A

PROJECT REPORT ON

"Real Estate Price Prediction Using Machine Learning"

SUBMITTED BY
Singh Abhishek Amar
Roll No – MC22064
Seat No - 2783

UNDER THE GUIDANCE OF
"Prof. Yugandhara Patil"

IN PARTIAL FULFILMENT OF
Award of the Degree of
MASTER OF COMPUTER APPLICATION
(Sem II)

SUBMITTED TO

SAVITRIBAI PHULE PUNE UNIVERSITY

THROUGH

YASHASWI EDUCATION SOCIETY'S
INTERNATIONAL INSTITUTE OF MANAGEMENT SCIENCE
CHINCHWAD, PUNE
ACADEMIC YEAR 2022-2023

DECLARATION

I, **Singh Abhishek Amar**, student of International Institute of Management Science, Chinchwad, Pune, hereby declare that this Mini Project report entitled "**Real Estate Price Prediction Using Machine Learning**".
Is a bonafide record of work done by me for the partial fulfilment of the requirement for the degree of **Master of Computer Application** (M.C.A) through Savitribai Phule Pune University.

I, hereby, declare that I have adequately referenced the original sources and this is my original work. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute andcan also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Singh Abhishek Amar
IIMS, Chinchwad, Pune.

ACKNOWLEDGEMENT

I am deeply indebted towards my project guide **"Prof. Yugandhara Patil"** who gave me the opportunity and was instrumental in providing me all the knowledge andinsight to do the research. It is their inspiration that has kept me motivated all along my project and the discipline and integrity they had expected from a mini project that made me to learn the real live projects.

I would like to express my earnest gratitude and thanks to **Dr. Shivaji Mundhe** (Director ofIIMS) and my project guide **Prof. Yugandhara Patil** for providing me all the knowledge andskills, resources, technical support, guidance as required to achieve this Endeavour.

I thank my all-faculty members and friends for their support and blessings. The report is the result of contribution of numerous people to mention individually.

I also thank all respondent who have given their value time, views and authentic information for this mini project. I thank each and everybody who has contributed directlyor indirectly to the successful completion of this project.

(Signature)
Singh Abhishek Amar
IIMS, Chinchwad, Pune.

Yashaswi Education Society's
Reg No. Maha. : 417/2007/Pune
INTERNATIONAL INSTITUTE OF MANAGEMENT SCIENCE
An ISO 9001 Certified Institute
(Approved by AICTE Ministry of HRD Govt. of India, Recognised by Govt. of Maharashtra and Affiliated to Savitribai Phule Pune University)
Campus. : IIMS Bldg, S. No. 169/1/A, Opp. Elpro International, Chinchwad, Pune - 411033. Ph.: (020) 27353730/32/33/34, Fax : (020) 27354731
Website. : www.iims.ac.in E-mail : info@iims.ac.in

CERTIFICATE

This is to Certify that ***Abhishek Amar Singh*** *is a Bonafide student of* ***International Institute of Management Science, Chinchwad, Pune****, worked on* ***Online Blogging System*** *and has successfully completed project work in partial fulfillment for award of degree* ***Master of Computer Application (MCA) Sem III*** *of Savitribai PhulePune University.*

This report is the record of Student's own efforts under our supervision and guidelines.

Date: 21.12.2023

Internal Guide *HOD-MCA* *Dr. Shivaji MundheDirector, IIMS, Chinchwad*

Internal Examiner *External Examiner*

PLAGIARISM REPORT

Plagiarism Checker X - Report

Originality Assessment

2%

Overall Similarity

Date: Dec 19, 2023
Matches: 100 / 4452 words
Sources: 3

Remarks: Low similarity detected, check with your supervisor if changes are required.

Verify Report:
Scan this QR Code

TABLE OF CONTENTS

1. INTRODUCTION

- Welcome to our **Real Estate Price Prediction Using Machine Learning**! We are excited to bring you a seamless and convenient way to Price Prediction Using ML. The purpose of **Real Estate Price Prediction Using Machine Learning** is to automate the existing manual system by the help of computerized and full-fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same.
- This Project Provides us an overview on how to Real Estate Price Prediction Using Machine Learning various ML models with the help of different python libraries.

- This proposed model considers as the most accurate model used for calculating the house price and provides a most accurate prediction. This project consists of what and how the Real Estate Price Prediction model works with the assistance of machine learning technique using scikit-learn and which datasets we will be using in our proposed model.
- By using real world data entities, we are going to predict the price of the house in that area. For better results we require data pre-processing units to improve the efficiency of the model. for this project we are using supervised learning, which is a part of machine learning. We have to go through different attributes of the dataset.

- By using real world data entities, we are going to predict the price of the house in that area. For better results we require data pre-processing units to improve the efficiency of the model. for this project we are using supervised learning, which is a part of machine learning. We have to go through different attributes of the dataset.

- Keywords: - Python, Machine learning, scikit-learn, python libraries, data pre-processing, Linear Regression, Decision Tree, RandomForestRegressor algorithm, Supervised learning.

2. SYSTEM DETAIL

2.1 Proposed System

One of the basic requirements of livelihood in the recent world is to buy a house of your own. The price of the house may depend on various factors. Real estate agents and many who are involved in selling the house want a price tag on the house which would be the real worth of buying the Real Estate Or House.

Thousands of houses are sold every day. There are some questions every buyer asks himself like: What is the actual price that this house deserves? Am I paying a fair price? In this paper, a machine learning model is proposed to predict a house price based on data related to the house and This will facilitate the reproducibility of our work. In this study, Python programming language with a number of Python packages will be used.

2.2Tools and Technologies used

Tools

➢ Jupyter Notebook	: An interactive python environment for data analysis and scientific computing.

Technologies

➢ Numpy	: Library for numerical computing.
➢ Matplotlib	: Library which provides collection of functions for creating variety of static, animated and interactive visualizations in python.
➢ Pandas	: A data manipulation library for data analysis
➢ Sklearn	: A machine learning library for python

2.3 System requirements

Hardware Requirements (Optimum)

- Processor : intel i3 and above
- RAM : 4 GB and above
- Operating System : Windows 7 and above
- Hard Disk : 512 GB and above

Software requirements

- Tools : MS Excel, Python.
- Platform : Vs Code
- IDE : Jupyter

3. SOFTWARE REQUIREMENT SPECIFICATION

3.1 Functional requirements

Functional Requirement defines a function of a software system and how the system must behave when presented with specific inputs or conditions. These may include calculations, data manipulation and processing and other specific functionality. Following are the functional requirements on the system: The whole process can be handled at minimal human interaction with android and web both. The application automatically receives the captured data from server.

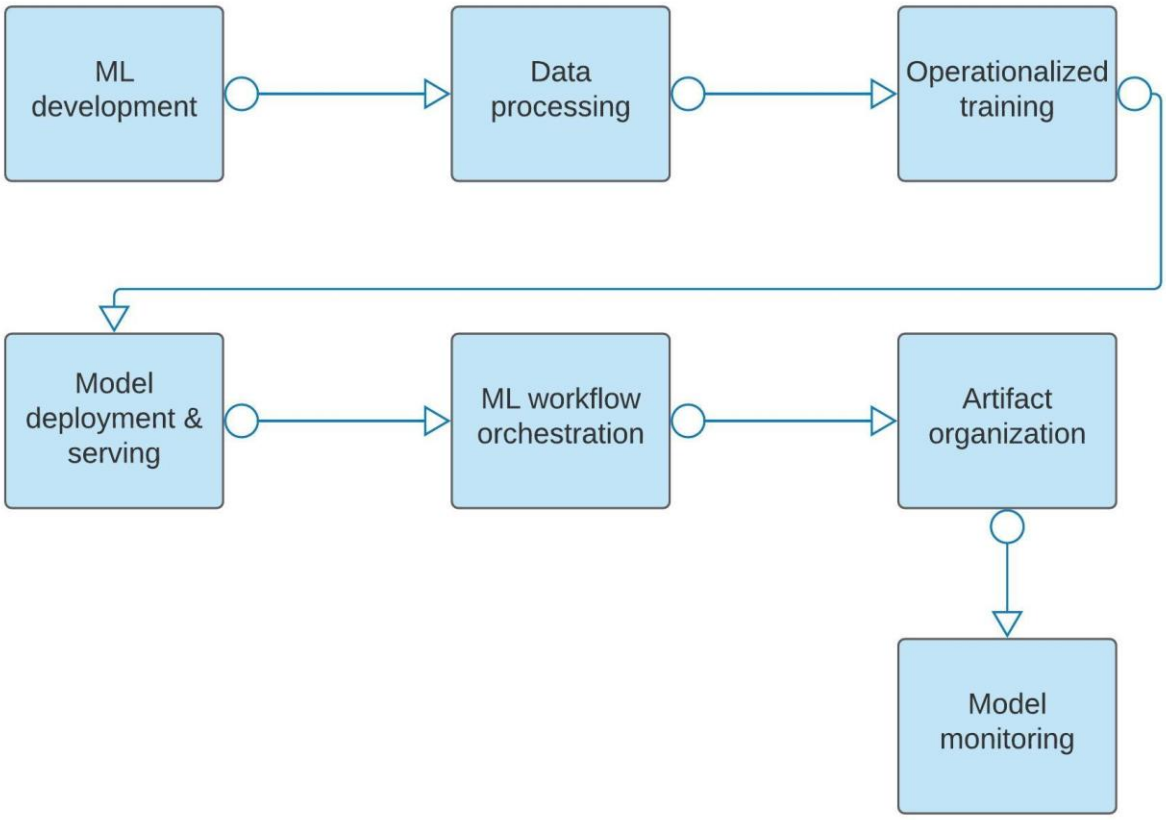
3.2 Non-Functional Requirements

Non-functional requirements are the requirements which are not directly concerned with the specific function delivered by the system. They specify the criteria that can be used to judge the operation of a system rather than specific behaviors. They may relate to emergent system properties such as reliability, response time and store occupancy. Non-functional requirements arise through the user needs, because of budget constraints, organizational policies, the need for interoperability with other software and hardware systems or because of external factors such as

- ✓ Performance Requirements
- ✓ Design Requirements
- ✓ Security Constraints
- ✓ Basic Operational Requirements

4. SYSTEM DESIGN

4.1 System Architecture



5. DETAILED DESIGN

Design is a meaningful engineering representation of something that is to be built. It is the most crucial phase in the developments of a system. Software design is a process through which the requirements are translated into a representation of software. Design is a place where design is fostered in software Engineering. Based on the user requirements and the detailed analysis of the existing system, the new system must be designed.

This is the phase of system designing. Design is the perfect way to accurately translate a customer's requirement in the finished software product. Design creates a representation or model, provides details about software data structure, architecture, interfaces and components that are necessary to implement

a system. The logical system design arrived at as a result of systems analysis is converted into physical system design.

5.1 System Development Methodology

System development method is a process through which a product will get completed or a product gets rid from any problem. Software development process is described as a number of phases, procedures and steps that gives the complete software. It follows series of steps which is used for product progress. The development method followed in this project is waterfall model.

