standardization

March 8, 2024

```
[2]: import pandas as pd
     import matplotlib.pyplot as plt
     import numpy as np
     import seaborn as sns
```

```
[3]: df=pd.read_csv('Social_Network_Ads.csv')
```

```
[4]: df=df.iloc[:,2:]
     df
```

```
[4]:      Age  EstimatedSalary  Purchased
     0     19            19000          0
     1     35            20000          0
     2     26            43000          0
     3     27            57000          0
     4     19            76000          0
     ..   …               …         …
     395   46            41000          1
     396   51            23000          1
     397   50            20000          1
     398   36            33000          0
     399   49            36000          1

     [400 rows x 3 columns]
```

0.1 Train Test Split

```
[5]: from sklearn.model_selection import train_test_split
     X_train,X_test,y_train,y_test=train_test_split(df.
      ↪drop('Purchased',axis=1),df['Purchased'],test_size=0.3,random_state=0)
```

```
[6]: X_train.shape,X_test.shape
```

```
[6]: ((280, 2), (120, 2))
```

0.2 Standard Scaler

[7]:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()

scaler.fit(X_train)

X_train_scaled=scaler.transform(X_train)
X_test_scaled=scaler.transform(X_test)
```

[8]:

```
X_train_scaled=pd.DataFrame(X_train_scaled,columns=X_train.columns)
X_test_scaled=pd.DataFrame(X_test_scaled,columns=X_test.columns)
```

[9]:

```
np.round(X_train_scaled.describe(),1)
```

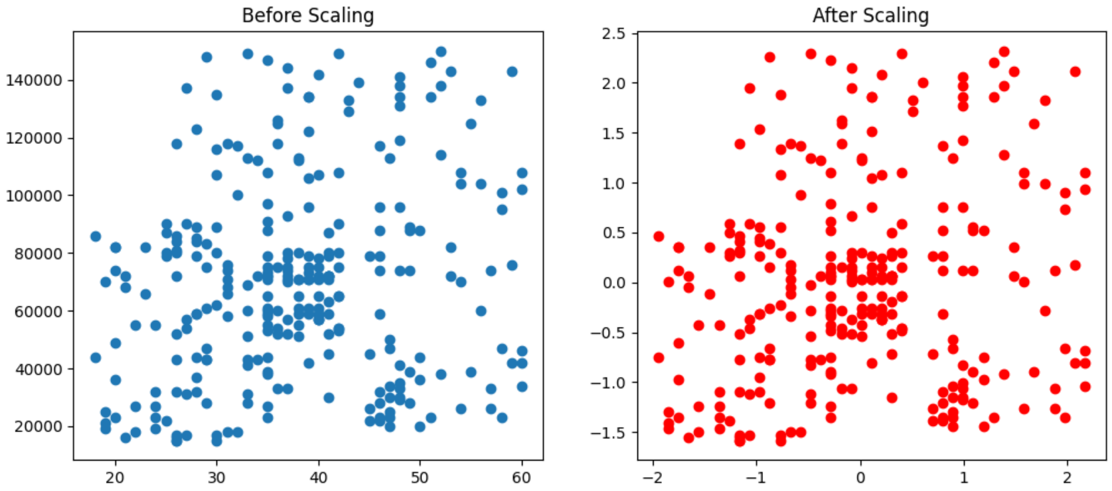
[9]:

```
         Age  EstimatedSalary
count  280.0            280.0
mean     0.0              0.0
std      1.0              1.0
min     -1.9             -1.6
25%     -0.8             -0.8
50%     -0.1              0.0
75%      0.8              0.5
max      2.2              2.3
```

0.3 Visualization of Standardisation

[10]:

```
fig,(ax1,ax2)=plt.subplots(ncols=2,figsize=(12,5))
ax1.scatter(X_train['Age'],X_train['EstimatedSalary'])
ax1.set_title('Before Scaling')
ax2.scatter(X_train_scaled['Age'],X_train_scaled['EstimatedSalary'],color='red')
ax2.set_title('After Scaling')
plt.show()
```



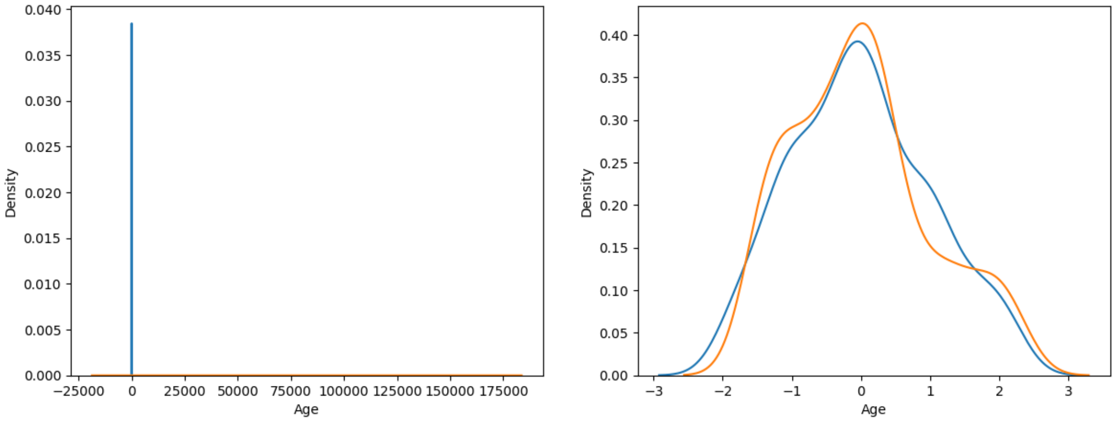
[11]:

```
fig,(ax1,ax2)=plt.subplots(ncols=2,figsize=(14,5))

#before scaling
sns.kdeplot(X_train['Age'],ax=ax1)
sns.kdeplot(X_train['EstimatedSalary'],ax=ax1)

#after scaling
sns.kdeplot(X_train_scaled['Age'],ax=ax2)
sns.kdeplot(X_train_scaled['EstimatedSalary'],ax=ax2)

plt.show()
```



0.4 Comparison between Scaled data and Unscaled data

[12]:

```
from sklearn.linear_model import LogisticRegression
```

[13]:

```
lr=LogisticRegression()
lr_scaled=LogisticRegression()
```

[14]:

```
lr.fit(X_train,y_train)
lr_scaled.fit(X_train_scaled,y_train)
```

[14]:

```
LogisticRegression()
```

[15]:

```
y_pred=lr.predict(X_test)
y_pred_scaled=lr_scaled.predict(X_test_scaled)
```

[16]:

```
from sklearn.metrics import accuracy_score
```

0.4.1 Here we can see the Scaled(standardised data) data gives more Accuracy than Previous data.

[17]:

```
print('Actual',accuracy_score(y_test,y_pred))
print('Scaled',accuracy_score(y_test,y_pred_scaled))
```

```
Actual 0.6583333333333333
Scaled 0.8666666666666667
```