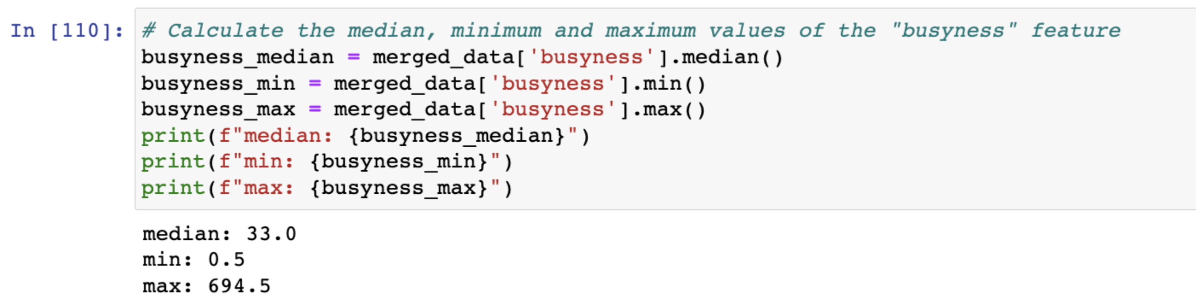
Busyness Classification:

1. Calculate the median, minimum and maximum values of the "busyness" feature:

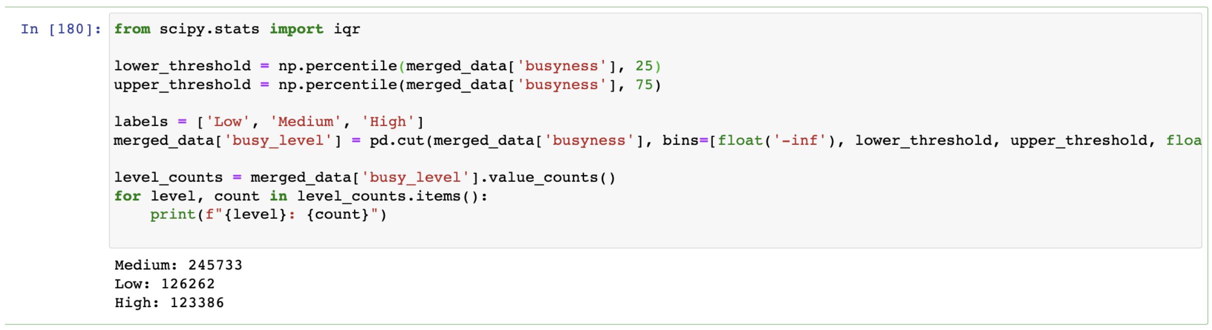


```
In [110]: # Calculate the median, minimum and maximum values of the "busyness" feature
          busyness_median = merged_data['busyness'].median()
          busyness_min = merged_data['busyness'].min()
          busyness_max = merged_data['busyness'].max()
          print(f"median: {busyness_median}")
          print(f"min: {busyness_min}")
          print(f"max: {busyness_max}")

          median: 33.0
          min: 0.5
          max: 694.5
```

Code:
# Calculate the median, minimum and maximum values of the "busyness" feature
busyness_median = merged_data['busyness'].median()
busyness_min = merged_data['busyness'].min()
busyness_max = merged_data['busyness'].max()
print(f"median: {busyness_median}")
print(f"min: {busyness_min}")
print(f"max: {busyness_max}")

2. Defining lower and upper thresholds:
   determine the values at the 25th and 75th percentiles of the 'busyness' column in the 'merged_data' DataFrame. The 25th percentile corresponds to the lower boundary, and the 75th percentile corresponds to the upper boundary.



```
In [180]: from scipy.stats import iqr

          lower_threshold = np.percentile(merged_data['busyness'], 25)
          upper_threshold = np.percentile(merged_data['busyness'], 75)

          labels = ['Low', 'Medium', 'High']
          merged_data['busy_level'] = pd.cut(merged_data['busyness'], bins=[float('-inf'), lower_threshold, upper_threshold, floa

          level_counts = merged_data['busy_level'].value_counts()
          for level, count in level_counts.items():
              print(f"{level}: {count}")

          Medium: 245733
          Low: 126262
          High: 123386
```

Code:

from scipy.stats import iqr

lower_threshold = np.percentile(merged_data['busyness'], 25)
upper_threshold = np.percentile(merged_data['busyness'], 75)

labels = ['Low', 'Medium', 'High']
merged_data['busy_level'] = pd.cut(merged_data['busyness'], bins=[float('-inf'), lower_threshold, upper_threshold, float('inf')], labels=labels, include_lowest=True)

```
level_counts = merged_data['busy_level'].value_counts()
for level, count in level_counts.items():
    print(f"{level}: {count}")
```

3.