Sktime

組員:黃子軒
張立勳
洪銘毅

目錄

相關套件安裝

- 所需要的相關套件:
  - Sktime
  - pmdarima
  - tsfresh
  - matplotlib
  - seaborn(*)
  - statsmodels
  - numpy
- 安裝方法
  - 首先, 須要使用管理者身分執行CMD
  - 接著在CMD上打"pip install Sktime" 以及上述其他套件

相關套件出錯解決方法

- 由於Sktime是個蠻新的package，所以安裝的時候可能會遇到版本不相容之類的問題，因此我們列出幾個我們安裝時有遇到的錯誤給大家參考。
- statsmodels，若是顯示版本出錯，建議uninstall後重新安裝。
- Sktime，若是安裝後與python版本不相容，可以安裝前一版。
- Seaborn，若是執行我們的sample code有問題的話，這個package才需要安裝

資料介紹

- 在接下來的介紹中，我們將使用sktime裡面的airline這筆資料來介紹相關的模型
- 資料第一欄為幾年幾月，第二欄為飛機上乘客的人數，總比數為144筆

Scikit-learn API

- scikit-learn提供元估計器 (meta-estimator)，有下列幾個特點
  - 將模組化與scikit-learn兼容，可使用sciki-learn的regressor來預測
  - 允許我們調整更進階的參數，像是決策方法...
  - 從某種意義上講，它可以使scikit-learn的estimator界面適應預測者的界面，從而確保我們可以調整和正確評估模型

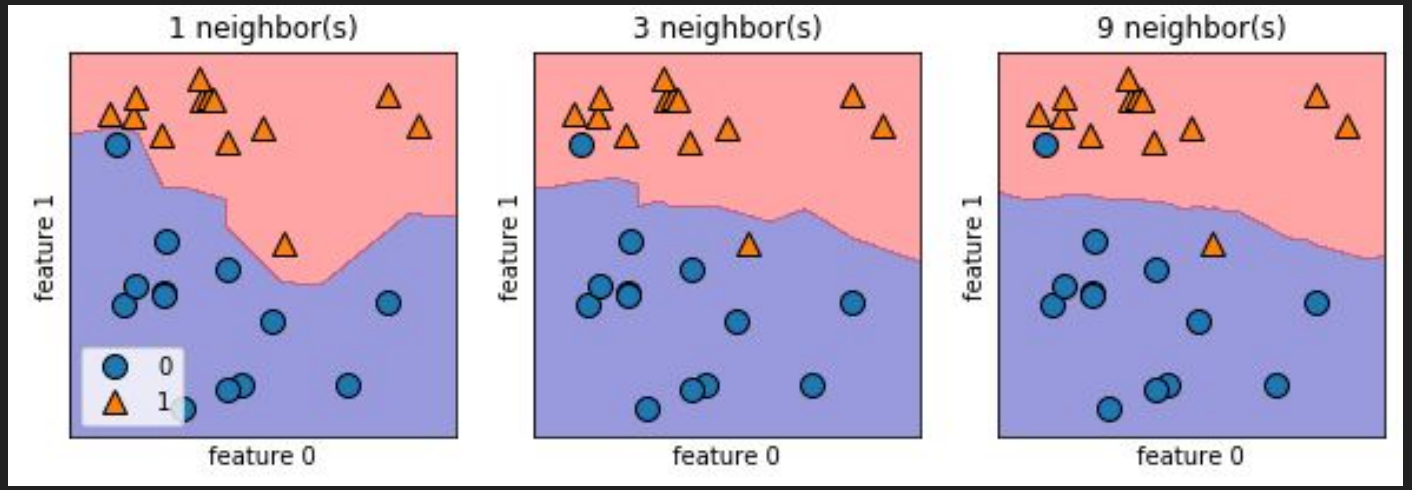
NaiveForecaster

- 最簡單且較為基本的預測方法
- Naive Method 是將後方預測值完全等於前些資料的觀察值
- 其中若有季節趨勢可將其加入參數 (sp)

ARIMA

- 常見處理時間資料的模型假設
- 一樣有季節趨勢可將其加入參數 (SARIMA)
- 如同R裡的auto.arima (自動配適p,d,q)

KNeighborsRegressor



ExponentialSmoothing

- $S_t = a \cdot y_t + (1 - a) S_{t-1}$
- St : 時間t的平滑值

  yt : 時間t的實際值

  St-1 : 時間t-1的平滑值

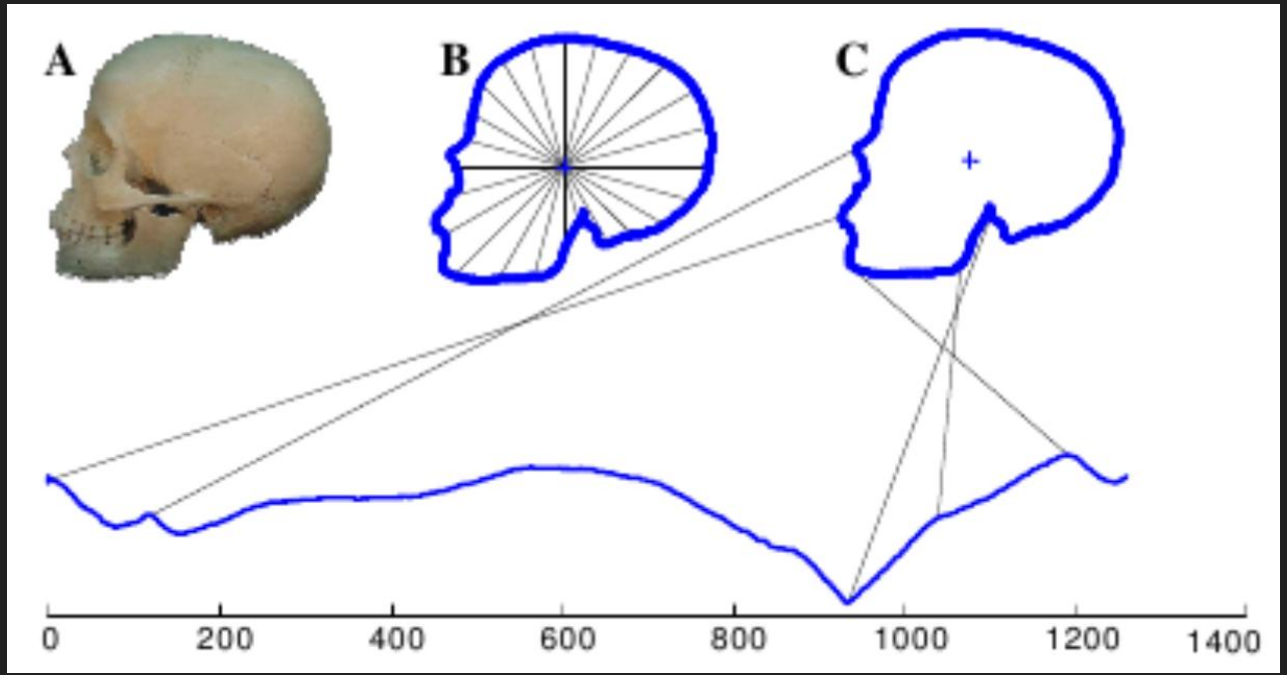
  a : 平滑指數, 其取值範圍為[0,1]

GridSearchCV

- 暴力搜尋
- 自動調參
- 根據資料量可能花費時間也較多

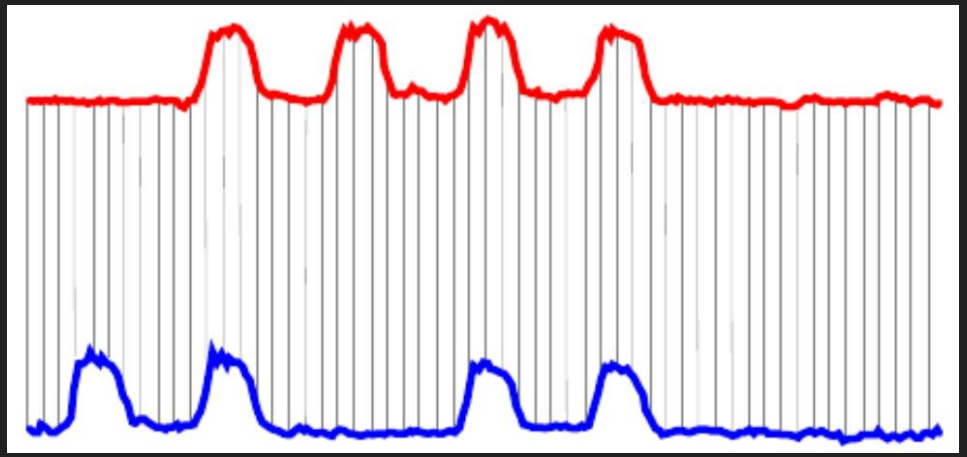
Time Series Classification

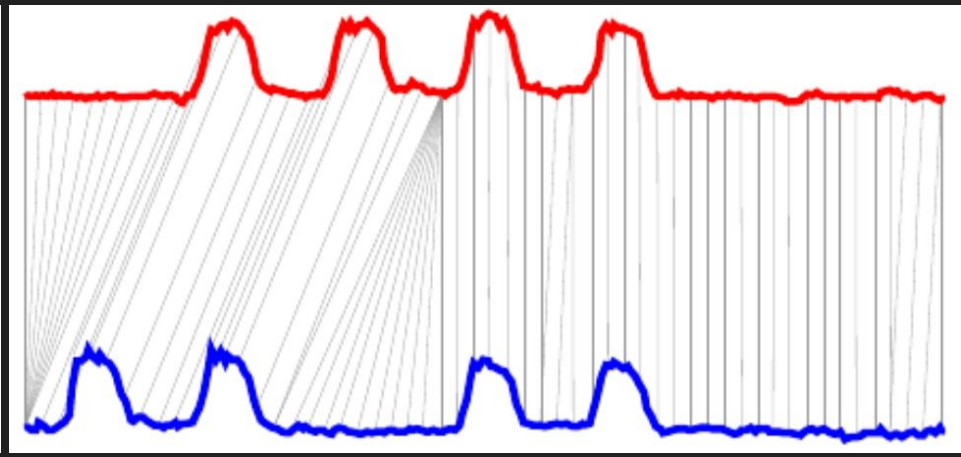
- Creating a time series by radial scanning



Comparing the distances

- Euclidean distance v.s. Dynamic time warping (DTW)

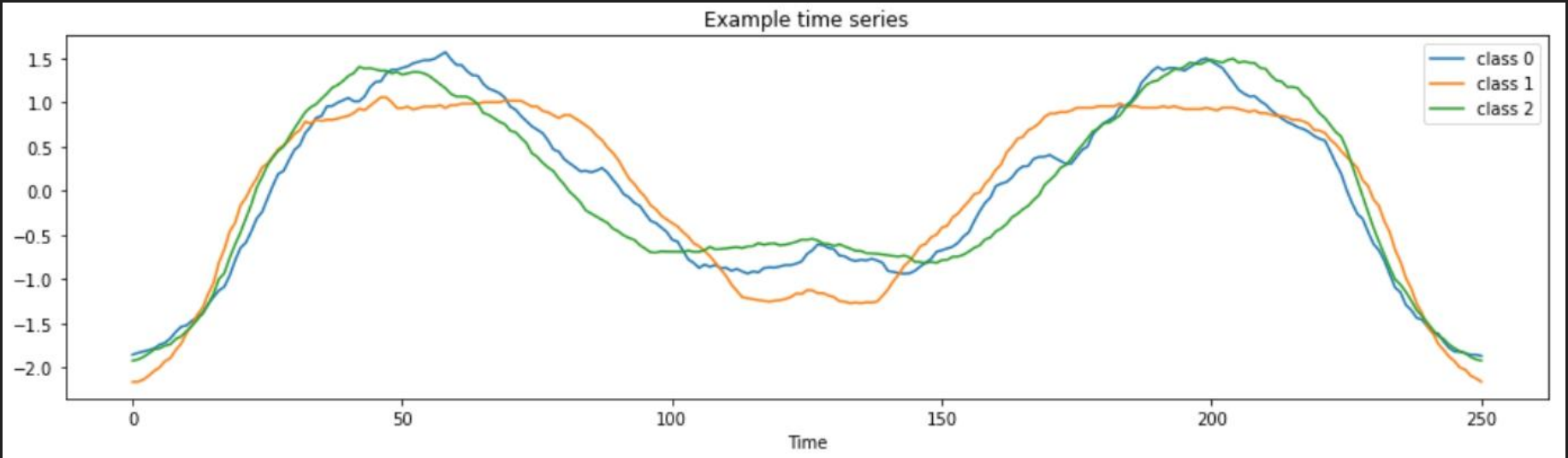




Ex: Arrow Head Classification

```
X_train.head()
```

	dim_0
20	0 -1.9226 1 -1.9080 2 -1.8801 3 ...
25	0 -1.8541 1 -1.8339 2 -1.8187 3 ...
10	0 -2.0336 1 -2.0052 2 -1.9754 3 ...
107	0 -2.1620 1 -2.1601 2 -2.1365 3 ...
145	0 -1.7427 1 -1.7399 2 -1.7102 3 ...


Example time series
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
50
100
150
200
250
Time
class 0
class 1
class 2

Time Series Classifier in Sktime

- TimeSeriesForestClassifier
  - Extract features
- KNeighborsTimeSeriesClassifier

Extract features

- TSFreshFeatureExtractor()

```
transformer = TSFreshFeatureExtractor(default_fc_parameters="efficient")
```

  - we use TSFreshFeatureExtractor() for automatic feature extraction
  - default_fc_parameters = “minimal”, “comprehensive”, “efficient”

RandomIntervalFeatureExtractor()

```
features = [np.mean, np.std, time_series_slope]
transformer = RandomIntervalFeatureExtractor(features=features, n_intervals = "sqrt")
```

- It segments time series into random intervals and subsequently extracts features from each interval
- n_intervals determines the number of intervals

TimeSeriesForestClassifier()

```
classifier = TimeSeriesForestClassifier(
    n_estimators = 500,
    criterion = "entropy",
    max_depth = 5)
```

- Build a random forest model with time series setting
- n_estimators determines the number of trees in the forest
- criterion determines the function to measure the quality of a split
- max_depth determines the maximum depth of the tree

Time series forest on extracted features

- Using Pipeline() to connect feature extractor function and classifier function

```
from sklearn.pipeline import Pipeline
features = [np.mean, np.std, time_series_slope]
steps = [('transform', RandomIntervalFeatureExtractor(features=features)),
         ('clf', DecisionTreeClassifier())]
estimator = Pipeline(steps)
classifier = TimeSeriesForestClassifier(estimator = estimator)
```

- DecisionTreeClassifier() must be added!!!

KNeighborsTimeSeriesClassifier()

```
from sktime.classification.distance_based import KNeighborsTimeSeriesClassifier
classifier = KNeighborsTimeSeriesClassifier(n_neighbors = 5, metric="dtw")
```

- Build a KNN model with time series setting
- n_neighbors determines the number of neighbors to use
- metric determines distance measure for time series

- ```
  classifier.fit(X_train, y_train)
  ```
  - Train the classifier
- ```
  classifier.predict(X_test)
  ```
  - Using this classifier to predict the result
- ```
  from sklearn.metrics import accuracy_score
  accuracy_score(y_test, y_pred)
  ```
  - Calaulate accuracy
  ```