The waterfall model is a successive programming improvement process, in which advance is seen as streaming relentlessly downwards (like a waterfall) through the periods of Requirement start, Analysis, Design, Implementation, Testing and upkeep.

5.1.1 Prerequisite Analysis

This stage is worried about gathering of necessity of the framework. This procedure includes producing record and necessity survey. Framework Design: Keeping the prerequisites at the top of the priority list the framework details are made an interpretation of into a product representation. In this stage the fashioner underlines on calculation, information structure, programming design and so on.

5.1.2 Coding

In this stage developer begins his coding with a specific end goal to give a full portray of item. At the end of the day framework particulars are just changed over into machine coherent register code.

5.1.3 Usage

The execution stage includes the genuine coding or programming of the product. The yield of this stage is regularly the library, executables, client manuals and extra programming documentation.

5.1.4 Testing

In this stage all projects (models) are coordinated and tried to guarantee that the complete framework meets the product prerequisites. The testing is worried with check and approval.

5.1.5 Support

The upkeep stage is the longest stage in which the product is upgraded to satisfy the changing client need, adjust to suit change in the outside environment, right mistakes and oversights beforehand undetected in the testing stage, improve the proficiency of the product.

5.2 Approach

The below figure explains the approach we have taken into building the predictive model using machine learning algorithms.

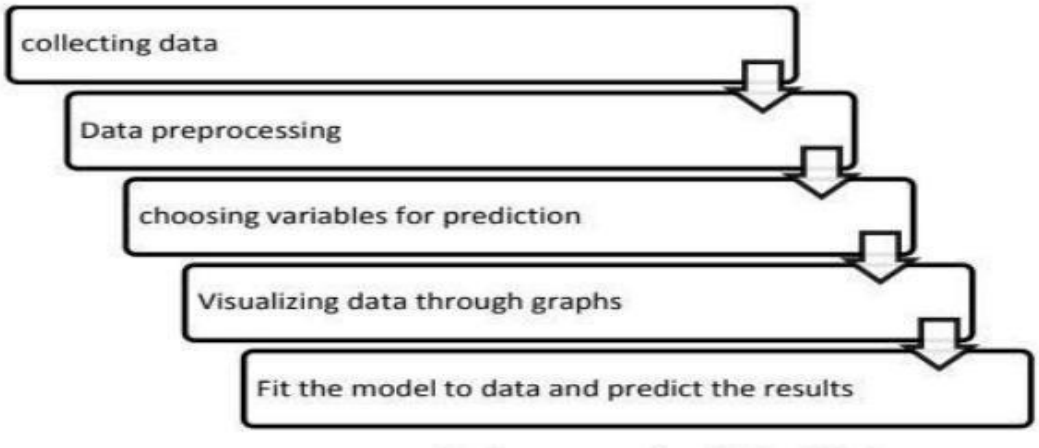


Fig 1: Process of Predicting Real Estate Price

5.2.1 Data Collection

Data collection is the process of gathering and measuring information from countless different sources. In order to use the data, we collect to develop practical machine learning solutions. Collecting data allows you to capture a record of past events so that we can use data analysis to find Predict Cost. From those Predict Cost, you build predictive models using machine learning algorithms that look for trends and predict future changes.

The Real Estate Price Prediction Using Machine Learning is the principal basis of data for this project. The data was web scrapped from the website and kept in the appropriate format using a python library called beautiful soup.

5.2.2 Data Preprocessing

5.2.2.1 Data cleaning

There are some null values in the dataset in the columns. Due to the presence of these null values, the classification cannot be done accurately. So, we tried to replace the null values in different columns with dummy values.

5.2.2.2 Choosing Required Attributes

This step is the main part where we can eliminate some columns of the dataset that are not useful for the estimation of match winning team. This is estimated using feature importance. The considered attributes have the following feature importance.

5.3 Product Requirements Platform independency

A progressive web app will be developed and deployed so that users with a smartphone or a computer can access the voting site to cast their vote. Ease of use: The progressive web app provides an interface which is easy to use and eliminates the need for the voter to go to a voting booth. Modularity: The complete product is broken up into modules and well-defined interfaces are developed to explore the benefit of flexibility of the product. Robustness: This software is being developed in such a way that the overall performance is optimized, and the user can expect the results within a limited time with utmost relevancy and correctness**.**

6. IMPLEMENTATION

6.1 Programming and Naming Conventions

6.1.1 Program Name

- ✓ Comparison Analysis using Data Preprocessing and Linear Regression, Decision Tree, RandomForestRegressor
- ✓ Prediction analysis using Linear Regression.

6.1.2 Module Names

- ✓ import pandas as pd
- ✓ import numpy as np
- ✓ import seaborn as sns
- ✓ import matplotlib.pyplot as plt
- ✓ from sklearn.model_selection import train_test_split
- ✓ from sklearn.model_selection import StratifiedShuffleSplit

- ✓ from pandas.plotting import scatter_matrix
- ✓ from sklearn.impute import SimpleImputer
- ✓ from sklearn.pipeline import Pipeline
- ✓ from sklearn.preprocessing import StandardScaler
- ✓ from sklearn.metrics import mean_squared_error
- ✓ from sklearn.model_selection import cross_val_score
- ✓ from joblib import dump, load
- ✓ from sklearn.linear_model import LinearRegression
- ✓ from sklearn.tree import DecisionTreeRegressor
- ✓ from sklearn.ensemble import RandomForestRegressorfrom

6.1.3 Function Names

- ✓ pd.read_csv()
- ✓ np.array()
- ✓ housing.head ()
- ✓ housing.info()
- ✓ housing.describe()
- ✓ housing.hist()
- ✓ sns.pairplot()
- ✓ np.random.seed()
- ✓ np.random.permutation()
- ✓ housing.corr()
- ✓ housing.plot()
- ✓ housing.dropna()
- ✓ housing.drop()
- ✓ imputer.fit()
- ✓ imputer.transform()

- ✓ pd.DataFrame()
- ✓ model.fit()
- ✓ model.predict()
- ✓ np.sqrt()
- ✓ np.array()

6.1.4 Variable Mames

- ✓ housing
- ✓ object_cols
- ✓ num_cols
- ✓ fl_cols
- ✓ def
- ✓ shuffled
- ✓ train_set
- ✓ test_set
- ✓ split
- ✓ corr_matrix
- ✓ housing_labels
- ✓ X
- ✓ housing_tr
- ✓ my_pipeline
- ✓ housing_num_tr
- ✓ model
- ✓ some_data
- ✓ some_label
- ✓ prepared_data
- ✓ housing_predictions
- ✓ final_predictions

- ✓ X_test
- ✓ Y_test
- ✓ X_test_prepared
- ✓ final_mse
- ✓ final_rmse
- ✓ features

6.1.5 Comments

- ✓ #Importing Lab.
- ✓ #Reading Data.
- ✓ # We will create some simple plot for visualizing the data.
- ✓ #Data Preprocessing.
- ✓ #Splitting Dataset into Training and Testing.
- ✓ #For Correlations.
- ✓ #Plotting Size Attr.
- ✓ # To take care of missing attributes, you have three options.
- ✓ #For Imputer.
- ✓ #For Creating Pipeline from Sklearn.
- ✓ #Model and Accuracy.
    - Linear Regressor
    - DecisionTree Regressor
    - RandomForest Regresso
- ✓ #Evaluating the Model.
- ✓ #Cross Validaton Technique
- ✓ #Testing The Model * Test Data
- ✓ #Used Model ---

6.2 Snippet Code

Real Estate - Price Prediction Using Machine Learning

```
#Importing Lab
import pandas as pd
import numpy  as np
import seaborn as sns
import matplotlib.pyplot as plt

#Reading Data
housing = pd.read_csv("data.csv")

housing.head(9)
housing.info()
housing['CHAS'].value_counts()
housing.describe()
housing.isna().sum()

#For Plotting Histogram
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))

# We will create some simple plot for visualizing the data.
sns.pairplot(housing)

#Data Preprocessing
#we categorize the features depending on their datatype (int, float, object) and then
calculate the number of them!!
obj = (housing.dtypes == 'object')
```

```
object_cols = list(obj[obj].index)
print("Categorical variables:",len(object_cols))

int_ = (housing.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))

fl = (housing.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))
```

Splitting Dataset into Training and Testing

```
#For T-T
import numpy as np
def split_train_test(data, test_ratio):
    np.random.seed(42)
    shuffled = np.random.permutation(len(data))
    print(shuffled)
    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]
print(f"Rows in train set: {len(train_set)}\nRows in test set: {len(test_set)}\n")

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)}\nRows in test set: {len(test_set)}\n")
```

```
from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(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]


strat_test_set['CHAS'].value_counts()
strat_train_set['CHAS'].value_counts()
95/7
376/28
housing = strat_train_set.copy()
```

Looking For Correlations

```
#For Correlations
corr_matrix = housing.corr()
corr_matrix['MEDV'].sort_values(ascending=False)

#Plotting Size Attr
from pandas.plotting import scatter_matrix
attributes = ["MEDV", "RM", "ZN", "LSTAT"]
scatter_matrix(housing[attributes], figsize = (12,8))

housing.plot(kind="scatter", x="RM", y="MEDV", alpha=0.8)
```

Trying out Attribute Combintions

```
housing["TAXRM"] = housing['TAX']/housing['RM']
housing.head()

corr_matrix = housing.corr()
corr_matrix['MEDV'].sort_values(ascending=False)

housing.plot(kind="scatter", x="TAXRM", y="MEDV", alpha=0.8)
```

```
housing = strat_train_set.drop("MEDV", axis=1)
housing_labels = strat_train_set["MEDV"].copy()
```

Missing Attributes

```
# To take care of missing attributes, you have three options:
#     1. Get rid of the missing data points
#     2. Get rid of the whole attribute
#     3. Set the value to some value (0, mean or median)

a = housing.dropna(subset=["RM"]) #Option 1
a.shape
# Note that the original housing dataframe will remain unchanged

housing.drop("RM", axis=1).shape # Option 2
# Note that there is no RM column and also note that the original housing dataframe will
remain unchanged.

median = housing["RM"].median() # Compute median for Option 3
housing["RM"].fillna(median) # Option 3
# Note that the original housing dataframe will remain unchanged
housing.shape
housing.describe() # before we started filling missing attributes
housing.head()

#For Imputer
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
imputer.fit(housing)
```

SimpleImputer

```
SimpleImputer(strategy='median')
```

```
imputer.statistics_
imputer.statistics_.shape
X = imputer.transform(housing)
housing_tr = pd.DataFrame(X, columns=housing.columns)
housing_tr.describe()
```

Scikit-Learn Design

Primarily, three types of objects

1. Estimators - It estimates some parameter based on a dataset. Eg. imputer. It has a fit method and transform method. Fit method - Fits the dataset and calculates internal parameters
2. Transformers - transform method takes input and returns output based on the learnings from fit(). It also has a convenience function called fit_transform() which fits and then transforms.
3. Predictors - LinearRegression model is an example of predictor. fit() and predict() are two common functions. It also gives score() function which will evaluate the predictions.

Feature Scaling

Primarily, two types of feature scaling methods:

1. Min-max scaling (Normalization) (value - min)/(max - min) Sklearn provides a class called MinMaxScaler for this
2. Standardization (value - mean)/std Sklearn provides a class called StandardScaler for this

Creating a Pipeline

```
#For Creating Pipeline from Sklearn
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    #     ..... add as many as you want in your pipeline
    ('std_scaler', StandardScaler()),
])

housing_num_tr = my_pipeline.fit_transform(housing)

housing_num_tr.shape
```

Selecting a desired model for Real Estate Price Prediction Using Machine Learning

#Model and Accuracy As we have to train the model to determine the continuous values, so we will be using these regression models.

Linear Regressor - Linear Regression predicts the final output-dependent value based on the given independent features. Like, here we have to predict SalePrice depending on features like MEDV, RM, TAX Etc

1. DecisionTree Regressor - 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.

2. RandomForest Regressor - Random Forest is an ensemble technique that uses multiple of decision trees and can be used for both regression and classification tasks.

#Regression is a method for understanding the relationship between independent variables or features and a dependent variable or outcome. Outcomes can then be predicted once the relationship between independent and dependent variables has been estimated.

```
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_labels)
```

```
RandomForestRegressor
RandomForestRegressor()
```

```
some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
prepared_data = my_pipeline.transform(some_data)
model.predict(prepared_data)
list(some_labels)
```

Evaluating the Model

```
#Evaluating the Model
from sklearn.metrics import mean_squared_error
housing_predictions = model.predict(housing_num_tr)
mse = mean_squared_error(housing_labels, housing_predictions)
rmse = np.sqrt(mse)

rmse
```

Using better Evaluation Technique - Cross Validation

```
#Cross Validaton Technique
# 1 2 3 4 5 6 7 8 9 10
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, housing_num_tr, housing_labels,
scoring="neg_mean_squared_error", cv=10)
rmse_scores = np.sqrt(-scores)
```

```
rmse_scores

def print_scores(scores):
    print("Scores:", scores)
    print("Mean: ", scores.mean())
    print("Standard deviation: ", scores.std())
    print_scores(rmse_scores)
```

Saving the Model

```
from joblib import dump, load
dump(model, 'Real Estate.joblib')
```

Testing the Model on test data

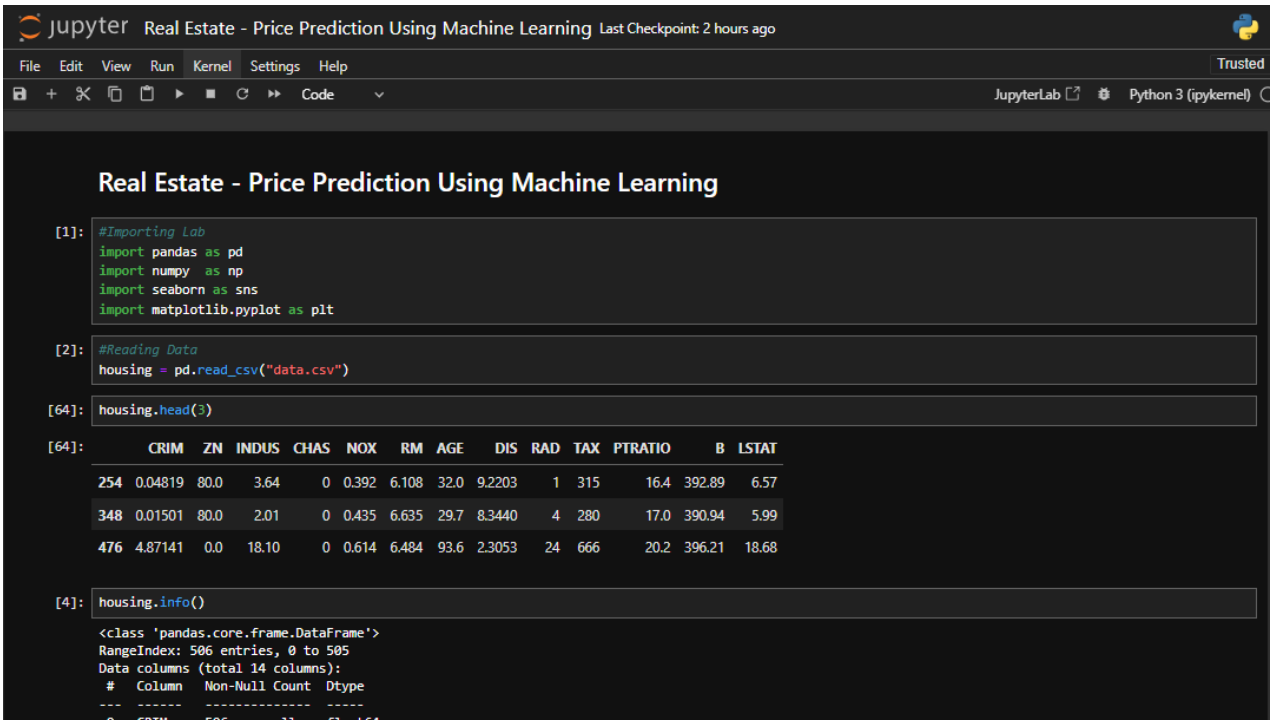
```
#Testing The Model * Test Data
X_test = strat_test_set.drop("MEDV", axis=1)
Y_test = strat_test_set["MEDV"].copy()
X_test_prepared = my_pipeline.transform(X_test)
final_predictions = model.predict(X_test_prepared)
final_mse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_mse)
# print(final_predictions, list(Y_test))

final_rmse

prepared_data[0]
```

Using the Model

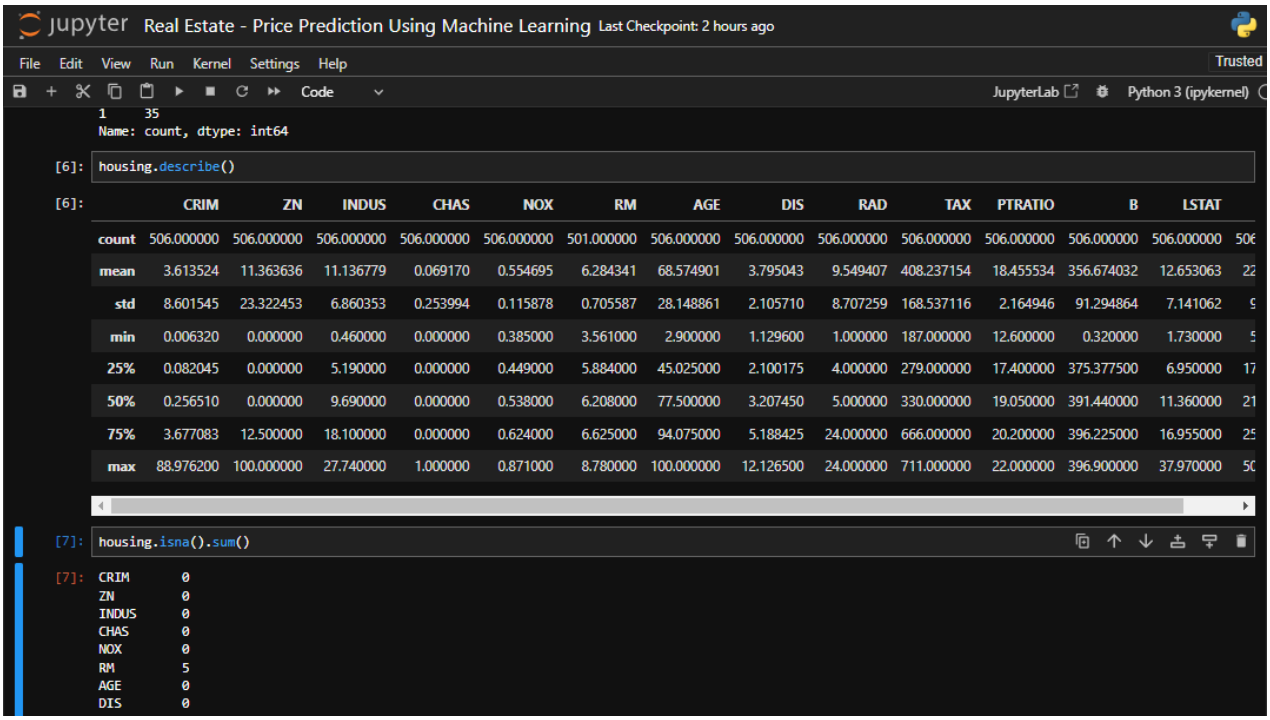
```
#Used Model ---
from joblib import dump
import numpy as np
model = load('Real Estate.joblib')
features = np.array([[-0.43942006,  11.12628155, -1.12165014, -0.27288841, -
1.42262747,
       -0.23979304, -1.31238772,  2.61111401, -1.0016859 , -0.5778192 ,
       -0.97491834,  0.41164221, -0.86091034],
      [-0.44352175,  3.12628155, -1.35893781, -0.27288841, -1.0542567 ,
        0.5009123 , -1.3938808 ,  2.19312325, -0.65766683, -0.78557904,
       -0.69277865,  0.39131918, -0.94116739],
      [ 0.15682292, -0.4898311 ,  0.98336806, -0.27288841,  0.47919371,
        0.28867984,  0.87020968, -0.68730678,  1.63579367,  1.50571521,
        0.81196637,  0.44624347,  0.81480158],
      [-0.42292925, -0.4898311 , -0.57719868, -0.27288841, -0.5573845 ,
        0.13688444, -0.52225911,  0.37882487, -0.5429938 , -0.74402708,
        0.52982668,  0.45343469, -0.81939807],
      [-5.40786253, -2.4898311 , -0.57719868, -0.27288841, -0.5573845 ,
        0.04693161, -1.42222622,  5.79643404, -0.5429938 , -0.74402708,
        0.52982668,  0.45343469, -0.91902752]])

model.predict(features)
```

6.2 Screenshots

#Reading data from data.csv



#Description Of Real Estate Attributes



#For Plotting Histogram For Model

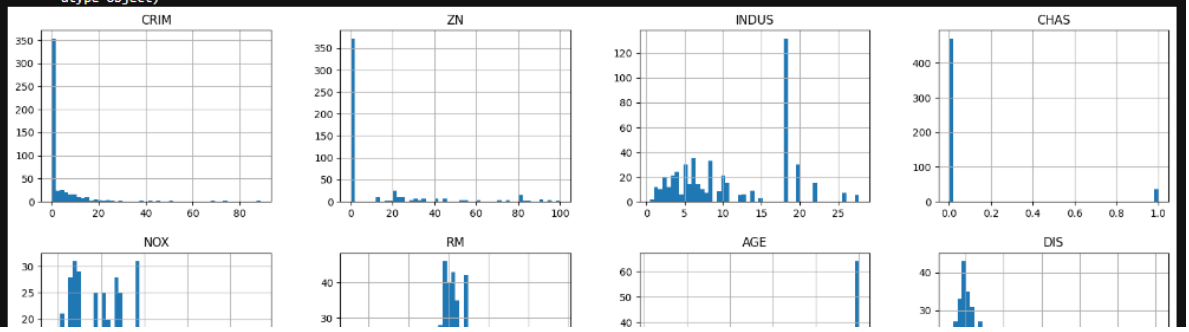
jupyter Real Estate - Price Prediction Using Machine Learning Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

```
[9]: #For Plotting Histogram
     import matplotlib.pyplot as plt
     housing.hist(bins=50, figsize=(20,15))
```

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



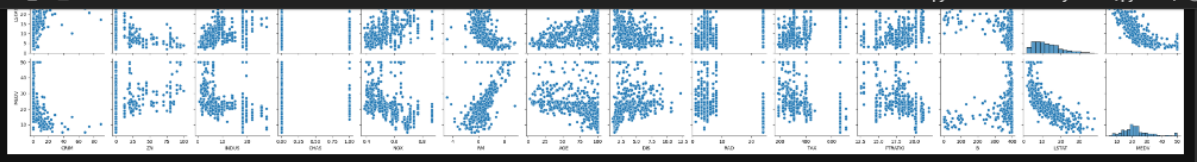
#Data Preprocessing

#we categorize the features depending on their datatype (int, float, object) and then calculate the number of them!!

jupyter Real Estate - Price Prediction Using Machine Learning Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)



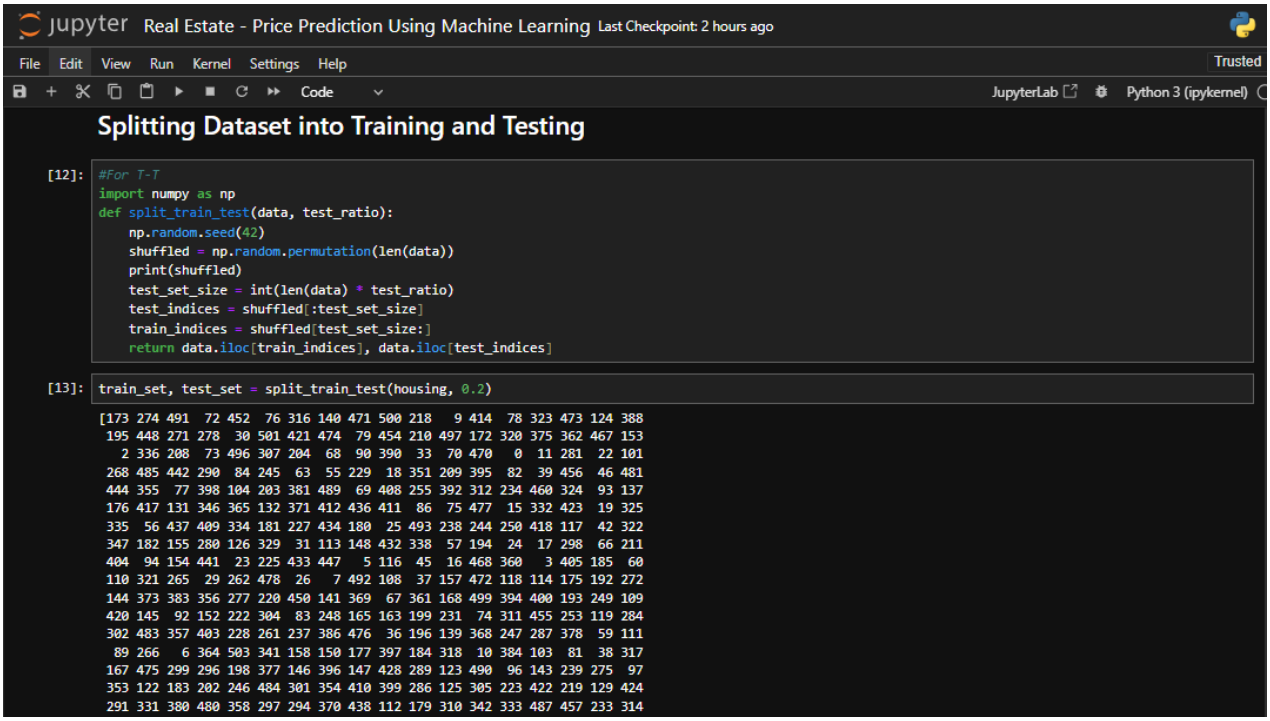
```
[11]: #Data Preprocessing
      #we categorize the features depending on their datatype (int, float, object) and then calculate the number of them!!
      obj = (housing.dtypes == 'object')
      object_cols = list(obj[obj].index)
      print("Categorical variables:",len(object_cols))

      int_ = (housing.dtypes == 'int')
      num_cols = list(int_[int_].index)
      print("Integer variables:",len(num_cols))

      fl = (housing.dtypes == 'float')
      fl_cols = list(fl[fl].index)
      print("Float variables:",len(fl_cols))
```

```
Categorical variables: 0
Integer variables: 0
Float variables: 11
```

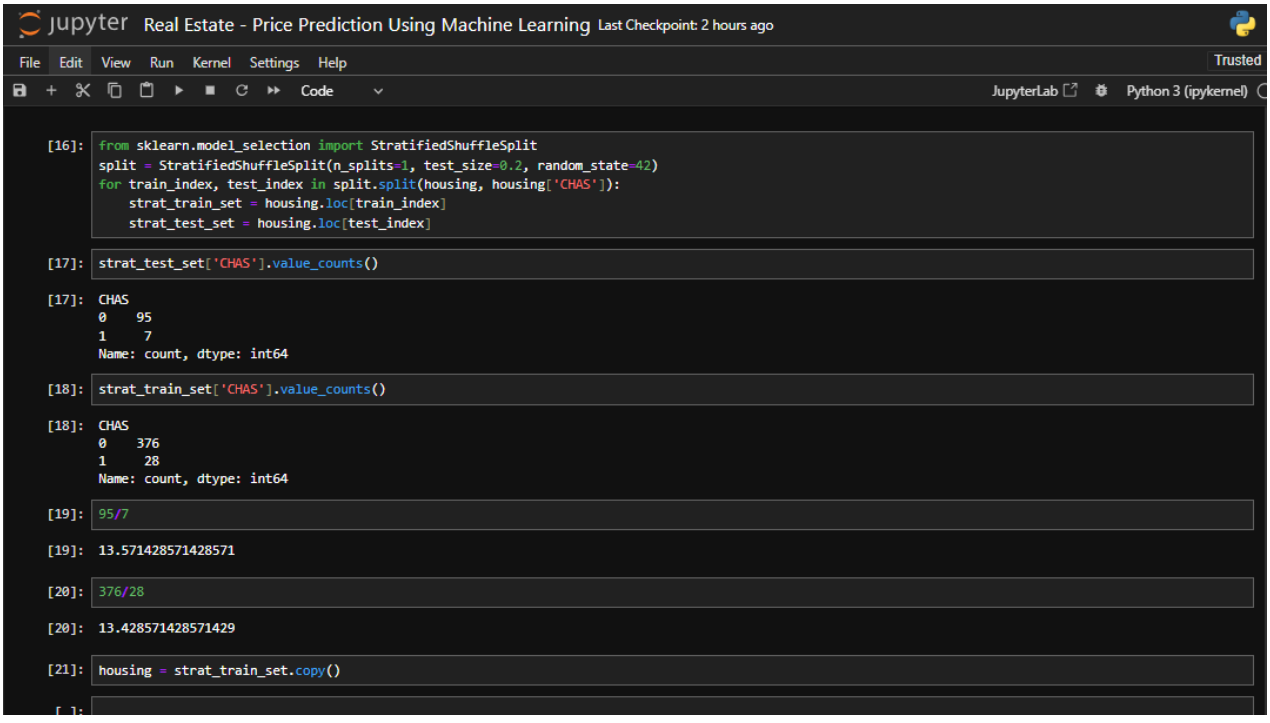
```
[ ]:
```

Splitting Dataset into Training and Testing

#Splitting Dataset into Train-Test



#. **StratifiedShuffleSplit** It is used for splitting a dataset into train and test sets while maintaining the distribution of the target variable across the splits



#Correlations – The `corr()` method finds the correlation of each column in a DataFrame.

jupyter Real Estate - Price Prediction Using Machine Learning Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

Looking For Correlations

```
[22]: #For Correlations
      corr_matrix = housing.corr()
      corr_matrix['MEDV'].sort_values(ascending=False)
```

```
[22]: MEDV       1.000000
      RM         0.680857
      B          0.361761
      ZN         0.339741
      DIS        0.240451
      CHAS       0.205066
      AGE       -0.364596
      RAD       -0.374693
      CRIM      -0.393715
      NOX       -0.422873
      TAX       -0.456657
      INDUS     -0.473516
      PTRATIO   -0.493534
      LSTAT     -0.740494
      Name: MEDV, dtype: float64
```

```
[23]: #Plotting Size Attr
      from pandas.plotting import scatter_matrix
      attributes = ["MEDV", "RM", "ZN", "LSTAT"]
      scatter_matrix(housing[attributes], figsize = (12,8))
```

```
[23]: array([[<Axes: xlabel='MEDV', ylabel='MEDV'>,
              <Axes: xlabel='RM', ylabel='MEDV'>,
              <Axes: xlabel='ZN', ylabel='MEDV'>,
              <Axes: xlabel='LSTAT', ylabel='MEDV'>],
             [<Axes: xlabel='MEDV', ylabel='RM'>,
```

The **scatter_matrix** function is part of the pandas plotting tools and is used to create a matrix of scatterplots.

