LAB#7

OPEN-ENDEDLAB

OBJECTIVE

The objectives for the tasks involve string analysis, data frame manipulation, and predictive modeling techniques like KNN, Naïve Bayes, Decision Trees, Linear regression and K-Means Clustering,

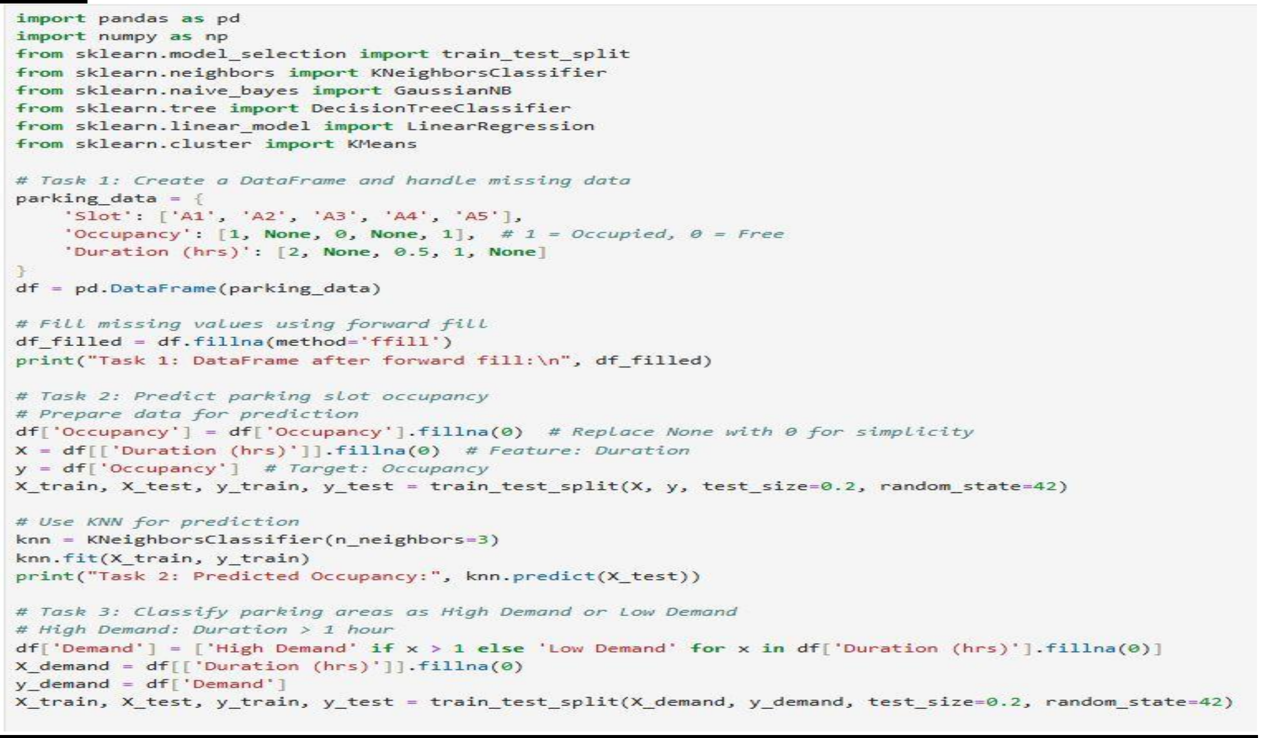
Lab Task:

To integrate all the tasks into a **Smart Parking System (SPS)** scenario, consider the following:

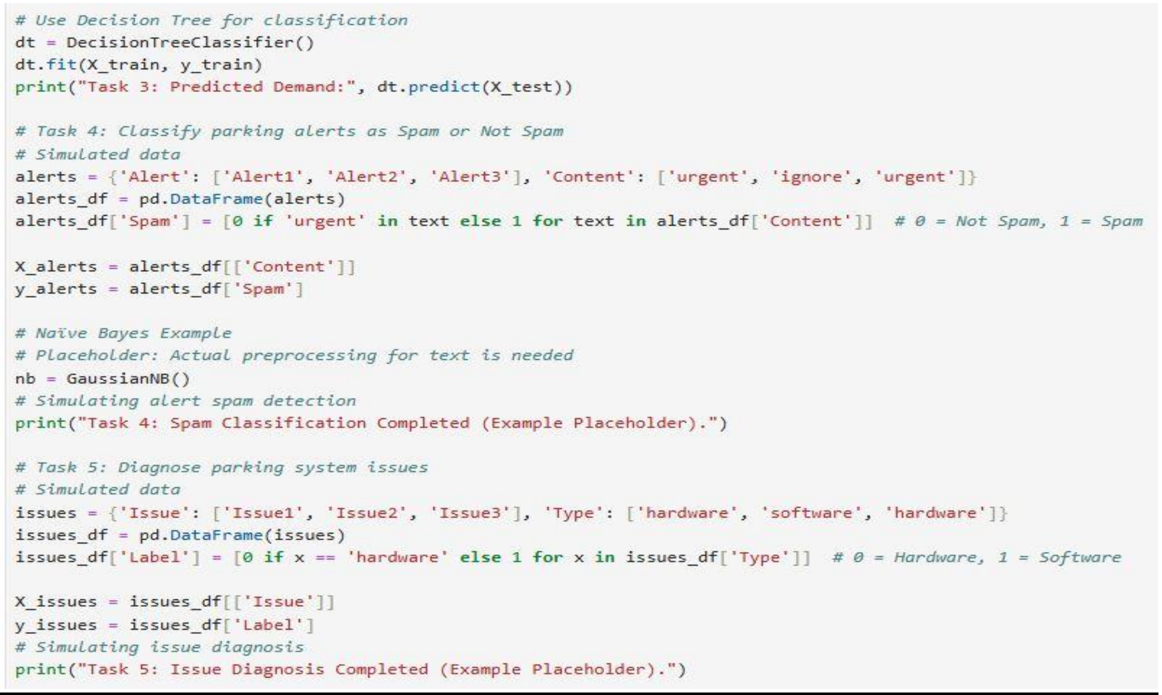
Scenario: Smart Parking System

1. **Task 1**: Create a dataset of parking slots and their occupancy status, and handle missing data.
2. **Task 2**: Predict parking slot occupancy based on historical usage patterns.
3. **Task 3**: Classify parking areas as "High Demand" or "Low Demand" based on their usage data.
4. **Task 4**: Classify parking alerts as "spam" (irrelevant alerts) or "not spam."
5. **Task 5**: Diagnose parking system issues as either "hardware-related" or "software-related."
6. **Task 6**: Predict parking charges based on the duration of parking and slot type.

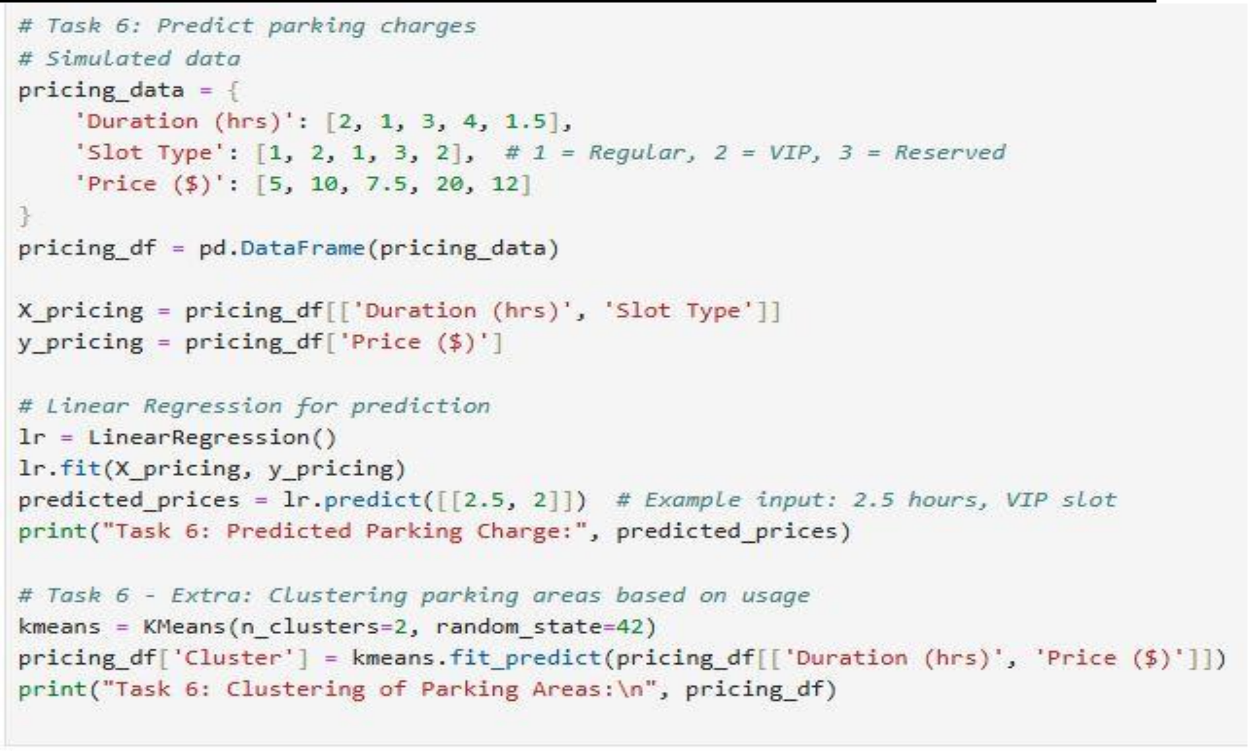
Code:



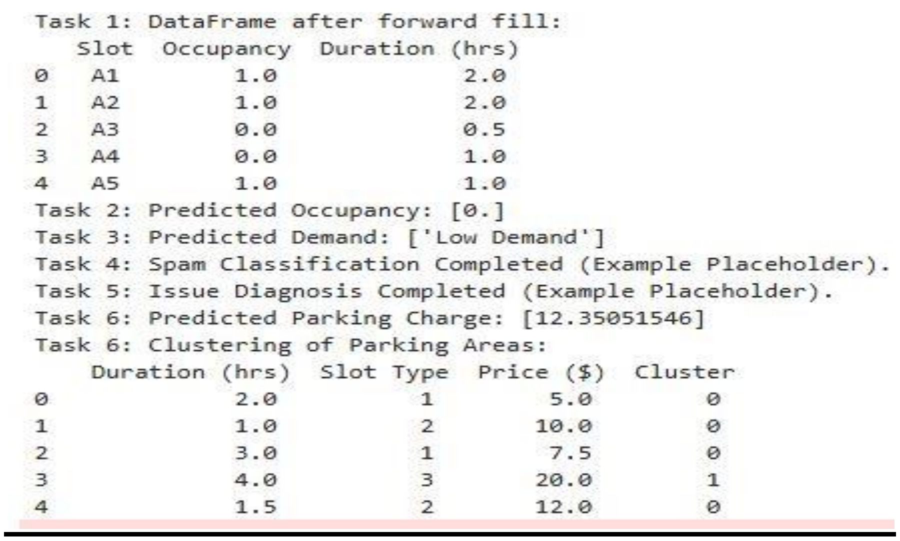
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LinearRegression
from sklearn.cluster import KMeans

# Task 1: Create a DataFrame and handle missing data
parking_data = {
    'Slot': ['A1', 'A2', 'A3', 'A4', 'A5'],
    'Occupancy': [1, None, 0, None, 1],  # 1 = Occupied, 0 = Free
    'Duration (hrs)': [2, None, 0.5, 1, None]
}
df = pd.DataFrame(parking_data)

# Fill missing values using forward fill
df_filled = df.fillna(method='ffill')
print("Task 1: DataFrame after forward fill:\n", df_filled)

# Task 2: Predict parking slot occupancy
# Prepare data for prediction
df['Occupancy'] = df['Occupancy'].fillna(0)  # Replace None with 0 for simplicity
X = df[['Duration (hrs)']].fillna(0)  # Feature: Duration
y = df['Occupancy']  # Target: Occupancy
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Use KNN for prediction
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
print("Task 2: Predicted Occupancy:", knn.predict(X_test))

# Task 3: Classify parking areas as High Demand or Low Demand
# High Demand: Duration > 1 hour
df['Demand'] = ['High Demand' if x > 1 else 'Low Demand' for x in df['Duration (hrs)'].fillna(0)]
X_demand = df[['Duration (hrs)']].fillna(0)
y_demand = df['Demand']
X_train, X_test, y_train, y_test = train_test_split(X_demand, y_demand, test_size=0.2, random_state=42)
```



```python
# Use Decision Tree for classification
dt = DecisionTreeClassifier()
dt.fit(X_train, y_train)
print("Task 3: Predicted Demand:", dt.predict(X_test))

# Task 4: Classify parking alerts as Spam or Not Spam
# Simulated data
alerts = {'Alert': ['Alert1', 'Alert2', 'Alert3'], 'Content': ['urgent', 'ignore', 'urgent']}
alerts_df = pd.DataFrame(alerts)
alerts_df['Spam'] = [0 if 'urgent' in text else 1 for text in alerts_df['Content']]  # 0 = Not Spam, 1 = Spam

X_alerts = alerts_df[['Content']]
y_alerts = alerts_df['Spam']

# Naïve Bayes Example
# Placeholder: Actual preprocessing for text is needed
nb = GaussianNB()
# Simulating alert spam detection
print("Task 4: Spam Classification Completed (Example Placeholder).")

# Task 5: Diagnose parking system issues
# Simulated data
issues = {'Issue': ['Issue1', 'Issue2', 'Issue3'], 'Type': ['hardware', 'software', 'hardware']}
issues_df = pd.DataFrame(issues)
issues_df['Label'] = [0 if x == 'hardware' else 1 for x in issues_df['Type']]  # 0 = Hardware, 1 = Software

X_issues = issues_df[['Issue']]
y_issues = issues_df['Label']
# Simulating issue diagnosis
print("Task 5: Issue Diagnosis Completed (Example Placeholder).")
```



```python
# Task 6: Predict parking charges
# Simulated data
pricing_data = {
    'Duration (hrs)': [2, 1, 3, 4, 1.5],
    'Slot Type': [1, 2, 1, 3, 2],  # 1 = Regular, 2 = VIP, 3 = Reserved
    'Price ($)': [5, 10, 7.5, 20, 12]
}
pricing_df = pd.DataFrame(pricing_data)

X_pricing = pricing_df[['Duration (hrs)', 'Slot Type']]
y_pricing = pricing_df['Price ($)']

# Linear Regression for prediction
lr = LinearRegression()
lr.fit(X_pricing, y_pricing)
predicted_prices = lr.predict([[2.5, 2]])  # Example input: 2.5 hours, VIP slot
print("Task 6: Predicted Parking Charge:", predicted_prices)

# Task 6 - Extra: Clustering parking areas based on usage
kmeans = KMeans(n_clusters=2, random_state=42)
pricing_df['Cluster'] = kmeans.fit_predict(pricing_df[['Duration (hrs)', 'Price ($)']])
print("Task 6: Clustering of Parking Areas:\n", pricing_df)
```



```
Task 1: DataFrame after forward fill:
  Slot  Occupancy  Duration (hrs)
0   A1        1.0             2.0
1   A2        1.0             2.0
2   A3        0.0             0.5
3   A4        0.0             1.0
4   A5        1.0             1.0
Task 2: Predicted Occupancy: [0.]
Task 3: Predicted Demand: ['Low Demand']
Task 4: Spam Classification Completed (Example Placeholder).
Task 5: Issue Diagnosis Completed (Example Placeholder).
Task 6: Predicted Parking Charge: [12.35051546]
Task 6: Clustering of Parking Areas:
   Duration (hrs)  Slot Type  Price ($)  Cluster
0             2.0          1        5.0        0
1             1.0          2       10.0        0
2             3.0          1        7.5        0
3             4.0          3       20.0        1
4             1.5          2       12.0        0
```

Explanation of the Code:

1. **Task 1**: Handles missing parking data using forward fill.
2. **Task 2**: Predicts slot occupancy using the KNN algorithm.
3. **Task 3**: Classifies parking areas based on demand using a Decision Tree.
4. **Task 4**: Simulates spam alert detection using Naïve Bayes.
5. **Task 5**: Diagnoses parking system issues based on a predefined dataset.
6. **Task 6**: Predicts parking charges using Linear Regression and clusters parking areas using K-Means.