7/8/2025 Lab·3· To study of the Classifier w.r.t Statistical Parameter

Aim : To study of the classifiers with respect to Statistical parameter

Objective :

- To implement and train classifiers (Decision Tree, SVM, Logistic Regression) on the digits dataset.
- To evaluate and compare the performance of the classifiers using statistical metrics.
- To understand how different algorithms behave in terms of classification accuracy.

Pseudocode :

```
I) Decision Tree classifier
   1. Load the digits datasets (sklearn)
   2. Split The dataset into training and
      testing sets
   3. Initialize the Decision Tree Classifier
   4. Fit the classifier on Training data
   5. Predict labels for Test data.
   6. Evaluate The model using accuracy &
      classification report.
```

2) SVM

1) Load the digits dataset
2) Split the dataset into training and testing sets
3) Initialize SVM classifier
4) Fit the Classifier into training data
5) Predict the labels using Testing data
6) Evaluate model using accuracy - score & classification report.

3) Logistic Regression

1) Load the digits dataset
2) Split the dataset into training & testing sets
3) Initialize Logistic Regression Classifier.
4) Fit the classifier on training data~~set~~.
5) Predict labels for the test data
6) Evaluate model using Accuracy.

Obsuvation

Classifier	Accuracy	Notes
Decision Tree	84.72%	Fast but slighty overfits lower generalization
SVM	98.61%	High precision & performs best
Logistic Regression	97.50%	Very High accuracy good generalization

* Decission Tree shows lower performance
* SVM achieved near-perfect classification strong fit for database
* LR also very accurate and competitive with SVM

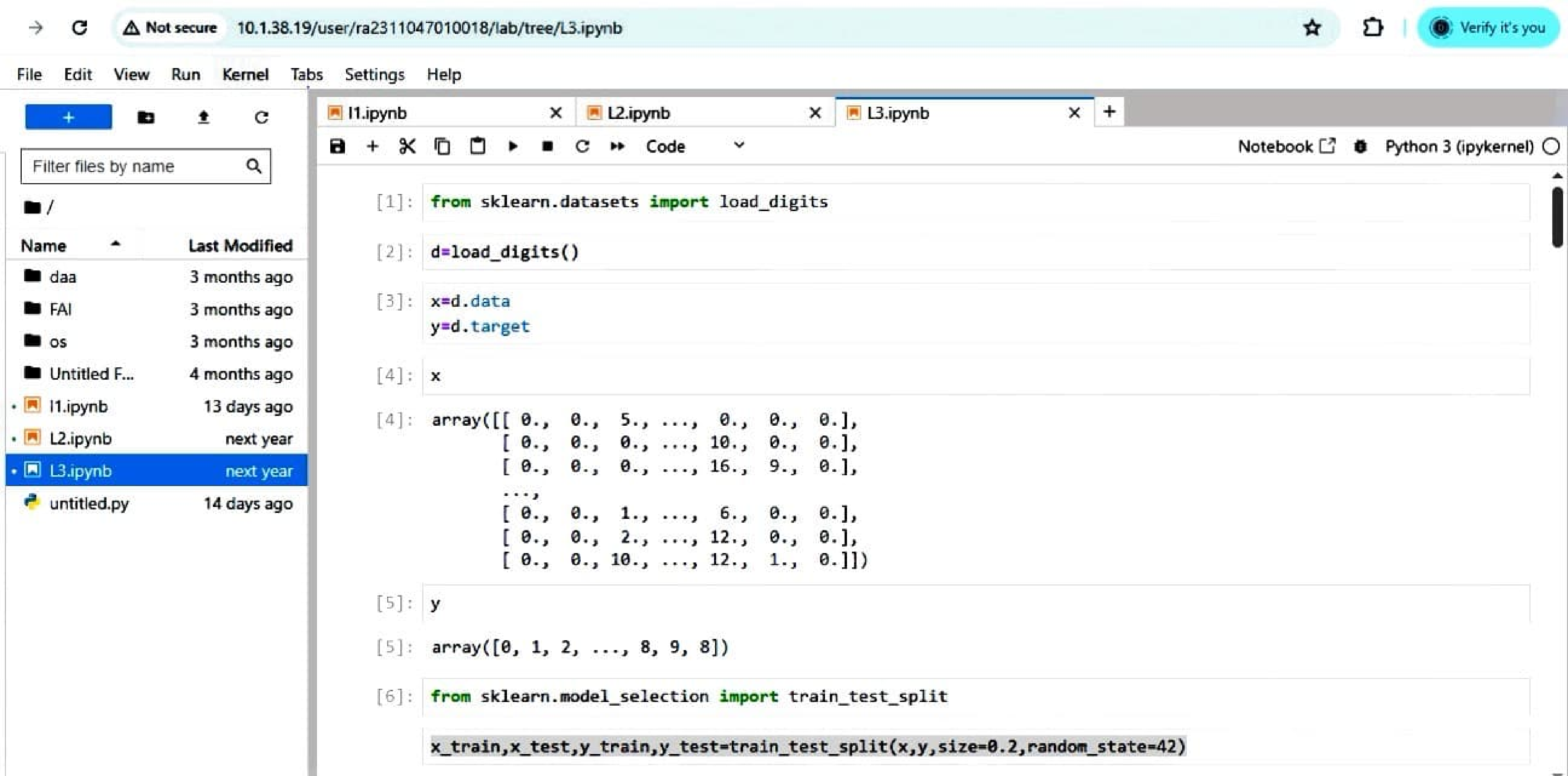
Classification Report

* Decision Tree
  - Lower Precision and recall for some classes
  - most confusion arises in predicting digits 3, 8, 9
* SVM
  - Achieves perfect (1.0000) precision
  - Very High consistency
* Logistic Regression
  - Near-perfect precision and recall for most digits
  - slight drop in F1-score

[illegible]

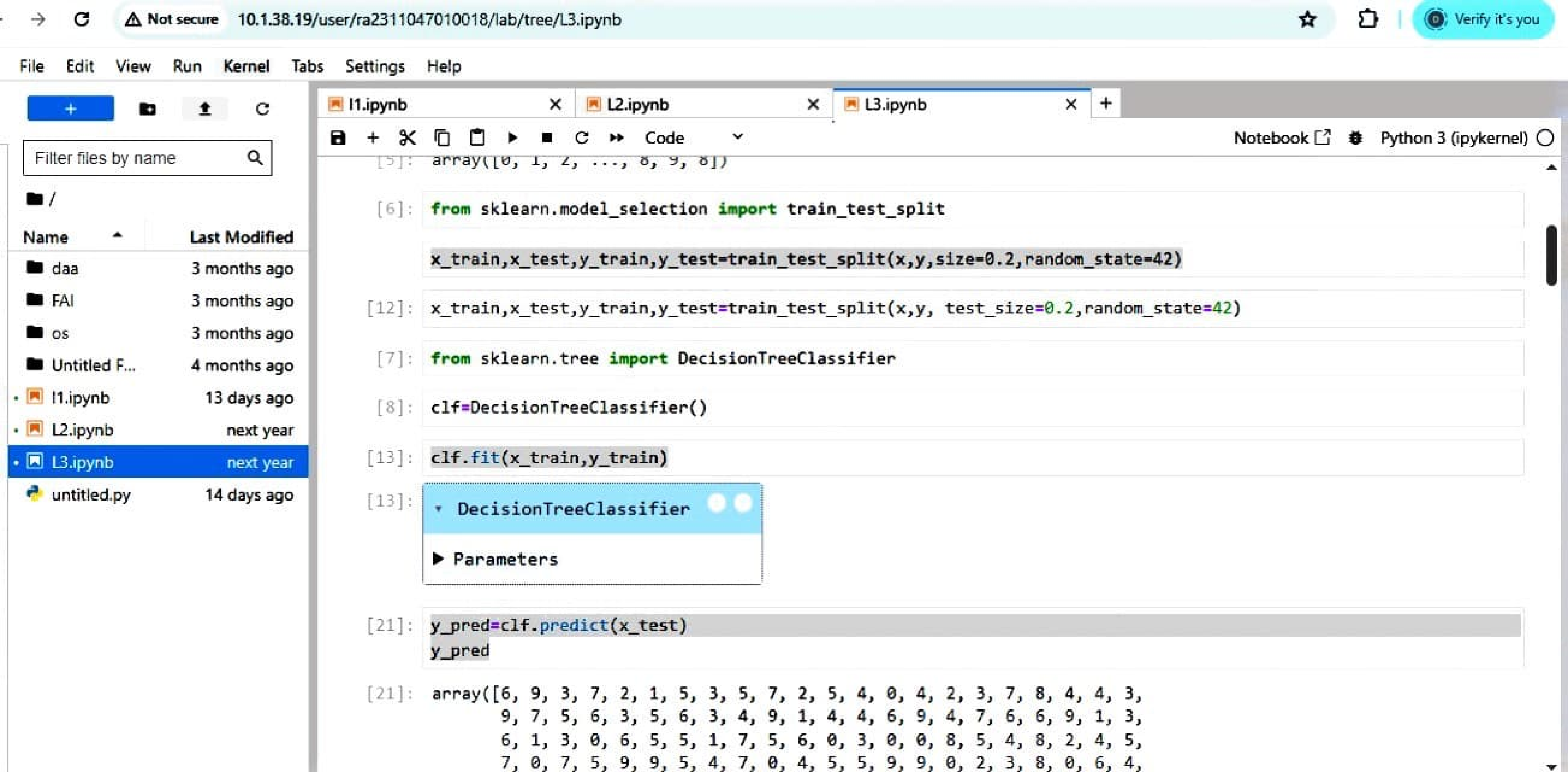
Result

Implemented the classifiers with respect to statistical parameter.

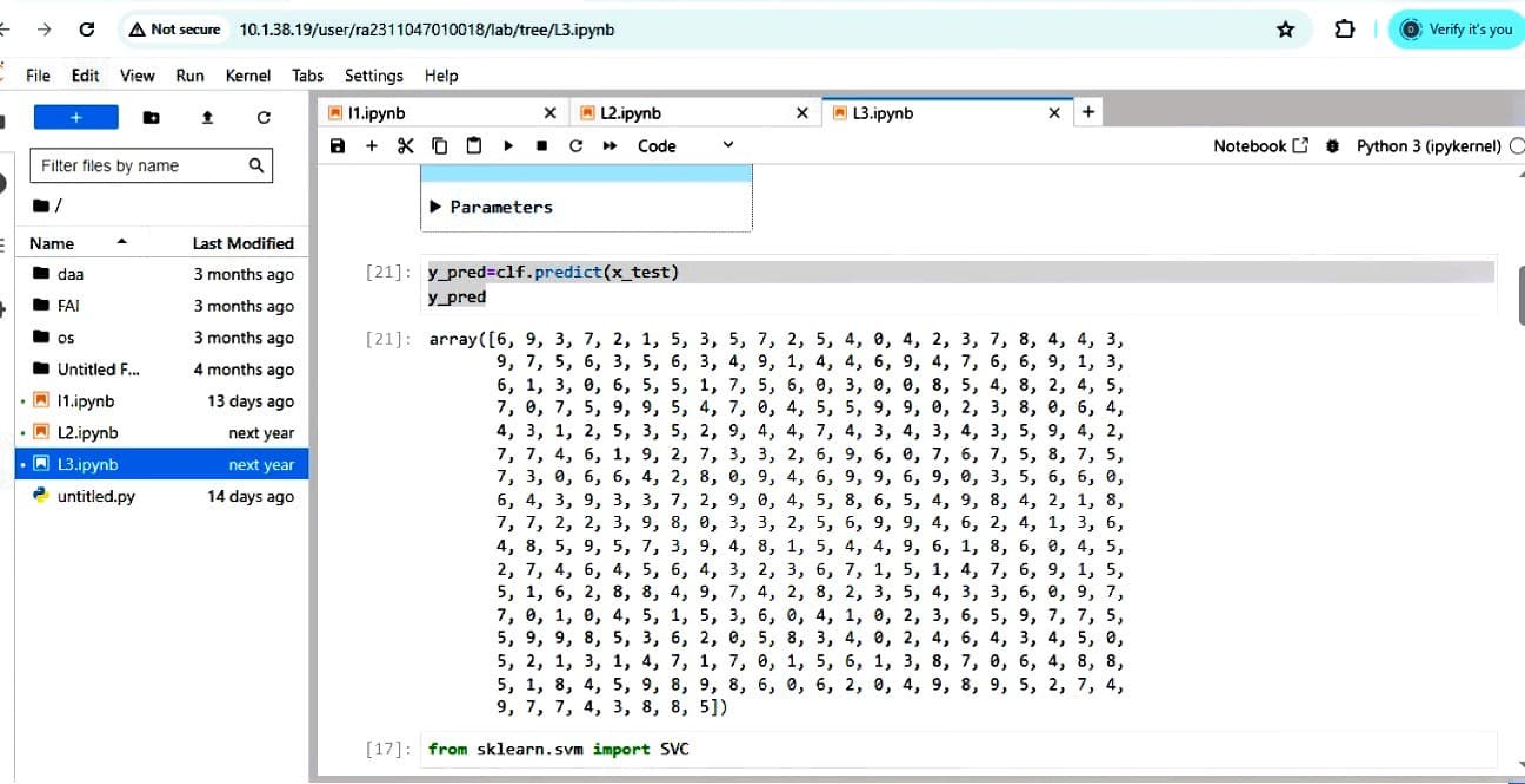


Name	Last Modified
daa	3 months ago
FAI	3 months ago
os	3 months ago
Untitled F...	4 months ago
l1.ipynb	13 days ago
L2.ipynb	next year
L3.ipynb	next year
untitled.py	14 days ago

```
[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

     x_train,x_test,y_train,y_test=train_test_split(x,y,size=0.2,random_state=42)
```



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

[6]:  from sklearn.model_selection import train_test_split

      x_train,x_test,y_train,y_test=train_test_split(x,y,size=0.2,random_state=42)

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

[7]:  from sklearn.tree import DecisionTreeClassifier

[8]:  clf=DecisionTreeClassifier()

[13]: clf.fit(x_train,y_train)

[13]: ▾ DecisionTreeClassifier
      ▶ Parameters

[21]: y_pred=clf.predict(x_test)
      y_pred

[21]: array([6, 9, 3, 7, 2, 1, 5, 3, 5, 7, 2, 5, 4, 0, 4, 2, 3, 7, 8, 4, 4, 3,
             9, 7, 5, 6, 3, 5, 6, 3, 4, 9, 1, 4, 4, 6, 9, 4, 7, 6, 6, 9, 1, 3,
             6, 1, 3, 0, 6, 5, 5, 1, 7, 5, 6, 0, 3, 0, 0, 8, 5, 4, 8, 2, 4, 5,
             7, 0, 7, 5, 9, 9, 5, 4, 7, 0, 4, 5, 5, 9, 9, 0, 2, 3, 8, 0, 6, 4,
```

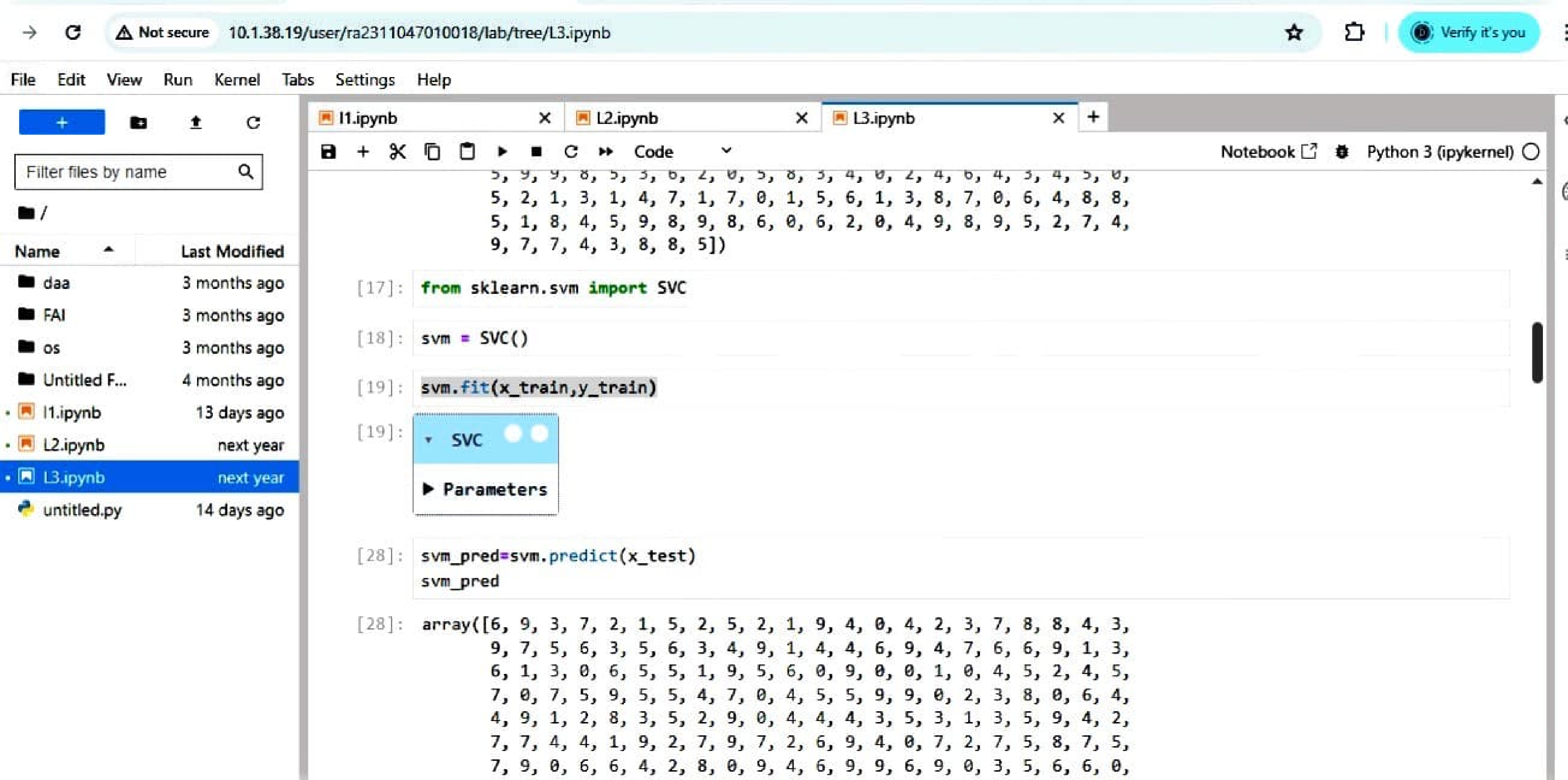


▶ Parameters

```
[21]: y_pred=clf.predict(x_test)
      y_pred
```

```
[21]: array([6, 9, 3, 7, 2, 1, 5, 3, 5, 7, 2, 5, 4, 0, 4, 2, 3, 7, 8, 4, 4, 3,
             9, 7, 5, 6, 3, 5, 6, 3, 4, 9, 1, 4, 4, 6, 9, 4, 7, 6, 6, 9, 1, 3,
             6, 1, 3, 0, 6, 5, 5, 1, 7, 5, 6, 0, 3, 0, 0, 8, 5, 4, 8, 2, 4, 5,
             7, 0, 7, 5, 9, 9, 5, 4, 7, 0, 4, 5, 5, 9, 9, 0, 2, 3, 8, 0, 6, 4,
             4, 3, 1, 2, 5, 3, 5, 2, 9, 4, 4, 7, 4, 3, 4, 3, 4, 3, 5, 9, 4, 2,
             7, 7, 4, 6, 1, 9, 2, 7, 3, 3, 2, 6, 9, 6, 0, 7, 6, 7, 5, 8, 7, 5,
             7, 3, 0, 6, 6, 4, 2, 8, 0, 9, 4, 6, 9, 9, 6, 9, 0, 3, 5, 6, 6, 0,
             6, 4, 3, 9, 3, 3, 7, 2, 9, 0, 4, 5, 8, 6, 5, 4, 9, 8, 4, 2, 1, 8,
             7, 7, 2, 2, 3, 9, 8, 0, 3, 3, 2, 5, 6, 9, 9, 4, 6, 2, 4, 1, 3, 6,
             4, 8, 5, 9, 5, 7, 3, 9, 4, 8, 1, 5, 4, 4, 9, 6, 1, 8, 6, 0, 4, 5,
             2, 7, 4, 6, 4, 5, 6, 4, 3, 2, 3, 6, 7, 1, 5, 1, 4, 7, 6, 9, 1, 5,
             5, 1, 6, 2, 8, 8, 4, 9, 7, 4, 2, 8, 2, 3, 5, 4, 3, 3, 6, 0, 9, 7,
             7, 0, 1, 0, 4, 5, 1, 5, 3, 6, 0, 4, 1, 0, 2, 3, 6, 5, 9, 7, 7, 5,
             5, 9, 9, 8, 5, 3, 6, 2, 0, 5, 8, 3, 4, 0, 2, 4, 6, 4, 3, 4, 5, 0,
             5, 2, 1, 3, 1, 4, 7, 1, 7, 0, 1, 5, 6, 1, 3, 8, 7, 0, 6, 4, 8, 8,
             5, 1, 8, 4, 5, 9, 8, 9, 8, 6, 0, 6, 2, 0, 4, 9, 8, 9, 5, 2, 7, 4,
             9, 7, 7, 4, 3, 8, 8, 5])
```

```
[17]: from sklearn.svm import SVC
```



Name	Last Modified
daa	3 months ago
FAI	3 months ago
os	3 months ago
Untitled F...	4 months ago
l1.ipynb	13 days ago
L2.ipynb	next year
L3.ipynb	next year
untitled.py	14 days ago

```
       5, 9, 9, 8, 5, 3, 6, 2, 0, 5, 8, 3, 4, 0, 2, 4, 6, 4, 3, 4, 5, 0,
       5, 2, 1, 3, 1, 4, 7, 1, 7, 0, 1, 5, 6, 1, 3, 8, 7, 0, 6, 4, 8, 8,
       5, 1, 8, 4, 5, 9, 8, 9, 8, 6, 0, 6, 2, 0, 4, 9, 8, 9, 5, 2, 7, 4,
       9, 7, 7, 4, 3, 8, 8, 5])
```

```
[17]: from sklearn.svm import SVC
```

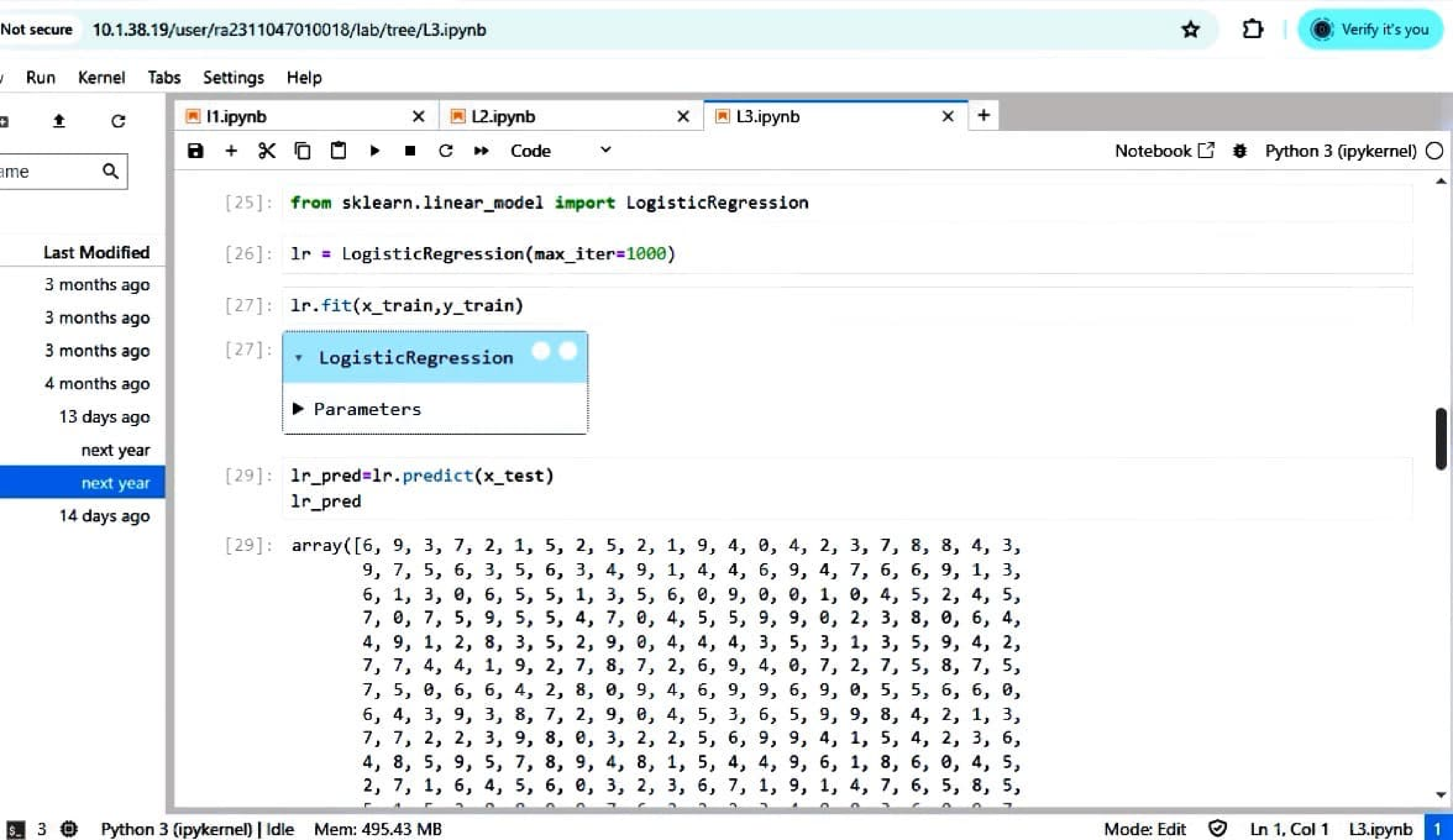
```
[18]: svm = SVC()
```

```
[19]: svm.fit(x_train,y_train)
```

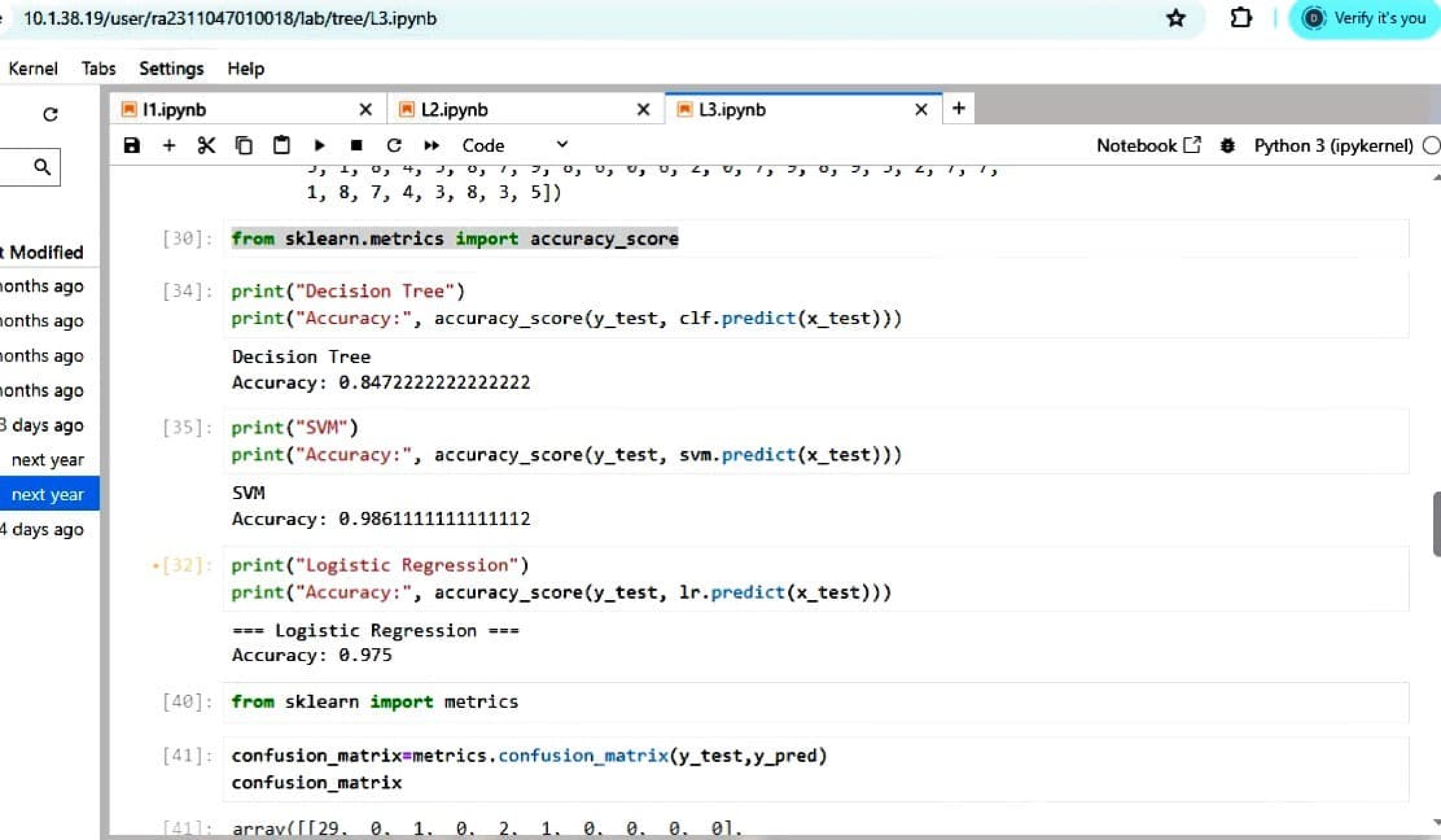
```
[19]: ▾ SVC
      ▶ Parameters
```

```
[28]: svm_pred=svm.predict(x_test)
      svm_pred
```

```
[28]: array([6, 9, 3, 7, 2, 1, 5, 2, 5, 2, 1, 9, 4, 0, 4, 2, 3, 7, 8, 8, 4, 3,
             9, 7, 5, 6, 3, 5, 6, 3, 4, 9, 1, 4, 4, 6, 9, 4, 7, 6, 6, 9, 1, 3,
             6, 1, 3, 0, 6, 5, 5, 1, 9, 5, 6, 0, 9, 0, 0, 1, 0, 4, 5, 2, 4, 5,
             7, 0, 7, 5, 9, 5, 5, 4, 7, 0, 4, 5, 5, 9, 9, 0, 2, 3, 8, 0, 6, 4,
             4, 9, 1, 2, 8, 3, 5, 2, 9, 0, 4, 4, 4, 3, 5, 3, 1, 3, 5, 9, 4, 2,
             7, 7, 4, 4, 1, 9, 2, 7, 9, 7, 2, 6, 9, 4, 0, 7, 2, 7, 5, 8, 7, 5,
             7, 9, 0, 6, 6, 4, 2, 8, 0, 9, 4, 6, 9, 9, 6, 9, 0, 3, 5, 6, 6, 0,
```



```
[25]: from sklearn.linear_model import LogisticRegression

[26]: lr = LogisticRegression(max_iter=1000)

[27]: lr.fit(x_train,y_train)

[27]: ▾ LogisticRegression
      ▶ Parameters

[29]: lr_pred=lr.predict(x_test)
      lr_pred

[29]: array([6, 9, 3, 7, 2, 1, 5, 2, 5, 2, 1, 9, 4, 0, 4, 2, 3, 7, 8, 8, 4, 3,
             9, 7, 5, 6, 3, 5, 6, 3, 4, 9, 1, 4, 4, 6, 9, 4, 7, 6, 6, 9, 1, 3,
             6, 1, 3, 0, 6, 5, 5, 1, 3, 5, 6, 0, 9, 0, 0, 1, 0, 4, 5, 2, 4, 5,
             7, 0, 7, 5, 9, 5, 5, 4, 7, 0, 4, 5, 5, 9, 9, 0, 2, 3, 8, 0, 6, 4,
             4, 9, 1, 2, 8, 3, 5, 2, 9, 0, 4, 4, 4, 3, 5, 3, 1, 3, 5, 9, 4, 2,
             7, 7, 4, 4, 1, 9, 2, 7, 8, 7, 2, 6, 9, 4, 0, 7, 2, 7, 5, 8, 7, 5,
             7, 5, 0, 6, 6, 4, 2, 8, 0, 9, 4, 6, 9, 9, 6, 9, 0, 5, 5, 6, 6, 0,
             6, 4, 3, 9, 3, 8, 7, 2, 9, 0, 4, 5, 3, 6, 5, 9, 9, 8, 4, 2, 1, 3,
             7, 7, 2, 2, 3, 9, 8, 0, 3, 2, 2, 5, 6, 9, 9, 4, 1, 5, 4, 2, 3, 6,
             4, 8, 5, 9, 5, 7, 8, 9, 4, 8, 1, 5, 4, 4, 9, 6, 1, 8, 6, 0, 4, 5,
             2, 7, 1, 6, 4, 5, 6, 0, 3, 2, 3, 6, 7, 1, 9, 1, 4, 7, 6, 5, 8, 5,
```



```
        1, 8, 7, 4, 3, 8, 3, 5])
```

```
[30]: from sklearn.metrics import accuracy_score
```

```
[34]: print("Decision Tree")
      print("Accuracy:", accuracy_score(y_test, clf.predict(x_test)))

      Decision Tree
      Accuracy: 0.8472222222222222
```

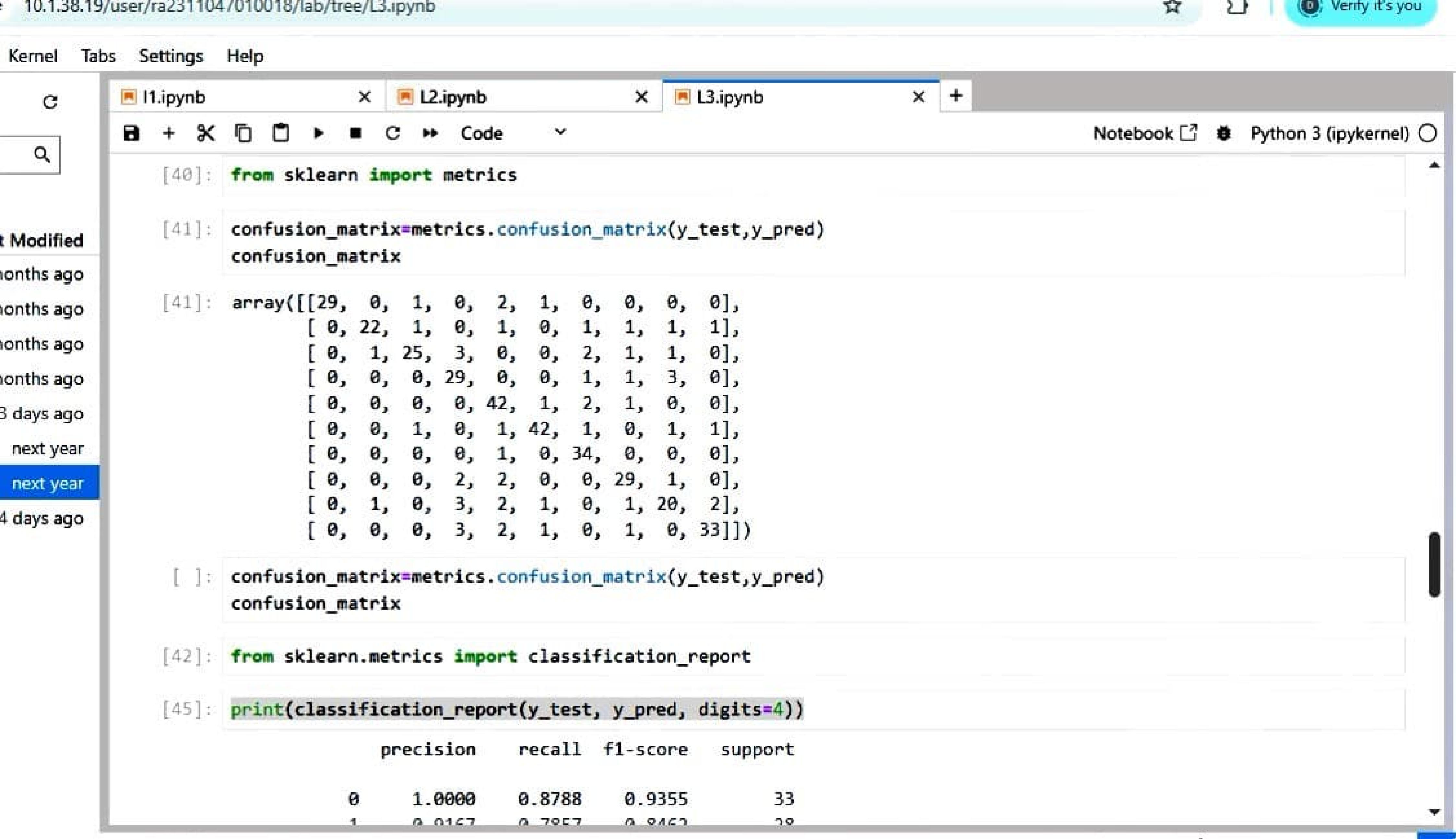
```
[35]: print("SVM")
      print("Accuracy:", accuracy_score(y_test, svm.predict(x_test)))

      SVM
      Accuracy: 0.9861111111111112
```

```
[32]: print("Logistic Regression")
      print("Accuracy:", accuracy_score(y_test, lr.predict(x_test)))

      === Logistic Regression ===
      Accuracy: 0.975
```

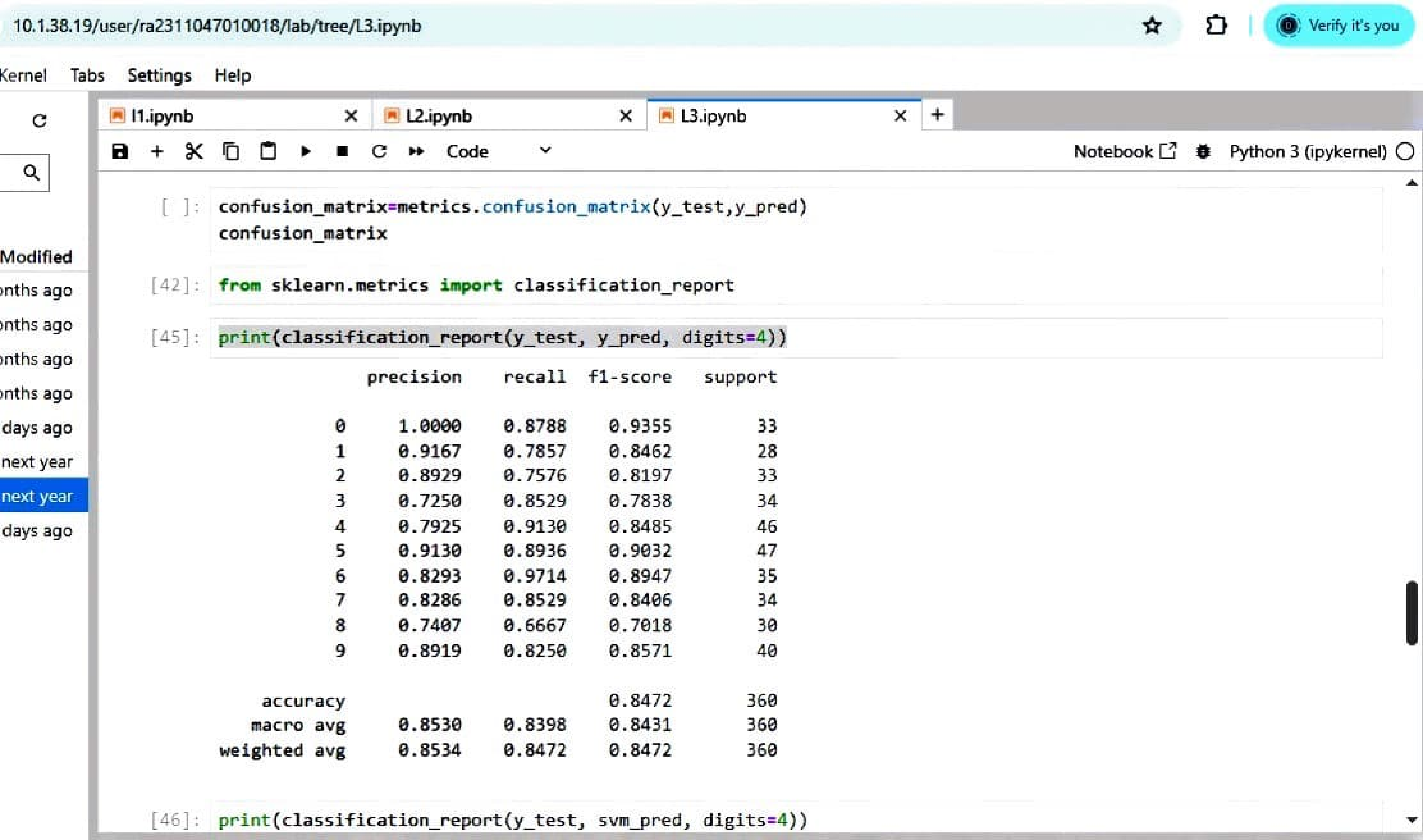
```
[40]: from sklearn import metrics
```

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

```
[41]: array([[29,  0,  1,  0,  2,  1,  0,  0,  0,  0],
```



```
[40]: from sklearn import metrics

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

[41]: array([[29,  0,  1,  0,  2,  1,  0,  0,  0,  0],
             [ 0, 22,  1,  0,  1,  0,  1,  1,  1,  1],
             [ 0,  1, 25,  3,  0,  0,  2,  1,  1,  0],
             [ 0,  0,  0, 29,  0,  0,  1,  1,  3,  0],
             [ 0,  0,  0,  0, 42,  1,  2,  1,  0,  0],
             [ 0,  0,  1,  0,  1, 42,  1,  0,  1,  1],
             [ 0,  0,  0,  0,  1,  0, 34,  0,  0,  0],
             [ 0,  0,  0,  2,  2,  0,  0, 29,  1,  0],
             [ 0,  1,  0,  3,  2,  1,  0,  1, 20,  2],
             [ 0,  0,  0,  3,  2,  1,  0,  1,  0, 33]])

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

[42]: from sklearn.metrics import classification_report

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

              precision    recall  f1-score   support

           0     1.0000    0.8788    0.9355        33
           1     0.9167    0.7857    0.8462        28
```



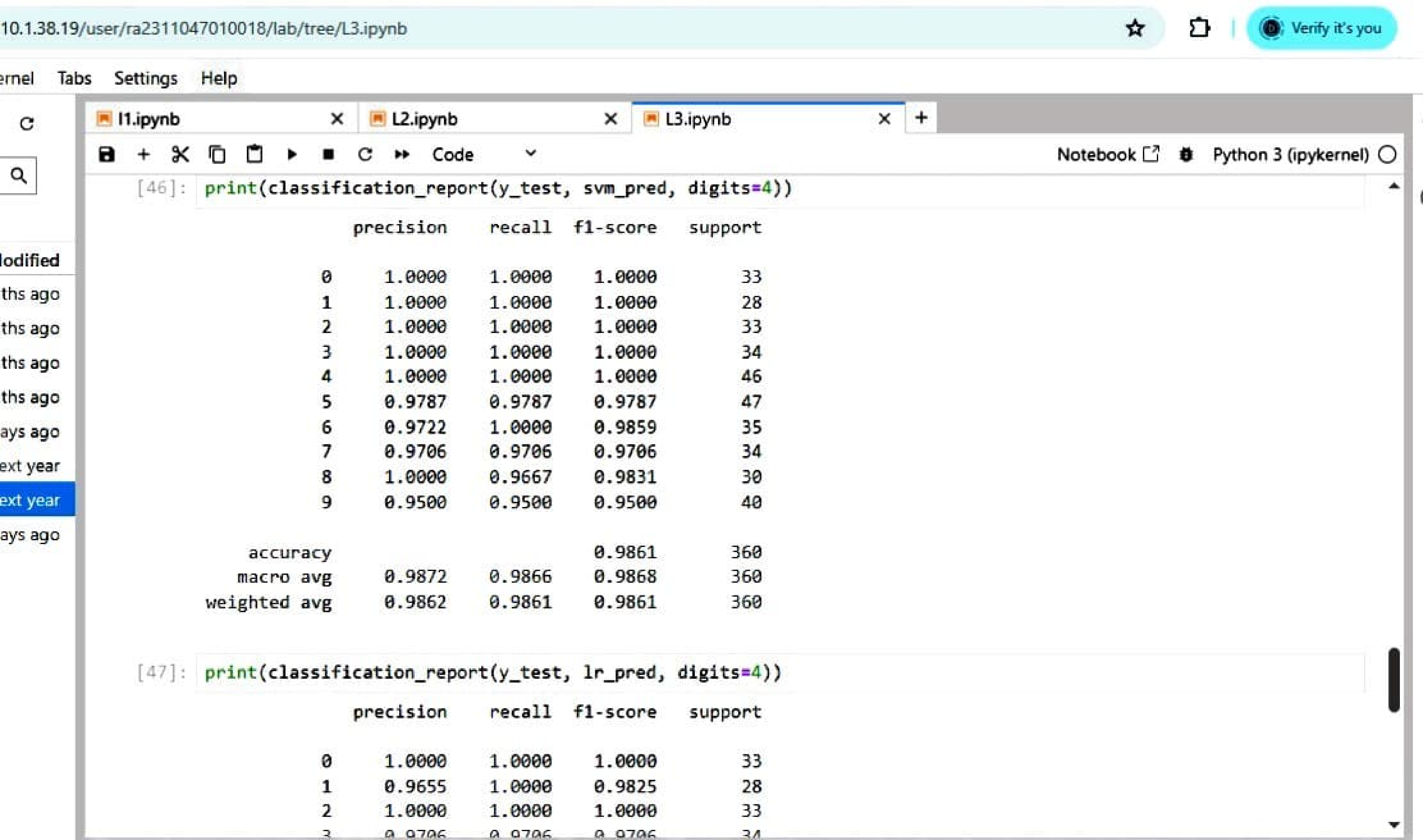
```
[ ]: confusion_matrix=metrics.confusion_matrix(y_test,y_pred)
     confusion_matrix
```

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

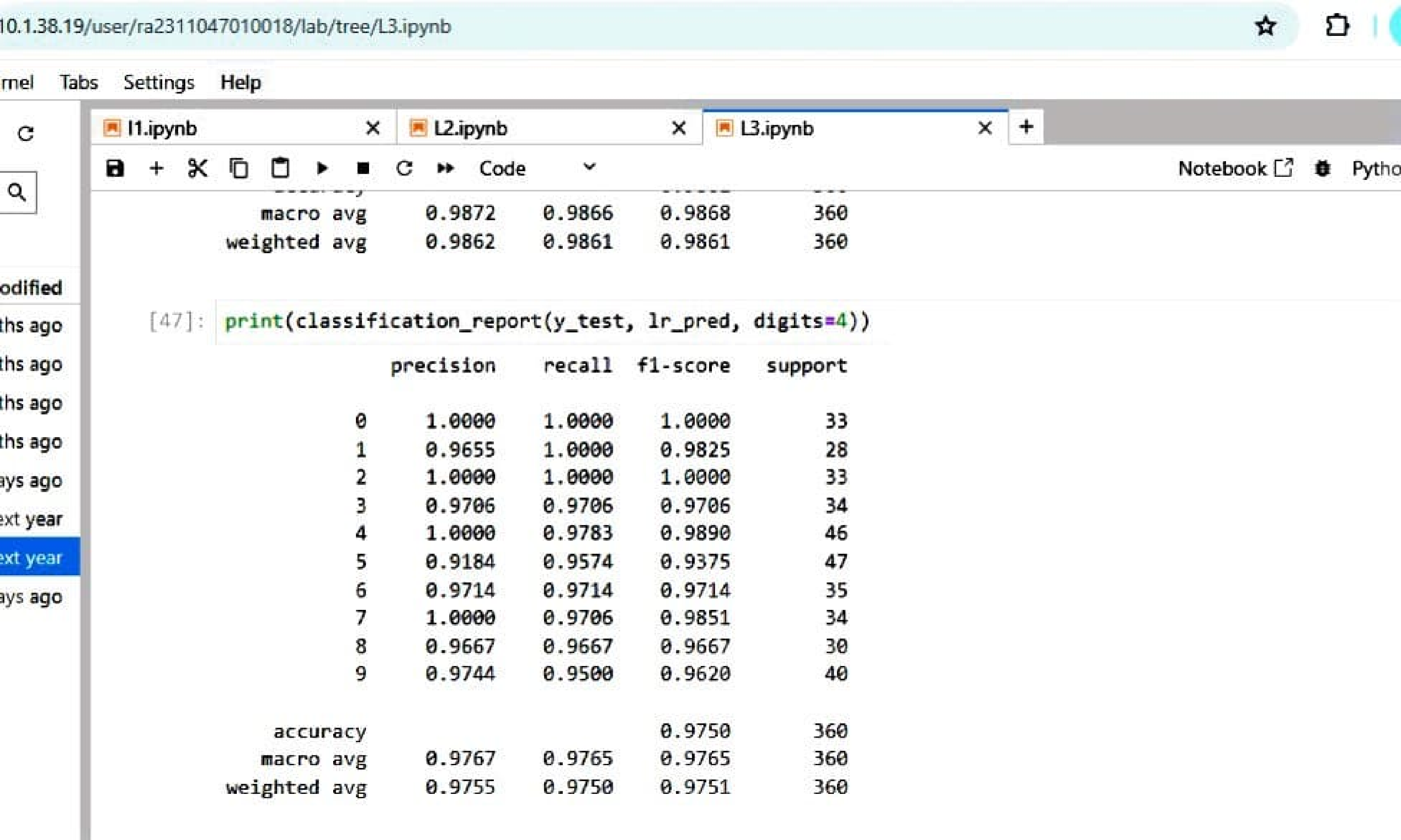
```
[45]: print(classification_report(y_test, y_pred, digits=4))
```

```
              precision    recall  f1-score   support

           0     1.0000    0.8788    0.9355        33
           1     0.9167    0.7857    0.8462        28
           2     0.8929    0.7576    0.8197        33
           3     0.7250    0.8529    0.7838        34
           4     0.7925    0.9130    0.8485        46
           5     0.9130    0.8936    0.9032        47
           6     0.8293    0.9714    0.8947        35
           7     0.8286    0.8529    0.8406        34
           8     0.7407    0.6667    0.7018        30
           9     0.8919    0.8250    0.8571        40

    accuracy                         0.8472       360
   macro avg     0.8530    0.8398    0.8431       360
weighted avg     0.8534    0.8472    0.8472       360
```

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



```
[46]: 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


[47]: 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     1.0000    1.0000    1.0000        33
           3     0.9706    0.9706    0.9706        34
```



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

```
[47]: 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     1.0000    1.0000    1.0000        33
           3     0.9706    0.9706    0.9706        34
           4     1.0000    0.9783    0.9890        46
           5     0.9184    0.9574    0.9375        47
           6     0.9714    0.9714    0.9714        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                         0.9750       360
   macro avg     0.9767    0.9765    0.9765       360
weighted avg     0.9755    0.9750    0.9751       360
```