AAI2250 Assignment2 Report

2022148071 컴퓨터과학과 이강원

1. Linear Regression

사용 data : 'Salary_dataset.csv' (in Kaggle)

Task : 근무 연차에 따른 연봉 선형 모델로 예측하기

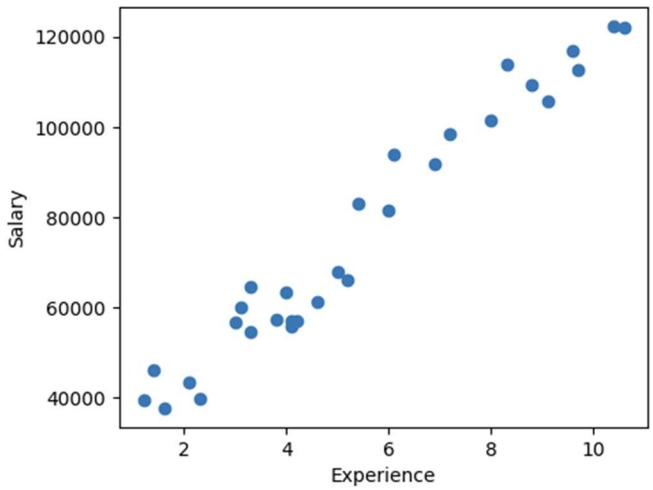
Data 전처리(Feature)

'Salary'변수에 csv data를 저장한 뒤, 'Salary.descibe()'를 통해 개괄적인 정보를 확인하였다. Column에는 Unnamed:0, YearsExperience, Salary정보가 담겨 있었다. Task가 근무 연차에 따른 Salary를 예측하는 것이므로, 필요없는 column인 Unnamed:0은 버린다. 즉, Salary = salary.drop(salary.columns[0], axis=1)을 수행해 feature을 추출하고, 데이터를 전처리하였다.

학습

학습에 앞서, 데이터를 시각화해보기 위해 matplotlib라이브러리의 메소드들을 사용해 산포도 형태로 표현하였다.

```
In [32]: plt.figure(figsize=(5,4))
         plt.scatter(salary['YearsExperience'],salary['Salary'])
         plt.xlabel('Experience')
         plt.ylabel('Salary')
         plt.show()
```



학습을 위해 sklearn라이브러리를 사용하였으며, train_test_split메소드를 사용해 데이터셋 내에서 학습 데이터와 테스트 데이터를 구분하였다.

```
In [33]: from sklearn.model_selection import train_test_split
         X_train, X_test, y_train, y_test = train_test_split(x,y,test_size=0.2)
```

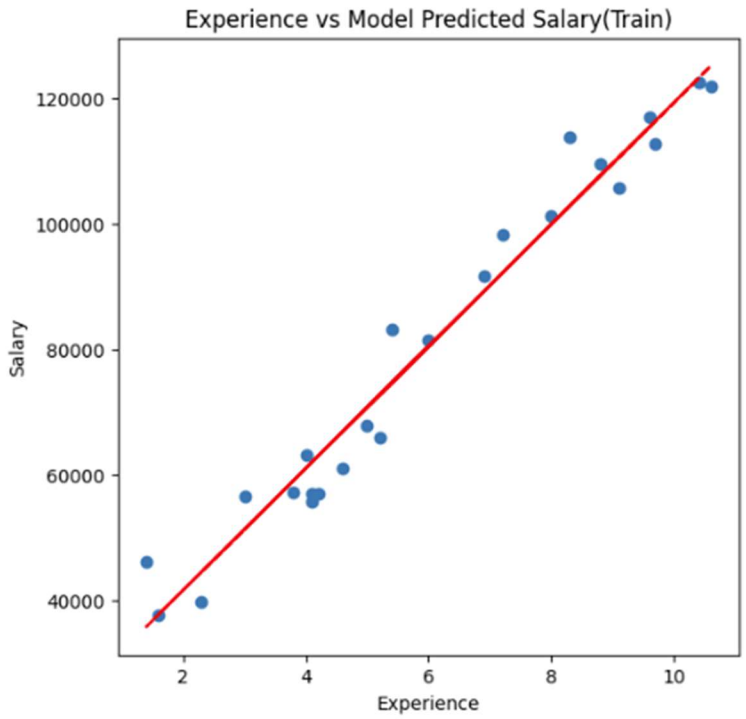
Sklearn.linear_model에서 LinearRegression을 import하여 fit메소드로 모델을 학습시킨 후, 예측데이터를 넣어 시각화하였다.

```
In [34]: from sklearn.linear_model import LinearRegression
         model = LinearRegression()
         model.fit(X_train, y_train)

In [35]: y_pred1 = model.predict(X_train)
         y_pred = model.predict(X_test)
         y_pred

Out[35]: array([125306.57600431, 101008.3195609 ,  82317.35306597,  61757.2899215
                65495.48322053,  47739.06505034])

In [27]: plt.figure(figsize=(6, 6))
         plt.scatter(X_train,y_train)
         plt.plot(X_train, y_pred1, '--r')
         plt.title('Experience vs Model Predicted Salary(Train)')
         plt.xlabel('Experience')
         plt.ylabel('Salary')
         plt.show()
```



Evalution

훈련데이터 X_train으로 예측한 값을 X_train에 대해 선형으로 그려봤을 때, 산포도(학습으로 얻은 데이터)와 예측치가 비슷한 형태를 나타냄을 알 수 있다. 따라서 좋은 모델이라 할 수 있다.

2. Logistic Regression

사용 data : sklearn dataset 내의 iris 데이터

Task : iris 품종 예측

Data 전처리(Feature)

Iris내의 'data'를 input_data로 사용하고, 'target'을 target_data로 사용하였다.

몇 가지 columns 할당 이후 dataframe을 만든다.

In [ ]:
```
data = datasets.load_iris()

input_data = data['data']
target_data = data['target']

flowers = data['target_names']
feature_names = data['feature_names']
```

In [ ]:
```
iris_df = pd.DataFrame(input_data, columns=feature_names)
iris_df['species'] = target_data

print(iris_df.head(10))
print(iris_df.describe())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
5                5.4               3.9                1.7               0.4
6                4.6               3.4                1.4               0.3
7                5.0               3.4                1.5               0.2
8                4.4               2.9                1.4               0.2
9                4.9               3.1                1.5               0.1

   species
0        0
1        0
2        0
3        0
```

학습

Train_test_split을 수행해 데이터를 학습데이터와 테스트데이터로 쪼개고, 값의 차이가 커 standardScaler()로 정규화를 수행한 후 학습을 수행함.

In [ ]:
```
train_input, test_input, train_target, test_target = train_test_
    input_data, target_data, random_state = 42)

scaler = StandardScaler()
train_scaled = scaler.fit_transform(train_input)
test_scaled = scaler.fit_transform(test_input)
```

In [ ]:
```
model = LogisticRegression(max_iter=1000)

model.fit(train_scaled, train_target)

pred = model.predict(test_scaled)
print(pred)
```

```
[1 0 2 1 1 0 1 2 2 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1
0 0 2 1
 0]
```

품종을 예측해 낸 학습 결과를 볼 수 있다

Evalution

```
print("학습 점수:{}".format(model.score(train_scaled, train_target)))
print("정확도:{}".format(model.score(test_scaled, test_target)))
```

```
학습 점수:0.9642857142857143
정확도:0.9736842105263158
```

높은 학습 점수와 정확도로 미루어 보아, 모델이 잘 구현되었다고 해석할 수 있다.

3. Clustering

사용 알고리즘 : K-means algorithm

사용 data : sklearn dataset 내의 iris 데이터

Task : iris의 품종을 구분하는 것

Data 전처리(Feature)

```
iris = load_iris()
print('target name:', iris.target_names)
iris_DF = pd.DataFrame(data=iris.data, columns=['sepal_length','sepal_width','petal_length','petal_width'])
iris_DF.head(10)
```

```
target name: ['setosa' 'versicolor' 'virginica']
```

Out[3]:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

분류할 데이터의 target_name 추출, sepal_length, sepal_width, petal_length, petal_width 로 dataframe을 만듦

학습

```
kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=300, random_state=0)
kmeans.fit(iris_DF)
```

KMeans로 수행할 학습의 군집 개수, 반복 횟수 등을 지정한 후 fit메소드로 학습을 수행함

Evalution

In [10]:
```
iris_DF['target'] = iris.target
iris_DF['cluster']=kmeans.labels_
iris_DF.head(5)
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width	target	cluster
0	5.1	3.5	1.4	0.2	0	1
1	4.9	3.0	1.4	0.2	0	1
2	4.7	3.2	1.3	0.2	0	1
3	4.6	3.1	1.5	0.2	0	1
4	5.0	3.6	1.4	0.2	0	1

In [12]:
```
iris_DF['target'] = iris.target
iris_DF['cluster'] = kmeans.labels_

iris_DF.groupby(['target', 'cluster']).count()
```

Out[12]:

		sepal_length	sepal_width	petal_length	petal_width
target	**cluster**				
0	**1**	50	50	50	50
1	**0**	48	48	48	48
	2	2	2	2	2
2	**0**	14	14	14	14
	2	36	36	36	36

Target과 cluster을 groupby했을 때 target0, target1은 대부분 군집화를 수행한 모습, target2는 나머지에는 뒤떨어지지만 성공적으로 clustering을 수행했다고 할 수 있다.