Making Your Scikit-Learn Model Persistent

After training a scikit-learn model, you may want to have a way to persist the model so that you can use it in the future to make predictions. There are a few methods that you can use to accomplish this. This exercise will demonstrate one of these methods using the Iris Classification KNN model.

For more information, consult scikit-learn's online documentation (https://scikit-learn.org/stable/model_persistence.html) on model persistence.

Import Packages

Before we get started, let us import a few packages. Run the code cell below.

In [1]:
```
import pandas as pd
import os
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Step 1: Load the Iris Data Set

In [2]:
```
filename = os.path.join(os.getcwd(), "data", "Iris_Data.csv")
df_iris = pd.read_csv(filename, header=0)
```

Step 2: Create Labeled Examples

In [3]:
```
X = df_iris[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y = df_iris['species']
```

Step 3: Create Training and Test Data Sets

In [4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4)
```

Step 4: Fit a KNN Model to the Training Set

In [5]:
```
# Initialize the model
model = KNeighborsClassifier(n_neighbors=3)

# Train the model using the training data sets
model.fit(X_train, y_train)
```

Out[5]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

Step 5: Save Your Model Using Pickle

One approach to make your model persistent is to use the python `pickle` module. `pickle` allows a Python object hierarchy to be converted into a byte stream, and vice versa.

The code cells below demonstrate how to use the `pickle` module.

1. Import the pickle module

In [7]:
```
import pickle
```

2. Use pickle to save your model to a file in the current working directory

In [8]:
```
pkl_model_filename = "Pickle_Iris_Classification_Model.pkl"

pickle.dump(model, open(pkl_model_filename, 'wb'))
```

3. Load your model back from the file to use in the future

In [10]:
```
persistent_model = pickle.load(open(pkl_model_filename, 'rb'))

persistent_model
```

Out[10]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
                     weights='uniform')
```

Step 6: Use Your Persistent Model to Make Predictions

In [11]:
```
prediction = persistent_model.predict(X_test)
print(prediction)
```

```
['Iris-virginica' 'Iris-setosa' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa'
 'Iris-setosa' 'Iris-virginica' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-virginica' 'Iris-setosa' 'Iris-versicolor'
 'Iris-setosa' 'Iris-setosa' 'Iris-virginica' 'Iris-setosa'
 'Iris-virginica' 'Iris-versicolor' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-virginica']
```

Considerations

There are certain things to consider when using `pickle` for model persistence. One consideration is maintainability.

Models saved using one version of scikit-learn may not behave expectedly when loaded in other versions. It is advisable to save metadata along with your model that includes information you can reference in the future, such as a reference to your training data, the Python source code that you used to build and evaluate the model, the version of scikit-learn and other packages that you used, and the results of evaluation metrics.