```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.metrics import accuracy_score
from IPython.display import Image, display
import pydotplus

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Create and train a Decision Tree classifier
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)

# Make predictions
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Visualize the Decision Tree
def visualize_tree(tree, feature_names):
```

```
    dot_data = export_graphviz(tree, out_file=None,
                               feature_names=feature_names,
                               class_names=iris.target_names,
                               filled=True, rounded=True,
                               special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data)
    return Image(graph.create_png())

# Display the tree
display(visualize_tree(clf, iris.feature_names))
```

Explanation:

1. **Import Libraries**: We import the necessary libraries including `matplotlib` for plotting, `sklearn` for machine learning tasks, and `pydotplus` and `graphviz` for visualizing the decision tree.
2. **Load Dataset**: The Iris dataset is loaded from `sklearn`.
3. **Split Dataset**: The dataset is split into training and testing sets using `train_test_split`.
4. **Train Classifier**: A `DecisionTreeClassifier` is created and trained on the training data.
5. **Make Predictions**: The trained model is used to make predictions on the test data.
6. **Evaluate Model**: The accuracy of the model is calculated using `accuracy_score`.
7. **Visualize Tree**: The decision tree is visualized using `export_graphviz` to create a DOT file, which is then converted to a PNG image using `pydotplus`. The image is displayed using IPython's `display`.