Smart parking using IoT

Smart parking:

- Smart parking is a system that uses the Internet of Things (IoT) to collect and analyze data about parking availability and occupancy. This data can then be used to improve the efficiency and convenience of parking for both drivers and parking lot operators.
- One of the key benefits of smart parking is that it can help drivers to find parking spaces more quickly and easily. This is done by providing drivers with real-time information about parking availability in nearby parking lots. Drivers can access this information through a mobile app or website, or through signs that are posted in the parking lot itself.

Step 1:Import the necessary libraries

Python code to import necessary libraries,

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Step 2: Load the dataset

Python code,

```
# Load the dataset
df = pd.read_csv('smart_parking_dataset.csv')
```

Step 3: Data cleaning

The goal of data cleaning is to remove any errors or inconsistencies from the dataset. This may involve:

- Checking for missing values: We can use the *isna()* function in Pandas to check for missing values in each column of the dataset. If we find any missing values, we can either drop the rows with missing values or impute the missing values with a reasonable value.
- Dropping duplicate rows: We can use the *duplicated()* function in Pandas to check for duplicate rows in the dataset. If we find any duplicate rows, we can drop them from the dataset.
- Converting the data types to the appropriate format: We should check the data types of each column in the dataset and convert them to the appropriate format if necessary. For example, we may want to convert the *datetime* column to a datetime object.

  Python code,

```
# Check for missing values
df.isna().sum()

# Drop any rows with missing values
df = df.dropna()

# Check for duplicate rows
df.duplicated().sum()

# Drop any duplicate rows
df = df.drop_duplicates()

# Convert the data types to the appropriate format
df['datetime'] = pd.to_datetime(df['datetime'])
df['parking_status'] = df['parking_status'].astype('category')
```

Step 4: Data analysis

Once the data has been cleaned, we can begin to analyze it. Here are some examples of data analysis that we can perform on a smart parking dataset:

- Get the number of parking slots available and occupied at each time interval: We can use the groupby() function in Pandas to group the data by

datetime and then use the value_counts() function to count the number of parking slots available and occupied at each time interval.

- Plot the number of parking slots available and occupied over time: We can use the plot() function in Matplotlib to plot the number of parking slots available and occupied over time. This can help us to identify peak parking times and trends in parking occupancy.
- Calculate the average parking occupancy rate: We can calculate the average parking occupancy rate by dividing the number of parking slots occupied by the total number of parking slots and then multiplying by 100.
- Identify the most popular parking areas: We can use the groupby() function in Pandas to group the data by parking_location and then use the value_counts() function to count the number of vehicles parked in each parking area. This can help us to identify the most popular parking areas.
- Analyze the relationship between parking occupancy rate and other factors: We can use correlation analysis to analyze the relationship between parking occupancy rate and other factors, such as weather, traffic conditions, and events. This can help us to understand how these factors impact parking demand.
- Use machine learning to predict future parking demand and recommend parking spots to drivers: We can use machine learning algorithms to predict future parking demand and recommend parking spots to drivers. This can help drivers to find parking more easily and reduce traffic congestion.

Python code,

```
# Get the number of parking slots available and occupied at each time interval
df_grouped = df.groupby('datetime')['parking_status'].value_counts()
df_grouped = df_grouped.unstack().fillna(0)

# Plot the number of parking slots available and occupied over time
plt.plot(df_grouped['available'], label='Available')
plt.plot(df_grouped['occupied'], label='Occupied')
plt.legend()
plt.xlabel('Datetime')
plt.ylabel('Number of parking slots')
plt.title('Number of parking slots available and occupied over time')
plt.show()
```

```
# Calculate the average parking occupancy rate
parking_occupancy_rate = df['parking_status'].value_counts()['occupied'] / len(df) * 100
print('Average parking occupancy rate:', parking_occupancy_rate)
```

This code will produce a plot of the number of parking slots available and occupied over time, as well as the average parking occupancy rate.

By performing data cleaning and analysis on a smart parking dataset, we can gain valuable insights into how parking resources are being used and identify areas for improvement. This information can be used to develop more efficient and sustainable parking solutions.