IBM Data Science Experience (DSX) platform for machine learning

Rafie Tarabay

eng_rafie@mans.edu.eg
mrafie@eg.ibm.com

2017

IBM one of the TOP data science platform leaders

Figure 1. Magic Quadrant for Data Science Platforms

Source: Gartner (February 2017)

the IBM Data Science Experience (DSX) platform

- IBM DSX is a powerful computational engine based on Apache Spark Executors. It has a strong computing capacity in the back end. It currently supports Python, R, and Scala.
- Using IBM DSX, you can create a Python, R, or Scala, notebook-based project and create a data connection to your data source. You have options to load all types of Machine Learning algorithms that are supported by runtime from KNN and RandomForest to TensorFlow.

You can use notebook to :

- Loaded your data
- Created data sets
- Modeled, trained, and validated your data

Register for IBM Cloud here : http://ibm.biz/MLChallenge

Log in to IBM DataScience Experience with your IBM Cloud credientials - https://datascience.ibm.com

Training Steps

- What is machine learning?
- How to develop machine learning using Python?
- How to develop machine learning using Spark?
- How to develop machine learning using R?
- How to use IBM Data Science Experience (DSX) platform to run your script?

MACHINE LEARNING

Machine Learning

- a machine learning-based system is not explicitly programmed but learnt from data.
- a machine learning algorithm infers patterns and relationships between different variables in a dataset then uses that knowledge to generalize beyond the training dataset.

Terms used in the context of machine learning

1. Data mining
2. Features
3. Labels
4. Models
5. Accuracy, and Precision

Data Mining

- Definition: "the application of specific algorithms for extracting patterns from data."
- **data mining** is a **process**, during which **machine learning** algorithms are utilized as **tools** to extract potentially-valuable patterns held within datasets.
- Data mining is exploratory analysis with unsupervised machine learning algorithms

The main application is text mining in a big data environment:

1. Text parsing
2. Sentiment analysis
3. Opinion mining
4. Natural language processing using *deep learning algorithm

* deep learning is the process of applying deep neural network technologies - that is, neural network architectures with multiple hidden layers - to solve problems.
Deep learning is a process, like data mining, which employs deep neural network architectures, which are particular types of machine learning algorithms.

Data Mining algorithms:

1. C4.5
2. k-means
3. Support vector machines
4. Apriori
5. EM
6. PageRank
7. AdaBoost
8. kNN
9. Naive Bayes
10. CART

For more information http://infolab.stanford.edu/~ullman/mmds/book.pdf

Features

- a feature represents an independent variable
- In a tabular dataset, a row represents an observation and column represents a feature.
- Features are also collectively referred to as dimensions.

Categorical Features

- It can take on one of a fixed number of discrete values with name or a label.
- The values of a categorical feature have no ordering.
- gender is a categorical feature. It can take on only one of two values.

Numerical Features

- It can take on any numerical value
- numerical feature have mathematical ordering
- discrete numerical feature : number of persons, number of rooms
- continuous numerical feature: temperature value

Labels

- label is a variable that a machine learning learns to predict
- categorical label: ex, category of a news article is a categorical label
- numerical label: ex, price is a numerical label

← Features →					Label
Position	Experience	Skill	Country	City	Salary ($)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	USA	New York	116100
Developer	8	1	USA	New York	117800
Developer	9	1	USA	New York	119700
Developer	10	1	USA	New York	121600

Models

- a mathematical construct for capturing patterns within a dataset and estimates the relationship between the dependent and independent variables and has predictive capability. and can calculate or predict the value for the dependent variable when get the values of the independent variables.

- Training a model is a compute intensive task, while using it is not as compute intensive.
- A model is generally saved to disk, so that it can be used without go to training step again.

Training Data (80% of data)

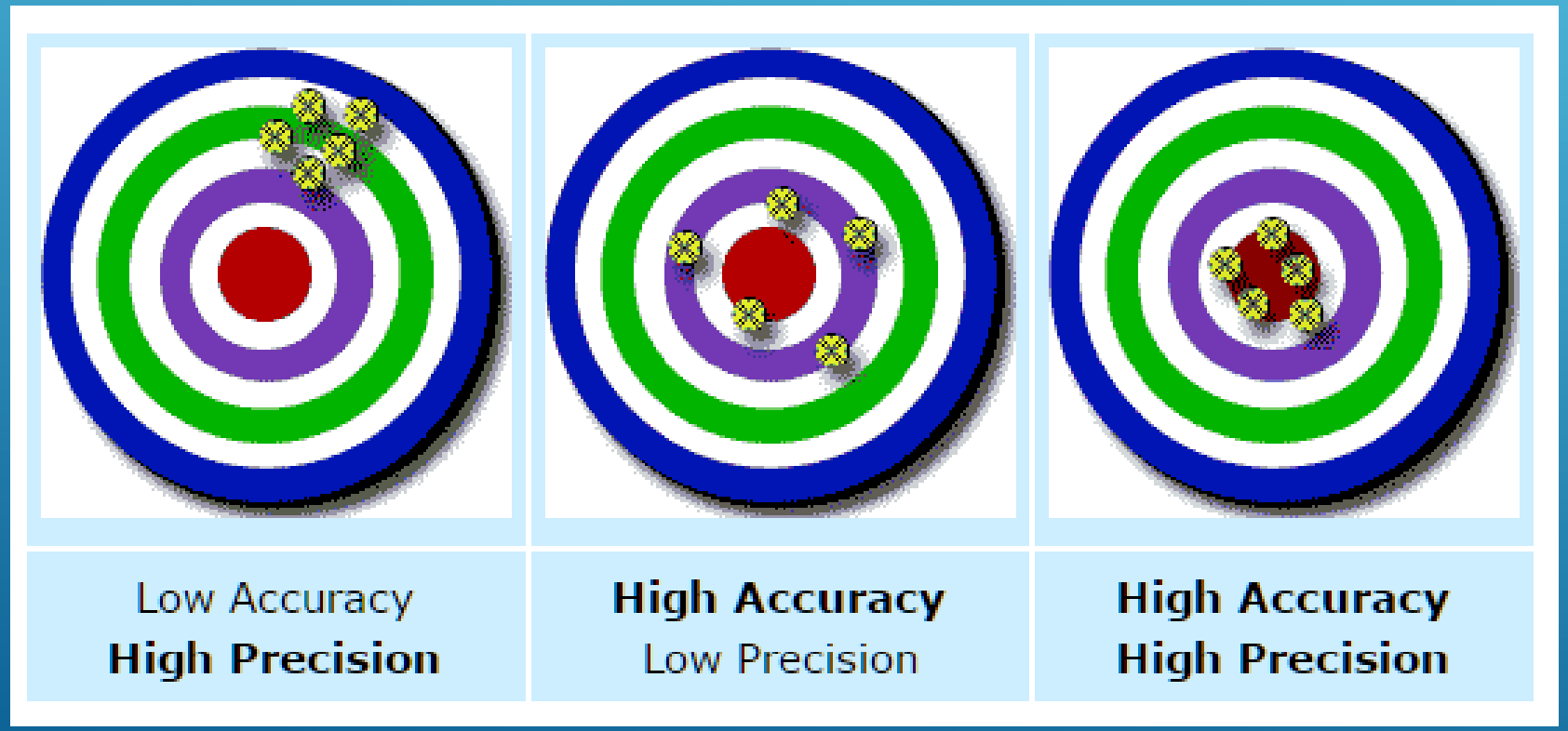
- The data used by a machine learning algorithm to train a model

Test Data (20% of data)

- The data used for evaluating the predictive performance of a model

Accuracy vs Precision

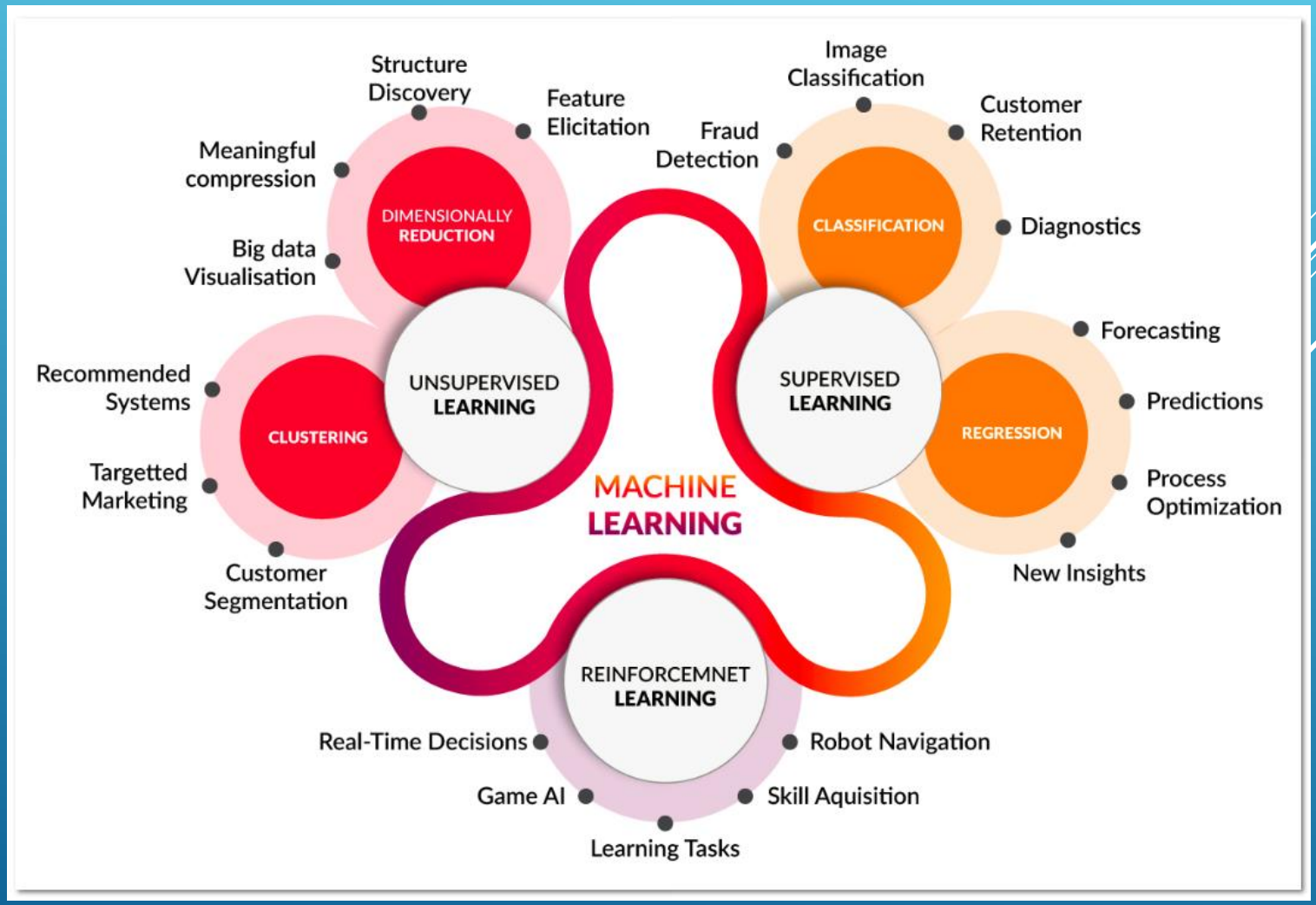
- **Accuracy** is how close a measured value is to the actual (true) value.
- **Precision** is how close the measured values are to each other.



Machine learning techniques

Machine learning mainly has three types of learning techniques:

- Supervised learning
- Unsupervised learning
- Reinforcement learning



Machine Learning tasks categories

1. Classification
2. Regression
3. Clustering
4. Anomaly detection
5. Association
6. Recommendation
7. Dimensionality reduction
8. Computer Vision
9. Text Analytics

Classification [supervised learning]

- Classification is concerned with building models that separate data into distinct classes. These models are built by inputting a set of training data for which the classes are pre-labelled in order for the algorithm to learn from. The model is then used by inputting a different dataset for which the classes are withheld, allowing the model to predict their class membership based on what it has learned from the training set.

Binary classification examples (divide data to two options only)

- spam filtering is a classification task
- Tumor diagnosis can be treated as a classification problem.
- determining credit risk using personal information such as income, outstanding debt

Multi-class classification examples

- handwritten recognition each character is a multi-class classification problem
- image recognition is a multi-class classification task
- Xbox Kinect360, which infers body parts and position

Regression [supervised learning]

- The goal is to predict a numerical label for an unlabeled observation

Regression algorithms: Linear regression, Decision trees

Examples

- home valuation
- Asset trading, and forecasting
- Sales or inventory forecasting

Features					Label
Position	Experience	Skill	Country	City	Salary ($)
Developer	0	1	USA	New York	103100
Developer	1	1	USA	New York	104900
Developer	2	1	USA	New York	106800
Developer	3	1	USA	New York	108700
Developer	4	1	USA	New York	110400
Developer	5	1	USA	New York	112300
Developer	6	1	USA	New York	114200
Developer	7	1	USA	New York	116100
Developer	8	1	USA	New York	117800
Developer	9	1	USA	New York	119700
Developer	10	1	USA	New York	121600

Anomaly Detection [supervised learning]

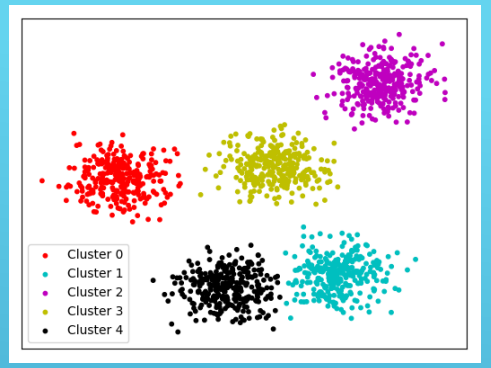
the goal is to find outliers, noise, deviations in a dataset

Anomaly detection applications

- In manufacturing, it is used for automatically finding defective products.
- In data centers, it is used for detecting bad systems.
- Websites use it for fraud detection.
- Detecting security attacks. Network traffic associated with a security attack is unlike normal network traffic. Similarly, hacker activity on a machine will be different from a normal user activity.

Clustering [unsupervised learning]

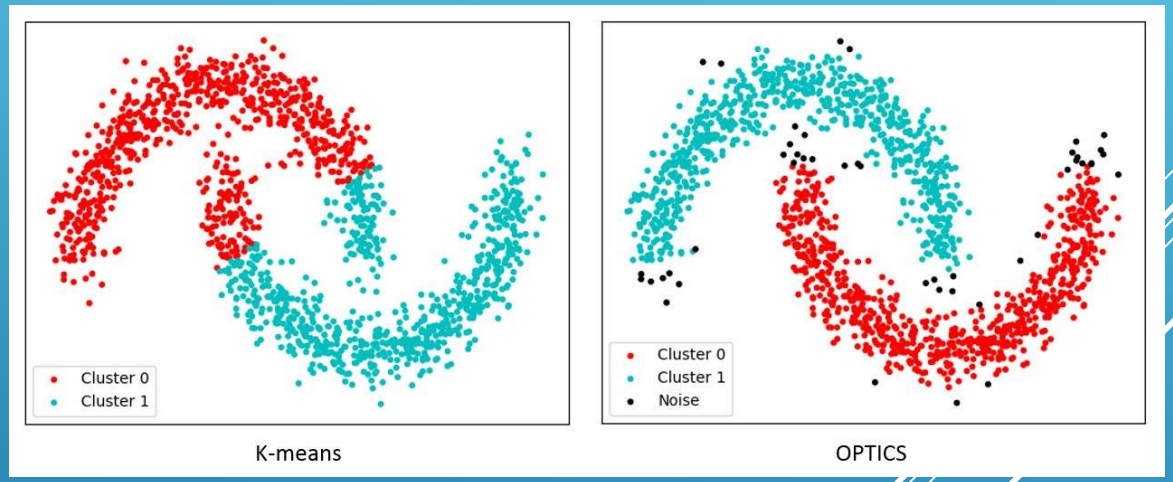
- The aim is to split a dataset into a specified number of clusters or segments.
- Elements in the same cluster are more similar to each other than to those in other clusters.



Clustering algorithms

1) k-means

- The number of clusters (k) must be given explicitly.
- Identify the best k cluster centers in an iterative manner
- Clusters are assumed to be spherical.



2) OPTICS /DBSCAN

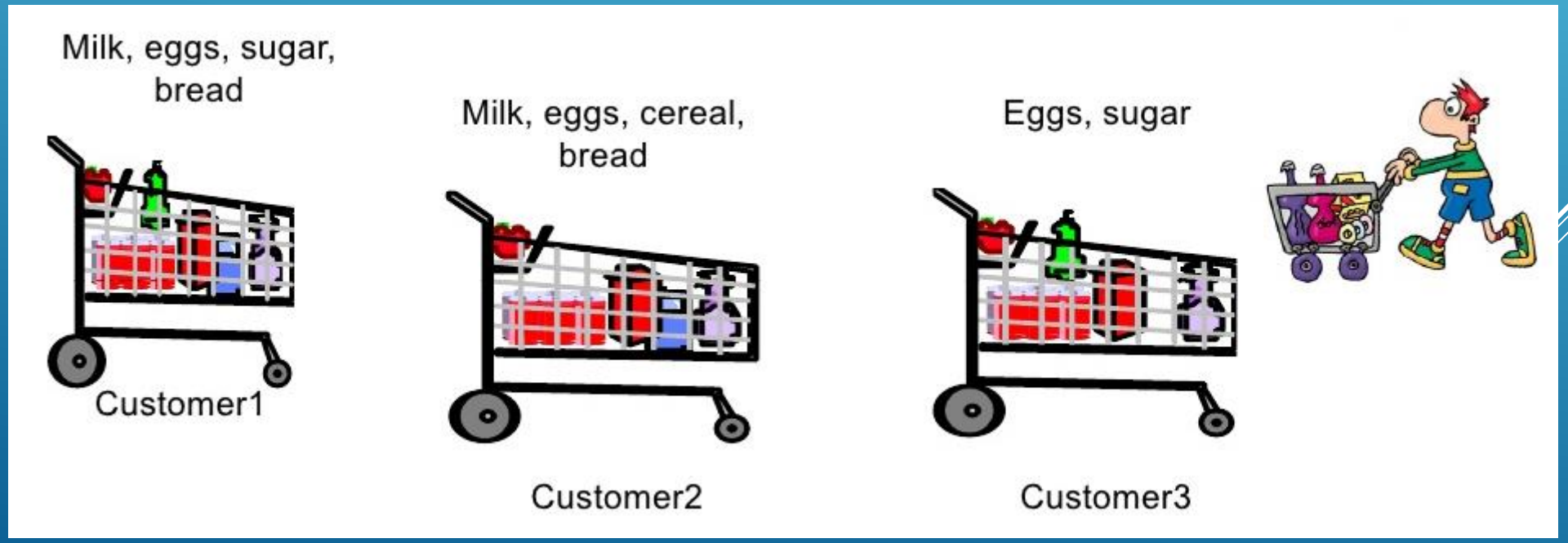
- it is a density-based clustering algorithm. represents clusters by its nature

Example

- creating customer segments, which can be targeted with different marketing programs

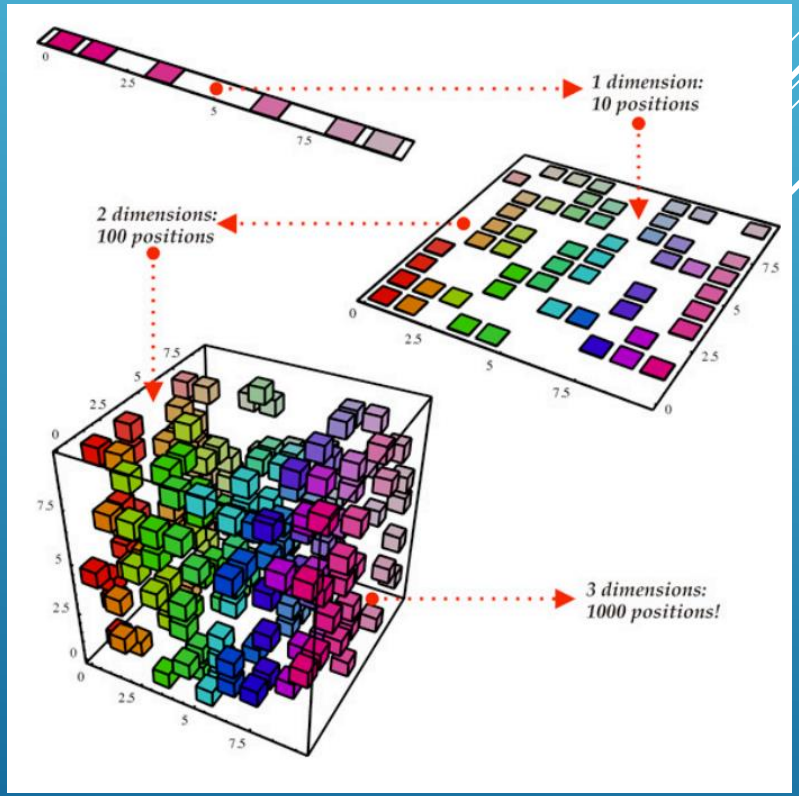
Association [unsupervised learning]

- Association is most easily explained by introducing market basket analysis, a typical task for which it is well-known.
- attempts to identify associations between the various items that have been chosen by a particular shopper and placed in their market basket and assigns support and confidence measures for comparison.
- The value of this lies in cross-marketing and customer behavior analysis.



Dimensionality Reduction [unsupervised learning]

- The goal in dimensionality reduction is to reduce the number of features in a dataset without significantly impacting the predictive performance of a model and this will reduce the computational complexity and cost of machine learning.



Recommendation [Reinforcement]

- The goal of a recommendation system is to recommend a product to a user based on past behavior to determine user preferences.
- Unlike Association, Recommendation focus on user behavior and suggest according to this user behavior.
- It is reinforcement because we not sure of result until user choose one of our recommendation, if not, then our recommendation was not correct.

Supervised **machine learning algorithms**

Classification

- **Two Class Classification**
  - Logistic Regression (Fast)
  - Decision Tree (Fast)
  - Decision jungle(Accurate)
  - SVM (Accurate) (>100 features)
  - Boosted Decision Tree (Fast - Large memory)
  - Bayes point machine (Fast)
- **Multi Class Classification**
  - Decision Tree (Fast)
  - Logistic Regression (Fast)
  - Random Forest (Accurate)
  - Gradient Boosting Tree (Accurate)
  - Naive Bayes (Big Data)
  - Decision jungle(Accurate)

Regression

- Linear Regression (Fast)
- Decision Tree (Fast)
- Random Forest (Accurate)
- Gradient Boosting Tree (Accurate)
- Ordinal regression
- Bayesian linear regression
- Boosted Decision Tree (Fast - Large memory)
- SGD Regressor (<100K rows)
- Lasso/ ElasticNet (Few Features)
- RidgeRegression
- SVR(kernel=linear/ rbf)
- EnsembleRegressors

Anomaly Detection

- One Class SVM (support vector machine)
- PCA based Anomaly Detection
- Time Series Anomaly Detection

Unsupervised **machine learning algorithms**

Clustering

- K-means Clustering
- K-modes (Categorical variables)
- DBScan (predict the number of clusters automatically)
- OPTICS (predict the number of clusters automatically)

Association

Apriori

Dimension Reduction

- PCA
- Singular value decomposition

Recommendation

- Matchbox Recommender

Computer Vision

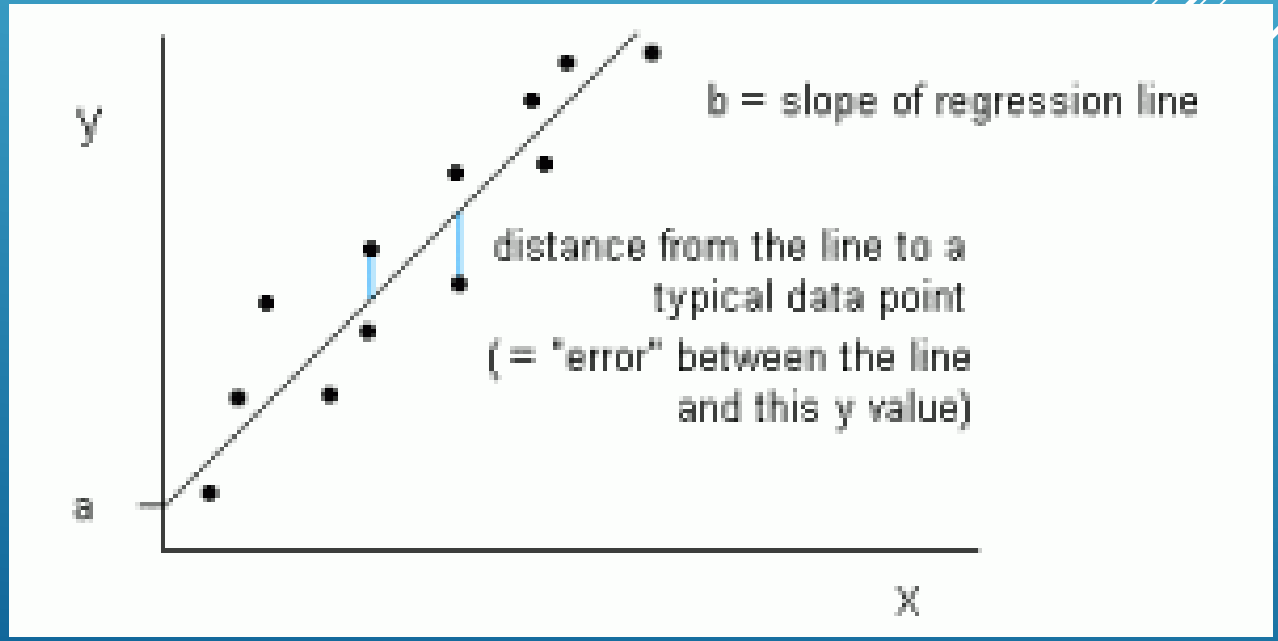
- OpenCV Library

Text Analytics (Supervised Learning)

- Named Entity Recognition
- Sentimental Analysis

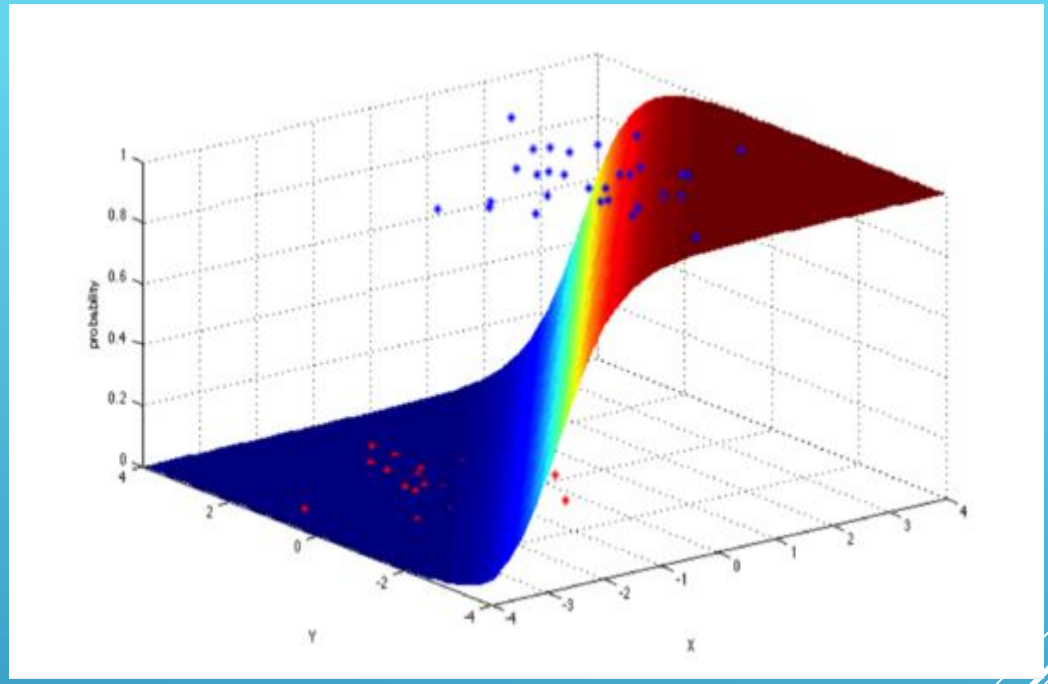
Linear regression algorithms

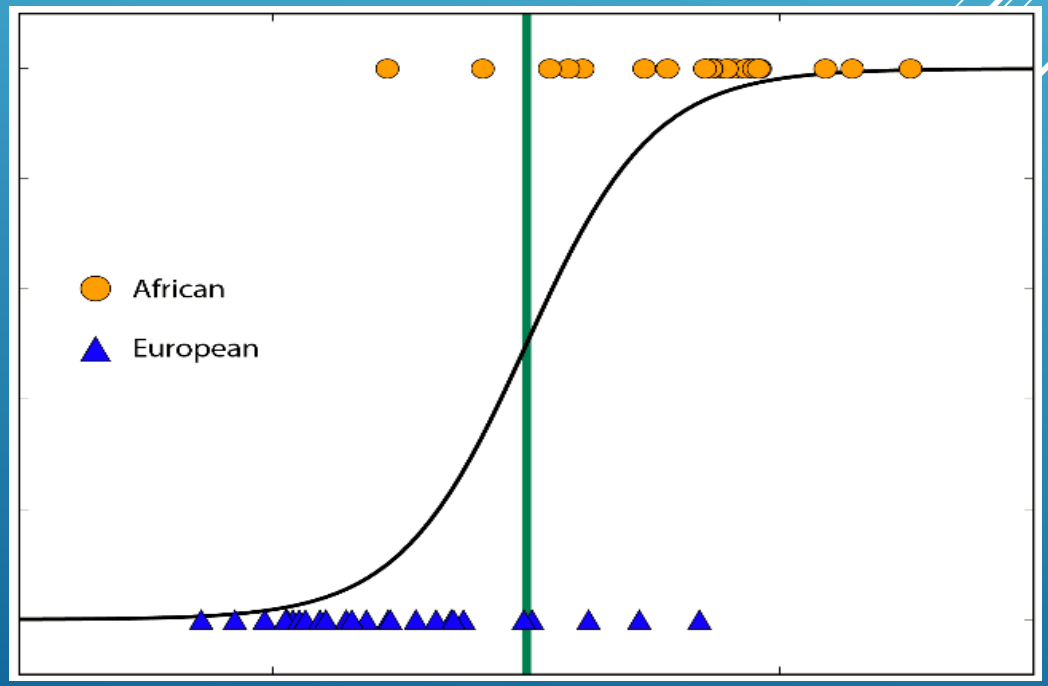
Linear Regression is used in problems where the label is of continuous nature e.g. Sales of a retail chain. It consists of ordinary least squares method fitting the best line that minimizes the sum of squared errors between the predicted and actual data points. Such algorithms are mostly used to for decision-making process to solve problems like – What should be my best marketing mix to optimize sales given the investment in various marketing channels or maximizing sales based on store layouts or pricing and promotional offers.



Logistic regression algorithms

Logistic Regression: When the label is that of categorical or discrete nature, we use log odds ratio to optimize business problems such as – scoring customers and then predict those that are most likely to default on the loan payment, or predict higher response rates to a certain marketing strategy.





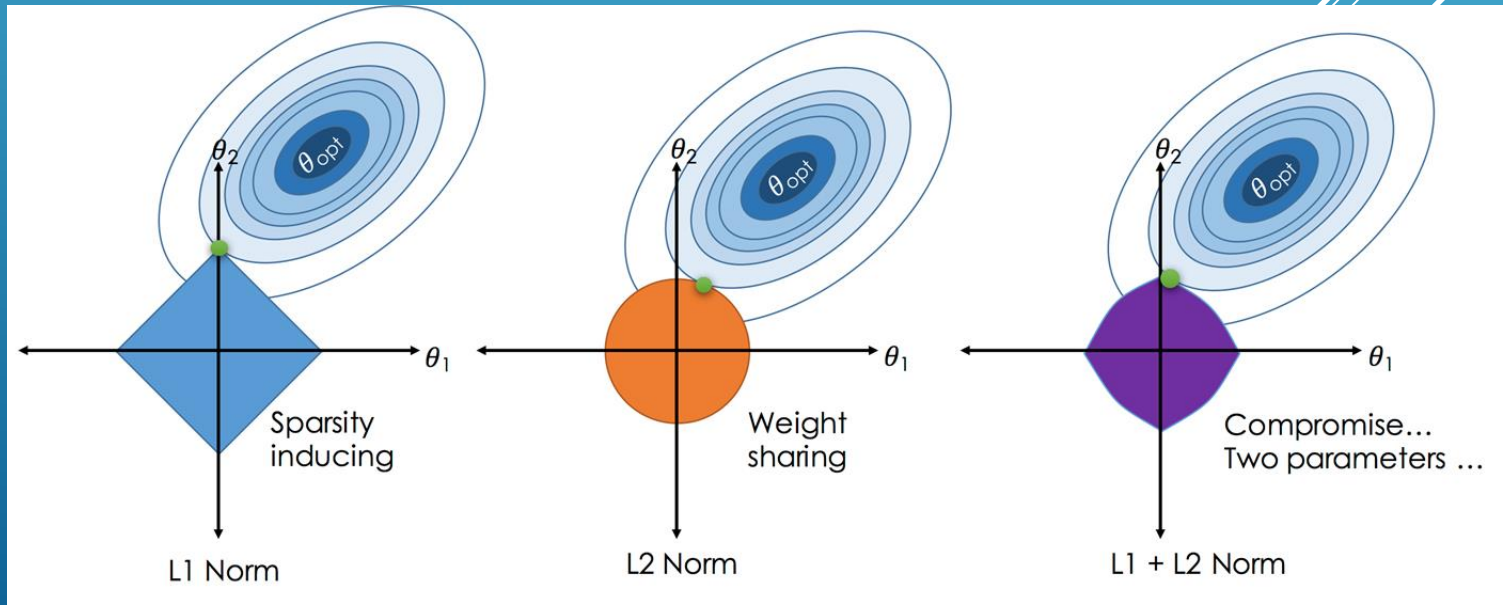
Ridge, Lasso, and Elasticnet algorithms

Ridge, Lasso, and Elasticnet are Best Linear Unbiased Estimator (BLUE)

- To find a BLUE estimator just need mean and variance
- BLUE restricts the estimator to be linear
- BLUE use norms, norms play a vital role in reducing the Mean Squared Error (MSE)

Ridge vs Lasso vs Elasticnet

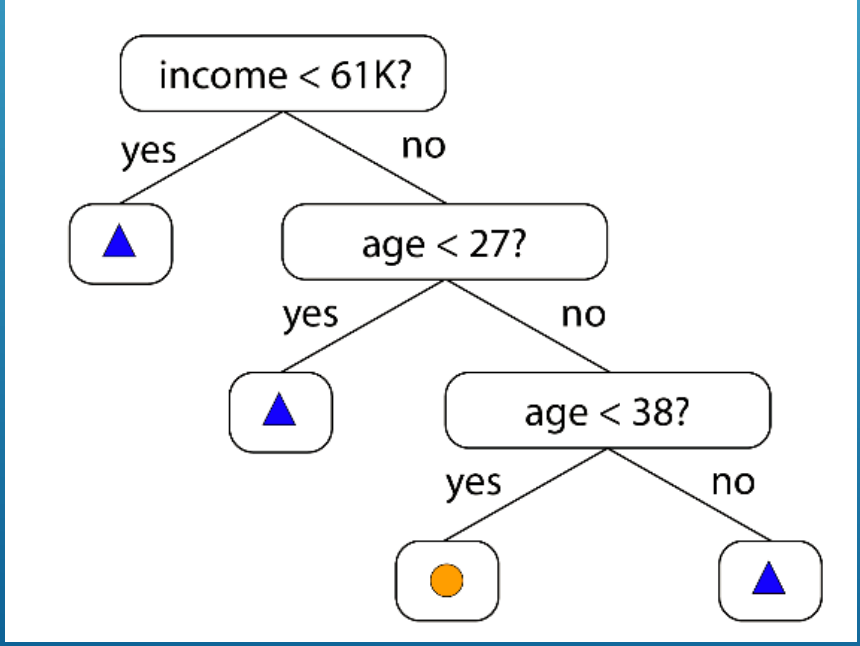
- **Ridge** uses L2 Norm penalty term which limits the size of the coefficient vector.
- **Lasso** uses L1 Norm penalty which imposes sparsity among the coefficients and thus, makes the fitted model more interpretable.
- **Elasticnet** has a penalty which is a mix of L1 and L2 norms.

