session3_decisiontree_demo

September 20, 2019

1 Machine Learning Demo - Decision Tree

Kim Hee (Graduate research assistant) Universitätsmedizin Mannheim, Mannheim (UMM)

- This is prepared for `Data analysis tools (Datenanalysewerkzeuge)` at MIRACUM summer school 2019

1.0.1 Protocol:

1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[2]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree␣
 ↪Classifier
from sklearn.model_selection import train_test_split # Import train_test_split␣
 ↪function
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix
```

1.0.2 Protocol:

1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[3]:
```
diabetes = pd.read_csv("data/diabetes.csv")
diabetes.Outcome.value_counts()
```

[3]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

1.0.3 Protocol:

1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[4]:
```
X = diabetes.iloc[:,:-1]
y = diabetes.Outcome # Target variable
```

[5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,␣
↪random_state=1)
```

1.0.4 Protocol:

1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[6]:
```
dt = DecisionTreeClassifier()
dt = dt.fit(X_train,y_train)
```

1.0.5 Protocol:

1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[7]:
```
y_pred = dt.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
pd.DataFrame(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.7142857142857143
```

[7]:
```
    0   1
0  38  10
1  12  17
```

1.0.6 Protocol:

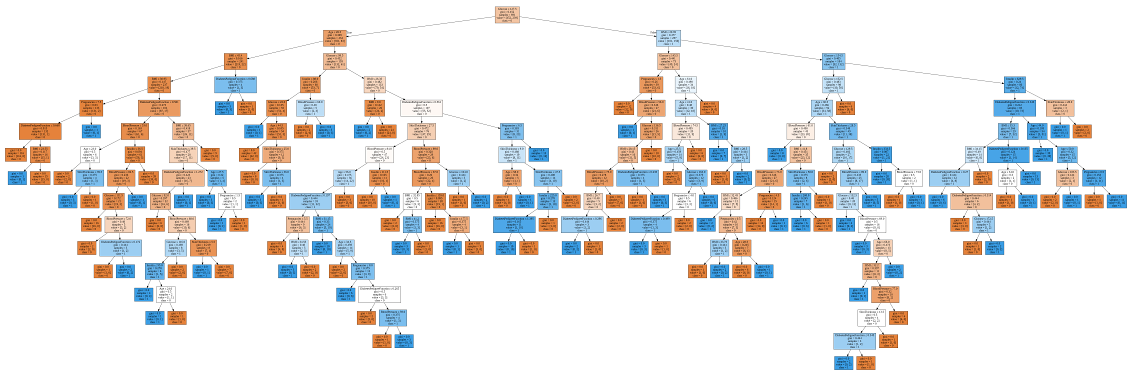
1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[8]:
```
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
from IPython.display import Image
import pydotplus

dot_data = StringIO()
export_graphviz(dt,
                out_file = dot_data,
                filled = True,
                special_characters = True,
                feature_names = X.columns,
                class_names = ['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('images/diabetes.png')
Image(graph.create_png())
```

```
/opt/conda/lib/python3.7/site-packages/sklearn/externals/six.py:31:
DeprecationWarning: The module is deprecated in version 0.21 and will be removed
in version 0.23 since we've dropped support for Python 2.7. Please rely on the
official version of six (https://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", DeprecationWarning)
```

[8]:



1.0.7 Protocol:

1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[9]:
```
dt = DecisionTreeClassifier(criterion="entropy", max_depth=2)
dt = dt.fit(X_train,y_train)
```

1.0.8 Protocol:

1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[10]:
```
y_pred = dt.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
pd.DataFrame(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.7922077922077922
```

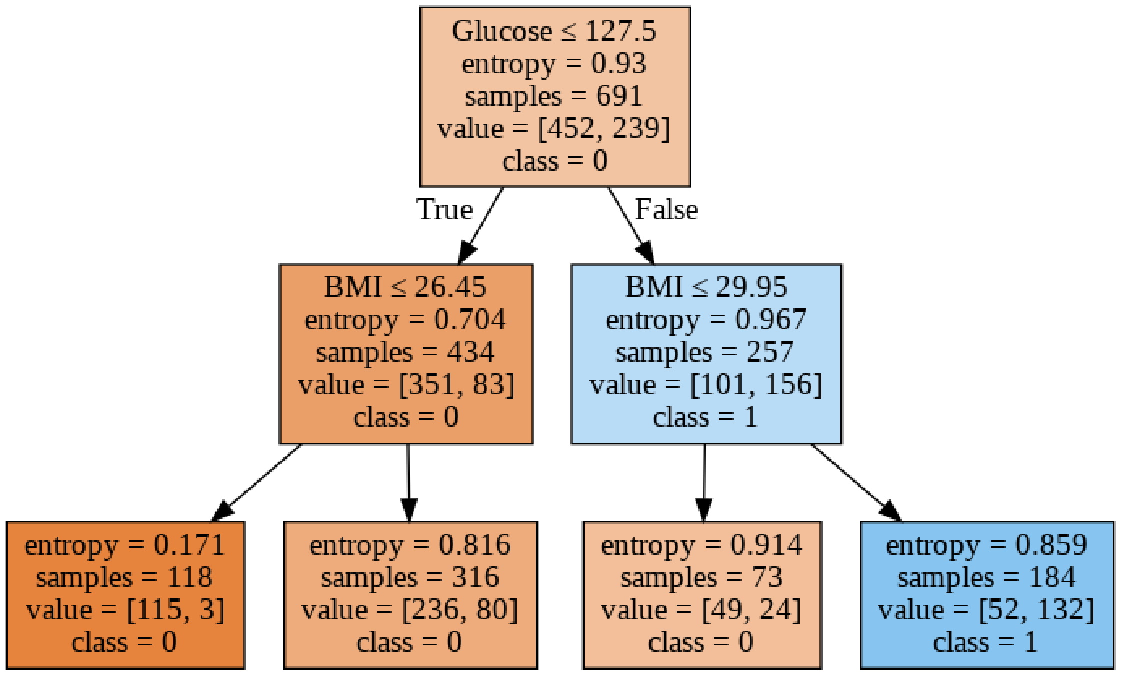
[10]:
```
    0   1
0  43   5
1  11  18
```

1.0.9 Protocol:

1. Import required libraries 2. Load data (download) and check the label balance 3. Split data to features and label and to train and test data 4. Create and Train a classifier 5. Apply the classifier to test data and print model performance 6. Visualize the classifier

[11]:
```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(dt,
                out_file = dot_data,
                filled = True,
                special_characters = True,
                feature_names = X.columns,
                class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('images/diabetes.png')
Image(graph.create_png())
```

[11]:


Glucose ≤ 127.5
entropy = 0.93
samples = 691
value = [452, 239]
class = 0
True
False
BMI ≤ 26.45
entropy = 0.704
samples = 434
value = [351, 83]
class = 0
BMI ≤ 29.95
entropy = 0.967
samples = 257
value = [101, 156]
class = 1
entropy = 0.171
samples = 118
value = [115, 3]
class = 0
entropy = 0.816
samples = 316
value = [236, 80]
class = 0
entropy = 0.914
samples = 73
value = [49, 24]
class = 0
entropy = 0.859
samples = 184
value = [52, 132]
class = 1

```
[ ]:
```