In [25]:

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

In [38]:

```
df = pd.read_csv('/home/joseph/Desktop/ml lab/lab1/dataset/Melbourne_housing_FULL.c
df.head(20)
```

Out[38]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Pos
0	Abbotsford	68 Studley St	2	h	NaN	SS	Jellis	3/09/2016	2.5	
1	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	2.5	
2	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	2.5	
3	Abbotsford	18/659 Victoria St	3	u	NaN	VB	Rounds	4/02/2016	2.5	
4	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	2.5	
5	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	2.5	
6	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	2.5	
7	Abbotsford	16 Maugie St	4	h	NaN	SN	Nelson	6/08/2016	2.5	
8	Abbotsford	53 Turner St	2	h	NaN	S	Biggin	6/08/2016	2.5	
9	Abbotsford	99 Turner St	2	h	NaN	S	Collins	6/08/2016	2.5	
10	Abbotsford	129 Charles St	2	h	941000.0	S	Jellis	7/05/2016	2.5	
11	Abbotsford	124 Yarra St	3	h	1876000.0	S	Nelson	7/05/2016	2.5	
12	Abbotsford	121/56 Nicholson St	2	u	NaN	PI	Biggin	7/11/2016	2.5	
13	Abbotsford	17 Raphael St	4	h	NaN	W	Biggin	7/11/2016	2.5	
14	Abbotsford	98 Charles St	2	h	1636000.0	S	Nelson	8/10/2016	2.5	
15	Abbotsford	217 Langridge St	3	h	1000000.0	S	Jellis	8/10/2016	2.5	
16	Abbotsford	18a Mollison St	2	t	745000.0	S	Jellis	8/10/2016	2.5	
17	Abbotsford	6/241 Nicholson St	1	u	300000.0	S	Biggin	8/10/2016	2.5	
18	Abbotsford	10 Valiant St	2	h	1097000.0	S	Biggin	8/10/2016	2.5	
19	Abbotsford	403/609 Victoria St	2	u	542000.0	S	Dingle	8/10/2016	2.5	

20 rows × 21 columns

Finding Unique Values

In [117]:

```
uniqueCounts = df.nunique();
print("Unique count across columns:")
print(uniqueCounts);
```

```
Unique count across columns:
Suburb             351
Address          34009
Rooms               12
Type                 3
Price             2872
Method               9
SellerG            388
Date                78
Distance           216
Postcode           212
Bedroom2            16
Bathroom            12
Car                 16
Landsize          1685
BuildingArea       741
YearBuilt          161
CouncilArea         34
Lattitude        13403
Longtitude       14525
Regionname           9
Propertycount      343
dtype: int64
```

Finding total number of null values

In [40]:

```
df.isnull().sum()
```

Out[40]:

```
Suburb              0
Address             0
Rooms               0
Type                0
Price            7610
Method              0
SellerG             0
Date                0
Distance            1
Postcode            1
Bedroom2         8217
Bathroom         8226
Car              8728
Landsize        11810
BuildingArea    21115
YearBuilt       19306
CouncilArea         3
Lattitude        7976
Longtitude       7976
Regionname          3
Propertycount       3
dtype: int64
```

Handling missing values using mean

In [41]:

```
df['Price'].fillna(value = df.Price.mean(), inplace = True)
df['Distance'].fillna(value = df.Distance.mean(), inplace = True)
df['Postcode'].fillna(value = df.Postcode.mean(), inplace = True)
df['Bedroom2'].fillna(value = df.Bedroom2.mean(), inplace = True)
df['Bathroom'].fillna(value = df.Bathroom.mean(), inplace = True)
df['Car'].fillna(value = df.Car.mean(), inplace = True)
df['Landsize'].fillna(value = df.Landsize.mean(), inplace = True)
df['Bedroom2'].fillna(value = df.Bedroom2.mean(), inplace = True)
df['YearBuilt'].fillna(value = df.YearBuilt.mean(), inplace = True)
df['Lattitude'].fillna(value = df.Lattitude.mean(), inplace = True)
df['Longtitude'].fillna(value = df.Longtitude.mean(), inplace = True)
df['Propertycount'].fillna(value = df.Propertycount.mean(), inplace = True)
df['BuildingArea'].fillna(value = df.BuildingArea.mean(), inplace = True)
```

In [42]:

```
df.isnull().sum()
```

Out[42]:

```
Suburb           0
Address          0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Date             0
Distance         0
Postcode         0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
YearBuilt        0
CouncilArea      3
Lattitude        0
Longtitude       0
Regionname       3
Propertycount    0
dtype: int64
```

