In [44]:

```
#import standard libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [45]:

```
data=pd.read_csv('/kaggle/input/mobile-device-usage-and-user-behavior-dataset/
data.head()
```

Out[45]:

	User ID	Device Model	Operating System	App Usage Time (min/day)	Screen On Time (hours/day)	Battery Drain (mAh/day)	Number of Apps Installed	Data Usage (MB/day)	Age	Gender
0	1	Google Pixel 5	Android	393	6.4	1872	67	1122	40	Male
1	2	OnePlus 9	Android	268	4.7	1331	42	944	47	Female
2	3	Xiaomi Mi 11	Android	154	4.0	761	32	322	42	Male
3	4	Google Pixel 5	Android	239	4.8	1676	56	871	20	Male
4	5	iPhone 12	iOS	187	4.3	1367	58	988	31	Female

In [46]:

```
data['Device Model'].unique()
```

Out[46]:

```
array(['Google Pixel 5', 'OnePlus 9', 'Xiaomi Mi 11', 'iPhone 12',
       'Samsung Galaxy S21'], dtype=object)
```

In [47]:

```
data['Device Model'].nunique()
```

Out[47]:

```
5
```

In [48]:

```
data['Operating System'].unique()
```

Out[48]:

```
array(['Android', 'iOS'], dtype=object)
```

In [49]:

```
duplicate=data.duplicated().sum()
print(f'The data set contains {duplicate} values')
```

