decision-trees

September 20, 2019

1 Instructor Do: Decision Trees

[1]:
```
# Initial imports
import pandas as pd
from path import Path
from sklearn import tree
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score,␣
 ↪classification_report

# Needed for decision tree visualization
import pydotplus
from IPython.display import Image
```

1.1 Loading and Preprocessing Loans Encoded Data

[2]:
```
# Loading data
file_path = Path("../Resources/loans_data_encoded.csv")
df_loans = pd.read_csv(file_path)
df_loans.head()
```

[2]:

	amount	term	age	bad	month_num	education_Bechalor	\
0	1000	30	45	0	6	0	
1	1000	30	50	0	7	1	
2	1000	30	33	0	8	1	
3	1000	15	27	0	9	0	
4	1000	30	28	0	10	0	

	education_High School or Below	education_Master or Above	\
0	1	0	
1	0	0	
2	0	0	
3	0	0	
4	0	0	

	education_college	gender_female	gender_male

```
0                   0              0             1
1                   0              1             0
2                   0              1             0
3                   1              0             1
4                   1              1             0
```

[3]:
```
# Define features set
X = df_loans.copy()
X.drop("bad", axis=1, inplace=True)
X.head()
```

[3]:
```
   amount  term  age  month_num  education_Bechalor  \
0    1000    30   45          6                   0
1    1000    30   50          7                   1
2    1000    30   33          8                   1
3    1000    15   27          9                   0
4    1000    30   28         10                   0

   education_High School or Below  education_Master or Above  \
0                                1                          0
1                                0                          0
2                                0                          0
3                                0                          0
4                                0                          0

   education_college  gender_female  gender_male
0                  0              0            1
1                  0              1            0
2                  0              1            0
3                  1              0            1
4                  1              1            0
```

[4]:
```
# Define target vector
y = df_loans["bad"].values.reshape(-1, 1)
y[:5]
```

[4]:
```
array([[0],
       [0],
       [0],
       [0],
       [0]])
```

[5]:
```
# Splitting into Train and Test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=78)
```

[6]:
```
# Creating StandardScaler instance
scaler = StandardScaler()
```

[7]:
```
# Fitting Standard Scaller
X_scaler = scaler.fit(X_train)
```

```
/anaconda3/envs/dev/lib/python3.7/site-
packages/sklearn/preprocessing/data.py:645: DataConversionWarning: Data with
input dtype int64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
```

[8]:
```
# Scaling data
X_train_scaled = X_scaler.transform(X_train)
X_test_scaled = X_scaler.transform(X_test)
```

```
/anaconda3/envs/dev/lib/python3.7/site-packages/ipykernel_launcher.py:2:
DataConversionWarning: Data with input dtype int64 were all converted to float64
by StandardScaler.

/anaconda3/envs/dev/lib/python3.7/site-packages/ipykernel_launcher.py:3:
DataConversionWarning: Data with input dtype int64 were all converted to float64
by StandardScaler.
  This is separate from the ipykernel package so we can avoid doing imports
until
```

1.2 Fitting the Decision Tree Model

[9]:
```
# Creating the decision tree classifier instance
model = tree.DecisionTreeClassifier()
```

[10]:
```
# Fitting the model
model = model.fit(X_train_scaled, y_train)
```

1.3 Making Predictions Using the Tree Model

[11]:
```
# Making predictions using the testing data
predictions = model.predict(X_test_scaled)
```

1.4 Model Evaluation

[12]:
```
# Calculating the confusion matrix
cm = confusion_matrix(y_test, predictions)
cm_df = pd.DataFrame(
    cm, index=["Actual 0", "Actual 1"], columns=["Predicted 0", "Predicted 1"]
)

# Calculating the accuracy score
acc_score = accuracy_score(y_test, predictions)
```

[13]:
```
# Displaying results
print("Confusion Matrix")
display(cm_df)
print(f"Accuracy Score : {acc_score}")
```

```
print("Classification Report")
print(classification_report(y_test, predictions))
```

```
Confusion Matrix
```

	Predicted 0	Predicted 1
Actual 0	50	34
Actual 1	22	19

```
Accuracy Score : 0.552
Classification Report
```

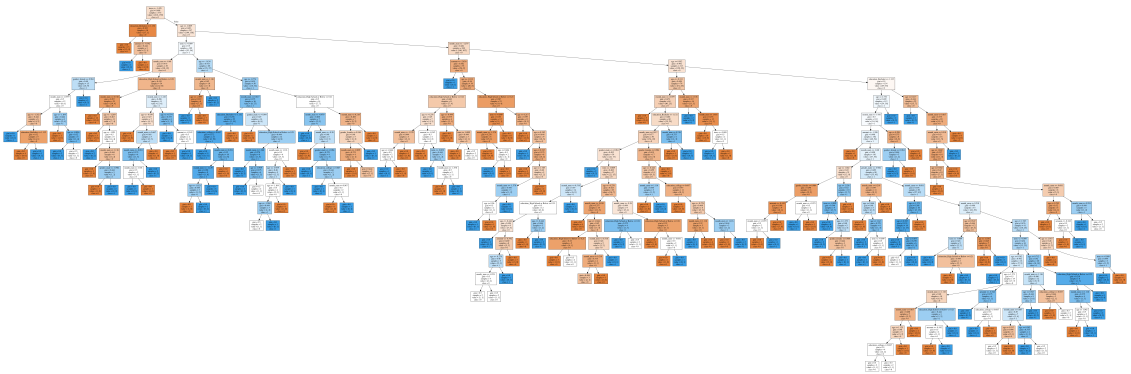
	precision	recall	f1-score	support
0	0.69	0.60	0.64	84
1	0.36	0.46	0.40	41
micro avg	0.55	0.55	0.55	125
macro avg	0.53	0.53	0.52	125
weighted avg	0.58	0.55	0.56	125

1.5 Visualizing the Decision Tree

[14]:
```
# Create DOT data
dot_data = tree.export_graphviz(
    model, out_file=None, feature_names=X.columns, class_names=["0", "1"],␣
 ↪filled=True
)

# Draw graph
graph = pydotplus.graph_from_dot_data(dot_data)

# Show graph
Image(graph.create_png())
```

[14]:


[15]:
```
# Saving the tree as PDF
file_path = Path("../Resources/loans_tree.pdf")
graph.write_pdf(file_path)

# Saving the tree as PNG
file_path = Path("../Resources/loans_tree.png")
graph.write_png(file_path)
```

[15]:
```
True
```