ASSIGNMENT - 4

Date	30 Oct 2022
Team ID	PNT2022TMID38667
Project Name	Project – Early Detection of Chronic Kidney Disease using Machine Learning

1. Download the dataset .

	A	B	C	D	E	F	G
1	CustomerI	Gender	Age	Annual Inc	Spending Score (1-100)		
2	1	Male	19	15	39		
3	2	Male	21	15	81		
4	3	Female	20	16	6		
5	4	Female	23	16	77		
6	5	Female	31	17	40		
7	6	Female	22	17	76		
8	7	Female	35	18	6		
9	8	Female	23	18	94		
10	9	Male	64	19	3		
11	10	Female	30	19	72		
12	11	Male	67	19	14		
13	12	Female	35	19	99		
14	13	Female	58	20	15		
15	14	Female	24	20	77		
16	15	Male	37	20	13		
17	16	Male	22	20	79		
18	17	Female	35	21	35		
19	18	Male	20	21	66		
20	19	Male	52	23	29		
21	20	Female	35	23	98		
22	21	Male	35	24	35		
23	22	Male	25	24	73		
24	23	Female	46	25	5		
25	24	Male	31	25	73		
26	25	Female	54	28	14		
27	26	Male	29	28	82		
28	27	Female	45	28	32		
29	28	Male	35	28	61		
30	29	Female	40	29	31		
31	30	Female	23	29	87		

2. Load the dataset into the tool.

In [6]:
```
df = pd.read_csv("Mall_Customers.csv")
df.head(10)
```

Out[6]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72

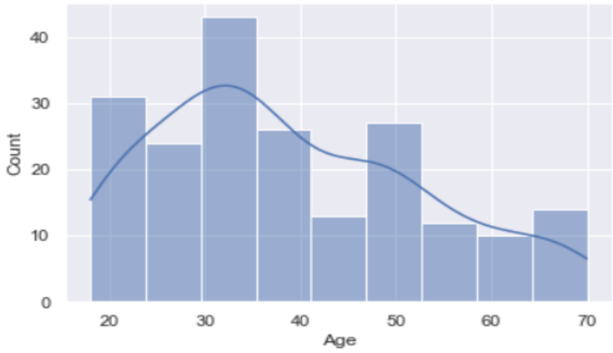
In [7]:
```
df.drop(['CustomerID'],axis=1,inplace=True)
df.head(10)
```
Out[7]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40
5	Female	22	17	76
6	Female	35	18	6
7	Female	23	18	94
8	Male	64	19	3
9	Female	30	19	72

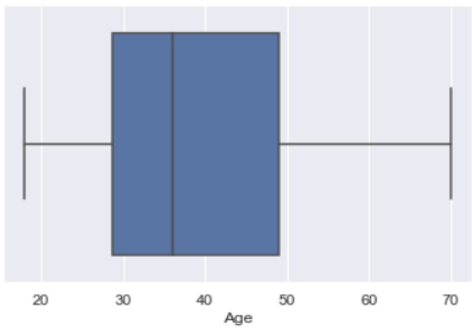
3. Perform Below Visualizations.

- **Univariate Analysis**

In [11]:
```
sns.histplot(df['Age'],kde=True)
```
Out[11]: <AxesSubplot:xlabel='Age', ylabel='Count'>

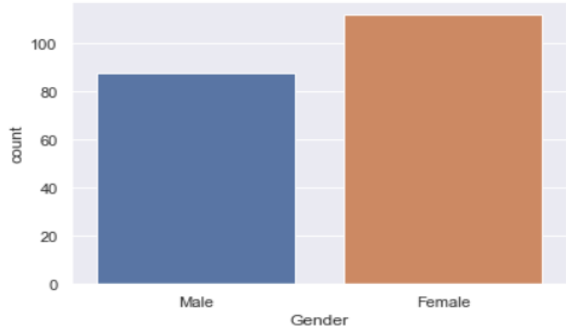


In [10]:
```
sns.boxplot(df['Age'],orient='h')
```
Out[10]: <AxesSubplot:xlabel='Age'>



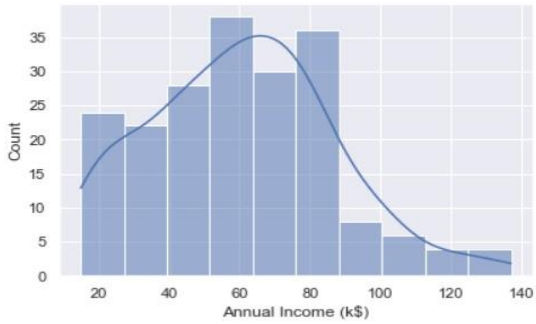
```
In [12]: sns.countplot(x='Gender',data=df)
```

```
Out[12]: <AxesSubplot:xlabel='Gender', ylabel='count'>
```



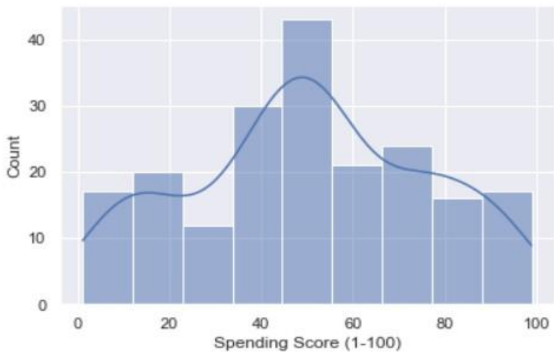

```
In [13]: sns.histplot(df['Annual Income (k$)'], kde=True)
```

```
Out[13]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Count'>
```



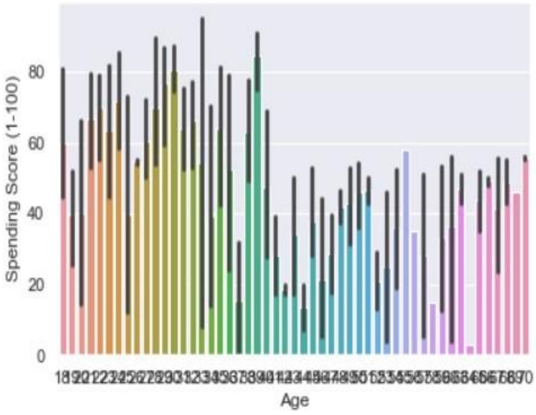

```
In [14]: sns.histplot(df['Spending Score (1-100)'], kde=True)
```

```
Out[14]: <AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Count'>
```

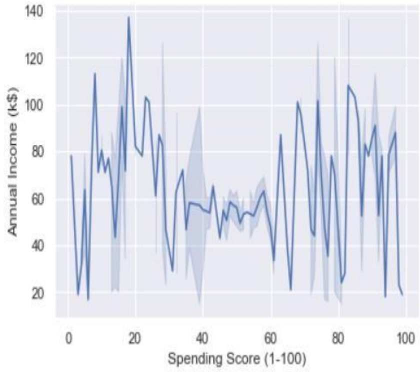


- **Bi- Variate Analysis**

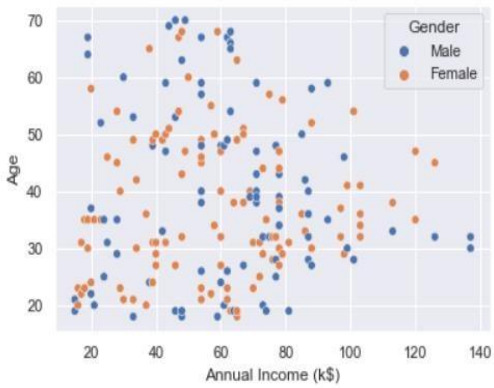
```
In [15]: sns.barplot(x='Age',y='Spending Score (1-100)',data=df)
Out[15]: <AxesSubplot:xlabel='Age', ylabel='Spending Score (1-100)'>
```




```
In [16]: sns.lineplot(x='Spending Score (1-100)', y='Annual Income (k$)', data=df)
Out[16]: <AxesSubplot:xlabel='Spending Score (1-100)', ylabel='Annual Income (k$)'>
```

