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, "FNN_scaler.pkl")
```

```
['FNN_scaler.pkl']
```

Train the module

```
from sklearn.neural_network import MLPClassifier

fnn = MLPClassifier(
    hidden_layer_sizes=(1024, 512), # 2 hidden layers, 1024/512
    activation='relu', # the used activation (for non-linearity)
    solver='adam', # the optimizer (the loss function is `log` by default)
    learning_rate='adaptive', # learning rate
    max_iter=300, # max number of epochs
    early_stopping=True, # early stop when over fitting start
    random_state=42
)
```

```
fnn.fit(X_train, y_train)
```

```
MLPClassifier(early_stopping=True, hidden_layer_sizes=(1024, 512),
              learning_rate='adaptive', max_iter=300, random_state=42)
```

Predict the testing data

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

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

        cats       0.78      0.74      0.76       200
       panda       0.77      0.80      0.79       199
     spiders       0.77      0.78      0.77       200

    accuracy                           0.77       599
   macro avg       0.77      0.77      0.77       599
weighted avg       0.77      0.77      0.77       599
```

Confusion Matrix Results

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

```
[[149  24  27]
 [ 21 159  19]
 [ 22  23 155]]
```

Load the module

```
import joblib
joblib.dump(fnn, "FNN_model.pkl")
```

```
['FNN_model.pkl']
```