大數據分析 HW1

M11015Q06 顏齊

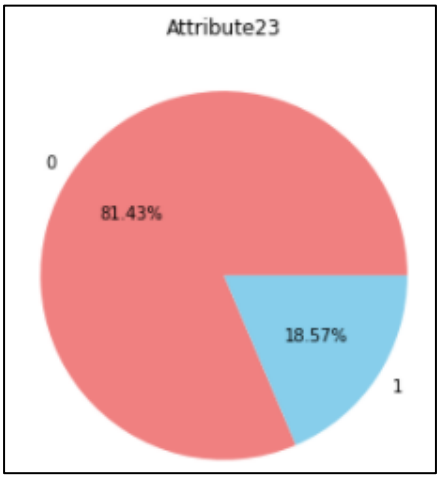
數據理解

1. 首先將有缺失值的資料全部捨棄，保留無缺失資料，利用無缺失資料進行分析，由下圖我們可以發現明天是否下雨(Attribute23)與當天陽光出現時數(Attribute7)最有關係，所以在撰寫程式時我們將以 Attribute7 為重點分析特徵。

```
df_data = df_data.dropna()
df_data.corr()['Attribute23']
```

```
Attribute2    -0.008408
Attribute3     0.080067
Attribute4    -0.134566
Attribute5     0.256521
Attribute6    -0.122929
Attribute7    -0.431394
Attribute8     0.078966
Attribute9     0.229648
Attribute10    0.039686
Attribute11    0.046196
Attribute12    0.063670
Attribute13    0.084580
Attribute14    0.256332
Attribute15    0.413247
Attribute16   -0.236084
Attribute17   -0.215623
Attribute18    0.308651
Attribute19    0.364257
Attribute20   -0.017706
Attribute21   -0.165463
Attribute22    0.306277
Attribute23    1.000000
year          -0.022761
month          0.011603
```

2. 根據 Attribute23 圓餅圖，可以發現 train_data 有數據不平衡的問題，所以在填補缺失值及訓練模型時，我們將調整 scale_pos_weight 參數平衡正負樣本。



程式架構

1. 首先將訓練集和測試集合併(同步處理)，並將類別型資料(風向、明日是否下雨、今日是否下雨)轉換成數字型態。

```
train_data = pd.read_csv("../input/bigdata-hw1/train.csv")#(17093, 23)
test_data = pd.read_csv("../input/bigdata-hw1/test.csv")#(790, 22)
location = dict(N=1, NNE=2, NE=3, ENE=4, E=5, ESE=6, SE=7, SSE=8, S=9, SSW=10, SW=11, WSW=12, W=13, WNW=14, NW=15, NNW=16)
df_data = train_data.append( test_data )
df_data['Attribute22'] = df_data['Attribute22'].map(dict(Yes=1, No=0))#將Attribute22 string換成num
df_data['Attribute23'] = df_data['Attribute23'].map(dict(Yes=1, No=0))#將Attribute23 string換成num
df_data['Attribute8'] = df_data['Attribute8'].map(location)#將string方位換成NUM
df_data['Attribute10'] = df_data['Attribute10'].map(location)#將string方位換成NUM
df_data['Attribute11'] = df_data['Attribute11'].map(location)#將string方位換成NUM
```

2. 透過 Attribute1 年月日期資料新增 Month、Year 特徵

```
df_data['Attribute1'] = df_data['Attribute1'].apply(pd.to_datetime)#將 年月日->月*30+日
df_data['year'] = df_data['Attribute1'].dt.year
df_data['month'] = df_data['Attribute1'].dt.month
```

3. 觀察缺失值，可以發現 Attribute6, 7, 18, 19 缺失值最多，缺失值占比過大的特徵將無法填補，否則會失去原始資料的真實性，所以我們將不使用 Attribute6, 18, 19 特徵訓練模型。

```
df_data.isnull().sum()
```

```
Attribute1       0
Attribute2       0
Attribute3      59
Attribute4      45
Attribute5     158
Attribute6    7300
Attribute7    8088
Attribute8    1102
Attribute9    1098
Attribute10   1186
Attribute11    417
Attribute12    152
Attribute13    285
Attribute14    198
Attribute15    412
Attribute16   1742
Attribute17   1740
Attribute18   6520
Attribute19   6875
Attribute20     96
Attribute21    304
Attribute22    158
Attribute23    790
year             0
month            0
```

4. 此外，因 Attribute7 與目標值關係密切，我們無法在不考慮 Attribute7 的情況下預測目標值，所以我先用 dp_data 資料集訓練 XGBRegressor，再對 Attribute7 進行預測填補。

(Note: dp_data 為訓練集資料除去 Attribute3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21, 22 缺失值的數據)

```
dp_data = df_data[:len(train_data)]
#train_data = train_data.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
dp_data = dp_data[(dp_data['Attribute3'].notna()) & (dp_data['Attribute4'].notna()) & (dp_data['Attribute5'].notna()) & (dp_data['Attribute7'].notna())
                  & (dp_data['Attribute8'].notna()) & (dp_data['Attribute9'].notna()) & (dp_data['Attribute10'].notna())
                   & (dp_data['Attribute11'].notna()) & (dp_data['Attribute12'].notna()) & (dp_data['Attribute13'].notna()) & (dp_data['Attribute14'].notna())
                  & (dp_data['Attribute15'].notna()) & (dp_data['Attribute16'].notna()) & (dp_data['Attribute17'].notna())
                  & (dp_data['Attribute20'].notna()) & (dp_data['Attribute21'].notna()) &(dp_data['Attribute22'].notna())]

predict_7 = ['month','year','Attribute2','Attribute3','Attribute4','Attribute5','Attribute8','Attribute9','Attribute10','Attribute11',
         'Attribute12','Attribute13','Attribute14','Attribute15','Attribute16','Attribute17','Attribute20','Attribute21','Attribute22','Attribute23']

from xgboost import XGBRegressor
model7 = XGBRegressor(scale_pos_weight = 3)

model7.fit(dp_data[predict_7], dp_data['Attribute7'])
```

5. 將步驟 1 合併的資料拆分數據集、測試集，再對訓練集缺失的 Attribute7 進行填補。

```
train_data = df_data[:len(train_data)]
test_data = df_data[len(train_data):len(train_data) + len(test_data)]
#train_data = train_data.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
df_data = train_data[(train_data['Attribute3'].notna()) & (train_data['Attribute4'].notna()) & (train_data['Attribute5'].notna())
                     & (train_data['Attribute8'].notna()) & (train_data['Attribute9'].notna()) & (train_data['Attribute10'].notna())
                     & (train_data['Attribute11'].notna()) & (train_data['Attribute12'].notna()) & (train_data['Attribute13'].notna()) & (train_data['Attribute14'].notna())
                     & (train_data['Attribute15'].notna()) & (train_data['Attribute16'].notna()) & (train_data['Attribute17'].notna())
                     & (train_data['Attribute20'].notna()) & (train_data['Attribute21'].notna()) &(train_data['Attribute22'].notna())]

pred7 = model7.predict(df_data[df_data['Attribute7'].isna()][predict_7])
index7 = df_data[df_data['Attribute7'].isna()].index
train_data.iloc[index7,6:7] = pred7
train_data = train_data[(train_data['Attribute3'].notna()) & (train_data['Attribute4'].notna()) & (train_data['Attribute5'].notna())
                     & (train_data['Attribute8'].notna()) & (train_data['Attribute9'].notna()) & (train_data['Attribute10'].notna())
                     & (train_data['Attribute11'].notna()) & (train_data['Attribute12'].notna()) & (train_data['Attribute13'].notna()) & (train_data['Attribute14'].notna())
                     & (train_data['Attribute15'].notna()) & (train_data['Attribute16'].notna()) & (train_data['Attribute17'].notna())
                     & (train_data['Attribute20'].notna()) & (train_data['Attribute21'].notna()) &(train_data['Attribute22'].notna())]
```

6. 最後，使用數據集訓練 XGBClassifier 並預測測試集目標值。

```
from xgboost import XGBClassifier
predict_final = ['Attribute2','Attribute3','Attribute4','Attribute5','Attribute6','Attribute7','Attribute8','Attribute9','Attribute10','Attribute11',
          'Attribute12','Attribute13','Attribute14','Attribute15','Attribute16','Attribute17','Attribute20','Attribute21','Attribute22','year','month']
XGBC = XGBClassifier(learning_rate=0.1,scale_pos_weight = 5,n_estimators = 28)
XGBC.fit(train_data[predict_final],train_data["Attribute23"])

pred_XGBC = XGBC.predict(test_data[predict_final])
pred_XGBC = pd.DataFrame(pred_XGBC)
pred_XGBC = pred_XGBC.astype(int)
id = pd.DataFrame(test_data.index)
id = id.astype(float)
submission = pd.DataFrame({
                           "id": id[0],
                           "ans": pred_XGBC[0]
                          })
submission.to_csv('submission.csv', index=False)
```

7. 交叉驗證

```
from sklearn.model_selection import train_test_split, cross_val_score, cross_val_predict, cross_validate, GridSearchCV
from sklearn.metrics import accuracy_score, balanced_accuracy_score
dict_models = {
'XGBClassifier': XGBClassifier(learning_rate=0.1,scale_pos_weight = 5,n_estimators = 28),
}

for name, model in dict_models.items():
    cv_predict = cross_val_predict(model,train_data[predict_final],train_data["Attribute23"], cv=10)
    accuracy = round(accuracy_score(train_data["Attribute23"], cv_predict), 3)
    balanced_accuracy = round(balanced_accuracy_score(train_data["Attribute23"], cv_predict), 3)
    print(f'model:{name}, accuracy:{accuracy}, balanced_accuracy:{balanced_accuracy}')
```

```
model:XGBClassifier, accuracy:0.798, balanced_accuracy:0.796
```