## **Experiment-03**

In [ ]:

*#Tushar [ 18SCSE1010487 ]*

In [21]:

**from** sklearn.ensemble **import** RandomForestClassifier

**from** sklearn.datasets **import** make\_regression

**from** sklearn.linear\_model **import** LinearRegression **from** sklearn.model\_selection **import** cross\_validate **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 clf **=** RandomForestClassifier(random\_state**=**0) X **=** [[ 1, 2, 3],[11, 12, 13]]

y **=** [0, 1]

clf.fit(X, y)

RandomForestClassifier(random\_state**=**0)

Out[21]: RandomForestClassifier(random\_state=0)

In [22]:

clf.predict(X)

Out[22]:

array([0, 1])

In [23]:

clf.predict([[4, 5, 6], [14, 15, 16]])

Out[23]:

array([0, 1])

9/13/2020 Experiment3-18SCSE1010487 - Jupyter Notebook

localhost:8888/notebooks/Documents/Experiment3-18SCSE1010487.ipynb

In [24]:

**from** sklearn.preprocessing **import** StandardScaler X **=** [[0, 15],[1, **-**10]]

StandardScaler().fit(X).transform(X)

Out[24]: array([[-1., 1.],

[ 1., -1.]])

In [25]:

pipe **=** make\_pipeline(StandardScaler(), LogisticRegression(random\_state**=**0)) X, y **=** load\_iris(return\_X\_y**=True**)

X\_train, X\_test, y\_train, y\_test **=** train\_test\_split(X, y, random\_state**=**0) pipe.fit(X\_train, y\_train)

Out[25]: Pipeline(steps=[('standardscaler', StandardScaler()),

('logisticregression', LogisticRegression(random\_state=0))])

In [26]:

accuracy\_score(pipe.predict(X\_test), y\_test)

Out[26]: 0.9736842105263158

In [27]:

X, y **=** make\_regression(n\_samples**=**1000, random\_state**=**0) lr **=** LinearRegression()

9/13/2020 Experiment3-18SCSE1010487 - Jupyter Notebook

localhost:8888/notebooks/Documents/Experiment3-18SCSE1010487.ipynb

In [29]:

result **=** cross\_validate(lr, X, y) result['test\_score']

Out[29]: array([1., 1., 1., 1., 1.])

In [30]:

|  |  |  |
| --- | --- | --- |
| **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  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)  param\_distributions **=** {'n\_estimators': randint(1, 5),'max\_depth': randint(5, 10)  search **=** RandomizedSearchCV(estimator**=**RandomForestRegressor(random\_state**=**0), n\_i search.fit(X\_train, y\_train) | | |
|  |  |  |

Out[30]: RandomizedSearchCV(estimator=RandomForestRegressor(random\_state=0), n\_iter=5,

param\_distributions={'max\_depth': <scipy.stats.\_distn\_infras tructure.rv\_frozen object at 0x000002544520C400>,

'n\_estimators': <scipy.stats.\_distn\_inf rastructure.rv\_frozen object at 0x0000025442624978>},

random\_state=0)

In [31]:

search.best\_params\_

Out[31]: {'max\_depth': 9, 'n\_estimators': 4}

In [32]:

search.score(X\_test, y\_test)

Out[32]: 0.735363411343253

In [ ]: