**AIFactory 제 2 회 USG AI⋅데이터 문제해결 제조혁신 경진대회 코드 설명자료**

**- USG AI 데이터 문제해결 제조혁신 경진대회 생산설비 이상 진단 -**

**대회 참가자 (노정현, 김석진, 정제완, 추호성)**

**2022년 03월 29일**

1. **라이브러리 및 데이터 (Library & Data)**

* 코드 내에 사용할 library import
* 사용할 데이터를 pandas의 read\_csv를 사용해 Dataframe 형태로 load

|  |
| --- |
| # 필요한 library import  import pandas as pd # 데이터를 Dataframe 형태로 load하기 위해  import numpy as np # 데이터를 numpy 형태로 변환하기 위해  import seaborn as sns # 데이터 EDA를 위해  import matplotlib.pyplot as plt # 데이터 EDA를 위해  from tensorflow.keras.models import Sequential # 데이터 훈련 모델을 위해  import tensorflow as tf # 데이터 훈련 모델을 위해  from tensorflow.keras import layers # 데이터 훈련 모델을 위해  # 데이터 불러오기  train\_input = pd.read\_csv('train\_data.csv')  test\_input = pd.read\_csv('test\_data.csv')  train\_output = pd.read\_csv('train\_label.csv') |

1. **데이터 전처리 (Data Cleansing & Pre-Processing)**

* 훈련시 필요없는 ID 열을 drop
* LSTM을 사용해 train 하기위해 train 데이터를 numpy 로 변환 한 후 ID 개수 별로 reshape해서 시계열 형식의 데이터 제작
* label data의 ID열을 drop, 출력값을 4개로 분류하기 위해 정답 열인 type을 onehot 인코딩을 사용 후 train 데이터와 마찬가지로 numpy 로 변환
* test 데이터셋도 마찬가지로 제작

|  |
| --- |
| # ID열 삭제  train = train.drop('ID',axis=1)  # numpy로 변환  train = np.array(train)  # 같은 ID가 131개가 있으므로 8690,131,3으로 reshape  train = train.reshape((8960,131,3))  # ID열 삭제  label = train\_label.drop('ID', axis=1)  #pandas의 onehot 인코딩 메소드를 사용하여 type열을 onehot 벡터 형태로 변환  label = pd.get\_dummies(label['type'])  # 훈련 데이터와 형태를 같게 하기위해 numpy 형태로 변환  label = np.array(label)  # test데이터도 훈련 데이터와 마찬가지로 제작  test = test.drop('ID', axis=1)  test = np.array(test)  test = test.reshape(2240,131,3) |

1. **탐색적 자료 분석 (Exploratory Data Analysis)**

* 데이터의 shape, 정보 등을 알기 위해 shape, head, info, describe 메소드를 사용
* corr 메소드를 사용해서 나타난 상관관계를 hitmap으로 출력
* 각 열 별 데이터 분포도를 그래프를 사용해서 출력

|  |
| --- |
| # data shape 출력  print(train\_input.shape)  print(train\_output.shape)  print(test\_input.shape)  # head 함수를 사용하여 데이터 출력  train\_input.head()  # info 함수를 사용하여 데이터 정보 획득, 개수, type, 결측 치 확인 등  train\_input.info()  train\_output.info()  # ID열을 삭제 한 후 describe 함수를 사용하여 데이터의 개수, 평균, 분산, 최대 최소 값 등을 출력  data = train\_input.drop('ID', axis=1)  target = train\_output.drop('ID', axis=1)  data = data.join(target)  data.describe()  # corr 메소드를 사용해 나타난 상관관계를 seaborn 라이브러리와 matplotlib.pyplot 라이브러리를 사용해 hitmap으로 표현  heatmap = sns.heatmap(data.corr(),annot=True,cmap='RdYlGn',linewidths=0.2)  fig=plt.gcf()  fig.set\_size\_inches(10,10)  heatmap.set\_xticklabels(heatmap.get\_xticklabels(), fontsize=20)  heatmap.set\_yticklabels(heatmap.get\_yticklabels(), fontsize=20)  plt.title('correlation between features', fontsize=40)  plt.show()  # 센서별 분포도를 seaborn 라이브러리와 matplotlib.pyplot 라이브러리를 사용해 그래프로 표현  feature = data.columns  plt.figure(figsize=(20,60))  for i in range(len(feature)):  plt.subplot(11,3,i+1)  plt.title(feature[i])  plt.xlim(-3,3)  plt.ylim(0,1)  sns.distplot(data[feature[i]],color='magenta')  plt.show() |

1. **변수 선택 및 모델 구축 (Feature Engineering & Initial Modeling)**

* tensorflow 라이브러리를 사용하여 시계열 데이터를 훈련하는데 큰 효과를 보이는 RNN 계열인 LSTM layer를 구축
* EDA결과 데이터가 단조롭게 보여 적은 unit을 사용해도 결과가 잘 나올것 같으므로 unit의 개수는 16개씩 4개의 LSTM layer를 구축,activation fucntion은 default값인 tanh를 사용
* 마지막 layer는 1~4인 4가지로 분류 해야 하므로 출력층을 dense layer로 설정하고, unit을 4개, activation function은 다중 분류 이므로 softmax를 사용
* 분류 문제 이므로 loss는 다중 분류일 때 사용하는 categorical\_crossentropy, optimizer는 가장 많이 사용되고 효과가 좋다고 알려진 Adam을 사용, learning\_rate는 반복횟수를 증가시킬 것 이기 때문에 0.001로 설정, 훈련 평가 방법인 metrics는 정확도인 accuracy로 설정하여 모델을 컴파일
* 성능 평가가 가장 좋은 모델을 저장 할 것이기 때문에 callback method를 정의

|  |
| --- |
| # RNN 모델을 LSTM layer로 구축, 출력 레이어는 4개로 구축  rnn = Sequential([  layers.LSTM(16, input\_shape=(131,3), return\_sequences=True),  layers.LSTM(16, return\_sequences=True),  layers.LSTM(16, return\_sequences=True),  layers.LSTM(16, return\_sequences=False),  layers.Dense(4, activation='softmax')  ])  # 구축한 모델을 compile  optimizer = tf.keras.optimizers.Adam(learning\_rate=0.001)  rnn.compile(loss=tf.keras.losses.categorical\_crossentropy, optimizer=optimizer, metrics=['accuracy'])  # 성능이 가장 좋은 모델을 저장하기위해 callback method를 구현  callbacklist =[tf.keras.callbacks.ModelCheckpoint(filepath = 'LSTM.h5', moniter = 'val\_accuracy', save\_best\_only=True)] |

1. **모델 학습 및 검증 (Model Tuning & Evaluation)**

* 반복 횟수를 100으로 설정
* fit메소드를 사용한 훈련, 검증 세트는 훈련세트의 0.1만큼 사용하여 검증

|  |
| --- |
| # 반복횟수 100 설정  EPOCHS = 100  # model fitting  history = rnn.fit(train, label, callbacks=callbacklist,  epochs=EPOCHS, validation\_split = 0.1, verbose=1) |

1. **결과 및 결언 (Conclusion & Discussion)**

* 성능이 가장 좋았던 모델을 다시 load
* test 데이터셋을 사용하여 예측
* 예측 결과가 예측 확률로 나오므로 argmx메소드를 사용하여 가장 높은 확률로 예측한 인덱스 값으로 변경
* 인덱스 값은 0~3인 값이므로 1씩 더해주어서 예측 해야하는 결과인 1~4인 값으로 변경
* answer\_sample.csv을 불러와 type 열에 변경된 값들을 저장 후 index를 제거하여 LSTM\_answer.csv파일에 저장
* 시계열 데이터 이므로 시계열 데이터를 훈련하기에 가장 적합한 모델인 RNN을 사용하는 것이 모델 구축의 핵심
* 데이터가 매우 단조로워 모델 구축하기가 쉬웠지만 실제 Global한 데이터는 단조롭지 않을수도 있으므로 해당 데이터를 학습할 때 모델의 깊이를 늘리거나 학습률을 조절할 필요가 있어 보인다.

|  |
| --- |
| # 가장 성능이 좋은 모델 load  model = tf.keras.models.load\_model('LSTM.h5')  # 모델 예측 및 label에 맞게 예측 값 변경  pred = model.predict(test)  prediction = pred.argmax(axis=1)  prediction = prediction + 1  # 예측 결과를 파일에 저장  submit = pd.read\_csv('answer\_sample.csv')  submit['type'] = prediction  submit.to\_csv('LSTM\_answer.csv', index=False) |