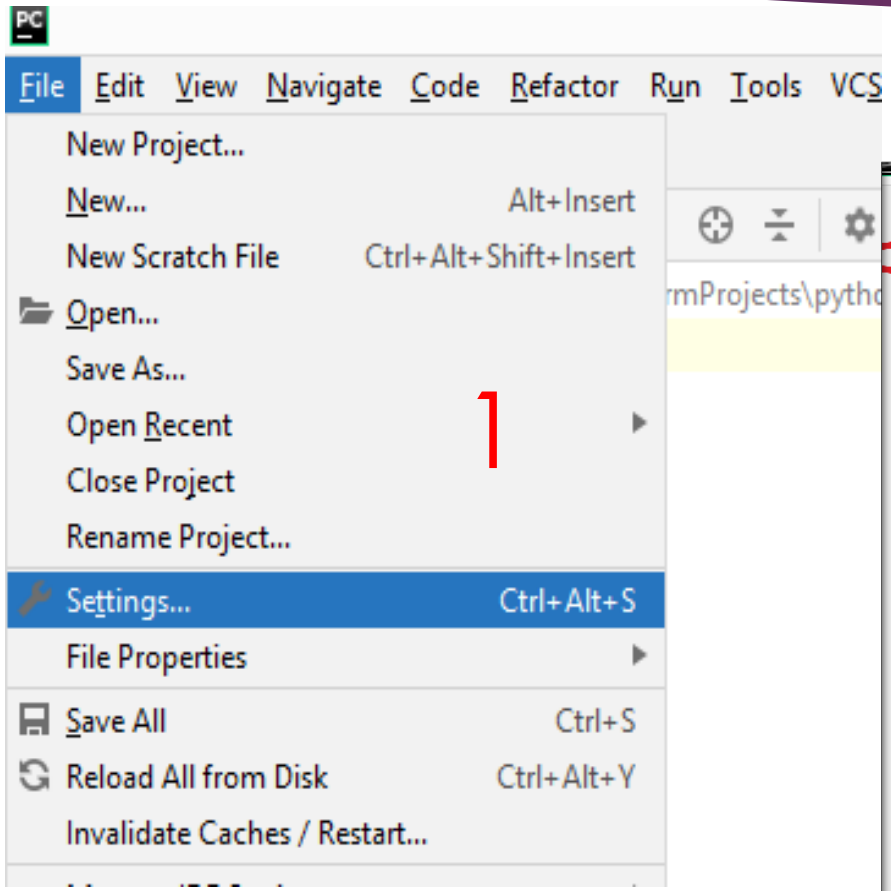
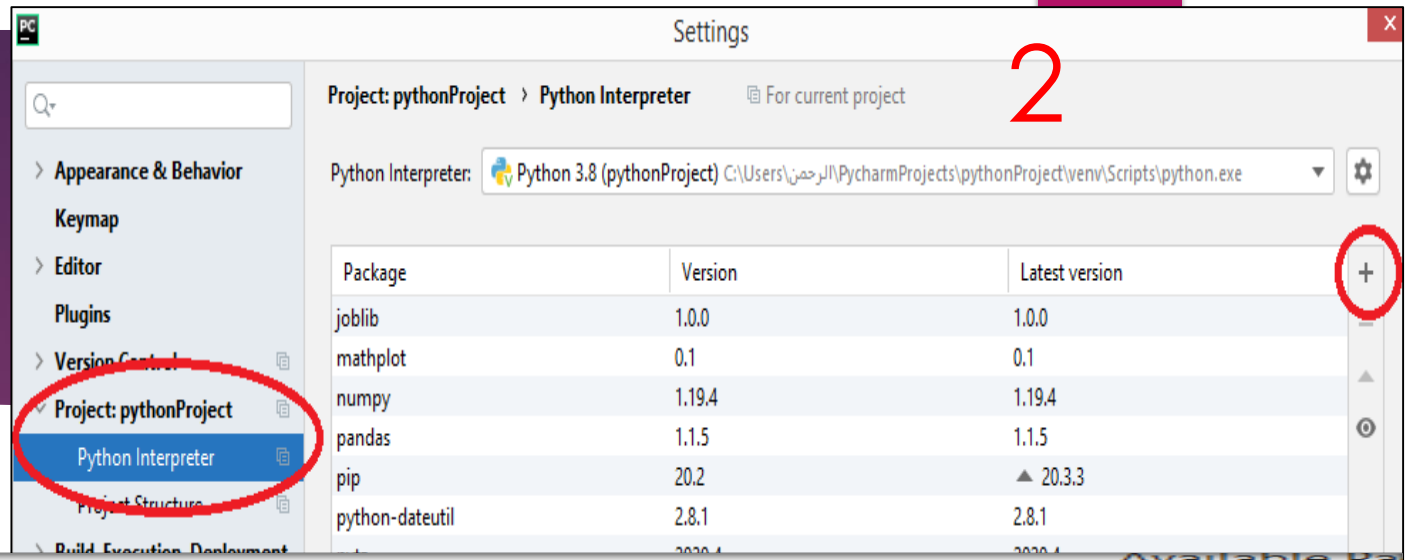
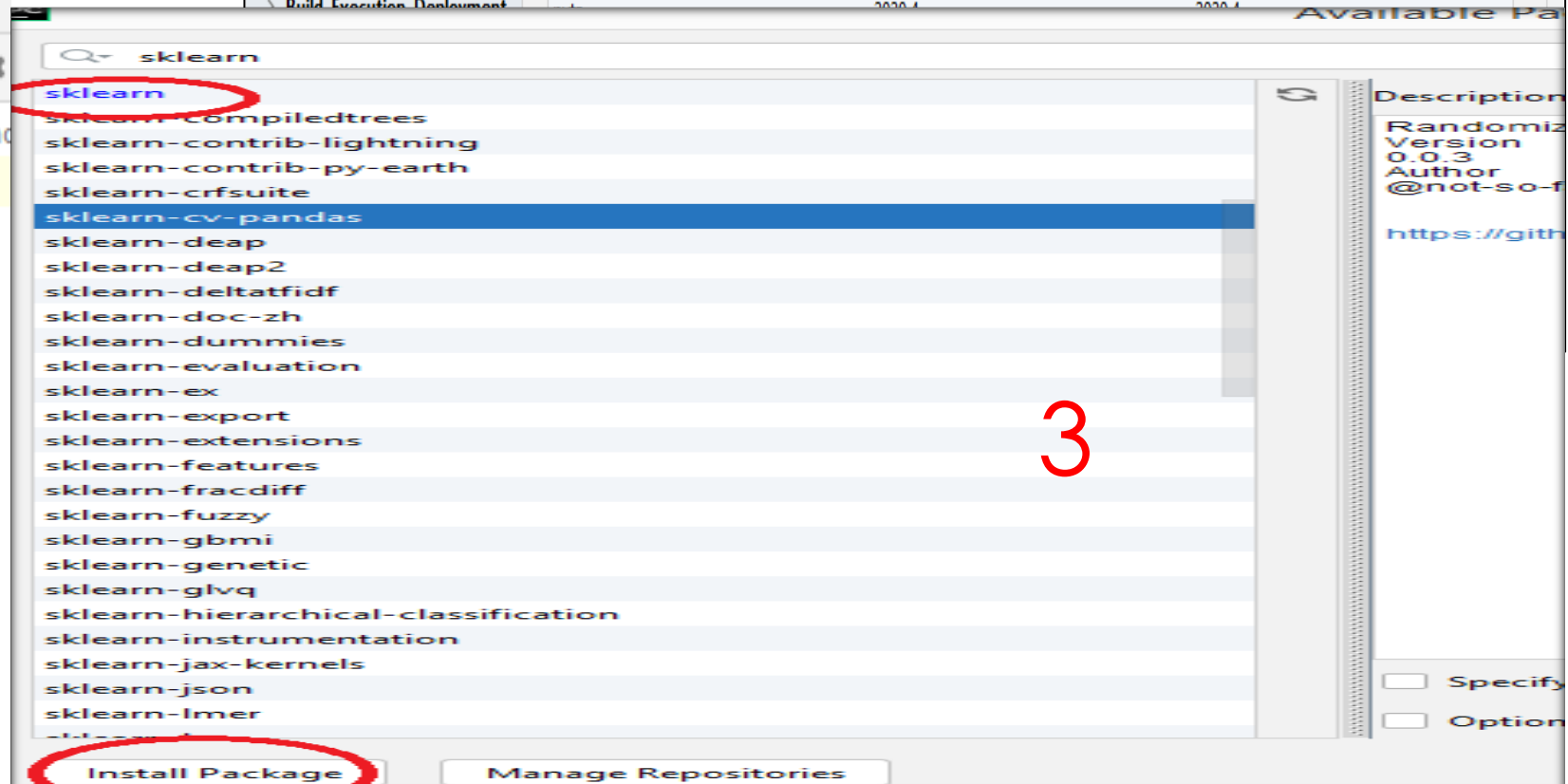
Section1: Sklearn

SCIKIT LEARN

Sklearn installing







Test SKlearn

```
1  import sklearn
2  print(sklearn.__version__)
3
```

Sklearn library

Data Preparation

- Data files from SKlearn
- Data cleaning
- Metrics module
- Feature selection
- Data Scaling
- Data Split

ML Algorithms

- Linear Regression
- Logistic Regression
- Neural Network
- SVR
- SVC
- Decision Tree
- Ensemble Regression

Algorithm Evaluation :

- Model Check
- Grid Search
- Pipeline
- Model Save

Terminologies in Sklearn

- Library : مكتبة سياكيتليرن كلها
- Module : الفروع
- Class : الالجوريثم المستخدم
- Object : الكائن المستنسخ من الكلاس لعمل التدريب و التوقع
- Parameters : البيانات المطلوبة في الالجوريثم
- Attributes : المعلومات الخارجة من الالجوريثم
- Methods : الدوال الموجودة في الألجوريثم

Terminologies in Sklearn
Ex: LinearRegression Algorithm

from **sklearn**.linear model import LinearRegression	**Library**
from sklearn.**linear model** import LinearRegression	**Module**
from sklearn.linear_model import **LinearRegression**	**Class**
LR = LinearRegression(fit_intercept=True, normalize=True)	**Object**
LR = LinearRegression(**fit_intercept=True, normalize=True**)	**Parameters**
LR.**coef_** , LR.**intercept_**	**Attributes**
LR.**fit**(X, y) , LR.**score**(X, y) , LR.**predict**(X_test)	**Methods**

Terminologies in Sklearn (Module)

Terminologies in Sklearn (Class)

ML Algorithms

- Linear Regression
- Logistic Regression
- Neural Network
- SVR
- SVC
- Decision Tree
- Ensemble Regression

```
1  import sklearn
2  print(sklearn.__version__)
3
4  from sklearn.svm import SVR
5
6  from sklearn.linear_model import LinearRegression
7
8
```

Terminologies in Sklearn
Ex: LinearRegression Algorithm

from **sklearn**.linear model import LinearRegression	**Library**
from sklearn.**linear model** import LinearRegression	**Module**
from sklearn.linear_model import **LinearRegression**	**Class**
LR = LinearRegression(fit_intercept=True, normalize=True)	**Object**
LR = LinearRegression(**fit_intercept=True, normalize=True**)	**Parameters**
LR.**coef_** , LR.**intercept_**	**Attributes**
LR.**fit**(X, y) , LR.**score**(X, y) , LR.**predict**(X_test)	**Methods**

Terminologies in Sklearn (Object)

```
1
2   from sklearn.linear_model import LinearRegression
3
4   LR = LinearRegression()
5
6
7
```

```
from sklearn.linear_model import LinearRegression

LR1 = LinearRegression()
LR2 = LinearRegression()
LR3 = LinearRegression()
LR4 = LinearRegression()
```

Terminologies in Sklearn
Ex: LinearRegression Algorithm

from **sklearn**.linear model import LinearRegression	**Library**
from sklearn.**linear model** import LinearRegression	**Module**
from sklearn.linear_model import **LinearRegression**	**Class**
LR = LinearRegression(fit_intercept=True, normalize=True)	**Object**
LR = LinearRegression(**fit_intercept=True, normalize=True**)	**Parameters**
LR.**coef_** , LR.**intercept_**	**Attributes**
LR.**fit**(X, y) , LR.**score**(X, y) , LR.**predict**(X_test)	**Methods**

Terminologies in Sklearn (Parameters)

29.18.8 sklearn.linear_model.LinearRegression

class sklearn.linear_model.**LinearRegression** (*fit_intercept=True, normalize=False, copy_X=True, n_jobs=1*)

Ordinary least squares Linear Regression.

Parametersfit_intercept : boolean, optional

whether to calculate the intercept for this model. If set to false, no intercept will be used in calculations (e.g. data is expected to be already centered).

normalize : boolean, optional, default False

If True, the regressors X will be normalized before regression. This parameter is ignored when *fit_intercept* is set to False. When the regressors are normalized, note that this makes the hyperparameters learnt more robust and almost independent of the number of samples. The same property is not valid for standardized data. However, if you wish to standardize, please use *preprocessing.StandardScaler* before calling *fit* on an estimator with *normalize=False*.

Terminologies in Sklearn (Parameters)

```
1
2  from sklearn.linear_model import LinearRegression
3  LR1 = LinearRegression()
4  LR2 = LinearRegression(normalize=True)
5  LR3 = LinearRegression(normalize=False)
6
7
8
9
```

The model uses the default values of all parameters

Terminologies in Sklearn
Ex: LinearRegression Algorithm

from **sklearn**.linear model import LinearRegression	**Library**
from sklearn.**linear model** import LinearRegression	**Module**
from sklearn.linear_model import **LinearRegression**	**Class**
LR = LinearRegression(fit_intercept=True, normalize=True)	**Object**
LR = LinearRegression(**fit_intercept=True, normalize=True**)	**Parameters**
LR.**coef_** , LR.**intercept_**	**Attributes**
LR.**fit**(X, y) , LR.**score**(X, y) , LR.**predict**(X_test)	**Methods**

Parameters:

fit_intercept : *bool, default=True*

Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

copy_X : *bool, default=True*

If True, X will be copied; else, it may be overwritten.

n_jobs : *int, default=None*

The number of jobs to use for the computation. This will only provide speedup in case of sufficiently large problems, that is if firstly `n_targets > 1` and secondly `X` is sparse or if `positive` is set to `True`. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

positive : *bool, default=False*

When set to `True`, forces the coefficients to be positive. This option is only supported for dense arrays.

New in version 0.24.

Attributes:

coef_ : *array of shape (n_features,) or (n_targets, n_features)*

Estimated coefficients for the linear regression problem. If multiple targets are passed during the fit (y 2D), this is a 2D array of shape (n_targets, n_features), while if only one target is passed, this is a 1D array of length n_features.

rank_ : *int*

Rank of matrix `X`. Only available when `X` is dense.

singular_ : *array of shape (min(X, y),)*

Singular values of `X`. Only available when `X` is dense.

Terminologies in Sklearn (Methods)

```
1
2   from sklearn.linear_model import LinearRegression
3   LR1 = LinearRegression()
4   LR2 = LinearRegression(normalize=True)
5
6   LR1.fit(x,y)
7
8
```

Terminologies in Sklearn

```
from sklearn.linear_model import LinearRegression

# Applying Linear Regression model
LR=LinearRegression(fit_intercept=True, n_jobs=-1,copy_X=True)
LR.fit(x,y)

#Calculating Details
print('Linear Regression train score=',LR.score(x,y))
print('Linear Regression coef=',LR.coef_)
print('Linear Regression intercept=',LR.intercept_)
```

Data Preparation

1. Data files from SKlearn
2. Data cleaning
3. Metrics module
4. Feature selection
5. Data Scaling
6. Data Split

Data Files in Sklearn

- Iris data
- Digits data
- Boston data
- Wine data
- Breast cancer data
- Diabetes data
- Sample Regression
- Sample Classification
- Sample images

Data Files in Sklearn

Data Files in Sklearn

	A	B	C	D	E	F	G	H	I	J
1	V1	V2	V3	V4	V5	V6	V7	V8	V9	Class
2	87	92	89	32	97	59	91	32	5	'2'
3	61	4	85	77	56	47	99	84	86	'2'
4	91	28	20	42	18	87	105	39	62	'2'
5	71	39	87	18	88	75	40	16	98	'2'
6	68	98	82	101	59	48	98	1	88	'2'
7	72	80	44	41	38	39	85	41	90	'2'
8	51	76	17	5	2	20	66	96	72	'2'
9	48	81	70	62	30	32	71	4	74	'2'
10	81	103	80	14	5	85	8	15	29	'2'
11	77	104	84	11	94	67	4	13	6	'2'
12	75	61	92	39	41	81	94	37	100	'2'
13	85	5	15	30	84	86	104	26	8	'2'
14	80	102	19	65	82	76	6	8	102	'2'
15	69	50	83	12	99	78	103	14	94	'2'
16	84	6	57	38	4	63	92	36	104	'2'
17	73	72	73	36	1	79	100	35	96	'2'
18	44	101	5	8	33	45	80	97	71	'2'
19	54	95	68	73	49	60	87	3	82	'2'
20	59	3	90	33	92	46	64	33	93	'2'
21	53	100	96	20	90	56	69	22	92	'2'
22	85	96	71	95	54	42	62	53	105	'2'
23	26	11	43	48	23	93	101	43	37	'4'

Data Files in Sklearn (iris dataset)

```
1  # Import Libraries
2  from sklearn.datasets import load_iris
3  #-----------------------------------------
4
5  #load iris data
6
7  IrisData = load_iris()
8
9  #X Data
10 X = IrisData.data
11 print('X Data is \n' , X[:10])
12 print('X data is \n' , IrisData.data[:10])
13 #-----------------------------------------------------
14 print('X shape is ' , X.shape)
15 print('X shape is ' , IrisData.data.shape)
16 #-----------------------------------------------------
17 print('X Features are \n' , IrisData.feature_names)
18
```

data attribute contains the feature data of the Iris dataset

Print samples from sample 0 to sample number 10

Shape: returns a tuple representing the dimensions of the Dataset.

Feature_names: provides the names of the features (i.e., the columns) in the Iris dataset.

Data Files in Sklearn (iris dataset)

```
# Import Libraries
from sklearn.datasets import load_iris
#----------------------------------------------------
#load iris data

IrisData = load_iris()

#X Data
X = IrisData.data
print('X Data is \n' , X)
print('X Data is \n' , X[:10])
print('X data is \n' , IrisData.data[:10])
#----------------------------------------------------
print('X shape is ' , X.shape)
print('X shape is ' , IrisData.data.shape)
#----------------------------------------------------
print('X Features are \n' , IrisData.feature_names)
#----------------------------------------------------

y = IrisData.target
print('y Data is \n' , y)
print('y Data is \n' , y[:10])
print('y shape is ' , y.shape)
print('y Columns are \n' , IrisData.target_names)
```

target attribute contains the target variable or labels corresponding to each sample in the Iris dataset.

target_names contains the names of the classes

Data Files in Sklearn (load digits dataset)

```
#Import Libraries
from sklearn.datasets import load_digits
#-----------------------------------------------------------

#load digits data

DigitsData = load_digits()

#X Data
X = DigitsData.data
print('X Data is \n' , X[:2])
print('X shape is ' , X.shape)
print('X Features are \n' , DigitsData.feature_names)

#y Data
y = DigitsData.target
print('y Data is \n' , y[:2])
print('y shape is ' , y.shape)
print('y Columns are \n' , DigitsData.target_names)
```

a dataset consisting of 8x8 pixel images of handwritten digits.

Data Files in Sklearn (load digits dataset)

```
11
12  #y Data
13  y = DigitsData.target
14  print('y Data is \n', y[:2])
15  print('y shape is ', y.shape)
16  print('y Columns are \n', DigitsData.target_names)
17
18  #-------------------------------------------------------------------
19  import matplotlib.pyplot as plt
20  plt.gray()
21  plt.matshow(DigitsData.images[0])
22  print('------------------------------')
23  plt.show()
```

Data Files in Sklearn (load digits dataset)

```
#-----------------------------------------------------------
import matplotlib.pyplot as plt
plt.gray()
for g in range(3):
    print('Images of Number : ', g)
    plt.matshow(DigitsData.images[g])
    print('-----------------------------')
    plt.show()
```

Data Files in Sklearn (load-boston dataset)

```
2  from sklearn.datasets import load_boston
3  #-------------------------------------------------------
4
5  #load boston data
6
7  BostonData = load_boston()
8
9  #X Data
10 X = BostonData.data
11 print('X Data is \n' , X[:3])
12 print('X shape is ' , X.shape)
13 print('X Features are \n' , BostonData.feature_names)
14
15 #y Data
16 y = BostonData.target
17 print('y Data is \n' , y[:3])
18 print('y shape is ' , y.shape)
```

Data Files in Sklearn: Task

- Import Class : load_breast_cancer
- Module name : datasets
- Print the first three records
- Print the size of the dataset
- Print the classes names of the first 100 records
- Print the name of the features
- Print the names of the targets

Data Files in Sklearn (Answer)

```
#Import Libraries
from sklearn.datasets import load_breast_cancer
#-----------------------------------------------

BreastData = load_breast_cancer()

#X Data
X = BreastData.data
print('X Data is \n' , X[:3])
print('X shape is ' , X.shape)
print('X Features are \n' , BreastData.feature_names)

#y Data
y = BreastData.target
print('y Data is \n' , y[:100])
print('y shape is ' , y.shape)
print('y Columns are \n' , BreastData.target_names)
```