In [43]:

```
df = df.fillna(0)
```

In [118]:

```
df
```

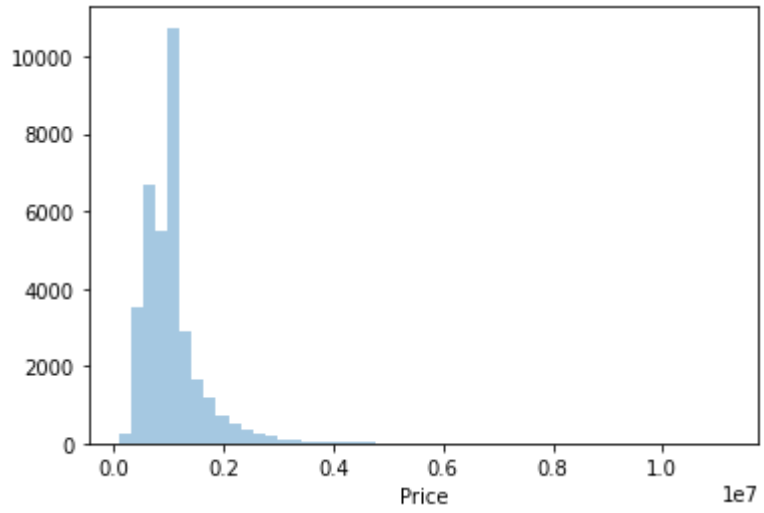
Out[118]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Postc
0	Abbotsford	68 Studley St	2	0	1.182301e-16	SS	Jellis	3/09/2016	-1.279322	30
1	Abbotsford	85 Turner St	2	0	7.579011e-01	S	Biggin	3/12/2016	-1.279322	30
2	Abbotsford	25 Bloomburg St	2	0	-2.675473e-02	S	Biggin	4/02/2016	-1.279322	30
3	Abbotsford	18/659 Victoria St	3	2	1.182301e-16	VB	Rounds	4/02/2016	-1.279322	30
4	Abbotsford	5 Charles St	3	0	7.314521e-01	SP	Biggin	4/03/2017	-1.279322	30
...	...	...	...	...	...	...	...	...	...	

Price before Scaling - Right skewed

In [47]:

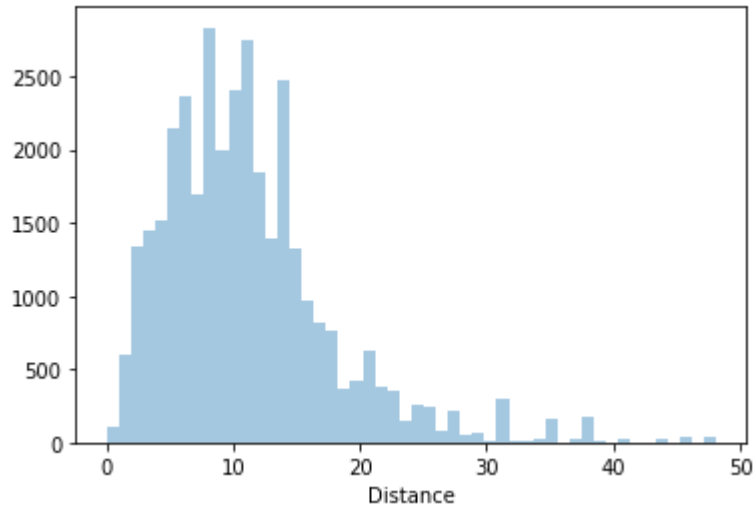
```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Price'],kde = False)
plt.show()
```



Distance before Scaling - Right skewed

In [62]:

```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Distance'],kde = False)
plt.show()
```



Standard Scaler

In [120]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

In [121]:

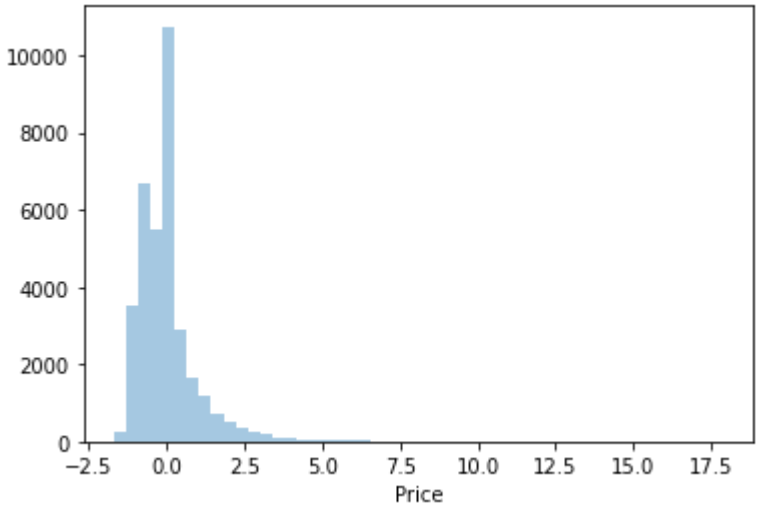
```
df[['Price','Distance','Landsize','Propertycount']] = scaler.fit_transform(df[['Pri
```

After Scaling

In [68]:

```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Price'],kde = False)
plt.show()
```

