In [1]:

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

In [2]:

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

Out[2]:

	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 [3]:

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

```
Unique count across columns:
Suburb             351
Address          34009
Rooms               12
Type                 3
Price             2871
Method               9
SellerG            388
Date                78
Distance           215
Postcode           211
Bedroom2            15
Bathroom            11
Car                 15
Landsize          1684
BuildingArea       740
YearBuilt          160
CouncilArea         33
Lattitude        13402
Longtitude       14524
Regionname           8
Propertycount      342
dtype: int64
```

Finding total number of null values

In [4]:

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

Out[4]:

```
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 [5]:

```
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 [6]:

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

Out[6]:

```
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 [7]:

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

In [8]:

```
df
```

Out[8]:

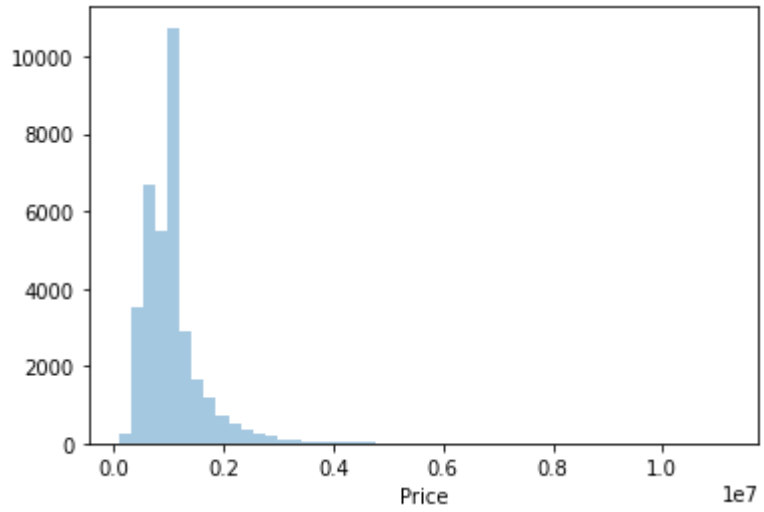
	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Distance	Post
0	Abbotsford	68 Studley St	2	h	1.050173e+06	SS	Jellis	3/09/2016	2.5	3
1	Abbotsford	85 Turner St	2	h	1.480000e+06	S	Biggin	3/12/2016	2.5	3
2	Abbotsford	25 Bloomburg St	2	h	1.035000e+06	S	Biggin	4/02/2016	2.5	3
3	Abbotsford	18/659 Victoria St	3	u	1.050173e+06	VB	Rounds	4/02/2016	2.5	3
4	Abbotsford	5 Charles St	3	h	1.465000e+06	SP	Biggin	4/03/2017	2.5	3
...	...	...	...	...	...	...	...	...	...	
34852	Yarraville	13 Burns St	4	h	1.480000e+06	PI	Jas	24/02/2018	6.3	3

Price before Scaling - Right skewed

In [9]:

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

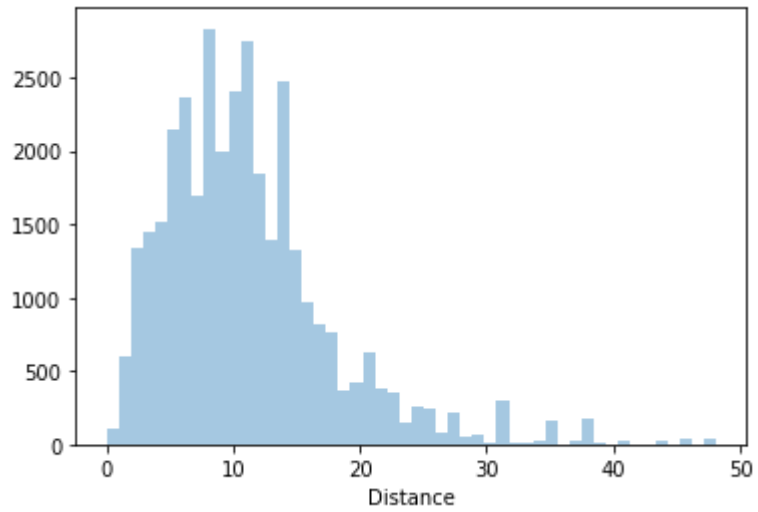
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated
function and will be removed in a future version. Please adapt your co
de to use either `displot` (a figure-level function with similar flexi
bility) or `histplot`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



Distance before Scaling - Right skewed

In [10]:

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



Standard Scaler

In [11]:

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

In [12]:

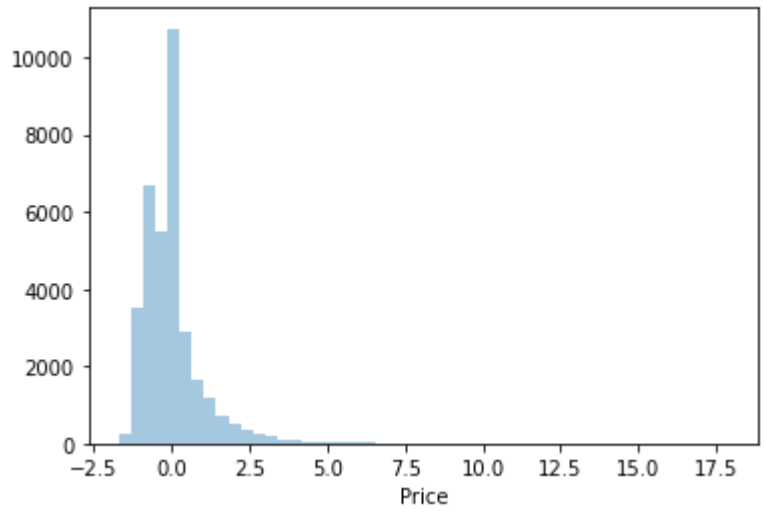
```
df[['Price','Distance','Landsize','Propertycount']] = scaler.fit_transform(df[['Pri
```

After Scaling

In [13]:

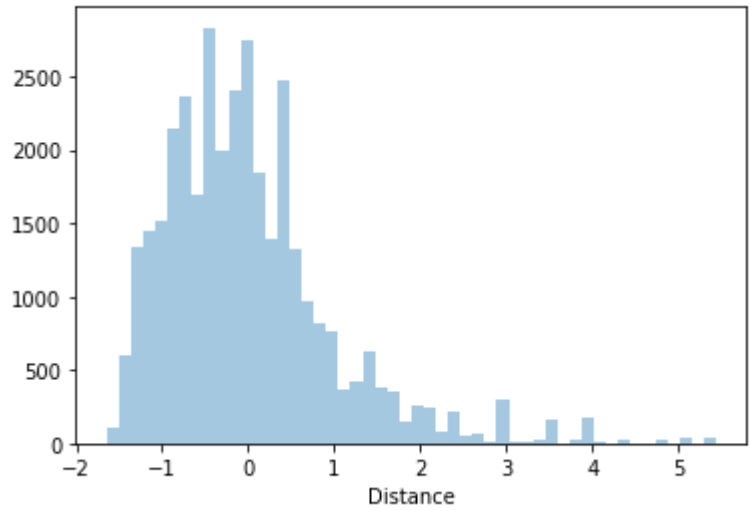
```
import seaborn as sb
from matplotlib import pyplot as plt
sb.distplot(df['Price'],kde = False)
plt.show()
```

```
/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 [14]:

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



In [15]:

```
df
```

Out[15]:

Price	Method	SellerG	Date	Distance	Postcode	...	Bathroom	Car	Lanc
0.000000	SS	Jellis	3/09/2016	-1.279322	3067.0	...	1.000000	1.000000	-0.16
0.757901	S	Biggin	3/12/2016	-1.279322	3067.0	...	1.000000	1.000000	-0.14
-0.026755	S	Biggin	4/02/2016	-1.279322	3067.0	...	1.000000	0.000000	-0.15
0.000000	VB	Rounds	4/02/2016	-1.279322	3067.0	...	2.000000	1.000000	-0.21
0.731452	SP	Biggin	4/03/2017	-1.279322	3067.0	...	2.000000	0.000000	-0.16
...	...	...	...	...	...	...	...	...	
0.757901	PI	Jas	24/02/2018	-0.719568	3013.0	...	1.000000	3.000000	-0.00
-0.285956	SP	Sweeney	24/02/2018	-0.719568	3013.0	...	2.000000	1.000000	-0.17
-0.608634	S	Jas	24/02/2018	-0.719568	3013.0	...	1.000000	2.000000	-0.13
0.158389	SP	hockingstuart	24/02/2018	-0.719568	3013.0	...	1.624798	1.728845	0.00
-0.053204	PI	RW	24/02/2018	-0.719568	3013.0	...	1.000000	0.000000	-0.12

KNN

In [82]:

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

In [83]:

```
from sklearn import preprocessing

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

In [84]:

```
df
```

Out[84]:

SellerG	Date	Distance	Postcode	...	Bathroom	Car	Landsize	BuildingArea	
Jellis	3/09/2016	-1.279322	3067.0	...	1.000000	1.000000	-0.169196	160.2564	196
Biggin	3/12/2016	-1.279322	3067.0	...	1.000000	1.000000	-0.141696	160.2564	196
Biggin	4/02/2016	-1.279322	3067.0	...	1.000000	0.000000	-0.158341	79.0000	190
Rounds	4/02/2016	-1.279322	3067.0	...	2.000000	1.000000	-0.214788	160.2564	196
Biggin	4/03/2017	-1.279322	3067.0	...	2.000000	0.000000	-0.166301	150.0000	190
...	...	...	...	...	...	...	...	...	
Jas	24/02/2018	-0.719568	3013.0	...	1.000000	3.000000	-0.000217	160.2564	196
Sweeney	24/02/2018	-0.719568	3013.0	...	2.000000	1.000000	-0.179327	104.0000	201
Jas	24/02/2018	-0.719568	3013.0	...	1.000000	2.000000	-0.135183	120.0000	200
ckingstuart	24/02/2018	-0.719568	3013.0	...	1.624798	1.728845	0.000000	160.2564	196
RW	24/02/2018	-0.719568	3013.0	...	1.000000	0.000000	-0.124328	103.0000	193

Type column is classified with the dataset

In [210]:

```
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 [211]:

```
from sklearn.preprocessing import MinMaxScaler #fixed import

scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [212]:

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

Out[212]:

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

Minkowski Distance

In [213]:

```
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 [214]:

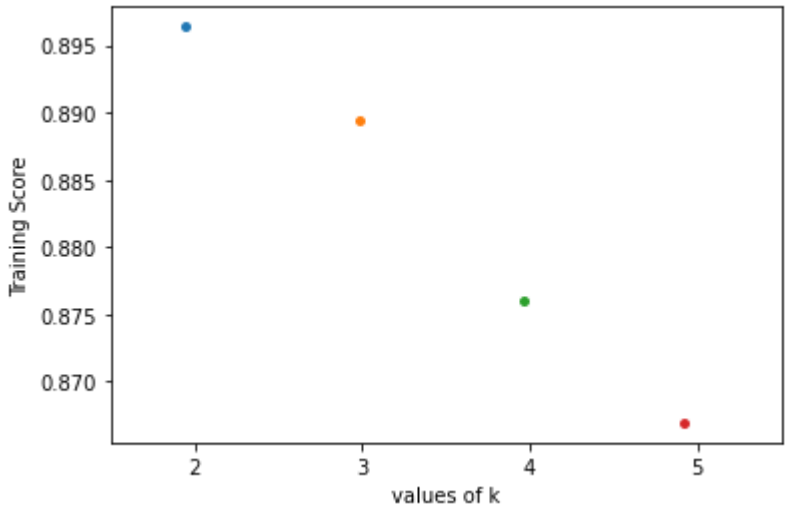
```
print("Accuracy Scores")
for keys, values in scores.items():
    print(keys, ':', values)
```

```
Accuracy Scores
2 : [0.8963959117805271, 0.8027825588066552]
3 : [0.8894746279361664, 0.8112449799196787]
4 : [0.875990675990676, 0.8115318416523236]
5 : [0.8669535592612516, 0.8156913367756741]
```

In [215]:

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

plt.show()
```

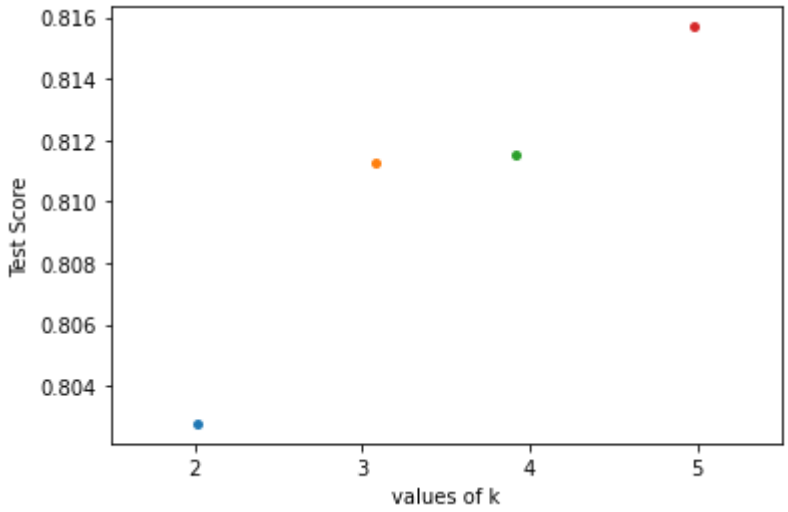
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result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



In [216]:

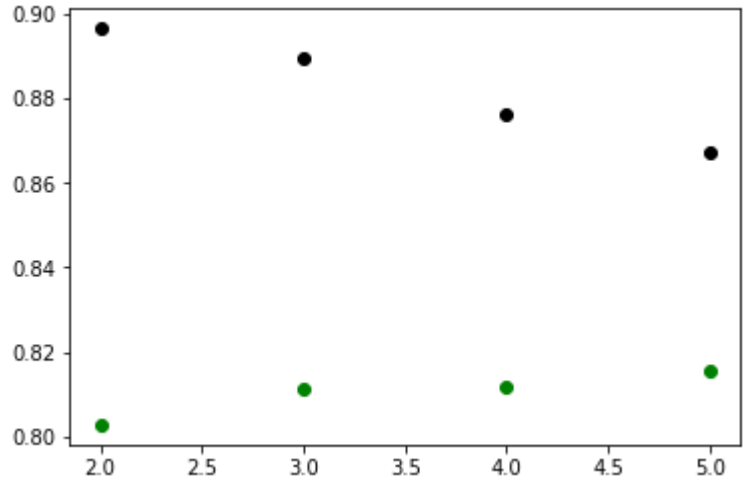
```
ax = sns.stripplot(K, test);
ax.set(xlabel ='values of k', ylabel ='Test Score')
plt.show()
```

```
/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 [217]:

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



In [218]:

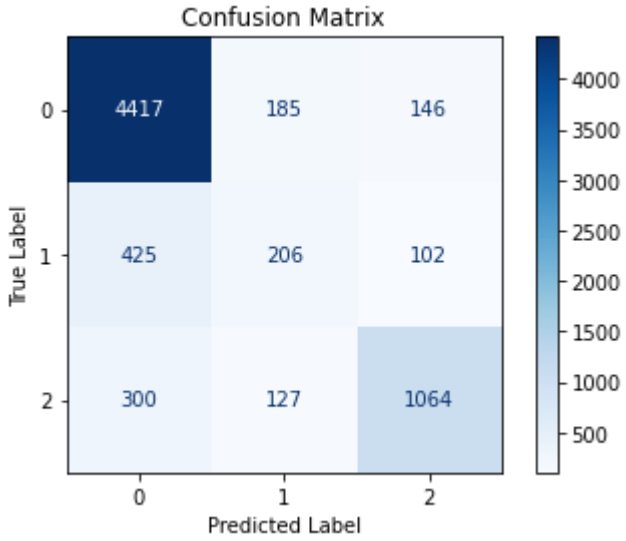
```
y_pred = clf.predict(X_test)

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

```
Accuracy: 0.8156913367756741
```

In [219]:

```
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      0.93      0.89      4748
           1       0.40      0.28      0.33       733
           2       0.81      0.71      0.76      1491

    accuracy                           0.82      6972
    macro avg       0.69      0.64      0.66      6972
    weighted avg       0.80      0.82      0.81      6972
```

In [220]:

```
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       0.40      0.28      0.33       733
           2       0.81      0.71      0.76      1491

    accuracy                           0.82      6972
   macro avg       0.69      0.64      0.66      6972
weighted avg       0.80      0.82      0.81      6972
```

Manhattan distance

In [98]:

```
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 [99]:

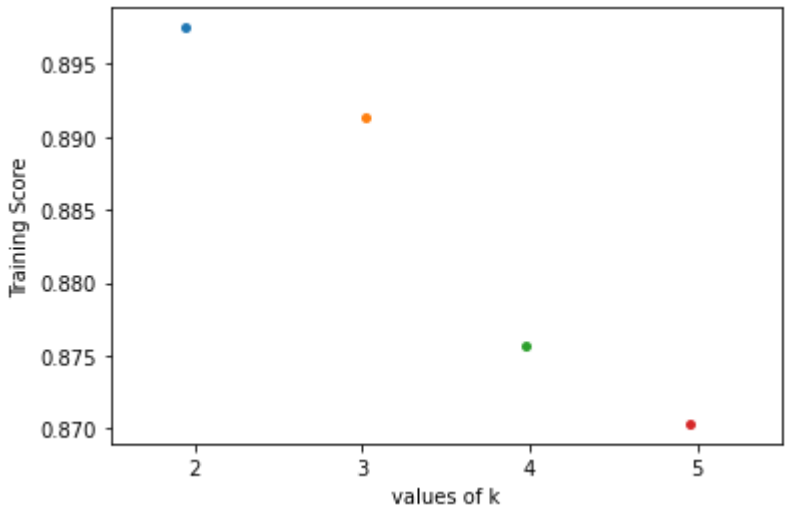
```
print("Accuracy Scores")
for keys, values in scores.items():
    print(keys, ':', values)
```

```
Accuracy Scores
2 : [0.8975076205845437, 0.8030694205393001]
3 : [0.8913752913752914, 0.8111015490533563]
4 : [0.875667921821768, 0.8125358577165807]
5 : [0.8703245472476242, 0.8172690763052208]
```

In [100]:

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

plt.show()
```

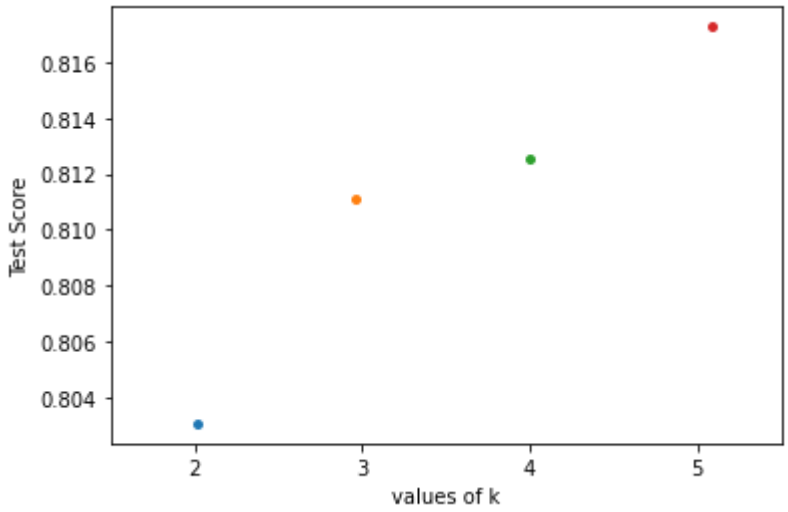
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result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



In [101]:

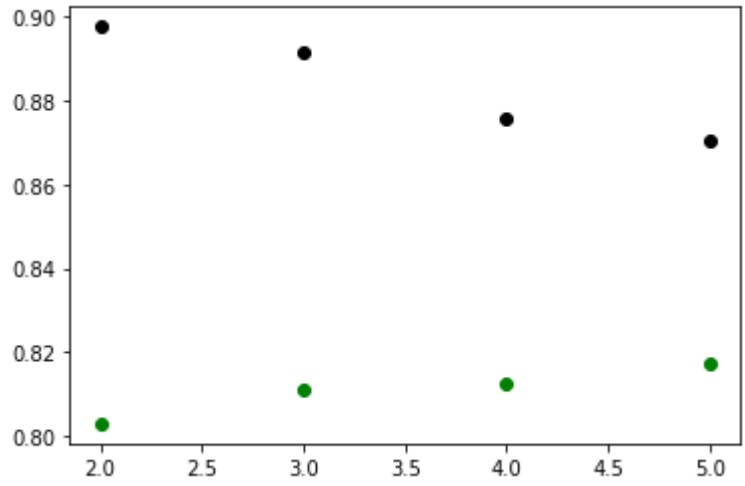
```
ax = sns.stripplot(K, test);
ax.set(xlabel ='values of k', ylabel ='Test Score')
plt.show()
```

```
/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 [102]:

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



In [103]:

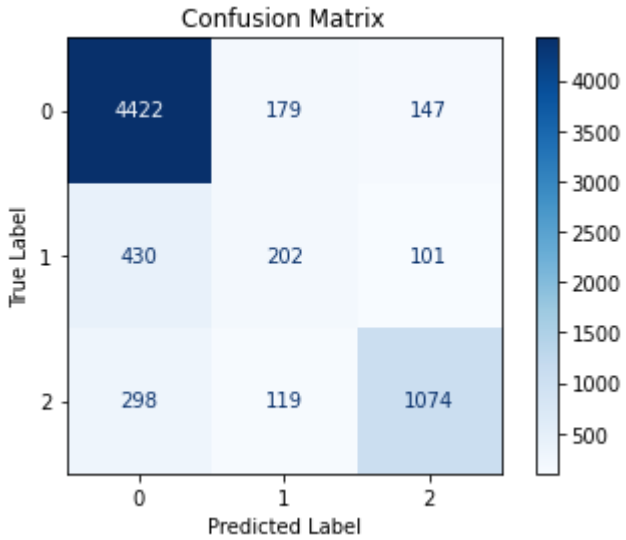
```
y_pred = clf.predict(X_test)

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

```
Accuracy: 0.8172690763052208
```

In [104]:

```
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 [105]:

```
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       0.40      0.28      0.33       733
           2       0.81      0.72      0.76      1491

    accuracy                           0.82      6972
   macro avg       0.69      0.64      0.66      6972
weighted avg       0.80      0.82      0.81      6972
```

We can infer that

precision recall f1-score support

```
           0       0.86      0.93      0.89      4748
           1       0.40      0.28      0.33       733
           2       0.81      0.72      0.76      1491

    accuracy                           0.82      6972
    macro avg       0.69      0.64      0.66      6972
    weighted avg       0.80      0.82      0.81      6972
```

Euclidean Distance

In [106]:

```
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 [107]:

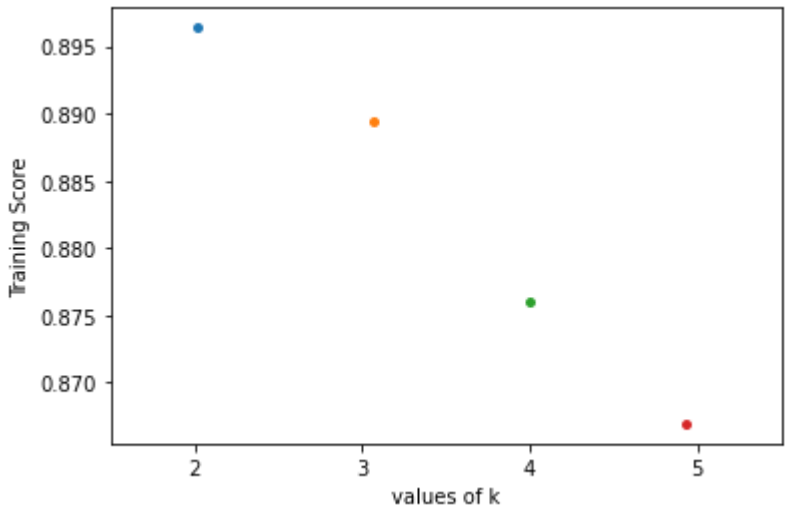
```
print("Accuracy Scores")
for keys, values in scores.items():
    print(keys, ':', values)
```

```
Accuracy Scores
2 : [0.8963959117805271, 0.8027825588066552]
3 : [0.8894746279361664, 0.8112449799196787]
4 : [0.875990675990676, 0.8115318416523236]
5 : [0.8669535592612516, 0.8156913367756741]
```

In [108]:

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

plt.show()
```

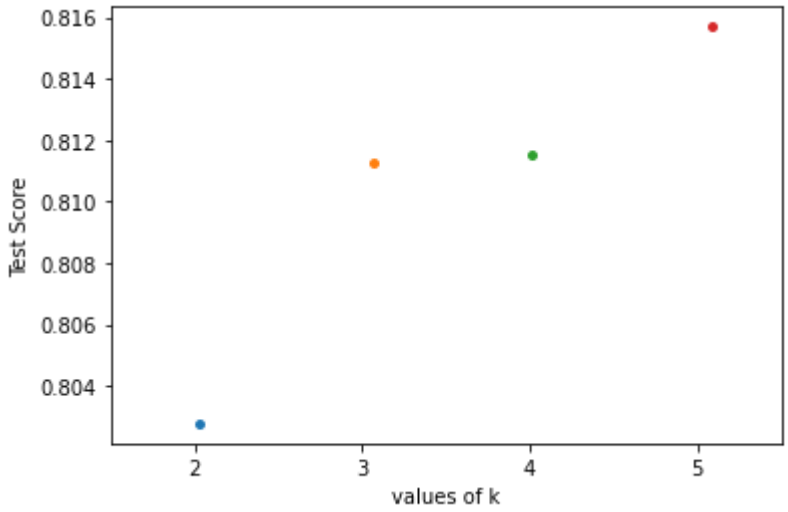
```
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/se
aborn/_decorators.py:36: FutureWarning: Pass the following variables a
s keyword args: x, y. From version 0.12, the only valid positional arg
ument will be `data`, and passing other arguments without an explicit
keyword will result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



In [109]:

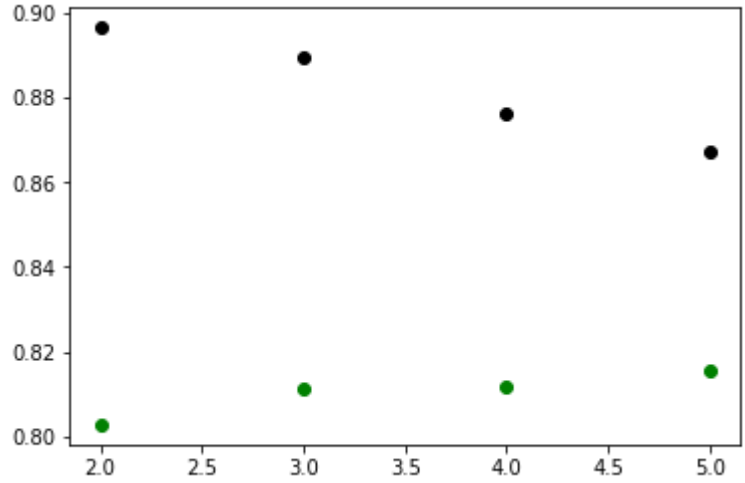
```
ax = sns.stripplot(K, test);
ax.set(xlabel ='values of k', ylabel ='Test Score')
plt.show()
```

```
/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 [110]:

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



In [111]:

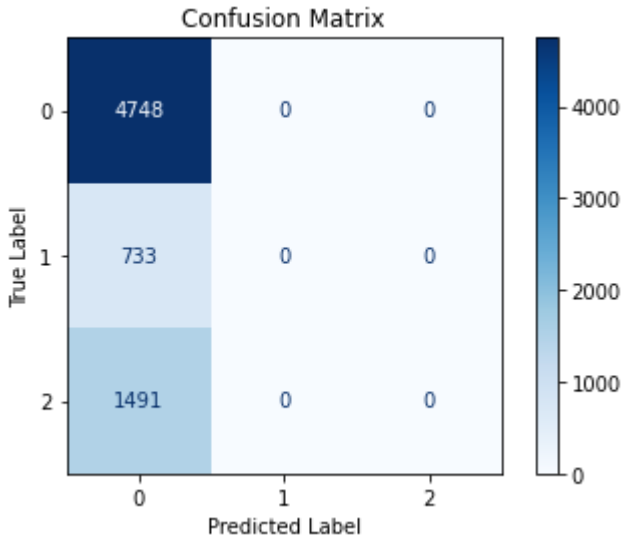
```
y_pred = clf.predict(X_test)

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

```
Accuracy: 0.8156913367756741
```

In [168]:

```
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 [113]:

```
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       0.40      0.28      0.33       733
           2       0.81      0.71      0.76      1491

    accuracy                           0.82      6972
   macro avg       0.69      0.64      0.66      6972
weighted avg       0.80      0.82      0.81      6972
```

We can infer that

```
              precision    recall  f1-score   support

           0       0.86      0.93      0.89      4748
           1       0.40      0.28      0.33       733
           2       0.81      0.71      0.76      1491

    accuracy                           0.82      6972
    macro avg       0.69      0.64      0.66      6972
    weighted avg       0.80      0.82      0.81      6972
```

Naïve Bayes Classifier

In [228]:

```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
```

Out[228]:

▾ GaussianNB

GaussianNB()

In [229]:

```
y_pred = clf.predict(X_test)
```

In [257]:

```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6810097532989099
```

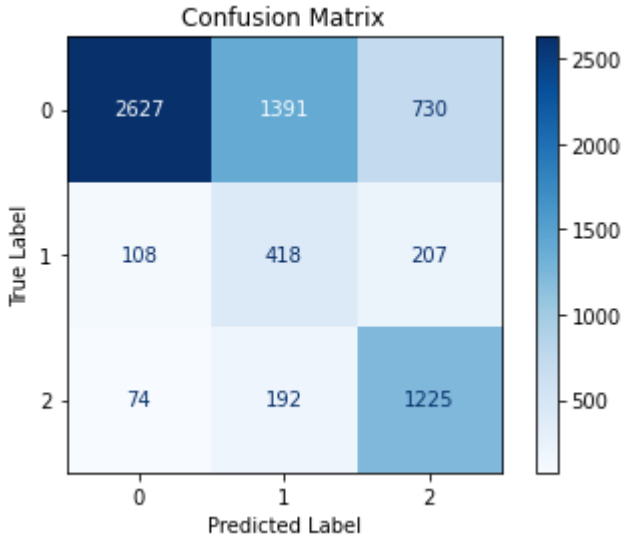
In [ ]:

```

```

In [231]:

```
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 [232]:

```
from sklearn.naive_bayes import BernoulliNB
```

In [233]:

```
clf = BernoulliNB()
```

In [234]:

```
clf.fit(X_train, y_train)
```

Out[234]:

▾ BernoulliNB

BernoulliNB()

In [235]:

```
y_pred = clf.predict(X_test)
```

In [236]:

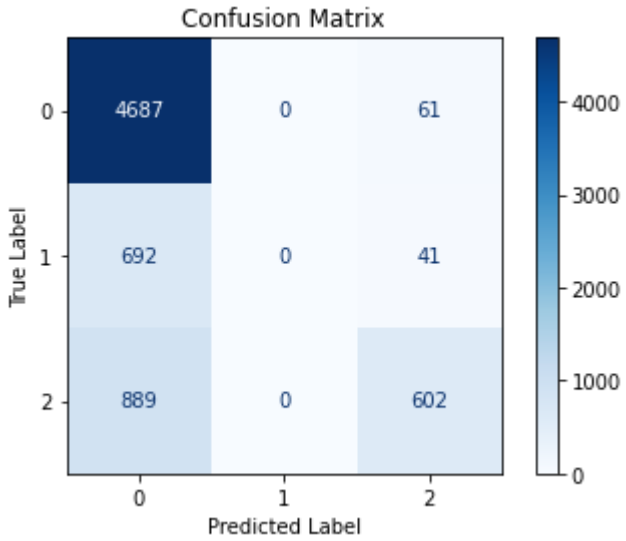
```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.7586058519793459
```

In [237]:

```
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 [238]:

```
from sklearn.naive_bayes import MultinomialNB
```

In [239]:

```
clf = MultinomialNB()
```

In [240]:

```
clf.fit(X_train, y_train)
```

Out[240]:

▾ MultinomialNB

MultinomialNB()

In [241]:

```
y_pred = clf.predict(X_test)
```

In [242]:

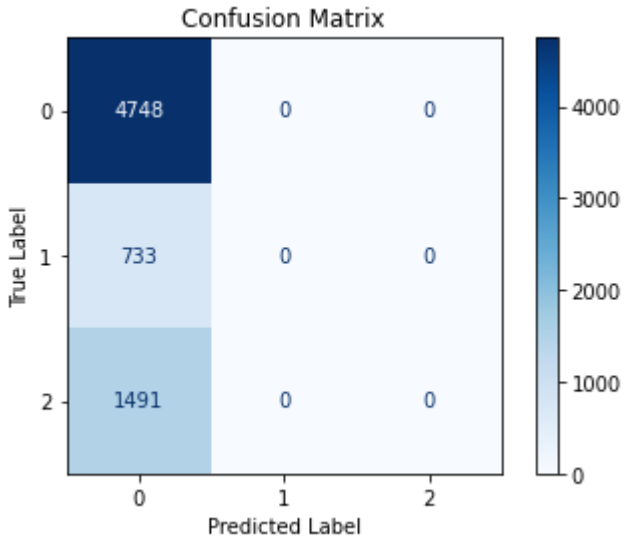
```
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.6810097532989099
```

In [244]:

```
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 [243]:

```
from sklearn.metrics import classification_report

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

```
              precision    recall  f1-score   support

           0       0.68      1.00      0.81      4748
           1       0.00      0.00      0.00       733
           2       0.00      0.00      0.00      1491

    accuracy                           0.68      6972
   macro avg       0.23      0.33      0.27      6972
weighted avg       0.46      0.68      0.55      6972



/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1327: UndefinedMetricWarning: Precisi
on and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1327: UndefinedMetricWarning: Precisi
on and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/joseph/Desktop/ml lab/lab1/mlenv/lib/python3.10/site-packages/sk
learn/metrics/_classification.py:1327: UndefinedMetricWarning: Precisi
on and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behav
ior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

We can infer that Naive Bayes provides less accuracy than KNN

Comparing different Naive Bayes methods, we can see that an accuracy rate of 68% is shown when compared with KNN, we can infer that KNN shows an accuracy of 81%. Hence for this particular dataset KNN is a well suited algorithm than Naive Bayes

In [ ]: