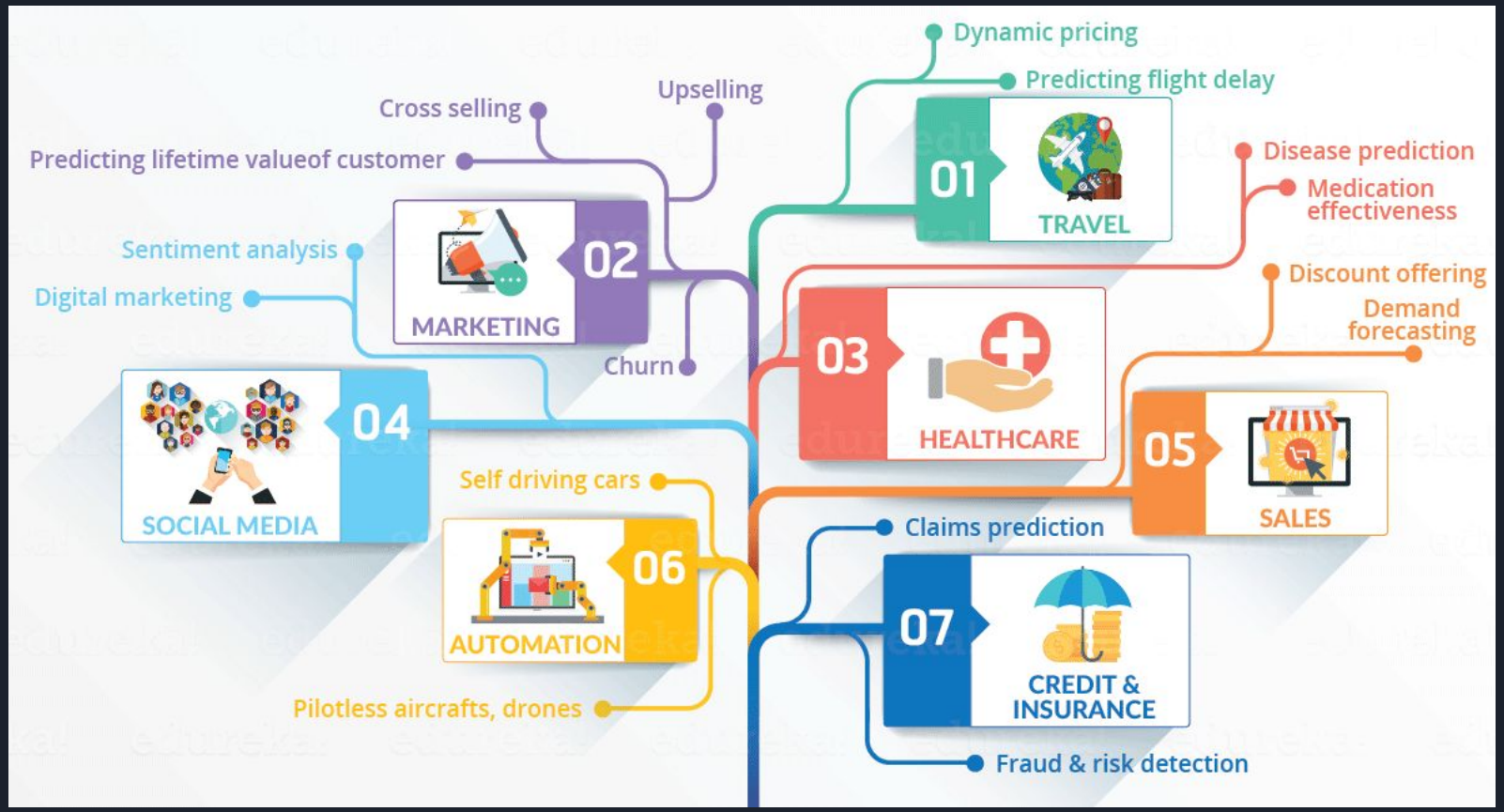
Data Analysis with Python

Agenda

- Introduction to Data Science
- Introduction to Python
- Python for Data Science
- Data Science with Python

Why Data Science?

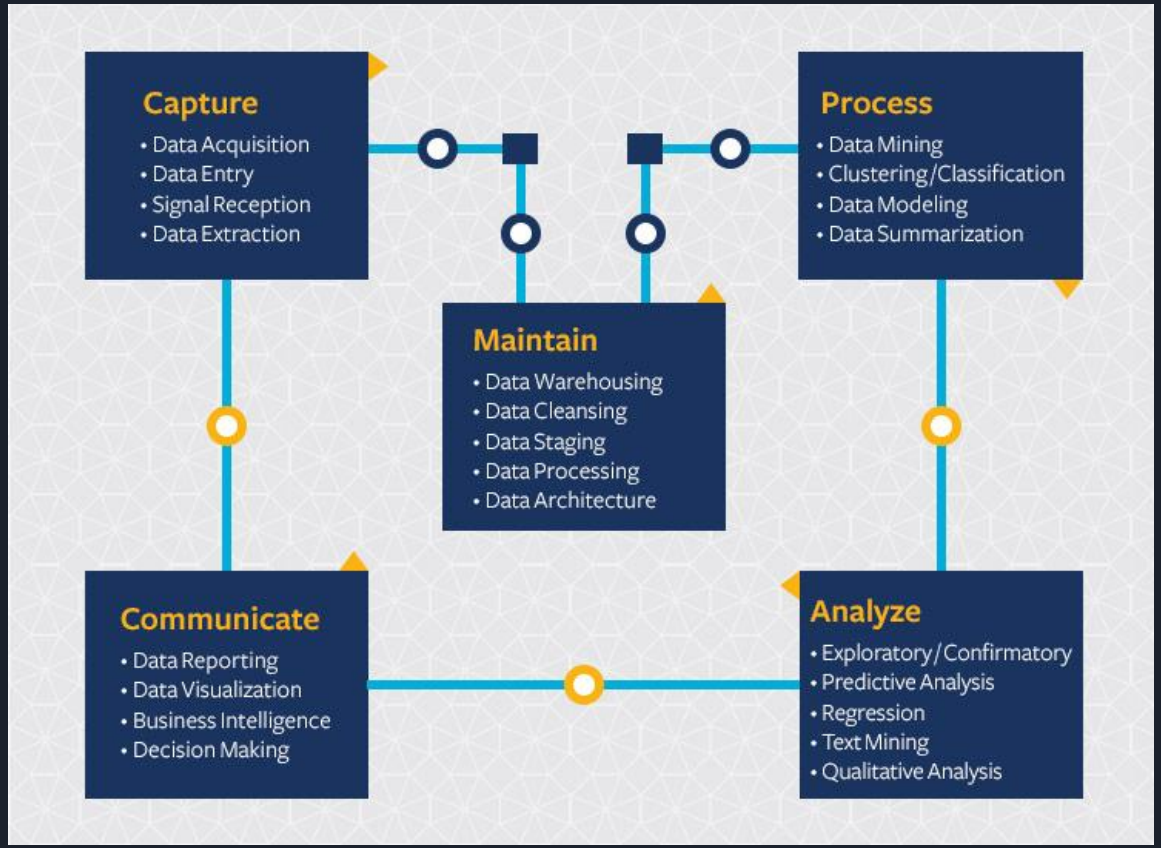


Data Science Use Cases

What is Data Science?

Data Science is a blend of various tools, algorithms and machine learning principles with a goal to discover hidden patterns from raw data.

Data Science Life Cycle



Data Science Life Cycle

Future for Data Science

28%

Demand increase for Data Scientists in 2020.

$120,931

Average Base Salary of a Data Scientist.

Languages Used in Data Science

- Python
- R
- Matlab
- Octave

Python

Why Python?

- Beginner friendly
- Open source
- A great library ecosystem
- Flexible
- Platform independence
- Readability
- Good visualization options
- Community support

Python Topics

- Variables
- Data types
  - Numbers
  - Strings
  - Print formatting
  - Lists
  - Dictionaries
  - Booleans
  - Tuples and Sets
- Conditional expressions
- Conditional Statements
- Loops
- List comprehension
- Functions
- Lambda expressions
- Map and filter
- Classes and objects

Variables

Harsh MacBook Air

```
>>> a = 10

>>> b = 3.5

>>> decision = true

>>> name = 'some name'

>>> type(a)
int
>>> type(name)
str
```

Expressions

Harsh MacBook Air

```
>>> 1 + 1
2
>>> 2 * 3
6
>>> 2 ** 4
16
>>> 2 + 3 * 5 + 5
22
>>> 5 % 2
1
```

Print Formatting

Harsh MacBook Air

```
>>> name = 'sam'

>>> age = 12


>>>print('My name is {} and my age is {}'.format(name, age))

My name is sam and my age is 12


>>> print('My name is {first} and my age is {second}'
        .format(first = name, second =  age))

My name is sam and my age is 12


>>> print('My name is {1} and my age is {0}'.format(name, age))

My name is 12 and my age is sam
```

Strings

Harsh MacBook Air

```
>>> 'this is a string'
this is a string

>>> "this is also a string"
this is also a string

>>> "I can't walk anymore"
I can't walk anymore
```

Strings

Harsh MacBook Air

```
>>> s = ‘abcdefghijk’
>>> s[0]
a
>>> s[0:]
abcdefghijk
>>> s[:3]
abc
>>> s[3:6]
def
>>> s[0:5:2]
ace
```

Strings

Harsh MacBook Air

```
>>> s1 = 'john'
>>> s2 = 'doe'

>>>s1 + s2
johndoe
>>> s1 * 3
johnjohnjohn
```

String methods

Harsh MacBook Air

```
>>> s = 'abcdefghijk'
>>> s.upper()
ABCDEFGHIJK
>>> s.lower()
abcdefghijk
>>> s.replace('a', 'z')
zbcdefghijk
>>> s.find('g)
6
```

Lists

Harsh MacBook Air

```
>>> my_list = ['a', 'b', 'c']
>>> my_list.append('d')
>>> my_list
['a', 'b', 'c', 'd']
>>> my_list[0]
a
>>> my_list[1:3]
['b', 'c']
>>> my_list[0] = 'NEW'
['NEW', 'b', 'c', 'd']
```

Dictionaries

Harsh MacBook Air

```
>>> d1 = {'key1' : 'value1', 'key2' : 123}
>>> d1['key1']
value1
>>> d1['key1'] = 'new'
{'key1' : 'new', 'key2' : 123}
>>> d2 = {'k1' : [1, 2, 3]}
>>> d2['k1']
[1, 2, 3]
>>> d['k1'][1]
```

Dictionaries

Harsh MacBook Air

```
>>> d1 = {'key1' : 'value1', 'key2' : 123}
>>> d1.keys()
dict_keys(['key2', 'key1'])
>>> d1.items()
dict_items([('key2', 123), ('key1', 'value1')])
>>> d1.values()
dict_values(['value1', 123])
```

Tuples

Harsh MacBook Air

```
>>> t = (1, 2, 3, 4, 5)
>>> t[0]
1
>>> t[1:4]
(2, 3, 4)
>>> len(t)
5
```

Sets

Harsh MacBook Air

```
>>> s = {1, 2, 3}
>>> set([1, 1, 1, 2, 2, 3, 3, 3])
{1, 2, 3}
>>> s.add(4)
{1, 2, 3, 4}
>>> s.remove(1)
>>> {2, 3, 4}
```

Relational Operators

Harsh MacBook Air

```
>>> 1 < 2
True
>>> 2 >= 3
False
>>> 3 == 4
False
>>> 3 != 4
True
>>> 'hi' != 'bye'
True
```

Relational Operators

Harsh MacBook Air

```
>>> 1 < 3 and 3 < 5
True
>>> (1 < 2) and (3 > 5)
False
>>> (2 < 5) or (3 > 4) or (1 == 1)
True
```

Conditional Statements

If Statement

Harsh MacBook Air

```
>>> if 1 < 2:
        print('1 is less than 2')

1 is less than 2

>>>if True:
        print('true')

true
```

If-elif-else Statement

Harsh MacBook Air

```
>>> if(1 != 3):
        print('not equal')
    else:
        print('equal')

not equal

>>>  if(1 == 3):
        print('first')
    elif(3 == 3):
        print('middle')
    else:
        print(last)

middle
```

Loops

For loop

Harsh MacBook Air

```
>>> for i in range(0, 6):
        print('hello')

hello
hello
hello
hello
hello

>>>  seq = [1, 2, 3, 4, 5]

>>> for item in seq:
        print(item)

1
2
3
4
5
```

While loop

Harsh MacBook Air

```
>>> i = 1

>>> while i < 5:
        print(i)
        i  = i + 1

1
2
3
4
5
```

List Comprehension

Harsh MacBook Air

```
>>> out = []

>>> for num in range(5):
        out.append(num ** 2)

[1, 4, 9, 16]

>>> [num ** 2 for num in range(5)]

 [1, 4, 9, 16]
```

Functions

Harsh MacBook Air

```
>>> def my_fun():
        print('Hello from function')

>>> my_fun()

Hello from function

>>> def my_fun(name = 'Default name'):
        print('Hello ' + name)

>>> my_fun('John')

Hello John

>>> my_fun()

Hello Default name
```

Functions

Harsh MacBook Air

```
>>> def square(num):
        """
        This is a docstring
        And can go multiple lines
        This function returns square of given number
        """
        return num ** 2

>>> output = square(2)

>>> output

4

>>> help(square)

This is a docstring
And can go multiple lines
This function returns square of given number
```

Map function

Harsh MacBook Air

```
>>> def times2(num):
        return num * 2

>>> seq = [1, 2, 3, 4, 5]

>>> list(map(times2, seq))

[2, 4, 6, 8, 10]
```

Lambda Expressions

Harsh MacBook Air

```
>>> def times2(num):
        return num * 2

>>> seq = [1, 2, 3, 4, 5]

>>> list(map(lambda num: num *2, seq))

[2, 4, 6, 8, 10]
```

Filter Function

Harsh MacBook Air

```
>>> seq = [1, 2, 3, 4, 5]
>>> list(filter(lambda num: num % 2 == 0, seq))
[2, 4]
```

Classes and Objects

Harsh MacBook Air

```
>>> class Circle:

        def __init__(self, radius = 3, color = 'red'):
                self.radius = radius
                self.color = color

        def add_radius(self, r):
                self.radius = self.radius + r
                return (self. radius)


>>> blueCircle = Circle(4, 'blue')

>>> blueCircle.radius

4

>>> blueCircle.add_radius(3)

7
```

Inheritance

Syntax:

```
class DrivedClassName(BaseClassName):
        # body of the class
```

Inheritance

```
class Polygon:
    def __init__(self, no_of_sides):
        self.n = no_of_sides
        self.sides = [0 for i in range(no_of_sides)]

    def inputSides(self):
        self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i in range(self.n)]


class Triangle(Polygon):
    def __init__(self):
        Polygon.__init__(self,3)

    def findArea(self):
        a, b, c = self.sides
        # calculate the semi-perimeter
        s = (a + b + c) / 2
        area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
        print('The area of the triangle is %0.2f' %area)

t = Triangle()
t.inputSides()
t.findArea()
```

Questions?

Recap

Libraries for Data Science

- Numpy
- Pandas
- Matplotlib
- Scikit-learn

Numpy

Numpy arrays

Harsh MacBook Air

```
>>> my_list = [1, 2, 3]

>>> import numpy as np

>>> arr = np.array(my_list)

array([1,2, 3])

>>> my_mat = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

>>> np.array(my_mat)

array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Built-in array methods

Harsh MacBook Air

```
>>> np.arange(0, 5)
array([0, 1, 2, 3, 4])
>>> np.arange(0, 11, 2)
array([0, 2, 4, 6, 8, 10])
>>> np.zeros(3)
array([0., 0., 0.])
>>> np.zeros(2, 3)
array([[0., 0., 0.],
       [0., 0., 0.]])
```

Built-in array methods

Harsh MacBook Air

```
>>> np.ones(4)

array([1., 1., 1., 1.])

>>> np.zeros(3, 3)

array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])

>>> np.linspace(0, 2, 5)

array([0. , 0.5, 1. , 1.5, 2.])

>>> np.eye(3)

array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Numpy random method

Harsh MacBook Air

```
>>> np.random.rand(2) # from uniform distribution over [0, 1]

array([ 0.11570539,  0.35279769])

>>>  np.random.rand(3,3)

array([[0.87151946, 0.9354061 , 0.52198032],
       [0.08266691, 0.80790276, 0.1859491 ],
       [0.58838556, 0.57997241, 0.11916838]])

>>> np.random.randint(1,100) # random value from low to high

44

>>> np.random.randint(1,100,10)

array([13, 64, 27, 63, 46, 68, 92, 10, 58, 24])
```

Array attributes and methods

Harsh MacBook Air

```
>>> arr = np.arange(25)

>>> arr.reshape(5,5)

array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])

>>> arr.shape

(25, )

>>> arr.dtype

dtype('int64')
```

Array attributes and methods

Harsh MacBook Air

```
>>> ranarr = np.random.randint(0,50,10)
array([10, 12, 41, 17, 49,  2, 46,  3, 19, 39])
>>> ranarr.max() # returns the value
49
>>> ranarr.argmax() # returns the index
4
>>> ranarr.min()
2
>>> ranarr.argmin()
5
```

Numpy indexing and selection

Harsh MacBook Air

```
>>> arr = np.arange(0,11)
>>> arr[8]
8
>>> arr[1:5]
array([1, 2, 3, 4])
>>> arr[0:5]
array([0, 1, 2, 3, 4])
>>> arr[6:]
array([ 6,  7,  8,  9, 10])
```

Broadcasting

Harsh MacBook Air

```
>>> arr[0:5]=100
array([100, 100, 100, 100, 100,  5,  6,  7,  8,  9, 10])
>>> arr = np.arange(0,11)
>>> slice_of_arr = arr[0:6]
>>> slice_of_arr[:]=99
>>> slice_of_arr
array([99, 99, 99, 99, 99, 99])
>>> arr
array([99, 99, 99, 99, 99, 99,  6,  7,  8,  9, 10])
>>> arr_copy = arr.copy()
```

2D arrays

Harsh MacBook Air

```
>>> arr_2d = np.array(([5,10,15],[20,25,30],[35,40,45]))

>>> arr_2d[1]

array([20, 25, 30])

>>> arr_2d[1][0]

20

>>> arr_2d[1, 0] # preferred

20

>>> arr_2d[:2,1:]

array([[10, 15],
       [25, 30]])
```

Slicing exercise

Harsh MacBook Air

```
>>> arr_2d = np.arange(50).reshape(5, 10)

array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49]])

>>> ??

array([[13, 14],
       [23, 24]])

>>> ??

array([[28, 29],
       [38, 39],
       [48, 49]])
```

Fancy indexing

Harsh MacBook Air

```
>>> arr2d = np.zeros((10,10))

>>> arr_length = arr2d.shape[1]

>>> for i in range(arr_length):
        arr2d[i] = i

>>> arr2d

array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.],
       [ 2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.],
       [ 3.,  3.,  3.,  3.,  3.,  3.,  3.,  3.,  3.,  3.],
       [ 4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.],
       [ 5.,  5.,  5.,  5.,  5.,  5.,  5.,  5.,  5.,  5.],
       [ 6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.],
       [ 7.,  7.,  7.,  7.,  7.,  7.,  7.,  7.,  7.,  7.],
       [ 8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.],
       [ 9.,  9.,  9.,  9.,  9.,  9.,  9.,  9.,  9.,  9.]])
```

Fancy indexing

Harsh MacBook Air

```
>>> arr2d[[2,4,6,8]]

array([[ 2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.],
       [ 4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.],
       [ 6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.],
       [ 8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.]])

>>> arr2d[[6,4,2,7]]

array([[ 6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.,  6.],
       [ 4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.,  4.],
       [ 2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.,  2.],
       [ 7.,  7.,  7.,  7.,  7.,  7.,  7.,  7.,  7.,  7.]])
```

Selection

Harsh MacBook Air

```
>>> arr = np.arange(1,11)

>>> arr > 4

array([False, False, False, False,  True,  True,  True,  True,  True,  True],
       dtype=bool)

>>> bool_arr = arr>4

>>> arr[bool_arr]

array([ 5,  6,  7,  8,  9, 10])

>>> arr[arr>2]

array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

Numpy operations

Harsh MacBook Air

```
>>> arr = np.arange(0,10)
>>> arr + arr
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
>>> arr * arr
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
>>> arr - arr
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

Numpy operations

Harsh MacBook Air

```
>>> arr/arr
RuntimeWarning: invalid value encountered in true_divide
array([nan, nan, nan, nan, nan, nan, nan, nan, nan, nan])

>>> 1/arr
RuntimeWarning: divide by zero encountered in true_divide
array([inf, inf, inf, inf, inf, inf, inf, inf, inf, inf])
>>> arr**3
array([  0,   1,   8,  27,  64, 125, 216, 343, 512, 729])
```

Universal array functions

Harsh MacBook Air

```
>>> np.sqrt(arr)

array([0.        , 1.        , 1.41421356, 1.73205081, 2.        ,
       2.23606798, 2.44948974, 2.64575131, 2.82842712, 3.        ])

>>> np.exp(arr) # for exponentiation

array([1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01,
       5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03,
       2.98095799e+03, 8.10308393e+03])

>>> np.max(arr)

9

>>> np.sin(arr)

array([ 0.        ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ,
       -0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849])

>>> np.log(arr)
```

https://github.com/harshshinde07/Data-Science-with-Python

Pandas

Pandas

- Series
- Dataframes
- Missing data
- Groupby
- Merge, join, concatenation
- Operations
- Data input and output

https://github.com/harshshinde07/Data-Science-with-Python

Recap

Matplotlib

https://github.com/harshshinde07/Data-Science-with-Python

Scikit-learn

Features of Scikit-learn

- Simple and efficient tools for data mining and machine learning
- Easily accessible and reusable
- Built on top of numpy, scipy, matplotlib
- Open source

Popular models provided

- Datasets
- Clustering
- Supervised models
- Dimensionality reduction
- Feature extraction
- Feature selection
- Parameter tuning

Machine Learning

Machine learning methods

- Supervised learning
  - Classification
  - Regression
- Semi-supervised learning
- Unsupervised learning
  - Clustering
- Reinforcement learning

Machine learning applications

- Credit card fraud detection
- Email filtering
- Housing price calculation
- Handwriting recognition
- Sentiment analysis
- Machine translation

Data Preprocessing

Data Preprocessing Steps

- Importing libraries
- Importing the Dataset
- Handling missing values
- Handling categorical data
- Splitting data into train and test sets
- Feature scaling

https://github.com/harshshinde07/Data-Science-with-Python

1. Importing libraries

1. Importing libraries

Harsh MacBook Air

```
>>> import numpy as np
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
```

2. Importing the dataset

2. Importing the dataset

Harsh MacBook Air

```
>>> dataset = pd.read_csv('Data.csv')
>>> X = dataset.iloc[:, :-1].values
>>> y = dataset.iloc[:, 3].values
```

3. Missing values

3. Missing values

Harsh MacBook Air

```
>>> from sklearn.impute import SimpleImputer
>>> imputer = SimpleImputer(missing_values = np.nan,
        strategy = 'mean')
>>> imputer = imputer.fit(X[:, 1:3])
>>> X[:, 1:3] = imputer.transform(X[:, 1:3])
```

4. Categorical data

4. Categorical data

Harsh MacBook Air

```
>>> # Encoding the Independent Variable
>>> from sklearn.preprocessing import LabelEncoder, OneHotEncoder
>>> from sklearn.compose import ColumnTransformer
>>> transformer = ColumnTransformer([('one_hot_encoder',
        OneHotEncoder(), [0])],remainder='passthrough')
>>> X = np.array(transformer.fit_transform(X), dtype=np.float)

>>> # Encoding the Dependent Variable
>>> labelencoder_y = LabelEncoder()
>>> y = labelencoder_y.fit_transform(y)
```

5. Train test split

5. Splitting data into train and test set

Harsh MacBook Air

```
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(X, y,
        test_size = 0.2, random_state = 0)
```

6. Feature scaling

6. Feature scaling

Harsh MacBook Air

```
>>> from sklearn.preprocessing import StandardScaler
>>> sc_X = StandardScaler()
>>> X_train = sc_X.fit_transform(X_train)
>>> X_test = sc_X.transform(X_test)
>>> sc_y = StandardScaler()
>>> y_train = sc_y.fit_transform(y_train)
```

Data Preprocessing Template

Any Questions??

Thanks!

https://github.com/harshshinde07

https://www.linkedin.com/in/harshshinde07