Report

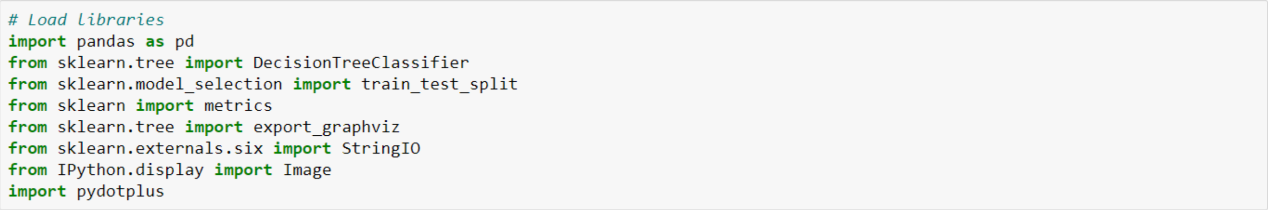
Objective

In this project, I have a dataset containing features extracted by a deep network on a 3-class classification problem. Each record corresponds to features generated for a sample. My objective is to build a decision tree model using these features. Your decision tree will be evaluated by its accuracy on the test set.

Dataset

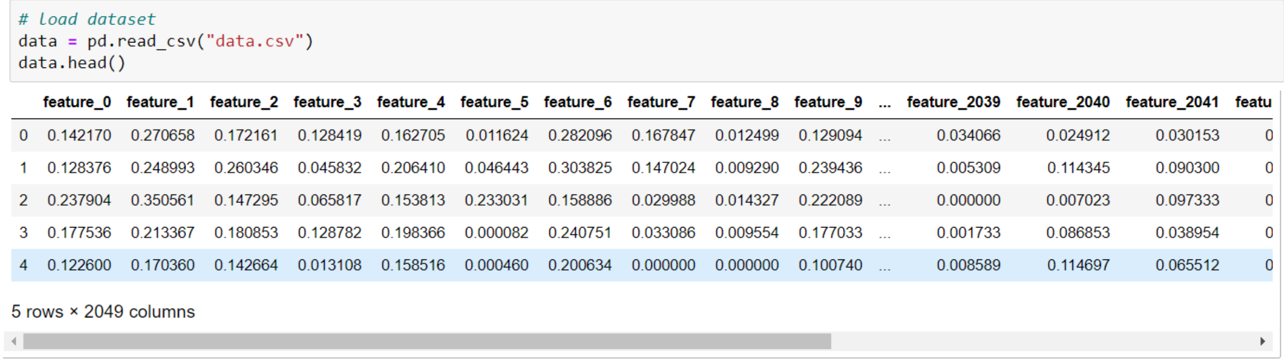
I have a csv file named data.csv. This dataset contains 2048 features for each record, with their labels in the last column indicating the class (“0”, “1” or “‘2”). There are a total of 2060 records.

1. In Python, sklearn is the package which contains all the required packages to implement Machine learning algorithm. I will use DecisionTreeClassifier.
   First, I import all the necessary modules.



```
# Load libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus
```

2. Now, I load the dataset using pandas' read CSV function.

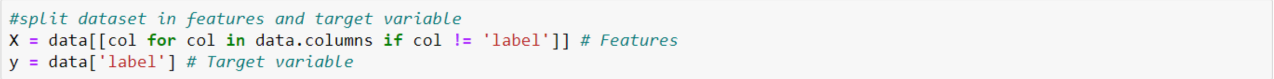


```
# Load dataset
data = pd.read_csv("data.csv")
data.head()
```

	feature_0	feature_1	feature_2	feature_3	feature_4	feature_5	feature_6	feature_7	feature_8	feature_9	...	feature_2039	feature_2040	feature_2041	featu
0	0.142170	0.270658	0.172161	0.128419	0.162705	0.011624	0.282096	0.167847	0.012499	0.129094	...	0.034066	0.024912	0.030153	0
1	0.128376	0.248993	0.260346	0.045832	0.206410	0.046443	0.303825	0.147024	0.009290	0.239436	...	0.005309	0.114345	0.090300	0
2	0.237904	0.350561	0.147295	0.065817	0.153813	0.233031	0.158886	0.029988	0.014327	0.222089	...	0.000000	0.007023	0.097333	0
3	0.177536	0.213367	0.180853	0.128782	0.198366	0.000082	0.240751	0.033086	0.009554	0.177033	...	0.001733	0.086853	0.038954	0
4	0.122600	0.170360	0.142664	0.013108	0.158516	0.000460	0.200634	0.000000	0.000000	0.100740	...	0.008589	0.114697	0.065512	0

