Decision tree predictor

In [2]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
import pickle
```

Load Data

load the training and testing data

In [3]:
```
# Load feature vectors and labels
X = np.load("../X.npy")
y = np.load("../y.npy")

with open("../label_map.pkl", "rb") as f:
    label_map = pickle.load(f)
class_names = [label_map[i] for i in range(len(label_map))]
```

Split the data

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

Scale Photos

In [5]:
```
# Normalize to improve model performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [6]:
```
import joblib
joblib.dump(scaler, "FNN_scaler.pkl")
```

Out[6]:
```
['FNN_scaler.pkl']
```

Train the module

In [7]:
```
from sklearn.neural_network import MLPClassifier

fnn = MLPClassifier(
    hidden_layer_sizes=(2048, 1024),# 2 hidden layers: first with 2048 neuro
    activation='relu',              # the activation function (for non-linea
    solver='adam',                  # my fav optimizer (adaaaam)
    alpha=0.0001,                   # L2 regularization term to avoid overfi
```

```
    batch_size='auto',              # giving auto batches
    learning_rate='adaptive',       # adaptive learning change
    learning_rate_init=0.001,       # initial learning rate
    max_iter=300,                   # max number of epochs
    early_stopping=True,            # early stop if validation data doesn't
    validation_fraction=0.1,        # how much of the data for validation
    n_iter_no_change=10,            # how much patient for validation (to pr
    random_state=42,                # the same random we used all along the
    shuffle=True,                   # shuffle training data every epoch
)

fnn.fit(X_train, y_train)
```

Out[7]:

MLPClassifier

```
MLPClassifier(early_stopping=True, hidden_layer_sizes=(2048, 1024),
              learning_rate='adaptive', max_iter=300, random_state=
42)
```

Predict the testing data

In [8]:
```
y_pred_fnn = fnn.predict(X_test)
print("Random Forest Report:\n", classification_report(y_test, y_pred_fnn, t
```

```
Random Forest Report:
               precision    recall  f1-score   support

         cats       0.78      0.78      0.78       200
        panda       0.80      0.82      0.81       199
      spiders       0.79      0.78      0.78       200

     accuracy                           0.79       599
    macro avg       0.79      0.79      0.79       599
weighted avg       0.79      0.79      0.79       599
```

Confusion Matrix Results

In [9]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_fnn)
print(cm)
```

```
[[156  20  24]
 [ 19 163  17]
 [ 24  21 155]]
```

Load the module

In [10]:
```
import joblib
joblib.dump(fnn, "FNN_model.pkl")
```

Out[10]: ['FNN_model.pkl']

In [ ]: