SOURCE CODE

```
#import Libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import mean_absolute_error
# Read data
data = pd.read_csv('/content/drive/MyDrive/submission', encoding='latin-1')
# Categorical features for Encoding labels
categorical_features = ['Fuel_type', 'Hotel_type']
# Encoding labels
label_encoders = {}
for feature in categorical_features:
 label_encoders[feature] = LabelEncoder()
 data[feature] = label_encoders[feature].fit_transform(data[feature])
features = ['Starting_Place', 'Ending_Place', 'Members', 'Fuel_type', 'Hotel_type',
'Miscellaneous_Charge'] # Define feature
targets = ['Total_Distance', 'Travel_Days', 'Food', 'Travel', 'Total_Budget'] #
Define Targets
X = data[features]
y = data[targets]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2) # Split data
for Train and Test
scaler = StandardScaler() # Standardize numerical features
X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)
# Define model with early stopping
base_model = GradientBoostingRegressor(n_estimators=300, learning_rate=0.1,
max_depth=10, random_state=42,validation_fraction=0.2, n_iter_no_change=5,
tol=0.001)
model = MultiOutputRegressor(base_model)
model.fit(X_train_scaled, y_train) # Train model
y_pred = model.predict(X_test_scaled) # Evaluate model
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean absolute Error: {mae}")
new_data = pd.DataFrame({
 'Starting_Place': [361], # Give the Starting_place from '1' to '503'
 'Ending_Place': [396], # Give the Ending_place from '1' to '503'
 'Members': [5], # Maximum seven '7' Members limit
 'Fuel_type': ['Petrol'], # Give the Fuel_type as 'Petrol' or 'Diesel'
 'Hotel_type': ['Mid_range_eats'], # Give the Hotel_type as 'Cheap_eats' or
'Mid_range_eats' or 'Fine_Dining_eats'
 'Miscellaneous_Charge': [1000]
})
# Encoding labels for new data
for feature in categorical_features:
 new_data[feature] = label_encoders[feature].transform(new_data[feature])
new_data_scaled = scaler.transform(new_data[features]) # Scale new data
predictions = model.predict(new_data_scaled)[0] # Predict
print("Predicted values:")
for i, target in enumerate(targets):
 print(f"{target}: {predictions[i]}") # print the predicted value
```