Assignment 3

1) given, $\hat{y}_i = x_i \hat{\beta}$, where

$$\hat{\beta} = \left(\sum_{i=1}^{n} x_i y_i \right) / \left(\sum_{i=1}^{n} x_i^2 \right)$$

To show, $\hat{y}_i = \sum_{i=1}^{n} a_i y_i$

$$\rightarrow \quad \hat{y}_i = x_i \frac{\left(\sum_{i=1}^{n} x_i y_i \right)}{\sum_{i=1}^{n} x_i^2}$$

$$\hat{y}_i = \left(\frac{x_i}{\sum_{i=1}^{n} x_i^2} \right) \left(\sum_{i=1}^{n} x_i y_i \right)$$

let ~~weight~~ $\frac{x_i}{\sum x_i^2} = w_i$ (for each data point)

$$\hat{y}_i = w_i \left(\sum_{i=1}^{n} x_i y_i \right)$$

$$\left[\hat{y}_i = \sum_{i=1}^{n} (w_i x_i) y_i = \sum_{i=1}^{n} a_i y_i \right]$$

$\rightarrow$ so $\left[a_i = w_i x_i \right]$, where $w_i = \frac{x_i}{\sum_{i=1}^{n} x_i^2}$

2) Given, $J(\beta) = \frac{1}{2}||X\beta - Y||_2^2 + \frac{\lambda}{2}||\beta||^2$

$\rightarrow J(\beta) = \frac{1}{2}(X\beta - Y)^T(X\beta - Y) + \frac{\lambda}{2}\beta^T\beta$

$J(\beta) = \frac{1}{2}(\beta^T X^T - Y^T)(X\beta - Y) + \frac{\lambda}{2}\beta^T\beta$

$J(\beta) = \frac{1}{2}[\beta^T X^T X\beta - \beta^T X^T Y - Y^T X\beta + Y^T Y] + \frac{\lambda}{2}\beta^T\beta$

$\therefore$ β is scalar $\beta = \beta^T$

$J(\beta) = \frac{1}{2}[X^T X\beta^2 - 2\beta X^T Y + Y^T Y] + \frac{\lambda}{2}\beta^2$

$$\frac{\partial(J(\beta))}{\partial\beta} = X^T X\beta - X^T Y + \lambda\beta = 0$$

$$(X^T X + \lambda I)\beta = X^T Y$$

$$\boxed{\beta = (X^T X + \lambda I)^{-1} X^T Y}$$

$$\frac{\partial^2(J(\beta))}{\partial\beta} = X^T X + \lambda \quad > 0$$

So, $\beta = (X^T X + \lambda I)^{-1} X^T Y$ minimizes $J(\beta)$.

3) given a design matrix X

n-vector of labels $y = [y_1, \dots y_n]^T$

$\rightarrow X^T X = nI$

let x_{*i} denote the i^{th} column of X.

a) for L_1 - regularized least squares, we have

$$J(\beta) = \|X\beta - Y\|^2 + \lambda\|\beta\|$$

$$J(\beta) = (X\beta - Y)^T (X\beta - Y) + \lambda\|\beta\|$$

$$J(\beta) = (\beta^T X^T - Y^T)(X\beta - Y) + \lambda\|\beta\|$$

$\therefore \beta$ is scalar $\beta^T = \beta$

$$J(\beta) = \|\beta\|^2 X^T X - 2\|\beta\| X^T Y + \|Y\|^2 + \lambda\|\beta\|$$

$$J(\beta) = \|Y\|^2 + n\|\beta\|^2 + \|\beta\|(\lambda - 2X^T Y)$$

~~for~~ we can also write above equation in terms of summation of the calculation for each column.

$$\left[J(\beta) = \|Y\|^2 + \sum_{i=1}^{d} \|\beta_i\| \left(n\|\beta_i\| + \lambda - 2x_{*i} Y_i \right) \right]$$

$$\left[J(\beta) = \|Y\|^2 + \sum_{i=1}^{d} f(x_{*i}, \beta_i) \right]$$

where $f(x_{*i}, \beta_i) = \|\beta_i\| (n\|\beta_i\| + \lambda - 2x_{*i} Y)$

$$\boxed{f(x_{*i}, \beta_i) = n\beta_i^2 - 2Y^T x_i \beta_i + \lambda|\beta_i|}$$

b) We have

$$J(\beta) = ||y||^2 + \sum_{i=1}^{d} (n\beta_i^2 - 2x_{*i}^T y \beta_i + \lambda|\beta_i|)$$

i) case $\beta_j > 0$

$$\frac{\partial(J(\beta_j))}{\partial \beta_j} = 2n\beta_i - 2y^T x_{*i} + \lambda = 0$$

$$\left[\beta_i = \frac{1}{n}\left(y^T x_{*i} - \frac{\lambda}{2}\right)\right]$$

Since $y^T x_{*i} - \frac{\lambda}{2}$ can be < 0. So, we will take

$$\left[\beta_i = \max\left\{\frac{1}{n}\left(y^T x_{*i} - \frac{\lambda}{2}\right), 0\right\}\right]$$

ii) $\beta_i < 0$

by similar calculation

$$\beta_i = \frac{1}{n}\left(y^T x_{*i} + \frac{\lambda}{2}\right)$$

and $\beta_i = \min\left\{\frac{1}{n}\left(y^T x_{*i} + \frac{\lambda}{2}\right), 0\right\}$

iii) when $\beta_i = 0$, for other cases

So

$$\hat{\beta}_i^* = \begin{cases} \min\left\{\frac{1}{n}\left(y^T x_{*i} + \frac{\lambda}{2}\right), 0\right\} \text{ if } \beta_i < 0 \\ \max\left\{\frac{1}{n}\left(y^T x_{*i} - \frac{\lambda}{2}\right), 0\right\} \text{ if } \beta_i > 0 \\ 0 \text{ , } \cancel{\text{if } \beta_i = 0} \text{ otherwise} \end{cases}$$

c) given $J_2(\beta) = \|X\beta - Y\|^2 + \lambda\|\beta\|^2$, $\lambda > 0$

- $J_2(\beta_i) = (X\beta_i - Y)^T(X\beta_i - Y) + \lambda\beta_i^2$

$J_2(\beta_i) = (\beta_i X^T - Y^T)(X\beta_i - Y) + \lambda\beta_i^2$

$J_2(\beta_i) = \beta_i^2 X^T X - 2\beta_i X^T Y - Y^T Y + \lambda\beta_i^2$

$$\frac{\partial(J_2(\beta_i^{\#}))}{\partial\beta_i^{\#}} = 2\beta_i^{\#} X^T X - 2X^T Y + 2\lambda\beta_i^{\#} = 0$$

$$\left[(X^T X + \lambda)\beta_i^{\#} = X^T Y\right]$$

→ ∴ $X^T Y$ will never going to be zero and λ is > 0, hence there is no condition such that

$\beta_i^{\#} = 0$

∴ $\beta_i^{\#}$ will always have non-zero values

d) from our solution β^* will going to be sparse because it is possible for it to have its components 0 but it is not possible for $\beta^{\#}$.

Problem 4

Objectives

1. Calculate and interpret the correlation matrix to understand relationships among features.
2. Create a scatterplot matrix to visualize relationships among features. Explain the insights they can gain from these visualizations.
3. Perform data preprocessing and cleaning, which involves addressing missing values and handling categorical features, followed by conducting a train-test split of the data.
4. Implementing and training the linear regression model (apply Ridge and Lasso regression techniques) using appropriate Python libraries.
5. Evaluate the model's performance by calculating relevant metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared. Additionally, interpret the model's coefficients and discuss how various features impact predictions of medical expenses.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from sklearn import preprocessing

path = '/content/drive/MyDrive/sem 7/ID5055/Assignment 3/Problem
4/insurance.csv'

data = pd.read_csv(path)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1338 non-null   int64  
 1   sex       1338 non-null   object 
 2   bmi       1338 non-null   float64
 3   children  1338 non-null   int64  
 4   smoker    1338 non-null   object 
 5   region    1338 non-null   object 
 6   expenses  1338 non-null   float64
dtypes: float64(2), int64(2), object(3)
memory usage: 73.3+ KB
```

There are intotal 1338 entries for 7 features and we have 3 'object' datatype features (sex, smoker, and region).

```
correlation = data.corr()
correlation
```

```
<ipython-input-417-521f87fcc686>:1: FutureWarning: The default value
of numeric_only in DataFrame.corr is deprecated. In a future version,
it will default to False. Select only valid columns or specify the
value of numeric_only to silence this warning.
  correlation = data.corr()
```

```
               age       bmi  children  expenses
age       1.000000  0.109341  0.042469  0.299008
bmi       0.109341  1.000000  0.012645  0.198576
children  0.042469  0.012645  1.000000  0.067998
expenses  0.299008  0.198576  0.067998  1.000000
```

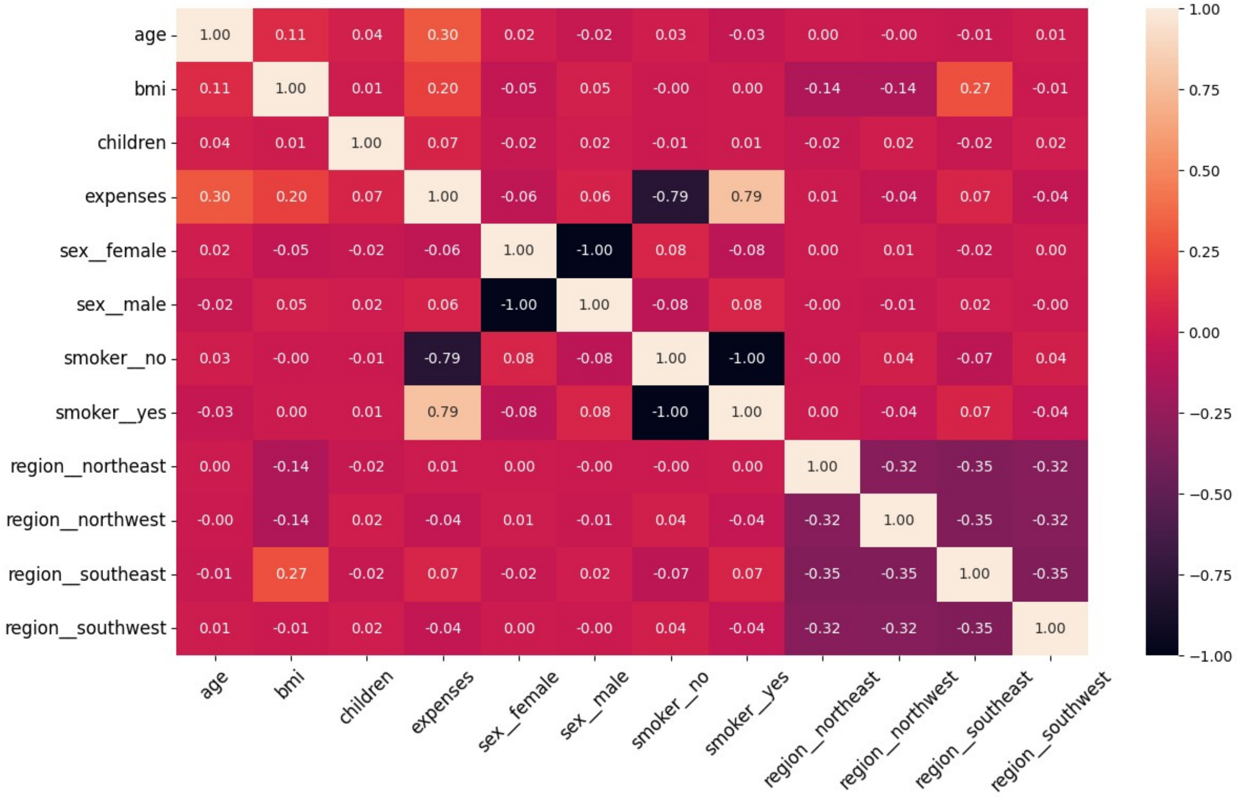
From correlation matrix between numerical datasets it is clear that-

1. Age and bmi associate strongly with expenses and otherway round i.e. expenses associate with age and bmi.
2. The expenses does not associate that strongly with number of children.
3. BMI and age are correlated weakly.
4. We still have to check the categorical features to get a better idea.

```
df2 = data.copy()

sex_dummies = pd.get_dummies(df2['sex'], prefix = 'sex_')
df2.drop(['sex'], axis = 1, inplace = True)
df2 = pd.concat([df2, sex_dummies], axis = 1)

smoker_dummies = pd.get_dummies(df2['smoker'], prefix = 'smoker_')
df2.drop(['smoker'], axis = 1, inplace = True)
df2 = pd.concat([df2, smoker_dummies], axis = 1)

region_dummies = pd.get_dummies(df2['region'], prefix = 'region_')
df2.drop(['region'], axis = 1, inplace = True)
df2 = pd.concat([df2, region_dummies], axis = 1)

df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1338 entries, 0 to 1337
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   age                1338 non-null   int64
 1   bmi                1338 non-null   float64
 2   children           1338 non-null   int64
 3   expenses           1338 non-null   float64
 4   sex__female        1338 non-null   uint8
 5   sex__male          1338 non-null   uint8
 6   smoker__no         1338 non-null   uint8
 7   smoker__yes        1338 non-null   uint8
 8   region__northeast  1338 non-null   uint8
 9   region__northwest  1338 non-null   uint8
 10  region__southeast  1338 non-null   uint8
 11  region__southwest  1338 non-null   uint8
dtypes: float64(2), int64(2), uint8(8)
memory usage: 52.4 KB
```

```
fig2, ax2 = plt.subplots(figsize=(14, 8))
corr_matrix_2 = df2.corr()

sns.heatmap(corr_matrix_2, annot=True, xticklabels=True,
yticklabels=True,
            annot_kws={"size": 10}, fmt=f'.{2}f', ax=ax2)
ax2.set_yticklabels(ax2.get_yticklabels(), rotation=0)
ax2.set_xticklabels(ax2.get_xticklabels(), rotation=45)
ax2.tick_params(axis='both', which='both', labelsize=12)

plt.show()
```



Observations and insights

1. Age vs. Expenses: Age has a positive correlation of approximately 0.30 with expenses. This suggests that as people get older, their medical expenses tend to increase. This correlation is moderately strong.

2. BMI vs. Expenses: BMI (Body Mass Index) also has a positive correlation with expenses, but it is weaker compared to age, with a correlation of approximately 0.20. This indicates that individuals with higher BMIs tend to have somewhat higher medical expenses.

3. Smoking Status vs. Expenses: Smoking status has a strong correlation with expenses. "smoker__yes" (indicating a smoker) has a positive correlation of approximately 0.79 with expenses, while "smoker__no" (indicating a non-smoker) has a negative correlation of approximately -0.79. This indicates that smokers tend to have significantly higher medical expenses compared to non-smokers.

4. Region vs. Expenses: The region where a person lives also has some correlation with expenses, although these correlations are relatively weak. None of the regional variables have a strong impact on medical expenses, but there are some variations.

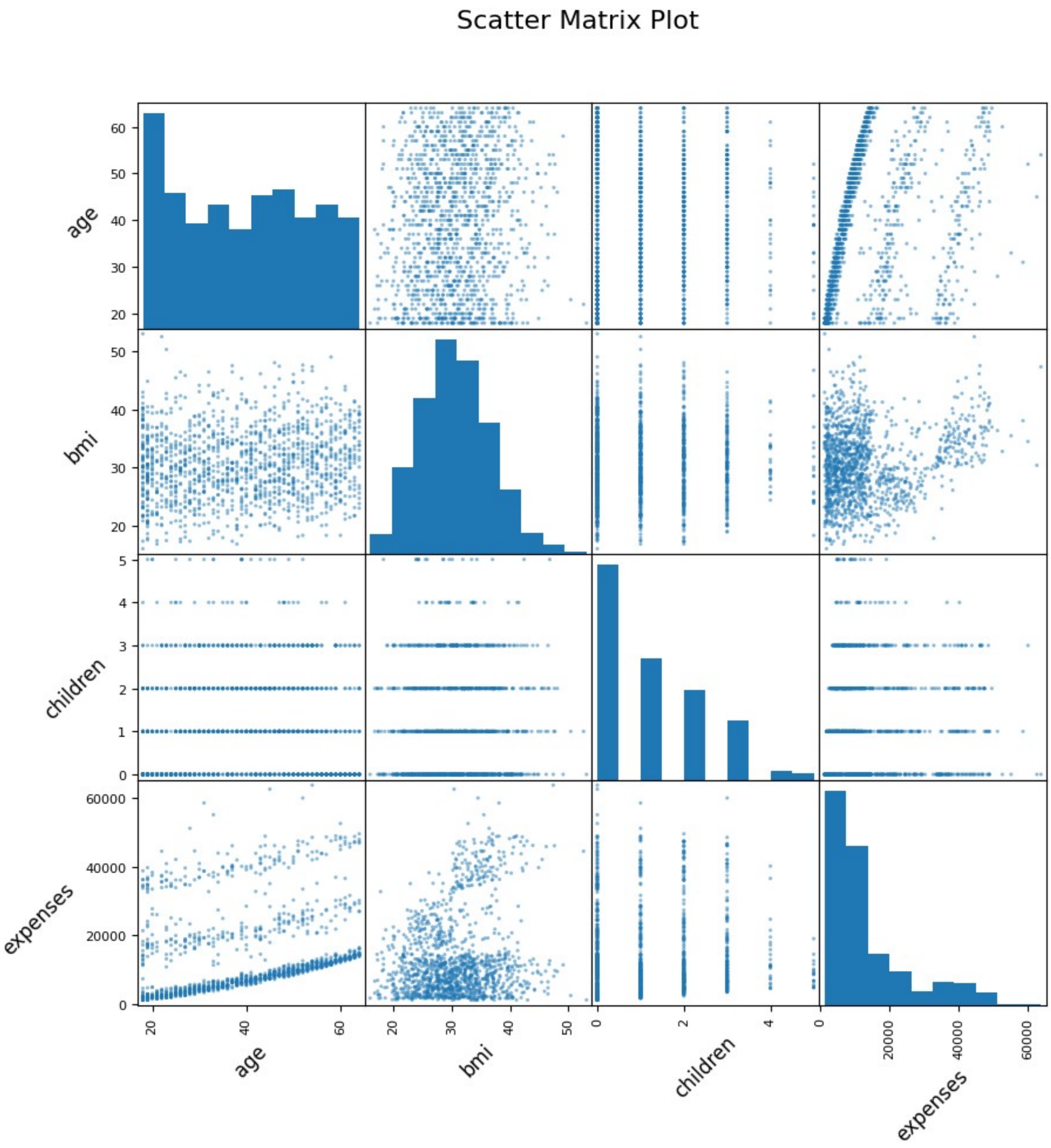
5. Gender vs. Expenses: Gender has a relatively weak correlation with expenses. "sex__female" has a negative correlation of approximately -0.06, while "sex__male" has a positive correlation of approximately 0.06. This suggests that, on average, females may have slightly lower medical expenses than males in the dataset, although the effect is not very significant.

6. Number of Children vs. Expenses: The number of children a person has ("children" variable) has a relatively weak positive correlation of approximately 0.07 with expenses. This implies that individuals with more children may have slightly higher medical expenses, but the effect is not very strong.

```
df_plot = data.copy()
df_plot
```

```
      age     sex   bmi  children smoker     region  expenses
0      19  female  27.9         0    yes  southwest  16884.92
1      18    male  33.8         1     no  southeast   1725.55
2      28    male  33.0         3     no  southeast   4449.46
3      33    male  22.7         0     no  northwest  21984.47
4      32    male  28.9         0     no  northwest   3866.86
...   ...     ...   ...       ...    ...        ...       ...
1333   50    male  31.0         3     no  northwest  10600.55
1334   18  female  31.9         0     no  northeast   2205.98
1335   18  female  36.9         0     no  southeast   1629.83
```

```
1336   21  female  25.8         0     no  southwest   2007.95
1337   61  female  29.1         0    yes  northwest  29141.36

[1338 rows x 7 columns]

scatter_matrix = pd.plotting.scatter_matrix(
    df_plot, figsize=(10, 10), alpha=0.5, marker='o', grid=True, s = 5
)

for ax in scatter_matrix.ravel():
    ax.set_xlabel(ax.get_xlabel(), fontsize=12)
    ax.set_ylabel(ax.get_ylabel(), fontsize=12)
    ax.xaxis.label.set_rotation(45)
    ax.yaxis.label.set_rotation(45)
    ax.yaxis.label.set_ha('right')

plt.suptitle("Scatter Matrix Plot", y=0.96, fontsize=16)

plt.show()
```



Insights

1. Expenses vs age - It is clear from the scatter plot that with age the medical expenses increases.
2. Expenses vs bmi - from the plot it can seen that there is a concentration of values and however there is not a clear trend but we can observe increase in expenses with increase in bmi.
3. Expenses vs children - There is no trend present between two features. Number of children has no major role in determining the expenses.

Observations

1. There are no missing datapoints in the given data.
2. We have already converted our categorical datasets into numerical datasets with the help of get_dummies.
3. Now moving towards train test split.

```
df_reg = data

for col in list(df_reg.columns):
  if str(df_reg[col].dtypes) == 'object':
    print(df_reg[col].unique())

['female' 'male']
['yes' 'no']
['southwest' 'southeast' 'northwest' 'northeast']

def cat_to_num(col_data, col_name, class_lis ):
  col_data[col_name] = col_data[col_name].apply(lambda x:
class_lis.index(x) + 1)

for cols in list(df_reg.columns):
  if str(df_reg[cols].dtypes) == 'object':
    cat_to_num(df_reg, cols, list(df_reg[cols].unique()))
```

For smoker: 1 = yes, 2 = no For sex: 1 = female, 2 = male For region: 1 = southwest, 2 = southeast, 3 = northwest, 4 = northeast.

```
X = df2.drop(['expenses'], axis = 1)
y = df2['expenses']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)
```

```
# Train the Linear Regression model
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)

# Train the Ridge Regression model
ridge_reg = Ridge(alpha=0.5)
ridge_reg.fit(X_train, y_train)
```

```python
# Train the Lasso Regression model
lasso_reg = Lasso(alpha=0.5)
lasso_reg.fit(X_train, y_train)
```

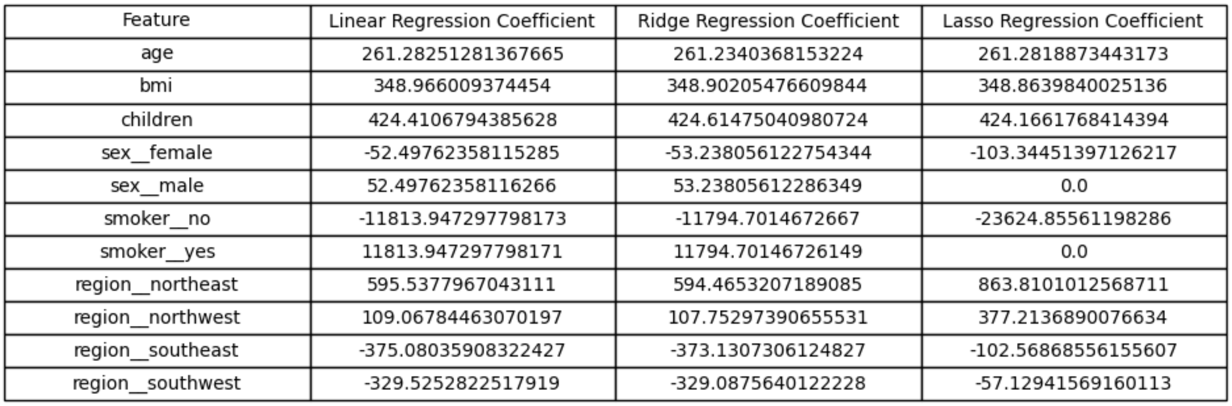
```
Lasso(alpha=0.5)
```

```python
# Make predictions on the test set
linear_pred = linear_reg.predict(X_test)
ridge_pred = ridge_reg.predict(X_test)
lasso_pred = lasso_reg.predict(X_test)

# Calculate evaluation metrics
linear_mae = mean_absolute_error(y_test, linear_pred)
ridge_mae = mean_absolute_error(y_test, ridge_pred)
lasso_mae = mean_absolute_error(y_test, lasso_pred)

linear_mse = mean_squared_error(y_test, linear_pred)
ridge_mse = mean_squared_error(y_test, ridge_pred)
lasso_mse = mean_squared_error(y_test, lasso_pred)

linear_r2 = r2_score(y_test, linear_pred)
ridge_r2 = r2_score(y_test, ridge_pred)
lasso_r2 = r2_score(y_test, lasso_pred)

# Print the evaluation metrics
print("Linear Regression Metrics:")
print(f"MAE: {linear_mae}")
print(f"MSE: {linear_mse}")
print(f"R-squared: {linear_r2}")
print("\nRidge Regression Metrics:")
print(f"MAE: {ridge_mae}")
print(f"MSE: {ridge_mse}")
print(f"R-squared: {ridge_r2}")
print("\nLasso Regression Metrics:")
print(f"MAE: {lasso_mae}")
print(f"MSE: {lasso_mse}")
print(f"R-squared: {lasso_r2}")
```

```
Linear Regression Metrics:
MAE: 4144.88640999345
MSE: 33777093.10084606
R-squared: 0.7696351080608884

Ridge Regression Metrics:
MAE: 4148.229580129345
MSE: 33786028.61035601
```

```
R-squared: 0.7695741665323639

Lasso Regression Metrics:
MAE: 4145.170098628805
MSE: 33777925.44532053
R-squared: 0.7696294313454782
```

```
coefficients_df = pd.DataFrame({
    'Feature': X.columns,
    'Linear Regression Coefficient': linear_reg.coef_,
    'Ridge Regression Coefficient': ridge_reg.coef_,
    'Lasso Regression Coefficient': lasso_reg.coef_
})

fig, ax = plt.subplots(figsize=(12, 4))
ax.axis('tight')
ax.axis('off')

table = ax.table(cellText=coefficients_df.values,
colLabels=coefficients_df.columns, loc='center', cellLoc='center')
table.auto_set_font_size(False)
table.set_fontsize(10)
table.scale(1, 1.5)
plt.show()
```



Feature	Linear Regression Coefficient	Ridge Regression Coefficient	Lasso Regression Coefficient
age	261.28251281367665	261.2340368153224	261.2818873443173
bmi	348.966009374454	348.90205476609844	348.8639840025136
children	424.4106794385628	424.61475040980724	424.1661768414394
sex__female	-52.49762358115285	-53.238056122754344	-103.34451397126217
sex__male	52.49762358116266	53.23805612286349	0.0
smoker__no	-11813.947297798173	-11794.7014672667	-23624.85561198286
smoker__yes	11813.947297798171	11794.70146726149	0.0
region__northeast	595.5377967043111	594.4653207189085	863.8101012568711
region__northwest	109.06784463070197	107.75297390655531	377.2136890076634
region__southeast	-375.08035908322427	-373.1307306124827	-102.56868556155607
region__southwest	-329.5252822517919	-329.0875640122228	-57.12941569160113

1. For $\alpha = 0.5$ we are getting the lowest MAE ans MSE score and highest R^2 score for both ridge and lasso regression.
2. Intrestingly the lasso regression is making smoker_yes and sex_male 0, i.e., they are irrelevant features according to it but it is not true, both smoker_yes and sex_male show good correlation with expenses.

Observations

1. From the table it is clear that except sex_female, smoker_no, region_southeast, and region_southwest all have positive coefficents, which implies that these features will proportionately increase the expenses.

2. Based on sex, sex_male feature has positive coeffiecent whereas sex_female has negative coefficents implying that females have less medical expenses as compared to males.

3. Similarly, for people who are smokers have more medical expenses as compared to non-smokers, which can found in the nature of their coefficents, also the value of coefficent is large implying that it is a major feature.

4. Finally, region does not associate well with medical expenses as per the correlation but here we can see a person from northeast and northwest have more medical expenses than a person who is from either southeast or southwest.

57. for multiple linear regression,

$$\left[\hat{Y}_i = \beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} \cdots \beta_p X_{p,i} + \varepsilon_i\right]$$

now, here ε_i has gaussian distribution with 0 mean and σ^2, unknown parameter, so joint density for ε_i is

$$L(\beta, \sigma^2 | Y, X) = \frac{1}{(\sqrt{2\pi}\,\sigma)^n} \exp\left(-\sum \frac{\varepsilon_i^2}{2\sigma^2}\right)$$

Also, $\left[\varepsilon_i = Y_i - (\beta_0 + \beta_1 X_{1,i} + \beta_2 X_{2,i} \cdots \beta_p X_{p,i})\right]$

$$\ln(L(\beta, \sigma^2 | Y, X)) = -\frac{n}{2}\ln 2\pi - n\ln\sigma - \frac{1}{2\sigma^2}\sum \varepsilon_i^2$$

So, maximizing the $\ln(L(\beta, \sigma^2 | Y, X))$ is same as minimizing

$$SS(\beta) = \sum \varepsilon_i^2$$

$$\left[SS(\beta) = \sum\left(\beta_0 + \sum_{j=1}^{p} \beta_j X_{j,i} - Y_i\right)^2\right]$$

hence, MLE for the matrix of the coefficients is same as that obtained via solving the normal equation.

Problem 6

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from skimage import io
from sklearn.metrics import silhouette_score

image = io.imread('/content/drive/MyDrive/sem 7/ID5055/Assignment 
3/Problem 6/frog.jpg')
pixels = np.array(image)
print(pixels.shape)
```

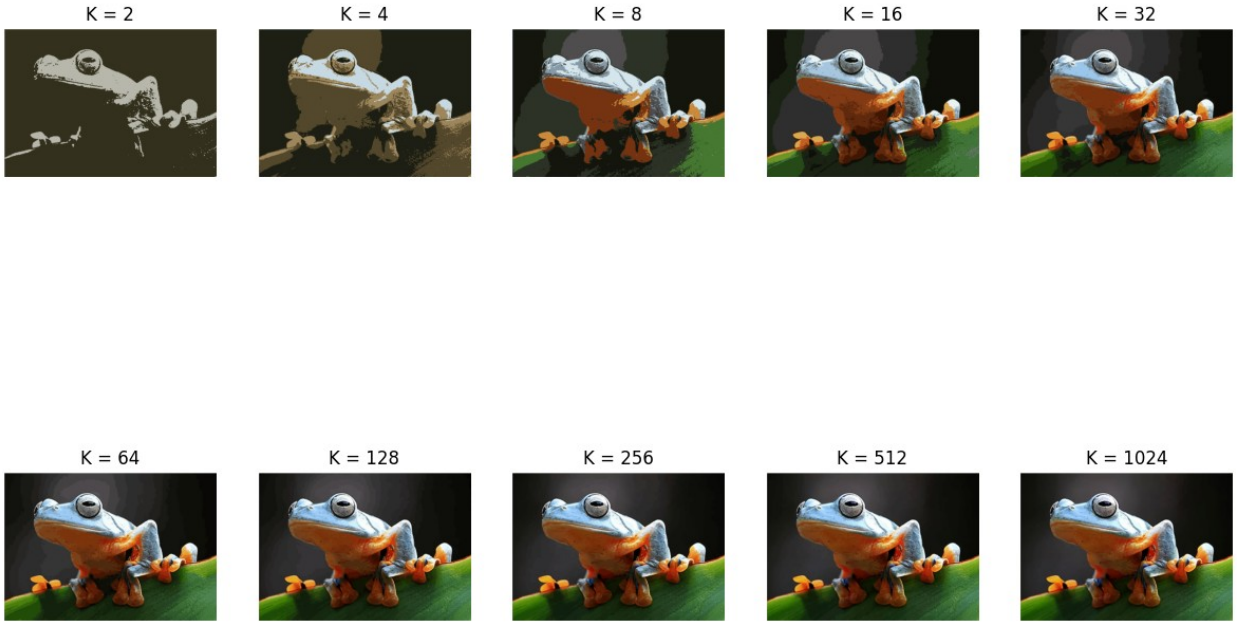
```
(392, 562, 3)
```

Converting the pixels super matrix into 2D matrix that represent all pixels.

```
pixels = pixels.reshape(-1, 3)
```

Visualizing the image after compression

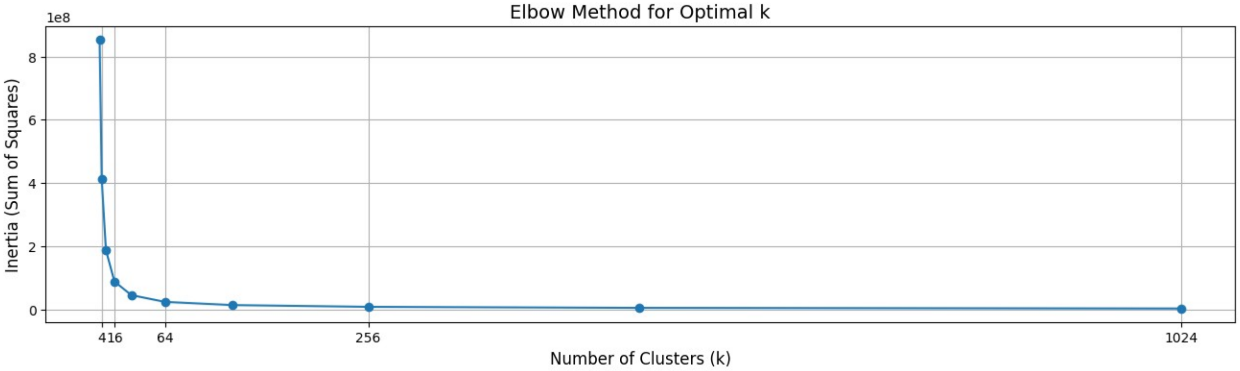
```
k_values = [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
ss_dist_elbow_check = []
# ss_dist_silhouette_score = []

plt.figure(figsize=(15, 10))

for i, k in enumerate(k_values):

    kmeans = KMeans(n_clusters = k, random_state = 0, init = 'k-means+
+', n_init = 1, max_iter = 30)
    clustered_pixels = kmeans.fit_predict(pixels)
    ss_dist_elbow_check.append(kmeans.inertia_)

    # score = silhouette_score(pixels, clustered_pixels, 
metric='euclidean')
    # ss_dist_silhouette_score.append(score)

    compressed_pixels = 
kmeans.cluster_centers_[clustered_pixels].astype(int)

    compressed_image = compressed_pixels.reshape(image.shape)

    plt.subplot(2, 5, i + 1)
    plt.title(f'K = {k}')
    plt.imshow(compressed_image)
    plt.axis('off')
plt.figure(figsize=(10, 8))
plt.show()
```



```
<Figure size 1000x800 with 0 Axes>
```

Getting the optimal k value based on elbow test and silhouette score.

```
# Elbow curve
plt.figure(figsize = (16, 4))
plt.plot(k_values, ss_dist_elbow_check, marker='o')
plt.xlabel('Number of Clusters (k)', fontsize = 12)
plt.ylabel('Inertia (Sum of Squares)', fontsize = 12)
plt.title('Elbow Method for Optimal k', fontsize = 14)
plt.xticks(k_values[1::2])
plt.grid(True)
```



1. From the above curve it is clear that k = 64 is the optimal value as per the elbow method for k value selection.
2. From the images shown above it is also clear that for k = 64 the image has almost all the features of the original image.

NOTE : I tried getting the Silhouette score but it is taking too much computational power, which my laptop can't process.

Is the compression obtained lossy or lossless? What is the effect of varying the value of K in terms of overfitting or underfitting the data?

1. The k mean compression is a lossy compression because we are approximating each pixel colour using nearest centroid. We are loosing information in the process and can't revet back, hence it is also known as irreversible compression.
2. If we increase the k values then it will result in overfitting of the data, and if we use low value of k it will result in underfitting. This is evident from the elbow curve, where as we increase the k values the sum of square of error decreases.

Problem 7

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import AgglomerativeClustering

# Load Online Retail data (assuming you have the dataset as a CSV
file)
data = pd.read_csv('/content/drive/MyDrive/sem 7/ID5055/Assignment
3/Problem 7/OnlineRetail.csv', encoding='ISO-8859-1')

# Drop rows with missing values (you may need more extensive
preprocessing)
data = data.dropna()

data.head()
```

```
  InvoiceNo StockCode                          Description
Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6

1    536365     71053                  WHITE METAL LANTERN         6

2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8

3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6

4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6


        InvoiceDate  UnitPrice  CustomerID         Country
0  01-12-2010 08:26       2.55     17850.0  United Kingdom
1  01-12-2010 08:26       3.39     17850.0  United Kingdom
2  01-12-2010 08:26       2.75     17850.0  United Kingdom
3  01-12-2010 08:26       3.39     17850.0  United Kingdom
4  01-12-2010 08:26       3.39     17850.0  United Kingdom
```

```
data.shape
```

```
(406829, 8)
```

```
data.info()
```

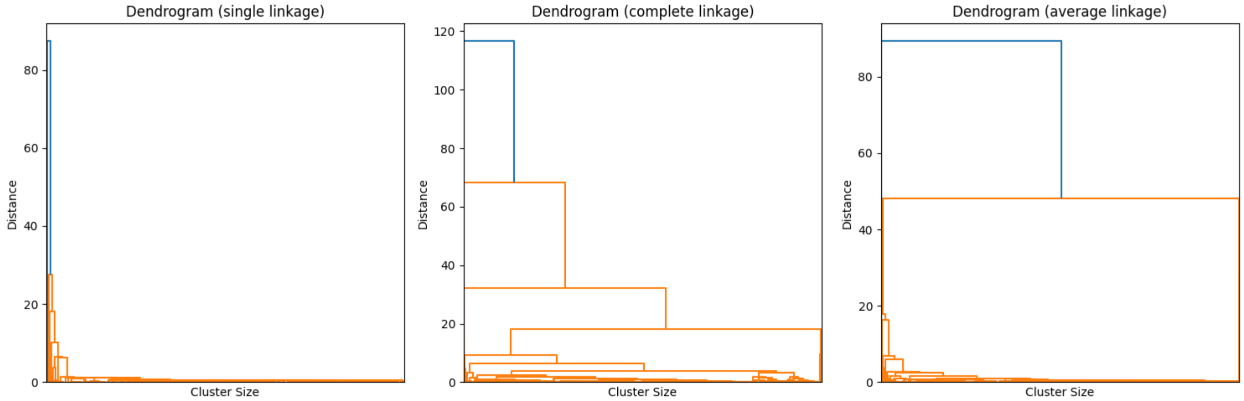
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 406829 entries, 0 to 541908
```

```
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype  
---  ------       --------------   -----  
 0   InvoiceNo    406829 non-null  object 
 1   StockCode    406829 non-null  object 
 2   Description  406829 non-null  object 
 3   Quantity     406829 non-null  int64  
 4   InvoiceDate  406829 non-null  object 
 5   UnitPrice    406829 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      406829 non-null  object 
dtypes: float64(2), int64(1), object(5)
memory usage: 27.9+ MB
```

```
for items in list(data.columns):
  if data[items].dtype == 'object':
    print(data[items].unique())
```

```
['536365' '536366' '536367' ... '581585' '581586' '581587']
['85123A' '71053' '84406B' ... '90214Z' '90089' '23843']
['WHITE HANGING HEART T-LIGHT HOLDER' 'WHITE METAL LANTERN'
 'CREAM CUPID HEARTS COAT HANGER' ... 'PINK CRYSTAL SKULL PHONE CHARM'
 'CREAM HANGING HEART T-LIGHT HOLDER' 'PAPER CRAFT , LITTLE BIRDIE']
['01-12-2010 08:26' '01-12-2010 08:28' '01-12-2010 08:34' ...
 '09-12-2011 12:31' '09-12-2011 12:49' '09-12-2011 12:50']
['United Kingdom' 'France' 'Australia' 'Netherlands' 'Germany'
'Norway'
 'EIRE' 'Switzerland' 'Spain' 'Poland' 'Portugal' 'Italy' 'Belgium'
 'Lithuania' 'Japan' 'Iceland' 'Channel Islands' 'Denmark' 'Cyprus'
 'Sweden' 'Austria' 'Israel' 'Finland' 'Greece' 'Singapore' 'Lebanon'
 'United Arab Emirates' 'Saudi Arabia' 'Czech Republic' 'Canada'
 'Unspecified' 'Brazil' 'USA' 'European Community' 'Bahrain' 'Malta'
'RSA']
```

```
df = data.iloc[:10000, :]

X = df[['Quantity', 'UnitPrice']]

# Standardize the feature matrix (important for hierarchical
clustering)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Define the number of clusters (k=3)
n_clusters = 3

# Apply Hierarchical Clustering with different linkage methods
linkage_methods = ['single', 'complete', 'average']
labels = {}

for method in linkage_methods:
    Z = linkage(X_scaled, method=method, metric='euclidean')
    labels[method] =
AgglomerativeClustering(n_clusters=n_clusters).fit_predict(Z)

# Plot the dendrograms
plt.figure(figsize=(15, 5))
for i, method in enumerate(linkage_methods):
    plt.subplot(1, 3, i + 1)
    plt.title(f'Dendrogram ({method} linkage)')
    dendrogram(linkage(X_scaled, method=method, metric='euclidean'),
no_labels=True, truncate_mode='level')
    plt.xlabel('Cluster Size')
    plt.ylabel('Distance')
plt.tight_layout()
plt.show()
```



1. Single Linkage: This method computes the distance between the closest points in the clusters. It tends to create long, chain-like clusters and is sensitive to outliers. Use cases include cases where clusters are expected to be non-convex and where there may be noise in the data.

2. Complete Linkage: This method computes the distance between the farthest points in the clusters. It tends to create compact, spherical clusters. It's suitable for situations where we expect well-separated, compact clusters in our data.

3. Average Linkage: This method calculates the average distance between all pairs of data points in the clusters. It is a compromise between single and complete linkage and is less sensitive to outliers. Average linkage is often a good choice when we have data with varying cluster shapes and sizes.

By visualizing the dendrograms and the resulting clusters, we can assess which linkage method is most suitable for our specific Online Retail dataset, depending on the characteristics of our data and our clustering objectives.

Problem 8

Loading the data

```
import pandas as pd
from sklearn.cluster import SpectralClustering
from sklearn.metrics import adjusted_rand_score,
adjusted_mutual_info_score, silhouette_score
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler

data = pd.read_csv('/content/drive/MyDrive/sem 7/ID5055/Assignment
3/Problem 8/gene_expression.csv')
data.head()
```

```
   Gene One  Gene Two  Cancer Present
0       4.3       3.9               1
1       2.5       6.3               0
2       5.7       3.9               1
3       6.1       6.2               0
4       7.4       3.4               1
```

```
df = data
```

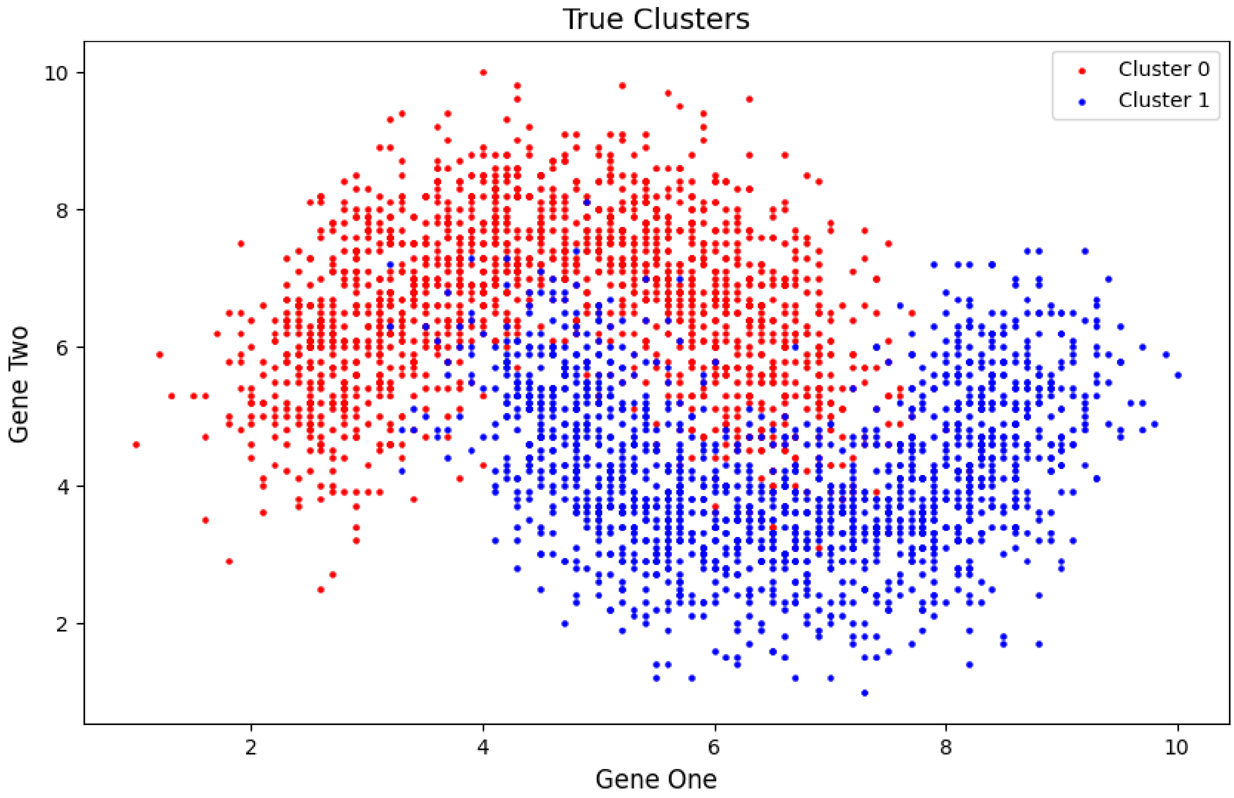
Implement spectral clustering using Python and scikit-learn to identify clusters of co-expressed genes within the dataset.

```
n_clusters = 2
X = df.iloc[:, :2]
X_scaled = StandardScaler().fit_transform(X)
spectral = SpectralClustering(n_clusters=n_clusters,
affinity='nearest_neighbors', random_state=0)
df['Cluster'] = spectral.fit_predict(X_scaled)

df.head()
```

```
   Gene One  Gene Two  Cancer Present  Cluster
0       4.3       3.9               1        0
1       2.5       6.3               0        1
2       5.7       3.9               1        0
3       6.1       6.2               0        1
4       7.4       3.4               1        0
```

Create visualizations for the true clusters based on the information in the 3rd column of the dataset.

```
plt.figure(figsize=(10, 6))
colors = ['red', 'blue']

for i in range(n_clusters):
    cluster_data = df[df['Cancer Present'] == i]
    plt.scatter(cluster_data['Gene One'], cluster_data['Gene Two'],
color=colors[i], label=f'Cluster {i}', s = 5)

plt.title('True Clusters', fontsize = 14)
plt.xlabel('Gene One', fontsize = 12)
plt.ylabel('Gene Two', fontsize = 12)
plt.legend()
plt.show()
```



Evaluate and provide insights on the outcomes, including a comprehensive report on performance metrics such as Adjusted Rand Index, Adjusted Mutual Information, and Silhouette Score.

```
ari = adjusted_rand_score(df['Cancer Present'], df['Cluster'])
ami = adjusted_mutual_info_score(df['Cancer Present'], df['Cluster'])
silhouette = silhouette_score(X_scaled, df['Cluster'])
```

```
print(f'Adjusted Rand Index: {ari}')
print(f'Adjusted Mutual Information: {ami}')
print(f'Silhouette Score: {silhouette}')
```

```
Adjusted Rand Index: 0.5153624060863754
Adjusted Mutual Information: 0.413040052343147
Silhouette Score: 0.4540574993371185
```

1. Adjusted Rand Index (ARI): 0.515 -

- The ARI measures the similarity between the true clusters (Cancer Present) and the clusters generated by the spectral clustering algorithm.

- An ARI score of 0.515 indicates a moderate degree of similarity between the true clusters and the clusters identified by the algorithm.
- This suggests that while spectral clustering is capturing some underlying structure in the data, there may still be room for improvement.

1. Adjusted Mutual Information (AMI): 0.413

- The AMI is another measure of the agreement between the true clusters and the clusters produced by the algorithm.
- An AMI score of 0.413 suggests a moderate level of mutual information between the true clusters and the algorithm's clusters.
- Similar to the ARI, this indicates that the spectral clustering algorithm is providing some meaningful clustering, but it may not be capturing all of the underlying patterns in the data.

1. Silhouette Score: 0.454

- The Silhouette Score measures the quality of the clusters themselves. It assesses how well-separated the clusters are and how similar the data points within each cluster are to each other.
- A Silhouette Score of 0.454 is relatively high, indicating that the clusters are reasonably well-separated and that data points within each cluster are similar to each other.
- This suggests that the algorithm is successful in creating meaningful clusters, and the clusters are relatively distinct from each other.

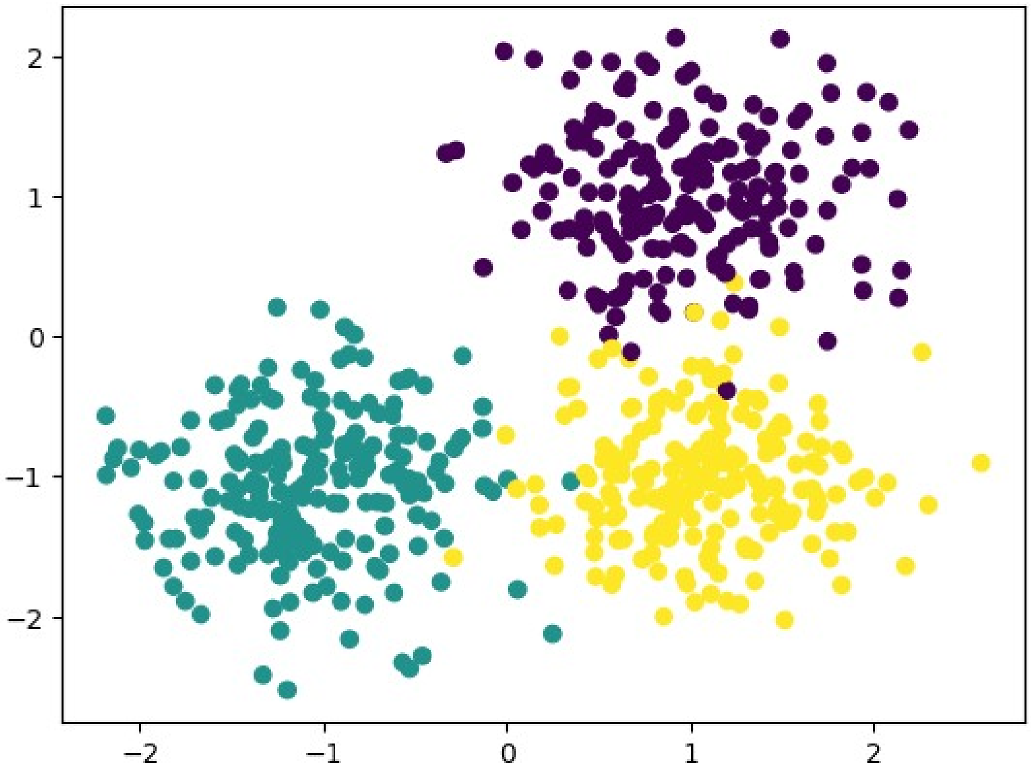
Problem 9

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

import seaborn as sns
import scipy.cluster.hierarchy as shc
from sklearn import metrics
from sklearn.datasets import make_blobs, make_circles, make_moons
from sklearn.cluster import DBSCAN
from sklearn.preprocessing import StandardScaler

seed = 0
```

Generate a dummy toy dataset with varying densities and shapes. Set the eps (Epsilon) andmin samples (MinPts) parameters, and then fit DBSCAN to the generated dataset.

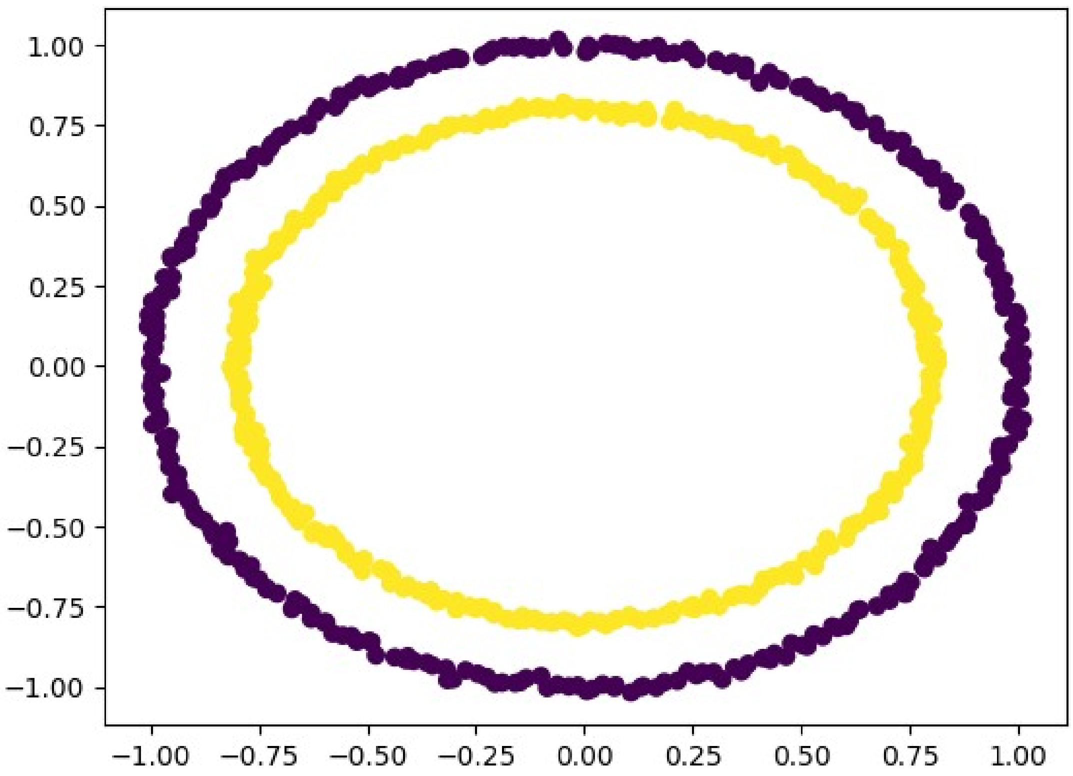
```
# blob data

centers = [[1, 1], [-1, -1], [1, -1]]
blob_X, blob_true_labels = make_blobs(n_samples = 600, centers =
centers, cluster_std= 0.5, random_state =seed)
scaled_blob_X = StandardScaler().fit_transform(blob_X)
plt.scatter(blob_X[:, 0], blob_X[:, 1], c = blob_true_labels)
```

```
<matplotlib.collections.PathCollection at 0x7936dc1ae2f0>
```



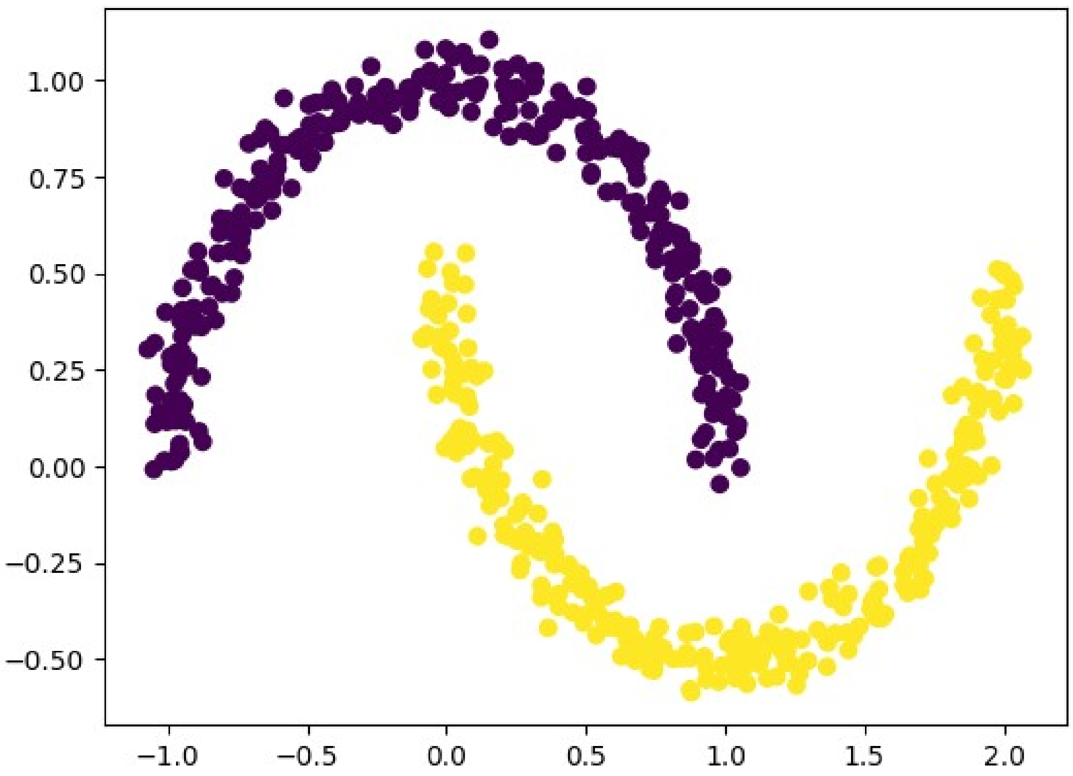
```
# Concentric circles data

circle_X, circle_true_labels = make_circles(n_samples = 600, noise =
.01, random_state = seed)
scaled_circle_X = StandardScaler().fit_transform(circle_X)
plt.scatter(circle_X[:, 0], circle_X[:, 1], c = circle_true_labels)
```

```
<matplotlib.collections.PathCollection at 0x7936dbbf5600>
```



```
# Moon shaped data

moon_X, moon_true_labels = make_moons(n_samples = 600, noise = 0.05,
random_state = seed)
scaled_moon_X = StandardScaler().fit_transform(moon_X)
plt.scatter(moon_X[:, 0], moon_X[:, 1], c = moon_true_labels)
```

```
<matplotlib.collections.PathCollection at 0x7936dbc755a0>
```



```
# Helper function to plot

def plot_clusters(data, true_labels = None, cluster_labels = None,
title_true = 'True Cluster', title_cluster = 'DBSCAN clustering'):
  fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (14, 6))
  ax1.scatter(data[:, 0], data[:, 1], c = true_labels)
  ax1.set_title(title_true)

  if cluster_labels is not None:
    ax2.scatter(data[:, 0], data[:, 1], c = cluster_labels)
    ax2.set_title(title_cluster)

  plt.show()
```

Objectives

1. Experiment with each combination of eps and min samples (consider at least 3 values of each) for these parameters. Report the values of the performance metrics to evaluate DBSCAN's sensitivity to parameter choices.

2. Visualize the clustering results using a scatter plot, where each cluster is assigned a different color. Additionally, use a different marker shape for noise points.

3. Calculate and report the following performance metrics: Silhouette Score, Adjusted Rand Index, Adjusted Mutual Information.

For Blob data

```
eps_values_blob = [0.2, 0.3, 0.5, 0.8, 1.3]
min_samples_values_blob = [2, 3, 5, 8, 13]

for i, eps in enumerate(eps_values_blob):
  for j, min_samples in enumerate(min_samples_values_blob):
    # Fit DBSCAN with current parameter combination
    dbscan_blob = DBSCAN(eps=eps, min_samples=min_samples)
    dbscan_blob.fit_predict(scaled_blob_X)

    # Calculate performance metrics
    X, pred_labels, true_labels = scaled_blob_X, dbscan_blob.labels_,
blob_true_labels
    if len(set(pred_labels)) > 1:
      silhouette = metrics.silhouette_score(X, pred_labels)
      ari = metrics.adjusted_rand_score(true_labels, pred_labels)
      ami = metrics.adjusted_mutual_info_score(true_labels,
pred_labels)
      print(f'For eps = {eps} and minimum samples = {min_samples}')
      print("Silhouette score: %0.3f " % silhouette)
      print('Adjusted Rand Index: %0.3f'% ari)
      print('Adjusted mutual information: %0.3f' % ami)
      plot_clusters(scaled_blob_X, true_labels, pred_labels)
      print('*********************************************')
      print('\n')

Output hidden; open in https://colab.research.google.com to view.
```

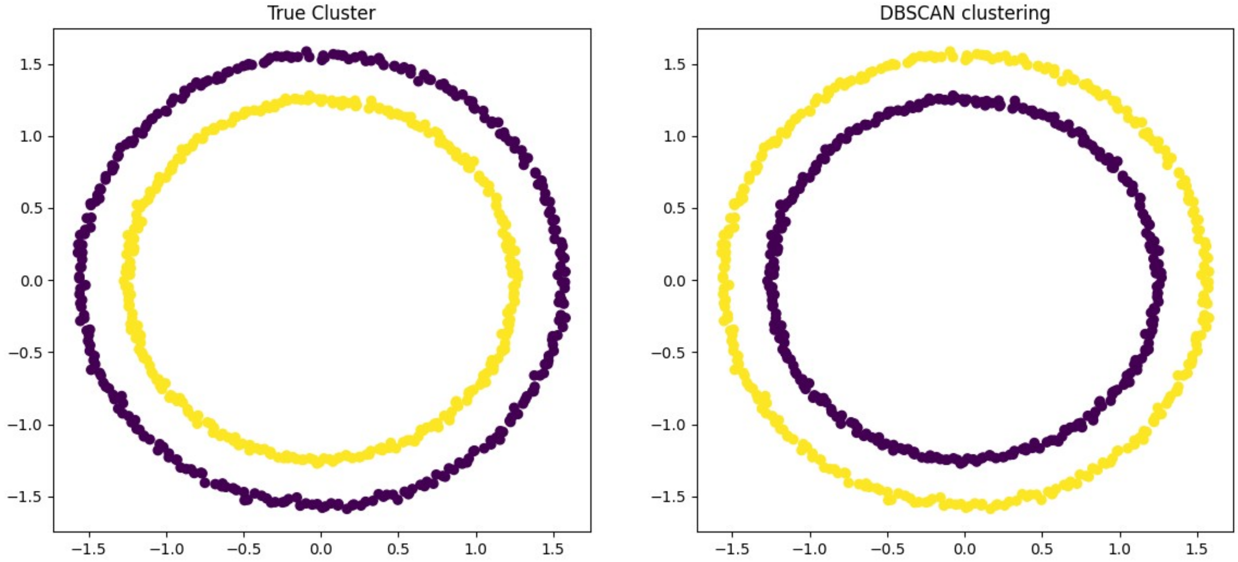
Based on the scoring for different eps and min_samples we found that the eps = 0.3 and min_samples = 13 is the best combination for blob dataset.

For Circle data

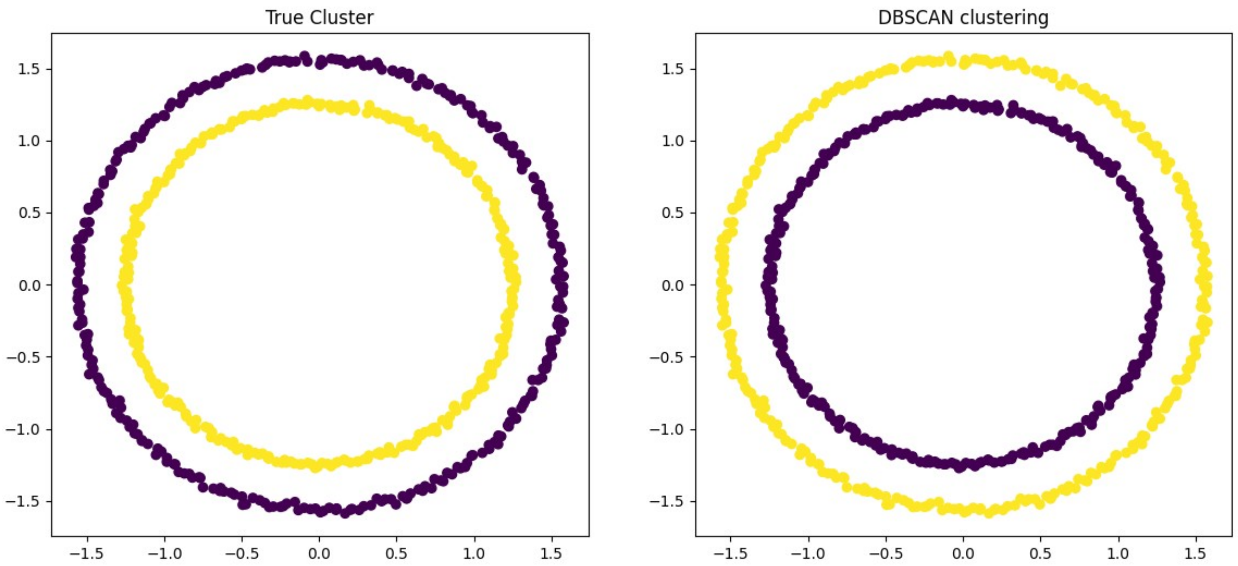
```
eps_values_circle = [0.2, 0.3, 0.5, 0.7]
min_samples_values_circle = [5, 10, 15, 20, 25]

for i, eps in enumerate(eps_values_circle):
  for j, min_samples in enumerate(min_samples_values_circle):
    # Fit DBSCAN with current parameter combination
    dbscan_circle = DBSCAN(eps=eps, min_samples=min_samples)
    dbscan_circle.fit_predict(scaled_circle_X)

    # Calculate performance metrics
    X, pred_labels, true_labels = scaled_circle_X,
dbscan_circle.labels_, circle_true_labels
    if len(set(pred_labels)) > 1:
      silhouette = metrics.silhouette_score(X, pred_labels)
      ari = metrics.adjusted_rand_score(true_labels, pred_labels)
      ami = metrics.adjusted_mutual_info_score(true_labels,
pred_labels)
```

```
        print(f'For eps = {eps} and minimum samples = {min_samples}')
        print("Silhouette score: %0.3f " % silhouette)
        print('Adjusted Rand Index: %0.3f'% ari)
        print('Adjusted mutual information: %0.3f' % ami)
        plot_clusters(X, true_labels, pred_labels)
        print('*********************************************')
        print('\n')
```

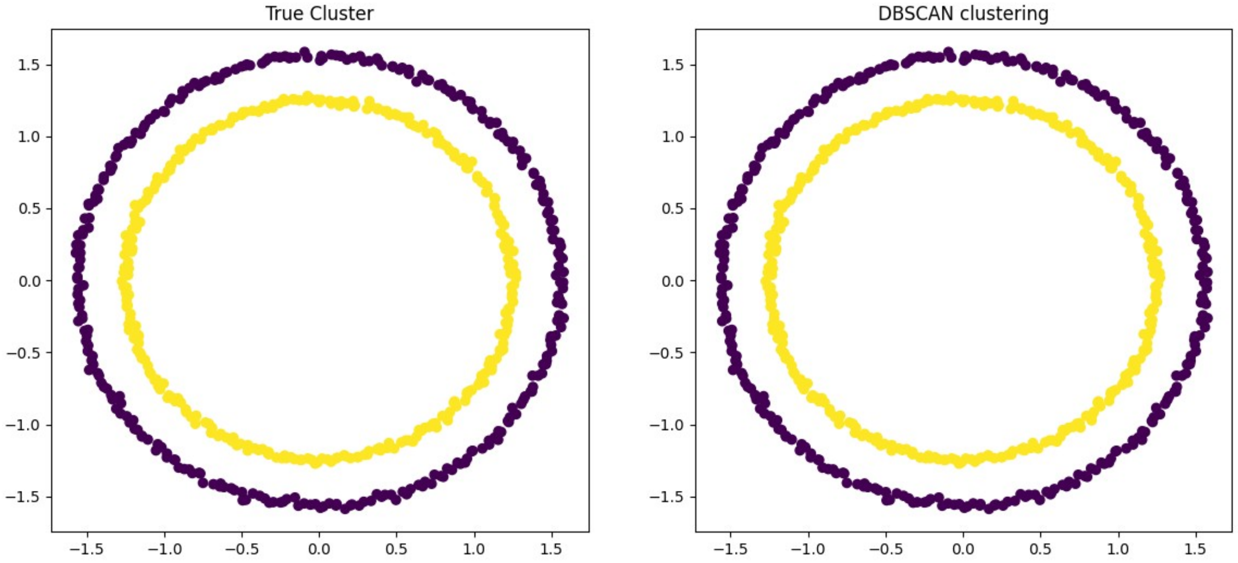
```
For eps = 0.2 and minimum samples = 5
Silhouette score: 0.019
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```



```
*********************************************


For eps = 0.2 and minimum samples = 10
Silhouette score: 0.019
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```

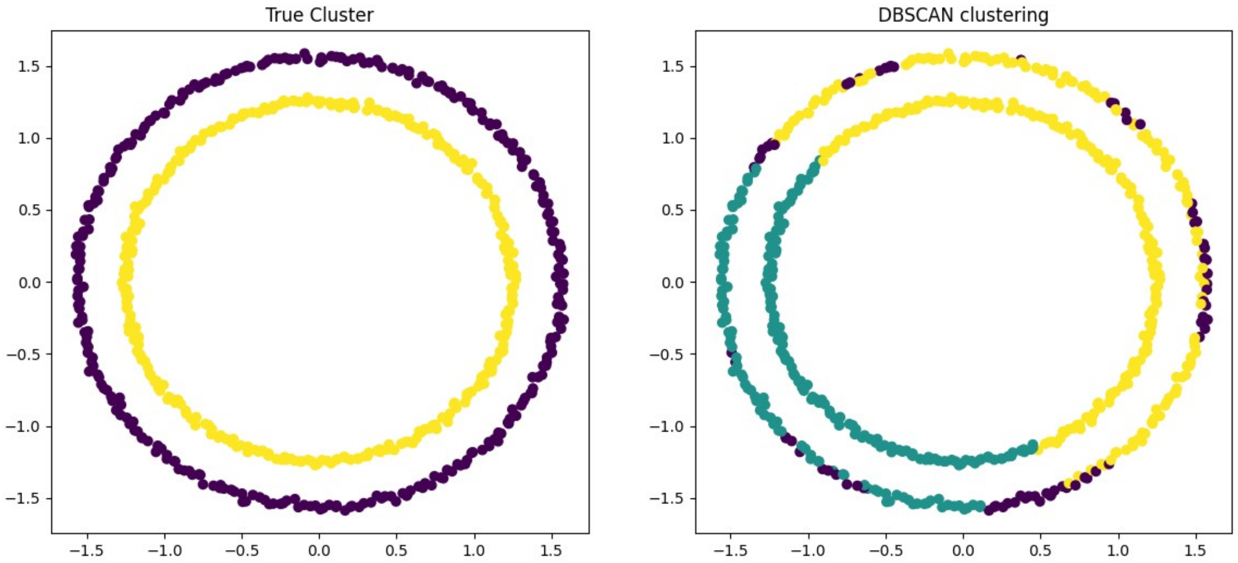


```
*********************************************


For eps = 0.2 and minimum samples = 15
Silhouette score: 0.019
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```



```
*********************************************


For eps = 0.3 and minimum samples = 25
Silhouette score: 0.230
```

```
Adjusted Rand Index: 0.043
Adjusted mutual information: 0.110
```



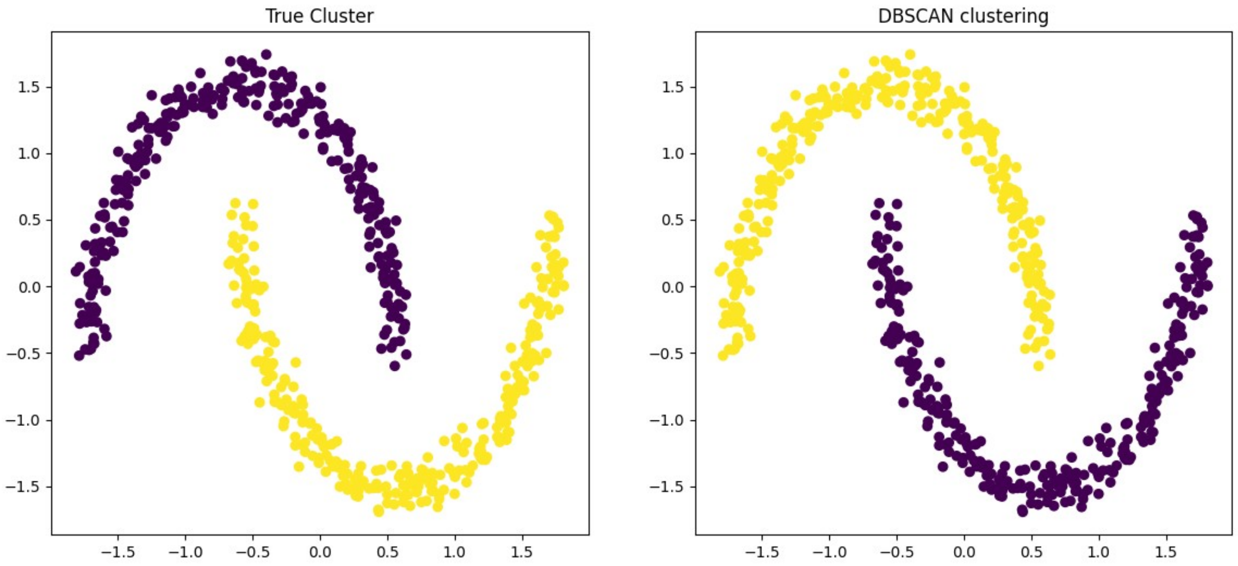
```
*********************************************
```

Observations

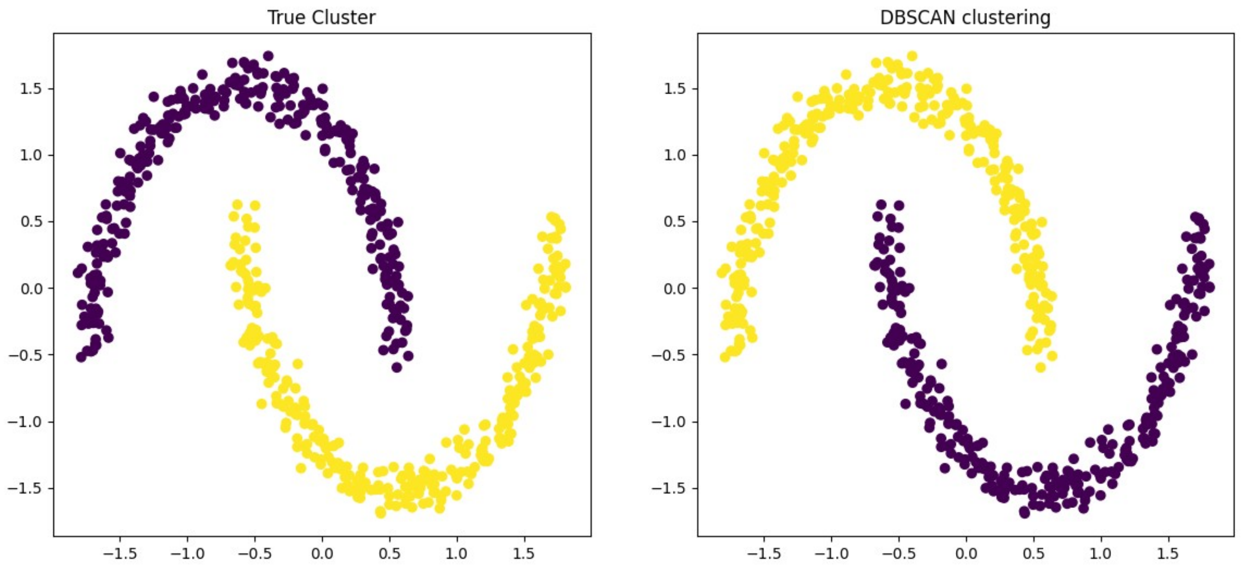
1. The scores are very bad for circular data.
2. Intrestingly, for eps = 0.2, any number of min_samples is generating a ari and ami score = 1, which implies that both the true cluster and predicted one agrees upon the placing of the points but the silhouette score is very bad.
3. We can conclude that DBSCAN clustering the data in proper number of clusters but it is not assigning correct datapoints to right cluster.

For moon data

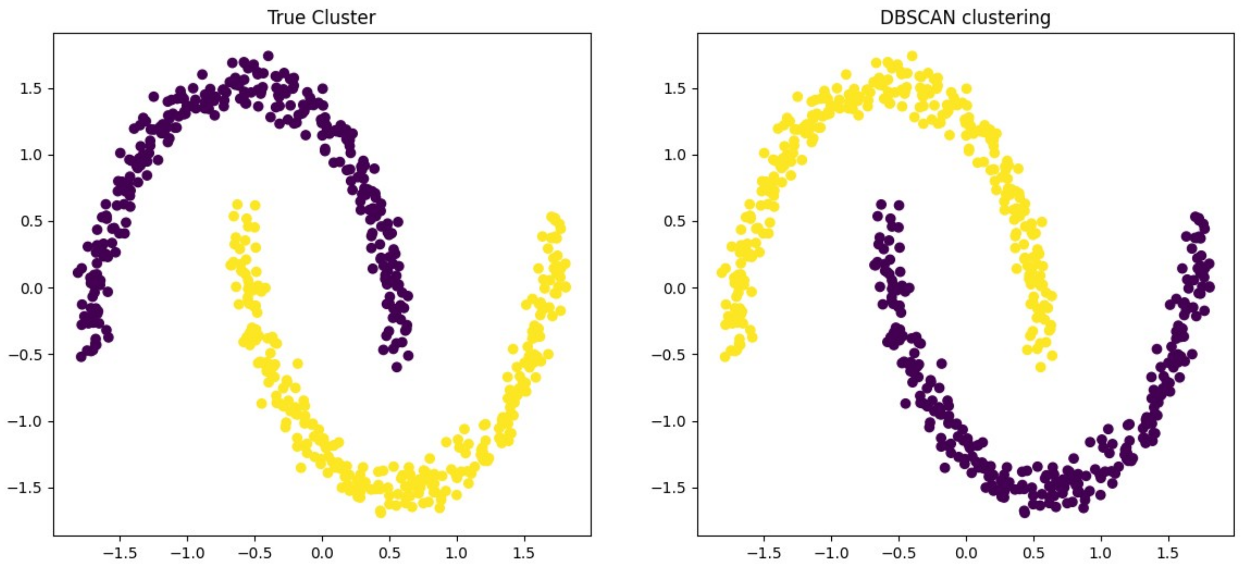
```
eps_values_moon = [0.3, 0.5, 0.7]
min_samples_values_moon = [5, 10, 15, 20]

for i, eps in enumerate(eps_values_moon):
  for j, min_samples in enumerate(min_samples_values_moon):
    # Fit DBSCAN with current parameter combination
    dbscan_moon = DBSCAN(eps=eps, min_samples=min_samples)
    dbscan_moon.fit_predict(scaled_moon_X)

    # Calculate performance metrics
    X, pred_labels, true_labels = scaled_moon_X, dbscan_moon.labels_,
moon_true_labels
    if len(set(pred_labels)) > 1:
      silhouette = metrics.silhouette_score(X, pred_labels)
      ari = metrics.adjusted_rand_score(true_labels, pred_labels)
      ami = metrics.adjusted_mutual_info_score(true_labels,
pred_labels)
      print(f'For eps = {eps} and minimum samples = {min_samples}')
      print("Silhouette score: %0.3f " % silhouette)
      print('Adjusted Rand Index: %0.3f'% ari)
      print('Adjusted mutual information: %0.3f' % ami)
      plot_clusters(X, true_labels, pred_labels)
      print('*********************************************')
      print('\n')

For eps = 0.3 and minimum samples = 5
Silhouette score: 0.389
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```



```
*********************************************


For eps = 0.3 and minimum samples = 10
Silhouette score: 0.389
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```

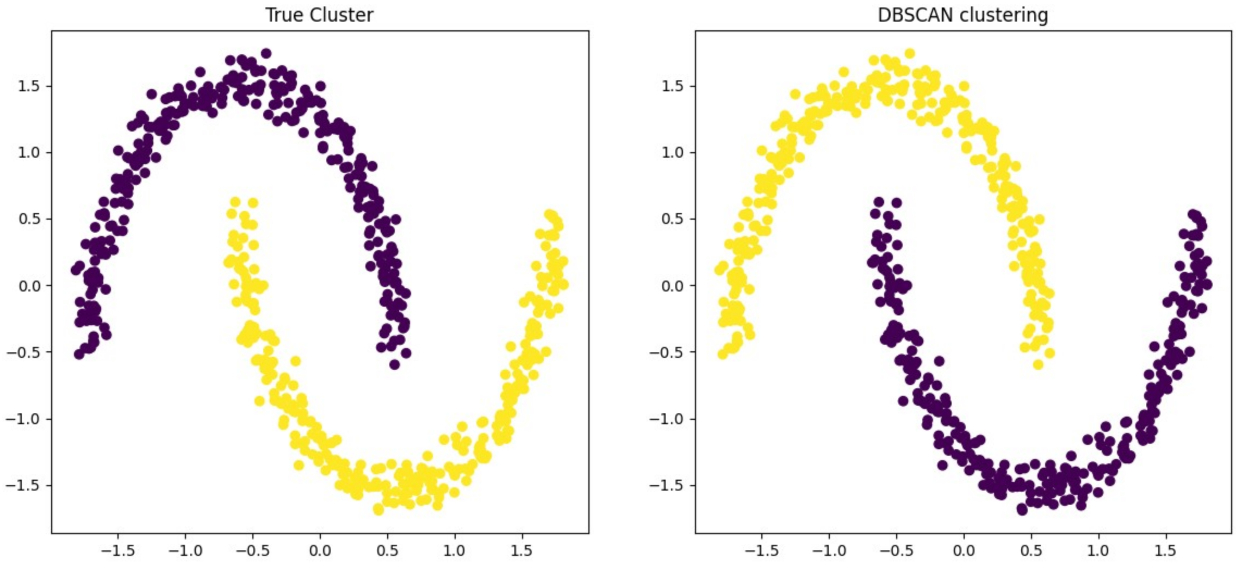


```
*********************************************


For eps = 0.3 and minimum samples = 15
Silhouette score: 0.389
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```

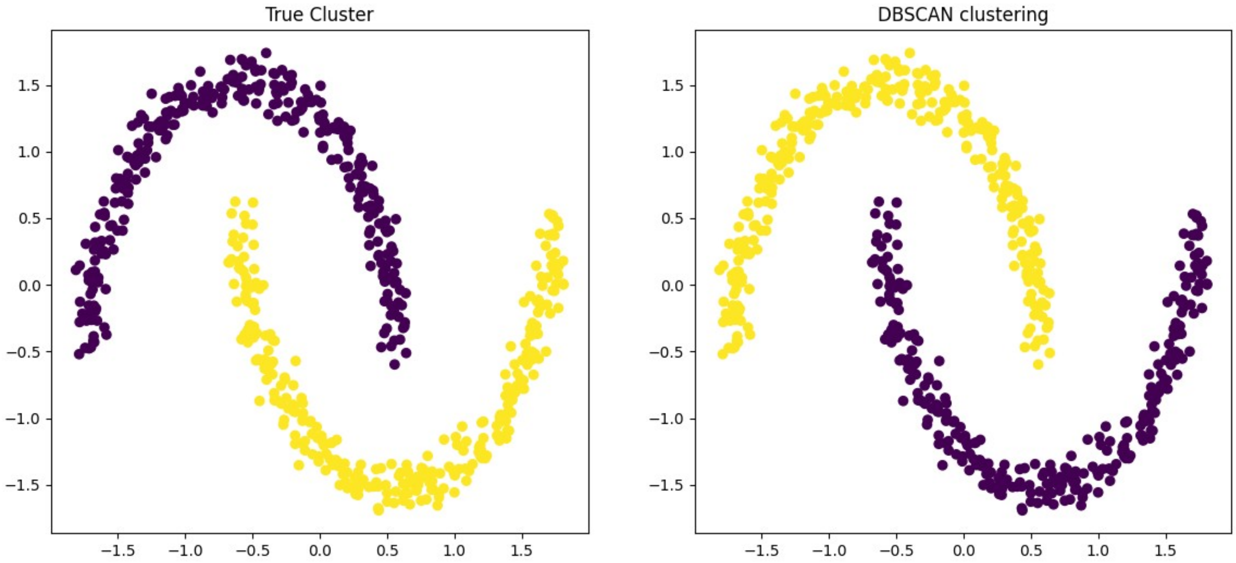


```
*********************************************


For eps = 0.3 and minimum samples = 20
Silhouette score: 0.389
```

```
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```

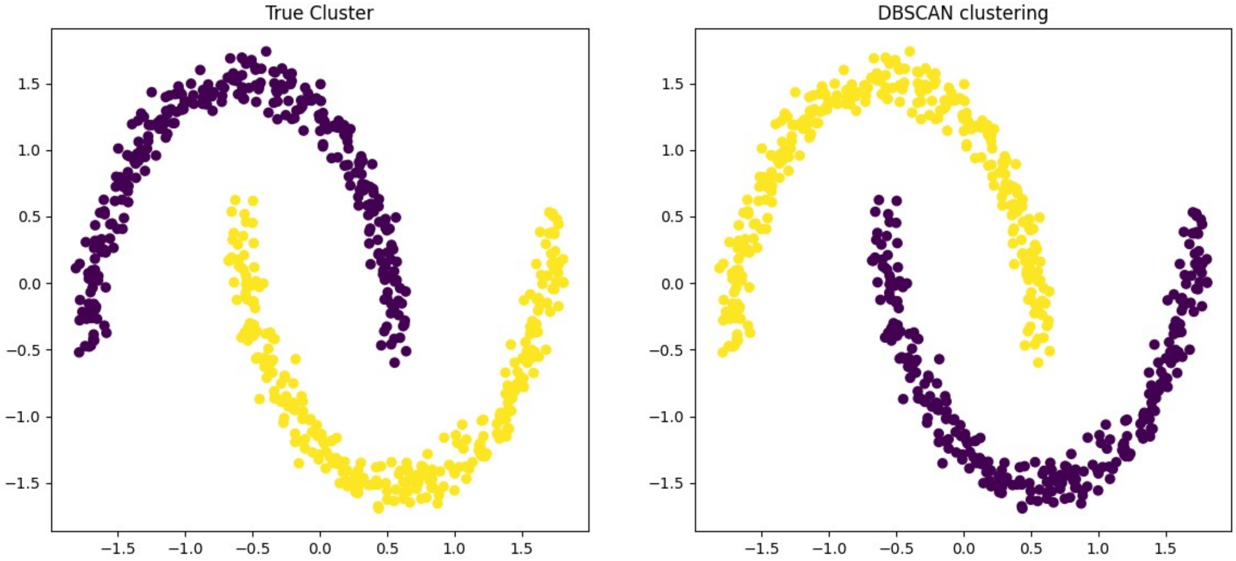


```
*********************************************


For eps = 0.5 and minimum samples = 5
Silhouette score: 0.389
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```

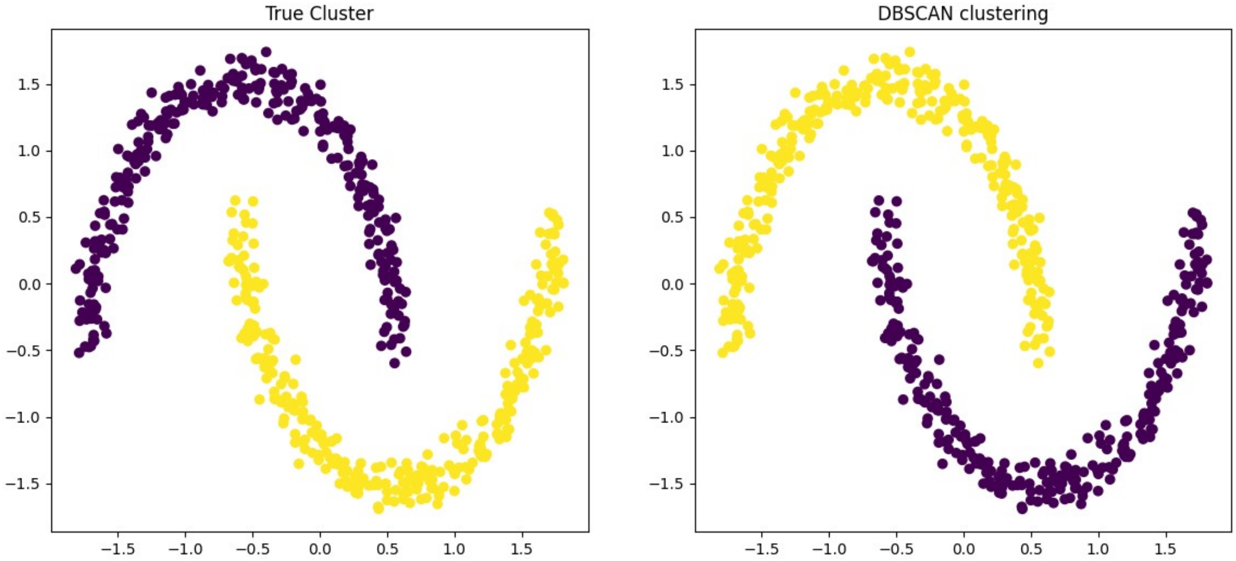


```
*********************************************
```

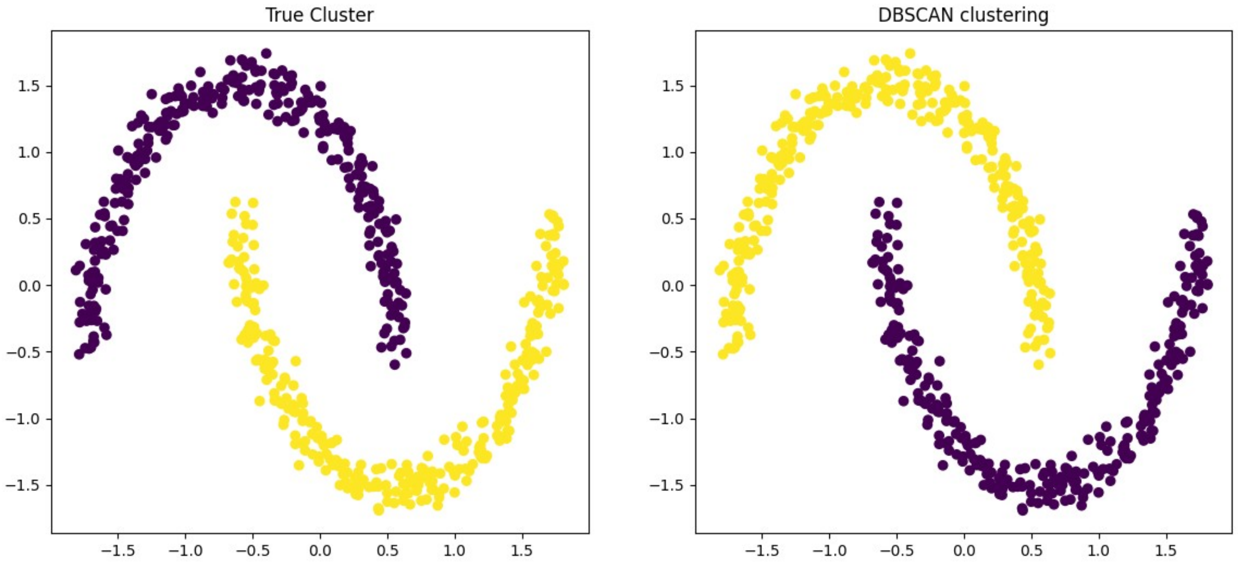
```

For eps = 0.5 and minimum samples = 10
Silhouette score: 0.389
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```



```
*********************************************


For eps = 0.5 and minimum samples = 15
Silhouette score: 0.389
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```



```
*********************************************


For eps = 0.5 and minimum samples = 20
Silhouette score: 0.389
Adjusted Rand Index: 1.000
Adjusted mutual information: 1.000
```



```
*********************************************
```

Observations

1. For moon dataset, as we can see the scores are consistent irrespective of the eps and min_samples values.
2. Again we are getting ari and ami = 1, which implies that DBSCAN is doing the placing of the points.
3. So we can take any eps and min_sample value for moon dataset.