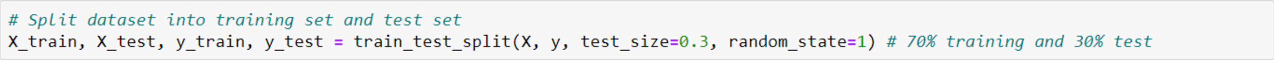
5 rows × 2049 columns

3. Here, I need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).



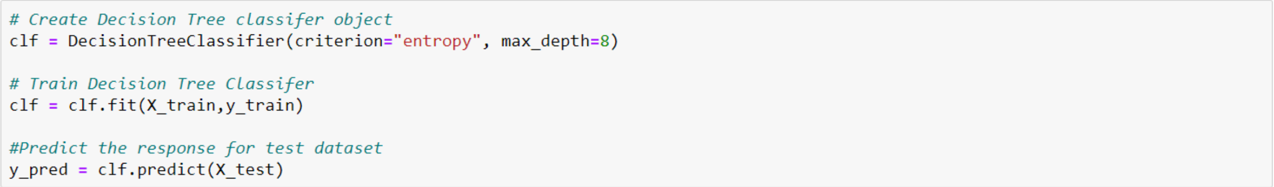
```
#split dataset in features and target variable
X = data[[col for col in data.columns if col != 'label']] # Features
y = data['label'] # Target variable
```

4. To understand model performance, dividing the dataset into a training set and a test set is a good strategy. Let's split the dataset by using function train_test_split(). I need to pass 3 parameters features, target, and test_set size.



```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test
```

5. I create a Decision Tree Model using Scikit-learn.



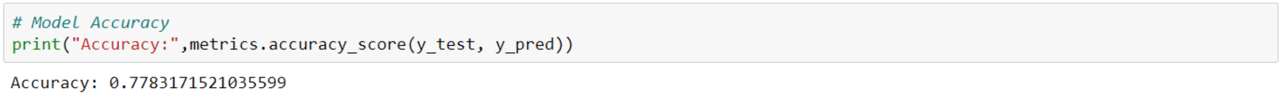
```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=8)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

6. I estimate, how accurately the classifier or model can predict labels.

Accuracy can be computed by comparing actual test set values and predicted values.

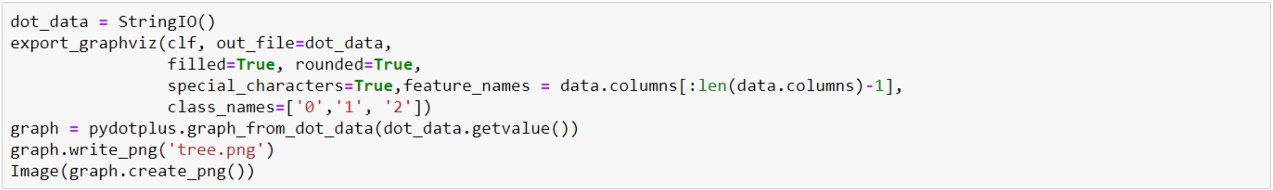


```
# Model Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7783171521035599
```

Well, I got a classification rate of 77.83%, considered as good accuracy.

7. I can use Scikit-learn's export_graphviz function for display the tree within a Jupyter notebook. For plotting tree, you also need to install graphviz and pydotplus.



```
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True,feature_names = data.columns[:len(data.columns)-1],
                class_names=['0','1', '2'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tree.png')
Image(graph.create_png())
```

In the decision tree chart, each internal node has a decision rule that splits the data. Gini referred as Gini ratio, which measures the impurity of the node. I can say a node is pure when all of its records belong to the same class, such nodes known as the leaf node.