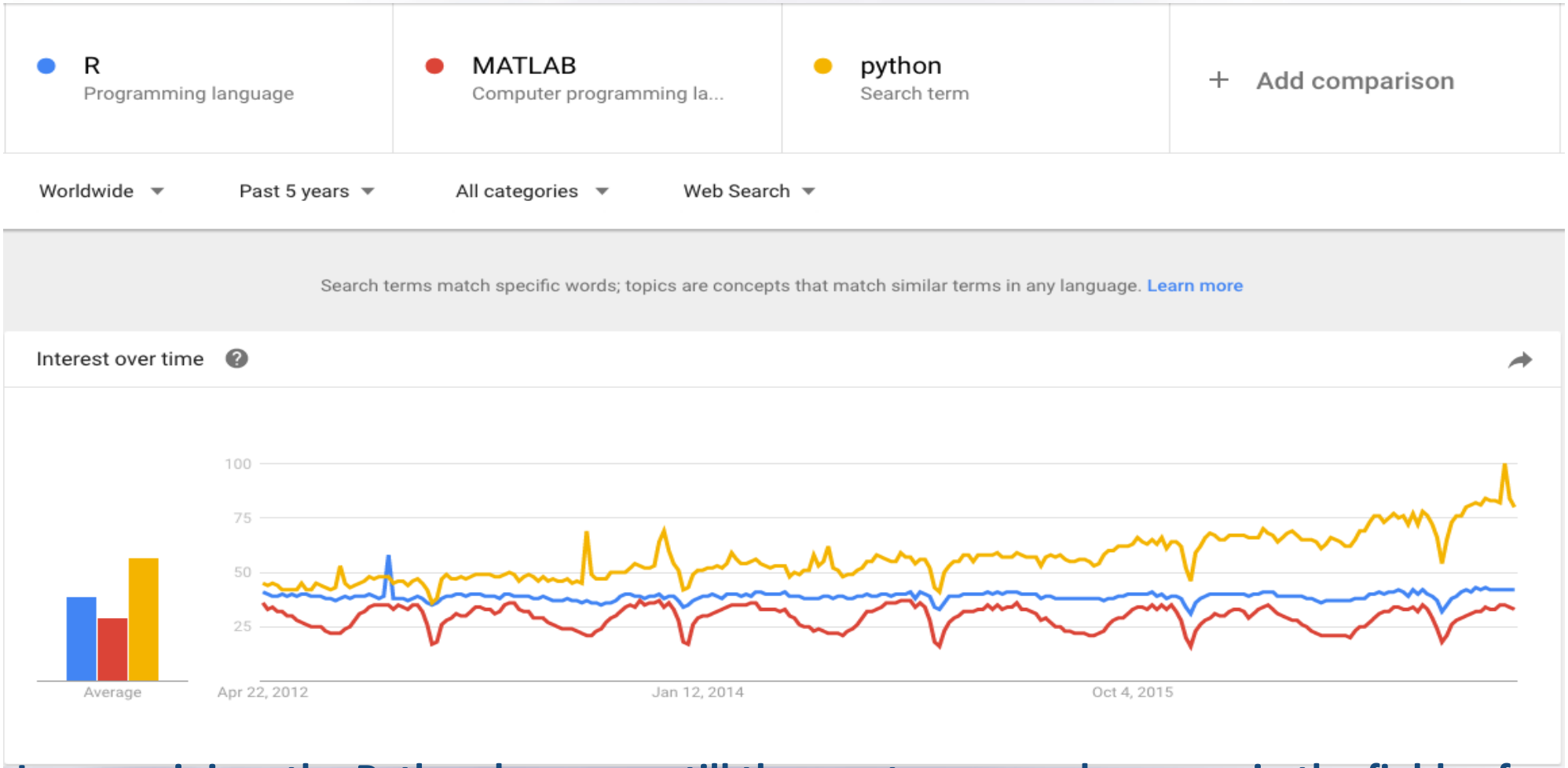
Data Analytic Methods



In my opinion, the Python language still the most common language in the fields of Machine Learning and Data Analysis.

Plan

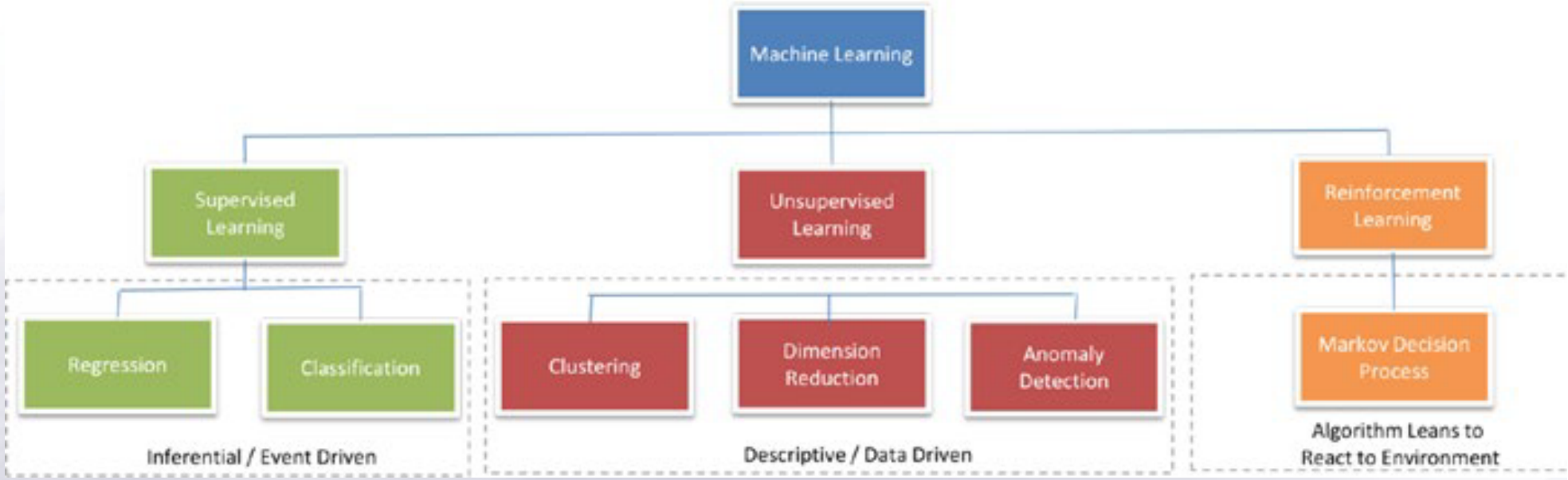
Introduction : Types of machine learning
1. Downloading, Installing and Starting Python and Libraries
2. Load The Data and libraries
3. Data Exploration
3.1 View of the dataset
3.2 Statistical description
4. Data Visualization
5 Developing a Model
5.1 Create a Validation Dataset
5.2 Build Model: Linear regression in scikit-learn
5.3 Model evaluation metrics for regression
6. Make Predictions
Summary

Introduction to Machine Learning in Python

Definition :

“Machine Learning is a collection of algorithms and techniques used to create computational systems that learn from data in order to make predictions and inferences.”

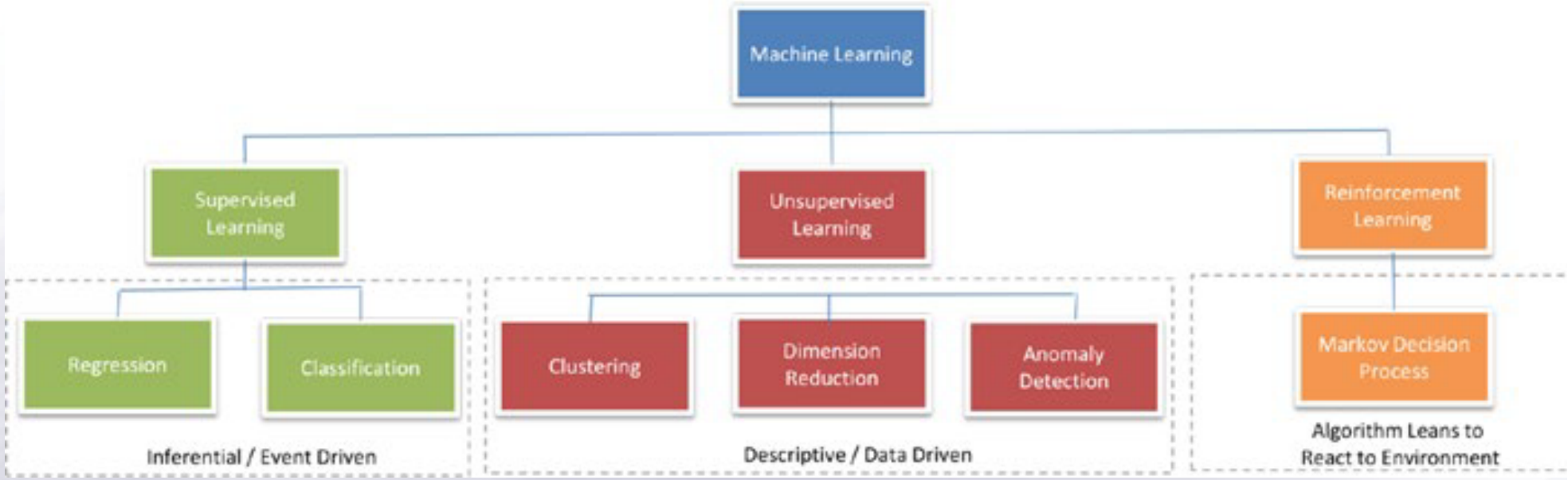
Types of Machine Learning



Supervised learning: making predictions using data

This is also called predictive modelling

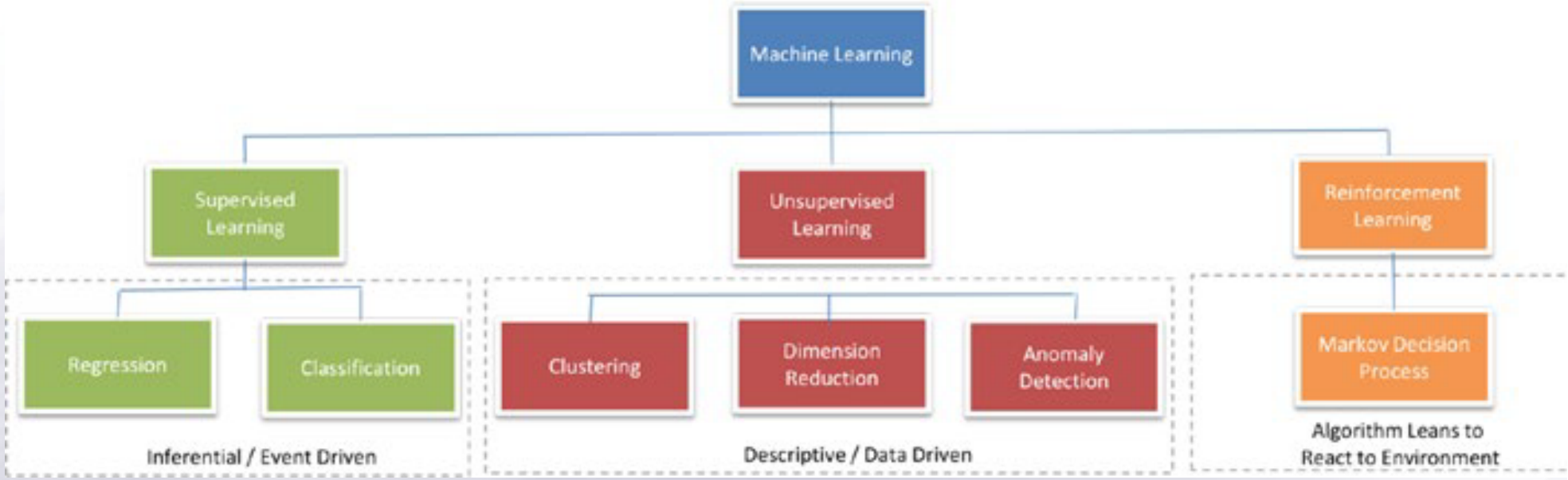
Types of Machine Learning



1) Regression

The output to be predicted is a continuous number in relevance with a given input dataset.

Types of Machine Learning



2) Classification

The output to be predicted is the actual or the probability of an event/class and the number of classes to be predicted can be two or more.

The algorithm should learn the patterns in the relevant input of each class from historical data and be able to predict the unseen class or event in the future considering their input.

Building supervised learning machine learning models

Building supervised learning machine learning models has three stages:

1. Training: The algorithm will be provided with historical input data with the mapped output. The algorithm will learn the patterns within the input data for each output and represent that as a statistical equation, which is also commonly known as a model.

2. Testing or validation: In this phase the performance of the trained model is evaluated, usually by applying it on a dataset
(that was not used as part of the training) to predict the class or event.

3. Prediction: Here we apply the trained model to a data set that was not part of either the training or testing. The prediction will be used to drive business decisions.

Exploratory Data Analytic Methods Using Python

Use EDA methods and Modeling processes to investigate a data set and make prediction with certain accuracy.

Use Python as a tool for programming.

Exploratory Data Analytic Methods Using Python

- Putting the Data Analytics Lifecycle into Practice
- Use a strategy to approach any data analytics problem:
  - Discovery and Data Preparation
  - Model Building and Evaluation
- To begin to analyze the data you need:
  - 1. A tool that allows you to look at the data – that is "python library".
  - 2. Skill in basic statistics.

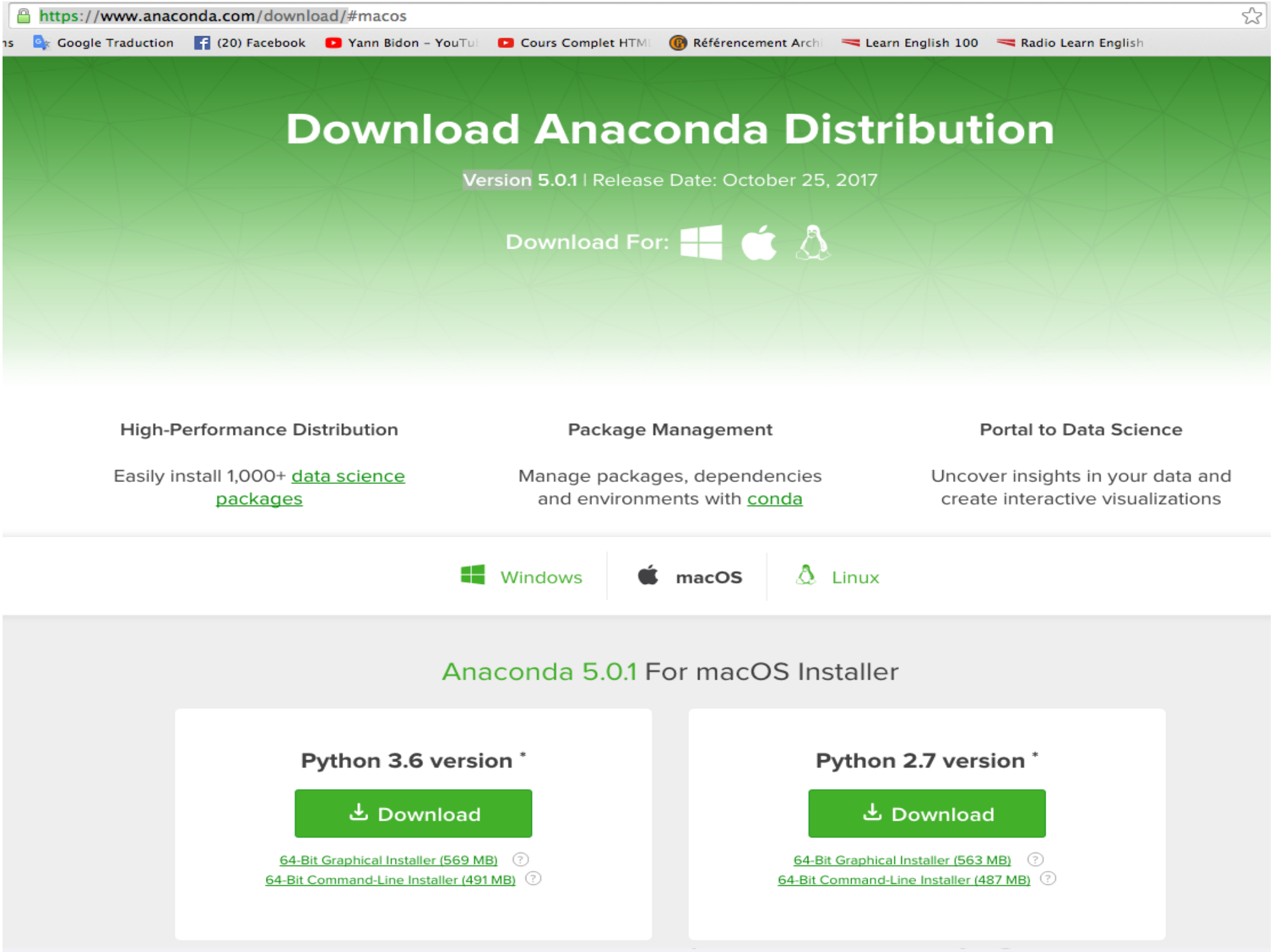
1. Downloading, Installing and Starting Python and Libraries

1.1 install Python version 2.7 or 3.5.

1.2 There are 5 key libraries that you will need to install.

scipy
numpy
matplotlib
pandas
sklearn

I would recommend installing the free version of Anaconda that includes everything you need.


https://www.anaconda.com/download/#macos
Google Traduction
(20) Facebook
Yann Bidon – YouTu
Cours Complet HTM
Référencement Arch
Learn English 100
Radio Learn English
Download Anaconda Distribution
Version 5.0.1 | Release Date: October 25, 2017
Download For:
High-Performance Distribution
Easily install 1,000+ data science packages
Package Management
Manage packages, dependencies and environments with conda
Portal to Data Science
Uncover insights in your data and create interactive visualizations
Windows
macOS
Linux
Anaconda 5.0.1 For macOS Installer
Python 3.6 version *
Download
64-Bit Graphical Installer (569 MB)
64-Bit Command-Line Installer (491 MB)
Python 2.7 version *
Download
64-Bit Graphical Installer (563 MB)
64-Bit Command-Line Installer (487 MB)

R vs. SAS vs. Julia vs. Python

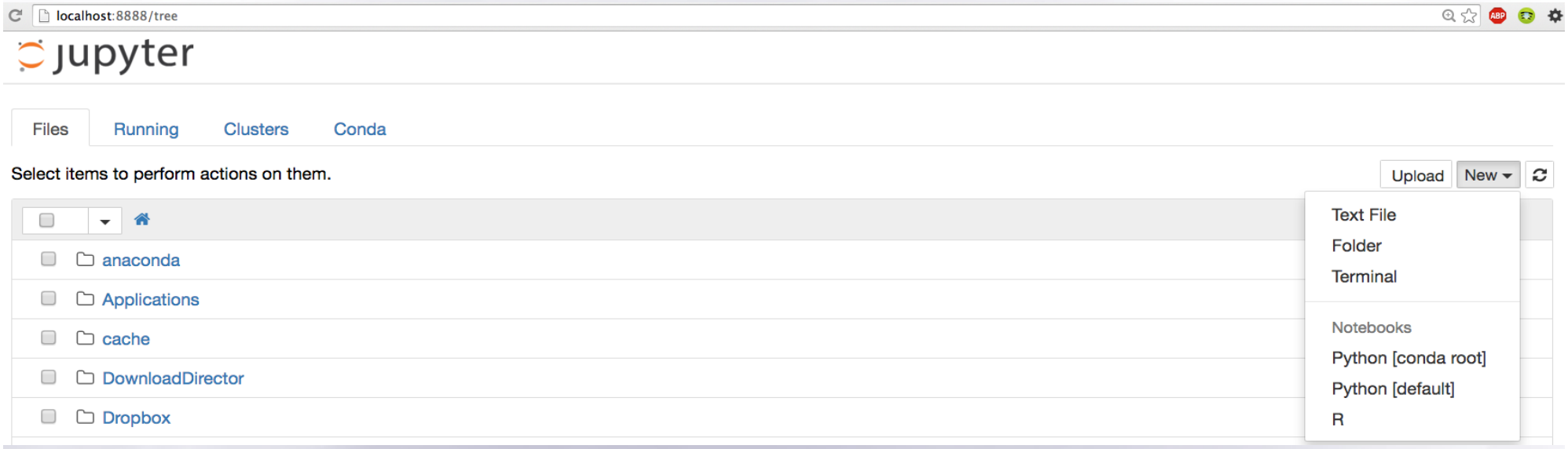
R is open source, SAS is a commercial product, Julia a very new dynamic programming language, ...

- **R** is free and available to everyone
- **R** code is open source and can be modified by everyone
- **R** is a complete and enclosed programming language
- **R** has a **big and active community**

⇔

⇔

JUPYTER


localhost:8888/tree
jupyter
Files
Running
Clusters
Conda
Select items to perform actions on them.
Upload
New
Text File
Folder
Terminal
Notebooks
Python [conda root]
Python [default]
R
anaconda
Applications
cache
DownloadDirector
Dropbox

Define Problem for case study

Prediction housing price

Goal: Model Evaluation and Validation for the Prediction of Boston Housing Prices.

For this project we will investigate the Boston House Price dataset.

Each record in the database describes a Boston suburb or town.

The data was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970.

https://archive.ics.uci.edu/ml/machine-learning-databases/housing/

Load The Data

We are going to use the Boston House Price dataset.

This dataset is used in machine learning and statistics by pretty much everyone.

The dataset contains 506 Instances 14 of Attributes

https://archive.ics.uci.edu/ml/machine-learning-databases/housing/

Import packages and dataset

jupyter tutoML (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python [default]

Run Markdown Publish your notebook into Anaconda.org Edit Presentation Show Presentation

Linear Regrssion on US Housing Price

- load the Python libraries needed for this case study and run the code below to load the Boston housing dataset.

Import packages and dataset

In [ ]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

- We are going to use the Boston House Price dataset.
- This dataset is used in machine learning and statistics by pretty much everyone.
- The dataset contains 506 Instances 14 of Attributes

https://archive.ics.uci.edu/ml/machine-learning-databases/housing/

In [ ]:
```
# Load dataset
filename = "https://archive.ics.uci.edu/ml/machine-learning-databases/housing/housing.data"
names = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']
dataset = pd.read_csv(filename, delim_whitespace=True, names=names)
```

- We are using pandas to load the data.
- We will also use pandas next to explore the data both with descriptive statistics and data visualization.

The dataset should load without incident.
If you do have network problems, you can download the " housing.data " file into your working directory and load it using the same method, changing URL to the local file name. (you must change .data to .csv format file)

Analyzing and Exploring the Data

In this step we are going to take a look at the data a two different ways:
1- View of the dataset.
2- Statistical summary of all attributes.

```
# Dimension and Descriptive statistics
dataset.shape
```

```
(506, 14)
```

```
# display the forst 10 and the last 5 rows

dataset.head(10)

dataset.tail()
```

1- View of the dataset

In [15]:
```
# Dimension and Descriptive statistics

dataset.shape
```

Out[15]: (506, 14)

In [14]:
```
dataset.head(10)
```

Out[14]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222.0	18.7	396.90	5.33	36.2
5	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222.0	18.7	394.12	5.21	28.7
6	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311.0	15.2	395.60	12.43	22.9
7	0.14455	12.5	7.87	0	0.524	6.172	96.1	5.9505	5	311.0	15.2	396.90	19.15	27.1
8	0.21124	12.5	7.87	0	0.524	5.631	100.0	6.0821	5	311.0	15.2	386.63	29.93	16.5
9	0.17004	12.5	7.87	0	0.524	6.004	85.9	6.5921	5	311.0	15.2	386.71	17.10	18.9

In [9]:
```
# display the last 5 rows
dataset.tail()
```

Out[9]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273.0	21.0	391.99	9.67	22.4
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273.0	21.0	396.90	9.08	20.6
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273.0	21.0	396.90	5.64	23.9
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273.0	21.0	393.45	6.48	22.0
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273.0	21.0	396.90	7.88	11.9

Statistics for Model Building and Evaluation

A statistical analysis may be descriptive, simply reporting, visualizing and summarizing a data set, but usually it is also inferential.

- **Descriptive Statistics**
- **Linear Regression**
- **Metrics**

Descriptive Statistics

Now we can take a look at a summary of each attribute.
This includes the count, mean, the min and max values as well as some percentiles.

'describe( )' ; is a method to get the statistical summary of the various features of the data set

```
dataset.describe()
```

Description

**'describe()' method to get the statistical summary of the various features of the data set

```
dataset.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LST
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000(
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653(
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141(
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730(
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950(
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360(
75%	3.677082	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955(
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970(

**Correlation matrix

