|  |  |  |  |
| --- | --- | --- | --- |
| **4주. Regression** | | | |
| 학번 | 32200327 | 이름 | 김경민 |

※ 이번 실습에 사용된 데이터셋은 공지에 있는 데이터셋 압축파일에 포함되어 있음

BostonHousing 데이터셋은 보스턴 지역의 지역정보 및 평균주택 가격 (medv) 정보를 담고 있다.

**BostonHousing dataset을 가지고 단순 선형 회귀 분석을 하고자 한다.**

|  |
| --- |
| Q1 **lstat** (소득분위가 하위인 사람들의 비율) 로 **medv** (주택가격)을 예측하는 단순 선형회귀 모델을 만드시오 (tain:test = 7:3, random\_state는 1234). 모델의 내용을 보이시오 |

**Source code :**

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  import pandas as pd  import numpy as np  from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  #prepare dataset  data = \  pd.read\_csv("C:/Users/user/PycharmProjects/deepLearning/data/BostonHousing.csv")  lstat = data['lstat']  medv = data['medv']  #data frame ro np.array  lstat = np.array(lstat).reshape(506,1)  medv = np.array(medv).reshape(506,1)  #Split the data into train/test sets  train\_X, test\_X, train\_y, test\_y = \  train\_test\_split(lstat, medv, test\_size=0.3, random\_state=1234)  #Define model  model = LinearRegression()  #Train the model  model.fit(train\_X,train\_y)  print(model)  print('Cofficients: {0:.2f}, Intercept {1:.3f}'\  .format(model.coef\_[0][0], model.intercept\_[0])) |

**실행화면 캡쳐:**



|  |
| --- |
| Q2. 모델에서 만들어진 회귀식을 쓰시오 (medv = W x lstat + b 의 형태) |

medv = -0.94 x lstat + 34.321

|  |
| --- |
| Q3. 회귀식을 이용하여 lstat 의 값이 각각 2.0, 3.0, 4.0, 5.0 일 때 medv 의 값을 예측하여 제시하시오. |

**Source code :**

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  print(model.predict([[2.0]]))  print(model.predict([[3.0]]))  print(model.predict([[4.0]]))  print(model.predict([[5.0]])) |

**실행화면 캡쳐:**



|  |
| --- |
| Q4. 모델에 대해 rooted mean square error (RMSE)와 R2score를 보이시오 |

**Source code :**

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  from sklearn.metrics import mean\_squared\_error, r2\_score  pred = model.predict(test\_X)  print('Mean square error: {0:.2f}'.\  format(mean\_squared\_error(test\_y,pred)))  print('Coefficient of determination: %.2f'% r2\_score(test\_y, pred)) |

**실행화면 캡쳐:**



**BostonHousing dataset을 가지고 다중 선형 회귀 분석을 하고자 한다.**

|  |
| --- |
| Q5. **lstat** (소득분위가 하위인 사람들의 비율), **ptratio**(초등교사비율), **tax**(세금), **rad**(고속도로접근성)로 **mdev** (주택가격)을 예측하는 단순 선형회귀 모델을 만드시오 (tain:test = 7:3, random\_state는 1234)). 모델의 내용을 보이시오 |

**Source code :**

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  import pandas as pd  import numpy as np  from sklearn.linear\_model import LinearRegression  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import mean\_squared\_error, r2\_score  #prepare dataset  data =\  pd.read\_csv("C:/Users/user/PycharmProjects/deepLearning/data/BostonHousing.csv")  #print(data)  df\_X = data[['lstat','ptratio','tax','rad']]  df\_y = data['medv']  #Split the data into train/test sets  train\_X, test\_X, train\_y, test\_y = \  train\_test\_split(df\_X, df\_y, test\_size=0.3, random\_state=1234)  #Define model  model = LinearRegression()  #Train the model  model.fit(train\_X,train\_y)  print(model)  print('Cofficients: {0:.2f},{1:.2f},{2:.2f},{3:.2f} Intercept {4:.3f}'\  .format(model.coef\_[0],model.coef\_[1],model.coef\_[2],model.coef\_[3],\  model.intercept\_)) |

**실행화면 캡쳐:**



|  |
| --- |
| Q6. 모델에서 만들어진 회귀식을 쓰시오 |

medv = -0.81 x lstat -1.28 x ptratio -0.02 x tax + -0.35 x rad + 59.262

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q7. lstat, ptratio, tax, rad 의 값이 다음과 같을 때 mdev 의 예측값을 보이시오.   |  |  |  |  | | --- | --- | --- | --- | | lstat | ptratio | tax | rad | | 2.0 | 14 | 296 | 1 | | 3.0 | 15 | 222 | 2 | | 4.0 | 15 | 250 | 3 | |

**Source code :**

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  my\_test\_x = np.array([[2.0,14,296,1],[3.0,15,222,2],[4.0,15,250,3]]).reshape(3,-1)  print()  my\_pred\_y = model.predict(my\_test\_x)  print(my\_pred\_y) |

**실행화면 캡쳐:**



|  |
| --- |
| Q8. 모델에 대해 rooted mean square error (RMSE)와 R2score를 보이시오 |

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  pred = model.predict(test\_X)  print('Mean squared error: {0: .2f}'.\  format(mean\_squared\_error(test\_y,pred)))  print('Coefficient of determination: %.2f'% r2\_score(test\_y, pred)) |

**실행화면 캡쳐:**



|  |
| --- |
| Q9. lstat 하나만 가지고 모델을 만든 경우와 4개 변수를 가지고 모델을 만든 경우 어느쪽이 더 좋은 모델이라고 할수 있는가? 그 이유는? |

4개의 변수를 가지고 만든 모델이 더 좋다. 4개의 변수를 사용했을 때가 Mean squared error가 더 적게 나왔는데 이는 실제값과 예측값의 오차가 더 적다는 의미이기 때문이다. 4개의 변수가 복합적으로 예측에 영향을 끼치면서 예측을 더 잘하게 되었기 때문에 더 좋은 모델이 되었다고 생각한다.

**ucla\_admit.csv 파일은 미국 UCLA 의 대학원 입학에 대한 정보를 담고 있다. 컬럼(변수)에 대한 설명은 다음과 같다.**

**admit : 합격여부 (1:합격, 0:불합격)**

**gre : GRE 점수**

**gpa : GPA 점수**

**rank : 성적 석차**

**이 데이터셋에 대해 다음의 문제를 해결하시오**

|  |
| --- |
| Q10. **gre**, **gpa**, **rank**를 가지고 합격여부를 예측하는 logistic regression 모델을 만드시오. (tain:test = 7:3, random\_state는 1234). |

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  import pandas as pd  from sklearn.linear\_model import LogisticRegression  from sklearn.model\_selection import train\_test\_split  #prepare dataset  data = pd.read\_csv("C:/Users/user/PycharmProjects/deepLearning/data/ucla\_admit.csv")  df\_X = data[['gre','gpa','rank']]  df\_y = data['admit']  #Split the data into training/testing sets  train\_X,test\_X, train\_y, test\_y = train\_test\_split(df\_X,df\_y,test\_size=0.3,random\_state=1234)  #Define model  model = LogisticRegression()  #Train the model  model.fit(train\_X,train\_y)  print("coef\_: ",model.coef\_)  print("intercept\_: ",model.intercept\_) |

**실행화면 캡쳐:**



|  |
| --- |
| Q11. 모델을 테스트 하여 training accuracy 와 test accuracy를 보이시오 |

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  print(f'training accuracy{model.score(train\_X, train\_y)}') #training accuracy  pred\_y = model.predict(test\_X)  acc = accuracy\_score(test\_y,pred\_y) #testing accuracy  print(f'testing accuracy{acc}') |

**실행화면 캡쳐:**



|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Q12. **gre**, **gpa**, **rank 가 다음과 같을 때 합격 여부를 예측하여 보이시오**   |  |  |  | | --- | --- | --- | | gre | gpa | rank | | 400 | 3.5 | 5 | | 550 | 3.8 | 2 | | 700 | 4.0 | 2 | |

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  import numpy as np  my\_test\_x = np.array([[400,3.5,5],[550,3.8,2],[700,4.0,2]]).reshape(3,-1)  my\_pred\_y = model.predict(my\_test\_x)  print(my\_pred\_y) |

**실행화면 캡쳐:**



|  |
| --- |
| Q13.이번에는 **gre**, **gpa만 가지고 합격 여부를 예측하는 모델을 만드시오**  (tain:test = 7:3, random\_state는 1234). |

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  import pandas as pd  from sklearn.linear\_model import LogisticRegression  from sklearn.model\_selection import train\_test\_split  #prepare dataset  data = pd.read\_csv("C:/Users/user/PycharmProjects/deepLearning/data/ucla\_admit.csv")  df\_X = data[['gre','gpa']]  df\_y = data['admit']  #Split the data into training/testing sets  train\_X, test\_X, train\_y, test\_y =  train\_test\_split(df\_X,df\_y,test\_size=0.3,random\_state=1234)  #Define model  model = LogisticRegression()  #Train the model  model.fit(train\_X,train\_y)  print("coef\_: ",model.coef\_)  print("intercept\_: ",model.intercept\_) |

**실행화면 캡쳐:**



|  |
| --- |
| Q14. 모델을 테스트 하여 training accuracy 와 test accuracy를 보이시오 |

|  |
| --- |
| // source code 의 폰트는 Courier10 BT Bold으로 하시오  from sklearn.metrics import accuracy\_score #기존 코드에 추가  print("\n\n")  #Make predictios using the testing sets  print(f'training accuracy = {model.score(train\_X, train\_y)}') #training accuracy  pred\_y = model.predict(test\_X)  acc = accuracy\_score(test\_y,pred\_y) #testing accuracy  print(f'testing accuracy = {acc}') |

**실행화면 캡쳐:**



|  |
| --- |
| Q15. 3가지 변수로 모델을 만든 경우와 2가지 변수로 모델을 만든 경우를 비교하여 어떤 모델이 더 좋은 모델인지 자신의 의견을 제시하시오 (근거도 제시) |

training accuracy는 2가지 변수로 했을 때 조금 더 낮지만 testing accuracy는 2가지 변수 모델이 훨씬 높기 때문에 2가지 변수 모델이 더 좋은 모델이라고 생각한다. 변수의 개수가 꼭 많다고 해서 좋은 모델이 나오는 것이 아니라 변수가 예측을 하는데 있어서 유의미한 값을 가져야 좋은 모델이 되는 것인데 rank는 예측에 크게 도움을 주는 변수가 아니였기 때문에 2가지 변수 모델이 더 좋은 모델이고 더 좋은 결과가 나왔다고 생각한다.