```
The data set contains 0 values
```

In [50]:
```
data.isna().sum()
```

Out[50]:
```
User ID                       0
Device Model                  0
Operating System              0
App Usage Time (min/day)      0
Screen On Time (hours/day)    0
Battery Drain (mAh/day)       0
Number of Apps Installed      0
Data Usage (MB/day)           0
Age                           0
Gender                        0
User Behavior Class           0
dtype: int64
```

In [51]:
```
data.shape
```

Out[51]:
```
(700, 11)
```

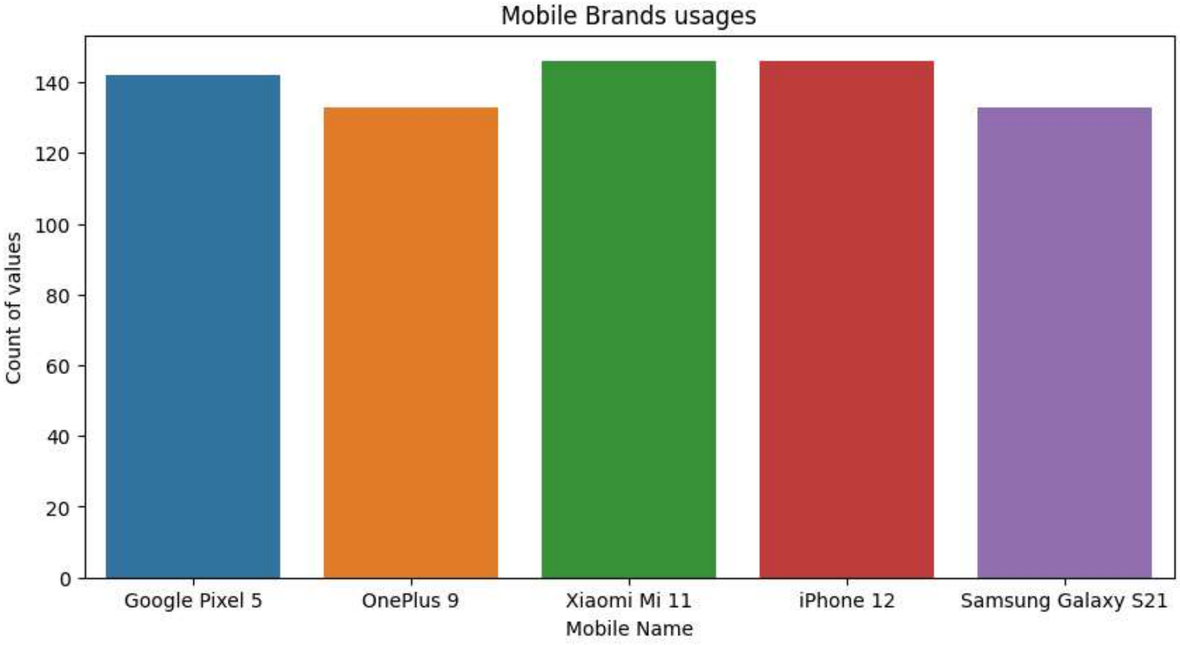
In [52]:
```
data.describe().style.background_gradient(cmap='winter_r')
```

Out[52]:

	User ID	App Usage Time (min/day)	Screen On Time (hours/day)	Battery Drain (mAh/day)	Number of Apps Installed	Data Usage (MB/day)	Age	
count	700.000000	700.000000	700.000000	700.000000	700.000000	700.000000	700.000000	7(
mean	350.500000	271.128571	5.272714	1525.158571	50.681429	929.742857	38.482857	
std	202.216880	177.199484	3.068584	819.136414	26.943324	640.451729	12.012916	
min	1.000000	30.000000	1.000000	302.000000	10.000000	102.000000	18.000000	
25%	175.750000	113.250000	2.500000	722.250000	26.000000	373.000000	28.000000	
50%	350.500000	227.500000	4.900000	1502.500000	49.000000	823.500000	38.000000	
75%	525.250000	434.250000	7.400000	2229.500000	74.000000	1341.000000	49.000000	
max	700.000000	598.000000	12.000000	2993.000000	99.000000	2497.000000	59.000000	

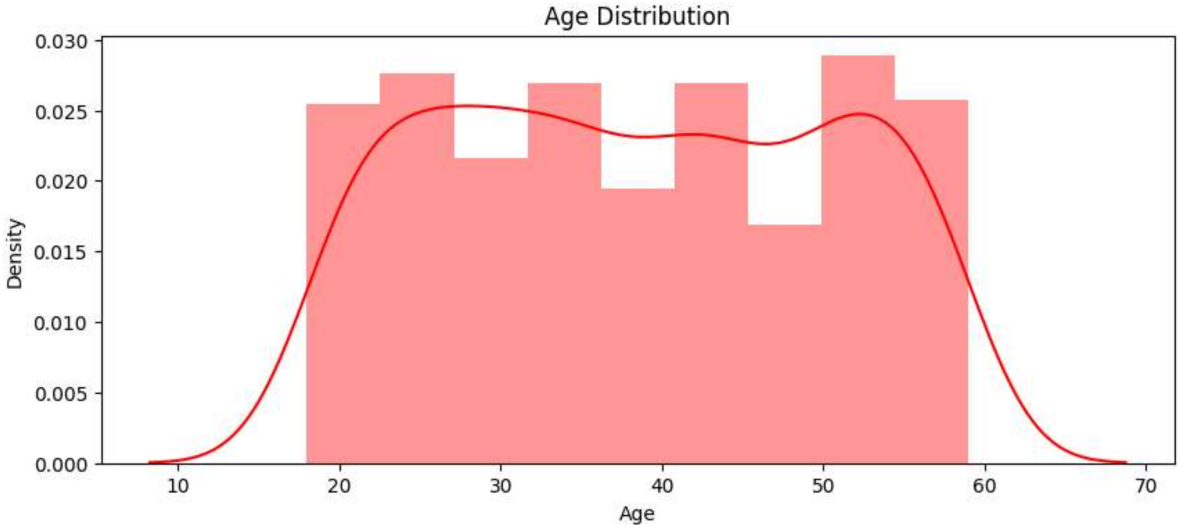
In [53]:

```
plt.figure(figsize=(10,5))
sns.countplot(data=data,x='Device Model')
plt.xlabel('Mobile Name')
plt.ylabel('Count of values')
plt.title('Mobile Brands usages')
plt.show()
```

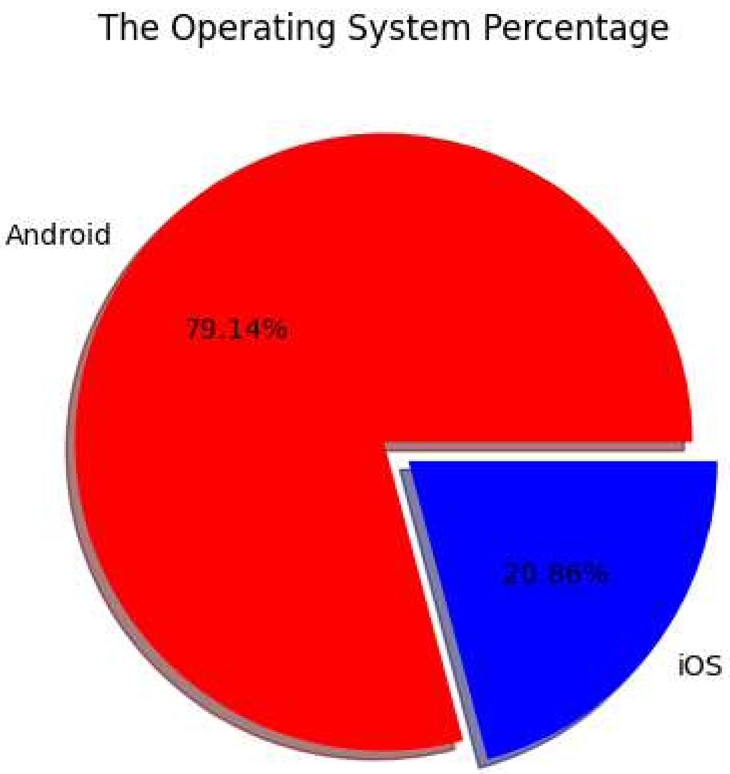


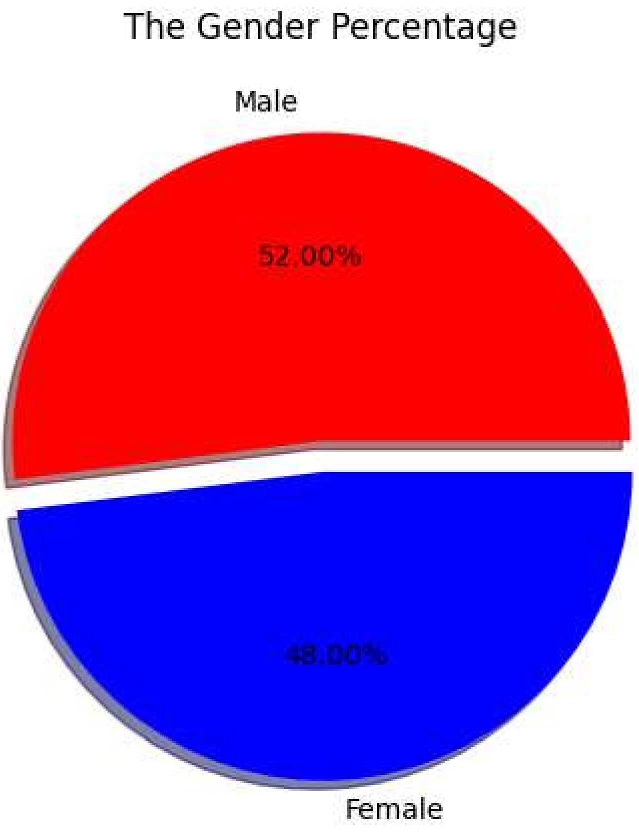
In [54]:

```
plt.figure(figsize=(10,4))
sns.distplot(data['Age'],kde=True,color='red')
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Density')
plt.show()
```

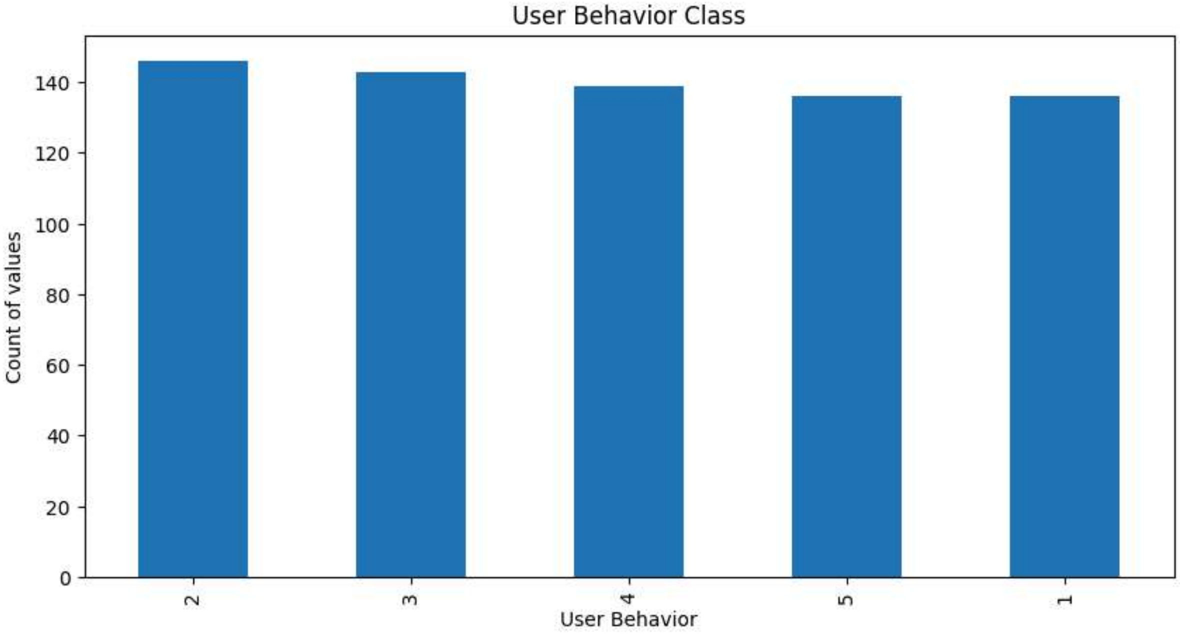


```
In [55]: for i in ['Operating System','Gender']:
             values=data[i].value_counts()
             plt.figure(figsize=(10,5))
             plt.pie(values,
             explode=[0,0.1],
             labels=values.index,
             colors=['red','blue'],
             autopct='%1.2f%%',
             shadow=True)
             plt.title(f'The {i} Percentage')
             plt.show()
```





In [56]:
```
data['User Behavior Class'].value_counts().sort_values(ascending=False)\
.plot(kind='bar',figsize=(10,5))
plt.title('User Behavior Class')
plt.xlabel('User Behavior')
plt.ylabel('Count of values')
plt.show()
```



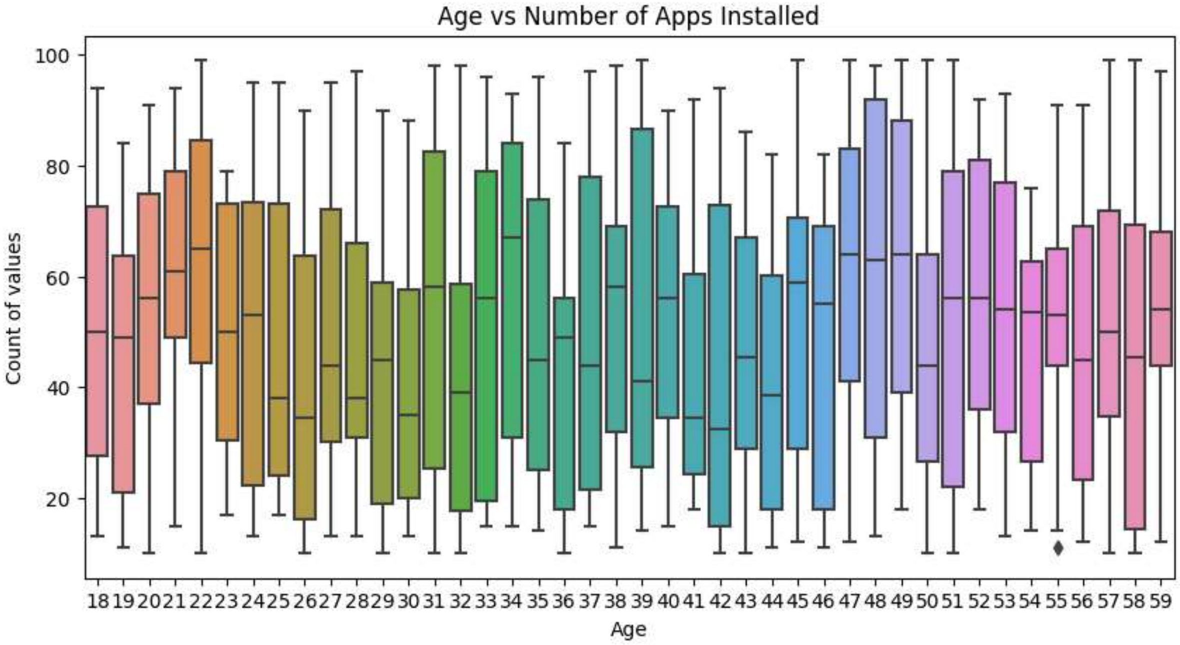
```
In [57]: maxtime=data['App Usage Time (min/day)'].max()
         average=data['App Usage Time (min/day)'].mean()
         print(f'The maximum time for the app usages {maxtime} min')
         print(f'The avarage time for the app usages {average} min')
```

```
The maximum time for the app usages 598 min
The avarage time for the app usages 271.12857142857143 min
```

```
In [58]: maxapps=data['Number of Apps Installed'].max()
         minimum=data['Number of Apps Installed'].mean()
         print(f'The maximum app in mobile is {maxapps}')
         print(f'The minimum apps intstalled in mobile is {minimum}')
```

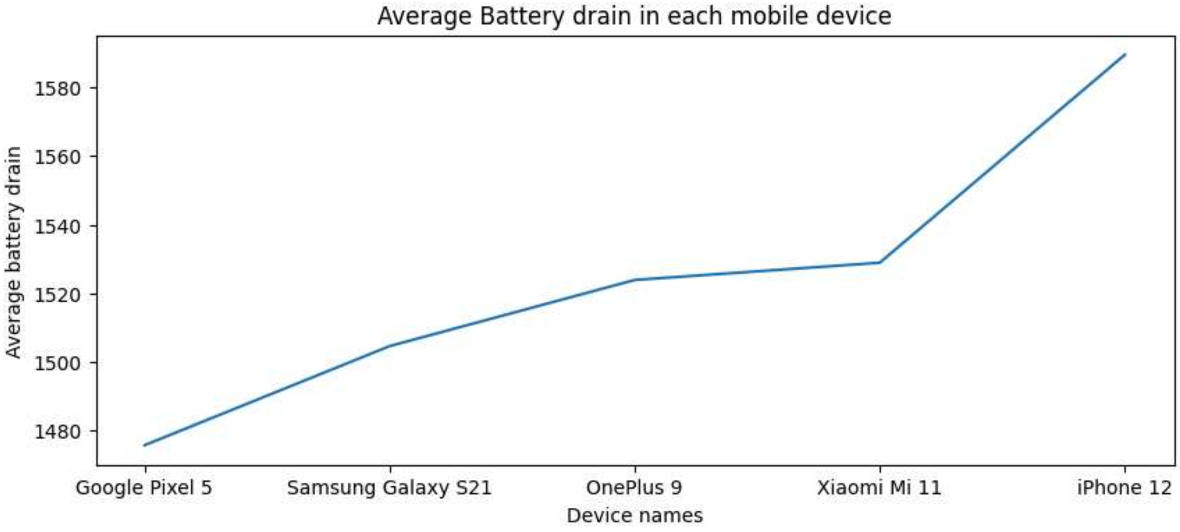
```
The maximum app in mobile is 99
The minimum apps intstalled in mobile is 50.68142857142857
```

```
In [59]: plt.figure(figsize=(10,5))
         sns.boxplot(data=data,x='Age',y='Number of Apps Installed')
         plt.title('Age vs Number of Apps Installed')
         plt.xlabel('Age')
         plt.ylabel('Count of values')
         plt.show()
```



In [60]:

```
battery_drain=data.groupby('Device Model')['Battery Drain (mAh/day)'].mean().s
plt.figure(figsize=(10,4))
sns.lineplot(data=battery_drain,x=battery_drain.index,y=battery_drain.values)
plt.title('Average Battery drain in each mobile device')
plt.xlabel('Device names')
plt.ylabel('Average battery drain')
plt.show()
```



In [61]:

```
operating=pd.DataFrame(data.groupby('Operating System')['Data Usage (MB/day)']
operating.style.background_gradient(cmap='Pastel2_r')
```

Out[61]:

	Data Usage (MB/day)
Operating System	
Android	920.317690
iOS	965.506849

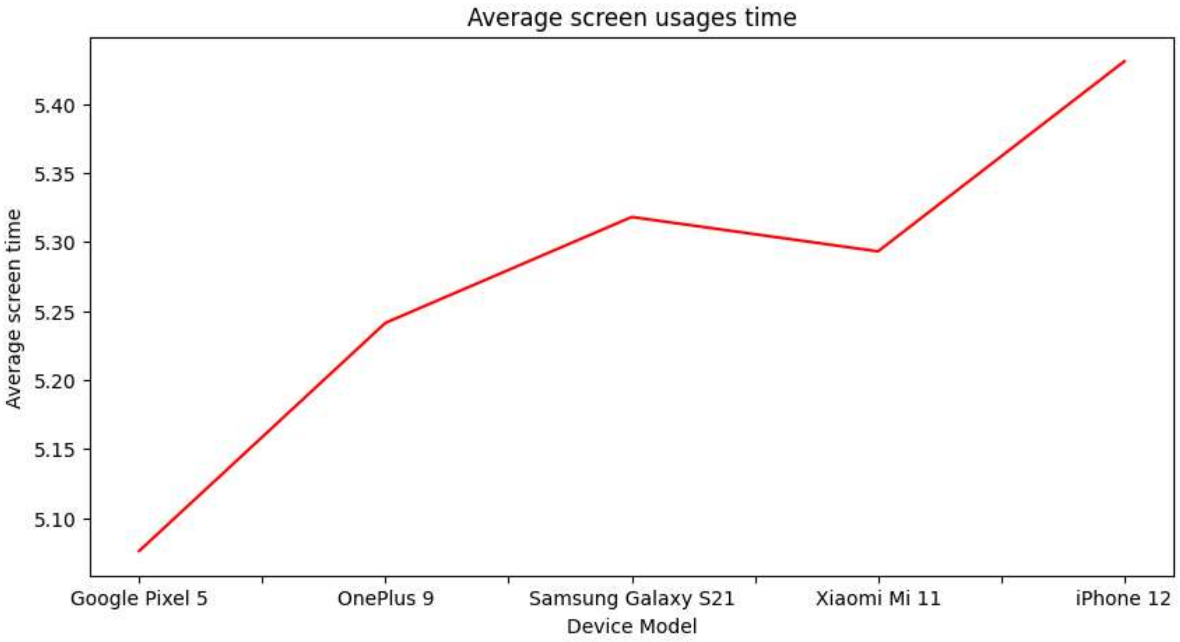
In [62]:

```
data.head(1)
```

Out[62]:

	User ID	Device Model	Operating System	App Usage Time (min/day)	Screen On Time (hours/day)	Battery Drain (mAh/day)	Number of Apps Installed	Data Usage (MB/day)	Age	Gender
0	1	Google Pixel 5	Android	393	6.4	1872	67	1122	40	Male

In [63]:
```
data.groupby('Device Model')['Screen On Time (hours/day)'].mean().plot(kind='l
plt.title('Average screen usages time')
plt.xlabel('Device Model')
plt.ylabel('Average screen time')
plt.show()
```



In [64]:

```
mobile_gender=pd.DataFrame(pd.pivot_table(data=data,values='Gender',columns='D
mobile_gender.style.background_gradient(cmap='gist_earth_r')
```

Out[64]:

Device Model	Google Pixel 5	OnePlus 9	Samsung Galaxy S21	Xiaomi Mi 11	iPhone 12
Age					
18	3.000000	nan	4.000000	3.000000	1.000000
19	1.000000	3.000000	1.000000	4.000000	3.000000
20	5.000000	2.000000	4.000000	1.000000	5.000000
21	5.000000	2.000000	2.000000	6.000000	2.000000
22	5.000000	3.000000	4.000000	4.000000	8.000000
23	3.000000	3.000000	3.000000	1.000000	5.000000
24	3.000000	6.000000	2.000000	3.000000	nan
25	7.000000	3.000000	4.000000	4.000000	3.000000
26	2.000000	4.000000	3.000000	3.000000	2.000000
27	3.000000	4.000000	5.000000	7.000000	5.000000
28	3.000000	3.000000	3.000000	4.000000	nan
29	2.000000	4.000000	4.000000	5.000000	6.000000
30	4.000000	3.000000	2.000000	2.000000	4.000000
31	3.000000	4.000000	3.000000	5.000000	5.000000
32	3.000000	4.000000	3.000000	5.000000	3.000000
33	6.000000	2.000000	2.000000	nan	1.000000
34	4.000000	3.000000	6.000000	5.000000	7.000000
35	5.000000	1.000000	1.000000	4.000000	4.000000
36	5.000000	4.000000	2.000000	4.000000	2.000000
37	2.000000	5.000000	3.000000	5.000000	4.000000
38	3.000000	nan	1.000000	3.000000	2.000000
39	2.000000	2.000000	5.000000	1.000000	5.000000
40	5.000000	3.000000	2.000000	4.000000	5.000000
41	2.000000	7.000000	1.000000	1.000000	1.000000
42	2.000000	4.000000	4.000000	6.000000	4.000000
43	4.000000	4.000000	2.000000	7.000000	5.000000
44	2.000000	2.000000	4.000000	4.000000	2.000000
45	4.000000	2.000000	8.000000	1.000000	3.000000
46	1.000000	3.000000	3.000000	1.000000	3.000000
47	3.000000	8.000000	2.000000	2.000000	2.000000
48	nan	nan	2.000000	4.000000	3.000000
49	5.000000	3.000000	2.000000	6.000000	1.000000
50	4.000000	2.000000	2.000000	7.000000	nan
51	5.000000	7.000000	5.000000	2.000000	6.000000
52	4.000000	3.000000	4.000000	4.000000	2.000000

Device Model	Google Pixel 5	OnePlus 9	Samsung Galaxy S21	Xiaomi Mi 11	iPhone 12
Age					
53	5.000000	2.000000	2.000000	3.000000	7.000000
54	3.000000	4.000000	5.000000	4.000000	nan
55	4.000000	4.000000	5.000000	3.000000	5.000000
56	3.000000	2.000000	1.000000	2.000000	8.000000
57	3.000000	4.000000	5.000000	1.000000	5.000000
58	3.000000	3.000000	4.000000	1.000000	3.000000
59	1.000000	1.000000	3.000000	4.000000	4.000000

In [65]:

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score,classification_report,confusion_matr
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

In [66]:

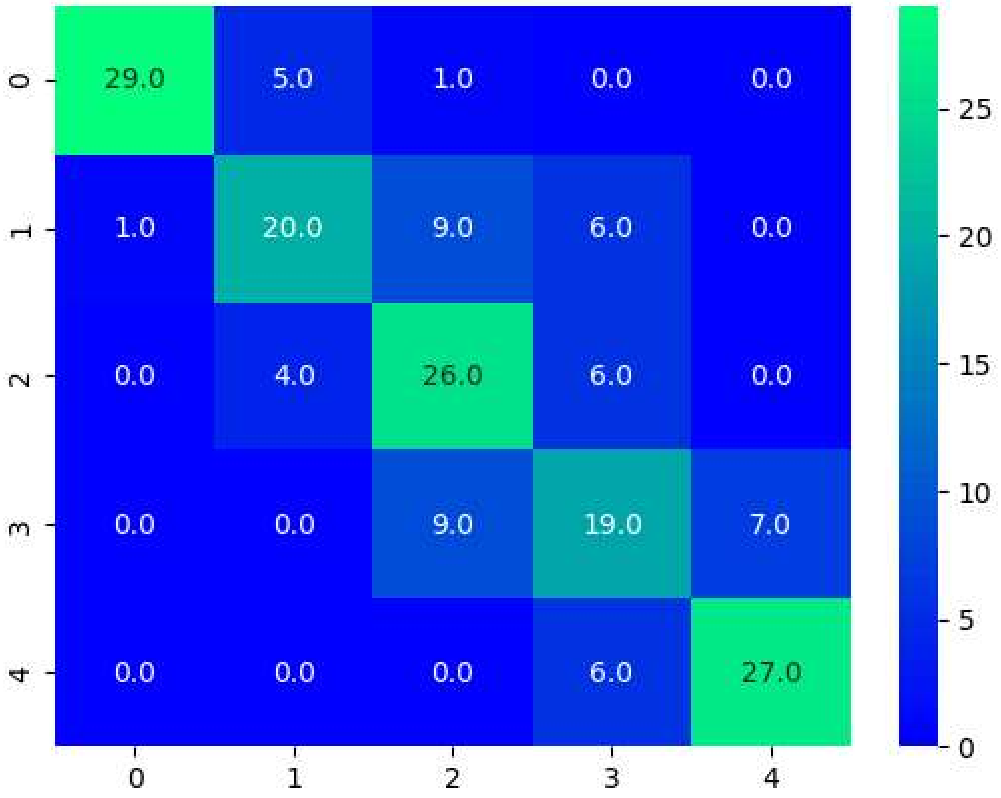
```

label=LabelEncoder()
data['Device Model']=label.fit_transform(data['Device Model'])
data['Operating System']=label.fit_transform(data['Operating System'])
data['Gender']=label.fit_transform(data['Gender'])
X=data.drop('User Behavior Class',axis=1)
y=data['User Behavior Class']
X.head(1)
```

Out[66]:

	User ID	Device Model	Operating System	App Usage Time (min/day)	Screen On Time (hours/day)	Battery Drain (mAh/day)	Number of Apps Installed	Data Usage (MB/day)	Age	Gender
0	1	0	0	393	6.4	1872	67	1122	40	1

In [79]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25,random_state
liner=LogisticRegression()
liner.fit(X_train,y_train)
liner_pred=liner.predict(X_test)
accuracy_score(y_test,liner_pred)
classification=confusion_matrix(y_test,liner_pred)
sns.heatmap(classification,annot=True, fmt='.1f',cmap='winter')
plt.show()
```



In [80]:
```
def model_selection(models,X_train,X_test,y_train,y_test):
        print(f'{models} Name')
        models.fit(X_train,y_train)
        model_pred=models.predict(X_test)
        print(f'The model accuracy Score {accuracy_score(y_test,model_pred)*10
        print(classification_report(y_test,model_pred))
        classification=confusion_matrix(y_test,model_pred)
        sns.heatmap(classification,annot=True, fmt='.2f',cmap='winter')
        plt.show()
```

In [81]:
```
models={
   'liner':LogisticRegression(),
   'decison':DecisionTreeClassifier(criterion='log_loss',max_depth=5),
   'random':RandomForestClassifier( n_estimators=50,criterion='entropy')
}
for i in range(len(models)):
    model_names=list(models.values())[i]
    feature=list(models.keys())[i]
    model_selection(model_names,X_train,X_test,y_train,y_test)
```

```
LogisticRegression() Name
The model accuracy Score 69.14
              precision    recall  f1-score   support

           1       0.97      0.83      0.89        35
           2       0.69      0.56      0.62        36
           3       0.58      0.72      0.64        36
           4       0.51      0.54      0.53        35
           5       0.79      0.82      0.81        33

    accuracy                           0.69       175
   macro avg       0.71      0.69      0.70       175
weighted avg       0.71      0.69      0.69       175
```

0
29.00
5.00
1.00
0.00
0.00
25

In [ ]: