mobile-price-prediction-knn

December 2, 2023

[1]:
```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import confusion_matrix, classification_report
```

[2]:
```
df = pd.read_csv('/kaggle/input/mobile-price-classification/train.csv')
```

[3]:
```
df.head()
```

[3]:
```
   battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  m_dep  \
0            842     0          2.2         0   1       0           7    0.6
1           1021     1          0.5         1   0       1          53    0.7
2            563     1          0.5         1   2       1          41    0.9
3            615     1          2.5         0   0       0          10    0.8
4           1821     1          1.2         0  13       1          44    0.6

   mobile_wt  n_cores  …  px_height  px_width   ram  sc_h  sc_w  talk_time  \
0        188        2  …         20       756  2549     9     7         19
1        136        3  …        905      1988  2631    17     3          7
2        145        5  …       1263      1716  2603    11     2          9
3        131        6  …       1216      1786  2769    16     8         11
4        141        2  …       1208      1212  1411     8     2         15

   three_g  touch_screen  wifi  price_range
0        0             0     1            1
1        1             1     0            2
2        1             1     0            2
3        1             0     0            2
4        1             1     0            1

[5 rows x 21 columns]
```

[4]:

```
df.tail()
```

[4]:

```
      battery_power  blue  clock_speed  dual_sim  fc  four_g  int_memory  \
1995            794     1          0.5         1   0       1           2
1996           1965     1          2.6         1   0       0          39
1997           1911     0          0.9         1   1       1          36
1998           1512     0          0.9         0   4       1          46
1999            510     1          2.0         1   5       1          45

      m_dep  mobile_wt  n_cores  …  px_height  px_width   ram  sc_h  sc_w  \
1995    0.8        106        6  …       1222      1890   668    13     4
1996    0.2        187        4  …        915      1965  2032    11    10
1997    0.7        108        8  …        868      1632  3057     9     1
1998    0.1        145        5  …        336       670   869    18    10
1999    0.9        168        6  …        483       754  3919    19     4

      talk_time  three_g  touch_screen  wifi  price_range
1995         19        1             1     0            0
1996         16        1             1     1            2
1997          5        1             1     0            3
1998         19        1             1     1            0
1999          2        1             1     1            3

[5 rows x 21 columns]
```

[5]:

```
df.shape
```

[5]:

```
(2000, 21)
```

[6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   battery_power  2000 non-null   int64
 1   blue           2000 non-null   int64
 2   clock_speed    2000 non-null   float64
 3   dual_sim       2000 non-null   int64
 4   fc             2000 non-null   int64
 5   four_g         2000 non-null   int64
 6   int_memory     2000 non-null   int64
 7   m_dep          2000 non-null   float64
 8   mobile_wt      2000 non-null   int64
 9   n_cores        2000 non-null   int64
 10  pc             2000 non-null   int64
 11  px_height      2000 non-null   int64
 12  px_width       2000 non-null   int64
 13  ram            2000 non-null   int64
 14  sc_h           2000 non-null   int64
 15  sc_w           2000 non-null   int64
 16  talk_time      2000 non-null   int64
 17  three_g        2000 non-null   int64
 18  touch_screen   2000 non-null   int64
 19  wifi           2000 non-null   int64
 20  price_range    2000 non-null   int64
dtypes: float64(2), int64(19)
memory usage: 328.2 KB
```

[7]:
```
df.describe()
```

[7]:
```
       battery_power       blue  clock_speed     dual_sim           fc  \
count    2000.000000  2000.0000  2000.000000  2000.000000  2000.000000
mean     1238.518500     0.4950     1.522250     0.509500     4.309500
std       439.418206     0.5001     0.816004     0.500035     4.341444
min       501.000000     0.0000     0.500000     0.000000     0.000000
25%       851.750000     0.0000     0.700000     0.000000     1.000000
50%      1226.000000     0.0000     1.500000     1.000000     3.000000
75%      1615.250000     1.0000     2.200000     1.000000     7.000000
max      1998.000000     1.0000     3.000000     1.000000    19.000000

            four_g   int_memory        m_dep    mobile_wt      n_cores  …  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000  …
mean      0.521500    32.046500     0.501750   140.249000     4.520500  …
std       0.499662    18.145715     0.288416    35.399655     2.287837  …
min       0.000000     2.000000     0.100000    80.000000     1.000000  …
25%       0.000000    16.000000     0.200000   109.000000     3.000000  …
50%       1.000000    32.000000     0.500000   141.000000     4.000000  …
75%       1.000000    48.000000     0.800000   170.000000     7.000000  …
max       1.000000    64.000000     1.000000   200.000000     8.000000  …

         px_height     px_width          ram         sc_h         sc_w  \
count  2000.000000  2000.000000  2000.000000  2000.000000  2000.000000
mean    645.108000  1251.515500  2124.213000    12.306500     5.767000
std     443.780811   432.199447  1084.732044     4.213245     4.356398
min       0.000000   500.000000   256.000000     5.000000     0.000000
25%     282.750000   874.750000  1207.500000     9.000000     2.000000
50%     564.000000  1247.000000  2146.500000    12.000000     5.000000
75%     947.250000  1633.000000  3064.500000    16.000000     9.000000
max    1960.000000  1998.000000  3998.000000    19.000000    18.000000

         talk_time      three_g  touch_screen         wifi  price_range
count  2000.000000  2000.000000   2000.000000  2000.000000  2000.000000
```

```
mean     11.011000     0.761500     0.503000     0.507000     1.500000
std       5.463955     0.426273     0.500116     0.500076     1.118314
min       2.000000     0.000000     0.000000     0.000000     0.000000
25%       6.000000     1.000000     0.000000     0.000000     0.750000
50%      11.000000     1.000000     1.000000     1.000000     1.500000
75%      16.000000     1.000000     1.000000     1.000000     2.250000
max      20.000000     1.000000     1.000000     1.000000     3.000000

[8 rows x 21 columns]
```

```
[27]: df.isnull().sum().sum()
```

```
[27]: 0
```

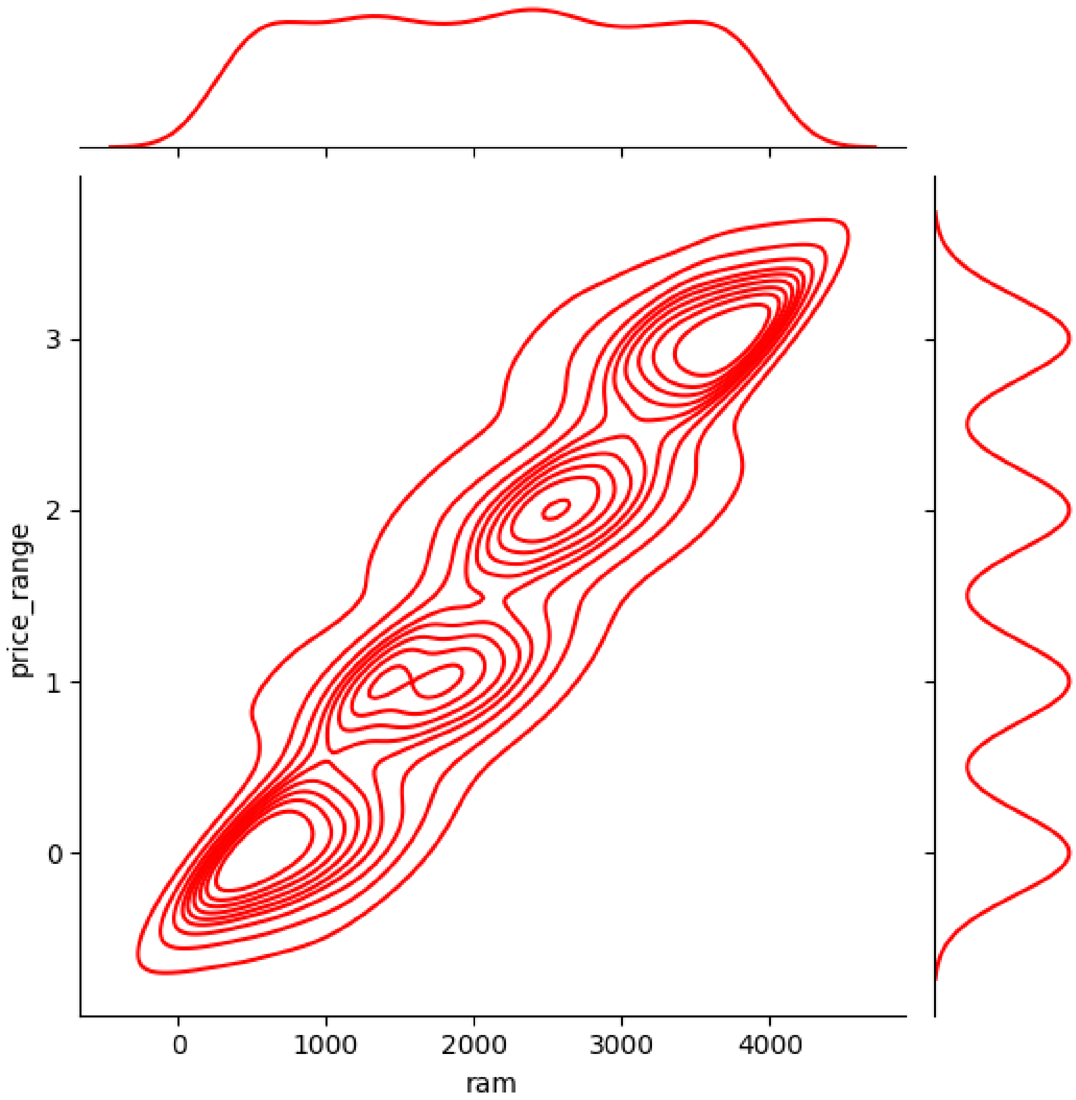
```
[9]: df.duplicated().any()
```

```
[9]: False
```

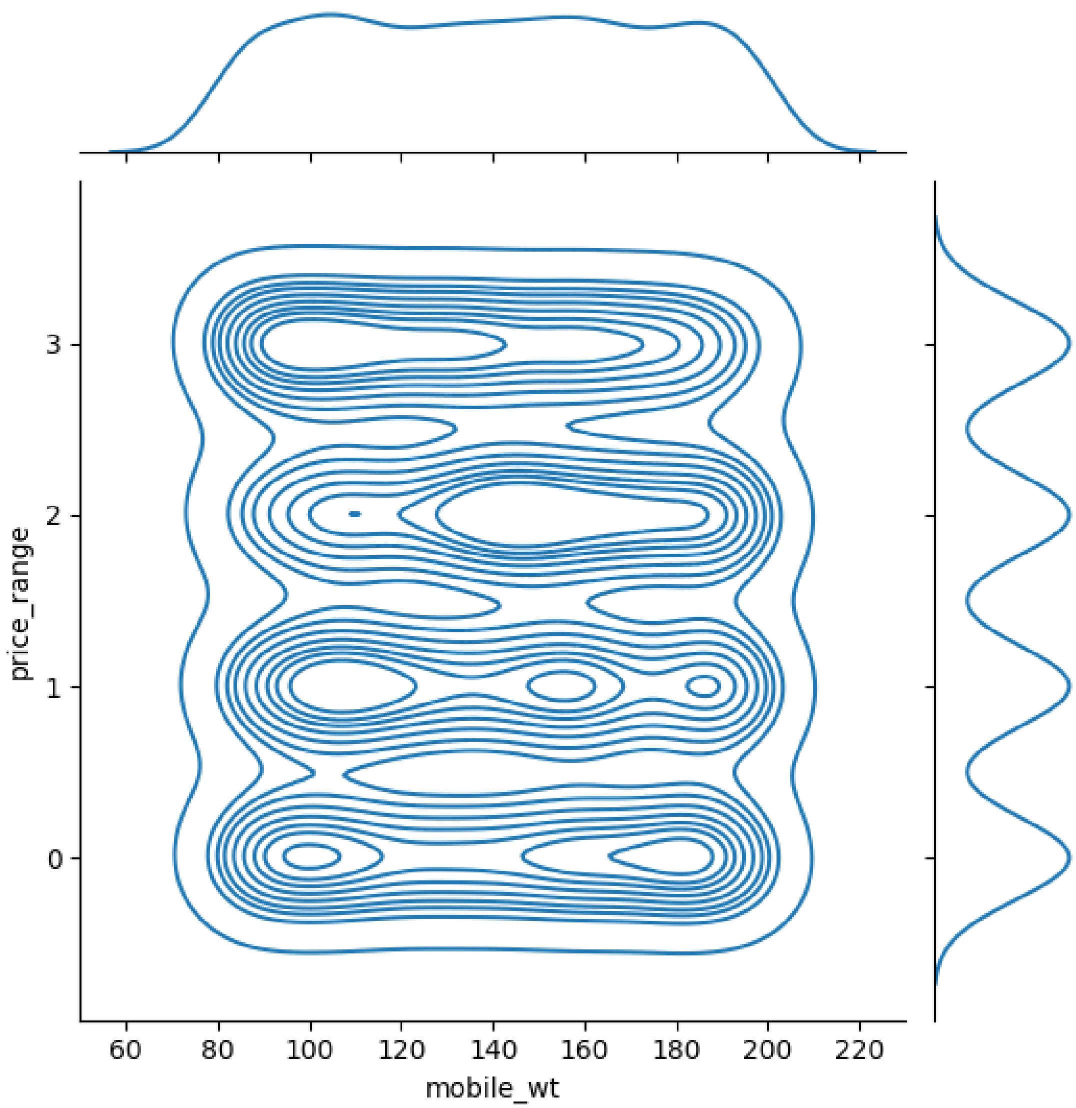
```
[10]: df.columns
```

```
[10]: Index(['battery_power', 'blue', 'clock_speed', 'dual_sim', 'fc', 'four_g',
             'int_memory', 'm_dep', 'mobile_wt', 'n_cores', 'pc', 'px_height',
             'px_width', 'ram', 'sc_h', 'sc_w', 'talk_time', 'three_g',
             'touch_screen', 'wifi', 'price_range'],
            dtype='object')
```

