3a. MinMax Algorithm

In [1]:
```
def minmax(depth, nodeIndex, maximizingPlayer, values, path):
    if depth == 3:
        return values[nodeIndex], path + [nodeIndex]

    if maximizingPlayer:
        best = float('-inf')
        best_path = []
        for i in range(2):
            val, new_path = minmax(depth + 1, nodeIndex * 2 + i, False, values, path + [nodeIndex])
            if val > best:
                best = val
                best_path = new_path
        return best, best_path
    else:
        best = float('inf')
        best_path = []
        for i in range(2):
            val, new_path = minmax(depth + 1, nodeIndex * 2 + i, True, values, path + [nodeIndex])
            if val < best:
                best = val
                best_path = new_path
        return best, best_path
```

In [2]:
```
# Example tree with depth 3 and 8 terminal nodes
values = [3, 5, 2, 9, 12, 5, 23, 23]

# Start the Min-Max algorithm
optimal_value, optimal_path = minmax(0, 0, True, values, [])
print("The optimal value is:", optimal_value)
print("The path taken is:", optimal_path)
```

```
The optimal value is: 12
The path taken is: [0, 1, 2, 4]
```

3b. Heat Map

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [5]:
```
data=pd.read_csv("./corolla.csv")
plt.figure(figsize = ( 5 , 3 ))
sns.heatmap(data[["Price","KM","Doors", "Weight"]].corr(),cmap='jet')
plt.show()
```