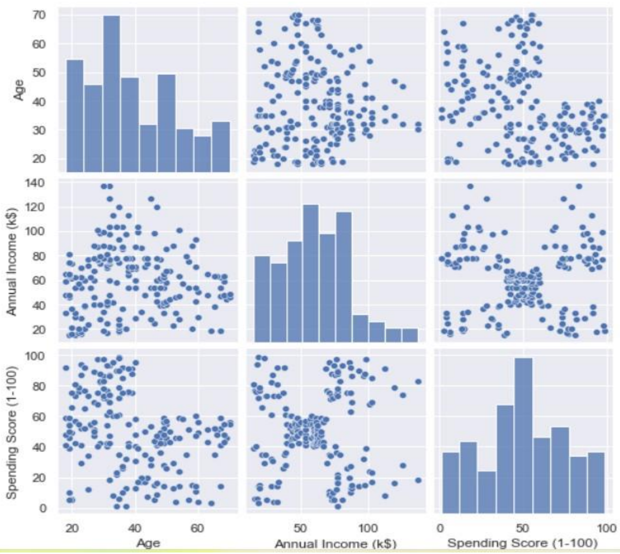



```
In [17]: sns.scatterplot(x='Annual Income (k$)',y='Age',hue='Gender',data=df)
Out[17]: <AxesSubplot:xlabel='Annual Income (k$)', ylabel='Age'>
```

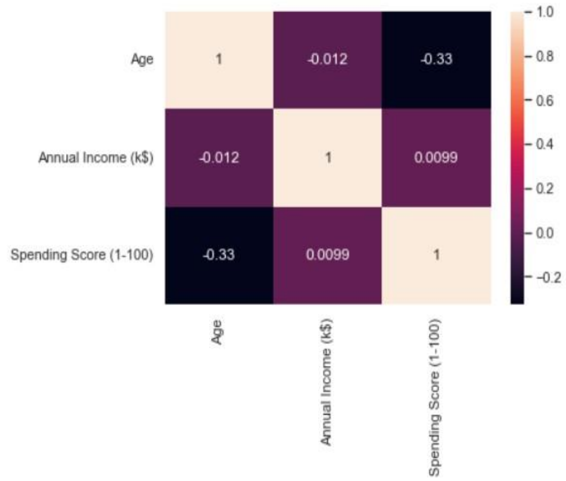


- **Multi-Variate Analysis**

In [19]:
```
sns.pairplot(data=df[["Gender", "Age","Annual Income (k$)","Spending Score (1-100)"]])
```
Out[19]: <seaborn.axisgrid.PairGrid at 0x1dd12de6190>



In [20]:
```
sns.heatmap(df.corr(),annot=True)
```
Out[20]: <AxesSubplot:>



4. Perform descriptive statistics on the dataset.

In [21]:
```
df.describe()
```
Out[21]:

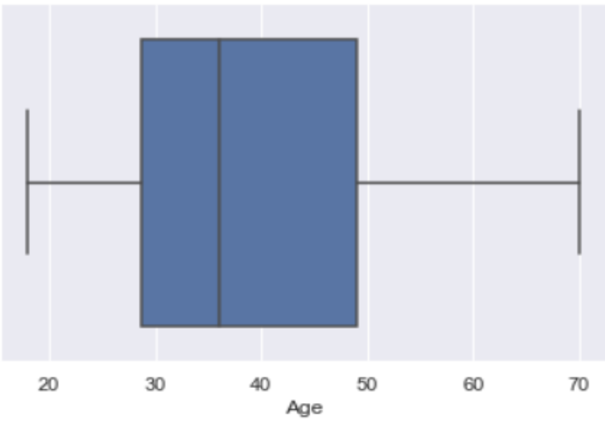
	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000
mean	38.850000	60.560000	50.200000
std	13.969007	26.264721	25.823522
min	18.000000	15.000000	1.000000
25%	28.750000	41.500000	34.750000
50%	36.000000	61.500000	50.000000
75%	49.000000	78.000000	73.000000
max	70.000000	137.000000	99.000000

5. Check for Missing values and deal with them.

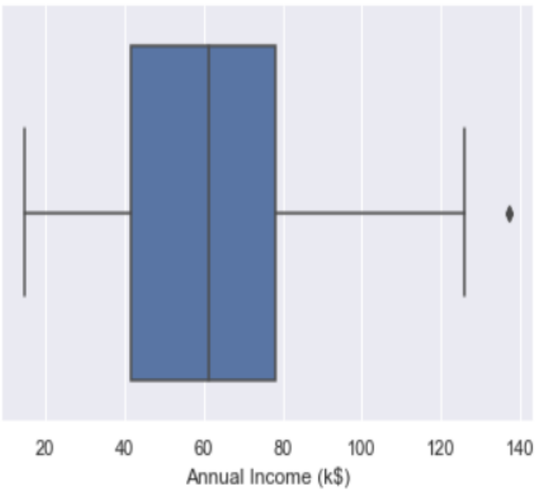
```
In [22]: df.isnull().sum()

Out[22]: Gender                    0
         Age                       0
         Annual Income (k$)        0
         Spending Score (1-100)    0
         dtype: int64
```

6. Find the outliers and replace them outliers.

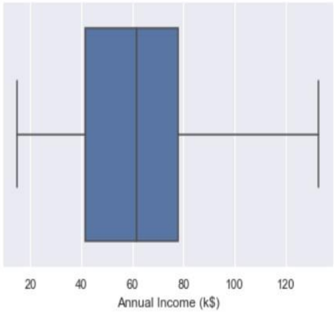
```
In [23]: sns.boxplot(df['Age'], orient='h')

Out[23]: <AxesSubplot:xlabel='Age'>
```



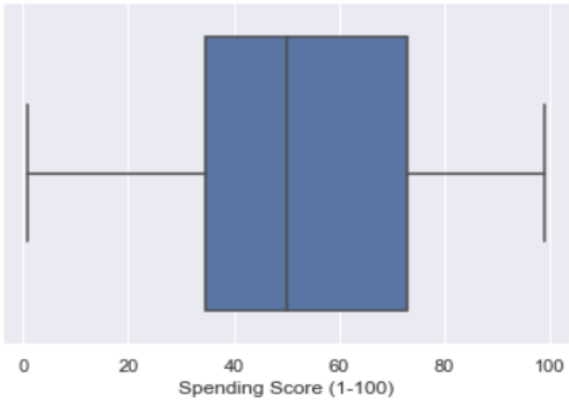

```
In [26]: sns.boxplot(df['Annual Income (k$)'], orient='h')

Out[26]: <AxesSubplot:xlabel='Annual Income (k$)'>
```



```
In [34]: q = df['Annual Income (k$)'].quantile(q=[0.75,0.25])
         iqr=q.iloc[0]-q.iloc[1]
         lower = q.iloc[1] - 1.5*iqr
         upper = q.iloc[0] + 1.5*iqr
         df['Annual Income (k$)'] = np.where(df['Annual Income (k$)']>upper,upper,np.where(df['Annual Income (k$)']<lower,lower,df['Annual
         sns.boxplot(df['Annual Income (k$)'], orient='h')

Out[34]: <AxesSubplot:xlabel='Annual Income (k$)'>
```




```
In [31]: sns.boxplot(df['Spending Score (1-100)'], orient='h')

Out[31]: <AxesSubplot:xlabel='Spending Score (1-100)'>
```



7. Check for Categorical columns and perform encoding.

```
In [33]: l_en = LabelEncoder()
         df['Gender'] = l_en.fit_transform(df['Gender'])
         df.head(10)

Out[33]:
```

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15.0	39
1	1	21	15.0	81
2	0	20	16.0	6
3	0	23	16.0	77
4	0	31	17.0	40
5	0	22	17.0	76
6	0	35	18.0	6
7	0	23	18.0	94
8	1	64	19.0	3
9	0	30	19.0	72

8. Scaling the data.

In [36]:
```
scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(df)
scaled_data[0:5]
```

Out[36]:
```
array([[1.        , 0.01923077, 0.        , 0.3877551 ],
       [1.        , 0.05769231, 0.        , 0.81632653],
       [0.        , 0.03846154, 0.00849257, 0.05102041],
       [0.        , 0.09615385, 0.00849257, 0.7755102 ],
       [0.        , 0.25      , 0.01698514, 0.39795918]])
```

9. Perform any of the clustering algorithms

In [37]:
```
from sklearn.cluster import KMeans
km = KMeans(algorithm='elkan',n_init=100, max_iter=3000)
res = km.fit_predict(scaled_data)
res
```

Out[37]:
```
array([6, 6, 4, 4, 4, 4, 7, 4, 5, 4, 5, 4, 7, 4, 3, 6, 4, 6, 5, 4, 6, 6,
       7, 6, 7, 6, 7, 6, 7, 4, 5, 4, 5, 6, 7, 4, 7, 4, 7, 4, 7, 6, 5, 4,
       7, 4, 7, 4, 4, 4, 7, 6, 4, 5, 7, 5, 7, 5, 4, 5, 5, 6, 7, 7, 5, 6,
       7, 7, 6, 4, 5, 7, 7, 7, 5, 6, 7, 6, 4, 7, 5, 6, 5, 7, 4, 5, 7, 4,
       4, 7, 7, 6, 5, 7, 4, 6, 7, 4, 5, 6, 4, 7, 5, 6, 5, 4, 7, 5, 5, 5,
       5, 4, 7, 6, 4, 4, 7, 7, 7, 7, 6, 7, 2, 1, 4, 2, 3, 1, 5, 1, 3, 1,
       4, 2, 3, 2, 0, 1, 3, 2, 0, 1, 4, 2, 3, 1, 5, 2, 0, 1, 3, 1, 0, 2,
       0, 2, 3, 2, 3, 2, 7, 2, 3, 2, 3, 2, 3, 2, 0, 1, 3, 1, 3, 1, 0, 2,
       5, 1, 5, 1, 0, 2, 3, 2, 0, 1, 0, 1, 0, 2, 0, 2, 3, 2, 0, 2, 0, 1,
       3, 1])
```

In [38]:
```
df1 = pd.DataFrame(scaled_data, columns = df.columns)
df1.head(10)
```

Out[38]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.0	0.019231	0.000000	0.387755
1	1.0	0.057692	0.000000	0.816327
2	0.0	0.038462	0.008493	0.051020
3	0.0	0.096154	0.008493	0.775510
4	0.0	0.250000	0.016985	0.397959
5	0.0	0.076923	0.016985	0.765306
6	0.0	0.326923	0.025478	0.051020
7	0.0	0.096154	0.025478	0.948980
8	1.0	0.884615	0.033970	0.020408
9	0.0	0.230769	0.033970	0.724490

10. Add the cluster data with the primary dataset

In [39]:
```
df1['Cluster']  = pd.Series(res)
df1.head(10)
```

Out[39]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)	Cluster
0	1.0	0.019231	0.000000	0.387755	6
1	1.0	0.057692	0.000000	0.816327	6
2	0.0	0.038462	0.008493	0.051020	4
3	0.0	0.096154	0.008493	0.775510	4
4	0.0	0.250000	0.016985	0.397959	4
5	0.0	0.076923	0.016985	0.765306	4
6	0.0	0.326923	0.025478	0.051020	7
7	0.0	0.096154	0.025478	0.948980	4
8	1.0	0.884615	0.033970	0.020408	5
9	0.0	0.230769	0.033970	0.724490	4

In [41]:
```
df1['Cluster'].unique()
```

Out[41]:
```
array([6, 4, 7, 5, 3, 2, 1, 0])
```

In [42]:
```
df1['Cluster'].value_counts()
```

Out[42]:
```
7    39
4    37
5    29
6    24
2    22
1    18
3    17
0    14
Name: Cluster, dtype: int64
```

11. Split the data into dependent and independent variables.

In [43]:
```
# independent variable
X = df1.iloc[:,0:4]
X.head(10)
```

Out[43]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1.0	0.019231	0.000000	0.387755
1	1.0	0.057692	0.000000	0.816327
2	0.0	0.038462	0.008493	0.051020
3	0.0	0.096154	0.008493	0.775510
4	0.0	0.250000	0.016985	0.397959
5	0.0	0.076923	0.016985	0.765306
6	0.0	0.326923	0.025478	0.051020
7	0.0	0.096154	0.025478	0.948980
8	1.0	0.884615	0.033970	0.020408
9	0.0	0.230769	0.033970	0.724490

In [44]:
```
# dependent variable
y = df1.iloc[:,4:]
y.head(10)
```

Out[44]:

	Cluster
0	6
1	6
2	4
3	4
4	4
5	4
6	7
7	4
8	5
9	4

12. Split the data into training and testing

In [45]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=1)
X_train.head(10)
```

Out[45]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
116	0.0	0.865385	0.424628	0.428571
67	0.0	0.961538	0.280255	0.479592
78	0.0	0.096154	0.331210	0.520408
42	1.0	0.576923	0.203822	0.357143
17	1.0	0.038462	0.050955	0.663265
5	0.0	0.076923	0.016985	0.765306
127	1.0	0.423077	0.475584	0.959184
105	0.0	0.057692	0.399151	0.418367
48	0.0	0.211538	0.212314	0.418367
66	0.0	0.480769	0.280255	0.500000

In [46]:
```
X_test.head(10)
```

Out[46]:

	Gender	Age	Annual Income (k$)	Spending Score (1-100)
58	0.0	0.173077	0.263270	0.510204
40	0.0	0.903846	0.195329	0.346939
34	0.0	0.596154	0.152866	0.132653
102	1.0	0.942308	0.399151	0.591837
184	0.0	0.442308	0.713376	0.387755
198	1.0	0.269231	1.000000	0.173469
95	1.0	0.115385	0.382166	0.520408
4	0.0	0.250000	0.016985	0.397959
29	0.0	0.096154	0.118896	0.877551
168	0.0	0.346154	0.611465	0.265306

In [47]:
```
y_train.head(10)
```

Out[47]:

	Cluster
116	7
67	7
78	4
42	5
17	6
5	4
127	1
105	4
48	4
66	7

```
In [48]: y_test.head(10)
Out[48]:
```

	Cluster
58	4
40	7
34	7
102	5
184	0
198	3
95	6
4	4
29	4
168	0

13. Build the Model

```
In [49]: # classification algorithm
         classifier_model = SVC(decision_function_shape='ovo')
```

14. Train the Model

```
In [50]: classifier_model.fit(X_train,y_train.values.flatten())

Out[50]: SVC(decision_function_shape='ovo')
```

15. Test the Model

```
In [51]: pred_y = classifier_model.predict(X_test)
         pred_y[0:5]

Out[51]: array([4, 7, 7, 5, 0])
```

16. Measure the performance using Evaluation Metrics.

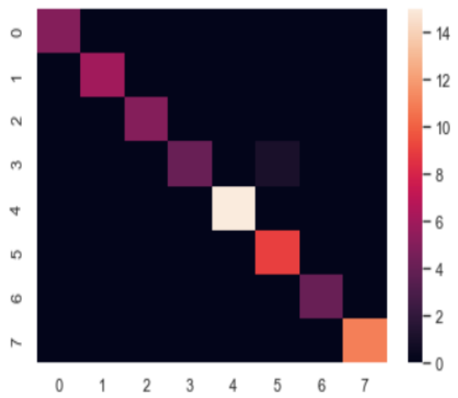
In [52]:
```
print('Classification Report: ')
print(classification_report(y_test, pred_y))
```

```
Classification Report: 
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         5
           1       1.00      1.00      1.00         6
           2       1.00      1.00      1.00         5
           3       1.00      0.80      0.89         5
           4       1.00      1.00      1.00        15
           5       0.90      1.00      0.95         9
           6       1.00      1.00      1.00         4
           7       1.00      1.00      1.00        11

    accuracy                           0.98        60
   macro avg       0.99      0.97      0.98        60
weighted avg       0.98      0.98      0.98        60
```

In [53]:
```
print('Confusion Matrix: ')
sns.heatmap(confusion_matrix(y_test,pred_y))
```

```
Confusion Matrix:
```

Out[53]: <AxesSubplot:>



In [54]:
```
print('F1 Score: ',f1_score(y_test,pred_y, average='weighted'))
```

```
F1 Score:  0.9828460038986354
```

In [55]:
```
# Hamming loss gives the fraction of labels that are incorrectly predicted
print('Hamming Loss: ',hamming_loss(y_test,pred_y))
```

```
Hamming Loss:  0.016666666666666666
```

In [56]:
```
print('Accuracy: ',accuracy_score(y_test,pred_y))
```

```
Accuracy:  0.9833333333333333
```