Wheelseye Online Truck Booking Machine Learning Project

Project By : PRASAD JADHAV

In [1]:
```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_csv('wheelseye_data.csv')
pd.set_option('display.max_columns',20)
print(df.shape)
```

```
(10000, 17)
```

In [4]:
```
df.head()
```

Out[4]:

	order_id	customer_id	city_from	city_to	route_distance_km	vehicle_type	material_weight	tru
0	ORD1000	CUS9726	Chennai	Delhi	1144	Open Truck	1.61 Tons	
1	ORD1001	CUS7464	Chennai	Pune	1912	Open Truck	1.98 Tons	
2	ORD1002	CUS4668	Kanpur	Indore	1145	Open Truck	7.28 Tons	
3	ORD1003	CUS2876	Indore	Chennai	1247	Open Truck	8.69 Tons	
4	ORD1004	CUS4137	Indore	Kanpur	569	Small Pickup Truck	7.6 Tons	

In [6]:
```
df.tail()
```

Out[6]:

	order_id	customer_id	city_from	city_to	route_distance_km	vehicle_type	material_weigh
9995	ORD10995	CUS9223	Chennai	Ludhiana	643	Container	9.29 Ton
9996	ORD10996	CUS6562	Ludhiana	Pune	917	Small Pickup Truck	2.71 Ton
9997	ORD10997	CUS3532	Pune	Delhi	1687	Container	9.76 Ton
9998	ORD10998	CUS3105	Chennai	Indore	484	Open Truck	3.93 Ton
9999	ORD10999	CUS2121	Pune	Ahmedabad	1018	Small Pickup Truck	5.72 Ton

Dataset Overview

- *order_id*: Unique identifier for each transport order (e.g., ORD12345)
- *customer_id*: Unique identifier for each customer (e.g., CUS5678)
- *city_from*: Starting city (e.g., Delhi, Mumbai)
- *city_to*: Destination city (e.g., Kanpur, Chennai)
- *route_distance_km*: The total distance in kilometers between the start and destination city
- *vehicle_type*: Type of vehicle used for transport (Open Truck, Container, Small Pickup Truck)
- *material_weight*: Weight of the material being transported (in Tons or Kg)
- *truck_length_ft*: Length of the truck (7, 8, 14, 17, 19, 20, 22 ft)
- *truck_height_ft*: Height of the truck (6.0 ft for smaller trucks, 8.0 ft for larger trucks)
- *business_category*: Category of the customer (Manufacturer, Transporter, Truck Owner, Individual, Other)
- *weather_condition*: Weather conditions during the transport (e.g., Sunny, Rainy, Cloudy)
- *traffic_condition*: Traffic conditions on the route (e.g., Light, Moderate, Heavy)
- *driver_rating*: Rating of the driver out of 5 stars (e.g., 4.5)
- *customer_rating*: Customer's rating of the service out of 5 stars (e.g., 4.8)
- *order_price*: Price of the transport service (to be fetched based on route, weight, vehicle type, etc.)
- *delivery_time_hours*: The time taken for the transport service in hours
- *is_delayed*: Boolean indicating if the delivery was delayed (True or False)

Data Exploration

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 17 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   order_id             10000 non-null  object 
 1   customer_id          10000 non-null  object 
 2   city_from            10000 non-null  object 
 3   city_to              10000 non-null  object 
 4   route_distance_km    10000 non-null  int64  
 5   vehicle_type         10000 non-null  object 
 6   material_weight      10000 non-null  object 
 7   truck_length_ft      10000 non-null  int64  
 8   truck_height_ft      10000 non-null  float64
 9   business_category    10000 non-null  object 
 10  weather_condition    10000 non-null  object 
 11  traffic_condition    10000 non-null  object 
 12  driver_rating        10000 non-null  float64
 13  customer_rating      10000 non-null  float64
 14  order_price          10000 non-null  object 
 15  delivery_time_hours  10000 non-null  float64
 16  is_delayed           10000 non-null  bool   
dtypes: bool(1), float64(4), int64(2), object(10)
memory usage: 1.2+ MB
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
order_id               0
customer_id            0
city_from              0
city_to                0
route_distance_km      0
vehicle_type           0
material_weight        0
truck_length_ft        0
truck_height_ft        0
business_category      0
weather_condition      0
traffic_condition      0
driver_rating          0
customer_rating        0
order_price            0
delivery_time_hours    0
is_delayed             0
dtype: int64
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
np.int64(0)
```

In [10]:
```
df.describe()
```

Out[10]:

	route_distance_km	truck_length_ft	truck_height_ft	driver_rating	customer_rating	delivery_
count	10000.000000	10000.00000	10000.00000	10000.000000	10000.000000	
mean	1048.489900	15.33730	6.86220	4.003230	3.998540	
std	547.719651	5.45189	0.99051	0.576817	0.580458	
min	100.000000	7.00000	6.00000	3.000000	3.000000	
25%	576.000000	8.00000	6.00000	3.500000	3.500000	
50%	1048.500000	17.00000	6.00000	4.000000	4.000000	
75%	1526.000000	20.00000	8.00000	4.500000	4.500000	
max	2000.000000	22.00000	8.00000	5.000000	5.000000	

In [11]:

```
df.head()
```

Out[11]:

	order_id	customer_id	city_from	city_to	route_distance_km	vehicle_type	material_weight	tru
0	ORD1000	CUS9726	Chennai	Delhi	1144	Open Truck	1.61 Tons	
1	ORD1001	CUS7464	Chennai	Pune	1912	Open Truck	1.98 Tons	
2	ORD1002	CUS4668	Kanpur	Indore	1145	Open Truck	7.28 Tons	
3	ORD1003	CUS2876	Indore	Chennai	1247	Open Truck	8.69 Tons	
4	ORD1004	CUS4137	Indore	Kanpur	569	Small Pickup Truck	7.6 Tons	

Step by setp columns price predict ml problem # city_from city_to, Vehicle Type, Material Weight, Truck Length, Truck Height, Business Category:,

In [12]:

```
num_cols = [x for x in df.columns if df[x].dtypes != 'float64']

for col in num_cols:
    print(f"Value counts for column '{col}':")
    print(df[col].value_counts())
    print("\n" + "_"*40 + "\n")
```

```
Value counts for column 'order_id':
order_id
ORD1000     1
ORD7670     1
ORD7663     1
ORD7664     1
ORD7665     1
           ..
ORD4333     1
ORD4334     1
ORD4335     1
ORD4336     1
ORD10999    1
Name: count, Length: 10000, dtype: int64
```

```
Value counts for column 'customer_id':
customer_id
CUS1076    6
CUS8970    6
CUS3108    6
CUS9981    6
CUS4917    6
          ..
CUS5144    1
CUS1596    1
CUS7660    1
CUS7743    1
CUS6562    1
Name: count, Length: 6033, dtype: int64
```

```
Value counts for column 'city_from':
city_from
Kanpur       1153
Indore       1136
Mumbai       1133
Ludhiana     1121
Pune         1116
Ahmedabad    1112
Chennai      1100
Delhi        1089
Kolkata      1040
Name: count, dtype: int64
```

```
Value counts for column 'city_to':
city_to
Chennai      1181
Kanpur       1148
Ludhiana     1140
Pune         1109
Ahmedabad    1108
```

```
Indore       1096
Delhi        1084
Kolkata      1067
Mumbai       1067
Name: count, dtype: int64
```

```
Value counts for column 'route_distance_km':
route_distance_km
1436    19
1545    14
727     13
1804    13
675     13
        ..
667      1
368      1
890      1
748      1
1627     1
Name: count, Length: 1888, dtype: int64
```

```
Value counts for column 'vehicle_type':
vehicle_type
Small Pickup Truck    3377
Container             3329
Open Truck            3294
Name: count, dtype: int64
```

```
Value counts for column 'material_weight':
material_weight
0.64 Tons    22
6.57 Tons    21
7.6 Tons     21
1.44 Tons    21
9.21 Tons    19
             ..
0.5 Tons      3
3.05 Tons     3
3.57 Tons     3
5.77 Tons     3
4.13 Tons     1
Name: count, Length: 951, dtype: int64
```

```
Value counts for column 'truck_length_ft':
truck_length_ft
22    1512
17    1444
8     1435
```

```
19    1432
14    1409
7     1401
20    1367
Name: count, dtype: int64
```

```
Value counts for column 'business_category':
business_category
Other           2066
Transporter     2045
Truck Owner     2038
Individual      1945
Manufacturer    1906
Name: count, dtype: int64
```

```
Value counts for column 'weather_condition':
weather_condition
Sunny     3371
Rainy     3360
Cloudy    3269
Name: count, dtype: int64
```

```
Value counts for column 'traffic_condition':
traffic_condition
Heavy       3381
Moderate    3321
Light       3298
Name: count, dtype: int64
```

```
Value counts for column 'order_price':
order_price
₹21979.46    2
₹23487.09    2
₹21104.62    2
₹13382.77    2
₹29033.66    2
            ..
₹25786.97    1
₹33208.61    1
₹26283.9     1
₹21896.34    1
₹18041.51    1
Name: count, Length: 9977, dtype: int64
```

```
Value counts for column 'is_delayed':
is_delayed
```

```
False    5052
True     4948
Name: count, dtype: int64

________________________________________
```

In [13]:

```
cat_cols = [col for col in df.columns if df[col].dtype == 'object' or df[col

for col in cat_cols:
    print(f"Value counts for column '{col}':")
    print(df[col].value_counts())
    print("\n" + "_"*40 + "\n")
```

```
Value counts for column 'order_id':
order_id
ORD1000     1
ORD7670     1
ORD7663     1
ORD7664     1
ORD7665     1
           ..
ORD4333     1
ORD4334     1
ORD4335     1
ORD4336     1
ORD10999    1
Name: count, Length: 10000, dtype: int64
```

```
Value counts for column 'customer_id':
customer_id
CUS1076    6
CUS8970    6
CUS3108    6
CUS9981    6
CUS4917    6
          ..
CUS5144    1
CUS1596    1
CUS7660    1
CUS7743    1
CUS6562    1
Name: count, Length: 6033, dtype: int64
```

```
Value counts for column 'city_from':
city_from
Kanpur       1153
Indore       1136
Mumbai       1133
Ludhiana     1121
Pune         1116
Ahmedabad    1112
Chennai      1100
Delhi        1089
Kolkata      1040
Name: count, dtype: int64
```

```
Value counts for column 'city_to':
city_to
Chennai      1181
Kanpur       1148
Ludhiana     1140
Pune         1109
Ahmedabad    1108
```

```
Indore       1096
Delhi        1084
Kolkata      1067
Mumbai       1067
Name: count, dtype: int64
```

```
Value counts for column 'vehicle_type':
vehicle_type
Small Pickup Truck    3377
Container             3329
Open Truck            3294
Name: count, dtype: int64
```

```
Value counts for column 'material_weight':
material_weight
0.64 Tons    22
6.57 Tons    21
7.6 Tons     21
1.44 Tons    21
9.21 Tons    19
             ..
0.5 Tons      3
3.05 Tons     3
3.57 Tons     3
5.77 Tons     3
4.13 Tons     1
Name: count, Length: 951, dtype: int64
```

```
Value counts for column 'business_category':
business_category
Other           2066
Transporter     2045
Truck Owner     2038
Individual      1945
Manufacturer    1906
Name: count, dtype: int64
```

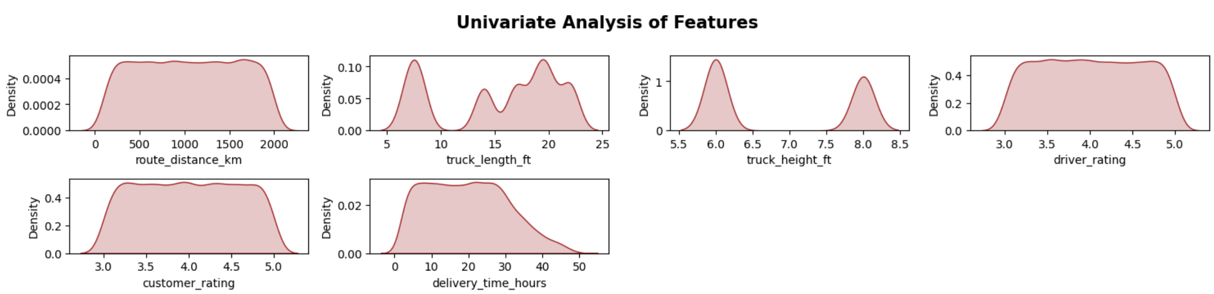
```
Value counts for column 'weather_condition':
weather_condition
Sunny     3371
Rainy     3360
Cloudy    3269
Name: count, dtype: int64
```

```
Value counts for column 'traffic_condition':
traffic_condition
```

```
Heavy       3381
Moderate    3321
Light       3298
Name: count, dtype: int64

________________________________________

Value counts for column 'order_price':
order_price
₹21979.46    2
₹23487.09    2
₹21104.62    2
₹13382.77    2
₹29033.66    2
            ..
₹25786.97    1
₹33208.61    1
₹26283.9     1
₹21896.34    1
₹18041.51    1
Name: count, Length: 9977, dtype: int64

________________________________________
```

In [14]:
```python
num_features = df.select_dtypes(include = ['int64', 'float64']).dtypes.index
```

In [15]:
```python
plt.figure(figsize=(15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=15

for i in range(0,len(num_features)):
    plt.subplot(10,4,i+1)
    sns.kdeplot(x=df[num_features[i]],shade=True,color='brown')
    plt.tight_layout()
```



In [ ]:
```python
plt.figure(figsize = (15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=20

for i in range(0,len(num_features)):
    plt.subplot(10,5,i+1)
    sns.boxplot(data=df,x=num_features[i],color='brown')
    plt.xlabel(num_features[i])
    plt.tight_layout()
```

In [ ]:
```python
cat_features = df.select_dtypes(include='object').dtypes.index
```

In [ ]:
```
plt.figure(figsize=(15,15))
plt.suptitle('Univariate Analysis of Features',fontweight='bold',fontsize=15

for i in range(0,len(cat_features)):
    plt.subplot(10,4,i+1)
    sns.countplot(x=df[cat_features[i]],color='brown')
    plt.tight_layout()
```

In [ ]:
```
# Example Machine Learning Use Cases:
```

In [ ]:
```
# Preprocessing: Convert categorical variables into dummy/one-hot encoding
# df = pd.get_dummies(df, columns=['city_from', 'city_to', 'vehicle_type', '
```

In [16]:
```
# Define mappings for each categorical feature
city_mapping = {city: idx for idx, city in enumerate(df['city_from'].unique(
vehicle_mapping = {vehicle: idx for idx, vehicle in enumerate(df['vehicle_ty
business_mapping = {business: idx for idx, business in enumerate(df['busines

# Apply the mappings to the columns
df['city_from'] = df['city_from'].map(city_mapping)
df['city_to'] = df['city_to'].map(city_mapping)  # Assuming city_to has simi
df['vehicle_type'] = df['vehicle_type'].map(vehicle_mapping)
df['business_category'] = df['business_category'].map(business_mapping)
```

In [17]:
```
# Remove the word "Tons" from the material_weight column and convert to floa
df['material_weight'] = df['material_weight'].str.replace("Tons", "").astype
```

In [18]:
```
# Define mappings for the categorical features
weather_mapping = {condition: idx for idx, condition in enumerate(df['weathe
traffic_mapping = {condition: idx for idx, condition in enumerate(df['traffi
# is_delayed_mapping = {'True': 1, 'False': 0}  # Assuming is_delayed has 'T

# Apply the mappings to each column
df['weather_condition'] = df['weather_condition'].map(weather_mapping)
df['traffic_condition'] = df['traffic_condition'].map(traffic_mapping)
# df['is_delayed'] = df['is_delayed'].map(is_delayed_mapping)
```

In [19]:
```
# df['is_delayed'] = df['is_delayed'].map({'False':0, 'True':1}).astype(int)
```

In [20]:
```
df['is_delayed'] = df['is_delayed'].astype(int)
```

Price Prediction (Regression)

In [22]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
```

In [23]:
```
price_features = df[['city_from', 'city_to', 'vehicle_type', 'material_weigh
```

In [24]:
```
# Feature Engineering: Remove unnecessary columns and extract target (price)
df['order_price'] = df['order_price'].replace('[₹,]', '', regex=True).astype
```

```
# X = df.drop(['order_id', 'customer_id', 'order_price', 'is_delayed'], axis
X = price_features
y = df['order_price']  # Target (Price)
```

In [25]:
```
# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

In [26]:
```
# Model Training: Random Forest Regressor
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[26]:

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

In [27]:
```
# Prediction and Evaluation
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Price Prediction RMSE: {rmse}")
```

```
Price Prediction RMSE: 8673.3491955699
```

Delivery Time Estimation (Regression)

In [28]:
```
from sklearn.tree import DecisionTreeRegressor
```

In [29]:
```
# Preprocessing: Use the same df but now we focus on predicting delivery tim
X = df.drop(['order_id', 'customer_id', 'delivery_time_hours', 'is_delayed']
y = df['delivery_time_hours']  # Target (Delivery Time)
```

In [30]:
```
# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

In [31]:
```
# Model Training: Random Forest Regressor for Delivery Time
model = DecisionTreeRegressor(random_state=42)
model.fit(X_train, y_train)
```

Out[31]:

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

In [32]:
```
# Prediction and Evaluation
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Delivery Time Estimation RMSE: {rmse}")
```

```
Delivery Time Estimation RMSE: 5.053821865677499
```

Customer Satisfaction Analysis (Regression/Classification)

In [33]:
```
from sklearn.linear_model import LinearRegression
```

In [34]:
```
# Regression (Predict Customer Rating)
```

In [35]:
```
# Preprocessing: Use the same df but now focus on predicting customer rating
X = df.drop(['order_id', 'customer_id', 'customer_rating', 'is_delayed'], ax
y = df['customer_rating']  # Target (Customer Rating)
```

In [36]:
```
# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

In [37]:
```
# Model Training: Random Forest Regressor for Customer Rating
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[37]:
```
LinearRegression
LinearRegression()
```

In [38]:
```
# Prediction and Evaluation
y_pred = model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Customer Rating Prediction RMSE: {rmse}")
```

```
Customer Rating Prediction RMSE: 0.5816678982940906
```

In [39]:
```
# Classification (High/Low Satisfaction)
```

In [40]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [41]:
```
# Create a binary column based on customer rating threshold
df['satisfied'] = df['customer_rating'] >= 4.0  # True for ratings >= 4, Fal
```

In [42]:
```
# Preprocessing: Drop unnecessary columns
X = df.drop(['order_id', 'customer_id', 'customer_rating', 'is_delayed', 'sa
y = df['satisfied'].astype(int)  # Target (Satisfied/Not Satisfied)
```

In [43]:
```
# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

In [44]:
```
model = LogisticRegression()
model.fit(X_train, y_train)
```

Out[44]:
```
LogisticRegression
LogisticRegression()
```

In [45]:
```
# Prediction and Evaluation
y_pred = model.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Customer Satisfaction Classification Accuracy: {accuracy}")
```

```
Customer Satisfaction Classification Accuracy: 0.5155
```

Delay Prediction (Classification)

In [46]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [47]:
```
# Preprocessing: Predict delay status (is_delayed) as binary classification
X = df.drop(['order_id', 'customer_id', 'is_delayed'], axis=1)  # Features
y = df['is_delayed'].astype(int)  # Target (0: On-time, 1: Delayed)
```

In [48]:
```
# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

In [49]:
```
# Model Training: Random Forest Classifier for Delay Prediction
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

Out[49]:
```
▾ RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [50]:
```
# Prediction and Evaluation
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Delay Prediction Accuracy: {accuracy}")
```

```
Delay Prediction Accuracy: 0.504
```

In [51]:
```
df.head()
```

Out[51]:

	order_id	customer_id	city_from	city_to	route_distance_km	vehicle_type	material_weight	tru
0	ORD1000	CUS9726	0	3	1144	0	1.61	
1	ORD1001	CUS7464	0	4	1912	0	1.98	
2	ORD1002	CUS4668	1	2	1145	0	7.28	
3	ORD1003	CUS2876	2	0	1247	0	8.69	
4	ORD1004	CUS4137	2	1	569	1	7.60	

In [52]:
```
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error

X = price_features  # `price_features` should be a DataFrame with relevant p
y = df['order_price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran
```

```
# Train an XGBRegressor with default hyperparameters
xg_model = XGBRegressor()
xg_model.fit(X_train, y_train)

# Prediction and Evaluation
y_pred = xg_model.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Price Prediction RMSE: {rmse}")
```

```
Price Prediction RMSE: 8963.408836849994
```

In [53]:
```
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Ensure price_features and df['order_price'] are defined
# Example: df['order_price'] = df['order_price'].replace('[₹,]', '', regex=T
# X = price_features

# Define features and target variable
X = price_features  # `price_features` should be a DataFrame with relevant p
y = df['order_price']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Train an XGBRegressor with default hyperparameters
model = XGBRegressor()
model.fit(X_train, y_train)

# Evaluate the model on the training set
train_preds = model.predict(X_train)
train_mse = mean_squared_error(y_train, train_preds)
train_rmse = np.sqrt(train_mse)

# Evaluate the model on the testing set
test_preds = model.predict(X_test)
test_mse = mean_squared_error(y_test, test_preds)
test_rmse = np.sqrt(test_mse)

print(f"Training RMSE: {train_rmse:.2f}")
print(f"Testing RMSE: {test_rmse:.2f}")
```

```
Training RMSE: 5779.81
Testing RMSE: 8963.41
```

In [54]:
```
# XGBoost for Price Prediction (Regression)

import xgboost as xgb
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

# Define features and target variable
X = price_features  # price_features should contain only the predictor varia
y = df['order_price']
```

```python
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Instantiate XGBoost Regressor
xgb_reg = xgb.XGBRegressor(objective='reg:squarederror', random_state=42)

# Define a comprehensive hyperparameter grid for fine-tuning
param_grid = {
    'n_estimators': [100, 300, 500, 700],
    'max_depth': [3, 5, 7, 9],
    'learning_rate': [0.01, 0.05, 0.1, 0.2, 0.3],
    'subsample': [0.6, 0.7, 0.8, 0.9, 1.0],
    'colsample_bytree': [0.5, 0.7, 0.8, 0.9, 1.0],
    'gamma': [0, 0.1, 0.2, 0.3, 0.4],
    'min_child_weight': [1, 3, 5, 7]
}

# Set up RandomizedSearchCV with cross-validation
random_search = RandomizedSearchCV(
    estimator=xgb_reg,
    param_distributions=param_grid,
    n_iter=20,  # Increased for more comprehensive search
    scoring='neg_root_mean_squared_error',
    n_jobs=-1,
    cv=3,
    verbose=2,
    random_state=42
)

# Fit the model to find the best hyperparameters
random_search.fit(X_train, y_train)

# Display the best parameters found
print(f"Best parameters: {random_search.best_params_}")

# Predict on the test set and evaluate performance
y_pred = random_search.best_estimator_.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"XGBoost Price Prediction RMSE: {rmse}")
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
Best parameters: {'subsample': 0.9, 'n_estimators': 100, 'min_child_weight':
3, 'max_depth': 3, 'learning_rate': 0.01, 'gamma': 0, 'colsample_bytree': 0.
8}
XGBoost Price Prediction RMSE: 8336.01500656909
```

In [55]:

```python
# XGBoost for Delay Prediction (Classification)

import xgboost as xgb
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score

# Preprocessing for delay classification
X = df.drop(['order_id', 'customer_id', 'is_delayed'], axis=1)
y = df['is_delayed'].astype(int)
```

```python
# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Hyperparameter tuning using RandomizedSearchCV with XGBoost
xgb_clf = xgb.XGBClassifier()

# Define the hyperparameters and the ranges for tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
    'learning_rate': [0.01, 0.1, 0.2],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

# RandomizedSearchCV
random_search = RandomizedSearchCV(xgb_clf, param_distributions=param_grid,
                                   n_jobs=-1, cv=3, verbose=2, random_state=

# Fit the model
random_search.fit(X_train, y_train)

# Best parameters
print(f"Best parameters: {random_search.best_params_}")

# Evaluate the model on test data
y_pred = random_search.best_estimator_.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"XGBoost Delay Prediction Accuracy: {accuracy}")
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
Best parameters: {'subsample': 1.0, 'n_estimators': 300, 'max_depth': 7, 'lea
rning_rate': 0.1, 'colsample_bytree': 1.0}
XGBoost Delay Prediction Accuracy: 0.5085
```

In [56]:

```python
#  LightGBM for Price Prediction (Regression)

import lightgbm as lgb
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import mean_squared_error

# Preprocessing as before
X = price_features  # price_features should contain only the predictor varia
y = df['order_price']

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Hyperparameter tuning using RandomizedSearchCV with LightGBM
lgb_reg = lgb.LGBMRegressor()

# Define the hyperparameters and the ranges for tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [3, 5, 7],
```

```python
    'learning_rate': [0.01, 0.1, 0.2],
    'num_leaves': [31, 50, 100],
    'subsample': [0.6, 0.8, 1.0],
    'colsample_bytree': [0.6, 0.8, 1.0]
}

# RandomizedSearchCV
random_search = RandomizedSearchCV(lgb_reg, param_distributions=param_grid,
                                   n_jobs=-1, cv=3, verbose=2, random_state=

# Fit the model
random_search.fit(X_train, y_train)

# Best parameters
print(f"Best parameters: {random_search.best_params_}")

# Evaluate the model on test data
y_pred = random_search.best_estimator_.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"LightGBM Price Prediction RMSE: {rmse}")
```

```
Fitting 3 folds for each of 10 candidates, totalling 30 fits
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000171 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 8000, number of use
d features: 7
[LightGBM] [Info] Start training from score 19795.171793
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Best parameters: {'subsample': 1.0, 'num_leaves': 50, 'n_estimators': 300, 'm
ax_depth': 3, 'learning_rate': 0.01, 'colsample_bytree': 0.8}
LightGBM Price Prediction RMSE: 8341.408667469945
```

In [ ]:

```python
from catboost import CatBoostRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import mean_squared_error

# Preprocessing as before
X = price_features  # price_features should contain only the predictor varia
y = df['order_price']

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Hyperparameter tuning using RandomizedSearchCV with CatBoostRegressor
```

```
cat_reg = CatBoostRegressor(verbose=0)

# Define the hyperparameters and the ranges for tuning
param_grid = {
    'iterations': [100, 200, 300],
    'depth': [4, 6, 8],
    'learning_rate': [0.01, 0.05, 0.1, 0.2],
    'l2_leaf_reg': [3, 5, 7],
    'bagging_temperature': [0, 1, 2],  # Specific for regression to control
    'random_strength': [1, 2, 5]       # Controls the amount of randomness i
}

# RandomizedSearchCV
random_search = RandomizedSearchCV(cat_reg, param_distributions=param_grid,
                                   n_jobs=-1, cv=3, verbose=2, random_state=

# Fit the model
random_search.fit(X_train, y_train)

# Best parameters
print(f"Best parameters: {random_search.best_params_}")

# Evaluate the model on test data
y_pred = random_search.best_estimator_.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"CatBoost Delay Prediction RMSE: {rmse}")
```

In [ ]:

```
# CatBoost for Delay Prediction (Classification)

from catboost import CatBoostClassifier
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import accuracy_score

# Preprocessing for CatBoost (no need to one-hot encode)
X = df.drop(['order_id', 'customer_id', 'is_delayed'], axis=1)
y = df['is_delayed'].astype(int)

# Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Hyperparameter tuning using RandomizedSearchCV with CatBoost
cat_clf = CatBoostClassifier(verbose=0)

# Define the hyperparameters and the ranges for tuning
param_grid = {
    'iterations': [100, 200, 300],
    'depth': [4, 6, 8],
    'learning_rate': [0.01, 0.1, 0.2],
    'l2_leaf_reg': [3, 5, 7]
}

# RandomizedSearchCV
random_search = RandomizedSearchCV(cat_clf, param_distributions=param_grid,
                                   n_jobs=-1, cv=3, verbose=2, random_state=

# Fit the model
```

```python
random_search.fit(X_train, y_train)

# Best parameters
print(f"Best parameters: {random_search.best_params_}")

# Evaluate the model on test data
y_pred = random_search.best_estimator_.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"CatBoost Delay Prediction Accuracy: {accuracy}")
```

In [59]:

```python
# Cross-validation

from sklearn.model_selection import cross_val_score

# Use cross-validation to evaluate the model
xgb_best = random_search.best_estimator_
scores = cross_val_score(xgb_best, X_train, y_train, cv=5, scoring='neg_root
print(f"Cross-Validation RMSE: {-scores.mean():.4f}")
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000122 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19775.630414
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000338 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19823.678847
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000120 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19810.009741
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000254 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19839.108048
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000247 seconds.
```

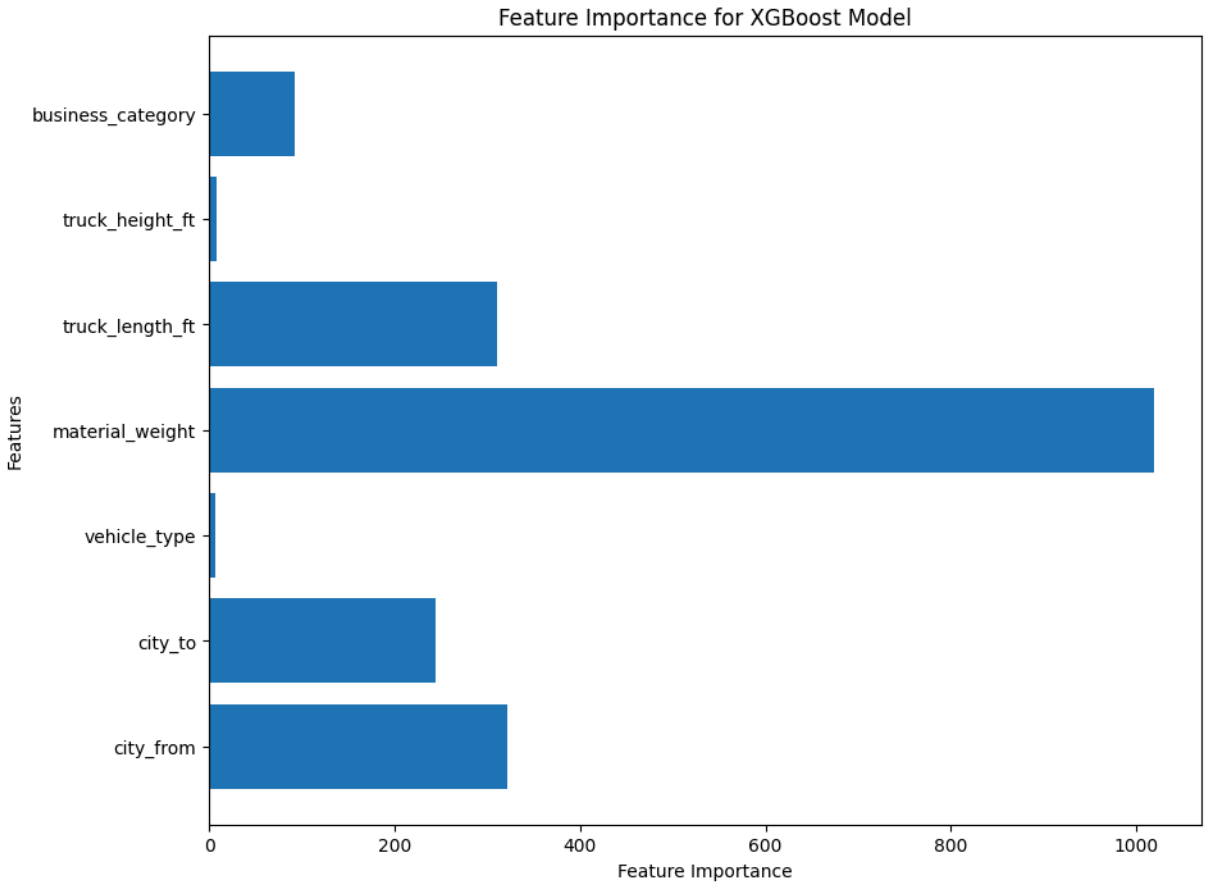
```
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19727.431917
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Cross-Validation RMSE: 8300.4867
```

In [60]:

```python
# Feature Importance Analysis
# Get feature importances
importance = random_search.best_estimator_.feature_importances_

# Plot feature importance
plt.figure(figsize=(10, 8))
plt.barh(X_train.columns, importance)
plt.xlabel('Feature Importance')
plt.ylabel('Features')
```

```
plt.title('Feature Importance for XGBoost Model')
plt.show()
```



In [62]:
```
# Ensemble Methods

from sklearn.ensemble import StackingRegressor
from sklearn.linear_model import LinearRegression

# Define base models
estimators = [
    ('xgb', xgb.XGBRegressor(**random_search.best_params_)),
    ('lgb', lgb.LGBMRegressor(**random_search.best_params_)),
#    ('cat', CatBoostRegressor(verbose=0))
]

# Stacking regressor with Linear Regression as the final estimator
stacking_reg = StackingRegressor(estimators=estimators, final_estimator=Line

# Fit the stacking model
stacking_reg.fit(X_train, y_train)

# Predict and evaluate
y_pred_stack = stacking_reg.predict(X_test)
stack_rmse = np.sqrt(mean_squared_error(y_test, y_pred_stack))
print(f"Stacking RMSE: {stack_rmse:.4f}")
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000426 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 8000, number of use
d features: 7
[LightGBM] [Info] Start training from score 19795.171793
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000322 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19775.630414
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of tes
ting was 0.000421 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19823.678847
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000140 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19810.009741
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of tes
ting was 0.000116 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19839.108048
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of tes
```

```
ting was 0.000209 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 6400, number of use
d features: 7
[LightGBM] [Info] Start training from score 19727.431917
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Stacking RMSE: 8334.7480
```

In [66]:

```python
# Final Model Training

# Train the best model on the entire training set
xgb_best.fit(X_train, y_train)

# Make predictions on the test set
y_final_pred = xgb_best.predict(X_test)

# Evaluate the final model
```

```
final_rmse = np.sqrt(mean_squared_error(y_test, y_final_pred))
print(f"Final Model RMSE: {final_rmse:.4f}")
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of tes
ting was 0.001262 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 292
[LightGBM] [Info] Number of data points in the train set: 8000, number of use
d features: 7
[LightGBM] [Info] Start training from score 19795.171793
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
```

```
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
Final Model RMSE: 8341.4087
```

In [67]:

```python
# Model Deployment

import joblib

# Save the model to disk
joblib.dump(xgb_best, 'final_model_xgboost.pkl')

# Load the model (for later use)
loaded_model = joblib.load('final_model_xgboost.pkl')

# Make predictions with the loaded model
y_loaded_pred = loaded_model.predict(X_test)
```

In [69]:

```python
# Deploy the Model Using Flask (Example)

from flask import Flask, request, jsonify
```

```python
import joblib
import pandas as pd

app = Flask(__name__)

# Load the model
model = joblib.load('final_model_xgboost.pkl')

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()  # Get data from request
    df = pd.DataFrame(data)  # Convert to DataFrame
    prediction = model.predict(df)  # Make prediction
    return jsonify({'prediction': list(prediction)})

if __name__ == '__main__':
    app.run(debug=True)
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
```

```
WARNING: This is a development server. Do not use it in a production deployme
nt. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```

In [70]:

```python
# Flask Deployment

from flask import Flask, request, jsonify
import joblib
import pandas as pd

app = Flask(__name__)

# Load the trained model
try:
    model = joblib.load('final_model_xgboost.pkl')
except Exception as e:
    print(f"Error loading model: {str(e)}")

@app.route('/predict', methods=['POST'])
def predict():
    try:
        # Get the JSON data from the POST request
        data = request.get_json()

        # Convert the JSON data into a Pandas DataFrame
        df = pd.DataFrame([data])

        # Ensure the data structure is correct before making predictions
        if df.empty:
            return jsonify({"error": "Empty input data"})
```

```python
        # Make predictions using the model
        prediction = model.predict(df)

        # Return the prediction as a JSON response
        return jsonify({'prediction': prediction.tolist()})

    except ValueError as ve:
        return jsonify({"error": f"Value Error: {str(ve)}"}), 400
    except Exception as e:
        return jsonify({"error": f"An error occurred: {str(e)}"}), 500

if __name__ == '__main__':
    try:
        app.run(debug=True)
    except Exception as e:
        print(f"Error starting the Flask app: {str(e)}")
```

```
 * Serving Flask app '__main__'
 * Debug mode: on
```

```
WARNING: This is a development server. Do not use it in a production deployme
nt. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (windowsapi)
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```

In [71]:

```python
# Testing the API with curl:
# curl -X POST http://127.0.0.1:5000/predict -H "Content-Type: application/j
```

In [72]:

```python
#  Streamlit Deployment

import streamlit as st
import joblib
import pandas as pd

# Load the trained model
try:
    model = joblib.load('final_model_xgboost.pkl')
except Exception as e:
    st.error(f"Error loading model: {str(e)}")

# Define the Streamlit app
def predict():
    st.title("Truck Delivery Prediction App")

    # Input fields for the model
    city_from = st.selectbox('City From', ['Delhi', 'Kanpur', 'Mumbai', 'Che
    city_to = st.selectbox('City To', ['Delhi', 'Kanpur', 'Mumbai', 'Chennai
    route_distance_km = st.number_input('Route Distance (km)', min_value=1,
    vehicle_type = st.selectbox('Vehicle Type', ['Open Truck', 'Container',
    material_weight = st.number_input('Material Weight (Tons)', min_value=0.
    truck_length_ft = st.selectbox('Truck Length (ft)', [7, 8, 14, 17, 19, 2
    truck_height_ft = st.selectbox('Truck Height (ft)', [6.0, 8.0])
    business_category = st.selectbox('Business Category', ['Manufacturer', '
```

```
    weather_condition = st.selectbox('Weather Condition', ['Sunny', 'Rainy',
    traffic_condition = st.selectbox('Traffic Condition', ['Light', 'Moderat
    driver_rating = st.slider('Driver Rating', 1.0, 5.0, 4.5)
    customer_rating = st.slider('Customer Rating', 1.0, 5.0, 4.8)

    # Create a button for prediction
    if st.button('Predict'):
        try:
            # Create a DataFrame with the input values
            input_data = pd.DataFrame({
                'city_from': [city_from],
                'city_to': [city_to],
                'route_distance_km': [route_distance_km],
                'vehicle_type': [vehicle_type],
                'material_weight': [material_weight],
                'truck_length_ft': [truck_length_ft],
                'truck_height_ft': [truck_height_ft],
                'business_category': [business_category],
                'weather_condition': [weather_condition],
                'traffic_condition': [traffic_condition],
                'driver_rating': [driver_rating],
                'customer_rating': [customer_rating]
            })

            # Make predictions using the loaded model
            prediction = model.predict(input_data)
            st.success(f"Predicted Price: ₹{prediction[0]:.2f}")

        except ValueError as ve:
            st.error(f"Value Error: {str(ve)}")
        except Exception as e:
            st.error(f"An error occurred: {str(e)}")

# Run the Streamlit app
if __name__ == '__main__':
    predict()
```

```
2024-11-01 10:22:03.385
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run C:\Users\prasad jadhav\AppData\Local\Programs\Python\Python
310\lib\site-packages\ipykernel_launcher.py [ARGUMENTS]
```

In [73]:

```
# streamlit run your_app_name.py
```

In [76]:

```
# Code to Optimize Multiple Models Using Optuna:

import optuna
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error, accuracy_score
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegresso
```

```python
from xgboost import XGBRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler #, LabelEncoder

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

# Define the target and features
X = price_features  # price_features should contain only the predictor varia
y = df['order_price']

# Label encoding for categorical features
# categorical_cols = ['city_from', 'city_to', 'vehicle_type', 'business_cate
# for col in categorical_cols:
#     le = LabelEncoder()
#     X[col] = le.fit_transform(X[col])

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Standardize the data for models that require scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Optuna objective function to optimize different models
def objective(trial):
    # Choose the model type
    model_type = trial.suggest_categorical("model", ["ridge", "decision_tree

    if model_type == "ridge":
        # Ridge Regression
        alpha = trial.suggest_float("alpha", 1e-3, 10.0, log=True)
        model = Ridge(alpha=alpha)

    elif model_type == "decision_tree":
        # Decision Tree Regressor
        max_depth = trial.suggest_int("max_depth", 2, 32)
        min_samples_split = trial.suggest_int("min_samples_split", 2, 20)
        model = DecisionTreeRegressor(max_depth=max_depth, min_samples_split

    elif model_type == "random_forest":
        # Random Forest Regressor
        n_estimators = trial.suggest_int("n_estimators", 100, 1000)
        max_depth = trial.suggest_int("max_depth", 2, 32)
        model = RandomForestRegressor(n_estimators=n_estimators, max_depth=m

    elif model_type == "gradient_boosting":
        # Gradient Boosting Regressor
        learning_rate = trial.suggest_float("learning_rate", 1e-3, 0.1, log=
        n_estimators = trial.suggest_int("n_estimators", 100, 1000)
        max_depth = trial.suggest_int("max_depth", 2, 32)
        model = GradientBoostingRegressor(learning_rate=learning_rate, n_est

    elif model_type == "xgboost":
        # XGBoost Regressor
```

```
        learning_rate = trial.suggest_float("learning_rate", 1e-3, 0.1, log=
        n_estimators = trial.suggest_int("n_estimators", 100, 1000)
        max_depth = trial.suggest_int("max_depth", 2, 32)
        model = XGBRegressor(learning_rate=learning_rate, n_estimators=n_est

    elif model_type == "mlp":
        # Neural Network (MLP)
        hidden_layer_sizes = trial.suggest_categorical("hidden_layer_sizes",
        activation = trial.suggest_categorical("activation", ["relu", "tanh"
        learning_rate_init = trial.suggest_float("learning_rate_init", 1e-4,
        model = MLPRegressor(hidden_layer_sizes=hidden_layer_sizes, activati

    # Cross-validation scoring
    score = cross_val_score(model, X_train_scaled, y_train, cv=3, scoring='n

    return -score

# Create an Optuna study for hyperparameter optimization
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=100)

# Print the best parameters and best model type
print("Best model:", study.best_trial.params)

# Train the best model on the full training set
best_model_params = study.best_trial.params
model_type = best_model_params.pop("model")

if model_type == "ridge":
    model = Ridge(**best_model_params)
elif model_type == "decision_tree":
    model = DecisionTreeRegressor(**best_model_params)
elif model_type == "random_forest":
    model = RandomForestRegressor(**best_model_params)
elif model_type == "gradient_boosting":
    model = GradientBoostingRegressor(**best_model_params)
elif model_type == "xgboost":
    model = XGBRegressor(**best_model_params)
elif model_type == "mlp":
    model = MLPRegressor(**best_model_params)

# Fit the model on the full training set
model.fit(X_train_scaled, y_train)

# Predict on the test set
y_pred = model.predict(X_test_scaled)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f"Test MSE: {mse}")
```

```
[I 2024-11-01 10:25:24,081] A new study created in memory with name: no-name-d98e71a2-aa33-466d-9229-14e44e159002
[I 2024-11-01 10:25:24,139] Trial 0 finished with value: 68663428.11762954 and parameters: {'model': 'ridge', 'alpha': 0.4161175979445562}. Best is trial 0 with value: 68663428.11762954.
[I 2024-11-01 10:25:27,445] Trial 1 finished with value: 70283473.631244 and parameters: {'model': 'xgboost', 'learning_rate': 0.004460538071699895, 'n_estimators': 171, 'max_depth': 10}. Best is trial 0 with value: 68663428.11762954.
[I 2024-11-01 10:25:27,469] Trial 2 finished with value: 68663386.20057793 and parameters: {'model': 'ridge', 'alpha': 1.9722089055681706}. Best is trial 2 with value: 68663386.20057793.
[I 2024-11-01 10:25:27,605] Trial 3 finished with value: 108750411.9074618 and parameters: {'model': 'decision_tree', 'max_depth': 30, 'min_samples_split': 11}. Best is trial 2 with value: 68663386.20057793.
[I 2024-11-01 10:26:11,235] Trial 4 finished with value: 74483055.68066157 and parameters: {'model': 'random_forest', 'n_estimators': 291, 'max_depth': 28}. Best is trial 2 with value: 68663386.20057793.
[I 2024-11-01 10:27:11,769] Trial 5 finished with value: 91219335.35091932 and parameters: {'model': 'xgboost', 'learning_rate': 0.04524494161449243, 'n_estimators': 407, 'max_depth': 23}. Best is trial 2 with value: 68663386.20057793.
[I 2024-11-01 10:27:11,799] Trial 6 finished with value: 68663425.3773238 and parameters: {'model': 'ridge', 'alpha': 0.5176822621674193}. Best is trial 2 with value: 68663386.20057793.
[I 2024-11-01 10:28:26,231] Trial 7 finished with value: 447143412.4074345 and parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 100), 'activation': 'tanh', 'learning_rate_init': 0.0005981755838983597}. Best is trial 2 with value: 68663386.20057793.
[I 2024-11-01 10:28:59,329] Trial 8 finished with value: 80779523.79128397 and parameters: {'model': 'gradient_boosting', 'learning_rate': 0.003200889164938601, 'n_estimators': 171, 'max_depth': 28}. Best is trial 2 with value: 68663386.20057793.
[I 2024-11-01 10:30:28,089] Trial 9 finished with value: 70686232.53790055 and parameters: {'model': 'mlp', 'hidden_layer_sizes': (100, 100), 'activation': 'relu', 'learning_rate_init': 0.0003510009541667591}. Best is trial 2 with value: 68663386.20057793.
[I 2024-11-01 10:30:28,135] Trial 10 finished with value: 68663239.51365662 and parameters: {'model': 'ridge', 'alpha': 7.4604331373263815}. Best is trial 10 with value: 68663239.51365662.
[I 2024-11-01 10:30:28,181] Trial 11 finished with value: 68663197.8349422 and parameters: {'model': 'ridge', 'alpha': 9.032128043222931}. Best is trial 11 with value: 68663197.8349422.
[I 2024-11-01 10:30:28,214] Trial 12 finished with value: 68663182.00951947 and parameters: {'model': 'ridge', 'alpha': 9.630345653458786}. Best is trial 12 with value: 68663182.00951947.
[I 2024-11-01 10:30:28,267] Trial 13 finished with value: 68663439.15099548 and parameters: {'model': 'ridge', 'alpha': 0.00741650100015105}. Best is trial 12 with value: 68663182.00951947.
[I 2024-11-01 10:30:47,297] Trial 14 finished with value: 71649207.14858495 and parameters: {'model': 'gradient_boosting', 'learning_rate': 0.09221526022065295, 'n_estimators': 988, 'max_depth': 2}. Best is trial 12 with value: 68663182.00951947.
[I 2024-11-01 10:30:47,435] Trial 15 finished with value: 90155355.30729155 and parameters: {'model': 'decision_tree', 'max_depth': 15, 'min_samples_split': 20}. Best is trial 12 with value: 68663182.00951947.
```

```
[I 2024-11-01 10:30:47,481] Trial 16 finished with value: 68663202.36706114 a
nd parameters: {'model': 'ridge', 'alpha': 8.860955974262941}. Best is trial
12 with value: 68663182.00951947.
[I 2024-11-01 10:31:04,929] Trial 17 finished with value: 68754764.53153673 a
nd parameters: {'model': 'random_forest', 'n_estimators': 761, 'max_depth':
3}. Best is trial 12 with value: 68663182.00951947.
[I 2024-11-01 10:31:04,966] Trial 18 finished with value: 68663438.73461847 a
nd parameters: {'model': 'ridge', 'alpha': 0.02283333148843314}. Best is tria
l 12 with value: 68663182.00951947.
[I 2024-11-01 10:31:05,004] Trial 19 finished with value: 68663439.32132785 a
nd parameters: {'model': 'ridge', 'alpha': 0.0011099043745178626}. Best is tr
ial 12 with value: 68663182.00951947.
[I 2024-11-01 10:31:57,254] Trial 20 finished with value: 375510295.734067 an
d parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 50), 'activation':
'tanh', 'learning_rate_init': 0.008383116576355697}. Best is trial 12 with va
lue: 68663182.00951947.
[I 2024-11-01 10:31:57,308] Trial 21 finished with value: 68663172.4173289 an
d parameters: {'model': 'ridge', 'alpha': 9.993329168835508}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:31:57,354] Trial 22 finished with value: 68663204.93295261 a
nd parameters: {'model': 'ridge', 'alpha': 8.764074621023136}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:31:57,446] Trial 23 finished with value: 68663409.46826835 a
nd parameters: {'model': 'ridge', 'alpha': 1.1077763342027287}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:31:57,527] Trial 24 finished with value: 68663362.68649116 a
nd parameters: {'model': 'ridge', 'alpha': 2.847484363675904}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:33:30,378] Trial 25 finished with value: 74235026.76154555 a
nd parameters: {'model': 'xgboost', 'learning_rate': 0.0010842199043400134,
'n_estimators': 633, 'max_depth': 19}. Best is trial 21 with value: 68663172.
4173289.
[I 2024-11-01 10:34:21,692] Trial 26 finished with value: 70279978.13412084 a
nd parameters: {'model': 'random_forest', 'n_estimators': 818, 'max_depth': 1
0}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:35:51,740] Trial 27 finished with value: 117580609.75659454
and parameters: {'model': 'gradient_boosting', 'learning_rate': 0.02155304124
740829, 'n_estimators': 480, 'max_depth': 22}. Best is trial 21 with value: 6
8663172.4173289.
[I 2024-11-01 10:35:51,857] Trial 28 finished with value: 81616099.29211701 a
nd parameters: {'model': 'decision_tree', 'max_depth': 9, 'min_samples_spli
t': 3}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:35:51,905] Trial 29 finished with value: 68663435.94352134 a
nd parameters: {'model': 'ridge', 'alpha': 0.12619047702739875}. Best is tria
l 21 with value: 68663172.4173289.
[I 2024-11-01 10:35:51,943] Trial 30 finished with value: 68663354.68117605 a
nd parameters: {'model': 'ridge', 'alpha': 3.145858507648962}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:35:51,981] Trial 31 finished with value: 68663204.80646567 a
nd parameters: {'model': 'ridge', 'alpha': 8.76884994698507}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:35:52,017] Trial 32 finished with value: 68663263.18795009 a
nd parameters: {'model': 'ridge', 'alpha': 6.570124331041535}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:35:52,058] Trial 33 finished with value: 68663360.39689736 a
nd parameters: {'model': 'ridge', 'alpha': 2.9328019237659544}. Best is trial
```

```
21 with value: 68663172.4173289.
[I 2024-11-01 10:35:52,096] Trial 34 finished with value: 68663193.9039505 an
d parameters: {'model': 'ridge', 'alpha': 9.18064923146643}. Best is trial 21
with value: 68663172.4173289.
[I 2024-11-01 10:36:29,589] Trial 35 finished with value: 86181072.42693734 a
nd parameters: {'model': 'xgboost', 'learning_rate': 0.011264266172384322, 'n
_estimators': 991, 'max_depth': 13}. Best is trial 21 with value: 68663172.41
73289.
[I 2024-11-01 10:36:29,632] Trial 36 finished with value: 68663412.04716446 a
nd parameters: {'model': 'ridge', 'alpha': 1.0120681728682561}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:36:29,765] Trial 37 finished with value: 94194875.37447363 a
nd parameters: {'model': 'decision_tree', 'max_depth': 32, 'min_samples_spli
t': 20}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:36:52,076] Trial 38 finished with value: 68965976.72235405 a
nd parameters: {'model': 'random_forest', 'n_estimators': 624, 'max_depth':
5}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:36:52,130] Trial 39 finished with value: 68663341.20960145 a
nd parameters: {'model': 'ridge', 'alpha': 3.648418023356419}. Best is trial
21 with value: 68663172.4173289.
```

```
[I 2024-11-01 10:37:53,478] Trial 40 finished with value: 331672719.26505977
and parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 100), 'activatio
n': 'relu', 'learning_rate_init': 0.00010459811575120866}. Best is trial 21 w
ith value: 68663172.4173289.
[I 2024-11-01 10:37:53,523] Trial 41 finished with value: 68663200.39053816 a
nd parameters: {'model': 'ridge', 'alpha': 8.935598576306964}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:37:53,563] Trial 42 finished with value: 68663176.23936951 a
nd parameters: {'model': 'ridge', 'alpha': 9.848661964720929}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:37:53,603] Trial 43 finished with value: 68663324.30895157 a
nd parameters: {'model': 'ridge', 'alpha': 4.279694590967343}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:37:53,639] Trial 44 finished with value: 68663399.53962703 a
nd parameters: {'model': 'ridge', 'alpha': 1.47643864612499}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:39:08,018] Trial 45 finished with value: 74858885.09051067 a
nd parameters: {'model': 'gradient_boosting', 'learning_rate': 0.001047984785
929564, 'n_estimators': 349, 'max_depth': 24}. Best is trial 21 with value: 6
8663172.4173289.
[I 2024-11-01 10:41:07,096] Trial 46 finished with value: 88022859.70089124 a
nd parameters: {'model': 'xgboost', 'learning_rate': 0.004177727654344027, 'n
_estimators': 811, 'max_depth': 18}. Best is trial 21 with value: 68663172.41
73289.
[I 2024-11-01 10:41:07,137] Trial 47 finished with value: 68663327.4942287 an
d parameters: {'model': 'ridge', 'alpha': 4.16064970521638}. Best is trial 21
with value: 68663172.4173289.
[I 2024-11-01 10:41:07,179] Trial 48 finished with value: 68663180.15557887 a
nd parameters: {'model': 'ridge', 'alpha': 9.700478777027246}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:41:07,220] Trial 49 finished with value: 68663318.1245788 an
d parameters: {'model': 'ridge', 'alpha': 4.510916109929547}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:42:50,900] Trial 50 finished with value: 321605198.13700265
and parameters: {'model': 'mlp', 'hidden_layer_sizes': (100, 100), 'activatio
n': 'tanh', 'learning_rate_init': 0.007320204607182357}. Best is trial 21 wit
h value: 68663172.4173289.
[I 2024-11-01 10:42:50,936] Trial 51 finished with value: 68663172.81617485 a
nd parameters: {'model': 'ridge', 'alpha': 9.978230359395363}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:42:50,981] Trial 52 finished with value: 68663190.04134978 a
nd parameters: {'model': 'ridge', 'alpha': 9.32663435882943}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:42:51,018] Trial 53 finished with value: 68663311.05879058 a
nd parameters: {'model': 'ridge', 'alpha': 4.775237573421291}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:42:51,066] Trial 54 finished with value: 68663380.99942403 a
nd parameters: {'model': 'ridge', 'alpha': 2.1656673157052944}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:42:51,146] Trial 55 finished with value: 70467598.43991484 a
nd parameters: {'model': 'decision_tree', 'max_depth': 5, 'min_samples_spli
t': 2}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:42:51,179] Trial 56 finished with value: 68663306.93011063 a
nd parameters: {'model': 'ridge', 'alpha': 4.929757771205937}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:43:04,359] Trial 57 finished with value: 74639003.82406723 a
```

```
nd parameters: {'model': 'random_forest', 'n_estimators': 114, 'max_depth': 2
6}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:43:04,406] Trial 58 finished with value: 68663384.30286078 a
nd parameters: {'model': 'ridge', 'alpha': 2.0427854096082965}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:44:42,708] Trial 59 finished with value: 112164899.2254107 a
nd parameters: {'model': 'gradient_boosting', 'learning_rate': 0.020833738131
19872, 'n_estimators': 548, 'max_depth': 21}. Best is trial 21 with value: 68
663172.4173289.
[I 2024-11-01 10:44:42,755] Trial 60 finished with value: 68663422.98343092 a
nd parameters: {'model': 'ridge', 'alpha': 0.6064264712563425}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:44:42,801] Trial 61 finished with value: 68663201.57490735 a
nd parameters: {'model': 'ridge', 'alpha': 8.890869855545956}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:44:42,833] Trial 62 finished with value: 68663292.4542823 an
d parameters: {'model': 'ridge', 'alpha': 5.471950722456521}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:44:42,882] Trial 63 finished with value: 68663180.17649023 a
nd parameters: {'model': 'ridge', 'alpha': 9.699687655532726}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:44:42,930] Trial 64 finished with value: 68663433.76283385 a
nd parameters: {'model': 'ridge', 'alpha': 0.20696003067822813}. Best is tria
l 21 with value: 68663172.4173289.
[I 2024-11-01 10:44:42,985] Trial 65 finished with value: 68663275.93168502 a
nd parameters: {'model': 'ridge', 'alpha': 6.091605657378613}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:44:52,721] Trial 66 finished with value: 88156609.23803066 a
nd parameters: {'model': 'xgboost', 'learning_rate': 0.08621203286088601, 'n_
estimators': 250, 'max_depth': 13}. Best is trial 21 with value: 68663172.417
3289.
[I 2024-11-01 10:44:52,768] Trial 67 finished with value: 68663438.19897753 a
nd parameters: {'model': 'ridge', 'alpha': 0.04266681891898758}. Best is tria
l 21 with value: 68663172.4173289.
[I 2024-11-01 10:45:47,092] Trial 68 finished with value: 69786952.87031387 a
nd parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 50), 'activation':
'relu', 'learning_rate_init': 0.002634158937293017}. Best is trial 21 with va
lue: 68663172.4173289.
[I 2024-11-01 10:45:47,140] Trial 69 finished with value: 68663280.76578356 a
nd parameters: {'model': 'ridge', 'alpha': 5.910221712939825}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:45:47,184] Trial 70 finished with value: 68663258.72906186 a
nd parameters: {'model': 'ridge', 'alpha': 6.737672991333914}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:45:47,233] Trial 71 finished with value: 68663239.13354975 a
nd parameters: {'model': 'ridge', 'alpha': 7.474742052536655}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:45:47,276] Trial 72 finished with value: 68663192.20251927 a
nd parameters: {'model': 'ridge', 'alpha': 9.244948150254551}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:45:47,328] Trial 73 finished with value: 68663189.22902602 a
nd parameters: {'model': 'ridge', 'alpha': 9.357341788760497}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:45:47,382] Trial 74 finished with value: 68663183.66536163 a
nd parameters: {'model': 'ridge', 'alpha': 9.567715707450418}. Best is trial
21 with value: 68663172.4173289.
```

```
[I 2024-11-01 10:45:47,495] Trial 75 finished with value: 73922826.24768418 a
nd parameters: {'model': 'decision_tree', 'max_depth': 7, 'min_samples_spli
t': 11}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:47:34,024] Trial 76 finished with value: 73044488.19842939 a
nd parameters: {'model': 'random_forest', 'n_estimators': 875, 'max_depth': 1
6}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:47:34,063] Trial 77 finished with value: 68663362.6934464 an
d parameters: {'model': 'ridge', 'alpha': 2.8472252146974903}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:49:19,847] Trial 78 finished with value: 76912734.52317987 a
nd parameters: {'model': 'gradient_boosting', 'learning_rate': 0.002156541791
947052, 'n_estimators': 707, 'max_depth': 13}. Best is trial 21 with value: 6
8663172.4173289.
[I 2024-11-01 10:49:19,882] Trial 79 finished with value: 68663283.67910737 a
nd parameters: {'model': 'ridge', 'alpha': 5.800943979334838}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:49:19,916] Trial 80 finished with value: 68663439.24001162 a
nd parameters: {'model': 'ridge', 'alpha': 0.004120645592304338}. Best is tri
al 21 with value: 68663172.4173289.
```

```
[I 2024-11-01 10:49:19,959] Trial 81 finished with value: 68663177.9432019 an
d parameters: {'model': 'ridge', 'alpha': 9.78418561826308}. Best is trial 21
with value: 68663172.4173289.
[I 2024-11-01 10:49:19,998] Trial 82 finished with value: 68663176.17035209 a
nd parameters: {'model': 'ridge', 'alpha': 9.85127391421915}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:49:20,041] Trial 83 finished with value: 68663175.0925818 an
d parameters: {'model': 'ridge', 'alpha': 9.892063870761616}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:49:20,082] Trial 84 finished with value: 68663280.19485657 a
nd parameters: {'model': 'ridge', 'alpha': 5.931640095245019}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:49:20,119] Trial 85 finished with value: 68663347.02712584 a
nd parameters: {'model': 'ridge', 'alpha': 3.431325380658104}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:50:43,899] Trial 86 finished with value: 90032087.15485202 a
nd parameters: {'model': 'xgboost', 'learning_rate': 0.008632417827405066, 'n
_estimators': 477, 'max_depth': 26}. Best is trial 21 with value: 68663172.41
73289.
[I 2024-11-01 10:50:43,943] Trial 87 finished with value: 68663277.00856246 a
nd parameters: {'model': 'ridge', 'alpha': 6.051192970005129}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:50:43,985] Trial 88 finished with value: 68663337.41720887 a
nd parameters: {'model': 'ridge', 'alpha': 3.7899952030713893}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:52:04,682] Trial 89 finished with value: 457368078.3260831 a
nd parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 100), 'activatio
n': 'tanh', 'learning_rate_init': 0.0001278985892255755}. Best is trial 21 wi
th value: 68663172.4173289.
[I 2024-11-01 10:52:04,720] Trial 90 finished with value: 68663273.24849682 a
nd parameters: {'model': 'ridge', 'alpha': 6.192315233273207}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:52:04,758] Trial 91 finished with value: 68663186.95080769 a
nd parameters: {'model': 'ridge', 'alpha': 9.443474117044277}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:52:04,789] Trial 92 finished with value: 68663183.91001399 a
nd parameters: {'model': 'ridge', 'alpha': 9.558462810645214}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:52:04,822] Trial 93 finished with value: 68663172.59836781 a
nd parameters: {'model': 'ridge', 'alpha': 9.986475652231903}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:52:04,871] Trial 94 finished with value: 68663253.33098139 a
nd parameters: {'model': 'ridge', 'alpha': 6.94059669266395}. Best is trial 2
1 with value: 68663172.4173289.
[I 2024-11-01 10:52:04,902] Trial 95 finished with value: 68663321.15183832 a
nd parameters: {'model': 'ridge', 'alpha': 4.397717988601275}. Best is trial
21 with value: 68663172.4173289.
[I 2024-11-01 10:54:34,314] Trial 96 finished with value: 74366920.8828585 an
d parameters: {'model': 'random_forest', 'n_estimators': 909, 'max_depth': 3
2}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:54:34,506] Trial 97 finished with value: 97986079.92063189 a
nd parameters: {'model': 'decision_tree', 'max_depth': 20, 'min_samples_spli
t': 16}. Best is trial 21 with value: 68663172.4173289.
[I 2024-11-01 10:54:34,557] Trial 98 finished with value: 68663364.21888255 a
nd parameters: {'model': 'ridge', 'alpha': 2.7903916094161874}. Best is trial
21 with value: 68663172.4173289.
```

```
[I 2024-11-01 10:54:34,592] Trial 99 finished with value: 68663242.07998374 a
nd parameters: {'model': 'ridge', 'alpha': 7.363837080302506}. Best is trial
21 with value: 68663172.4173289.
Best model: {'model': 'ridge', 'alpha': 9.993329168835508}
Test MSE: 69594019.82830097
```

In [77]:

```
import optuna
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegresso
from xgboost import XGBRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import StandardScaler
import pickle

# Define the target and features
X = price_features  # price_features should contain only the predictor varia
y = df['order_price']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

# Standardize the data for models that require scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Optuna objective function to optimize different models
def objective(trial):
    # Choose the model type
    model_type = trial.suggest_categorical("model", ["ridge", "decision_tree

    if model_type == "ridge":
        # Ridge Regression
        alpha = trial.suggest_float("alpha", 1e-3, 10.0, log=True)
        model = Ridge(alpha=alpha)

    elif model_type == "decision_tree":
        # Decision Tree Regressor
        max_depth = trial.suggest_int("max_depth", 2, 32)
        min_samples_split = trial.suggest_int("min_samples_split", 2, 20)
        model = DecisionTreeRegressor(max_depth=max_depth, min_samples_split

    elif model_type == "random_forest":
        # Random Forest Regressor
        n_estimators = trial.suggest_int("n_estimators", 100, 1000)
        max_depth = trial.suggest_int("max_depth", 2, 32)
        model = RandomForestRegressor(n_estimators=n_estimators, max_depth=m

    elif model_type == "gradient_boosting":
        # Gradient Boosting Regressor
        learning_rate = trial.suggest_float("learning_rate", 1e-3, 0.1, log=
        n_estimators = trial.suggest_int("n_estimators", 100, 1000)
        max_depth = trial.suggest_int("max_depth", 2, 32)
```

```python
        model = GradientBoostingRegressor(learning_rate=learning_rate, n_est

    elif model_type == "xgboost":
        # XGBoost Regressor
        learning_rate = trial.suggest_float("learning_rate", 1e-3, 0.1, log=
        n_estimators = trial.suggest_int("n_estimators", 100, 1000)
        max_depth = trial.suggest_int("max_depth", 2, 32)
        model = XGBRegressor(learning_rate=learning_rate, n_estimators=n_est

    elif model_type == "mlp":
        # Neural Network (MLP)
        hidden_layer_sizes = trial.suggest_categorical("hidden_layer_sizes",
        activation = trial.suggest_categorical("activation", ["relu", "tanh"
        learning_rate_init = trial.suggest_float("learning_rate_init", 1e-4,
        model = MLPRegressor(hidden_layer_sizes=hidden_layer_sizes, activati

    # Cross-validation scoring
    score = cross_val_score(model, X_train_scaled, y_train, cv=3, scoring='n

    return -score

# Create an Optuna study for hyperparameter optimization
study = optuna.create_study(direction='minimize')
study.optimize(objective, n_trials=100)

# Print the best parameters and best model type
print("Best model parameters:", study.best_trial.params)

# Train the best model on the full training set
best_model_params = study.best_trial.params
model_type = best_model_params.pop("model")

# Define the best model using the best parameters
if model_type == "ridge":
    best_model = Ridge(**best_model_params)
elif model_type == "decision_tree":
    best_model = DecisionTreeRegressor(**best_model_params)
elif model_type == "random_forest":
    best_model = RandomForestRegressor(**best_model_params)
elif model_type == "gradient_boosting":
    best_model = GradientBoostingRegressor(**best_model_params)
elif model_type == "xgboost":
    best_model = XGBRegressor(**best_model_params)
elif model_type == "mlp":
    best_model = MLPRegressor(**best_model_params)

# Fit the best model on the full training set
best_model.fit(X_train_scaled, y_train)

# Save the best model as a .pkl file
pickle.dump(best_model, 'best_model.pkl', 'wb')
# print("Best model saved as 'best_model.pkl'")
```

```
[I 2024-11-01 10:59:33,971] A new study created in memory with name: no-name-ee069f38-6f6b-4a74-80ec-60ea99f0ad5f
[I 2024-11-01 11:00:31,375] Trial 0 finished with value: 112131791.75526951 and parameters: {'model': 'gradient_boosting', 'learning_rate': 0.03546117757472154, 'n_estimators': 278, 'max_depth': 21}. Best is trial 0 with value: 112131791.75526951.
[I 2024-11-01 11:01:40,648] Trial 1 finished with value: 73926164.257764 and parameters: {'model': 'random_forest', 'n_estimators': 444, 'max_depth': 20}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:03:10,566] Trial 2 finished with value: 91020685.58255081 and parameters: {'model': 'mlp', 'hidden_layer_sizes': (100, 100), 'activation': 'relu', 'learning_rate_init': 0.0001625450090123917}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:03:43,593] Trial 3 finished with value: 90674104.99670553 and parameters: {'model': 'xgboost', 'learning_rate': 0.03553147724738988, 'n_estimators': 196, 'max_depth': 23}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:06:41,352] Trial 4 finished with value: 91653761.8219328 and parameters: {'model': 'xgboost', 'learning_rate': 0.007402053426589449, 'n_estimators': 934, 'max_depth': 31}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:08:02,658] Trial 5 finished with value: 439621011.68819284 and parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 100), 'activation': 'tanh', 'learning_rate_init': 0.0009533560316975296}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:08:18,910] Trial 6 finished with value: 82569769.00903039 and parameters: {'model': 'xgboost', 'learning_rate': 0.013702583608935476, 'n_estimators': 241, 'max_depth': 14}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:08:19,007] Trial 7 finished with value: 78595527.29848376 and parameters: {'model': 'decision_tree', 'max_depth': 9, 'min_samples_split': 19}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:09:07,008] Trial 8 finished with value: 85423377.56786168 and parameters: {'model': 'xgboost', 'learning_rate': 0.007265910183203576, 'n_estimators': 866, 'max_depth': 14}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:09:07,170] Trial 9 finished with value: 98950063.68894637 and parameters: {'model': 'decision_tree', 'max_depth': 13, 'min_samples_split': 8}. Best is trial 1 with value: 73926164.257764.
[I 2024-11-01 11:09:30,441] Trial 10 finished with value: 68937098.82760148 and parameters: {'model': 'random_forest', 'n_estimators': 555, 'max_depth': 5}. Best is trial 10 with value: 68937098.82760148.
[I 2024-11-01 11:09:49,397] Trial 11 finished with value: 68829507.82523823 and parameters: {'model': 'random_forest', 'n_estimators': 530, 'max_depth': 4}. Best is trial 11 with value: 68829507.82523823.
[I 2024-11-01 11:10:04,335] Trial 12 finished with value: 68685605.69353737 and parameters: {'model': 'random_forest', 'n_estimators': 639, 'max_depth': 2}. Best is trial 12 with value: 68685605.69353737.
[I 2024-11-01 11:10:04,384] Trial 13 finished with value: 68663326.62438862 and parameters: {'model': 'ridge', 'alpha': 4.193155528525295}. Best is trial 13 with value: 68663326.62438862.
[I 2024-11-01 11:10:04,444] Trial 14 finished with value: 68663281.28098387 and parameters: {'model': 'ridge', 'alpha': 5.890894800544558}. Best is trial 14 with value: 68663281.28098387.
[I 2024-11-01 11:10:04,500] Trial 15 finished with value: 68663228.91314222 and parameters: {'model': 'ridge', 'alpha': 7.859654740136146}. Best is trial
```

```
15 with value: 68663228.91314222.
[I 2024-11-01 11:10:04,548] Trial 16 finished with value: 68663258.35382725 a
nd parameters: {'model': 'ridge', 'alpha': 6.751775781663592}. Best is trial
15 with value: 68663228.91314222.
[I 2024-11-01 11:10:04,605] Trial 17 finished with value: 68663198.66260736 a
nd parameters: {'model': 'ridge', 'alpha': 9.000863364753389}. Best is trial
17 with value: 68663198.66260736.
[I 2024-11-01 11:10:04,662] Trial 18 finished with value: 68663439.28329735 a
nd parameters: {'model': 'ridge', 'alpha': 0.002517984937074765}. Best is tri
al 17 with value: 68663198.66260736.
[I 2024-11-01 11:10:04,708] Trial 19 finished with value: 68663433.7354982 an
d parameters: {'model': 'ridge', 'alpha': 0.20797259589413097}. Best is trial
17 with value: 68663198.66260736.
[I 2024-11-01 11:13:18,190] Trial 20 finished with value: 89547803.9417217 an
d parameters: {'model': 'gradient_boosting', 'learning_rate': 0.0010416368443
338197, 'n_estimators': 780, 'max_depth': 31}. Best is trial 17 with value: 6
8663198.66260736.
[I 2024-11-01 11:13:18,233] Trial 21 finished with value: 68663187.63629206 a
nd parameters: {'model': 'ridge', 'alpha': 9.417556352371095}. Best is trial
21 with value: 68663187.63629206.
[I 2024-11-01 11:13:18,279] Trial 22 finished with value: 68663175.29959507 a
nd parameters: {'model': 'ridge', 'alpha': 9.884228831525226}. Best is trial
22 with value: 68663175.29959507.
[I 2024-11-01 11:13:18,318] Trial 23 finished with value: 68663425.15387116 a
nd parameters: {'model': 'ridge', 'alpha': 0.5259651553580619}. Best is trial
22 with value: 68663175.29959507.
[I 2024-11-01 11:13:18,358] Trial 24 finished with value: 68663411.70680614 a
nd parameters: {'model': 'ridge', 'alpha': 1.0246984088127367}. Best is trial
22 with value: 68663175.29959507.
[I 2024-11-01 11:13:18,399] Trial 25 finished with value: 68663439.15953805 a
nd parameters: {'model': 'ridge', 'alpha': 0.0071002080373900696}. Best is tr
ial 22 with value: 68663175.29959507.
[I 2024-11-01 11:13:18,436] Trial 26 finished with value: 68663400.7782608 an
d parameters: {'model': 'ridge', 'alpha': 1.430430221479741}. Best is trial 2
2 with value: 68663175.29959507.
[I 2024-11-01 11:13:18,475] Trial 27 finished with value: 68663174.89673787 a
nd parameters: {'model': 'ridge', 'alpha': 9.899476298425093}. Best is trial
27 with value: 68663174.89673787.
[I 2024-11-01 11:13:59,893] Trial 28 finished with value: 69698177.55892675 a
nd parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 50), 'activation':
'relu', 'learning_rate_init': 0.008998111703920574}. Best is trial 27 with va
lue: 68663174.89673787.
[I 2024-11-01 11:14:26,187] Trial 29 finished with value: 70213313.39012311 a
nd parameters: {'model': 'gradient_boosting', 'learning_rate': 0.001254653851
7486736, 'n_estimators': 117, 'max_depth': 27}. Best is trial 27 with value:
68663174.89673787.
[I 2024-11-01 11:14:26,410] Trial 30 finished with value: 142390031.3780454 a
nd parameters: {'model': 'decision_tree', 'max_depth': 26, 'min_samples_spli
t': 2}. Best is trial 27 with value: 68663174.89673787.
[I 2024-11-01 11:14:26,447] Trial 31 finished with value: 68663186.31992017 a
nd parameters: {'model': 'ridge', 'alpha': 9.467328930608}. Best is trial 27
with value: 68663174.89673787.
[I 2024-11-01 11:14:26,476] Trial 32 finished with value: 68663390.48541798 a
nd parameters: {'model': 'ridge', 'alpha': 1.812895458047031}. Best is trial
27 with value: 68663174.89673787.
[I 2024-11-01 11:14:26,523] Trial 33 finished with value: 68663178.50488697 a
```

```
nd parameters: {'model': 'ridge', 'alpha': 9.762932387863508}. Best is trial
27 with value: 68663174.89673787.
[I 2024-11-01 11:14:55,700] Trial 34 finished with value: 70100413.61892541 a
nd parameters: {'model': 'gradient_boosting', 'learning_rate': 0.002979747069
716516, 'n_estimators': 392, 'max_depth': 8}. Best is trial 27 with value: 68
663174.89673787.
[I 2024-11-01 11:14:55,746] Trial 35 finished with value: 68663381.95354599 a
nd parameters: {'model': 'ridge', 'alpha': 2.130172251069295}. Best is trial
27 with value: 68663174.89673787.
[I 2024-11-01 11:16:15,779] Trial 36 finished with value: 299667035.41928643
and parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 100), 'activatio
n': 'tanh', 'learning_rate_init': 0.008696218991372793}. Best is trial 27 wit
h value: 68663174.89673787.
[I 2024-11-01 11:16:15,819] Trial 37 finished with value: 68663438.62875076 a
nd parameters: {'model': 'ridge', 'alpha': 0.026753287247137946}. Best is tri
al 27 with value: 68663174.89673787.
[I 2024-11-01 11:16:15,857] Trial 38 finished with value: 68663368.7463354 an
d parameters: {'model': 'ridge', 'alpha': 2.621753256139958}. Best is trial 2
7 with value: 68663174.89673787.
```