```
[12]: sns.jointplot(x='ram',y='price_range',data=df,color='red',kind='kde');
```

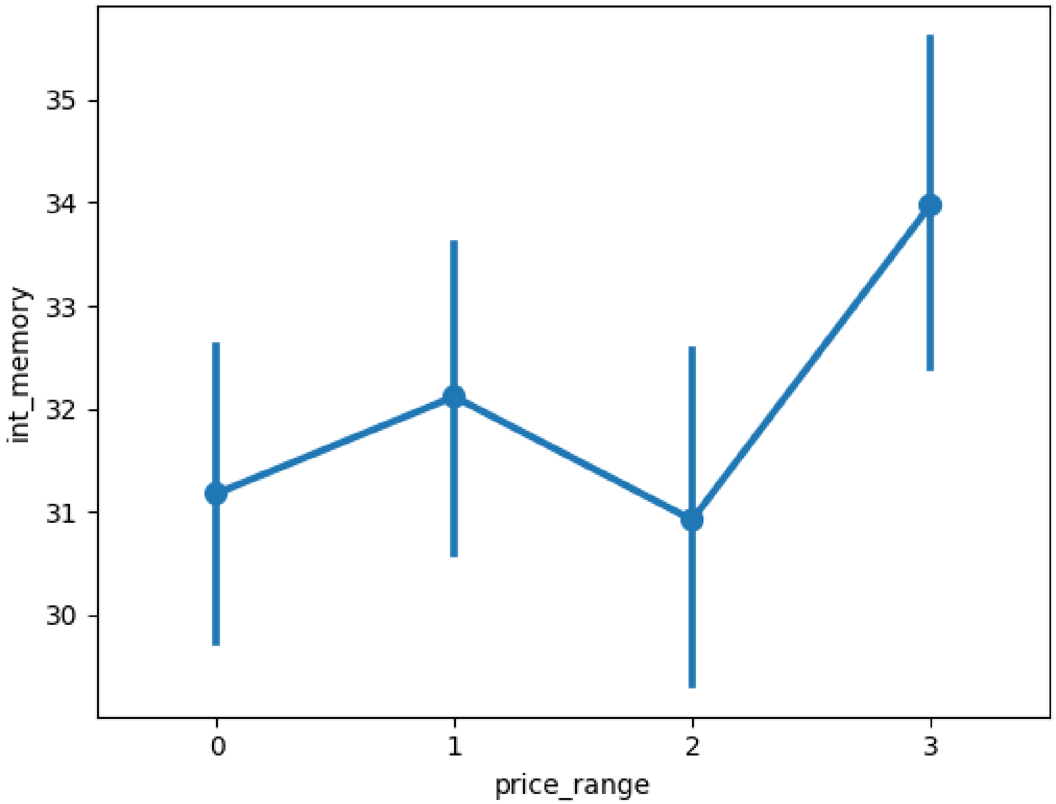


```
[13]: sns.jointplot(x='mobile_wt',y='price_range',data=df,kind='kde');
```

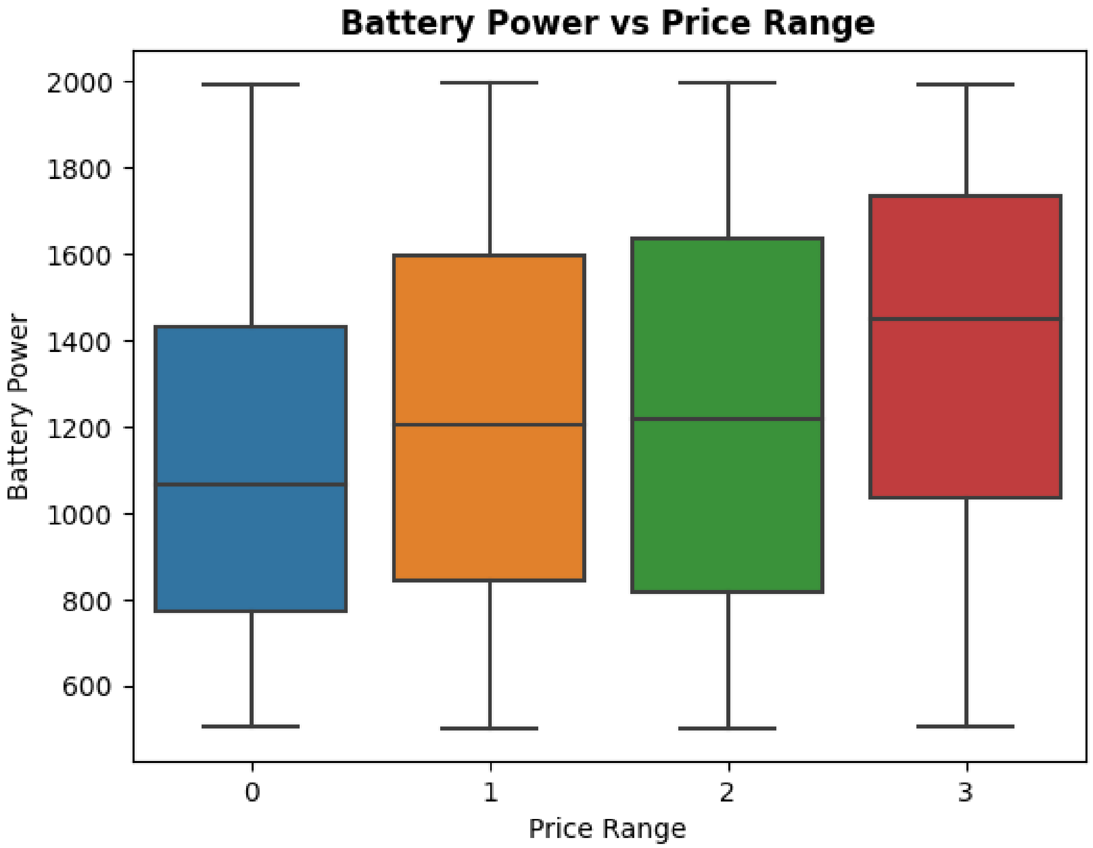


```
[14]: sns.pointplot(y="int_memory", x="price_range", data=df)
```

```
[14]: <Axes: xlabel='price_range', ylabel='int_memory'>
```

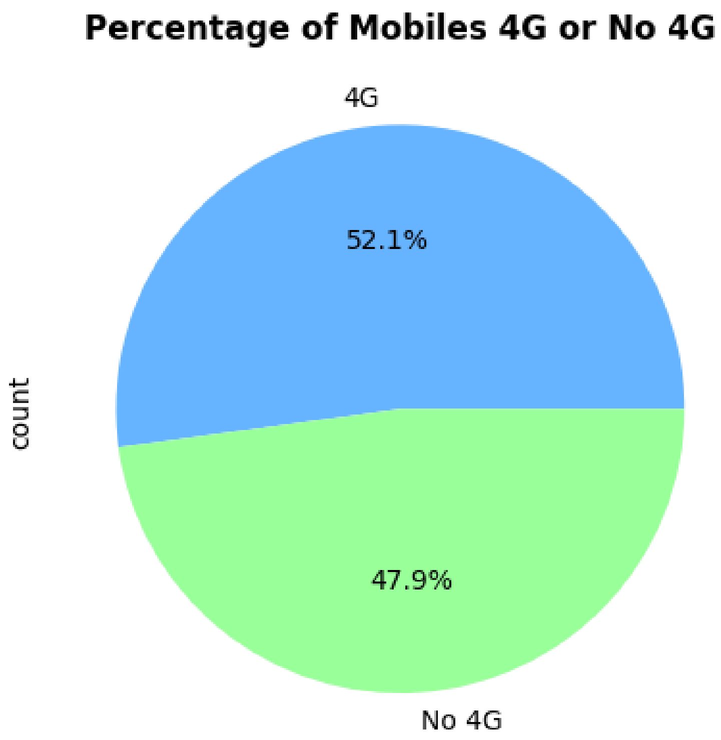


[15]:
```
sns.boxplot(x='price_range', y='battery_power', data=df)
plt.xlabel('Price Range')
plt.ylabel('Battery Power')
plt.title('Battery Power vs Price Range', weight='bold')
plt.show()
```



[16]:
```
import plotly.express as px
fig = px.scatter_3d(df.head(1000), x='ram', y='battery_power', z='px_width',␣
  ↪color='price_range')
fig.show()
```

[18]:
```
four_g = df['four_g'].value_counts()
plt.title('Percentage of Mobiles 4G or No 4G', weight='bold')
four_g.plot.pie(autopct="%.1f%%", labels=['4G', 'No 4G'],colors =␣
  ↪['#66b3ff','#99ff99'])
plt.show()
```



[19]:
```python
X = df.drop('price_range', axis=1)
y = df['price_range']
```

[20]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.33,␣
 ↪random_state=100)
```

[21]:
```python
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=10)
knn.fit(X_train,y_train)
```

