모두를 위한 딥러닝 (입문편)

허 찬

강의 계획표 – 모두를 위한 딥러닝 (입문편)

학습주제(강좌명)	차시(모듈)	차시별 주제	학습내용
모두를 위한 딥러닝 (입문편)	1	Dive into deep learning	머신러닝, 딥러닝, 인공신경망의 개념 이해 딥러닝의 분류 및 응용 분야 이해 딥러닝 모델의 장/단점 이해
	2	딥러닝 코딩 환경 준비	머신러닝/딥러닝 워크플로우의 이해 파이썬 기반 머신러닝, 딥러닝 프레임워크의 이해 패키지 설치 및 기초 벡터 연산 설명
	3	딥러닝 모델 학습의 4steps	딥러닝 학습 과정의 단계별 이해
	4	데이터 시각화	데이터 시각화 Pandas, Matplotlib, Seaborn 패키지의 사용
	5	딥러닝을 이용한 분류 모델 실습	딥러닝 코드 실습을 통한 분류 문제 해결
	6	딥러닝을 이용한 회귀 모델 실습	딥러닝 코드 실습을 통한 회귀 문제 해결
	7	딥러닝 모델의 성능 향상 기법	딥러닝 기본 모델의 성능 향상 기법

5차시

1.5	딥러닝을 이용한 분류 모델 실습	딥러닝 코드 실습을 통한 분류 문제 해결

Contents

- **딥러닝을 이용한 문제 해결 과정**
- **딥러닝 코드 실습을 통한 분류 문제 해결**
- **다양한 데이터 전처리 및 모델링 과정 실습**

1.5 딥러닝을 이용한 분류 문제 실습

딥러닝을 이용한 문제 해결 과정

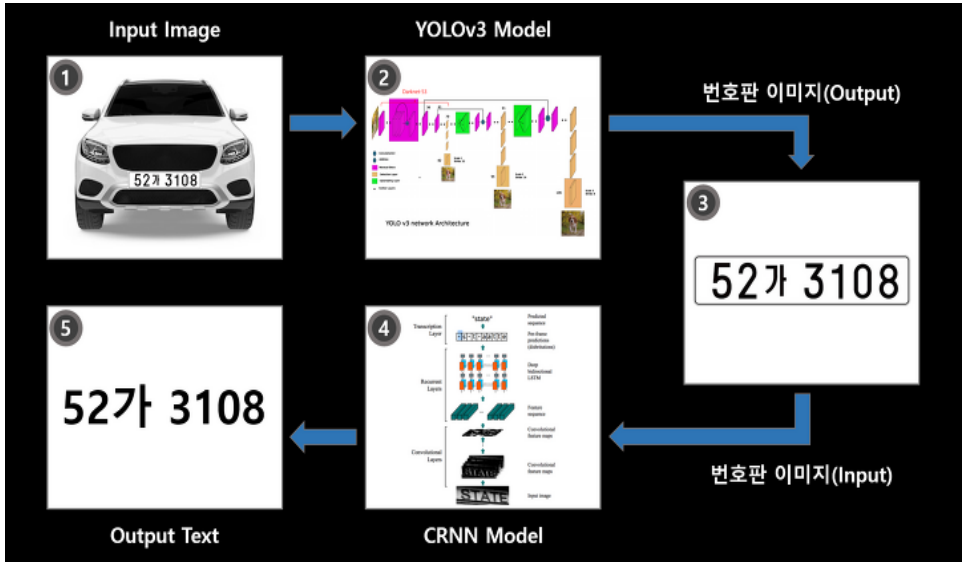
0. 문제 정의
1. 데이터 정의
2. 모델 정의
3. 모델 구현 및 학습
4. 모델 테스트 및 시각화

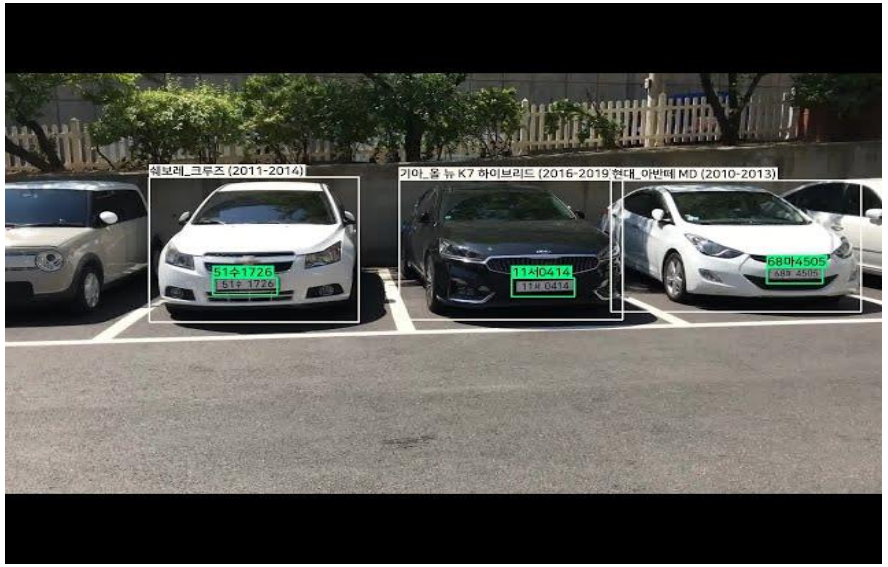
1.5 딥러닝을 이용한 분류 문제 실습

딥러닝을 이용한 분류 문제?

1. 이미지 인식 - 자동차 번호판 인식:

- 딥러닝은 자동차 번호판을 인식하는 데 적용됩니다. 주로 보안 카메라, 도로 교통 관제 등에서 사용되며, 딥러닝 모델은 차량 이미지를 입력으로 받아 번호판의 문자 및 숫자를 식별하여 차량을 추적하거나 교통 위반을 감지합니다.





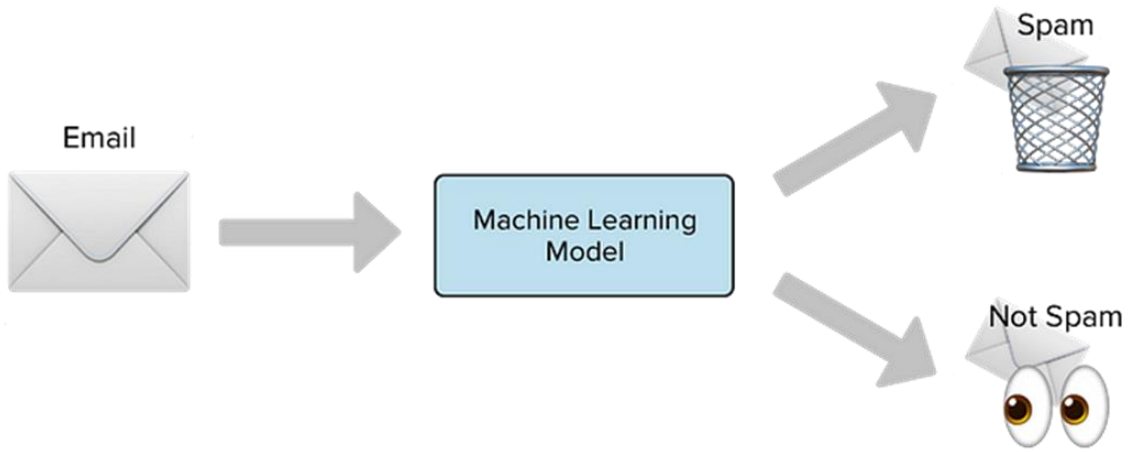
번호판 인식

1.5 딥러닝을 이용한 분류 문제 실습

딥러닝을 이용한 분류 문제?

2. 자연어 처리 - 스팸 필터링:

이메일 제공업체들은 딥러닝을 사용하여 스팸 이메일을 식별하고 차단하는데 활용합니다. 모델은 이메일의 내용을 분석하여 스팸으로 의심되는 메일을 식별하고, 사용자의 스팸함으로 이동시킵니다.



뉴스홈 | 최신기사

SKT, AI기술로 스팸문자 걸러낸다…변형 문구도 차단

송고시간 | 2018-10-11 09:26

채새롬 기자
기자 페이지

(서울=연합뉴스) 채새롬 기자 = SK텔레콤[017670]은 인공지능(AI) 핵심 기술인 딥러닝 기반의 스팸 문자 차단 기술을 기존 스팸 차단 서비스에 적용했다고 11일 밝혔다.

딥러닝은 사람의 뇌와 유사한 인공지능 신경망을 이용해 컴퓨터가 수많은 데이터를 학습한 후 그 안에서 일정한 패턴을 발견해 분류 모델을 만드는 기술이다.

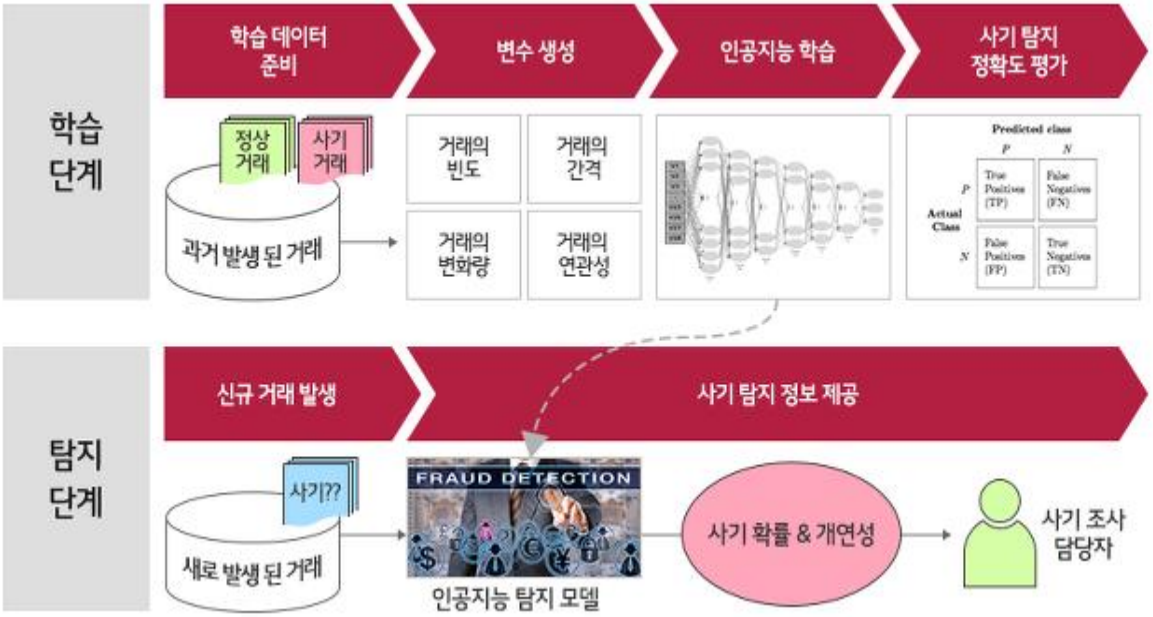
기존에는 SK텔레콤으로 전송되는 문자 수억 건을 한국인터넷진흥원에 신고된 스팸 문자 데이터베이스와 비교해 동일 문자를 필터링하는 방식으로 스팸을 걸렀다. 문자 발송자가 특정 문구를 바꾸면 정확한 필터링이 어려웠다.

1.5 딥러닝을 이용한 분류 문제 실습

딥러닝을 이용한 분류 문제?

3.금융 - 사기 탐지:

금융 기관들은 딥러닝을 사용하여 금융 거래 데이터를 분석하여 사기 행위를 탐지합니다. 사용자의 거래 패턴, 지리적 위치 등을 고려하여 이상 거래를 식별하고 예방합니다.



1.5 딥러닝을 이용한 분류 문제 실습

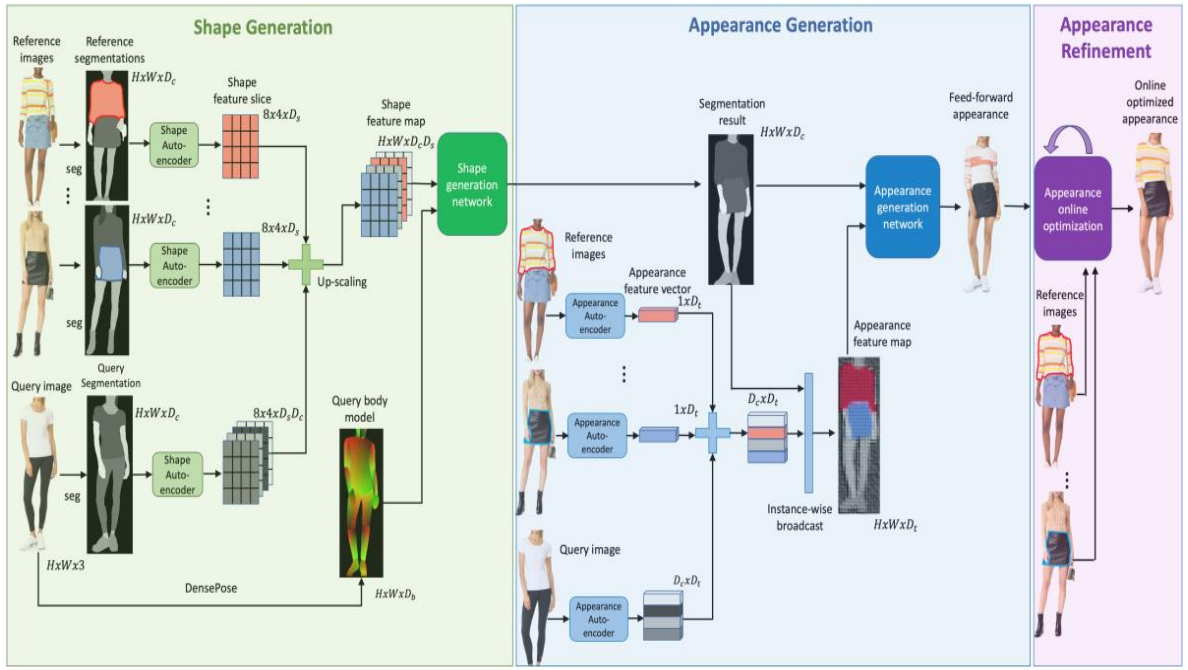
딥러닝을 이용한 분류 문제?

4.상품 추천 - 온라인 쇼핑:

온라인 쇼핑 플랫폼들은 딥러닝을 사용하여 사용자의 이전 구매 기록, 검색 기록, 클릭 패턴 등을 분석하여 맞춤 상품을 추천합니다.

상품 기반 AI 알고리즘 (10종)		방문자 이력 기반 AI 알고리즘 (6종)		통계형 알고리즘(12종)	
추천 알고리즘 유형	추천명	추천 알고리즘 유형	추천명	데이터 조회 기간	추천명
추천 알고리즘 유형 (딥러닝 활용)	함께 본 상품 함께 담은 상품 함께 구매한 상품 연관 카테고리 상품 딥러닝 연관 기획	딥러닝 추천	딥러닝 기반 다음에 볼 상품 딥러닝 기반 유사 고객 선호 상품	일(24h) 기준	많이 본 상품 TOP N 많이 본 기획전 TOP N 많이 담은 상품 TOP N 많이 구매한 상품 TOP N 주문 전환율 TOP N 추천 유입율 TOP N 추천 클릭률 TOP N 할인율 높은 상품 TOP N 카테고리 TOP N
고도화된 협업 필터링 추천	구매 패턴 유사 상품	클러스터링 추천	RFM 세그먼트 별 선호 상품		
딥러닝 추천	딥러닝 기반 유사 상품 이미지 기반 유사 상품	자연어 처리 X 클러스터링 추천	유사 취향 고객 선호 상품		
		개인화 추천(분류기법)	고객 프로파일링 기반 선호 상품	실시간(3h) 기준	실시간 많이 본 상품 TOP N 실시간 많이 구매한 상품 TOP N 실시간 카테고리 TOP N
자연어 처리	상품명 기반 유사 상품 검색어 추천	고도화된 협업 필터링 추천	구매 패턴 유사 고객 선호 상품		

딥러닝 기반 추천 알고리즘



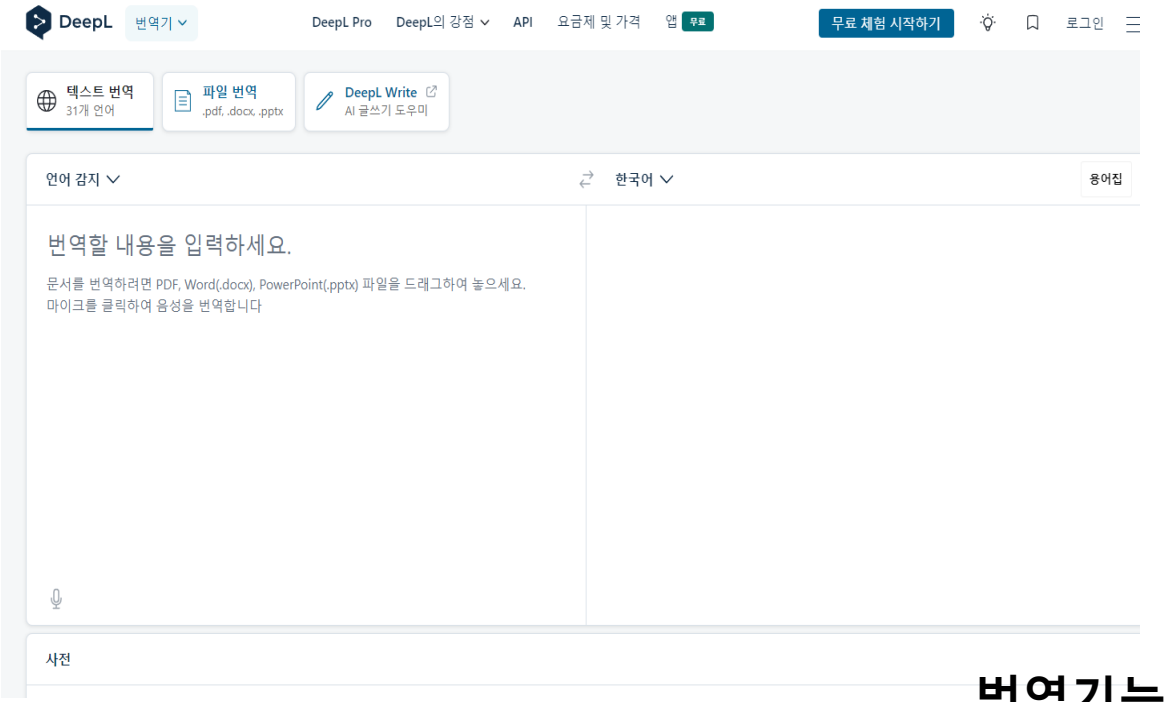
이미지 분석을 통한 추천

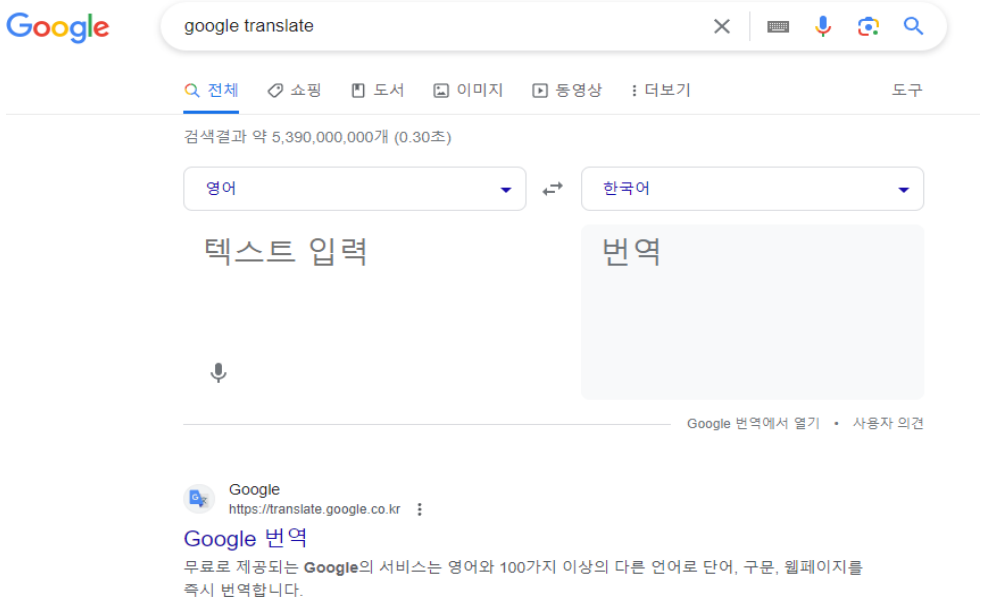
1.5 딥러닝을 이용한 분류 문제 실습

딥러닝을 이용한 분류 문제?

5.언어 번역 - 실시간 번역 서비스:

딥러닝을 사용한 기계 번역 서비스는 다양한 언어 간의 실시간 번역을 제공합니다. 이 서비스는 온라인 커뮤니케이션, 비즈니스, 여행 등 다양한 상황에서 사용됩니다.





번역기는 단어 분류기!

1.5 딥러닝을 이용한 분류 문제 실습

딥러닝을 이용한 분류 문제 예제

1. 딥러닝을 이용한 손글씨 데이터 분류 (이미지 데이터)
- 이미지 데이터의 기초 처리 방법
- 모델 설계 방법

2. 와인 데이터 분류 (CSV 데이터)
- CSV 데이터의 기초 처리 방법
- 기본적인 모델 설계 방법
- 딥러닝 모델 저장 방법

3. 심장 질환 여부 분류 모델 (CSV 데이터)
- CSV 데이터의 기초 처리 방법
- 데이터 시각화 방법
- Target(label)의 전처리 방법
- 이진 분류기의 구성 방법

4. 딥러닝을 이용한 당뇨 데이터 분류 (CSV 데이터)
- 케글 데이터 사용 방법
- 각종 데이터 전처리 방법

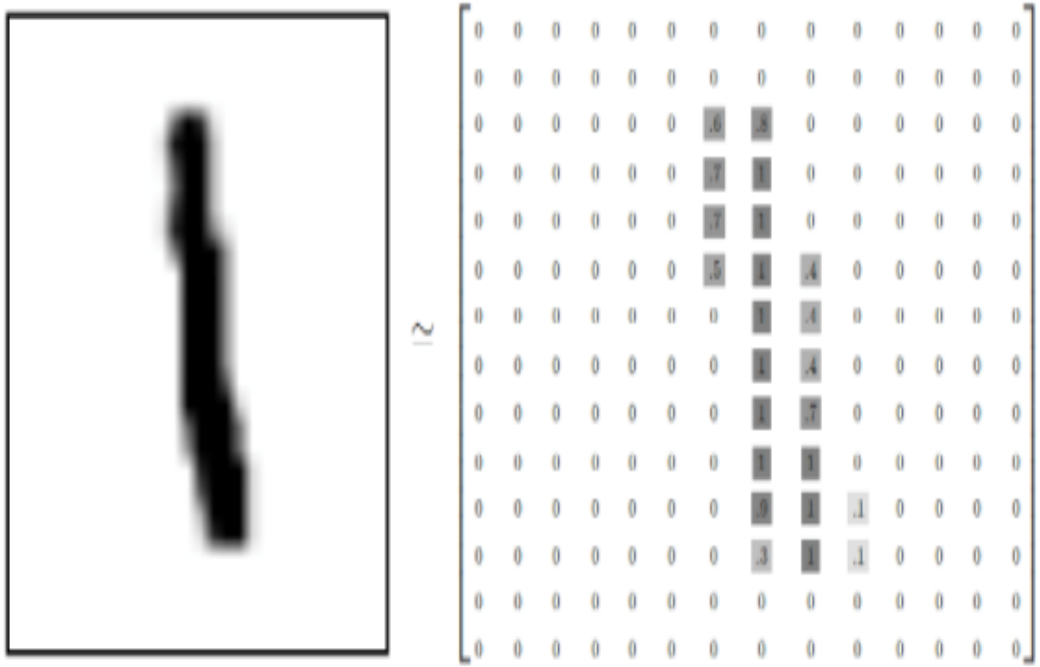
1.5 딥러닝을 이용한 분류 문제 실습

Point!

- 이미지 데이터의 기초 처리 방법
- 기본적인 모델 설계 방법

1.5 딥러닝을 이용한 손글씨 데이터 분류

데이터셋 설명



MNIST dataset

문제 정의:

- **0~9까지의 손글씨 데이터를 보고 어떤 숫자인지 맞추는 분류 모델 설계**

데이터 정의

Input data:

- [0,0,0,0,.....0.6,0.8,0,0,0,....0.7....0,0,0.1,0,0,..]로 이루어진 60000개 학습 데이터/10000개 테스트 데이터.

Label:

- [1,0,0,0,0,0,0,0,0,0]

1.5 딥러닝을 이용한 손글씨 데이터 분류

1. Module import

```
from keras.datasets import mnist
from keras.models import Sequential
from keras import layers
from keras.utils import to_categorical
import matplotlib.pyplot as plt
from keras.utils import plot_model
```

1.5 딥러닝을 이용한 손글씨 데이터 분류

2. Data preprocessing

```
# MNIST 데이터셋 불러오기
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# 이미지 데이터 준비하기 (모델에 맞는 크기로 바꾸고 0과 1사이로 스케일링)
train_images = train_images.reshape((60000, 28 * 28))
train_images = train_images.astype('float32') / 255
test_images = test_images.reshape((10000, 28 * 28))
test_images = test_images.astype('float32') / 255

print(train_labels.shape)
print(test_labels.shape)

# 레이블을 범주형으로 인코딩
train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)
```

mnist.load_data 함수:

- 손글씨 데이터셋을 불러오는 함수

train_images.reshape 함수:

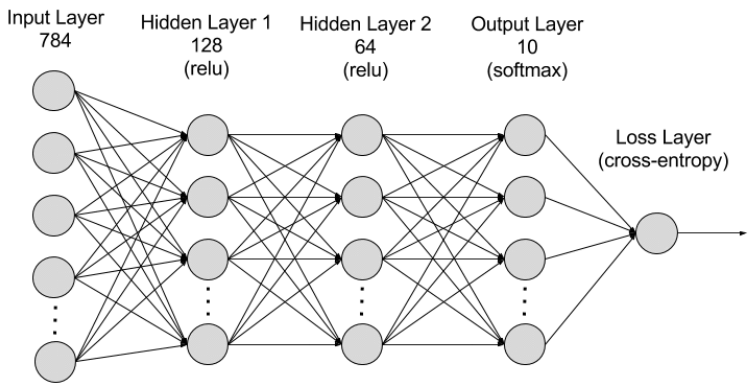
- 이미지를 한 줄의 vector로 형태 변경

1.5 딥러닝을 이용한 손글씨 데이터 분류

3. Model build

```
# 모델 정의하기 (여기에서는 Sequential 클래스 사용)
model = Sequential()
model.add(layers.Dense(100, activation='relu', input_shape=(28 * 28,)))
model.add(layers.Dense(150, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

- model.add() : 신경망을 쌓는 함수



1.5 딥러닝을 이용한 손글씨 데이터 분류

4. Model compile & training

```
# 모델 컴파일 하기
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# fit() 함수로 모델 훈련 시키기
hist=model.fit(train_images, train_labels, epochs=3, batch_size=64, verbose=1)
```

- Epochs: 전체 데이터를 몇 번 반복하여 학습시킬지 정하는 파라미터
- batch_size: 한 번에 학습하는 데이터의 개수를 지정하는 파라미터
- model.fit은 return값을 주면 학습 과정의 loss 그래프를 그릴 수 있음 (옵션)

1.5 딥러닝을 이용한 손글씨 데이터 분류

5. Model test

```
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=0)
print('loss:',test_loss)
print('test_accuracy:',test_acc)
```

딥러닝 분류 문제를 위한 손실함수 정리

DNN을 이용한 Iris data 분류 - 1.4장 (class 0~2) 문제

```
model.compile(loss='categorical_crossentropy',
              optimizer='Adam',
              metrics=['accuracy'])
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=50, verbose=1)
```

DNN을 이용한 손글씨 분류 1.5장 (class 0~9) 문제

```
# 모델 컴파일 하기
model.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])

# fit() 함수로 모델 훈련 시키기
model.fit(train_images, train_labels,epochs=3, batch_size=64, verbose=1)
```

Crossentropy???

딥러닝 분류 문제를 위한 손실함수 정리

1. 크로스엔트로피 손실함수 (Crossentropy function)

- 분류 문제에 맞게 고안된 목적 함수 (손실 함수, 에러 함수...etc)

- 엔트로피(entropy): information theorem(정보 이론) 에서 놀람도를 의미

Ex) 주사위의 각 눈이 나올 확률은 1/6 전혀 놀라운 결과가 아님 (놀람도 낮음)
>> 하지만 내가 모델링한 주사위 3 눈이 1/2로 나온다면 놀람도가 커지고 이는 모델링에 에러가 있음을 암시.

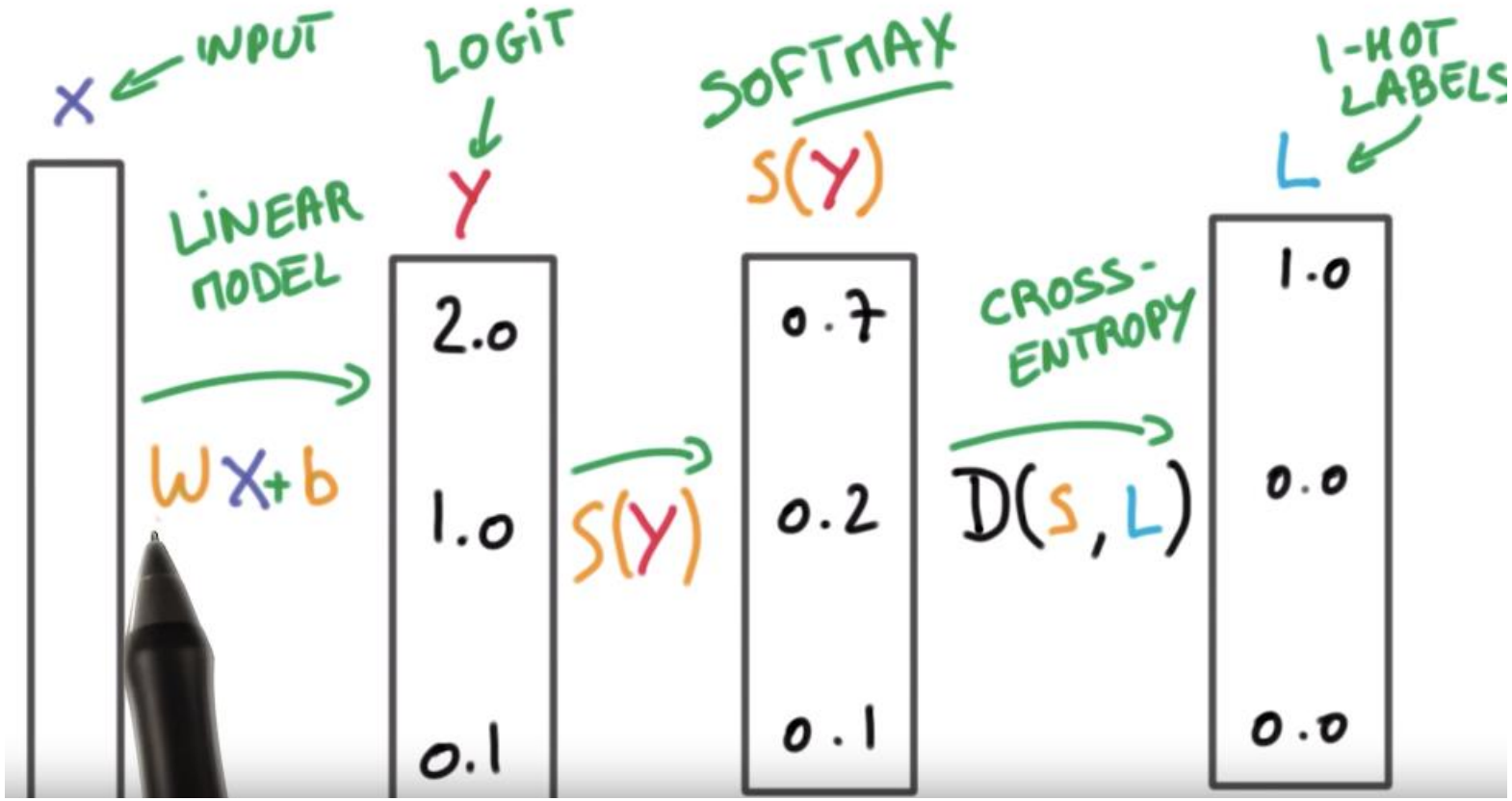
<직관적 설명>
- 모델링 결과의 엔트로피를 낮춘다는 것은 어떤 모델이 학습을 통해 목적함수와 같은 output을 산출하여 놀람도를 낮춘다는 것과 마찬가지.

$$H(P,Q) = -\mathbb{E}_{X\sim P}[\log Q(x)] = -\sum P(x)\log Q(x)$$

P: 올바른 확률분포 / Q: 내가 예측한 확률분포

딥러닝 분류 문제를 위한 손실함수 정리

1. 크로스엔트로피 손실함수 (Crossentropy function)



- S(Y)분포(모델의 예측 결과) 와 L분포(실제 정답)가 최대한 유사해야 함

딥러닝 분류 문제를 위한 손실함수 정리

1. 크로스엔트로피 손실함수 계산 예시(Crossentropy function)

$$\text{softmax(input)} = \begin{bmatrix} 0.2 \\ 0.7 \\ 0.1 \end{bmatrix} \Rightarrow \begin{aligned} Q(X = c_1) &= 0.2 \\ Q(X = c_2) &= 0.7 \\ Q(X = c_3) &= 0.1 \end{aligned} \Rightarrow \begin{aligned} H(P, Q) &= -\sum P(x) \log Q(x) \\ &= -(0 \cdot \log 0.2 + 1 \cdot \log 0.7 + 0 \cdot \log 0.1) \\ &= -\log 0.7 \approx 0.357 \end{aligned}$$

1. 모델의 예측값 (Q 분포)

2. 정답분포(P)와의 비교

$$\begin{aligned} D_{\mathrm{KL}}(p \parallel q) &= \sum_x p(x) \log \frac{p(x)}{q(x)} \\ &= \sum_x p(x) \log p(x) - \sum_x p(x) \log q(x) \\ &= -\mathrm{H}(p) + \mathrm{CE}(p, q) \end{aligned}$$

- 분류 문제에서 Cross-entropy는 두 분포의 Kullback–Leibler divergence를 최소화하는것과 같은 문제

Recommended Reference:
https://3months.tistory.com/436
https://theeluwin.postype.com/post/6080524

딥러닝 분류 문제를 위한 손실함수 정리

2. Binary Crossentropy function

$$l = -(\, y\, log(p) \;+\; (1-y)\, log(1-p)\,)$$

- 이진분류 문제의 경우
- 모델의 마지막 node의 output(예측값): p
- Label (정답) : y (0 또는 1)
- 정답이 0이면 p값이 0에 가까워져야 loss가 최소, 반대도 마찬가지 >> 직접 숫자를 넣어보자

1.5 딥러닝을 이용한 와인 등급 분류

Point!

- 다양한 특성을 가진 데이터의 분류 방법
- 기본적인 모델 설계 방법
- 학습 과정의 시각화 방법 (loss, accuracy flow)

1.5 딥러닝을 이용한 와인 등급 분류

1. 크뤼부르주아 샤토 브리에의 세컨드 와인 레오드 브리에
2. 그랑크뤼 2등급 샤토 비방의 세컨드 와인 비방
3. 크뤼부르주아 샤토 샤스 스플린의 세컨드 와인 오라뚜아드
4. 그랑크뤼 2등급 샤토 그오 라로즈의 세컨드 와인 라로즈 드 크뤼오
5. 그랑크뤼 1등급 샤토 마고의 세컨드 와인 빠삐용 블랑 드

6. 크뤼부르주아 샤토 브리에
7. 크뤼부르주아 샤토 샤스 스플린
8. 그랑 크뤼 2등급 그랑크뤼 1등급 샤토 비방
9. 그랑크뤼 1등급 샤토 마고의 세컨드 와인 빠삐용 루즈드
10. 그랑크뤼 2등급 샤토 그리오 라로즈

1.5 딥러닝을 이용한 와인 등급 분류

문제 정의:

- **와인의 화학적 특성을 기반으로 해당 와인이 세 가지 클래스 중 어느 것에 속하는지를 분류하는 문제**

데이터 정의:
특징변수:
- Alcohol (알코올 함량): 와인의 알콜 함량을 나타내는 특성입니다.
- Malic Acid (말릭 산): 와인에 함유된 말릭 산의 양을 나타내는 특성입니다.
- Ash (회분): 와인의 회분(잔여 물질)을 나타내는 특성입니다.
- Alcalinity of Ash (회분의 알칼리도): 와인의 회분이 얼마나 알칼리성인지를 나타내는 특성입니다.
- Magnesium (마그네슘): 와인에 함유된 마그네슘의 양을 나타내는 특성입니다.
- Total Phenols (총 페놀): 와인에 함유된 총 페놀의 양을 나타내는 특성입니다.
- Flavanoids (플라보노이드): 와인에 함유된 플라보노이드의 양을 나타내는 특성입니다.
- Nonflavanoid Phenols (비 플라보노이드 페놀): 와인에 함유된 비 플라보노이드 페놀의 양을 나타내는 특성입니다.
- Proanthocyanins (프로안소시아닌): 와인에 함유된 프로안소시아닌의 양을 나타내는 특성입니다.
- Color Intensity (색 강도): 와인의 색의 강도를 나타내는 특성입니다.
- Hue (색상): 와인의 색상을 나타내는 특성입니다.
- OD280/OD315 of diluted wines (희석 와인의 OD280/OD315 비율): 희석된 와인의 광학 밀도의 비율을 나타내는 특성입니다.
- Proline (프로린): 와인에 함유된 프로린의 양을 나타내는 특성입니다.

타겟 변수(label):
- class: 0,1,2로 와인의 클래스를 의미합니다

1.5 딥러닝을 이용한 와인 등급 분류

1. Module import & data preprocessing

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from keras.callbacks import History

# 데이터 로드
wine_data = load_wine()
X = wine_data.data
y = wine_data.target

# 특성 정규화
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 레이블을 one-hot 인코딩
y_one_hot = to_categorical(y)

# 훈련 데이터와 테스트 데이터로 분리
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_one_hot, test_size=0.2, random_state=42)
```

1.5 딥러닝을 이용한 와인 등급 분류

2. Data visualization

```
# 클래스 분포 확인
sns.countplot(x=y)
plt.title('Class Distribution')
plt.show()

# 특성 간 상관관계 시각화
df = pd.DataFrame(X, columns=wine_data.feature_names)
plt.figure(figsize=(15, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

# 특성별 분포 시각화
df.hist(figsize=(15, 10), bins=20)
plt.suptitle('Feature Distribution', x=0.5, y=0.92, ha='center', fontsize='large')
plt.show()
```

1.5 딥러닝을 이용한 와인 등급 분류

3. Model build & training

```
# 케라스 모델 생성
model = Sequential()
model.add(Dense(16, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(3, activation='softmax'))  # 클래스의 개수에 맞게 출력 뉴런 수 조정

# 모델 컴파일
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

# 모델 학습
history = History()  # 학습 과정을 저장하기 위한 객체
model.fit(X_train, y_train, epochs=50, batch_size=8, validation_data=(X_test,
y_test), callbacks=[history])
```

1.5 딥러닝을 이용한 심장 질환 데이터 분류

4. 학습 과정 시각화

```
# 학습 과정 시각화
plt.figure(figsize=(15, 5))

# 정확도 시각화
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

# 손실 시각화
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

1.5 딥러닝을 이용한 심장 질환 데이터 분류

Point!

- CSV 데이터의 기초 처리 방법
- 데이터 시각화 방법
- Target(label)의 전처리 방법
- 이진 분류기의 구성 방법

1.5 딥러닝을 이용한 심장 질환 데이터 분류

문제 정의:
- 심장 질환 여부를 판별하기 위한 데이터셋으로, 환자들의 심장 관련 정보를 가지고 심장 질환 유/무 예측

데이터 정의:
"Heart Disease UCI"dataset

- Age: 환자의 나이
- Sex: 환자의 성별 (1: 남성, 0: 여성)
- Cp: 가슴 통증 유형 (chest pain type)
- Trestbps: 안정시 혈압 (resting blood pressure)
- Chol: 혈청 콜레스테롤 수치 (serum cholesterol)
- fbs: 공복 혈당 수치 (fasting blood sugar)
- Restecg: 안정시 심전도 결과 (resting electrocardiographic results)
- Thalach: 최대 심박동 수 (maximum heart rate achieved)
- Exang: 운동 유발 협심증 여부 (exercise induced angina)
- Oldpeak: 운동으로 인한 ST depression
- Slope: 운동의 기울기
- Ca: 주요 혈관의 수 (number of major vessels colored by fluoroscopy)
- Thal: Thalassemia 유형 (thalassemia type)
- Target: 0(심장질환없음)/1(심장질환있음)

1.5 딥러닝을 이용한 심장 질환 데이터 분류

1. Module import 및 데이터 로드

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical

# 데이터 로드
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/processed.cleveland.data"
names = ["age", "sex", "cp", "trestbps", "chol", "fbs", "restecg", "thalach", "exang", "oldpeak", "slope", "ca", "thal", "target"]
df = pd.read_csv(url, names=names, na_values="?")
```

1.5 딥러닝을 이용한 심장 질환 데이터 분류

2. 데이터 전처리

```
# 결측치 제거
df = df.dropna()

# target 열을 0과 1로 변환
df['target'] = df['target'].apply(lambda x: 1
if x > 0 else 0) #이부분 전처리 과정 중요!

# 특성과 레이블 분리
X = df.iloc[:, :-1].values
y = df["target"].values

# 훈련 데이터와 테스트 데이터로 분리
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)

# 데이터 정규화
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

코드 설명

- df['target'] = df['target'].apply(lambda x: 1 if x > 0 else 0)

- Python lambda 함수:
Iterable한 요소에 대해 반복적인 작업 수행

1.5 딥러닝을 이용한 심장 질환 데이터 분류

3. 데이터 시각화

```
# 클래스 분포 확인
sns.countplot(x='target', data=df)
plt.title('Class Distribution')
plt.show()

# 특성 간 상관관계 시각화
plt.figure(figsize=(15, 10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

# 특성별 분포 시각화
sns.set(style="whitegrid")
df_long = pd.melt(df, id_vars="target", var_name="Features", value_name='Values')
plt.figure(figsize=(15, 10))
sns.boxplot(x='Features', y='Values', hue='target', data=df_long)
plt.xticks(rotation=45)
plt.title('Features Distribution by Target')
plt.show()

# Pair Plot
sns.pairplot(df, hue="target", palette="husl")
plt.suptitle('Pair Plot of Features by Target', y=1.02)
plt.show()
```

1.5 딥러닝을 이용한 심장 질환 데이터 분류

4. 데이터 전처리2

```
# 특성과 레이블 분리
X = df.iloc[:, :-1].values
y = df["target"].values

# 훈련 데이터와 테스트 데이터로 분리
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 데이터 정규화
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

1.5 딥러닝을 이용한 심장 질환 데이터 분류

5. 모델 구성 및 훈련

```
# 케라스 모델 생성
model = Sequential()
model.add(Dense(16, input_dim=X_train.shape[1], activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

# 모델 컴파일
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# 모델 훈련
model.fit(X_train, y_train, epochs=50, batch_size=8, validation_data=(X_test, y_test))
```

6. 모델 테스트

```
# 테스트 데이터에서 성능 평가
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss}, Test Accuracy: {accuracy}')
```

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

Point!

- 케글 데이터 사용 방법
- 각종 데이터 전처리 방법

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

문제 정의:

- **환자의 데이터를 기반으로 당뇨병 발병 여부를 예측.**

데이터 정의:

특징변수:
- Pregnancies (임신 횟수): 환자가 현재까지 임신한 횟수
- Glucose (혈당 수치): 2시간 동안의 경구 포도당 내성 검사에서 혈장 포도당 농도
- BloodPressure (혈압): 환자의 혈압 (mm Hg)
- SkinThickness (피부 두께): 삼두근 피부 주름의 두께 (mm)
- Insulin (인슐린): 2시간 동안의 혈청 인슐린 (mu U/ml)
- BMI (체질량 지수): 체중(kg) / 키(m)^2
- DiabetesPedigreeFunction (당뇨가족력 지수): 당뇨병 혈통의 가중치 함수
- Age (나이): 환자의 나이 (세)

타겟 변수(label):
- class: 0,1로 0은 당뇨병이 없음을, 1은 당뇨병이 있음을 나타냅니다.

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

당뇨병 : 실제 혈당 수치가 상승해 발생하는 만성 질환

1형 당뇨 :

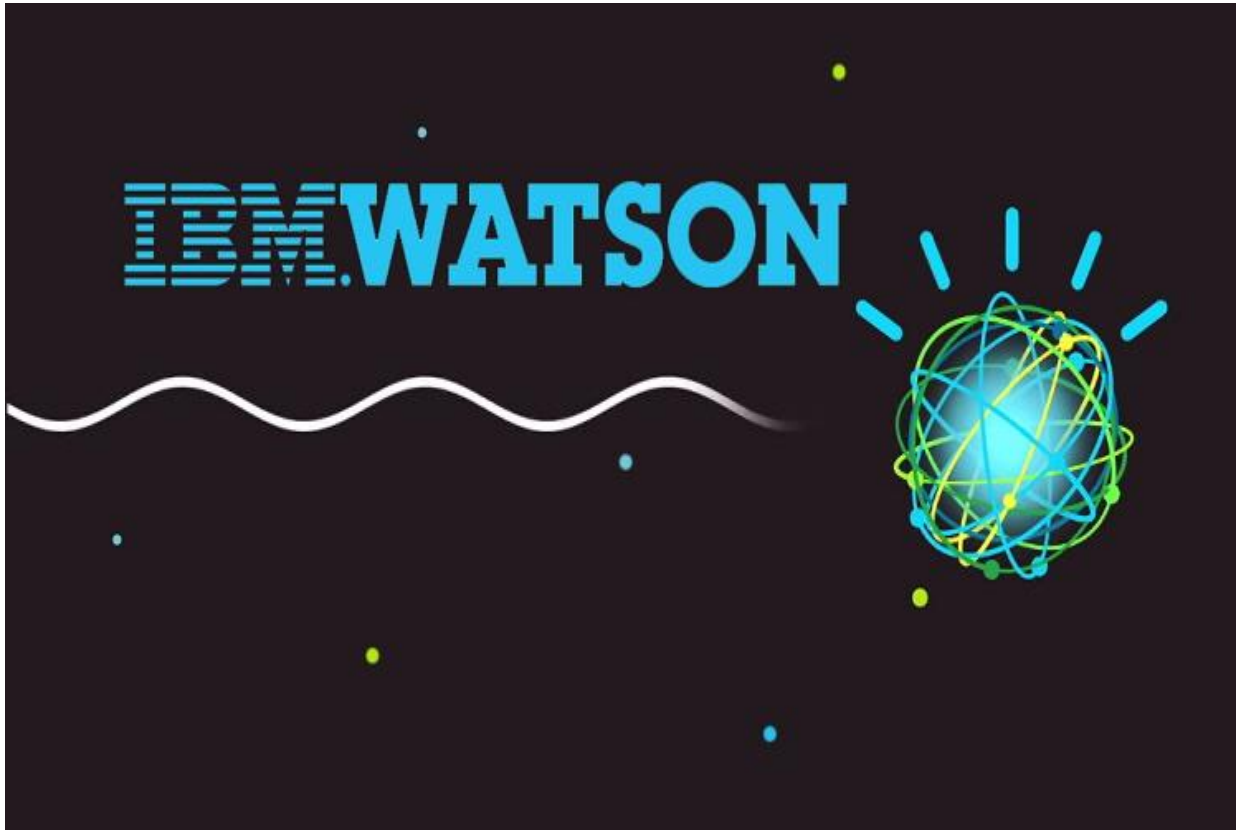
- 체내에서 인슐린을 생산하지 못하여 발생
- 당뇨병 중 상대적으로 희귀하고 당뇨병 중 5%를 차지

2형 당뇨 :

- 점진적인 인슐린 저항 때문에 발생
- 과체중, 운동 부족, 잘못된 식습관 때문에 발생
- 조기 진단으로 예방하고 치료 가능

>> 당뇨 예측이 필요!

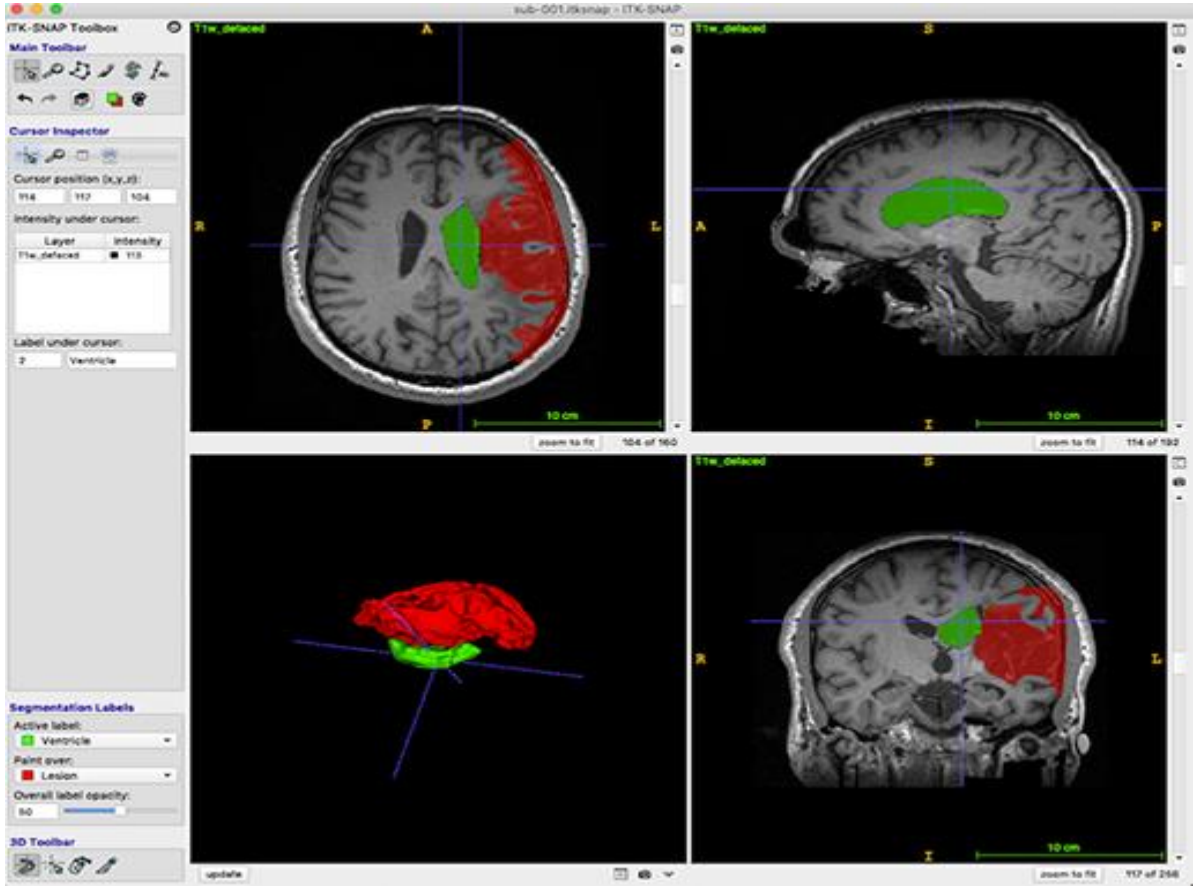
1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류



의료분야의 인공지능 이용

- 챗봇 기반 의료 지식 상담

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류



의료분야의 인공지능 이용

- Image segmentation 기반 의사 결정

Kaggle???

- 데이터 과학 및 기계 학습 프로젝트를 위한 플랫폼
- 데이터 과학자, 기계 학습 엔지니어, 연구자 및 기타 관련 분야의 전문가들이 데이터를 탐색하고 모델을 개발하며 경쟁하고 협력할 수 있는 온라인 커뮤니티.
- 케글은 데이터 세트를 제공하고 사용자들이 해당 데이터를 기반으로 예측 모델을 개발하고 경쟁하게 하여, 데이터 과학 및 기계 학습 분야에서의 지식을 공유하고 발전시키는 데 기여

데이터 다운로드 및 load

예제 프로젝트에 필요한 데이터셋을 내려받으려면 다음 과정을 거쳐야 한다.

1. https://www.kaggle.com/uciml/pima-indians-diabetes-database에 접속한다.
2. 상단 탭 메뉴 중 Download를 클릭한다.
3. 캐글 계정으로 로그인하거나 가입하라는 화면이 나올 것이다. 캐글 계정이 없다면 하나 만들자.
4. 로그인을 완료하면 자동으로 내려받는다. (자동으로 시작하지 않는다면 Download를 한 번 더 클릭하자.)
5. 다 받았다면 ZIP 파일 압축을 풀고 diabetes.csv 파일을 프로젝트 폴더로 옮겨놓자.

```
try:
    input_data = pd.read_csv('diabetes.csv')
except:
    print("""
      Dataset not found in your computer.
      Please follow the instructions in the link below to download the dataset:
      https://raw.githubusercontent.com/PacktPublishing/Neural-Network-Projects-with-Python/master/chapter2/how_to_download_the_dataset.txt
      """)
    quit()

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
```

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

데이터 다운로드 및 load

day2.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

..
sample_data
diabetes.csv

```
[5] pip install matplotlib==3.0.2
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: matplotlib==3.0.2 in /usr/local/lib/python3.8/dist-packages (3.0.2)
Requirement already satisfied: numpy>=1.10.0 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.0.2) (1.21.6)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.0.2) (3.0.9)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.0.2) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.0.2) (1.4.4)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.8/dist-packages (from matplotlib==3.0.2) (2.8.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.8/dist-packages (from python-dateutil>=2.1->matplotlib==3.0.2) (1.15.0)
```

```
[7] import matplotlib
    #matplotlib.use("TkAgg")
    #from utils import preprocess
    import pandas as pd
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import roc_curve
    from keras.models import Sequential
    from keras.layers import Dense
    import matplotlib.pyplot as plt
    import numpy as np
    np.random.seed(16)
```

```
try:
    input_data = pd.read_csv('diabetes.csv')
```

- 패키지 설치: pip install <패키지명>
- 구글드라이브에 csv파일 업로드

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

1. Module import 및 data load

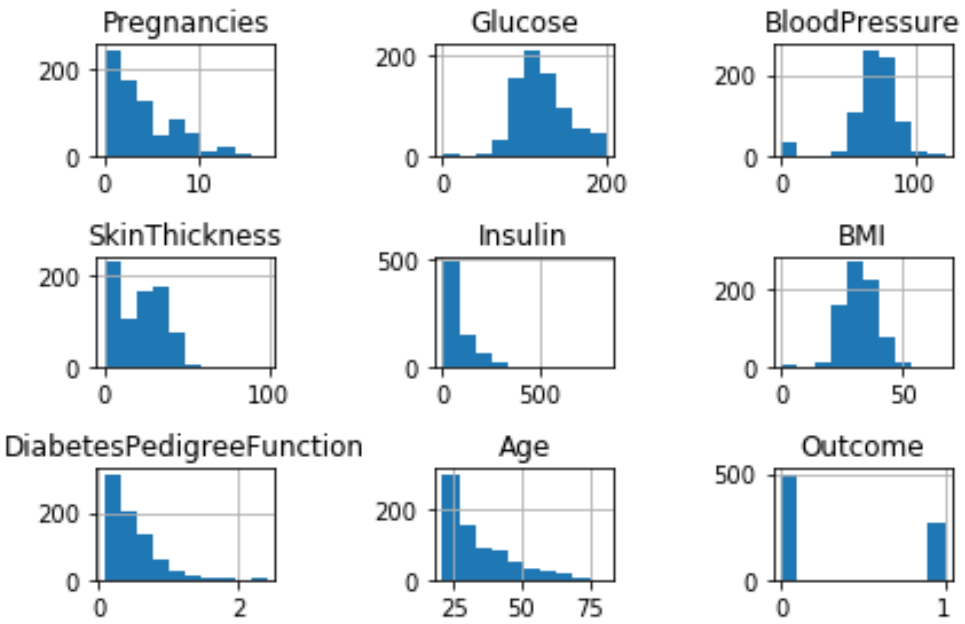
```
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
import warnings
warnings.filterwarnings("ignore")

input_data=pd.read_csv('diabetes.csv')
input_data
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

2. 데이터 시각화

```
# 히스토그램을 살펴본다
input_data.hist()
plt.tight_layout()
plt.show()
```



히스토그램

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

3. 데이터 전처리

전처리시 고려해야 할 사항

- 변수의 범위를 수정해야한다 (표준화 문제)
- 데이터 안의 결측치를 해결해야 한다 (이상/결측치 문제)

코드 설명

preprocessing.scale():

- 데이터들의 scale을 맞추어주는 함수

df.fillna():

- 데이터 안의 결측치를 해결하는 함수

```python
def preprocess(df):
    print('-----------------------------------------------')
    print("Before preprocessing")
    print("Number of rows with 0 values for each variable")
    for col in df.columns:
        missing_rows = df.loc[df[col]==0].shape[0]
        print(col + ": " + str(missing_rows))
    print('-----------------------------------------------')

    # 0이 아닌 값의 평균을 계산해 0 값을 대체
    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)
    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())

    print('-----------------------------------------------')
    print("After preprocessing")
    print("Number of rows with 0 values for each variable")
    for col in df.columns:
        missing_rows = df.loc[df[col]==0].shape[0]
        print(col + ": " + str(missing_rows))
    print('-----------------------------------------------')

    # 데이터 표준화
    df_scaled = preprocessing.scale(df)
    df_scaled = pd.DataFrame(df_scaled, columns=df.columns)
    df_scaled['Outcome'] = df['Outcome']
    df = df_scaled
    return df

# 데이터 전처리
preprocessed_df = preprocess(input_data)
```

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

3. 데이터 전처리

전처리 전/후

```
---------------------------------------------
Before preprocessing
Number of rows with 0 values for each variable
Pregnancies: 111
Glucose: 5
BloodPressure: 35
SkinThickness: 227
Insulin: 374
BMI: 11
DiabetesPedigreeFunction: 0
Age: 0
Outcome: 500
---------------------------------------------
---------------------------------------------
After preprocessing
Number of rows with 0 values for each variable
Pregnancies: 111
Glucose: 0
BloodPressure: 0
SkinThickness: 0
Insulin: 0
BMI: 0
DiabetesPedigreeFunction: 0
Age: 0
Outcome: 500
---------------------------------------------
```

```python
def preprocess(df):
    print('---------------------------------------------')
    print("Before preprocessing")
    print("Number of rows with 0 values for each variable")
    for col in df.columns:
        missing_rows = df.loc[df[col]==0].shape[0]
        print(col + ": " + str(missing_rows))
    print('---------------------------------------------')

    # 0이 아닌 값의 평균을 계산해 0 값을 대체
    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)
    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())

    print('---------------------------------------------')
    print("After preprocessing")
    print("Number of rows with 0 values for each variable")
    for col in df.columns:
        missing_rows = df.loc[df[col]==0].shape[0]
        print(col + ": " + str(missing_rows))
    print('---------------------------------------------')

    # 데이터 표준화
    df_scaled = preprocessing.scale(df)
    df_scaled = pd.DataFrame(df_scaled, columns=df.columns)
    df_scaled['Outcome'] = df['Outcome']
    df = df_scaled
```

⋮

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

4. 데이터셋 분할

```
# 데이터를 훈련 데이터셋과 테스트 데이터셋으로 분할
X = preprocessed_df.loc[:, preprocessed_df.columns != 'Outcome']
y = preprocessed_df.loc[:, 'Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

코드 설명

1. 전체 dataset에서 입력 데이터 / 정답 분리

2. train_test_split() 함수:

- dataframe 안의 데이터를 train/test로 분할

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

5. Model build

```
# 케라스 신경망을 구축
model = Sequential()
model.add(Dense(32, activation='relu', input_dim=8))
model.add(Dense(16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

6. Model compile & training

```
model.compile(optimizer='adam', loss='binary_crossentropy
', metrics=['accuracy'])
model.fit(X_train, y_train, epochs=200, verbose=0)
```

코드 설명

Sequential():

- 레이어를 쌓기 위한 틀

model.compile() 함수:

- 모델 구성을 위한 함수

▷loss 파라미터:

- 손실함수를 지정하는 파라미터.

▷optimizer:

- Adam, rmsprop, sgd 등.

▷metric 파라미터:

- 정확도를 계산하는 파라미터.

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

7. Test accuracy check

```
loss, accuracy = model.evaluate(X_test, y_test)
print("Accuracy = {:.2f}".format(accuracy))
```

코드 설명

evaluate 함수:

- Train/test accuracy를 나누어서 평가

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

Confusion matrix

```
# 결과 - 혼동 행렬
y_test_pred = model.predict(X_test)
y_test_pred = np.where(y_test_pred > 0.5, 1 , 0)
c_matrix = confusion_matrix(y_test, y_test_pred)
ax = sns.heatmap(c_matrix, annot=True, xticklabels=['No Diabetes',
 'Diabetes'],
                 yticklabels=['No Diabetes', 'Diabetes'], cbar=Fal
se, cmap='Blues')
ax.set_xlabel("Prediction")
ax.set_ylabel("Actual")
plt.show()
plt.clf()
```

코드 설명

np.where 함수:

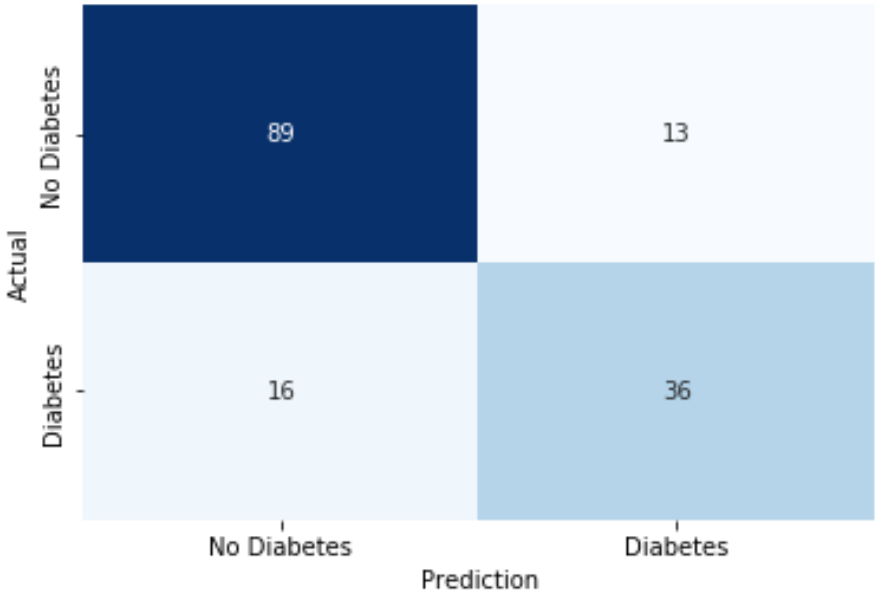
- Predict 값을 category (0,1)로 할당하는 조건
- 여기에선 0.5 이상=1, 그 이하=0으로 할당

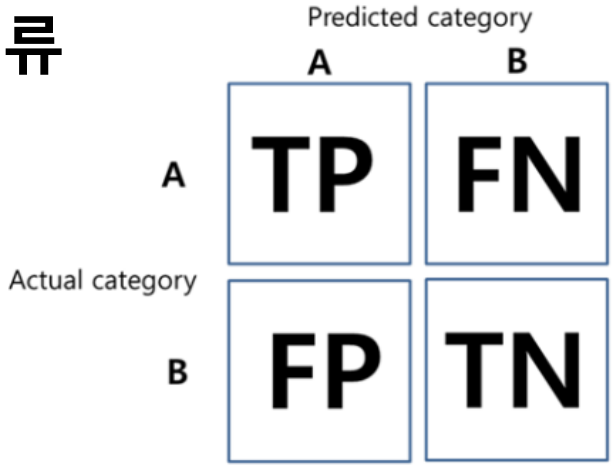
sns.heatmap 함수:

- Seaborn 패키지를 써서 시각화

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

혼동행렬 출력값





TP (True Positive) :

관심 범주를 정확하게 분류함

TN (True Negative) :

관심 범주가 아닌것을 정확하게 분류함

FP (False Positive) :

관심 범주로 잘못 분류함

FN (False Negative) :

관심 범주가 아닌 것으로 잘못 분류함

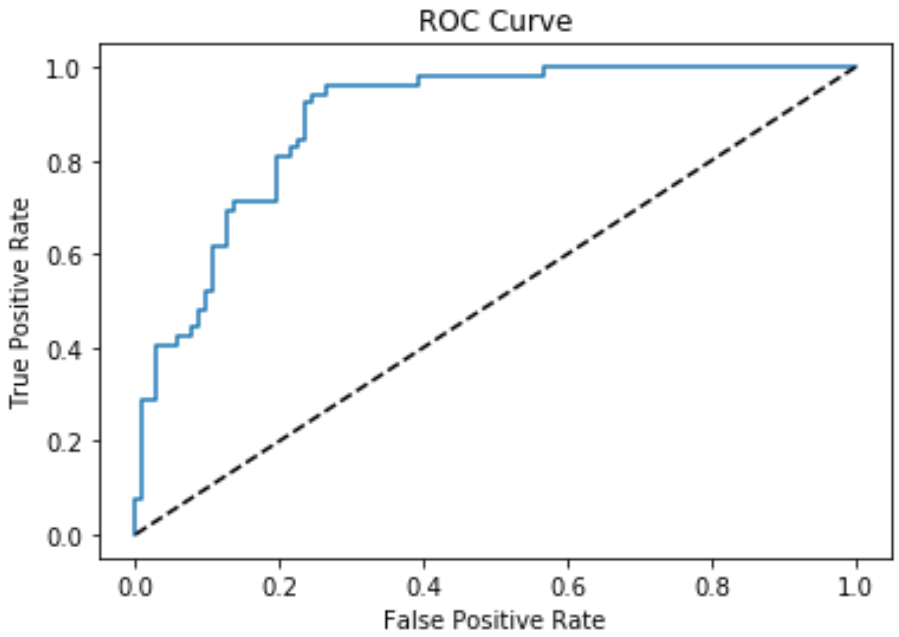
좋은 모델이란?

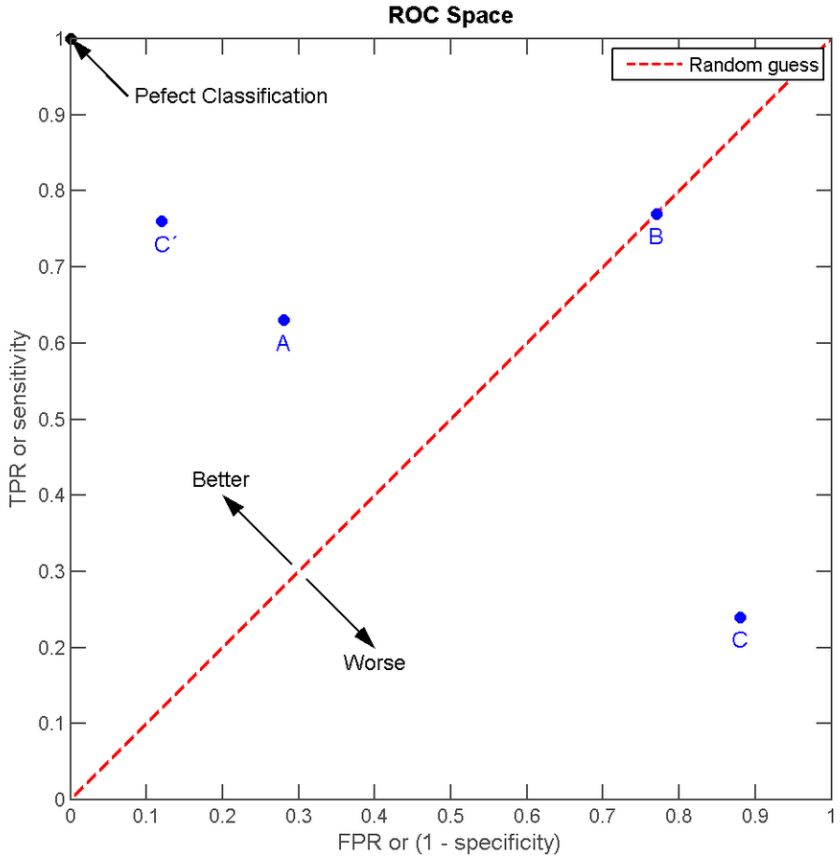
- FP, FN이 최소 / TP, TN이 최대!

1.5 딥러닝을 이용한 당뇨병 환자 데이터 분류

결과분석 2 : ROC 곡선

ROC 출력값





- 곡선이 왼쪽 위로 가면서 아래 면적(AUC)이 넓을수록 클래스 분류 정확도가 높음을 의미

감사합니다