Decision tree predictor

```
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

```
# 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

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
stratify=y, random_state=42)
```

Scale Photos

```
# Normalize to improve model performance
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

import joblib
joblib.dump(scaler, "DT_scaler.pkl")
```

```
['DT_scaler.pkl']
```

Train the module

```
from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier(
    n_estimators=600,          # More trees = better ensemble (200–500 is good
range)
    max_depth=None,              # Slightly deeper trees (try 20–30)
    min_samples_split=4,       # Prevents overfitting, needs at least 4 samples
to split a node
    min_samples_leaf=2,        # Prevents very small leaves (try 1–4)
    max_features='sqrt',       # Good for classification tasks (try also
'log2')
    class_weight='balanced',  # Handles class imbalance by weighting classes
```

```
automatically
    bootstrap=True,           # Enables bootstrap sampling (default)
    random_state=42,          # For reproducibility
    n_jobs=-1                 # Use all CPU cores to speed up training
)

rf.fit(X_train, y_train)
```

```
RandomForestClassifier(class_weight='balanced', min_samples_leaf=2,
                       min_samples_split=4, n_estimators=600, n_jobs=-1,
                       random_state=42)
```

Predict the testing data

```
y_pred_rf = rf.predict(X_test)
print("Random Forest Report:\n", classification_report(y_test, y_pred_rf,
target_names=class_names))
```

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

        cats       0.78      0.72      0.75       200
       panda       0.82      0.84      0.83       199
     spiders       0.82      0.86      0.84       200

    accuracy                           0.81       599
   macro avg       0.81      0.81      0.81       599
weighted avg       0.81      0.81      0.81       599
```

Confusion Matrix Results

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_rf)
print(cm)
```

```
[[145  28  27]
 [ 20 167  12]
 [ 20   8 172]]
```

Load the module

```
import joblib
joblib.dump(rf, "DT_model.pkl")
```

```
['DT_model.pkl']
```