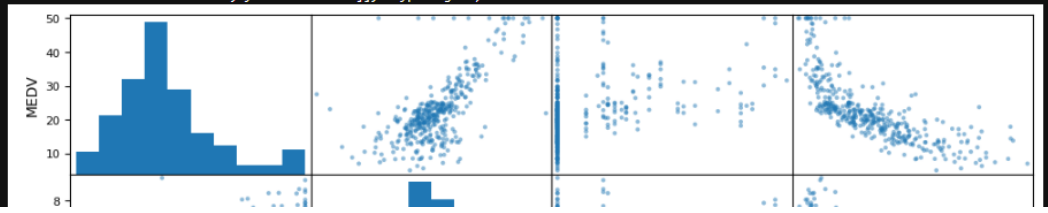
jupyter Real Estate - Price Prediction Using Machine Learning Last Checkpoint: 2 hours ago

File Edit View Run Kernel Settings Help Trusted

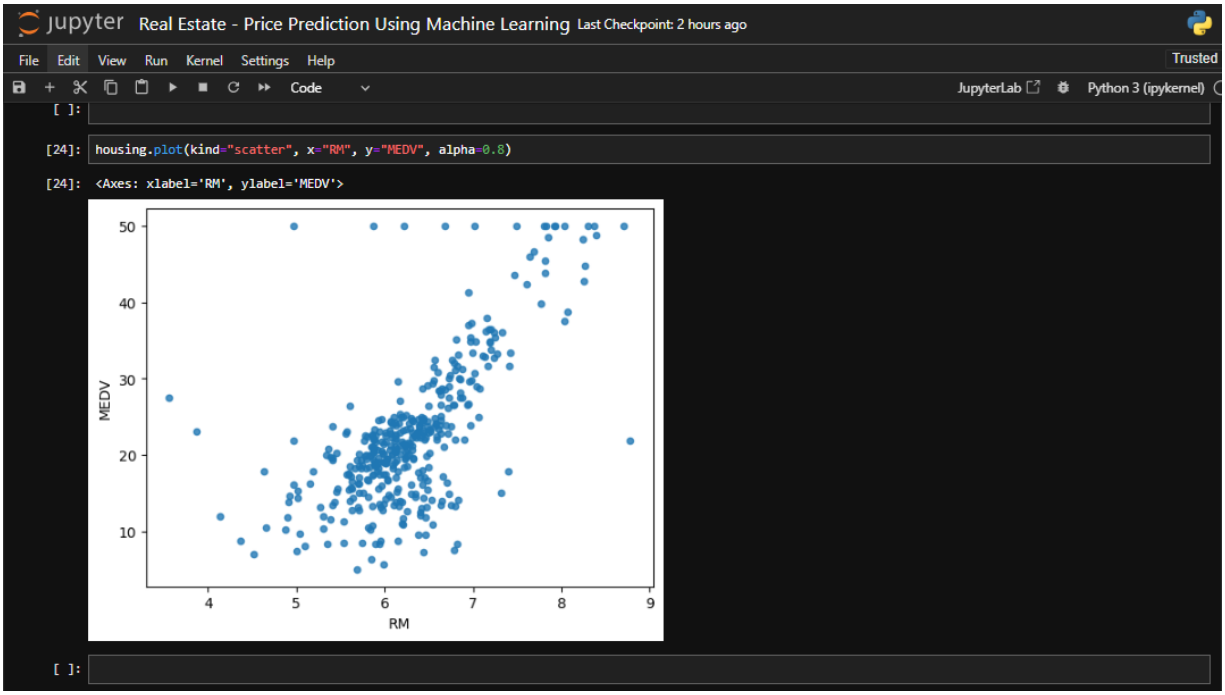
Code JupyterLab Python 3 (ipykernel)

```
[23]: #Plotting Size Attr
      from pandas.plotting import scatter_matrix
      attributes = ["MEDV", "RM", "ZN", "LSTAT"]
      scatter_matrix(housing[attributes], figsize = (12,8))
```

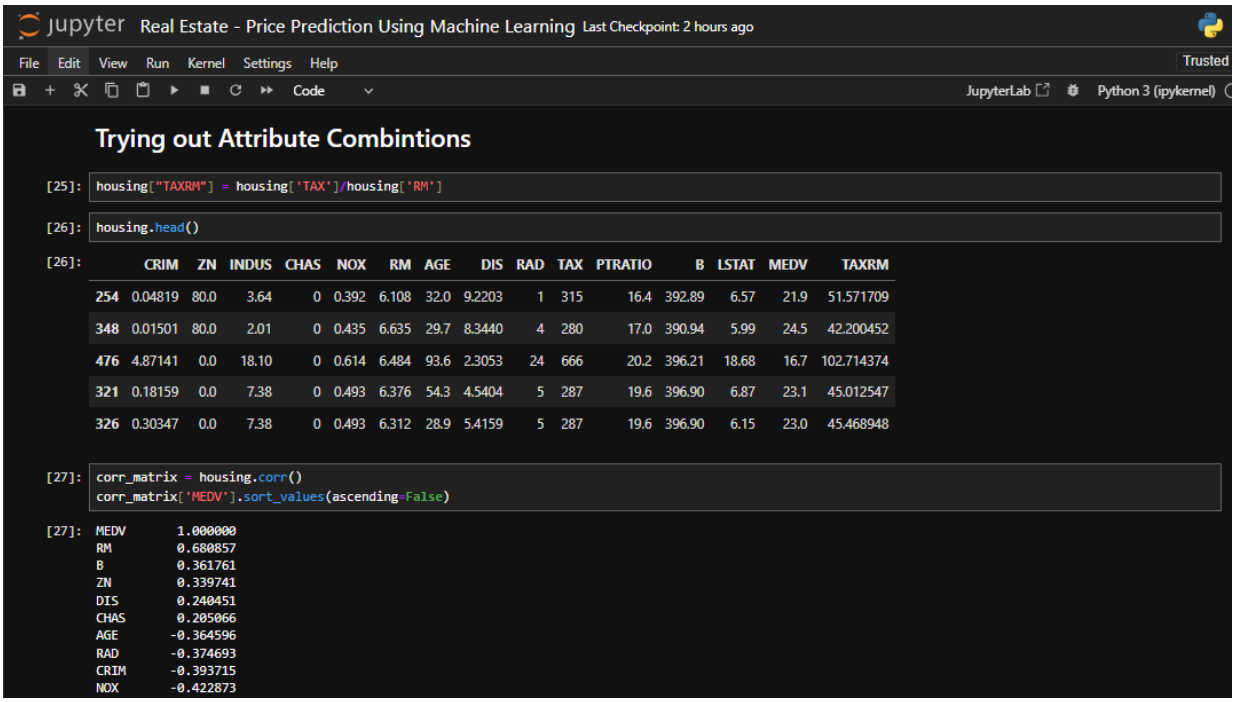
```
[23]: array([[<Axes: xlabel='MEDV', ylabel='MEDV'>,
              <Axes: xlabel='RM', ylabel='MEDV'>,
              <Axes: xlabel='ZN', ylabel='MEDV'>,
              <Axes: xlabel='LSTAT', ylabel='MEDV'>],
             [<Axes: xlabel='MEDV', ylabel='RM'>,
              <Axes: xlabel='RM', ylabel='RM'>,
              <Axes: xlabel='ZN', ylabel='RM'>,
              <Axes: xlabel='LSTAT', ylabel='RM'>],
             [<Axes: xlabel='MEDV', ylabel='ZN'>,
              <Axes: xlabel='RM', ylabel='ZN'>,
              <Axes: xlabel='ZN', ylabel='ZN'>,
              <Axes: xlabel='LSTAT', ylabel='ZN'>],
             [<Axes: xlabel='MEDV', ylabel='LSTAT'>,
              <Axes: xlabel='RM', ylabel='LSTAT'>,
              <Axes: xlabel='ZN', ylabel='LSTAT'>,
              <Axes: xlabel='LSTAT', ylabel='LSTAT'>]], dtype=object)
```



#Pandas DataFrames, you can use the **scatter** method directly on the DataFrame for quick plotting.



Trying out Attribute Combintions

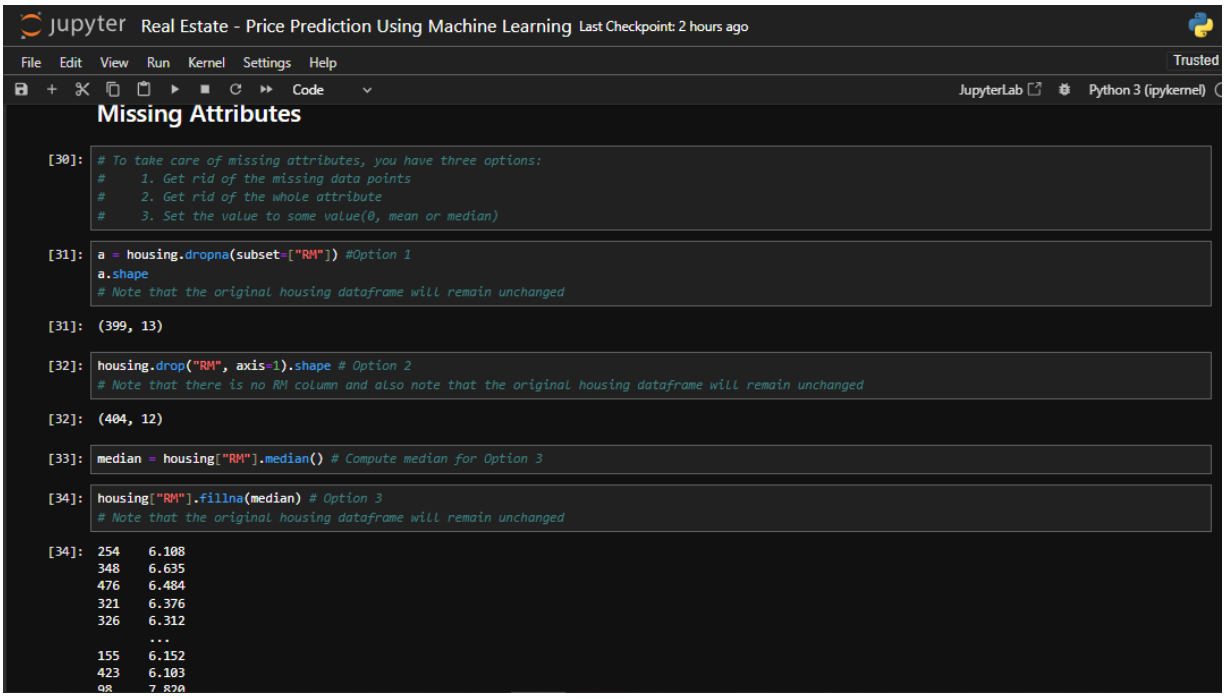


 # To take care of missing attributes, you have three options:

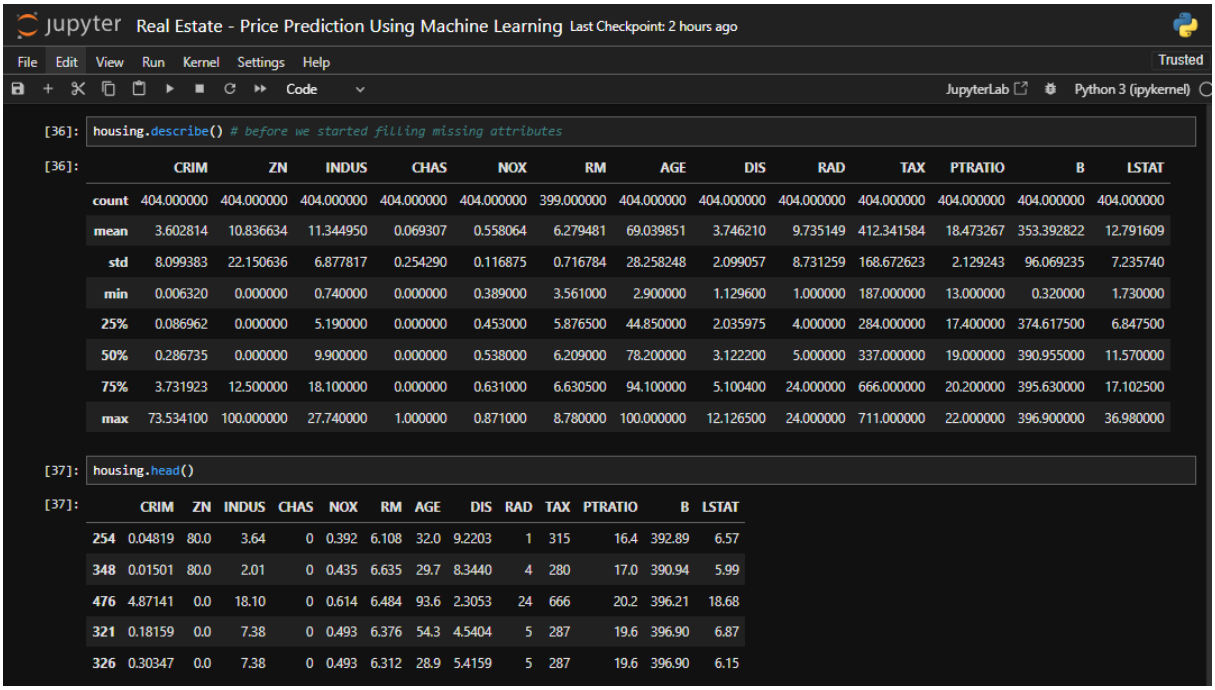
1. Get rid of the missing data points

2. Get rid of the whole attribute

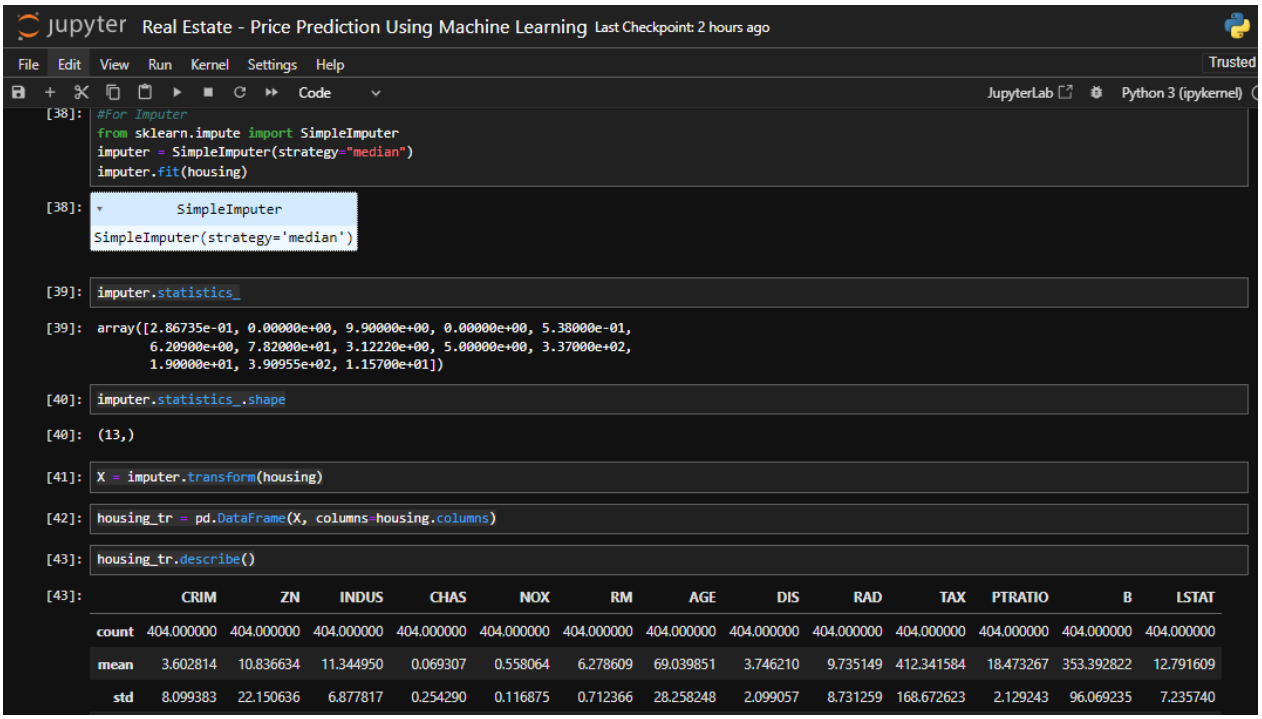
3. Set the value to some value(0, mean or median)



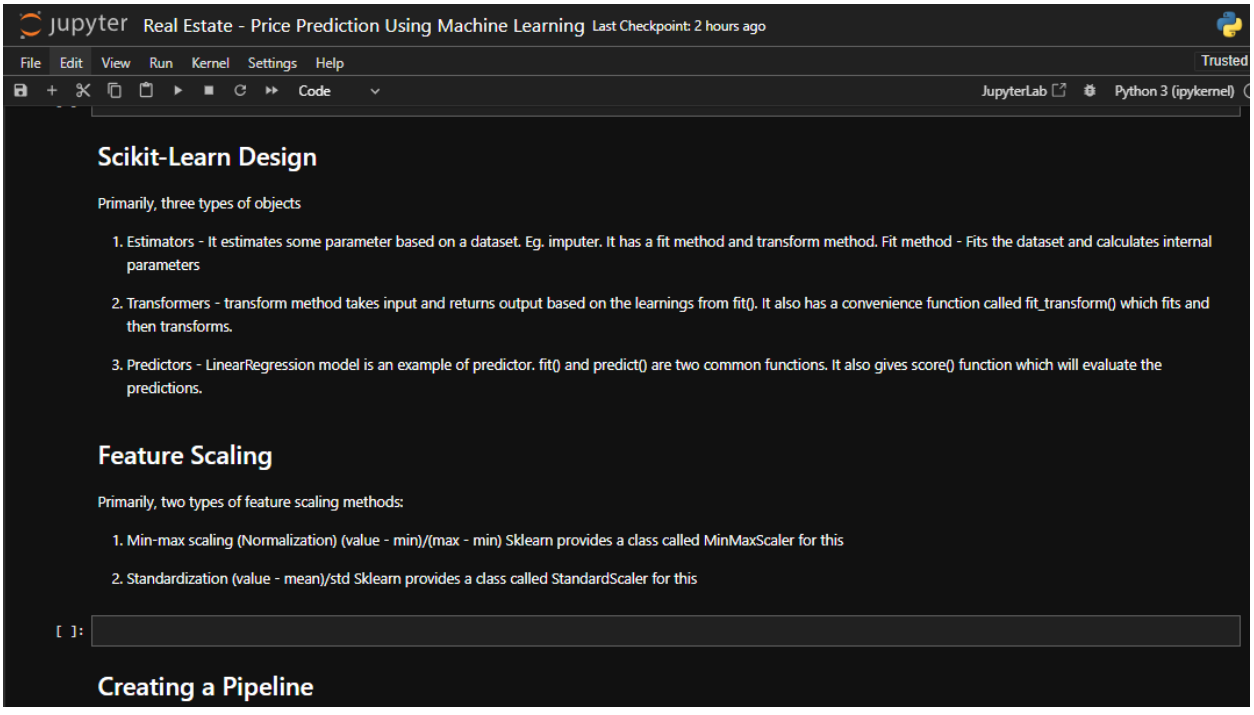
 # Hosing Describe and Count Attributes



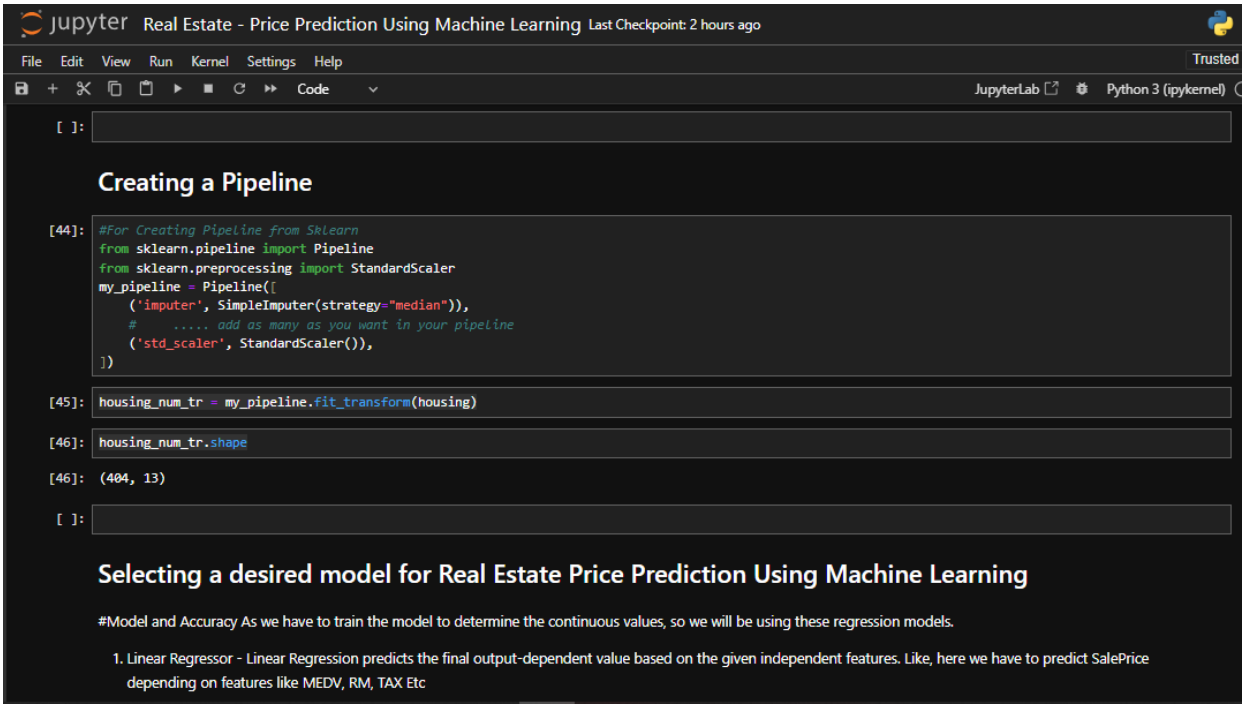
SimpleImputer class provides a simple strategy to handle missing values by replacing them with a constant value or the mean, median, or most frequent value along each column.



Scikit-learn provides simple and efficient tools for data analysis and modeling, including various machine learning algorithms for classification, regression, clustering, dimensionality reduction, and more.

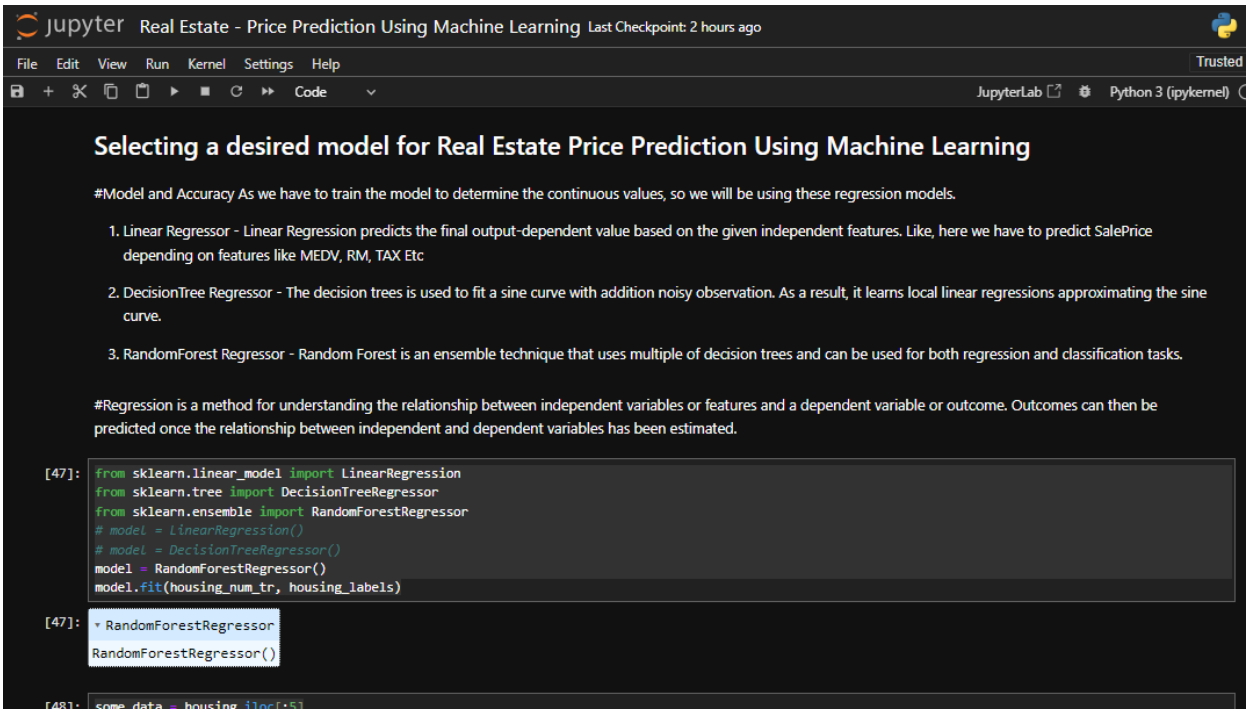


The **Pipeline** class in scikit-learn is a tool to simplify the construction, training, and evaluation of complex machine learning workflows.

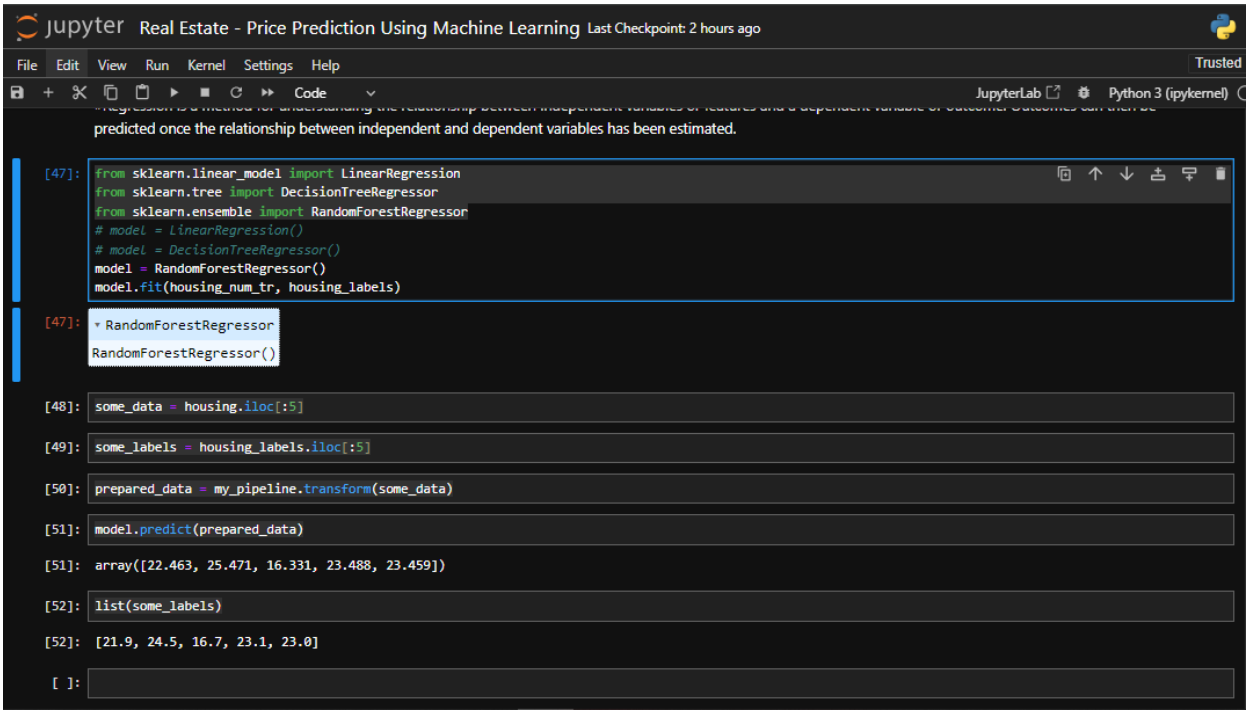


#Model For Predictions ----

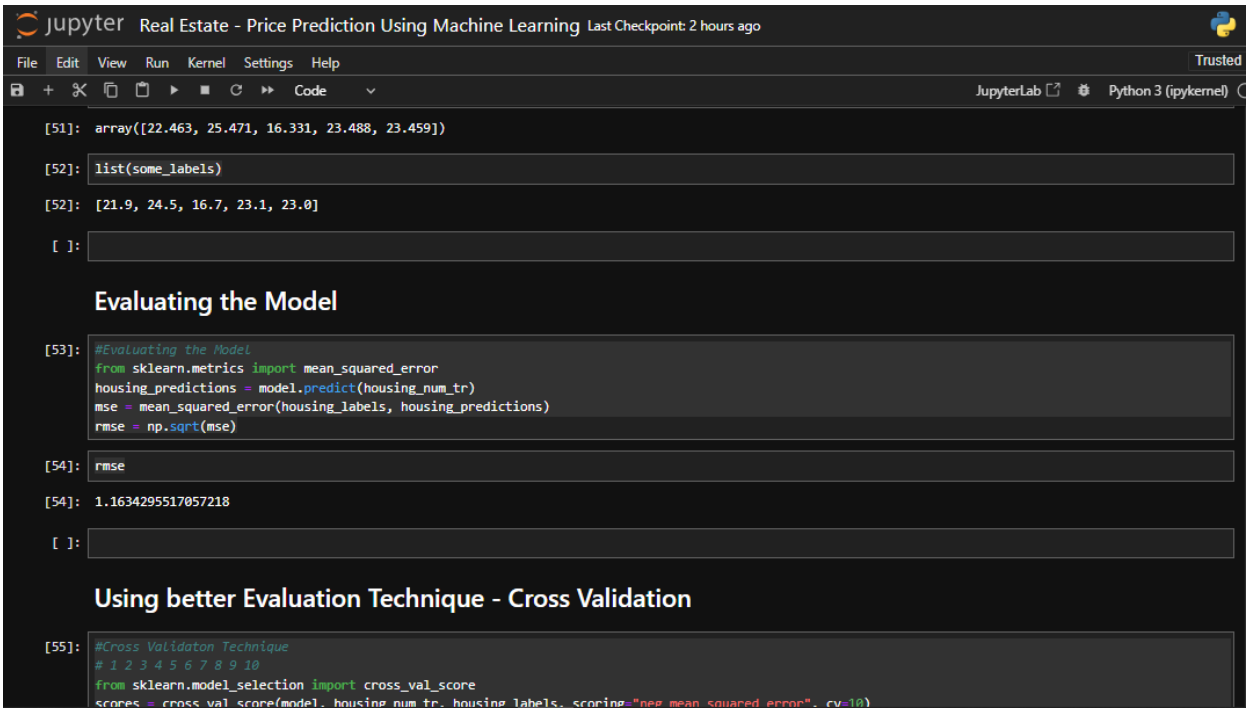
- Liner Regression
- DecisionTress Regression
- RandomForest Regression



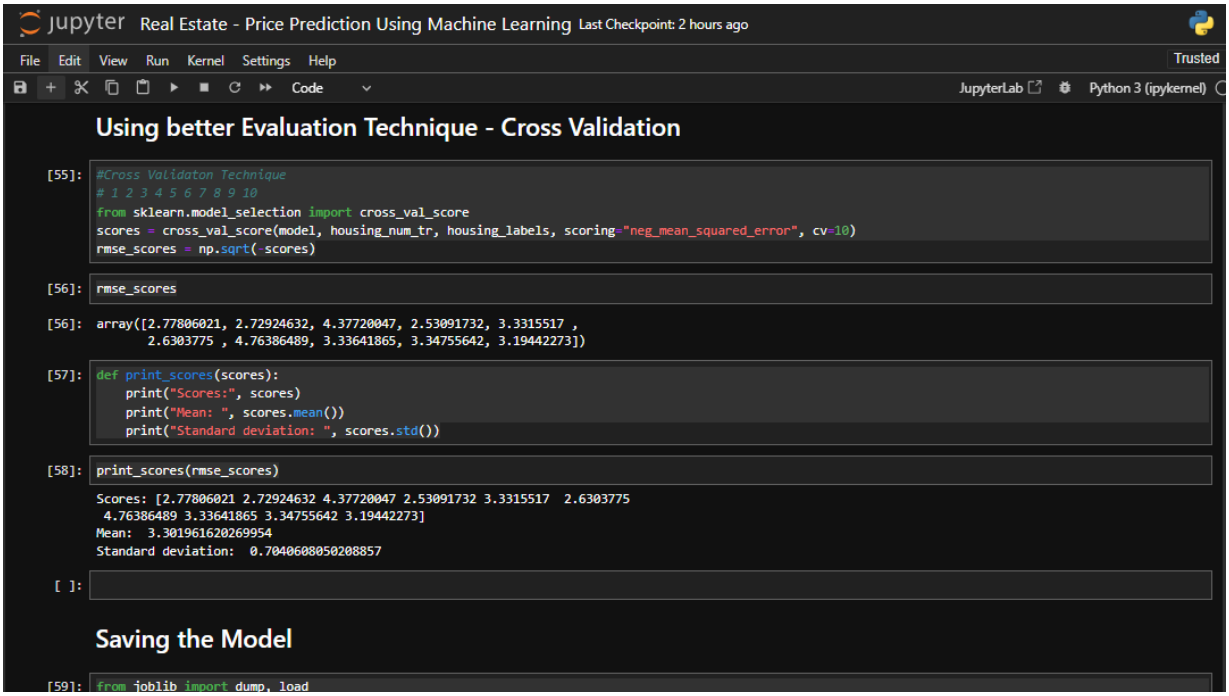
#**Random Forest** is an ensemble learning method that constructs a multitude of decision trees during training and outputs the mean prediction of the individual trees for regression tasks And Predict Price