```
dataset.corr()
```

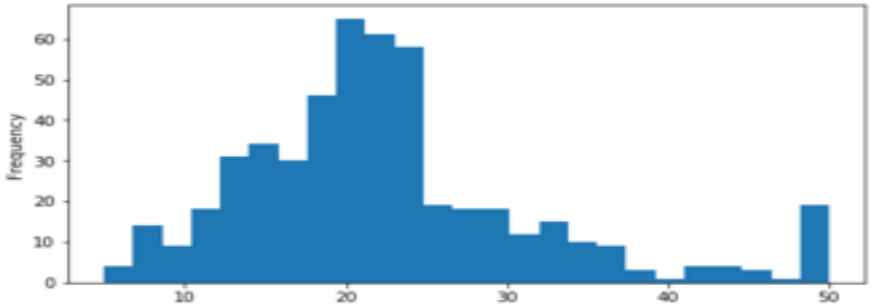
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
CRIM	1.000000	-0.200469	0.406583	-0.055892	0.420972	-0.219247	0.352734	-0.379670	0.625505	0.582764	0.289946	-0.385064	0.455621	-0.388305
ZN	-0.200469	1.000000	-0.533828	-0.042697	-0.516604	0.311991	-0.569537	0.664408	-0.311948	-0.314563	-0.391679	0.175520	-0.412995	0.360445
INDUS	0.406583	-0.533828	1.000000	0.062938	0.763651	-0.391676	0.644779	-0.708027	0.595129	0.720760	0.383248	-0.356977	0.603800	-0.483725
CHAS	-0.055892	-0.042697	0.062938	1.000000	0.091203	0.091251	0.086518	-0.099176	-0.007368	-0.035587	-0.121515	0.048788	-0.053929	0.175260
NOX	0.420972	-0.516604	0.763651	0.091203	1.000000	-0.302188	0.731470	-0.769230	0.611441	0.668023	0.188933	-0.380051	0.590879	-0.427321
RM	-0.219247	0.311991	-0.391676	0.091251	-0.302188	1.000000	-0.240265	0.205246	-0.209847	-0.292048	-0.355501	0.128069	-0.613808	0.695360
AGE	0.352734	-0.569537	0.644779	0.086518	0.731470	-0.240265	1.000000	-0.747881	0.456022	0.506456	0.261515	-0.273534	0.602339	-0.376955
DIS	-0.379670	0.664408	-0.708027	-0.099176	-0.769230	0.205246	-0.747881	1.000000	-0.494588	-0.534432	-0.232471	0.291512	-0.496996	0.249929
RAD	0.625505	-0.311948	0.595129	-0.007368	0.611441	-0.209847	0.456022	-0.494588	1.000000	0.910228	0.464741	-0.444413	0.488676	-0.381626
TAX	0.582764	-0.314563	0.720760	-0.035587	0.668023	-0.292048	0.506456	-0.534432	0.910228	1.000000	0.460853	-0.441808	0.543993	-0.468536
PTRATIO	0.289946	-0.391679	0.383248	-0.121515	0.188933	-0.355501	0.261515	-0.232471	0.464741	0.460853	1.000000	-0.177383	0.374044	-0.507787
B	-0.385064	0.175520	-0.356977	0.048788	-0.380051	0.128069	-0.273534	0.291512	-0.444413	-0.441808	-0.177383	1.000000	-0.366087	0.333461
LSTAT	0.455621	-0.412995	0.603800	-0.053929	0.590879	-0.613808	0.602339	-0.496996	0.488676	0.543993	0.374044	-0.366087	1.000000	-0.737663
MEDV	-0.388305	0.360445	-0.483725	0.175260	-0.427321	0.695360	-0.376955	0.249929	-0.381626	-0.468536	-0.507787	0.333461	-0.737663	1.000000

Visualization : Feature and variable sets

```
# Feature and variable sets
```

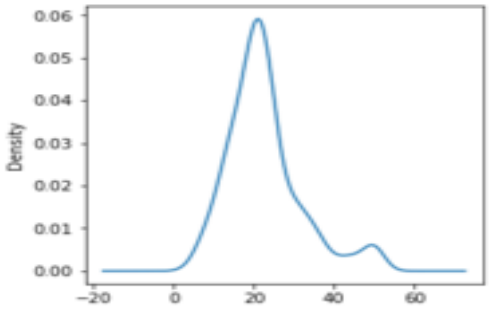
In [34]:
```
prices = dataset['MEDV']
features = dataset.drop('MEDV', axis = 1)
```

In [18]:
```
prices.plot.hist(bins=25,figsize=(8,4))
plt.show()
```



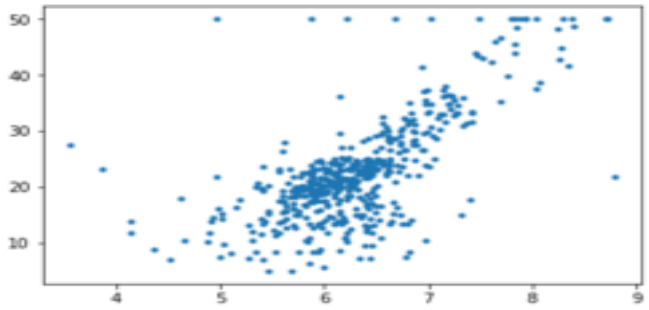
Histogram of the prices: Is a fast way to get an idea of the distribution of attribute.

In [19]:
```
prices.plot.density(figsize=(4,4))
plt.show()
```



Density of the prices; Density plots are another way of getting a quick idea of the distribution of attribute.

In [25]:
```
plt.plot(dataset['RM'],prices,'.')
plt.show()
```



prices vs RM: A scatter plot shows the relationship between two variables as dots in two dimensions, one axis for each attribute.

Developin a Model : Linear Regression

Regression analysis is a very widely used statistical tool to establish a relationship model between variables.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_n x_n$$

- y is the response
- β_0 is the intercept
- β_1 is the coefficient for x_1 (the first feature)
- β_n is the coefficient for x_n (the nth feature)

We develop the tools and techniques necessary for a model to make a prediction and estimate their accuracy on unseen data.

steps:

1. **Shuffle and Split Data and use 10-fold cross validation.**
2. **Build and Select the best models to predict Selling Prices**

Create a Validation Dataset

we take the Boston housing dataset and split the data into training and testing subsets.

We will split the loaded dataset into two, 80% of which we will use to train our models and 20% that we will hold back as a validation dataset.

Typically, the data is also shuffled into a random order when creating the training and testing subsets to remove any bias in the ordering of the dataset.

Developin a Model : Linear Regression

Developing a Model

1. Prepare Data: Test-train split

****Import train_test_split function from scikit-learn**

```
from sklearn.cross_validation import train_test_split
# Split-out validation dataset
validation_size = 0.20
seed = 7
X_train, X_test, Y_train, Y_test = train_test_split(features, prices, test_size=validation_size, random_state=seed)
```

Check the size and shape of train/test splits

```
print("Training feature set size:",X_train.shape)
print("Test feature set size:",X_test.shape)
print("Training variable set size:",Y_train.shape)
print("Test variable set size:",Y_test.shape)
```

```
Training feature set size: (404, 13)
Test feature set size: (102, 13)
Training variable set size: (404,)
Test variable set size: (102,)
```

Developin a Model : Linear Regression

Model fit and training

Import linear regression model estimator from scikit-learn and instantiate

```
from sklearn.linear_model import LinearRegression
from sklearn import metrics
regressor = LinearRegression()
regressor.fit(X_train, Y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

model coefficients

```
print ("coefficients: %s\nintercept: %0.3f" % \
(regressor.coef_,regressor.intercept_))
```

```
coefficients: [ -1.24649091e-01   3.04735052e-02   2.17990089e-02   2.79225761e+00
  -1.52135247e+01   5.27249266e+00  -1.10577742e-02  -1.27320872e+00
   2.65804711e-01  -1.15043029e-02  -9.19571148e-01   1.01624292e-02
  -3.89712044e-01]
intercept: 23.554
```

Make Predictions

Evaluation Metrics

For regression metrics, the Boston House Price dataset is used as demonstration. this is a regression problem where all of the input variables are also numeric.

Regression Metrics:

We will review 3 of the most common metrics for evaluating predictions on regression machine learning problems:

. **Mean Absolute Error.**

. **Mean Squared Error And (RMSE)**

. **R2**.

Model evaluation metrics for regression

- Evaluation metrics for classification problems, such as accuracy, are not useful for regression problems.
- Instead, we need evaluation metrics designed for comparing continuous values.

:
```
# make predictions on the testing set

y_pred = regressor.predict(X_test)
```

- Mean Absolute Error** (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

- Mean Squared Error** (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

:
```
# Calculate (MSE) and root-mean-square error (RMSE)
from sklearn.metrics import r2_score , mean_absolute_error, mean_squared_error

lin_mse = mean_squared_error(y_pred, Y_test)
lin_rmse = np.sqrt(lin_mse)

print('Linear Regression MSE: %.4f' % lin_mse)
print('Linear Regression RMSE: %.4f' % lin_rmse)
```

```
Linear Regression MSE: 34.0565
Linear Regression RMSE: 5.8358
```

:
```
# function to calculate MAE; MSE and RMSE
print("Mean absolute error (MAE):", metrics.mean_absolute_error(Y_test,y_pred))
print("Mean square error (MSE):", metrics.mean_squared_error(Y_test,y_pred))
print("Root mean square error (RMSE):", np.sqrt(metrics.mean_squared_error(Y_test,y_pred)))
```

```
Mean absolute error (MAE): 3.78076290917
Mean square error (MSE): 34.0564813489
Root mean square error (RMSE): 5.83579312081
```

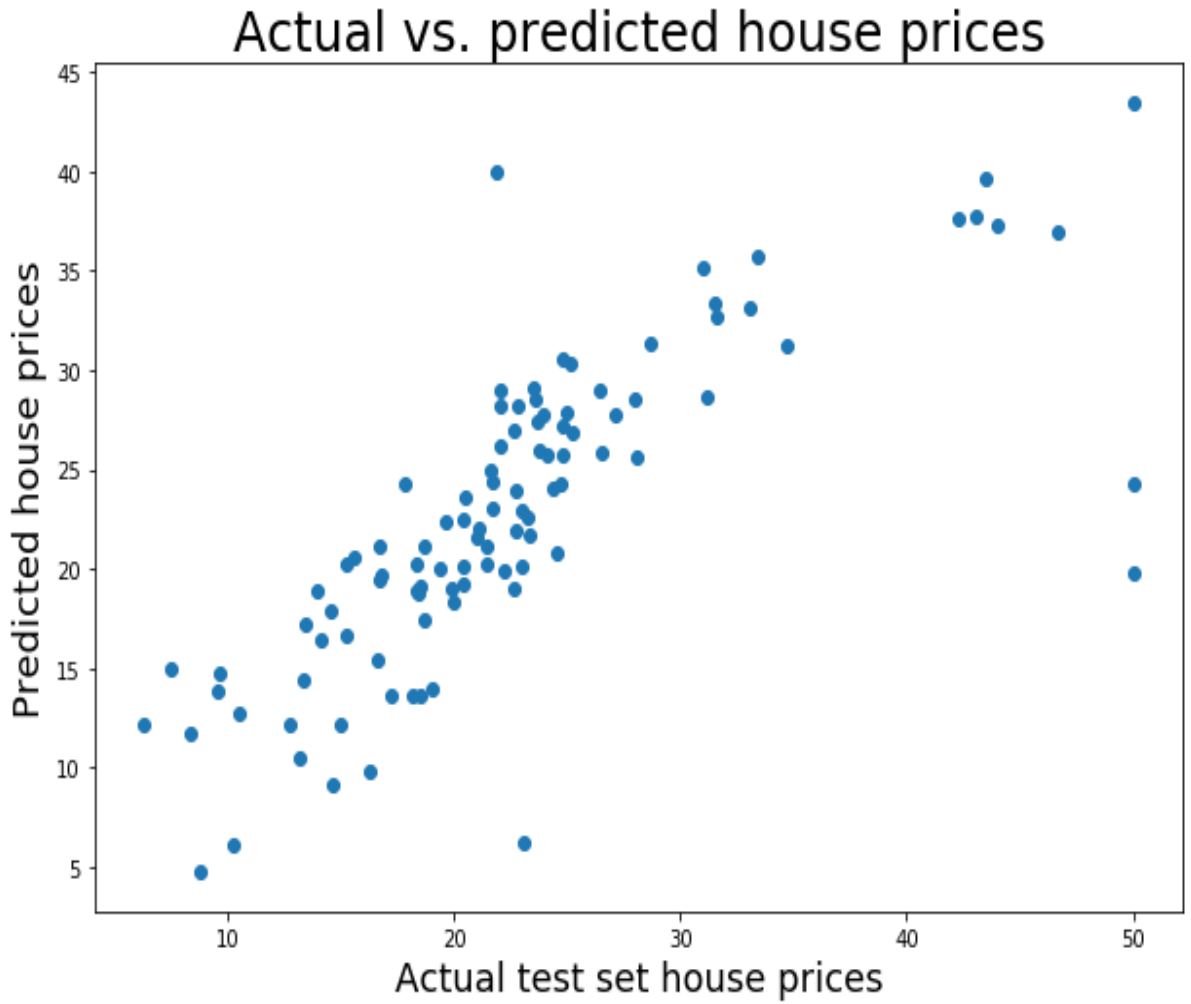
:
```
# function to calculate r-squared

print ("R Squared : ", r2_score(y_pred, Y_test))
```

```
R Squared :  0.443156217752
```

Actual vs. predicted house prices

```
plt.figure(figsize=(10,7))
plt.title("Actual vs. predicted house prices",fontsize=25)
plt.xlabel("Actual test set house prices",fontsize=18)
plt.ylabel("Predicted house prices", fontsize=18)
plt.scatter(x=Y_test,y=y_pred)
plt.show()
```



STATMODEL

```
import statsmodels.api as sm
# create a fitted model & print the summary
X_train = sm.add_constant(X_train)
lm = sm.OLS(Y_train, X_train).fit()
print (lm.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MEDV   R-squared:                       0.770
Model:                            OLS   Adj. R-squared:                  0.762
Method:                 Least Squares   F-statistic:                     100.4
Date:                Sat, 20 Jan 2018   Prob (F-statistic):          1.40e-115
Time:                        11:37:52   Log-Likelihood:                -1174.7
No. Observations:                 404   AIC:                             2377.
Df Residuals:                     390   BIC:                             2433.
Df Model:                          13
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         23.5542      5.656      4.164      0.000      12.434      34.674
CRIM          -0.1246      0.038     -3.316      0.001      -0.199      -0.051
ZN             0.0305      0.014      2.140      0.033       0.002       0.058
INDUS          0.0218      0.063      0.348      0.728      -0.101       0.145
CHAS           2.7923      0.920      3.034      0.003       0.983       4.602
NOX          -15.2135      4.104     -3.707      0.000     -23.281      -7.146
RM             5.2725      0.462     11.400      0.000       4.363       6.182
AGE           -0.0111      0.014     -0.778      0.437      -0.039       0.017
DIS           -1.2732      0.207     -6.156      0.000      -1.680      -0.867
RAD            0.2658      0.068      3.894      0.000       0.132       0.400
TAX           -0.0115      0.004     -3.089      0.002      -0.019      -0.004
PTRATIO       -0.9196      0.140     -6.545      0.000      -1.196      -0.643
B              0.0102      0.003      3.406      0.001       0.004       0.016
LSTAT         -0.3897      0.057     -6.880      0.000      -0.501      -0.278
==============================================================================
Omnibus:                      140.357   Durbin-Watson:                   2.019
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              568.509
Skew:                           1.494   Prob(JB):                    3.55e-124
Kurtosis:                       7.985   Cond. No.                     1.57e+04
==============================================================================
```

Summary

Key points covered in this part:

- How to use step-by-step machine learning to investigate a dataset in Python.
- How to discover a small end-to-end project from loading the data to making predictions in the best way with a new platform.

THANK YOU