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

```
#read file
df=pd.read_csv('/content/sample_data/Opportunity Score(6).csv')
df.head()
```

	phrase	Search Volume	Relevancy	Seller1	Seller2	Seller3	Seller4	Seller5	Seller6
0	lunchbox	9159	NaN	9.0	81	3	6	5	18
1	bread box	105	NaN	4.0	129	5	11	2	21
2	bread boxes	694	NaN	4.0	151	13	12	1	19

```
#cleaning data and extracting important one
df = df[['Search Volume','Seller1','Seller2','Seller3','Seller4','Seller5','Seller6','Sell
df = df[df != '-']
df = df[df != '>306']
df = df[df != 'N/R']
df.fillna(0, inplace = True)
df
```

	Search Volume	Seller1	Seller2	Seller3	Seller4	Seller5	Seller6	Seller7	Seller8	S
0	9159	9.0	81	3	6	5	18	4	23	
1	105	4.0	129	5	11	2	21	6	9	
2	694	4.0	151	13	12	1	19	6	11	
3	223	9.0	166	26	4	2	7	12	28	
4	2229	12.0	178	4	7	1	8	9	24	
5	50794	4.0	185	16	5	1	20	7	14	
6	516	18.0	8	3	2	1	10	21	118	
7	122	1.0	15	19	13	11	3	41	17	
8	1368	5.0	147	9	12	15	7	1	8	
9	159	1.0	91	19	8	24	5	13	23	
10	185	1.0	73	13	24	4	7	23	17	
11	522	1.0	81	9	22	4	8	16	17	
12	177	20.0	8	5	23	3	16	10	217	
13	103	11.0	9	2	5	0	10	10	6	
14	104	0.0	18	1	14	4	5	17	16	
15	100	1.0	70	10	12	38	14	8	26	
16	799	13.0	31	20	10	5	42	7	1	
17	107	20.0	14	2	43	1	35	23	6	
18	101	6.0	7	3	13	5	25	17	53	
19	115	14.0	27	12	47	6	10	29	69	
20	101	16.0	108	2	3	15	4	26	87	
21	114	1.0	58	15	22	14	7	24	97	
22	316	5.0	151	9	13	1	205	3	10	
23	105	8.0	222	4	15	14	1	21	264	
24	280	10.0	0	1	11	34	48	5	7	
25	132	7.0	6	4	19	8	57	3	76	
26	107	4.0	48	37	29	22	89	15	45	
27	3349	4.0	40	64	1	3	81	8	5	
28	110	19.0	143	62	6	7	48	28	2	
29	132	9.0	15	42	14	25	34	20	70	
30	326	9.0	212	16	10	4	39	7	64	
31	103	19.0	111	23	42	6	2	27	76	

32	3388	17.0	120	23	43	7	4	29	58
33	167	12.0	170	66	14	69	22	28	11
34	100	17.0	87	16	21	0	0	28	14
35	122	10.0	21	47	29	16	149	8	282
36	125	14.0	43	22	10	21	49	41	8
37	132	16.0	0	5	45	6	41	3	32
38	134	16.0	0	24	7	18	32	10	36
39	175	0.0	0	11	19	0	0	7	6
40	1406	3.0	120	33	19	21	80	1	78
41	347	8.0	89	13	7	11	65	1	86
42	762	2.0	94	44	4	42	110	3	17

```
#converting dataset into float
df= df.astype(float)
```

44	1228	5.0	56	19	16	45	159	15	80

```
#calculating relevancy
df['Relevancy'] = np.where((df['Seller1'] < 16) & (df['Seller1'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller2'] < 16) & (df['Seller2'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller3'] < 16) & (df['Seller3'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller4'] < 16) & (df['Seller4'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller5'] < 16) & (df['Seller5'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller6'] < 16) & (df['Seller6'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller7'] < 16) & (df['Seller7'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller8'] < 16) & (df['Seller8'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller9'] < 16) & (df['Seller9'] > 0), df['Relevancy'] + 1
df['Relevancy'] = np.where((df['Seller10'] < 16) & (df['Seller10'] > 0), df['Relevancy'] +
df
```

	Search Volume	Seller1	Seller2	Seller3	Seller4	Seller5	Seller6	Seller7	Seller8
0	9159.0	9.0	81.0	3.0	6.0	5.0	18.0	4.0	23.0
1	105.0	4.0	129.0	5.0	11.0	2.0	21.0	6.0	9.0
2	694.0	4.0	151.0	13.0	12.0	1.0	19.0	6.0	11.0
3	223.0	9.0	166.0	26.0	4.0	2.0	7.0	12.0	28.0
4	2229.0	12.0	178.0	4.0	7.0	1.0	8.0	9.0	24.0
5	50794.0	4.0	185.0	16.0	5.0	1.0	20.0	7.0	14.0
6	516.0	18.0	8.0	3.0	2.0	1.0	10.0	21.0	118.0
7	122.0	1.0	15.0	19.0	13.0	11.0	3.0	41.0	17.0
8	1368.0	5.0	147.0	9.0	12.0	15.0	7.0	1.0	8.0
9	159.0	1.0	91.0	19.0	8.0	24.0	5.0	13.0	23.0
10	185.0	1.0	73.0	13.0	24.0	4.0	7.0	23.0	17.0
11	522.0	1.0	81.0	9.0	22.0	4.0	8.0	16.0	17.0
12	177.0	20.0	8.0	5.0	23.0	3.0	16.0	10.0	217.0
13	103.0	11.0	9.0	2.0	5.0	0.0	10.0	10.0	6.0
14	104.0	0.0	18.0	1.0	14.0	4.0	5.0	17.0	16.0
15	100.0	1.0	70.0	10.0	12.0	38.0	14.0	8.0	26.0
16	799.0	13.0	31.0	20.0	10.0	5.0	42.0	7.0	1.0
17	107.0	20.0	14.0	2.0	43.0	1.0	35.0	23.0	6.0
18	101.0	6.0	7.0	3.0	13.0	5.0	25.0	17.0	53.0
19	115.0	14.0	27.0	12.0	47.0	6.0	10.0	29.0	69.0
20	101.0	16.0	108.0	2.0	3.0	15.0	4.0	26.0	87.0
21	114.0	1.0	58.0	15.0	22.0	14.0	7.0	24.0	97.0
22	316.0	5.0	151.0	9.0	13.0	1.0	205.0	3.0	10.0
23	105.0	8.0	222.0	4.0	15.0	14.0	1.0	21.0	264.0
24	280.0	10.0	0.0	1.0	11.0	34.0	48.0	5.0	7.0
25	132.0	7.0	6.0	4.0	19.0	8.0	57.0	3.0	76.0
26	107.0	4.0	48.0	37.0	29.0	22.0	89.0	15.0	45.0
27	3349.0	4.0	40.0	64.0	1.0	3.0	81.0	8.0	5.0
28	110.0	19.0	143.0	62.0	6.0	7.0	48.0	28.0	2.0
29	132.0	9.0	15.0	42.0	14.0	25.0	34.0	20.0	70.0
30	326.0	9.0	212.0	16.0	10.0	4.0	39.0	7.0	64.0
31	103.0	19.0	111.0	23.0	42.0	6.0	2.0	27.0	76.0

32	3388.0	17.0	120.0	23.0	43.0	7.0	4.0	29.0	58.0
33	167.0	12.0	170.0	66.0	14.0	69.0	22.0	28.0	11.0
34	100.0	17.0	87.0	16.0	21.0	0.0	0.0	28.0	14.0
35	122.0	10.0	21.0	47.0	29.0	16.0	149.0	8.0	282.0
36	125.0	14.0	43.0	22.0	10.0	21.0	49.0	41.0	8.0
37	132.0	16.0	0.0	5.0	45.0	6.0	41.0	3.0	32.0
38	134.0	16.0	0.0	24.0	7.0	18.0	32.0	10.0	36.0
39	175.0	0.0	0.0	11.0	19.0	0.0	0.0	7.0	6.0
40	1406.0	3.0	120.0	33.0	19.0	21.0	80.0	1.0	78.0
41	347.0	8.0	89.0	13.0	7.0	11.0	65.0	1.0	86.0
42	762.0	2.0	94.0	44.0	4.0	42.0	110.0	3.0	17.0
43	542.0	10.0	165.0	35.0	27.0	25.0	20.0	59.0	124.0
44	1228.0	5.0	56.0	19.0	16.0	45.0	159.0	15.0	80.0

```
#if relevancy < 3 drop that record
df.drop(df[(df['Relevancy'] < 3)].index,axis = 0,inplace=True)
df
```

	Search Volume	Seller1	Seller2	Seller3	Seller4	Seller5	Seller6	Seller7	Seller8
0	9159.0	9.0	81.0	3.0	6.0	5.0	18.0	4.0	23.0
1	105.0	4.0	129.0	5.0	11.0	2.0	21.0	6.0	9.0
2	694.0	4.0	151.0	13.0	12.0	1.0	19.0	6.0	11.0
3	223.0	9.0	166.0	26.0	4.0	2.0	7.0	12.0	28.0
4	2229.0	12.0	178.0	4.0	7.0	1.0	8.0	9.0	24.0
5	50794.0	4.0	185.0	16.0	5.0	1.0	20.0	7.0	14.0
6	516.0	18.0	8.0	3.0	2.0	1.0	10.0	21.0	118.0
7	122.0	1.0	15.0	19.0	13.0	11.0	3.0	41.0	17.0
8	1368.0	5.0	147.0	9.0	12.0	15.0	7.0	1.0	8.0
9	159.0	1.0	91.0	19.0	8.0	24.0	5.0	13.0	23.0
10	185.0	1.0	73.0	13.0	24.0	4.0	7.0	23.0	17.0
11	522.0	1.0	81.0	9.0	22.0	4.0	8.0	16.0	17.0
12	177.0	20.0	8.0	5.0	23.0	3.0	16.0	10.0	217.0
13	103.0	11.0	9.0	2.0	5.0	0.0	10.0	10.0	6.0
14	104.0	0.0	18.0	1.0	14.0	4.0	5.0	17.0	16.0
15	100.0	1.0	70.0	10.0	12.0	38.0	14.0	8.0	26.0
16	799.0	13.0	31.0	20.0	10.0	5.0	42.0	7.0	1.0
17	107.0	20.0	14.0	2.0	43.0	1.0	35.0	23.0	6.0
18	101.0	6.0	7.0	3.0	13.0	5.0	25.0	17.0	53.0
19	115.0	14.0	27.0	12.0	47.0	6.0	10.0	29.0	69.0
20	101.0	16.0	108.0	2.0	3.0	15.0	4.0	26.0	87.0
21	114.0	1.0	58.0	15.0	22.0	14.0	7.0	24.0	97.0
22	316.0	5.0	151.0	9.0	13.0	1.0	205.0	3.0	10.0
23	105.0	8.0	222.0	4.0	15.0	14.0	1.0	21.0	264.0
24	280.0	10.0	0.0	1.0	11.0	34.0	48.0	5.0	7.0
25	132.0	7.0	6.0	4.0	19.0	8.0	57.0	3.0	76.0
27	3349.0	4.0	40.0	64.0	1.0	3.0	81.0	8.0	5.0
28	110.0	19.0	143.0	62.0	6.0	7.0	48.0	28.0	2.0
29	132.0	9.0	15.0	42.0	14.0	25.0	34.0	20.0	70.0

```
# calculating CKWS
totalrows = len(df)

x = len(df.loc[(df['Seller1'] < 16) & (df['Seller1'] > 0)])
```

```
x = ((x/totalrows)*100)

x2 = len(df.loc[(df['Seller2'] < 16) & (df['Seller2'] > 0)])
x2 = ((x2/totalrows)*100)

x3 = len(df.loc[(df['Seller3'] < 16) & (df['Seller3'] > 0)])
x3 = ((x3/totalrows)*100)

x4 = len(df.loc[(df['Seller4'] < 16) & (df['Seller4'] > 0)])
x4 = ((x4/totalrows)*100)

x5 = len(df.loc[(df['Seller5'] < 16) & (df['Seller5'] > 0)])
x5 = ((x5/totalrows)*100)

x6 = len(df.loc[(df['Seller6'] < 16) & (df['Seller6'] > 0)])
x6 = ((x6/totalrows)*100)

x7 = len(df.loc[(df['Seller7'] < 16) & (df['Seller7'] > 0)])
x7 = ((x7/totalrows)*100)

x8 = len(df.loc[(df['Seller8'] < 16) & (df['Seller8'] > 0)])
x8 = ((x8/totalrows)*100)

x9 = len(df.loc[(df['Seller9'] < 16) & (df['Seller9'] > 0)])
x9 = ((x9/totalrows)*100)

x10 = len(df.loc[(df['Seller10'] < 16) & (df['Seller10'] > 0)])
x10 = ((x10/totalrows)*100)

df1=pd.read_csv('/content/sample_data/train_test_data.csv')
data_frame = pd.DataFrame([x, x2, x3, x4, x5, x6, x7, x8, x9, x10])
df1['CKWS'] = data_frame
```

```
# calculating CVS
totalCVS = df['Search Volume'].sum()

CVS = df.loc[(df['Seller1'] < 16) & (df['Seller1'] > 0)]
x = ((CVS['Search Volume'].sum()/totalCVS)*100)

CVS = df.loc[(df['Seller2'] < 16) & (df['Seller2'] > 0)]
x2 = ((CVS['Search Volume'].sum()/totalCVS)*100)

CVS = df.loc[(df['Seller3'] < 16) & (df['Seller3'] > 0)]
x3 = ((CVS['Search Volume'].sum()/totalCVS)*100)

CVS = df.loc[(df['Seller4'] < 16) & (df['Seller4'] > 0)]
x4 = ((CVS['Search Volume'].sum()/totalCVS)*100)

CVS = df.loc[(df['Seller5'] < 16) & (df['Seller5'] > 0)]
x5 = ((CVS['Search Volume'].sum()/totalCVS)*100)

CVS = df.loc[(df['Seller6'] < 16) & (df['Seller6'] > 0)]
x6 = ((CVS['Search Volume'].sum()/totalCVS)*100)
```

```
CVS = df.loc[(df['Seller7'] < 16) & (df['Seller7'] > 0)]
x7 = ((CVS['Search Volume'].sum()/totalCVS)*100)

CVS = df.loc[(df['Seller8'] < 16) & (df['Seller8'] > 0)]
x8 = ((CVS['Search Volume'].sum()/totalCVS)*100)

CVS = df.loc[(df['Seller9'] < 16) & (df['Seller9'] > 0)]
x9 = ((CVS['Search Volume'].sum()/totalCVS)*100)

CVS = df.loc[(df['Seller10'] < 16) & (df['Seller10'] > 0)]
x10 = ((CVS['Search Volume'].sum()/totalCVS)*100)

data_frame = pd.DataFrame([x, x2, x3, x4, x5, x6, x7, x8, x9, x10])
df1['CSV'] = data_frame
```

```
#classifying viability through loc
x = df1.sum(axis = 1)
df1.loc[x > 150, 'Viable'] = 'Low'
df1.loc[((x <= 150) & (x > 100)), 'Viable'] = 'Medium'
df1.loc[x <= 100, 'Viable'] = 'High'
df1.to_csv('/content/sample_data/train_test_data.csv')
df1
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarning: Droppi
```

	Sellers	CKWS	CSV	Viable
0	Seller1	76.086957	93.939321	Low
1	Seller2	19.565217	1.807338	High
2	Seller3	56.521739	21.242560	High
3	Seller4	63.043478	89.409506	Low
4	Seller5	63.043478	91.565756	Low
5	Seller6	39.130435	11.780898	High
6	Seller7	58.695652	91.337575	Low
7	Seller8	32.608696	71.226261	Medium
8	Seller9	45.652174	86.555433	Medium
9	Seller10	23.913043	70.689010	High

```
#read training/testing dataset
df2 = pd.read_csv('/content/sample_data/Final Output.csv')
df2
```

	Sellers	CKWS	CSV	Viable
0	Seller1	97.222222	99.572795	Low
1	Seller2	22.222222	72.683140	High
2	Seller3	38.888889	89.897568	Medium
3	Seller4	13.888889	46.128453	High
4	Seller5	66.666667	27.748920	High
...	...	...	...	...
491	Seller6	15.828292	92.487205	Medium
492	Seller7	66.842396	30.174558	High
493	Seller8	83.553085	25.134668	Medium
494	Seller9	76.824917	95.468479	Low

```
#encoding viable into Categorical_Viable
df2.loc[df2['Viable'] == 'High', 'Categorical_Viable'] = '0'
df2.loc[df2['Viable'] == 'Medium', 'Categorical_Viable'] = '1'
df2.loc[df2['Viable'] == 'Low', 'Categorical_Viable'] = '2'
df2 = df2[['CKWS','CSV','Categorical_Viable']]
df2= df2.astype(float)
df2
```

	CKWS	CSV	Categorical_Viable
0	97.222222	99.572795	2.0
1	22.222222	72.683140	0.0
2	38.888889	89.897568	1.0
3	13.888889	46.128453	0.0
4	66.666667	27.748920	0.0
...	...	...	...
491	15.828292	92.487205	1.0
492	66.842396	30.174558	0.0
493	83.553085	25.134668	1.0
494	76.824917	95.468479	2.0
495	32.644893	20.162299	0.0

496 rows × 3 columns

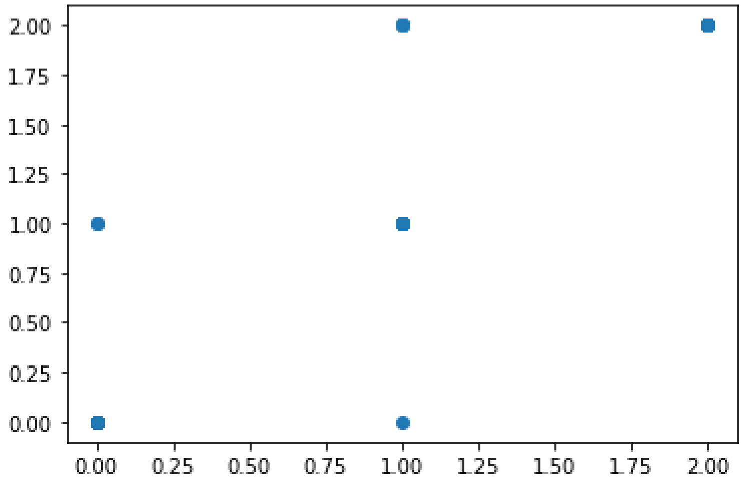
```
#train testing command/classfication import
x = df2[['CKWS','CSV']]
y = df2['Categorical_Viable']
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3)
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

```
#Apply KNN,training data,score
from sklearn.neighbors import KNeighborsClassifier
knn = (KNeighborsClassifier(n_neighbors = 30))
knn.fit(x_train,y_train)
knn.score(x_test,y_test)
```

```
0.9530201342281879
```

```
#predicting and scattering graph
predict = knn.predict(x_test)
plt.scatter(predict,y_test)
```

```
<matplotlib.collections.PathCollection at 0x7f9722b65b50>
```



```
#syntax for piechart(Testing data)
from collections import Counter
counter_object = Counter(y_test)
keyst = counter_object.keys()
valuest = counter_object.values()
keyst, valuest
```

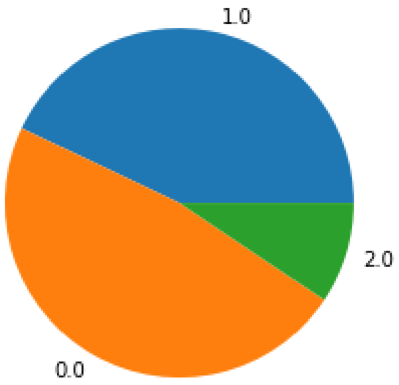
```
(dict_keys([1.0, 0.0, 2.0]), dict_values([61, 70, 18]))
```

```
#KNN testing data
plt.pie( valuest, labels=keyst )
plt.show()
```

```
#syntax for piechart(KNN)
from collections import Counter
counter_object = Counter(predict)
keysk = counter_object.keys()
valuesk = counter_object.values()
keysk, valuesk

# print(num_values)
```

```
(dict_keys([1.0, 0.0, 2.0]), dict_values([64, 71, 14]))
```

```
#KNN predict
plt.pie( valuesk, labels=keysk )
plt.show()
```



```
#classification report
print(classification_report(y_test,predict))
print(classification_report(y_train,knn.predict(x_train)))
```

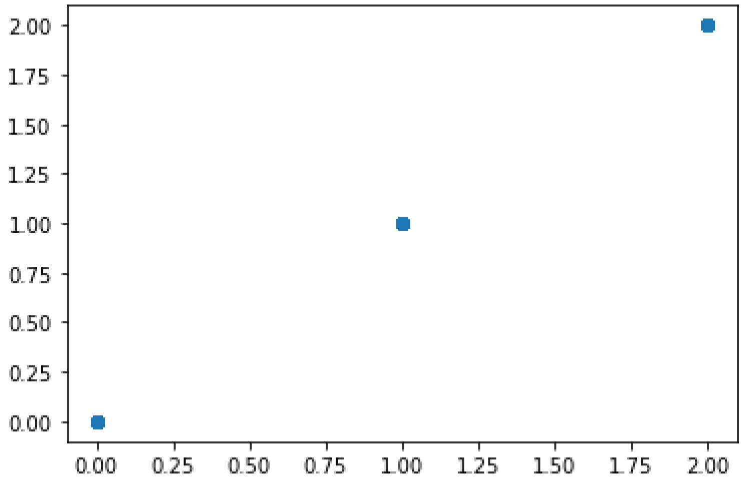
```
              precision    recall  f1-score   support

         0.0       0.97      0.99      0.98        70
         1.0       0.92      0.97      0.94        61
         2.0       1.00      0.78      0.88        18

    accuracy                           0.95       149
   macro avg       0.96      0.91      0.93       149
weighted avg       0.95      0.95      0.95       149

              precision    recall  f1-score   support

         0.0       1.00      0.97      0.98       163
         1.0       0.94      1.00      0.97       130
         2.0       1.00      0.94      0.97        54

    accuracy                           0.98       347
   macro avg       0.98      0.97      0.98       347
weighted avg       0.98      0.98      0.98       347
```

```
#Apply Logistic Regression
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
#predicting and scattering graph
predict1 = lr.predict(x_test)
plt.scatter(predict1, y_test)
```

```
<matplotlib.collections.PathCollection at 0x7f97224fc750>
```



```
#piechart syntax
from collections import Counter
counter_object = Counter(predict1)
keysl = counter_object.keys()
valuesl = counter_object.values()
keysl, valuesl

# print(num_values)
```

```
(dict_keys([1.0, 0.0, 2.0]), dict_values([61, 70, 18]))
```

```
#Logistic Regression Predict
plt.pie( valuesl, labels=keysl )
plt.show()
```

```
print(classification_report(y_test,predict1))
print(classification_report(y_train,lr.predict(x_train)))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00        70
         1.0       1.00      1.00      1.00        61
         2.0       1.00      1.00      1.00        18

    accuracy                           1.00       149
   macro avg       1.00      1.00      1.00       149
weighted avg       1.00      1.00      1.00       149

              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       163
         1.0       1.00      1.00      1.00       130
         2.0       1.00      1.00      1.00        54

    accuracy                           1.00       347
   macro avg       1.00      1.00      1.00       347
weighted avg       1.00      1.00      1.00       347
```

```
#Apply Naive Bayes

from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB().fit(x_train,y_train)
```

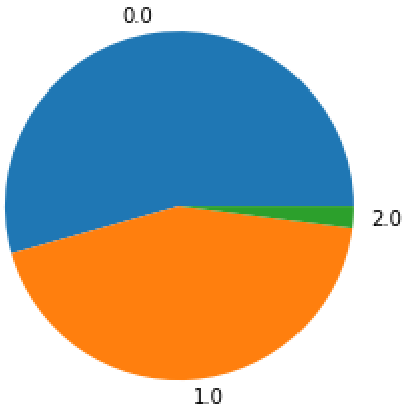
```
predict2 = classifier.predict(x_test)
print(classification_report(y_test,predict2))
```

```
              precision    recall  f1-score   support

         0.0       0.53      0.61      0.57        70
         1.0       0.43      0.46      0.44        61
         2.0       0.00      0.00      0.00        18

    accuracy                           0.48       149
   macro avg       0.32      0.36      0.34       149
weighted avg       0.43      0.48      0.45       149
```

```
from collections import Counter
counter_object = Counter(predict2)
keysn = counter_object.keys()
valuesn = counter_object.values()
keysn, valuesn

# print(num_values)
```

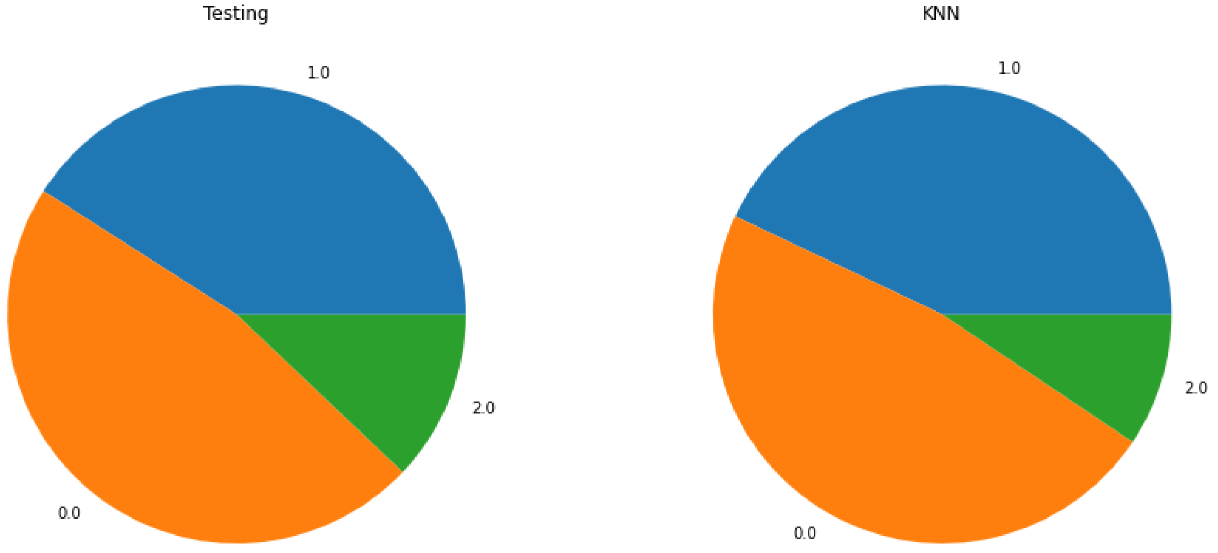
```
(dict_keys([0.0, 1.0, 2.0]), dict_values([81, 65, 3]))
```

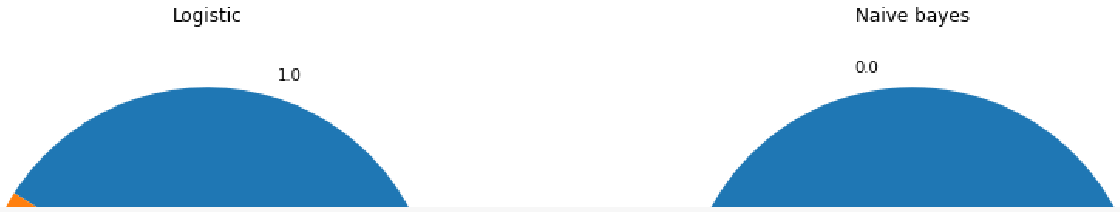
```
#Naive bayes Predict
plt.pie( valuesn, labels=keysn )
plt.show()
```



```
#comparing each pie chart
figure, axis = plt.subplots(2, 2,figsize=(15,15))

axis[0, 0].pie( valuest, labels=keyst )
axis[0, 0].set_title("Testing")

axis[0, 1].pie( valuesk, labels=keysk )
axis[0, 1].set_title("KNN")

axis[1, 0].pie( valuesl, labels=keysl )
axis[1, 0].set_title("Logistic")

axis[1, 1].pie( valuesn, labels=keysn )
axis[1, 1].set_title("Naive bayes")
plt.show
```

```
<function matplotlib.pyplot.show>
```





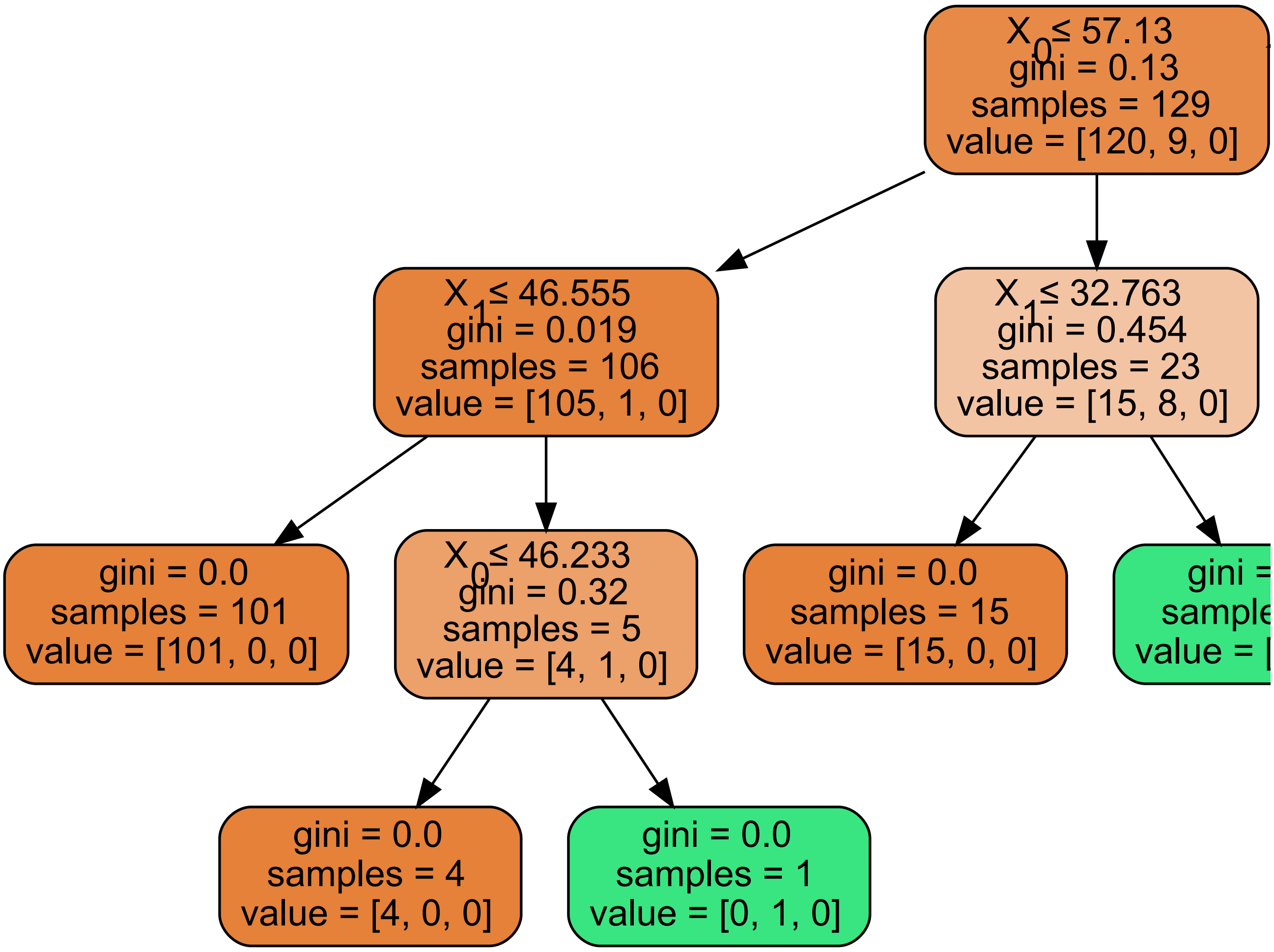
```
#training data(applying decision tree)

from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(x_train, y_train)
```

```
#predicting data
predict3=clf.predict(x_test)
```

```
#accuracy of decision tree
clf.score(x_test,y_test), clf.score(x_train,y_train)
```

```
(0.9463087248322147, 1.0)
```

```
#data visualization
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None,
                     filled=True, rounded=True,
                     special_characters=True)
graph = graphviz.Source(dot_data)
graph.view()
graph
```


X0 ≤ 57.13
gini = 0.13
samples = 129
value = [120, 9, 0]
X1 ≤ 46.555
gini = 0.019
samples = 106
value = [105, 1, 0]
X1 ≤ 32.763
gini = 0.454
samples = 23
value = [15, 8, 0]
gini = 0.0
samples = 101
value = [101, 0, 0]
X0 ≤ 46.233
gini = 0.32
samples = 5
value = [4, 1, 0]
gini = 0.0
samples = 15
value = [15, 0, 0]
gini =
sample
value = [
gini = 0.0
samples = 4
value = [4, 0, 0]
gini = 0.0
samples = 1
value = [0, 1, 0]

```
#KNN accuracy: 97%
#Logistic accuracy: 100%
#Naive Bayes accuracy: 34%
#decisiontree accuracy: 95%

print(knn.predict([[90,80]]))
print(lr.predict([[90,80]]))
print(classifier.predict([[90,80]]))
print(clf.predict([[90,80]]))
```

```
[2.]
[2.]
[0.]
[2.]
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
  "X does not have valid feature names, but"
```