課程閱讀

D27:實作樹型(Tree Base)模型

本日作業



問題討論

 $\blacksquare \checkmark$

學習心得(完成)

```
• 透過實作更加了樹型模型的運作

    實作與使用決策樹模型

(本日課程為實作導向課程,同學可參考附件"實作 TreeBase 模型.ipynb")
生成假數據
```

['Green', 3.1, 'Apple'], ['Yellow', 3.2, 'Apple'],

```
5
       ['Red', 1, 'Grape'],
6
       ['Yellow', 3.3, 'Lemon'],
       ['Yellow', 3.1, 'Lemon'],
       ['Green', 3, 'Apple'],
8
9
       ['Red', 1.1, 'Grape'],
10
       ['Yellow', 3, 'Lemon'],
       ['Red', 1.2, 'Grape'],
11
12 ]
數據分割
一般在進行訓練前會將數據集,會將數據集切分成訓練集(training set) 與驗證集(validation set),這
```

return train_df, test_df

4

5 6

8

9

10 11

12

分割測試集與訓練集 1 def train_test_split(df, test_size=0.1): 3

```
檢測節點函式
在樹型模型中,當節點中的資料都屬於同一類別時即不須再往下分割,此節點即為終端節點(leaf
node)。因此這部分實作檢測節點資料是否為同一類別的函式。
 # 檢查資料是否都為同一類別
 def check_purity(data):
   '''Function to check if input data all belong to the same class
   Parameter
   data: list
     Input data
                                                      1 check_purity(df.values)
```

False

Parameter 5 data: list 6

取得可分割特徵值

8

中取得。

9 10

11 12

判別特徵值型態

def determine_type_of_feature(df):
 '''Function to get features types

Parameter

df: pd.DataFrame

potential_splits = {}

別,而類別型可以用是否相同類別判斷。

#由給定的輸入DataFrane給個特徵值的型態(數值型特徵或類別型特徵)

, n columns = data.shape

Input data

#取的資料的label訊息 labels = data[:, -1]

9 #取的資料的label訊息 labels = data[:, -1] 10 11 12 #檢查是否所有的label都為同一種

樹型模型中,每個節點會根據所選取的特徵設定一個條件值,而這個條件值可以從特徵所有的獨特值

```
# 根據給定的資料,取得每個特徵(feature)可能做為樹型模型分割能點的值
# 可能作為分割膨胀得值的為每個特徵的獨特徵(unique value)
def get_potential_splits(data):
        Function to get all potential split value for tree base model
    data: list
Input data
    potential_splits = {}
                                                                                                                    1 get_potential_splits(df.values)
     , n_columns = data.shape
                                                                                                                    {0: array{['Green', 'Red', 'Yellow'], dtype=object),
1: array{[1.0, 1.1, 1.2, 3.0, 3.1, 3.2, 3.3], dtype=object)}
     #此處的-1是為了扣掉1abe1的欄位
     for column_index in range(n_columns - 1):
        #開譯機位取約計算的開計值(unique values)
values = data[:, column_index]
unique_values = np.unique(values)
        #蔣取得的可能分割值錄存在potential_split的字典中(key=特徵欄位的index, value:此特徵可能的分數值)
potential_splits[column_index] = unique_values
    return potential_splits
          根據給定的資料,取得每個特徵(feature)可能做為樹型模型分割節點的值
1
        # 可能作為分割節點得值即為每個特徵的獨特值(unique value)
3
        def get_potential_splits(data):
```

```
Input raw pd.DataFrame data
    feature_types = []
    #若特徵的獨特值個數較少。及當作類別型特徵資料(若為數值型,獨特值個數應該會很多)
#此處簡易的將判斷方法設為資料個數的1/3次方,此值可以自行修改選較為適合的個數
                                                                                                            1 determine_type_of_feature(df)
    n_unique_values_treshold = int(len(df)**(1/3))
                                                                                                          ['categorical', 'continuous']
    for feature in df.columns:
        if feature != "label":
unique_values = df[feature].unique()
rep_value = unique_values[0] #應出一個值數此特徵的代表
           if (isinstance(rep_value, str)) or (len(unique_values) <= s_unique_values_treshold):
    feature_types.append("categorical")</pre>
           else:
               feature_types.append("continuous")
   return feature_types
      #由給定的輸入DataFrame給個特徵值的型態(數值型特徵或類別型特徵)
      def determine type of feature(df):
             '''Function to get features types
4
            Parameter
            df: pd.DataFrame
6
```

結點再選取條件值時,根據特徵資料型態的不同,會有不同的條件判斷式。如數值型可以用大小於判

```
if type_of_feature == "continuous":
      #數值型特徵分割
      data_left = data[split_column_values <= split_value]</pre>
                                                                               左節點
      data_right = data[split_column_values > split_value]
       #類別型特徵分割
      data_left = data[split_column_values == split_value]
      data_right = data[split_column_values != split_value]
   return data_left, data_right
     def split_data(data, split_column, split_value):
1
         '''Function to splitted left and right nodes
2
3
         Parameter
5
         data: list
             Input data
6
         split_column: int
8
             index for feature column
9
         split_value: float or int or string
10
              value to be used as split benchmark
```

節點

存款 > 5000

Yes

```
if task_type == "regression":
       #回歸任務
       leaf = np.mean(label_column)
    else:
       #分類任務
       #取得所有輸入資料的獨立類別與其個數
       unique_classes, counts_unique_classes = np.unique(label_column, return_counts=True)
       #以個數最多的類別,作為此節點的輸出類別
       index = counts_unique_classes.argmax()
       leaf = unique_classes[index]
   return leaf
    def create_leaf(data, task_type):
2
       '''Function to create leaf node
3
       Parameters
4
       -----
5
       data: list
6
          Input data
7
       task_type: str
8
          indicate the type of tree (regression or classification)
9
10
11
       #取的資料的label欄位
12
       label_column = data[:, -1]
計算訊息增益
節點分割的依據在於取得最大的訊息增益,而計算訊息增益的方法之一就是計算樣本的熵。(同學可到
決策樹章節複習訊息增益的計算)
```

#計算機率 probabilities = counts / counts.sum() #計算entropy entropy = sum(probabilities * -np.log2(probabilities))

2 3

5

7

8

9

10 11

12

def calculate_entropy(data):

label_column = data[:, -1]

#取得所有輸入資料的獨立類別與其個數

_, counts = np.unique(label_column, return_counts=True)

#取的資料的label訊息

return entropy

def calculate_entropy(data):

#取的資料的label訊息

#計算機率

#計算entropy

計算總訊息增益

label_column = data[:, -1]

#取得所有輸入資料的獨立類別與其個數

probabilities = counts / counts.sum()

_, counts = np.unique(label_column, return_counts=True)

```
return overall metric
    def calculate_overall_metric(data_below, data_above, metric_function):
2
```

n = len(data_below) + len(data_above)

p_data_below = len(data_below) / n p_data_above = len(data_above) / n

```
return overall_metric
決策樹演算法
將上述介紹的函式組裝成決策樹模型,模型中的參數代表的意義如下:
  1. metric_function:代表欲使用的訊息增益計算函式(ex:Entropy)
  2. task_type: 此決策樹是分類樹(classification)或回歸樹(regression)
  3. counter:用來計算樹生成的深度
  4. min_sample:代表節點要分割最少需要的樣本數
  5. max_depth: 樹最深的層數
class decision_tree():
   '''Decision Tree model
  Parameters
   metric_function: function
     the metric function used to calculate information gain
```

def calculate overall metric(data below, data above, metric function):

overall_metric = (p_data_below * metric_function(data_below)

+ p_data_above * metric_function(data_above))

```
8
           indicate the type of tree (regression or classification)
9
       counter: int
10
           counter for recording number of splits
11
       min_samples: int
           minimum number of samples for a node to be able to split
詳細使用操作
請參照使用實做 TreeBase 模型 .ipynb 檔進行更詳細的使用操作
知識點回顧
```

task_type: str

5

6

metric_function: function

task_type: str

此 Youtube 介紹如何使用 Python 實作決策樹,內容詳細強烈建議學員觀看

```
Let's Write a Decision Tree Classifier from Scratch - Machine Learning R. Color Diam Label
                                                          Green
                                                                     Apple
                                                         Yellow
                                                                     Apple
                                                                     Grape
                                                          Red
                                                                     Grape
                                                         Yellow
                                                                    Lemon
                                                                 3
                                                                     Is diameter >= 3?
                                                                         G 3 Apple
                                                   R 1 Grape
                                                                         Y 3 Apple
                        Let's Write a Decision
Tree from Scratch
                                                                                    Apple 50%
                                                 Apple 100%
                                                                                    Lemon 50%
```

```
為了要驗證與使用時做的模型,這邊我們自行製作少量的假數據。此數據為類別是水果而特徵有顏色
與半徑兩個。下述的介紹都會基於此製作的假數據進行。
                                                                         特徵
                                                                                      標籤
  1 training_data = [
                                                                     color diameter
                                                                                      label
        ['Red', 1.2, 'Grape'],
        ['Red', 1, 'Grape'],
                                                                     Green
                                                                                     Apple
        ['Yellow', 3.3, 'Lemon'],
        ['Yellow', 3.1, 'Lemon'],
                                                                     Yellow
                                                                                     Apple
        ['Green', 3, 'Apple'],
        ['Red', 1.1, 'Grape'],
                                                                       Red
                                                                                     Grape
 10
        ['Yellow', 3, 'Lemon'],
        ['Red', 1.2, 'Grape'],
 11
                                                                                  1 Grape
 12 ]
                                                                       Red
 13
 14 header = ["color", "diameter", "label"]
                                                                                  3 Lemon
                                                                    Yellow
 15
 16 df = pd.DataFrame(data=training_data, columns=header)
 17 df.head()
    training_data = [
2
        ['Green', 3.1, 'Apple'],
3
        ['Yellow', 3.2, 'Apple'],
4
        ['Red', 1.2, 'Grape'],
裡我們實作數據切分函式。
 # 分割測試集與訓練集
 def train_test_split(df, test_size=0.1):
                                                                          資料集
    if isinstance(test_size, float):
       test_size = round(test_size * len(df))
    #以隨機的方式取的測試集資料點的index
    indices = list(df.index)
    test_indices = random.sample(population=indices, k=test_size)
                                                                   訓練集
                                                                                   驗證集
    #分割測試集與訓練集
    test df - df.loc[test indices]
    train_df = df.drop(test_indices)
```

#檢查是否所有的label都為同一種 unique_classes = np.unique(labels) if len(unique_classes) == 1: return True else: return False # 檢查資料是否都為同一類別 1 2 def check_purity(data): 3 '''Function to check if input data all belong to the same class

if isinstance(test_size, float):

#以隨機的方式取的測試集資料點的index

test_df = df.loc[test_indices]

indices = list(df.index)

#分割測試集與訓練集

test_size = round(test_size * len(df))

test_indices = random.sample(population=indices, k=test_size)

'''Function to get all potential split value for tree base model 4 5 Parameter 6 data: list 8 Input data

```
Input raw pd.DataFrame data
8
9
10
      feature_types = []
11
      #若特徵的獨特值個數較少,及當作類別型特徵資料(若為數值型,獨特值個數應該會很多)
12
取得分割節點
```

節點在分割後,會產生右節點(滿足分割條件)與左節點(不滿足分割條件)

def split_data(data, split_column, split_value): ''Function to splitted left and right nodes

index for feature column

split_value: float or int or string value to be used as split benchmark

split_column_values = data[:, split_column]

#取得用來分割的特徵欄位

#依據欄位值的型態(數值型特徵或類別型特徵)來進行節點分割 type_of_feature = FEATURE_TYPES[split_column]

Parameter data: list

11 12 13

14

15 16

17

18

取得終端節點

Parameters

data: list

Input data task_type: str

#取的資料的label欄位

label_column = data[:, -1]

(在此節點的所有樣本皆被分類為同一個類別)

'''Function to create leaf node

def create_leaf(data, task_type):

Input data split_column: int

#取得用來分割的特徵欄位

19 if type_of_feature == "continuous": #數值型特徵分割 20 21 data_left = data[split_column_values <= split_value]</pre> 22 data_right = data[split_column_values > split_value] 23

#依據欄位值的型態(數值型特徵或類別型特徵)來進行節點分割

split_column_values = data[:, split_column]

type_of_feature = FEATURE_TYPES[split_column]

24 #類別型特徵分割 25 data_left = data[split_column_values == split_value] 26 data_right = data[split_column_values != split_value] 27 28 return data_left, data_right

節點無法再分割時,此節點即成為終端節點(Leaf node),在終端節點需要取得此節點最後分類的類別

indicate the type of tree (regression or classification)

計算熵(Entropy) $Entropy = -\sum_{i=1}^{c} p(i)log_2 p(i)$

n = len(data_below) + len(data_above) 3 p_data_below = len(data_below) / n 5 p_data_above = len(data_above) / n 6 7 overall_metric = (p_data_below * metric_function(data_below) + p_data_above * metric_function(data_above)) 8 9 10

indicate the type of tree (regression or classification) counter: int counter for recording number of splits min_samples: int minimum number of samples for a node to be able to split max_depth: int Maximum depth for the decision tree def __init__(self, metric_function, task_type='classification', counter=0, min_samples=2, max_depth=5): self.metric_function = metric_function self.task_type = task_type self.counter = counter self.min_samples = min_samples self.max_depth = max_depth class decision_tree(): 1 2 '''Decision Tree model 3 Parameters

the metric function used to calculate information gain

在這章節我們學習到了 • 實作與使用決策樹模型 延伸閱讀 網站: 決策樹介紹影片

到以下平台觀看: NouTube

下一步:完成作業