Experiment No - 03

Getting Started With ScikitLearn Library

In [1]:
```
print('-------------------------------EXPERIEMENT-03-------------------
---------------')
print('NAME:  Pratyush Srivastava')
print('ROLL NO: 18SCSE1010128')
```

```
-------------------------------EXPERIEMENT-03--------------------------
--------
NAME:  Pratyush Srivastava
ROLL NO: 18SCSE1010128
```

Fitting and Predicting: Estimator Basics

In [2]:
```
#Importing of Random Forest Classifier From Sklearn ensemble
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(random_state=0)
```

In [3]:
```
#Creating a raw data as X as 2 samples and 3 features
X = [[ 1,  2,  3], [11, 12, 13]]
X
```

Out[3]:
```
[[1, 2, 3], [11, 12, 13]]
```

In [4]:
```
#Creating Classes of each sample
y = [0, 1]
```

In [5]:
```
#Train The model by using fit function
model.fit(X,y)
```

Out[5]:
```
RandomForestClassifier(random_state=0)
```

In [6]:
```
#Prediction of classes of training data
print(model.predict(X))
```

```
[0 1]
```

In [7]:
```
#Prediction of Classes on New data points
model.predict([[4, 5, 6], [14, 15, 16]])
```

Out[7]:
```
array([0, 1])
```

Transformers and pre-processors

In [8]:
```
#Importing of Standard Scaler From Sklearn Preprocessor
from sklearn.preprocessing import StandardScaler
```

In [9]:
```
#Create a raw Data as X
X = [[0, 15],[1, -10]]
X
```

Out[9]:
```
[[0, 15], [1, -10]]
```

In [10]:
```
#Standardization of Data by Using Standard Scaler , Fit and Transform
StandardScaler().fit(X).transform(X)
```

Out[10]:
```
array([[-1.,  1.],
       [ 1., -1.]])
```

Pipelines: chaining pre-processors and estimators

In [11]:

```
#Importing all important packages(like Standard Scaler , Logistic Regre
ssion , make_pipeline , load_iris,  train_test_split ,
# accuracy_score)  From Sklearn Library..............
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [12]:

```
#Create a pipeline object
pipe = make_pipeline(StandardScaler(),LogisticRegression(random_state=0
))
```

In [13]:

```
# load the iris dataset and split it into train and test sets
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=
0)
```

In [14]:

```
# Train the whole pipeline by using Fit Function.....
pipe.fit(X_train, y_train)
```

Out[14]:

```
Pipeline(steps=[('standardscaler', StandardScaler()),
                ('logisticregression', LogisticRegression(random_state=
0))])
```

In [15]:

```
# We can now See the Accuracy Score of The model...
accuracy_score(pipe.predict(X_test), y_test)
```

Out[15]:

```
0.9736842105263158
```

Model Evaluation

In [16]:

```
#Import Make Regression , Linear Regression and Cross validate from Skl
earn Library
```

```
from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
```

In [17]:

```
#Using make_regression create a linear regression model
X, y = make_regression(n_samples=1000, random_state=0)
lr = LinearRegression()
```

In [18]:

```
result = cross_validate(lr, X, y)  # defaults to 5-fold CV
result['test_score']  # r_squared score is high because dataset is easy
```

Out[18]:

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

Automatic parameter searches.......

In [19]:

```
#Importing all libraries like fetch_california_housing , RandomForestRe
gressor, RandomizedSearchCV , train_test_split
#randint from Sckitlearn and scipy

from sklearn.datasets import fetch_california_housing
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import train_test_split
from scipy.stats import randint
```

In [20]:

```
#Fetching of data
#Splitting of Data into training and testing data points

X, y = fetch_california_housing(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=
0)
```

In [21]:

```
#define the parameter space that will be searched over
param_distributions = {'n_estimators': randint(1, 5),'max_depth': randi
nt(5, 10)}
```

In [22]:

```
#now create a searchCV object and fit it to the data
search = RandomizedSearchCV(estimator=RandomForestRegressor(random_stat
e=0),n_iter=5,
                                    param_distributions=param_distributions
,
                                    random_state=0)
```

In [23]:

```
#Training of Data by using Fit function
search.fit(X_train, y_train)
```

Out[23]:

```
RandomizedSearchCV(estimator=RandomForestRegressor(random_state=0), n_i
ter=5,
                   param_distributions={'max_depth': <scipy.stats._dist
n_infrastructure.rv_frozen object at 0x000002AA9F214460>,
                                        'n_estimators': <scipy.stats._d
istn_infrastructure.rv_frozen object at 0x000002AA9F214220>},
                   random_state=0)
```

In [24]:

```
search.best_params_
```

Out[24]:

```
{'max_depth': 9, 'n_estimators': 4}
```

In [25]:

```
# the search object now acts like a normal random forest estimator
# with max_depth=9 and n_estimators=4
#Now we are going to check the score of the model
search.score(X_test, y_test)
```

Out[25]:

```
0.735363411343253
```

In [ ]:

```

```