[예제] 당뇨병 발병 예측

1. https://www.kaggle.com/uciml/pima-Indians-diabetes-database 접속 후,

2. Download 메뉴를 통해서 데이터 다운 후 kaggle_diabetes.csv 파일로 이름 변경

3. 머신러닝/딥러닝 기본 프로세스를 바탕으로, 당뇨병 발병 확률을 70% 이상으로 예측

1. 데이터 로드 및 기본 정보 확인

```
import matplotlib
import pandas as pd
from matplotlib import pyplot as plt

df = pd.read_csv('./kaggle_diabetes.csv')

df.head()
```

```
# 전체 히스토그램을 살펴본다
df.hist()

plt.tight_layout()
plt.show()
```

```
# 개별 히스토그램을 살펴본다

df['BloodPressure'].hist()

plt.tight_layout()
plt.show()
```

```
df.info()
```

```
df.describe()
```

2. 데이터 전처리

```
# missing value 확인

df.isnull().sum()
```

```
for col in df.columns:
    missing_rows = df.loc[df[col]==0].shape[0]
    print(col + ": " + str(missing_rows))
```

```
import numpy as np

# outlier 처리

df['Glucose'] = df['Glucose'].replace(0, np.nan)
df['BloodPressure'] = df['BloodPressure'].replace(0, np.nan)
df['SkinThickness'] = df['SkinThickness'].replace(0, np.nan)
df['Insulin'] = df['Insulin'].replace(0, np.nan)
df['BMI'] = df['BMI'].replace(0, np.nan)

# missing value 처리

df['Glucose'] = df['Glucose'].fillna(df['Glucose'].mean())
df['BloodPressure'] = df['BloodPressure'].fillna(df['BloodPressure'].mean())
df['SkinThickness'] = df['SkinThickness'].fillna(df['SkinThickness'].mean())
df['Insulin'] = df['Insulin'].fillna(df['Insulin'].mean())
df['BMI'] = df['BMI'].fillna(df['BMI'].mean())
```

2. 데이터 전처리

```
for col in df.columns:
    missing_rows = df.loc[df[col]==0].shape[0]
    print(col + ": " + str(missing_rows))
```

```
df_scaled = df  # 원본 DataFrame 보존

df_scaled.describe()
```

```
# feature column, label column 추출 후 DataFrame 생성

feature_df = df_scaled[df_scaled.columns.difference(['Outcome'])]

label_df = df_scaled['Outcome']

print(feature_df.shape, label_df.shape)
```

```
# pandas <=> numpy

feature_np = feature_df.to_numpy().astype('float32')
label_np = label_df.to_numpy().astype('float32')

print(feature_np.shape, label_np.shape)
```

3. 머신러닝 / 딥러닝

```
s = np.arange(len(feature_np))

np.random.shuffle(s)

feature_np = feature_np[s]
label_np = label_np[s]
```

```
# train / test data 분리

split = 0.15

test_num = int(split*len(label_np))

x_test = feature_np[0:test_num]
y_test = label_np[0:test_num]

x_train = feature_np[test_num:]
y_train = label_np[test_num:]

print(x_train.shape, y_train.shape)
print(x_test.shape, y_test.shape)
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout

model = Sequential()

model.add(Dense(1, activation='sigmoid', input_shape=(8,) ))
```

```
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=1e-3),
              loss='binary_crossentropy', metrics=['accuracy'])
```

```
from datetime import datetime

start_time = datetime.now()

hist = model.fit(x_train, y_train, epochs=400, validation_data=(x_test, y_test))

end_time = datetime.now()

print('elapsed time => ',end_time-start_time)
```

```
model.evaluate(x_test, y_test)                # 또는 pred = model.predict(...)
```

```
import matplotlib.pyplot as plt

plt.title('loss trend')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.grid()

plt.plot(hist.history['loss'], label='train loss')
plt.plot(hist.history['val_loss'], label='validation loss')
plt.legend(loc='best')

plt.show()
```

```
import matplotlib.pyplot as plt

plt.title('accuracy trend')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.grid()

plt.plot(hist.history['accuracy'], label='train accuracy')
plt.plot(hist.history['val_accuracy'], label='validation accuracy')
plt.legend(loc='best')

plt.show()
```