In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```python
data=pd.read_csv(r"D:\project\mobile_price_range_data.csv")
```

In [3]:
```python
data.describe()
```

Out[3]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.00000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.04650
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.14571
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.00000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.00000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.00000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.00000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.00000

8 rows × 21 columns

In [46]:
```python
x
```

Out[46]:
```
array([[ 9.59360742e-01,  4.56078645e+00],
       [ 1.26572308e+00,  6.20712897e-01],
       [ 1.20212540e+00,  3.64414685e+00],
       [ 1.48170052e+00,  6.90074595e-01],
       [ 1.43472182e+00,  1.30662037e+00],
       [ 1.50757419e+00,  1.56787343e+00],
       [ 1.49493180e+00,  3.85848832e+00],
       [ 2.13003529e+00,  5.19209620e+00],
       [ 2.76808540e+00,  1.08782923e+00],
       [ 3.33818506e-01,  4.93645836e+00],
       [-4.74920358e-02,  5.47425256e+00],
       [ 1.65066793e+00,  9.16761995e-01],
       [ 4.38990142e-01,  4.53592883e+00],
       [ 2.10616050e+00,  3.49513189e+00],
       [ 1.36155806e+00,  1.36638252e+00],
       [ 2.14043942e+00,  7.06066610e-01],
       [ 1.84287117e+00,  7.26928839e-02],
       [ 2.15527162e+00,  1.27868252e+00],
       [ 1.00372519e+00,  4.19147702e+00],
       [ 1.25566754e+00,  3.38204112e+00],
       [ 1.16051297e+00,  1.16129868e+00],
       [ 1.06269622e+00,  5.17635143e+00],
       [ 2.43934644e+00, -7.25099666e-02],
       [ 1.65991049e+00,  3.56289184e+00],
       [ 1.56724897e+00,  1.78090633e-02],
       [ 1.57322172e+00,  4.83933793e-01],
       [ 2.13979079e-01,  4.88542535e+00],
       [ 1.72955064e+00,  1.14729369e+00],
       [ 6.69786996e-01,  3.59540802e+00],
       [ 4.59534668e-01,  5.44982630e+00],
       [ 1.99619601e+00,  4.99576688e-01],
       [ 2.73124907e+00,  2.49704755e-01],
       [ 1.67375987e+00,  1.30352364e+00],
       [ 1.69687788e+00,  7.54910622e-01],
       [ 2.29287155e+00,  2.41826755e-01],
       [ 1.71444449e+00,  5.02521524e+00],
       [ 3.35320909e+00,  1.69958043e+00],
       [ 1.89593761e+00,  5.18540259e+00],
       [ 1.41372442e+00,  4.38117707e+00],
       [ 1.86922139e+00,  5.44132083e+00],
       [ 4.88382309e-01,  3.26801777e+00],
       [-1.84892963e-03,  4.58145668e+00],
       [ 1.45513831e+00, -2.91989981e-02],
       [ 5.95676822e-01,  4.08614263e+00],
       [ 7.89338559e-01,  4.33748653e+00],
       [ 4.43598630e-01,  3.11530945e+00],
       [ 1.20083098e+00,  6.01671730e-01],
       [ 9.14338767e-01,  4.55014643e+00],
       [ 1.54462126e+00,  4.21078127e+00],
       [ 1.54632313e+00,  4.21297300e+00],
       [ 5.72793810e-01,  4.08805543e+00],
       [ 1.59331788e+00,  1.22121317e+00],
       [ 3.47138300e-01,  3.45177657e+00],
       [ 2.14917144e+00,  1.03697228e+00],
       [ 2.03169783e+00,  1.96807561e-01],
       [ 2.04505527e+00,  1.12515470e+00],
       [ 1.68353782e+00,  4.19583243e+00],
       [ 7.67522789e-01,  4.39759671e+00],
       [ 1.32000621e+00,  1.40428145e+00],
       [ 2.09680545e+00,  4.84741412e+00],
       [ 2.56936589e+00,  5.07048304e-01],
       [ 1.87271752e+00,  4.18069237e+00],
       [ 2.60137487e+00,  1.08799459e+00],
       [ 2.95195825e+00, -3.44327355e-01],
       [ 1.06923853e+00,  4.53068484e+00],
       [ 2.33812285e+00,  3.43116792e+00],
       [ 1.77710994e+00,  1.18655254e+00],
       [ 1.05241733e+00,  4.54498095e+00],
       [ 5.59529363e-01,  4.21400660e+00],
       [ 7.15177948e-01,  5.41334556e+00],
       [ 1.13078931e+00,  9.35620856e-01],
       [ 1.86985974e+00, -1.07938624e-01],
       [ 7.93137001e-03,  4.17614316e+00],
       [ 2.71506328e+00,  1.29082190e+00],
       [ 8.15468056e-01,  4.78526116e+00],
       [ 1.67280531e+00,  6.59300571e-01],
       [ 2.22322228e+00,  8.38773426e-01],
       [ 1.97553917e+00,  7.18989132e-01],
       [ 1.36678633e+00,  6.34971633e-01],
       [ 2.43040639e+00, -6.35709334e-02],
       [ 1.37964693e+00,  4.54826443e+00],
       [ 4.53791789e-01,  3.95647753e+00],
       [ 2.62492001e+00,  9.50194405e-01],
       [ 1.48859977e+00,  6.51633844e-01],
       [ 2.74666646e+00,  1.54543482e+00],
       [ 6.70478769e-01,  4.04094275e+00],
       [ 2.40122201e+00,  7.72684407e-01],
       [ 1.21767506e+00,  3.89290127e+00],
       [ 7.43873988e-01,  4.12240568e+00],
       [ 1.27955338e+00,  1.05789418e+00],
       [ 1.43289271e+00,  4.37679234e+00],
       [ 1.19008992e+00,  4.72773123e+00],
       [ 2.72396035e-01,  5.46996004e+00],
       [ 1.34471770e+00,  4.85711133e+00],
       [ 1.15521298e+00,  5.09961887e+00],
       [ 1.10123507e+00,  4.88977075e+00],
       [ 3.48515439e+00,  1.46435135e+00],
       [ 4.31891060e-01,  4.33495456e+00],
       [ 1.75644805e+00,  2.05538289e+00],
       [ 1.01618041e+00,  4.48527047e+00],
       [ 2.31102276e+00,  1.30380848e+00],
       [ 2.51834185e+00,  1.39176615e+00],
       [ 1.78194802e+00,  9.08151155e-01],
       [ 1.83375842e+00,  7.54036153e-01],
       [ 1.60841463e+00,  4.01800537e-01],
       [ 1.66909648e+00, -4.36378231e-01],
       [ 2.77180174e-01,  4.84428322e+00],
       [ 7.28098690e-01,  3.85531444e+00],
       [ 2.52706430e+00,  6.17812202e-01],
       [ 2.36790645e+00,  5.52190878e-01],
       [ 9.82570091e-01,  5.37530962e+00],
       [ 7.34363910e-01,  5.03725437e+00],
       [-5.55523811e-01,  4.69595848e+00],
       [ 1.61152972e+00,  1.82347242e+00],
       [ 2.04067185e+00,  4.54845114e-01],
       [ 2.24592863e-01,  4.77028154e+00],
       [ 1.16411070e+00,  3.79132988e+00],
       [ 2.60778282e+00,  1.08890025e+00],
       [ 2.01432256e+00,  1.92566929e+00],
       [ 1.08272576e+00,  4.06271877e+00],
       [ 3.41085289e+00,  8.72309369e-01],
       [ 1.39263752e+00,  9.28962707e-01],
       [ 1.24258802e+00,  4.50399192e+00],
       [ 2.09680487e+00,  3.71742060e+00],
       [ 2.36923352e+00,  7.94735861e-01]])
```

In [47]:
```python
y
```

Out[47]:
```
array([0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1])
```

In [4]:
```python
data.columns
```

Out[4]:
```
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')
```

1)Remove handle null values (if any)

In [5]:
```python
data.isnull().sum()
```

Out[5]:
```
battery_power    0
blue             0
clock_speed      0
dual_sim         0
fc               0
four_g           0
int_memory       0
m_dep            0
mobile_wt        0
n_cores          0
pc               0
px_height        0
px_width         0
ram              0
sc_h             0
sc_w             0
talk_time        0
three_g          0
touch_screen     0
wifi             0
price_range      0
dtype: int64
```

2)Split data into training and test data.

In [6]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
```

In [7]:
```python
train=pd.read_csv(r'D:\project\mobile_price_range_data.csv')
test=pd.read_csv(r'D:\project\mobile_price_range_data.csv')
```

In [8]:
```python
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

In [9]:
```python
train.head()
```

Out[9]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_co
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

In [10]:
```python
test.head()
```

Out[10]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_co
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

In [12]:
```python
train.shape
```

Out[12]:
```
(2000, 21)
```

In [14]:
```python
train.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
```

3)Apply the following models on the training dataset and generate the predicted value for the

In [15]:
```python
import pandas as pd
from sklearn.preprocessing import StandardScaler
data = pd.read_csv("D:\project\mobile_price_range_data.csv")
```

In [16]:
```python
x = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
x = StandardScaler().fit_transform(x)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20
, random_state=0)
```

In [18]:
```python
from sklearn.linear_model import LogisticRegression
lr= LogisticRegression()
lr.fit(x_train, y_train)
pred=lr.predict(x_test)
```

In [19]:
```python
pred
```

Out[19]:
```
array([3, 0, 2, 2, 3, 0, 0, 3, 3, 1, 1, 3, 0, 2, 3, 0, 3, 2, 2, 1, 0, 0,
       3, 1, 2, 2, 3, 1, 3, 1, 1, 0, 2, 0, 2, 3, 0, 0, 3, 3, 3, 1, 3, 3,
       1, 3, 0, 1, 3, 1, 1, 3, 0, 3, 0, 2, 2, 2, 0, 3, 3, 1, 3, 2, 1, 2,
       3, 2, 2, 2, 3, 2, 1, 0, 1, 3, 2, 2, 1, 2, 3, 3, 3, 0, 0, 0, 2, 1,
       2, 3, 1, 2, 2, 1, 0, 3, 3, 3, 0, 3, 1, 1, 3, 1, 3, 2, 2, 3, 2, 3,
       3, 0, 0, 1, 3, 3, 0, 0, 1, 0, 0, 3, 2, 2, 1, 2, 1, 1, 0, 2, 1, 3,
       3, 3, 3, 3, 3, 2, 0, 1, 1, 2, 1, 3, 0, 3, 0, 0, 2, 0, 1, 1, 1, 1,
       3, 0, 0, 3, 1, 3, 2, 1, 3, 1, 2, 3, 3, 2, 1, 0, 3, 1, 2, 3, 3, 0,
       2, 2, 3, 1, 2, 1, 0, 1, 2, 2, 2, 0, 3, 3, 1, 1, 0, 2, 3, 0, 1, 2,
       2, 0, 3, 3, 3, 1, 2, 3, 3, 3, 0, 0, 0, 2, 3, 3, 0, 0, 1, 3, 2, 3,
       3, 3, 0, 0, 2, 3, 3, 1, 0, 2, 0, 0, 0, 3, 2, 1, 2, 2, 1, 1, 0, 2,
       3, 3, 0, 0, 1, 3, 3, 1, 3, 0, 3, 1, 1, 0, 2, 3, 3, 2, 0, 0, 1, 2,
       3, 2, 2, 3, 2, 1, 0, 3, 3, 2, 1, 3, 2, 2, 2, 1, 0, 2, 2, 1, 0, 0,
       2, 2, 2, 3, 0, 1, 3, 0, 2, 2, 3, 0, 2, 0, 1, 1, 3, 0, 0, 2, 3, 1,
       2, 0, 2, 0, 3, 0, 3, 3, 2, 3, 1, 2, 2, 1, 1, 1, 0, 1, 0, 3, 1, 0,
       3, 0, 0, 1, 3, 0, 3, 1, 1, 0, 1, 3, 0, 2, 1, 1, 2, 1, 1, 0, 2, 0,
       0, 3, 1, 2, 3, 2, 2, 0, 3, 2, 2, 1, 3, 2, 3, 3, 3, 0, 2, 0, 3, 0,
       1, 1, 2, 3, 1, 3, 1, 2, 0, 1, 2, 3, 0, 0, 1, 3, 0, 3, 0, 2, 2, 1,
       1, 0, 2, 0], dtype=int64)
```

In [20]:
```python
x_test
```

Out[20]:
```
array([[ 0.49050173,  1.0100505 , -1.2530642 , ...,  0.55964063,
         0.99401789, -1.01409939],
       [-0.33352087,  1.0100505 , -1.2530642 , ..., -1.78686097,
         0.99401789, -1.01409939],
       [ 0.64984312,  1.0100505 ,  0.34046327, ...,  0.55964063,
        -1.00601811,  0.98609664],
       ...,
       [-1.46712103,  1.0100505 ,  1.32109556, ..., -1.78686097,
         0.99401789,  0.98609664],
       [ 0.98901264,  1.0100505 ,  0.0953052 , ..., -1.78686097,
        -1.00601811, -1.01409939],
       [ 0.11718762, -0.9900495 , -1.2530642 , ..., -1.78686097,
         0.99401789,  0.98609664]])
```

In [21]:
```python
#Logistic Regression Confusion_Matrix

from sklearn.metrics import confusion_matrix
confusion_matrix(pred1,y_test)
```

Out[21]:
```
array([[ 93,   2,   0,   0],
       [  2,  85,   3,   0],
       [  0,   5,  91,   1],
       [  0,   0,   5, 113]], dtype=int64)
```

In [22]:
```python
#Logistic Regression classification_report

from sklearn.metrics import classification_report
```

In [23]:
```python
print(classification_report(pred1,y_test))
```

```
              precision    recall  f1-score   support

           0       0.98      0.98      0.98        95
           1       0.92      0.94      0.93        90
           2       0.92      0.94      0.93        97
           3       0.99      0.96      0.97       118

    accuracy                           0.95       400
   macro avg       0.95      0.95      0.95       400
weighted avg       0.96      0.95      0.96       400
```

In [24]:
```python
#Logistic Regression Best Accuracy

from sklearn.metrics import accuracy_score
lr_acc=accuracy_score(pred1,y_test)
lr_acc
```

Out[24]:
```
0.955
```

In [25]:
```python
#KNN Classification_report

from sklearn.metrics import classification_report
```

In [26]:
```python
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(x_train,y_train)
```

Out[26]:
```
KNeighborsClassifier()
```

In [27]:
```python
pred2=knn.predict(x_test)
pred2
```

Out[27]:
```
array([2, 0, 2, 0, 1, 1, 0, 2, 3, 3, 1, 3, 1, 2, 2, 0, 3, 2, 2, 0, 1, 1,
       3, 0, 2, 2, 3, 0, 3, 0, 0, 3, 1, 0, 1, 3, 0, 0, 3, 3, 3, 1, 3, 1,
       2, 3, 0, 2, 2, 0, 0, 2, 0, 2, 0, 2, 1, 2, 0, 3, 2, 3, 1, 3, 0, 1,
       2, 2, 1, 3, 3, 3, 1, 0, 1, 1, 3, 2, 1, 0, 3, 2, 2, 1, 0, 1, 1, 1,
       1, 3, 1, 2, 1, 0, 1, 3, 3, 2, 0, 3, 0, 1, 1, 1, 1, 1, 0, 2, 2, 2,
       2, 1, 2, 2, 1, 3, 0, 0, 0, 0, 0, 1, 1, 3, 1, 1, 0, 1, 0, 3, 0, 3,
       2, 3, 2, 3, 2, 2, 1, 0, 2, 2, 0, 2, 0, 2, 0, 0, 1, 0, 1, 1, 1, 0,
       3, 0, 0, 3, 2, 2, 1, 3, 2, 1, 2, 3, 2, 1, 0, 1, 2, 1, 0, 3, 3, 1,
       1, 1, 3, 2, 1, 1, 0, 2, 2, 1, 2, 0, 3, 2, 1, 1, 0, 2, 2, 0, 0, 2,
       3, 0, 2, 3, 2, 1, 3, 2, 3, 2, 0, 1, 0, 1, 3, 3, 0, 0, 0, 3, 1, 3,
       2, 3, 0, 0, 2, 2, 2, 0, 1, 1, 1, 0, 0, 3, 2, 0, 2, 1, 1, 1, 0, 1,
       3, 1, 1, 0, 1, 3, 2, 2, 3, 0, 3, 3, 1, 0, 2, 2, 2, 3, 0, 1, 1, 2,
       3, 3, 1, 2, 2, 1, 0, 3, 2, 2, 3, 2, 1, 1, 2, 2, 0, 0, 1, 1, 1, 0,
       1, 2, 1, 0, 0, 1, 3, 0, 3, 3, 3, 0, 3, 0, 0, 2, 3, 0, 0, 1, 3, 1,
       1, 0, 3, 1, 2, 1, 1, 2, 2, 2, 0, 0, 1, 2, 0, 1, 0, 1, 0, 2, 2, 0,
       3, 2, 0, 0, 2, 0, 2, 1, 1, 0, 0, 2, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 2, 1, 2, 2, 0, 2, 0, 3, 2, 1, 1, 2, 1, 3, 2, 3, 0, 1, 0, 1, 0,
       1, 0, 2, 2, 1, 3, 1, 1, 0, 1, 1, 3, 1, 2, 1, 3, 0, 3, 0, 2, 1, 1,
       2, 1, 1, 0], dtype=int64)
```

In [28]:
```python
# Confusion matrix

from sklearn.metrics import confusion_matrix
confusion_matrix(pred2,y_test)
```

Out[28]:
```
array([[68, 30,  9,  0],
       [22, 45, 41,  9],
       [ 4, 11, 36, 49],
       [ 1,  6, 13, 56]], dtype=int64)
```

In [31]:
```python
#classification report

print(classification_report(pred2,y_test))
```

```
              precision    recall  f1-score   support

           0       0.72      0.64      0.67       107
           1       0.49      0.38      0.43       117
           2       0.36      0.36      0.36       100
           3       0.49      0.74      0.59        76

    accuracy                           0.51       400
   macro avg       0.51      0.53      0.51       400
weighted avg       0.52      0.51      0.51       400
```

In [32]:
```python
#best accuracy

from sklearn.metrics import accuracy_score
knn.acc=accuracy_score(pred2,y_test)
knn.acc
```

Out[32]:
```
0.5125
```

SVM Classifier with linear and rbf kernel

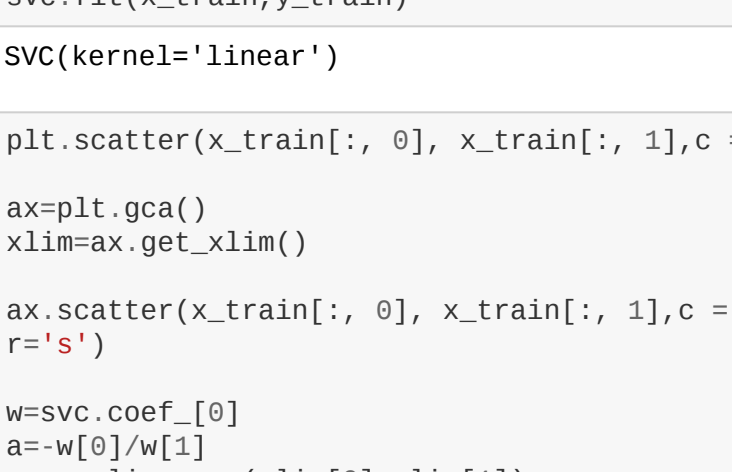
In [45]:
```python
from sklearn.datasets.samples_generator import make_blobs
from matplotlib import pyplot as plt
import numpy as np
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.svm import SVC
```

In [34]:
```python
data=read_csv(r'D:\project\mobile_price_range_data.csv')
```

In [35]:
```python
x,y=make_blobs(n_samples=125, centers=2, cluster_std=0.60, random_state=
0)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=20,
```

In [36]:
```python
plt.scatter(x_train[:, 0], x_train[:, 1],c = y_train, cmap='winter')
```

Out[36]:
```
<matplotlib.collections.PathCollection at 0x15fbffc54f0>
```



In [37]:
```python
svc=SVC(kernel='linear')
svc.fit(x_train,y_train)
```

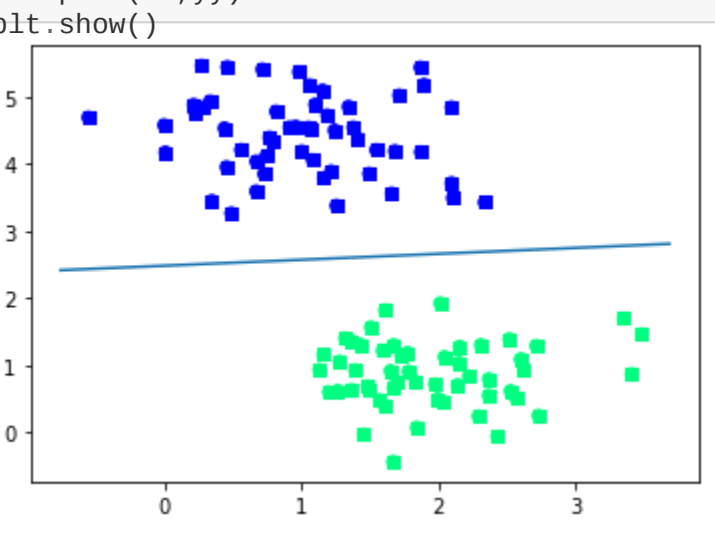
Out[37]:
```
SVC(kernel='linear')
```

In [38]:
```python
plt.scatter(x_train[:, 0], x_train[:, 1],c = y_train, cmap='winter')

ax=plt.gca()
xlim=ax.get_xlim()

ax.scatter(x_train[:, 0], x_train[:, 1],c = y_train, cmap='winter',marke
r='s')

w=svc.coef_[0]
a=-w[0]/w[1]
xx=np.linspace(xlim[0],xlim[1])
yy=a*xx-(svc.intercept_[0]/w[1])
plt.plot(xx,yy)
plt.show()
```



In [39]:
```python
y_pred=svc.predict(x_test)
```

In [41]:
```python
#Confusion matrix

confusion_matrix(y_test,y_pred)
```

Out[41]:
```
array([[11,  0],
       [ 0,  9]], dtype=int64)
```

In [42]:
```python
#classification report

from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        11
           1       1.00      1.00      1.00         9

    accuracy                           1.00        20
   macro avg       1.00      1.00      1.00        20
weighted avg       1.00      1.00      1.00        20
```

In [43]:
```python
#best accuracy.

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

```
Accuracy:  1.0
```

4)Predict the price range for test data

In [48]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
```

In [49]:
```python
train=pd.read_csv(r'D:\project\mobile_price_range_data.csv')
test=pd.read_csv(r'D:\project\mobile_price_range_data.csv')
```

In [50]:
```python
pd.set_option('display.max_rows',None)
pd.set_option('display.max_columns',None)
```

In [51]:
```python
train.head()
```

Out[51]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_co
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

In [52]:
```python
test.head()
```

Out[52]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_co
0	842	0	2.2	0	1	0	7	0.6	188	
1	1021	1	0.5	1	0	1	53	0.7	136	
2	563	1	0.5	1	2	1	41	0.9	145	
3	615	1	2.5	0	0	0	10	0.8	131	
4	1821	1	1.2	0	13	1	44	0.6	141	

In [53]:
```python
test.describe()
```

Out[53]:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory
count	2000.000000	2000.0000	2000.000000	2000.000000	2000.000000	2000.000000	2000.00000
mean	1238.518500	0.4950	1.522250	0.509500	4.309500	0.521500	32.04650
std	439.418206	0.5001	0.816004	0.500035	4.341444	0.499662	18.14571
min	501.000000	0.0000	0.500000	0.000000	0.000000	0.000000	2.00000
25%	851.750000	0.0000	0.700000	0.000000	1.000000	0.000000	16.00000
50%	1226.000000	0.0000	1.500000	1.000000	3.000000	1.000000	32.00000
75%	1615.250000	1.0000	2.200000	1.000000	7.000000	1.000000	48.00000
max	1998.000000	1.0000	3.000000	1.000000	19.000000	1.000000	64.00000

In [54]:
```python
sns.countplot(test['price_range'])
```

Out[54]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15fbfffb850>
```

In [55]:
```python
x_test
```

Out[55]:
```
array([[ 2.7680854 ,  1.08782923],
       [ 0.57279381,  4.08805543],
       [ 0.59567682,  4.08614263],
       [ 1.56724897,  0.01780906],
       [ 2.95195825, -0.34432736],
       [ 1.75644805,  2.05538289],
       [ 2.03169783,  0.19680756],
       [ 1.43289271,  4.37679234],
       [ 2.13003529,  5.1920962 ],
       [ 0.73436391,  5.03725437],
       [-0.04749204,  5.47425256],
       [ 1.86985974, -0.10793862],
       [ 0.43189106,  4.33495456],
       [ 1.54462126,  4.21078127],
       [ 2.40122201,  0.77268441],
       [ 1.2021254 ,  3.64414685],
       [ 1.01618041,  4.48527047],
       [ 2.74666646,  1.54543482],
       [ 2.43934644, -0.07250997],
       [ 0.44359863,  3.11530945]])
```

In [ ]: