automl

- 자동으로 예측모델 생성하는 라이브러리
- wsl에서 수행.

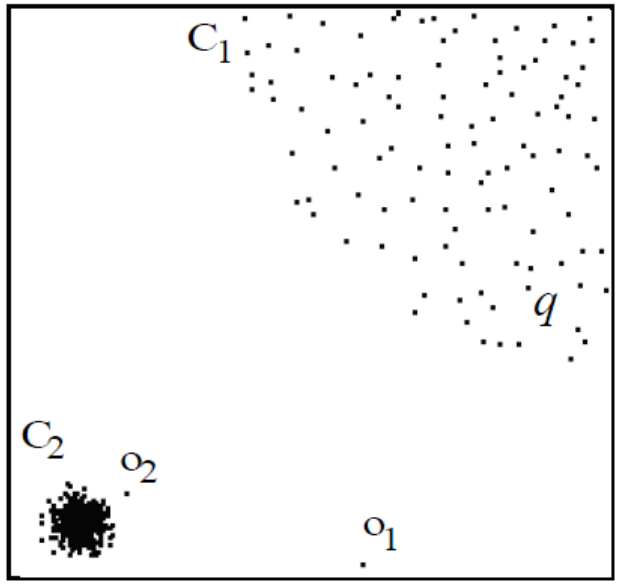
```
$   (pip install –-upgrade pip)
$   conda create -n ak python=3.8 tensorflow ipykernel
$   conda activate ak
$   python -c 'import tensorflow as tf'
$   pip install --user ipykernel  # 커널등록도구 설치
$   python -m ipykernel install --user --name=ak   # 새환경에서 커널 등록
$   pip install matplotlib pandas sklearn keras autokeras
$   sudo apt install graphviz
$   pip install pydot
```

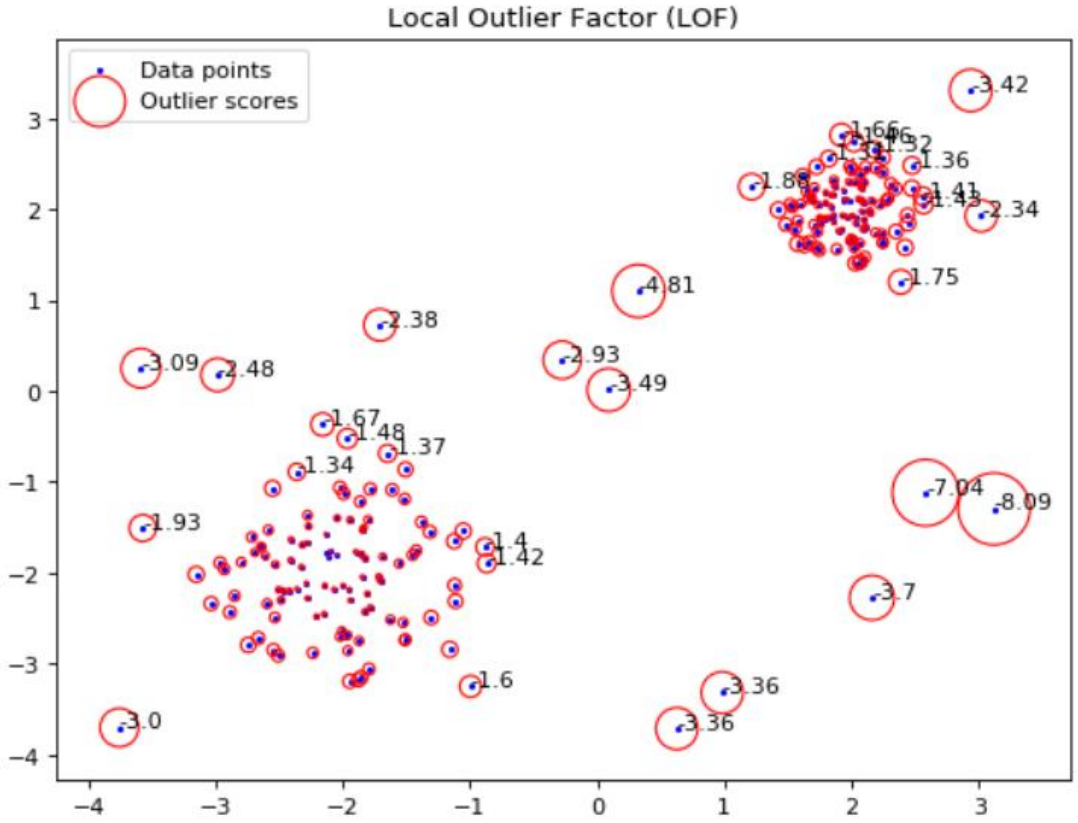
< 실습 내용 >

- 기본 keras model 구축 및 시험
  - 1. 1. basic+ak.py
- autokeras 실습 내용
  - model 생성하는 법.
  - cross validation 방법.
  - StandardScaler() 사용.
  - 참고 노트 : 6.4.1 Ch4_CV_Eval.ipynb
- pycaret 실습
  - 데이터 읽어오기

2. 참고 : 이상치

어떤 데이타포인트가 이상치일까.





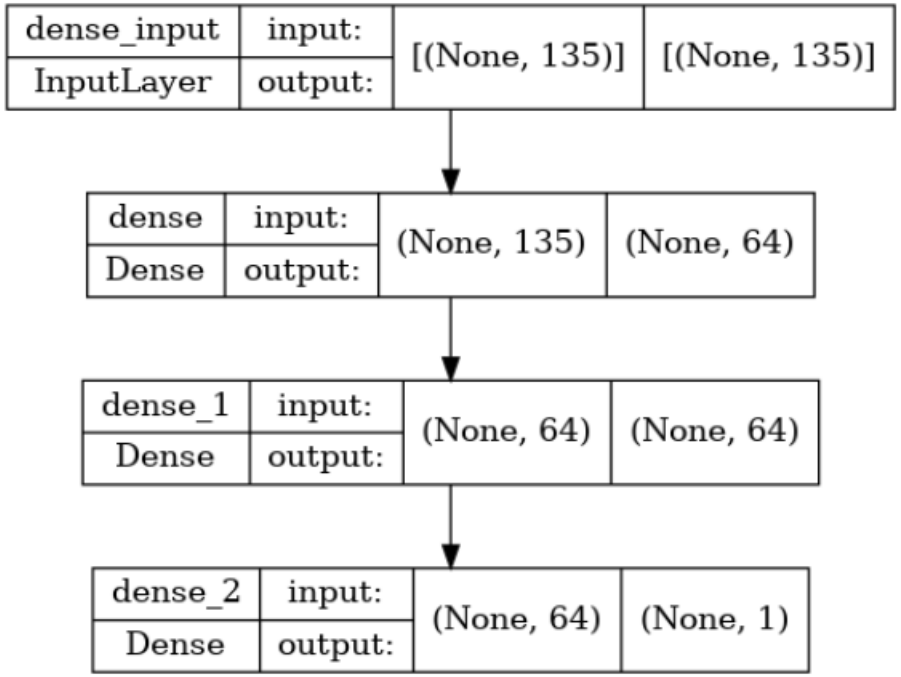
https://dl.acm.org/doi/10.1145/335191.335388

2. 기본 모델 구현 및 시험

- 모델 생성하는 함수 정의 실습

```
from tensorflow.keras import models
from tensorflow.keras import layers

def build_model():
    # 동일한 모델을 여러 번 생성할 것이므로 함수를 만들어 사용합니다
    model = models.Sequential()
    model.add(layers.Dense(??, activation='relu',
                           input_shape=(??,)))
    model.add(layers.Dense(??, activation='relu'))
    model.add(layers.Dense(??))
    model.compile(optimizer='rmsprop', loss='mse', metrics=['mae'])
    return model
```



- 실습 : 1. basic+ak.py
- cross validation library 사용방법 실습
  - 6.4.1 Ch4_CV_Eval.ipynb 참고
- scaler 사용방법 실습
  - 6.4.1 Ch4_CV_Eval.ipynb 참고

2. autokeras 설치 및 실습

- keras 기반 모델을 자동으로 생성해주는 유틸
- 3가지 주요 기능
  1) Auto Feature Engineering
  2) Neural Search Architecture
  3) Hyper Parameter Tuning

- 설치

  $ pip install git+https://github.com/keras-team/keras-tuner.git

  $ pip install autokeras

- (실습) : 1. basic+ak.py

```
Search: Running Trial #5

Value             |Best Value So Far |Hyperparameter
True              |True              |structured_data_block_1/normalize
False             |False             |structured_data_block_1/dense_block_1/use_batchnorm
2                 |2                 |structured_data_block_1/dense_block_1/num_layers
32                |32                |structured_data_block_1/dense_block_1/units_0
0                 |0                 |structured_data_block_1/dense_block_1/dropout
32                |32                |structured_data_block_1/dense_block_1/units_1
0                 |0                 |regression_head_1/dropout
adam              |adam              |optimizer
0.0001            |0.001             |learning_rate
```

< 기본 구조 >

```
reg = ak.StructuredDataRegressor(max_trials=max_trials,
metrics='mae', seed=5)
reg.fit(X_train, y_train, epochs=epochs)

model = reg.export_model()
model.save('berry.h5')
new_model = tf.keras.models.load_model('berry.h5')
new_model.predict(X_test)
```

2. pycaret 설치 및 실습

- pycaret 설치

```
$  conda create –n caret python=3.8 ipykernel
$  conda activate caret
$  python -m ipykernel install --user --name=caret   # 새환경에서 커널 등록
$  pip install pycaret numpy pandas
```

< 주요 수행 내용 >

- 데이터 준비
- Setting up Environment
  - 데이터 속성 정의, 각종 데이터 전처리 방법 정의(scaling, imputing …)
  - 모델구축을 위한 파이프라인을 정의
  - 자체 알고리즘을 이용하여 각 특성들의 데이터 유형을 유추함(numerical, categorical, ordinal) → → 만약 틀리면 수동으로 정정
- Compare Model
  - 모델별 기본 매개변수로 보유하고 있는 모든 모델 평가.
  - 특정모델 제외, 검증성능 변경, 폴드수 변경 등 가능.
- Create Model
  - 유력한 모델 대상으로 개별 모델 생성
- Tune Model
  - Random Grid Search 방법으로 매개변수 튜닝
  - 필요시 탐색대상 매개변수 범위 지정 가능
- Finalize Model
  - 교차검증을 위해 분리된 모든 데이터를 사용하여 모델 생성

2. pycaret 실습

- (실습 1) 2. (ref) pycaret_diamond.ipynb
- (실습 2) 2. (ref) pycaret_diamond.ipynb 를 복사하여 brix 데이터를 대상으로 모델 생성.
- 학습데이타 가져오기
  - 1. basic+ak.py 에서 X, y 데이터를 파일로 저장하기
    - y.to_csv()
    - X.savetxt()
  - 저장된 학습데이타 가져오기
    - pd.read_csv()
    - X, y 4856 개 데이터가 읽어졌는지 확인.
- 모델생성
  - setup()
  - compare_models()
  - create_model()
  - tune_model()
  - plot_model()
  - finalize_model()
  - predict_model()
  - save_model()