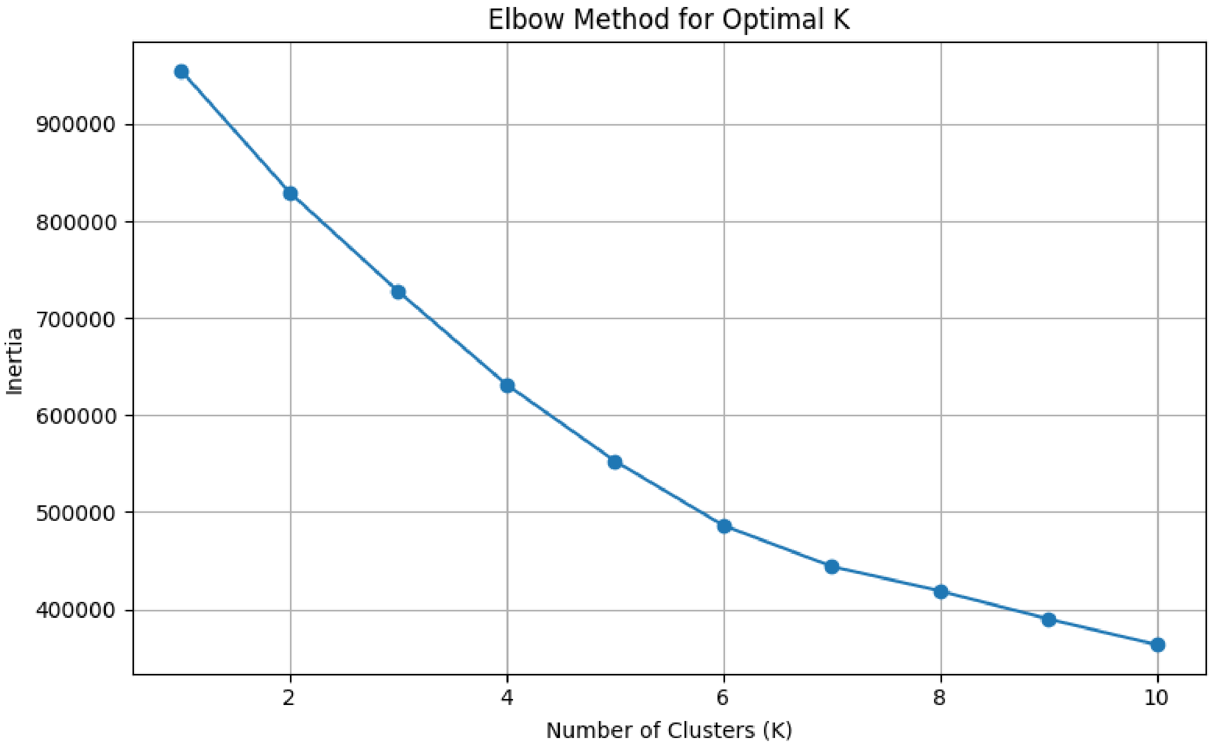
~\Downloads\DS MajorPro\Hotel Bookings[part-2].py

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import mean_squared_error, accuracy_score, confusion_matrix
import matplotlib.pyplot as plt

# Load dataset
file_path = r"C:\Users\K KRISHNAVINAYAKA\Downloads\Hotel Bookings.csv"
df = pd.read_csv(file_path)

# Selecting relevant features and making a copy to avoid SettingWithCopyWarning
df_model = df[['lead_time', 'stays_in_week_nights', 'stays_in_weekend_nights', 'adults', 'children', 'babies', 'adr', 'is_canceled','market_segment', 'total_of_special_requests']].copy()

# Handle missing values
df_model.fillna(0, inplace=True)

# Encode categorical variable
label_encoder = LabelEncoder()
df_model.loc[:, 'market_segment'] = label_encoder.fit_transform(df_model['market_segment'])

# Features and targets
X = df_model.drop(columns=['adr', 'is_canceled'])
y_reg = df_model['adr']  # For regression
y_clf = df_model['is_canceled']  # For classification

# Train-test split
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X, y_reg, test_size=0.2, random_state=42)
X_train_clf, X_test_clf, y_train_clf, y_test_clf = train_test_split(X, y_clf, test_size=0.2, random_state=42)

# Regression model
reg_model = LinearRegression()
reg_model.fit(X_train_reg, y_train_reg)
y_pred_reg = reg_model.predict(X_test_reg)
rmse = np.sqrt(mean_squared_error(y_test_reg, y_pred_reg))
print(f"Regression Model - RMSE: {rmse:.2f}")

# Classification model
clf_model = LogisticRegression(max_iter=500)
clf_model.fit(X_train_clf, y_train_clf)
y_pred_clf = clf_model.predict(X_test_clf)
accuracy = accuracy_score(y_test_clf, y_pred_clf)
```

```python
print(f"Classification Model - Accuracy: {accuracy:.2%}")
print("Confusion Matrix:\n", conf_matrix)

# Clustering
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Elbow method for optimal K
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

# Plot Elbow Curve
plt.figure(figsize=(8, 5))
plt.plot(range(1, 11), inertia, marker='o')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.title('Elbow Method for Optimal K')
plt.grid(True)
plt.tight_layout()
plt.show()

# Final clustering (assume K=3)
kmeans = KMeans(n_clusters=3, random_state=42, n_init=10)
df_model.loc[:, 'Cluster'] = kmeans.fit_predict(X_scaled)

print("Customer Segmentation - Clusters Assigned")
print(df_model['Cluster'].value_counts())

```

OUTPUT

```
Regression Model - RMSE: 42.65
Classification Model - Accuracy: 69.31%
Confusion Matrix:
 [[12881  2026]
 [ 5303  3668]]
Customer Segmentation - Clusters Assigned
Cluster
0    64597
2    53876
```



Elbow Method for Optimal K
Inertia
900000
800000
700000
600000
500000
400000
2
4
6
8
10
Number of Clusters (K)