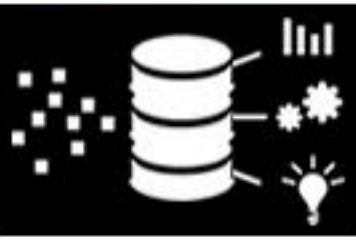
Implementing a Data Analytics Project with Machine Learning

A Hands-on Approach

Sulaimon Afolabi

Visualize Data



Process Data

Build Model

Measure Performance

"Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed".

Arthur Samuel (1959)

The Tools

python™

jupyter

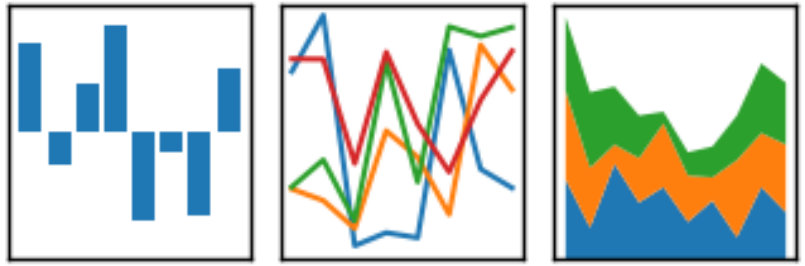
ANACONDA®

scikit learn

matplotlib

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Teaching Outline

1. Project Description, Resources & Checklist
2. Data Loading, Visualisation & Exploration
3. Feature Cleaning, Selection & Transformation
4. Machine Learning Algorithm Adoption
5. Model Performance Evaluation
6. Model Validation, Fine-Tuning & Ensembling

1

Project Description, Resources & Checklist

Project Description, Resources & Checklist

Description

To use machine learning techniques to predict median housing price at the district level.

1 Project Description, Resources & Checklist

Description

In the US, district is the smallest geographical unit for which the U.S. Census Bureau publishes sample data.

A district typically has a population of 600 to 3,000 people.

A district is also called a block group.

1 Project Description, Resources & Checklist

Resources

Ten Features

20640 Records

Labelled

1 Project Description, Resources & Checklist

Checklist

Checklist 1	Supervised or Unsupervised Learning
Checklist 2	Classification or Regression Task
Checklist 3	Identify the target feature
Checklist 4	Can I get extra data or feature?

1 Project Description, Resources & Checklist

Checklist

Define the objective in business terms.

How will your solution be used?

What are the current solutions (if any)?

How should performance be measured?

Is the performance measure aligned with the business objective?

What would be the minimum performance needed to reach the business objective?

What are comparable problems? Can you reuse experience or tools?

How would you solve the problem manually?

List the assumptions you (or others) have made so far.

Verify assumptions if possible.

2

Data Loading, Visualisation & Exploration

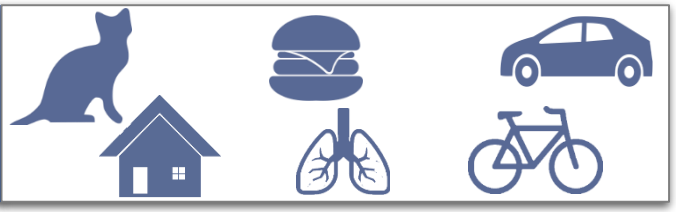
2 Data Loading, Visualisation & Exploration

Data Form

Alpha Numeric

1		Male	Female	
	2.0	2014-08-21		No
-5		Yes	$1,000	10

Image

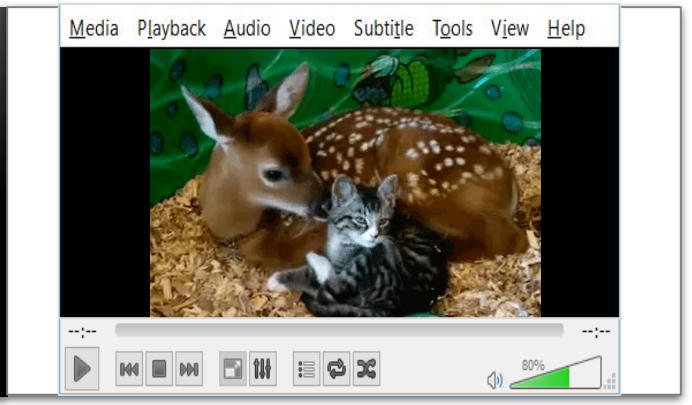


Text

If you cannot do great things
do small things in a great way.
This is a quote by Napoleon Hill.

Audio

Video



2 Data Loading, Visualisation & Exploration

Data Loading Checklist

Data Location

Where is the dataset located?

Computer | Server | Web | Cloud.

Data Form

The dataset is what form? Alpha-Numeric | Text | Image | Audio | Video.

Data Flow

Is it a real time data? Does it come as a stream or in batches?

Data Size

How big is the dataset? Is the size in kilo byte, megabyte, gigabyte or terabyte.

Analysis Platform

Can I analyse it on my computer or I need to engage the service of cloud based computing provider e.g. Microsoft Azure, Amazon web service (AWS), google cloud etc.

2 Data Loading, Visualisation & Exploration

Data Loading Steps

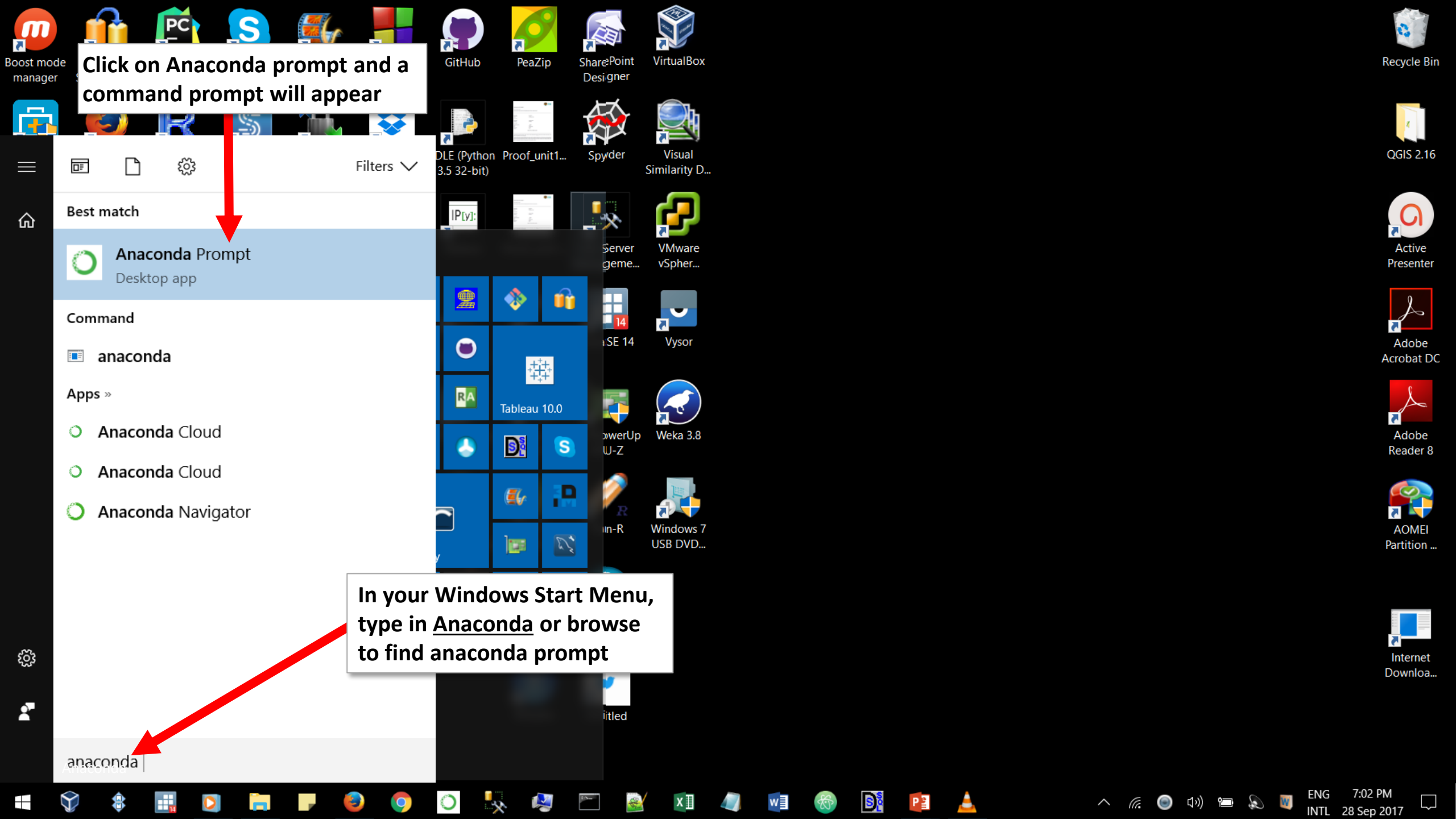
Step 1

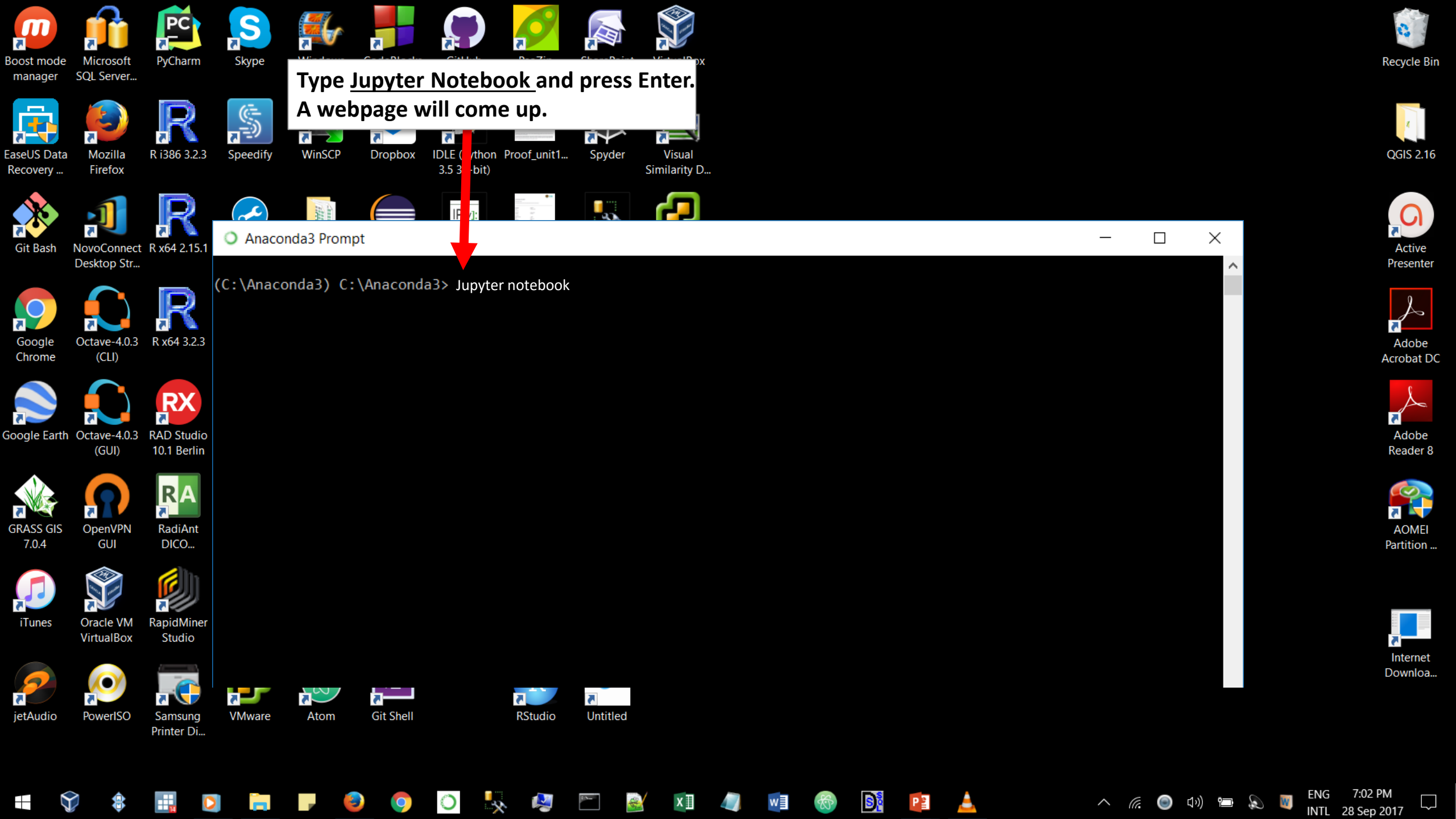
Start the Jupyter notebook or your

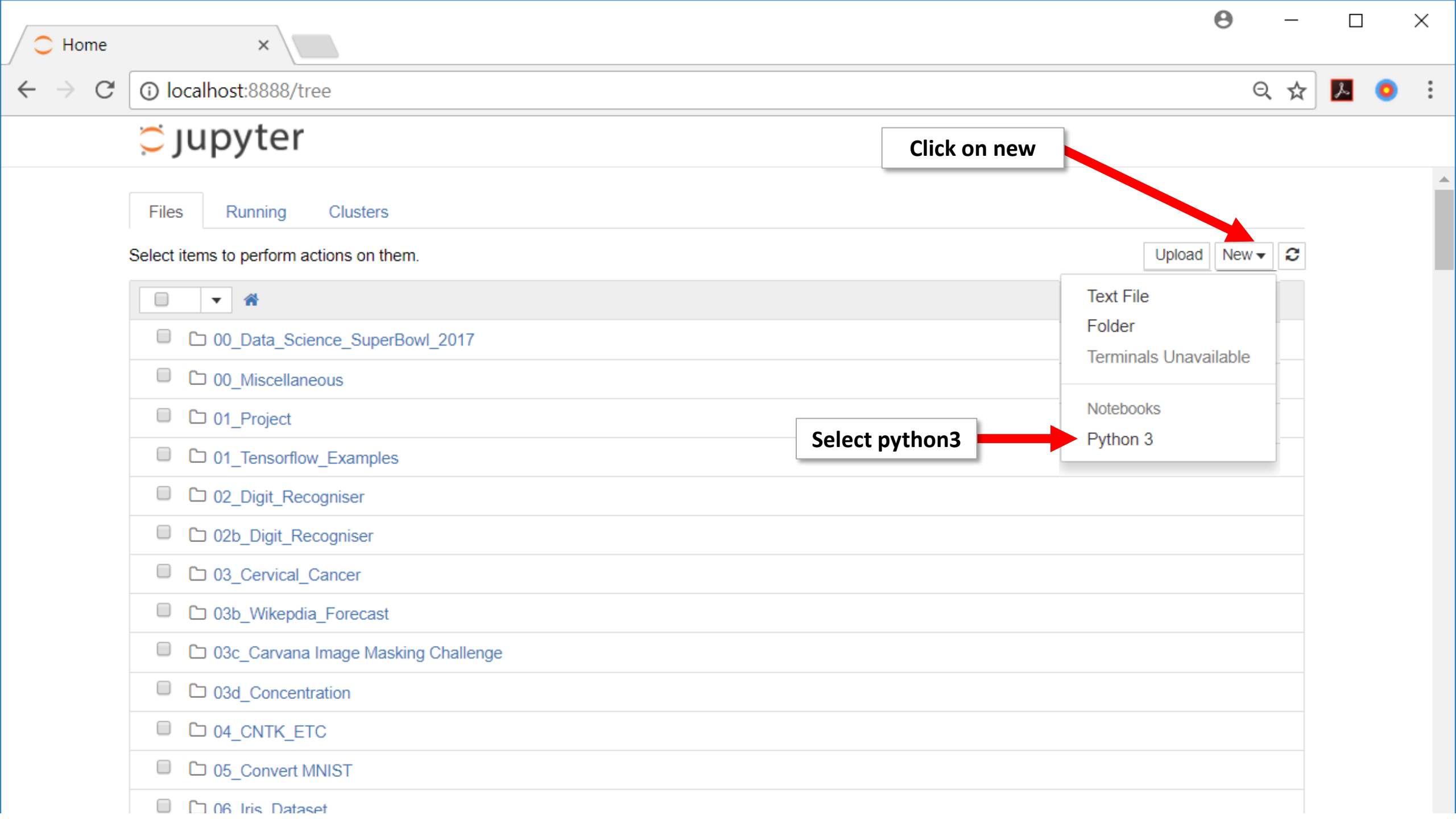
Start Menu → Anaconda Prompt → Type Jupyter Notebook→ a webpage will open

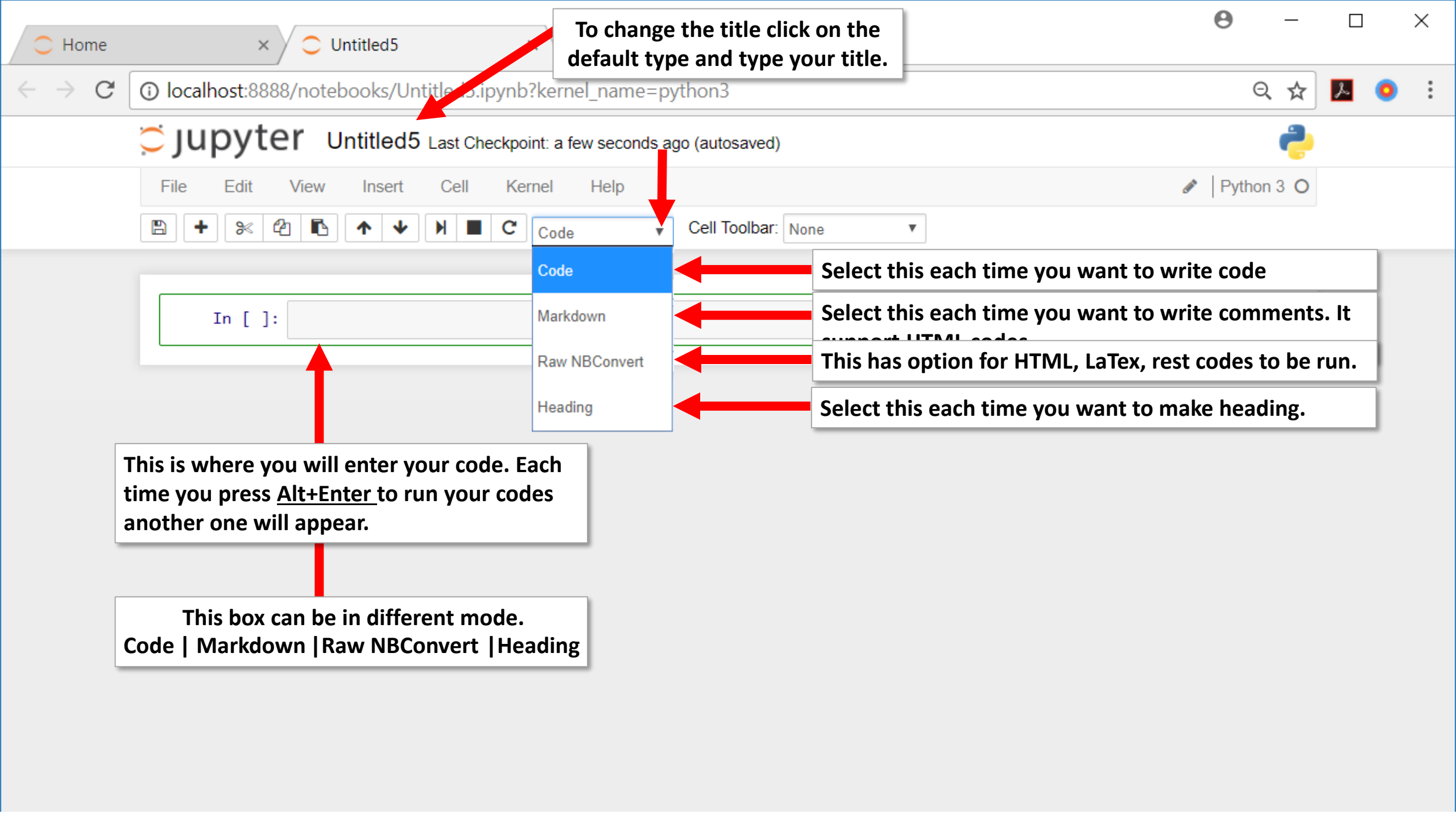
LET'S DEMONSTRATE THIS

It is assumed that you have already installed Anaconda


Click on Anaconda prompt and a command prompt will appear
Filters
Best match
Anaconda Prompt
Desktop app
Command
anaconda
Apps »
Anaconda Cloud
Anaconda Cloud
Anaconda Navigator
anaconda
In your Windows Start Menu, type in Anaconda or browse to find anaconda prompt
Tableau 10.0
Boost mode manager
GitHub
PeaZip
VirtualBox
Recycle Bin
QGIS 2.16
Spyder
Visual Similarity D...
VMware vSpher...
Active Presenter
Vysor
Adobe Acrobat DC
Weka 3.8
Adobe Reader 8
Windows 7 USB DVD...
AOMEI Partition ...
Internet Downloa...
ENG
INTL
7:02 PM
28 Sep 2017


Type Jupyter Notebook and press Enter.
A webpage will come up.
Anaconda3 Prompt
(C:\Anaconda3) C:\Anaconda3> Jupyter notebook
Boost mode manager
Microsoft SQL Server...
PyCharm
Skype
EaseUS Data Recovery ...
Mozilla Firefox
R i386 3.2.3
Speedify
WinSCP
Dropbox
Spyder
Visual Similarity D...
Git Bash
NovoConnect Desktop Str...
R x64 2.15.1
Google Chrome
Octave-4.0.3 (CLI)
R x64 3.2.3
Google Earth
Octave-4.0.3 (GUI)
RAD Studio 10.1 Berlin
GRASS GIS 7.0.4
OpenVPN GUI
RadiAnt DICO...
iTunes
Oracle VM VirtualBox
RapidMiner Studio
jetAudio
PowerISO
Samsung Printer Di...
VMware
Atom
Git Shell
RStudio
Untitled
Recycle Bin
QGIS 2.16
Active Presenter
Adobe Acrobat DC
Adobe Reader 8
AOMEI Partition ...
Internet Downloa...
ENG
INTL
7:02 PM
28 Sep 2017


Home
localhost:8888/tree
jupyter
Click on new
Files
Running
Clusters
Select items to perform actions on them.
Upload
New
Text File
Folder
Terminals Unavailable
Notebooks
Python 3
Select python3
00_Data_Science_SuperBowl_2017
00_Miscellaneous
01_Project
01_Tensorflow_Examples
02_Digit_Recogniser
02b_Digit_Recogniser
03_Cervical_Cancer
03b_Wikepdia_Forecast
03c_Carvana Image Masking Challenge
03d_Concentration
04_CNTK_ETC
05_Convert MNIST
06_Iris_Dataset


Home
Untitled5
To change the title click on the default type and type your title.
localhost:8888/notebooks/Untitled5.ipynb?kernel_name=python3
jupyter
Untitled5
Last Checkpoint: a few seconds ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Help
Python 3
Code
Cell Toolbar:
None
Code
Markdown
Raw NBConvert
Heading
Select this each time you want to write code
Select this each time you want to write comments. It
This has option for HTML, LaTex, rest codes to be run.
Select this each time you want to make heading.
In [ ]:
This is where you will enter your code. Each time you press Alt+Enter to run your codes another one will appear.
This box can be in different mode.
Code | Markdown |Raw NBConvert |Heading

2 Data Loading, Visualisation & Exploration

Data Loading Steps

Step 1

Start the Jupyter notebook or your

Start Menu → Anaconda Prompt → Type Jupyter Notebook → a webpage will open

Step 2

Import the python module for checking & changing your directory

```
import os
        os.getcwd()
                os.chdir('C:/Anaconda3')
```

LET'S SEE THE CODE ON JUPYTER NOTEBOOK

jupyter MyProject Last Checkpoint: 2 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

Import the python module for checking the current directory

In [3]:
```
import os
```

Checking the current working directory

In [11]:
```
os.getcwd()
```

Out[11]:
```
'C:\\Anaconda3\\01_Project'
```

Changing the current working directory

In [13]:
```
os.chdir('C:/Anaconda3')
```

In [16]:
```
# THis is an equivalent of the code up there
os.chdir(r'C:\Anaconda3')
```

Checking the directory again

In [14]:
```
os.getcwd()
```

Out[14]:
```
'C:\\Anaconda3'
```

2 Data Loading, Visualisation & Exploration

Data Loading Steps

Step 1

Start the Jupyter notebook or your

Start Menu → Anaconda Prompt → Type Jupyter Notebook → a webpage will open

Step 2

Import the python module for checking & changing your directory

```
import os
        os.getcwd()
                os.chdir('C:/Anaconda3')
```

Step 3

Import the python module for loading data i.e. pandas

```
import pandas as pd
```

jupyter MyProject-Copy1 Last Checkpoint: 15 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help Python 3

Markdown Cell Toolbar: None

Loading dataset using Pandas

Importing the python module for loading data

In [3]:
```
import pandas as pd
```

#Note : If pandas is not already installed, do the following (internet is needed)

- Go to Anaconda prompt (already opened)
- Pres CTRL+C to stop Jupyter Notebook (Do not close the Anaconda Prompt)
- To install pandas type conda install pandas
- When prompted, type y for yes
- Restart jupyter notebook by typing - Jupyter Notebook

2 Data Loading, Visualisation & Exploration

Data Loading Steps

Step 1

Start the Jupyter notebook or your

Start Menu → Anaconda Prompt → Type Jupyter Notebook→ a webpage will open

Step 2

Import the python module for checking & changing your directory

```
import os
        os.getcwd()
                os.chdir('C:/Anaconda3')
```

```
pd.read_clipboard
pd.read_csv
pd.read_excel
pd.read_feather
pd.read_fwf
pd.read_gbq
pd.read_hdf
pd.read_html
pd.read_json
pd.read_msgpack
pd.read_pickle
pd.read_sas
pd.read_sql
pd.read_sql_query
pd.read_sql_table
pd.read_stata
pd.read_table
```

The other kind of data Format that you can load

Step 3

Import the python module for loading das

```
import pandas as pd
```

Step 4

Load the data

```
Dataset = pd.read_csv('C:/MyDataset.csv')
```

jupyter MyProject-Copy1 Last Checkpoint: 25 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help Python 3

Markdown Cell Toolbar: None

Loading dataset using Pandas

Importing the python module for loading data

In [3]:
```
import pandas as pd
```

#Note : If pandas is not already installed, do the following (internet is needed)

- Go to Anaconda prompt (already opened)
- Pres CTRL+C to stop Jupyter Notebook (Do not close the Anaconda Prompt)
- To install pandas type conda install pandas
- When prompted, type y for yes
- Restart jupyter notebook by typing - Jupyter Notebook

In [ ]:
```
MyDataset = pd.read_csv('C/Anaconda/MyData.csv')
```

That is the folder where you put your dataset.
Note the direction of the slash (/)
If you want it like (\), type r's
r'C\Anaconda\MyData.csv'

```
Note:
- pd is the alias for pd
- MyDaset is arbitrary
```

2 Data Loading, Visualisation & Exploration

Data Loading Steps

Step 1

Start the Jupyter notebook or your

Start Menu → Anaconda Prompt → Type Jupyter Notebook → Webpage will open

Step 2

Import the python module for checking & changing your directory

```
import os
        os.getcwd()
                os.chdir('C:/Anaconda3')
```

Step 3

Import the python module for loading data i.e. pandas

```
import pandas as pd
```

Step 4

Load the data

```
Dataset = pd.read_csv('C:/MyDataset.csv')
```

2 Data Loading, Visualisation & Exploration

Data Visualisation

View the raw data

Show the variable properties

Handle the categorical variables

Display the descriptive statistics

Draw the Box and whisker plot

Plot histograms

Plot the GIS Coordinates /Map

Viewing the raw data and the variable properties

File Edit View Insert Cell Kernel Help Python 3

Code Cell Toolbar: None

Index column- Auto-generated by python and it always start from Zero

Viewing the Dataset

Numeric data also serves as GIS coordinates

Numeric data

Categorical variable

```
In [11]: MyDataset.head()
```

Out[11]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR I
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR I
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR I
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR I
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR I

Columns

Labelled

Viewing the attributes of the variables

Total number of records

```
In [12]: MyDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

Case of missing Data

Numbers

String (Categorical Data)

Display the descriptive statistics

Note:

```
I do not want the GIS coordinates (latitude and lungitude) and the categorical variables (ocean_proximity. <br>
This can be done by excluding or filtering them out. See the code below
```

In [73]:
```
MyDataset_Reduced0 = MyDataset[MyDataset.columns.difference(['longitude', 'latitude','ocean_proximity'])]
```

In [74]:
```
MyDataset_Reduced0.describe()
```

Out[74]:

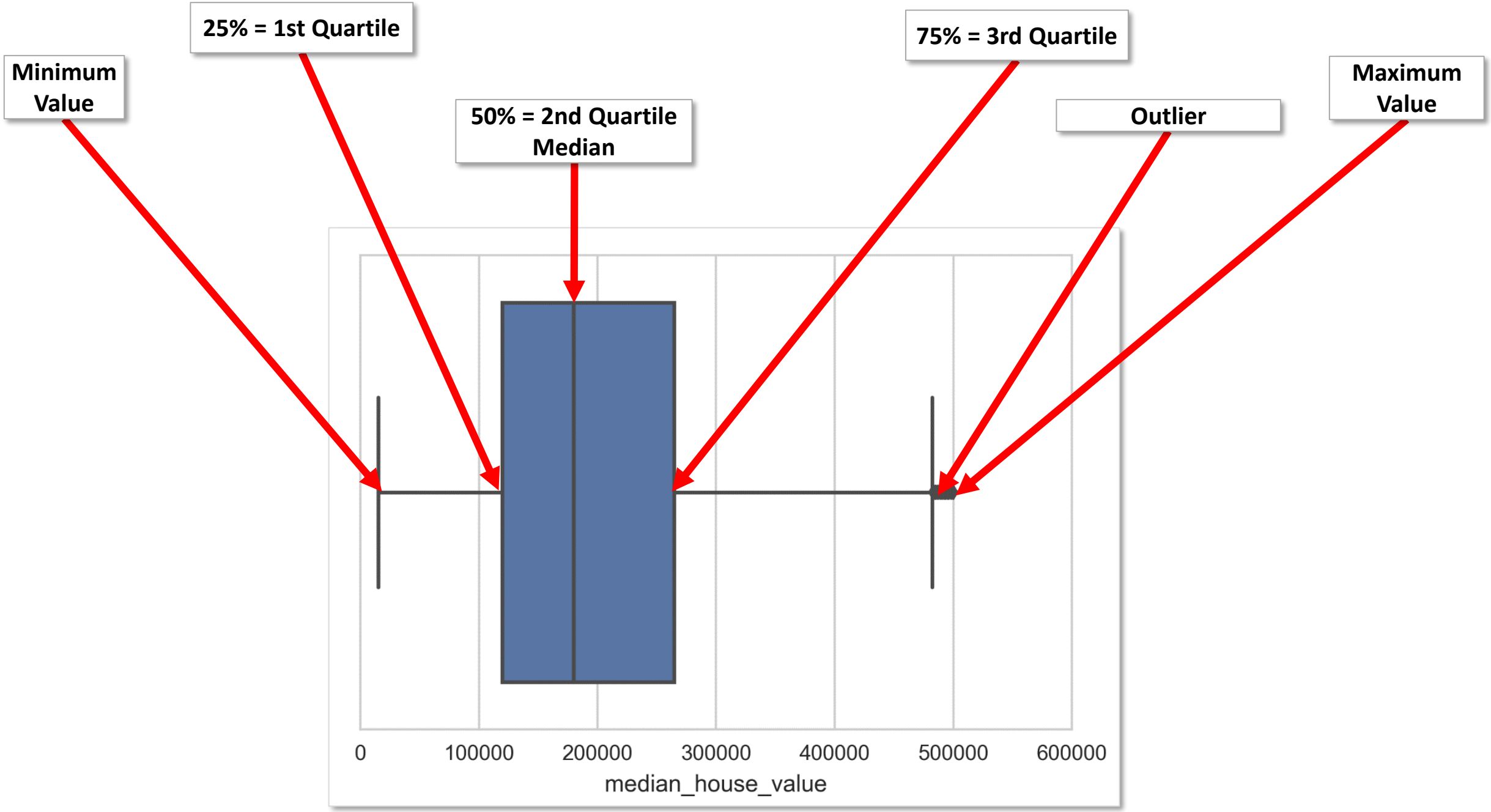
	households	housing_median_age	median_house_value	median_income	population	total_bedrooms	total_rooms
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000
mean	499.539680	28.639486	206855.816909	3.870671	1425.476744	537.870553	2635.763081
std	382.329753	12.585558	115395.615874	1.899822	1132.462122	421.385070	2181.615252
min	1.000000	1.000000	14999.000000	0.499900	3.000000	1.000000	2.000000
25%	280.000000	18.000000	119600.000000	2.563400	787.000000	296.000000	1447.750000
50%	409.000000	29.000000	179700.000000	3.534800	1166.000000	435.000000	2127.000000
75%	605.000000	37.000000	264725.000000	4.743250	1725.000000	647.000000	3148.000000
max	6082.000000	52.000000	500001.000000	15.000100	35682.000000	6445.000000	39320.000000

Count = Frequency
Mean= Average
Std = Standard Deviation

Min = Minimum value
Max = Maximum value

25%= 1st Quartile
50%= 2nd Quartile = Median
75%= 3rd Quartile

Drawing the Box and Whisker Plot



Drawing the Box and Whisker Plot

total_bedrooms

In [7]:
```
#Loading the necessary module
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

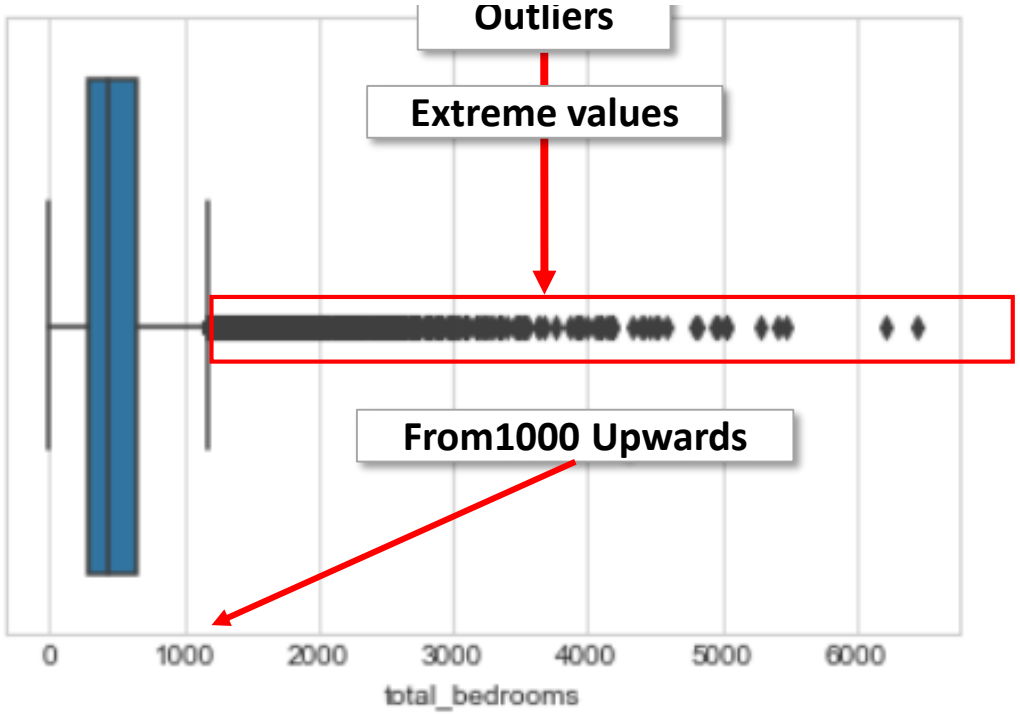
Python Modules

Needed for the plot to display in the notebook

In [8]:
```
sns.set_style("whitegrid")
ax = sns.boxplot(x = MyDataset['total_bedrooms'] )
#Saving the graph
plt.savefig("total_bedrooms_Plot1.png", format='png', dpi=300)
```

The feature of interest

Check your directory for the image with the specified name

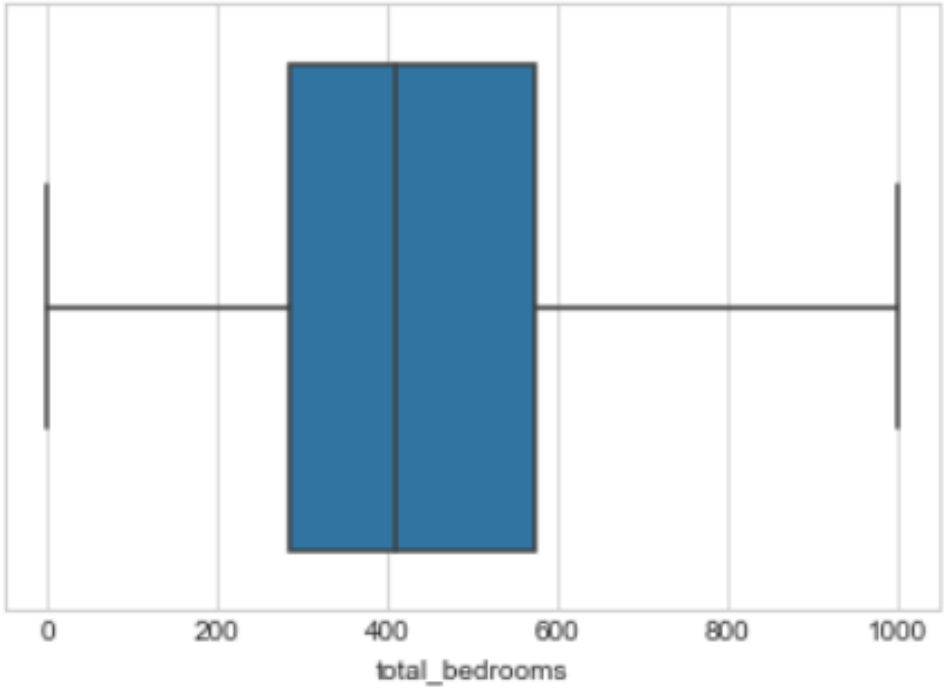


Drawing the Box and Whisker Plot

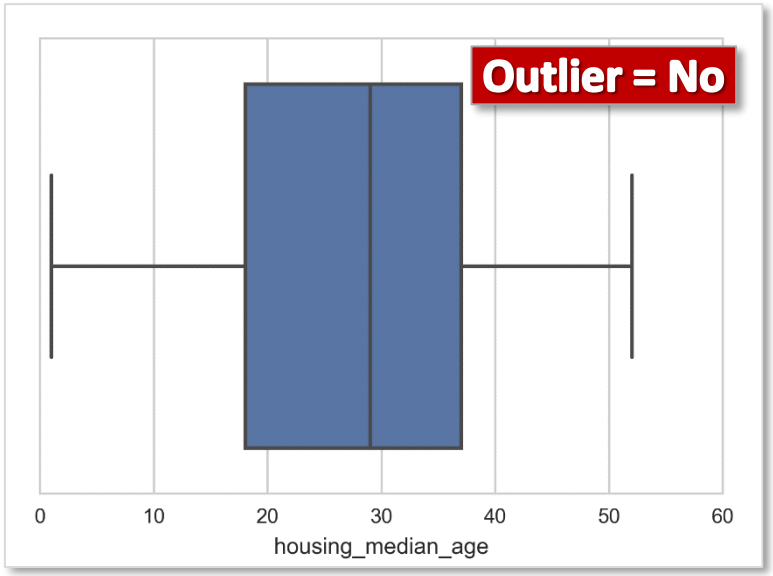
Let us limit the data to less than 1000 and re-draw the box & whisker plot

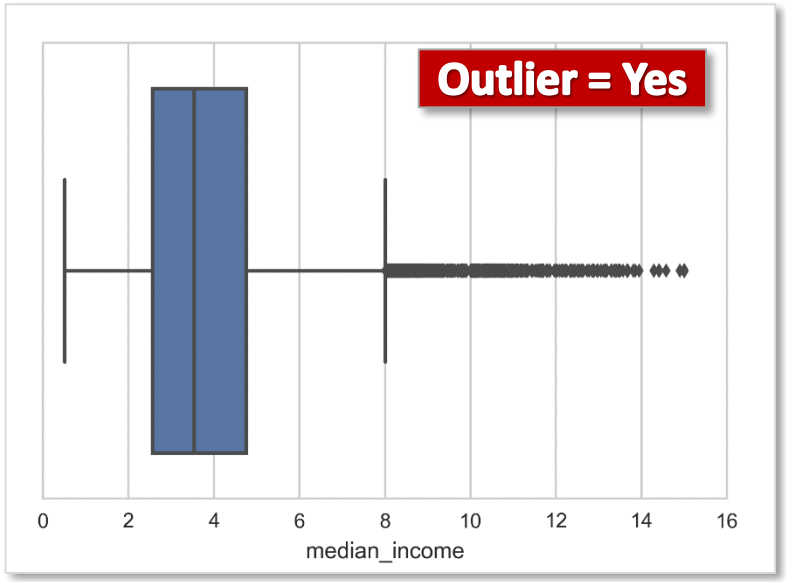
total_bedromms -Less than 1000

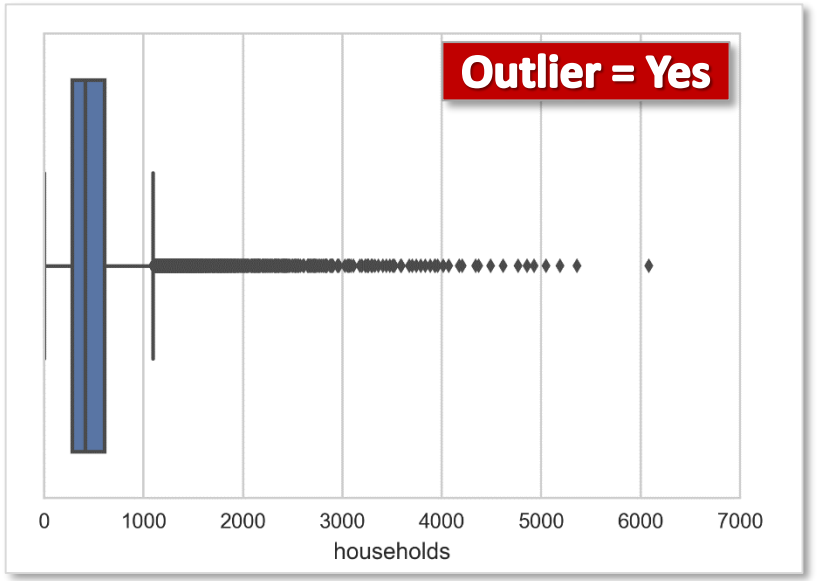
In [14]:
```
total_bedrooms_Filtered = MyDataset[MyDataset['total_bedrooms']<1000]
ax = sns.boxplot(x = total_bedrooms_Filtered['total_bedrooms'] )
plt.savefig("total_bedrooms_Plot2.png", format='png', dpi=300)
```

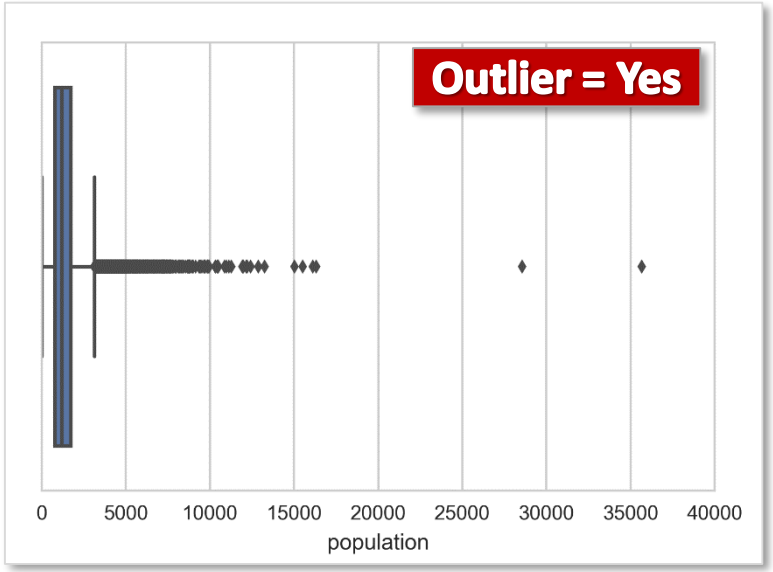


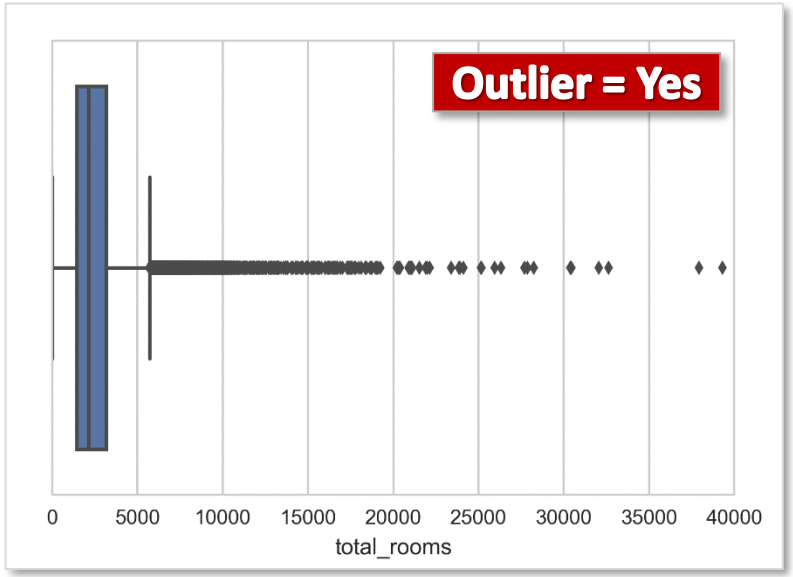
Plotting the Box and Whisker

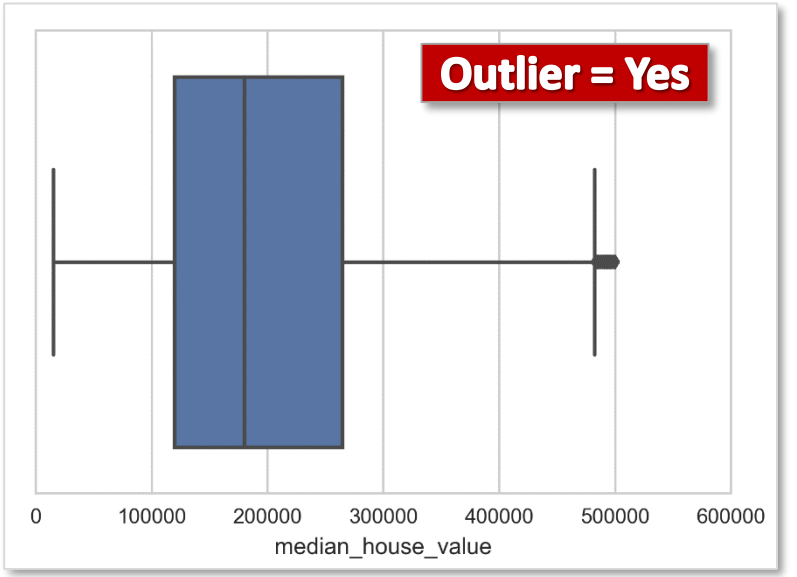


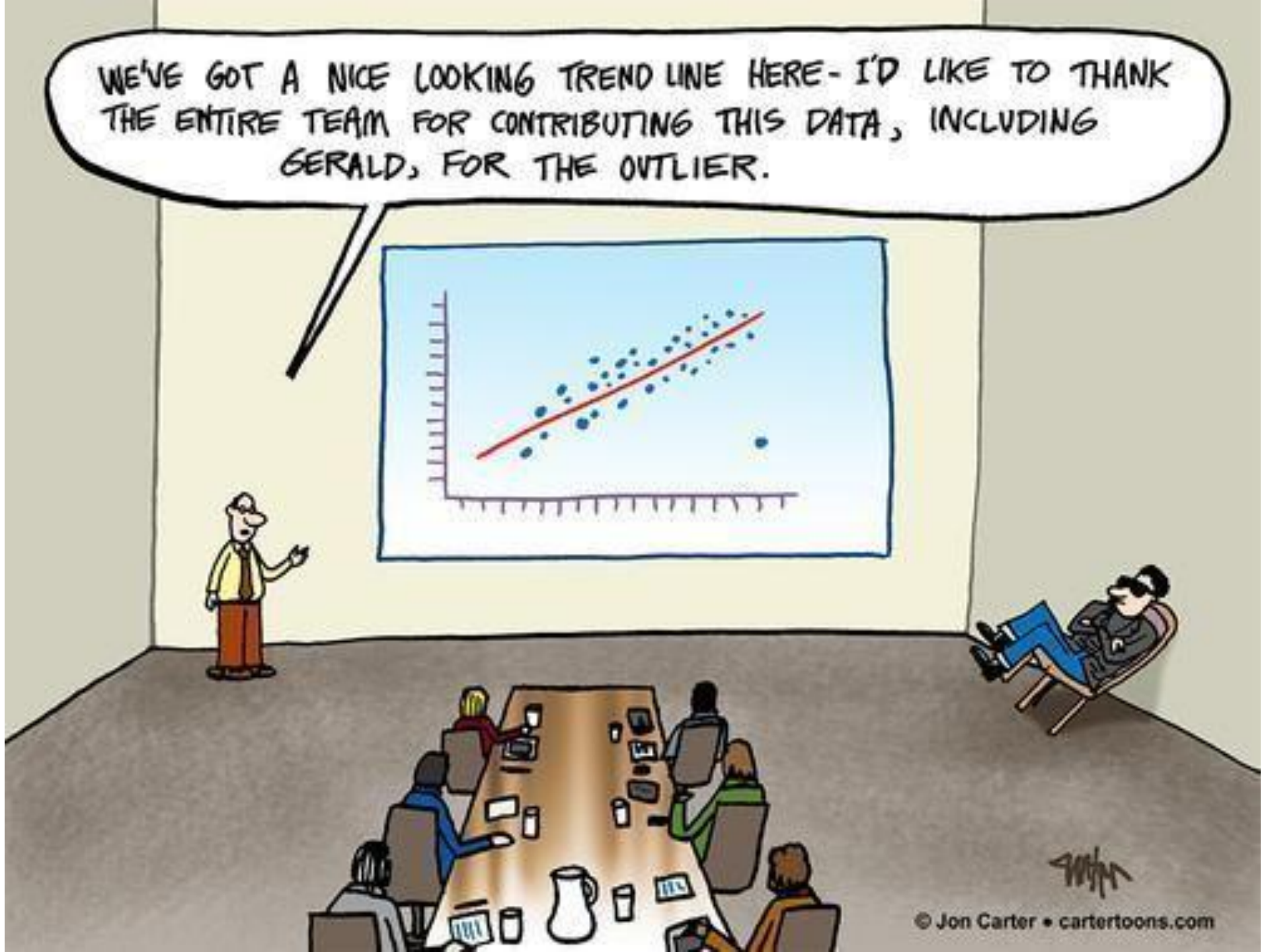











WE'VE GOT A NICE LOOKING TREND LINE HERE- I'D LIKE TO THANK THE ENTIRE TEAM FOR CONTRIBUTING THIS DATA, INCLUDING GERALD, FOR THE OUTLIER.
© Jon Carter • cartertoons.com

Without Outlier	With Outlier
4, 4, 5, 5, 5, 5, 6, 6, 6, 7, 7	4, 4, 5, 5, 5, 5, 6, 6, 6, 7, 7,300
Mean = 5.45	Mean = 30.00
Median = 5.00	Median = 5.50
Mode = 5.00	Mode = 5.00
Standard Deviation = 1.04	Standard Deviation = 85.03

Outliers

Causes of Outlier

Natural Cause: A legitimate but abnormal data point.

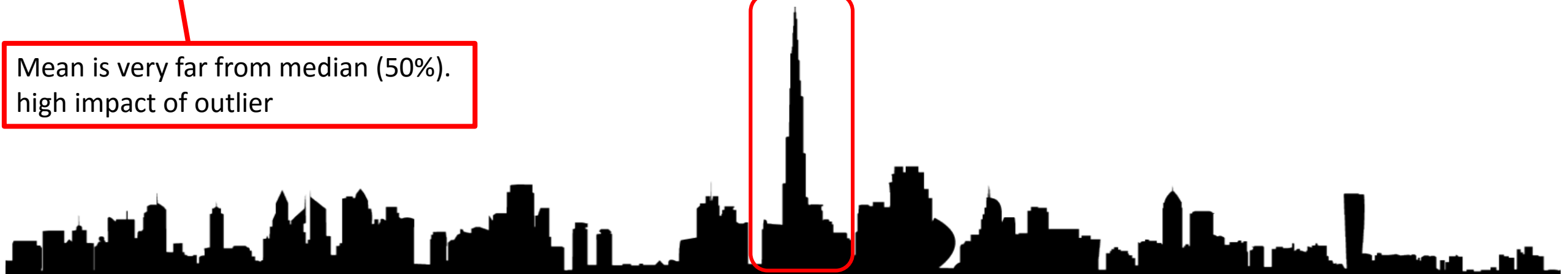
Human Factor: An error committed during manual data entry.

Instrument Failure: Due to momentary malfunctioning of a data collection device.

	households	housing_median_age	median_house_value	median_income	population	total_bedrooms	total_rooms
mean	499.539680	28.639486	206855.816909	3.870671	1425.476744	537.870553	2635.763081
50%	409.000000	29.000000	179700.000000	3.534800	1166.000000	435.000000	2127.000000
std	382.329753	12.585558	115395.615874	1.899822	1132.462122	421.385070	2181.615252

The Mean is very close to themedian (50%).
No impact of outlier

Mean is very far from median (50%).
high impact of outlier



Outliers

Treating Outliers

Feature Transformation: We can log-transform the data.

Delete Records: Deleting the corresponding records in the other feature.

Feature Removal : You can remove the outlier laden feature.

Value Imputation: You can replace the outlier value with the median value of the feature.

Plotting histograms

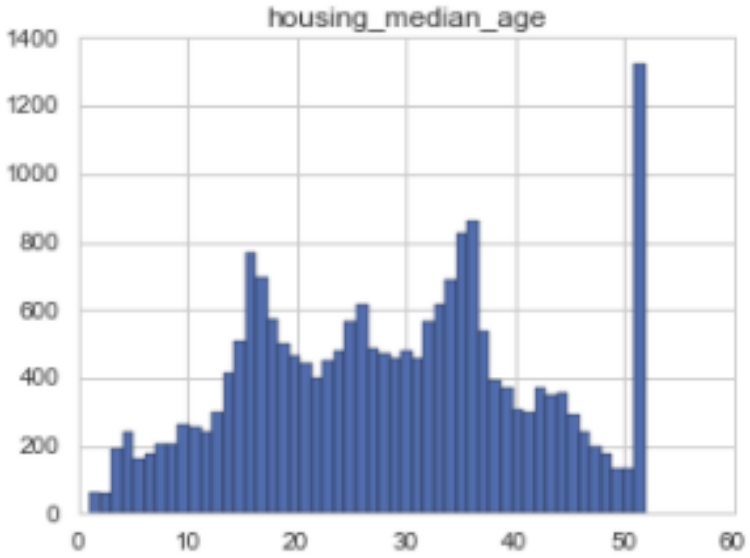
Selecting the columns of interest

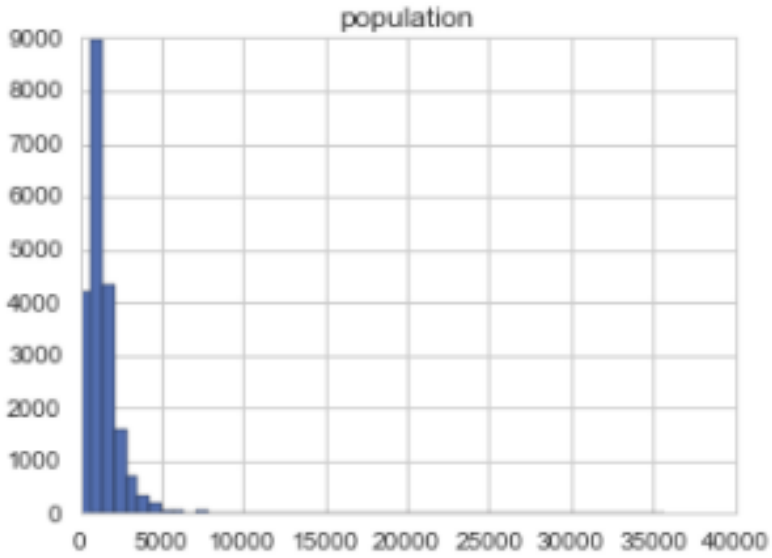
In [30]:
```
MyDataset_Reduced1 = MyDataset[['housing_median_age','total_rooms','total_bedrooms','population'
```

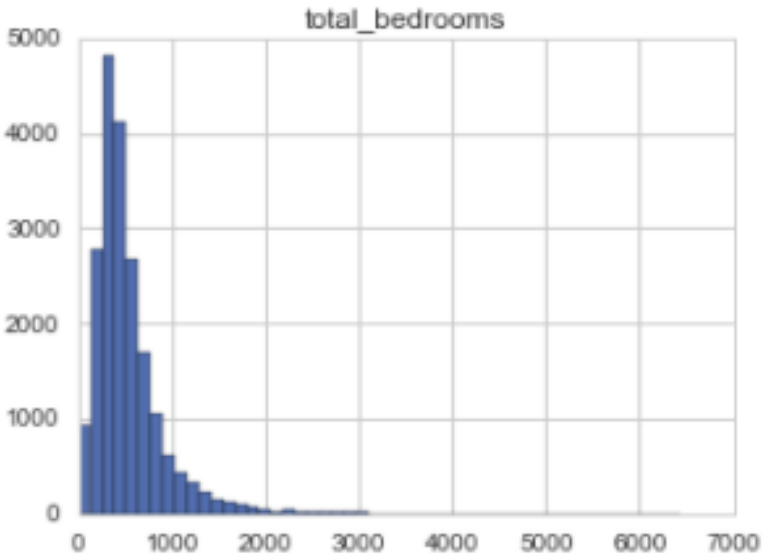
Note: You need a module called seaborn, matplotlib and %matplotlib inline (to make it appear on in the notebook)

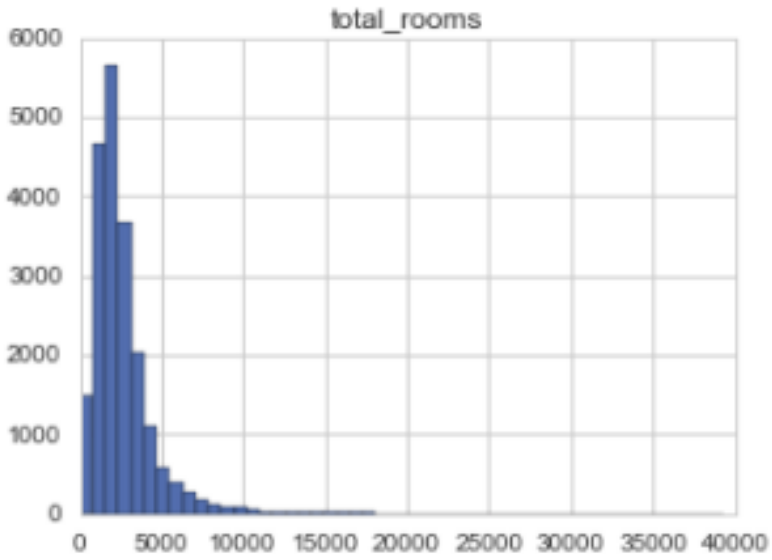
In [31]:
```
MyDataset_Reduced1.hist(bins=50, figsize=(11,8))
plt.show()
#Note:Saving the histogram in your current directory or folder
plt.savefig("HistogramPlot1.png", format='png', dpi=300)
#Note: Check your current directory for the image
```

Plotting the histogram



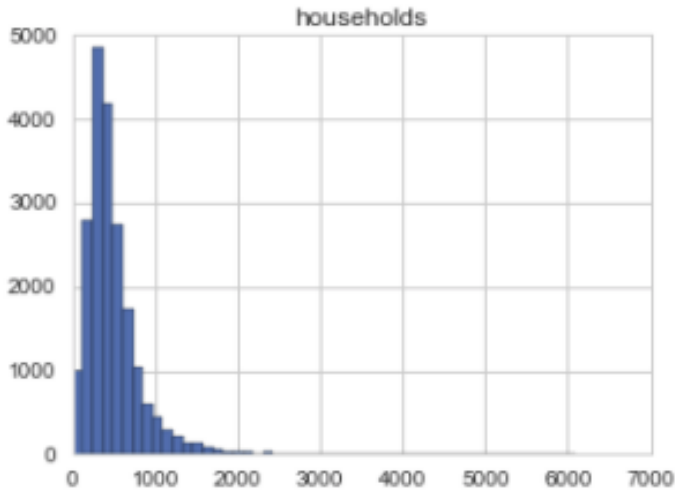


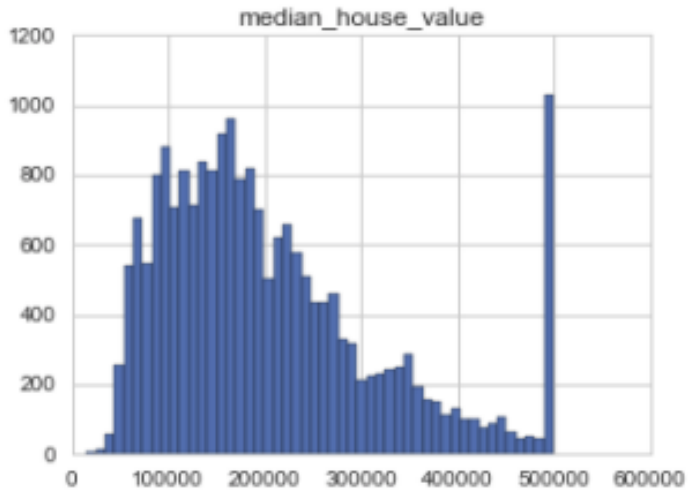


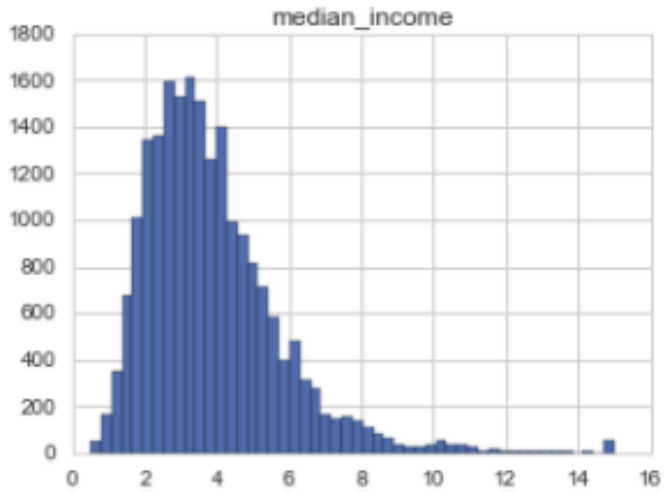


Plotting histograms

In [58]:
```
MyDataset_Reduced2 = MyDataset[['households','median_income','median_house_value']]
MyDataset_Reduced2.hist(bins=50, figsize=(11,8))
plt.show()
plt.savefig("HistogramPlot2.png", format='png', dpi=300)
```



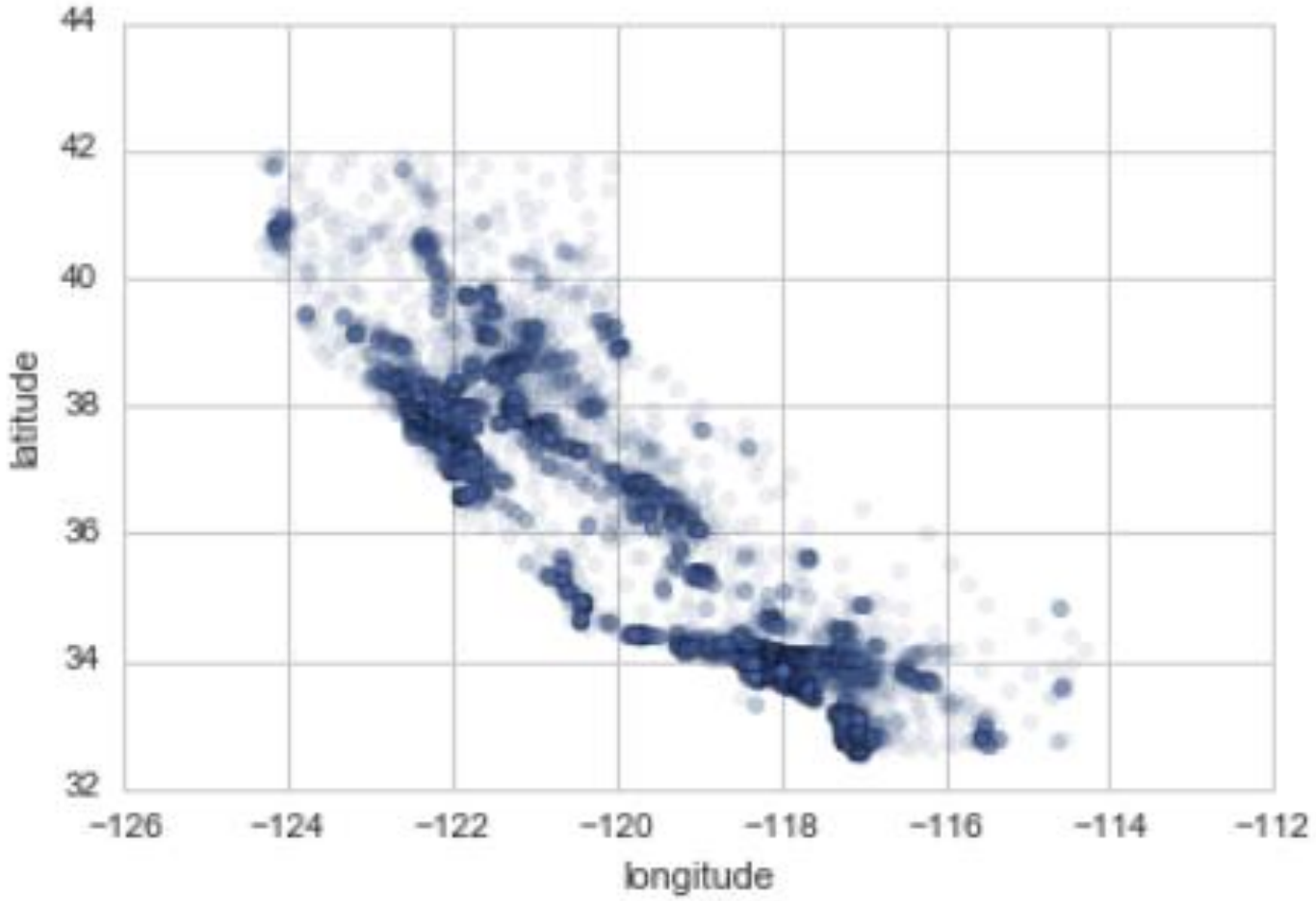




Plotting the GIS Coordinates

Plot 1

```
In [34]: MyDataset.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
         plt.savefig("Longitude_Latitude_Plot1",format='png', dpi=300)
```

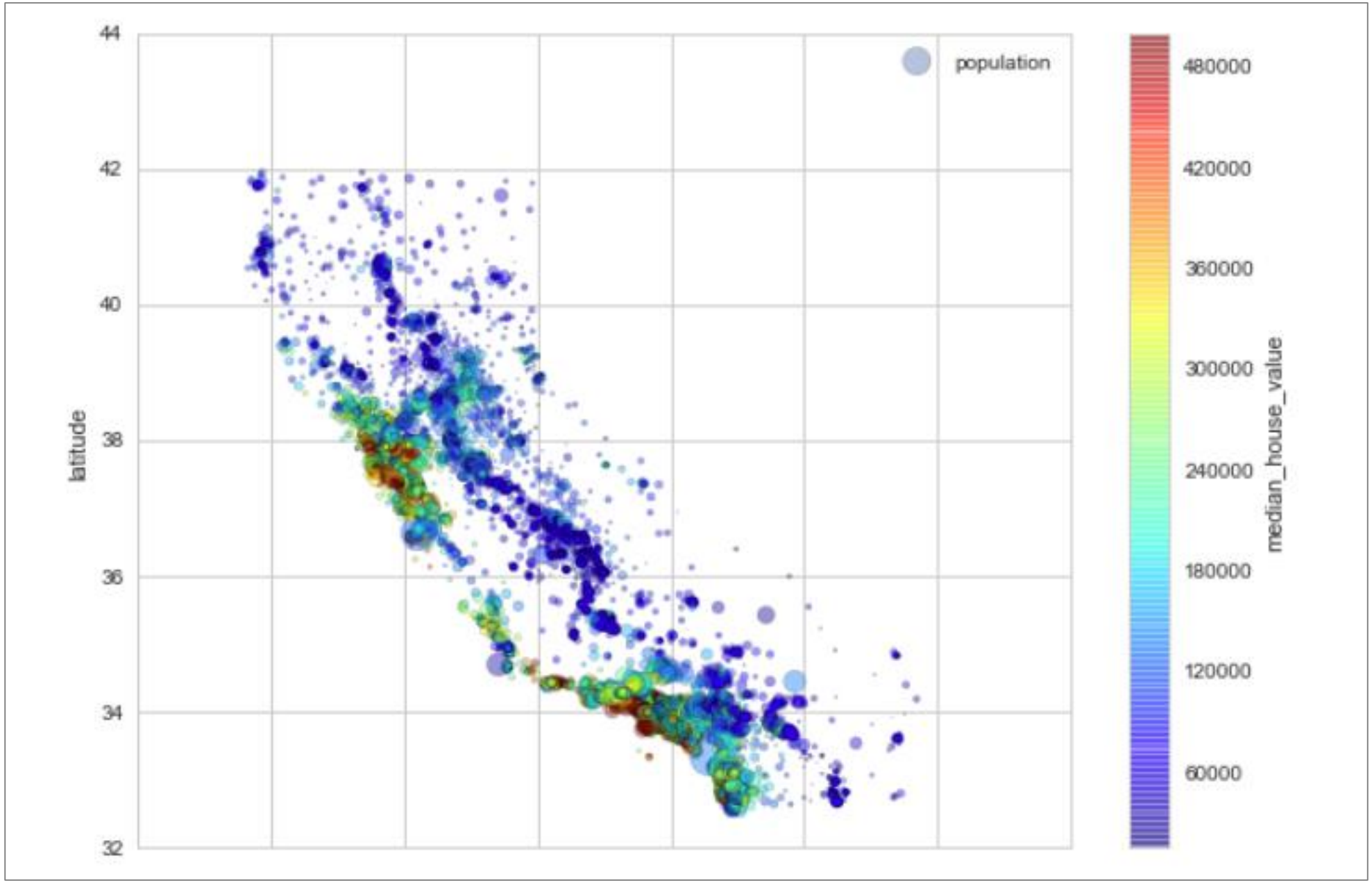


Plotting the GIS Coordinates

Plot 2

```
MyDataset.plot(kind="scatter", x="longitude", y="latitude",
    s=MyDataset['population']/100, label="population",
    c="median_house_value", cmap=plt.get_cmap("jet"),
    colorbar=True, alpha=0.4, figsize=(10,7),
)
plt.legend()
plt.savefig("Longitude_Latitude_Plot2")
plt.show()
```

Plotting the GIS Coordinates



Plotting the GIS Coordinates

Plot 3

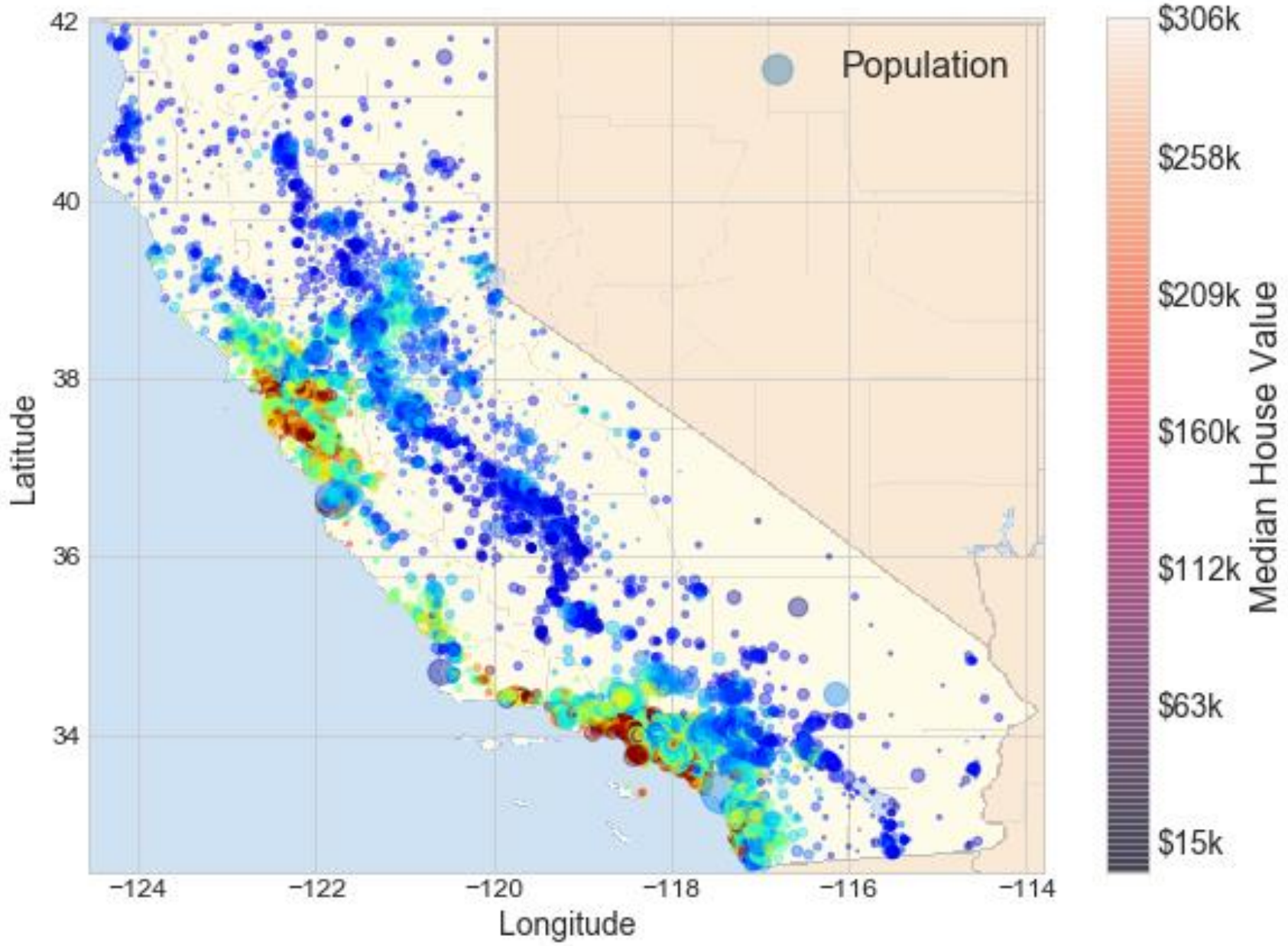
In [40]:

```
import matplotlib.image as mpimg
import numpy as np
california_img=mpimg.imread('california.png')
ax = MyDataset.plot(kind="scatter", x="longitude", y="latitude", figsize=(10,7),
                        s=MyDataset['population']/100, label="Population",
                        c="median_house_value", cmap=plt.get_cmap("jet"),
                        colorbar=False, alpha=0.4,
                       )
plt.imshow(california_img, extent=[-124.55, -113.80, 32.45, 42.05], alpha=0.5)
plt.ylabel("Latitude", fontsize=14)
plt.xlabel("Longitude", fontsize=14)

prices = MyDataset["median_house_value"]
tick_values = np.linspace(prices.min(), prices.max(), 11)
cbar = plt.colorbar()
cbar.ax.set_yticklabels(["$%dk"%(round(v/1000)) for v in tick_values], fontsize=14)
cbar.set_label('Median House Value', fontsize=16)

plt.legend(fontsize=16)
plt.savefig("california_housing_prices_plot")
plt.show()
```

Plotting the GIS Coordinates



3

Feature Cleaning, Selection and Transformation

Data Quality Rules

Rule 1 Attribute Domain Constraints	Optionality Constraints Format Constraints Valid Value Constraints Precision and Granularity Constraints
Rule 2 Relational Integrity Rules	Identity Rules Reference rules Cardinal Rules Inheritance rules
Rule 3 Rules for Historical Data	Timestamp Constraints Currency Rule Retention Rule Continuity Rule Granularity rule Timestamp Pattern Rule Direction of Change Magnitude of Change
Rule 4 Event History rules	Event Dependencies Event conditions Event-specific attribute constraints
Rule 5 Rules for State-Dependent Objects	State domain constraint State-transition constraints Continuity rules Action domain constraint Action pre-conditions Terminator domain constraint State-action constraints Duration rules

Data Quality Rule

Attribute domain constraints

Optionality Constraints

These constraints prevent attributes from taking Null, or missing values. Default values are often entered to circumvent the Not-Null constraints, i.e., the attribute is populated with a default value when actual value is not available.

Format Constraints:

These constraints define the expected form in which the attribute values are stored in the database field. To identify valid values, we first need to collect counts of all actual values. These counts can then be analysed, and actual values can be cross-referenced against the valid value list, if available. Values that have low frequency are suspect.

Valid Value Constraint

These constraints limit the permitted attribute values to a prescribed **list or range**.

Precision and Granularity Constraints

These constraints require all values of an attribute to have the same precision, granularity, and unit of measurement. Precision constraints can apply to both numeric and date/time attributes.

Data Quality Rules: An Introduction

July 2007: Originally published in IDQ Newsletter Vol 3 Issue 3

Arkady Maydanchik

Editor's Note: This article is the first in a series of excerpts from the book ***Data Quality Assessment*** by Arkady Maydanchik. The book presents systematic, step-by-step instructions for identifying, warehousing, and analyzing data errors.

This recently published book (May 2007, ISBN: 978-0977140022) can be purchased through our Bibliography page at bibliography.iaidq.org

Handling categorical variables

Integer Encoding

This is all about replacing the text values of the categorical variable with numbers e.g. INLAND = 1 .
Note: The allocated numeric values do not necessarily carry weight. Hence, there is need for one-hot encoding

One-Hot Encoding

One-hot encoding entails creating binary variable i.e. two values variable - 1 (hot) and 0 (cold) for each of the string or text values

Handling categorical variables

1. Integer Encoding

ocean_proximity (String Values)	ocean_proximity (Numeric Values)
<1H OCEAN	1
INLAND	2
NEAR OCEAN	3
NEAR BAY	4
ISLAND	5

Handling categorical variables

1. Integer Encoding

#1 Tabulate the values

In [4]:
```
MyDataset["ocean_proximity"].value_counts()
```

Out[4]:
```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

#2 Adding variable ocean_proximity_Recoded (duplicate of ocean_proximity) to the datset.

In [18]:
```
MyDataset["ocean_proximity_Recoded"] = MyDataset["ocean_proximity"]
```

In [20]:
```
#checking the datset with the code below, you will see that a new dataset has been added
MyDataset.head(2)
```

Out[20]:

de	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	ocean_proximity_Recoded
88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY	NEAR BAY
86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY	NEAR BAY

Handling categorical variables

#3 Create numeric label for each of the letters

In [24]:
```
Recoded_Values  = {"ocean_proximity_Recoded":{"<1H OCEAN": 1, "INLAND": 2, "NEAR OCEAN": 3, "NEAR BAY": 4, "ISLAND": 5}}
```

#4 Replacing the letter values of ocean_proximity_Recoded with numeric values

In [25]:
```
MyDataset.replace(Recoded_Values, inplace = True)
```

The values of ocean_proximity_Recoded are now numeric

In [26]:
```
MyDataset.head(2)
```

Out[26]:

g_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity	ocean_proximity_Recoded
41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY	4
21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY	4

Handlings the categorical variables

2. One-hot Encoding

Index	ocean_proximity
0	NEAR BAY
1	NEAR BAY
2	INLAND
3	NEAR OCEAN
4	<1H OCEAN
5	ISLAND
6	ISLAND
7	ISLAND
8	<1H OCEAN
9	NEAR BAY
10	INLAND

Index	Near Bay	Inland	Near Ocean	<1H Ocean	Island
0	1	0	0	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	0	0	0	1	0
5	0	0	0	0	1
6	0	0	0	0	1
7	0	0	0	0	1
8	0	0	0	1	0
9	1	0	0	0	0
10	0	1	0	0	0

Handling categorical variables

```
# Sorting the dataset by population so that we can see different examples
MyDataset_Sorted = MyDataset.sort_values(by='population', ascending = 0)
```

```
# Doing the one hot encoding
ocean_proximity_Encoded = pd.get_dummies(MyDataset_Sorted["ocean_proximity"])
```

```
ocean_proximity_Encoded.head(8)
```

Handling categorical variables

One-Hot Encoding

In [44]:
```
# Sorting the dataset by population so that we can see different examples
MyDataset_Sorted = MyDataset.sort_values(by='population', ascending = 0)
```

In [46]:
```
# Doing the one hot encoding
ocean_proximity_Encoded = pd.get_dummies(MyDataset_Sorted["ocean_proximity"])
```

In [40]:
```
ocean_proximity_Encoded.head(8)
```

Out[40]:

	<1H OCEAN	INLAND	ISLAND	NEAR BAY	NEAR OCEAN
15360	1	0	0	0	0
9880	1	0	0	0	0
13139	0	1	0	0	0
10309	1	0	0	0	0
6057	1	0	0	0	0
6066	1	0	0	0	0
12215	1	0	0	0	0
9019	0	0	0	0	1

Handling categorical variables

In [84]:
```
# Join or merge it with the original dataset for comparison
MyDataset_SortedJoined = MyDataset_Sorted.join(ocean_proximity_Encoded)
```

In [85]:
```
#Viewing selected variables and top 8 records
MyDataset_SortedJoined[['ocean_proximity','<1H OCEAN','INLAND','NEAR BAY','NEAR OCEAN']].head(8)
```

Out[85]:

	ocean_proximity	<1H OCEAN	INLAND	NEAR BAY	NEAR OCEAN
15360	<1H OCEAN	1	0	0	0
9880	<1H OCEAN	1	0	0	0
13139	INLAND	0	1	0	0
10309	<1H OCEAN	1	0	0	0
6057	<1H OCEAN	1	0	0	0
6066	<1H OCEAN	1	0	0	0
12215	<1H OCEAN	1	0	0	0
9019	NEAR OCEAN	0	0	0	1

Treating Missing Data

Method 1

You can delete the corresponding rows in the remaining columns or features.

Method 2

You can impute the values using based on the values generated from mean, mode, median etc.

Method 3

You can delete the feature.

Data Cleaning

Treating the missing values

Method 1: Deleting the corresponding records in the other columns or features

In [144]:
```
Mydaset_NoBedroom = MyDataset.dropna(subset=["total_bedrooms"])
Mydaset_NoBedroom.count()
# All the other featurees now 20433 from 20640
```

Out[144]:
```
longitude             20433
latitude              20433
housing_median_age    20433
total_rooms           20433
total_bedrooms        20433
population            20433
households            20433
median_income         20433
median_house_value    20433
ocean_proximity       20433
dtype: int64
```

Data Cleaning

Treating the missing values

Method 2: Imputing the missing values with the median or another values

In [145]:
```
median_value = MyDataset["total_bedrooms"].median()
MyDataset["total_bedrooms"].fillna(median_value, inplace=True)
MyDataset.count()
```

Out[145]:
```
longitude             20640
latitude              20640
housing_median_age    20640
total_rooms           20640
total_bedrooms        20640
population            20640
households            20640
median_income         20640
median_house_value    20640
ocean_proximity       20640
dtype: int64
```

Data Cleaning

Treating the missing values

Method 3: Deleting the variable or features with missing values

In [148]:
```
MyDataset.drop("total_bedrooms", axis=1)
MyDataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
total_rooms           20640 non-null float64
total_bedrooms        20640 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
```

Feature Selection

Filter Methods	**Features are selected based on their scores in various statistical tests. They are not dependent of any machine learning algorithms.**
Wrapper Methods	**We select some features and train a model using them. Based on the model performance, we can decide to remove or add more features.**
Embedded Methods	**Embedded methods combine the qualities' of filter and wrapper methods. It's implemented by algorithms that have their own built-in feature selection methods.**

Filter Method

Correlation
Linear discriminant analysis
Analysis of variance (ANOVA)
Chi-Square

Wrapper Method

Forward Selection
Backward Elimination
Recursive Feature elimination

Embedded Method

Lasso regression
Ridge regression

Correlation Matrix

Method 1

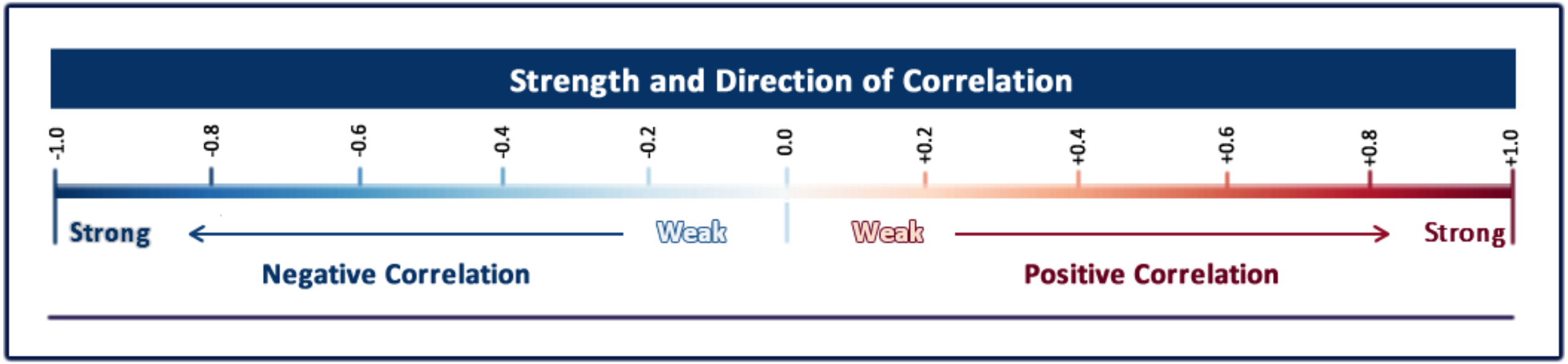
In [188]:
```
corr_matrix = MyDataset.corr()
corr_matrix
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_
longitude	1.000000	-0.924478	-0.105848	0.048871	0.076598	0.108030	0.063070	-0.019583	-0.0474
latitude	-0.924478	1.000000	0.005766	-0.039184	-0.072419	-0.115222	-0.077647	-0.075205	-0.1427
housing_median_age	-0.105848	0.005766	1.000000	-0.364509	-0.325047	-0.298710	-0.306428	-0.111360	0.11411
total_rooms	0.048871	-0.039184	-0.364509	1.000000	0.929379	0.855109	0.918392	0.200087	0.13509
total_bedrooms	0.076598	-0.072419	-0.325047	0.929379	1.000000	0.876320	0.980170	-0.009740	0.04768
population	0.108030	-0.115222	-0.298710	0.855109	0.876320	1.000000	0.904637	0.002380	-0.0269
households	0.063070	-0.077647	-0.306428	0.918392	0.980170	0.904637	1.000000	0.010781	0.06450
median_income	-0.019583	-0.075205	-0.111360	0.200087	-0.009740	0.002380	0.010781	1.000000	0.68716
median_house_value	-0.047432	-0.142724	0.114110	0.135097	0.047689	-0.026920	0.064506	0.687160	1.00000

Correlation Matrix

```
In [189]: f, ax = plt.subplots(figsize=(18, 10))
          sns.heatmap(corr_matrix,linewidths=2.0, ax=ax , annot=True)
          ax.set_title('Correlation Mtrix')
```

```
|: <matplotlib.text.Text at 0x958ff50>
```



When it is close to 1, it means that there is a strong positive correlation:
When the coefficient is close to -1, it means that there is a strong negative correlation
coefficients close to zero mean that there is no
linear correlation.

Feature Scaling

Feature scaling is a way of transforming your data

Min-max scaling	This is called Normalization. Values are shifted and rescaled so that they end up ranging from 0 to 1. This is done by subtracting the min value and dividing by the max minus the min.
Standardization	It subtracts the mean value (so standardized values always have a zero mean), and then it divides by the variance so that the resulting distribution has unit variance.

it is important to fit the scalers to the training data only, not to the full dataset (including the test set). Only then can you use them to transform the training set and the test set (and new data).

```
imputer.fit(housing_num)
```

The **imputer** has simply computed the median of each attribute and stored the result in its **statistics_** instance variable. Only the **total_bedrooms** attribute had missing values, but cannot be sure that there won't be any missing values in new data after the system goes live, so it is safer to apply the imputer to all the numerical attributes:

```
>>> imputer.statistics_
array([ -118.51 , 34.26 , 29. , 2119. , 433. , 1164. , 408. , 3.5414])
>>> housing_num.median().values
array([ -118.51 , 34.26 , 29. , 2119. , 433. , 1164. , 408. , 3.5414])
```

Now you can use this "trained" **imputer** to transform the training set by replacing missing values by the learned medians:

```
X = imputer.transform(housing_num)
```

The result is a plain Numpy array containing the transformed features. If you want to put it back into a Pandas DataFrame, it's simple:

```
housing_tr = pd.DataFrame(X,
columns=housing_num.columns)
```

Splitting the data

Random Splitting

In [57]:
```
#You need sckit-learn module for this
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(MyDataset, test_size=0.2, random_state=15)
print(len(train_set), "train +", len(test_set), "test")
```

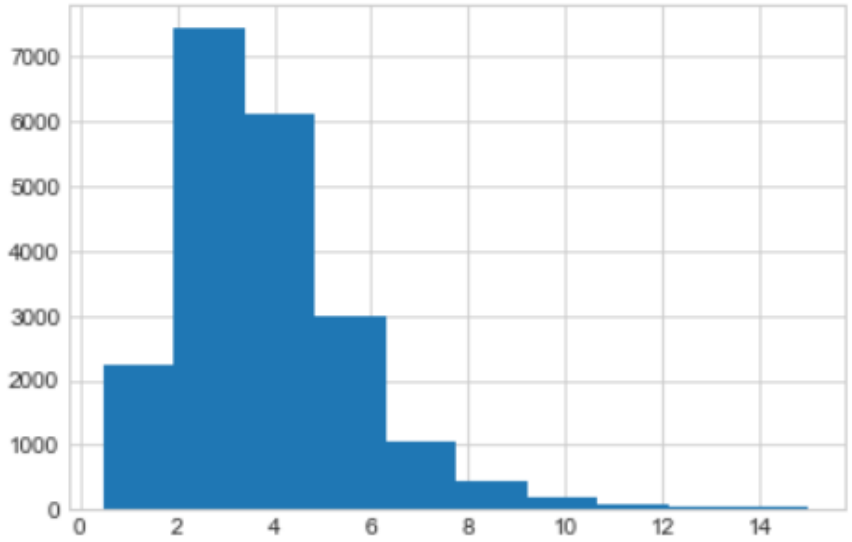
```
16512 train + 4128 test
```

Splitting the data

Stratified Splitting

```
In [61]: MyDataset["median_income"].hist()
```

```
Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0xb5d0fb0>
```




```
In [63]: MyDataset["income_cat"] = np.ceil(MyDataset["median_income"] / 1.5)
         MyDataset["income_cat"].where(MyDataset["income_cat"] < 5, 5.0, inplace=True)
         MyDataset["income_cat"].value_counts()
```

```
Out[63]: 3.0    7236
         2.0    6581
         4.0    3639
         5.0    2362
         1.0     822
         Name: income_cat, dtype: int64
```

Splitting the data

In [66]:
```
from sklearn.model_selection import StratifiedShuffleSplit

split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=15)

for train_index, test_index in split.split(MyDataset, MyDataset["income_cat"]):
    strat_train_set = MyDataset.loc[train_index]
    strat_test_set = MyDataset.loc[test_index]
```

In [68]:
```
print(len(strat_train_set), "train +", len(strat_test_set), "test")
```

```
16512 train + 4128 test
```

Splitting the data

Separating the labelled feature from other feeatures

In [69]:
```
# Droping the labelled feature, median_house_value
MyDataset = strat train set.drop("median house value", axis=1)
```

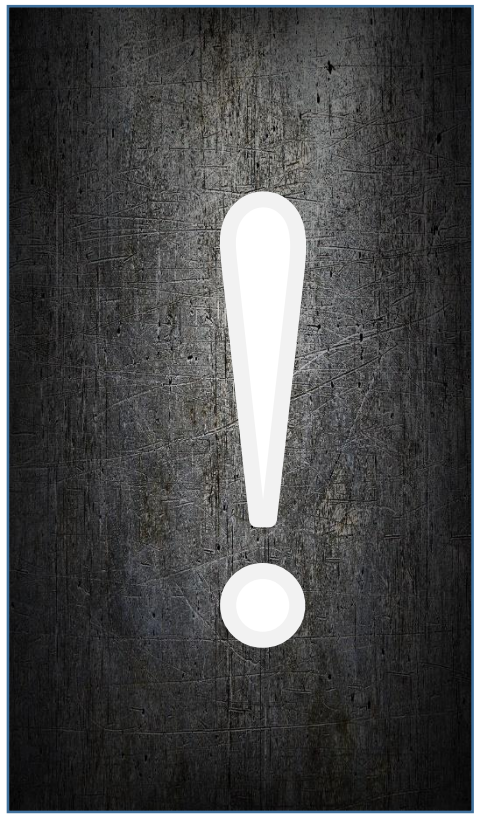
```
# Naking the labelled feature, median_house_value to stand alone
MyDataset_Label = strat_train_set["median_house_value"].copy()
```

Feature Creation

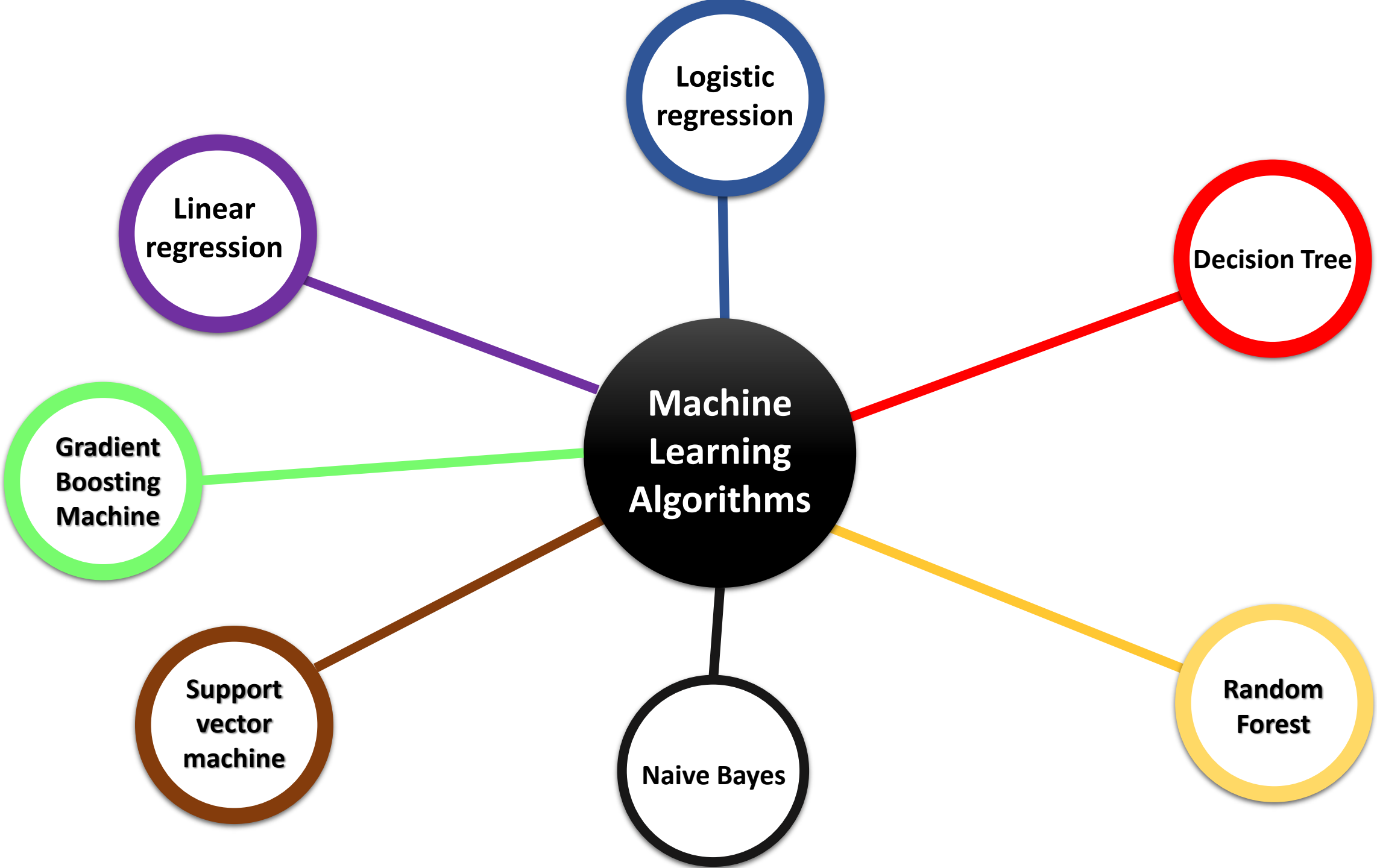
```
MyDataset["rooms_per_household"] =
MyDataset["total_rooms"]/MyDataset["households"]MyDataset["bedrooms_p
er_room"] =
MyDataset["total_bedrooms"]/MyDataset["total_rooms"]MyDataset["populati
on_per_household"]=MyDataset["population"]/MyDataset["households"]
```

4

Machine Learning Algorithm Adoption

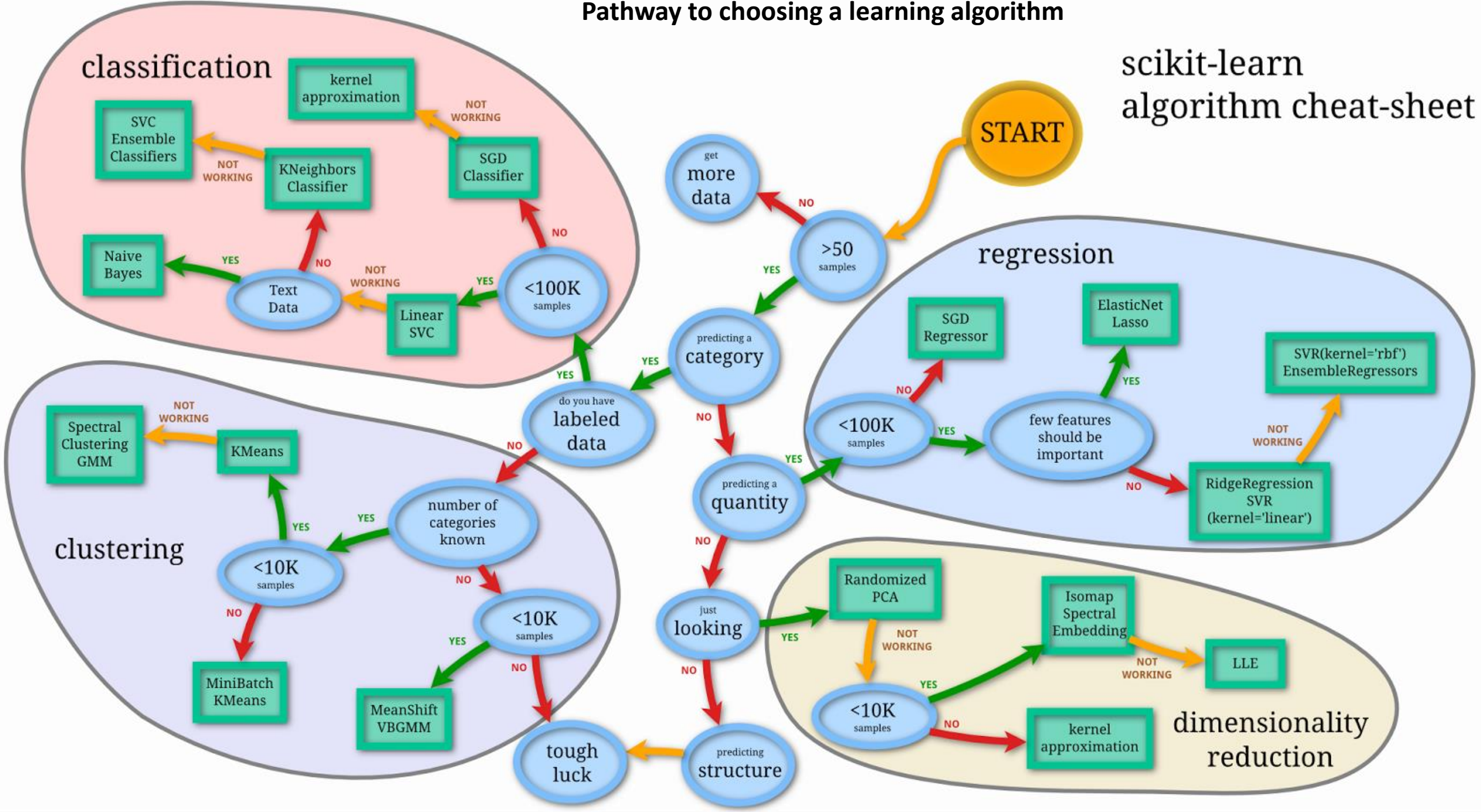


The list of machine learning algorithms provided in this presentation is not exhaustive of the available algorithms that can be applied to a data science project.


Logistic regression
Linear regression
Decision Tree
Machine Learning Algorithms
Gradient Boosting Machine
Support vector machine
Naive Bayes
Random Forest

Linear Regression	This is mathematically modelling the relationship between a predictor and an continuous outcome variables.
Logistic Regression	This is modelling the relationship between a predictor and a binary (two values) outcome variables.
Decision Tree	This modelling observations about an item (represented in the branches) to conclusions about the item's target value (represented in the leaves).
Random Forest	This constructs a multitude of decision trees at training time and outputs the class (classification) or mean prediction (regression) of the individual trees.
Naïve Bayes	This is the application of Bayes' theorem with the assumption of independence between the predictors. Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event.
Support Vector Machine	This is performs classification exercise by finding the hyperplane (decision surface) that maximizes the margin between the two classes.

Pathway to choosing a learning algorithm



Machine Learning Algorithms

	Unsupervised	Supervised

Continuous	• Clustering & Dimensionality Reduction
	  ○ SVD
	  ○ PCA
	  ○ K-means
		• Regression
		  ○ Linear
		  ○ Polynomial
		• Decision Trees
		• Random Forests

Categorical	• Association Analysis
	  ○ Apriori
	  ○ FP-Growth
	• Hidden Markov Model
		• Classification
		  ○ KNN
		  ○ Trees
		  ○ Logistic Regression
		  ○ Naive-Bayes
		  ○ SVM

Machine Learning Algorith Adoption

In [185]:
```
# Test Cases
some_data = MyDataset_Main.iloc[:4]
some_labels = MyDataset_Label.iloc[:4]
some_data_prepared = full_pipeline.transform(some_data)
```

1. Linear Regression

In [204]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(MyDataset_prepared, MyDataset_Label)
```

Out[204]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [197]:
```
print("Predictions:\t", lin_reg.predict(some_data_prepared))
print("Labels:\t\t", list(some_labels))
```

```
Predictions:    [ 210644.60459286  317768.80697211  210956.43331178   59218.98886849]
Labels:         [286600.0, 340600.0, 196900.0, 46300.0]
```

2. Decision Tree

In [206]:
```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor()
tree_reg.fit(MyDataset_prepared, MyDataset_Label)
```

```
print("Predictions:\t", tree_reg.predict(some_data_prepared))
print("Labels:\t\t", list(some_labels))
```

```
Predictions:	 [ 286600.  340600.  196900.   46300.]
Labels:		 [286600.0, 340600.0, 196900.0, 46300.0]
```

3. Random Forest

In [207]:
```
from sklearn.ensemble import RandomForestRegressor
forest_reg = RandomForestRegressor()
forest_reg.fit(MyDataset_prepared, MyDataset_Label)
```

```
print("Predictions:\t", forest_reg.predict(some_data_prepared))
print("Labels:\t\t", list(some_labels))
```

```
Predictions:	 [ 277590.  314160.  218800.   49360.]
Labels:		 [286600.0, 340600.0, 196900.0, 46300.0]
```

5 Model Performance Evaluation

The Measure

Mean Absolute Error (MSE)
Root Mean Absolute Error (RMAE)
log loss / multi-class logarithmic loss
Dice coefficient
SMAPE (Symmetric mean absolute percentage error)
Root Mean Squared Logarithmic Error – RMSLE
coefficient of determination (R^2)
ROC / area under the ROC curve
Global Average Precision (GAP)
mean F1 score.

1a. Linear Regression - The Model

In [228]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(MyDataset_prepared, MyDataset_Label)
```

Out[228]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [218]:
```
print("Predictions:\t", lin_reg.predict(some_data_prepared))
print("Labels:\t\t", list(some_labels))
```

```
Predictions:    [ 210644.60459286  317768.80697211  210956.43331178   59218.98886849]
Labels:         [286600.0, 340600.0, 196900.0, 46300.0]
```

1b. Linear Regression - The Model Performance Evaluation

Root Mean Squared Error (RMSE)

In [220]:
```
from sklearn.metrics import mean_squared_error
MyDataset_predictions_LR = lin_reg.predict(MyDataset_prepared)
lin_mse = mean_squared_error(MyDataset_Label, MyDataset_predictions_LR)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

Out[220]:
```
68628.198198489234
```

2a. Decision Tree -The Model

In [229]:
```
from sklearn.tree import DecisionTreeRegressor
tree_reg = DecisionTreeRegressor()
tree_reg.fit(MyDataset_prepared, MyDataset_Label)
housing_predictions = tree_reg.predict(MyDataset_prepared)

print("Predictions:\t", tree_reg.predict(some_data_prepared))
print("Labels:\t\t", list(some_labels))
```

```
Predictions:     [ 286600.  340600.  196900.   46300.]
Labels:          [286600.0, 340600.0, 196900.0, 46300.0]
```

2b. Decision Tree - The Model Performance Evaluation

In [225]:
```
MyDataset_predictions_DT = tree_reg.predict(MyDataset_prepared)
tree_mse = mean_squared_error(MyDataset_Label, MyDataset_predictions_DT)
tree_rmse = np.sqrt(tree_mse)
tree_rmse
```

Out[225]: 0.0

3a. Random Forest - The Model

In [226]:
```
from sklearn.ensemble import RandomForestRegressor
forest_reg = RandomForestRegressor()
forest_reg.fit(MyDataset_prepared, MyDataset_Label)

print("Predictions:\t", forest_reg.predict(some_data_prepared))
print("Labels:\t\t", list(some_labels))
```

```
Predictions:     [ 281400.  322030.  223700.   53430.]
Labels:          [286600.0, 340600.0, 196900.0, 46300.0]
```

3b. Random Forest - The Model Performance Evaluation

In [262]:
```
MyDataset_predictions_RF = forest_reg.predict(MyDataset_prepared)
forest_mse = mean_squared_error(MyDataset_Label, MyDataset_predictions_RF)
forest_mse = np.sqrt(forest_mse)
forest_mse
```

Out[262]: 22357.732495590219

Model Validation, Fine-Tuning & Ensembling

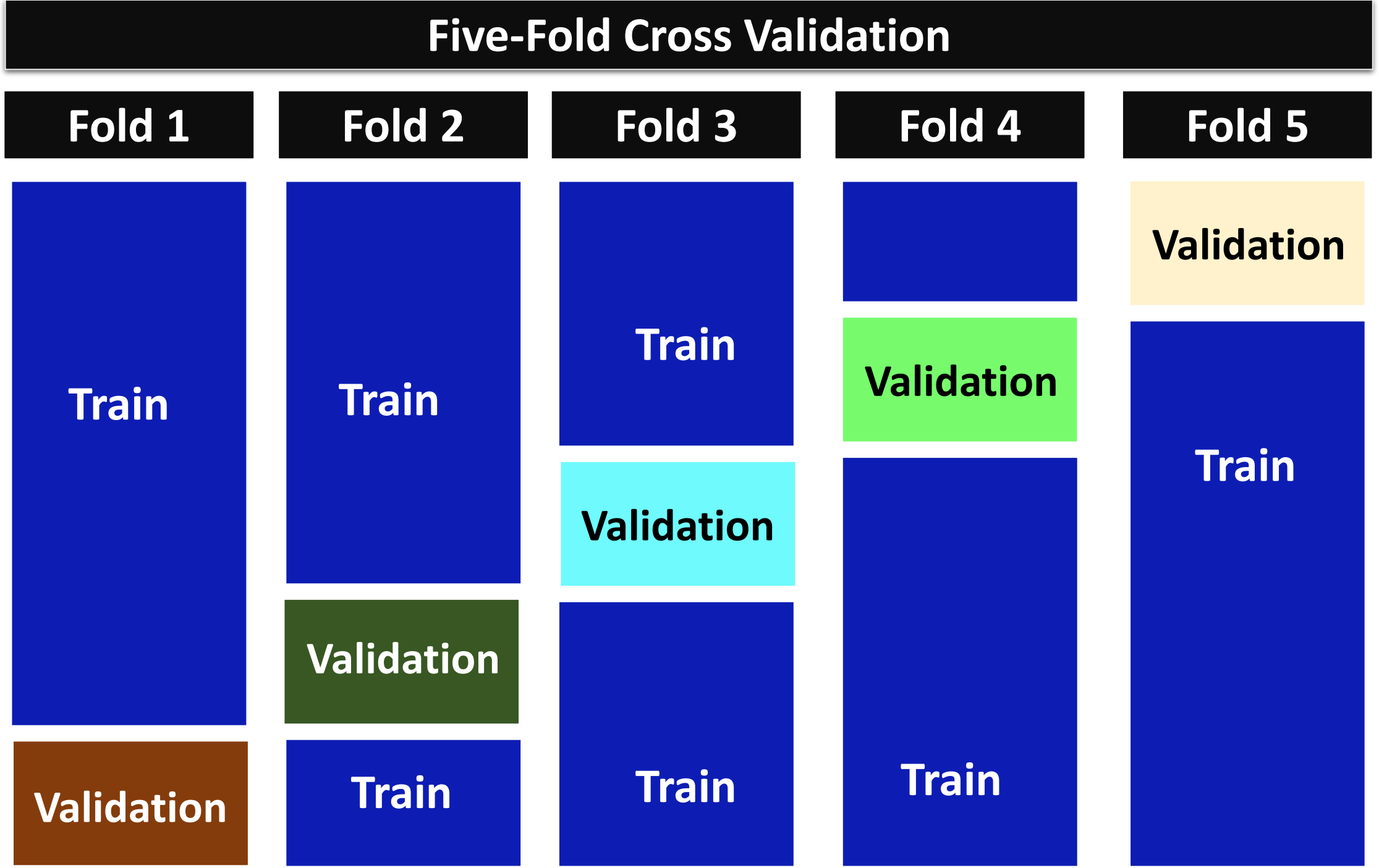
Cross Validation

Model Fine-Tuning

Ensembling Method

Cross Validation

This is a technique often adopted to validate the predicted results by randomly splitting the training data into training and validating set folds and afterwards train it.


Five-Fold Cross Validation
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Train
Validation
Train
Validation
Train
Train
Validation
Train
Train
Validation
Train
Validation
Train

Cross Validation

1 Linear Regression - Model Cross-Validation

In [244]:
```
from sklearn.model_selection import cross_val_score
scores_LR = cross_val_score(lin_reg, MyDataset_prepared, MyDataset_Label,
                            scoring="neg_mean_squared_error", cv=10)
rmse_scores_LR = np.sqrt(-scores_LR)
```

In [247]:
```
def display_scores(scores_LR):
    print("Scores:", scores_LR)
    print("Mean:", scores_LR.mean())
    print("Standard deviation:", scores_LR.std())

display_scores(rmse_scores_LR)
```

```
Scores: [ 66782.73843989  66960.118071    70347.95244419  74739.57052552
  68031.13388938  71193.84183426  64969.63056405  68281.61137997
  71552.91566558  67665.10082067]
Mean: 69052.4613635
Standard deviation: 2731.6740018
```

Cross Validation

2 Decision Tree - Model Cross-Validation

In [255]:
```
from sklearn.model_selection import cross_val_score
scores_DT = cross_val_score(tree_reg, MyDataset_prepared, MyDataset_Label,
                            scoring="neg_mean_squared_error", cv=10)
rmse_scores_DT = np.sqrt(-scores_DT)
```

In [272]:
```
def display_scores(scores_DT):
    print("Scores:", scores_DT)
    print("Mean:", scores_DT.mean())
    print("Standard deviation:", scores_DT.std())

display_scores(rmse_scores_DT)
```

```
Scores: [ 68216.27023026  67342.56932389  71866.05915468  68629.96135561
  70777.58555932  75264.91701834  71188.9289897   70546.634258
  75229.70787919  71948.61856934]
Mean: 71101.1252338
Standard deviation: 2542.85970979
```

Cross Validation

3 Random Forest - Model Cross-Validation

In [*]:
```
from sklearn.model_selection import cross_val_score
scores_RF = cross_val_score(forest_reg, MyDataset_prepared, MyDataset_Label,
                            scoring="neg_mean_squared_error", cv=10)
rmse_scores_RF = np.sqrt(-scores_RF)
```

In [274]:
```
def display_scores(scores_RF):
    print("Scores:", scores_RF)
    print("Mean:", scores_RF.mean())
    print("Standard deviation:", scores_RF.std())

display_scores(rmse_scores_RF)
```

```
Scores: [ 52030.29127654  49652.20128983  52801.96656611  55605.89563021
  52396.05640201  55498.64640096  52356.88154027  50088.67660153
  57024.01587125  52077.25665183]
Mean: 52953.1888231
Standard deviation: 2269.30900573
```

Model Fine-Tuning

GridSearch CV

Randomized Search

Model Fine-Tuning

1. Grid Search

In [281]:
```
from sklearn.model_selection import GridSearchCV

param_grid = [
        {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
        {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
    ]

forest_reg = RandomForestRegressor()
grid_search = GridSearchCV(forest_reg, param_grid, cv=5, scoring='neg_mean_squared_error')
grid_search.fit(MyDataset_prepared, MyDataset_Label)
```

Out[281]:
```
GridSearchCV(cv=5, error_score='raise',
       estimator=RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False),
       fit_params=None, iid=True, n_jobs=1,
       param_grid=[{'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]}, {'bootstrap': [False], 'n_e
: [3, 10], 'max_features': [2, 3, 4]}],
       pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
       scoring='neg_mean_squared_error', verbose=0)
```

In [81]:
```
grid_search.best_params_
```

Out[81]:
```
{'max_features': 6, 'n_estimators': 30}
```

Model Fine-Tuning

In [82]:
```
grid_search.best_estimator_
```

Out[82]:
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features=6, max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           n_estimators=30, n_jobs=1, oob_score=False, random_state=None,
           verbose=0, warm_start=False)
```

In [83]:
```
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
64912.0351358 {'max_features': 2, 'n_estimators': 3}
55606.8827598 {'max_features': 2, 'n_estimators': 10}
52946.9325185 {'max_features': 2, 'n_estimators': 30}
60384.0908354 {'max_features': 4, 'n_estimators': 3}
52709.9199934 {'max_features': 4, 'n_estimators': 10}
50503.5985321 {'max_features': 4, 'n_estimators': 30}
59058.1153485 {'max_features': 6, 'n_estimators': 3}
52172.0292957 {'max_features': 6, 'n_estimators': 10}
49958.9555932 {'max_features': 6, 'n_estimators': 30}
59122.260006 {'max_features': 8, 'n_estimators': 3}
52441.5896087 {'max_features': 8, 'n_estimators': 10}
50041.4899416 {'max_features': 8, 'n_estimators': 30}
62195.5242392 {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
54571.1333781 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
59634.0533132 {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
52456.0883904 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
58825.665239 {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
52002.8021556 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
```

Model Fine-Tuning

```
In [*]: pd.DataFrame(grid_search.cv_results_)
```

	mean_fit_time	mean_score_time	mean_test_score	mean_train_score	param_bootstrap	param_max_features	param_n_estimators	params
0	0.120785	0.003209	-4.116630e+09	-1.097586e+09	NaN	2	3	{'max_features': 2, 'n_estimators': 3}
1	0.349773	0.009121	-3.080688e+09	-5.656775e+08	NaN	2	10	{'max_features': 2, 'n_estimators': 10}
2	0.783634	0.030079	-2.797432e+09	-4.360042e+08	NaN	2	30	{'max_features': 2, 'n_estimators': 30}
3	0.105681	0.002808	-3.624491e+09	-9.541444e+08	NaN	4	3	{'max_features': 4, 'n_estimators': 3}

Model Fine-Tuning

rank_test_score	split0_test_score	...	split2_test_score	split2_train_score	split3_test_score	split3_train_score	split4_test_score	split4_train_score	s
18	-4.165303e+09	...	-4.238704e+09	-1.104297e+09	-3.900271e+09	-1.107609e+09	-4.208729e+09	-1.108423e+09	
11	-2.887620e+09	...	-3.053425e+09	-5.477571e+08	-2.816779e+09	-5.580728e+08	-3.376297e+09	-5.875674e+08	
9	-2.667445e+09	...	-2.874605e+09	-4.242697e+08	-2.652672e+09	-4.402507e+08	-2.866445e+09	-4.313052e+08	
16	-3.573981e+09	...	-3.660380e+09	-9.721007e+08	-3.573177e+09	-1.026416e+09	-3.657956e+09	-9.099575e+08	

Model Fine-Tuning

std_fit_time	std_score_time	std_test_score	std_train_score
0.035551	0.000751	1.222439e+08	1.237332e+07
0.106957	0.001655	2.147691e+08	1.317910e+07
0.148584	0.015302	1.141064e+08	7.147830e+06
0.006003	0.000750	4.158621e+07	5.440154e+07

Model Fine-Tuning

Randomized Search

In [*]:
```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {
        'n_estimators': randint(low=1, high=200),
        'max_features': randint(low=1, high=8),
    }

forest_reg = RandomForestRegressor()
rnd_search = RandomizedSearchCV(forest_reg, param_distributions=param_distribs,
                                n_iter=10, cv=5, scoring='neg_mean_squared_error')
rnd_search.fit(MyDataset_prepared, MyDataset_Label)
```

Out[124]:
```
RandomizedSearchCV(cv=5, error_score='raise',
          estimator=RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
           max_features='auto', max_leaf_nodes=None,
           min_impurity_decrease=0.0, min_impurity_split=None,
           min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
           oob_score=False, random_state=None, verbose=0, warm_start=False),
          fit_params=None, iid=True, n_iter=10, n_jobs=1,
          param_distributions={'n_estimators': <scipy.stats._distn_infrastructure.rv_frozen object at 0x0B31ACD0>
ax_features': <scipy.stats._distn_infrastructure.rv_frozen object at 0x0B31A870>},
          pre_dispatch='2*n_jobs', random_state=None, refit=True,
          return_train_score=True, scoring='neg_mean_squared_error',
          verbose=0)
```

Model Fine-Tuning

In [125]:
```
cvres = rnd_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
50783.2363071 {'max_features': 3, 'n_estimators': 49}
49232.4012125 {'max_features': 5, 'n_estimators': 148}
49218.9609239 {'max_features': 5, 'n_estimators': 145}
51771.8705727 {'max_features': 2, 'n_estimators': 105}
54160.3504342 {'max_features': 6, 'n_estimators': 6}
50199.557368 {'max_features': 6, 'n_estimators': 31}
49289.1190103 {'max_features': 6, 'n_estimators': 119}
52427.9576753 {'max_features': 4, 'n_estimators': 12}
49138.8009337 {'max_features': 7, 'n_estimators': 144}
54483.294949 {'max_features': 1, 'n_estimators': 150}
```

In [125]:
```
cvres = rnd_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
50783.2363071 {'max_features': 3, 'n_estimators': 49}
49232.4012125 {'max_features': 5, 'n_estimators': 148}
49218.9609239 {'max_features': 5, 'n_estimators': 145}
51771.8705727 {'max_features': 2, 'n_estimators': 105}
54160.3504342 {'max_features': 6, 'n_estimators': 6}
50199.557368 {'max_features': 6, 'n_estimators': 31}
49289.1190103 {'max_features': 6, 'n_estimators': 119}
52427.9576753 {'max_features': 4, 'n_estimators': 12}
49138.8009337 {'max_features': 7, 'n_estimators': 144}
54483.294949 {'max_features': 1, 'n_estimators': 150}
```