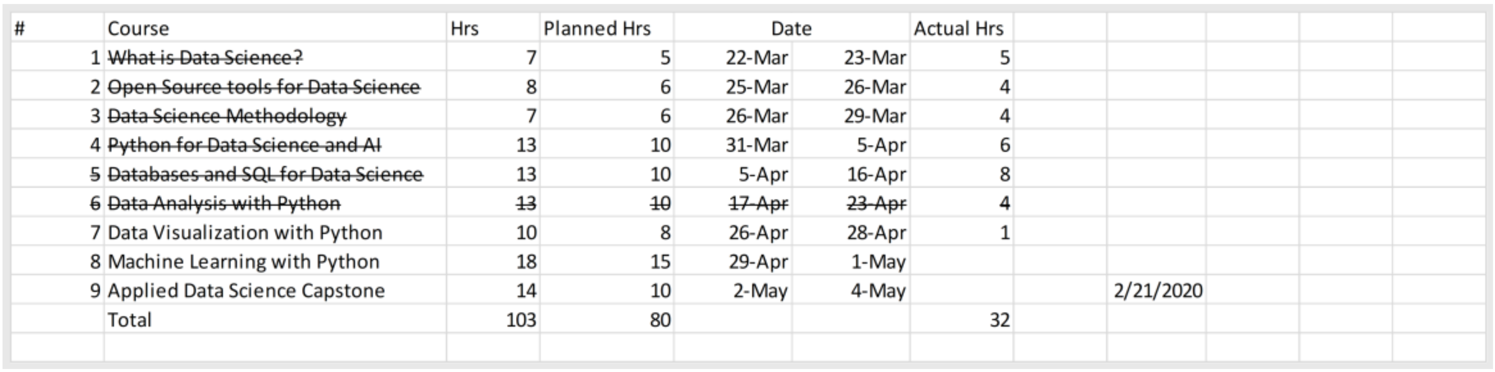
Courses

Thursday, March 26, 2020 11:43 AM

Courses - Spreadsheet



#	Course	Hrs	Planned Hrs	Date		Actual Hrs					
1	~~What is Data Science?~~	7	5	22-Mar	23-Mar	5					
2	~~Open Source tools for Data Science~~	8	6	25-Mar	26-Mar	4					
3	~~Data Science Methodology~~	7	6	26-Mar	29-Mar	4					
4	~~Python for Data Science and AI~~	13	10	31-Mar	5-Apr	6					
5	~~Databases and SQL for Data Science~~	13	10	5-Apr	16-Apr	8					
6	~~Data Analysis with Python~~	~~13~~	~~10~~	~~17-Apr~~	~~23-Apr~~	4					
7	Data Visualization with Python	10	8	26-Apr	28-Apr	1					
8	Machine Learning with Python	18	15	29-Apr	1-May						
9	Applied Data Science Capstone	14	10	2-May	4-May			2/21/2020			
	Total	103	80			32					

IBM cloud API key:
QuNcAVXzvsrR-8im-xpVDtgES8HUjaJmPgDiMAC1BabL
https://api.us-south.speech-to-text.watson.cloud.ibm.com/instances/18070760-abc0-4146-9f93-f2d1e0e96c82

Lang Transl
2Wnly8nETXLdnkVUoSjKIJhvDLRp3w_8Y9wf3-DthhOY
https://api.us-south.language-translator.watson.cloud.ibm.com/instances/9933dca8-73e1-41f0-89d0-c6e900073081

Good Stuff

Monday, March 23, 2020 9:27 PM

- use the keyboard shortcuts: [a] - Insert a Cell Above; [b] - Insert a Cell Below.
- https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

- max_value = df.['colname].idxmax() --> highest value in column
    - item_w_max_value = df.at[max_value , 'item_col_name]

Structural Bioinformatics Laboratory

https://github.com/sbl-sdsc

What is Data Science?

Monday, March 23, 2020 9:11 PM

Tools

Monday, March 23, 2020 9:16 PM

https://labs.cognitiveclass.ai/login?next=https%3A%2F%2Flabs.cognitiveclass.ai%2F

https://towardsdatascience.com/interactive-visualizations-in-jupyter-notebook-3be02ab2b8cd
https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks

R :
In your R script window, type "**x <- 1**" or "x = 1". Both do the same thing, but in R, variables are traditionally assigned using the "<-" syntax, which resembles a leftward-pointing arrow.

From <https://www.coursera.org/learn/open-source-tools-for-data-science/ungradedLti/bVDJr/lab-rstudio-the-basics>

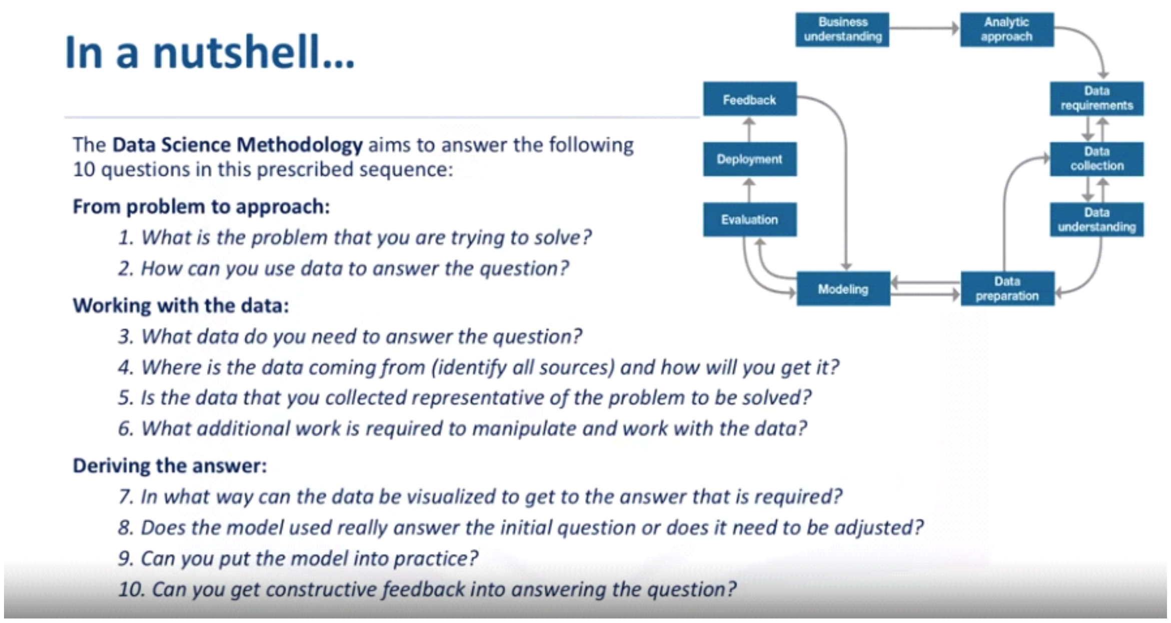
You can register for IBM Watson Studio here: https://cocl.us/Watson_Studio_Coursera_DS0105

Zapellin notebooks, you can use multiples languages in the same notebook

DS Methodology

Thursday, March 26, 2020 9:17 PM

John Rollins



In a nutshell...

