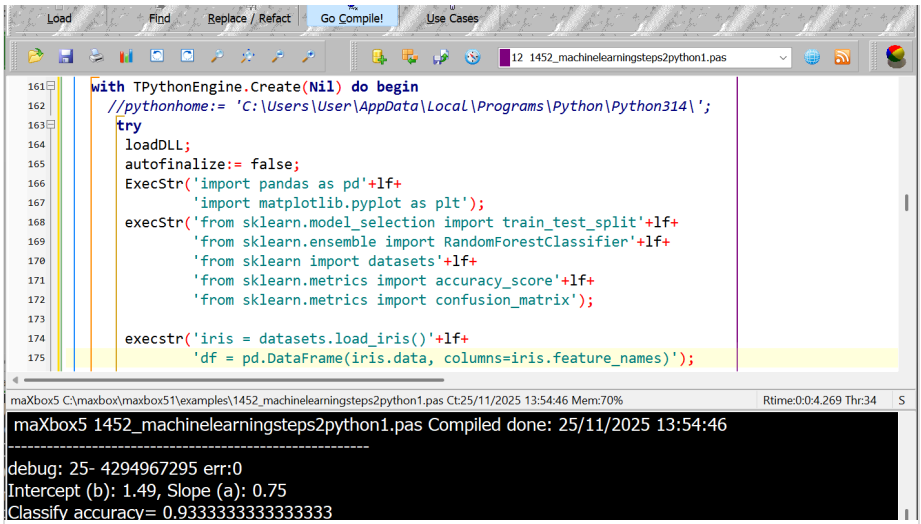


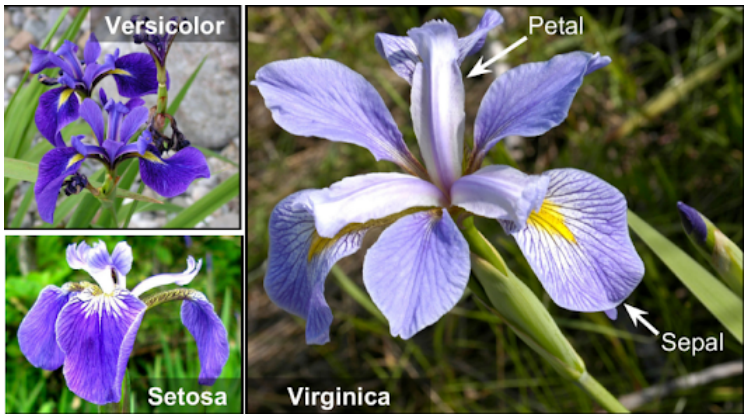
Machinelearning Steps

November 25, 2025

This article will provide the clear cut understanding of Iris dataset and how to do classification on Iris flowers dataset using python, seaborn and sklearn.

The Iris dataset is often used in machine learning and data science courses, because it's simple to understand and well-defined, yet interesting enough to present real challenges to new learners. This single code snippet will use Python with Python4Delphi to classify the Iris dataset into one of three flower species: Setosa, Versicolor, or Virginica.

The iris data consisted of 150 samples of three species of Iris. The first column represented sepal length, the second column represented sepal width, the third column represented petal length, and the fourth column represented petal width. I'm going to use sci-kit-learn to classify these instances according to their species.



The provided Delphi code snippet uses the Python4Delphi library, specifically the TPythonEngine component, to embed Python code execution into a Delphi application. It creates an instance of TPythonEngine and uses it to run a sequence of Python commands that implement a machine learning workflow with the iris dataset using popular Python libraries like pandas, matplotlib, pyplot, scikit-learn (sklearn), and seaborn.

