KiSTi 한국과학기술정보연구원
Korea Institute of Science and Technology Information

2022

스마트교통 빅데이터 분석

VDS 데이터 분석 및 기계학습 적용 실습하기

2022.3.28.

이 홍 석 (hsyi@kisti.re.kr)

교통 데이터 다운로드

국가교통데이터 오픈마켓

대전시 VDS 운행 원시 이력 데이터

- VDS(Vehicle Detection System) : 차량검지기
  - ✓도로 상에 약 1km 간격으로 설치되어 실시간으로 교통량, 점유율, 속도, 대기행렬길이, 차량길이 등의 정보를 검지하여 소통 및 돌발상황 등을 감시하는 장치로 도로 환경적 특성에 따라 설치하며 종류는 루프식, 영상식, 레이더식 등이 있다
  - ✓대전시 VDS 운행 원시 이력 데이터 (등록일자, 검지기 ID, VDS ID, VDS 구간 ID, 요일 구분, 교통량 (소/중/대형), 속도, 점유율, 차두 길이, 차두 시간 등)

도로 0 대전광역시

대전시 VDS 운행 원시 이력 데이터

대전시 VDS 운행 원시 이력 데이터 [vds-collect-info.zip] 외 5건(80.74MB)

결제금액 합계 : 0 원

결제완료

상품 다운로드

문의하기

교통 데이터 다운로드

http://tportal.daejeon.go.kr/stats/content01.view?search=1

교통데이터 DW 시스템 (대전시)

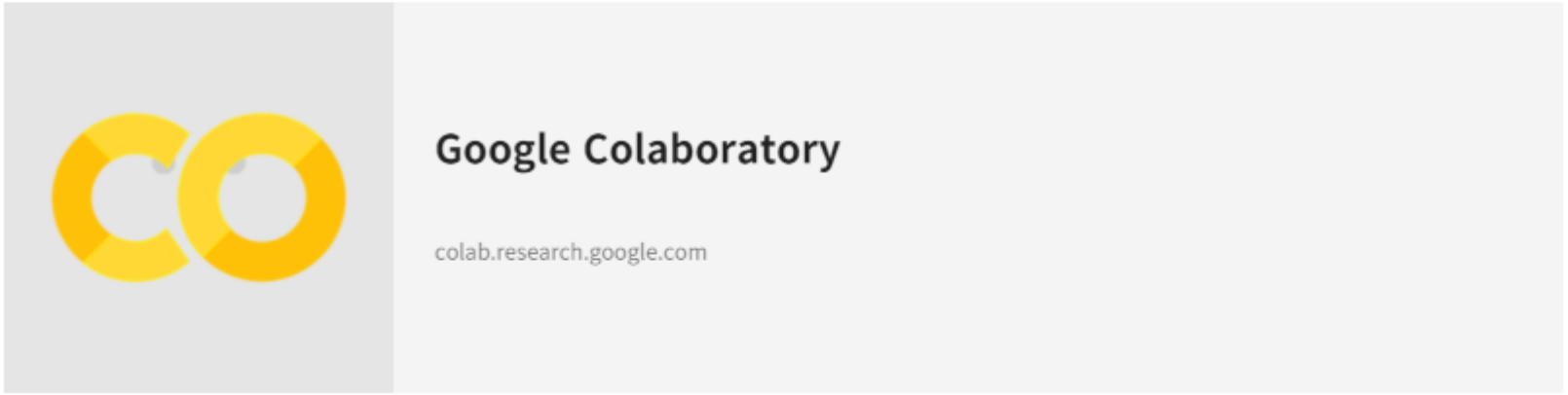
Colab 사용법

실습

코랩 Colab이란?

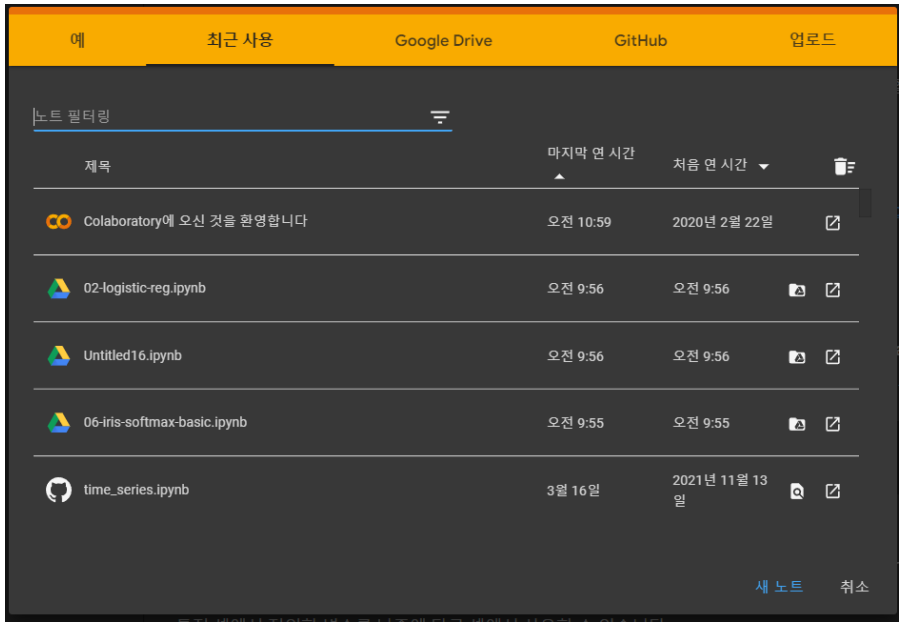
- Colaboratory는 설치가 필요 없으며 완전히 클라우드에서 실행되는 무료 Jupyter 노트 환경이다.
  - ✓즉, 브라우저에서 클라우드 환경을 이용해 Python 코드를 실행할 수 있으며,
  - ✓무료로 GPU와 TPU 등의 컴퓨팅 자원을 함께 사용할 수 있다.
  - ✓Colaboratory를 사용하면 브라우저를 통해 무료로 코드를 작성 및 실행하고
  - ✓분석을 저장 및 공유하며, 강력한 컴퓨팅 리소스를 이용할 수 있습니다.
- Google Colaboratory 접속하기

https://colab.research.google.com/



Google Colab 스펙

- **Google Colab 스펙**
  - ✓CPU : 제온
  - ✓Memory : 13GB
  - ✓HDD : 320GB
  - ✓GPU : NVIDIA Tesla K80
  - ✓ 이하 Google Cloab이라고도 부르며 사용법은 아주 간단합니다.
- **파일 생성/접근 방법**
  - ✓개인 구글 개정으로 접근
  - ✓https://colab.research.google.com 접속
  - ✓GOOGLE 새드라이브 PYTHON 노트탭 선택
  - ✓실습은python3로 진행할 예정.



코랩 사용법

Code cell, Text cell

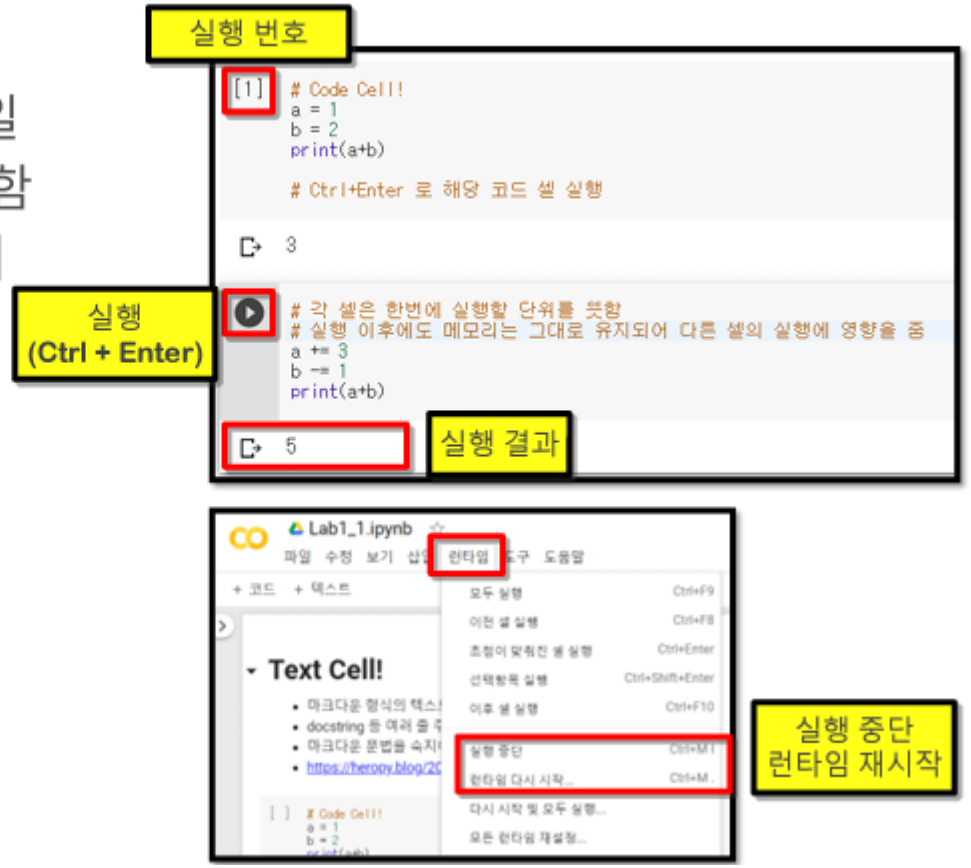
- .ipynb 파일은 code cell과 text cell로 구성
- 각 셀 하단에 마우스를 대거나, 화면 좌상단 버튼으로 셀 추가 가능
- 셀 선택(마우스) 후 셀 우상단 삭제버튼으로 셀 삭제 가능

코랩 사용법

Code cell

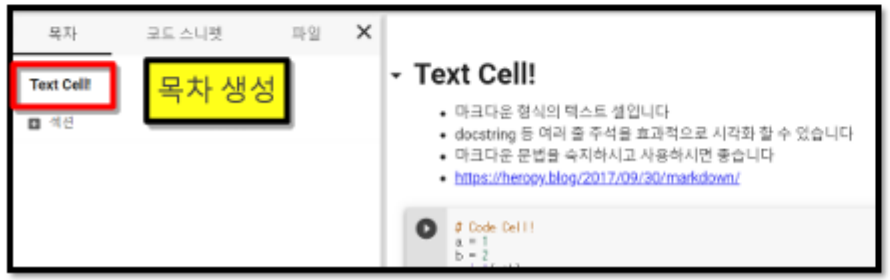
- 일반적인 파이썬 코딩 방식과 동일
- 각 셀은 한번에 실행할 단위를 뜻함
- 실행 이후에도 메모리는 유지되어 다른 셀 실행 시 영향을 줌
  - 런타임 다시 시작 시 초기화

- 상단 메뉴의 런타임
  - 실행 중인 셀 중단
  - 런타임 다시 시작



코랩 사용법

- Text cell
  - 여러 줄 주석의 효과적인 시각화
  - 마크다운(Markdown) 문법
  - 자동 목차 생성

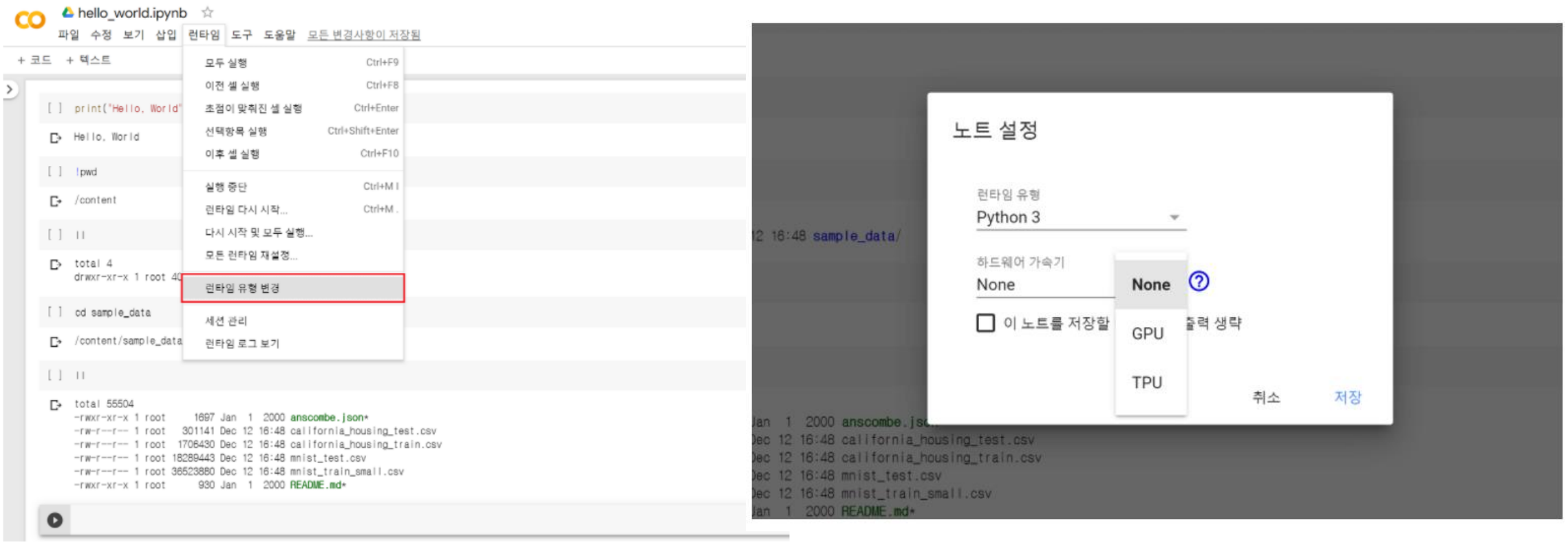


11

런타임 유형 GPU, TPU 설정하기

무료답지 않게 엄청난 서비스가 무료입니다.

GPU, TPU 등을 지원하기 때문에 요즘 핫한 Tensorflow등을 동작할 수 있습니다.



GPU 확인 : !nvidia-smi 명령어


```
[1]  1 !nvidia-smi

Mon Mar 21 02:06:32 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   39C    P8    27W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

VDS 데이터 머신러닝

차량검지기 VDS 데이터 분석

plot_ml_curve 라이브러리 설정하기


```
import numpy as np
def plot_ml_curve(estimator, title, X, y, ylim=None, cv=None,
                        n_jobs=None, train_sizes=np.linspace(.1, 1.0, 20)):
    plt.figure()
    plt.title(title)
    if ylim is not None:
        plt.ylim(*ylim)
    plt.xlabel("Training examples")
    plt.ylabel("Score")
    train_sizes, train_scores, test_scores = learning_curve(
        estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)
    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)
    plt.grid()

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1, color="r")
    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1, color="g")
    plt.plot(train_sizes, train_scores_mean, 'o-', color="r", label="Training score")
    plt.plot(train_sizes, test_scores_mean, 'o-', color="g", label="Cross-validation score")

    plt.legend(loc='lower right')
    return plt
```

데이터 가져오기

```
[10] import pandas as pd
```

```
[11] from pandas import datetime
```

```
[12] def parser(x):
         return datetime.strptime(x, '%Y-%m-%d %H:%M')
```

```
[13] df = pd.read_csv('./daejeon_vds16.csv',  date_parser=parser)
```

```
[14] df.head()
```

	Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
0	2017-04-02 0:00	43	34	9	0	50.3	1.90
1	2017-04-02 0:05	45	32	13	0	58.9	1.84
2	2017-04-02 0:10	46	34	12	0	50.6	1.87
3	2017-04-02 0:15	45	36	9	0	50.9	1.72
4	2017-04-02 0:20	27	13	13	1	62.2	1.12

df.info()

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8064 entries, 0 to 8063
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Date      8064 non-null   object 
 1   ToVol     8064 non-null   int64  
 2   SmVol     8064 non-null   int64  
 3   MeVol     8064 non-null   int64  
 4   LaVol     8064 non-null   int64  
 5   Speed     8064 non-null   float64
 6   Occ.Rate  8064 non-null   float64
dtypes: float64(2), int64(4), object(1)
memory usage: 441.1+ KB
```

df.describe()


```
[16] df.describe()
```

초

	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate
count	8064.000000	8064.000000	8064.000000	8064.000000	8064.000000	8064.000000
mean	110.459945	79.353299	29.948537	1.158110	49.327431	6.166941
std	63.954451	46.802106	19.081136	1.530192	7.921856	6.739946
min	6.000000	2.000000	0.000000	0.000000	9.100000	0.230000
25%	50.000000	35.000000	13.000000	0.000000	44.900000	2.140000
50%	122.000000	87.000000	29.000000	1.000000	48.500000	5.550000
75%	155.000000	111.000000	44.000000	2.000000	54.200000	7.290000
max	338.000000	250.000000	145.000000	16.000000	87.800000	82.100000

maxs = df.max()


```
maxs = df.max()
print(maxs)
```

```
Date        2017-04-29 9:55
ToVol                  338
SmVol                  250
MeVol                  145
LaVol                   16
Speed                 87.8
Occ.Rate              82.1
dtype: object
```

라벨을 정하기

속도 Speed를 라벨로 정하자

```
def get_score(v):
    if v < 20:
        score = 'Jam'
    elif v < 40:
        score = 'Slow'
    else :
        score = 'Normal'
    return score
```

```
[20] df["label"] = df["Speed"].apply(lambda v: get_score(v))
     df
```

	Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate	label
0	2017-04-02 0:00	43	34	9	0	50.3	1.90	Normal
1	2017-04-02 0:05	45	32	13	0	58.9	1.84	Normal

df.head()

```
df.head()
```

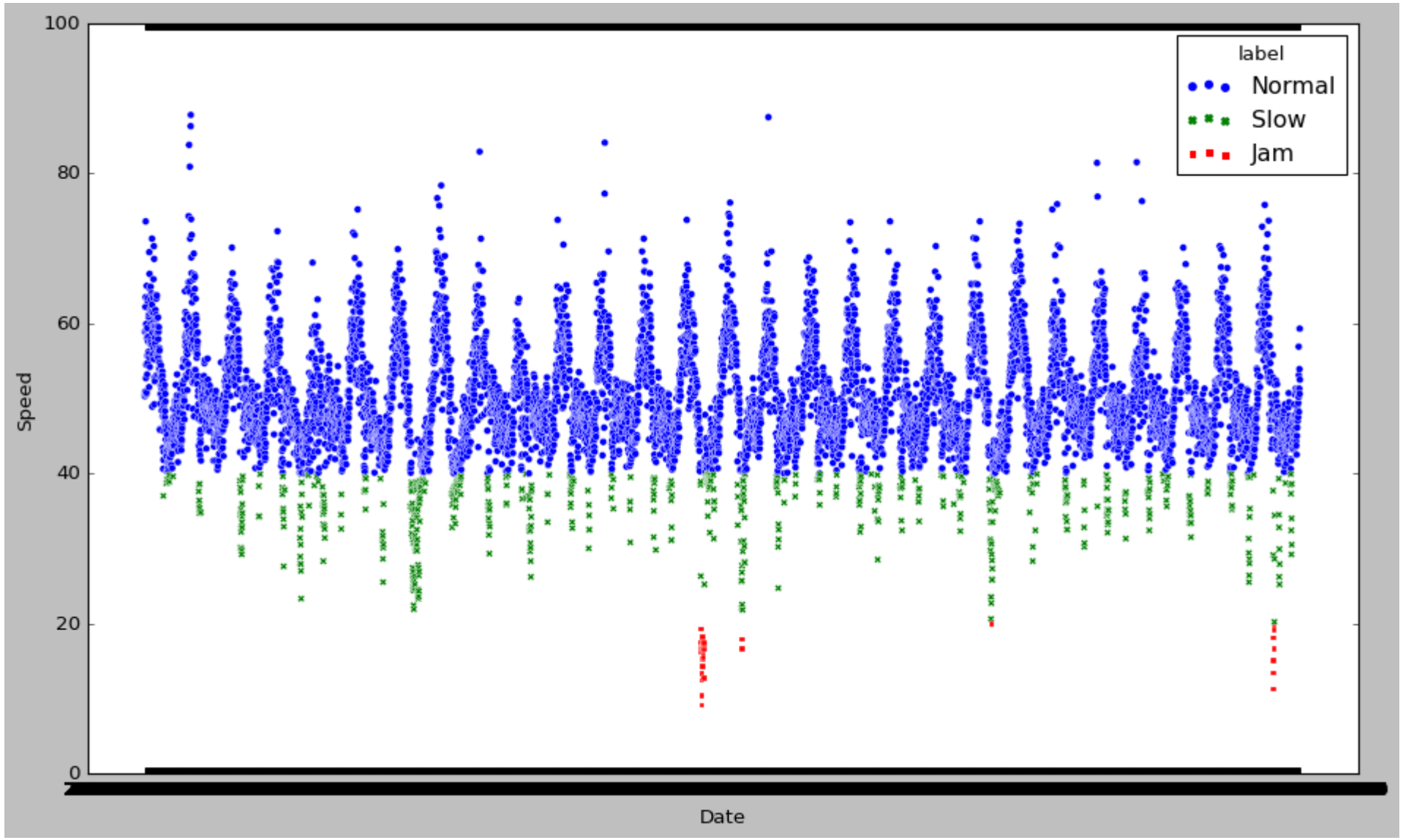
	Date	ToVol	SmVol	MeVol	LaVol	Speed	Occ.Rate	label
0	2017-04-02 0:00	43	34	9	0	50.3	1.90	Normal
1	2017-04-02 0:05	45	32	13	0	58.9	1.84	Normal
2	2017-04-02 0:10	46	34	12	0	50.6	1.87	Normal
3	2017-04-02 0:15	45	36	9	0	50.9	1.72	Normal
4	2017-04-02 0:20	27	13	13	1	62.2	1.12	Normal

3. 데이터 가시화

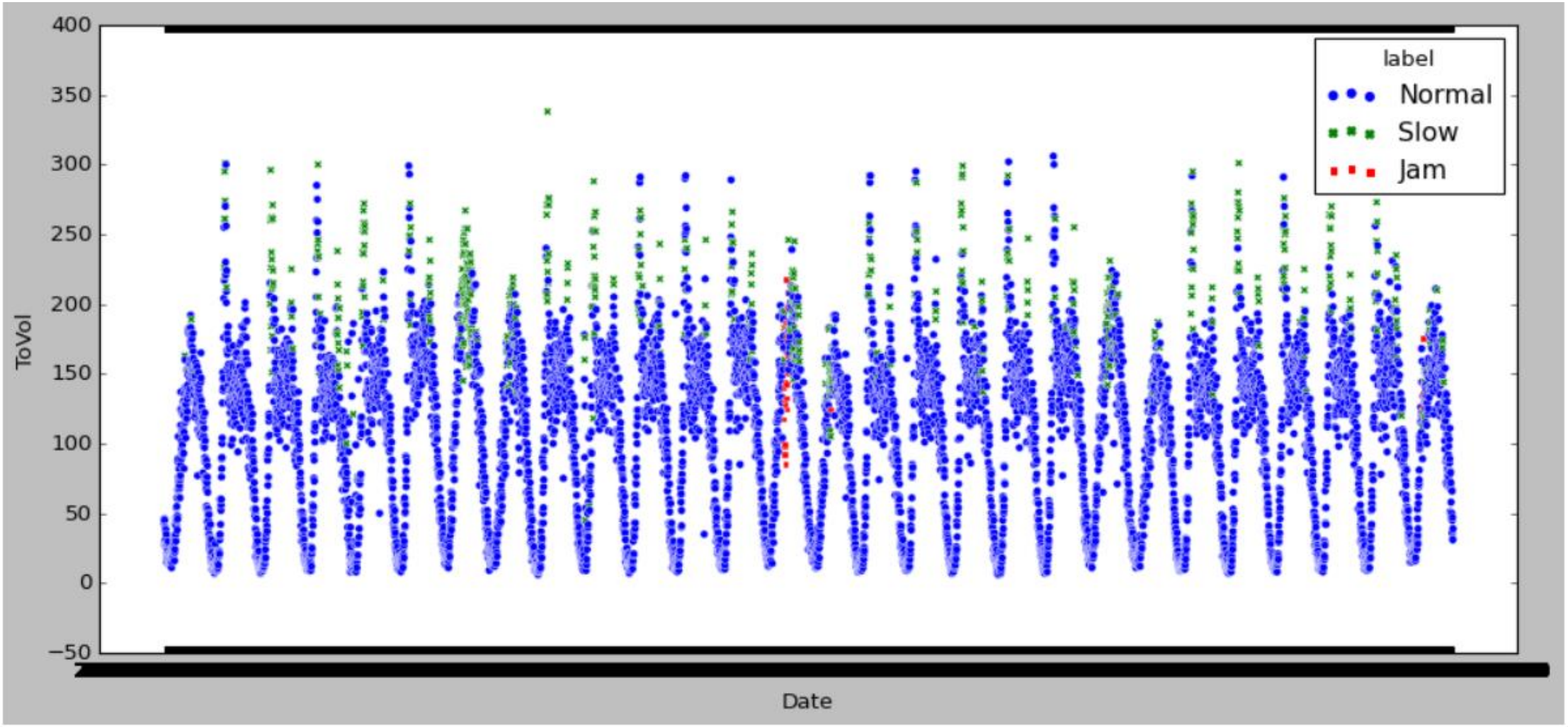
[23]
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use("classic")
```

[24]
```
import seaborn as sns
plt.figure(figsize=(14,8))
sns.scatterplot(data=df, x = 'Date', y = 'Speed', hue='label', style='label')
```

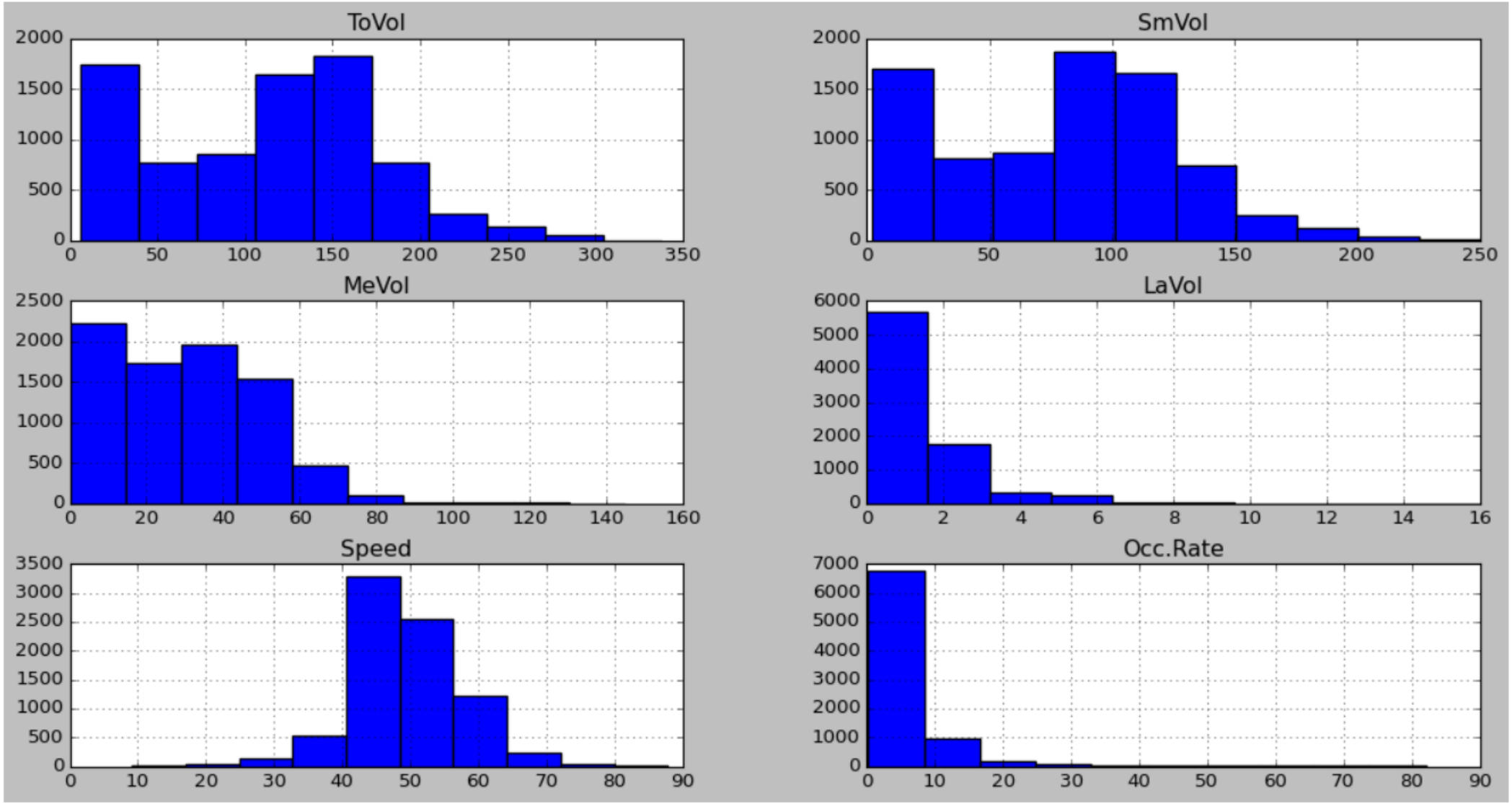
데이터 가시화



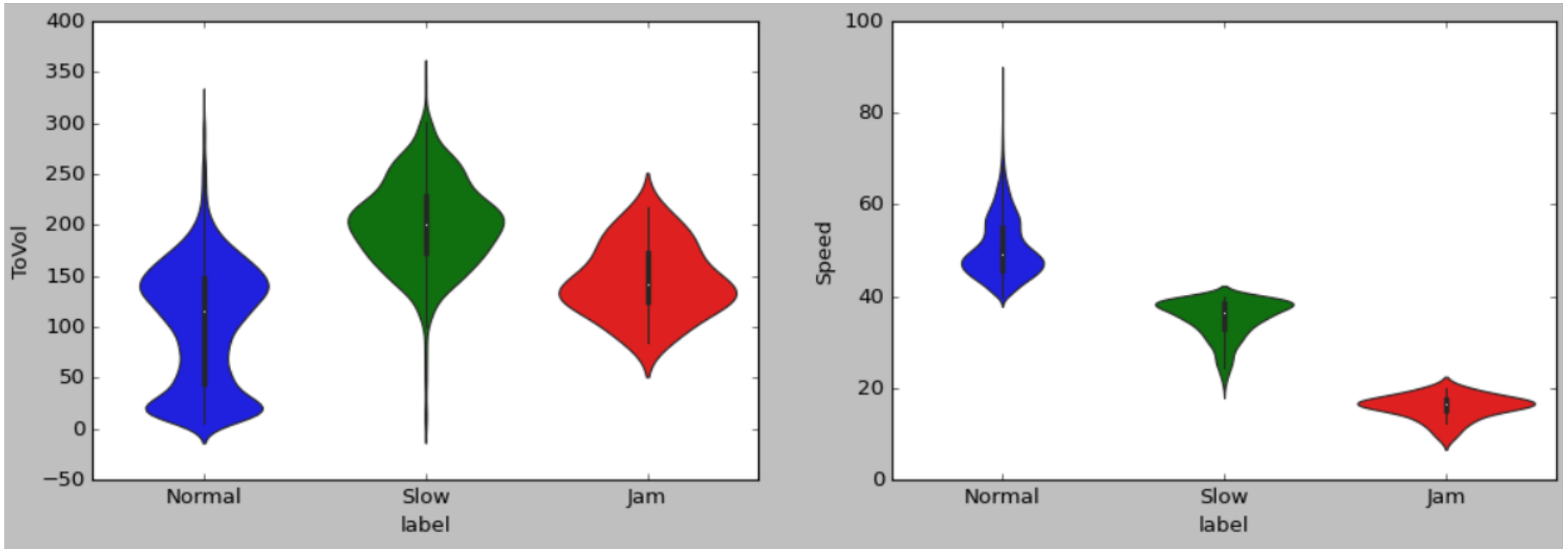
sns.scatterplot(data=df, x = 'Date', y = 'ToVol', hue='label')



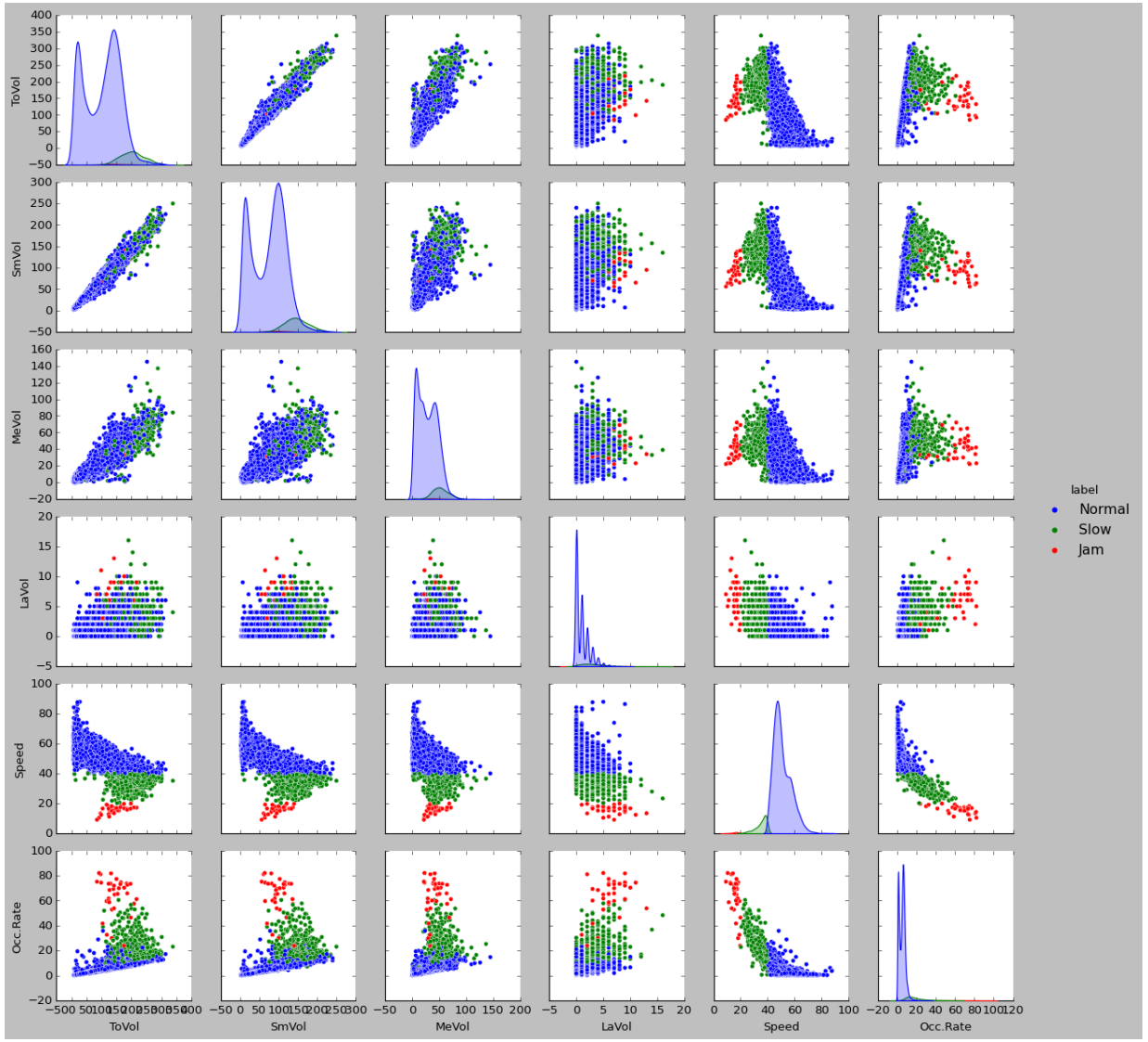
df.hist(edgecolor='black', linewidth=1.2)

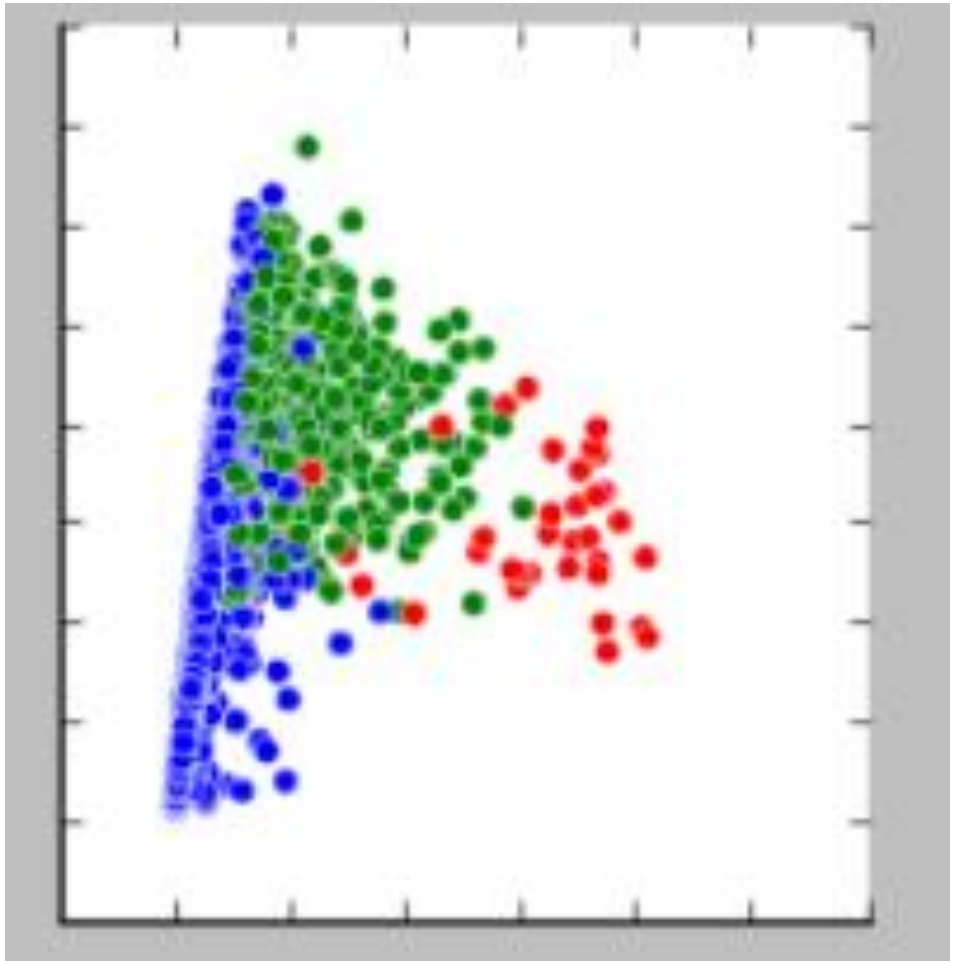


sns.violinplot



sns.pairplot(df,hue='label')





교통량

점유율

sns.heatmap(df.corr(),annot=True,cmap='YlGnBu')

머신러닝을 위한 입력 데이터 만들자

1) 입력 X와 출력 y의 값을 정하기

```
df['label'].unique()
```

```
array(['Normal', 'Slow', 'Jam'], dtype=object)
```

```
#feature_cols = ['ToVol', 'SmVol', 'Speed', 'Occ.Rate']
#feature_cols = ['ToVol', 'SmVol', 'LaVol', 'MeVol']
feature_cols = ['ToVol','Occ.Rate']
target_col = 'label'
X = df[feature_cols]
y = df[target_col]
```

```
X.head()
```

	ToVol	Occ.Rate
0	43	1.90
1	45	1.84
2	46	1.87
3	45	1.72
4	27	1.12

```
y.head()
```

```
0    Normal
1    Normal
2    Normal
3    Normal
4    Normal
Name: label, dtype: object
```

2) 출력용 라벨을 머신러닝

텍스트를 숫자로 바꾸자

In [34]:
```
class_dic = {'Jam':0, 'Slow':1, 'Normal':2}
y_ohc = y.apply(lambda z: class_dic[z])
```

In [35]:
```
y_ohc.head()
```

Out [35] :
```
0    2
1    2
2    2
3    2
4    2
Name: label, dtype: int64
```

데이터를 훈련과 테스트로 나누자

3) 데이터를 훈련과 테스트로 나누자

- (실전) 데이터를 validation을 포함해서 나눌수 있다.
- (해보기) 전체 데이터를 train : validation : test = 0.6: 0.2: 0.2 로 나누어라

```
from sklearn.model_selection import train_test_split,  ShuffleSplit, learning_curve
#from sklearn.model_selection import learning_curve, train_test_split, KFold,  ShuffleSplit

X_train, X_test, y_train, y_test = train_test_split(X, y_ohc, test_size=0.20, random_state=30)
```

```
print(X_train.shape, y_train.shape)
print(X_test.shape,  y_test.shape)
```

```
(6451, 2) (6451,)
(1613, 2) (1613,)
```

데이터 정규화


```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

5. sklearn을 이용하여 머신러닝 모델을 만들자


```
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
```

1) 로지스틱 회귀

In [41]:
```
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

In [42]:
```
m_lr = LogisticRegression()
m_lr.fit(X_train,y_train)
```

Out [42] : LogisticRegression()

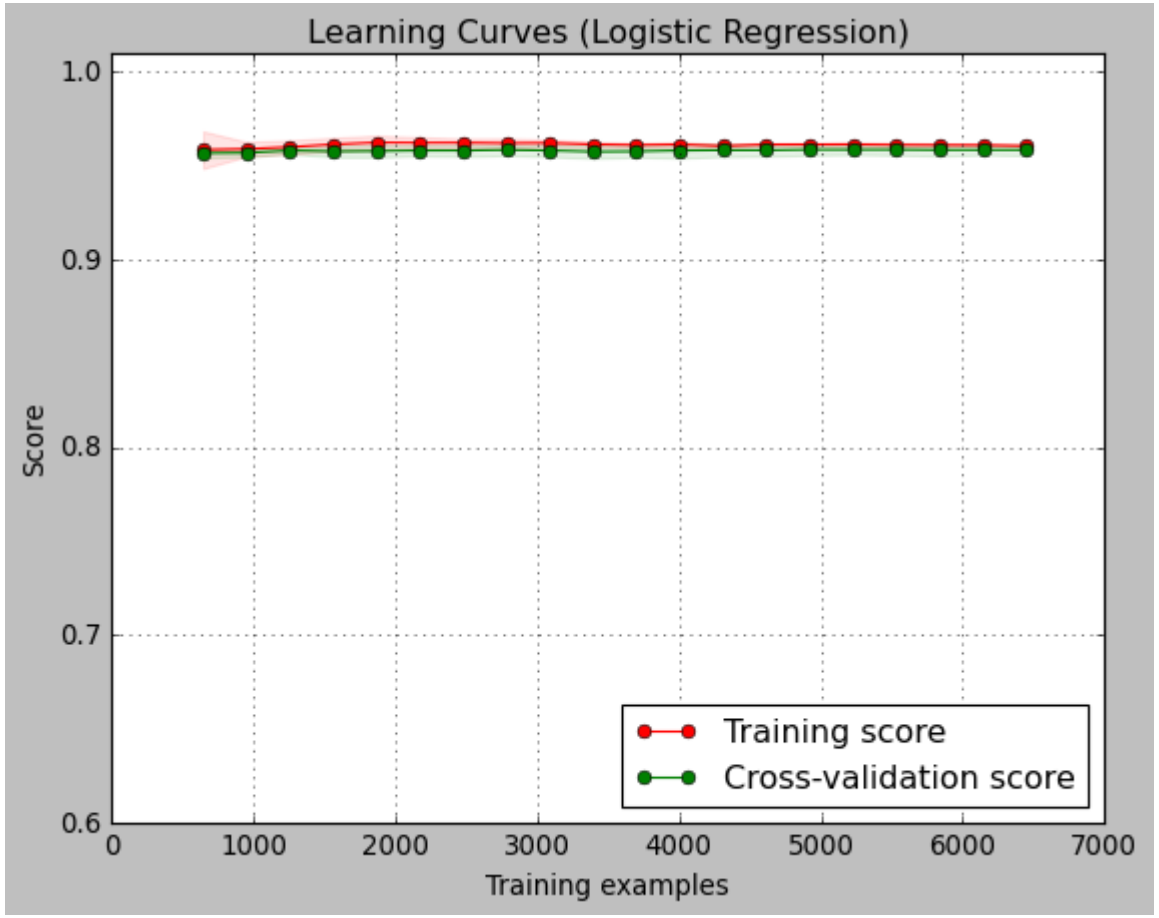
In [43]:
```
pred = m_lr.predict(X_test)

acc_lr = metrics.accuracy_score(pred,y_test)
print('The accuracy of the Logistic Regression is', acc_lr)
```

```
The accuracy of the Logistic Regression is 0.9603223806571606
```

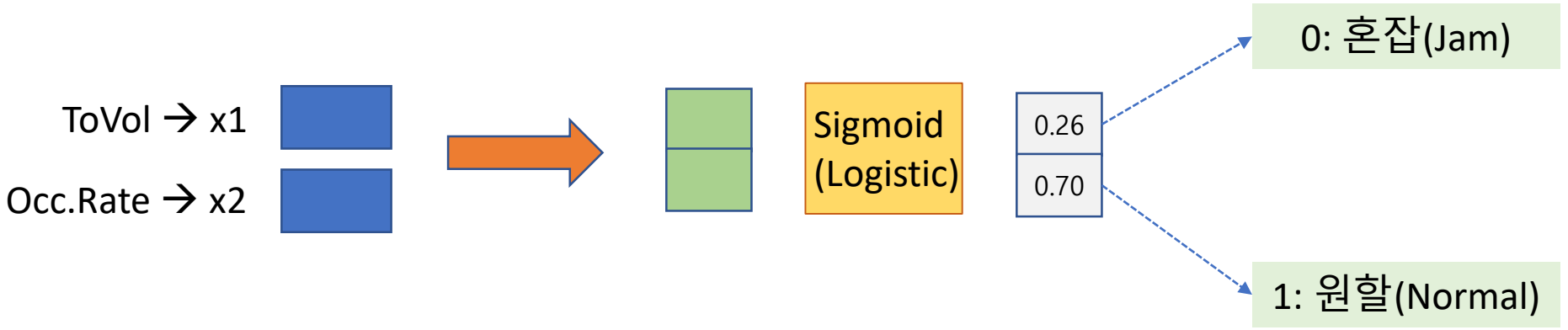
In [44]:
```
title = "Learning Curves (Logistic Regression)"
cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)
```

1) 로지스틱 회귀

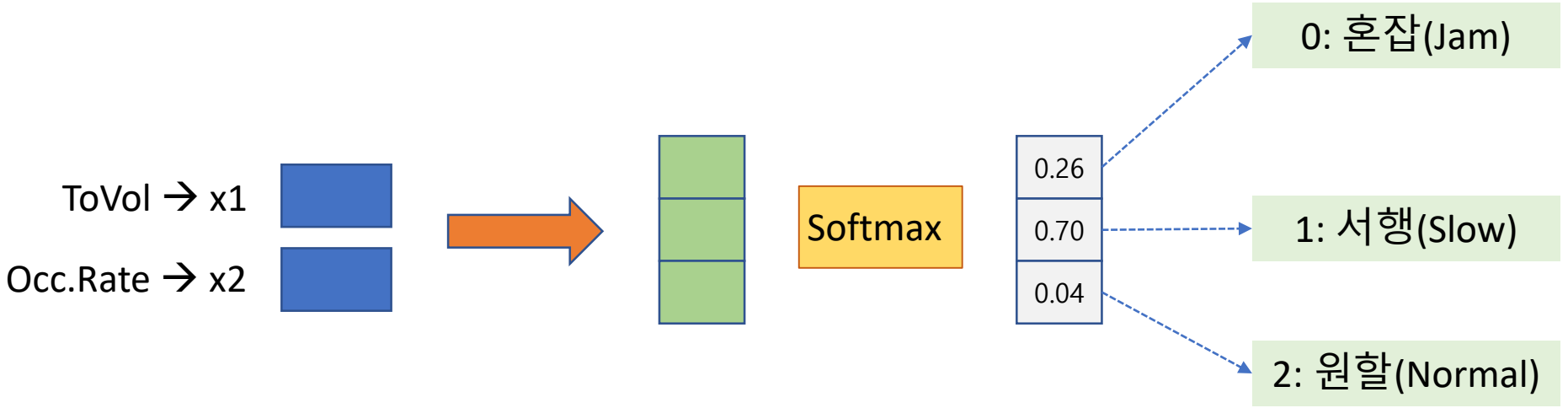
```
#import myUtil as myutil
plot_ml_curve(m_lr, title, X, y, ylim=(0.6, 1.01), cv=cv)
```

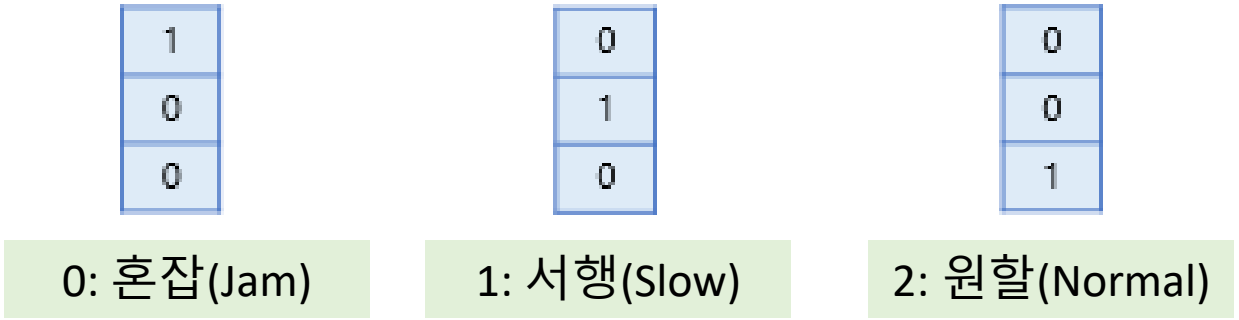


라벨이 2개 일경우


ToVol → x1
Occ.Rate → x2
Sigmoid
(Logistic)
0.26
0.70
0: 혼잡(Jam)
1: 원할(Normal)

라벨이 3개 일경우

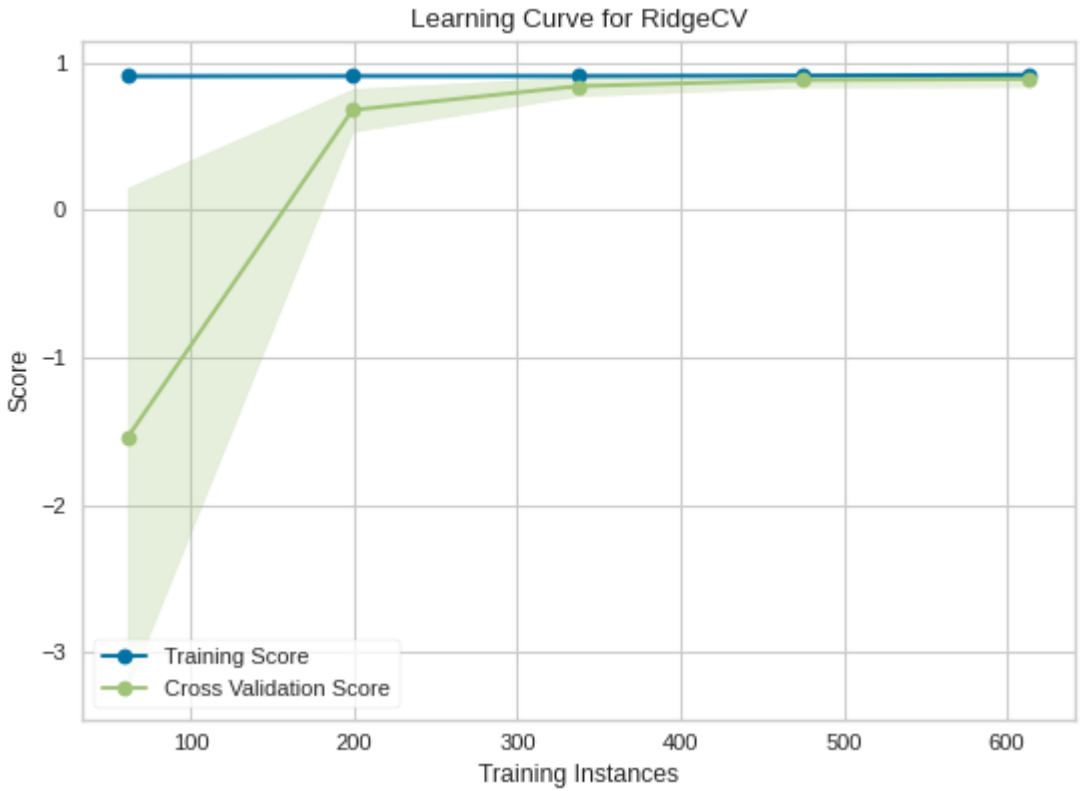

ToVol → x1
Occ.Rate → x2
Softmax
0.26
0.70
0.04
0: 혼잡(Jam)
1: 서행(Slow)
2: 원할(Normal)


1
0
0
0: 혼잡(Jam)
0
1
0
1: 서행(Slow)
0
0
1
2: 원할(Normal)

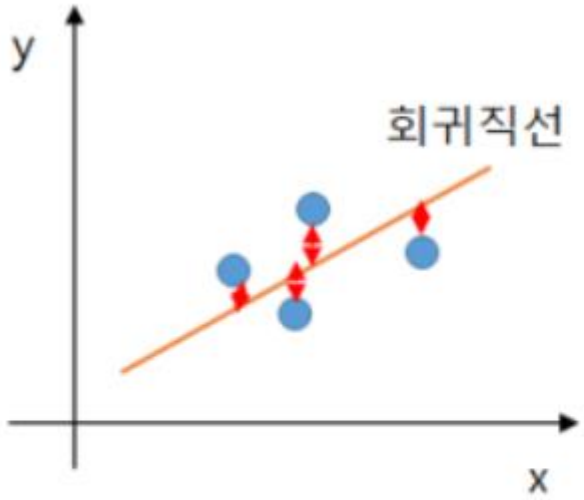
sklearn learning curve : Regression에서 R2 스코어 사용

- sklearn에서 learning_curve의 score는 높은 값이 좋은 성능으로 표현함.
  - ✓회귀에서 MSE 보다는 Negative MSE를 사용함
  - ✓R2도 사용함
- sklearn에서 Regression learning_curve의 score는 deafult로 R2를 사용함.

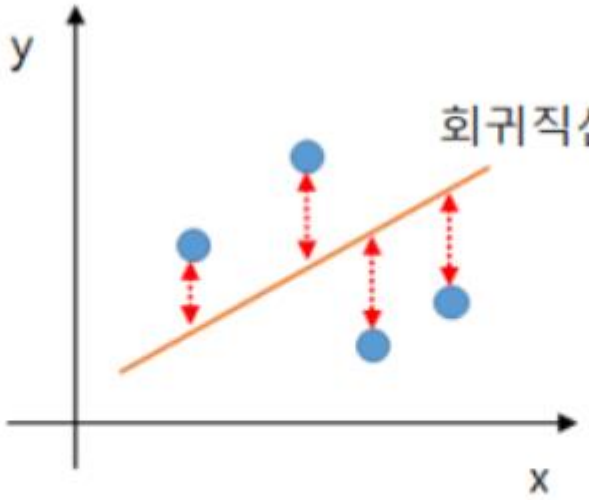
$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$



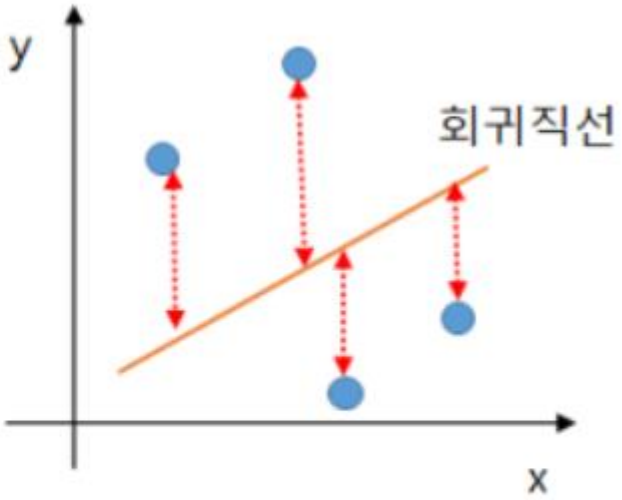
회귀의 평가지표(default): 설명력(결정계수, R2)



R^2가 1에 가까운 경우



R^2가 0.5에 가까운 경우



R^2가 0에 가까운 경우

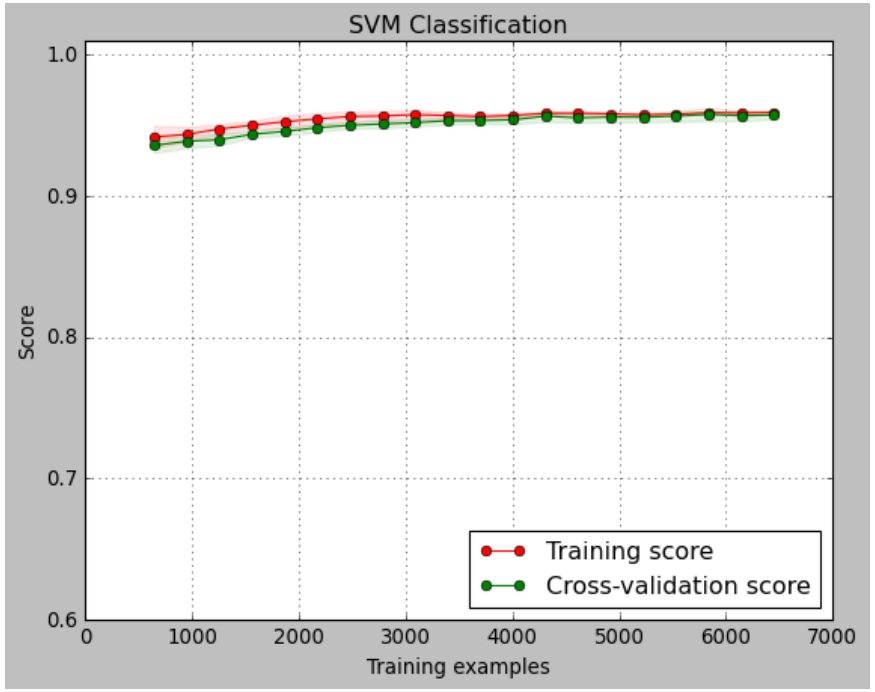
2) Support Vector Machine (SVM)

```
sv = svm.SVC()
sv.fit(X_train,y_train)
pred = sv.predict(X_test)
acc_svm = metrics.accuracy_score(pred,y_test)
print('The accuracy of the SVM is:', acc_svm)
```

The accuracy of the SVM is: 0.9671419714817111

```
%%time
title = "SVM Classification"
cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)
plot_ml_curve(sv, title, X, y, ylim=(0.6, 1.01), cv=cv )
```

Wall time: 34.6 s

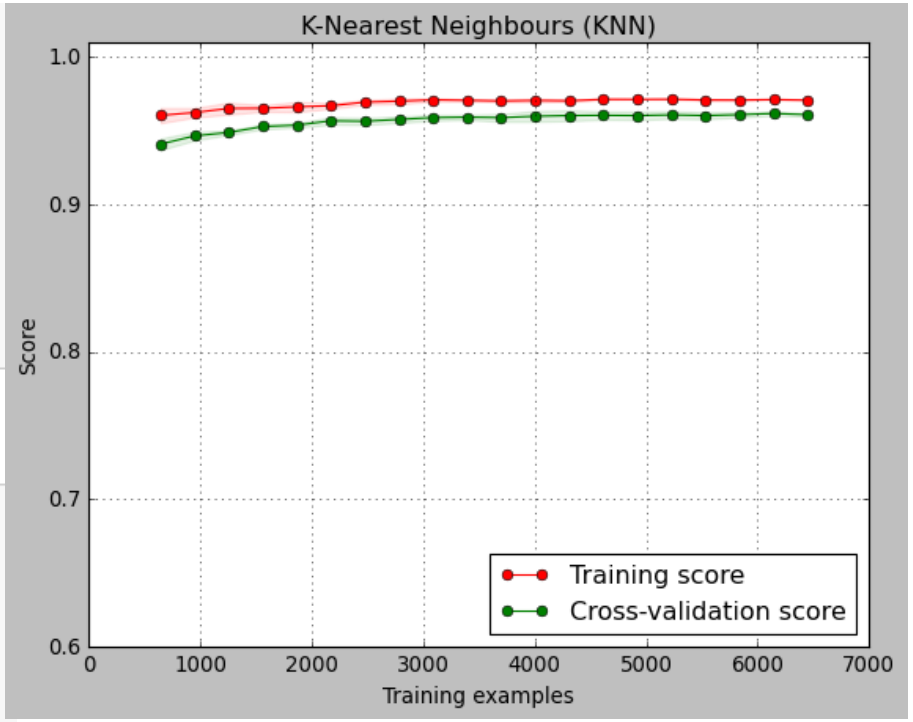


3) K-Nearest Neighbours

```
knc = KNeighborsClassifier(n_neighbors=6)
knc.fit(X_train,y_train)
pred = knc.predict(X_test)
acc_knn = metrics.accuracy_score(pred,y_test)
print('The accuracy of the KNN is', acc_knn)
```

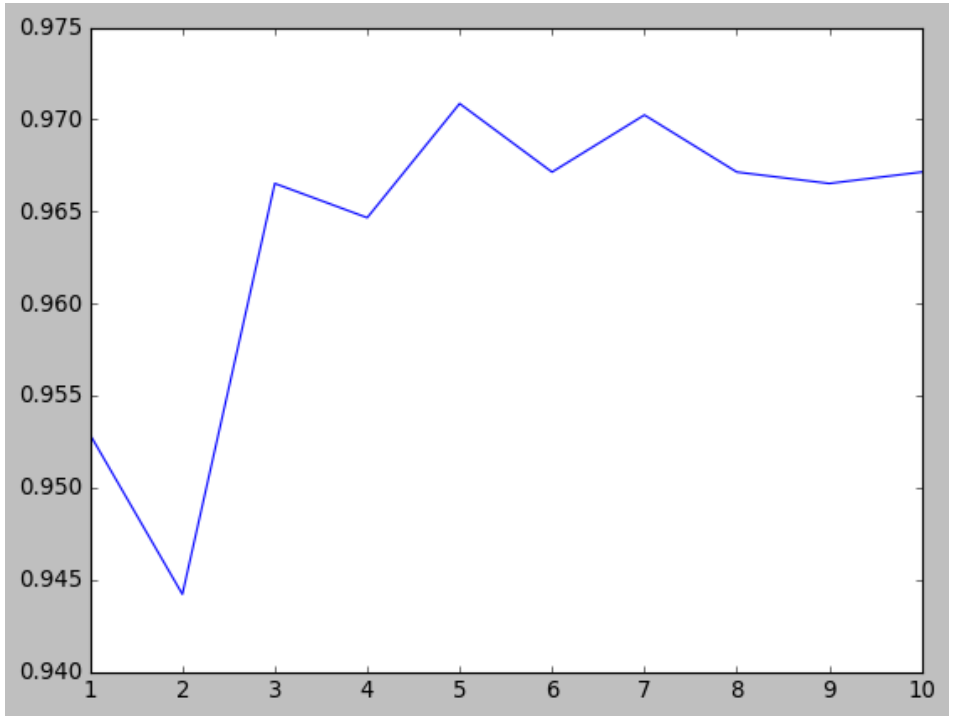
```
The accuracy of the KNN is 0.9671419714817111
```

```
title = "K-Nearest Neighbours (KNN)"
cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)

plot_ml_curve(knc, title, X, y, ylim=(0.6, 1.01), cv=cv, n_jobs=4)
```



3.1 최적의 k-NN의 정확도 확인


```
a_index = list(range(1,11))
a = pd.Series()
x = [1,2,3,4,5,6,7,8,9,10]
for i in list(range(1,11)):
    kcs = KNeighborsClassifier(n_neighbors=i)
    kcs.fit(X_train,y_train)
    y_pred = kcs.predict(X_test)
    a=a.append(pd.Series(
        metrics.accuracy_score(y_pred,y_test)))

plt.plot(a_index, a)
plt.xticks(x)
```



4) 랜덤포레스트 (RandomForestClassifier)

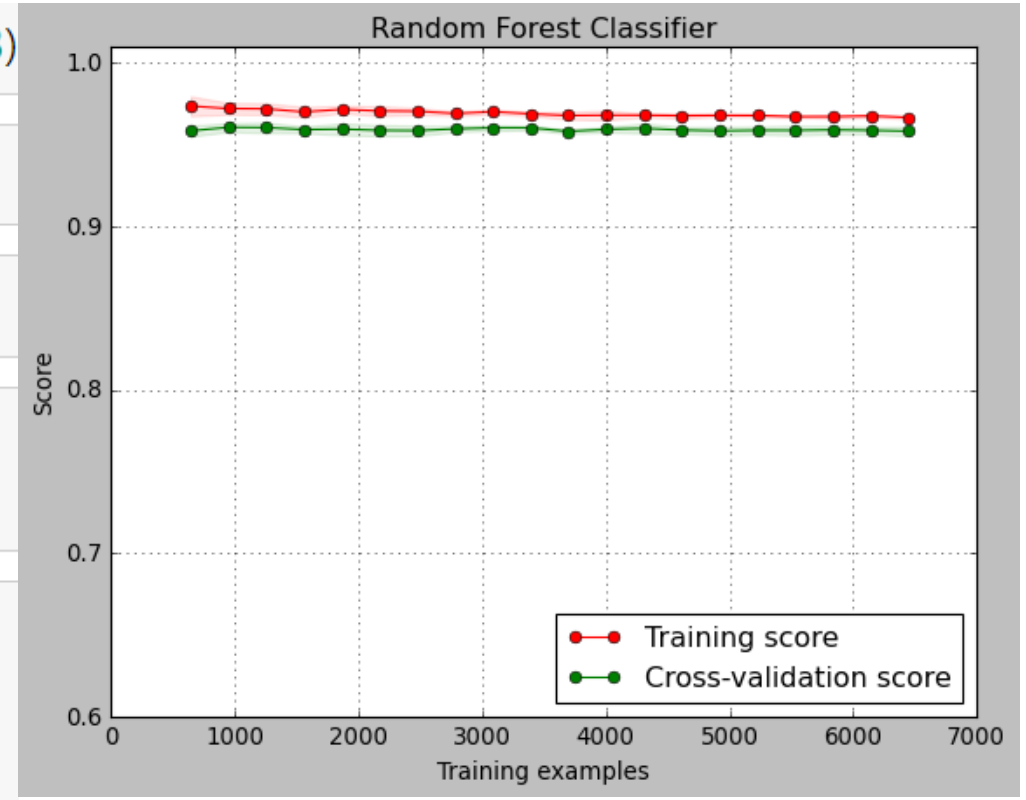

```
m_rf = RandomForestClassifier(n_estimators=100, max_depth = 3)
```

```
m_rf.fit(X_train, y_train)
```

```
pred = m_rf.predict(X_test)
```

```
acc_rf = metrics.accuracy_score(pred,y_test)
print('The accuracy of the RFC is:', acc_rf)
```

```
title = "Random Forest Classifier"
cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)
plot_ml_curve(m_rf, title, X, y, ylim=(0.6, 1.01), cv=cv)
```



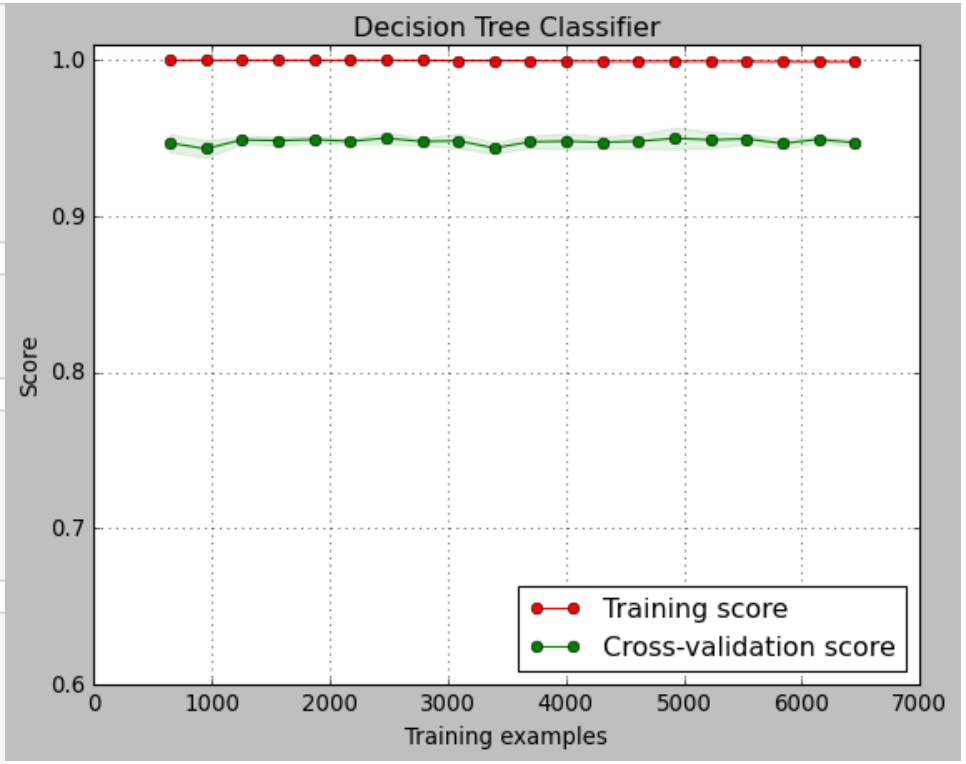
5) 의사결정나무 (DecisionTreeClassifier)


```
m_tree = DecisionTreeClassifier()

m_tree.fit(X_train, y_train)
```

```
prd = m_tree.predict(X_test)
```

```
acc_dt = metrics.accuracy_score(pred,y_test)
print('The accuracy of the Decision Tree is:', acc_dt)
```

```
title = "Decision Tree Classifier"
cv = ShuffleSplit(n_splits=4, test_size=0.2, random_state=0)

plot_ml_curve(m_tree, title, X, y, ylim=(0.6, 1.01), cv=cv)
```



기계학습 성능 비교


```
models = pd.DataFrame({
    'Model': ['Logistic Regression',  'Support Vector Machines','RandomForest',
              'K-Nearest Neighbours', 'Decision Tree'],
    'Score': [acc_lr, acc_svm, acc_rf, acc_knn, acc_dt]})
models.sort_values(by='Score', ascending=False)
```

	Model	Score
1	Support Vector Machines	0.967142
3	K-Nearest Neighbours	0.967142
0	Logistic Regression	0.960322
2	RandomForest	0.960322
4	Decision Tree	0.001860

2022 TRUST KISTI

Korea Institute of Science
and Technology Information