14/08

EX-3 : STUDY OF CLASSIFIERS WITH RESPECT TO STATISTICAL PARAMETERS

AIM: To study of classifiers with respect to statistical parameters.

Algorithm:

=> Load an open source dataset

=> Split the dataset into training and testing sets

=> Initialize multiple classifiers

=> Train each classifiers

=> Predict on test set using each model

=> Evaluate each model using:

- Accuracy
- Recall
- Precision
- F1-score

=> Compare results.

PSEUDOCODE:

```
BEGIN
    load data set
    split data into training and test
    standardize the features
    Initialize classifier
    For each classifier:
            Train
            Predict
            calculate Accuracy
            store results
    END FOR
        Display and compare
END
```

observation:

classifier	Decision Tree	SUM	Logistic Regression
Accuracy	84.7%	98.6%	97.4%
Avg precision	85.3%	98.7%	97.6%
Avg. Recall	83.9%	98.6%	97.3%
Avg F1-score	84.3%	98.6%	97.61%
weighted F1 score	84.7%	98.61%	97.8%

Code:

```
from sklearn.datacits import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import Standand Scalar
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborclassifier
from sklearn.svm import SVC
from sklearn.tree import decision_tree_classifier
from sklearn.metrices import classification_report,
                                 confusionmatrix

data = load_iris()
x = data.data
y = data.target
x_train, x_test, y_train, y_test = train_test_split
(x, y, test_size = 0.3, random slate = 43)
scalar = Standard_Scalar()
x_train = Scalar fit_transform(x_train)
x_test = Scalar.transform(x_test)
classifers = {
    "Logistic Regression" : Logistic Regression()
    "K-Neighbor : K NC (n_neighbor = 3),
    " Support Vector Machine" : SVC()
    " Decision Tree" : Decision Tree ()
}
```

Overall :

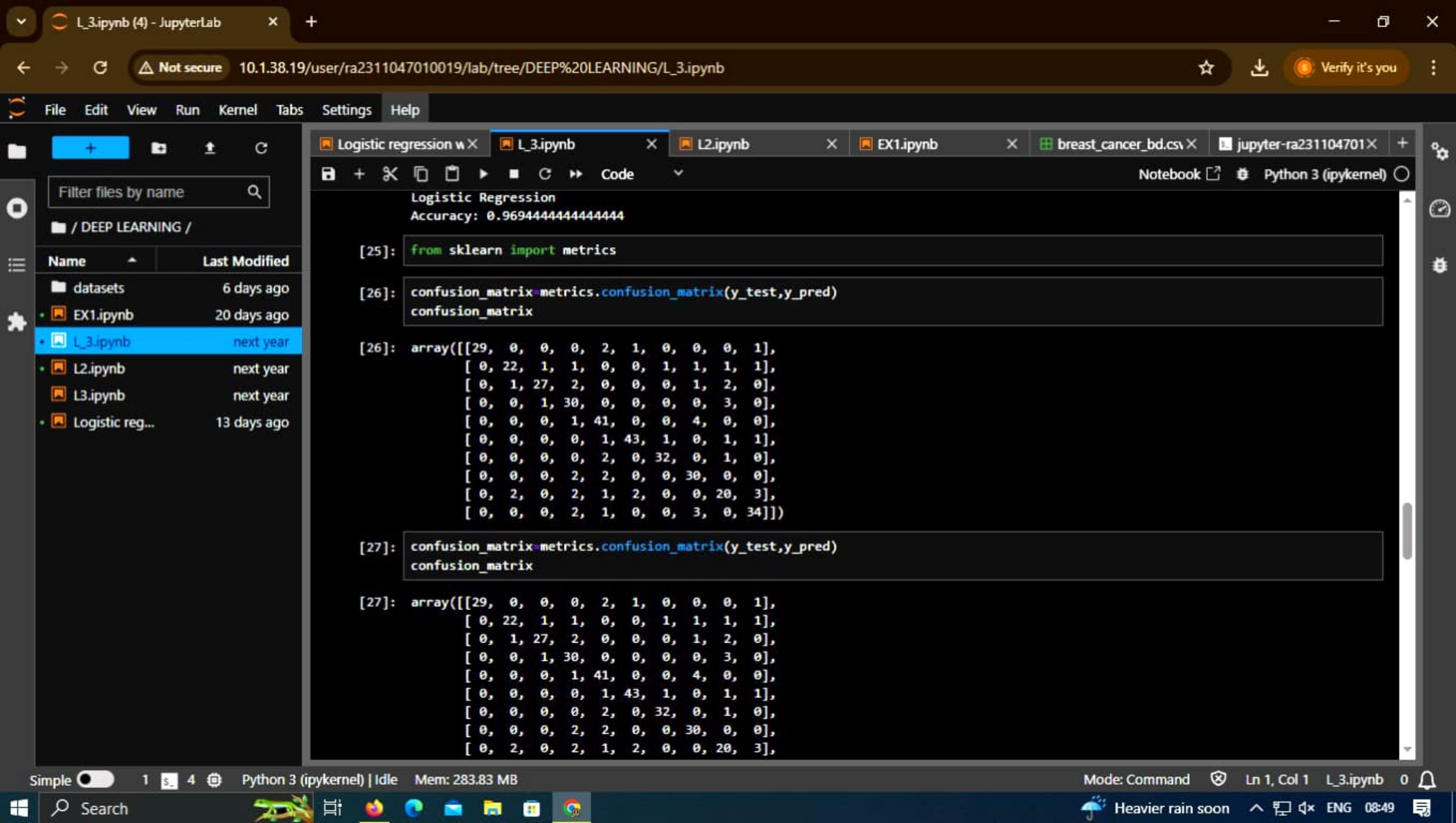
- SVM provides best overall classification
- Decision tree shows sign of overfitting.
- Logistic Regression is very close in performance.

11/4/25

Result :

Study of classifers with respect to statistical parameters completed.

```
for name, clf in classifiers.items():
    clf.fit (x_train, y_train)
    ypred = clf.predict (x_test)
    print (f"=== {name} ===")
    print ("Conf mat")
    print(confusion_matrix (y_test, y_pred))
    print (classification_report (y_test, y_pred,
      target_names= data.target_names))
```



```
Logistic Regression
Accuracy: 0.9694444444444444
```

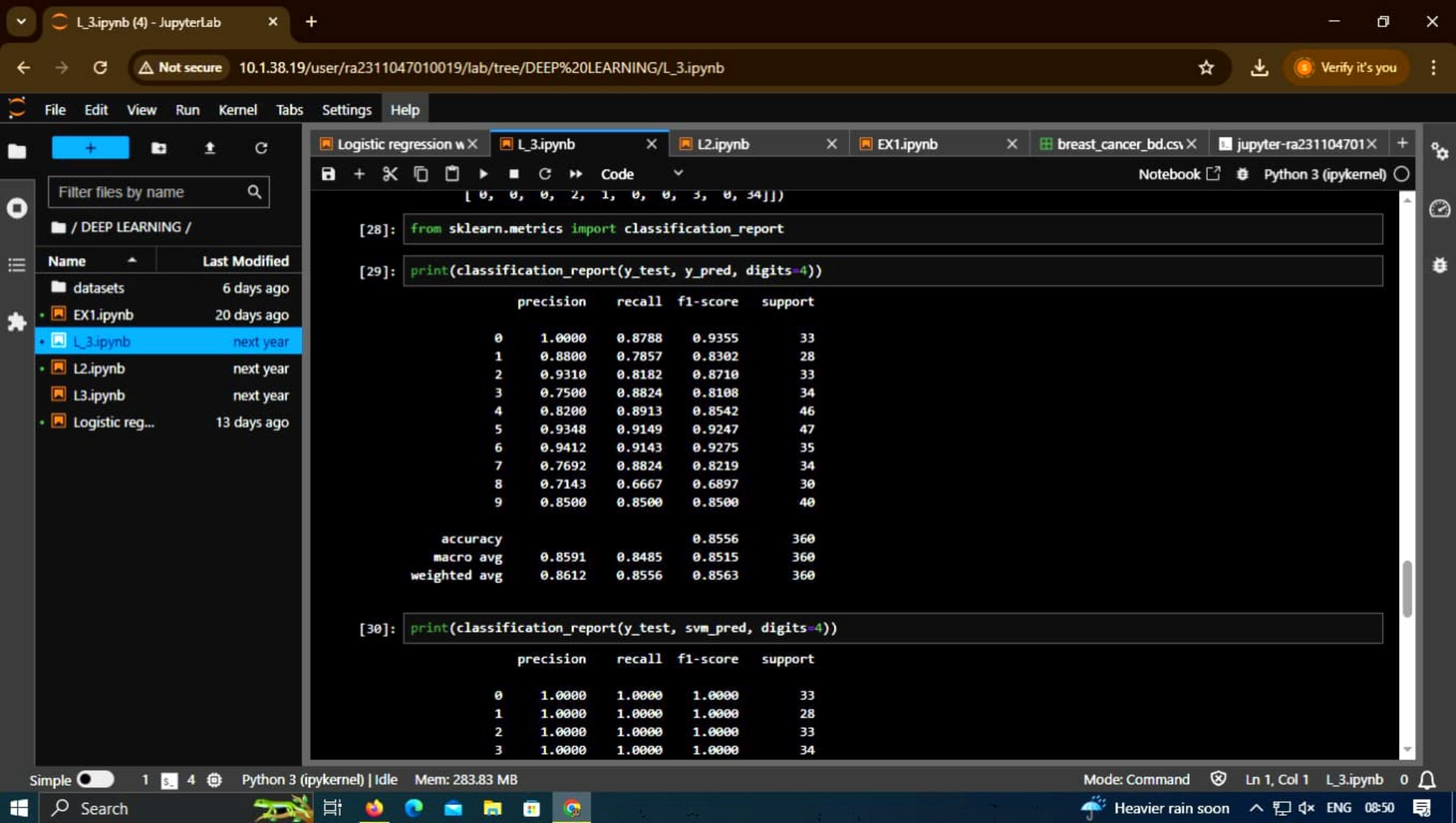
```
[25]: from sklearn import metrics
```

```
[26]: confusion_matrix=metrics.confusion_matrix(y_test,y_pred)
      confusion_matrix
```

```
[26]: array([[29,  0,  0,  0,  2,  1,  0,  0,  0,  1],
             [ 0, 22,  1,  1,  0,  0,  1,  1,  1,  1],
             [ 0,  1, 27,  2,  0,  0,  0,  1,  2,  0],
             [ 0,  0,  1, 30,  0,  0,  0,  0,  3,  0],
             [ 0,  0,  0,  1, 41,  0,  0,  4,  0,  0],
             [ 0,  0,  0,  0,  1, 43,  1,  0,  1,  1],
             [ 0,  0,  0,  0,  2,  0, 32,  0,  1,  0],
             [ 0,  0,  0,  2,  2,  0,  0, 30,  0,  0],
             [ 0,  2,  0,  2,  1,  2,  0,  0, 20,  3],
             [ 0,  0,  0,  2,  1,  0,  0,  3,  0, 34]])
```

```
[27]: confusion_matrix=metrics.confusion_matrix(y_test,y_pred)
      confusion_matrix
```

```
[27]: array([[29,  0,  0,  0,  2,  1,  0,  0,  0,  1],
             [ 0, 22,  1,  1,  0,  0,  1,  1,  1,  1],
             [ 0,  1, 27,  2,  0,  0,  0,  1,  2,  0],
             [ 0,  0,  1, 30,  0,  0,  0,  0,  3,  0],
             [ 0,  0,  0,  1, 41,  0,  0,  4,  0,  0],
             [ 0,  0,  0,  0,  1, 43,  1,  0,  1,  1],
             [ 0,  0,  0,  0,  2,  0, 32,  0,  1,  0],
             [ 0,  0,  0,  2,  2,  0,  0, 30,  0,  0],
             [ 0,  2,  0,  2,  1,  2,  0,  0, 20,  3],
```



```
 [ 0,  0,  0,  2,  1,  0,  0,  3,  0, 34]])
```

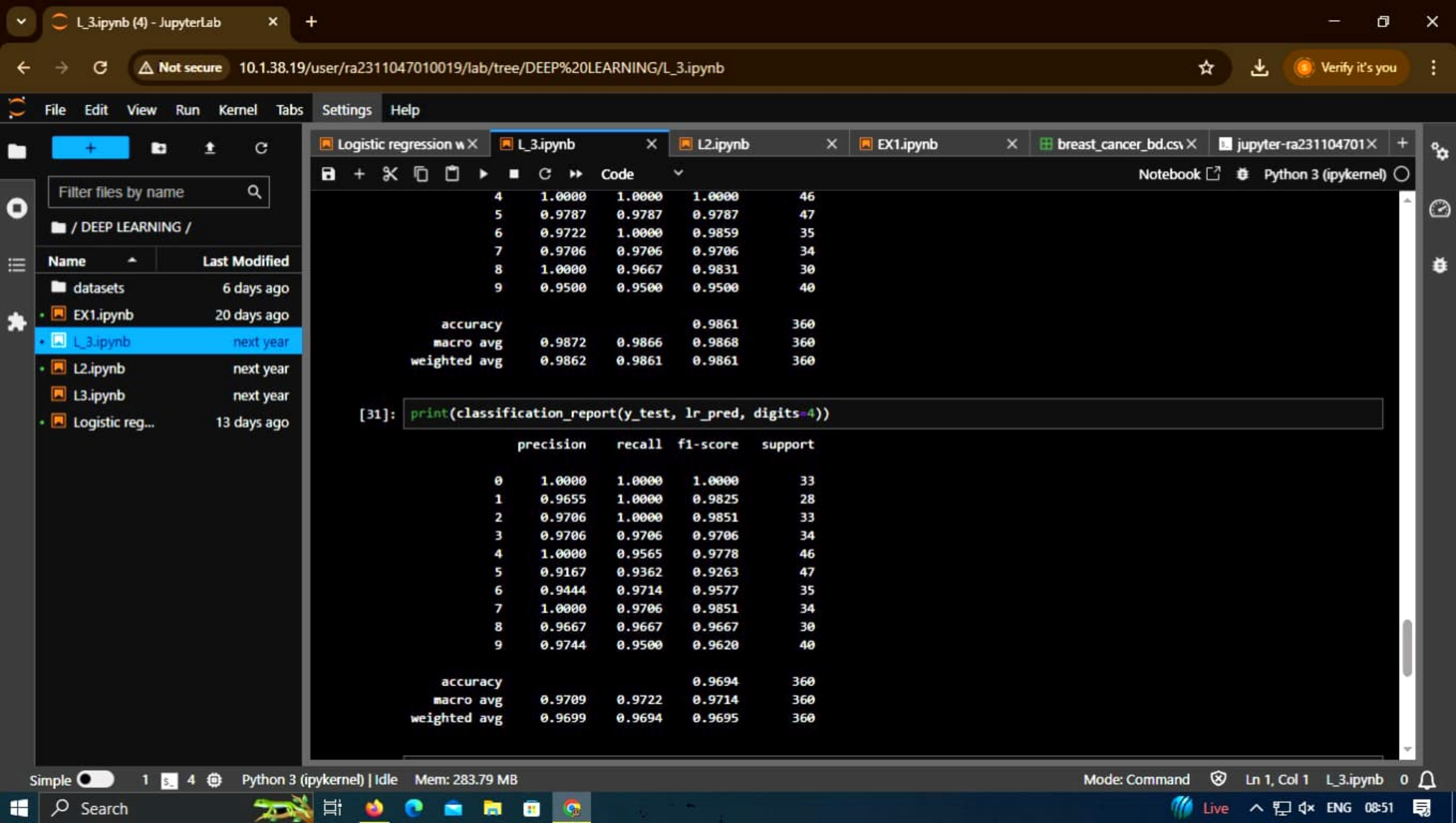
```
[28]: from sklearn.metrics import classification_report
```

```
[29]: print(classification_report(y_test, y_pred, digits=4))
```

```
              precision    recall  f1-score   support

           0     1.0000    0.8788    0.9355        33
           1     0.8800    0.7857    0.8302        28
           2     0.9310    0.8182    0.8710        33
           3     0.7500    0.8824    0.8108        34
           4     0.8200    0.8913    0.8542        46
           5     0.9348    0.9149    0.9247        47
           6     0.9412    0.9143    0.9275        35
           7     0.7692    0.8824    0.8219        34
           8     0.7143    0.6667    0.6897        30
           9     0.8500    0.8500    0.8500        40

    accuracy                         0.8556       360
   macro avg     0.8591    0.8485    0.8515       360
weighted avg     0.8612    0.8556    0.8563       360
```

```
[30]: print(classification_report(y_test, svm_pred, digits=4))
```

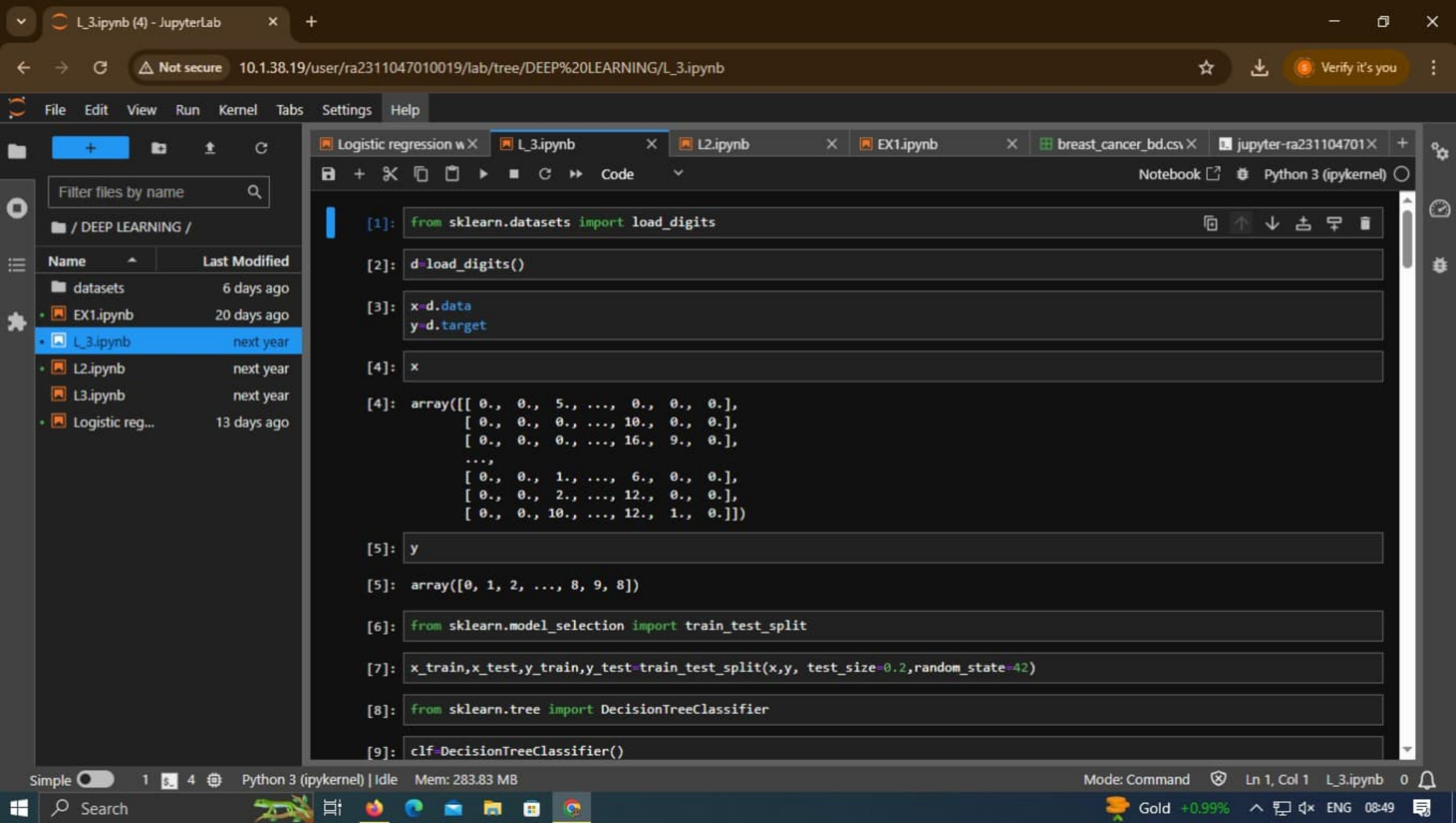
```
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000        33
           1     1.0000    1.0000    1.0000        28
           2     1.0000    1.0000    1.0000        33
           3     1.0000    1.0000    1.0000        34
```



```
           4     1.0000    1.0000    1.0000        46
           5     0.9787    0.9787    0.9787        47
           6     0.9722    1.0000    0.9859        35
           7     0.9706    0.9706    0.9706        34
           8     1.0000    0.9667    0.9831        30
           9     0.9500    0.9500    0.9500        40

    accuracy                         0.9861       360
   macro avg     0.9872    0.9866    0.9868       360
weighted avg     0.9862    0.9861    0.9861       360
```

```
[31]: print(classification_report(y_test, lr_pred, digits=4))
```

```
              precision    recall  f1-score   support

           0     1.0000    1.0000    1.0000        33
           1     0.9655    1.0000    0.9825        28
           2     0.9706    1.0000    0.9851        33
           3     0.9706    0.9706    0.9706        34
           4     1.0000    0.9565    0.9778        46
           5     0.9167    0.9362    0.9263        47
           6     0.9444    0.9714    0.9577        35
           7     1.0000    0.9706    0.9851        34
           8     0.9667    0.9667    0.9667        30
           9     0.9744    0.9500    0.9620        40

    accuracy                         0.9694       360
   macro avg     0.9709    0.9722    0.9714       360
weighted avg     0.9699    0.9694    0.9695       360
```



Name	Last Modified
datasets	6 days ago
EX1.ipynb	20 days ago
L_3.ipynb	next year
L2.ipynb	next year
L3.ipynb	next year
Logistic reg...	13 days ago

```
[1]: from sklearn.datasets import load_digits

[2]: d=load_digits()

[3]: x=d.data
     y=d.target

[4]: x

[4]: array([[ 0.,  0.,  5., ...,  0.,  0.,  0.],
            [ 0.,  0.,  0., ..., 10.,  0.,  0.],
            [ 0.,  0.,  0., ..., 16.,  9.,  0.],
            ...,
            [ 0.,  0.,  1., ...,  6.,  0.,  0.],
            [ 0.,  0.,  2., ..., 12.,  0.,  0.],
            [ 0.,  0., 10., ..., 12.,  1.,  0.]])

[5]: y

[5]: array([0, 1, 2, ..., 8, 9, 8])

[6]: from sklearn.model_selection import train_test_split

[7]: x_train,x_test,y_train,y_test=train_test_split(x,y, test_size=0.2,random_state=42)

[8]: from sklearn.tree import DecisionTreeClassifier

[9]: clf=DecisionTreeClassifier()
```