The **Data Science Methodology** aims to answer the following 10 questions in this prescribed sequence:

From problem to approach:

1. *What is the problem that you are trying to solve?*
2. *How can you use data to answer the question?*

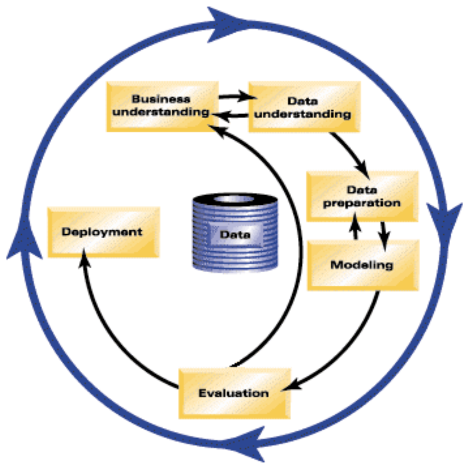
Working with the data:

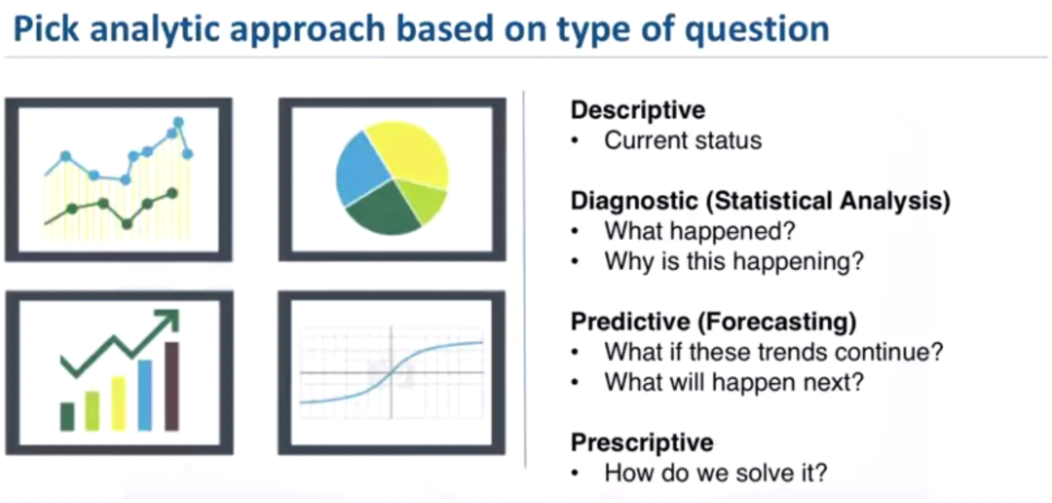
3. *What data do you need to answer the question?*
4. *Where is the data coming from (identify all sources) and how will you get it?*
5. *Is the data that you collected representative of the problem to be solved?*
6. *What additional work is required to manipulate and work with the data?*

Deriving the answer:

7. *In what way can the data be visualized to get to the answer that is required?*
8. *Does the model used really answer the initial question or does it need to be adjusted?*
9. *Can you put the model into practice?*
10. *Can you get constructive feedback into answering the question?*

CRISP-MD





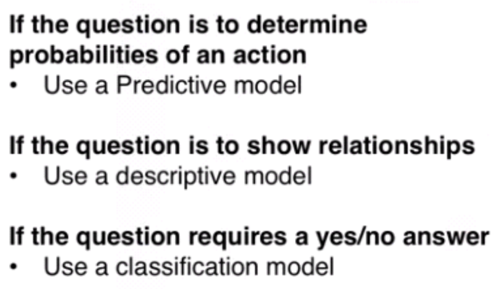
Pick analytic approach based on type of question

Descriptive
- Current status

Diagnostic (Statistical Analysis)
- What happened?
- Why is this happening?

Predictive (Forecasting)
- What if these trends continue?
- What will happen next?

Prescriptive
- How do we solve it?



If the question is to determine probabilities of an action
- Use a Predictive model

If the question is to show relationships
- Use a descriptive model

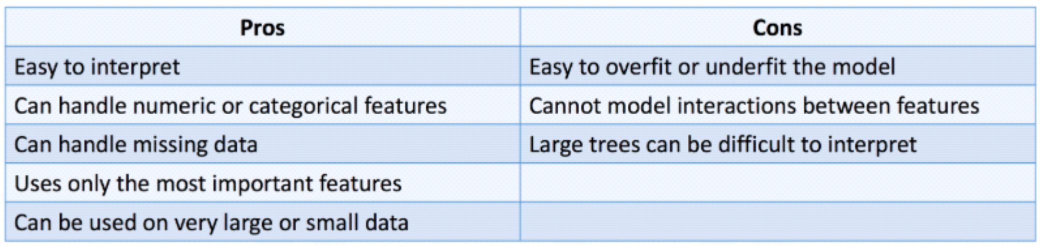
If the question requires a yes/no answer
- Use a classification model

Decision trees

- Decision trees are built using recursive partitioning to classify the data.
- When partitioning the data, decision trees use the most predictive feature (ingredient in this case) to split the data.
- Predictiveness is based on decrease in entropy - gain in information, or impurity

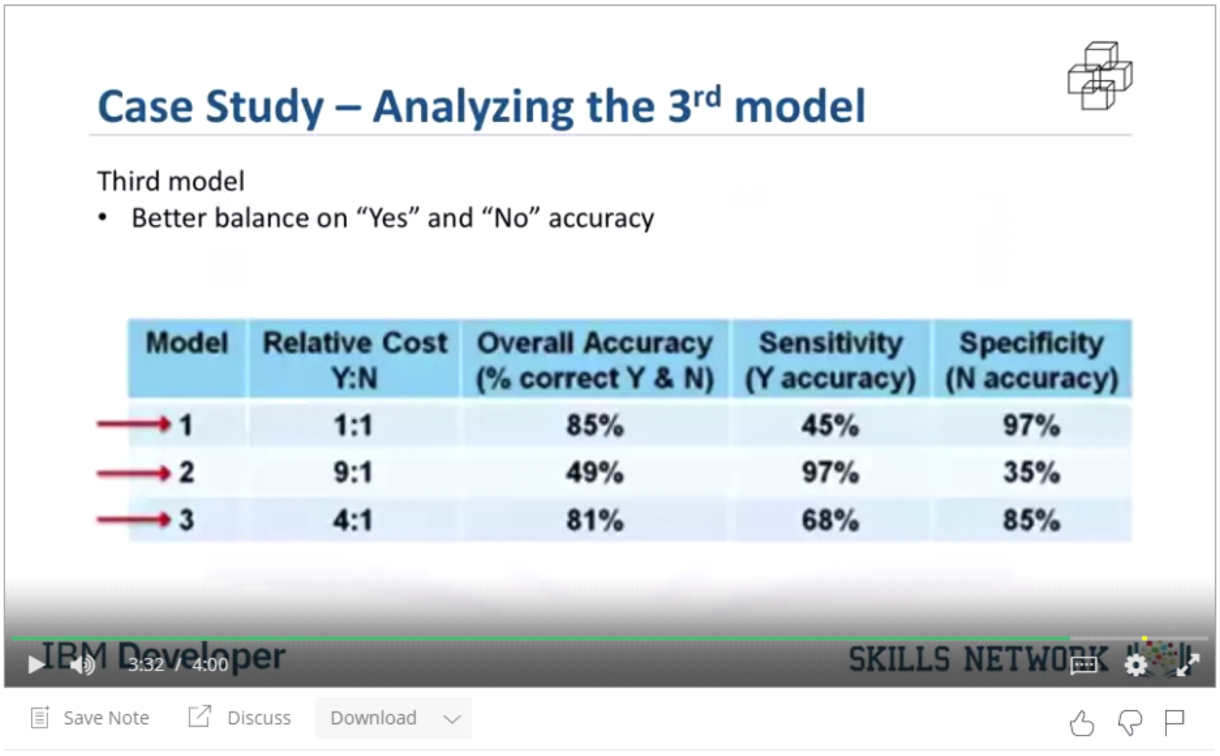
- A tree stops growing at a node when:
  - Pure or nearly pure.
  - No remaining variables on which to further subset the data.
  - The tree has grown to a preselected size limit.

