1. Download the dataset: Dataset

2. Load the dataset into the tool.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import warnings
```

```
data=pd.read_csv("Mall_Customers.csv",encoding='ISO-8859-1')
data.head()
```

	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

```
data.describe()
```

	CustomerID	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

```
data.dtypes
```

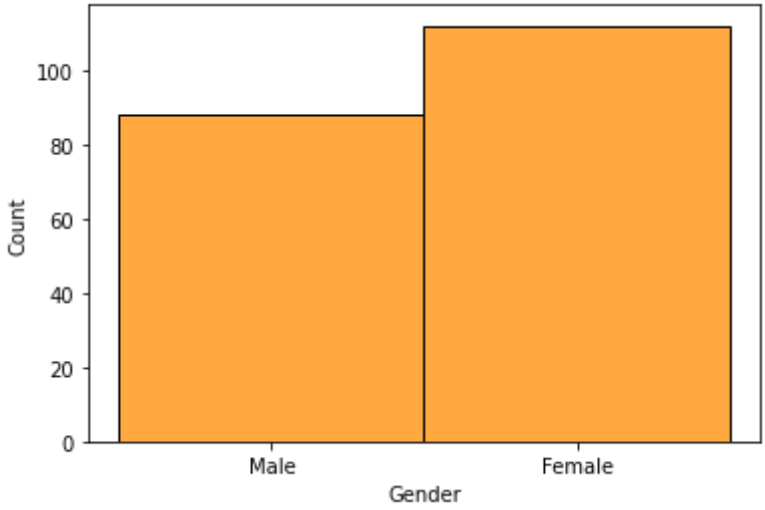
```
CustomerID                int64
Gender                   object
```

```
Age                     int64
Annual Income (k$)      int64
Spending Score (1-100)  int64
dtype: object
```

3. Perform Below Visualizations. · Univariate Analysis · Bi- Variate Analysis · Multi-Variate Analysis

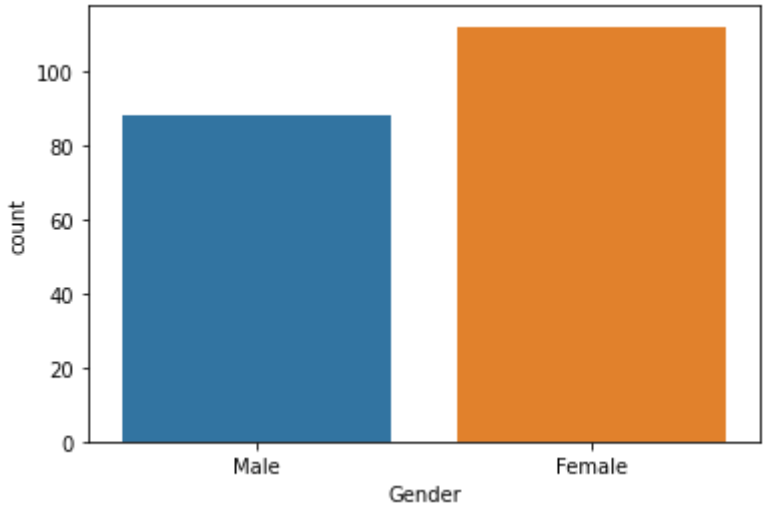
```
#univariate analysis "Histogram"
sns.histplot(data["Gender"],color='darkorange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7815375f50>
```



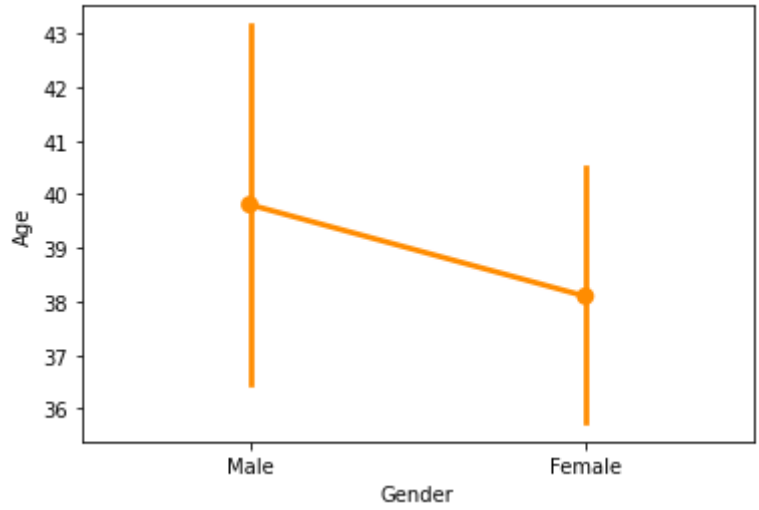

```
#univariate analysis "Countlot"
sns.countplot(data['Gender'])
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f78152d6f90>
```




```
#bivariate analysis"Pointplot"
sns.pointplot(x='Gender',y='Age',data=data,color='darkorange')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7814e002d0>
```




```
#Multivariate analysis"Pairplot"
sns.pairplot(data)
```

```
<seaborn.axisgrid.PairGrid at 0x7f7814d7f3d0>
```

4. Perform descriptive statistics on the dataset.

```
# Descriptive statistics of the data set accessed.
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
CustomerID	200.0	100.50	57.879185	1.0	50.75	100.5	150.25	200.0
Age	200.0	38.85	13.969007	18.0	28.75	36.0	49.00	70.0
Annual Income (k$)	200.0	60.56	26.264721	15.0	41.50	61.5	78.00	137.0
Spending Score (1-100)	200.0	50.20	25.823522	1.0	34.75	50.0	73.00	99.0

```
data.isnull().any().any()
```

```
False
```

```
data.isnull().any()
```

```
CustomerID              False
Gender                  False
Age                     False
Annual Income (k$)      False
Spending Score (1-100)  False
dtype: bool
```

```
df2=data.dropna(how='all')
```

5. Check for Missing values and deal with them.

```
df2.isnull().sum().sum()
```

```
0
```

This dataset does not contain any missing value

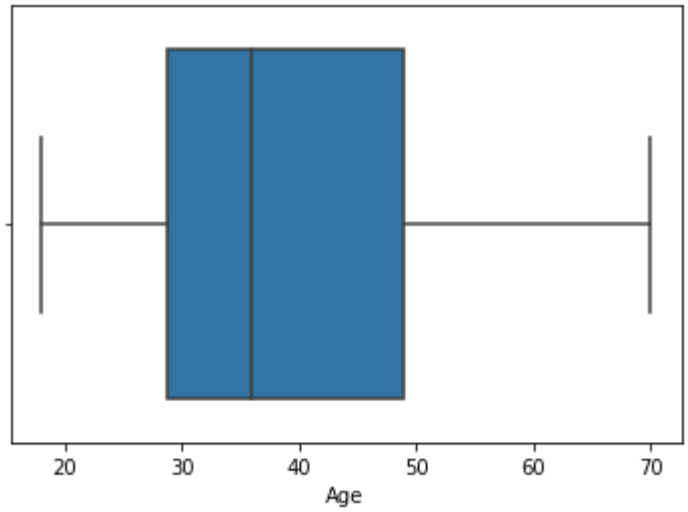
```
missing_values=data.isnull().sum()
missing_values[missing_values>0]/len(data)*100
```

```
Series([], dtype: float64)
```

6. Find the outliers and replace them outliers

```
sns.boxplot(data['Age'],data=data)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f7810504090>
```




```
for x in ['Age']:
    q75,q25 = np.percentile(data.loc[:,x],[75,25])
    intr_qr = q75-q25

    max = q75+(1.5*intr_qr)
    min = q25-(1.5*intr_qr)

    data.loc[data[x] < min,x] = np.nan
    data.loc[data[x] > max,x] = np.nan
```

```
data.isnull().sum()
```

```
CustomerID                0
Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64
```

7. Check for Categorical columns and perform encoding.

```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
data['Gender']=encoder.fit_transform(data['Gender'])
```

```
data.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	1	19.0	15	39
1	2	1	21.0	15	81
2	3	0	20.0	16	6
3	4	0	23.0	16	77

8. Scaling the data

```
from sklearn.preprocessing import StandardScaler
df=StandardScaler()
data1=df.fit_transform(data)
```

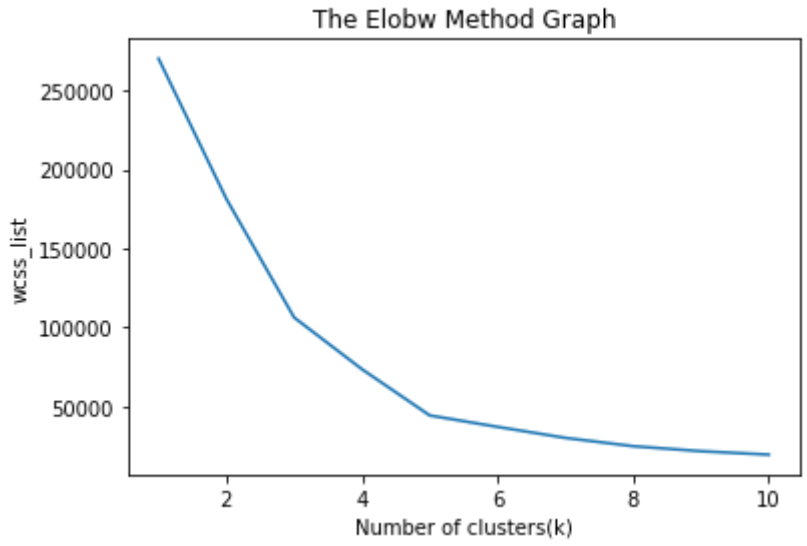
```
data1
```

```
array([[-1.7234121 ,  1.12815215, -1.42456879, -1.73899919, -0.43480148],
       [-1.70609137,  1.12815215, -1.28103541, -1.73899919,  1.19570407],
       [-1.68877065, -0.88640526, -1.3528021 , -1.70082976, -1.71591298],
       [-1.67144992, -0.88640526, -1.13750203, -1.70082976,  1.04041783],
       [-1.6541292 , -0.88640526, -0.56336851, -1.66266033, -0.39597992],
       [-1.63680847, -0.88640526, -1.20926872, -1.66266033,  1.00159627],
       [-1.61948775, -0.88640526, -0.27630176, -1.62449091, -1.71591298],
       [-1.60216702, -0.88640526, -1.13750203, -1.62449091,  1.70038436],
       [-1.5848463 ,  1.12815215,  1.80493225, -1.58632148, -1.83237767],
       [-1.56752558, -0.88640526, -0.6351352 , -1.58632148,  0.84631002],
       [-1.55020485,  1.12815215,  2.02023231, -1.58632148, -1.4053405 ],
       [-1.53288413, -0.88640526, -0.27630176, -1.58632148,  1.89449216],
       [-1.5155634 , -0.88640526,  1.37433211, -1.54815205, -1.36651894],
       [-1.49824268, -0.88640526, -1.06573534, -1.54815205,  1.04041783],
       [-1.48092195,  1.12815215, -0.13276838, -1.54815205, -1.44416206],
       [-1.46360123,  1.12815215, -1.20926872, -1.54815205,  1.11806095],
       [-1.4462805 , -0.88640526, -0.27630176, -1.50998262, -0.59008772],
       [-1.42895978,  1.12815215, -1.3528021 , -1.50998262,  0.61338066],
       [-1.41163905,  1.12815215,  0.94373197, -1.43364376, -0.82301709],
       [-1.39431833, -0.88640526, -0.27630176, -1.43364376,  1.8556706 ],
       [-1.3769976 ,  1.12815215, -0.27630176, -1.39547433, -0.59008772],
       [-1.35967688,  1.12815215, -0.99396865, -1.39547433,  0.88513158],
       [-1.34235616, -0.88640526,  0.51313183, -1.3573049 , -1.75473454],
       [-1.32503543,  1.12815215, -0.56336851, -1.3573049 ,  0.88513158],
       [-1.30771471, -0.88640526,  1.08726535, -1.24279661, -1.4053405 ],
       [-1.29039398,  1.12815215, -0.70690189, -1.24279661,  1.23452563],
       [-1.27307326, -0.88640526,  0.44136514, -1.24279661, -0.7065524 ],
       [-1.25575253,  1.12815215, -0.27630176, -1.24279661,  0.41927286],
       [-1.23843181, -0.88640526,  0.08253169, -1.20462718, -0.74537397],
       [-1.22111108, -0.88640526, -1.13750203, -1.20462718,  1.42863343],
       [-1.20379036,  1.12815215,  1.51786549, -1.16645776, -1.7935561 ],
       [-1.18646963, -0.88640526, -1.28103541, -1.16645776,  0.88513158],
       [-1.16914891,  1.12815215,  1.01549866, -1.05194947, -1.7935561 ],
       [-1.15182818,  1.12815215, -1.49633548, -1.05194947,  1.62274124],
       [-1.13450746, -0.88640526,  0.7284319 , -1.05194947, -1.4053405 ],
       [-1.11718674, -0.88640526, -1.28103541, -1.05194947,  1.19570407],
       [-1.09986601, -0.88640526,  0.22606507, -1.01378004, -1.28887582],
       [-1.08254529, -0.88640526, -0.6351352 , -1.01378004,  0.88513158],
       [-1.06522456, -0.88640526, -0.20453507, -0.89927175, -0.93948177],
```

```
[-1.04790384, -0.88640526, -1.3528021 , -0.89927175,  0.96277471],
[-1.03058311, -0.88640526,  1.87669894, -0.86110232, -0.59008772],
[-1.01326239,  1.12815215, -1.06573534, -0.86110232,  1.62274124],
[-0.99594166,  1.12815215,  0.65666521, -0.82293289, -0.55126616],
[-0.97862094, -0.88640526, -0.56336851, -0.82293289,  0.41927286],
[-0.96130021, -0.88640526,  0.7284319 , -0.82293289, -0.86183865],
[-0.94397949, -0.88640526, -1.06573534, -0.82293289,  0.5745591 ],
[-0.92665877, -0.88640526,  0.80019859, -0.78476346,  0.18634349],
[-0.90933804, -0.88640526, -0.85043527, -0.78476346, -0.12422899],
[-0.89201732, -0.88640526, -0.70690189, -0.78476346, -0.3183368 ],
[-0.87469659, -0.88640526, -0.56336851, -0.78476346, -0.3183368 ],
[-0.85737587, -0.88640526,  0.7284319 , -0.70842461,  0.06987881],
[-0.84005514,  1.12815215, -0.41983513, -0.70842461,  0.38045129],
[-0.82273442, -0.88640526, -0.56336851, -0.67025518,  0.14752193],
[-0.80541369,  1.12815215,  1.4460988 , -0.67025518,  0.38045129],
[-0.78809297, -0.88640526,  0.80019859, -0.67025518, -0.20187212],
[-0.77077224,  1.12815215,  0.58489852, -0.67025518, -0.35715836],
[-0.75345152, -0.88640526,  0.87196528, -0.63208575, -0.00776431],
[-0.73613079,  1.12815215,  2.16376569, -0.63208575, -0.16305055],
```

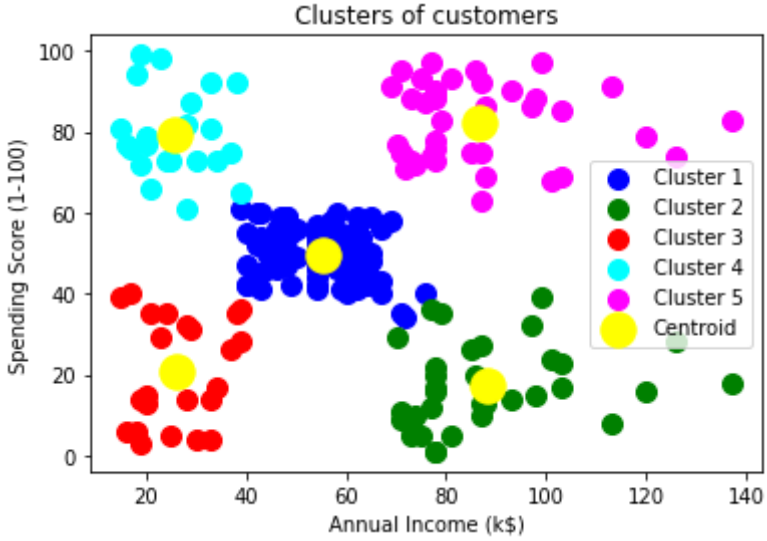
9. Perform any of the clustering algorithms 10. Add the cluster data with the primary dataset

```
x = data.iloc[:, [3, 4]].values
from sklearn.cluster import KMeans
wcss_list= []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
    kmeans.fit(x)
    wcss_list.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss_list)
plt.title('The Elobw Method Graph')
plt.xlabel('Number of clusters(k)')
plt.ylabel('wcss_list')
plt.show()
```




```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
plt.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'blue', label = 'Clus
```

```
plt.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Clu
plt.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluste
plt.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Clus
plt.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'C
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'ye
plt.title('Clusters of customers')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.legend()
plt.show()
```



11. Split the data into dependent and independent variables

```
#target variable
y=data['Age']
y.head()
```

```
0    19.0
1    21.0
2    20.0
3    23.0
4    31.0
Name: Age, dtype: float64
```

```
#independent
x=data.drop(columns=['Age'],axis=1)
x.head()
```

```
CustomerID  Gender  Annual Income (k$)  Spending Score (1-100)
```

```
data=pd.get_dummies(data,columns=['Age'])
data.head()
```

	CustomerID	Gender	Annual Income (k$)	Spending Score (1-100)	Age_18.0	Age_19.0	Age_20.0	Age_21.0	Age_
0	1	1	15	39	0	1	0	0	
1	2	1	15	81	0	0	0	1	
2	3	0	16	6	0	0	1	0	
3	4	0	16	77	0	0	0	0	
4	5	0	17	40	0	0	0	0	

5 rows × 55 columns

12. Split the data into training and testing

```
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,random_state=0)
x_train.shape
```

```
(160, 4)
```

```
x_test.shape
```

```
(40, 4)
```

```
y_train.shape
```

```
(160,)
```

```
y_test.shape
```

```
(40,)
```

13. Build the Model
14. Train the Model
15. Test the Model
16. Measure the performance using Evaluation Metrics.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
x_train.shape
```

```
(160, 4)
```

```
x_test.shape
```

```
(40, 4)
```

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

```
model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
LogisticRegression()
```

```
train_pred = model.predict(x_train)
train_pred
```

```
array([57., 35., 60., 32., 34., 35., 53., 32., 32., 32., 32., 32., 32.,
       32., 32., 53., 53., 36., 60., 35., 32., 32., 32., 21., 32., 32.,
       35., 39., 35., 32., 32., 32., 32., 32., 34., 32., 32., 53., 35.,
       29., 32., 35., 34., 32., 32., 35., 53., 20., 41., 32., 35., 53.,
       47., 35., 47., 34., 32., 32., 32., 32., 57., 32., 35., 35., 32.,
       32., 32., 35., 32., 35., 32., 20., 20., 32., 21., 39., 21., 32.,
       32., 20., 21., 32., 32., 35., 32., 32., 21., 32., 34., 45., 53.,
       35., 32., 32., 39., 32., 57., 32., 32., 21., 32., 32., 35., 35.,
       35., 21., 32., 52., 60., 32., 32., 32., 21., 53., 32., 35., 32.,
       59., 53., 60., 32., 32., 35., 34., 29., 34., 53., 32., 32., 32.,
       21., 32., 32., 32., 32., 34., 32., 32., 32., 21., 32., 32., 32.,
       34., 32., 35., 35., 57., 32., 35., 32., 53., 21., 21., 32., 32.,
       37., 32., 35., 34.])
```

```
test_pred= model.predict(x_test)
test_pred
```

```
array([53., 34., 32., 32., 28., 34., 21., 32., 53., 34., 35., 32., 32.,
       34., 32., 21., 21., 34., 35., 32., 32., 29., 35., 28., 35., 29.,
       32., 32., 32., 53., 35., 20., 34., 32., 53., 32., 32., 35., 32.,
       32.])
```

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report,roc_curv
```

```
accuracy_score(y_test,test_pred)
```

```
0.025
```

```
accuracy_score(y_train,train_pred)
```

```
0.1
```

```
confusion_matrix(y_test,test_pred)
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
pd.crosstab(y_test,test_pred)
```

col_0	20.0	21.0	28.0	29.0	32.0	34.0	35.0	53.0
Age								
18.0	0	1	0	0	0	0	0	0
19.0	0	0	0	0	1	0	1	0
22.0	0	1	0	0	0	0	0	0
23.0	0	1	0	0	0	0	0	0
24.0	0	0	0	0	0	0	1	0
27.0	0	0	1	0	0	0	0	0
28.0	0	0	0	1	0	0	0	0
29.0	0	0	0	0	1	0	0	0
30.0	0	0	1	0	1	0	1	0
31.0	0	0	0	0	1	0	1	0

```
print(classification_report(y_test,test_pred))
```

```
              precision    recall  f1-score   support

        18.0       0.00      0.00      0.00         1
        19.0       0.00      0.00      0.00         2
        20.0       0.00      0.00      0.00         0
        21.0       0.00      0.00      0.00         0
        22.0       0.00      0.00      0.00         1
        23.0       0.00      0.00      0.00         1
        24.0       0.00      0.00      0.00         1
        27.0       0.00      0.00      0.00         1
        28.0       0.00      0.00      0.00         1
        29.0       0.00      0.00      0.00         1
        30.0       0.00      0.00      0.00         3
        31.0       0.00      0.00      0.00         2
        32.0       0.00      0.00      0.00         0
        34.0       0.00      0.00      0.00         0
        35.0       0.17      0.50      0.25         2
        36.0       0.00      0.00      0.00         1
        37.0       0.00      0.00      0.00         1
        38.0       0.00      0.00      0.00         1
        39.0       0.00      0.00      0.00         1
        40.0       0.00      0.00      0.00         2
        43.0       0.00      0.00      0.00         1
        44.0       0.00      0.00      0.00         1
        46.0       0.00      0.00      0.00         3
        47.0       0.00      0.00      0.00         3
        48.0       0.00      0.00      0.00         2
        49.0       0.00      0.00      0.00         1
        52.0       0.00      0.00      0.00         1
        53.0       0.00      0.00      0.00         0
        54.0       0.00      0.00      0.00         1
        57.0       0.00      0.00      0.00         1
        58.0       0.00      0.00      0.00         1
        59.0       0.00      0.00      0.00         1
        66.0       0.00      0.00      0.00         1
```

```
        70.0       0.00      0.00      0.00         1

    accuracy                           0.03        40
   macro avg       0.00      0.01      0.01        40
weighted avg       0.01      0.03      0.01        40

/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undef
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undef
  _warn_prf(average, modifier, msg_start, len(result))
```