

easy integration with other libs.
 ↑ speed & efficiency
 ↑ operations
 ↑ optimized

Features

Numpy

Mathematical Arithmetic Funcs

`np.add(a1, a2)`
`np.subtract(a1, a2)`
`np.multiply(a1, a2)`
`np.divide(a1, a2)`
`np.true_divide(a1, a2)`
`np.floor_divide(a1, a2)`
`np.positive(a)`
`np.negative(a)`
`np.absolute(a)`
`np.mod(a1, a2)`
`np.fmod(a1, a2)`
`np.modf(a1, a2)`
`np.divmod(a1, a2)`
`np.remainder(a1, a2)`
`np.reciprocal(a1)`

Arithmetic Operation

`a1 + a2`

`a1 += a2`

`a1 - a2`

`a1 -= a2`

`a1 * a2`

`a1 *= a2`

`a1 ** a2`

`a1 **= a2`

`a1 / a2`

`a1 /= a2`

`a1 % a2`

`a1 %= a2`

`a1 // a2`

`a1 //= a2`

`a1 @ a2`

(matrix product
`a1.dot(a2)`)

Extrema Finding

`np.maximum(a1, a2)`
`np.max(a)`
`np.amin(a)`
`np.minimum(a1, a2)`
`np.min(a)`
`np.amin(a)`
`np.fmin(a)`
`np.fmax(a)`
`np.average(a)`
`np.nanmax(a)`
`np.nanmin(a)`

Rounding Funcs

`np.round(a)`
`np.around(a)`
`np.trunc(a)`
`np.floor(a)`
`np.ceil(a)`
`np.fix(a)`

Trig/hyperbolic

`np.sin(a)` add 'h'
`np.cos(a)` at every
`np.tan(a)` func end
`np.arcsin(a)` to have
`np.arccos(a)` it a
`np.arctan(a)` hyperbolic
`np.degree(a)` function
`np.radians(a)`
`np.deg2rad(a)`
`np.rad2deg(a)`

Statistical

`np.sum(a)`
`np.prod(a)`
`np.nansum(a)`
`np.nanprod(a)`
`np.amin(a)`
`np.amin(a)`
`np.cumsum(a)`
`np.cumprod(a)`
`np.diff(a)`
`np.nandiff(a)`
`np.cumdiff(a)`
`np.ediff1d(a)`
`np.corrcoef(a, b)`

Others

`np.lcm(a1, a2)`
`np.gcd(a1, a2)`
`np.clip(a, a-min, a-max)`
`np.nan_to_num(a, [copy, nan])`
`np.sign(a)`
`np.fabs(a) // float abs`

Numpy 2.0

Attributes

a.dtype
a.size
a.itemsize
a.data
a.ndim
a.shape

Creating Array

from pylist np.array(list)
from pydict np.array(dict)
from default values generators
↳ np.ones((2,)) or np.ones((2,3))
↳ np.zeros(tuple)
↳ np.empty(tuple)
from arange np.arange(start, end, jump)
from reshape
a.reshape(tuple) (x, y, z...)
np.reshape(a, (x, y, z))

Access

a[index]
a[start]
a[start:end]
a[start:end:jump]

Array Stacking

np.concatenate((a1, a2))
np.vstack((a1, a2))
np.hstack((a1, a2))

Array Splitting

np.split(a, int split)
np.hsplit(a, int split)
np.vsplit(a, int split)

Pandas

Features

- ↑ performance
- dataFrame
- ↑ funs
- ↑ speed, efficiency, plotting
- csv, etc read, write
- missing data handling
- time series analysis
- groupby operations

Series

create: pd

- ↳ From list: `pd.Series(l)`
- ↳ From dict: `pd.Series(d)`
- ↳ From ndarray: `pd.Series(ndarr)`

attributes

- ↳ `pd.index` ↳ `pd.shape`
- ↳ `pd.values` ↳ `pd.size`
- ↳ `pd.dtype`

info

- `pd.head(n)`
- `pd.tail(n)`
- `pd.describe()`
- `pd.unique()`

math & stats

- `pd.sum` `pd.min`
- `pd.max` `pd.mean`
- `pd.mode` `pd.median`
- `pd.std` `pd.var`

missing data handling

- `pd.dropna`
- `pd.fillna`

conversion

- `pd.to_list()`
- `pd.to_numpy()`
- `pd.to_frame()`

DataFrame

create

- From list: `pd.DataFrame(l)`
- From dict: `pd.DataFrame(d)`
- From ndarray: `pd.DataFrame(ndarr)`

access

can be 1st or an array
`df[column][rows]`
 ↓
 can be sliced (`rs: e: js`)
 too (start, end, jump)

attributes

- `df.shape` `df.name`
- `df.size` `df.dtype`

info.info

- `df.head(n)` `df.tail(n)`
- `df.describe()` `df.unique()`

math & stats

- `df.sum()` `df.min()`
- `df.max()` `df.mean()`
- `df.mode()` `df.median()`
- `df.std()` `df.var()`

sorting

- `df.sort_values(by, ascending)` T/F
- `df.sort_index()`
- `df.rank()`

merging

- `df.groupby('col')`
- `df.merge()`
- `df.concat()`
- `df.join()`

convert

- `df.to_numpy()`
- `df.to_excel()`
- `df.to_json()`
- `df.to_csv()`
- `df.to_dict()`

Pandas 2.0

CSVs

read pd.read_csv

```
pd.read_csv(filename, buffer,  
sep=',', engine=None,  
usecols=None, header='infer',  
nrows=None, skiprows=None)
```

Save to CSV

```
pd.to_csv(filename, buffer,  
sep=',', header=None)
```

Data cleaning

NaN → `dropna()`

`fillna()`





`pduplicate` → `drop_duplicates()`

outliers → manual...

Plotting

```
df.plot(x='col1',  
y='col2',  
kind='line')
```

x & y are column names
and it is displayed by
`plt.show()`
type of plottings...

	1> scatter	3> hist	
	2> line	4> bar	

MatPlot Lib

Features

plotting, animation
image save
object oriented interface
customization
multiple plotting styles
3D plotting
animation
integration with Django

Terms

Figure: main plot area
axes: graph plain
artist: everything visible
Axis: You should know
xlabel, ylabel, title
majortick, minortick
majorticklabel, minorticklabel
marker, line, legends
grids, spine (outline)

Basic Names

plt.show()
plt.legend()
plt.grid(True/False)
plt.title()
plt.xlabel()
plt.ylabel()

plt.plot(x, y, color, linestyle, marker, label)

or simply

plt.plot(x, y, 'fmt')

where fmt is

'colormarkerlinestyle'
↓
'r', 'g', 'b', 'k' (black)

marker: 'o' (circle)

's' (square)

'.' (point)

'^' (triangle)

linestyle: '-' (solid line)

'--' (dashed)

'.' (dot)

'-.' (dashdot)

eg. fmt can be

'ro--', 'k^'

'gs.', 'b.-'

Plot types

plt.line()



plt.pie()



plt.bar()



plt.hist()



plt.scatter()



Sub Plots

plt.subplot(2,2)
creation

access ... (1)

plt.subplot(2,2,1)

plt.xlabel ...

SKLearn

Features

ml learnings
inbuilt ml algo
inbuilt onaylis algo
inbuilt datasets
inbuilt testing algos
consistent API
community & document
integration with lib
other ml evaluation
and more

Steps to Build ml model

1. import lib and datasets
2. clean, optimise, handle dataset
(missing data, duplicate,
outliers)
3. Seperate, identify features
and outcomes if any.
4. split dataset into training &
testing
5. select appropriate ml model
6. evaluate ~~neural~~ train
7. evaluate result
8. refine/improve
accordingly.

PREPARING TO MODEL AND PREPROCESSING

Machine Learning Activities

1. Preparing to Model

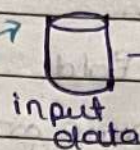
- understand data type
- exploring data, nature
- exploring interrelation between features
- handling missing, outliers, duplicate data errors.

2. Learning

- Partitioning (Semi test, but like practice exam, we can change para of train, validate, test)
 (mode by its performance)
- Structure (Feature, o/p)
- Model Selection
 - Supervised
 - classification
 - regression
 - unsupervised
 - clustering
 - association analyses
- Training (let it learn)
 (evaluate)
- Validating (change para according validation dataset result)
- Improve performance (para)
- Test.

3. Performance Evaluation

'analyzing results, &'



Preparing To Model

Learning

refined data

performance improvement

performance evaluation

Types of Data

1. Quantitative

- Interval
- Ratio

2. Qualitative → nominal Ordinal

Data Quality

'prc of apply quality measurement techs'

1. Incorrect Sample Selection (biased data)
2. Errors in Data Set (missing, duplicates, empty, outliers)

Data Remediation

- | | |
|--------------|-----------|
| Outliers | missing |
| - replace | - replace |
| - remove | - mean |
| - clip (cap) | - remove |

Performance Evaluation:

(continue...)

- Cross Validation
- Confusion Matrix
 - K Fold
 - leave one out etc.

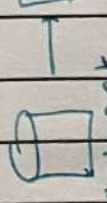
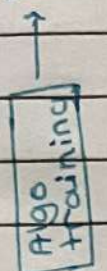
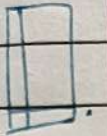
4. Performance Improvement

- Tuning model (change param)
- Ensemble model
(use many models, avg result)

Data Pre-Processing

1. Dimensionality Reduction
(remove, compress, create one for many)
Features to optimize
2. Feature Subset Selection
(selecting few which are necessary)

decision making system



A computer is said to be learn from experience E with respect to performance P and on the specific task T , if its performance at t improve with measure of P , with experience E

Date _____
Page _____

INTRODUCTION TO MACHINE LEARNING

MACHINE LEARNING: Field of AI that focuses on creating algorithms and statistical models that enable the machine to find pattern and learn from data given in order to improve its performance without being programmed externally with

Applications

- Fraud detection
- Marketing
- Predicting tumor probability
- Analyzing medical reports
- Stock market analysis
- Agricultural analysis (crop yield, crop, diseases)
- Health experts
- Sports
- weather forecast
- manufacturing
- servicing
- Tech improve
- Scientific predictions
- entertainment (movie analysis)
- generative AI
- computer vision

Human vs Machine

Types

required data to train	↓ less	↑ more
speed (x complex)	less	more
power	less	more
noise	yes	depend on data
emotions	yes	no
input	senses (hear, speak, see, etc)	data sets... (visuals etc)
storage & processing	brain, natural neural networks	device, data structure, artificial NN...
long term storage (olp)	less probability	none until manually instructed
learning improve-ments	Feedback, observations	parameters changes etc...
ethical	more, based on environment	based on motive of programmer
e.g.	child learning	fraud detection

1. Supervised
 - ↳ Simple linear $1F \rightarrow 1D$
 - ↳ multi linear $1+F \rightarrow 1D$
 - ↳ multivariant $1+olp$
2. Unsupervised
 - ↳ clustering
 - ↳ hard (1 point \rightarrow 1 cluster)
 - ↳ soft (1 point \rightarrow 1+ cluster)
 - ↳ dimensionality Reduction
 - ↳ Association Analyses
3. Reinforcement

Tools & Tech

lang
Py, R

lib for data
numpy, pandas

lib for ml
sklearn, tensor flow, pytorch

lib for visual
matplotlib, seaborn

collab
jupyter
cree

INTRODUCTION TO MACHINE LEARNING

MACHINE LEARNING: Field of AI that focuses on creating algorithms and statistical models that enable the machine to find pattern and learn from data given in order to improve its performance without being programmed externally with fixed rules.

Human vs Machine

Types

required data to train	↓ less	↑ more	1. <u>Supervised</u>
Speed (xcomplex)	less	more	→ Classification
power	less	more	↳ binary
noise	yes	depend on data	↳ multiclass
emotions	yes	no	↳ multilabel
input	person (hear, speak, see, etc)	data sets (images etc)	→ Regression
storage & processing	brain, natural neural networks	device, data structure, artificial NN...	↳ Simple linear $1f \rightarrow 1op$
long term storage (elp)	less probably	more until manually instructed	↳ multi linear $1+f \rightarrow 1op$
learning improvements	Feedback, observations	parameters change etc...	↳ multivariant $1+op$
ethical	more, based on environment	based on motive of programmer	2. <u>Unsupervised</u>
e.g.	child learning	fraud detection	→ clustering
			↳ hard (point $\rightarrow 1$ cluster)
			↳ soft (1 point $\rightarrow 1+$ cluster)
			→ Dimensionality Reduction
			→ Association Analyses
			3. <u>Reinforcement</u>

Tools & Tech

lang
Py, R

lib for data
numpy, pandas

lib for ml
sklearn, tensorflow, pytorch

lib for visual
matplotlib, seaborn

could
colab, jupyter, CRED etc