Here are some characteristics of decision trees:



Pros	Cons
Easy to interpret	Easy to overfit or underfit the model
Can handle numeric or categorical features	Cannot model interactions between features
Can handle missing data	Large trees can be difficult to interpret
Uses only the most important features	
Can be used on very large or small data	

Modeling - Case Study



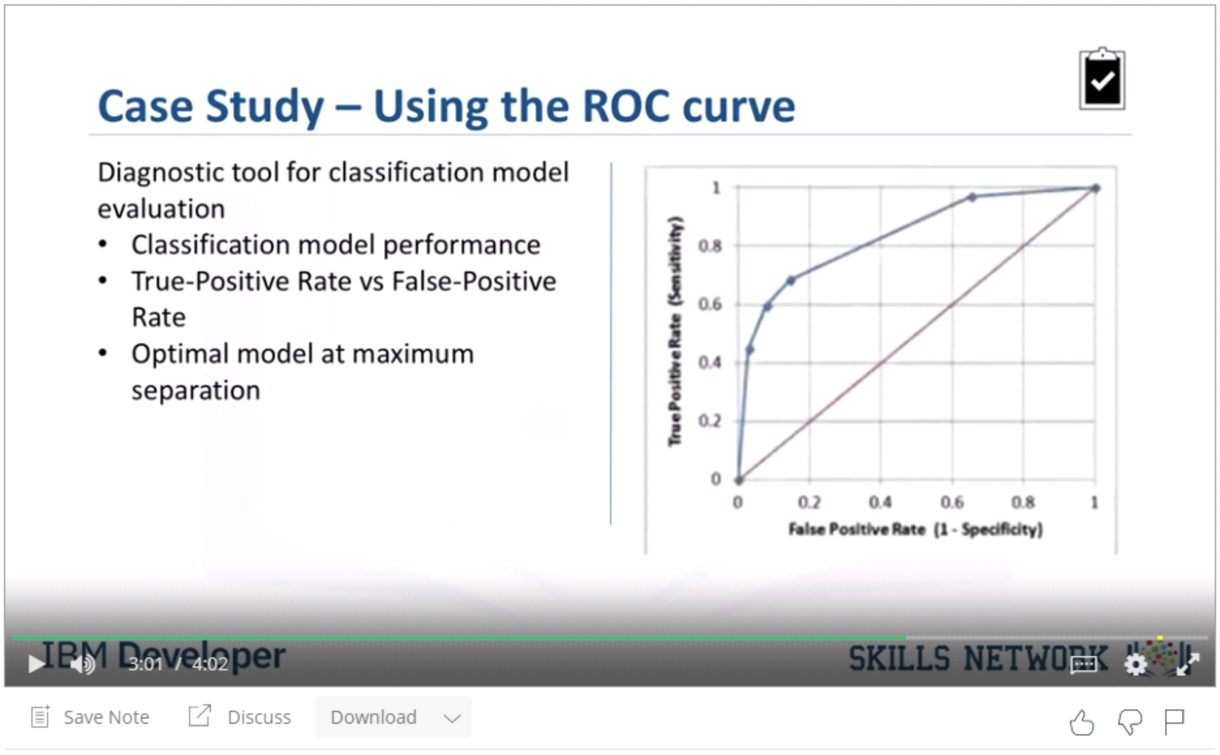
Case Study – Analyzing the 3rd model

Third model

- Better balance on "Yes" and "No" accuracy

Model	Relative Cost Y:N	Overall Accuracy (% correct Y & N)	Sensitivity (Y accuracy)	Specificity (N accuracy)
1	1:1	85%	45%	97%
2	9:1	49%	97%	35%
3	4:1	81%	68%	85%

Evaluation



Case Study – Using the ROC curve

Diagnostic tool for classification model evaluation

- Classification model performance
- True-Positive Rate vs False-Positive Rate
- Optimal model at maximum separation

Optimal = further distance

Python Intro

Tuesday, March 31, 2020 9:25 PM

help(var) gives me info on all the methods I can use on this var type

- Types
  - type(var) : gives me the type of a variable
  - Typecasting : changing the type of a variable
  - Boolean needs to have a capital first letter , int(True) = 1 and bool(1) = True
  - 25//6 = 4 (It does integer division and rounds)

  - Strings
    - Strings: \ is an escape character or use the letter r before the string
    - Strings have sequence methods and string methods
      - String methods B=A.method() , e.g fimd , replace
    - Strings use **[beg, end, step]**
    - **len**(var) is used for size
    - Immutable
    - Split strings to list e.g "A,B,C,D".split(",") = ["A","B","C","D"]

  - Tuples
    - Ordered sequences
    - Tuples are created with **( )** but accessed with **[beg, end, step]**
    - **len**(var) is used for size
    - Immutable
    - Methods : sorted

  - Lists
    - Ordered sequences : indexes - elements
      - Indexes need to be integers
    - Lists are created with **[ ]** and accessed with **[beg, end, step]**
    - **len**(var) is used for size
    - Mutable
    - Method : extend , del, append on the same list L.append
      - Append only adds one element to a list
        - A=[1] , A.append([2,3,4,5]) gets [1,[2,4,5,6]] wih only2 elemetns
    - Aliasing : multiple var are referring to the same list (e.g. B=A)
    - Cloning : referencing a copy of a list (e.g. B=A[:])

  - Dictionaries
    - Unordered sequences : keys - values
      - Keys are immutable and unique
    - Dictionaries are created with **{ }** and accessed with **[beg, end, step]**
      - {key1 : value1 , key2 : value2} ; keys are usually strings so they are in ""
    - Key3 **in** my_dictionary : checks if the key3 exists as a key in my_dictionary (result True/False)
    - del(my_dictionary['Key3 ']) deletes that entries

  - Sets
    - Unordered sequences
    - Sets only have unique elements
    - Sets are creates with **{ }**
      - My_set = { 0, 5, 10, 7, 5, 7} --> {0, 10, 5, 7}
    - Typecast a list to a set set(my_list)
    - Methods: my_set.add("a") -->{0, 10, 5, 7, "a"} / my_set.remove(10) -->{0, 5, 7, "a"} / 5 in my_set --> True
    - Logic Operators : Can use & to find intersection of 2 sets / Union : seta.union(setb) /seta.issubset(setb)

- Loops
  - Range(N)--> list [0, 1, 2, .. N] ; range(2,5)--> [2, 3, 4]
  - for i,x in enumerate --> prints index and value

- Function
  - Len on str or dict or list
  - A variable created inside a function with the keyword global can be used outside without returning it
    def printer(artist):
      global internal_var
  - dir() lidt all functions , dir(nameofoblect): list of methods

- Open
  - You can get properties of the file object such as name/mode
  - With open(file,"r") as filename: --> better than just open because it closes the file
  - Readline(n) read n characters , if there is no n it reads a line of the file

- Pandas
  - **loc** is primarily label based df.loc[0, "ColName"]
  - **iloc** is integer-based. You use column numbers and row numbers to get rows or columns at particular positions in the data frame. df.iloc[0, 2]
  - By default, **ix** looks for a label. If ix doesn't find a label, it will use an integer. This means you can select data by using either column numbers and row numbers or column headers and row names using ix. In Pandas version 0.20.0 and later, ix is deprecated.

- Numpy
  - 1D list-like accessed with [] = array
  - Dtype gives the type of the elements
  - Asstributes : size = # of elem, ndim=# of dim, shape= size of array in each dimensions
  - Basic operation are faster and contain less memory in np than python, like Vector addition or linear combination
  - 2d : nested lists
  - Hadamant product --> multiplication of every element || dot product = [m,p]x[p,n]=[m,n]
-

SQL

Sunday, April 5, 2020 6:26 PM

- ## Statements
  - DDL (Data Definition Languages) : Define, change & drop data

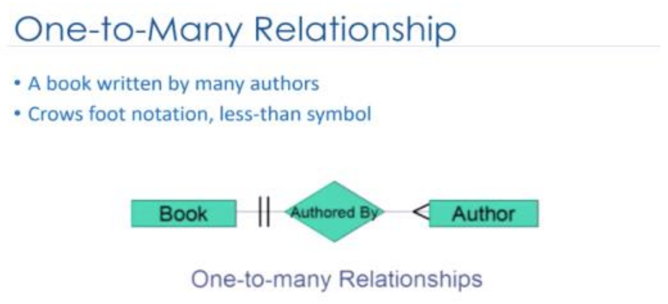
    -
      ```
      create table TABLENAME (
          COLUMN1 datatype,
          COLUMN2 datatype,
          COLUMN3 datatype,
              ...
          ) ;
      ```
  - DML (Data Manipulation Languages) :Read & modify data
    - `select COLUMN1, COLUMN2, ... from TABLE1 ;`
      - `select * from TABLE1 where condition ;`
      - `select COUNT(*) from TABLE1 ;  ==> number of rows matching`
      - `select DISTINCT colname from TABLE1 ;  ==>remove duplicates`
      - `select * from TABLE1 LIMIT x;  ==> limit to only x rows`

    -
      ```
      insert into TABLENAME
           (colname 1, colname 2. …)
          values
           ( val1_col1, val1_col2, ..)
           ( val1_col1, val1_col2, ..)
      ```

    -
      ```
      update TABLENAME
          set Colname = Value
           Where Condition
      ```

    -
      ```
      Delete TABLENAME
           Where Condition
      ```
  - Primary ID: uniquely identifies each tuple or row in a table
    - A Primary Key is a unique identifier in a table, and using Primary Keys can help speed up your queries significantly.
  - It is quite common to issue a DROP before doing a CREATE in test and development scenarios.
  - Where clause is like an if loop : The WHERE clause can be added to your query to filter results or get specific rows of data

- ## Relational models
  - Informational : abstract model for designers and operators --> relationships
    - Hierarchical model
  - Data model : Concrete/detailed model for implementors
    - Relational model : Data is organized in tables with entity relationships
      - data independence
      - stored in tables
    - Entity-Relationship(ER) model
      - Entities and attributes
  - Relationships:
    - Entities sets : rectangle
    - Attributes : ovals
    - Relationship sets : diamond
    - Crows foot notations



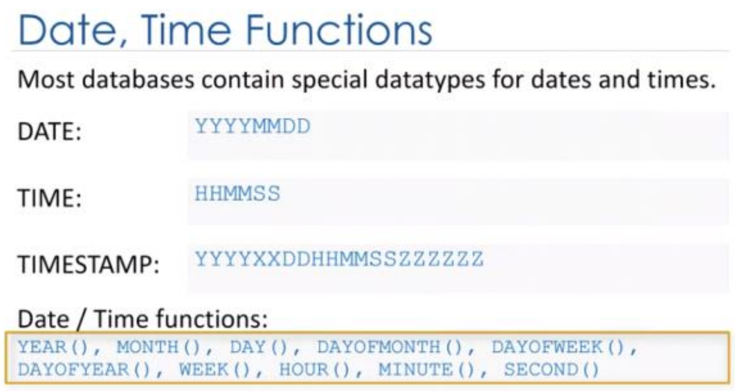
One-to-Many Relationship

- A book written by many authors
- Crows foot notation, less-than symbol

One-to-many Relationships

- Entity --> Table | Attributes --> Columns
- Relation=Table | Sets: unordered collection if distinct elements
- Relation Schema : name of relation and attributes
- Relation Instance: table made of the data
  - Degree = # of columns
  - Cardinality= # of tuples or rows

- ## Retrieving Data
  - Patterns & ranges:
    - Where com like 'R% ' : all rows whose col element starts with R l
    - Where col >= 10 AND <=12
    - Where col between 10 and 12 : same as above , inclusive
    - Where col='a' or col='b'
    - Where col in ('a', 'b', 'c')
  - Sort
    - Add order cause in a select command
    - Order by col desc
    - Order by 2 : sorts by the 2nd col
  - Remove duplicates/ group/count:
    - Select distinct(col) from table
    - Add group close to group
      - Group by col
    - Group & Count
      - Select count(col) from table group by col
      - Select count(col) as count from table group by col
    - Add condition to a group by clause:
      - Select count(col) as count from table group by col having count(col)>4
      - Having works only with group, where works only with whole set
- Build in functions: Databases dependent
  - Aggregate : sum , min, max - cannot use in the where clause
  - Scalar: length , ucase, lcase
  - Date:



Date, Time Functions

Most databases contain special datatypes for dates and times.

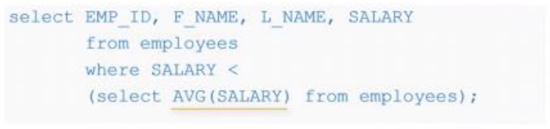
DATE:	YYYYMMDD
TIME:	HHMMSS
TIMESTAMP:	YYYYXXDDHHMMSSZZZZZZ

Date / Time functions:

```
YEAR(), MONTH(), DAY(), DAYOFMONTH(), DAYOFWEEK(),
DAYOFYEAR(), WEEK(), HOUR(), MINUTE(), SECOND()
```

        - Also allows for date arithmetic

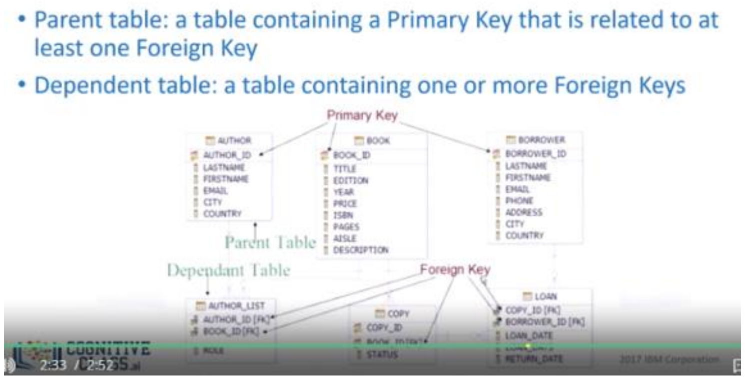
- Sub-Queries and Nested Selects
    - Needed subqueries for an aggregate function



```
select EMP_ID, F_NAME, L_NAME, SALARY
       from employees
       where SALARY <
       (select AVG(SALARY) from employees);
```

    - Column expressions: substitute col name w a sub query
    - Derived Tables or Table expressions : substitute the table name w a sub query
- Multiple Tables
    - Implicit join : select from 2 tables --> full or Cartesian join
        - Every row in the first table is joined with every row in the 2nd table

- ## Relational Model
    - Foreign key = col in a table that is a primary key in another table
    - Parent Table = a table containing a primary key related to at least a foreign key
    - Dependent Table = a table containing one or more foreign keya



- Parent table: a table containing a Primary Key that is related to at least one Foreign Key
- Dependent table: a table containing one or more Foreign Keys

    - <u>Constrains:</u>
        - Entity Integrity Constraint : PK is unique and not NULL
        - Referential Integrity Constraint : e.g. one book need to be written by at least 1 author
        - Semantic Integrity Constraint : correctness of the data
        - Semantic Constraints: defined by the user/business case not schema:
            - Domain constraint: similar type
            - NULL constraint: specific attributes cannot be NULL
            - Check constraint: limit values , eg not a year beyond current year

- ## Python
  - dbAPI = python database API
  - Connection object : how to connect
  - Cursor operates like a file handle in that it enables you to scan through a result set of a query.
  - % sql magic
    - Load with %load_ext sql
    - You can use python variables in your SQL statements by adding a ":" prefix to your python variable names.

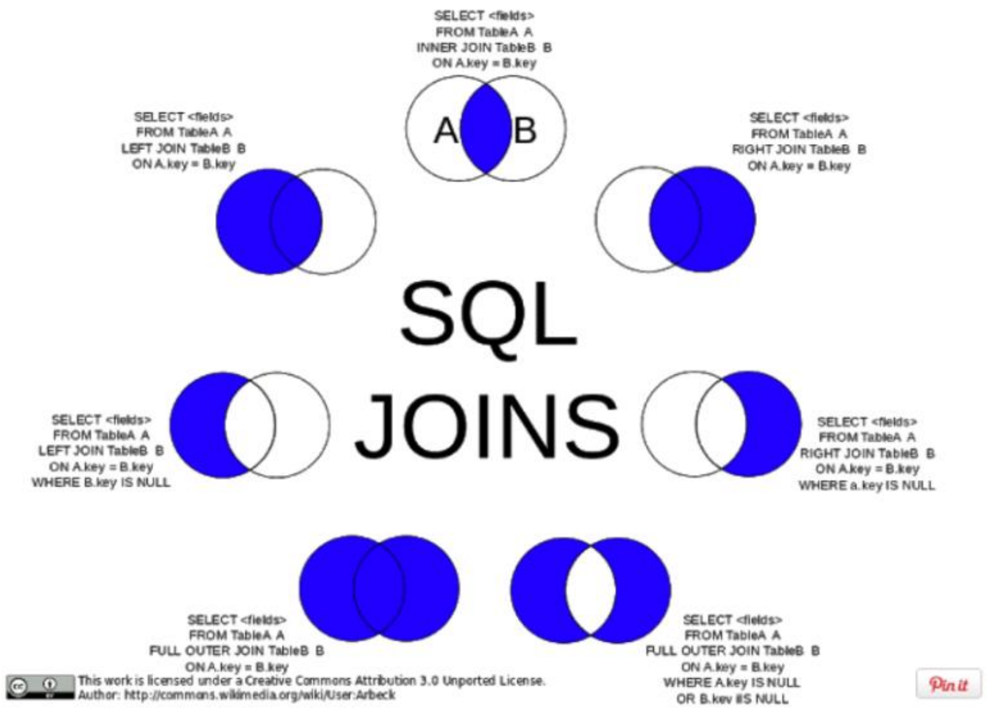
      country = "Canada"

      %sql select * from INTERNATIONAL_STUDENT_TEST_SCORES where country = :country

- ## Join
  - dbAPI = python database API
  - Inner join : intersection --> only row s that match

    - **select** A.col1, B.col2, C.col3
      from tableA A
        inner join tableB B **on** A.colm = B.colm
        inner join tableC C **on** B.colx = C.colx

  - Left Outer join or left join: intersection --> all rows from left & any matching from right

    - On the rows that don't match in the right table the values are null
  - Full Outer join or left join: intersection --> all rows from all tables
    - On the rows that don't match in the right table the values are null
  - 

```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE a.key IS NULL
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

credentials

Sunday, April 12, 2020 6:00 PM

```
"db": "BLUDB",
  "dsn": "DATABASE=BLUDB;HOSTNAME=dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net;PORT=50000;PROTOCOL=TCPIP;UID=mfp19233;PWD=9x2cknz0m@gzp3ds;",
  "host": "dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net",
  "hostname": "dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net",
  "https_url": "https://dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net",
  "jdbcurl": "jdbc:db2://dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net:50000/BLUDB",
  "parameters": {},
  "password": "9x2cknz0m@gzp3ds",
  "port": 50000,
  "ssldsn": "DATABASE=BLUDB;HOSTNAME=dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net;PORT=50001;PROTOCOL=TCPIP;UID=mfp19233;PWD=9x2cknz0m@gzp3ds;Security=SSL;",
  "ssljdbcurl": "jdbc:db2://dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net:50001/BLUDB:sslConnection=true;",
  "uri": "db2://mfp19233:9x2cknz0m%40gzp3ds@dashdb-txn-sbox-yp-dal09-08.services.dal.bluemix.net:50000/BLUDB",
  "username": "mfp19233"
```

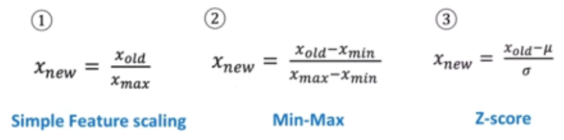
Python Data Analysis

Thursday, April 16, 2020 10:02 PM

Pandas

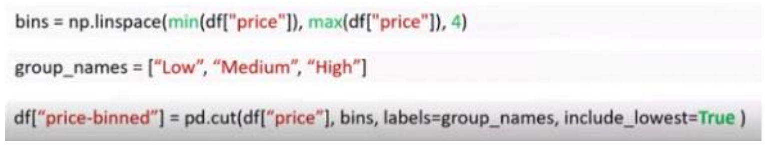
- df.dtypes returns the datatypes of each column
- df.describe() returns df statistics , df.describe("all") will incule info about objects too
- df.info : shows top and bottom 30 rows
- df['col_name'].value_counts() --> which values are present in col_name
    - df['col_name'].value_counts().idxmax() --> most common value

Data Wrangling

- Missing Values : Drop values / Replace w average value of the column or similar type, by frequency, based on other columns
    - df.dropna(), axis=0 or axis=1 drops rows or columns
    - df.replace(mis value, new value)
- Normalization:
    - 

      ① $x_{new} = \frac{x_{old}}{x_{max}}$ Simple Feature scaling

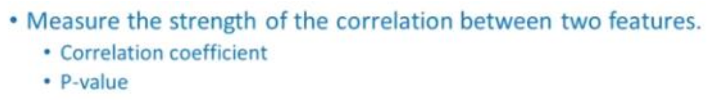
      ② $x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}}$ Min-Max

      ③ $x_{new} = \frac{x_{old} - \mu}{\sigma}$ Z-score
    - Range of the above normalized values is 0-1, 0-1, usually -3 to 3
- Binning
    - 

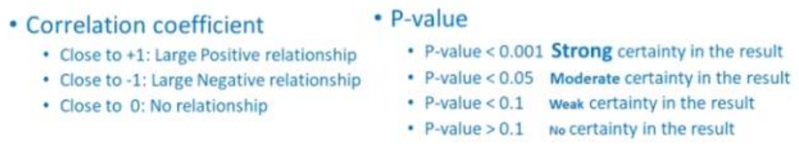
      ```
      bins = np.linspace(min(df["price"]), max(df["price"]), 4)
      group_names = ["Low", "Medium", "High"]
      df["price-binned"] = pd.cut(df["price"], bins, labels=group_names, include_lowest=True )
      ```
- Categorical to numerical
    - one hot encoding
    - get_dummies()

Exploratory Data Analysis (EDA)

- Descriptive Statistics:
    - df.describe() , df.describe(include=['object'])
    - value_counts()
    - boxplot for statistics & scatter plot for relationships
- Group by:
    - uses categorial variables
    - pivot table if grouping by more than 1 categories
    - Hetamap: plot target variable over multiple variables
    - get_froup returns the values for each group
- Correlation:
    - regplot line --> fitted regression line for data
    - df.corr() --> calculate the correlation between variables of type "int64" or "float64
    - Pearson correlation

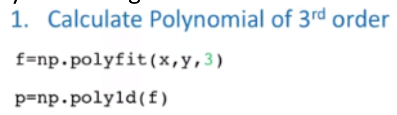
      - Measure the strength of the correlation between two features.
          - Correlation coefficient
          - P-value

        - 

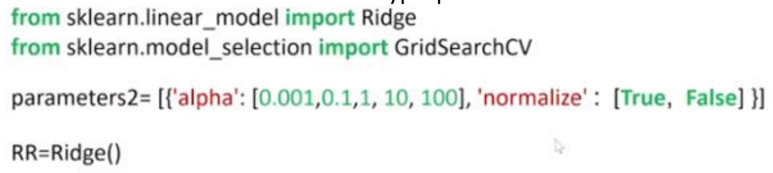
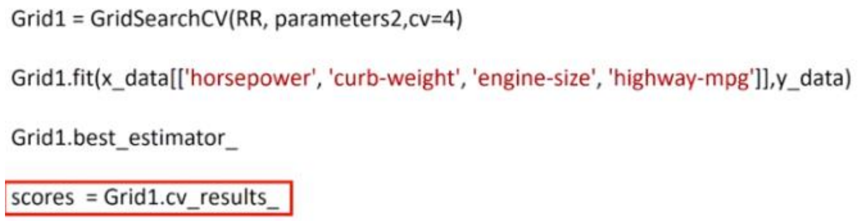
          - Correlation coefficient
              - Close to +1: Large Positive relationship
              - Close to -1: Large Negative relationship
              - Close to 0: No relationship
          - P-value
              - $P\text{-value} < 0.001$ **Strong** certainty in the result
              - $P\text{-value} < 0.05$ **Moderate** certainty in the result
              - $P\text{-value} < 0.1$ **Weak** certainty in the result
              - $P\text{-value} > 0.1$ **No** certainty in the result

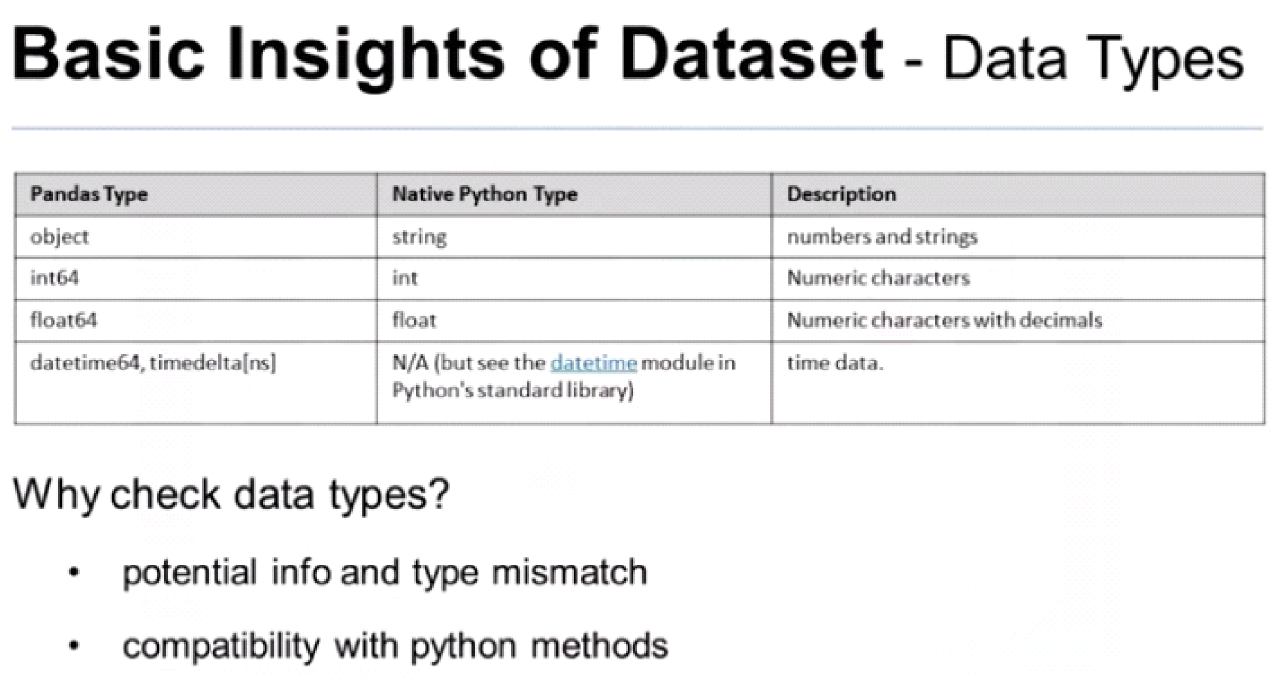
        ▪ correlation heatmap : heatmap with pairwise corelation across all variables (including target variable as well)
    ○ ANOVA : analysis of variance --> corelation between categorical variables
        ▪ F-Test : correlation btw the averages of two groups wrt their the variance
        ▪ we want large F test values and small p value
            ◻ F-test score: ANOVA assumes the means of all groups are the same, calculates how much the actual means deviate from the assumption, and reports it as the F-test score. A larger score means there is a larger difference between the means.
            ◻ P-value: P-value tells how statistically significant is our calculated score value.

Regression

- Model evaluation with visualization:
    - seaborn resplot - residual plot : error vs x--> if there is no pattern (not spread randomly or variance is dependent to x) then linear regression is good
    - distribution plots : binning of x and y ==> distribution plot of the actual vs predicted values
- 1D Polynomial Regression:
    - 
      1. Calculate Polynomial of 3rd order
      ```
      f=np.polyfit(x,y,3)
      p=np.poly1d(f)
      ```
    - Print (p) shows the symbolic form of the equation
- Multidimensional polynomial
    - use the preprocessing module to transform the features to all the possible terms (e.g. X1, X2, X1^2, X2^2, X1*X2
    - use the preprocessing library to normalize the features
- Use a **pipeline** to sequence perform normalization --> polynomial transformation --> linear regression
- In sample evaluation of a model:
    - **MSE** : mean_squarred_error
        - SLR has higher MSE than MLR, that has higher than polynomila because there are more elements (?)
    - **R^2** : % of variation explained in the linear model - lm.score(X,Y)

Evaluation - out of sample

- train_test_split
- generalization error : error of prediction vs test data
- Cross validation:
    - Data is split to groups (folds)
    - some folds are used for training some for testing
    - cross_val_score --> R2 scores of cross validation , cross_val_predict() --> yhat of cross validaton
- Ridge Regression : bias fit in training data to fit the test data the best
- Grid search - hyperparameters iteration with cross validation
    - split in training, test and validation
    - use training and test with cross validation
    - then use validation test to choose hyperparameter value
      
      ```
      from sklearn.linear_model import Ridge
      from sklearn.model_selection import GridSearchCV

      parameters2= [{'alpha': [0.001,0.1,1, 10, 100], 'normalize' :  [True,  False] }]

      RR=Ridge()
      ```
    - 
      ```
      Grid1 = GridSearchCV(RR, parameters2,cv=4)

      Grid1.fit(x_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']],y_data)

      Grid1.best_estimator_

      scores  = Grid1.cv_results_
      ```

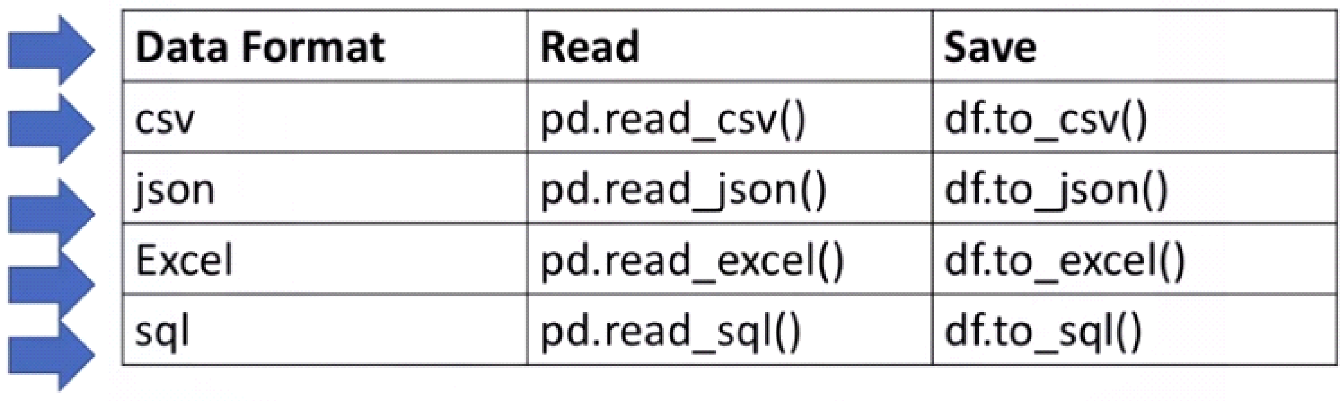


Basic Insights of Dataset - Data Types

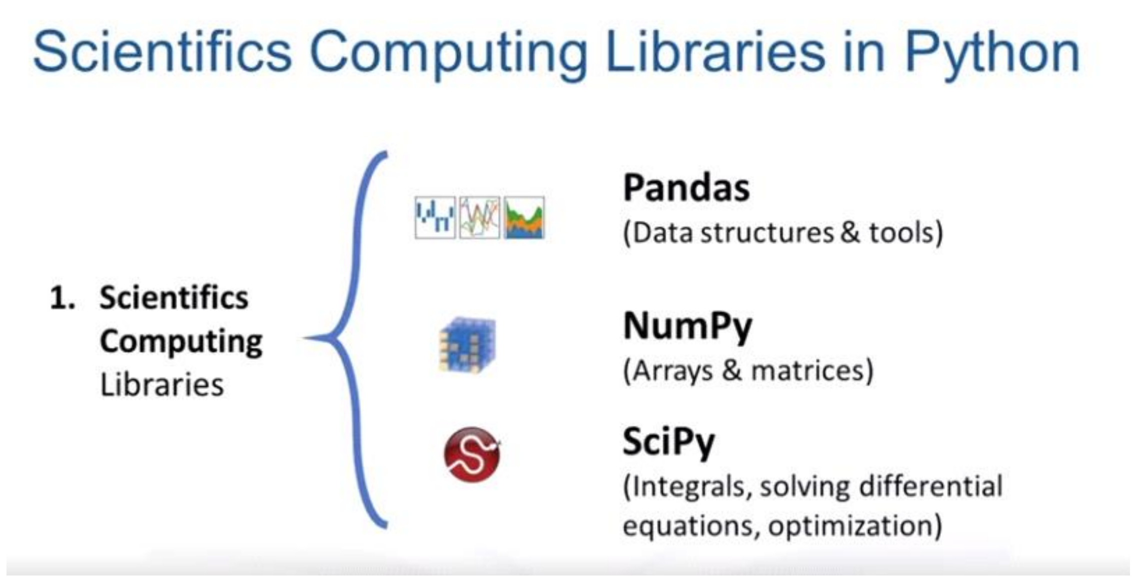
Pandas Type	Native Python Type	Description
object	string	numbers and strings
int64	int	Numeric characters
float64	float	Numeric characters with decimals
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	time data.

Why check data types?

- potential info and type mismatch
- compatibility with python methods



Data Format	Read	Save
csv	pd.read_csv()	df.to_csv()
json	pd.read_json()	df.to_json()
Excel	pd.read_excel()	df.to_excel()
sql	pd.read_sql()	df.to_sql()

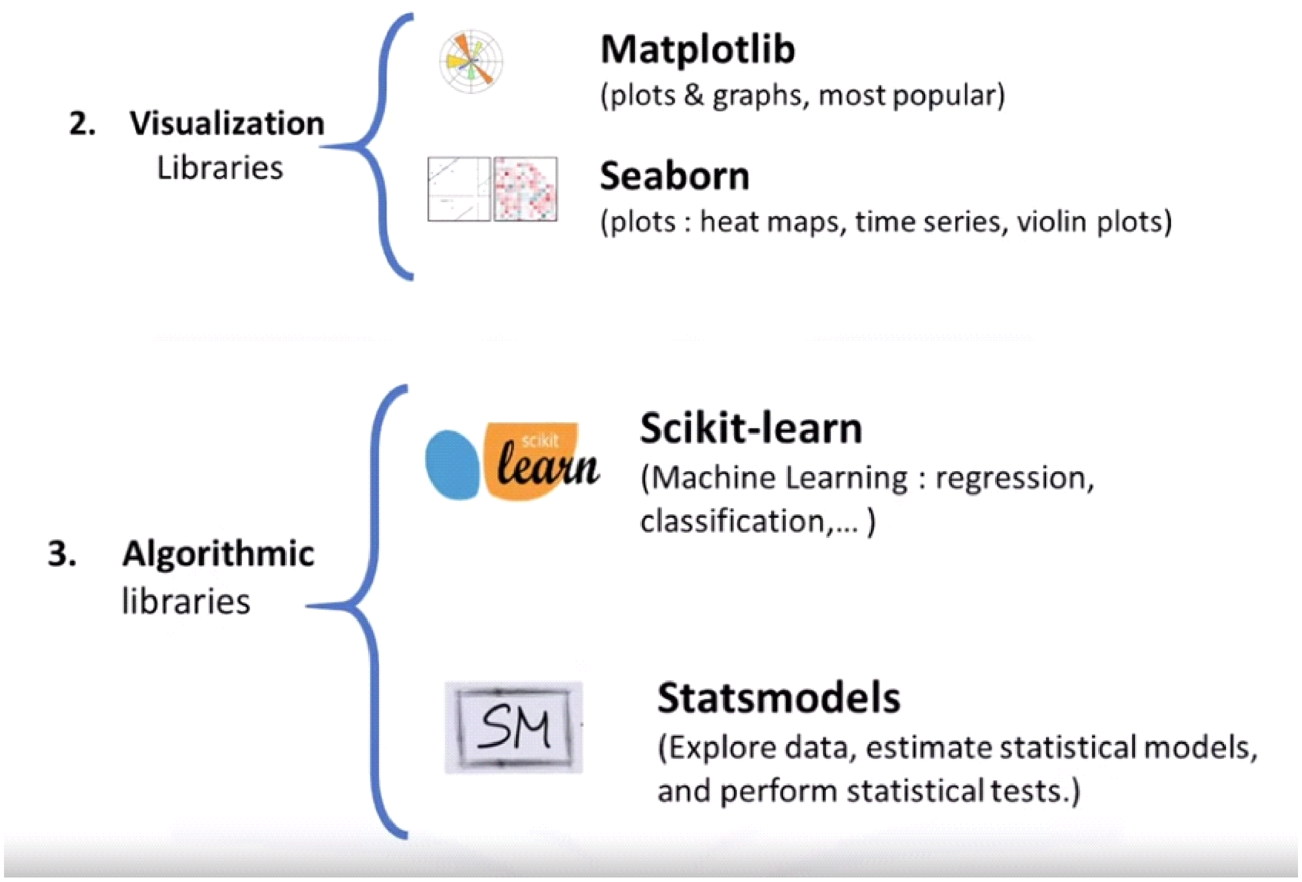


Scientifics Computing Libraries in Python

1. Scientifics Computing Libraries

- **Pandas** (Data structures & tools)
- **NumPy** (Arrays & matrices)
- **SciPy** (Integrals, solving differential equations, optimization)

Visualization Libraries in Python



2. Visualization Libraries

Matplotlib
(plots & graphs, most popular)

Seaborn
(plots : heat maps, time series, violin plots)

3. Algorithmic libraries

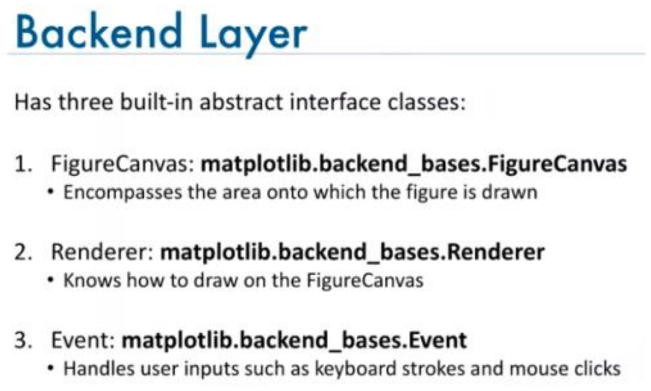
Scikit-learn
(Machine Learning : regression, classification,...)

Statsmodels
(Explore data, estimate statistical models, and perform statistical tests.)

Data Visualization

Thursday, April 23, 2020 10:54 PM

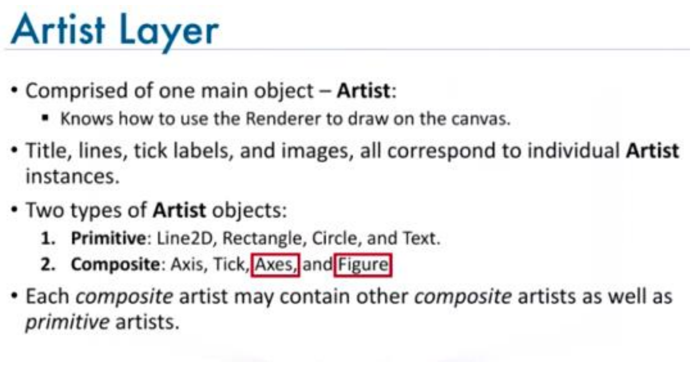
MatplotLib



Backend Layer

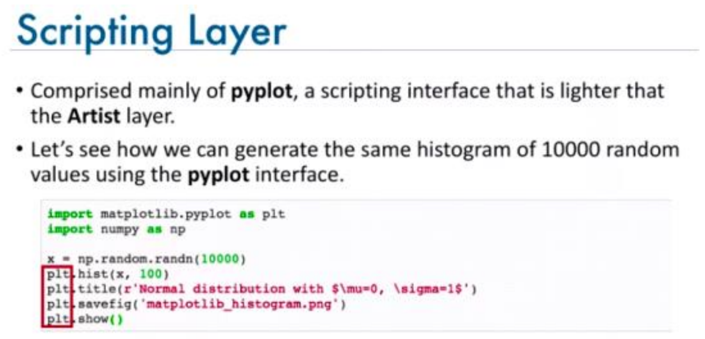
Has three built-in abstract interface classes:

1. FigureCanvas: **matplotlib.backend_bases.FigureCanvas**
   - Encompasses the area onto which the figure is drawn
2. Renderer: **matplotlib.backend_bases.Renderer**
   - Knows how to draw on the FigureCanvas
3. Event: **matplotlib.backend_bases.Event**
   - Handles user inputs such as keyboard strokes and mouse clicks



Artist Layer

- Comprised of one main object – **Artist**:
  - Knows how to use the Renderer to draw on the canvas.
- Title, lines, tick labels, and images, all correspond to individual **Artist** instances.
- Two types of **Artist** objects:
  1. **Primitive**: Line2D, Rectangle, Circle, and Text.
  2. **Composite**: Axis, Tick, Axes, and Figure
- Each *composite* artist may contain other *composite* artists as well as *primitive* artists.



Scripting Layer

- Comprised mainly of **pyplot**, a scripting interface that is lighter that the **Artist** layer.
- Let's see how we can generate the same histogram of 10000 random values using the **pyplot** interface.

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

x = np.random.randn(10000)
plt.hist(x, 100)
plt.title(r'Normal distribution with $\mu=0, \sigma=1$')
plt.savefig('matplotlib_histogram.png')
plt.show()
```