In [34]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [127]:
```
df = pd.read_csv('online_retail_customer_churn.csv')
df
```

Out[127]:

Transaction_Amount	Num_of_Returns	Num_of_Support_Contacts	Satisfaction_Score	Last_Purchase_Days_Ago	Email_Opt_In	Promotion_Re
453.80	2	0	3	129	True	Res
22.90	2	2	3	227	False	Res
50.53	5	2	2	283	False	Res
411.83	5	3	5	226	True	
101.19	3	0	5	242	False	Unsul
...	...	...	...	...	...	
77.75	0	3	2	88	True	
34.45	6	4	4	352	False	Res
187.37	7	3	1	172	True	Unsul
483.80	1	2	5	55	False	Res
420.91	6	0	1	269	True	

In [36]:
```
df.describe()
```

Out[36]:

	Customer_ID	Age	Annual_Income	Total_Spend	Years_as_Customer	Num_of_Purchases	Average_Transaction_Amount	N
count	1000.000000	1000.000000	1000.000000	1000.00000	1000.000000	1000.000000	1000.000000	
mean	500.500000	43.267000	111.962960	5080.79265	9.727000	49.456000	266.876530	
std	288.819436	15.242311	52.844111	2862.12335	5.536346	28.543595	145.873445	
min	1.000000	18.000000	20.010000	108.94000	1.000000	1.000000	10.460000	
25%	250.750000	30.000000	67.800000	2678.67500	5.000000	25.000000	139.682500	
50%	500.500000	43.000000	114.140000	4986.19500	9.000000	49.000000	270.100000	
75%	750.250000	56.000000	158.452500	7606.47000	14.000000	74.000000	401.602500	
max	1000.000000	69.000000	199.730000	9999.64000	19.000000	99.000000	499.570000	

In [37]:
```
df.shape
```

Out[37]:
```
(1000, 15)
```

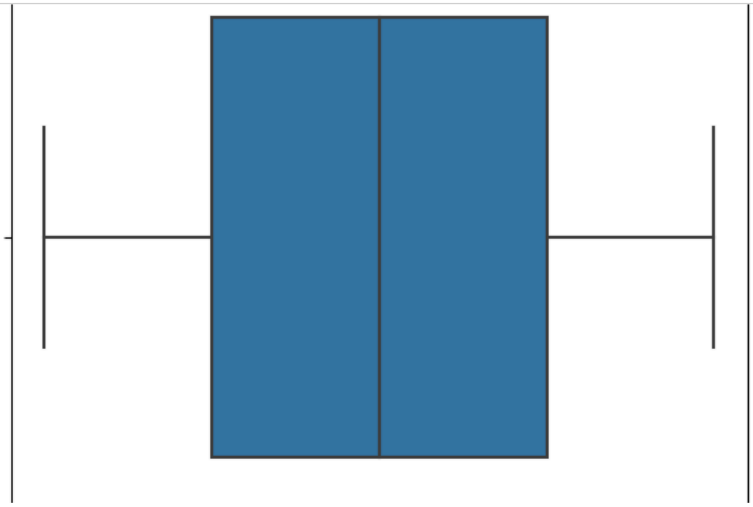
In [38]:
```
df.isnull().sum()
```

Out[38]:
```
Customer_ID                 0
Age                         0
Gender                      0
Annual_Income               0
Total_Spend                 0
Years_as_Customer           0
Num_of_Purchases            0
Average_Transaction_Amount  0
Num_of_Returns              0
Num_of_Support_Contacts     0
Satisfaction_Score          0
Last_Purchase_Days_Ago      0
Email_Opt_In                0
Promotion_Response          0
Target_Churn                0
dtype: int64
```

In [39]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 15 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Customer_ID                  1000 non-null   int64  
 1   Age                          1000 non-null   int64  
 2   Gender                       1000 non-null   object 
 3   Annual_Income                1000 non-null   float64
 4   Total_Spend                  1000 non-null   float64
 5   Years_as_Customer            1000 non-null   int64  
 6   Num_of_Purchases             1000 non-null   int64  
 7   Average_Transaction_Amount   1000 non-null   float64
 8   Num_of_Returns               1000 non-null   int64  
 9   Num_of_Support_Contacts      1000 non-null   int64  
 10  Satisfaction_Score           1000 non-null   int64  
 11  Last_Purchase_Days_Ago       1000 non-null   int64  
 12  Email_Opt_In                 1000 non-null   bool   
 13  Promotion_Response           1000 non-null   object 
 14  Target_Churn                 1000 non-null   bool   
dtypes: bool(2), float64(3), int64(8), object(2)
memory usage: 103.6+ KB
```

In [40]:
```
for i in df.select_dtypes(['int','float']):
    sns.boxplot(data=df,x=i)
    plt.show()
```

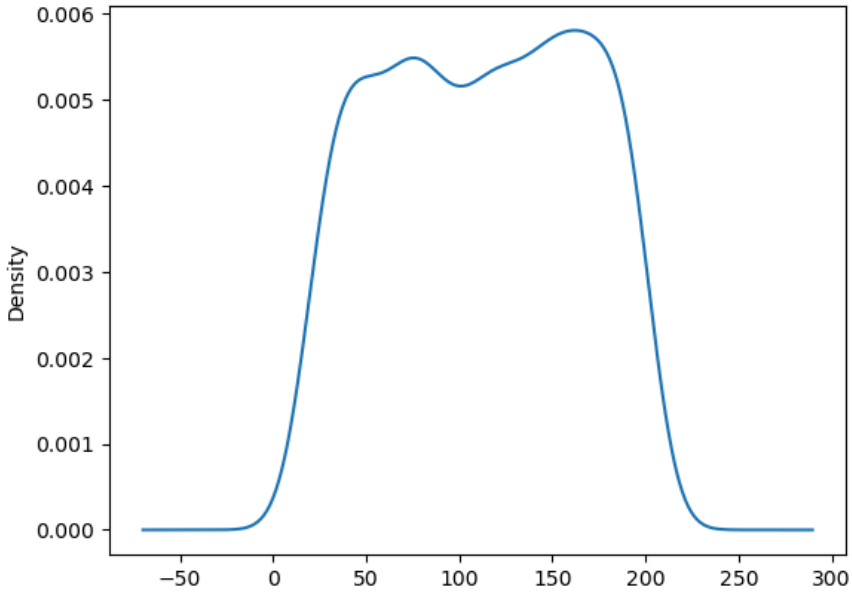


In [41]:
```
df.select_dtypes(['int','float']).skew()
```

Out[41]:
```
Customer_ID                  0.000000
Age                          0.013168
Annual_Income               -0.044687
Total_Spend                  0.017604
Years_as_Customer            0.067083
Num_of_Purchases             0.060150
Average_Transaction_Amount  -0.097007
Num_of_Returns              -0.056576
Num_of_Support_Contacts      0.082670
Satisfaction_Score           0.004082
Last_Purchase_Days_Ago       0.023327
dtype: float64
```

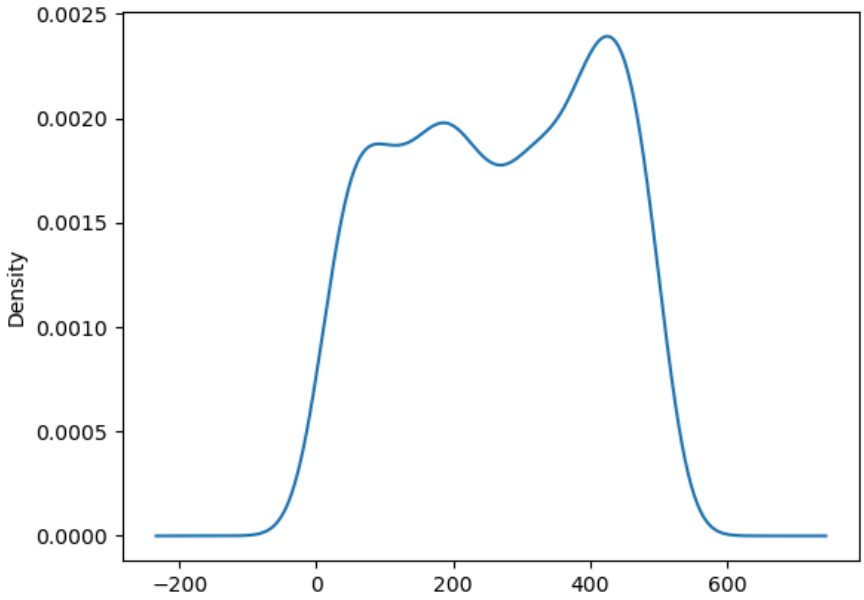
In [42]:
```
df['Annual_Income'].plot(kind = 'kde')
```

Out[42]: <Axes: ylabel='Density'>



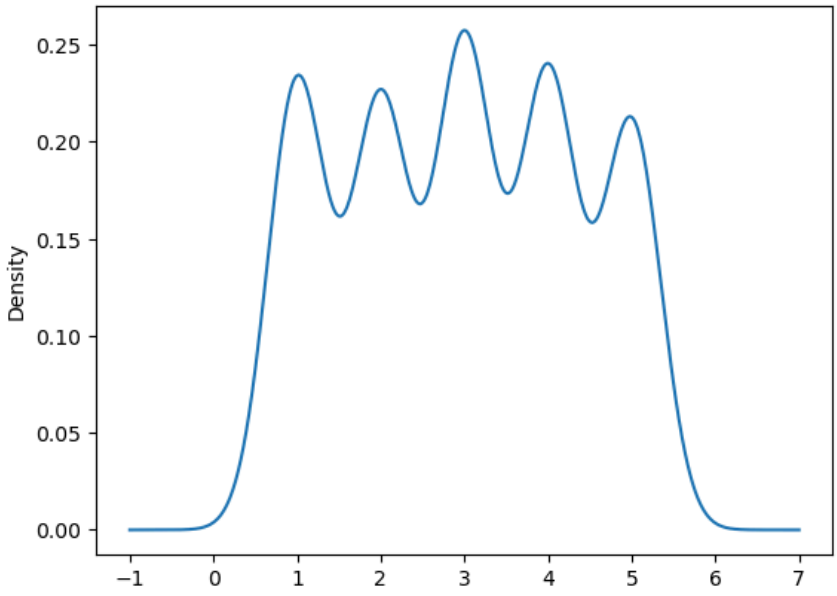
In [43]:
```
df['Average_Transaction_Amount'].plot(kind = 'kde')
```

Out[43]: <Axes: ylabel='Density'>



In [44]:
```
df['Satisfaction_Score'].plot(kind = 'kde')
```

Out[44]: <Axes: ylabel='Density'>

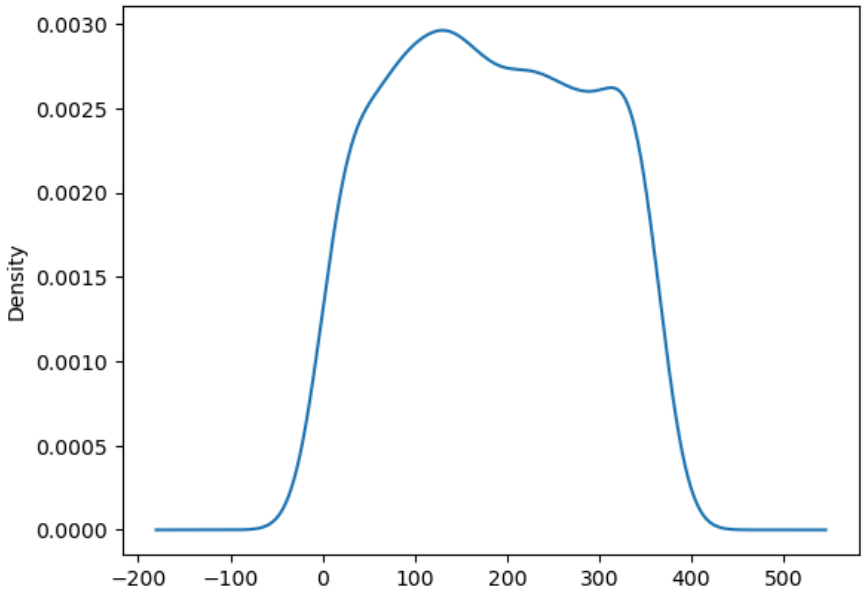


In [45]:
```
df.select_dtypes(['int','float']).kurtosis()
```

Out[45]:
```
Customer_ID                 -1.200000
Age                         -1.212970
Annual_Income               -1.228979
Total_Spend                 -1.207383
Years_as_Customer           -1.191698
Num_of_Purchases            -1.216627
Average_Transaction_Amount  -1.274298
Num_of_Returns              -1.227017
Num_of_Support_Contacts     -1.277017
Satisfaction_Score          -1.251670
Last_Purchase_Days_Ago      -1.174978
dtype: float64
```

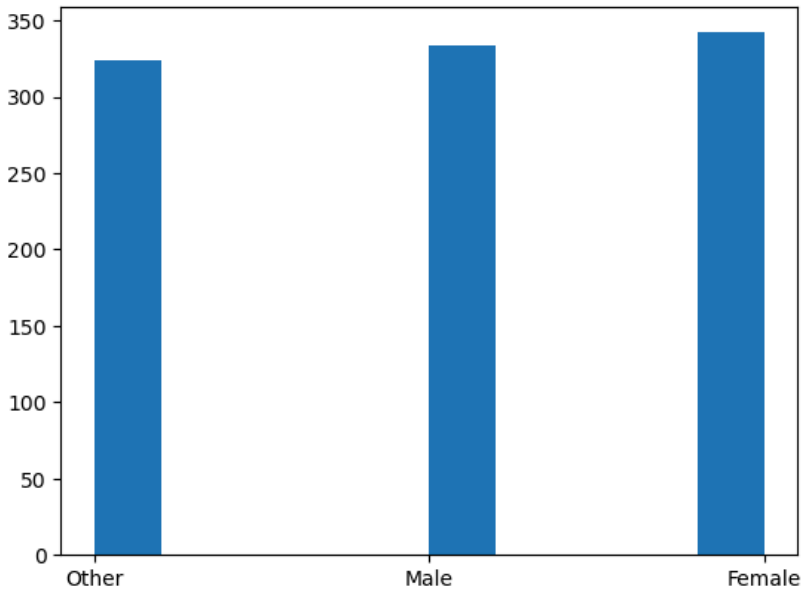
In [46]:
```
df['Last_Purchase_Days_Ago'].plot(kind = 'kde')
```

Out[46]: <Axes: ylabel='Density'>



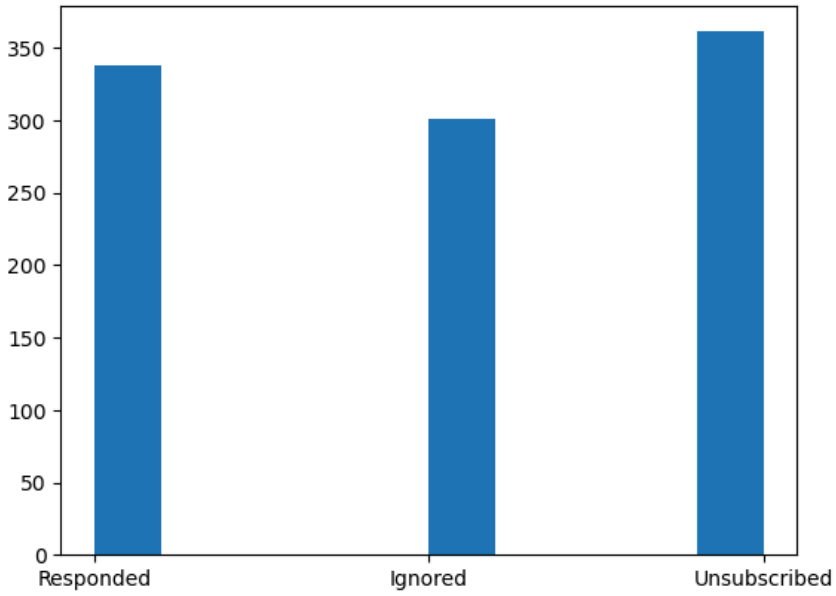
In [47]:
```
plt.hist(df["Gender"])
```

Out[47]:
```
(array([324.,   0.,   0.,   0.,   0., 334.,   0.,   0.,   0., 342.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <BarContainer object of 10 artists>)
```



In [48]:
```
plt.hist(df["Promotion_Response"])
```

Out[48]:
```
(array([338.,   0.,   0.,   0.,   0., 301.,   0.,   0.,   0., 361.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <BarContainer object of 10 artists>)
```



Logistic Regression

In [128]:
```
df
```

Out[128]:

	Customer_ID	Age	Gender	Annual_Income	Total_Spend	Years_as_Customer	Num_of_Purchases	Average_Transaction_Amount	Nu
0	1	62	Other	45.15	5892.58	5	22	453.80	
1	2	65	Male	79.51	9025.47	13	77	22.90	
2	3	18	Male	29.19	618.83	13	71	50.53	
3	4	21	Other	79.63	9110.30	3	33	411.83	
4	5	21	Other	77.66	5390.88	15	43	101.19	
...	...	...	...	...	...	...	...	...	
995	996	54	Male	143.72	1089.09	2	29	77.75	
996	997	19	Male	164.19	3700.24	9	90	34.45	
997	998	47	Female	113.31	705.85	17	69	187.37	
998	999	23	Male	72.98	3891.60	7	31	483.80	
999	1000	34	Other	134.86	3956.71	15	48	420.91	

1000 rows × 15 columns

In [254]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import warnings
warnings.filterwarnings('ignore')
```

In [255]:
```
df1 = pd.get_dummies(df,columns = ['Gender','Promotion_Response'])
df1
```

Out[255]:

	Customer_ID	Age	Annual_Income	Total_Spend	Years_as_Customer	Num_of_Purchases	Average_Transaction_Amount	Num_of_Re
0	1	62	45.15	5892.58	5	22	453.80	
1	2	65	79.51	9025.47	13	77	22.90	
2	3	18	29.19	618.83	13	71	50.53	
3	4	21	79.63	9110.30	3	33	411.83	
4	5	21	77.66	5390.88	15	43	101.19	
...	...	...	...	...	...	...	...	
995	996	54	143.72	1089.09	2	29	77.75	
996	997	19	164.19	3700.24	9	90	34.45	
997	998	47	113.31	705.85	17	69	187.37	
998	999	23	72.98	3891.60	7	31	483.80	
999	1000	34	134.86	3956.71	15	48	420.91	

1000 rows × 19 columns

In [256]:
```
X = df1.drop(['Target_Churn'],axis = 1)
y = df1['Target_Churn']
```

In [257]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

```
model = LogisticRegression()
model
```

In [258]:
```
model.fit(X_train, y_train)
```

Out[258]:

▾ LogisticRegression

LogisticRegression()

In [259]:
```
y_pred = model.predict(X_test)
y_pred
```

Out[259]:
```
array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
       False,  True, False,  True,  True,  True,  True, False,  True,
        True,  True,  True, False,  True, False, False,  True,  True,
        True,  True,  True, False,  True,  True,  True, False, False,
        True,  True,  True,  True,  True, False,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
       False, False,  True, False, False,  True,  True, False,  True,
       False,  True,  True,  True,  True, False,  True, False,  True,
       False,  True, False, False,  True, False,  True,  True,  True,
       False,  True, False, False,  True,  True,  True,  True, False,
        True, False,  True,  True,  True,  True,  True, False, False,
        True,  True, False,  True, False,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True, False,  True, False,
        True,  True, False,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True,  True,  True,  True,  True, False,
       False, False, False,  True, False, False, False,  True,  True,
        True,  True,  True, False,  True,  True,  True, False,  True,
        True,  True,  True, False,  True, False,  True, False,  True,
        True,  True,  True,  True, False,  True, False, False,  True,
       False, False,  True,  True, False,  True,  True,  True,  True,
       False,  True,  True,  True,  True, False,  True,  True,  True,
        True,  True,  True,  True,  True,  True, False,  True,  True,
       False, False])
```

In [260]:
```
y_test
```

Out[260]:
```
521     True
737     True
740     True
660     True
411    False
       ...
408    False
332    False
208    False
613    False
78     False
Name: Target_Churn, Length: 200, dtype: bool
```

In [261]:
```
accuracy_score(y_test,y_pred)
```

Out[261]:
```
0.545
```

In [264]:
```
confusion_matrix(y_test,y_pred)
```

Out[264]:
```
array([[31, 63],
       [28, 78]])
```

In [262]:
```
model.score(X_train,y_train)
```

Out[262]:
```
0.5375
```

In [263]:
```
model.score(X_test,y_test)
```

Out[263]:
```
0.545
```

Linear Regression

In [175]:
```
df
```

Out[175]:

Transaction_Amount	Num_of_Returns	Num_of_Support_Contacts	Satisfaction_Score	Last_Purchase_Days_Ago	Email_Opt_In	Promotion_Re
453.80	2	0	3	129	True	Res
22.90	2	2	3	227	False	Res
50.53	5	2	2	283	False	Res
411.83	5	3	5	226	True	
101.19	3	0	5	242	False	Unsul
...	...	...	...	...	...	
77.75	0	3	2	88	True	
34.45	6	4	4	352	False	Res
187.37	7	3	1	172	True	Unsul
483.80	1	2	5	55	False	Res
420.91	6	0	1	269	True	

In [286]:
```
from sklearn.linear_model import LinearRegression
```

In [287]:
```
from sklearn.preprocessing import LabelEncoder
```

In [288]:
```
le = LabelEncoder()
```

In [289]:
```
df1['Target_Churn']= le.fit_transform(df1['Target_Churn'])
```

In [290]:
```
df1
```

Out[290]:

	Customer_ID	Age	Annual_Income	Total_Spend	Years_as_Customer	Num_of_Purchases	Average_Transaction_Amount	Num_of_Re
0	1	62	45.15	5892.58	5	22	453.80	
1	2	65	79.51	9025.47	13	77	22.90	
2	3	18	29.19	618.83	13	71	50.53	
3	4	21	79.63	9110.30	3	33	411.83	
4	5	21	77.66	5390.88	15	43	101.19	
...	...	...	...	...	...	...	...	
995	996	54	143.72	1089.09	2	29	77.75	
996	997	19	164.19	3700.24	9	90	34.45	
997	998	47	113.31	705.85	17	69	187.37	
998	999	23	72.98	3891.60	7	31	483.80	
999	1000	34	134.86	3956.71	15	48	420.91	

1000 rows × 19 columns

In [291]:
```
x = df1.drop('Target_Churn',axis=1)


y = df1['Target_Churn']
```

In [292]:
```
x
```

Out[292]:

	Customer_ID	Age	Annual_Income	Total_Spend	Years_as_Customer	Num_of_Purchases	Average_Transaction_Amount	Num_of_Re
0	1	62	45.15	5892.58	5	22	453.80	
1	2	65	79.51	9025.47	13	77	22.90	
2	3	18	29.19	618.83	13	71	50.53	
3	4	21	79.63	9110.30	3	33	411.83	
4	5	21	77.66	5390.88	15	43	101.19	
...	...	...	...	...	...	...	...	
995	996	54	143.72	1089.09	2	29	77.75	
996	997	19	164.19	3700.24	9	90	34.45	
997	998	47	113.31	705.85	17	69	187.37	
998	999	23	72.98	3891.60	7	31	483.80	
999	1000	34	134.86	3956.71	15	48	420.91	

1000 rows × 18 columns

In [293]:
```
y
```

Out[293]:
```
0      1
1      0
2      1
3      1
4      0
      ..
995    0
996    1
997    0
998    1
999    1
Name: Target_Churn, Length: 1000, dtype: int64
```

In [294]:
```
from sklearn.model_selection import train_test_split
```

In [295]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size= 0.30,random_state=0)
```

In [296]:
```
lr=LinearRegression()
```

In [297]:
```
lr.fit(x_train,y_train)
```

Out[297]:
```
▾ LinearRegression
LinearRegression()
```

In [298]:
```
y_pred = lr.predict(x_test)

y_pred
```

Out[298]:
```
array([0.47589588, 0.57449366, 0.47671102, 0.59849231, 0.52660338,
       0.36552957, 0.44828967, 0.45581864, 0.4957535 , 0.47616082,
       0.49270534, 0.44918209, 0.4369767 , 0.51985429, 0.58112607,
       0.52270435, 0.54889595, 0.4611776 , 0.47174831, 0.4716946 ,
       0.47489979, 0.48660588, 0.49153534, 0.4516617 , 0.55098009,
       0.44164917, 0.54999091, 0.55048433, 0.53759787, 0.49559667,
       0.54749914, 0.43529441, 0.3851065 , 0.48621625, 0.50640098,
       0.44362649, 0.6062375 , 0.51963237, 0.44193439, 0.56289907,
       0.62649779, 0.42371559, 0.6119746 , 0.43911744, 0.56105472,
       0.62048637, 0.55668555, 0.45814039, 0.53128969, 0.52629143,
       0.62886754, 0.51523958, 0.43419154, 0.49262347, 0.53951524,
       0.59623906, 0.56289107, 0.57493656, 0.64761287, 0.54177858,
       0.56060122, 0.48921969, 0.43488219, 0.48993998, 0.50568293,
       0.4617335 , 0.44019857, 0.53686465, 0.49459303, 0.46997931,
       0.49497995, 0.61553957, 0.46431834, 0.55678434, 0.38591003,
       0.51705192, 0.55161874, 0.53597204, 0.52007492, 0.44696776,
       0.644336  , 0.55341865, 0.59851422, 0.54051097, 0.54521876,
       0.51230973, 0.56562135, 0.50702809, 0.46965101, 0.56571119,
       0.47900067, 0.42387528, 0.50431665, 0.51103552, 0.59904315,
       0.38501222, 0.56467901, 0.51137357, 0.43649618, 0.53340495,
       0.62429898, 0.56707366, 0.54068747, 0.48748388, 0.45990924,
       0.59536389, 0.53783974, 0.52467969, 0.49048577, 0.67051517,
       0.51594445, 0.5146918 , 0.5740159 , 0.46437843, 0.57572762,
       0.50576768, 0.46683969, 0.60963476, 0.46583152, 0.54663028,
       0.58579234, 0.3500987 , 0.53025568, 0.56443727, 0.57429178,
       0.4655763 , 0.57820187, 0.48836826, 0.65479406, 0.54395051,
       0.55967605, 0.50348687, 0.46777321, 0.57065989, 0.49884741,
       0.54593047, 0.55644189, 0.70990519, 0.39421718, 0.51607154,
       0.6269134 , 0.50396045, 0.49660918, 0.60935246, 0.55485833,
       0.56761624, 0.44687198, 0.49844504, 0.50933624, 0.50514083,
       0.52118331, 0.45699403, 0.41338503, 0.50607024, 0.53207014,
       0.49543446, 0.53514654, 0.51130888, 0.47256737, 0.46922424,
       0.58833782, 0.40981992, 0.50967817, 0.57603932, 0.45163688,
       0.44593531, 0.62924485, 0.46267325, 0.55196985, 0.56724898,
       0.47230343, 0.38681561, 0.49523124, 0.44638381, 0.58054789,
       0.60003911, 0.45235936, 0.57763272, 0.55749269, 0.57696219,
       0.49937566, 0.46379781, 0.34685654, 0.46397992, 0.60703252,
       0.4867421 , 0.50486531, 0.40925132, 0.51946846, 0.5989905 ,
       0.4591386 , 0.49516722, 0.58368114, 0.32353318, 0.62053747,
       0.59245185, 0.52311349, 0.41798634, 0.51021864, 0.50596334,
       0.56631392, 0.42117664, 0.61180828, 0.49476688, 0.62212461,
       0.4977281 , 0.55639094, 0.51575679, 0.62883105, 0.43355309,
       0.57919159, 0.5007626 , 0.59186013, 0.54030243, 0.6047451 ,
       0.58554823, 0.44636918, 0.55192959, 0.49187186, 0.52772932,
       0.44262356, 0.54087311, 0.45789179, 0.42877965, 0.39799644,
       0.43291394, 0.58446806, 0.44910585, 0.52934905, 0.43828595,
       0.48781016, 0.49032848, 0.46168065, 0.6367614 , 0.61561083,
       0.56469904, 0.5779474 , 0.46059759, 0.46395339, 0.60294668,
       0.60809608, 0.52684939, 0.58046024, 0.62392595, 0.558628  ,
       0.45521454, 0.62369218, 0.4490054 , 0.4824217 , 0.47803897,
       0.52551417, 0.53453835, 0.49632286, 0.55722057, 0.48680604,
       0.53855094, 0.50927051, 0.54826679, 0.44727278, 0.54409556,
       0.55778423, 0.49813246, 0.64695646, 0.58807226, 0.46774071,
       0.47855411, 0.47911732, 0.56105488, 0.47071948, 0.45847307,
       0.50174348, 0.59329844, 0.49406077, 0.45011253, 0.50932322,
       0.57775156, 0.42689451, 0.51707697, 0.48882346, 0.41997741,
       0.62461931, 0.49465378, 0.4774423 , 0.43315395, 0.55812967,
       0.43548927, 0.59451152, 0.46451228, 0.57493064, 0.4315745 ,
       0.51631878, 0.52630934, 0.65219925, 0.45979141, 0.52320113,
       0.50169863, 0.52178545, 0.44414319, 0.49670614, 0.55088171])
```

In [299]:
```
y_test
```

Out[299]:
```
993    0
859    1
298    1
553    0
672    1
      ..
167    1
998    1
984    0
491    1
10     0
Name: Target_Churn, Length: 300, dtype: int64
```

In [300]:
```
from sklearn.metrics import r2_score