Import Libraries

First we load the libraries:

```
with TPythonEngine.Create(Nil) do begin
     //pythonhome:= 'C:\Users\User\AppData\Local\Programs\Pyt
  try
   loadDLL;
   autofinalize:= false;
   ExecStr('import pandas as pd'+lf+
        'import matplotlib.pyplot as plt');
   execStr('from sklearn.model_selection import train_test_sp
        'from sklearn.ensemble import RandomForestClassifier'
        'from sklearn import datasets'+lf+
        'from sklearn.metrics import accuracy_score'+lf+
        'from sklearn.metrics import confusion_matrix');
```

Load Dataset

But the data are not well structured for us to understand. So we need to convert it into a pandas DataFrame. Pandas is a great tool for doing all sorts of things related to datasets, including preprocessing and exploring them.

```
execstr('iris = datasets.load_iris()'+lf+
        'df = pd.DataFrame(iris.data, columns=iris.feature_nam
```

- It loads the iris dataset from sklearn.datasets and converts it into a pandas DataFrame.
- The data is split into training and test sets with stratification and fixed random seed as random_state for reproducibility and comparision.

Classify the species

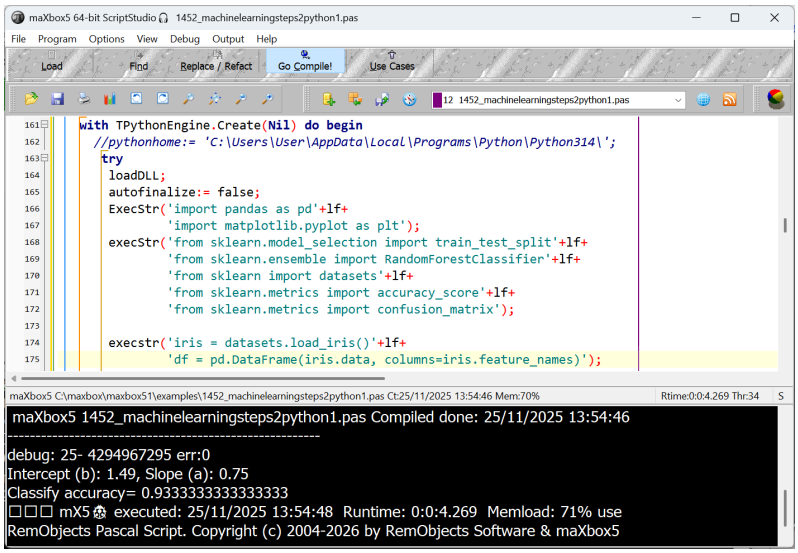
So we don't need to hardcode the algorithm for classification since there are already many algorithms available in the sci-kit learn package. We can simply choose any of them and use them. Here, I am going to use the Random Forest Classifier model. Now, after training our model on training data, we can predict petal and sepal measurements on testing data as the features. And that's the non binary cassification for the first block!

```
execstr('X_train,X_test, y_train,y_test = train_test_split(df[
        'iris.target,test_size=0.5, stratify=iris.target, rand
execstr('rf = RandomForestClassifier(n_estimators=100, oob_sco
                                     'random_state=123456)'+lf

writeln('Classify accuracy= '+Evalstr('accuracy_score(y_test,
```

Also, we can use the train_test_split function in sklearn in order to split the dataset into train and test.

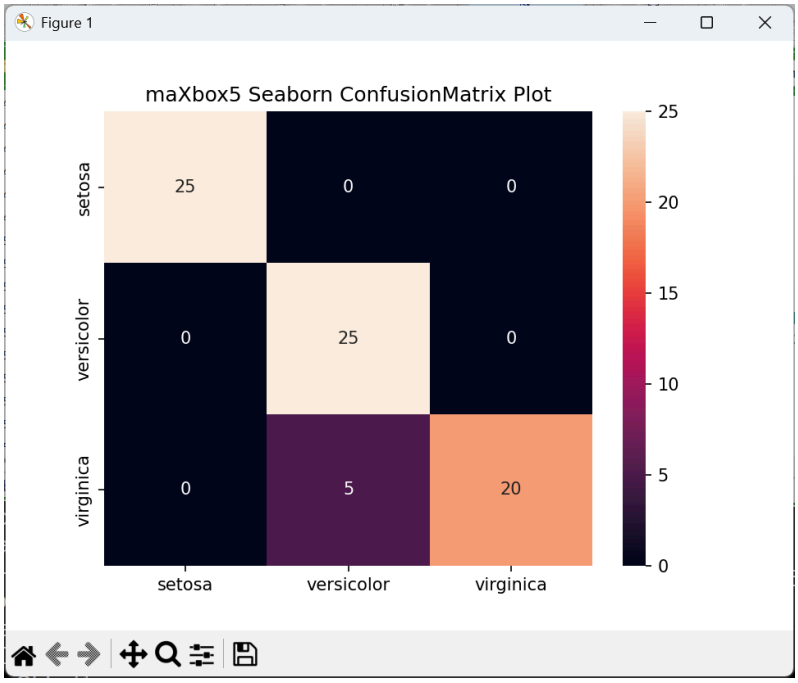
We get Classify accuracy= 0.9333333333333333



