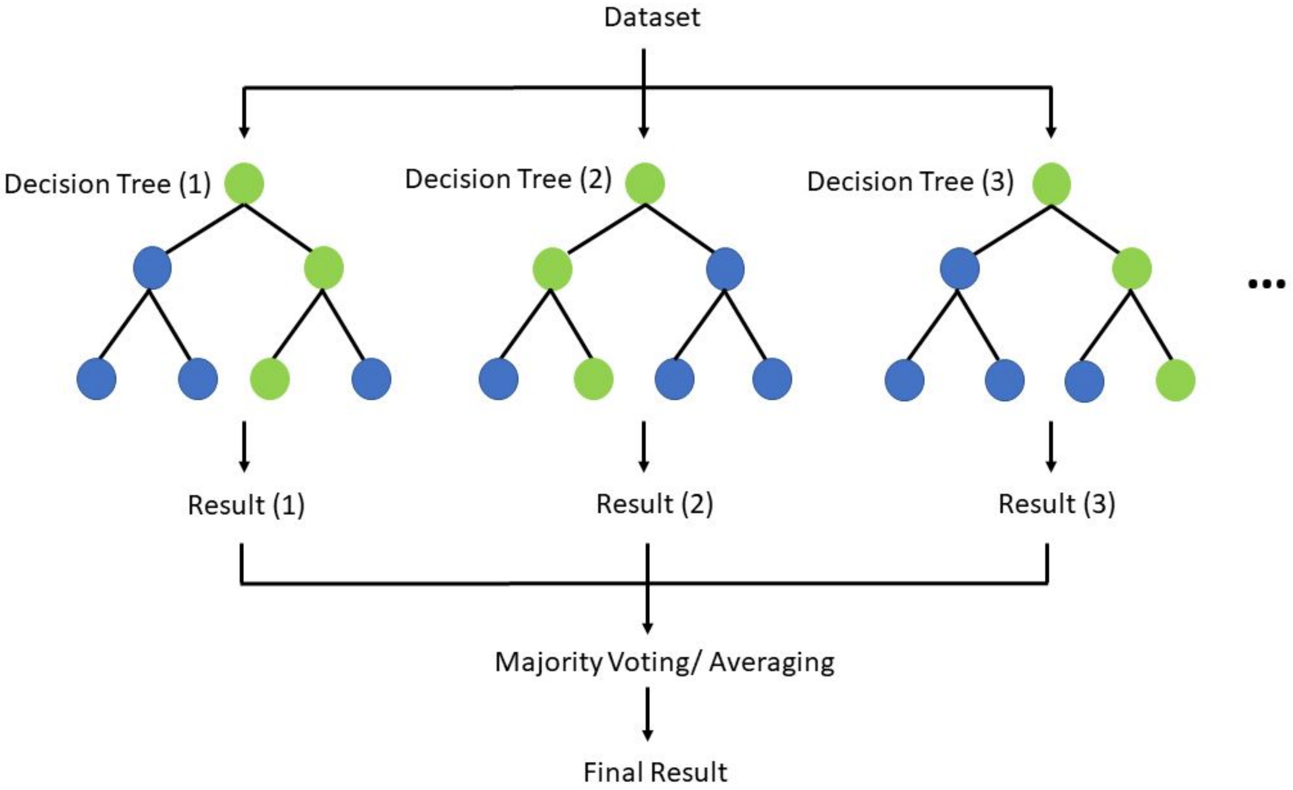
Machine Learning

By EvolkAI

Model Name:
Decision Tree Classifier

A Decision Tree Classifier is a simple yet powerful machine learning algorithm that creates a flowchart-like structure to make predictions. It starts with a root node that represents the entire dataset and splits it based on different features. The splits are determined by finding the most informative feature that best separates the data. This process is repeated recursively, creating branches and sub-branches until a stopping criterion is met, such as reaching a maximum depth or no further improvement in classification. Each leaf node represents a class or a decision.

During prediction, a new data point follows the path through the tree, eventually reaching a leaf node and obtaining its corresponding class label. Decision Trees are intuitive, easily interpretable, and capable of handling both numerical and categorical data. They can be prone to overfitting but can be mitigated by using techniques like pruning or ensembling. Decision Tree Classifier is widely used in various domains, including finance, healthcare, and marketing, due to its simplicity and effectiveness in solving classification problems.



Dataset

Decision Tree (1)

Decision Tree (2)

Decision Tree (3)

...

Result (1)

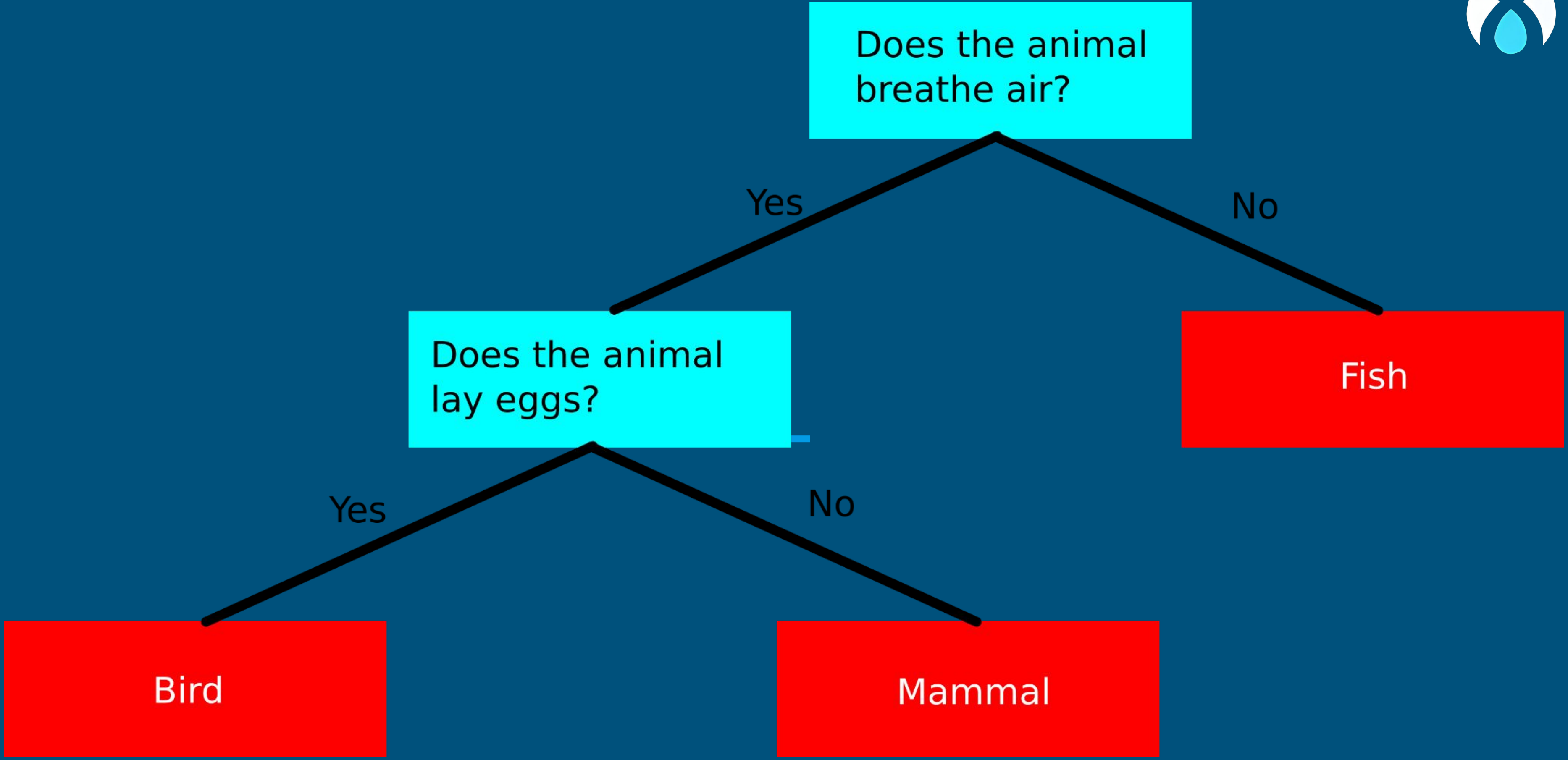
Result (2)

Result (3)

Majority Voting/ Averaging

Final Result

Random Forest



Does the animal breathe air?

Yes

No

Does the animal lay eggs?

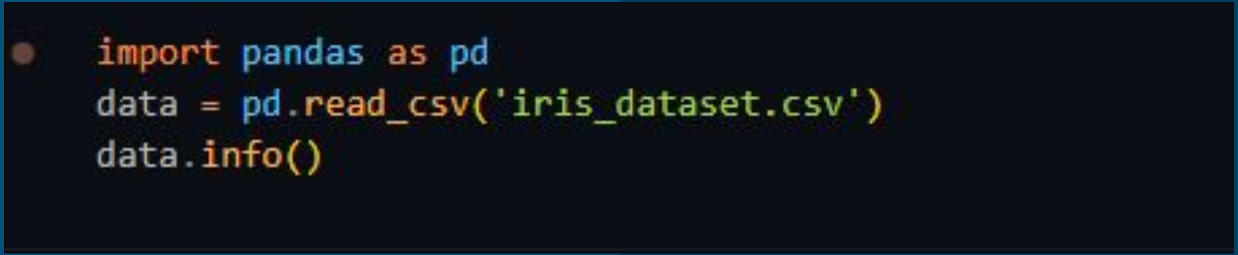
Fish

Yes

No

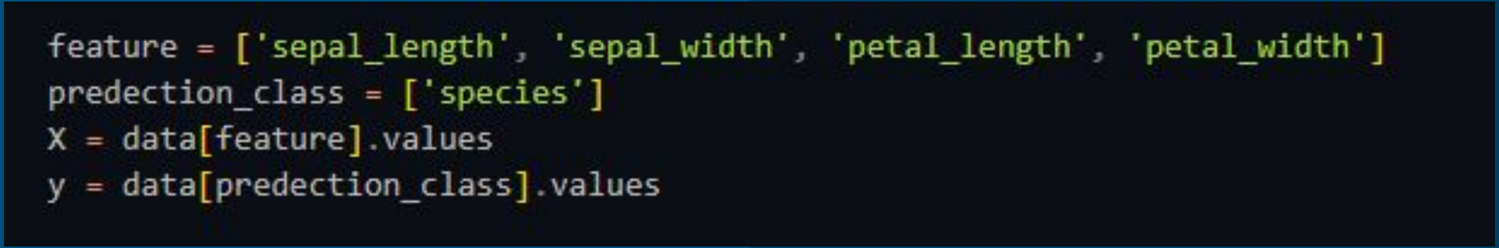
Bird

Mammal



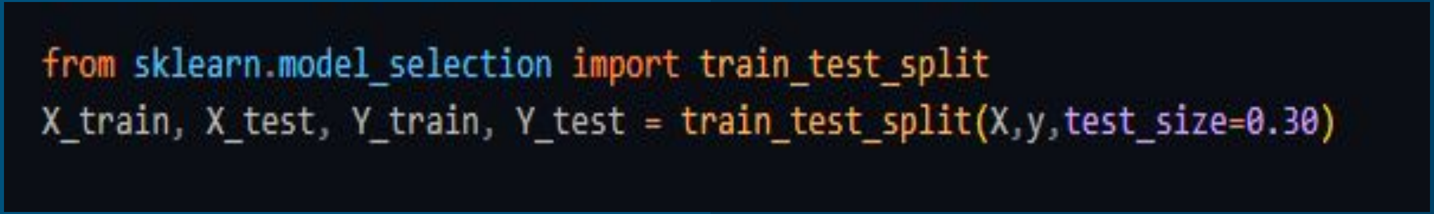
```python
import pandas as pd
data = pd.read_csv('iris_dataset.csv')
data.info()
```

In First line we Import pandas library as pd, then we read iris_dataset.csv file using the read_csv() function, and prints information about the data using the info() method.



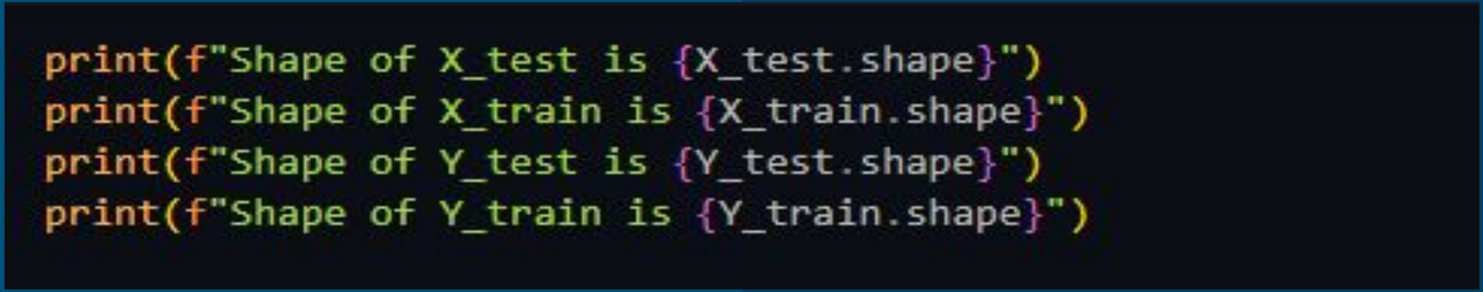
```
feature = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']
predection_class = ['species']
X = data[feature].values
y = data[predection_class].values
```

This defines the ‘feature’ and ‘predection_class’ variables, which specify the columns of the data to use as ‘features’ and the column to use as the prediction target.
The code then creates ‘X’ and ‘y’ arrays containing the values of these columns from the data DataFrame.



```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,y,test_size=0.30)
```

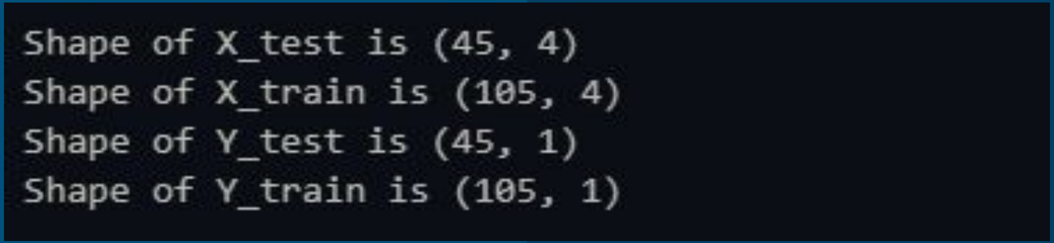
This imports the train_test_split function from the sklearn.model_selection module and uses it to split the data into training and testing sets. The test_size parameter specifies that 30% of the data should be used for testing.



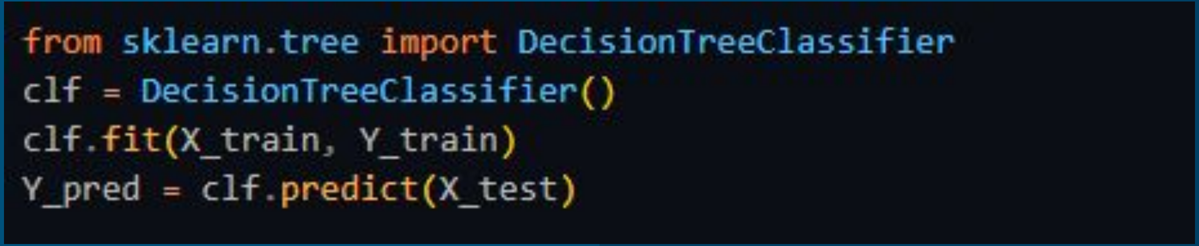
```
print(f"Shape of X_test is {X_test.shape}")
print(f"Shape of X_train is {X_train.shape}")
print(f"Shape of Y_test is {Y_test.shape}")
print(f"Shape of Y_train is {Y_train.shape}")
```

These lines print the shapes of the training and testing data arrays. This output totally depends on the test size we took while train_test_split.

OUTPUT

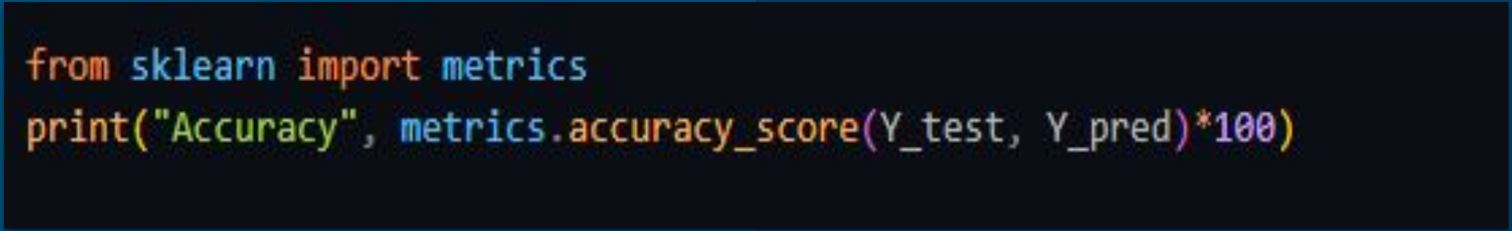


```
Shape of X_test is (45, 4)
Shape of X_train is (105, 4)
Shape of Y_test is (45, 1)
Shape of Y_train is (105, 1)
```



```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier()
clf.fit(X_train, Y_train)
Y_pred = clf.predict(X_test)
```

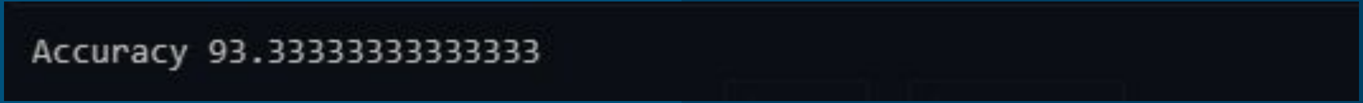
This imports the *Decision Tree Classifier* module from the sklearn tree library, creates a *Decision Tree Classifier* classifier object, fits the classifier with the training data using the *fit()* method, and use the *predict()* method to generate predictions for the testing data.



```
from sklearn import metrics
print("Accuracy", metrics.accuracy_score(Y_test, Y_pred)*100)
```

This imports the metrics module from sklearn and uses the accuracy_score() function to calculate the accuracy of the model on the testing data. The result is printed in the console.

OUTPUT



```
Accuracy 93.33333333333333
```

Conclusion

In conclusion, the Decision Tree Classifier algorithm is a versatile and intuitive tool for classification tasks. By constructing a tree-like structure based on informative features, it can make predictions with ease. Decision trees are easily interpretable, allowing for a clear understanding of the decision-making process. They handle both numerical and categorical data effectively and find applications in diverse domains such as finance, healthcare, and marketing. While decision trees can be prone to overfitting, techniques like pruning help to mitigate this issue.

Overall, the Decision Tree Classifier algorithm offers a powerful and widely-used solution for classification problems, providing accurate predictions and insights that aid in decision-making and problem-solving.

Thank You

🙏