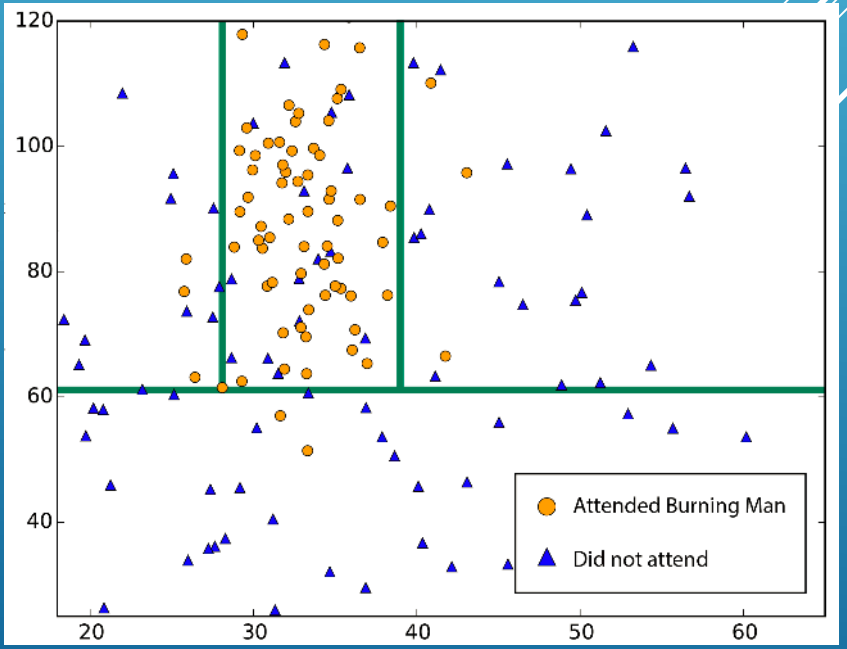


Decision Trees, forests, and jungles

Decision Trees methods construct a tree of predictive decisions made based on actual values of attributes in the data. It used for classification and regression problems.

Decision Trees, forests, and jungles: all do the same thing - subdivide the feature space into regions with mostly the same label. These can be regions of consistent category or of constant value, depending on whether you are doing classification or regression.





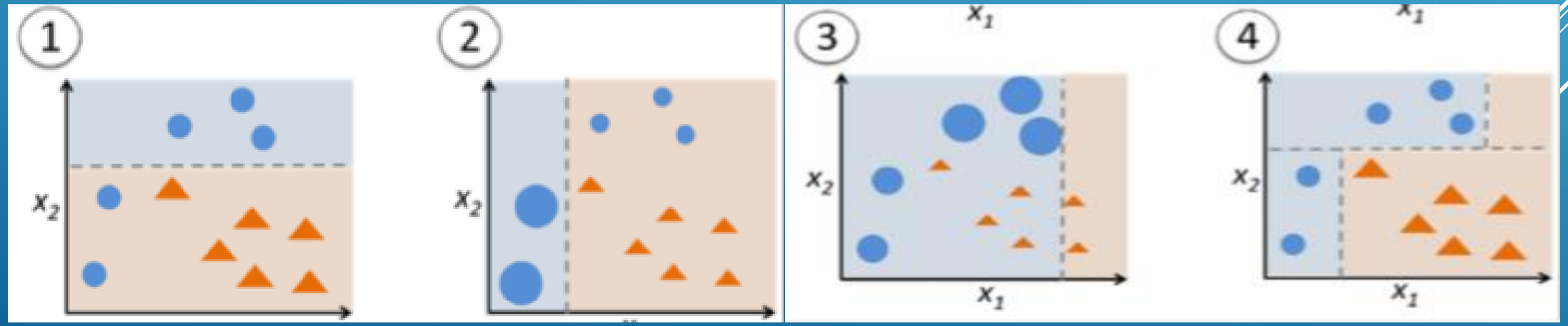
Ensemble Learning Algorithms (Gradient-Boosted/ Random Forests)

Decision trees do not have the same level of predictive accuracy.

This problem is addressed by using a collection of decision trees instead of just one decision tree.

this algorithm called ensemble learning algorithms and it is one of top performers for classification and regression tasks.

The commonly used ensemble algorithms are Random Forests and Gradient-Boosted Trees



Random Forests

Random Forest algorithm trains each decision tree in an ensemble independently using a random sample of data. In addition, each decision tree is trained using a subset of the features. The number of trees in an ensemble is of the order of hundreds.

Random Forest creates an ensemble model that has a better predictive performance than that of a single decision tree model.

For a regression task, a Random Forest model takes an input feature vector and gets a prediction from each decision tree in the ensemble. It averages the numeric labels return by all the trees and returns the average as its prediction.

Gradient-Boosted Trees

Gradient-Boosted Trees (GBTs) algorithm also trains an ensemble of decision trees. However, it sequentially trains each decision tree.

It optimizes each new tree using information from previously trained trees. Thus, the model becomes better with each new tree.

GBT can take longer to train a model since it trains one tree at a time. In addition, it is prone to overfitting if a large number of trees are used in an ensemble.

However, each tree in a GBT ensemble can be shallow, which are faster to train.

Random Forests vs Gradient-Boosted Trees (GBTs)

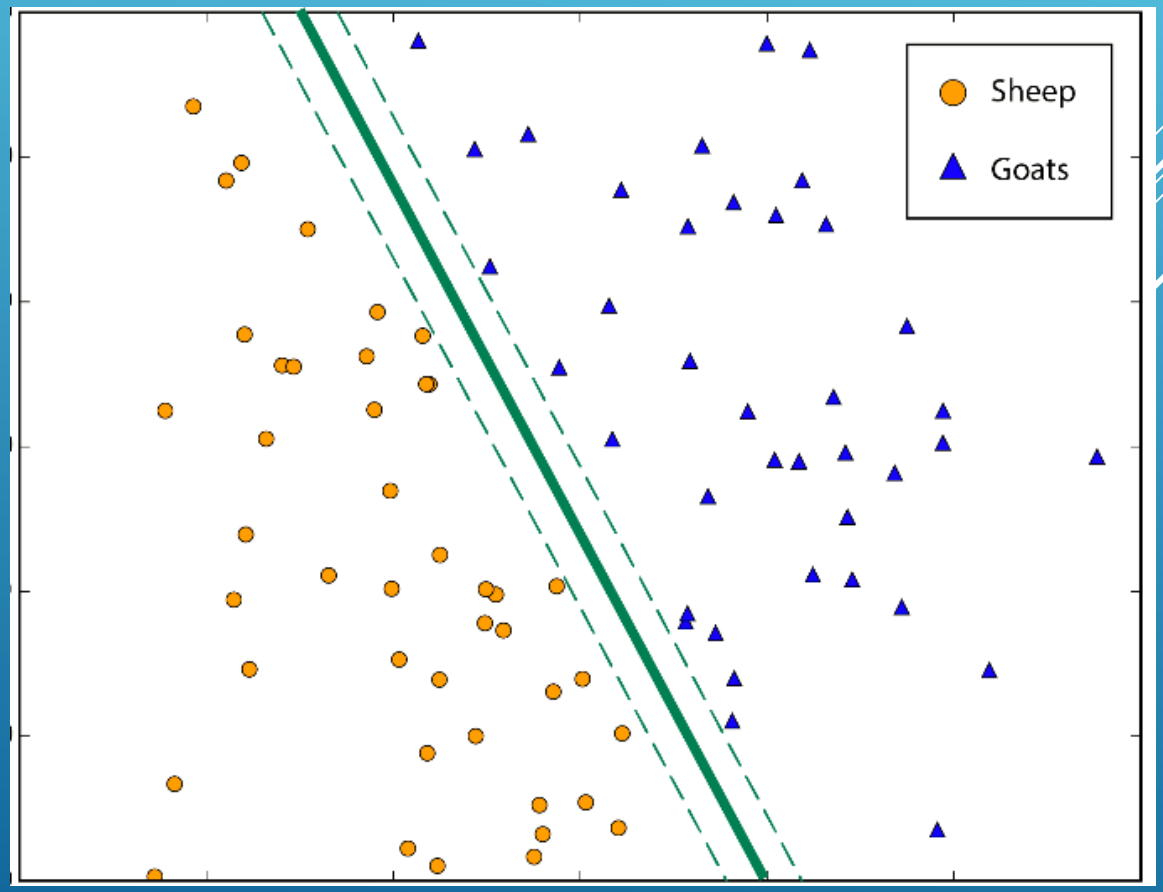
- Random Forests can train multiple trees in parallel, but GBTs train one tree at a time
- Random Forest reduces the likelihood of overfitting
- Random Forests reduce variance by using more trees, whereas GBTs reduce bias by using more trees

To make a prediction on a new instance, a random forest must aggregate the predictions from its set of decision trees. This aggregation is done differently for classification and regression.

- *Classification*: Majority vote. Each tree's prediction is counted as a vote for one class. The label is predicted to be the class which receives the most votes.
- *Regression*: Averaging. Each tree predicts a real value. The label is predicted to be the average of the tree predictions.

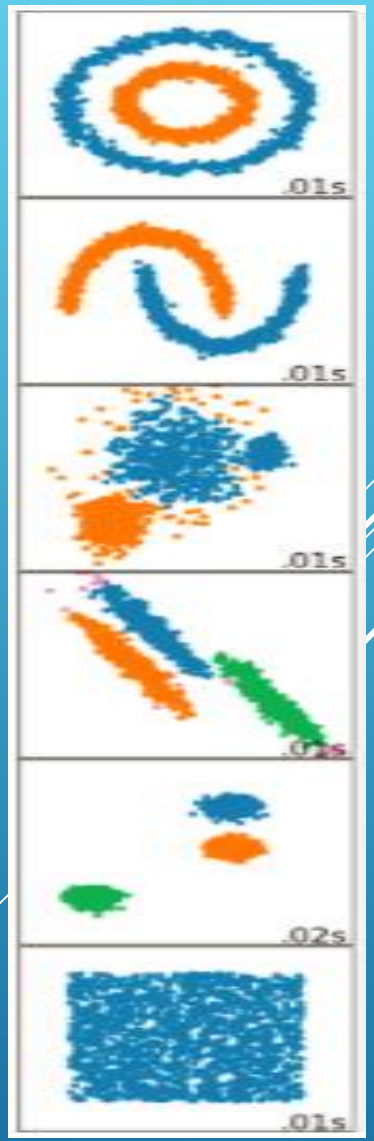
SVM

Support Vector Machines (SVM) are most popularly used in ML to deal with problems related to image segmentation, the stock market, text categorization and biological sciences, it try to find the line that give largest margin between two classes of observations.



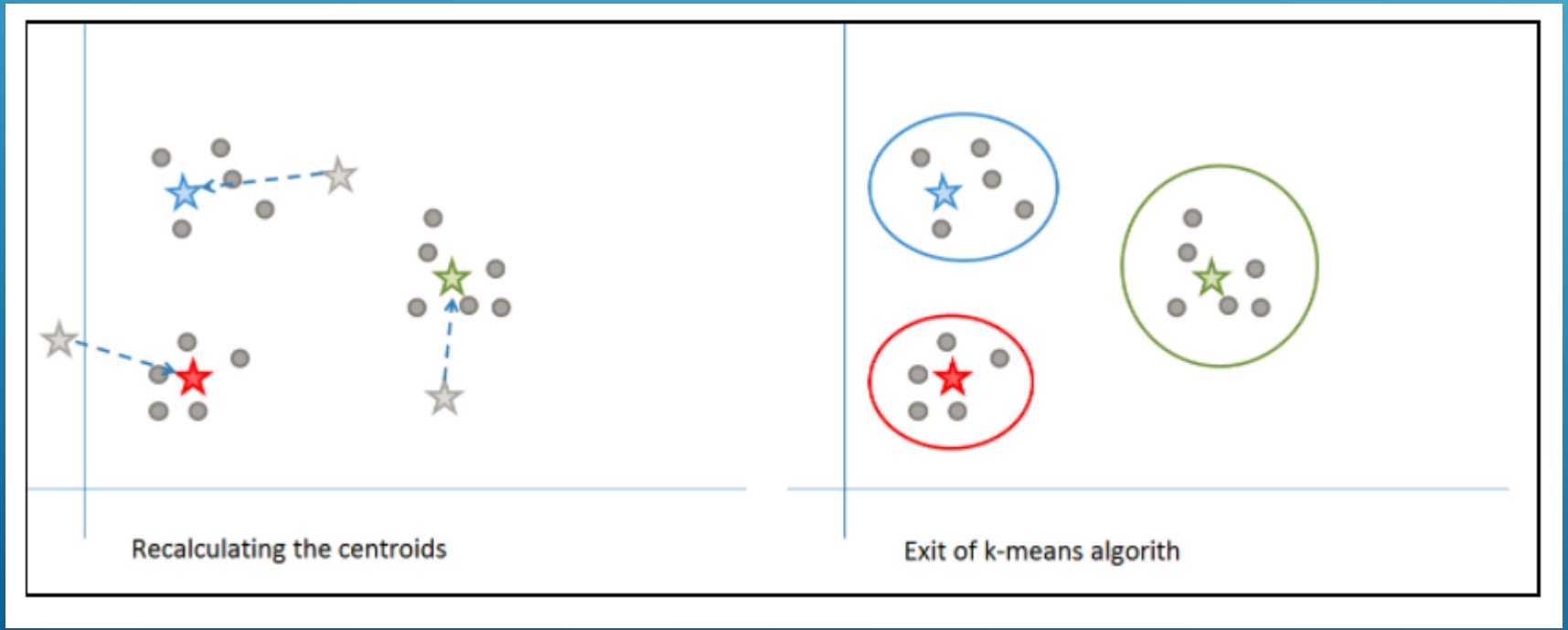
DBScan / Optics

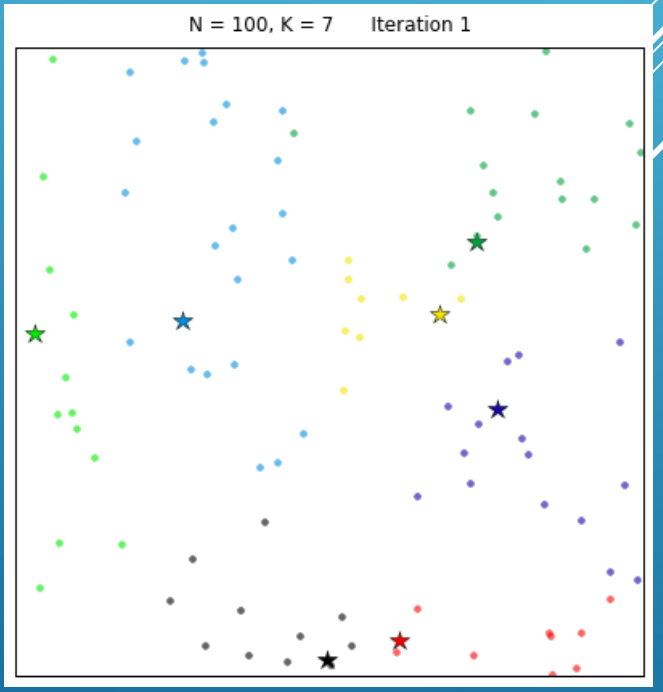
- It is a Clustering algorithm that detect the continues patterns
- The distance between points in DBScan are fixed. So, it is fast but not work well with Clusters with different densities.
- Optics slower than DB Scan, identify clusters with different densities



K-means

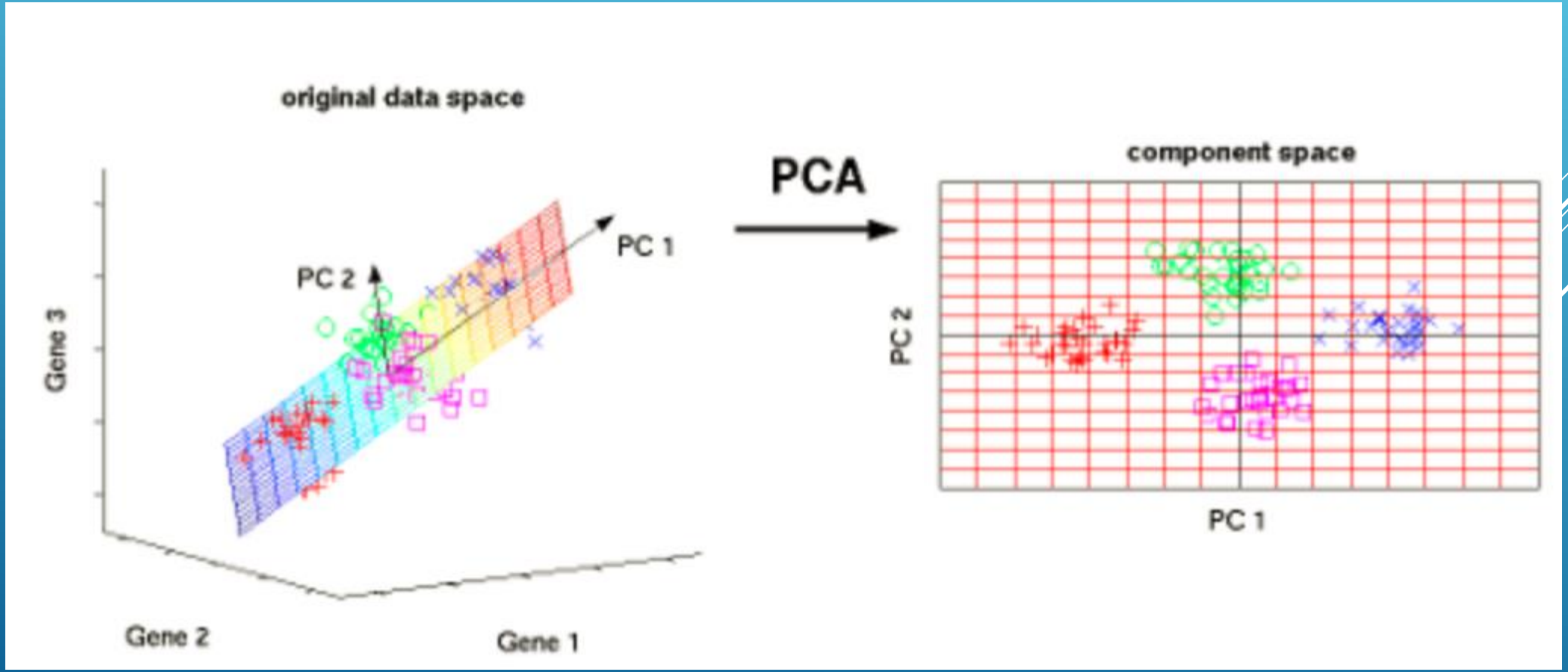
- Clustering/ K-Means: This is a undirected or unsupervised data mining activity typically seen to be used in problems that involve market segmentation, fraud detection, recommendation engines, clustering web pages by similarity.





PCA

- Used for dimension reduction



Naïve Bayes

- Naïve Bayes mostly used in Real time Prediction , Multi class Prediction , Text classification, sentiment analysis, document categorization, spam filtering, disease prediction, Recommendation System.
- Naïve Bayes Classifier is based on the Bayes Theorem of Probability and assumes independence of attributes.
- Independence of attributes example:
  a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive' =(child like/inexperienced).

Naive Bayes Pros:

- It is easy and fast to predict class/multi class prediction
- Performs better compare to other models, need less training data.
- It perform well in case of categorical input variables compared to numerical variable(s). For numerical variable, normal distribution is assumed

Naive Bayes Cons:

- If categorical variable has a category (in test data set), which was not observed in training data set, then model will assign a 0 (zero) probability and will be unable to make a prediction.
- Naive Bayes is also known as a bad estimator, so the probability outputs from predict_proba are not to be taken too seriously.
- In real life, it is almost impossible to get a set of predictors which are completely independent.

There are three types of Naive Bayes model

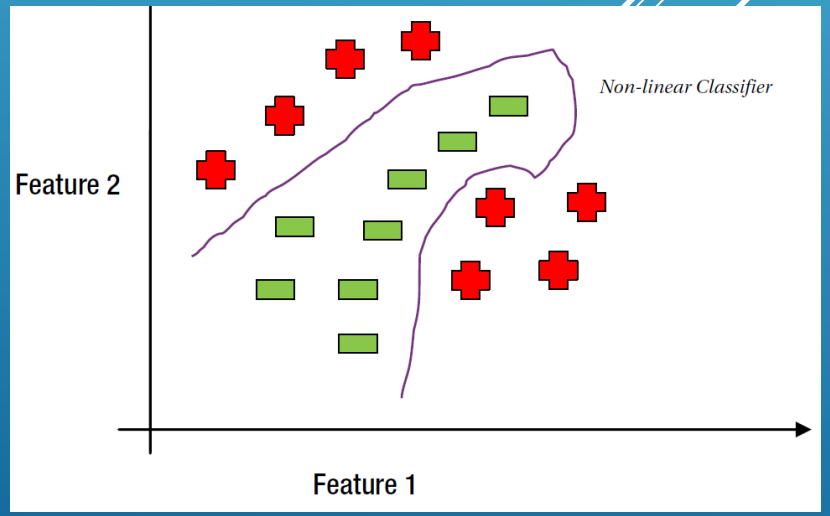
- **Gaussian**: It is used in classification and it assumes that features follow a normal distribution.

- **Multinomial**: It is used for discrete counts. For example, in text classification problem. "count how often negative word occurs in the document"

- **Bernoulli**: The binomial model is useful if your feature vectors are binary (i.e. zeros and ones).

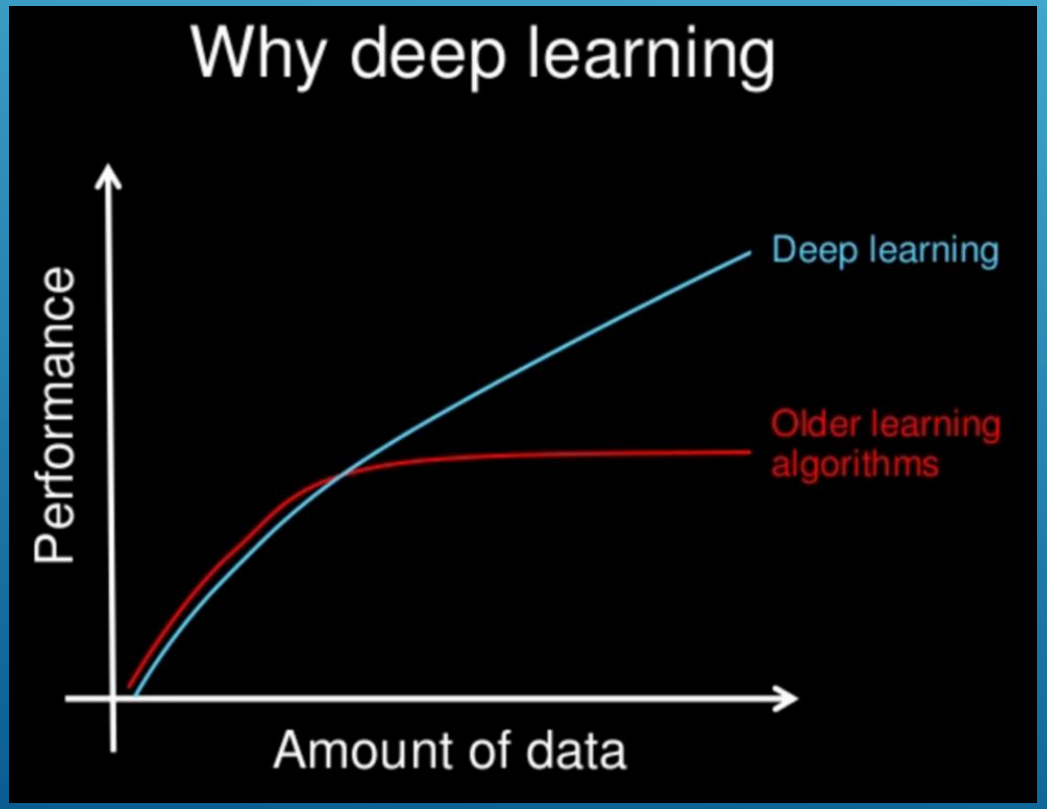
Neural Network

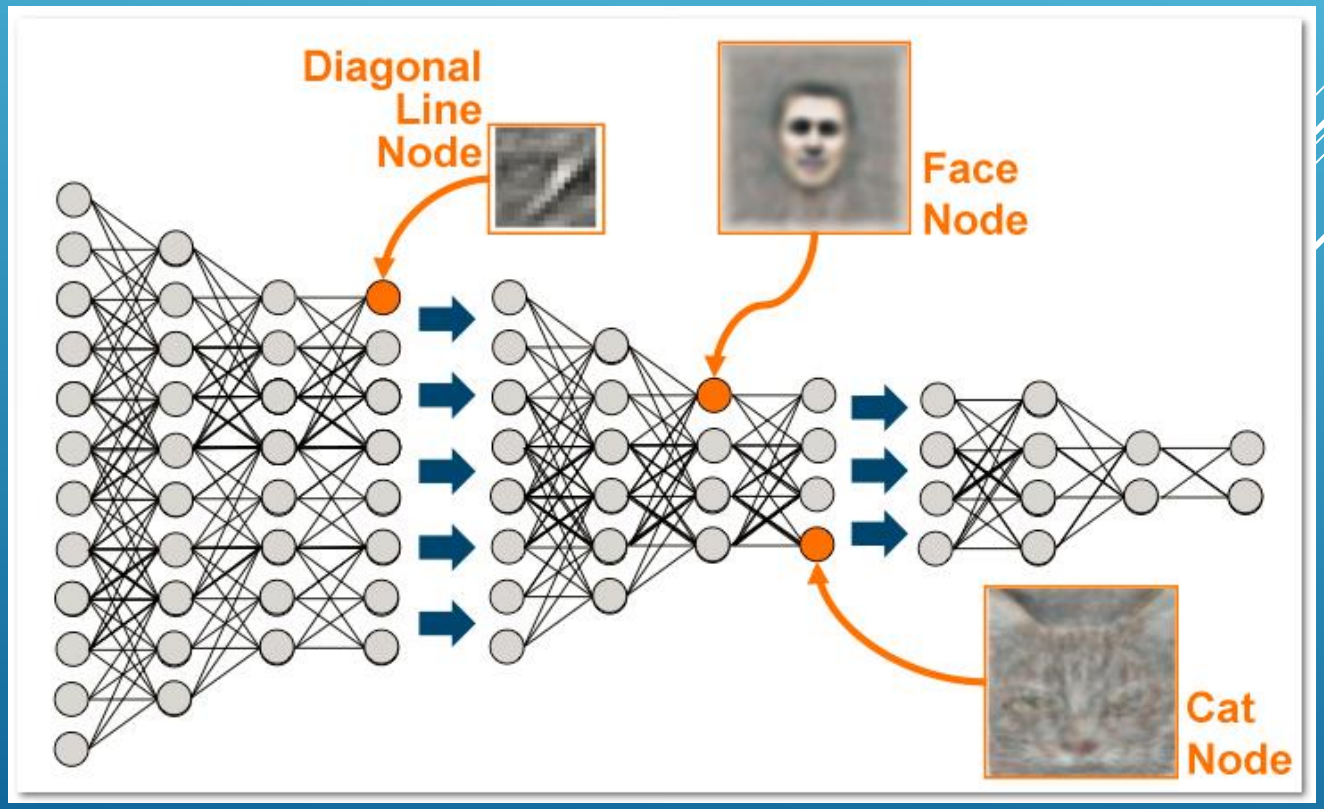
- The feedforward neural network algorithm uses a technique known as backpropagation to train a model. During the training phase, prediction errors are fed back to the network. The algorithm uses this information for adjusting the weights of the edges connecting the nodes to minimize prediction errors. This process is repeated until the prediction errors converge to value less than a predefined threshold.
  Generally, a neural network with one layer is sufficient in most cases. If more than one hidden layers are used, it is recommended to have the same number of nodes in each hidden layer.
  Neural networks are better suited for classifying data that is not linearly separable.



Deep learning

- It is consists of many layers/nodes of neural networks.
- Deep learning automatically finds out the features which are important for classification, but in Machine Learning we had to manually give the features
- deep learning algorithms need a large amount of data to work perfect.



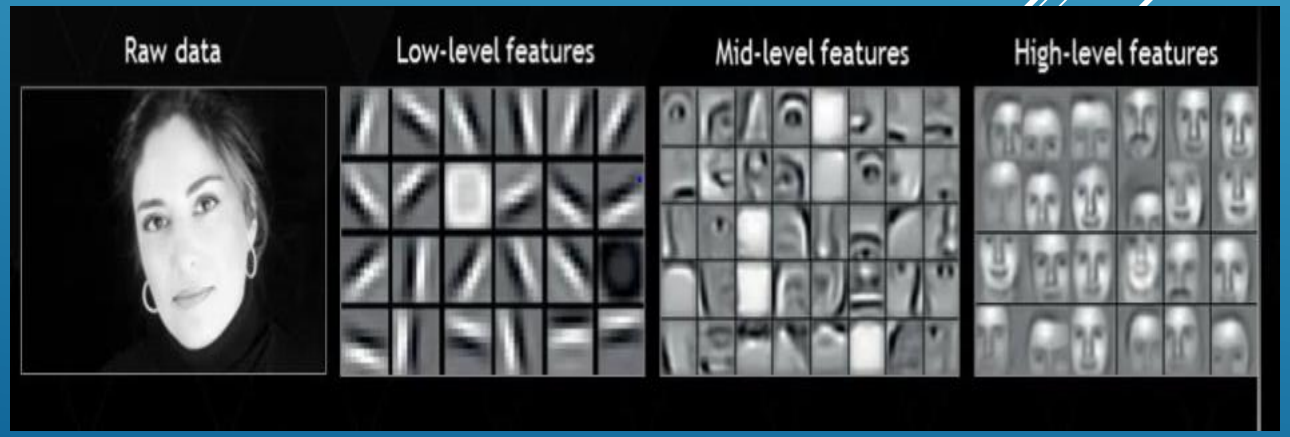


Traditional ML Algorithms

- can work on low-end machines
- break the problem down into different parts, solve them individually and combine them to get the result.
- takes much less time to train, ranging from a few seconds to a few hours.
- gives us the selection rules, so it is easy to interpret, safe to use in industry for interpretability.

Deep learning algorithms

- need a large amount of data to work perfect
- heavily depend on GPUs
- solve the problem end-to-end.
- takes a long time to train. (like two weeks)
- It is excellent and is near human performance, but no guarantee to be always like this!, because most of operations are hidden and we can't review selection logic!



No Free Lunch

- In machine learning, there's something called the "No Free Lunch" theorem. In a nutshell, it states that no one algorithm works best for every problem.
- As a result, one should **try many different algorithms for the problem**, while using a hold-out "test set" of data to evaluate performance and select the winner.

PYTHON MACHINE LEARNING

Why Python?

- A **large community**
- **tons of machine learning specific libraries**
- **easy to learn**
- **TensorFlow** make Python the **leading language** in the data science community.

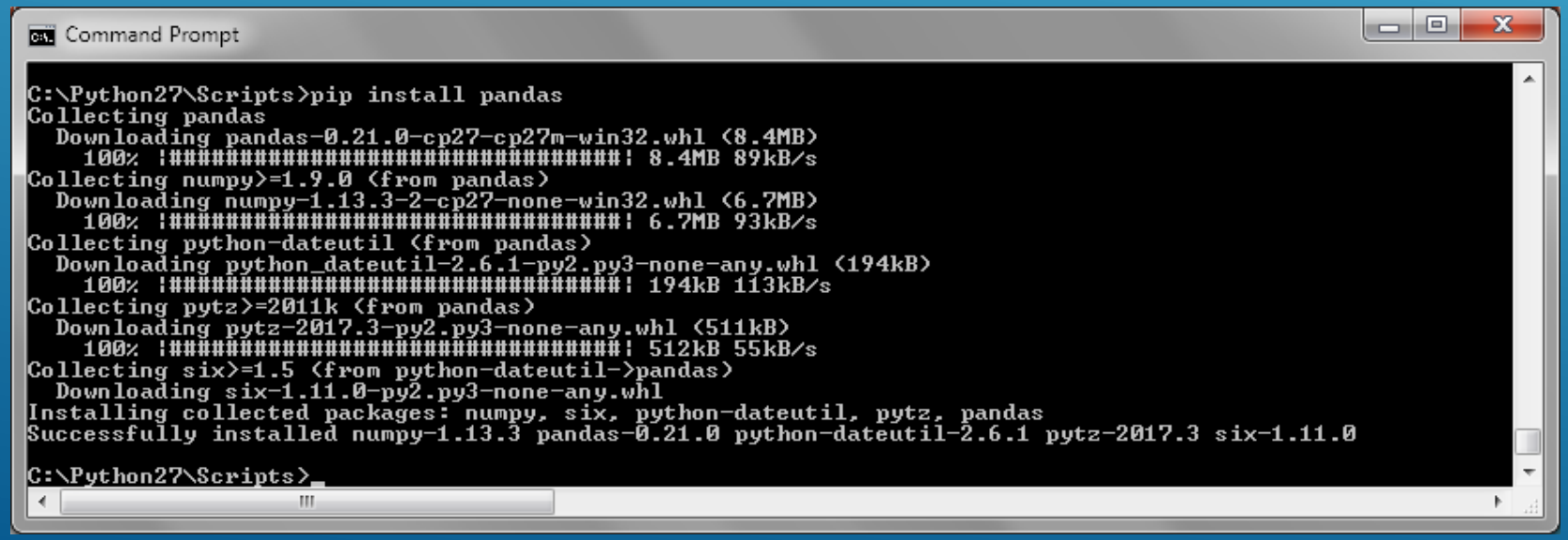
About Python

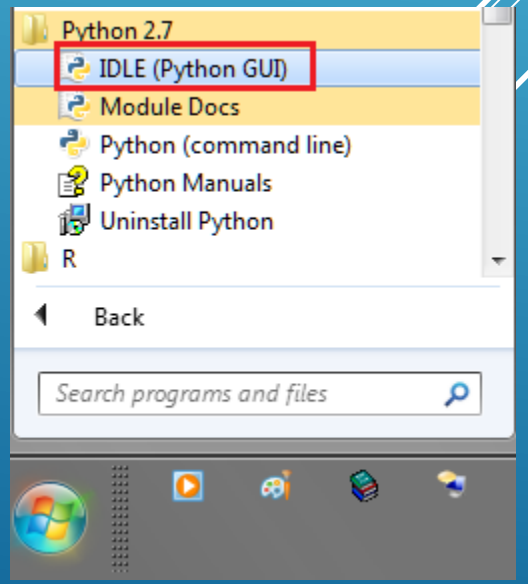
- It is case sensitive
- Text indentation is important in logic flow!
- Use **#** symbol to add a code comment
- Use ‘’’ ‘’’ to comment a block of code

Prepare your machine

Although you can develop machine learning algorithm directly on IBM cloud, but you can use your machine also to develop some simple tasks using the next simple steps.

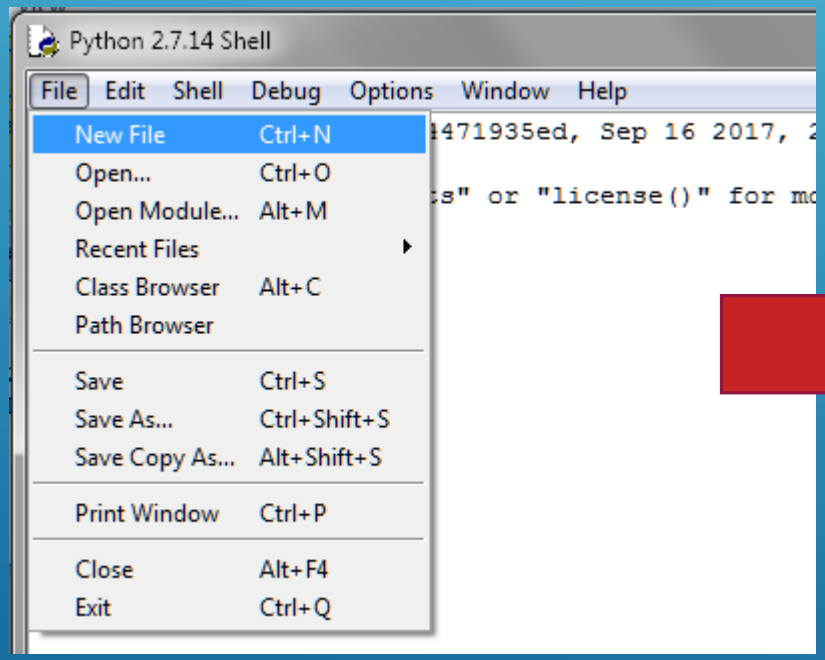
- Install **Python 2.7.xx**
- Pip install pandas
- Pip install matplotlib Sklearn Statistics scipy seaborn Ipython
  pip install --no-cache-dir pystan #to force redownload exist package in case of error
- To get sample Datasets for training check http://archive.ics.uci.edu/ml/datasets.html

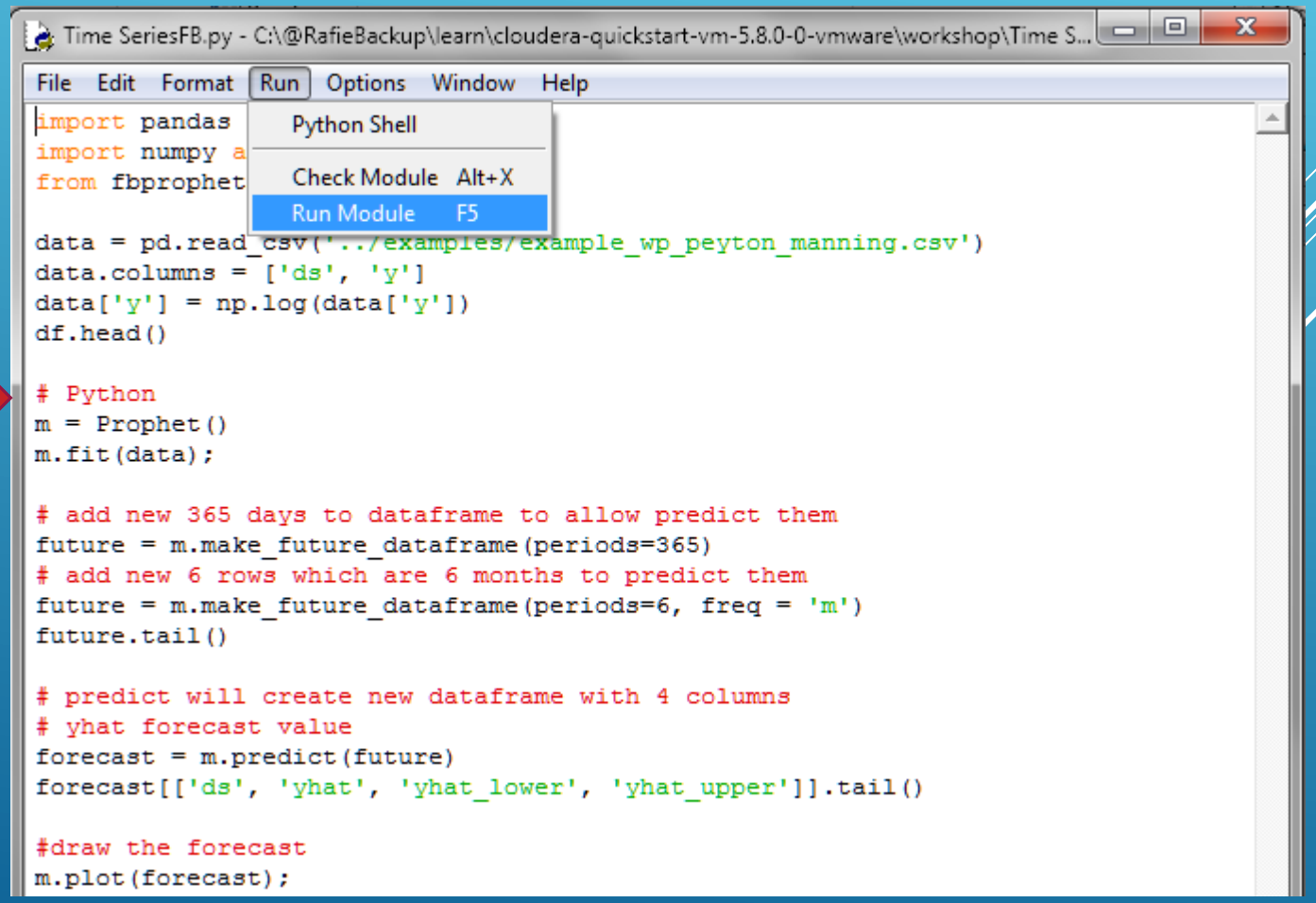




How to Run Python ML application

- Open "IDLE (Python GUI)" → File → New File
- Write your code on the new window then → Run → Run Module





Alt+3 → Comment lines
Alt+4 → Remove lines comments

Anaconda

- Anaconda is Python distribution for data scientists .
- It has around 270 packages including the most important ones for most scientific applications, data analysis, and machine learning such as NumPy, SciPy, Pandas, IPython, matplotlib, and scikit-learn.
- Download it for free from https://www.continuum.io/downloads

Import Statistical libraries

```
import time
import random
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import statistics
from scipy import stats
import sklearn
import seaborn
from IPython.display import Image
import numpy as np
```

import pandas as pd

- *Pandas* ***DataFrame*** similar to *Python* ***lists*** but it has extra column (index) and Any operation to perform on a DataFrame , get's performed on every single element inside it.
- Example of convert python list to DataFrame →

```
baloons_data = [
            {'red' : 1, 'blue' : '2'},
            {'blue' : 4, 'green' : '1'},
            {'red' : 2, 'blue' : '4'}
        ]
```

```
pd.DataFrame(baloons_data, index)
```

	blue	green	red
0	2	NaN	1.0
1	4	1	NaN
2	4	NaN	2.0

The missing values are dennoted by NaN __

Common Operations

1) Set dataset columns names → **data.columns = ['blue','green','red']**
2) Get value of any cell use cell index and column index/name like this : **data.ix[2,'red']**
3) replace NaN with zero using : **data = data.replace(np.nan,0)**
4) Select a subset of columns in new dataFrame : **NewDF= data [['blue' , 'red']]**
5) Count of rows that has data on a column: **data['red'].value_counts().sum()**
6) Select count of distinct value on a column: **data['blue'].nunique()**
7) Drop duplicates rows: **data = data.drop_duplicates()**

Basic commands

Reading the Data from disk into Memory

- data = pd.read_csv('examples/trip.csv')

Reading the Data from HDFS into Memory

- with hd.open("/home/file.csv") as f:
- data = pd.read_csv(f)

Printing Size of the Dataset and Printing First/Last Few Rows

- print len(data)
- data.head()
- data.tail()
- data.describe() #get data summary count, mean, the min and max values

Order Dataset and get first and last element of a column

- data = data.sort_values(by= 'birthyear')
- data.reset_index()
- print data.ix[1, 'birthyear'] #first element
- print data.ix[len(data)-1, 'birthyear'] #last element

from_station_id	to_station_id	usertype	gender	birthyear
CBD-06	PS-04	Member	Male	1960.0
CBD-06	PS-04	Member	Male	1970.0
CBD-06	PS-04	Member	Female	1988.0
CBD-06	PS-04	Member	Female	1977.0
CBD-06	PS-04	Member	Male	1971.0

DataFrame cleaning

- If need to convert column with negative values
  `data['AwaitingTime'] = data['AwaitingTime'].apply(lambda x: abs(x))`
- Convert string values to integer values (DayOfTheWeek to Integers)
  `Day_mapping = {'Monday' : 0, 'Tuesday' : 1, 'Wednesday' : 2, 'Thursday' : 3, 'Saturday' : 5, 'Sunday' : 6}`
  `data['DayOfTheWeek'] = data['DayOfTheWeek'].map(Day_mapping)`
- Remove rows that has age column less than zero
  `data = data[data['Age'] >= 0]`
- Delete column from Dataset
  `del data['from_station_id']`

Filter data to remove rows that has NA/0

```
> df = pd.DataFrame({'a':[0,0,1,1], 'b':[0,1,0,1]})
➢  df = df[(df.T != 0).any()]        #keep row if any feature contain values
> print df
> df = df[(df.T != 0).all()]         #keep row if all features contain values
> print df
```

Original DataFrame

	a	b
1	0	0
2	1	0
3	0	1
4	1	1

Keep raw if ANY contains a value

	a	b
2	1	0
3	0	1
4	1	1

df[(df.T != 0).any()]

Keep raw if ALL contains values

	a	b
4	1	1

df[(df.T != 0).all()]

Drop the columns where Any/All elements are NaN

- data.dropna(how='any')

how : {'any', 'all'}

- any : if any NA values are present, drop that label
- all : if all values are NA, drop that label

Drop duplicate rows

Consider duplicate if all the row are the same, return result in new dataframe

- **dataframe1 = df.drop_duplicates(subset=None, inplace=False)**

Consider duplicate if Column1, Column3 are the same, drop duplicate except last row, return result in the same dataframe

- **df.drop_duplicates(subset=['Column1', 'Column3'], keep='last')**

Save unique data

- df.to_csv(file_name_output)

DataFrame.**drop_duplicates**(*subset=None, keep='first', inplace=False*)

Return DataFrame with duplicate rows removed, optionally only considering certain columns

Parameters:

subset : *column label or sequence of labels, optional*

Only consider certain columns for identifying duplicates, by default use all of the columns

keep : *{'first', 'last', False}, default 'first'*

- first : Drop duplicates except for the first occurrence.
- last : Drop duplicates except for the last occurrence.
- False : Drop all duplicates.

inplace : *boolean, default False*

Whether to drop duplicates in place or to return a copy

Returns:

deduplicated : *DataFrame*

Split Dataset, Create training, and test dataset

How to split dataset by 70-30?
[70% of the observations fall into training and 30% of observations fall into the test dataset.]

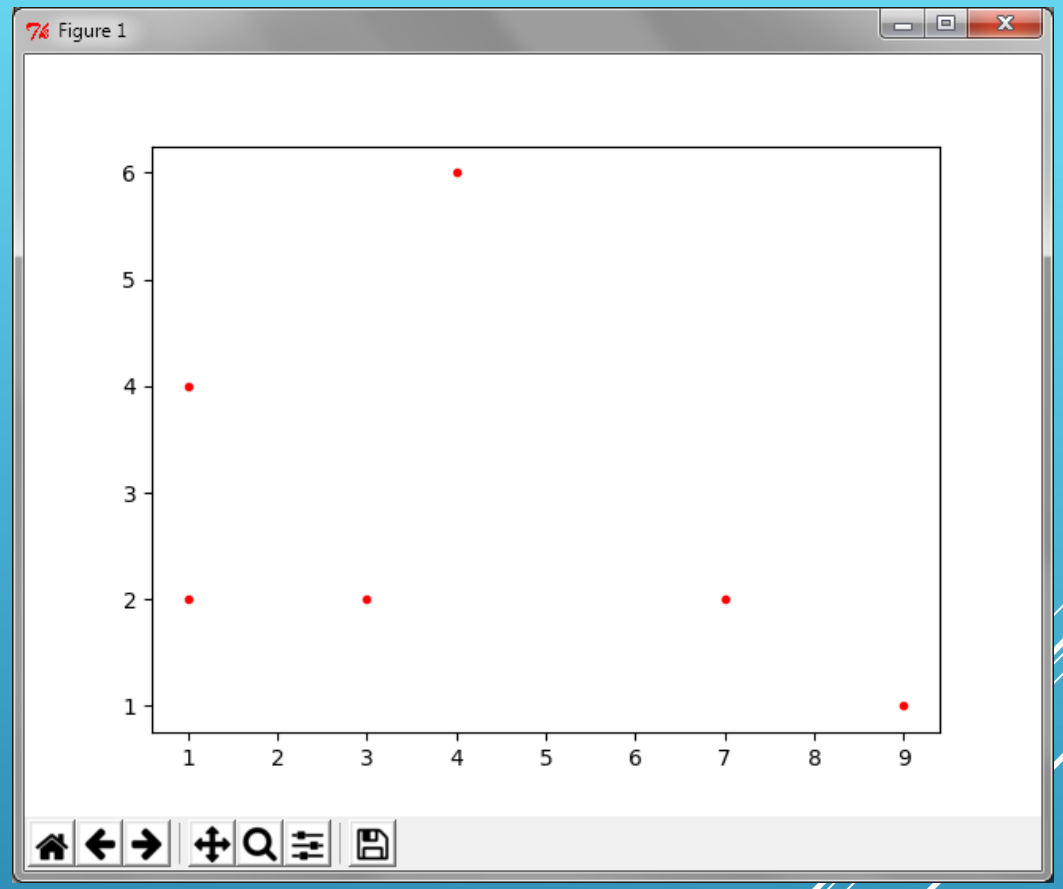
- x_train,x_test, y_train, y_test = train_test_split(data['x'],data['y'], test_size=0.3, random_state=1)

import matplotlib.pyplot as plt

- plt.bar – creates a bar chart
- plt.scatter – makes a scatter plot
- plt.boxplot – makes a box and whisker plot
- plt.hist – makes a histogram
- plt.plot – creates a line plot

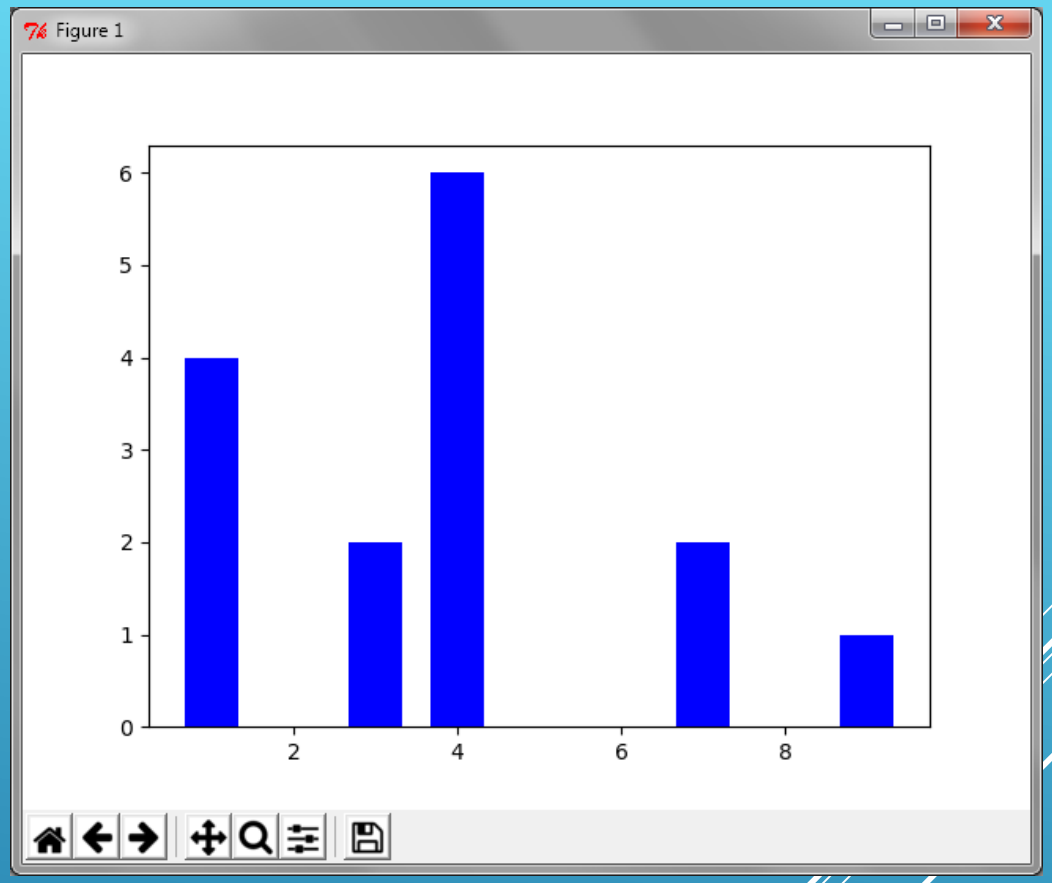
Plot for Data array

```
import numpy as np
import matplotlib.pyplot as plt

Data=np.array([ [1,2], [3,2], [4,6], [7,2], [1,4], [9,1] ])
Column1=0
Column2=1

for  i in range( len(Data) ):
    plt.plot( Data[ i ] [Column1] ,  Data[ i ] [Column2]  ,'r.')        #r. :red

plt.show()
```



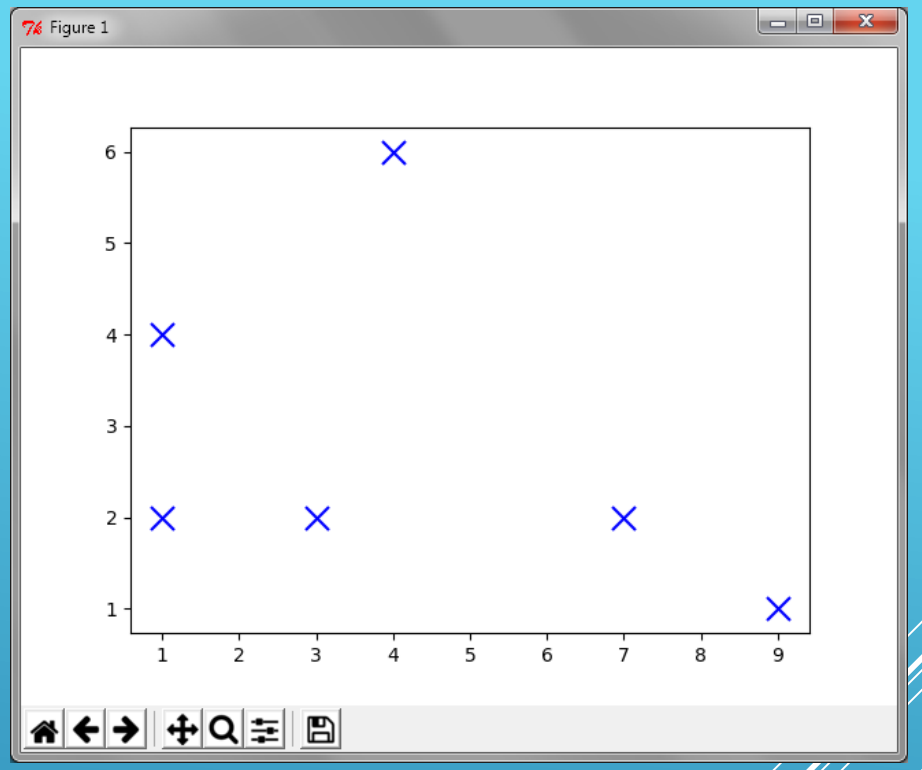
Bar plot for Data array

```
import numpy as np
import matplotlib.pyplot as plt

Data=np.array([ [1,2], [3,2], [4,6], [7,2], [1,4], [9,1] ])
Column1=0
Column2=1
BarWidth=1/1.5
for  i   in range( len(Data) ):
      plt.bar( Data[ i ] [Column1] , Data[ i ] [Column2] , BarWidth  , color="blue")
plt.show()
```



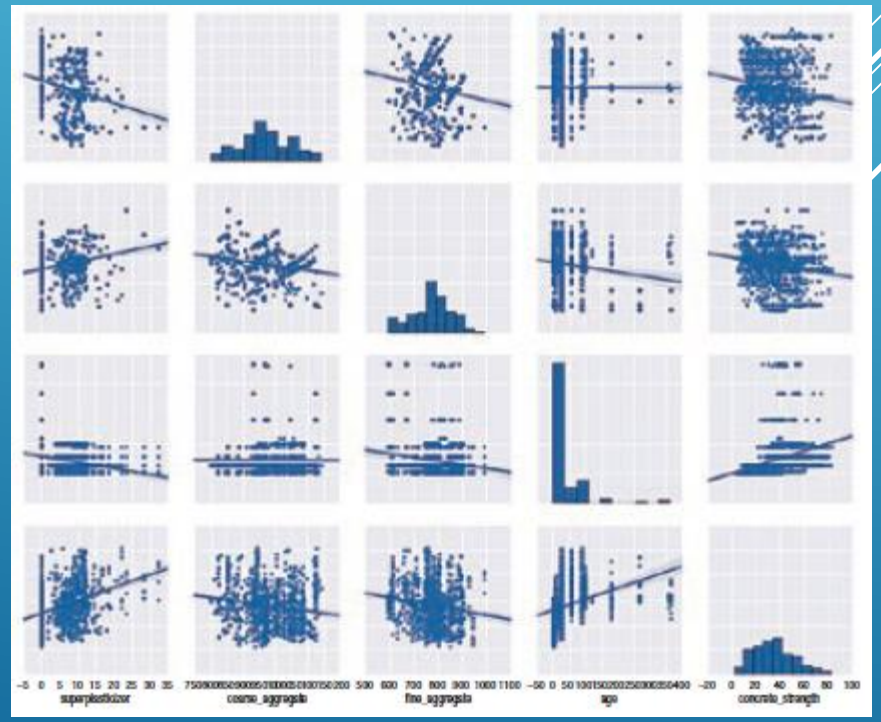
Scatter plot for Data

```
import numpy as np
import matplotlib.pyplot as plt
Data=np.array([ [1,2], [3,2], [4,6], [7,2], [1,4], [9,1] ])
Column1=0
Column2=1

plt.scatter( Data[:, Column1] , Data[:, Column2] , marker = "x“ ,  s=150 , linewidths = 5 , zorder = 1)
plt.show()


#Where s is the size of the marker
```



Calculating Pair Plot Between All Features

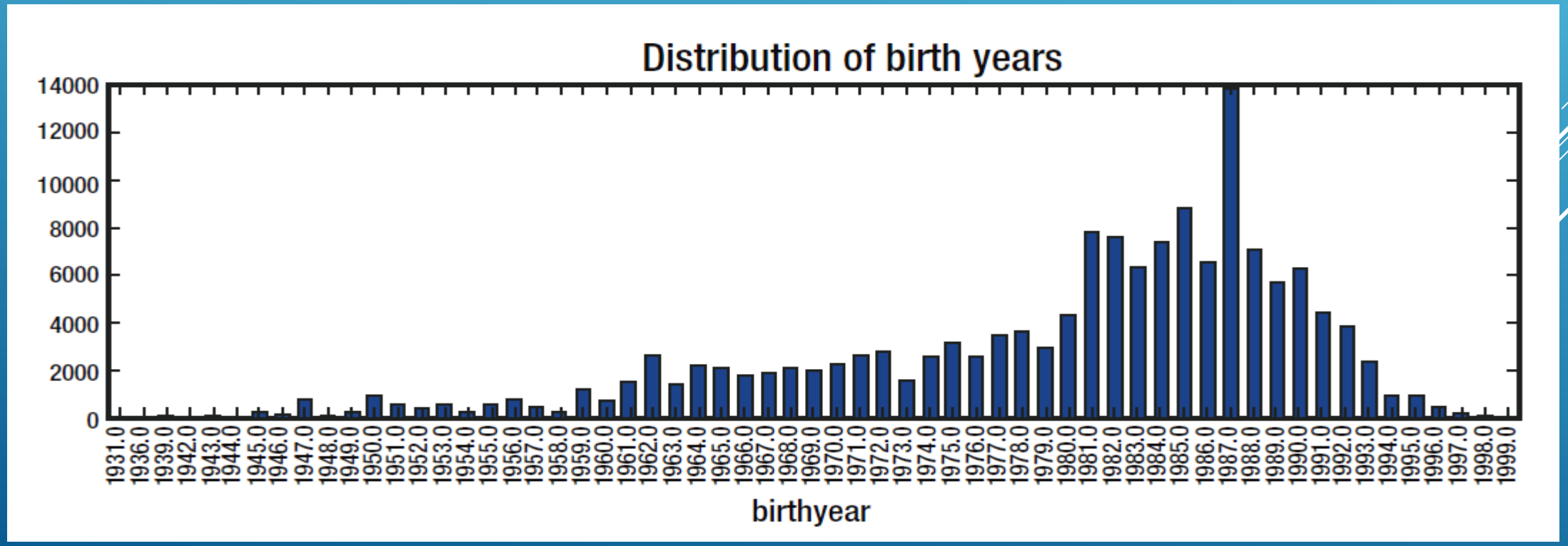
```
import numpy as np
import matplotlib.pyplot as plt
data=np.array([ [1,2], [3,2], [4,6], [7,2], [1,4], [9,1] ])

import pandas
df = pandas.DataFrame(Data)

#Calculating Pair Plot Between All Features
seaborn.pairplot(df)

plt.show()
```



Basic commands

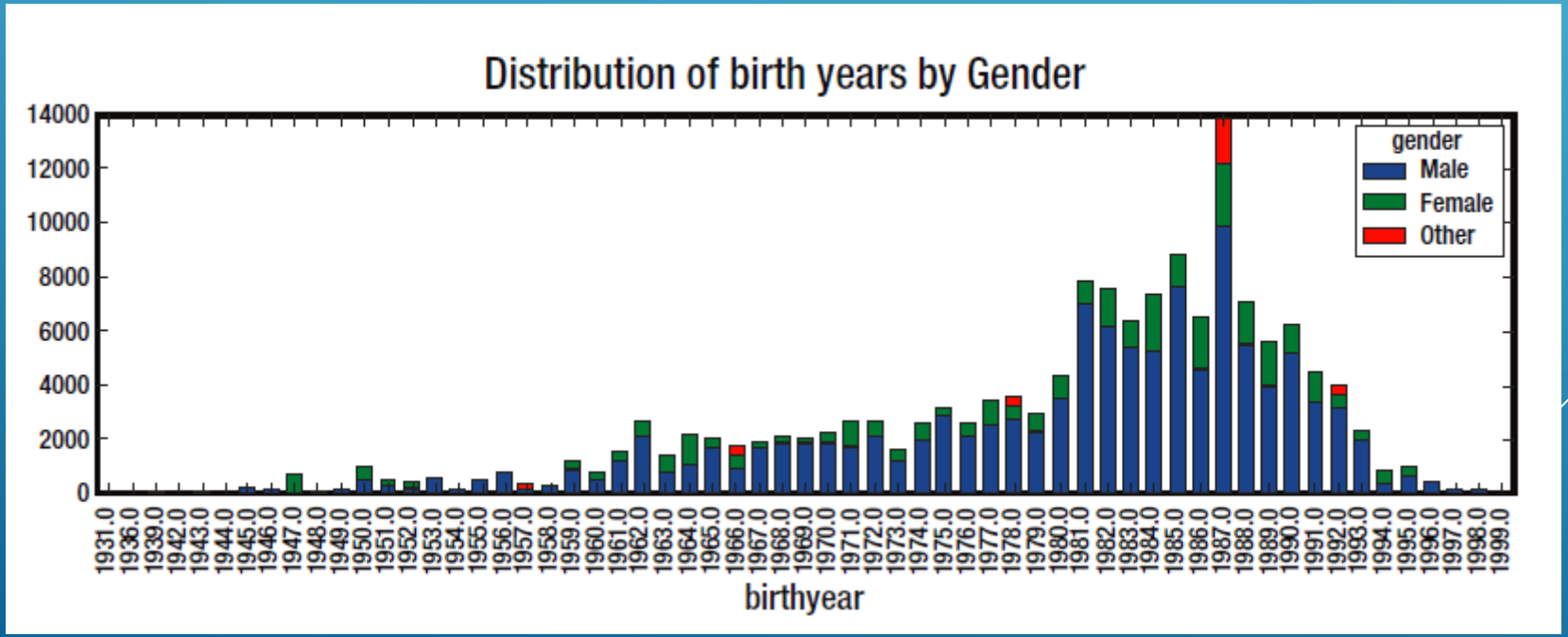
Sort, Filter, group, and Plotting the Data

- data = data.sort_values(by='birthyear')
- data = data[(data['birthyear'] >= 1931) & (data['birthyear']<=1999)]
- groupby_birthyear = data.groupby('birthyear').size()
- groupby_birthyear.plot.bar(title = 'Distribution of birth years', figsize = (15,4))
- plt.show()



Stacked column chart

```
data1 = data.groupby(['birthyear', 'gender']).size().unstack('gender').fillna(0)
data1.plot.bar(title ='Distribution of birth years by Gender', stacked=True, figsize = (15,4))
```



Convert string to DateTime

trip_id	starttime	stoptime	bikeid	tripduration	from_station_name	to_station_name
431	10/13/2014 10:31	10/13/2014 10:48	SEA00298	985.935	2nd Ave & Spring St	Occidental Park/ Occidental Ave S & S Washing...
432	10/13/2014 10:32	10/13/2014 10:48	SEA00195	926.375	2nd Ave & Spring St	Occidental Park/ Occidental Ave S & S Washing...
433	10/13/2014 10:33	10/13/2014 10:48	SEA00486	883.831	2nd Ave & Spring St	Occidental Park/ Occidental Ave S & S Washing...
434	10/13/2014 10:34	10/13/2014 10:48	SEA00333	865.937	2nd Ave & Spring St	Occidental Park/ Occidental Ave S & S Washing...
435	10/13/2014 10:34	10/13/2014 10:49	SEA00202	923.923	2nd Ave & Spring St	Occidental Park/ Occidental Ave S & S Washing...

Create new column as date/time

- data['StartTime1'] = data['starttime'].apply(lambda x: datetime.datetime.strptime(x, "%m/%d/%Y %H:%M"))

Extract Year, Month, Day, and Hour

- data['year'] = data['StartTime1'].apply(lambda x: x.year)
- data['month'] = data['StartTime1'].apply(lambda x: x.month)
- data['day'] = data['StartTime1'].apply(lambda x: x.day)
- data['hour'] = data['StartTime1'].apply(lambda x: x.hour)

Split String column to create new columns for Year-Month-Day

```
for index, component in enumerate(['year', 'month', 'day']):
    data[component] = data['AppointmentRegistration'].apply(lambda x: int(x.split('T')[0].split('-')[index]))
```

OR

```
data['year'] = data['AppointmentRegistration'].apply(lambda x: int(x.split('T')[0].split('-')[0]))
data['month'] = data['AppointmentRegistration'].apply(lambda x: int(x.split('T')[0].split('-')[1]))
data['day'] = data['AppointmentRegistration'].apply(lambda x: int(x.split('T')[0].split('-')[2]))
```

Notes

'abcde'[0:2] → 'ab'

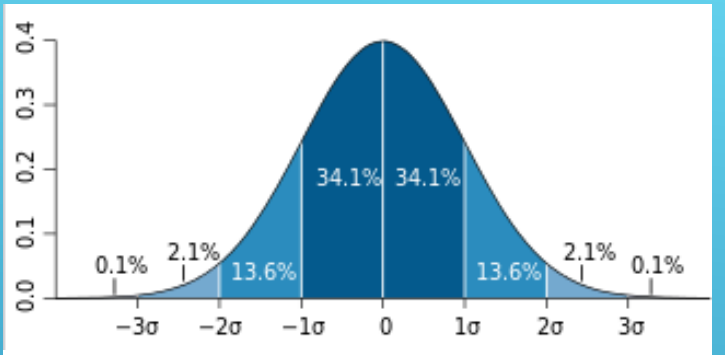
'abcde'[:2] → 'ab'

'abcde'[:-1] → 'abcd'

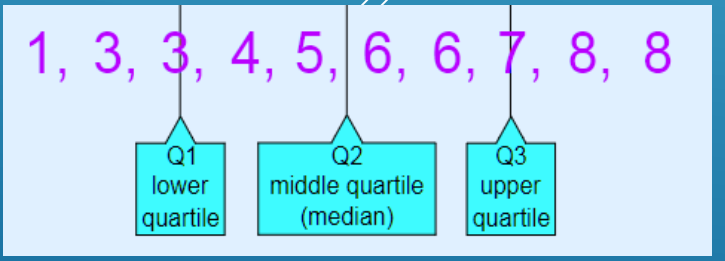
	Age	Gender	AppointmentRegistration
0	19	M	2014-12-16T14:46:25Z
1	24	F	2015-08-18T07:01:26Z
2	4	F	2014-02-17T12:53:46Z
3	5	M	2014-07-23T17:02:11Z
4	38	M	2015-10-21T15:20:09Z

Concepts

- **Mean** : Average
- **Median**: When the frequency of observations in the data is odd, the middle data point is returned as the median.
- **Mode**: returns the observation in the dataset with the highest frequency.
- **Variance:** represents variability of data points about the mean.
- **Standard deviation:** just like variance, also captures the spread of data along the mean. The only difference is that it is a square root of the variance.
- **Normal distribution (Gaussian distribution):** the mean lies at the center of this distribution with a spread (i.e., standard deviation) around it. Some 68% of the observations lie within 1 standard deviation from the mean; 95% of the observations lie within 2 standard deviations from the mean, whereas 99.7% of the observations lie within 3 standard deviations from the mean.
- **Outliers:** refer to the values distinct from majority of the observations. These occur either naturally, due to equipment failure, or because of entry mistakes.



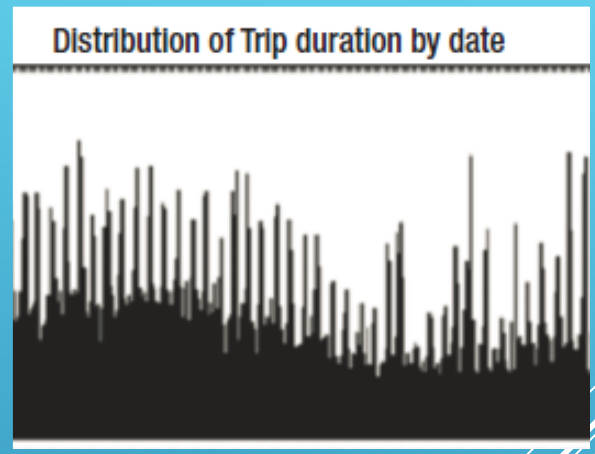
Confidence level	Formula
68%	Mean ± 1 std.
95%	Mean ± 2 std.
99.7%	Mean ± 3 std.



Plot mean

Draw mean of 'TripDuration' group by 'StartTime_date'

- data.groupby('starttime_date')['tripduration'].mean()
  .plot.bar(title = 'Distribution of Trip duration by date', figsize = (15,4))



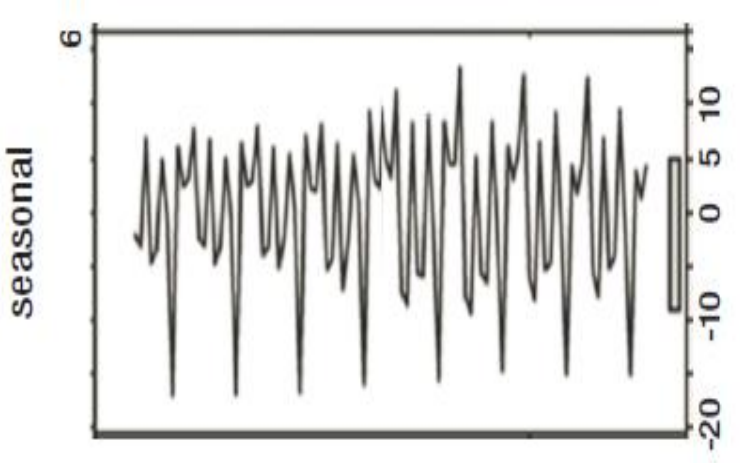
Calculate Mean, Standard deviation, Median

- print data['tripduration_mean'].mean()
- print data['tripduration_mean'].std()
- print data['tripduration_mean'].median()

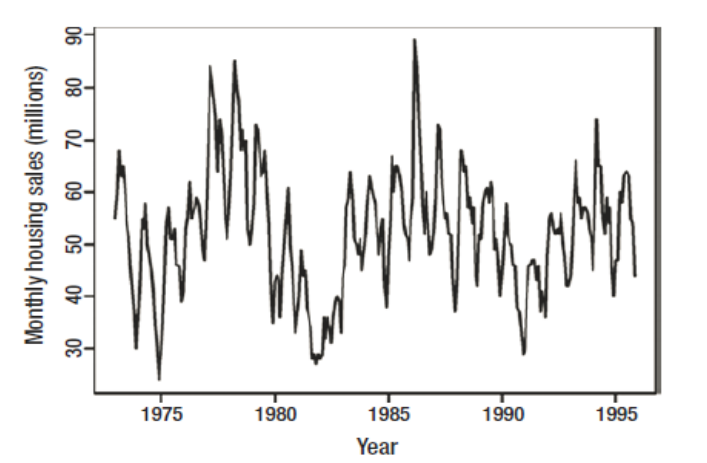
Seasonal pattern vs Cyclic Pattern vs Trend

- Seasonal: pattern over fixed period, like monthly pattern
- Cycle: pattern overall without worry about periods, to check if patterns repeat over non-periodic time cycles.
- Trend: In long-term does the continuous variable increase/decrease

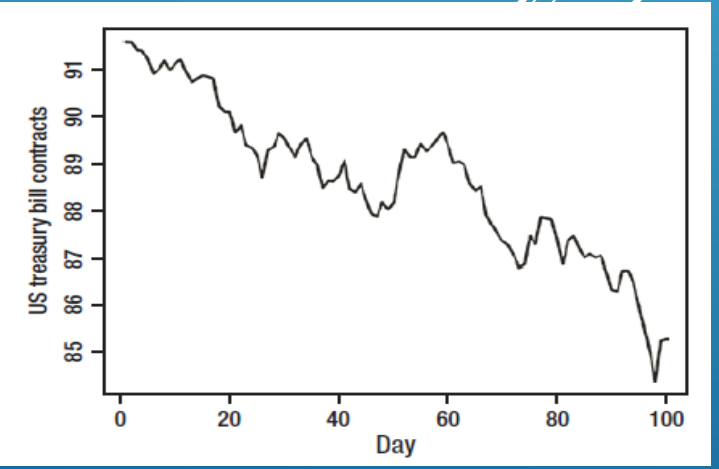
Seasonal pattern



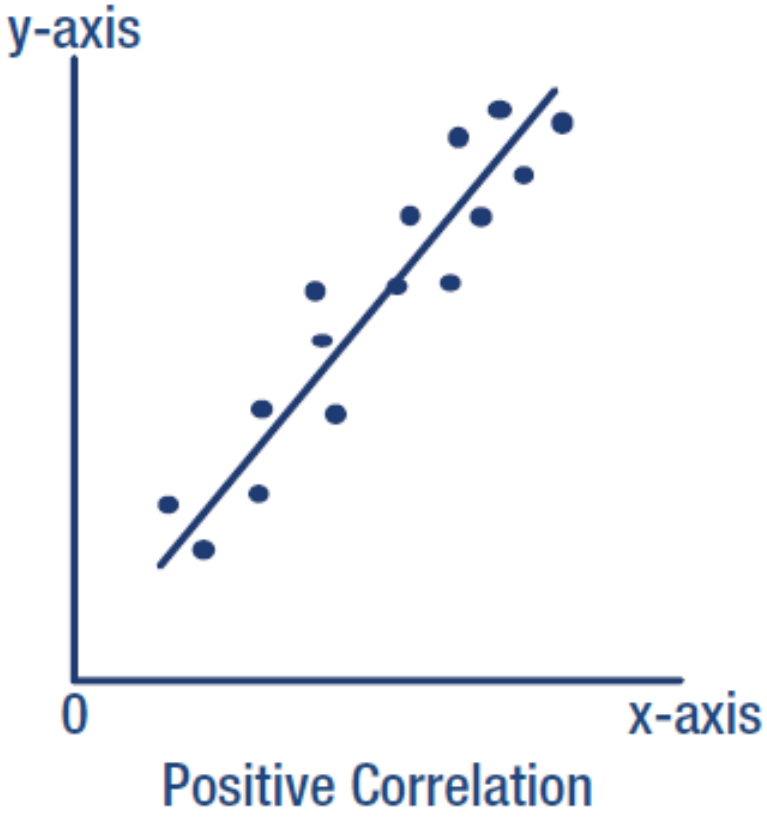
Cycle pattern



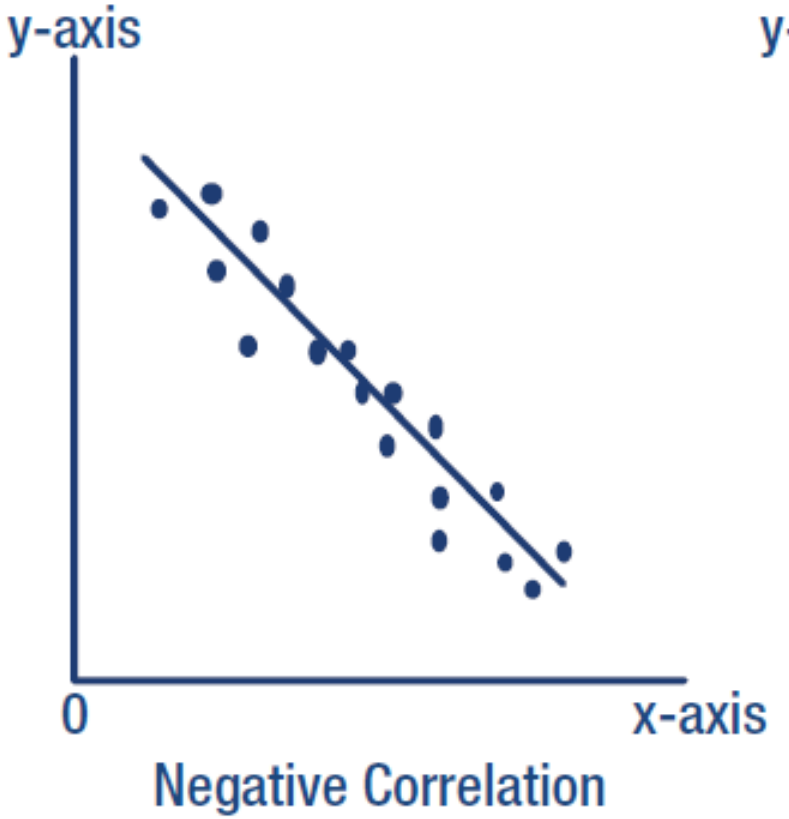
Trend



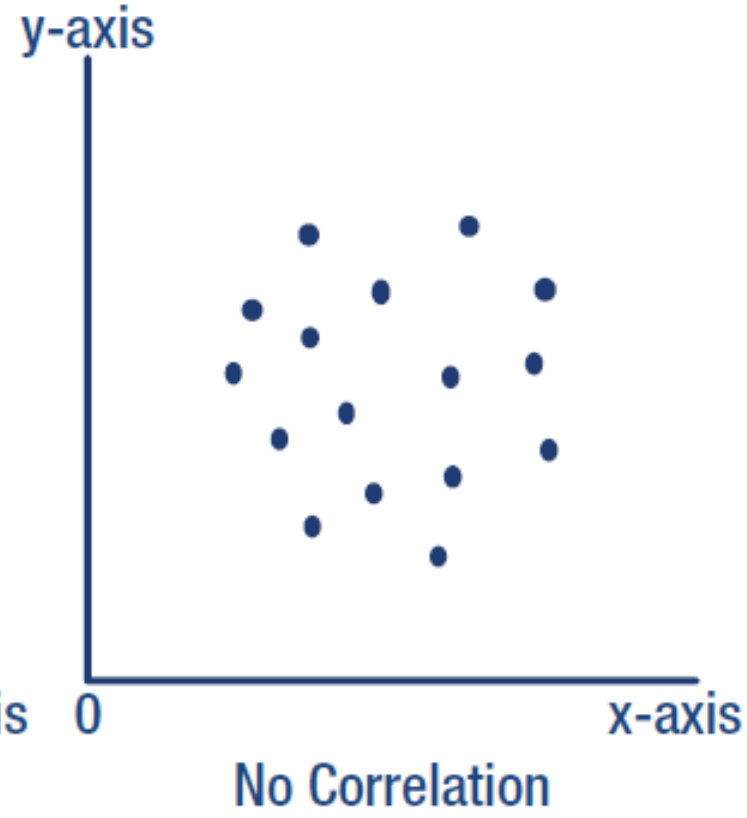
Correlation directions



Positive Correlation



Negative Correlation



No Correlation

Calculate the correlation

- pd.set_option('precision', 3)
- correlations = data[['tripduration','age']]**.corr()**
- print(correlations)

.corr(method='pearson')
method : {'pearson', 'kendall', 'spearman'}

pearson : standard correlation coefficient
kendall : Kendall Tau correlation coefficient
spearman : Spearman rank correlation

Output

	tripduration	age
tripduration	1.000	0.058
age	0.058	1.000

For more information http://www.statisticssolutions.com/correlation-pearson-kendall-spearman/

Rename Dataset columns' names

- Get data from the next URL and save as “concrete_data.csv”
  http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

- data = pd.read_csv('examples/concrete_data.csv’)
- print len(data)
- data.head()

	Cement (component 1)(kg in a m^3 mixture)	Blast Furnace Slag (component 2)(kg in a m^3 mixture)	Fly Ash (component 3)(kg in a m^3 mixture)	Water (component 4)(kg in a m^3 mixture)	Superplasticizer (component 5) (kg in a m^3 mixture)	Coarse Aggregate (component 6) (kg in a m^3 mixture)	Fine Aggregate (component 7) (kg in a m^3 mixture)	Age (day)	Concrete compressive strength(MPa, megapascals)
0	540.0	0.0	0.0	162.0	2.5	1040.0	676.0	28	79.99
1	540.0	0.0	0.0	162.0	2.5	1055.0	676.0	28	61.89
2	332.5	142.5	0.0	228.0	0.0	932.0	594.0	270	40.27
3	332.5	142.5	0.0	228.0	0.0	932.0	594.0	365	41.05
4	198.6	132.4	0.0	192.0	0.0	978.4	825.5	360	44.30

Renaming the Columns

- data.columns = ['cement_component', 'furnace_slag', 'flay_ash’,\
  'water_component’, 'superplasticizer', 'coarse_aggregate’, \
  'fine_aggregate', 'age', 'concrete_strength']

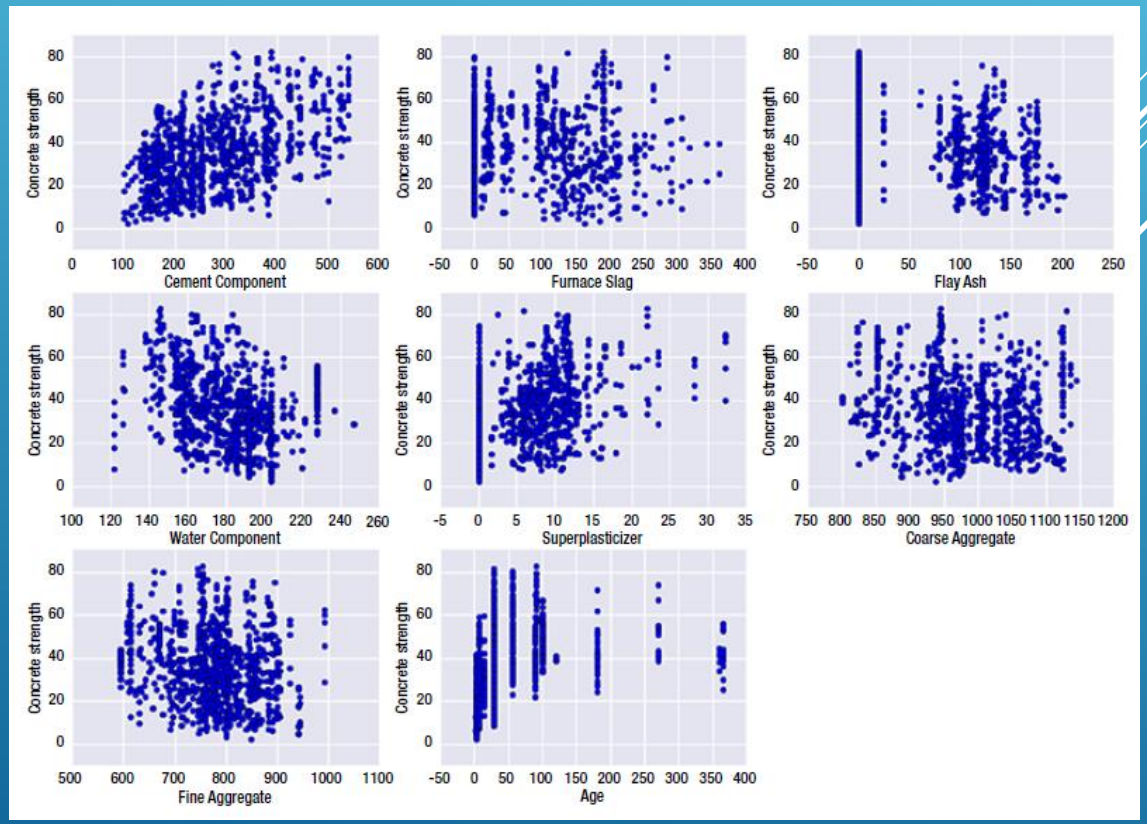
Loop through dataset

Draw relation between each feature and concrete_strength

```
plt.figure(figsize=(15,10.5))
plot_count = 1
for feature in list(data.columns)[:-1]:
        plt.subplot(3,3,plot_count)
        plt.scatter(data[feature], data['concrete_strength'])
        plt.xlabel(feature.replace('_',' ').title())
        plt.ylabel('Concrete strength')
        plot_count+=1
plt.show()
```

#//Get the correlation between data

```
pd.set_option('display.width', 100)
pd.set_option('precision', 3)
correlations = data.corr(method='pearson')
print(correlations)
```



Dimensionality Reduction

PCA Dimensionality Reduction

When we have a big dataset and we need to reduce the number of columns for fast training (remove redundancy features).

Example : Dataset (data) contains 20 features and we need to reduce them to 4 features

```
from sklearn.decomposition import PCA

NewDataFrame = PCA(n_components = 4).fit_transform(data)
```

Association Algorithm

Association Rules

Transaction 1	🍎 🍺 ⚪ 🍗
Transaction 2	🍎 🍺 ⚪
Transaction 3	🍎 🍺
Transaction 4	🍎 🍐
Transaction 5	💉 🍺 ⚪ 🍗
Transaction 6	💉 🍺 ⚪
Transaction 7	💉 🍺
Transaction 8	💉 🍐

Association rules analysis is a technique to uncover how items are associated to each other.

There are three common ways to measure association:

1) Support: This says how popular an itemset is, as measured by the proportion of transactions in which an itemset appears.

$$\text{Support}\{🍎\} = \frac{4}{8}$$

2) Confidence: This says how likely item Y is purchased when item X is purchased (not good if one item is common)

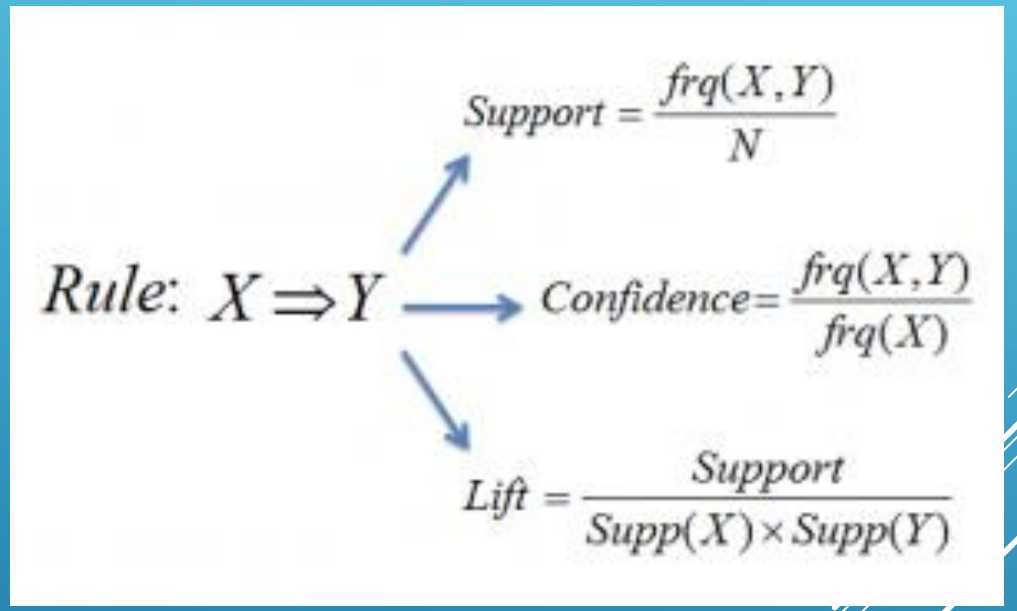
$$\text{Confidence}\{🍎 \rightarrow 🍺\} = \frac{\text{Support}\{🍎, 🍺\}}{\text{Support}\{🍎\}}$$

3) Lift: This says how likely item Y is purchased when item X is purchased

$$\text{Lift}\{🍎 \rightarrow 🍺\} = \frac{\text{Support}\{🍎, 🍺\}}{\text{Support}\{🍎\} \times \text{Support}\{🍺\}}$$

Association Rules (Apriori Algorithm)

```
from apyori import apriori

transactions = [
    ['beer', 'nuts'],
    ['beer', 'cheese'],
]
results = list(apriori(transactions))
```



Association Rules (Example)

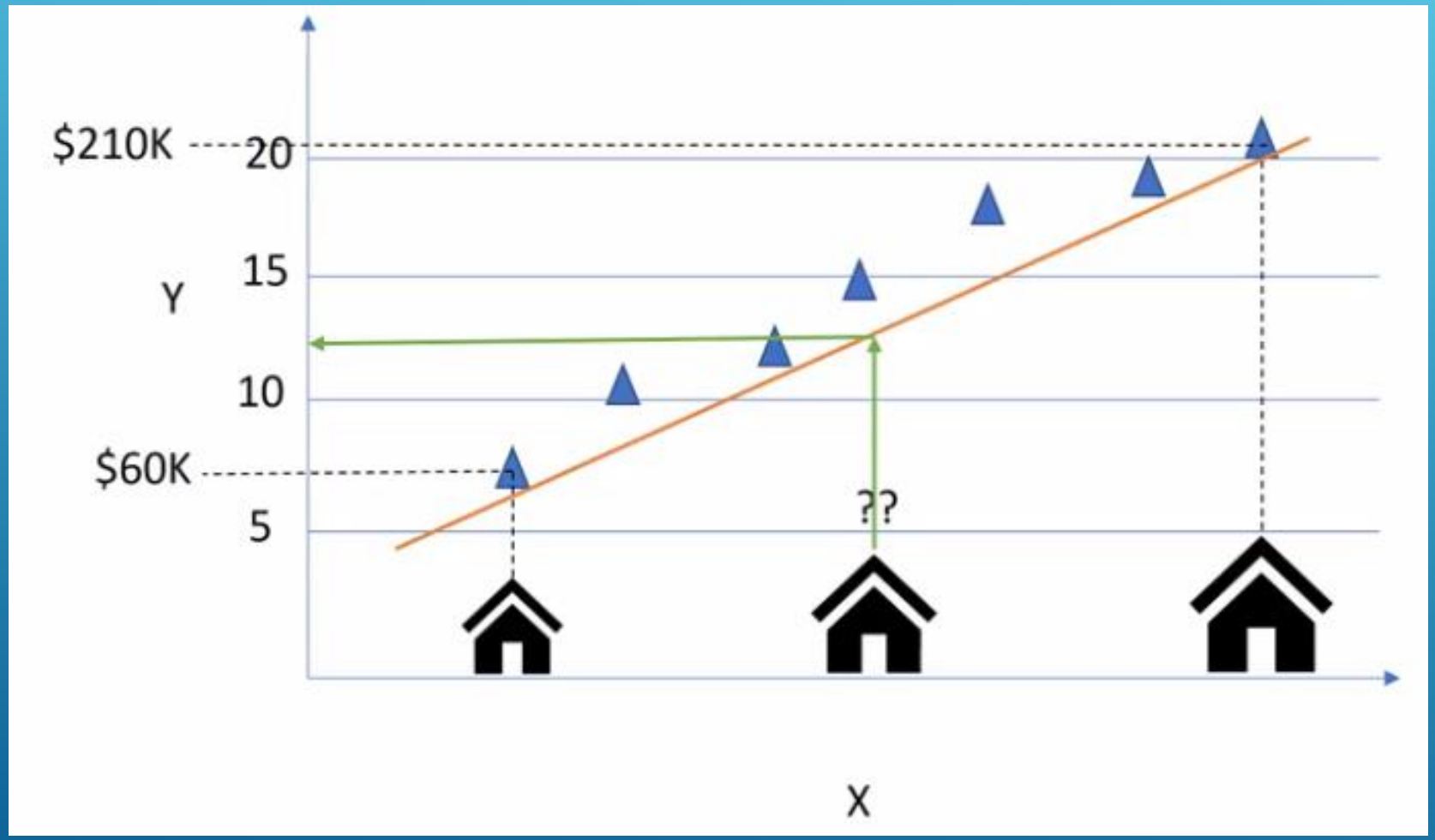
- pip install apyori

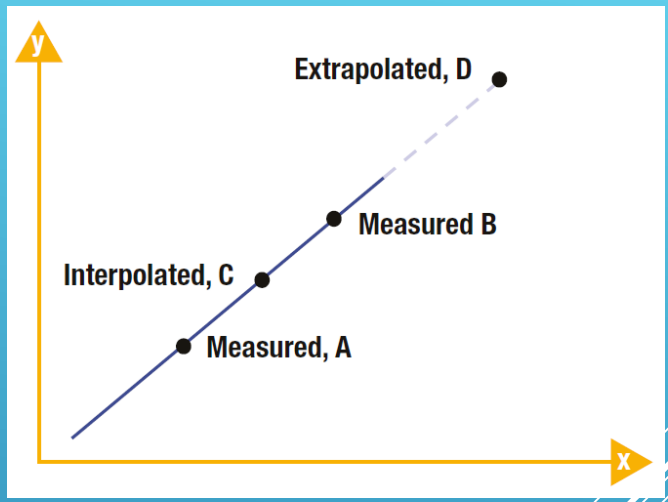
```
# Import the libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# Import Dataset
dataset = pd.read_csv('apriori_data2.csv', header = None)
records = [] for i in range(0, 11):
records.append([str(dataset.values[i,j]) for j in range(0, 10)])

# Train Apriori Model
from apyori import apriori
rules = apriori(records, min_support = 0.003, min_confidence = 0.2, min_lift = 3, min_length = 2)

# Visualising the results
results = list(rules)
```

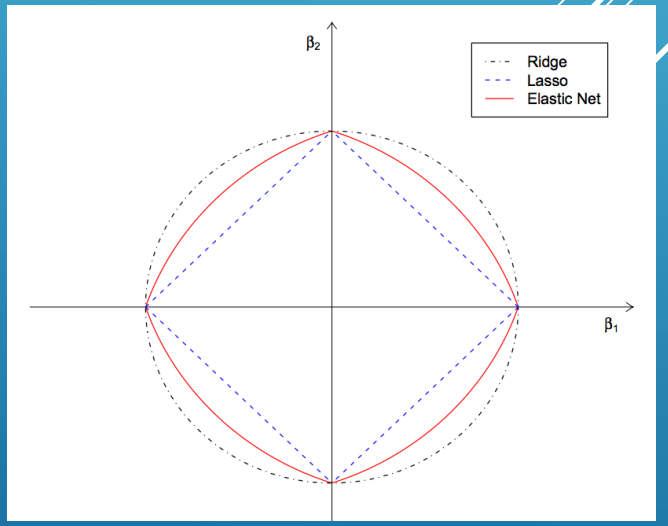
Regression Algorithm

Supervised machine learning

Regression


$210K
20
15
Y
10
$60K
5
??
X

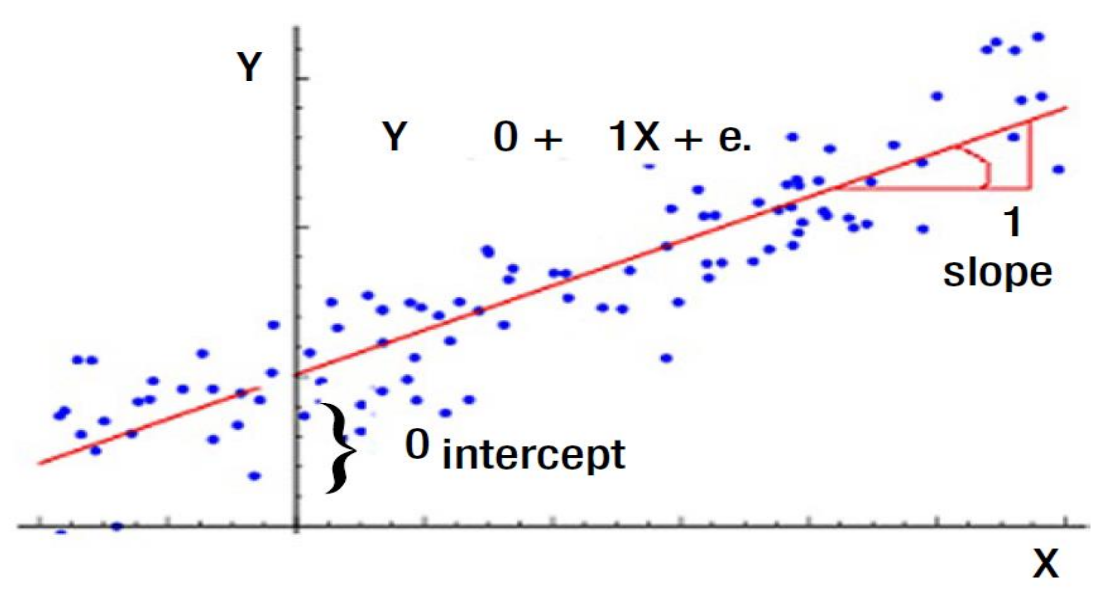

y
Extrapolated, D
Measured B
Interpolated, C
Measured, A
x

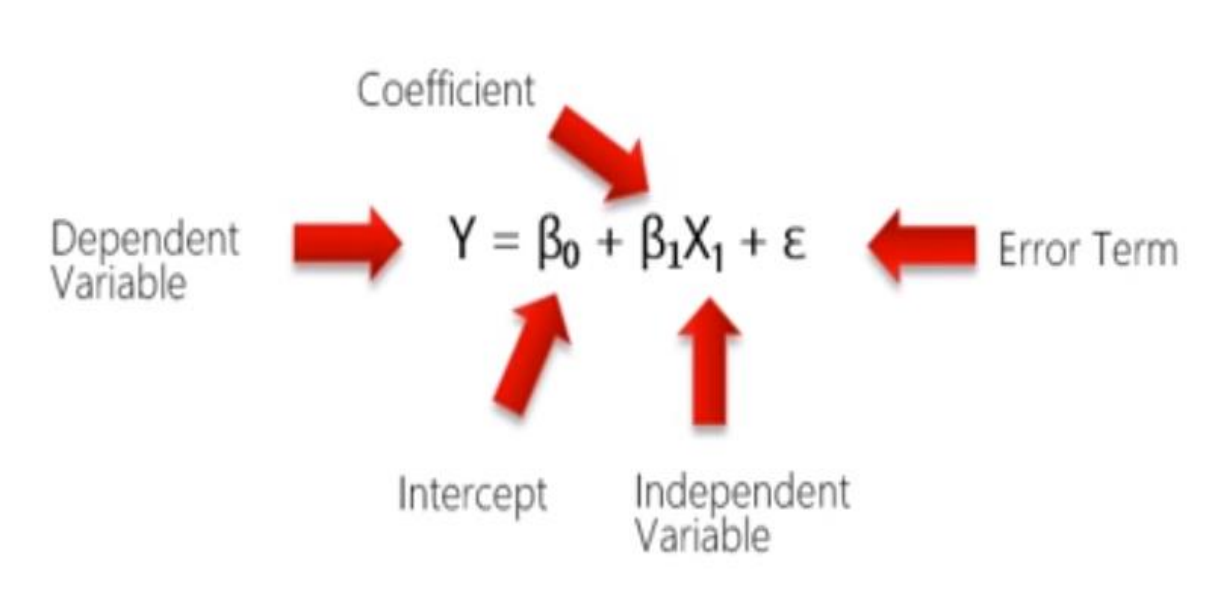

β₂
Ridge
Lasso
Elastic Net
β₁

Linear Regression

- x_train,x_test, y_train, y_test = train_test_split(data['x'],data['y'], test_size=0.3, random_state=1)
- regr = LinearRegression()
- regr.fit(x_train, y_train)
- y_pred = regr.predict(x_test)
- print 'R2 score: %f'%r2_score(y_test, y_pred)
- print 'Intercept: %f'%regr.intercept_
- print 'Coefficients: %s'%str(regr.coef_)

What is Intercept and Coefficient?



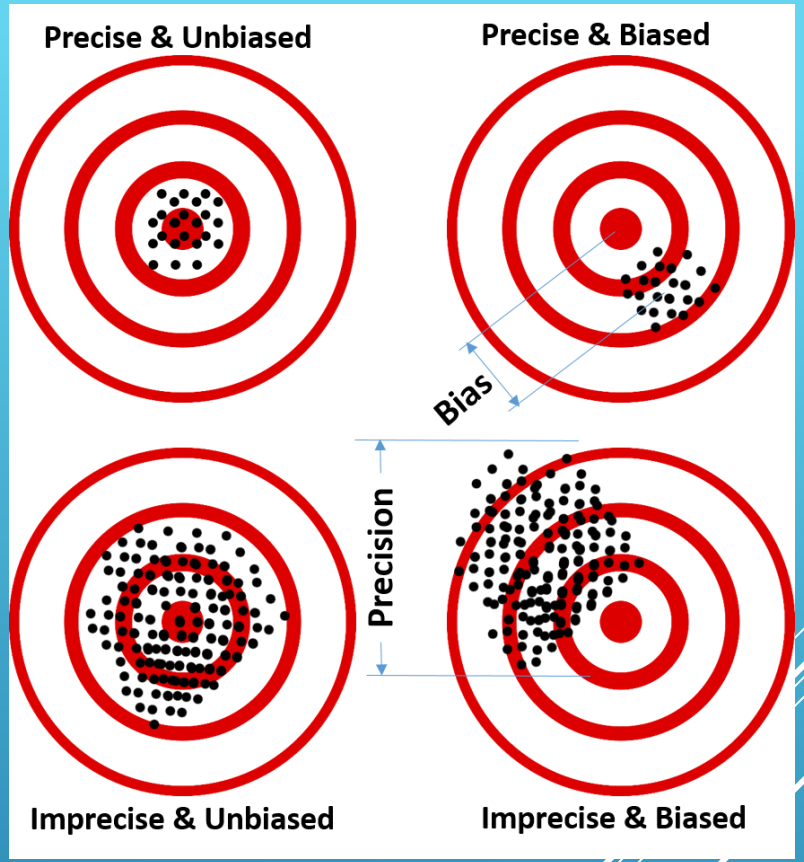


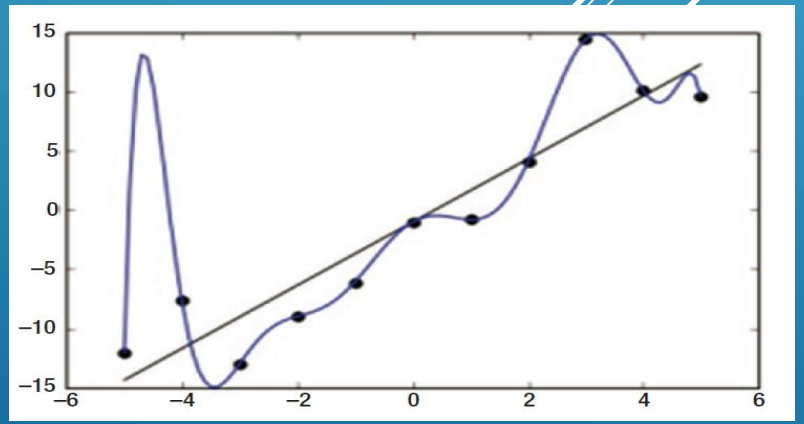
Bias, Overfitting

- Outliers can create a bias in the outcome of a regression model such that they will pull the best fit line toward themselves. Hence treatment of outliers is critical in regression.

- Overfitting refers to the phenomenon where a model is highly fitted on a dataset. This generalization thus deprives the model from making highly accurate predictions about unseen data.

- Overfitting = high variance
- Bias = Under fitting

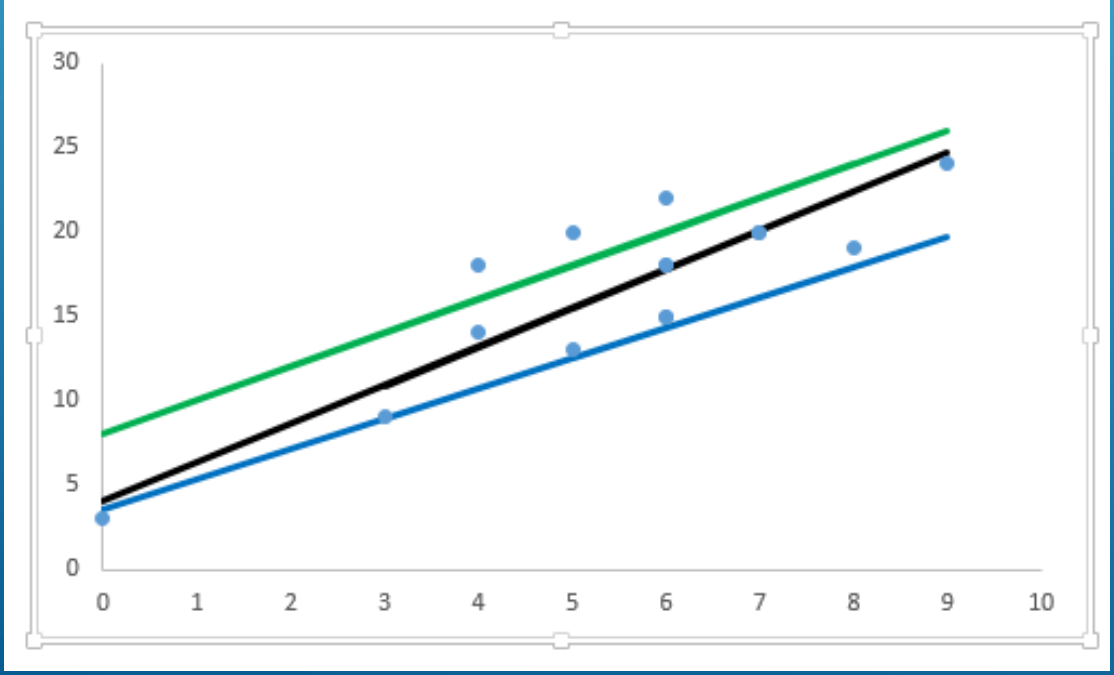


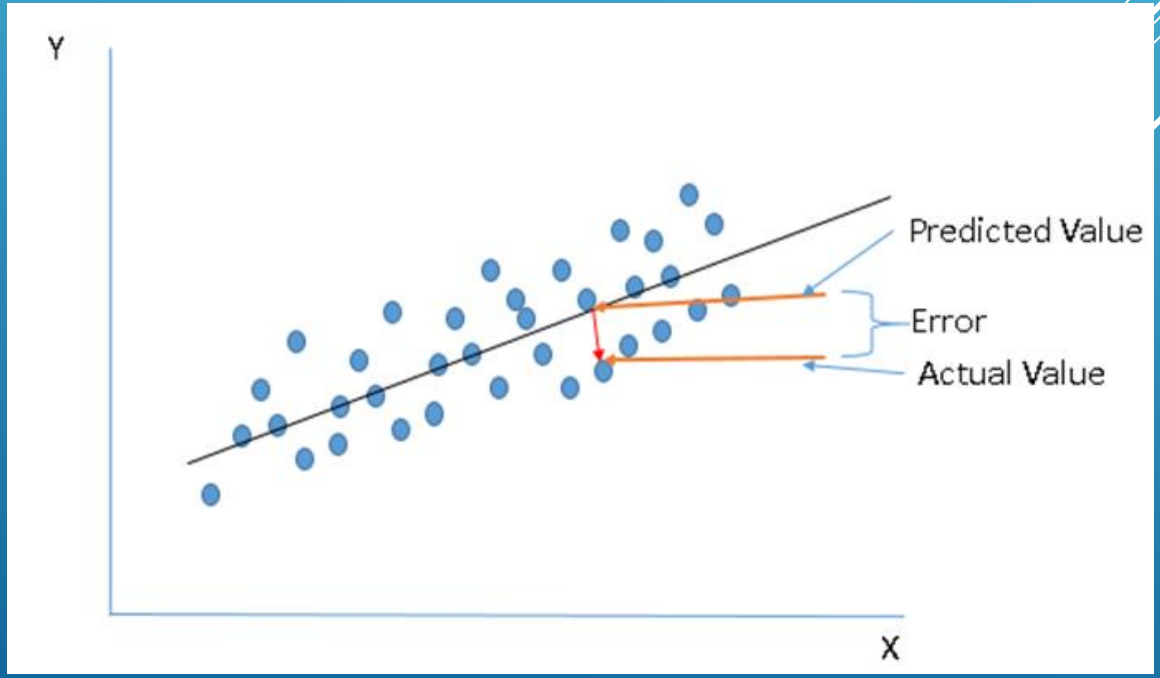


How to find the best regression line?

Here are some methods which check for error:

- Sum of all errors (∑error)
- Sum of absolute value of all errors (∑|error|)
- Sum of square of all errors (∑error^2)





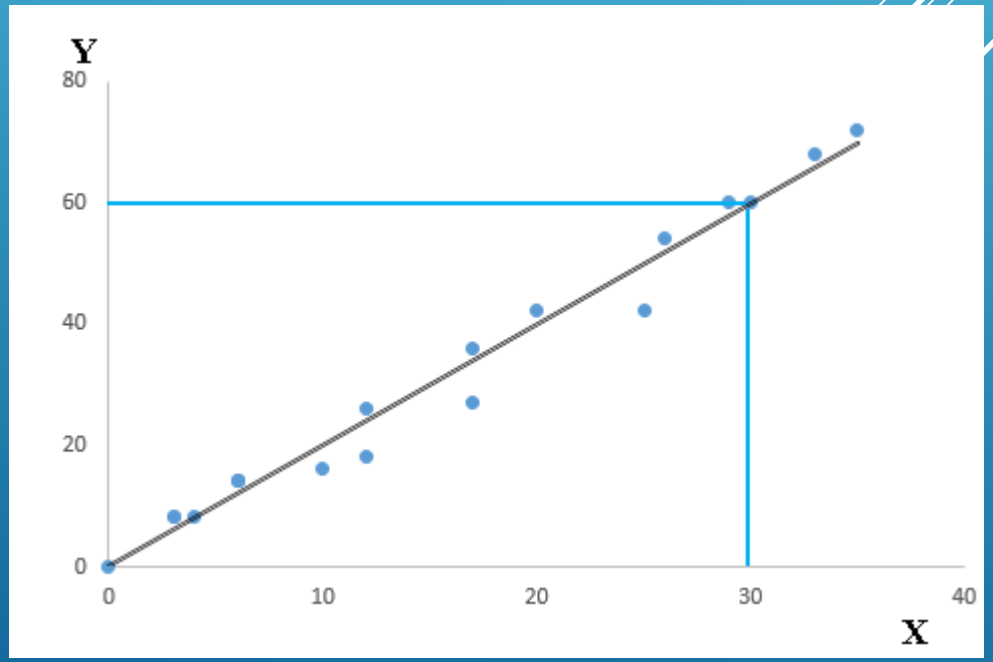
Ordinary Least Square (OLS)

The simplest form of regression formula is : Y= a * X + b

Now, find the value of a and b?

coefficients a and b are derived based on minimizing the sum of squared difference of distance between data points and regression line.

common algorithm to find the right coefficients for minimum sum of squared errors is Ordinary Least Square (OLS)



Regression regularization techniques

- if you are using regression without regularization, you have to be very special!
- 'regularization' techniques means penalizing the magnitude of coefficients of features along with minimizing the error between predicted and actual observations.

- **Residual Sum of Squares (RSS):** sum of square of errors between the predicted and actual values in the training data set.

$$RSS = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Ridge Regression:

- Performs L2 regularization, i.e. adds penalty equivalent to **square of the magnitude** of coefficients
- Optimizes(Minimization) objective = RSS + a * (sum of square of coefficients)

Lasso Regression:

- Performs L1 regularization, i.e. adds penalty equivalent to **absolute value of the magnitude** of coefficients
- Optimizes(Minimization) objective = RSS + a * (sum of absolute value of coefficients)

Where a is alpha parameter, user should change to get the best fit

Regression Metrics of Evaluation

- Explained Variance Score : The best score for this metric is 1.
- Mean absolute error (MAE) : values closer to 0 are better.
- Mean Squared Error : values closer to 0 are better.
- R Square : values between 0% and 100% and closer to 100% is better

#Python sample code

- print "Train MAE: ", metrics.mean_absolute_error(y_train, y_train_pred)
- print "Train RMSE: ", np.sqrt(metrics.mean_squared_error(y_train, y_train_pred))

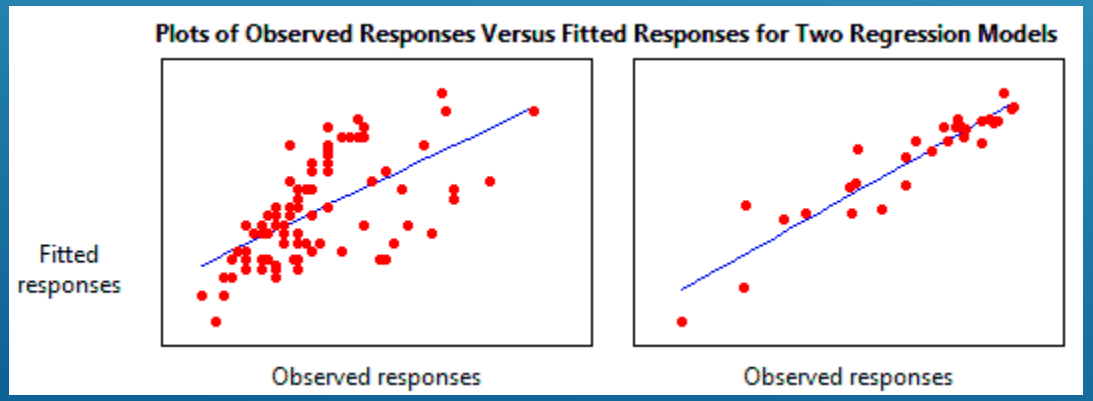
$$\frac{1}{n}\sum_{i=1}^{n}|y_i-\hat{y}_i|$$

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

What Is R-squared, (Coefficient of determination)?

$$\text{R-Square} = 1 - \frac{\sum(Y\_actual - Y\_predicted)^2}{\sum(Y\_actual - Y\_mean)^2}$$

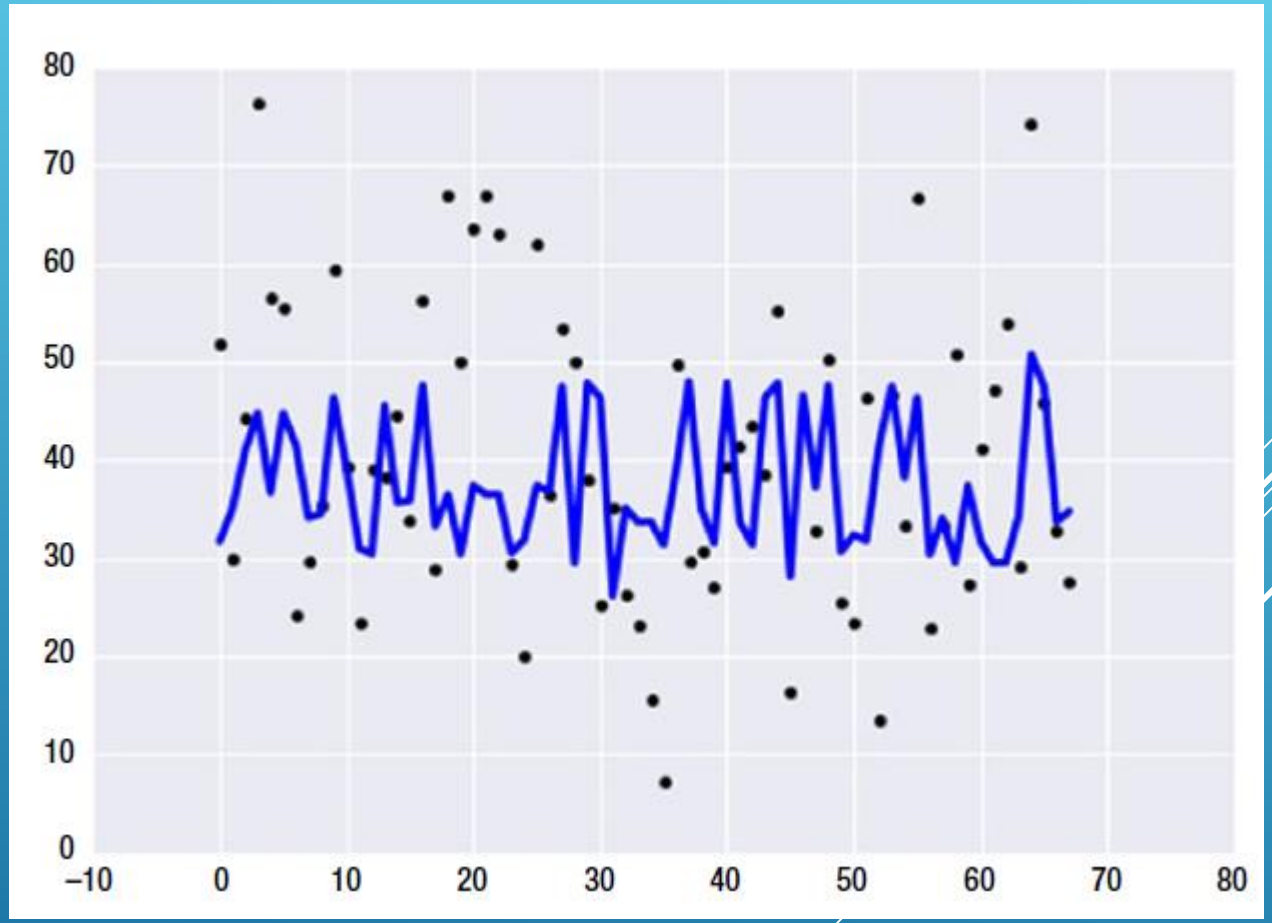
- R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.
- The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model. Or: **R-squared = Explained variation / Total variation**
- R-squared is always between 0 and 100%:
- 0% indicates that the model explains none of the variability of the response data around its mean.
- 100% indicates that the model explains all the variability of the response data around its mean.
- In general, the higher the R-squared, the better the model fits your data



```
import time
import random
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import statistics
from scipy import stats
from sklearn.linear_model import RANSACRegressor, LinearRegression, TheilSenRegressor
from sklearn.metrics import explained_variance_score, mean_absolute_error, mean_squared_error, median_absolute_error, r2_score
from sklearn.svm import SVR
from sklearn.linear_model import Ridge,Lasso,ElasticNet,BayesianRidge
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
import seaborn
from IPython.display import Image
import numpy as np

data = pd.read_csv('c:/concrete_data.csv')
data.columns = ['cement_component', 'furnace_slag', 'flay_ash', 'water_component', 'superplasticizer', 'coarse_aggregate', 'fine_aggregate', 'age', 'concrete_strength']

plt.figure(figsize=(15,7))
plot_count = 1
for feature1 in ['cement_component', 'flay_ash', 'water_component', 'superplasticizer', 'coarse_aggregate']:
        TwoRowsDataSet = data[[feature1,'concrete_strength']]
        TwoRowsDataSet=TwoRowsDataSet[(TwoRowsDataSet.T != 0).all()]
        x_train,x_test, y_train,  y_test = train_test_split(TwoRowsDataSet[feature1],TwoRowsDataSet['concrete_strength'], test_size=0.25, random_state=42)
        # Create linear regression object
        regr = LinearRegression()
        # Train the model using the training sets
        #in scikit-learn version everything has to be a 2d matrix, even a single column or row.
        #convert x_train to 2d by values.reshape(-1, 1)
        regr.fit(x_train.values.reshape(-1, 1), y_train.values.reshape(-1, 1))
        y_pred = regr.predict(x_test.values.reshape(-1, 1))
        # Plot outputs
        plt.subplot(2,3,plot_count)
        plt.scatter(x_test, y_test, color='black')
        plt.plot(x_test, y_pred, color='blue', linewidth=3)

        plt.xlabel(feature1.replace('_',' ').title())
        plt.ylabel('Concrete strength')
        print feature1, r2_score(y_test, y_pred)
        plot_count+=1
plt.show()
```

Multiple linear regression plot

- Ridge Regression
- Lasso Regression
- ElasticNet
- Gradient Boosting Regression
- Support Vector Machines



Ridge Regression

- x_train,x_test, y_train, y_test = train_test_split(data['x'],data['y'], test_size=0.3, random_state=1)
- **model = Ridge()**
- alphas = np.arange(0.1 , 5 , 0.1) #generate an array of values in the range of 0.1 to 5, with an offset of 0.1
- cv = GridSearchCV(estimator=model, param_grid=dict(alpha=alphas))
- y_pred = cv.fit(x_train, y_train).predict(x_test)

- print 'R2 score: %f'%r2_score(y_test, y_pred)
- print 'Intercept: %f'% model.intercept_
- print 'Coefficients: %s'%str(model.coef_)

Lasso Regression

- x_train,x_test, y_train, y_test = train_test_split(data['x'],data['y'], test_size=0.3, random_state=1)
- **model = Lasso()**
- alphas = np.arange(0.1 , 5 , 0.1) #generate an array of values in the range of 0.1 to 5, with an offset of 0.1
- cv = GridSearchCV(estimator=model, param_grid=dict(alpha=alphas))
- y_pred = cv.fit(x_train, y_train).predict(x_test)

- print 'R2 score: %f'%r2_score(y_test, y_pred)
- print 'Intercept: %f'%model.intercept_
- print 'Coefficients: %s'%str(model.coef_)

ElasticNet

- x_train,x_test, y_train, y_test = train_test_split(data['x'],data['y'], test_size=0.3, random_state=1)
- **model = ElasticNet()**
- alphas = np.arange(0.1 , 5 , 0.1)
- cv = GridSearchCV(estimator=model, param_grid=dict(alpha=alphas))
- y_pred = cv.fit(x_train, y_train).predict(x_test)

- print 'R2 score: %f ' % r2_score(y_test, y_pred)
- print 'Intercept: %f' % model.intercept_
- print 'Coefficients: %s ' % str(model.coef_)

Gradient Boosting Regression

- x_train,x_test, y_train, y_test = train_test_split(data['x'],data['y'], test_size=0.3, random_state=1)
- **regr = GradientBoostingRegressor()**
- regr.fit(x_train, y_train)
- y_pred = regr.predict(x_test)

- print 'R2 score: %f'%r2_score(y_test, y_pred)
- print 'Intercept: %f'%regr.intercept_
- print 'Coefficients: %s'%str(regr.coef_)

- plt.scatter(x_test, y_test, color='black')
- plt.plot(x_test, y_pred, color='blue', linewidth=3)
- plt.xlabel('X axis')
- plt.ylabel('Y axis')

Support Vector Machines

- x_train,x_test, y_train, y_test = train_test_split(data['x'],data['y'], test_size=0.3, random_state=1)
- regr = SVR(kernel='linear')
- regr.fit(x_train, y_train)
- y_pred = regr.predict(x_test)

- print 'R2 score: %f'%r2_score(y_test, y_pred)
- print 'Intercept: %f'%regr.intercept_
- print 'Coefficients: %s'%str(regr.coef_)

- plt.scatter(x_test, y_test, color='black')
- plt.plot(x_test, y_pred, color='blue', linewidth=3)
- plt.xlabel('X axis')
- plt.ylabel('Y axis')

Time Series

Supervised machine learning

Time Series

A time series forecast is different from regression in that time acts as an exploratory variable and should be continuous along equal intervals.

Algorithms

1) Autoregressive Forecast Model: uses observations at previous time steps to predict observations at future time step.

2) ARIMA Forecast Model: linear model, predict after remove trends and seasonality.

3) Prophet: Facebook library, it is quick and gives very good results, forecasting time series data.
Related Packages: **pip install fbprophet**

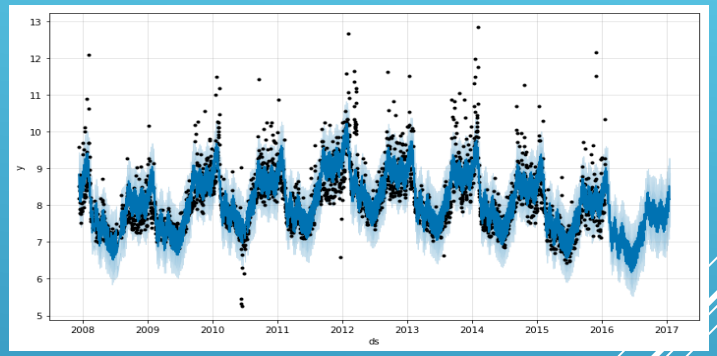
The input to Prophet is always a dataframe with two columns: ds and y.
The ds (datestamp) column must contain a date or datetime (either is fine).
The y column must be numeric, and represents the measurement we wish to forecast,
It is good to use log(y) and not the actual y to remove trends and noise data

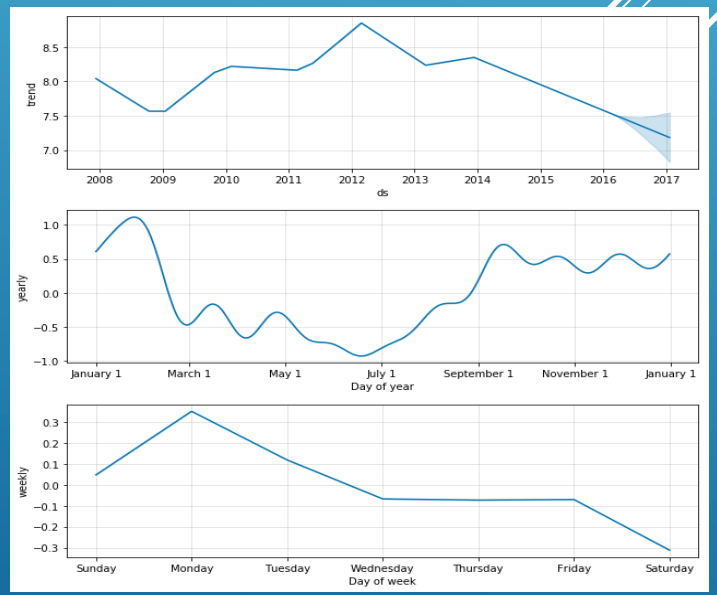
	ds	y
0	1992-01-01	146376
1	1992-02-01	147079
2	1992-03-01	159336
3	1992-04-01	163669
4	1992-05-01	170068

FB Prophet

```python
import pandas as pd
import numpy as np
from fbprophet import Prophet

data = pd.read_csv('../examples/example_wp_peyton_manning.csv')
data['y'] = np.log(data['y'])
df.head()

# Python
m = Prophet()
m.fit(data);

# add new 365 days to dataframe to allow predict them
future = m.make_future_dataframe(periods=365)
# add new 6 rows which are 6 months to predict them
future = m.make_future_dataframe(periods=6, freq = 'm')
future.tail()

# predict will create new dataframe with 4 columns
# yhat forecast value
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()

#draw the forecast
m.plot(forecast);

# draw weekly,monthly, and yearly trends
m.plot_components(forecast);

# get forecast without log transformation
data['y'] = np.exp(data['y'])
forecast['yhat'] = np.exp(forecast['yhat'])
forecast['yhat_lower'] = np.exp(forecast['yhat_lower'])
forecast['yhat_upper'] = np.exp(forecast['yhat_upper'])
m.plot(forecast);
m.plot_components(forecast);
```

	ds	yhat	yhat_lower	yhat_upper
294	2016-06-30	13.057100	13.037462	13.075858
295	2016-07-31	13.081213	13.061035	13.100568
296	2016-08-31	13.014274	12.993798	13.034220
297	2016-09-30	13.036700	13.016954	13.056166
298	2016-10-31	13.054154	13.031567	13.074649





Classification

Supervised machine learning

Classification

Algorithms

- Naïve bayes
- Decision Tree
- Kernel Approximation
- Random Forest Classification
- Gradient Bosting

Binary classification example

Diabetes Data Set

Detect Diabetes Disease based on analysis

Dataset Attributes:

1. Number of times pregnant
2. Plasma
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

```
#Import Library of Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
import numpy

# load pima indians dataset
dataset = numpy.loadtxt("c:\pima-indians-diabetes.csv", delimiter=",",skiprows =1)
# split into input (X) and output (Y) variables
X = dataset[:,0:8]
Y = dataset[:,8]

#Create a Gaussian Classifier
model = GaussianNB()

# Train the model using the training sets
model.fit(X, Y)

#Predict Output
predicted= model.predict(dataset[:,0:8])
print predicted

#to Predict 1 raw
RowNo2TestWith=2
predicted= model.predict(dataset[RowNo2TestWith,0:8].reshape(1, -1))
print predicted
print 'Actual value:%s, predicted value:%s' %(dataset[RowNo2TestWith,8],predicted)
```

```
def model_performance(model_name, X_train, y_train, y_test, Y_pred):
        print 'Model name: %s'%model_name
        print 'Test accuracy (Accuracy Score): %f'%metrics.accuracy_score(y_test, Y_pred)
        print 'Test accuracy (ROC AUC Score): %f'%metrics.roc_auc_score(y_test, Y_pred)
        print 'Train accuracy: %f'%clf.score(X_train, y_train)

        fpr, tpr, thresholds = metrics.precision_recall_curve(y_test, Y_pred)
        print 'Area Under the Precision-Recall Curve: %f'%metrics.auc(fpr, tpr)


features_of_choice = [u'Age', u'Gender']
x = np.array(data[features_of_choice])
y = np.array(data['Status'])
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3,random_state=1)


#Decision Tree Classification
clf = DecisionTreeClassifier()
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
model_performance('Decision tree classifier', x_train, y_train, y_test,y_pred)

#Kernel Approximation / SGD Classifier
rbf_feature = kernel_approximation.RBFSampler(gamma=1, random_state=1)
X_train = rbf_feature.fit_transform(x_train)
clf = SGDClassifier()
clf.fit(X_train, y_train)
X_test = rbf_feature.fit_transform(x_test)
Y_pred = clf.predict(X_test)
model_performance('Kernel approximation', X_train, y_train, y_test, Y_pred)

#Bagging(Bootstrap method) / Random Forest Classification
clf = RandomForestClassifier()
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
model_performance('Random Forest', x_train, y_train, y_test, y_pred)

#Gradient Boosting
clf = GradientBoostingClassifier(random_state=10, learning_rate=0.1,n_estimators=200, max_depth=5, max_features=10)
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)
model_performance('Gradient Boosting', x_train, y_train, y_test, y_pred)
```

Clustering

Unsupervised machine learning

Cluster

k-means clustering: needs the number of clusters before work!

we can get the expected numbers of clusters using one of the 3 next methods (Elbow method, Variance explained, BIC score)

- **Silhouette score:** measure how close observations within a cluster are.
  The best value is 1 and the worst is -1.
  A coefficient value of 0 indicates overlapping clusters.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

Data=np.array([ [1,2], [3,2], [4,6], [7,2], [1,4], [9,1],[10,3],[3,10] ])
Column1=0
Column2=1


model = KMeans(n_clusters=3)
model.fit(Data)

#array contains the center point for each cluster
CenterOfEachCluster = model.cluster_centers_

#scatter chart for clusters centers
plt.scatter( CenterOfEachCluster[:, Column1] , CenterOfEachCluster[:, Column2]
             , marker = "x",  s=150 ,color='blue', linewidths = 5 , zorder = 1)


labels = model.labels_  # array contains the cluster associated for each dataset points in order

print("CenterOfEachCluster %s "%CenterOfEachCluster)
print("labels %s" %labels)


colors = ["g.","r.","c.","y."]

for i in range(len(Data)):
    print("coordinate:",Data[i], "label:", labels[i])
    plt.plot(Data[i][Column1], Data[i][Column2], colors[labels[i]], markersize = 10)


plt.show()
```

COMPUTER VISION

Open CV for computer vision

- To install OpenCV pip install opencv-python
- Normal images are 4 dimensional array for each pixel (Blue ,Green, Red, and Alpha)
- It is normal to process images in Gray scale for fast performance
- Video is just a loop of images. So, any code processing images can process video too

For more information about open CV please visit

- https://www.tutorialspoint.com/opencv/index.htm
- https://www.youtube.com/playlist?list=PLQVvvaa0QuDdttJXlLtAJxJetJcqmqlQq

Common OpenCV functions

- image = cv2.imread(ImgPATH) #Read image
- cv2.cvtColor(ImgPATH, cv2.COLOR_BGR2GRAY) #Convert to gray scale
- cv2.imshow('Window Title', image) #Show Image in a window
- cv2.imwrite(ImgPATH, image) # to write image to HD
- _ , image = cv2.threshold(image, threshold, maxval, **thresholdType**) #change pixel to maxval if value greater than threshold
- image = cv2.adaptiveThreshold(image, maxValue, **adaptiveMethod**, **thresholdType**, blockSize, C) #Threshold is automatic calculated

thresholdType

Original	THRESH_BINARY	THRESH_BINARY_INV	THRESH_TRUNC	THRESH_TOZERO	THRESH_TOZERO_INV
					

adaptiveMethod

- ADAPTIVE_THRESH_MEAN_C : threshold value is the mean of neighborhood area.
- ADAPTIVE_THRESH_GAUSSIAN_C : threshold value is the weighted sum of neighborhood values where weights are a Gaussian window.

blockSize : A variable of the integer type representing size of the pixelneighborhood used to calculate the threshold value.

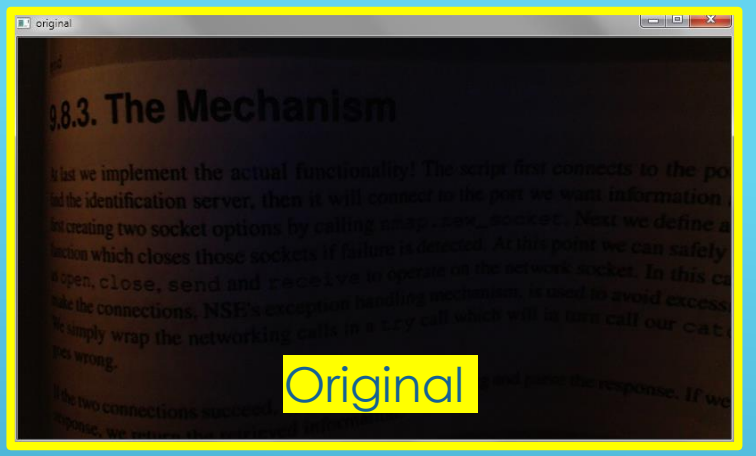
C : A variable of double type representing the constant used in the both methods (subtracted from the mean or weighted mean).

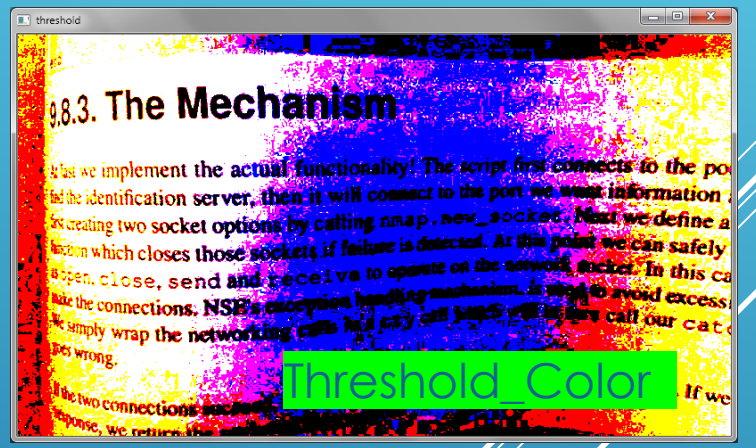
OpenCV Video Stream Example

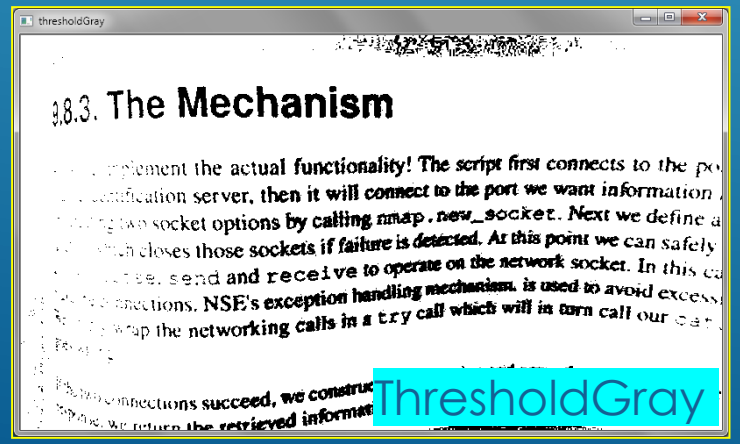
```
#VideoStream1 = cv2.VideoCapture('c:/boxed-correct.avi')  ➔ read stream from avi video file
VideoStream1 = cv2.VideoCapture(0)  ➔ read stream from the first connected video cam
while True:
    IsReturnFrame1,ImgFrame1=VideoStream1.read()
    if IsReturnFrame1 == 0:  break
    GrayImgFrame1=cv2.cvtColor(ImgFrame1,cv2.COLOR_BGR2GRAY)
    cv2.imshow('ImageTitle1',ImgFrame1)
    cv2.imshow('ImageTitle2',GrayImgFrame1)
    if cv2.waitKey(0): break

VideoStream1.release()
cv2.destroyAllWindows()
```

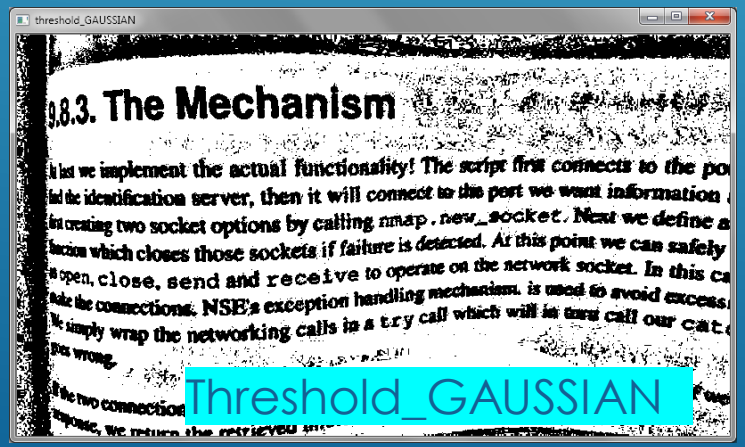
Image processing using OpenCV

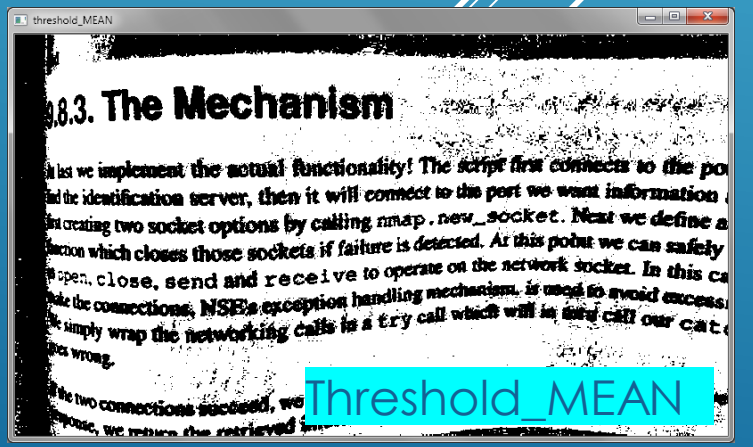
```
import cv2
import numpy as np

img = cv2.imread('c:/bookpage.jpg')
cv2.imshow('original',img)

retval, threshold_Color = cv2.threshold(img, 12, 255, cv2.THRESH_BINARY)
cv2.imshow('threshold_Color',threshold_Color)

grayscaled=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

retval, thresholdGray = cv2.threshold(grayscaled, 8, 255, cv2.THRESH_BINARY)
cv2.imshow('thresholdGray',thresholdGray)

thr_GU = cv2.adaptiveThreshold(grayscaled,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 115, 1)
cv2.imshow('threshold_GAUSSIAN',thr_GU)

thr_MEAN = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 115, 1)
cv2.imshow('threshold_MEAN',threshold_MEAN)

cv2.waitKey(0)
cv2.destroyAllWindows()
```





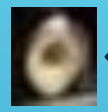






Object Recognition

MainImage



template

Search for image with exact lighting/scale/angle

```
img_rgb = cv2.imread('c:/MainImage.jpg')
img_gray = cv2.cvtColor(img_rgb, cv2.COLOR_BGR2GRAY)

template = cv2.imread('c:/template-for-matching.jpg',0)
w, h = template.shape[::-1]

res = cv2.matchTemplate(img_gray , template , cv2.TM_CCOEFF_NORMED)
threshold = 0.75
loc = np.where( res >= threshold)

for pt in zip(*loc[::-1]):
    cv2.rectangle(img_rgb, pt, (pt[0] + w, pt[1] + h), (0,255,255), 2)

cv2.imshow('Detected',img_rgb)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Result

NLP

Natural Language Processing

NLP (Natural Language Processing)

Why we need NLP packages?

- NLP Package handle a wide range of tasks such as Named-entity recognition , part-of-speech (POS) tagging, sentiment analysis, document classification, topic modeling, and much more.

Named-entity recognition: means extract names of persons, organizations, locations, time, quantities, percentages, etc.
example: Jim bought 300 shares of Acme Corp. in 2006. ➔ [Jim]Person bought 300 shares of [Acme Corp.]Organization in [2006]Time.

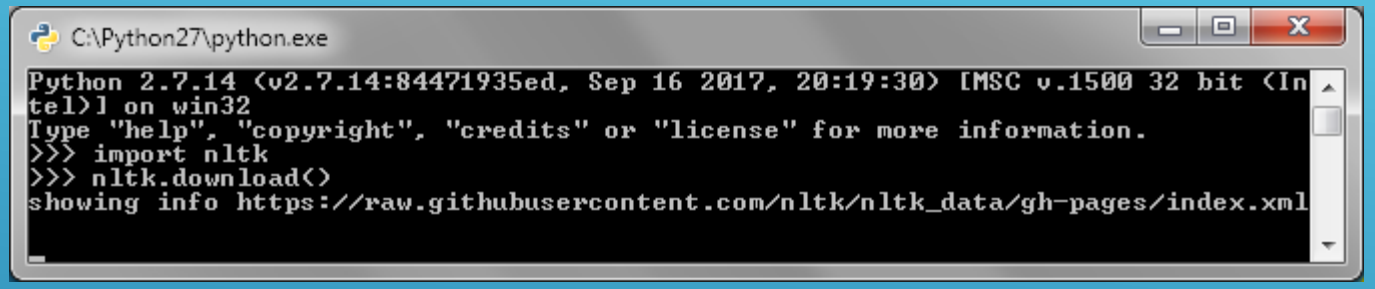
Part-of-speech tagging: used to identification of words as nouns, verbs, adjectives, adverbs, etc.

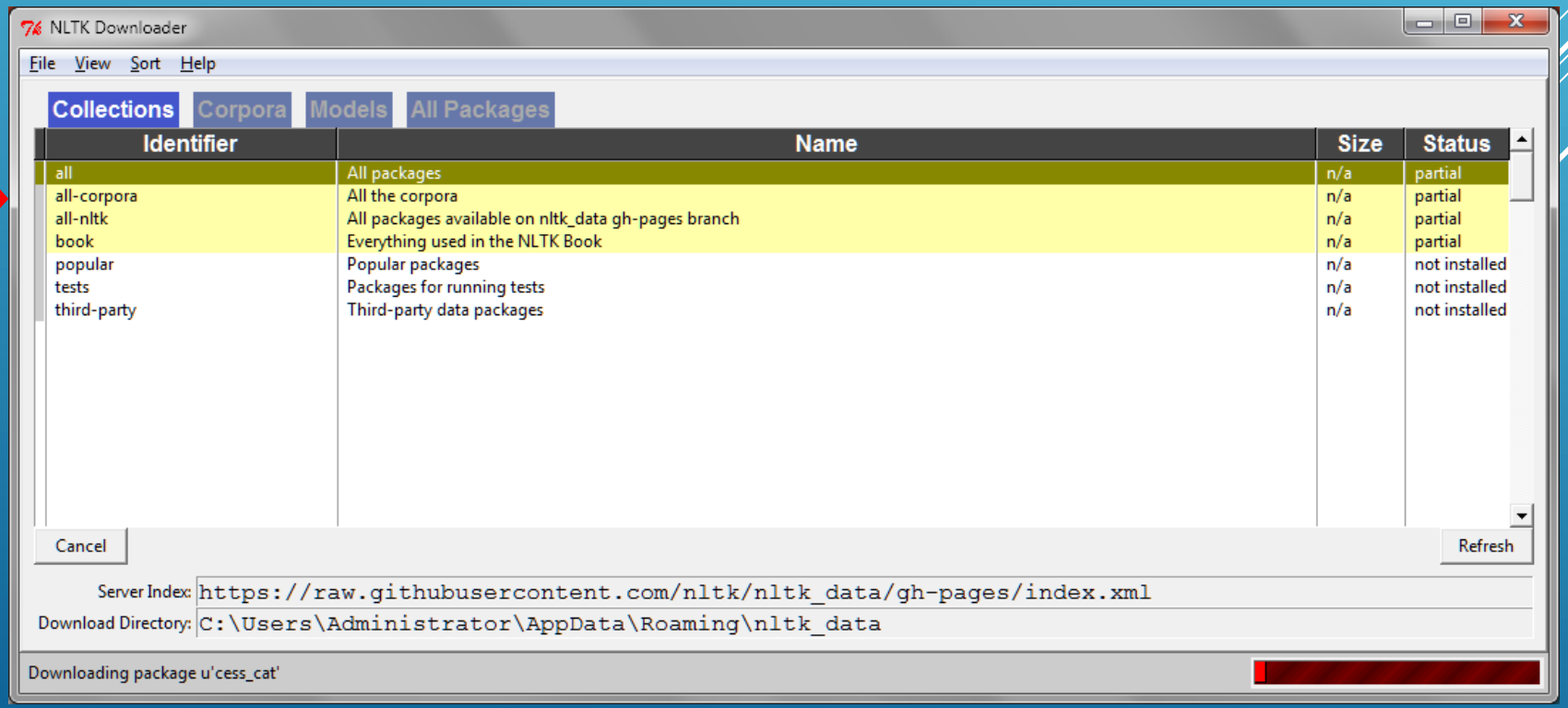
Top 5 Python NLP libraries

- **NLTK** good as education and research tool. Its modularized structure makes it excellent for learning and exploring NLP concepts, but it's not meant for production.
- **TextBlob** is built on top of NLTK, and it's more easily-accessible. good library for fast-prototyping or building applications that don't require highly optimized performance. Beginners should start here.
- **Stanford's CoreNLP** is a Java library with Python wrappers. It's in many existing production systems due to its speed.
- **SpaCy** is a new NLP library that's designed to be fast, streamlined, and production-ready. It's not as widely adopted.
- **Gensim** is most commonly used for topic modeling and similarity detection. It's not a general-purpose NLP library, but for the tasks it does handle, it does them well.

NLTK Package for Natural Language Processing

- Pip install nltk
- To download all packages using GUI, write ➔ nltk.download()





NLTK concepts

Tokenizing

- Word tokenizers : split text by words
- Sentence tokenizers : split text by paragraphs

Lexicon

- Get words and their actual means in the context

Corpora

- Text classification. Ex: medical journals, presidential speech

Lemmatizing (stemming)

- return the word to its root, ie (gone, went, going) ➔ go

WordNet

- List of different words that have the same meaning for the given word

NLTK Simple example

```
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords

EXAMPLE_TEXT = "Hello Mr. Smith, how are you doing today? The weather is great, and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard."
WordsArray = sent_tokenize(EXAMPLE_TEXT)
print(WordsArray)

ListOfAvailableStopWords= set(stopwords.words("english"))
print(ListOfAvailableStopWords)
```

textblob

- pip install textblob
- python -m textblob.download_corpora

```
from textblob import TextBlob


EXAMPLE_TEXT = ("Hello Mr. Mohamed, how are you doing today? The weather is great, "
                "and Python is awesome. The sky is pinkish-blue. You shouldn't eat cardboard.")


blob = TextBlob(EXAMPLE_TEXT)

print (blob.words) #split to words array
print (blob.sentences) #split to paragraphs array

for sentence in blob.sentences:
    print (sentence)

animals = TextBlob("cat dog octopus")
print animals.words                 #WordList(['cat', 'dog', 'octopus'])
print animals.words.pluralize()     #WordList(['cats', 'dogs', 'octopodes'])
print animals.words.singularize()   #WordList(['cat', 'dog', 'octopus'])

SpellingCorrection= TextBlob("I havv goood speling!")
print(SpellingCorrection.correct())    # I have good spelling!

from textblob import Word
w = Word("went")
print w.lemmatize("v")    #go

w = Word("doing")
print w.lemmatize("v")    #go

#Use internet to translate!
print (blob.translate(to="es")  )  #Spanish
```

Sentimental Analysis Example

- We have a sample of 3000 random users comments on imdb.com, amazon.com, and yelp.com website http://archive.ics.uci.edu/ml/machine-learning-databases/00331/
- Each comment has a score, Score is either 1 (for positive) or 0 (for negative)

```python
# -*- coding: utf-8 -*-
from textblob.classifiers import NaiveBayesClassifier
from textblob import TextBlob
import numpy

#Dataset contains 3000 ==> 1:positive, 0:negative
#http://archive.ics.uci.edu/ml/machine-learning-databases/00331/
train_dataset = numpy.loadtxt("c:/amazon_cells_labelled.txt", delimiter="\t",skiprows =0,dtype='str')
test_dataset1 = numpy.loadtxt(        "c:/imdb_labelled.txt", delimiter="\t",skiprows =0,dtype='str')
test_dataset2 = numpy.loadtxt(        "c:/yelp_labelled.txt", delimiter="\t",skiprows =0,dtype='str')


cl = NaiveBayesClassifier(train_dataset)
Current_accuracy = cl.accuracy(test_dataset1)
print("Dataset 1 Accuracy: {0}".format(Current_accuracy))

Current_accuracy = cl.accuracy(test_dataset2)
print("Dataset 2 Accuracy: {0}".format(Current_accuracy))

# Most Informative Features, top 5
cl.show_informative_features(5)

blob = TextBlob("The beer was amazing. But the hangover was horrible. My boss was not happy.",
                classifier=cl)


myTextAnalysis = blob.classify()
print("myTextAnalysis: {0}".format(myTextAnalysis))

##for sentence in blob.sentences:
##    print(sentence)
##    print(sentence.classify())
```

Python 2.7.14 Shell

File Edit Shell Debug Options Window Help

```
impleTextClassification.py
Dataset 1 Accuracy: 0.673
Dataset 2 Accuracy: 0.716
Most Informative Features
         contains(works) = True              1 : 0      =     21.0 : 1.0
         contains(money) = True              0 : 1      =     12.3 : 1.0
         contains(great) = True              1 : 0      =     11.0 : 1.0
         contains(price) = True              1 : 0      =      9.8 : 1.0
           contains(Not) = True              0 : 1      =      9.6 : 1.0
myTextAnalysis: 0
>>>
```

Ln: 24 Col: 4

Python Deep Learning

Common Deep learning fields

Main Fields

- Computer vision
- Speech recognition
- Natural language processing
- Audio recognition
- Social network filtering
- Machine translation

Visual examples:

- Colorization of Black and White Images. https://www.youtube.com/watch?v=_MJU8VK2PI4
- Adding Sounds To Silent Movies. https://www.youtube.com/watch?v=0FW99AQmMc8
- Automatic Machine Translation.
- Object Classification in Photographs.
- Automatic Handwriting Generation.
- Character Text Generation.
- Image Caption Generation.
- Automatic Game Playing. https://www.youtube.com/watch?v=TmPfTpjtdgg

Common Deep learning development tool (Keras)

- Keras is a free Artificial Neural Networks (ANN) library (deep learning library).
- it is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
- It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Deep learning Computer vision (image processing)

3 common ways to detect objects

- median based features
- edge based features
- threshold based features

How to use Neural Network Models in Keras

Five steps

1. Define Network.
2. Compile Network.
3. Fit Network.
4. Evaluate Network.
5. Make Predictions.

Step 1. Define Network

- Neural networks are defined in Keras as a sequence of layers.
- The first layer in the network must define the number of inputs to expect. for a Multilayer Perceptron model this is specified by the input_dim attribute.
- Example of small Multilayer Perceptron model (2 inputs, 5 hidden layers, 1 output)

```
model = Sequential()
model.add(Dense(5, input_dim=2))
model.add(Dense(1))
```

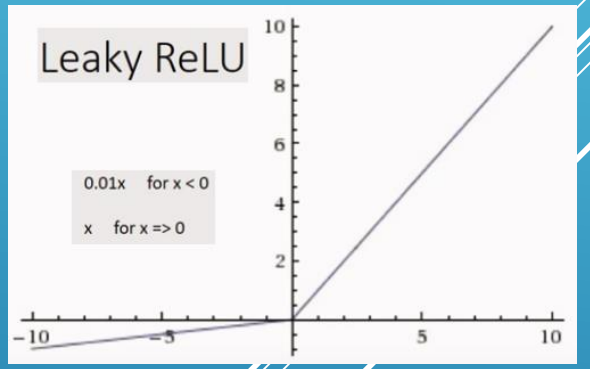
- Re-write after add activation function

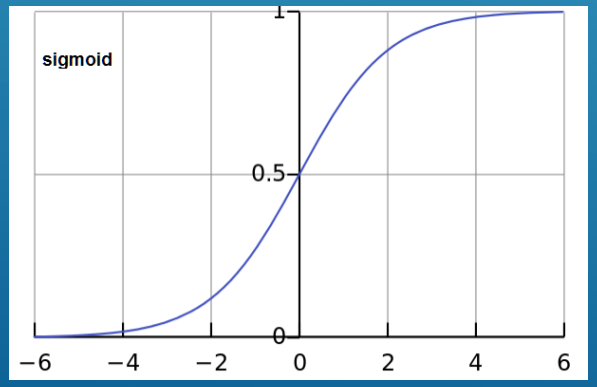
```
model = Sequential()
model.add(Dense(5, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

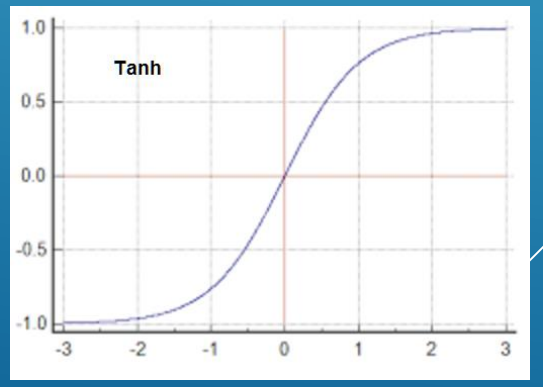
Available Activation Functions

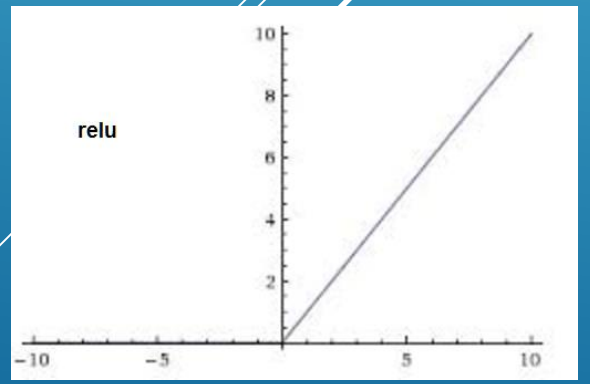
Optional, it is like a filter, used to solve some common predictive modeling problem, to get significant boost in performance.

- Sigmoid: used for Binary Classification (2 class) one neuron the output layer. What ever the input it will map to zero or one.
- Softmax: used for Multiclass Classification (>2 class), one output neuron per class value.
- Linear: used for Regression, the number of neurons matching the number of outputs.
- Tanh: what ever the input it will convert to number between -1 and 1
- Relu: either 0 for a<0 or a for a>0. so, it just remove the negative values and pass the positive as it is.
- LeakyReLU: minimize the value of negative values and pass as it is if positive
- elu
- selu
- softplus
- softsign
- hard_sigmoid
- PReLU
- ELU
- ThresholdedReLU









Step 2. Compile Network

- Specifically the optimization algorithm to use to train the network and the loss function used to evaluate the network that is minimized by the optimization algorithm.

```
model.compile(optimizer='sgd', loss='mse')
```

Optimizers tool to minimize loss between prediction and real value. Commonly used optimization algorithms:

- 'sgd' (Stochastic Gradient Descent) requires the tuning of a learning rate and momentum.
- ADAM requires the tuning of learning rate.
- RMSprop requires the tuning of learning rate.

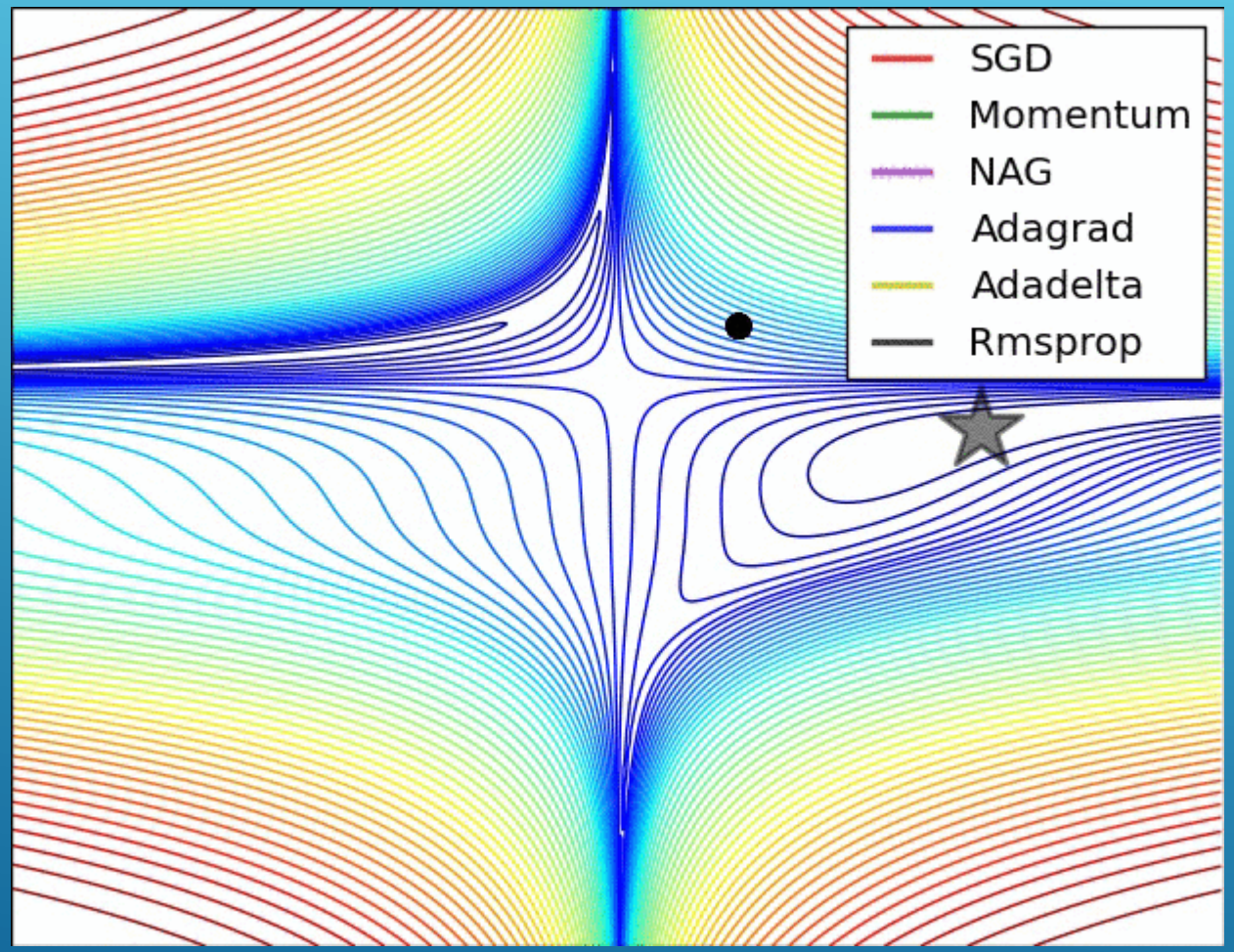
Loss functions:

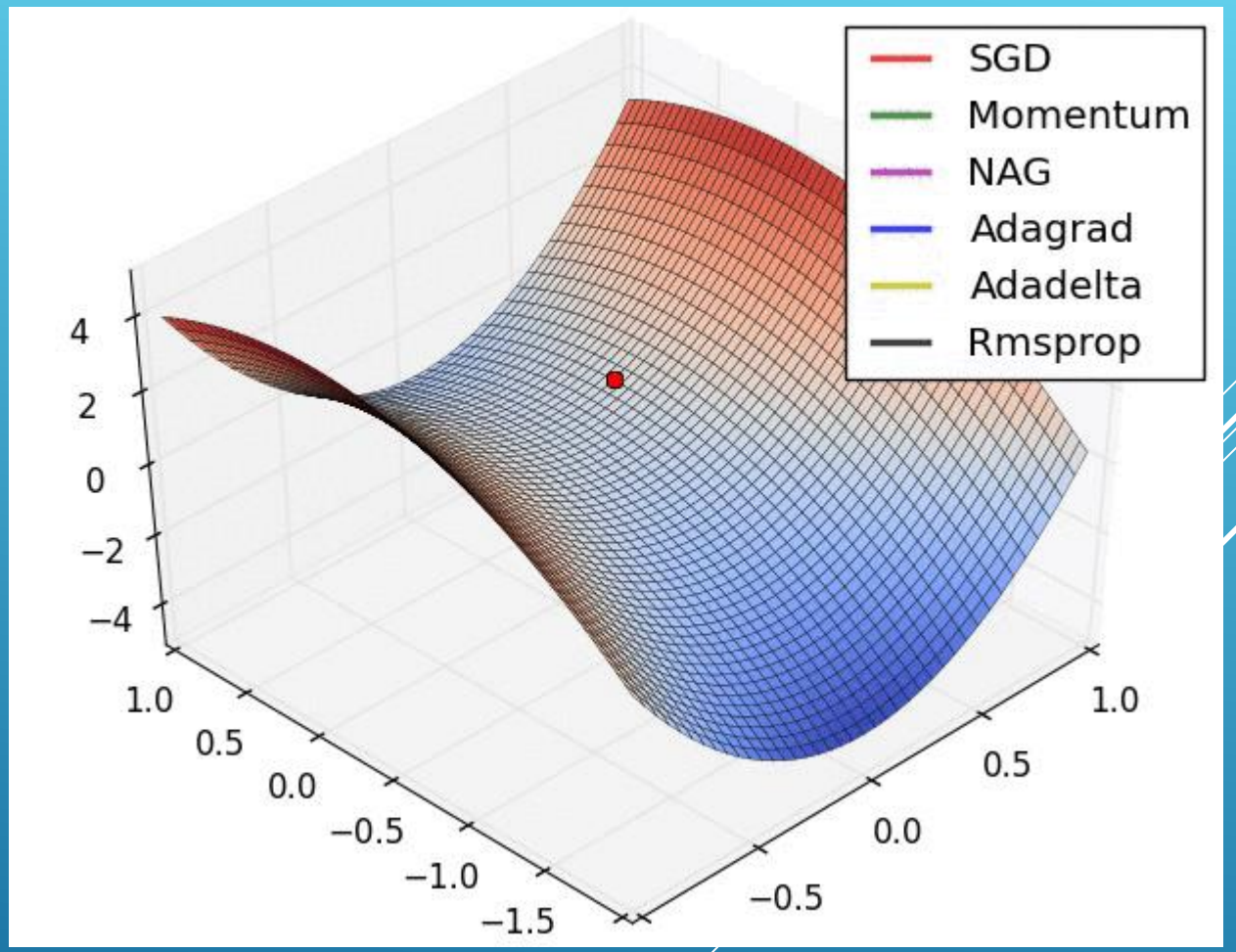
- Regression: Mean Squared Error or 'mse'.
- Binary Classification (2 class): 'binary_crossentropy'.
- Multiclass Classification (>2 class): 'categorical_crossentropy'.

Finally, you can also specify metrics to collect while fitting the model in addition to the loss function. Generally, the most useful additional metric to collect is accuracy.

```
model.compile(optimizer='sgd', loss='mse', metrics=['accuracy'])
```

Optimizers


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop


SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5

Step 3. Fit Network

```
history = model.fit(X, y, batch_size=10, epochs=100)
```

The network is trained using the backpropagation algorithm

- Batch size is the number of samples that going to be propagated through the network.
- epochs is the number of training times (for **ALL** the training examples)

Example: if you have 1000 training examples, and your batch size is 500, then it will take 2 iterations to complete 1 epoch.

-

Step 4. Evaluate Network

- The model evaluates the loss across all of the test patterns, as well as any other metrics specified when the model was compiled.

- For example, for a model compiled with the accuracy metric, we could evaluate it on a new dataset as follows:

```
loss, accuracy = model.evaluate(X, Y)
print("Loss: %.2f, Accuracy: %.2f% %" % (loss, accuracy*100))
```

Step 5. Make Predictions

```
probabilities = model.predict(X)
predictions = [float(round(x)) for x in probabilities]
accuracy = numpy.mean(predictions == Y)  #count the number of True and divide by the total size
print("Prediction Accuracy: %.2f% %" % (accuracy*100))
```

Binary classification using Neural Network in Keras

Diabetes Data Set

Detect Diabetes Disease based on analysis

Dataset Attributes:

1. Number of times pregnant
2. Plasma
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index
7. Diabetes pedigree function
8. Age (years)
9. Class variable (0 or 1)

```
# Sample Multilayer Perceptron Neural Network in Keras
from keras.models import Sequential
from keras.layers import Dense
import numpy
# load and prepare the dataset
dataset = numpy.loadtxt("pima-indians-diabetes.csv", delimiter=",")
X = dataset[:,0:8]
Y = dataset[:,8]
# 1. define the network
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# 2. compile the network
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# 3. fit the network
history = model.fit(X, Y, epochs=100, batch_size=10)
# 4. evaluate the network
loss, accuracy = model.evaluate(X, Y)
print("\nLoss: %.2f, Accuracy: %.2f%%" % (loss, accuracy*100))
# 5. make predictions
probabilities = model.predict(X)
predictions = [float(round(x)) for x in probabilities]
accuracy = numpy.mean(predictions == Y)
print("Prediction Accuracy: %.2f%%" % (accuracy*100))
```

Save prediction model

- After train our model, ie, model.fit(X_train, Y_train), we can save this traning to use later.
- This task can done by **Pickle** package(Python Object Serialization Library), using dump and load methods. Pickle can save any object not just the prediction model.

```
import pickle
.....
model.fit(X_train, Y_train)
# save the model to disk
pickle.dump(model, open("c:/data.dump", 'wb'))      #wb= write bytes

# some time later... load the model from disk
model = pickle.load(open("c:/data.dump", 'rb'))      #rb= read bytes
```

Spark

How to handel spark using Python (PySpark)

SPARK

- Platform designed for handling very large datasets, that allows you to use data transforms and machine learning algorithms on top of it.
- Cloudera is now replacing MapReduce with Spark as the default processing engine in all of its Hadoop implementations. also, Spark does not include its own distributed storage layer, and as such it may take advantage of Hadoop's distributed filesystem (HDFS), among other technologies unrelated to Hadoop

- To run spark type pyspark
- In previous versions of Spark, the spark-shell created a **SparkContext** (sc), but in new Spark 2.0, the spark-shell creates a **SparkSession** (*spark*).

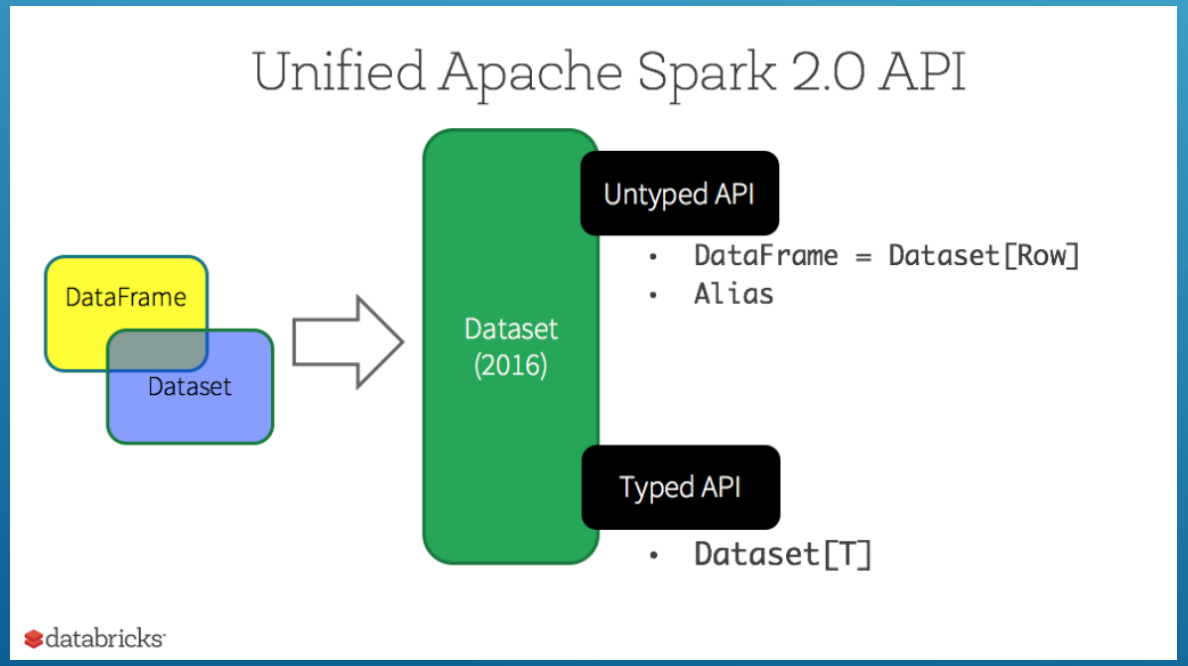
- **Spark Context** (deprecated) allows the users to handle the managed spark cluster and PySpark has Spark Context available as **sc**.

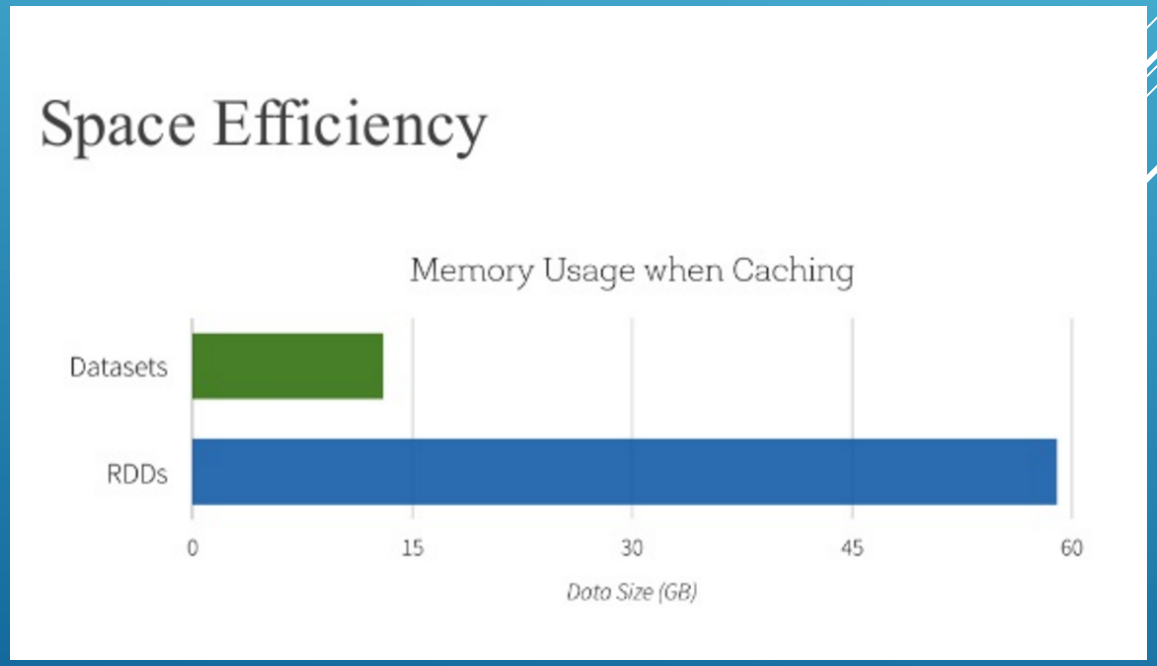
- In Spark 2.0, **SparkSession** creating SparkConf, SparkContext and SQLContext, as they're encapsulated within the SparkSession, SparkSession available as **'spark'**

Spark data containers

- Spark 1.0 used the RDD API
- Spark 1.3 introduced DataFrame API to improve the performance and scalability
- Spark 1.6 introduces a preview of the new Dataset API , introduces the concept of a schema
- Spark 2.0 merge DataFrame and Dataset to become one component (DataFrame become an alias to Dataset[Row])

Note: Since Python and R have no compile-time type-safety, we only have untyped APIs, namely DataFrames.





Load Data to Spark 2.0

```
# Load plain text files (e.g. CSVs) from different sources
df = spark.read.load("hdfs:///user/cloudera/ml-100k/u.data", minPartitions=1)
df = spark.read.load("file:///home/alex/ml-100k/u.data")
df = spark.read.load("examples/src/main/resources/users.parquet")
df = spark.read.load("examples/src/main/resources/people.json", format="json")
df = spark.read.parquet("examples/src/main/resources/users.parquet")
df = spark.read.json("examples/src/main/resources/people.json")
df = spark.read.csv('file:///home/filamentData.csv', mode="DROPMALFORMED",inferSchema=True, header = True)



#Load data from existing Hive table
df = spark.sql("SELECT key, value FROM src WHERE key < 10 ORDER BY key")

df.show()                #Show dataframe data
df.printSchema()      #Show Columns types
```

Spark: Load data with predefined schema

```
from pyspark.sql.types import *
Column1 = StructField("Column1",StringType(),True)
Column2 = StructField("Column2",StringType(),True)
Column3 = StructField("Column3",FloatType() ,True)
DataSchema = StructType([Column1,Column2, Column3])

data1 = spark.read.csv('file:///home/filamentData.csv',header=True, schema =DataSchema, mode="DROPMALFORMED")

Data_DataFrame2 = data1.filter(data1.Column2 == '100W')
Data_DataFrame3 = data1.filter((data1.Column2 == '100W') & (data1.Column3 > 650.0))
```

Convert DataFrame to memory table to query using SQL

```
data1.createOrReplaceTempView("TableName1")
data1.cache()

TableName1DataFrame = spark.sql('select age, income from TableName1 limit 500')
TableName1DataFrame.show(10)



#Convert Memory Table to physical  hive table
spark.table("TableName1").write.saveAsTable("hive_table_Name")
```

DataFrame GroupBy

##In PySparkSQL, groupBy() function returns GroupedData object

##then apply avg(),sum(), count(), min(), max(), and sum() on GroupedData.

```
DataFrame6 = data1.groupBy('income').count()                    #Return two columns (income , count)
DataFrame7 = data1.groupBy(['income', 'sex']).mean('age')       #Return three columns (income, sex , age)
DataFrame8 = data1.groupBy('income').count().sort(['income','count'],ascending= 0)
```

SparkSQL Example

```
Data = [['filamentA','100W',605],['filamentB','100W',683],['filamentB','100W',691]]

from pyspark.sql.types import *
Column1 = StructField("Column1",StringType(),True)
Column2 = StructField("Column2",StringType(),True)
Column3 = StructField("Column3",StringType(),True)
DataSchema = StructType([Column1,Column2, Column3])

DataFrame1 =spark.createDataFrame(Data, DataSchema)
DataFrame1 = DataFrame1.withColumn('Column3',DataFrame1.Column3.cast(FloatType()))    #Update Column DataType

Data_DataFrame2 = DataFrame1.filter(DataFrame1.Column2 == '100W')
Data_DataFrame3 = DataFrame1.filter((DataFrame1.Column2 == '100W') & (DataFrame1.Column3 > 650.0))

DataFrame1.createOrReplaceTempView("Table1")         #Convert dataFrame to MemoryTable
SQL_DataFrame1 = spark.sql("SELECT * FROM Table1 where Column2='100W' ")

spark.table("Table1").write.saveAsTable("zips_hive_table")
#Select from Hive table
sqlDF = spark.sql("SELECT * FROM zips_hive_table")
sqlDF.show()
```

Prepare dataset for Spark Machine Learning

- In order for the features to be used by a machine learning algorithm, the features are transformed and put into Feature Vectors, which are vectors of numbers representing the value for each feature.
- If the label column is of type string, it will be first transformed to double with StringIndexer.

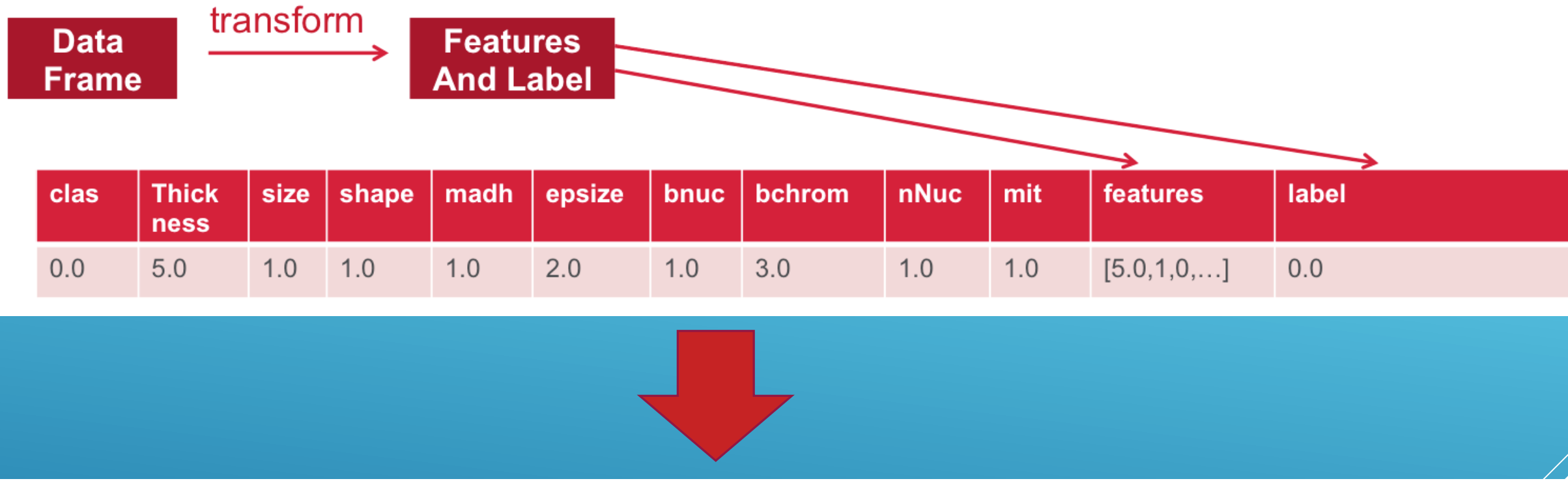
At the end we should have dataset contains two columns :

- First column is **Features** Column that contains a Vector of the input data features
- Second column is **Label** Column with type double and this is the target that we need to predict

- We can prepare dataset using **VectorAssembler**/**StringIndexer** or using **RFormula**

- **RFormula** produces a vector column of features and a double or string column of label. Like when formulas are used in R for linear regression, string input columns will be one-hot encoded, and numeric columns will be cast to doubles. If the label column is of type string, it will be first transformed to double with StringIndexer. If the label column does not exist in the DataFrame, the output label column will be created from the specified response variable in the formula.

After prepare the dataset we need to split it 70% for traning and 30% for testing using **randomSplit** function


Data Frame
transform
Features And Label
clas
Thick ness
size
shape
madh
epsize
bnuc
bchrom
nNuc
mit
features
label
0.0
5.0
1.0
1.0
1.0
2.0
1.0
3.0
1.0
1.0
[5.0,1,0,…]
0.0
Data Frame
transform
Features And Label
fit
model
transform
predictions
Test split
clas
Thick ness
size
…
features
label
Raw rediction
probability
prediciton
0.0
5.0
1.0
[5.0,1,0,…]
0.0
1.79
0.76
0.0

Prepare dataset using RFormula

```
from pyspark.ml.feature import RFormula
#dataset = spark.read.format("csv").  optVectorAssemblerion("header", "true").  option("inferSchema", "true"). load( "/gen_data.csv")
dataset = spark.createDataFrame( [(7, "US", 18, 1.0),    (8, "CA", 12, 0.0),   (9, "NZ", 15, 0.0)],    ["id", "country", "hour", "clicked"])

formula = RFormula(    formula="clicked ~ country + hour",   featuresCol="features",    labelCol="label")
Data4Train = formula.fit(dataset).transform(dataset)
Data4Train.select("features", "label").show()

training, test = Data4Train.randomSplit([0.6, 0.4], seed=11L)          # Split data into training (60%) and test (40%)
training.cache()
```

Notes:

Now, **Data4Train** will have two additional columns, features and label, and features's type would be Vector.

RFormula formula **clicked ~ country + hour**, indicates that we want to predict **clicked** based on **country** and **hour**

Prepare dataset using VectorAssembler, StringIndexer

```
dataset = spark.read.format("csv").  optVectorAssemblerion("header", "true").  option("inferSchema", "true"). load( "/generate.csv")
dataset.printSchema()

from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
#Assume dataset contains the next columns ["id", "hour", "mobile", "userFeatures", "clicked"]
assembler = VectorAssembler(    inputCols=["hour", "mobile", "userFeatures"],    outputCol="features")
data = assembler.transform(dataset)

indexer = StringIndexer(inputCol="clicked", outputCol="label")
data1 = indexer.fit(data).transform(data)
data1.select("features", "label").show()

training, test = data1.randomSplit([0.6, 0.4], seed=11L)          # Split data into training (60%) and test (40%)
training.cache()
```

Spark ML LogisticRegression Example

```
dataset = spark.read.format("csv").  option("header", "true").  option("inferSchema", "true"). load( "/generated_data.csv")
dataset.printSchema()

from pyspark.ml.feature import RFormula
formula = RFormula(formula="Class ~Number_pregnant",  featuresCol="features",  labelCol="label")
data = formula.fit(dataset).transform(dataset)
data.select("features", "label").show()
training, test = data.randomSplit([0.6, 0.4], seed=11L)          # Split data into training (60%) and test (40%)
training.cache()

from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)
lrModel = lr.fit(training)
print("Coefficients: \n" + str(lrModel.coefficientMatrix))
print("Intercept: " + str(lrModel.interceptVector))
predictions = lrModel.transform(test)
predictions.show()
```

Other training algorithms (LinearSVC)

```
#Assume we have training, test datasets ready, like we do before
training, test = data.randomSplit([0.6, 0.4], seed=11L)          # Split data into training (60%) and test (40%)
```

```
from pyspark.ml.classification import LinearSVC
lsvc = LinearSVC(maxIter=10, regParam=0.1)
lsvcModel = lsvc.fit(training)
print("Coefficients: " + str(lsvcModel.coefficients))
print("Intercept: " + str(lsvcModel.intercept))

predictions = lsvcModel.transform(test)
```

Other training algorithms (NaiveBayes)

```
#Assume we have training, test datasets ready, like we do before
training, test = data.randomSplit([0.6, 0.4], seed=11L)          # Split data into training (60%) and test (40%)


from pyspark.ml.classification import NaiveBayes
nb = NaiveBayes(smoothing=1.0, modelType="multinomial")
model = nb.fit(train)
predictions = model.transform(test)
predictions.show()
# compute accuracy on the test set
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction",metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy = " + str(accuracy))
```

Other training algorithms (LinearRegression)

```
#Assume we have training, test datasets ready, like we do before
training, test = data.randomSplit([0.6, 0.4], seed=11L)          # Split data into training (60%) and test (40%)


from pyspark.ml.regression import LinearRegression
lr = LinearRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)
lrModel = lr.fit(training)
print("Coefficients: %s" % str(lrModel.coefficients))
print("Intercept: %s" % str(lrModel.intercept))
trainingSummary = lrModel.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)
```

IBM Data Science Experience

Run your machine learning/Deep learning projects on IBM Cloud!

Welcome Mohammed!

IBM Data Science Experience and IBM Data Catalog are part of Watson Data Platform.

Try out other Watson Data Platform apps.

Get started with key tasks

- New project
- Catalog and find data
- New notebook
- New SPSS Modeler flow
- New model
- New streams flow

Hide

Recently updated projects

No projects to show

You haven't made a project yet.

New Project

New project

Define project details

Name

Project1

92

Description

Project description

3000

Choose project options

- [x] Restrict who can be a collaborator
- [x] Add a compute engine for data analysis

Define storage

Select storage type

- ○ Object Storage (Swift API)
- ● IBM Cloud Object Storage

Target Cloud Object Storage Instance

cloud-object-storage-mh

Define compute engine

Select Spark service

Spark™-em

⚠ If you associate the same Spark service with multiple projects, the Spark history server will display job history information for all the projects.

Cancel Create

Overview Assets Bookmarks Deployments Collaborators Settings

What assets are you looking for?

Data assets

0 assets selected.

NAME	TYPE	SERVICE	CREATED BY	LAST MODIFIED	ACTIONS
you currently have no data assets					

Notebooks

New notebook

NAME	SHARED	SCHEDULED	STATUS	LANGUAGE	LAST EDITOR	LAST MODIFIED	ACTIONS
you currently have no notebooks							

Streams flows

New streams flow

NAME	MODIFIED BY	LAST MODIFIED	ACTIONS
you currently have no streams flows			

Models

New model

NAME	STATUS	RUNTIME	LAST MODIFIED	ACTIONS
you currently have no models				

Load Files Catalog

Drop file here or browse your files to add a new file

Overview Assets Bookmarks Deployments Collaborators Settings

What assets are you looking for?

Data assets

0 assets selected.

NAME	TYPE	SERVICE	CREATED BY	LAST MODIFIED	ACTIONS
Concrete_Data.xls	Data Asset	Project	Mohammed Rafie	14 Dec 2017	⋮

Notebooks

New notebook

NAME	SHARED	SCHEDULED	STATUS	LANGUAGE	LAST EDITOR	LAST MODIFIED	ACTIONS
you currently have no notebooks							

Streams flows

New streams flow

NAME	MODIFIED BY	LAST MODIFIED	ACTIONS
you currently have no streams flows			

Models

New model

NAME	STATUS	RUNTIME	LAST MODIFIED	ACTIONS
you currently have no models				

Load Files Catalog

Find in storage

0 selected

Concrete_Data.xls

New notebook

Blank From file From URL

Name*

FileAnalysis1

37 Characters Remaining

Description

Type your Description here

Language*

● Python 2 ○ R ○ Scala ○ Python 3.5 Experimental

Spark version*

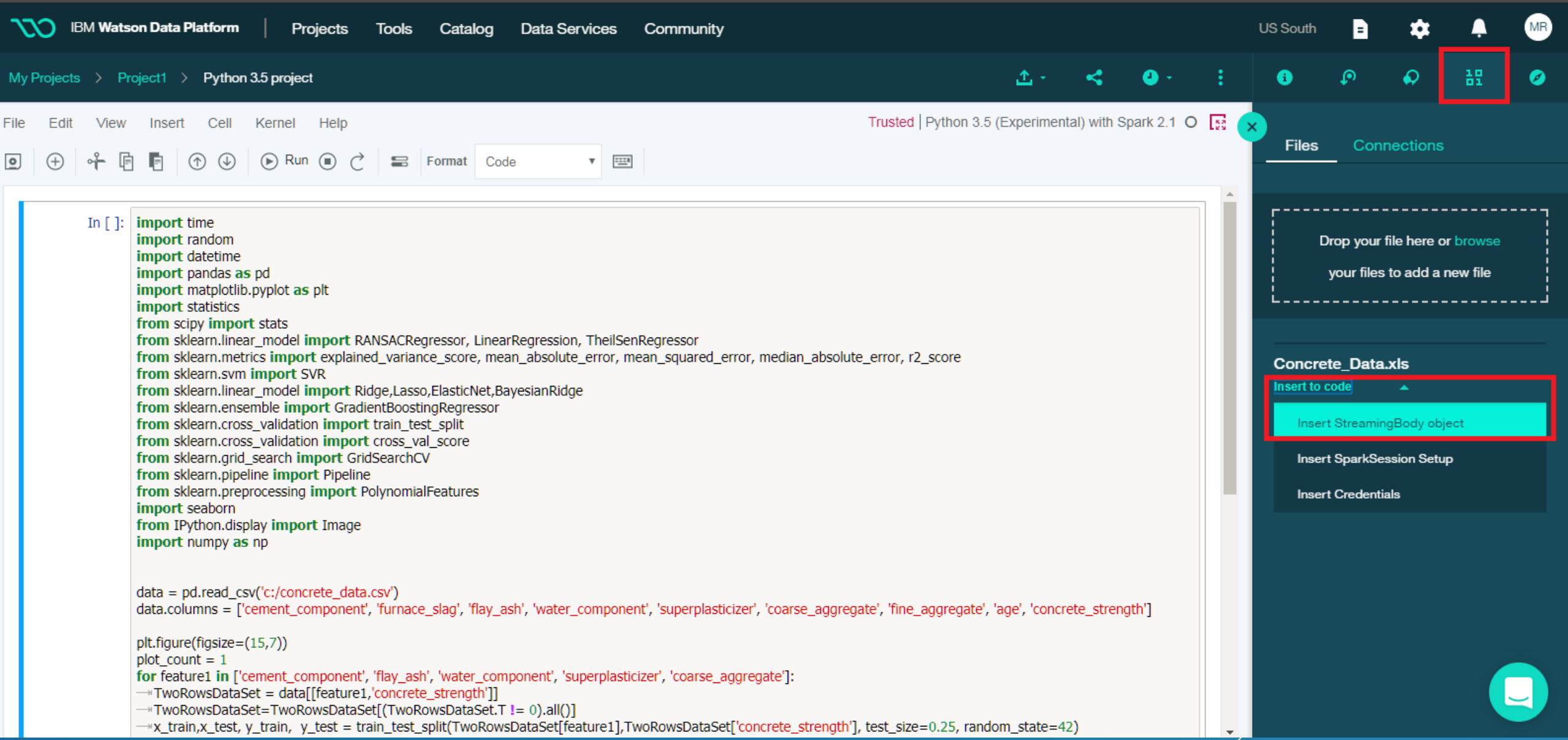
● 2.1 ○ 2.0

Spark service*

Spark™-em

Cancel Create Notebook

In [ ]:



File should be cvs with coma separators between fields not Excel to handle it without problems!

In [ ]:

```
import time
import random
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import statistics
from scipy import stats
from sklearn.linear_model import RANSACRegressor, LinearRegression, TheilSenRegressor
from sklearn.metrics import explained_variance_score, mean_absolute_error, mean_squared_error, median_absolute_error, r2_score
from sklearn.svm import SVR
from sklearn.linear_model import Ridge,Lasso,ElasticNet,BayesianRidge
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.model_selection import train_test_split
#from sklearn.cross_validation import train_test_split
from sklearn.model_selection import cross_val_score
#from sklearn.cross_validation import cross_val_score
from sklearn.model_selection import GridSearchCV
#from sklearn.grid_search import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
import seaborn
from IPython.display import Image
import numpy as np

data = pd.read_csv('c:/concrete_data.csv')
data.columns = ['cement_component', 'furnace_slag', 'flay_ash', 'water_component', 'superplasticizer', 'coarse_aggregate', 'fine_aggregate', 'age', 'concrete_strength']

plt.figure(figsize=(15,7))
plot_count = 1
for feature1 in ['cement_component', 'flay_ash', 'water_component', 'superplasticizer', 'coarse_aggregate']:
    TwoRowsDataSet = data[[feature1,'concrete_strength']]
    TwoRowsDataSet=TwoRowsDataSet[(TwoRowsDataSet.T != 0).all()]
    x_train,x_test, y_train,  y_test = train_test_split(TwoRowsDataSet[feature1],TwoRowsDataSet['concrete_strength'], test_size=0.25, random_state=42)
    # Create regression object
    regr = LinearRegression()
    # Train the model using the training sets
    #in scikit-learn version everything has to be a 2d matrix, even a single column or row.
    #convert x_train to 2d by values.reshape(-1, 1)
    regr.fit(x_train.values.reshape(-1, 1), y_train.values.reshape(-1, 1))
    y_pred = regr.predict(x_test.values.reshape(-1, 1))
```

```
import numpy as np
import sys
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share your notebook.
client_39564de3342c490d97f3c4a1978ea981 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='l-3xXOs-djRK4JuqKKTaJd2CSbQ_GM8xHKI7uEaYsd-c',
    ibm_service_instance_id="iam-ServiceId-9c5676b7-9515-42ea-9a59-38cdf766a5d1",
    ibm_auth_endpoint="https://iam.ng.bluemix.net/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

body = client_39564de3342c490d97f3c4a1978ea981.get_object(Bucket='project10ade4e75ae9a444d84d00a2b659f9117',Key='Concrete_Data.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

data = pd.read_csv(body)  #data = pd.read_csv('c:/concrete_data.csv')

data.head()

data.columns = ['cement_component', 'furnace_slag', 'flay_ash', 'water_component', 'superplasticizer', 'coarse_aggregate', 'fine_aggregate', 'age', 'concrete_strength']

plt.figure(figsize=(15,7))
plot_count = 1
for feature1 in ['cement_component', 'flay_ash', 'water_component', 'superplasticizer', 'coarse_aggregate']:
    TwoRowsDataSet = data[[feature1,'concrete_strength']]
    TwoRowsDataSet=TwoRowsDataSet[(TwoRowsDataSet.T != 0).all()]
    x_train,x_test, y_train,  y_test = train_test_split(TwoRowsDataSet[feature1],TwoRowsDataSet['concrete_strength'], test_size=0.25, random_state=42)
    # Create regression object
    regr = LinearRegression()
    # Train the model using the training sets
    #in scikit-learn version everything has to be a 2d matrix, even a single column or row.
    #convert x_train to 2d by values.reshape(-1, 1)
    regr.fit(x_train.values.reshape(-1, 1), y_train.values.reshape(-1, 1))
    y_pred = regr.predict(x_test.values.reshape(-1, 1))
```


Drop your file here or browse

your files to add a new file

Concrete_Data.csv

Insert to code

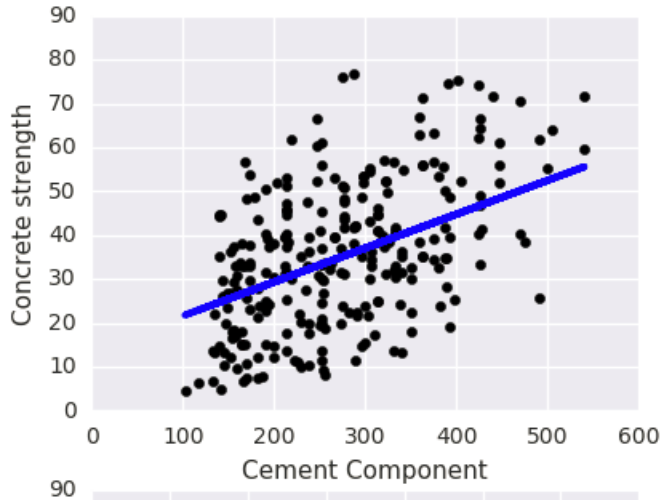
daily-total-female-births-in-cal....

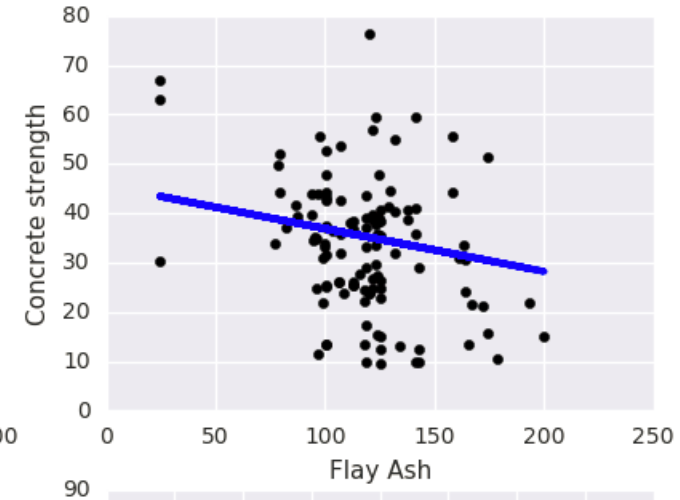
Insert to code

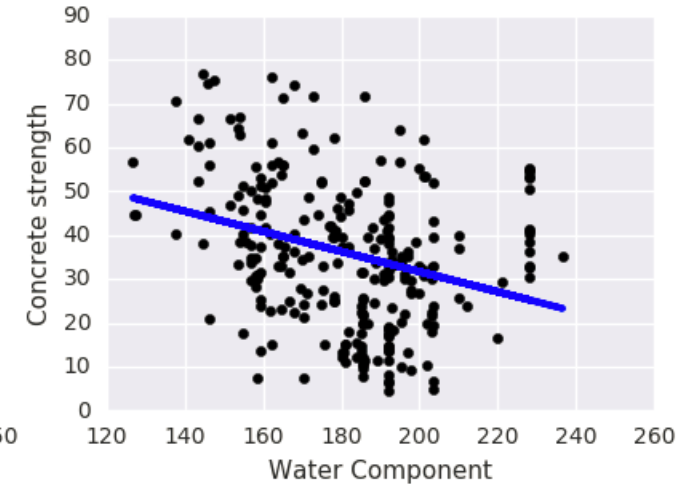
In [6]:

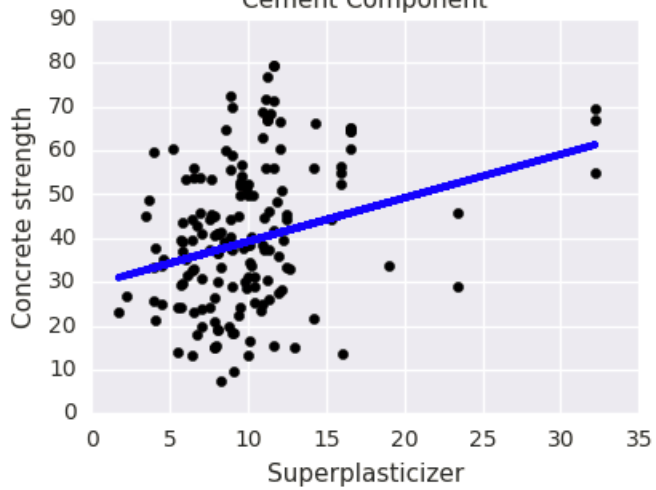
```
import time, random, datetime
import pandas as pd
import matplotlib.pyplot as plt
import statistics
from scipy import stats
from sklearn.linear_model import RANSACRegressor, LinearRegression, TheilSenRegressor,Ridge,Lasso,ElasticNet,BayesianRidge
from sklearn.metrics import explained_variance_score, mean_absolute_error, mean_squared_error, median_absolute_error, r2_score
from sklearn.svm import SVR
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.cross_validation import train_test_split, cross_val_score
from sklearn.grid_search import GridSearchCV
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
import seaborn
from IPython.display import Image
import numpy as np

data = pd.read_csv('c:/concrete_data.csv')
data.head()
data.columns = ['cement_component', 'furnace_slag', 'flay_ash', 'water_component', 'superplasticizer', 'coarse_aggregate', 'fine_aggregate', 'age', 'concrete_strength']
plt.figure(figsize=(15,7))
plot_count = 1
for feature1 in ['cement_component', 'flay_ash', 'water_component', 'superplasticizer', 'coarse_aggregate']:
    TwoRowsDataSet = data[[feature1,'concrete_strength']]
    TwoRowsDataSet=TwoRowsDataSet[(TwoRowsDataSet.T != 0).all()]
    x_train,x_test, y_train,  y_test = train_test_split(TwoRowsDataSet[feature1],TwoRowsDataSet['concrete_strength'], test_size=0.25, random_state=42)
    regr = LinearRegression()
    # Train the model using the training sets
    #in scikit-learn version everything has to be a 2d matrix, even a single column or row. convert x_train to 2d by values.reshape(-1, 1)
    regr.fit(x_train.values.reshape(-1, 1), y_train.values.reshape(-1, 1))
    y_pred = regr.predict(x_test.values.reshape(-1, 1))
    # Plot outputs
    plt.subplot(2,3,plot_count)
    plt.scatter(x_test, y_test, color='black')
    plt.plot(x_test, y_pred, color='blue', linewidth=3)
    plt.xlabel(feature1.replace('_',' ').title())
    plt.ylabel('Concrete strength')
    #print feature1, r2_score(y_test, y_pred)
    plot_count+=1
plt.show()
```

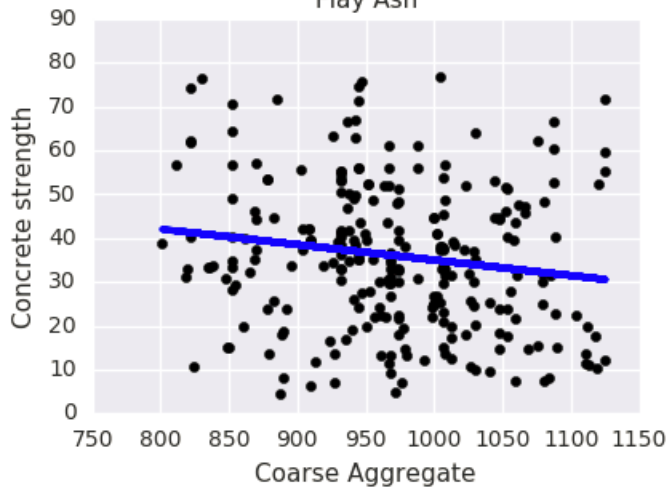
```
    plt.ylabel('Concrete strength')
    #print feature1, r2_score(y_test, y_pred)
    plot_count+=1
plt.show()
```









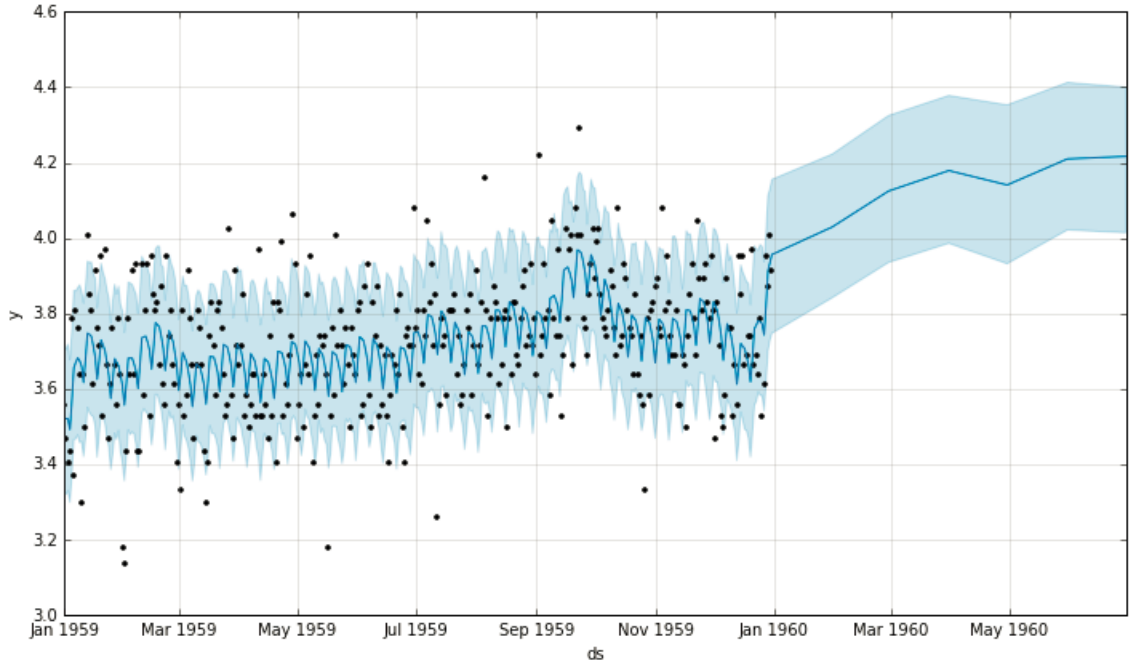


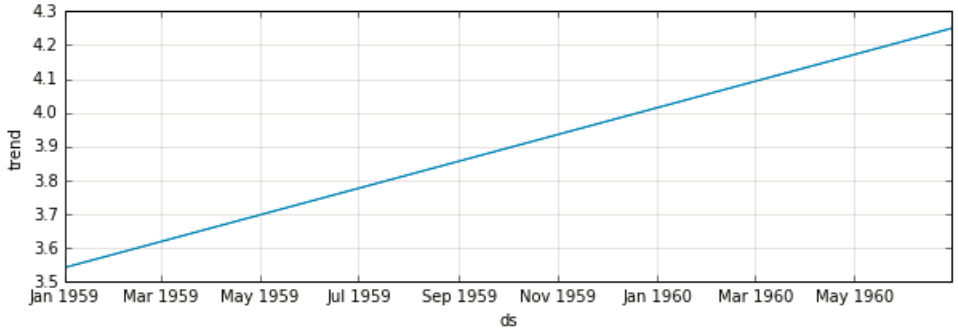
To install new package

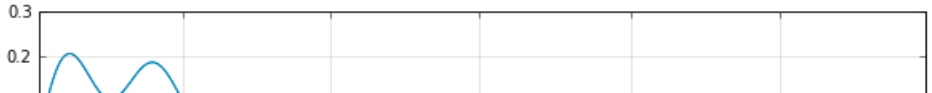
- !pip install --user fbprophet

In [7]:

```
import pandas as pd
import numpy as np
from fbprophet import Prophet
import matplotlib.pyplot as plt

data = pd.read_csv('../examples/example_wp_peyton_manning.csv')
data = pd.read_csv(body)
data.head()

data.columns = ['ds', 'y']
data['y'] = np.log(data['y'])
# Python
m = Prophet( yearly_seasonality=True , weekly_seasonality=True , daily_seasonality=True )
m.fit(data);

# add new 365 days to dataframe to allow predict them
future = m.make_future_dataframe(periods=365)
# add new 6 rows which are 6 months to predict them
future = m.make_future_dataframe(periods=6, freq = 'm')
future.tail()

# predict will create new dataframe with 4 columns
# yhat forecast value
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()

#draw the forecast with log transformation
#m.plot(forecast);

# draw weekly,monthly, and yearly trends  with log transformation
#m.plot_components(forecast);

# get forecast without log transformation
data['y'] = np.exp(data['y'])
forecast['yhat'] = np.exp(forecast['yhat'])
forecast['yhat_lower'] = np.exp(forecast['yhat_lower'])
forecast['yhat_upper'] = np.exp(forecast['yhat_upper'])
m.plot(forecast);
m.plot_components(forecast);
plt.show()
```







In [4]:
```
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

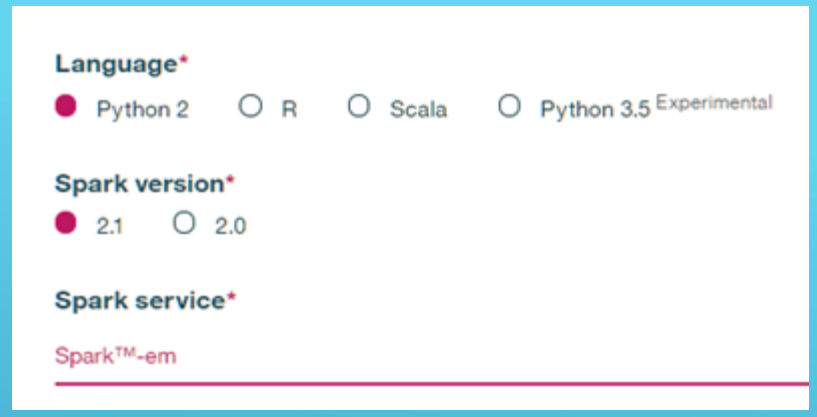
Out[4]:

	ds	yhat	yhat_lower	yhat_upper
366	1960-02-29	61.817855	50.285631	75.378767
367	1960-03-31	65.273125	53.744467	79.452100
368	1960-04-30	62.828136	51.031410	75.765555
369	1960-05-31	67.331808	54.366985	81.864730
370	1960-06-30	67.783788	55.422102	82.289539

Spark Example

Create new notebook to write Spark code (Python 2, Spark 2.1).

When click on "Insert SparkSession DataFrame" you will get the highlighted code, and to allow Spark to detect the correct data types add "option("inferSchema","True")"



IBM Data Science Experience | Projects Tools Data Services Community US South MR

My Projects > Spark > spark2

File Edit View Insert Cell Kernel Help

Not Connected Trusted | Python 2 with Spark 2.1

Run Format Code

In [14]:

```
import ibmos2spark

# @hidden_cell
credentials = {
    'endpoint': 'https://s3-api.us-geo.objectstorage.service.networklayer.com',
    'api_key': 's-8Rx0QNh1FP8CSael5cmFppdjZKAQlXJF_Xk8h7KgeZ',
    'service_id': 'iam-ServiceId-9fe9e05e-cd7b-4693-9884-4b3455d6c18b',
    'iam_service_endpoint': 'https://iam.ng.bluemix.net/oidc/token'}

configuration_name = 'os_f813519f89754a4bbc5257b29d035c70_configs'
cos = ibmos2spark.CloudObjectStorage(sc, credentials, configuration_name, 'bluemix_cos')

from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
dataset = spark.read\
  .format('org.apache.spark.sql.execution.datasources.csv.CSVFileFormat')\
  .option('header', 'true').option("inferSchema","True")\
  .load(cos.url('pima-indians-diabetes.csv', 'sparkb41f696acfd147cfad2de6d6f97cd67c'))
#dataset.take(5)
dataset.printSchema()
```

Files Connections

Drop your file here or browse

your files to add a new file

pima-indians-diabetes.csv

Insert to code

Insert pandas DataFrame

Insert SparkSession DataFrame

Insert Credentials

In [14]:

```
import ibmos2spark
credentials = {
    'endpoint': 'https://s3-api.us-geo.objectstorage.service.networklayer.com',
    'api_key': 's-8Rx0QNh1FP8CSael5cmFppdjZKAQlXJF_Xk8h7KgeZ',
    'service_id': 'iam-ServiceId-9fe9e05e-cd7b-4693-9884-4b3455d6c18b',
    'iam_service_endpoint': 'https://iam.ng.bluemix.net/oidc/token'}

configuration_name = 'os_f813519f89754a4bbc5257b29d035c70_configs'
cos = ibmos2spark.CloudObjectStorage(sc, credentials, configuration_name, 'bluemix_cos')

from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
dataset = spark.read\
  .format('org.apache.spark.sql.execution.datasources.csv.CSVFileFormat')\
  .option('header', 'true').option("inferSchema","True")\
  .load(cos.url('pima-indians-diabetes.csv', 'sparkb41f696acfd147cfad2de6d6f97cd67c'))
#dataset.take(5)
dataset.printSchema()


from pyspark.ml.feature import RFormula
formula = RFormula(
    formula="Class ~Number_pregnant",
    featuresCol="features",
    labelCol="label")

data = formula.fit(dataset).transform(dataset)
data.select("features", "label").show()
# Split data into training (60%) and test (40%)
training, test = data.randomSplit([0.6, 0.4], seed=11L)
training.cache()
##############################
from pyspark.ml.classification import LogisticRegression
lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)
lrModel = lr.fit(training)
print("Coefficients: \n" + str(lrModel.coefficientMatrix))
print("Intercept: " + str(lrModel.interceptVector))

predictions = lrModel.transform(test)
predictions.show()
```

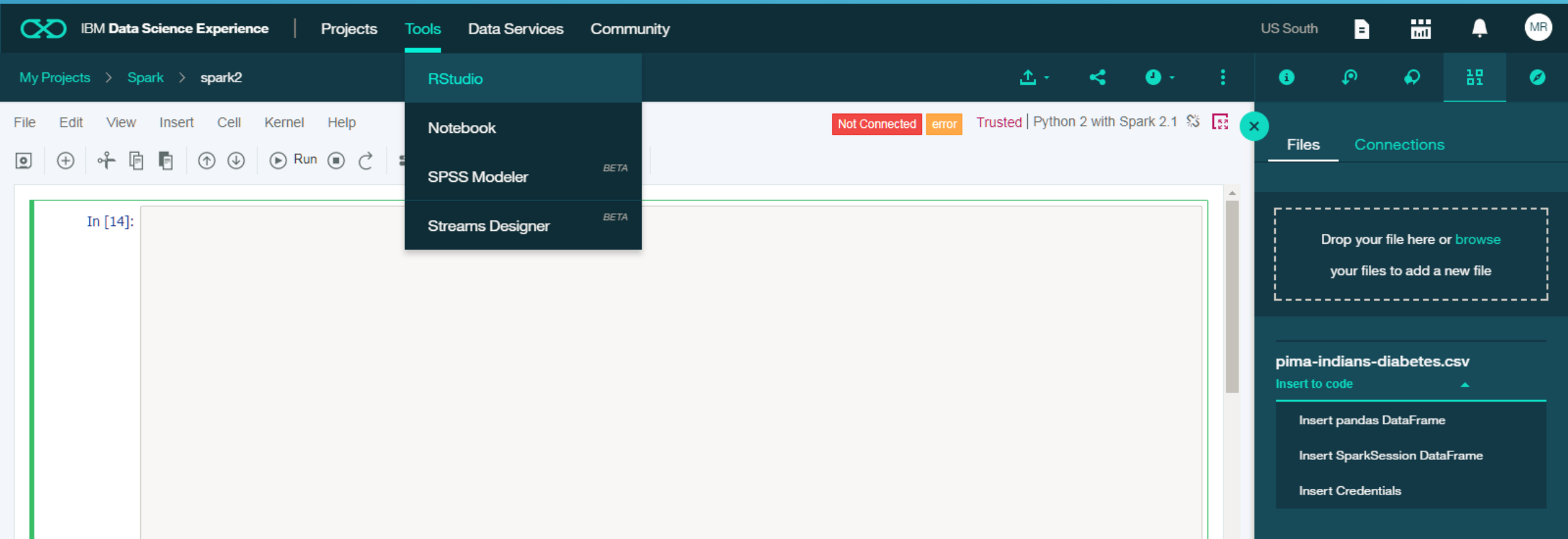
SPARK JOB PROGRESS Hide All

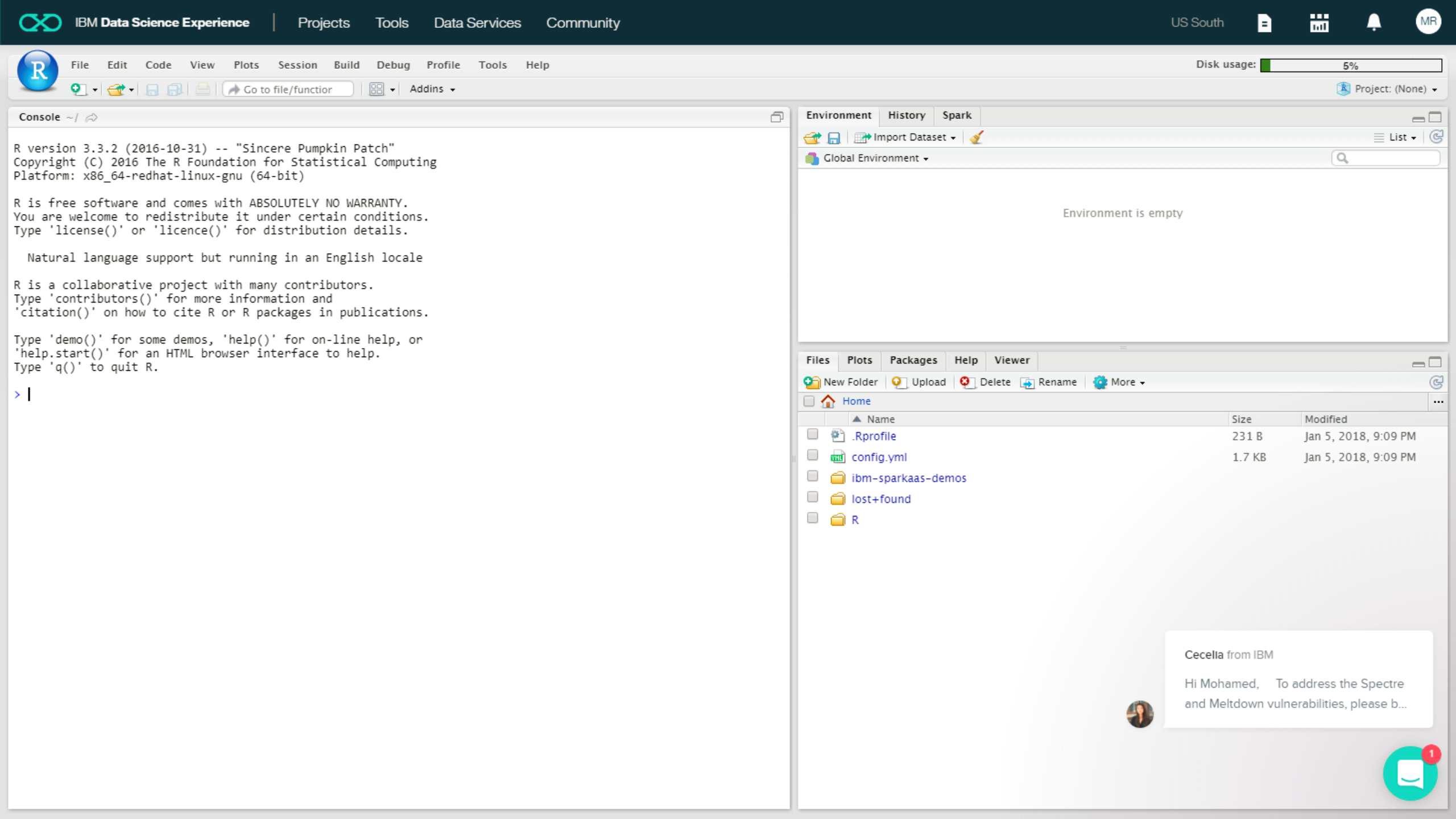
JOB	PROGRESS	DURATION	STATUS
43	1 stage	0.42 sec	
44	1 stage	0.28 sec	
45	1 stage	0.17 sec	
46	1 stage	0.95 sec	
47	1 stage	1.77 sec	
48	1 stage	0.05 sec	
49	1 stage	0.58 sec	

```
root
 |-- Number_pregnant: integer (nullable = true)
 |--  Plasma_test: integer (nullable = true)
 |-- blood_pressure : integer (nullable = true)
 |-- skin_fold_thickness: integer (nullable = true)
 |-- serum : integer (nullable = true)
 |-- Body_mass_index : double (nullable = true)
 |-- Diabetes_function : double (nullable = true)
 |-- Age : integer (nullable = true)
 |-- Class: integer (nullable = true)

+--------+-----+
|features|label|
+--------+-----+
|  [6.0]|  1.0|
|  [1.0]|  0.0|
|  [8.0]|  1.0|
|  [1.0]|  0.0|
|  [0.0]|  1.0|
|  [5.0]|  0.0|
|  [3.0]|  1.0|
| [10.0]|  0.0|
|  [2.0]|  1.0|
|  [8.0]|  1.0|
```

R Example

- Tools ➔ RStudio




IBM Data Science Experience
Projects
Tools
Data Services
Community
US South
MR
File Edit Code View Plots Session Build Debug Profile Tools Help
Disk usage: 5%
Go to file/functior
Addins
Project: (None)
Console ~/
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment History Spark
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder
Upload
Delete
Rename
More
Home
Name
Size
Modified
.Rprofile 231 B Jan 5, 2018, 9:09 PM
config.yml 1.7 KB Jan 5, 2018, 9:09 PM
ibm-sparkaas-demos
lost+found
R
Cecelia from IBM
Hi Mohamed, To address the Spectre and Meltdown vulnerabilities, please b...
1

Basket market analysis with R

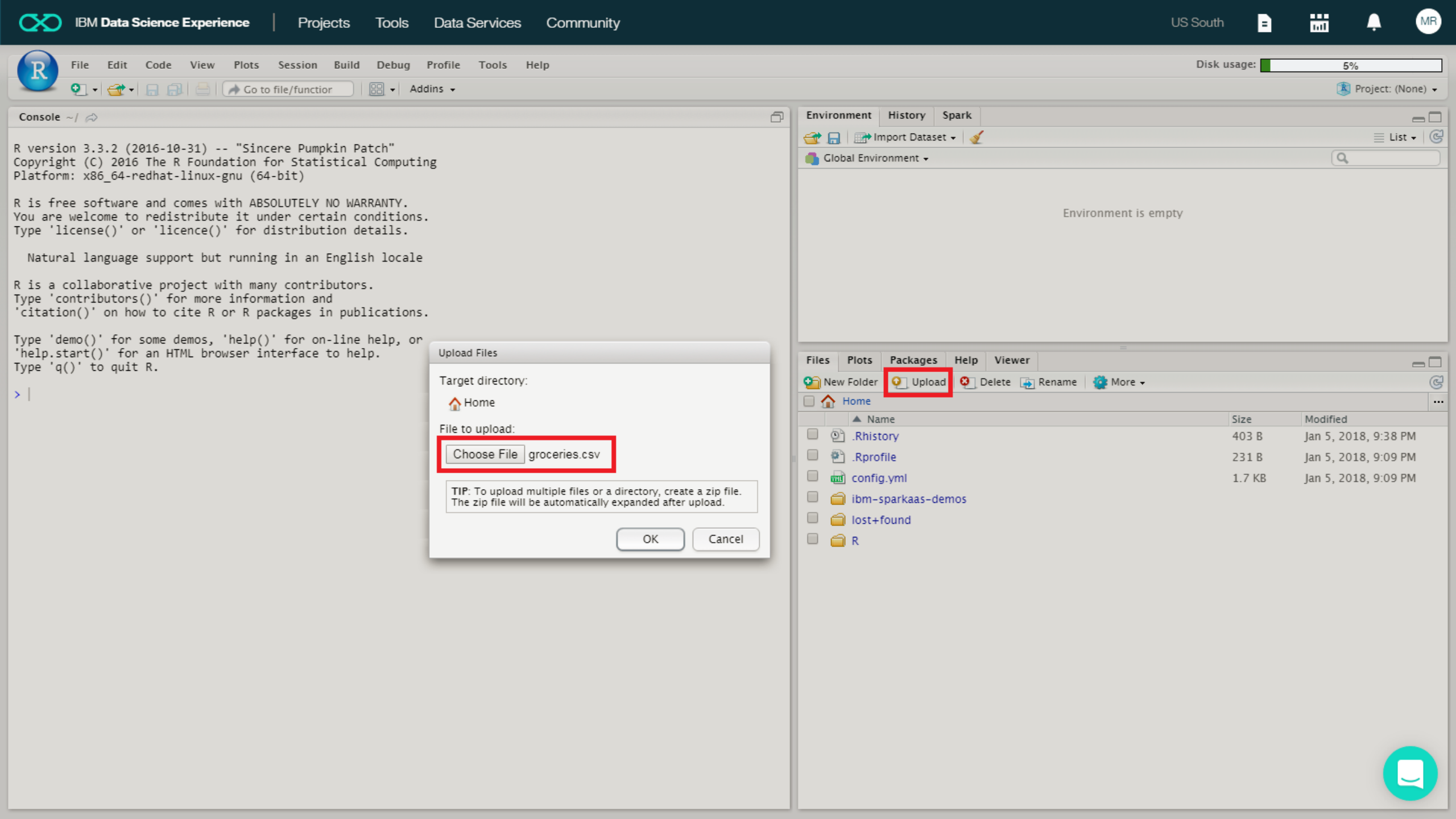
Upload “groceries.csv” dataset to the root R directory, the run the next code in sequence to get the relation between products that customers buy.

R code solution steps

- require(arules)
- SalesData =read.transactions("~/groceries.csv", sep=",")
- AssociationRules1 =apriori (SalesData, parameter = list (support = 0.007,confidence=0.25, minlen=2))
- inspect(sort(AssociationRules1, by="lift")[1:4])

groceries.csv

```
1  citrus fruit,semi-finished bread,margarine,ready soups
2  tropical fruit,yogurt,coffee
3  whole milk
4  pip fruit,yogurt,cream cheese ,meat spreads
5  other vegetables,whole milk,condensed milk,long life bakery product
6  whole milk,butter,yogurt,rice,abrasive cleaner
7  rolls/buns
8  other vegetables,UHT-milk,rolls/buns,bottled beer,liquor (appetizer)
9  pot plants
10 whole milk,cereals
11 tropical fruit,other vegetables,white bread,bottled water,chocolate
12 citrus fruit,tropical fruit,whole milk,butter,curd,yogurt,flour,bottled
13 beef
```


IBM Data Science Experience
Projects
Tools
Data Services
Community
US South
MR
File Edit Code View Plots Session Build Debug Profile Tools Help
Disk usage: 5%
Go to file/functior
Addins
Project: (None)
Console ~/
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-redhat-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Upload Files
Target directory:
Home
File to upload:
Choose File
groceries.csv
TIP: To upload multiple files or a directory, create a zip file.
The zip file will be automatically expanded after upload.
OK
Cancel
Environment History Spark
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
New Folder
Upload
Delete
Rename
More
Home
Name
Size
Modified
.Rhistory 403 B Jan 5, 2018, 9:38 PM
.Rprofile 231 B Jan 5, 2018, 9:09 PM
config.yml 1.7 KB Jan 5, 2018, 9:09 PM
ibm-sparkaas-demos
lost+found
R

File Edit Code View Plots Session Build Debug Profile Tools Help

Disk usage: 5%

Go to file/functior | Addins | Project: (None)

Console ~/

```
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> require(arules)
Loading required package: arules
Loading required package: Matrix

Attaching package: 'arules'

The following objects are masked from 'package:base':

    %in%, write

> SalesData =read.transactions("~/groceries.csv", sep=",")
> AssociationRules1 =apriori (SalesData, parameter = list (support = 0.007,confidence=0.25, minlen=2))

Parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
       0.25    0.1    1 none FALSE            TRUE   0.007      2     10  rules FALSE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [104 item(s)] done [0.00s].
creating transaction tree ... done [0.01s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [363 rule(s)] done [0.00s].
creating S4 object  ... done [0.01s].
> inspect(sort(AssociationRules1, by="lift")[1:4])
  lhs                     rhs                    support confidence     lift
1 {herbs}              => {root vegetables}    0.007015760  0.4312500 3.956477
2 {berries}            => {whipped/sour cream} 0.009049314  0.2721713 3.796886
3 {other vegetables,
   tropical fruit,
   whole milk}         => {root vegetables}    0.007015760  0.4107143 3.768074
4 {beef,
   other vegetables}   => {root vegetables}    0.007930859  0.4020619 3.688692
>
```

Environment | History | Spark

Import Dataset | List

Global Environment

Values

AssociationRules1	Formal class rules
SalesData	Large transactions (9835 elements, 760.7 Kb)

Files | Plots | Packages | Help | Viewer

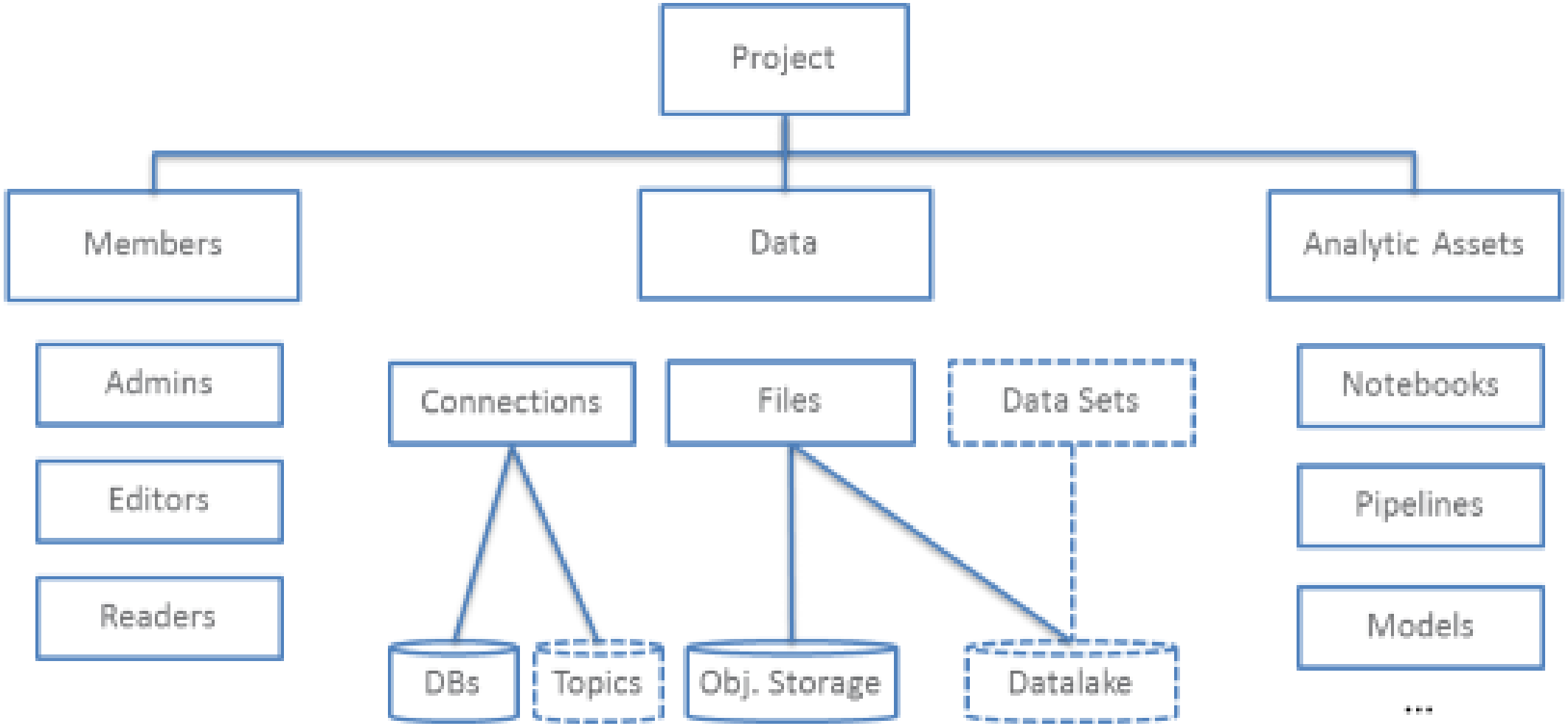
New Folder | Upload | Delete | Rename | More

Home

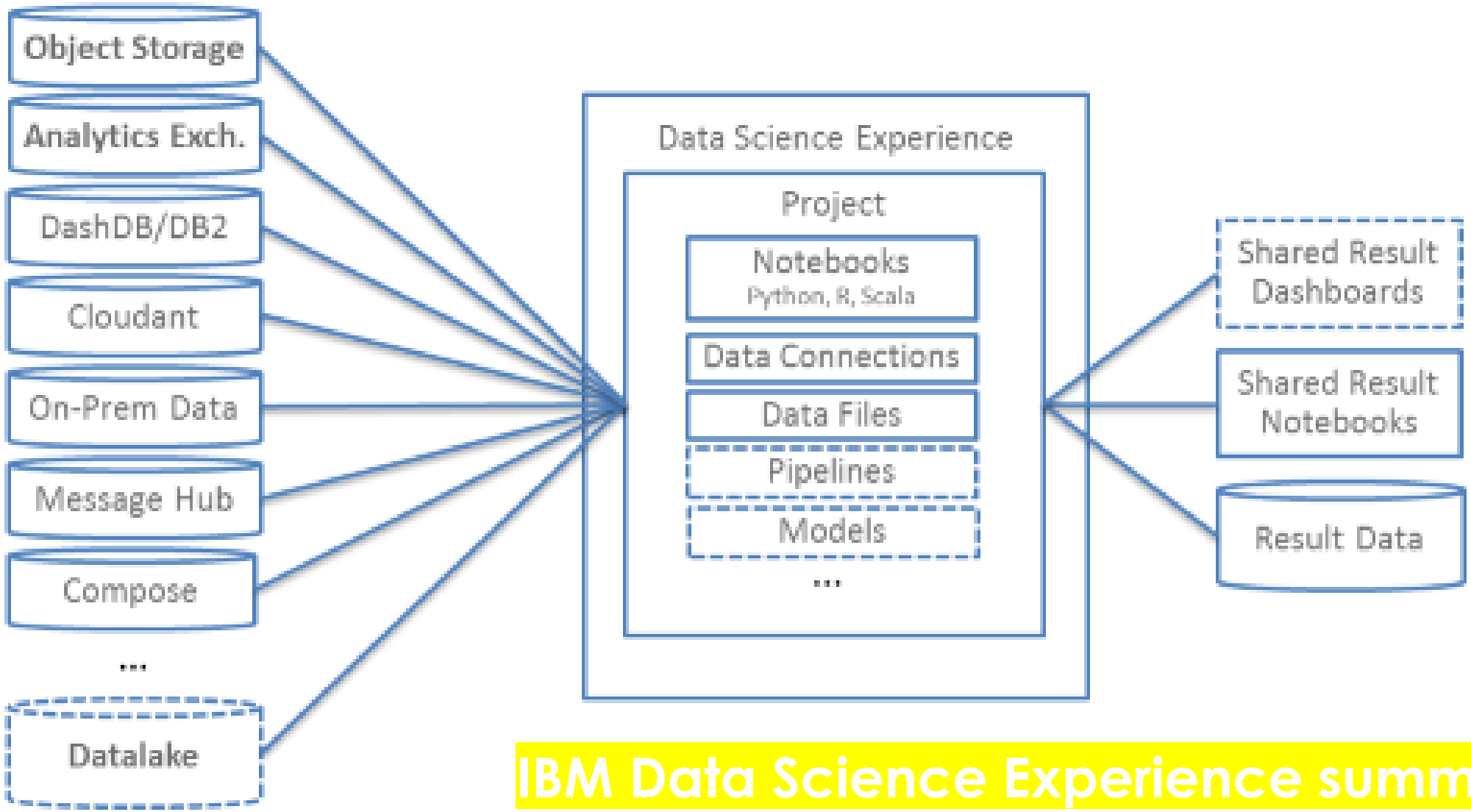
Name	Size	Modified
.Rhistory	403 B	Jan 5, 2018, 9:38 PM
.Rprofile	231 B	Jan 5, 2018, 9:09 PM
config.yml	1.7 KB	Jan 5, 2018, 9:09 PM
ibm-sparkaas-demos		
lost+found		
R		
groceries.csv	489.1 KB	Jan 5, 2018, 9:40 PM

IBM Data Science Experience summary

Projects enable Collaboration and bringing together Data and Analytics



Analyze Data from many Sources to generate Insights



IBM Data Science Experience summary

the IBM Data Science Experience (DSX) platform

Additional learning resources

- Learn more by trying out some interesting code patterns on DSX here : https://developer.ibm.com/code/patterns/category/data-science/

Perform a Machine Learning Exercise:

- https://developer.ibm.com/code/patterns/perform-a-machine-learning-exercise/ Correlate documents: https://developer.ibm.com/code/patterns/watson-document-correlation/

Lab - Create and deploy a scoring model to predict heart failure w/Bluemix and IBM Data Science Experience https://github.com/justinmccoy/watson-dojo-pm-tester