https://sourceforge.net/projects/maxbox5/files/examples/1452_machinelearningsteps2python1.pas/download

Plotting the Performance Measures

Performance measures are used to evaluate effectiveness of classifiers on different datasets with different characteristics. For classification problems, there are three main measures for evaluating the model, the precision(the accuracy of positive predictions or the number of most relevant values from retrieved values.), Recall(ratio of positive instances that are truly detected by the classifier), and confusion matrix.



Confusion Matrix for Performance Measure

The scores are pretty good in this case only 5 misclassified. The main usage of the confusion matrix is to identify how many of the classes are misclassified by the classifier (false-positive or false-negative).

With the following call you get a report of the performance:

execstr('print(metrics.classification_report(y_test, rf.predict(X_test), digits=3))');

precision recall f1-score support eg: Setosa(0), Versicolor(1), Virginica(2)

```
              precision recall    f1-score     support
         0      1.000     1.000     1.000        25
         1      0.833     1.000     0.909        25
         2      1.000     0.800     0.889        25

accuracy                           0.933        75
```

macro avg 0.944 0.933 0.933 75
weighted avg 0.944 0.933 0.933 75

We check the recall of virginica (2) its 80%, that means only 80% are recognized (recall) as virginica so 20% are misclassified as versicolor!

On the other side the precision of versicolor (1) is only 83% because 5 from virginica are misclassified as versicolor and that means 25/30= 0.8333 precision!

Conclusion and Summary

The provided Delphi code snippet uses the Python4Delphi library, specifically the TPythonEngine component, to embed Python code execution into a Delphi application. It creates an instance of TPythonEngine and uses it to run a sequence of Python commands that implement a machine learning workflow with the iris dataset using popular Python libraries like pandas, matplotlib, scikit-learn, and seaborn.

Here are the detailed steps in 9 points and purpose of the code for your tests and document:

- It initializes the Python engine component (TPythonEngine.Create(nil)) and loads the Python DLL dynamically (loadDLL) as 3.13 or 3.14.
- The code sets autofinalize to false to control Python finalization manually.
- It executes multiple Python import statements to load common data science and machine learning libraries: pandas (for data manipulation), matplotlib.pyplot (plotting), sklearn's modules including train_test_split, RandomForestClassifier, dataset loaders, and metrics like accuracy_score and confusion_matrix and It loads the iris dataset from sklearn.datasets and converts it into a pandas DataFrame.
- The data is split into training and test sets with stratification and fixed random seed for reproducibility.
- A RandomForestClassifier model is created and trained on the training data.
- The accuracy of the model on the test set is evaluated and printed using the Python accuracy_score function and a confusion matrix is generated, converted into a pandas DataFrame, and plotted using seaborn's heatmap with an annotated matrix plot.
- The plot is finalized with a title and displayed using matplotlib.
- Exception handling surrounds the block to raise any Python-related errors caught during execution.
- Finally, resources are freed explicitly by calling Free on the TPythonEngine instance.

