bubble sort

In [5]:
```
import numpy as np
def bubble_sort(a):
    n=len(a)
    for i in range(n):
        for j in range(n-i-1):
            if (a[j] > a[j+1]):
                a[j],a[j+1]=a[j+1],a[j]

a=np.array([55,22,11,2,7])
bubble_sort(a)
print(a)
```

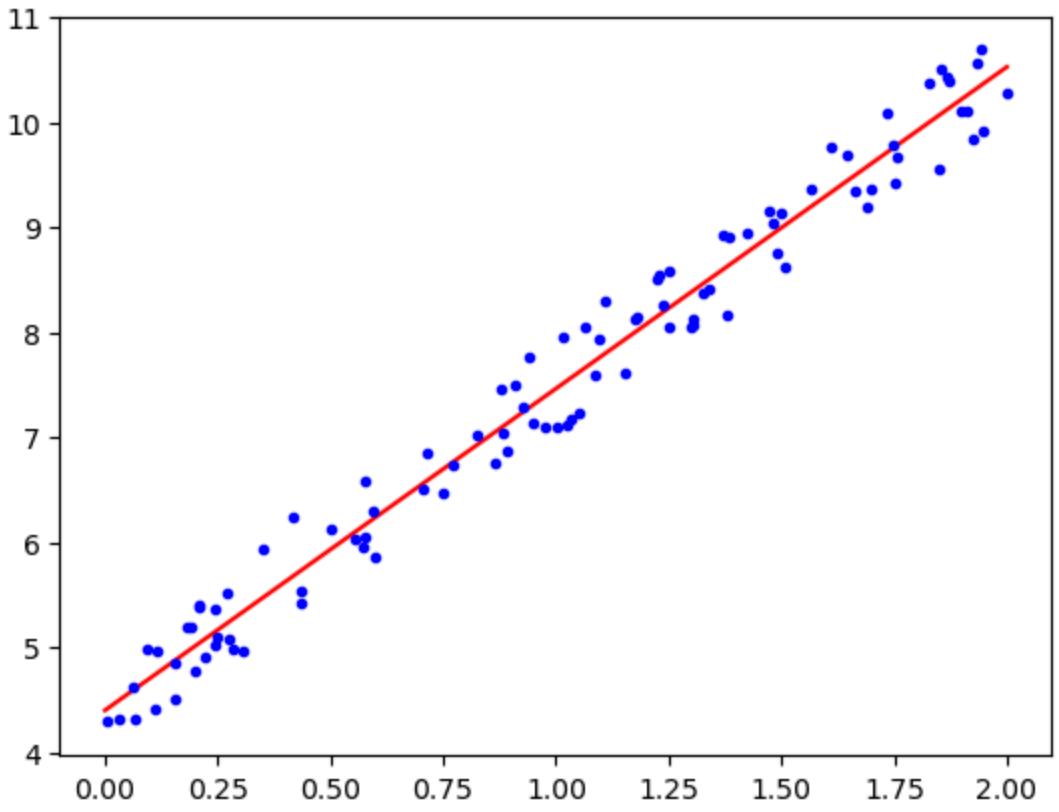
```
[ 2  7 11 22 55]
```

Linear Regression

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

X=2* np.random.rand(100,1)
y=4+3*X+np.random.rand(100,1)

model=LinearRegression().fit(X,y)
X_new=np.array([[0],[2]])
print(model.predict(X_new))

plt.plot(X_new, model.predict(X_new), 'r-')
plt.plot(X, y, 'b.')
```

```
[[ 4.40462139]
 [10.52463982]]
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x1e31342f890>]
```



KMEAN

In [19]:
```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans

df=pd.read_csv('KMeans.csv')
X=df[['X1','X2']]
model=KMeans(n_clusters=3,random_state=0).fit(X)
print(model.predict([[1.713, 1.586]]))
```

```
[0]
C:\Users\safwa\miniconda3\envs\ml_env\Lib\site-packages\sklearn\cluster\_kmeans.py:1419: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\safwa\miniconda3\envs\ml_env\Lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but KMeans was fitted with feature names
  warnings.warn(
```

In [ ]:

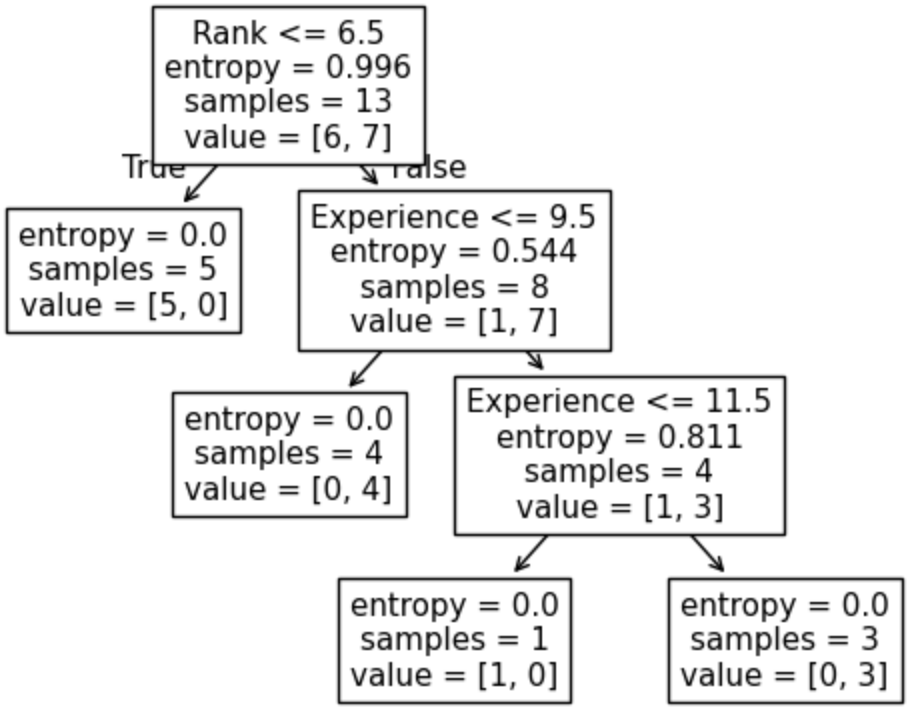
Decision Tree Classifier

In [31]:
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree

df = pd.read_csv("decision.csv")

df['Nationality'] = df['Nationality'].map({'UK': 0, 'USA': 1, 'N': 2})
df['Go'] = df['Go'].map({'NO': 0, 'YES': 1})

df = df.dropna(subset=['Go'])  # ✅ This line fixes the error

X = df[['Age', 'Experience', 'Rank', 'Nationality']]
y = df['Go']

model = DecisionTreeClassifier(criterion='entropy').fit(X, y)
print(model.predict([[40, 10, 7, 1]]))
plot_tree(model, feature_names=['Age', 'Experience', 'Rank', 'Nationality'])
```

```
[0]
```

```
C:\Users\safwa\miniconda3\envs\ml_env\Lib\site-packages\sklearn\utils\validation.py:2739: UserWarning: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature names
  warnings.warn(
```

Out[31]:
```
[Text(0.3333333333333333, 0.875, 'Rank <= 6.5\nentropy = 0.996\nsamples = 13\nvalue = [6, 7]'),
 Text(0.16666666666666666, 0.625, 'entropy = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(0.25, 0.75, 'True  '),
 Text(0.5, 0.625, 'Experience <= 9.5\nentropy = 0.544\nsamples = 8\nvalue = [1, 7]'),
 Text(0.41666666666666663, 0.75, '  False'),
 Text(0.3333333333333333, 0.375, 'entropy = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.6666666666666666, 0.375, 'Experience <= 11.5\nentropy = 0.811\nsamples = 4\nvalue = [1, 3]'),
 Text(0.5, 0.125, 'entropy = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.8333333333333334, 0.125, 'entropy = 0.0\nsamples = 3\nvalue = [0, 3]')]
```



In [ ]:

fuzzy set

In [32]:
```
import numpy as np
!pip install scikit-fuzzy
import skfuzzy as fuzz

np.random.seed(0)
data = np.random.rand(100, 2)

a, b, c, d, e, f, g = fuzz.cluster.cmeans(
    data.T,     # Transpose: shape must be (features, samples)
    3,          # Number of clusters
    2,          # Fuzziness
    error=0.005,
    maxiter=1000,
    init=None
)

print(a)
print(np.argmax(b, axis=0))
```

```
Collecting scikit-fuzzy
  Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl.metadata (2.6 kB)
Downloading scikit_fuzzy-0.5.0-py2.py3-none-any.whl (920 kB)
   ---------------------------------------- 0.0/920.8 kB ? eta -:--:--
   ---------------------------------- ----- 786.4/920.8 kB 5.4 MB/s eta 0:00:01
   ---------------------------------------- 920.8/920.8 kB 5.4 MB/s eta 0:00:00
Installing collected packages: scikit-fuzzy
Successfully installed scikit-fuzzy-0.5.0
[[0.22645397 0.71840176]
 [0.52083891 0.18668653]
 [0.76252289 0.60239021]]
[2 2 0 0 2 2 2 1 0 2 2 0 0 0 1 0 0 0 2 2 1 1 2 1 1 2 1 1 1 1 1 1 0 1 1 2 2
 1 1 1 1 0 1 1 2 0 0 1 1 1 1 2 0 2 0 0 1 2 2 2 2 2 0 0 1 2 1 2 2 2 2 0 2 0
 2 0 0 0 2 1 2 2 2 0 1 1 1 1 0 1 0 1 2 2 1 1 0 2 1 0]
```

In [37]:
```
import numpy as np
import skfuzzy as fuzz

data = np.random.rand(100, 2)
_, u, _, _, _, _, _ = fuzz.cluster.cmeans(data.T, 3, 2, 0.005, 1000)

print("Cluster Labels:", np.argmax(u, axis=0))
```

```
Cluster Labels: [1 0 1 1 1 0 0 2 1 1 1 1 1 0 1 1 2 1 1 2 1 0 0 1 1 1 2 0 0 1 1
0 2 0 0 1 0
 0 0 1 1 1 1 2 2 1 0 0 1 2 2 1 2 2 1 2 0 1 2 0 0 1 1 1 1 1 0 0 2 1 0 0 1 2
 2 0 0 2 2 1 2 2 2 1 0 1 0 1 2 0 1 2 0 0 2 2 2 0 0 1]
```

Gradient boosting classifier

In [42]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=10, n_features=5, n_informative=2, n_redu

model = GradientBoostingClassifier().fit(X, y)
print(model.predict([[0.19, 1.05, -0.72, -1.14, 1.44]]))
```

```
[0]
```

In [41]:
```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=10, n_features=5, n_informative=2, n_redu

model = GradientBoostingClassifier().fit(X, y)
print(model.predict([[0.19, 1.05, -0.72, -1.14, 1.44]]))
```

```
[0]
```

In [ ]:
```

```