from sklearn import metrics
```

In [301]:
```
metrics.r2_score(y_test,y_pred)
```

Out[301]:
```
-0.04178758255868886
```

In [302]:
```
lr.score(x_train,y_train)
```

Out[302]:
```
0.016946110193488684
```

In [303]:
```
lr.score(x_test,y_test)
```

Out[303]:
```
-0.04178758255868886
```

In [304]:
```
from sklearn.linear_model import Lasso
```

In [244]:
```
lasso1=Lasso(1)
lasso1
```

Out[244]:

▾ Lasso

Lasso(alpha=1)

In [245]:
```
lasso1.fit(x_train,y_train)
```

Out[245]:

▾ Lasso

Lasso(alpha=1)

In [246]:
```
y_pred1 = lasso1.predict(x_test)

y_pred1
```

Out[246]:
```
array([0.53464182, 0.55153753, 0.52628915, 0.49662945, 0.54877464,
       0.50450952, 0.49674897, 0.52337155, 0.49584509, 0.50448076,
       0.54814566, 0.50129629, 0.4897494 , 0.51392462, 0.51321707,
       0.50270294, 0.52542176, 0.53052843, 0.53884494, 0.51723128,
       0.52665481, 0.52375847, 0.50791945, 0.49811189, 0.52450892,
       0.54670405, 0.50777637, 0.5226336 , 0.51988592, 0.55563679,
       0.49533313, 0.52267864, 0.48786982, 0.48746628, 0.48686171,
       0.53345559, 0.54125438, 0.49327109, 0.50422327, 0.54795086,
       0.53439381, 0.51010866, 0.55270557, 0.50064567, 0.52211387,
       0.51488096, 0.51952139, 0.52336423, 0.54375263, 0.50851542,
       0.51908089, 0.54522431, 0.49666425, 0.50419776, 0.5440904 ,
       0.54163323, 0.52222202, 0.55766762, 0.54360428, 0.53768008,
       0.51895283, 0.51627601, 0.48838784, 0.51453939, 0.49163574,
       0.52393199, 0.51234664, 0.5142862 , 0.54721641, 0.50768202,
       0.55327297, 0.54550842, 0.49509879, 0.52869125, 0.49375763,
       0.54284225, 0.54506344, 0.54314333, 0.55243587, 0.49966253,
       0.5251955 , 0.54669208, 0.52046554, 0.55007251, 0.53942849,
       0.55529631, 0.55309525, 0.53350667, 0.51232794, 0.52907115,
       0.51848517, 0.49469423, 0.50351302, 0.49420804, 0.52791041,
       0.49693676, 0.51358555, 0.52450335, 0.48756191, 0.51998592,
       0.52276844, 0.55250245, 0.5061784 , 0.48931772, 0.54217259,
       0.55063353, 0.52022087, 0.51178654, 0.53700288, 0.54999992,
       0.50748478, 0.50105763, 0.52094363, 0.50731984, 0.52642678,
       0.54352488, 0.486606  , 0.54407158, 0.50084305, 0.5184643 ,
       0.54104426, 0.50756275, 0.50431826, 0.54047201, 0.50089517,
       0.49624834, 0.5523241 , 0.51664533, 0.54842354, 0.52883471,
       0.51300706, 0.54569382, 0.55159158, 0.53429924, 0.53359346,
       0.49397472, 0.54470976, 0.52536352, 0.50952402, 0.49539345,
       0.55760267, 0.51343665, 0.50648277, 0.56009391, 0.52272927,
       0.52673608, 0.52451863, 0.55582562, 0.49506742, 0.54124277,
       0.525013  , 0.54136917, 0.51217623, 0.50514554, 0.52651132,
       0.5175902 , 0.53289962, 0.54121183, 0.50707286, 0.50295366,
       0.5053261 , 0.51876609, 0.49300894, 0.52373623, 0.51343753,
       0.51671317, 0.53776226, 0.53598055, 0.49785329, 0.55703143,
       0.51281505, 0.49756079, 0.49265314, 0.50261573, 0.4942255 ,
       0.49006406, 0.55402249, 0.54689969, 0.52521804, 0.5580938 ,
       0.55051319, 0.50980385, 0.51658797, 0.53195772, 0.53720679,
       0.52579559, 0.54437701, 0.50046394, 0.50950311, 0.53930871,
       0.55054962, 0.54412243, 0.52006105, 0.49040816, 0.51471237,
       0.49559669, 0.49930032, 0.50524691, 0.51943388, 0.50888363,
       0.50440869, 0.49538782, 0.55035749, 0.55074493, 0.55534372,
       0.55013277, 0.49853108, 0.48686277, 0.55699499, 0.49491615,
       0.52557326, 0.55251239, 0.55096551, 0.49441626, 0.50324419,
       0.53652173, 0.49957418, 0.49028159, 0.50920792, 0.54554518,
       0.54210237, 0.52686883, 0.53748479, 0.55080781, 0.52037975,
       0.51709161, 0.52606011, 0.51191523, 0.49942113, 0.55421032,
       0.50899714, 0.49208174, 0.52031006, 0.54149147, 0.52045084,
       0.53092388, 0.49041257, 0.52747267, 0.49842687, 0.5163128 ,
       0.5207841 , 0.54937253, 0.54165357, 0.55504995, 0.52890532,
       0.50099228, 0.54444669, 0.49506366, 0.52977849, 0.48757309,
       0.49809133, 0.49242623, 0.49639482, 0.50906697, 0.54376221,
       0.51707493, 0.54155767, 0.49492144, 0.49880123, 0.52809831,
       0.49849268, 0.52875672, 0.54388238, 0.4922618 , 0.53439921,
       0.53880564, 0.53307073, 0.49343332, 0.53004398, 0.5387145 ,
       0.50827437, 0.54647078, 0.54116997, 0.53148126, 0.51261234,
       0.49431204, 0.52835068, 0.49743585, 0.52027491, 0.49397356,
       0.55165324, 0.49463129, 0.51551735, 0.49789132, 0.53784617,
       0.52104058, 0.55234946, 0.49354236, 0.53761308, 0.50780125,
       0.50738351, 0.53814762, 0.49046304, 0.5181294 , 0.52112408,
       0.50997513, 0.50899717, 0.49256217, 0.51122565, 0.52109102])
```

In [247]:
```
y_test
```

Out[247]:
```
993    0
859    1
298    1
553    0
672    1
      ..
167    1
998    1
984    0
491    1
10     0
Name: Target_Churn, Length: 300, dtype: int64
```

In [248]:
```
lasso1.score(x_train,y_train)
```

Out[248]:
```
0.001913698664254171
```

KNN Algorithm

In [265]:
```
df1
```

Out[265]:

Score	Last_Purchase_Days_Ago	Email_Opt_In	Target_Churn	Gender_Female	Gender_Male	Gender_Other	Promotion_Response_Ignored
3	129	True	True	0	0	1	0
3	227	False	False	0	1	0	0
2	283	False	True	0	1	0	0
5	226	True	True	0	0	1	1
5	242	False	False	0	0	1	0
...	...	...	...	...	...	...	...
2	88	True	False	0	1	0	1
4	352	False	True	0	1	0	0
1	172	True	False	1	0	0	0
5	55	False	True	0	1	0	0
1	269	True	True	0	0	1	1

In [266]:
```
x = df1.drop('Target_Churn',axis=1)
y = df1.Target_Churn
```

In [267]:
```
x
```

Out[267]:

	Customer_ID	Age	Annual_Income	Total_Spend	Years_as_Customer	Num_of_Purchases	Average_Transaction_Amount	Num_of_Re
0	1	62	45.15	5892.58	5	22	453.80	
1	2	65	79.51	9025.47	13	77	22.90	
2	3	18	29.19	618.83	13	71	50.53	
3	4	21	79.63	9110.30	3	33	411.83	
4	5	21	77.66	5390.88	15	43	101.19	
...	...	...	...	...	...	...	...	
995	996	54	143.72	1089.09	2	29	77.75	
996	997	19	164.19	3700.24	9	90	34.45	
997	998	47	113.31	705.85	17	69	187.37	
998	999	23	72.98	3891.60	7	31	483.80	
999	1000	34	134.86	3956.71	15	48	420.91	

1000 rows × 18 columns

In [268]:
```
y
```

Out[268]:
```
0       True
1      False
2       True
3       True
4      False
       ...
995    False
996     True
997    False
998     True
999     True
Name: Target_Churn, Length: 1000, dtype: bool
```

In [269]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.20,random_state=42)
```

In [270]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [371]:
```
knnC = KNeighborsClassifier(n_neighbors = 1)
knnC
```

Out[371]:
```
KNeighborsClassifier(n_neighbors=1)
```

In [372]:
```
knnC.fit(x_train,y_train)
```

Out[372]:
```
KNeighborsClassifier(n_neighbors=1)
```

In [373]:
```
y_pred=knnC.predict(x_test)
y_pred
```

Out[373]:
```
array([1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0])
```

In [374]:
```
y_test
```

Out[374]:
```
993    0
859    1
298    1
553    0
672    1
      ..
167    1
998    1
984    0
491    1
10     0
Name: Target_Churn, Length: 300, dtype: int64
```

In [375]:
```
knnC.score(x_train,y_train)
```

Out[375]:
```
1.0
```

In [376]:
```
knnC.score(x_test,y_test)
```

Out[376]:
```
0.46
```

In [377]:
```
accuracy_score(y_test,y_pred)
```

Out[377]:
```
0.46
```

In [320]:
```
from sklearn.neighbors import KNeighborsRegressor
```

In [378]:
```
knnr = KNeighborsRegressor(n_neighbors = 1)
knnr
```

Out[378]:
```
KNeighborsRegressor(n_neighbors=1)
```

In [379]:
```
knnr.fit(x_train,y_train)
```

Out[379]:
```
KNeighborsRegressor(n_neighbors=1)
```

In [380]:
```
y_pred=knnr.predict(x_test)
y_pred
```

Out[380]:
```
array([1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 0., 0.,
       0., 1., 0., 1., 0., 1., 1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 0.,
       0., 1., 1., 0., 0., 1., 0., 0., 1., 0., 0., 1., 1., 1., 0., 0., 1.,
       0., 0., 1., 1., 0., 1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 1., 0.,
       1., 0., 1., 1., 0., 0., 0., 0., 0., 1., 0., 1., 0., 1., 1., 1., 0.,
       0., 1., 0., 1., 1., 0., 0., 1., 0., 1., 1., 0., 1., 0., 0., 0., 0.,
       1., 1., 1., 0., 0., 0., 1., 0., 0., 1., 1., 1., 0., 1., 0., 1., 1.,
       1., 1., 1., 1., 1., 1., 1., 1., 0., 1., 0., 1., 1., 0., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 0., 0., 1., 0., 1., 1., 1.,
       1., 1., 0., 1., 0., 0., 1., 0., 0., 0., 0., 1., 0., 1., 1., 0., 1.,
       0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 0., 1.,
       0., 1., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0., 0., 1., 1.,
       1., 1., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1., 1.,
       1., 1., 0., 1., 1., 0., 1., 0., 1., 1., 0., 0., 0., 0., 1., 1., 0.,
       1., 0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 1., 1., 1., 1., 1., 0.,
       1., 1., 0., 0., 0., 0., 0., 0., 1., 0., 1., 0., 0., 1., 1., 1., 0.,
       0., 1., 0., 1., 0., 0., 1., 0., 1., 1., 1., 1., 0., 0., 1., 0., 1.,
       0., 0., 1., 1., 0., 1., 0., 1., 0., 0., 0.])
```

In [381]:
```
y_test
```

Out[381]:
```
993    0
859    1
298    1
553    0
672    1
      ..
167    1
998    1
984    0
491    1
10     0
Name: Target_Churn, Length: 300, dtype: int64
```

In [382]:
```
metrics.r2_score(y_test,y_pred)
```

Out[382]:
```
-1.1696428571428572
```

Decision Tree

In [390]:
```
df1
```

Out[390]:

Age	Annual_Income	Total_Spend	Years_as_Customer	Num_of_Purchases	Average_Transaction_Amount	Num_of_Returns	Num_of_Suppor
62	45.15	5892.58	5	22	453.80	2	
65	79.51	9025.47	13	77	22.90	2	
18	29.19	618.83	13	71	50.53	5	
21	79.63	9110.30	3	33	411.83	5	
21	77.66	5390.88	15	43	101.19	3	
...	...	...	...	...	...	...	
54	143.72	1089.09	2	29	77.75	0	
19	164.19	3700.24	9	90	34.45	6	
47	113.31	705.85	17	69	187.37	7	
23	72.98	3891.60	7	31	483.80	1	
34	134.86	3956.71	15	48	420.91	6	

mns

In [409]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.30,random_state=0)
```

In [410]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [418]:
```
dc = DecisionTreeClassifier(criterion='gini',max_depth=5,random_state=1)
```

In [419]:
```
dc.fit(x_train,y_train)
```

Out[419]:
```
DecisionTreeClassifier(max_depth=5, random_state=1)
```

In [420]:
```
y_pred = dc.predict(x_test)
y_pred
```

Out[420]:
```
array([0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1])
```

In [421]:
```
y_test
```

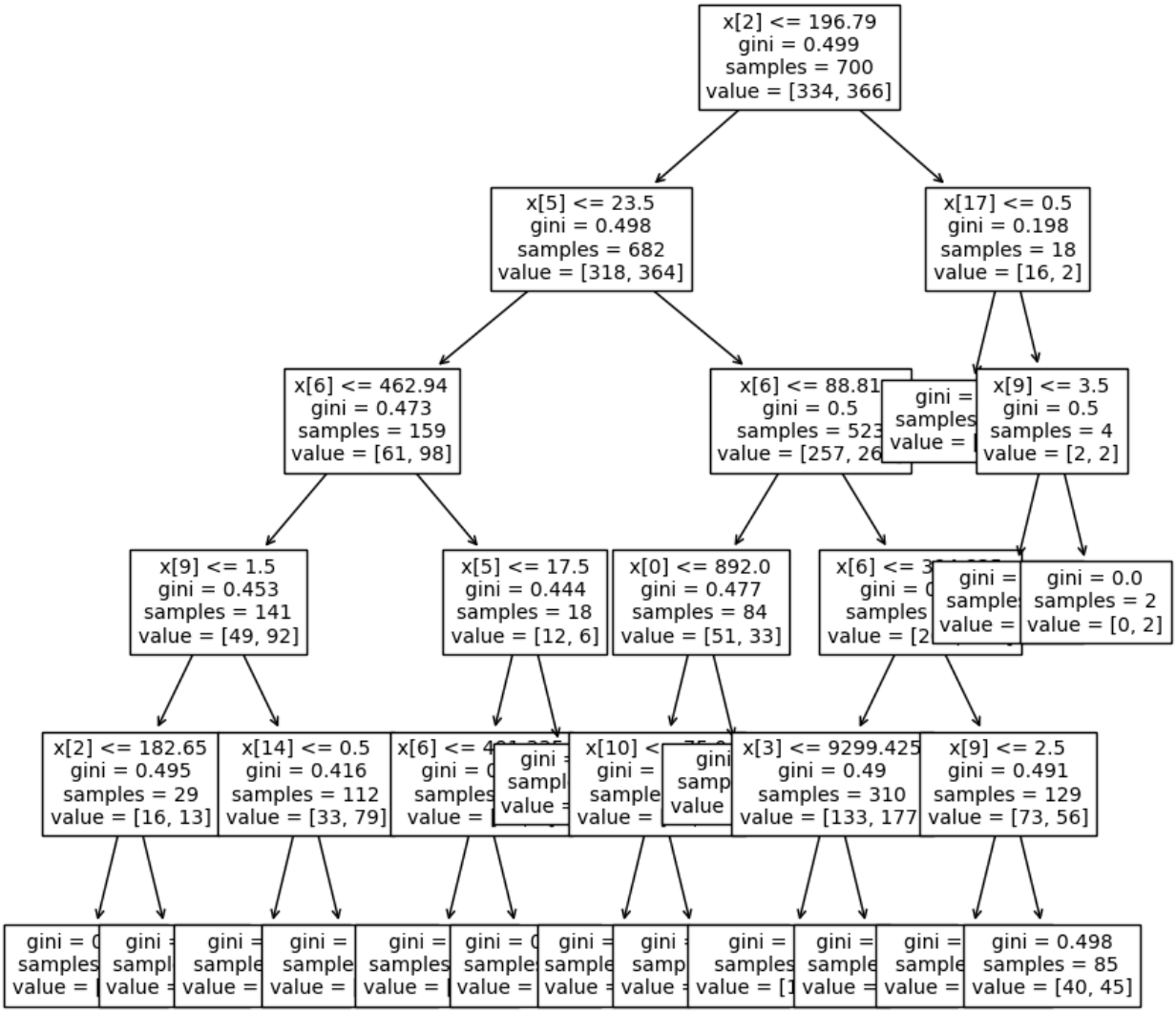
Out[421]:
```
993    0
859    1
298    1
553    0
672    1
      ..
167    1
998    1
984    0
491    1
10     0
Name: Target_Churn, Length: 300, dtype: int64
```

In [422]:
```
accuracy_score(y_test,y_pred)
```

Out[422]:
```
0.5133333333333333
```

In [423]:
```
from sklearn import tree
```

In [424]:
```
plt.figure(figsize = (10,10))
tree.plot_tree (dc,fontsize = 10)
plt.show()
```



Random Forest

In [425]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.20,random_state =43,stratify = y
```

In [426]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [494]:
```
RF_classifier = RandomForestClassifier(n_estimators= 51)
RF_classifier
```

Out[494]:
```
RandomForestClassifier
RandomForestClassifier(n_estimators=51)
```

In [495]:
```
RF_classifier.fit(x_train,y_train)
```

Out[495]:
```
RandomForestClassifier
RandomForestClassifier(n_estimators=51)
```

In [496]:
```
y_pred=RF_classifier.predict(x_test)
y_pred
```

Out[496]:
```
array([0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1])
```

In [497]:
```
y_test
```

Out[497]:
```
827    0
632    1
773    1
626    1
29     0
      ..
411    0
767    0
968    1
147    0
654    1
Name: Target_Churn, Length: 200, dtype: int64
```

In [498]:
```
accuracy_score(y_test,y_pred)
```

Out[498]:
```
0.535
```

In [499]:
```
confusion_matrix(y_pred,y_test)
```

Out[499]:
```
array([[41, 39],
       [54, 66]])
```

KMean Cluster

In [544]:
```
x = df1[['Satisfaction_Score','Total_Spend']]
```

In [545]:
```
from sklearn.cluster import KMeans
```

In [595]:
```
k_mean_algo = KMeans(n_clusters=8)
k_mean_algo
```

Out[595]:
```
KMeans
KMeans()
```

In [596]:
```
k_mean_algo.fit(X)
```

Out[596]:
```
KMeans
KMeans()
```

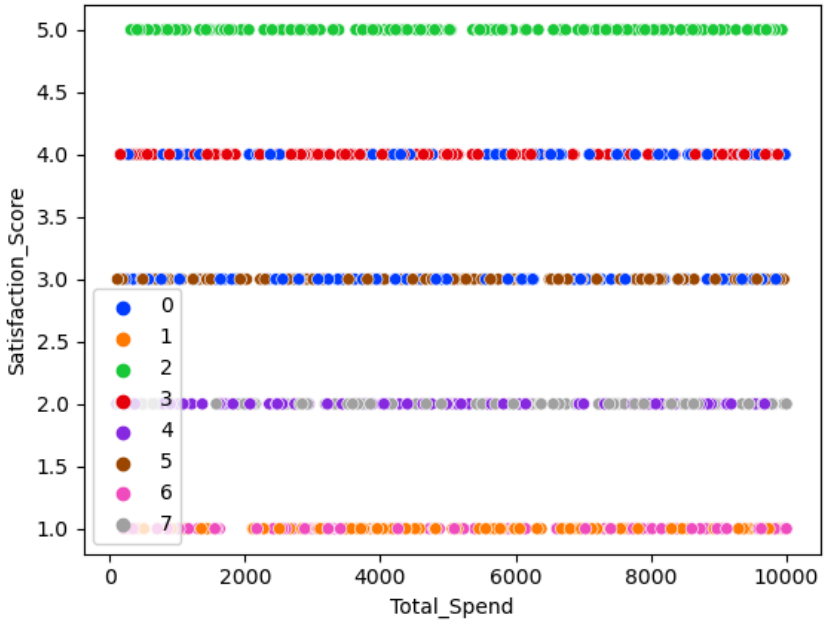
In [597]:
```
k_mean_algo.labels_
```

Out[597]:
```
array([5, 0, 7, 2, 2, 0, 6, 0, 3, 4, 4, 7, 2, 7, 1, 0, 7, 0, 0, 2, 7, 6,
       7, 7, 3, 6, 7, 6, 2, 6, 3, 1, 1, 2, 2, 5, 1, 4, 2, 2, 0, 1, 6, 1,
       3, 7, 4, 4, 5, 3, 0, 5, 6, 7, 2, 4, 3, 0, 0, 3, 3, 0, 0, 4, 0, 0,
       2, 7, 7, 6, 2, 3, 5, 2, 6, 1, 7, 5, 6, 0, 7, 2, 5, 7, 2, 1, 1, 4,
       1, 4, 0, 6, 5, 6, 5, 7, 0, 6, 2, 4, 5, 1, 6, 3, 3, 2, 7, 0, 2, 5,
       4, 4, 0, 4, 2, 7, 1, 2, 2, 4, 4, 3, 6, 0, 4, 6, 5, 3, 4, 2, 2, 4,
       5, 5, 1, 2, 5, 3, 0, 3, 2, 0, 7, 3, 4, 5, 7, 2, 2, 0, 2, 1, 3, 6,
       2, 5, 0, 2, 0, 5, 2, 2, 6, 7, 1, 3, 0, 5, 3, 5, 2, 7, 2, 2, 2, 5,
       0, 4, 2, 7, 5, 2, 7, 3, 2, 0, 0, 6, 2, 7, 7, 4, 0, 2, 6, 0, 6, 4,
       6, 0, 7, 6, 1, 0, 4, 0, 4, 5, 6, 0, 0, 4, 2, 2, 0, 6, 0, 3, 7, 2,
       7, 7, 5, 1, 4, 1, 0, 3, 6, 0, 0, 0, 2, 3, 0, 7, 1, 2, 2, 7, 1, 5,
       4, 5, 5, 0, 5, 0, 7, 2, 1, 3, 4, 5, 3, 6, 0, 7, 3, 5, 0, 5, 5, 0,
       3, 2, 1, 0, 4, 6, 1, 5, 4, 3, 0, 4, 4, 1, 4, 2, 7, 2, 0, 3, 3, 2,
       1, 0, 4, 0, 5, 7, 5, 0, 1, 4, 0, 2, 5, 2, 2, 6, 2, 4, 0, 5, 7, 2,
       4, 7, 6, 2, 4, 5, 6, 7, 0, 5, 2, 7, 2, 5, 4, 2, 2, 3, 1, 0, 5, 2,
       0, 0, 4, 3, 4, 7, 5, 5, 0, 7, 4, 3, 2, 4, 5, 4, 6, 7, 7, 6, 3, 4,
       6, 6, 1, 3, 6, 2, 5, 2, 7, 3, 0, 3, 4, 1, 4, 6, 6, 2, 1, 7, 6, 7,
       0, 2, 6, 3, 0, 7, 3, 3, 6, 0, 1, 5, 3, 5, 3, 1, 4, 0, 2, 7, 5, 3,
       3, 3, 3, 0, 1, 5, 7, 0, 0, 1, 6, 3, 0, 0, 6, 4, 5, 1, 5, 1, 0, 7,
       6, 2, 0, 1, 0, 3, 5, 1, 5, 6, 6, 3, 6, 7, 2, 4, 6, 2, 5, 5, 7, 0,
       2, 2, 3, 5, 6, 5, 1, 5, 2, 2, 6, 0, 1, 1, 0, 6, 6, 5, 7, 4, 3, 4,
       0, 6, 0, 2, 5, 4, 3, 1, 6, 4, 3, 2, 3, 3, 7, 0, 6, 3, 7, 3, 3, 1,
       6, 4, 0, 6, 4, 0, 6, 2, 4, 2, 1, 2, 0, 1, 0, 4, 2, 5, 2, 1, 2, 4,
       2, 2, 3, 5, 2, 3, 2, 2, 5, 2, 0, 2, 5, 2, 5, 1, 2, 3, 2, 2, 0, 2,
       6, 3, 0, 0, 0, 0, 3, 2, 1, 5, 4, 2, 2, 6, 5, 4, 0, 5, 6, 2, 6, 5,
       7, 2, 1, 0, 4, 0, 0, 7, 4, 2, 5, 6, 4, 1, 5, 5, 3, 6, 1, 2, 5, 5,
       0, 0, 2, 3, 0, 4, 0, 2, 3, 3, 2, 7, 0, 7, 5, 2, 1, 2, 3, 5, 2, 4,
       6, 4, 4, 0, 0, 5, 2, 0, 2, 5, 3, 5, 0, 1, 6, 5, 0, 2, 1, 2, 1, 0,
       2, 0, 6, 0, 4, 2, 6, 3, 0, 4, 7, 3, 2, 0, 0, 3, 3, 0, 0, 0, 5, 0,
       0, 5, 1, 6, 4, 1, 7, 5, 3, 3, 1, 5, 2, 0, 0, 3, 1, 2, 0, 5, 4, 0,
       7, 5, 0, 2, 6, 1, 0, 2, 2, 1, 7, 0, 2, 3, 0, 1, 3, 0, 0, 4, 3, 6,
       5, 7, 1, 3, 1, 3, 6, 3, 5, 0, 1, 2, 1, 6, 6, 2, 6, 1, 5, 4, 2, 2,
       1, 2, 2, 3, 7, 2, 7, 2, 5, 4, 0, 2, 1, 2, 2, 2, 7, 2, 4, 0, 5, 5,
       5, 2, 4, 5, 3, 0, 2, 1, 0, 2, 2, 1, 5, 0, 3, 2, 5, 0, 4, 1, 0, 1,
       7, 4, 4, 5, 6, 1, 2, 0, 3, 0, 2, 4, 5, 1, 2, 6, 1, 1, 4, 0, 7, 4,
       0, 0, 2, 7, 5, 1, 5, 0, 5, 3, 2, 7, 0, 2, 2, 2, 2, 6, 1, 0, 7, 4,
       0, 7, 3, 7, 0, 6, 3, 2, 7, 3, 5, 0, 1, 3, 0, 7, 2, 5, 2, 0, 4, 6,
       3, 5, 0, 0, 1, 0, 2, 4, 6, 2, 0, 1, 3, 6, 5, 7, 2, 3, 6, 7, 4, 2,
       1, 1, 2, 7, 6, 3, 0, 6, 0, 7, 2, 0, 3, 6, 2, 2, 3, 0, 4, 6, 2, 5,
       0, 5, 3, 2, 6, 0, 7, 2, 0, 3, 1, 3, 1, 0, 0, 7, 3, 5, 0, 3, 7, 6,
       5, 5, 5, 5, 0, 6, 3, 4, 1, 2, 7, 3, 0, 0, 3, 3, 4, 2, 5, 0, 0, 0,
       3, 0, 7, 4, 1, 1, 7, 3, 0, 7, 4, 1, 2, 0, 5, 2, 1, 0, 6, 6, 0, 2,
       2, 0, 1, 1, 0, 2, 5, 0, 2, 0, 2, 6, 6, 7, 0, 5, 2, 5, 4, 3, 0, 1,
       7, 6, 2, 3, 2, 7, 1, 6, 5, 4, 4, 1, 2, 0, 1, 3, 0, 5, 1, 3, 0, 2,
       1, 2, 2, 6, 2, 4, 3, 3, 2, 3, 5, 5, 3, 4, 7, 6, 6, 5, 3, 2, 1, 0,
       7, 0, 7, 6, 5, 4, 3, 6, 2, 1], dtype=int32)
```

In [598]:
```
centroids = k_mean_algo.cluster_centers_
centroids
```

Out[598]:
```
array([[1.11022302e-16, 3.47252747e+00],
       [1.00000000e+00, 1.00000000e+00],
       [4.91803279e-01, 5.00000000e+00],
       [1.00000000e+00, 4.00000000e+00],
       [2.22044605e-16, 2.00000000e+00],
       [1.00000000e+00, 3.00000000e+00],
       [1.11022302e-16, 1.00000000e+00],
       [1.00000000e+00, 2.00000000e+00]])
```

In [599]:
```
sns.scatterplot(data=x,x='Total_Spend',y='Satisfaction_Score',hue = k_mean_algo.labels_,palette='br
plt.show()
```



In [600]:
```
from sklearn.metrics import silhouette_score
```

In [601]:
```
silhouette_score(x,k_mean_algo.labels_)
```

Out[601]:
```
-0.07479118701339353
```

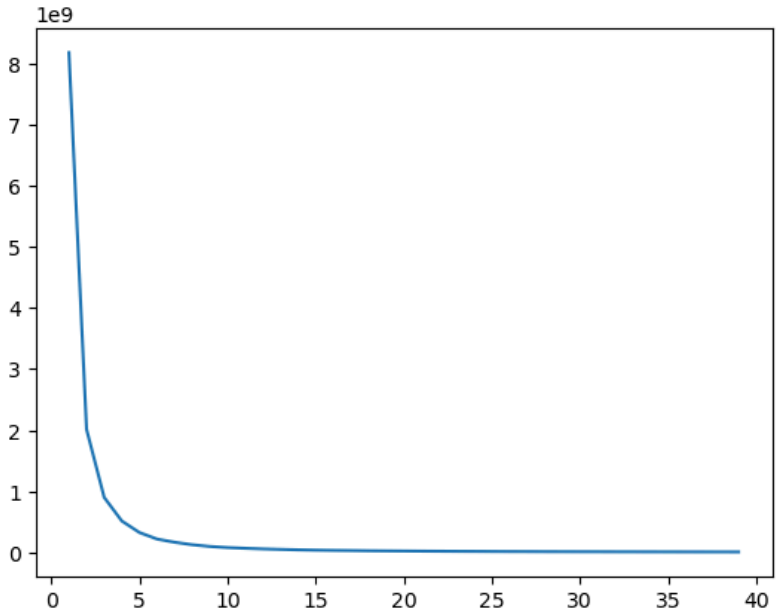
In [602]:
```
wcss = []
for i in range(1,40):
    k1 = KMeans(n_clusters =i)
    k1.fit(x)
    wcss.append(k1.inertia_)
```

In [603]:
```
wcss
```

Out[603]:
```
[8183560254.690079,
 2012667546.7588565,
 898206961.0979846,
 511666142.951803,
 322615398.3867538,
 215503493.71441016,
 163327315.69025815,
 122570789.16063347,
 94286597.58670473,
 76079902.70216402,
 64265169.467839494,
 54283349.400611214,
 46443986.19841979,
 39281189.26927517,
 33927061.97638519,
 30088584.419168554,
 27220073.14803481,
 23703408.535129912,
 21515378.19906824,
 19684362.873822525,
 17554902.28348212,
 15988813.202004304,
 14370763.012386,
 13509774.632138185,
 12149097.000864055,
 10975802.33969901,
 10470321.577828087,
 9492910.801070167,
 8895703.415116558,
 8318026.001274471,
 7887540.290504886,
 7303127.760329071,
 6956943.245081749,
 6442775.886441586,
 6091731.16792983,
 5723121.635259893,
 5451807.240624731,
 5081774.037546091,
 4920345.769676542]
```

In [605]:
```
plt.plot(range(1,40),wcss)
```

Out[605]:
```
[<matplotlib.lines.Line2D at 0x287031840>]
```



Hierarchical Clustering

In [606]:
```
df1
```

Out[606]:

	Customer_ID	Age	Annual_Income	Total_Spend	Years_as_Customer	Num_of_Purchases	Average_Transaction_Amount	Num_of_Re
0	1	62	45.15	5892.58	5	22	453.80	
1	2	65	79.51	9025.47	13	77	22.90	
2	3	18	29.19	618.83	13	71	50.53	
3	4	21	79.63	9110.30	3	33	411.83	
4	5	21	77.66	5390.88	15	43	101.19	
...	...	...	...	...	...	...	...	
995	996	54	143.72	1089.09	2	29	77.75	
996	997	19	164.19	3700.24	9	90	34.45	
997	998	47	113.31	705.85	17	69	187.37	
998	999	23	72.98	3891.60	7	31	483.80	
999	1000	34	134.86	3956.71	15	48	420.91	

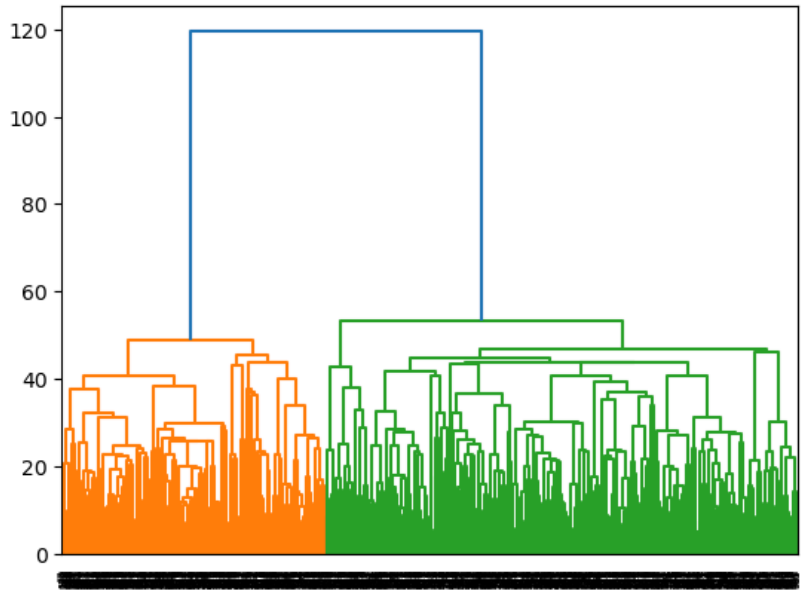
1000 rows × 19 columns

In [607]:
```
from scipy.cluster import hierarchy
```

In [608]:
```
hierarchy_algo = hierarchy.linkage(x,method ='single')
hierarchy_algo
```

Out[608]:
```
array([[2.40000000e+02, 4.25000000e+02, 1.70000000e-01, 2.00000000e+00],
       [8.46000000e+02, 9.58000000e+02, 2.40000000e-01, 2.00000000e+00],
       [5.22000000e+02, 9.69000000e+02, 3.40000000e-01, 2.00000000e+00],
       ...,
       [1.98400000e+03, 1.99300000e+03, 4.87910084e+01, 3.59000000e+02],
       [1.98600000e+03, 1.99500000e+03, 5.33193783e+01, 6.41000000e+02],
       [1.99600000e+03, 1.99700000e+03, 1.19486739e+02, 1.00000000e+03]])
```

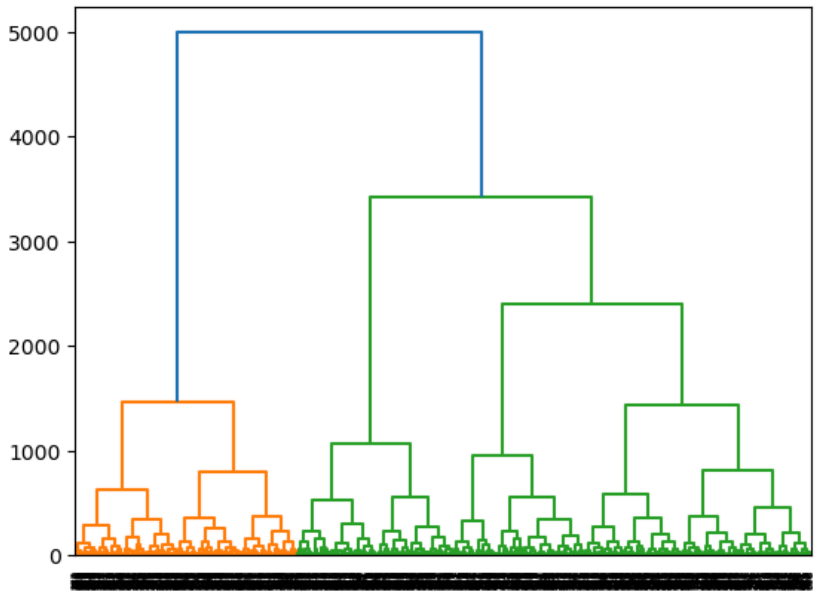
In [609]:
```
hierarchy.dendrogram(hierarchy_algo)
plt.show()
```



In [610]:
```
avg = hierarchy.linkage(x,method ='average')
avg
```

Out[610]:
```
array([[2.40000000e+02, 4.25000000e+02, 1.70000000e-01, 2.00000000e+00],
       [8.46000000e+02, 9.58000000e+02, 2.40000000e-01, 2.00000000e+00],
       [5.22000000e+02, 9.69000000e+02, 3.40000000e-01, 2.00000000e+00],
       ...,
       [1.99200000e+03, 1.99400000e+03, 2.40585337e+03, 4.84000000e+02],
       [1.99300000e+03, 1.99600000e+03, 3.41626812e+03, 6.98000000e+02],
       [1.99500000e+03, 1.99700000e+03, 4.99322951e+03, 1.00000000e+03]])
```

In [611]:
```
hierarchy.dendrogram(avg)
plt.show()
```



DBScan

In [616]:
```
from sklearn.cluster import DBSCAN
```

In [617]:
```
DBSCAN_algo = DBSCAN(eps = 9,min_samples = 3)
DBSCAN_algo
```

Out[617]:
```
DBSCAN(eps=9, min_samples=3)
```

In [618]:
```
DBSCAN_algo.fit(x)
```

Out[618]:
```
DBSCAN(eps=9, min_samples=3)
```

In [619]:
```
DBSCAN_algo.labels_
```

Out[619]:
```
array([  0,   1,  -1,   2,   3,  -1,  -1,  -1,  -1,   4,  -1, 131,  -1,
        -1,  53,  -1,  72,  -1,   5,   6,   7, 114,   2,   8,  -1, 144,
        -1,   9,   4,  -1,  -1,   8,  10,  11,  -1,  -1,  12,  -1,   3,
        13,  14,  17,  15,  16,  17,  18,  19,  74,  12,  -1,  20,  21,
        22,  -1,  -1,  23,  -1,  -1,  24,  -1,  25,  26,  27,  -1,  28,
        -1,  -1,  -1,  59, 142,  -1,  -1,  29,  -1,  30,  31,  16,  68,
        10,  32,  -1,  -1,  33,  -1,  -1,  -1,  -1,  -1,  -1,  -1,  34,
        -1,  -1,  31,   5,  35,  -1,  18,  25,  -1,  36,  -1,  -1,  37,
        -1, 103,  -1,  14,  38,  -1,  -1,  -1,  39,  -1,  -1,  -1,  -1,
       140,  40,  41,  -1,  42,  -1,  86,  61,  43,  43,  -1,  -1,  -1,
        44,  -1,  -1,  19,  -1,  41,  33,  -1,  -1,  87,  45,  83,  46,
        47,  48,  -1,  49,  43,  -1,   2,  -1,  50,  -1,  51,  -1,  52,
        53,  -1,  54, 119,  55,  56,   0,   9,  69,  36,  57,  58,  32,
        27,  59,  -1,  60,  -1,  -1,  61,  66,  62,  -1,  -1,  63,  -1,
        -1,  -1,  64,  83,  65,   3,  66,  -1,  -1,  21,  -1,  56,  63,
        56,  -1,  67,  -1,  -1,  -1,   7,  68,  69,  70,  -1,  71,  -1,
        72,  37,  73,  -1,  74,  75,  76,  -1,  77,  78,  81,  26,  79,
        58,  -1,  77,  20,  72,  -1,  80,  81,  42, 128,  17,  96,  14,
        58,  74,   1,  82,  83,  84,  85,  -1,  -1,  86,  74, 143,  75,
         6,  -1,  -1,  87,  35, 144,  -1,  53,  -1,  88,  -1,  19,  -1,
        15,  -1,  -1, 103,  33,  -1,  -1,  -1,  -1,  77,  89,  -1,  37,
        90,  -1,  82,  44,  -1,  17,  -1,  -1,  -1,  26,  -1,  91,  -1,
        34,  -1,  92,  -1,  93,  -1,  22,  -1,  13,  -1,  55, 137,   3,
        -1, 142,  94,  95,  -1, 130,  -1,  -1,  -1,  45,  29,  96,  79,
        97,  -1,  -1,  98,  67,  43,  99,  -1,  -1,  80,  93,  83, 111,
        -1,  56, 100,  -1,  91,  99,  91,  -1, 101,  90, 102,  -1, 103,
        -1,  -1, 104,  -1, 105, 106,  -1,  -1,  43, 101,  75,  87,  -1,
        -1,  50, 107,  -1,  62,  -1, 108,   0,  -1, 109, 110,  39,  26,
       111,  75,  28,  -1,  -1,  -1,  58,  -1, 112,  -1,   8,  56, 113,
        -1, 114, 115,  51,  -1,  -1,  -1,  93,  -1, 101,  65,  67,  -1,
        62, 108,   6,  70,  -1,  62,  14,  30, 116, 117,  -1,  -1, 145,
        33, 118,  32,  -1, 119,  46, 118,  19,  40,  31, 107,  -1, 120,
        39,  61,  -1,  -1, 121,  -1,  69, 105,  -1,  85,  -1,  -1,  79,
        -1, 122, 102,  -1,  -1,  -1, 123,  -1,  37,  77,  23,  42,  -1,
        88,  35,  40,  -1,  77,  60,  -1,  -1, 132, 124, 125,  75,  99,
        -1,  -1,  -1,  -1,  64, 126,  14,  -1,  -1,  -1,   5,  83, 118,
        -1, 121,   9,  99, 126,  29,  54, 122,  -1, 115, 143,  -1,  -1,
        -1,  30,  -1,  15,  79, 127,  -1,  16,  -1,  -1,  -1,  -1, 109,
         5,  -1,  92,  -1,  98,  -1,  -1, 128,  -1,  99, 119, 129,  57,
       130,  -1,  70,  -1,  20,  82, 123, 118,  -1,  46,  -1,   4,  57,
        -1, 111,  84,  -1,  -1,   1,  89, 131,  -1, 132, 130,  -1,  -1,
         5, 103,  -1,  18, 122,  67, 131,  73,  27, 111,  -1,  -1,  63,
        51, 115,  -1,  -1,  -1,  -1, 110,  23,  -1,  57,  73,  -1,  -1,
       106, 116, 127, 129,  95,  -1, 133,  -1,  -1, 108, 101, 119,  48,
       103,  -1,  -1,  55,  -1,  28,  22, 134,  16,  80,  -1,  -1, 100,
        -1,  11, 100,  -1,  71,  -1,  -1, 135, 136,  98,  -1,   2,  -1,
        -1,  97, 133, 137, 138,   5,  32,  96, 133,  -1, 102,  70,  -1,
        -1,  -1, 100,  -1,  95,  -1, 128,  41,  -1,   2,  -1,  -1,  27,
       124,  44,  -1,  -1, 121,  66,  82,  -1,  -1,  -1,  65,  49,  -1,
        -1,  61,  -1,  -1, 113,  79,  -1,  38,  12,  -1, 110,  76,  75,
        -1,  75,  -1, 139,  -1,  47, 140,  -1,  -1, 141,  94,  -1,  -1,
       120, 141, 121,  52,  21,  -1, 120,  57,  19,  59, 124,  -1,   1,
        93,  -1,  -1,  -1,  86, 126, 121,  -1,  -1,  71,  22,  -1,  99,
       127,  37, 103, 123,  -1, 109,  66,  -1,  -1,  18, 134,   8,  66,
        43,  56,  -1,  -1,  14,  15,  96, 137,  -1, 142, 104, 134,  -1,
        -1,  -1,  76, 103,  23,  -1,  -1, 115,  -1,  -1, 143,  -1,  -1,
        -1,  28,  17, 111, 102, 110,  81, 108, 134,  78,  62,  -1,  -1,
        -1,  47, 126, 139, 129,  28,  -1,  54,  69, 115,  -1, 110,  -1,
       135, 100, 144,  16,  26,  56,  10,  -1,  79,   8,  26,  -1,  20,
        -1,  43,  -1, 122, 145,  -1,  -1, 121, 138,  24,  -1,  97, 102,
        97,  -1, 145,  50,  -1,  -1,  91,  -1,  -1,  36,  -1,  36,  13,
        80, 117, 115,  24,  44,  -1, 107,  86,  -1,  -1, 139,  -1,  67,
       120,  -1,  -1,  16,  -1,  -1,  95, 107,  -1,  -1,  44,  -1,  70,
       146,  -1, 102, 133,  64,  -1,  -1, 147,  -1,  94,  -1,  -1,  -1,
        -1,  16, 116,  77,  45,  -1, 140,  -1, 132,  -1,  -1,  54,  -1,
        68, 145,  -1,  64, 119,  72,  -1, 133,   5,  -1, 117,  64,  -1,
        92,  85, 113,  -1,  82,  -1,  -1,  32, 110,  -1,  11,  88,  -1,
        -1,  -1, 111,  -1,  25,  64,  99,  74,  -1,  24,  -1,  95,  56,
        47,  58,  -1,  -1, 146,  -1,  83, 136, 125,  11, 121,  16, 147,
        20,  -1,   0,   9,  -1,  60,  63,  33, 147, 122, 112,  55,  -1,
        48,  78, 135,  69,  57, 108,  -1,  -1,  -1,  90,  -1,  -1,   6,
        16, 118,  75,  -1, 125,  -1,  -1,  -1,  84,  -1,  -1,  -1, 114,
         7,  -1,  -1, 112,  -1,  -1,  -1,  26,  -1, 136,  -1,   2,  -1,
        23, 141, 122,  -1, 104,  -1,  90,  65,  -1, 145,  79, 146,  -1,
        49,  -1, 119,  46,  -1,  -1,  -1,  84,  -1,  -1, 106, 115,  -1,
       105, 133,  89,  -1,  81,  -1,  -1,  34,  41,  -1,  -1,  38,  -1,
        -1,  43,  52, 128, 138,  39,  -1,  -1,  -1,  45,  36, 108])
```

In [620]:
```
silhouette_score(x,DBSCAN_algo.labels_)
```

Out[620]:
```
0.1007568411458883
```