KARNATAK LAW SOCIETY’S

**GOGTE INSTITUTE OF TECHNOLOGY**

UDYAMBAG, BELGAVI – 590008

Academic Year 2019-20



***Department of Computer Science and Engineering***

A Project Synopsis

On

**“A Real Estate Data analysis using machine learning model. ”**

***Submitted By***

Vismay gunda 2GI17CS169

Under the guidance of

**Prof. Umesh Kulkarni**

**TABLE OF CONTENTS:-**

|  |  |
| --- | --- |
| *Serial**No.* | *Title* |
| ***1.*** | *Abstract* |
| ***2.*** | *Introduction* |
| ***3.*** | *Objectives* |
| ***4*** | *Minimum Hardware and Software requirements* |
| ***5.*** | *Scope* |
| ***6.*** | *Functional and Non-functional Requirements* |
| ***7.*** | *Figures for the System* |
| ***8.*** | *Testing Summary* |
| ***9.*** | *Source Code* |
| ***10.*** | *Screenshots* |
| ***11.*** | *Conclusion* |
| ***12.*** | *References* |

**ABSTRACT :**

# People looking to buy a new home tend to be more conservative with their budgets and market strategies.

# The existing system involves calculation of house prices without the necessary prediction about future market trends

# and price increase.The real estate system Give the functionality for buyers, allowing them to search for houses by features or

# address.Along with this, when the user will search for the property, original property value and predicted property value will be displayed. By

# analysing previous market trends and price ranges, and also upcoming developments future prices will be predicted.

# For the price prediction we will be using classification algorithm.This application will help customers to invest in an estate without approaching an agent. It also decreases the risk involved in the transaction.

# The property, original property value and predicted property value will be displayed. By analysing previous market trends

# and price ranges, and also upcoming developments future prices will be predicted. For the price prediction we will be

# using classification algorithm.This application will help customers to invest in an estate

# without approaching an agent. It also decreases the risk involved in the transaction..

**INTRODUCTION :**

Problems faced during buying a house:

1) Buying a house is a stressful thing.

2) Buyers are generally not aware of factors that influence the house

prices.

3) Many problems are faced during buying a house.

4) Hence real estate agents are trusted with the communication

between buyers and sellers as well as laying down a legal contract for

the transfer. This just creates a middle man and increases the cost of

houses.

5)They believe that it depends upon:

a) The square foot area

b) Neighbourhood

c) The no. of bedrooms

6) But it depends upon many factors also.....Such as:

a) No. of storeys.

b) Area outside the house.

c) Rooms on one floor.

**Data set introduction:**

1. Title: Boston Housing Data

2. 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

3. 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.

4. Relevant Information:

Concerns housing values in suburbs of Boston.

5. Number of Instances: 506

6. Number of Attributes: 13 continuous attributes (including "class"

attribute "MEDV"), 1 binary-valued attribute.

7. 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: None.

**OBJECTIVES:**

* To predict price value for given estate using Batch processing machine learning model.
* Analysing standard deviation and mean values for predicting models.
* Analysing data with 3 supervised machine learning algorithms mainly

a)Decision Tree

b)Linear regression

c)Random forest Regression

* Finding least standard deviation among various model analysis.
* Concluding Random forest regression supervised learning as the best model with least deviation of data.

**MINIMUM HARDWARE & SOFTWARE REQUIREMENTS :**

Minimum Hardware Requirements

* + - Hard Disk space: 20GB
    - Ram: 2GB

Software Requirements

* + - Operating System: Windows 7 or above, macOS and Linux
    - Python: 3.6.x or above

Note:

The program should work fine with Python versions < 3.6.x as well. Since, I’ve developed the program using Python 3.6.x, it is being specified as a requirement.

**SCOPE :**

* The Machine learning model can be operated as mediator free Real estate money prediction for client.
* Users will be able to achieve a certain definite task or get an answer to the query.

**FUNCTIONAL & NON-FUNCTIONAL REQUIREMENTS :**

**Functional Requirements:**

* The Machine learning model should be able to respond to any textual input it receives.

1. If the Model doesn’t understand the input, it should ask for a more simplified input.
2. If the Model understands the input, it should respond with the correct prediction for given input.

* The machine learning model needs to analyse real estate data and Split into training data and testing data.
* Training data need to fit into three machine learning algorithms mainly random forest,linear regression and decision tree.
* Model is analysed using test data which needs to be predicted.

**Non Functional Requirements:**

* The software should be open source with the source code uploaded on Github.
* The Model should be able to respond under 30 seconds.

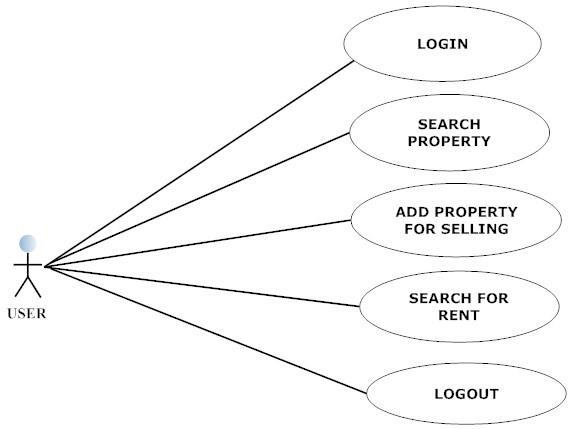
**Figures for the Real estate prediction machine learning model:**

**A. Use Case diagram**

Use case diagrams belong to the category of behavioural diagram of UML diagrams. Use case diagrams aim to present a graphical overview of the functionality provided by the system. It consists of a set of actions (referred to as use cases) that the concerned system can perform, one or more actors, and dependencies among them.

A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

Use case diagram is a platform that can provide a common understanding for the end-users, developers and the domain experts. It is used to capture the basic functionality i.e. use cases, and the users of those available functionality, i.e. actors, from a given problem statement.



­­­

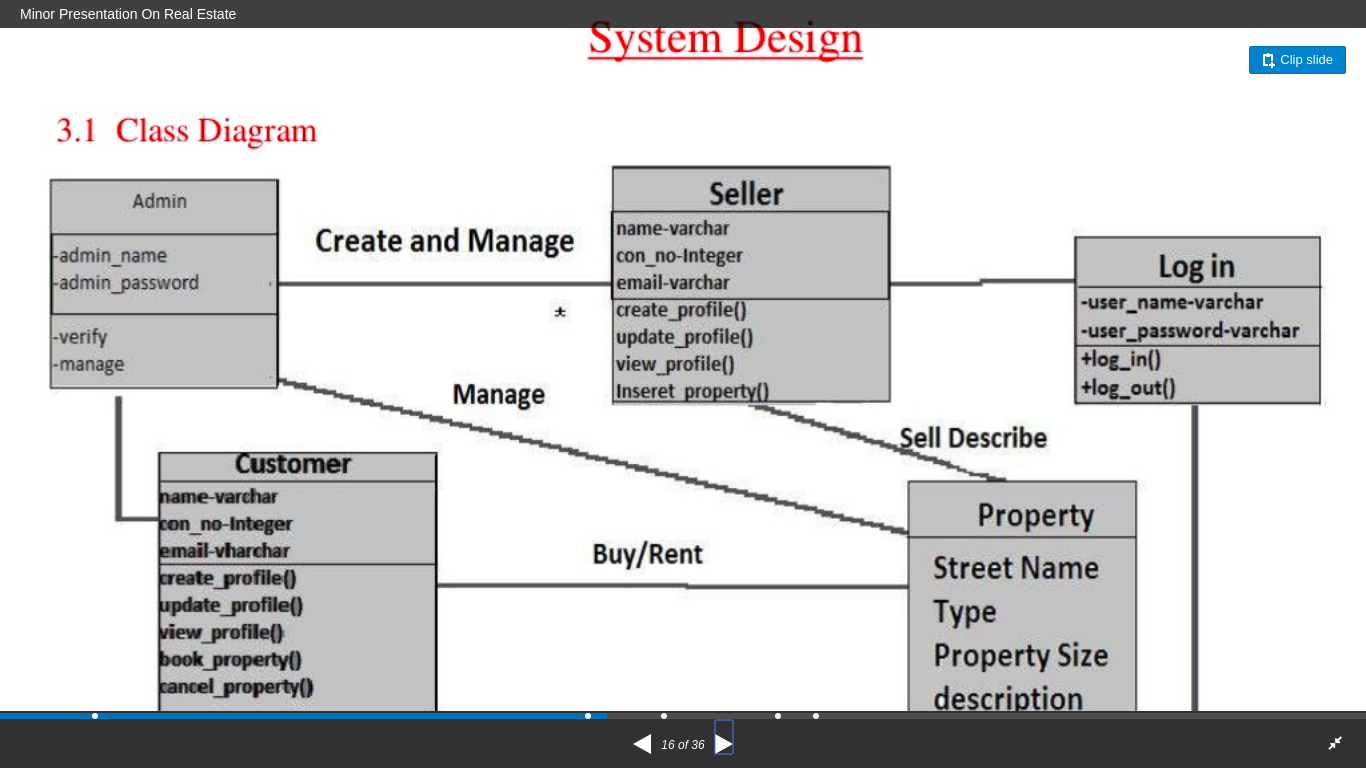
**B. Class diagram:**

In software engineering, a class diagram in the [Unified Modelling Language (UML)](https://en.wikipedia.org/wiki/Unified_Modeling_Language) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

It is a graphical representation for describing a system in context of it’s static construction. Class diagram contains the system classes with its data members, operations and relationships between classes.

A UML class diagram is made up of:

* A set of classes.
* A set of relationships between classes.

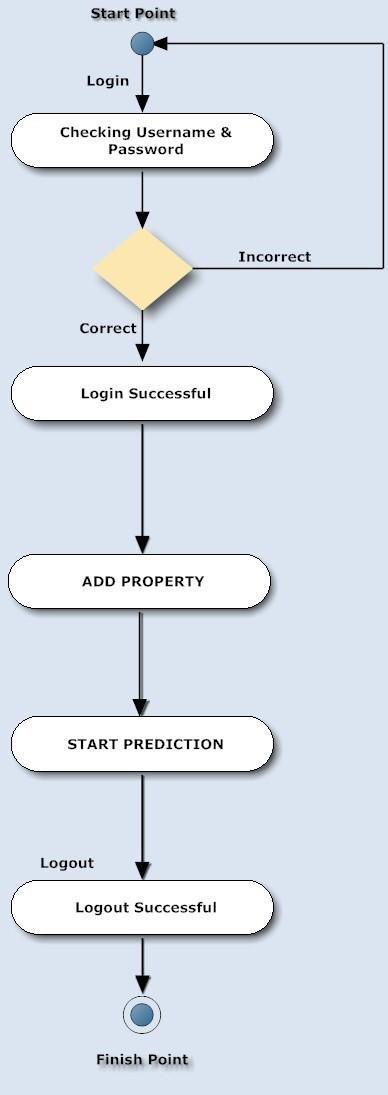
****

**C. Activity Diagram:**

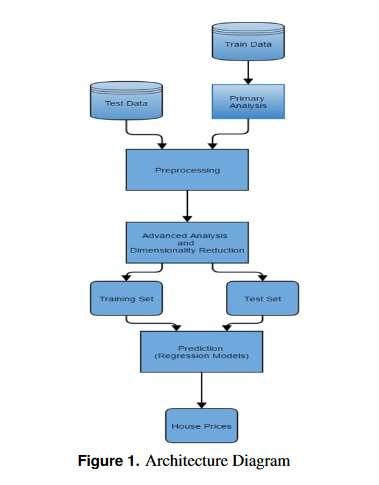
Activity diagram is another important behavioural diagram in [UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modelling the flow from one activity to another activity.

Activity Diagrams describe how activities are coordinated to provide a service which can be at different levels of abstraction. Typically, an event needs to be achieved by some operations, particularly where the operation is intended to achieve a number of different things that require coordination, or how the events in a single use case relate to one another, in particular, use cases where activities may overlap and require coordination. It is also suitable for modelling how a collection of use cases coordinate to represent business workflows.

An activity diagram visually presents a series of actions or flow of control in a system similar to a [flowchart](https://www.smartdraw.com/flowchart/) or a [data flow diagram.](https://www.smartdraw.com/data-flow-diagram/) Activity diagrams are often used in business process modeling. They can also describe the steps in a [use case diagram.](https://www.smartdraw.com/use-case-diagram/) Activities modeled can be sequential and concurrent. In both cases an activity diagram will have a beginning (an initial state) and an end (a final state).



**D. State Diagram:**

A state diagram is a diagram used in computer science to describe the behavior of a system considering all the possible states of an object when an event occurs. This behavior is represented and analyzed in a series of events that occur in one or more possible states. Each diagram represents objects and tracks the various states of these objects throughout the system.

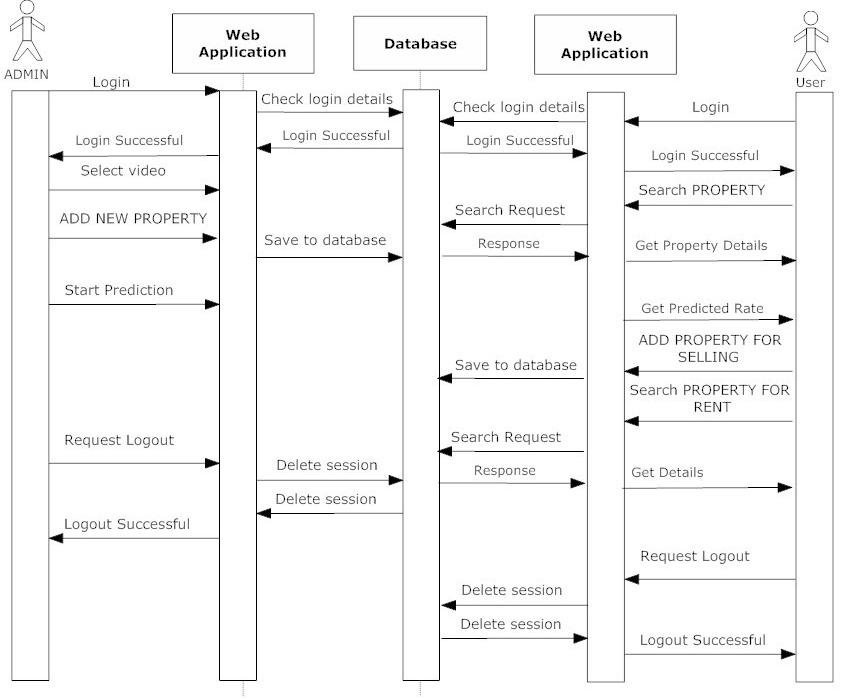
**E. Sequence Diagram**

Sequence diagram is a visual representation of various messages sent between the objects of a system and the order in which these messages occur.

[UML](https://en.wikipedia.org/wiki/Unified_Modeling_Language) Sequence Diagrams are interaction diagrams that detail how operations are carried out. They capture the interaction between objects in the context of a collaboration. Sequence Diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

Sequence Diagrams captures:

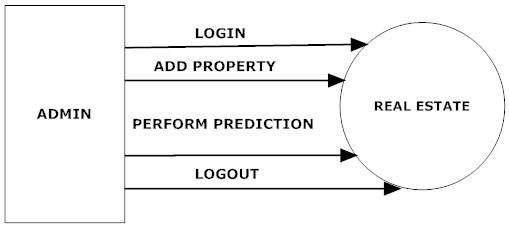
* the interaction that takes place in a collaboration that either realizes a use case or an operation (instance diagrams or generic diagrams)
* high-level interactions between user of the system and the system, between the system and other systems, or between subsystems (sometimes known as system sequence diagrams)



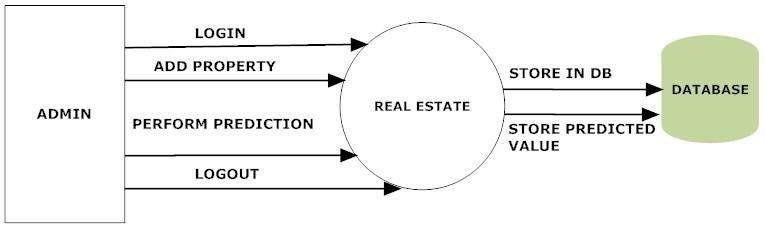
**F. Data flow Diagram:**

A picture is worth a thousand words. A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both.

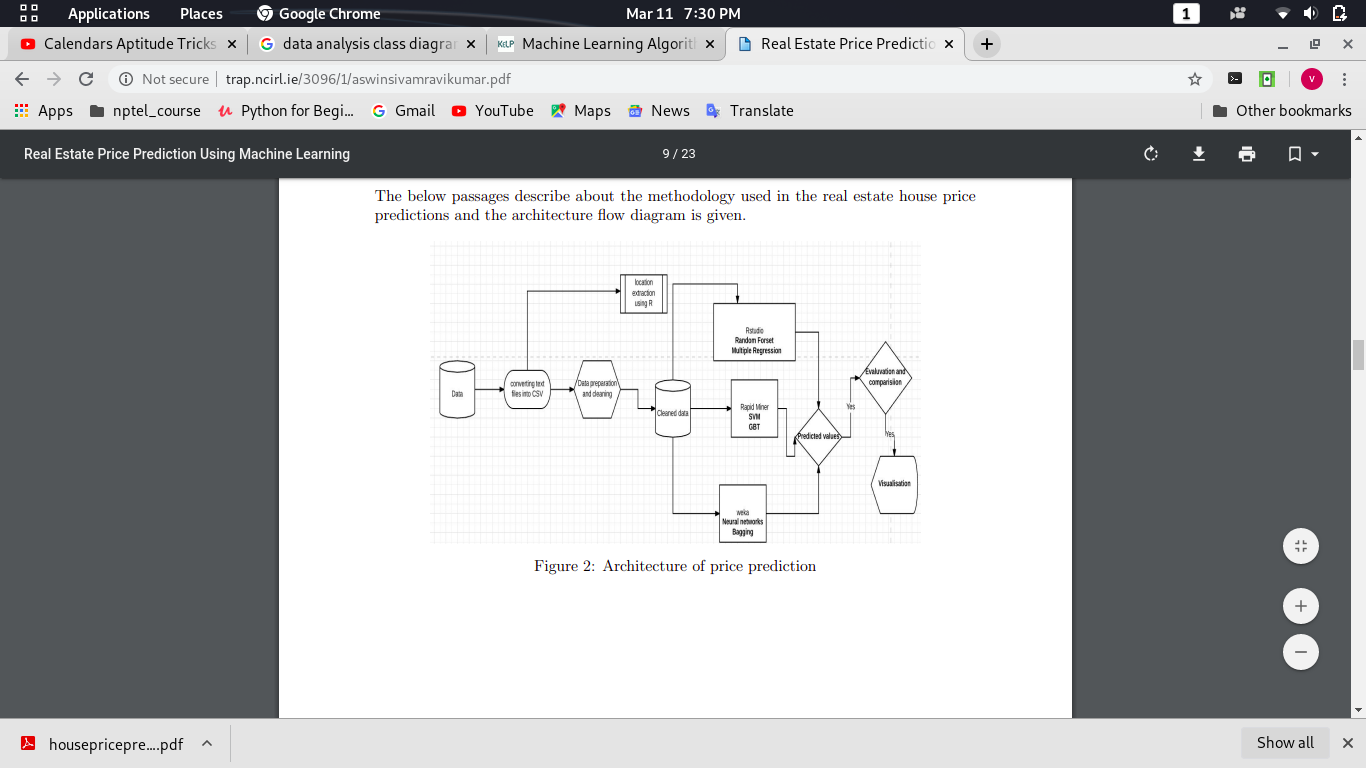
A data flow diagram (or DFD) is a graphical representation of the flow of data through an information system. It shows how information is input to and output from the system, information sources and destinations, and where it's stored.

****

DFD level(0)

 DFD level(1)

**G. Retrieval-Based Chatbot Architecture Diagram:**

****

.

**Different Testing Techniques used in the project:**

**The Testing Techniques used in this project are as follows:**

* **Unit Testing:**

Individual components of the Machine learning model are tested to ensure that they operate correctly.

* **Test for Intent Classification:**

To verify that the intent in the user’s message/query is being correctly identified.

* **Test for Entity Recognition:**

To verify that the entity present in the user’s message/query is being correctly recognised.

* **Module Testing:**

Module is collection of independent components such as an object class, an abstract data type or some loser collection of procedures and functions. A module encapsulates related components so can be tested without other modules.

* **Test for Response Selection:**

To verify that the response with the ‘best fit’ for the given input is being selected as a response from the set of available responses, to be returned as a reply to the user.

* **Sub System Testing:**

This phase involves testing collection of modules, which have been integrated into subsystems. Subsystems may be independently designed and implemented. The most common problems, which arise in the large software systems, are subsystems interface mismatches. The subsystem test process should therefore concentrate on the detection of interface errors by rigorously exercising these interfaces.

* **Test for the PredictionClass Implementation:**

To verify that the tasks of pre-processing input, Intent Classification, Entity Recognition, Response Generation and Selection work well when integrated together.

* **System Testing:**

The subsystems are integrated to makeup the entire system. The testing process is concerned with finding errors, which result from unanticipated interactions between subsystems and system components. It is also concerned with validating that the system is functional or non-functional requirements.

* **Test for the System as a whole:**

To verify that the system is able to grab user input, pre-process it, perform all the tasks necessary ( Intent Classification, Entity Recognition, Response Selection) for an appropriate response to be returned to the user.

**SOURCE CODE:**

ln[1]: import pandas as pd

In [2]: housing = pd.read\_csv("data.csv")

In [3]: housing.head()

In [4]: housing.info()

In [5]: housing['CHAS'].value\_counts()

In [6]: housing.describe()

In [7]: %matplotlib inline

In [8]: # # For plotting histogram

import matplotlib.pyplot as plt

housing.hist(bins=50, figsize=(20, 15))

ln [9]:# For learning purpose

#Train test split

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]

In [10]:train\_set, test\_set = split\_train\_test(housing, 0.2)

In [11]:print(f"Rows in train set: {len(train\_set)}\nRows in test set: {len(test\_set)}\n")

In [12]: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")

In [13]: from sklearn.model\_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n\_splits=1, test\_size=0.2, r andom\_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]

In [14]: strat\_test\_set['CHAS'].value\_counts()

In [15]: strat\_train\_set['CHAS'].value\_counts()

In [16]: housing = strat\_train\_set.copy()

In [17]: corr\_matrix = housing.corr()

corr\_matrix['MEDV'].sort\_values(ascending=False)

In [20]:from pandas.plotting import scatter\_matrix

attributes = ["MEDV", "RM", "ZN", "LSTAT"]

scatter\_matrix(housing[attributes], figsize = (12,8))

In [21]:housing.plot(kind="scatter", x="RM", y="MEDV", alpha=0.8)

In [22]: housing["TAXRM"] = housing['TAX']/housing['RM']

In [23]: housing.head()

In [24]: corr\_matrix = housing.corr()

corr\_matrix['MEDV'].sort\_values(ascending=False)

In [25]:housing.plot(kind="scatter", x="TAXRM", y="MEDV", alpha=0.8)

In [26]:housing = strat\_train\_set.drop("MEDV", axis=1)

housing\_labels = strat\_train\_set["MEDV"].copy()

In [28]: a = housing.dropna(subset=["RM"]) #Option 1

a.shape

# Note that the original housing dataframe will remain unchanged

In [29]:housing.drop("RM", axis=1).shape # Option 2

# Note that there is no RM column and also note that the original housing dataframe will rem

ain unchanged

In [30]: median = housing["RM"].median() # Compute median for Option 3

In [31]: housing["RM"].fillna(median) # Option 3

# Note that the original housing dataframe will remain unchanged

In [32]: housing.shape

In [33]: housing.describe() # before we started filling missing attributes

In [34]: from sklearn.impute import SimpleImputer

imputer = SimpleImputer(strategy="median")

imputer.fit(housing)

In [35]: imputer.statistics\_

In [36]: X = imputer.transform(housing)

In [37]: housing\_tr = pd.DataFrame(X, columns=housing.columns)

In [38]: housing\_tr.describe()

In [39]: 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()),

])

In [40]: housing\_num\_tr = my\_pipeline.fit\_transform(housing)

In [41]: housing\_num\_tr.shape

In [42]: 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)

In [43]: some\_data = housing.iloc[:5]

In [44]: some\_labels = housing\_labels.iloc[:5]

In [45]: prepared\_data = my\_pipeline.transform(some\_data)

In [46]: model.predict(prepared\_data)

In [47]: list(some\_labels)

In [48]: 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)

In [49]: rmse

In [50]: # 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\_er

ror", cv=10)

rmse\_scores = np.sqrt(-scores)

In [51]: rmse\_scores

In [52]: def print\_scores(scores):

print("Scores:", scores)

print("Mean: ", scores.mean())

print("Standard deviation: ", scores.std())

In [53]: print\_scores(rmse\_scores)

In [56]: from joblib import dump, load

dump(model, 'Dragon.joblib')

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))

In [58]: final\_rmse

In [64]: prepared\_data[0]

In [66]: from joblib import dump, load

import numpy as np

model = load('Dragon.joblib')

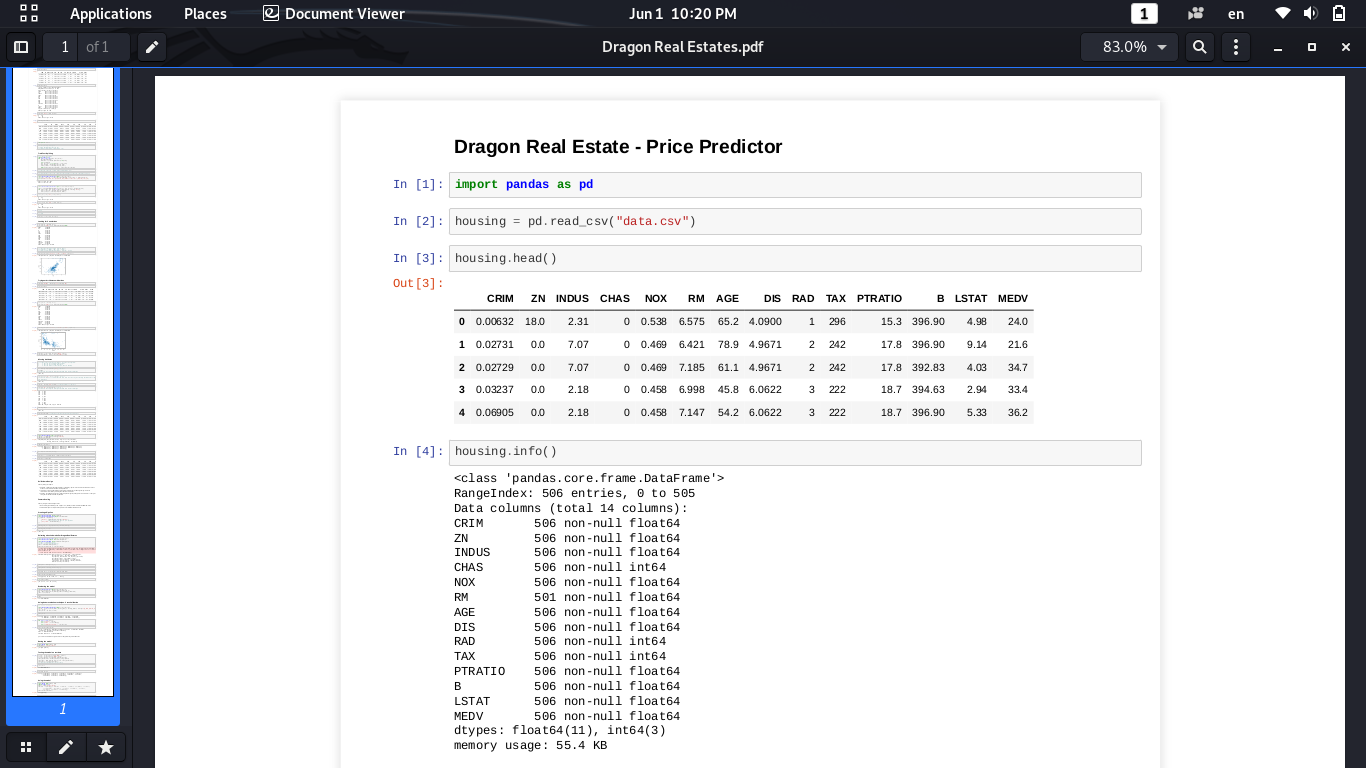
features = np.array([[-5.43942006, 4.12628155, - 1.6165014, -0.67288841, -1.42262747,

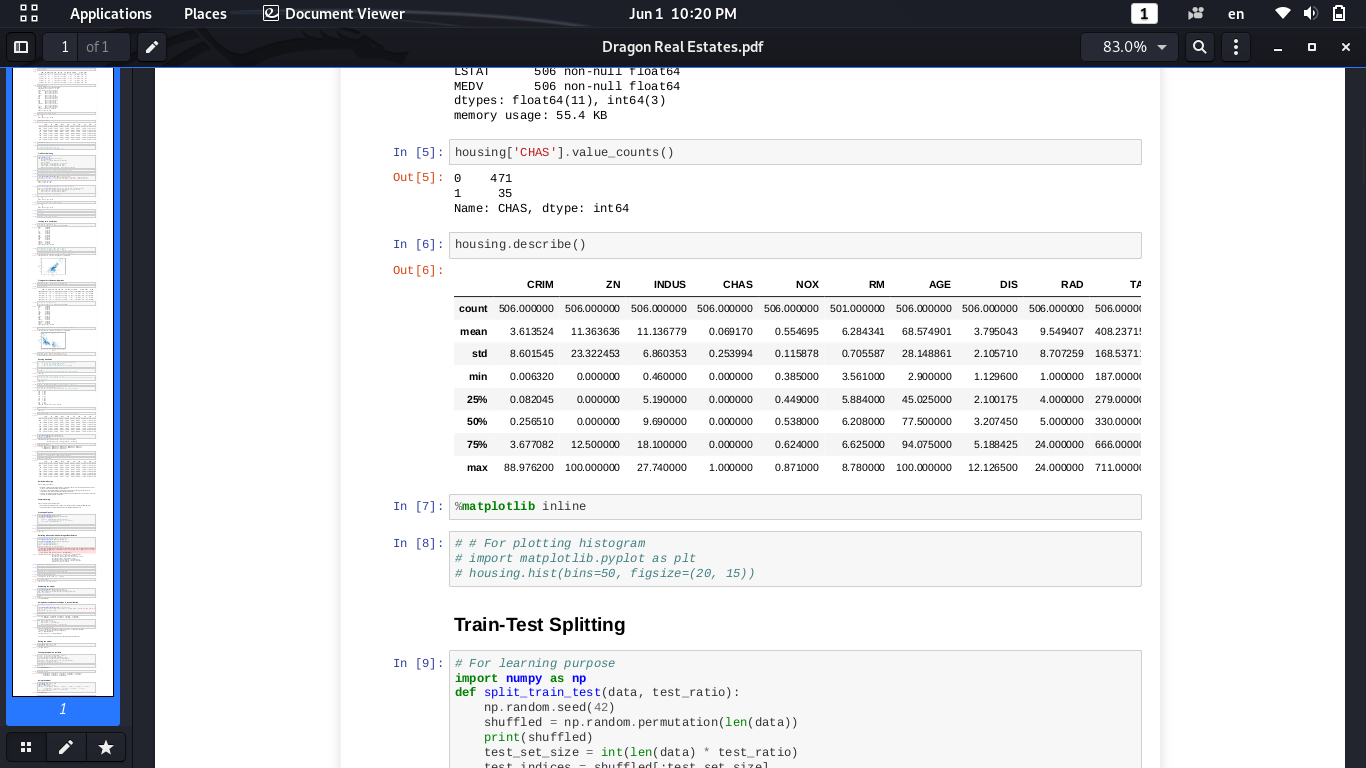
-11.44443979304, -49.31238772, 7.61111401, - 26.0016879 , -0.5778192 ,

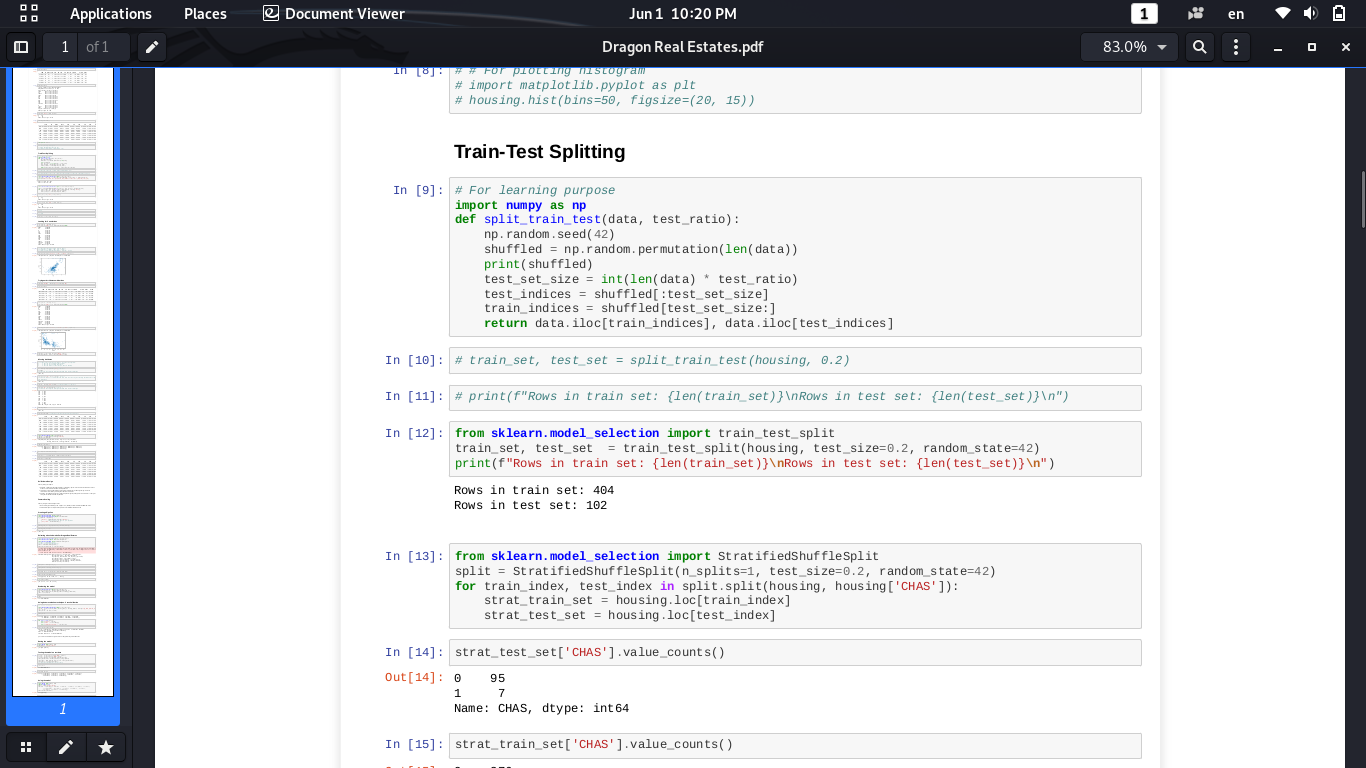
-0.97491834, 0.41164221, -66.86091034]])

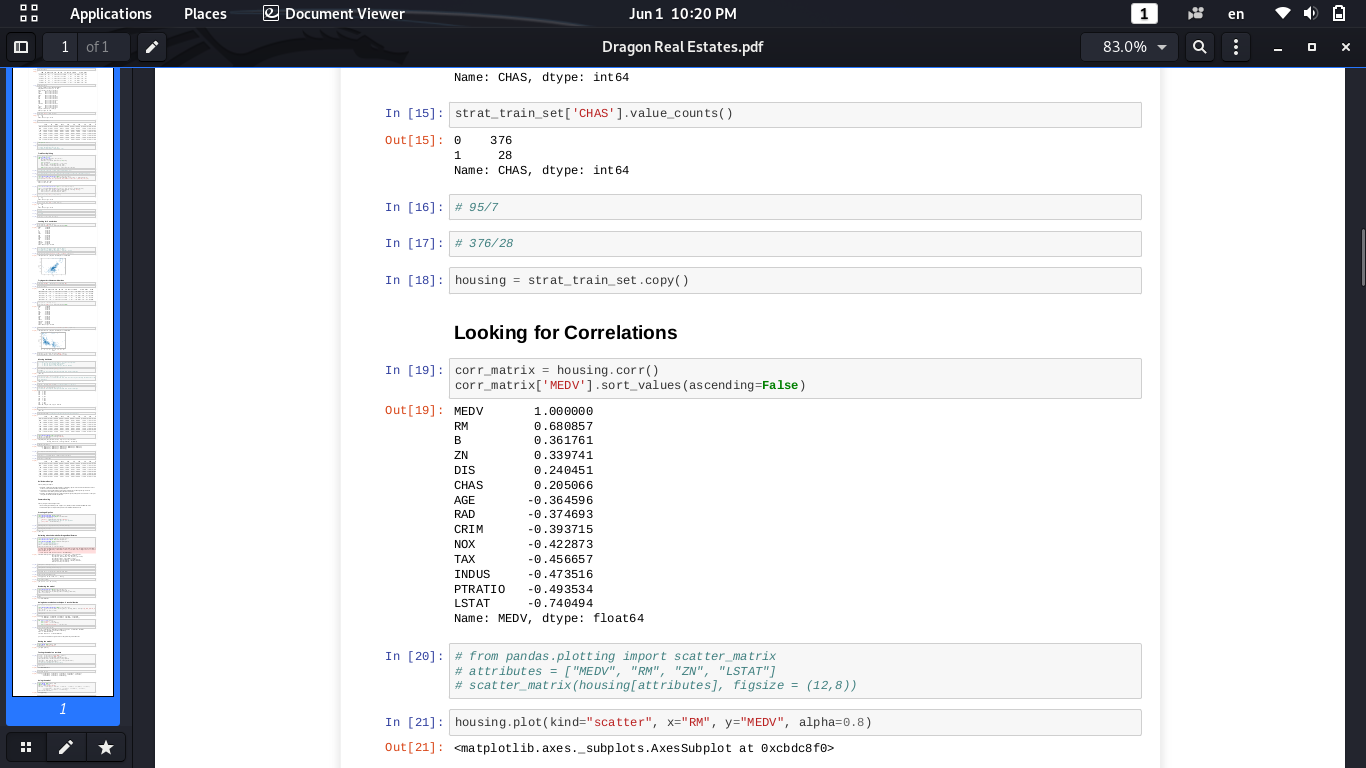
model.predict(features)

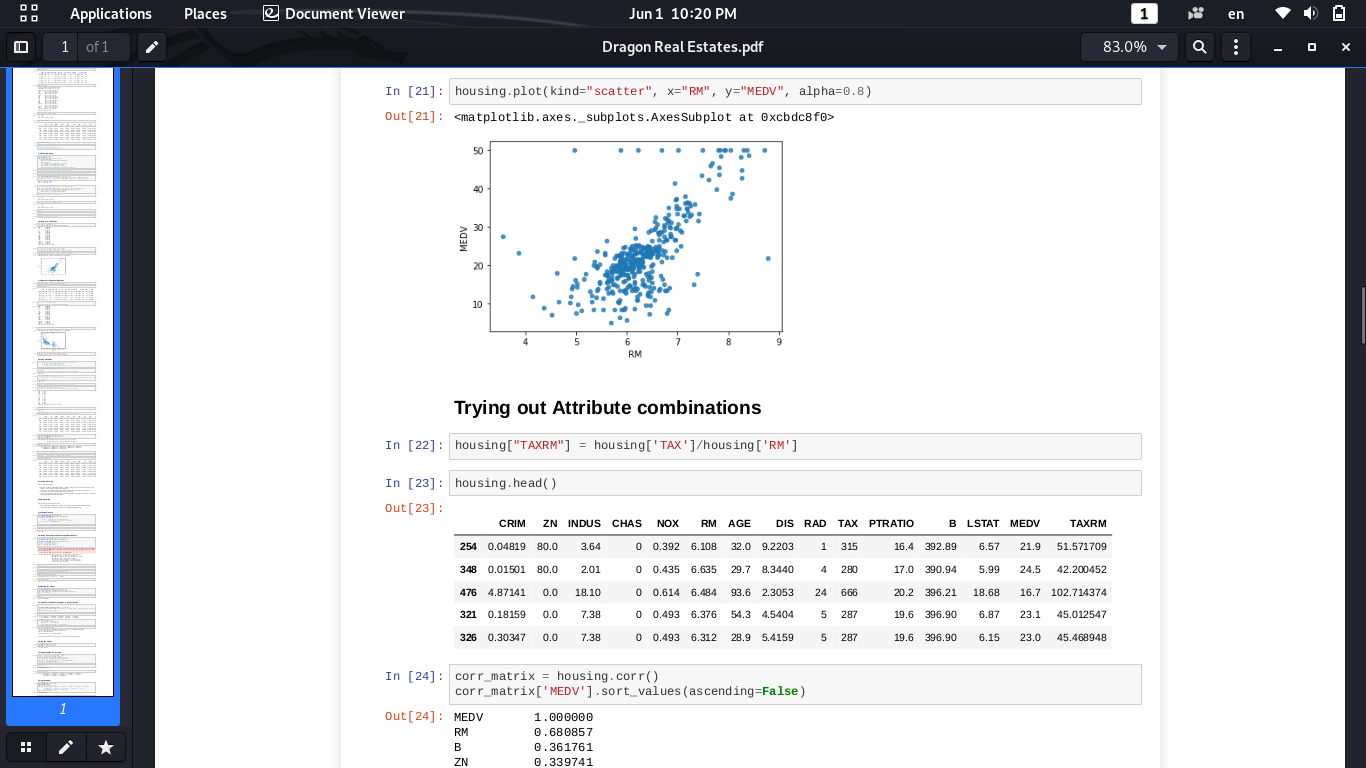
**Screenshots:**

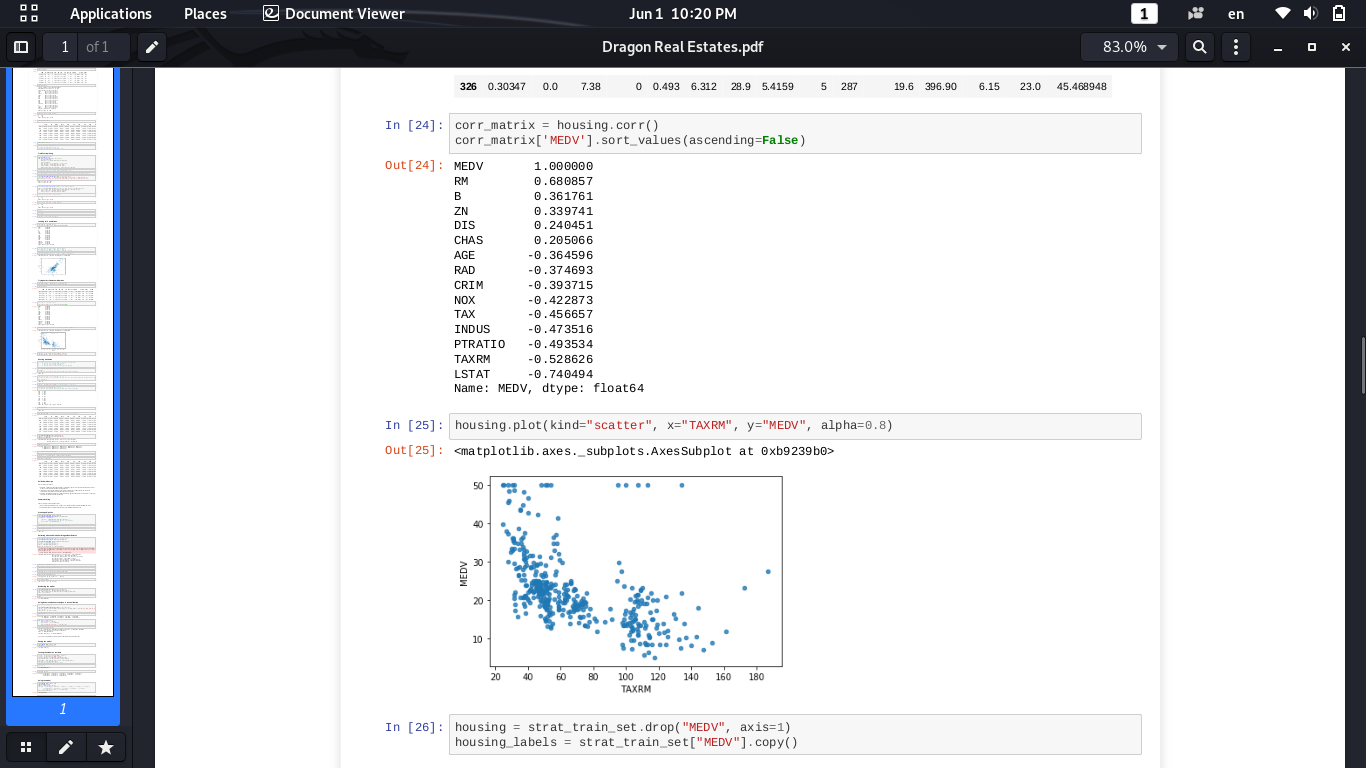
****

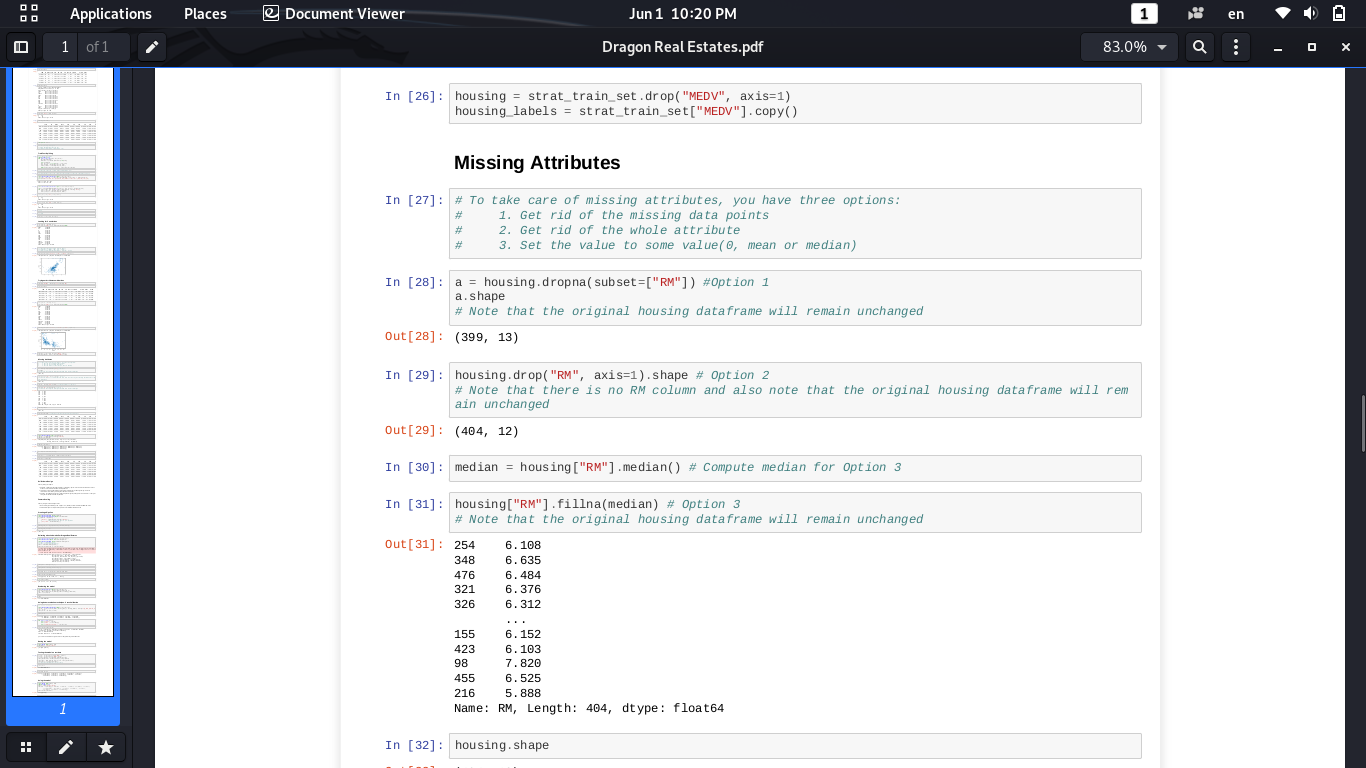


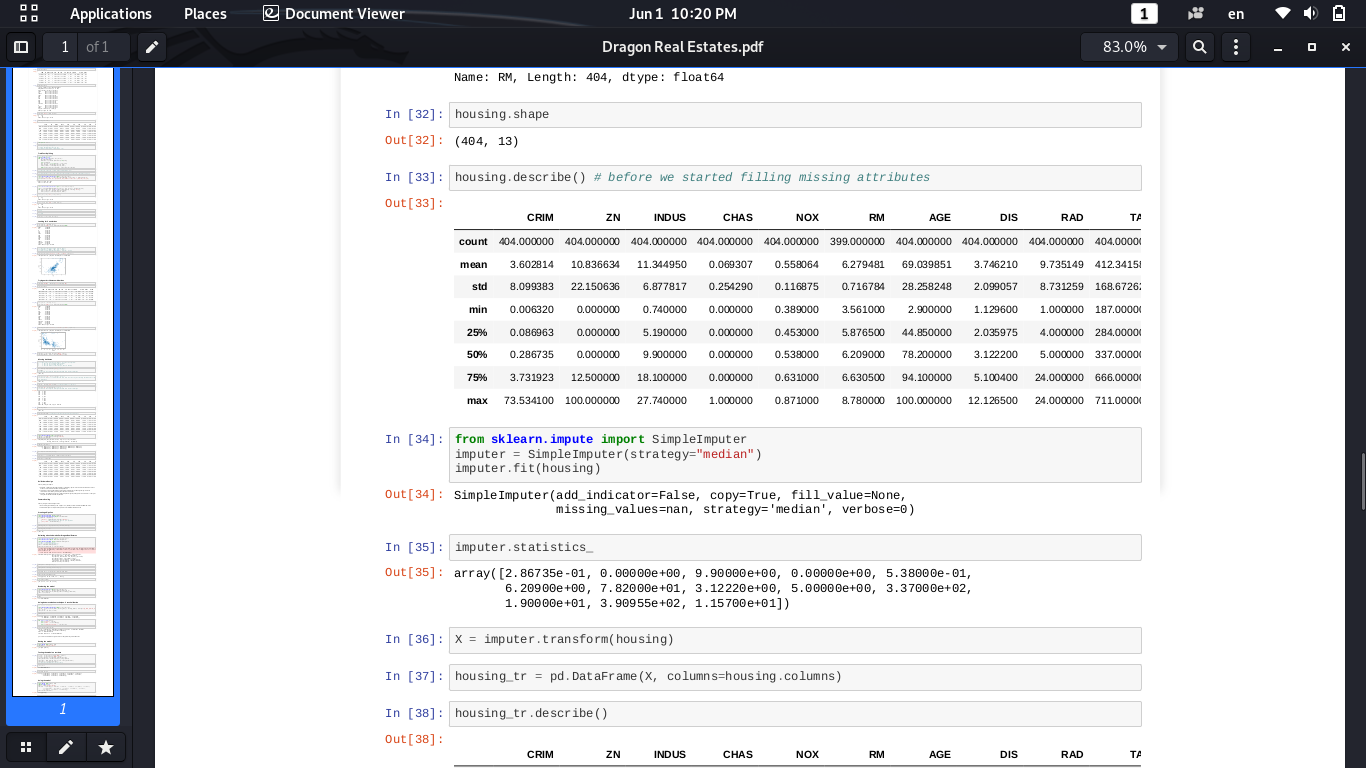


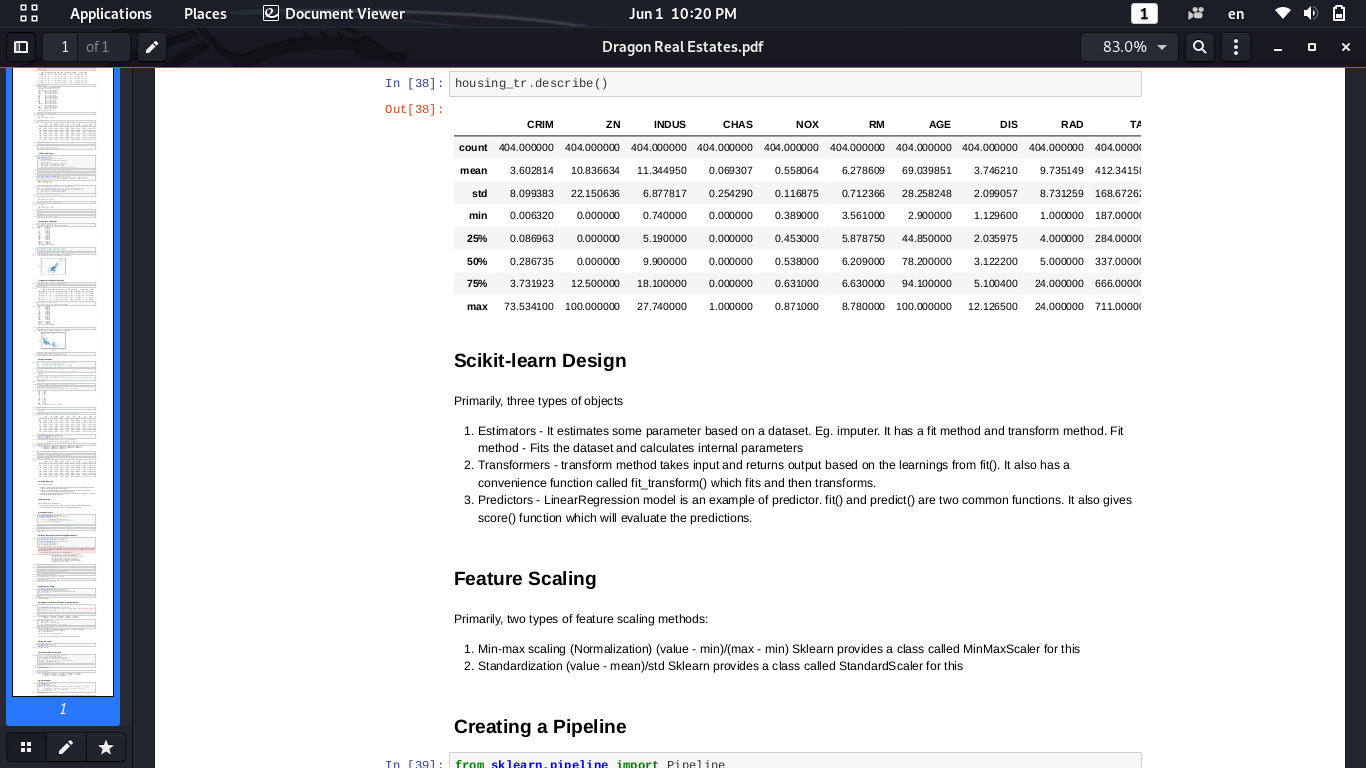


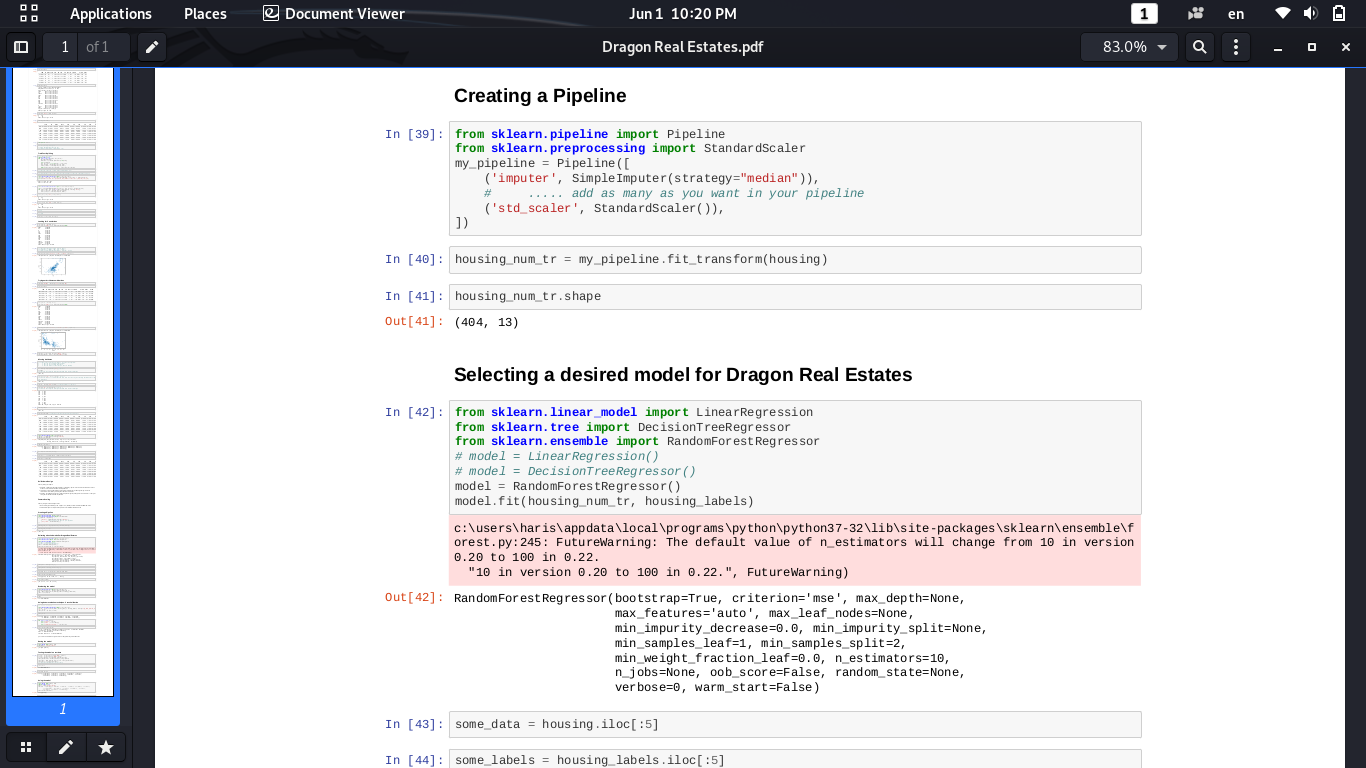


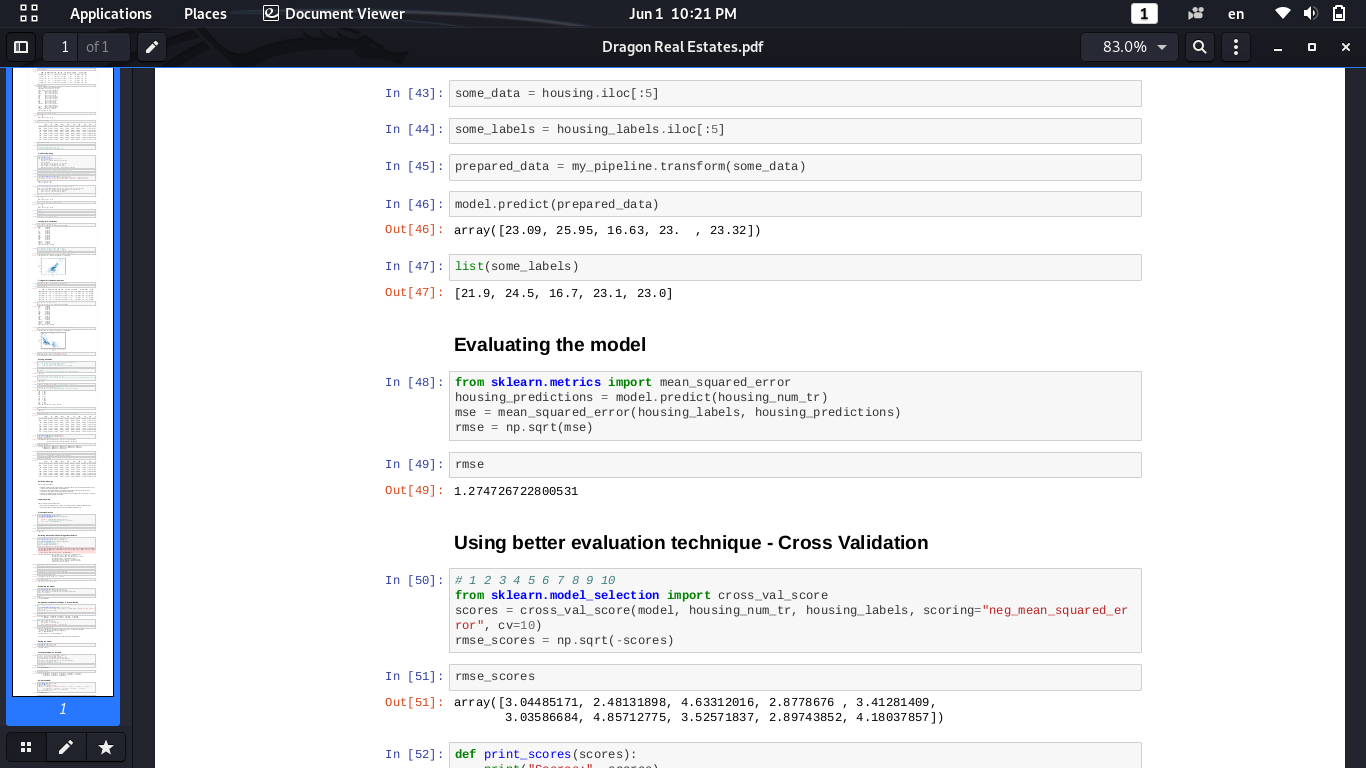


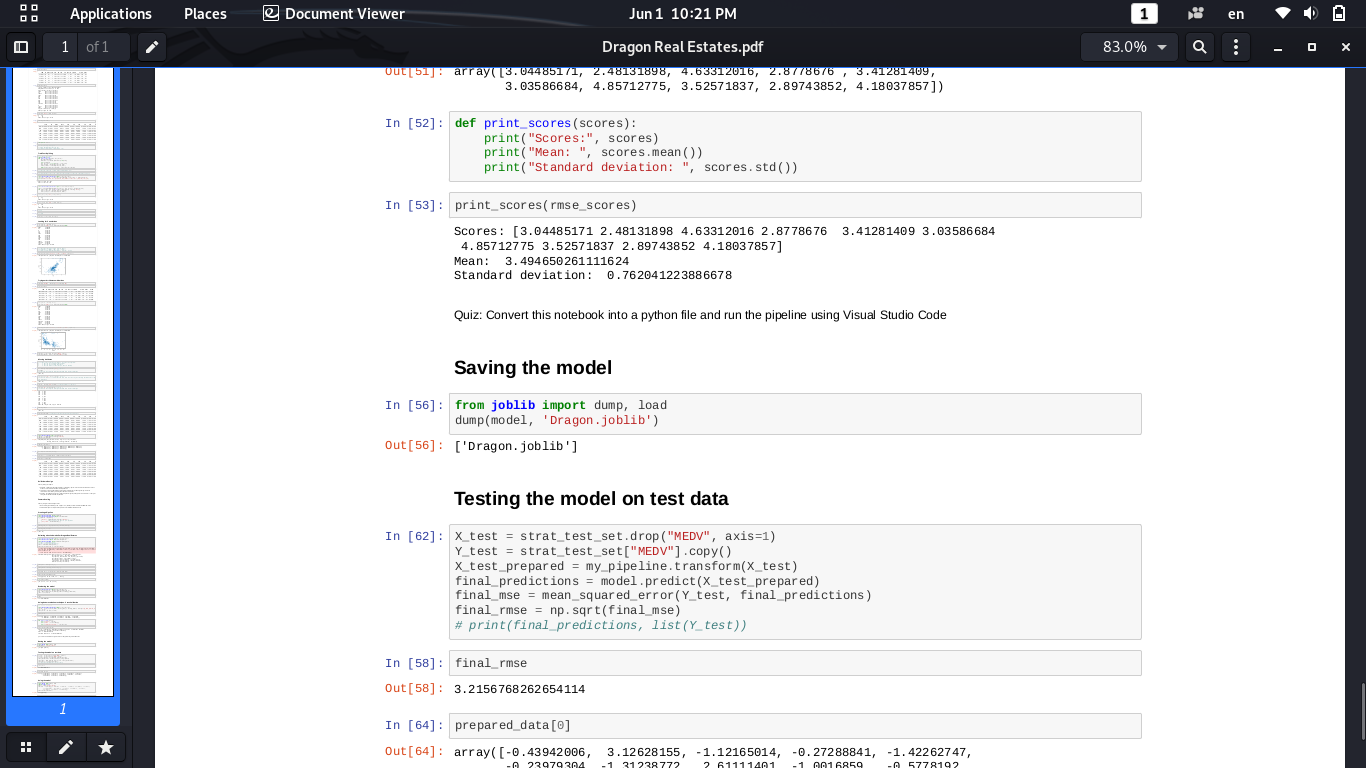


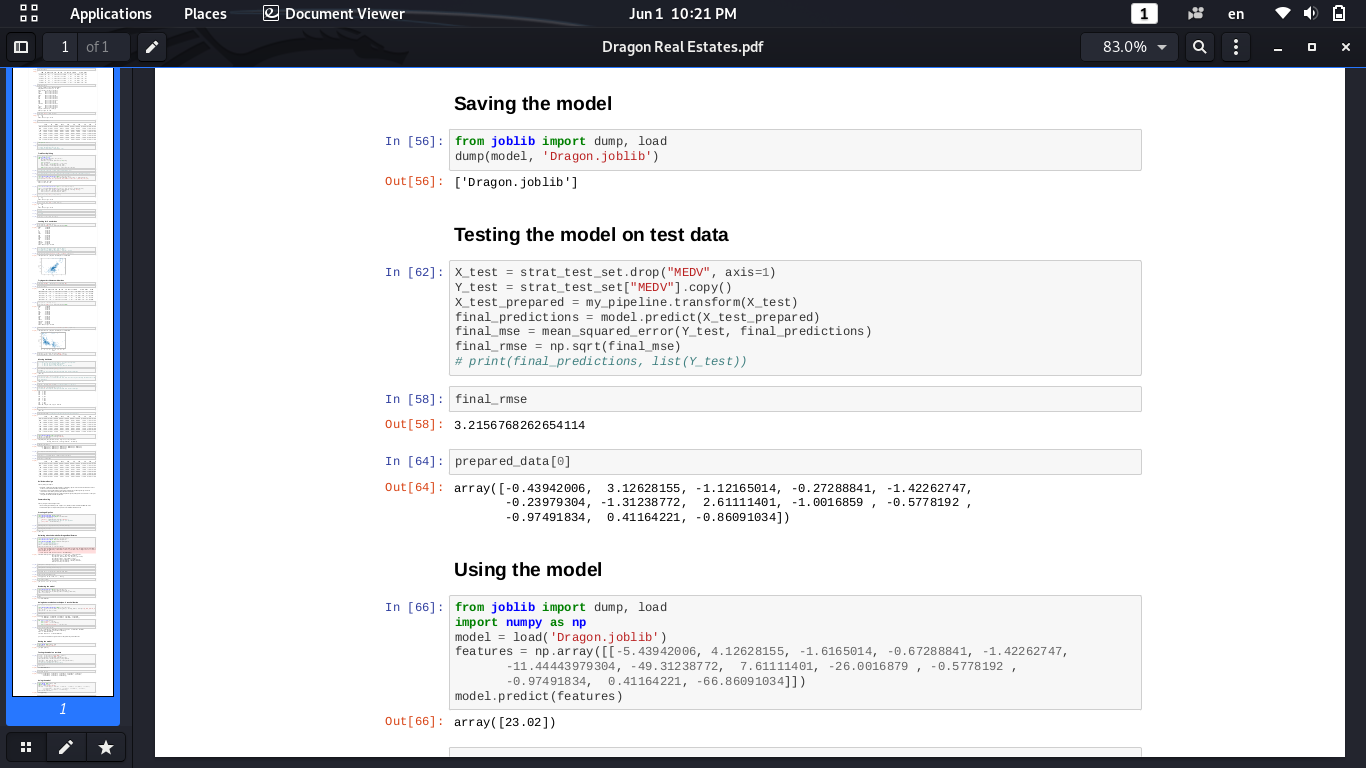












**OUTPUT:**



**Conclusion:**

In todays real estate world, it has become tough to store such huge data and extract them for ones own requirement.Also, the extracted data should be useful. The system makes optimal use of the Data mining Algorithm. The system makes use of such data in the most efficient way. The Data mining algorithm helps to fulfill customers by increasing the accuracy of estate choice and reducing the risk of investing in an estate. A lots of features that could be added to make the system more widely acceptable. One of the major future scopes is adding estate database of more cities which will provide the user to explore more estates and reach an accurate decision. More factors like recession that affect the house prices shall be added. In-depth details of every property will be added to provide ample details of a desired estate. This will help the system to run on a larger level. There are quite a few things that can be polished or add in the future work. • Though, we were able to identify most of the residential areas. There may be some more places that have

housing complexes or multi-storey apartments which are located in commercial areas. Such apartments were not included in this paper and can be counted in future to give a more accurate result. With more and more demand for housing in metropolitan cities, there is a definite increase in the number private builders that provide real estate with additional amenities to attract more customers. • There are several other models available that can be implemented for prediction. Data given as input to such model should be compatible with the tool used and the operators involved in the process. Also, more number of data sets can be used to increase the accuracy of the model. The main objective of using a different model should be to reduce the calculation time and carry out the whole process in ease.

**References:**

* 1. Natural Language Processing with Python – Edward Loper, Ewan Klein and Steven Bird.
  2. [www.python.org](http://www.python.org/)
  3. [www.spacy.io](http://www.spacy.io/)
  4. [www.nltk.org](http://www.nltk.org/)
  5. Software Engineering – Ian Sommerville.
  6. Anany Levitin, Introduction to The Design & Analysis of Algorithms, Pearson Education.
  7. IEEE SRS format.
  8. Michael Blaha, James Rumbaugh: Object-Oriented Modelling and Design with UML, Pearson Education, 2nd Edition.