This code showcases an integration approach where Delphi acts as a container to drive Python-based data science workflows, combining the strengths of Delphi in application development and Python in machine learning and visualization. The use of the embedded Python environment allows direct execution of Python commands from Delphi, enabling rich interaction between the two languages.

```
procedure PY_Machinelearning_steps;
 begin
   with TPythonEngine.Create(Nil) do begin
     //pythonhome:= 'C:\Users\User\AppData\Local\Programs\Pyt
    try
     loadDLL;
     autofinalize:= false;
     ExecStr('import pandas as pd'+lf+
             'import matplotlib.pyplot as plt');
     execStr('from sklearn.model_selection import train_test_
             'from sklearn.ensemble import RandomForestClassi
             'from sklearn import datasets'+lf+
             'from sklearn.metrics import accuracy_score'+lf+
             'from sklearn.metrics import confusion_matrix');

     execstr('iris = datasets.load_iris()'+lf+
             'df = pd.DataFrame(iris.data, columns=iris.featu
     execstr('X_train,X_test, y_train,y_test= train_test_spli
             'iris.target,test_size=0.5, stratify=iris.target
     execstr('rf = RandomForestClassifier(n_estimators=100, o
                                          'random_state=1234
             'rf.fit(X_train, y_train)');

     writeln('Classify accuracy= '+evalstr('accuracy_score(y_

//https://www.blopig.com/blog/2017/07/using-random-forests-in
     execstr('import seaborn as sns');
     execstr('cm = pd.DataFrame(confusion_matrix(y_test, rf.p
                     'columns=iris.target_names, index=iris.t
     execstr('sns.heatmap(cm, annot=True)'+lf+
             'plt.title("maXbox5 Seaborn ConfusionMatrix Plot
             'plt.show()');
     except
       raiseError;
     finally
       Free;
     end;
   end;
 end;
```

References:

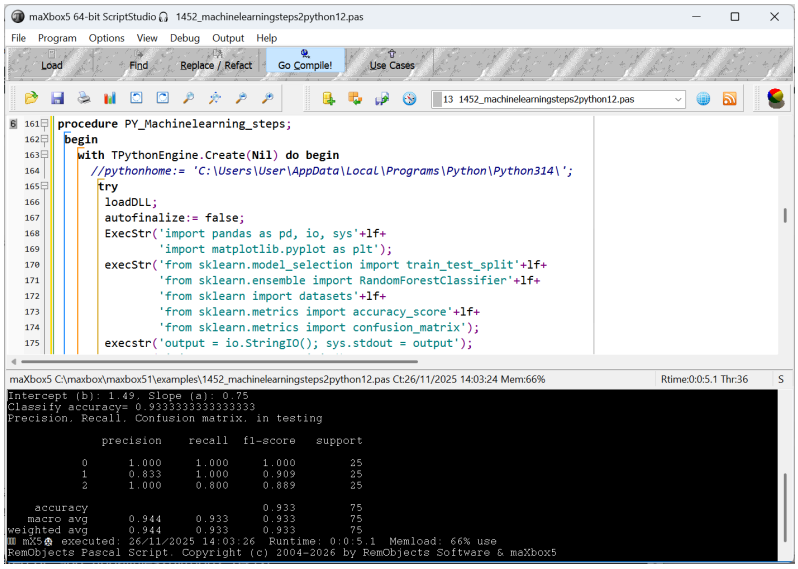
- TPythonEngine is a core component of Python4Delphi for embedding Python.
- The code sequence corresponds to classical iris dataset classification using Random Forest, a well-known machine learning example.
- maXbox5 as the scripting-studio compiler to execute code.

This code snippet can form a comprehensive description for your document highlighting how Delphi and Python4Delphi facilitate machine learning integration through Python scripts executed inside Delphi or maXbox applications. So with maXbox you we can think out of the box.

Classify 3 Locs as our next step

Classify with 4 Classes : Swiss, France, Germany and Belgian Locs

Update 1.2



Download 1452_machinelearningsteps2python12.pas (maXbox5)

Posted in Code, Data Science, Machinelearning, maXbox, Python

Leave a comment

← Statistics with Statsmodels