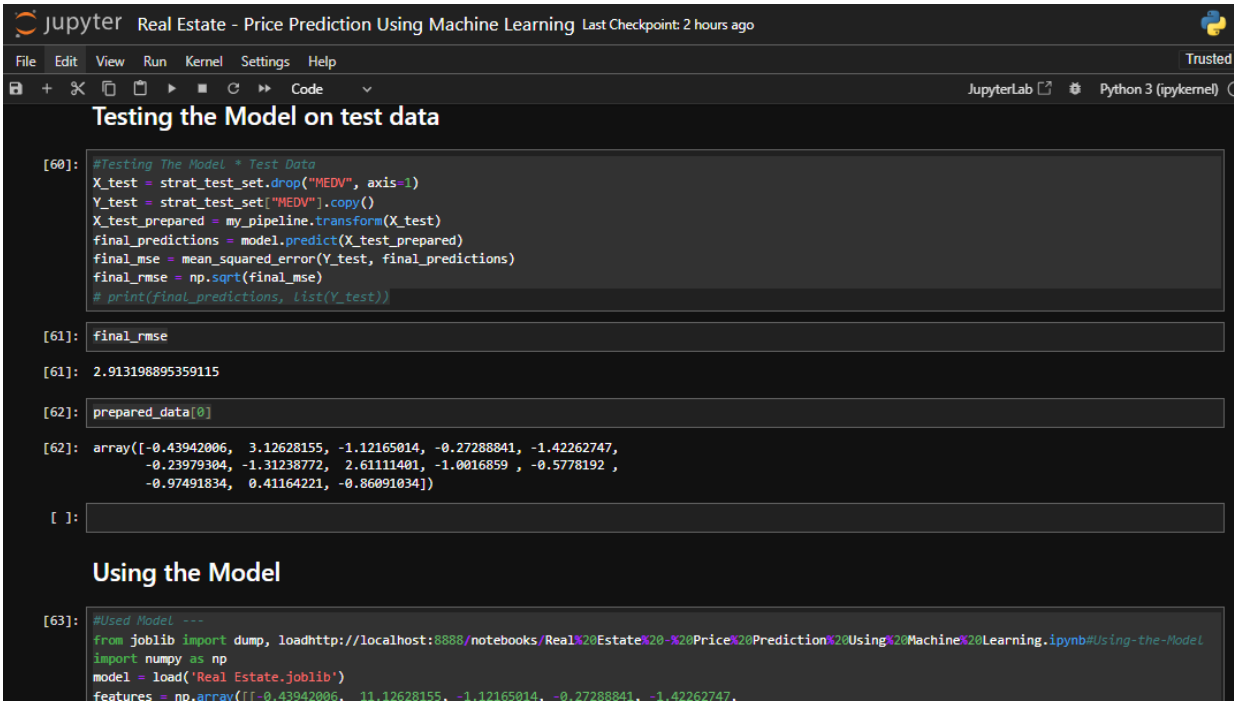
Evaluating a Machine learning Model--



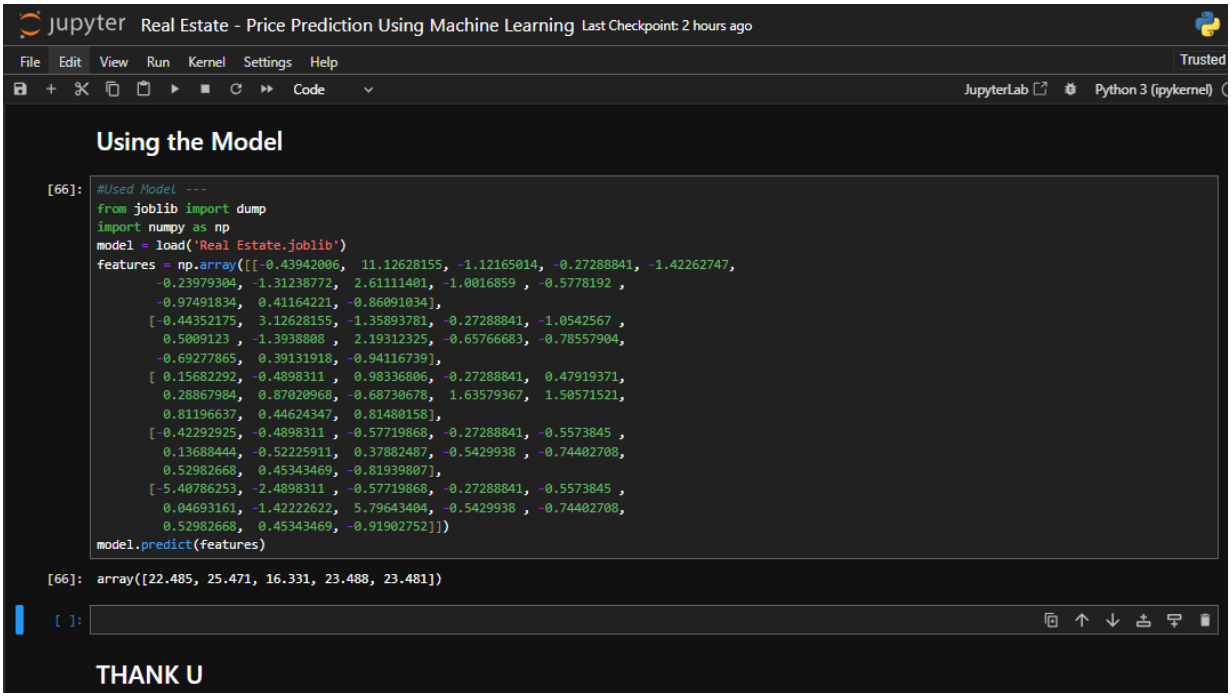
 # Using better Evaluation Technique - Cross Validation is used in machine learning to assess the performance of a model and to reduce the risk of overfitting.



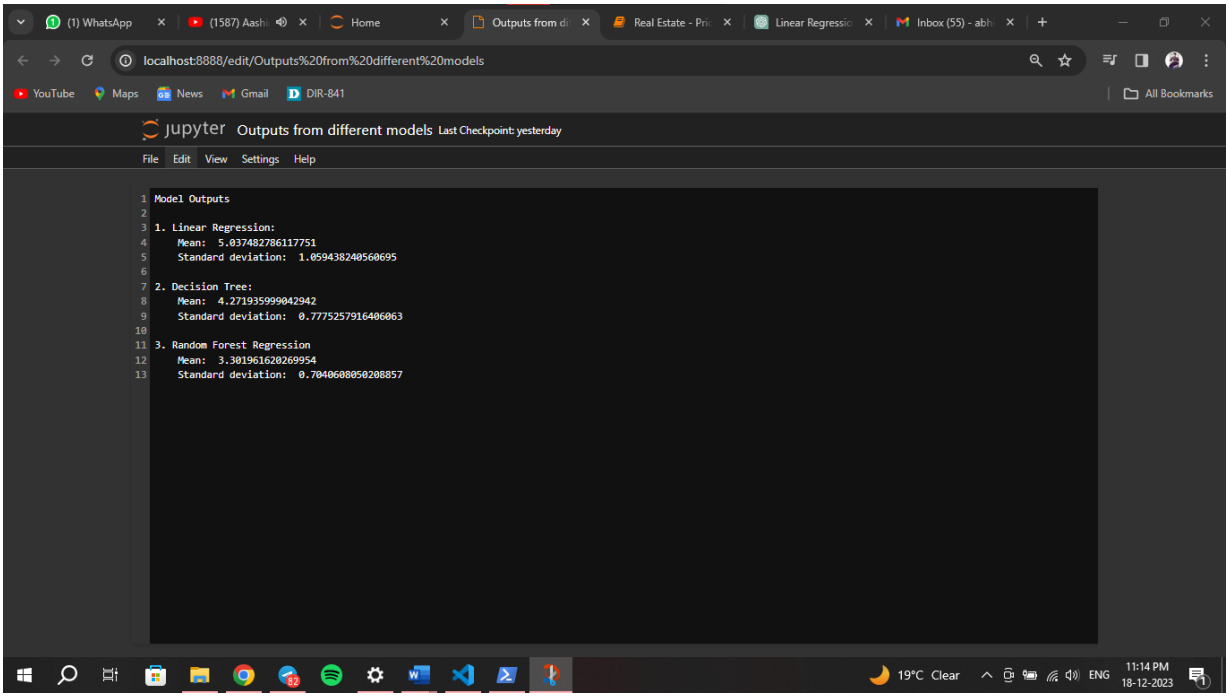
 # Testing The Model on The Test Data and predicts Aculeate Price and Cost



Uses Of The Model



Output From Different – Different Models (“Price Prediction”)- ML



7. SOFTWARE TESTING

7.1 UNIT TESTING

- for a particular Testing the accuracy of the calculation of the Real Estate Price Prediction period.
- Testing the accuracy of the Random Forest Regression algorithm in predicting the future values of the indices.

7.2 INTEGRATION TESTING

- Testing the accuracy of the data input and output between the Linear regression, Random Forest Regression algorithm and the Real Estate Price Prediction datasets.
- Testing the accuracy of the user interface that displays the analysis and prediction results.

7.3 SYSTEM TESTING

- Testing the overall accuracy of the analysis and prediction results over the Dataset Last Few Years.
- Testing the user interface, functionality, and performance of the system under different conditions and scenarios.

7.4 BLACK BOX TESTING

- Testing the user interface and functionality of the system without any knowledge of the underlying code and algorithms.
- Testing the accuracy of the analysis and prediction results without any knowledge of the data sources and statistical models used.

7.5 WHITE BOX TESTING

- Testing the accuracy of the linear regression algorithm by examining the source code and performing tests on its individual components.
- Testing the accuracy of the data input and output between the datasets and the DecisionTree regression algorithm by examining the code that processes the data.

8. CONCLUSION

This project provides a strong foundation for real estate price prediction, and further refinement and exploration of features could yield even more accurate predictions. The insights gained from this analysis contribute to a better understanding of the factors influencing real estate prices in the given context.

We also used different python packages like NumPy, pandas, matplotlib etc. For importing the dataset, and also for doing data pre-processing we used pandas. For doing exploratory data analysis we used matplotlib package in python.

This model further helps people understand whether this place is more suited for them based on heatmap correlation. It also helps people looking to sell a house at best time for greater profit.

9. FUTURE ENHANCEMENTS

Future enhancements for a real estate price prediction project can involve improving the model, expanding the dataset, incorporating additional features, and making the prediction system more user-friendly.

- ✓ Fine-Tuning Hyperparameters.
- ✓ Feature Engineering.
- ✓ Interactive Visualization.
- ✓ Cloud Deployment.
- ✓ Model Monitoring and Maintenance.

10. BIBLIOGRAPHY

- *The Hundred-Page Machine Learning Book* by Andriy Burkov
- *Machine Learning for Hackers* by Drew Conway and John Myles White
- https://chat.openai.com/chat.
- https://youtube.com/
- https://google.com/
- Hands-On Machine Learning for Algorithmic Trading" by Stefan Jansen