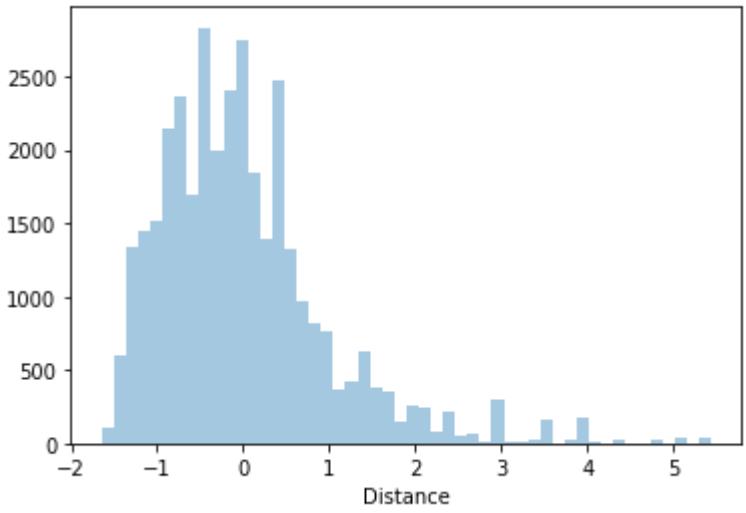
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your co
de to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [122]:

```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Distance'],kde = False)
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your co
de to use either `displot` (a figure-level function with similar flexi
bility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



In [123]:

```
df
```

Out[123]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	D
0	Abbotsford	68 Studley St	2	0	1.157840e-16	SS	Jellis	3/09/2016	-1
1	Abbotsford	85 Turner St	2	0	7.579011e-01	S	Biggin	3/12/2016	-1
2	Abbotsford	25 Bloomburg St	2	0	-2.675473e-02	S	Biggin	4/02/2016	-1
3	Abbotsford	18/659 Victoria St	3	2	1.157840e-16	VB	Rounds	4/02/2016	-1
4	Abbotsford	5 Charles St	3	0	7.314521e-01	SP	Biggin	4/03/2017	-1
...	...	...	...	...	...	...	...	...	
34852	Yarraville	13 Burns St	4	0	7.579011e-01	PI	Jas	24/02/2018	-0
34853	Yarraville	29A Murray St	2	0	-2.859557e-01	SP	Sweeney	24/02/2018	-0
34854	Yarraville	147A Severn St	2	1	-6.086344e-01	S	Jas	24/02/2018	-0
34855	Yarraville	12/37 Stephen St	3	0	1.583888e-01	SP	hockingstuart	24/02/2018	-0
34856	Yarraville	3 Tarrengower St	2	0	-5.320380e-02	PI	RW	24/02/2018	-0

34857 rows × 21 columns

KNN

In [71]:

```
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
import matplotlib.pyplot as plt
```

In [88]:

```
from sklearn import preprocessing

label_encoder = preprocessing.LabelEncoder()
df['Type']= label_encoder.fit_transform(df['Type'])
```

In [89]:

```
df
```

Out[89]:

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	D
0	Abbotsford	68 Studley St	2	0	1.182301e-16	SS	Jellis	3/09/2016	-1
1	Abbotsford	85 Turner St	2	0	7.579011e-01	S	Biggin	3/12/2016	-1
2	Abbotsford	25 Bloomburg St	2	0	-2.675473e-02	S	Biggin	4/02/2016	-1
3	Abbotsford	18/659 Victoria St	3	2	1.182301e-16	VB	Rounds	4/02/2016	-1
4	Abbotsford	5 Charles St	3	0	7.314521e-01	SP	Biggin	4/03/2017	-1
...	...	...	...	...	...	...	...	...	
34852	Yarraville	13 Burns St	4	0	7.579011e-01	PI	Jas	24/02/2018	-0
34853	Yarraville	29A Murray St	2	0	-2.859557e-01	SP	Sweeney	24/02/2018	-0
34854	Yarraville	147A Severn St	2	1	-6.086344e-01	S	Jas	24/02/2018	-0
34855	Yarraville	12/37 Stephen St	3	0	1.583888e-01	SP	hockingstuart	24/02/2018	-0
34856	Yarraville	3 Tarrengower St	2	0	-5.320380e-02	PI	RW	24/02/2018	-0

34857 rows × 21 columns

Type column is classified with the dataset

In [90]:

```
y = df ['Type'].values
X = df.drop('Address', axis = 1)
X = X.drop('CouncilArea', axis = 1)
X = X.drop('Regionname', axis = 1)
X = X.drop('Suburb', axis = 1)
X = X.drop('SellerG', axis = 1)
X = X.drop('Method', axis = 1)
X = X.drop('Type', axis = 1)

X = X.drop('Date', axis = 1)
# Separating the dependent and independent variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [73]:

```
X_train.info
```

Out[73]:

```
<bound method DataFrame.info of        Rooms         Price  Distance
Postcode  Bedroom2  Bathroom       Car  \
20967        4  8.284320e-01  0.385209    3165.0  4.000000  2.000000  2.
000000
26014        4  1.718884e+00  0.385209    3188.0  4.000000  2.000000  2.
000000
20468        4 -8.025942e-01  1.386874    3064.0  4.000000  1.000000  0.
000000
19303        2 -3.617763e-01 -0.100893    3186.0  2.000000  1.000000  1.
000000
6672         2 -1.199330e+00 -1.235131    3000.0  3.084647  1.624798  1.
728845
...        ...           ...       ...       ...       ...       ...
...
16850        3 -8.643088e-01  1.990819    3175.0  3.000000  1.000000  1.
000000
6265         2  1.182301e-16 -0.557534    3144.0  3.084647  1.624798  1.
728845
11284        4 -4.922584e-01  0.178984    3060.0  4.000000  2.000000  2.
000000
860          5  2.115620e+00 -0.218736    3103.0  5.000000  3.000000  2.
000000
15795        5  1.182301e-16  0.886042    3132.0  5.000000  2.000000  2.
000000

           Landsize  BuildingArea    YearBuilt  Lattitude  Longtitude
\
20967  1.968445e-02  1.128113e-16  1965.289885 -37.906090  145.055100
26014  2.475020e-02  7.836618e-02  1920.000000 -37.938000  145.013570
20468  1.715157e-02 -1.994774e-01  1990.000000 -37.611140  144.936830
19303 -1.786037e-01 -1.994774e-01  1960.000000 -37.895690  144.998810
6672   1.237085e-17  1.128113e-16  1965.289885 -37.810634  145.001851
...             ...           ...          ...        ...         ...
16850  7.830248e-02  1.128113e-16  1965.289885 -37.979380  145.230350
6265   1.237085e-17  1.128113e-16  1965.289885 -37.810634  145.001851
11284  1.932261e-02  1.128113e-16  1965.289885 -37.708900  144.969500
860    2.316139e-03  7.769443e-01  2006.000000 -37.806100  145.080100
15795  6.021050e-02  3.165178e-01  2010.000000 -37.825190  145.197710

       Propertycount
20967       0.766991
26014      -0.478538
20468       1.792549
19303       0.678912
6672        2.241075
...              ...
16850       0.750053
6265       -0.654471
11284      -0.565262
860        -0.427046
15795      -0.158517

[27885 rows x 13 columns]>
```

In [74]:

```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[74]:

```
((27885, 13), (6972, 13), (27885,), (6972,))
```

Minkowski Distance

In [91]:

```
K = []
training = []
test = []
scores = {}

for k in range(2, 6):
    clf = KNeighborsClassifier(n_neighbors = k)
    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    K.append(k)

    training.append(training_score)
    test.append(test_score)
    scores[k] = [training_score, test_score]
```

In [92]:

```
print("Accuracy Scores")
for keys, values in scores.items():
    print(keys, ':', values)
```

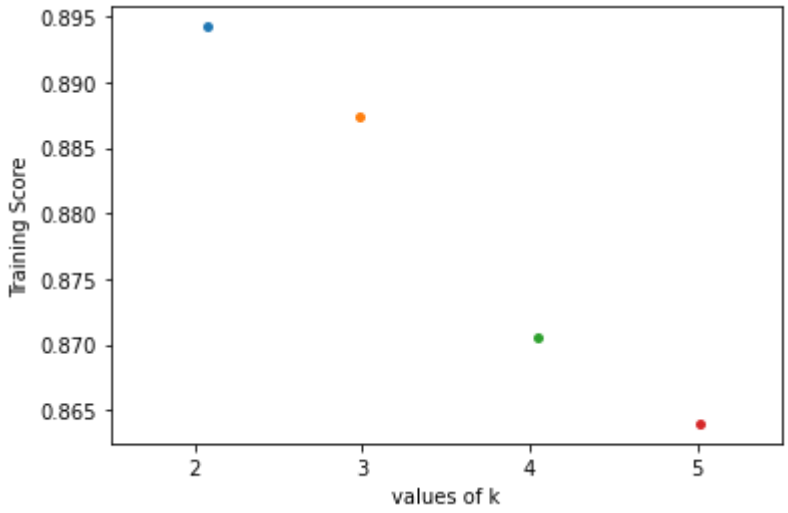
```
Accuracy Scores
2 : [0.8942442173211403, 0.7923121055651177]
3 : [0.8874305181997489, 0.8040734366035571]
4 : [0.8705038551192398, 0.8003442340791739]
5 : [0.8639770485924332, 0.8052208835341366]
```

In [93]:

```
ax = sns.stripplot(K, training);
ax.set(xlabel ='values of k', ylabel ='Training Score')

plt.show()
```

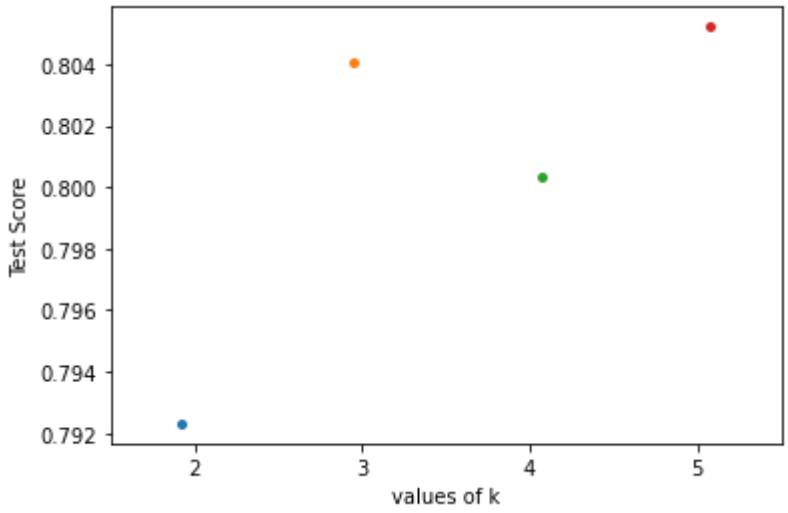
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [94]:

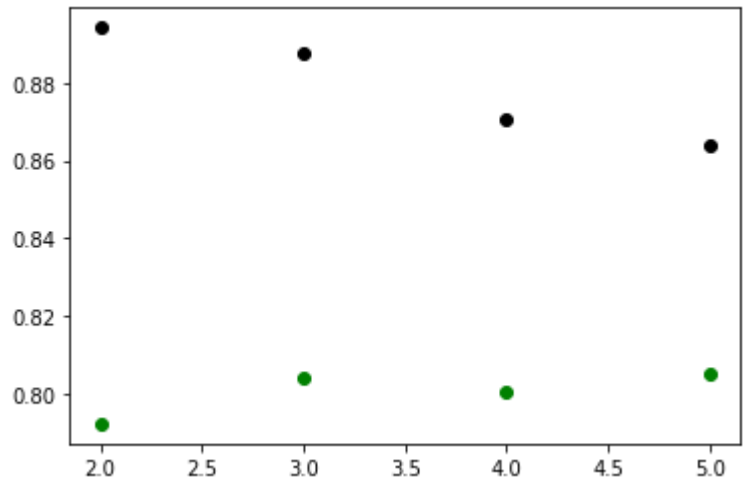
```
ax = sns.stripplot(K, test);
ax.set(xlabel ='values of k', ylabel ='Test Score')
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [95]:

```
plt.scatter(K, training, color ='k')
plt.scatter(K, test, color ='g')
plt.show()
# For overlapping scatter plots
```



In [96]:

```
y_pred = clf.predict(X_test)

from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

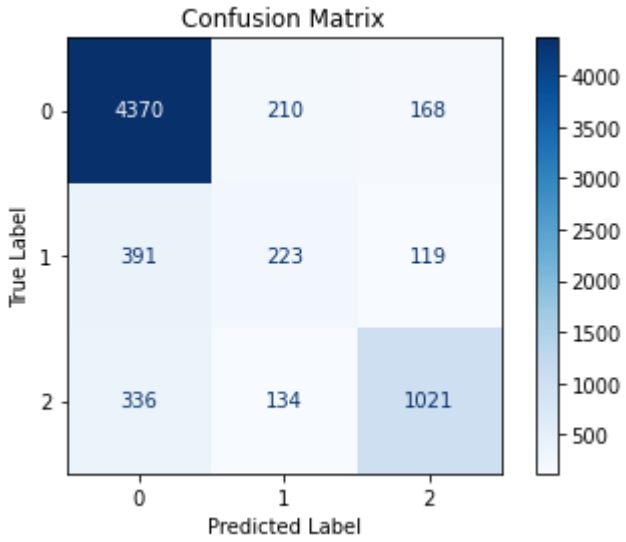
```
Accuracy: 0.8052208835341366
```

In [99]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix

color = 'black'
matrix = plot_confusion_matrix(clf, X_test, y_test, cmap=plt.cm.Blues)
matrix.ax_.set_title('Confusion Matrix', color=color)
plt.xlabel('Predicted Label', color=color)
plt.ylabel('True Label', color=color)
plt.gcf().axes[0].tick_params(colors=color)
plt.gcf().axes[1].tick_params(colors=color)
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sk
learn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_
matrix is deprecated; Function `plot_confusion_matrix` is deprecated i
n 1.0 and will be removed in 1.2. Use one of the class methods: Confus
ionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estim
ator.
  warnings.warn(msg, category=FutureWarning)
```



We can infer that

	precision	recall	f1-score	support
0	0.86	0.92	0.89	4748
1	0.39	0.30	0.34	733
2	0.78	0.68	0.73	1491
accuracy			0.81	6972

In [116]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.92      0.89      4748
           1       0.39      0.30      0.34       733
           2       0.78      0.68      0.73      1491

    accuracy                           0.81      6972
   macro avg       0.68      0.64      0.65      6972
weighted avg       0.79      0.81      0.80      6972
```

Manhattan distance

In [100]:

```
K = []
training = []
test = []
scores = {}

for k in range(2, 6):
    clf = KNeighborsClassifier(n_neighbors = k, p = 1)
    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    K.append(k)

    training.append(training_score)
    test.append(test_score)
    scores[k] = [training_score, test_score]
```

In [101]:

```
print("Accuracy Scores")
for keys, values in scores.items():
    print(keys, ':', values)
```

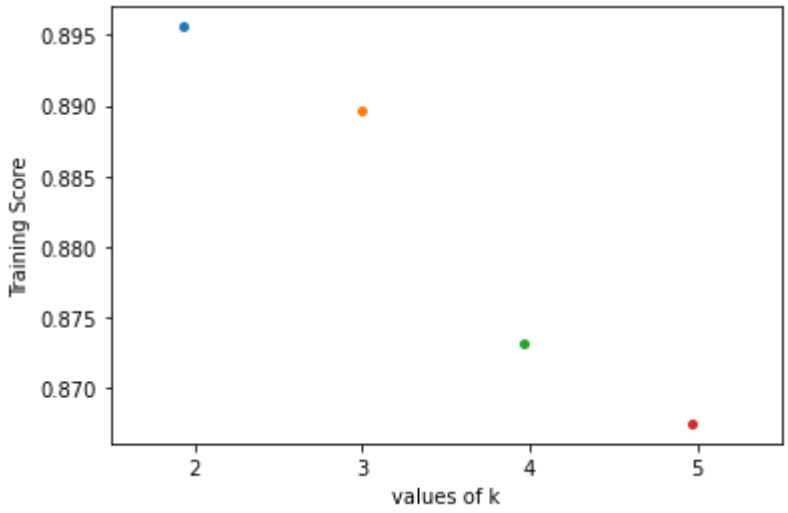
```
Accuracy Scores
2 : [0.8955710955710956, 0.7933161216293746]
3 : [0.8896897973821051, 0.8079460699942628]
4 : [0.8731576116191501, 0.8102409638554217]
5 : [0.8675273444504213, 0.8148307515777395]
```

In [102]:

```
ax = sns.stripplot(K, training);
ax.set(xlabel ='values of k', ylabel ='Training Score')

plt.show()
```

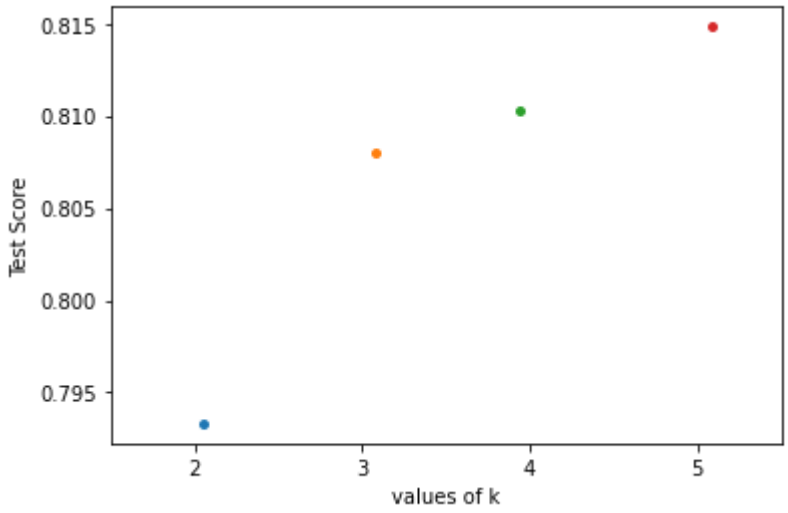
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [103]:

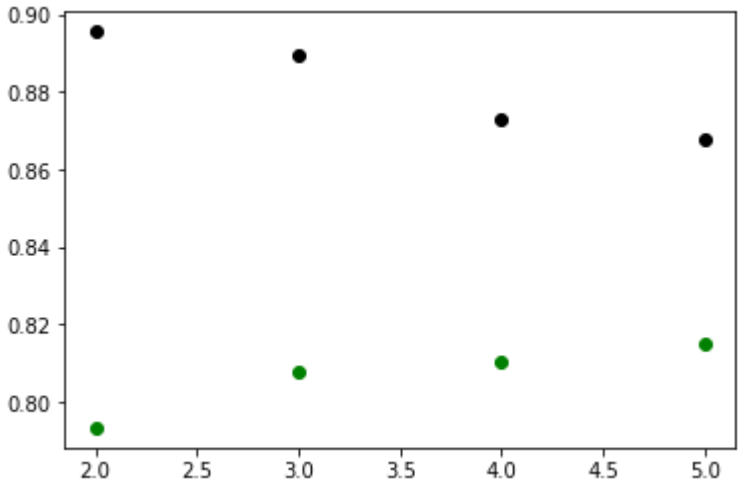
```
ax = sns.stripplot(K, test);
ax.set(xlabel ='values of k', ylabel ='Test Score')
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [104]:

```
plt.scatter(K, training, color ='k')
plt.scatter(K, test, color ='g')
plt.show()
# For overlapping scatter plots
```



In [105]:

```
y_pred = clf.predict(X_test)

from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

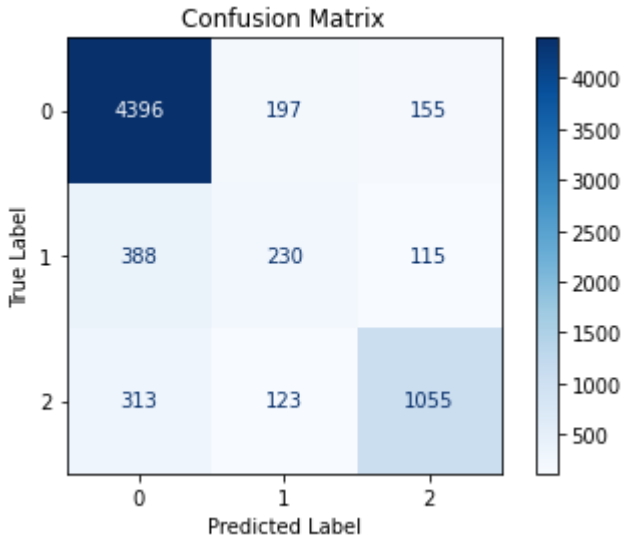
```
Accuracy: 0.8148307515777395
```

In [106]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix

color = 'black'
matrix = plot_confusion_matrix(clf, X_test, y_test, cmap=plt.cm.Blues)
matrix.ax_.set_title('Confusion Matrix', color=color)
plt.xlabel('Predicted Label', color=color)
plt.ylabel('True Label', color=color)
plt.gcf().axes[0].tick_params(colors=color)
plt.gcf().axes[1].tick_params(colors=color)
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sk
learn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_
matrix is deprecated; Function `plot_confusion_matrix` is deprecated i
n 1.0 and will be removed in 1.2. Use one of the class methods: Confus
ionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estim
ator.
  warnings.warn(msg, category=FutureWarning)
```



In [107]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.93      0.89      4748
           1       0.42      0.31      0.36       733
           2       0.80      0.71      0.75      1491

    accuracy                           0.81      6972
   macro avg       0.69      0.65      0.67      6972
weighted avg       0.80      0.81      0.81      6972
```

We can infer that

precision recall f1-score support

```
           0       0.86      0.93      0.89      4748
           1       0.42      0.31      0.36       733
           2       0.80      0.71      0.75      1491

    accuracy                           0.81      6972
```

Euclidean Distance

In [108]:

```
K = []
training = []
test = []
scores = {}

for k in range(2, 6):
    clf = KNeighborsClassifier(n_neighbors = k, p = 2)
    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    K.append(k)

    training.append(training_score)
    test.append(test_score)
    scores[k] = [training_score, test_score]
```

In [109]:

```
print("Accuracy Scores")
for keys, values in scores.items():
    print(keys, ':', values)
```

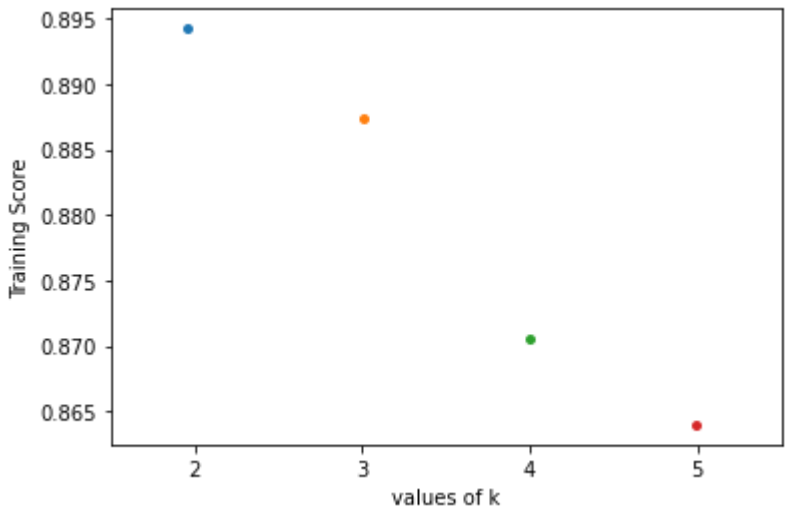
```
Accuracy Scores
2 : [0.8942442173211403, 0.7923121055651177]
3 : [0.8874305181997489, 0.8040734366035571]
4 : [0.8705038551192398, 0.8003442340791739]
5 : [0.8639770485924332, 0.8052208835341366]
```

In [110]:

```
ax = sns.stripplot(K, training);
ax.set(xlabel ='values of k', ylabel ='Training Score')

plt.show()
```

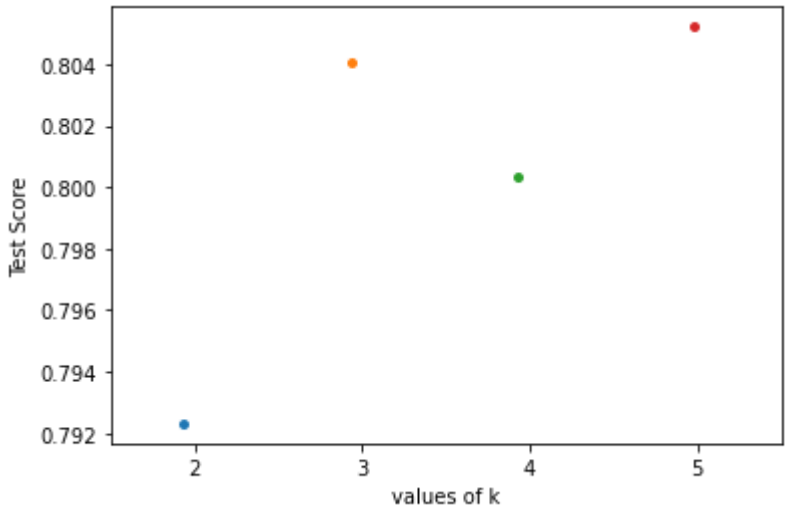
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [111]:

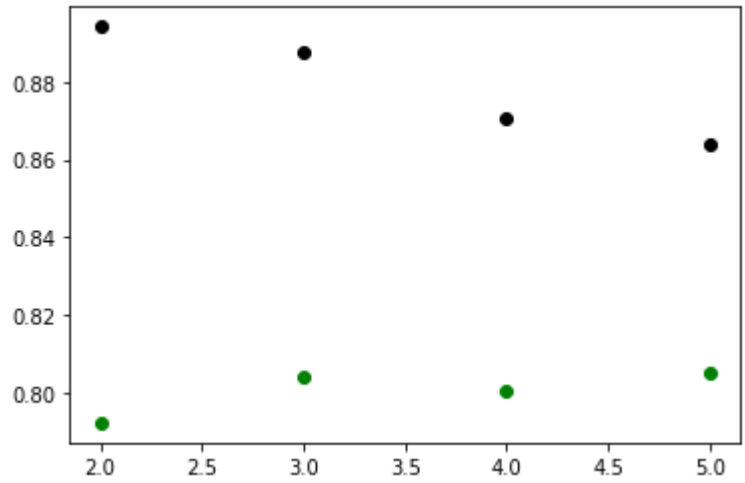
```
ax = sns.stripplot(K, test);
ax.set(xlabel ='values of k', ylabel ='Test Score')
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.
  warnings.warn(
```



In [112]:

```
plt.scatter(K, training, color ='k')
plt.scatter(K, test, color ='g')
plt.show()
# For overlapping scatter plots
```



In [113]:

```
y_pred = clf.predict(X_test)

from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

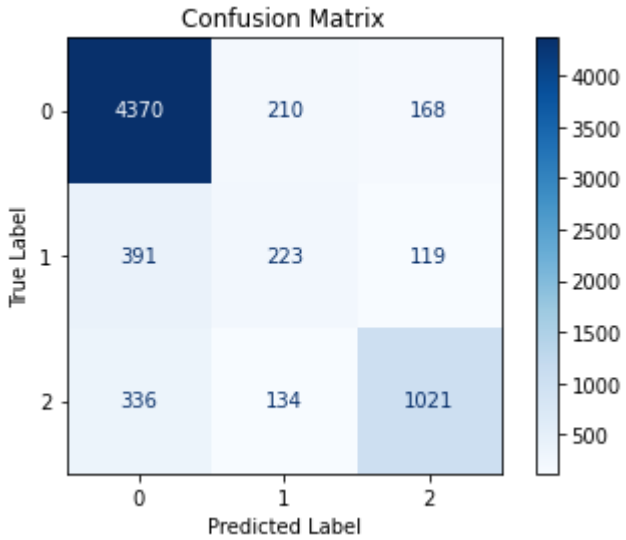
```
Accuracy: 0.8052208835341366
```

In [114]:

```
import matplotlib.pyplot as plt
from sklearn.metrics import plot_confusion_matrix

color = 'black'
matrix = plot_confusion_matrix(clf, X_test, y_test, cmap=plt.cm.Blues)
matrix.ax_.set_title('Confusion Matrix', color=color)
plt.xlabel('Predicted Label', color=color)
plt.ylabel('True Label', color=color)
plt.gcf().axes[0].tick_params(colors=color)
plt.gcf().axes[1].tick_params(colors=color)
plt.show()
```

```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sk
learn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_
matrix is deprecated; Function `plot_confusion_matrix` is deprecated i
n 1.0 and will be removed in 1.2. Use one of the class methods: Confus
ionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estim
ator.
  warnings.warn(msg, category=FutureWarning)
```



In [115]:

```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.92      0.89      4748
           1       0.39      0.30      0.34       733
           2       0.78      0.68      0.73      1491

    accuracy                           0.81      6972
   macro avg       0.68      0.64      0.65      6972
weighted avg       0.79      0.81      0.80      6972
```

We can infer that

	precision	recall	f1-score	support
0	0.86	0.92	0.89	4748
1	0.39	0.30	0.34	733
2	0.78	0.68	0.73	1491
accuracy			0.81	6972

In [ ]:

```

```