



Using Machine Learning Models

数据科学 –
机器学习模型入门

人类活动预测

**Random Forest with
Classification & Clustering**

Dec 2020

Microsoft Reactor | Ryan Chung

```
led by player to  
s.load_image("kg.png")  
(self):  
    initialize Dog object and create Text of  
g, self).__init__(image = Dog.image,  
                    x = games.mouse.x,  
                    bottom = games.screen  
re = games.Text(value = 0, size = 24,  
                  top = 5, right = game  
reen.add(self.score)  
1 = games.Text(value = 0, size = 24,  
                 top = 5, left = game
```



Ryan Chung

Instructor / DevelopIntelligence
Founder / MobileDev.TW

@ryanchung403 on WeChat
Ryan@MobileDev.TW





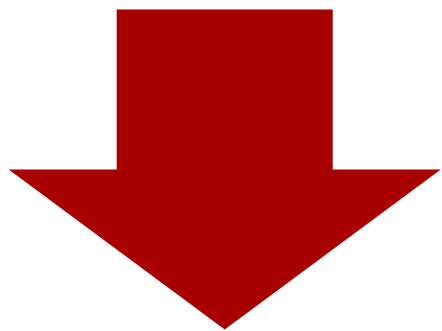
Reactor



developer.microsoft.com/reactor/
@MSFTReactor on Twitter

手机感测器数据 => 当前行为动作

- Accelerometer 加速度计
- Gyroscope 陀螺仪



1. Walking 行走
2. Walking Upstairs 上楼
3. Walking Downstairs 下楼
4. Sitting 坐
5. Standing 站立
6. Laying 躺着



分析步骤



分析步骤



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.linear_model import LogisticRegression, LogisticRegressionCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import precision_recall_fscore_support as error_metric
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, f1_score, recall_score
from sklearn.feature_selection import VarianceThreshold
```

分析步骤



```
train = pd.read_csv("Data/train.csv")
test = pd.read_csv("Data/test.csv")
```

train.head()

| | tBodyAcc-mean()-X | tBodyAcc-mean()-Y | tBodyAcc-mean()-Z | tBodyAcc-std()-X | tBodyAcc-std()-Y | tBodyAcc-std()-Z | tBodyAcc-mad()-X | tBodyAcc-mad()-Y | tBodyAcc-mad()-Z | tBodyAcc-max()-X | ... |
|---|-------------------|-------------------|-------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|-----|
| 0 | 0.288585 | -0.020294 | -0.132905 | -0.995279 | -0.983111 | -0.913526 | -0.995112 | -0.983185 | -0.923527 | -0.934724 | ... |
| 1 | 0.278419 | -0.016411 | -0.123520 | -0.998245 | -0.975300 | -0.960322 | -0.998807 | -0.974914 | -0.957686 | -0.943068 | ... |
| 2 | 0.279653 | -0.019467 | -0.113462 | -0.995380 | -0.967187 | -0.978944 | -0.996520 | -0.963668 | -0.977469 | -0.938692 | ... |
| 3 | 0.279174 | -0.026201 | -0.123283 | -0.996091 | -0.983403 | -0.990675 | -0.997099 | -0.982750 | -0.989302 | -0.938692 | ... |
| 4 | 0.276629 | -0.016570 | -0.115362 | -0.998139 | -0.980817 | -0.990482 | -0.998321 | -0.979672 | -0.990441 | -0.942469 | ... |

5 rows x 563 columns

| angle(X,gravityMean) | angle(Y,gravityMean) | angle(Z,gravityMean) | subject | Activity |
|----------------------|----------------------|----------------------|---------|----------|
| -0.841247 | 0.179941 | -0.058627 | 1 | STANDING |
| -0.844788 | 0.180289 | -0.054317 | 1 | STANDING |
| -0.848933 | 0.180637 | -0.049118 | 1 | STANDING |
| -0.848649 | 0.181935 | -0.047663 | 1 | STANDING |
| -0.847865 | 0.185151 | -0.043892 | 1 | STANDING |

遗漏值确认



```
train.isnull().sum()
```

```
test.isnull().sum()
```

```
train.isnull().values.any()
```

```
test.isnull().values.any()
```

```
[26] train.isnull().sum()
```



| | |
|---------------------------|---|
| tBodyAcc-mean()-X | 0 |
| tBodyAcc-mean()-Y | 0 |
| tBodyAcc-mean()-Z | 0 |
| tBodyAcc-std()-X | 0 |
| tBodyAcc-std()-Y | 0 |
| .. | |
| angle(X,gravityMean) | 0 |
| angle(Y,gravityMean) | 0 |
| angle(Z,gravityMean) | 0 |
| subject | 0 |
| Activity | 0 |
| Length: 563, dtype: int64 | |

```
[27] test.isnull().sum()
```



| | |
|---------------------------|---|
| tBodyAcc-mean()-X | 0 |
| tBodyAcc-mean()-Y | 0 |
| tBodyAcc-mean()-Z | 0 |
| tBodyAcc-std()-X | 0 |
| tBodyAcc-std()-Y | 0 |
| .. | |
| angle(X,gravityMean) | 0 |
| angle(Y,gravityMean) | 0 |
| angle(Z,gravityMean) | 0 |
| subject | 0 |
| Activity | 0 |
| Length: 563, dtype: int64 | |

```
[28] train.isnull().values.any()
```



```
False
```

```
[29] test.isnull().values.any()
```



```
False
```


整体观察



`train.info()`

`test.info()`

```
[30] train.info()
```



```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 7352 entries, 0 to 7351  
Columns: 563 entries, tBodyAcc-mean()-X to Activity  
dtypes: float64(561), int64(1), object(1)  
memory usage: 31.6+ MB
```

```
[31] test.info()
```



```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 2947 entries, 0 to 2946  
Columns: 563 entries, tBodyAcc-mean()-X to Activity  
dtypes: float64(561), int64(1), object(1)  
memory usage: 12.7+ MB
```

数据分析 – 移除可能不相关的栏位



- subject : 区分受测者的id

```
train.drop('subject',axis=1, inplace=True)
```

```
test.drop('subject',axis=1, inplace=True)
```

```
[36] train.head()
```



| oJerkMean,gravityMean) | angle(X,gravityMean) | angle(Y,gravityMean) | angle(Z,gravityMean) | Activity |
|------------------------|----------------------|----------------------|----------------------|----------|
| -0.018446 | -0.841247 | 0.179941 | -0.058627 | STANDING |
| 0.703511 | -0.844788 | 0.180289 | -0.054317 | STANDING |
| 0.808529 | -0.848933 | 0.180637 | -0.049118 | STANDING |
| -0.485366 | -0.848649 | 0.181935 | -0.047663 | STANDING |
| -0.615971 | -0.847865 | 0.185151 | -0.043892 | STANDING |

数据分析 – 分析所有栏位



- 将栏位名称另存一个List

```
rem_cols2 = train.columns.tolist()
```

```
[45] len(rem_cols2)
```



562

```
[42] rem_cols2
```



```
['tBodyAcc-mean()-X',  
'tBodyAcc-mean()-Y',  
'tBodyAcc-mean()-Z',  
'tBodyAcc-std()-X',  
'tBodyAcc-std()-Y',  
'tBodyAcc-std()-Z',  
'tBodyAcc-mad()-X',  
'tBodyAcc-mad()-Y',  
'tBodyAcc-mad()-Z',  
'tBodyAcc-max()-X',  
'tBodyAcc-max()-Y',  
'tBodyAcc-max()-Z',  
'tBodyAcc-min()-X',  
'tBodyAcc-min()-Y',
```

数据分析 – 分析所有栏位



- 观察目前各栏位的资料型态

```
[46] train.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7352 entries, 0 to 7351
Columns: 562 entries, tBodyAcc-mean()-X to Activity
dtypes: float64(561), object(1)
memory usage: 31.5+ MB
```

- 唯一的 object 就是目标值

```
[47] train['Activity'][0]
'STANDING'
```

```
[48] type(train['Activity'][0])
str
```

数据分析 – 分析所有栏位



- 另一个找到object的方式

```
is_object_type_feature = train.dtypes == np.object  
train.columns[is_object_type_feature]
```

```
[67] is_object_type_feature
```

```
✖  
tBodyAcc-mean()-X      False  
tBodyAcc-mean()-Y      False  
tBodyAcc-mean()-Z      False  
tBodyAcc-std()-X       False  
tBodyAcc-std()-Y       False  
...  
angle(tBodyGyroJerkMean,gravityMean) False  
angle(X,gravityMean)    False  
angle(Y,gravityMean)    False  
angle(Z,gravityMean)    False  
Activity                True  
Length: 562, dtype: bool
```

```
[68] train.columns
```

```
✖  
Index(['tBodyAcc-mean()-X', 'tBodyAcc-mean()-Y', 'tBodyAcc-mean()-Z',  
      'tBodyAcc-std()-X', 'tBodyAcc-std()-Y', 'tBodyAcc-std()-Z',  
      'tBodyAcc-mad()-X', 'tBodyAcc-mad()-Y', 'tBodyAcc-mad()-Z',  
      'tBodyAcc-max()-X',  
      ...  
      'fBodyBodyGyroJerkMag-skewness()', 'fBodyBodyGyroJerkMag-kurtosis()',  
      'angle(tBodyAccMean,gravity)', 'angle(tBodyAccJerkMean,gravityMean)',  
      'angle(tBodyGyroMean,gravityMean)',  
      'angle(tBodyGyroJerkMean,gravityMean)', 'angle(X,gravityMean)',  
      'angle(Y,gravityMean)', 'angle(Z,gravityMean)', 'Activity'],  
      dtype='object', length=562)
```

```
[69] train.columns[is_object_type_feature]
```

```
✖  
Index(['Activity'], dtype='object')
```


数据分析 – 分析所有栏位



- Activity的数值分布

```
train['Activity'].value_counts()
```

- 也可以写成

```
train.Activity.value_counts()
```

```
[72] train['Activity'].value_counts()
```



| | |
|------------------------------|------|
| LAYING | 1407 |
| STANDING | 1374 |
| SITTING | 1286 |
| WALKING | 1226 |
| WALKING_UPSTAIRS | 1073 |
| WALKING_DOWNSTAIRS | 986 |
| Name: Activity, dtype: int64 | |

```
[73] train.Activity.value_counts()
```



| | |
|------------------------------|------|
| LAYING | 1407 |
| STANDING | 1374 |
| SITTING | 1286 |
| WALKING | 1226 |
| WALKING_UPSTAIRS | 1073 |
| WALKING_DOWNSTAIRS | 986 |
| Name: Activity, dtype: int64 | |

数据分析 – 分析所有栏位



- 将Activity转换成数值型式

```
le = LabelEncoder()
for x in [train, test]:
    x['Activity'] = le.fit_transform(x['Activity'])
```

```
[77] train['Activity'].value_counts()
```



```
0    1407
2    1374
1    1286
3    1226
5    1073
4     986
```

```
Name: Activity, dtype: int64
```

数据分析 – 分析所有栏位



• 关联性观察

```
corr_val = train.corr()  
corr_val_activity_abs = corr_val['Activity'].abs()  
corr_val_activity_abs_sort = corr_val_activity_abs.sort_values(ascending=False)  
corr_val_activity_abs_sort[corr_val_activity_abs_sort>0.84]
```

```
[109] corr_val_activity_abs_sort[corr_val_activity_abs_sort>0.83]
```



×

| | |
|---------------------------|----------|
| Activity | 1.000000 |
| fBodyAccJerk-entropy()-X | 0.845190 |
| tBodyGyroJerk-entropy()-Z | 0.844754 |
| tBodyAccJerk-entropy()-Y | 0.837034 |
| tBodyAcc-sma() | 0.835621 |
| tBodyAccJerkMag-entropy() | 0.835376 |
| tGravityAccMag-sma() | 0.833126 |
| tBodyAccMag-sma() | 0.833126 |
| tBodyAccMag-mean() | 0.833126 |
| tGravityAccMag-mean() | 0.833126 |
| fBodyAccJerk-entropy()-Y | 0.832284 |
| fBodyAccMag-entropy() | 0.831222 |
| fBodyAcc-entropy()-Y | 0.831114 |
| fBodyAcc-std()-Y | 0.830052 |

Name: Activity, dtype: float64

```
[110] corr_val_activity_abs_sort[corr_val_activity_abs_sort>0.84]
```



×

| | |
|---------------------------|----------|
| Activity | 1.000000 |
| fBodyAccJerk-entropy()-X | 0.845190 |
| tBodyGyroJerk-entropy()-Z | 0.844754 |

Name: Activity, dtype: float64

数据分析 – 基本整理原则



- 无遗漏值
- 剔除明显不相关栏位
- 全面数值化
- 观察数值分布情况，必要时进行转换
 - Normalization：往常态分布迈进
 - Standardization：使用相同的尺规
- 关联性分析

分析步骤



```
feature_cols = train.columns[:-1]
split_data = StratifiedShuffleSplit(n_splits=3, test_size=0.3, random_state=42)
train_idx, val_idx = next(split_data.split(train[feature_cols], train.Activity))
```

```
X_train = train.loc[train_idx, feature_cols]
y_train = train.loc[train_idx, 'Activity']
```

```
X_val = train.loc[val_idx, feature_cols]
y_val = train.loc[val_idx, 'Activity']
```

| # | F1 | F2 | F3 | Target |
|---|---------|----|----|---------|
| | X_train | | | y_train |
| | X_val | | | y_val |
| | X_train | | | y_train |
| | X_train | | | y_train |
| | X_train | | | y_train |
| | X_val | | | y_val |
| | X_train | | | y_train |
| | X_val | | | y_val |
| | X_train | | | y_train |

分析步骤



```
feature_cols = train.columns[:-1]
split_data = StratifiedShuffleSplit(n_splits=3, test_size=0.3, random_state=42)
train_idx, val_idx = next(split_data.split(train[feature_cols], train.Activity))
```

```
X_train = train.loc[train_idx, feature_cols]
y_train = train.loc[train_idx, 'Activity']
```

```
X_val = train.loc[val_idx, feature_cols]
y_val = train.loc[val_idx, 'Activity']
```

```
y_train.value_counts()
y_train.value_counts(normalize=True)
y_val.value_counts(normalize=True)
```

```
[122] y_train.value_counts()
0    985
2    962
1    900
3    858
5    751
4    690
Name: Activity, dtype: int64
```

```
[126] y_val.value_counts(normalize=True)
0    0.191296
2    0.186763
1    0.174977
3    0.166818
5    0.145966
4    0.134180
Name: Activity, dtype: float64
```

```
[123] y_train.value_counts(normalize=True)
0    0.191411
2    0.186941
1    0.174893
3    0.166731
5    0.145939
4    0.134085
Name: Activity, dtype: float64
```

分析步骤



```
lr = LogisticRegression()
lr_l2 = LogisticRegressionCV()
rf = RandomForestClassifier()

lr_model = lr.fit(X_train, y_train)
lr_l2_model = lr_l2.fit(X_train, y_train)
rf_model = rf.fit(X_train, y_train)
```

```
lr_model_predict = lr_model.predict(X_val)
lr_l2_model_predict = lr_l2_model.predict(X_val)
rf_model_predict = rf_model.predict(X_val)
```

```
three_model_predict_df = pd.DataFrame({'lr':lr_model_predict,
'lr_l2':lr_l2_model_predict,'rf':rf_model_predict})
```

| | lr | lr_l2 | rf |
|------|------|-------|-----------|
| 0 | 0 | 0 | 0 |
| 1 | 5 | 5 | 5 |
| 2 | 1 | 1 | 1 |
| 3 | 0 | 0 | 0 |
| 4 | 3 | 3 | 3 |
| ... | ... | ... | ... |
| 2201 | 1 | 1 | 1 |
| 2202 | 5 | 5 | 5 |
| 2203 | 2 | 2 | 2 |
| 2204 | 1 | 1 | 1 |
| 2205 | 3 | 3 | 3 |
| 2206 | rows | x | 3 columns |

分析步骤



- 三个模型的信心指数也可输出进行观察

```
lr_model_proba = lr_model.predict_proba(X_val).max(axis=1)
lr_l2_model_proba = lr_l2_model.predict_proba(X_val).max(axis=1)
rf_model_proba = rf_model.predict_proba(X_val).max(axis=1)
```

```
three_model_proba_df = pd.DataFrame({'lr':lr_model_proba,
'lr_l2':lr_l2_model_proba, 'rf':rf_model_proba})
```

| | lr | lr_l2 | rf |
|------|------------------|----------|-----|
| 0 | 0.982767 | 0.999938 | 1.0 |
| 1 | 0.977773 | 0.999258 | 1.0 |
| 2 | 0.995116 | 0.998465 | 0.8 |
| 3 | 0.999999 | 1.000000 | 0.9 |
| 4 | 0.962265 | 0.998696 | 0.9 |
| ... | ... | ... | ... |
| 2201 | 0.998039 | 0.999863 | 1.0 |
| 2202 | 0.997980 | 0.999993 | 0.9 |
| 2203 | 0.992126 | 0.999752 | 1.0 |
| 2204 | 0.978954 | 0.989888 | 0.9 |
| 2205 | 0.991474 | 0.999875 | 1.0 |
| 2206 | rows x 3 columns | | |

分析步骤



• 三个模型的评量综整

```
accuracy_lr = accuracy_score(y_val, lr_model_predict)
accuracy_lr_12 = accuracy_score(y_val, lr_12_model_predict)
accuracy_rf = accuracy_score(y_val, rf_model_predict)
```

```
three_model_report = pd.DataFrame(data={'Accuracy': [accuracy_lr,
accuracy_lr_12, accuracy_rf]}, index=['lr', 'lr_12', 'rf'])
```

| Accuracy | |
|----------|----------|
| lr | 0.982774 |
| lr_12 | 0.986401 |
| rf | 0.966002 |

分析步骤



• 加上其他指标 - Precision

```
precision_lr = precision_score(y_val, lr_model_predict, average='weighted')
precision_lr_l2 = precision_score(y_val, lr_l2_model_predict, average='weighted')
precision_rf = precision_score(y_val, rf_model_predict, average='weighted')
```

```
three_model_report['Precision'] = [precision_lr, precision_lr_l2, precision_rf]
```

| | Accuracy | Precision |
|-------|----------|-----------|
| lr | 0.982774 | 0.982787 |
| lr_l2 | 0.986401 | 0.986408 |
| rf | 0.966002 | 0.966252 |

分析步骤



• 加上其他指标 - Recall

```
recall_lr = recall_score(y_val, lr_model_predict, average='weighted')
recall_lr_l2 = recall_score(y_val, lr_l2_model_predict, average='weighted')
recall_rf = recall_score(y_val, rf_model_predict, average='weighted')
```

```
three_model_report['Recall'] = [recall_lr, recall_lr_l2, recall_rf]
```

| | Accuracy | Precision | Recall |
|-------|----------|-----------|----------|
| lr | 0.982774 | 0.982787 | 0.982774 |
| lr_l2 | 0.986401 | 0.986408 | 0.986401 |
| rf | 0.966002 | 0.966252 | 0.966002 |

分析步骤



- 加上其他指标 – F1 Score

```
f1score_lr = f1_score(y_val, lr_model_predict, average='weighted')  
f1score_lr_l2 = f1_score(y_val, lr_l2_model_predict, average='weighted')  
f1score_rf = f1_score(y_val, rf_model_predict, average='weighted')
```

```
three_model_report['F1 Score'] = [f1score_lr, f1score_lr_l2, f1score_rf]
```

| | Accuracy | Precision | Recall | F1 Score |
|-------|----------|-----------|----------|----------|
| lr | 0.982774 | 0.982787 | 0.982774 | 0.982771 |
| lr_l2 | 0.986401 | 0.986408 | 0.986401 | 0.986399 |
| rf | 0.966002 | 0.966252 | 0.966002 | 0.966006 |



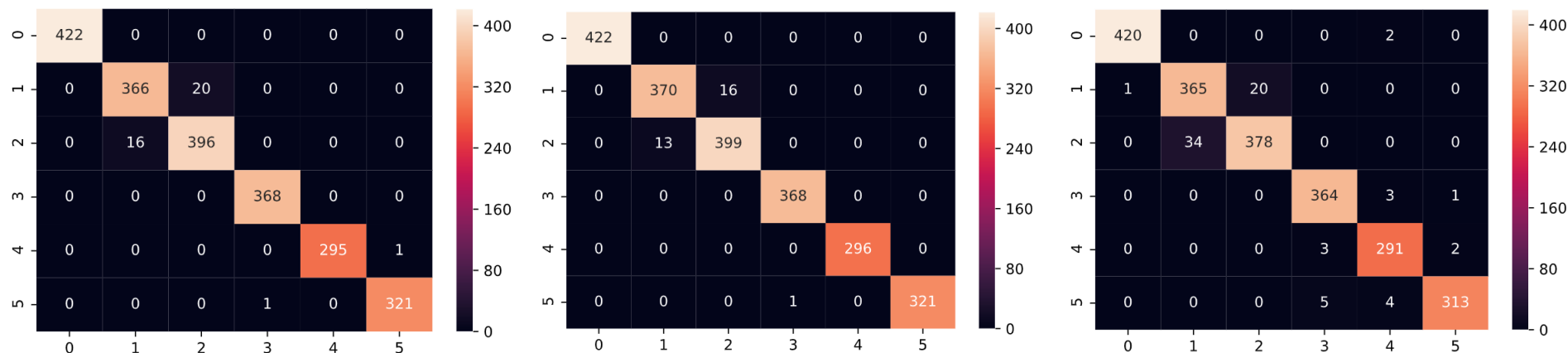
分析步骤



• 加上其他指标 – Confusion Matrix

```
cm_lr = confusion_matrix(y_val, lr_model_predict)
cm_lr_l2 = confusion_matrix(y_val, lr_l2_model_predict)
cm_rf = confusion_matrix(y_val, rf_model_predict)
sns.heatmap(cm_lr, annot=True, fmt="d")
sns.heatmap(cm_lr_l2, annot=True, fmt="d")
sns.heatmap(cm_rf, annot=True, fmt="d")
```

annot 是否标示数值
fmt 数值标示格式(d为整数)



小结

- 依数据的分布情形与范围，考量是否需要Normalization与Standardization
- 数据集的训练与测试切割，须同时考量目标分类的平均取样
- 综合比较各种评量指标的结果，了解不同模型的表现





Reactor



developer.microsoft.com/reactor/
@MSFTReactor on Twitter

议程结束 感谢聆听



请记得填写课程回馈问卷 (Event ID : **11888**)
<https://aka.ms/Reactor/Survey>

© 2019 Microsoft Corporation. All rights reserved. The text in this document is available under the Creative Commons Attribution 3.0 License, additional terms may apply. All other content contained in this document (including, without limitation, trademarks, logos, images, etc.) are not included within the Creative Commons license grant. This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it. Some examples are for illustration only and are fictitious. No real association is intended or inferred. Microsoft makes no warranties, express or implied, with respect to the information provided here.