[21]:
```
KNeighborsClassifier(n_neighbors=10)
```

[22]:
```python
knn.score(X_test,y_test)
```

[22]:
```
0.9196969696969697
```

[23]:
```python
from sklearn.metrics import classification_report,confusion_matrix
pred = knn.predict(X_test)
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support
```

```
           0       0.96      0.98      0.97       178
           1       0.88      0.93      0.90       163
           2       0.87      0.86      0.87       161
           3       0.97      0.90      0.93       158

    accuracy                           0.92       660
   macro avg       0.92      0.92      0.92       660
weighted avg       0.92      0.92      0.92       660
```

[24]:

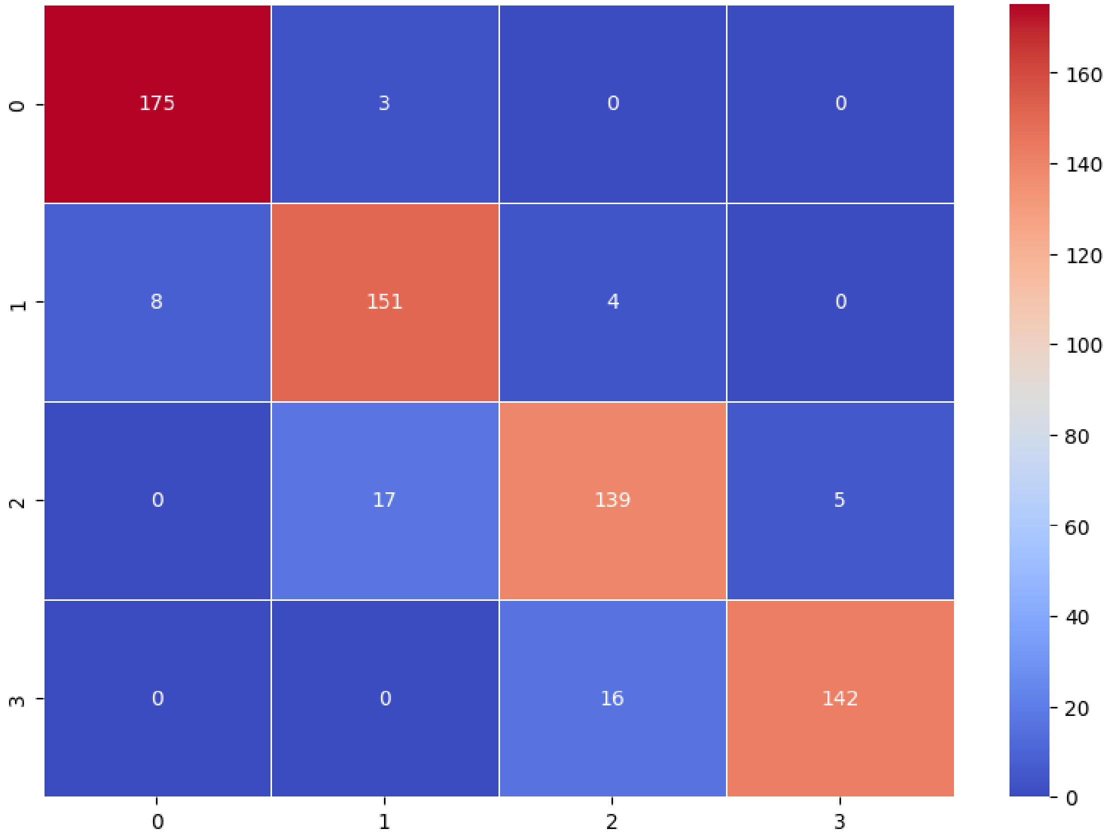
```
matrix=confusion_matrix(y_test,pred)
print(matrix)
```

```
[[175   3   0   0]
 [  8 151   4   0]
 [  0  17 139   5]
 [  0   0  16 142]]
```

[26]:

```
plt.figure(figsize = (10,7))
sns.heatmap(matrix, annot=True, cmap="coolwarm", linewidths=.5, fmt='g')
```

[26]: `<Axes: >`


175
3
0
0
8
151
4
0
0
17
139
5
0
0
16
142
0
1
2
3
0
1
2
3
160
140
120
100
80
60
40
20
0