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

In [2]:
```python
df = pd.read_csv('5G.csv')
```

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

Out[3]:

	Timestamp	User_ID	Application_Type	Signal_Strength	Latency	Required_Bandwidth	Allocated_Bandwidth
0	9/3/2023 10:00	User_1	Video_Call	-75 dBm	30 ms	10 Mbps	15 Mbps
1	9/3/2023 10:00	User_2	Voice_Call	-80 dBm	20 ms	100 Kbps	120 Kbps
2	9/3/2023 10:00	User_3	Streaming	-85 dBm	40 ms	5 Mbps	6 Mbps
3	9/3/2023 10:00	User_4	Emergency_Service	-70 dBm	10 ms	1 Mbps	1.5 Mbps
4	9/3/2023 10:00	User_5	Online_Gaming	-78 dBm	25 ms	2 Mbps	3 Mbps

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   Timestamp            400 non-null    object
 1   User_ID              400 non-null    object
 2   Application_Type     400 non-null    object
 3   Signal_Strength      400 non-null    object
 4   Latency              400 non-null    object
 5   Required_Bandwidth   400 non-null    object
 6   Allocated_Bandwidth  400 non-null    object
 7   Resource_Allocation  400 non-null    object
dtypes: object(8)
memory usage: 25.1+ KB
```

In [5]:
```python
df.describe()
```

Out[5]:

	Timestamp	User_ID	Application_Type	Signal_Strength	Latency	Required_Bandwidth	Allocated_Bandw
count	400	400	400	400	400	400	
unique	7	400	11	84	87	188	
top	9/3/2023 10:01	User_1	Video_Call	-97 dBm	5 ms	0.1 Mbps	0.1 M
freq	60	1	58	9	35	16	

In [6]:
```python
df['Application_Type'].value_counts()
```

Out[6]:
```
Application_Type
Video_Call             58
Web_Browsing           48
Streaming              47
Emergency_Service      47
Background_Download    47
Video_Streaming        47
VoIP_Call              46
Online_Gaming          45
IoT_Temperature        13
Voice_Call              1
File_Download           1
Name: count, dtype: int64
```

In [7]:
```python
df.drop('User_ID', inplace=True, axis=1)
```

In [8]:
```python
df['Latency'] = df['Latency'].str.extract('(\d+)').astype(int)
```

In [9]:
```python
df['Signal_Strength'] = df['Signal_Strength'].str.extract('(-?\d+)').astype(int)
```

In [10]:
```python
df
```

Out[10]:

	Timestamp	Application_Type	Signal_Strength	Latency	Required_Bandwidth	Allocated_Bandwidth	Resou
0	9/3/2023 10:00	Video_Call	-75	30	10 Mbps	15 Mbps	
1	9/3/2023 10:00	Voice_Call	-80	20	100 Kbps	120 Kbps	
2	9/3/2023 10:00	Streaming	-85	40	5 Mbps	6 Mbps	
3	9/3/2023 10:00	Emergency_Service	-70	10	1 Mbps	1.5 Mbps	
4	9/3/2023 10:00	Online_Gaming	-78	25	2 Mbps	3 Mbps	
...	...	...	...	...	...	...	
395	9/3/2023 10:06	Streaming	-110	61	1.3 Mbps	1.8 Mbps	
396	9/3/2023 10:06	Video_Call	-40	53	14.5 Mbps	15.8 Mbps	
397	9/3/2023 10:06	Video_Streaming	-113	58	1.0 Mbps	1.4 Mbps	
398	9/3/2023 10:06	Emergency_Service	-40	5	0.4 Mbps	0.4 Mbps	
399	9/3/2023 10:06	Web_Browsing	-113	0	0.1 Mbps	0.1 Mbps	

400 rows × 7 columns

In [11]:
```python
df['Required_Bandwidth'] = df['Required_Bandwidth'].str.extract('(\d+)').astype(int)
```

In [12]:
```python
df['Allocated_Bandwidth'] = df['Allocated_Bandwidth'].str.extract('(\d+)').astype(int)
```

In [13]:
```python
df
```

Out[13]:

	Timestamp	Application_Type	Signal_Strength	Latency	Required_Bandwidth	Allocated_Bandwidth	Resou
0	9/3/2023 10:00	Video_Call	-75	30	10	15	
1	9/3/2023 10:00	Voice_Call	-80	20	100	120	
2	9/3/2023 10:00	Streaming	-85	40	5	6	
3	9/3/2023 10:00	Emergency_Service	-70	10	1	1	
4	9/3/2023 10:00	Online_Gaming	-78	25	2	3	
...	...	...	...	...	...	...	
395	9/3/2023 10:06	Streaming	-110	61	1	1	
396	9/3/2023 10:06	Video_Call	-40	53	14	15	
397	9/3/2023 10:06	Video_Streaming	-113	58	1	1	
398	9/3/2023 10:06	Emergency_Service	-40	5	0	0	
399	9/3/2023 10:06	Web_Browsing	-113	0	0	0	

400 rows × 7 columns

In [14]:
```python
df['Resource_Allocation'] = df['Resource_Allocation'].str.rstrip('%').astype(float)
```

In [15]:
```python
df
```

Out[15]:

	Timestamp	Application_Type	Signal_Strength	Latency	Required_Bandwidth	Allocated_Bandwidth	Resou
0	9/3/2023 10:00	Video_Call	-75	30	10	15	
1	9/3/2023 10:00	Voice_Call	-80	20	100	120	
2	9/3/2023 10:00	Streaming	-85	40	5	6	
3	9/3/2023 10:00	Emergency_Service	-70	10	1	1	
4	9/3/2023 10:00	Online_Gaming	-78	25	2	3	
...	...	...	...	...	...	...	
395	9/3/2023 10:06	Streaming	-110	61	1	1	
396	9/3/2023 10:06	Video_Call	-40	53	14	15	
397	9/3/2023 10:06	Video_Streaming	-113	58	1	1	
398	9/3/2023 10:06	Emergency_Service	-40	5	0	0	
399	9/3/2023 10:06	Web_Browsing	-113	0	0	0	

400 rows × 7 columns

In [16]:
```python
from sklearn.preprocessing import LabelEncoder
label = LabelEncoder()
```

In [17]:
```python
df['Application_Type'] = label.fit_transform(df['Application_Type'])
```

In [18]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 7 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Timestamp            400 non-null    object 
 1   Application_Type     400 non-null    int32  
 2   Signal_Strength      400 non-null    int32  
 3   Latency              400 non-null    int32  
 4   Required_Bandwidth   400 non-null    int32  
 5   Allocated_Bandwidth  400 non-null    int32  
 6   Resource_Allocation  400 non-null    float64
dtypes: float64(1), int32(5), object(1)
memory usage: 14.2+ KB
```

In [19]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
```

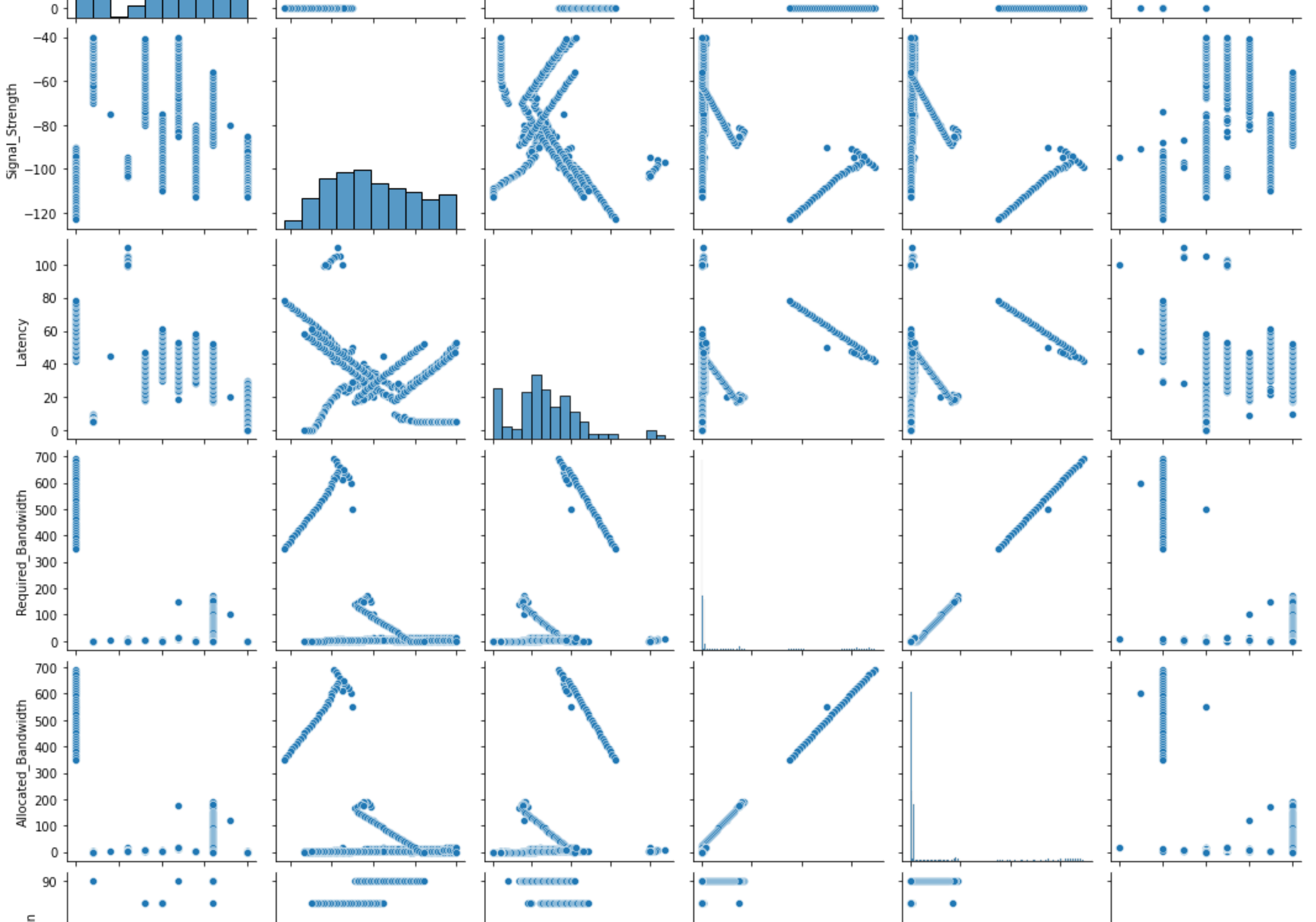
In [20]:
```python
df.drop('Timestamp', inplace=True, axis=1)
```

In [21]:
```python
sns.heatmap(df.corr(), annot=True, fmt='.2f')
```

Out[21]: <AxesSubplot:>

In [22]:
```python
sns.pairplot(df)
```

Out[22]: <seaborn.axisgrid.PairGrid at 0x1f901986f40>



In [23]:
```python
from sklearn.linear_model import LinearRegression as LR
lr = LR()
```

In [24]:
```python
from sklearn.model_selection import train_test_split as tt
X = df.drop('Resource_Allocation',axis=1)
y = df['Resource_Allocation']
```

In [25]:
```python
X_train,X_test,y_train,y_test = tt(X,y,test_size=0.2, random_state=0)
```

In [26]:
```python
X_train.shape
```

Out[26]: (320, 5)

In [27]:
```python
X_test.shape
```

Out[27]: (80, 5)

In [28]:
```python
y_train.shape
```

Out[28]: (320,)

In [29]:
```python
lr.fit(X_train,y_train)
```

Out[29]: LinearRegression()

In [30]:
```python
y_pred = lr.predict(X_test)
```

In [31]:
```python
from sklearn.metrics import r2_score
r2_score(y_pred, y_test)
```

Out[31]: 0.06613277337583978

In [32]:
```python
from sklearn.ensemble import RandomForestRegressor as RF
rf = RF(n_estimators=100, random_state=42)
```

In [33]:
```python
rf.fit(X_train,y_train)
```

Out[33]: RandomForestRegressor(random_state=42)

In [34]:
```python
y_rf = rf.predict(X_test)
```

In [35]:
```python
r2_score(y_rf,y_test)
```

Out[35]: 0.800333998377096

In [36]:
```python
from sklearn.tree import DecisionTreeRegressor as DTR
```

In [37]:
```python
dtr = DTR()
```

In [38]:
```python
dtr.fit(X_train,y_train)
```

Out[38]: DecisionTreeRegressor()

In [39]:
```python
y_dtr = dtr.predict(X_test)
```

In [40]:
```python
r2_score(y_dtr, y_test)
```

Out[40]: 0.7188613886239269

In [41]:
```python
from sklearn.neighbors import KNeighborsRegressor as KNN
```

In [42]:
```python
knn = KNN()
```

In [43]:
```python
knn.fit(X_train, y_train)
```

Out[43]: KNeighborsRegressor()

In [44]:
```python
y_knn = knn.predict(X_test)
```

In [45]:
```python
r2_score(y_knn, y_test)
```

Out[45]: 0.7927597117429814

RANDOM FOREST REGRESSOR

In [50]:
```python
import pickle
```

In [52]:
```python
pickle.dump(rf, open('5g.pkl', 'wb'))
```

In [ ]: