, 21.2, 18.5, 13.8, 18.8, 11.7, 50.0, 13.0, 25.2, 29.1, 12.7, 2
0.6, 14.2, 14.9, 20.3, 14.1, 21.7, 21.2, 23.1, 23.8, 50.0, 20.8, 19.1, 19.4, 3
4.7, 19.5, 24.4, 23.4, 18.3, 28.2, 50.0, 17.4, 22.6, 23.2, 13.1, 24.2, 19.6, 2
4.0, 18.9, 35.4, 15.2, 26.5, 43.5, 19.1, 16.7, 28.5, 23.9, 18.5, 25.0, 35.4, 3
1.5, 20.2, 24.1, 20.0, 12.5, 24.8, 30.8, 13.5, 20.0, 17.2, 16.4, 20.6, 20.8, 6.
3, 20.1, 48.5, 11.3, 8.1, 20.9, 17.2, 20.9, 13.8, 19.4, 29.0, 17.7, 25.0, 25.0,
17.1, 23.2, 12.1, 19.6, 17.2, 27.5, 23.0, 50.0, 25.0, 14.3, 17.2, 22.5, 21.4]
```

Here final rmse error comes:

3.72723778683462



```
In [63]: final_rmse
Out[63]: 3.72723778683462
```

FINAL OUTPUT

Final output with different data:

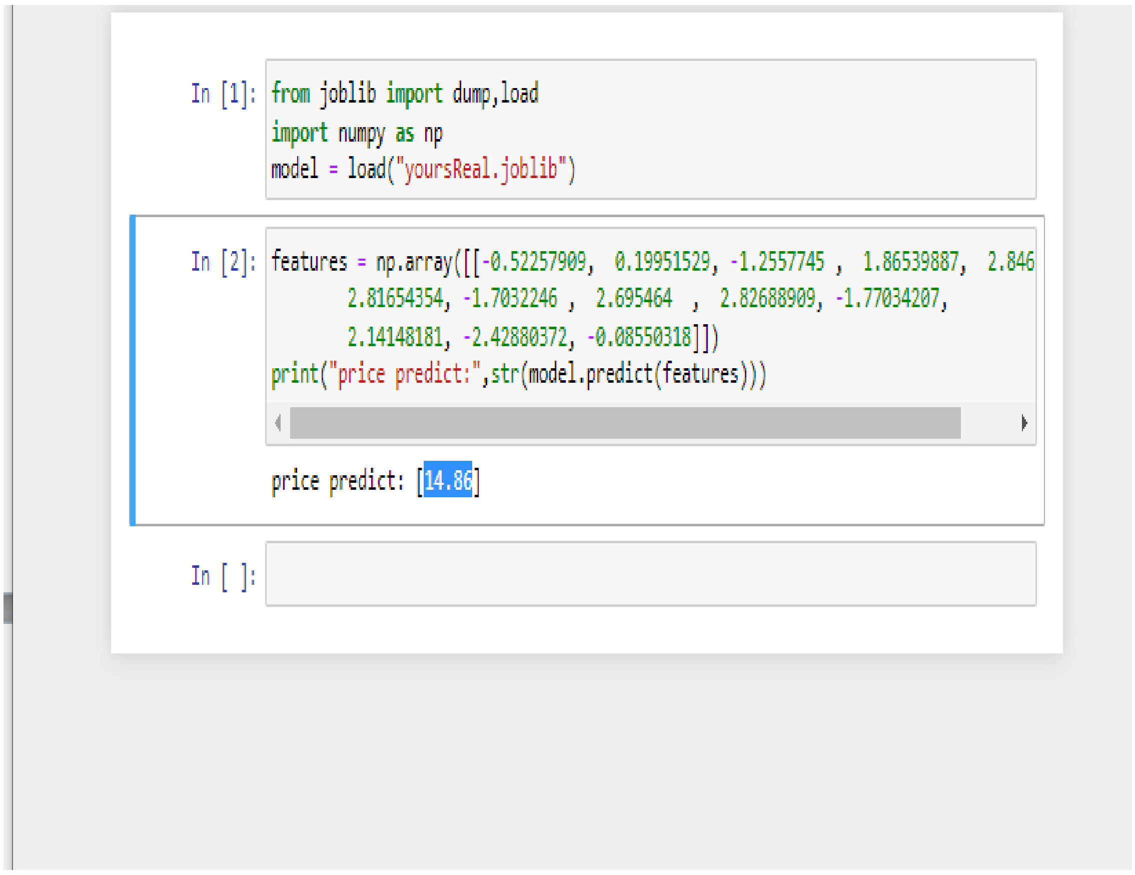
Array used:

-0.52257909, 0.19951529, -1.2557745 , 1.86539887, 2.84614798,
2.81654354, -1.7032246 , 2.695464 , 2.82688909, -1.77034207,
2.14148181, -2.42880372, -0.08550318

Features: **13**

Price predict: **14.86**

Model screen shot:



```
In [1]: from joblib import dump,load
        import numpy as np
        model = load("yoursReal.joblib")

In [2]: features = np.array([[-0.52257909,  0.19951529, -1.2557745 ,  1.86539887,  2.846
                  2.81654354, -1.7032246 ,  2.695464  ,  2.82688909, -1.77034207,
                  2.14148181, -2.42880372, -0.08550318]])
        print("price predict:",str(model.predict(features)))

        price predict: [14.86]

In [ ]:
```

PROS AND CONS

Pros:

- This model is build under **Random forest regression** so the at least lowest amount of error comes out from the model with correct price.
- All the assumption of this model is correct and
  Taken in the presence of expert. So there may be no problem of wrong model
- As all the assumption is correct so this model doesn't undergoes into classification method.
- This model is train in such a way that if new feature come with a Varity of other dataset then we can easily train the model again.

Cons

- Though the model train with correct dataset
  But there is a problem in missing value where we make impurity and median value in the missing attribute value place so there might be a problem in given wrong error(As we diagnosed before that the model give **rmse** error be 0.0 but actually it gives error *2*)
- Model is under batch learning so if new feature comes than we have to train our model again
- We try our model with three regression technique. But if we train our model with other than may be we find least error and may be our model behave better manner.

CONCLUSION & FUTURE SCOPE

Predicting housing prices from online advertisements is a task which requires insight into the data combined with powerful ML algorithms. In this work, we applied two different methods for this task, and combined them into a final prediction. We provided a strong baseline to overcome, as our final ensemble hit a score *1.64183.....* of RMSE. The two architectures (enriched RF and KISS) separately also presented good results and can be used as baselines for data enrichment, stacking models, or configuring RNNs. The Enriched RF works well with numeric features, as it can derive rules not only depending on the value of an attribute but also on its presence or absence. However, it cannot handle raw image or text data. KISS, on the other hand, can represent all kinds of data through the embedding layers. But it did not handle numeric attributes as well as the random forest, and had to rely on more data types to edge it out. Combining the two methods yielded the best result, combining the different strengths of each approach. As future work, feature selection algorithms can also be employed to leverage the training speed for the models. Another technique which we aim to apply on this dataset is weak supervised learning with pseudo-labeling to increase the number of training data instances for deep learning.

REFERENCE& BIBLIOGRAPHY

Alpaydin, E. (2009). Introduction to machine learning. MIT press. Associac¸ao Brasileira de Incorporadoras Imobili ˜ arias – ABRAINC (2017). An ´ alise ´ das necessidades habitacionais e suas tendencias para os pr ˆ oximos dez anos. ´ https://www.abrainc.org.br/wp-content/uploads/2018/10/ ANEHAB-Estudo-completo.pdf. Breiman, L. (2001). Random forests. Machine Learning, 45(1):5–32. De Souza, F. A. (1999). Land tenure security and housing improvements in recife, brazil. Habitat International, 23(1):19–33. Fan, C., Cui, Z., and Zhong, X. (2018). House prices prediction with machine learning algorithms. In Proceedings of the 2018 10th International Conference on Machine Learning and Computing, pages 6–10. ACM. Fik, T. J., Ling, D. C., and Mulligan, G. F. (2003). Modeling spatial variation in housing prices: a variable interaction approach. Real Estate Economics, 31(4):623–646. Goodman, A. C. and Thibodeau, T. G. (2003). Housing market segmentation and hedonic prediction accuracy. Journal of Housing Economics, 12(3):181–201. Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8):1735–1780. Huang, G., Liu, Z., v. d. Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2261–2269. Moreira de Aguiar, M., Simoes, R., and Braz Golgher, A. (2014). Housing market analysis ˜ using a hierarchical–spatial approach: the case of belo horizonte, minas gerais, brazil. Regional Studies, Regional Science, 1(1):116–137. Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. Javapoint lecture on machine learning, uci ml repository, code with harry note, medium.com/codex/house-price-prediction-with-machine-learning-in-python note. Anil Bikash Chowdhury Ml lecture, Tapas Kumar Mitra python lecture and note.