CS652: Applied Machine Learning
Assignment#4

การจำแนกด้วย Decision Tree และ Random Forest

Part 1: ทำงานกับชุดข้อมูลที่มี categorical features

ใช้ dataset ชื่อ adult.csv สามารถดูรายละเอียดข้อมูลได้ที่ http://archive.ics.uci.edu/ml/datasets/Adult ในการทำ binary classification เพื่อทำนายว่าคนอเมริกันแต่ละคนจะหารายได้ได้มากกว่า $50,000 ต่อปีหรือไม่ ชุดข้อมูลนี้จะประกอบด้วยฟีเชอร์ที่เป็น numerical และ categorical (ชุดข้อมูลนี้ถูกดึงมาจากฐานข้อมูลประชากรในปี 1994)

ขั้นตอนที่ 1: Reading the data

ให้ดาวน์โหลดไฟล์ csv สองไฟล์สำหรับ training set และ test set ที่กำหนดให้แล้วเก็บไว้ใน working folder ของตัวเอง ทั้งสองไฟล์นี้ถูกแบ่งสำหรับ train/test มาให้แล้วโดยผู้สร้าง dataset ชุดนี้

(http://www.cse.chalmers.se/~richajo/dit866/data/adult_train.csv)

(http://www.cse.chalmers.se/~richajo/dit866/data/adult_test.csv)

ให้เขียนโปรแกรมเพื่ออ่าน csv file ด้วย Pandas จากนั้นแบ่งข้อมูลเป็นอินพุต X และเอาต์พุต Y โดย ฟีเชอร์ที่จะทำนายมีชื่อว่า target

ขั้นตอนที่ 2: Encoding the features as numbers

ในชุดข้อมูลนี้มีหลายฟีเชอร์ที่มีค่าเป็น categorical values เช่น workclass, education เป็นต้น แต่โมเดลทั้งหมดของ scikit-learn จะทำงานกับข้อมูลที่เป็นตัวเลข ดังนั้นจึงต้องแปลงฟีเชอร์เหล่านี้ให้เป็นตัวเลข วิธีที่ตรงที่สุดที่สามารถแปลงข้อมูลให้เป็นตัวเลขคือ one-hot-encoding ซึ่งวิธีนี้เป็นการสร้าง column ใหม่สำหรับค่าแต่ละค่าของฟีเชอร์นั้น ๆ ใน scikit-learn มีเครื่องมือหลายตัวที่สามารถทำ one-hot-encoding กับ categorical features ได้ และหนึ่งในเครื่องมือนั้นก็คือ *DictVectorizer* โดยเราต้องเก็บฟีเชอร์ด้วยโครงสร้างแบบ dictionaries ด้านล่างเป็นตัวอย่างในการแทนข้อมูลหนึ่ง instance จาก Adult dataset เป็น dictionary

```
{'age': 44,
 'workclass': 'Private',
 'fnlwgt': 160323,
 'education': 'Some-college',
 'education-num': 10,
 'marital-status': 'Married-civ-spouse',
 'occupation': 'Machine-op-inspct',
 'relationship': 'Husband',
 'race': 'Black',
```

```
 'sex': 'Male',
 'capital-gain': 7688,
 'capital-loss': 0,
 'hours-per-week': 40,
 'native-country': 'United-States'}
```

Pandas มีเครื่องมือในการแปลง DataFrame เป็นลิสต์ของ dictionaries เช่น สมมติให้ my_training_data เป็น dataframe

```
dicts_for_my_training_data = my_training_data.to_dict('records')
```

จากนั้นสร้าง DictVectorizer และนำมาใช้งาน ดังนี้

```
dv = DictVectorizer()
X_train_encoded = dv.fit_transform(dicts_for_my_training_data)
```

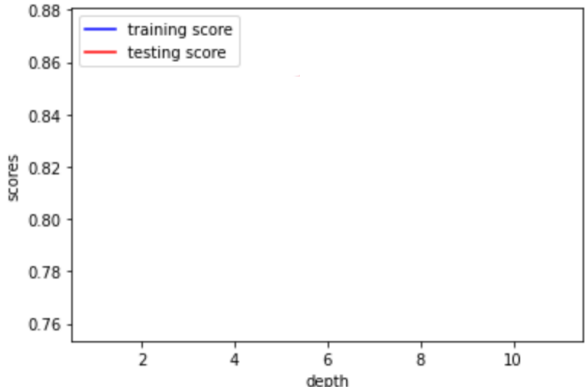
ส่วนของ train data เราจะใช้เมธอด fit_transform โดยเมธอดนี้จะเรียกเมธอด fit สำหรับการฝึกสอน (การฝึกสอนด้วย DictVectorizer จะ mapping ข้อมูล categories ไปเป็นตำแหน่งของ column) จากนั้นจะเรียก เมธอด transform เพื่อแปลงข้อมูลให้เป็น matrix

ส่วนของ test data เราจะเรียกเมธอด transform เพียงอย่างเดียว ดังนี้

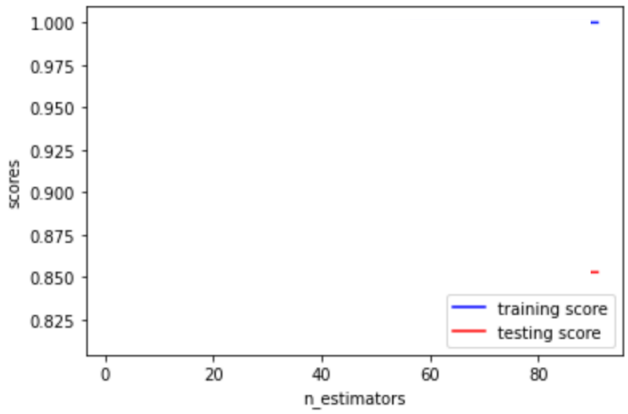
```
X_test_encoded = dv.transform(dicts_for_my_test_data)
```

Part 2: Decision trees and random forests

1. ให้ใช้ DecisionTreeClassifier ของ scikit-learn และประเมินผลด้วยการ plot เทียบค่า accuracy scores ของ training set และ test set ที่ระดับต่าง ๆ ของค่า *max_depth* (จาก 1 - 12) ตัวอย่างการ Plot แสดงดังภาพ



2. จากนั้นให้ใช้ RandomForestClassifier และลองปรับค่า *n_estimators* (จำนวนต้นไม้ตัดสินใจที่ใช้) เพื่อดูว่าจะกระทบกราฟของการ underfitting/overfitting อย่างไร โดยอาจจะพิจารณาจำนวนต้นไม้ตั้งแต่หนึ่งถึงร้อยต้น



Hint การทดลองนี้อาจจะใช้เวลาในการทำงาน ถ้าค่า *n_estimators* เยอะ เราอาจจะลดเวลาในการฝึกสอนได้โดยการปรับค่า hypermeter ที่ชื่อ *n_jobs* (จำนวน jobs ที่ต้องการให้ทำงานแบบขนาน) ถ้าไม่มีการกำหนดค่านี้ จะหมายถึงให้ทำทีละ jobs หรือ ใช้เพียงหนึ่ง core ของ CPU แต่ถ้าค่าของ *n_jobs* มีค่าเป็น -1 ทุก ๆ core ของ CPU จะถูกนำมาใช้ทำงาน ทำให้ฝึกสอนต้นไม้หลาย ๆ ต้นทำได้พร้อม ๆ กัน

3. ให้สรุปผลที่ได้จาก 1 และ 2 เป็นเซลล์ Text ในโปรแกรม ด้วยการตอบคำถามต่อไปนี้
   - การใช้ decision tree และ random forest ที่มีจำนวนต้นไม้เพียง 1 ต้น ให้ผลที่ต่างกันหรือไม่ ถ้าต่างกัน ท่านคิดว่าเหตุใดจึงเป็นเช่นนั้น
   - ค่าของ test set accuracy เป็นอย่างไรเมื่อจำนวนต้นไม้ใน random forests เพิ่มขึ้น
   - เวลาที่ใช้ในการฝึกสอนเป็นอย่างไรเมื่อจำนวนต้นไม้ใน random forests เพิ่มขึ้น

Part 3: Feature importances in random forest classifiers

การทำงานของ Decision trees และ Random forests จะมีการคำนวณค่า *importance scores* ของแต่ละฟีเชอร์เพื่อเลือกฟีเชอร์สำหรับการแตกกิ่งต้นไม้ ใน scikit-learn เราสามารถพิมพ์ค่า *importance scores* ของโมเดลได้จาก attribute ที่ชื่อ *feature_importances_* ซึ่งเป็นโครงสร้างแบบ NumPy array ที่เก็บค่า *importance scores* ของแต่ละฟีเชอร์ในรูปของ matrix สำหรับ random forests ค่า *importance scores* จะเป็นค่าที่ได้จากการเฉลี่ย scores ของต้นไม้ทุกต้นใน random forests

ให้แสดงค่า *importance scores* ของแต่ละฟีเชอร์ในรูปของตารางหรือกราฟ และเพื่อให้ค่า *importance scores* เข้าใจได้ง่ายขึ้น ให้พิมพ์ชื่อของ features จาก attribute ชื่อ *feature_names_* ออกมาด้วย โดยทำการเรียงลำดับ ฟีเชอร์ตาม *importance scores* จากมากไปน้อย ให้สังเกตว่าอะไรคือฟีเชอร์ที่มีความสำคัญในระดับต้น ๆ และวิเคราะห์ว่าทำไมผลลัพธ์จึงเป็นเช่นนี้

Hint. ค่า *importance scores* แค่บอกเราว่าฟีเชอร์นั้นดีสำหรับการจำแนกคลาสของข้อมูลหรือไม่ แต่ไม่ได้บอกเราว่าความสัมพันธ์ระหว่างฟีเชอร์และคลาสเอาต์พุตคืออะไร กล่าวคือไม่ได้บอกว่าฟีเชอร์นี้ทำให้มีโอกาสมากหรือน้อยที่บุคคลนั้นจะเป็นผู้มีรายได้สูง

การส่งงาน

- กำหนดส่งงาน **26 มีนาคม 2565** ก่อน 23:59 น.
- ตั้งชื่อไฟล์ด้วยเลขทะเบียน ตามด้วยขีดล่างและชื่อการบ้าน เช่น 640961XXXX_ass4.ipynb
- ภายในไฟล์ให้ใช้ Label เพื่อแบ่งงานออกเป็นตอน ๆ ตามที่กำหนด พร้อมทั้งตอบคำถามในแต่ละส่วนด้วยการใช้เซลล์แบบที่เป็น Text
- ส่งงานทาง courses.cs.tu.ac.th