R + Python via reticulate in R

Nitish Kumar Mishra

22 October, 2019

R Markdown

This is an R Markdown for the R and Python by following this URL https://www.business-science.io/business/2018/10/08/python-and-r.html.

I used both python and R chunks and parse output in each other:

```
library(reticulate)
conda_list() # Use this command to get list of conda environment, and select python from the list.
```

```
##        name
## 1 anaconda3
##                                                                          python
## 1 C:\\Users\\nitish.mishra\\AppData\\Local\\Continuum\\anaconda3\\python.exe
```

```
use_condaenv("anaconda3")
setwd("D:/OneDrive - University of Nebraska Medical Center/Plot from Twitter/")
#summary(cars)
```

```
import numpy as np
import pandas as pd

dataset_url = 'http://mlr.cs.umass.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv'
data = pd.read_csv(dataset_url, sep = ";")
print(data.head())
```

```
##    fixed acidity  volatile acidity  citric acid  ...  sulphates  alcohol  quality
## 0            7.4              0.70         0.00  ...       0.56      9.4        5
## 1            7.8              0.88         0.00  ...       0.68      9.8        5
## 2            7.8              0.76         0.04  ...       0.65      9.8        5
## 3           11.2              0.28         0.56  ...       0.58      9.8        6
## 4            7.4              0.70         0.00  ...       0.56      9.4        5
##
## [5 rows x 12 columns]
```

```
library(tidyverse)
summary(py$data)
```

```
##  fixed acidity   volatile acidity  citric acid    residual sugar
##  Min.   : 4.60   Min.   :0.1200   Min.   :0.000   Min.   : 0.900
##  1st Qu.: 7.10   1st Qu.:0.3900   1st Qu.:0.090   1st Qu.: 1.900
##  Median : 7.90   Median :0.5200   Median :0.260   Median : 2.200
##  Mean   : 8.32   Mean   :0.5278   Mean   :0.271   Mean   : 2.539
##  3rd Qu.: 9.20   3rd Qu.:0.6400   3rd Qu.:0.420   3rd Qu.: 2.600
##  Max.   :15.90   Max.   :1.5800   Max.   :1.000   Max.   :15.500
##    chlorides       free sulfur dioxide total sulfur dioxide
##  Min.   :0.01200   Min.   : 1.00       Min.   :  6.00
##  1st Qu.:0.07000   1st Qu.: 7.00       1st Qu.: 22.00
##  Median :0.07900   Median :14.00       Median : 38.00
##  Mean   :0.08747   Mean   :15.87       Mean   : 46.47
```

```
##   3rd Qu.:0.09000   3rd Qu.:21.00       3rd Qu.: 62.00
##   Max.   :0.61100   Max.   :72.00       Max.   :289.00
##      density             pH           sulphates         alcohol
##   Min.   :0.9901   Min.   :2.740   Min.   :0.3300   Min.   : 8.40
##   1st Qu.:0.9956   1st Qu.:3.210   1st Qu.:0.5500   1st Qu.: 9.50
##   Median :0.9968   Median :3.310   Median :0.6200   Median :10.20
##   Mean   :0.9967   Mean   :3.311   Mean   :0.6581   Mean   :10.42
##   3rd Qu.:0.9978   3rd Qu.:3.400   3rd Qu.:0.7300   3rd Qu.:11.10
##   Max.   :1.0037   Max.   :4.010   Max.   :2.0000   Max.   :14.90
##      quality
##   Min.   :3.000
##   1st Qu.:5.000
##   Median :6.000
##   Mean   :5.636
##   3rd Qu.:6.000
##   Max.   :8.000
```

```
py$data %>%
    as_tibble() %>%
    glimpse()
```

```
## Observations: 1,599
## Variables: 12
## $ `fixed acidity`        <dbl> 7.4, 7.8, 7.8, 11.2, 7.4, 7.4, 7.9, 7.3...
## $ `volatile acidity`     <dbl> 0.700, 0.880, 0.760, 0.280, 0.700, 0.66...
## $ `citric acid`          <dbl> 0.00, 0.00, 0.04, 0.56, 0.00, 0.00, 0.0...
## $ `residual sugar`       <dbl> 1.9, 2.6, 2.3, 1.9, 1.9, 1.8, 1.6, 1.2,...
## $ chlorides              <dbl> 0.076, 0.098, 0.092, 0.075, 0.076, 0.07...
## $ `free sulfur dioxide`  <dbl> 11, 25, 15, 17, 11, 13, 15, 15, 9, 17, ...
## $ `total sulfur dioxide` <dbl> 34, 67, 54, 60, 34, 40, 59, 21, 18, 102...
## $ density                <dbl> 0.9978, 0.9968, 0.9970, 0.9980, 0.9978,...
## $ pH                     <dbl> 3.51, 3.20, 3.26, 3.16, 3.51, 3.51, 3.3...
## $ sulphates              <dbl> 0.56, 0.68, 0.65, 0.58, 0.56, 0.56, 0.4...
## $ alcohol                <dbl> 9.4, 9.8, 9.8, 9.8, 9.4, 9.4, 9.4, 10.0...
## $ quality                <dbl> 5, 5, 5, 6, 5, 5, 5, 7, 7, 5, 5, 5, 5, ...
```

```
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.externals import joblib
```

```
## C:\Users\NITISH~1.MIS\AppData\Local\CONTIN~1\ANACON~1\lib\site-packages\sklearn\externals\joblib\__i
##   warnings.warn(msg, category=DeprecationWarning)
```

```
y = data.quality
X = data.drop("quality", axis=1)

X_train, X_test, y_train, y_test = train_test_split(
  X, y,
  test_size    = 0.2,
  random_state = 123,
  stratify     = y
)
```

```python
scaler = preprocessing.StandardScaler().fit(X_train)
X_test_scaled = scaler.transform(X_test)
pipeline = make_pipeline(
    preprocessing.StandardScaler(),
    RandomForestRegressor(n_estimators = 100)
)
```

```python
hyperparameters = {
    "randomforestregressor__max_features" : ["auto", "sqrt", "log2"],
    "randomforestregressor__max_depth"    : [None, 5, 3, 1]
}

clf = GridSearchCV(pipeline, hyperparameters, cv = 10)
clf.fit(X_train, y_train)

#print(clf.best_params_)
```

```
## GridSearchCV(cv=10, error_score='raise-deprecating',
##              estimator=Pipeline(memory=None,
##                                 steps=[('standardscaler',
##                                         StandardScaler(copy=True,
##                                                        with_mean=True,
##                                                        with_std=True)),
##                                        ('randomforestregressor',
##                                         RandomForestRegressor(bootstrap=True,
##                                                               criterion='mse',
##                                                               max_depth=None,
##                                                               max_features='auto',
##                                                               max_leaf_nodes=None,
##                                                               min_impurity_decrease=0.0,
##                                                               min_impurity_split=None,
##                                                               min_...
##                                                               min_weight_fraction_leaf=0.0,
##                                                               n_estimators=100,
##                                                               n_jobs=None,
##                                                               oob_score=False,
##                                                               random_state=None,
##                                                               verbose=0,
##                                                               warm_start=False))],
##                                 verbose=False),
##              iid='warn', n_jobs=None,
##              param_grid={'randomforestregressor__max_depth': [None, 5, 3, 1],
##                          'randomforestregressor__max_features': ['auto', 'sqrt',
##                                                                  'log2']},
##              pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
##              scoring=None, verbose=0)
```

```python
y_pred = clf.predict(X_test)
print(r2_score(y_test, y_pred))
```

```
## 0.46815230946183184
```

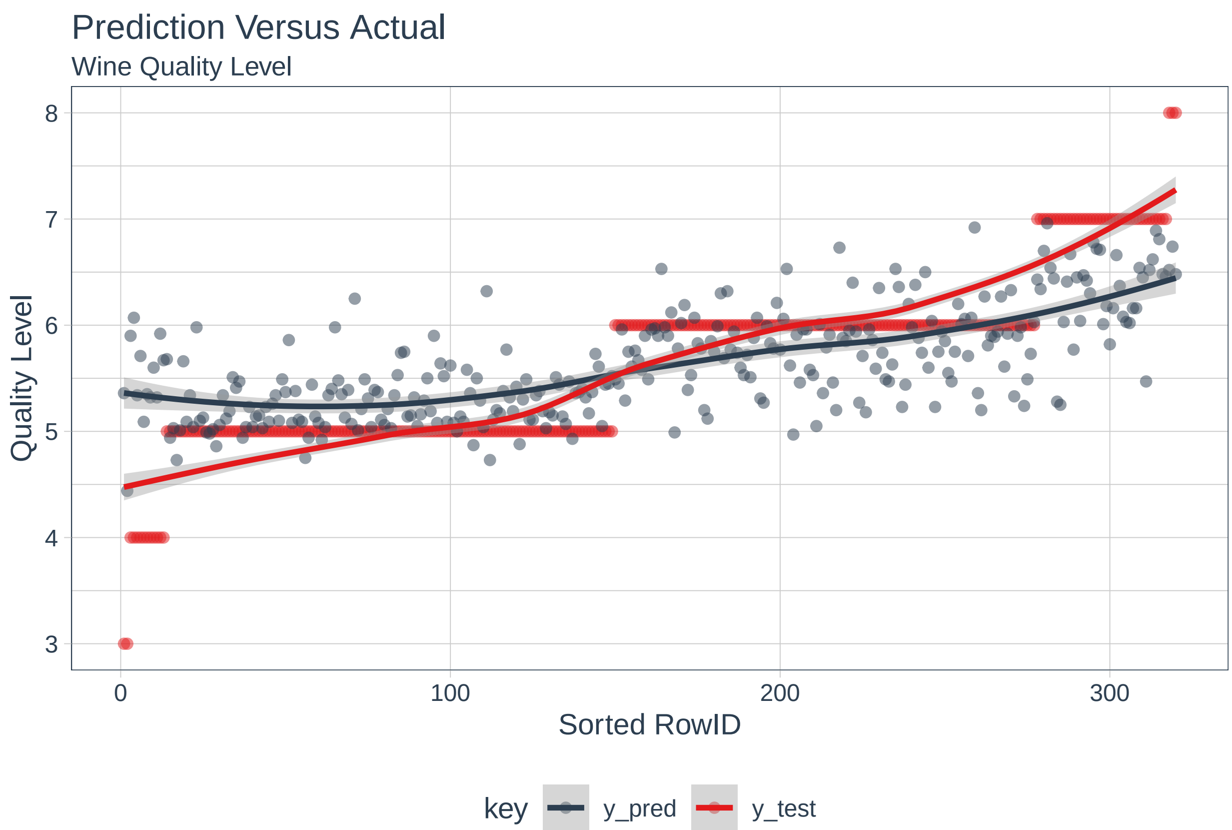
```python
print(mean_squared_error(y_test, y_pred))
```

```
## 0.34318718750000005
```

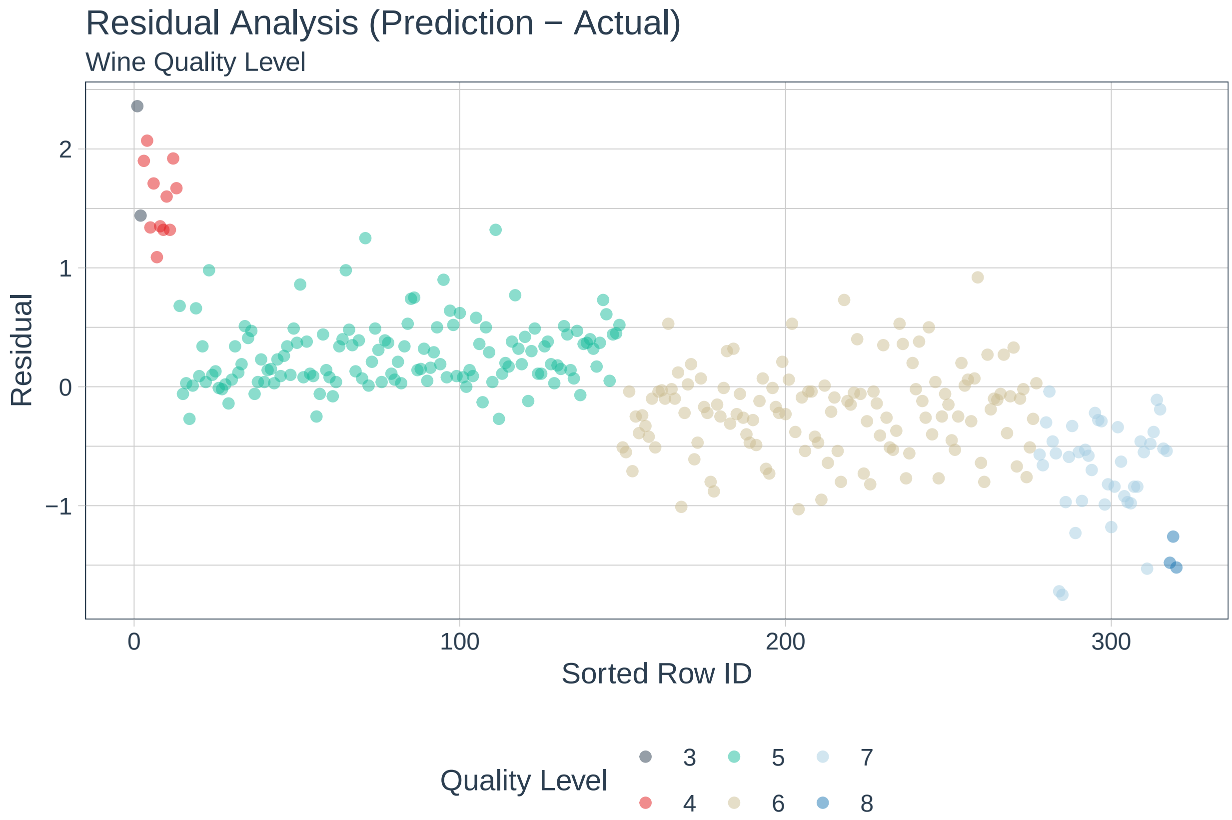
```r
#R
library(tidyverse)
library(tidyquant) # for theme_tq()

# Manipulate data for ggplot
results_tbl <- tibble(
    y_test = py$y_test,
    y_pred = py$y_pred
) %>%
    rowid_to_column() %>%
    arrange(y_test) %>%
    mutate(rowid = as_factor(as.character(rowid))) %>%
    rowid_to_column("sorted_rowid") %>%
    gather(key = "key", value = "value", -c(rowid, sorted_rowid))

# Make ggplot
results_tbl %>%
    ggplot(aes(sorted_rowid, value, color = key)) +
    geom_point(alpha = 0.5) +
    geom_smooth() +
    theme_tq() +
    scale_color_tq() +
    labs(
        title = "Prediction Versus Actual",
        subtitle = "Wine Quality Level",
        x = "Sorted RowID", y = "Quality Level"
    )
```




```
results_tbl %>%
  # Manipulation
  spread(key, value) %>%
  mutate(resid = y_pred - y_test) %>%
  # Plot
  ggplot(aes(sorted_rowid, resid, color = as.character(y_test))) +
    geom_point(alpha = 0.5) +
    theme_tq() +
    scale_color_tq() +
    labs(
        title = "Residual Analysis (Prediction - Actual)",
        subtitle = "Wine Quality Level",
        x = "Sorted Row ID", y = "Residual",
        color = "Quality Level"
    )
```



Residual Analysis (Prediction − Actual)
Wine Quality Level
Residual
2
1
0
−1
0
100
200
300
Sorted Row ID
Quality Level
3
4
5
6
7
8