```
[I 2024-11-01 11:17:17,713] Trial 39 finished with value: 89942805.6074364 an
d parameters: {'model': 'xgboost', 'learning_rate': 0.08495524965717507, 'n_e
stimators': 686, 'max_depth': 17}. Best is trial 27 with value: 68663174.8967
3787.
[I 2024-11-01 11:17:17,840] Trial 40 finished with value: 81611605.49670173 a
nd parameters: {'model': 'decision_tree', 'max_depth': 10, 'min_samples_spli
t': 18}. Best is trial 27 with value: 68663174.89673787.
[I 2024-11-01 11:17:17,880] Trial 41 finished with value: 68663185.62875855 a
nd parameters: {'model': 'ridge', 'alpha': 9.49346425378212}. Best is trial 2
7 with value: 68663174.89673787.
[I 2024-11-01 11:17:17,927] Trial 42 finished with value: 68663174.37925433 a
nd parameters: {'model': 'ridge', 'alpha': 9.919062941112978}. Best is trial 
42 with value: 68663174.37925433.
[I 2024-11-01 11:17:17,973] Trial 43 finished with value: 68663345.03537782 a
nd parameters: {'model': 'ridge', 'alpha': 3.5056396822374385}. Best is trial
42 with value: 68663174.37925433.
[I 2024-11-01 11:18:49,891] Trial 44 finished with value: 108854886.76997614 
and parameters: {'model': 'mlp', 'hidden_layer_sizes': (100, 100), 'activatio
n': 'relu', 'learning_rate_init': 0.00014477675902466267}. Best is trial 42 w
ith value: 68663174.37925433.
[I 2024-11-01 11:21:30,463] Trial 45 finished with value: 74285111.55797626 a
nd parameters: {'model': 'random_forest', 'n_estimators': 966, 'max_depth': 2
7}. Best is trial 42 with value: 68663174.37925433.
[I 2024-11-01 11:22:16,401] Trial 46 finished with value: 77355441.188477 and
parameters: {'model': 'xgboost', 'learning_rate': 0.0028366704544452483, 'n_e
stimators': 370, 'max_depth': 18}. Best is trial 42 with value: 68663174.3792
5433.
[I 2024-11-01 11:22:16,464] Trial 47 finished with value: 68663334.13179599 a
nd parameters: {'model': 'ridge', 'alpha': 3.9126818927669604}. Best is trial
42 with value: 68663174.37925433.
[I 2024-11-01 11:22:16,512] Trial 48 finished with value: 68663177.46812922 a
nd parameters: {'model': 'ridge', 'alpha': 9.80216236523409}. Best is trial 4
2 with value: 68663174.37925433.
[I 2024-11-01 11:22:16,565] Trial 49 finished with value: 68663419.68861735 a
nd parameters: {'model': 'ridge', 'alpha': 0.7285973724861334}. Best is trial
42 with value: 68663174.37925433.
[I 2024-11-01 11:24:27,724] Trial 50 finished with value: 74278734.28929675 a
nd parameters: {'model': 'random_forest', 'n_estimators': 784, 'max_depth': 3
2}. Best is trial 42 with value: 68663174.37925433.
[I 2024-11-01 11:24:27,777] Trial 51 finished with value: 68663192.42292733 a
nd parameters: {'model': 'ridge', 'alpha': 9.2366181718989}. Best is trial 42
with value: 68663174.37925433.
[I 2024-11-01 11:24:27,833] Trial 52 finished with value: 68663335.07308923 a
nd parameters: {'model': 'ridge', 'alpha': 3.8775278988253854}. Best is trial
42 with value: 68663174.37925433.
[I 2024-11-01 11:24:27,882] Trial 53 finished with value: 68663315.64682506 a
nd parameters: {'model': 'ridge', 'alpha': 4.6035878018735215}. Best is trial
42 with value: 68663174.37925433.
[I 2024-11-01 11:24:27,944] Trial 54 finished with value: 68663190.83166659 a
nd parameters: {'model': 'ridge', 'alpha': 9.296760852351381}. Best is trial 
42 with value: 68663174.37925433.
[I 2024-11-01 11:24:28,134] Trial 55 finished with value: 108485618.36304604 
and parameters: {'model': 'decision_tree', 'max_depth': 24, 'min_samples_spli
t': 11}. Best is trial 42 with value: 68663174.37925433.
[I 2024-11-01 11:24:28,182] Trial 56 finished with value: 68663358.29708788 a
nd parameters: {'model': 'ridge', 'alpha': 3.011061733234698}. Best is trial 
```

```
42 with value: 68663174.37925433.
[I 2024-11-01 11:24:41,454] Trial 57 finished with value: 83547932.46947806 a
nd parameters: {'model': 'gradient_boosting', 'learning_rate': 0.095180752547
72504, 'n_estimators': 121, 'max_depth': 11}. Best is trial 42 with value: 68
663174.37925433.
[I 2024-11-01 11:24:41,523] Trial 58 finished with value: 68663437.4447334 an
d parameters: {'model': 'ridge', 'alpha': 0.0705961327605018}. Best is trial 
42 with value: 68663174.37925433.
[I 2024-11-01 11:24:44,210] Trial 59 finished with value: 73154585.47437084 a
nd parameters: {'model': 'xgboost', 'learning_rate': 0.01747375772181485, 'n_
estimators': 527, 'max_depth': 6}. Best is trial 42 with value: 68663174.3792
5433.
[I 2024-11-01 11:24:44,275] Trial 60 finished with value: 68663293.31926104 a
nd parameters: {'model': 'ridge', 'alpha': 5.439534523153324}. Best is trial 
42 with value: 68663174.37925433.
[I 2024-11-01 11:24:44,338] Trial 61 finished with value: 68663172.73313762 a
nd parameters: {'model': 'ridge', 'alpha': 9.981373794788786}. Best is trial 
61 with value: 68663172.73313762.
[I 2024-11-01 11:24:44,404] Trial 62 finished with value: 68663315.50653814 a
nd parameters: {'model': 'ridge', 'alpha': 4.6088353135013085}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:24:44,452] Trial 63 finished with value: 68663290.8689352 an
d parameters: {'model': 'ridge', 'alpha': 5.531369727467178}. Best is trial 6
1 with value: 68663172.73313762.
[I 2024-11-01 11:24:44,509] Trial 64 finished with value: 68663177.78757979 a
nd parameters: {'model': 'ridge', 'alpha': 9.790074278004049}. Best is trial 
61 with value: 68663172.73313762.
[I 2024-11-01 11:24:44,559] Trial 65 finished with value: 68663293.33412008 a
nd parameters: {'model': 'ridge', 'alpha': 5.438977682007707}. Best is trial 
61 with value: 68663172.73313762.
[I 2024-11-01 11:24:44,606] Trial 66 finished with value: 68663282.09293216 a
nd parameters: {'model': 'ridge', 'alpha': 5.860437548966841}. Best is trial 
61 with value: 68663172.73313762.
[I 2024-11-01 11:25:51,125] Trial 67 finished with value: 445410800.2677758 a
nd parameters: {'model': 'mlp', 'hidden_layer_sizes': (50, 50), 'activation':
'tanh', 'learning_rate_init': 0.0013768945306068395}. Best is trial 61 with v
alue: 68663172.73313762.
[I 2024-11-01 11:25:51,170] Trial 68 finished with value: 68663174.38910888 a
nd parameters: {'model': 'ridge', 'alpha': 9.91868993939105}. Best is trial 6
1 with value: 68663172.73313762.
[I 2024-11-01 11:27:46,304] Trial 69 finished with value: 74351048.20409407 a
nd parameters: {'model': 'random_forest', 'n_estimators': 706, 'max_depth': 2
9}. Best is trial 61 with value: 68663172.73313762.
[I 2024-11-01 11:27:46,352] Trial 70 finished with value: 68663288.15204097 a
nd parameters: {'model': 'ridge', 'alpha': 5.633217523550235}. Best is trial 
61 with value: 68663172.73313762.
[I 2024-11-01 11:27:46,416] Trial 71 finished with value: 68663175.1574439 an
d parameters: {'model': 'ridge', 'alpha': 9.889608955454245}. Best is trial 6
1 with value: 68663172.73313762.
[I 2024-11-01 11:27:46,465] Trial 72 finished with value: 68663262.2950107 an
d parameters: {'model': 'ridge', 'alpha': 6.603672714168007}. Best is trial 6
1 with value: 68663172.73313762.
[I 2024-11-01 11:27:46,526] Trial 73 finished with value: 68663366.37530492 a
nd parameters: {'model': 'ridge', 'alpha': 2.7100614111436814}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:27:46,558] Trial 74 finished with value: 68663269.3475378 an
```

```
d parameters: {'model': 'ridge', 'alpha': 6.3387723359618295}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:30:26,766] Trial 75 finished with value: 89019247.82036363 a
nd parameters: {'model': 'gradient_boosting', 'learning_rate': 0.003699630388
100432, 'n_estimators': 852, 'max_depth': 16}. Best is trial 61 with value: 6
8663172.73313762.
[I 2024-11-01 11:30:26,864] Trial 76 finished with value: 68663439.3239317 an
d parameters: {'model': 'ridge', 'alpha': 0.0010134965549025556}. Best is tri
al 61 with value: 68663172.73313762.
[I 2024-11-01 11:30:26,960] Trial 77 finished with value: 68851525.15931912 a
nd parameters: {'model': 'decision_tree', 'max_depth': 2, 'min_samples_spli
t': 2}. Best is trial 61 with value: 68663172.73313762.
[I 2024-11-01 11:30:27,007] Trial 78 finished with value: 68663185.8259406 an
d parameters: {'model': 'ridge', 'alpha': 9.48600793171511}. Best is trial 61
with value: 68663172.73313762.
[I 2024-11-01 11:30:27,070] Trial 79 finished with value: 68663430.19996034 a
nd parameters: {'model': 'ridge', 'alpha': 0.3389549308785647}. Best is trial
61 with value: 68663172.73313762.
```

```
[I 2024-11-01 11:30:27,121] Trial 80 finished with value: 68663259.83238366 a
nd parameters: {'model': 'ridge', 'alpha': 6.696208388613386}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:30:27,179] Trial 81 finished with value: 68663179.84050132 a
nd parameters: {'model': 'ridge', 'alpha': 9.712399004011218}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:30:27,229] Trial 82 finished with value: 68663175.29283471 a
nd parameters: {'model': 'ridge', 'alpha': 9.884484695120214}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:30:27,290] Trial 83 finished with value: 68663337.34738299 a
nd parameters: {'model': 'ridge', 'alpha': 3.7926023522392978}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:30:27,339] Trial 84 finished with value: 68663239.58409439 a
nd parameters: {'model': 'ridge', 'alpha': 7.457781596249952}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:30:27,387] Trial 85 finished with value: 68663259.0809636 an
d parameters: {'model': 'ridge', 'alpha': 6.724447544442034}. Best is trial 6
1 with value: 68663172.73313762.
[I 2024-11-01 11:31:53,148] Trial 86 finished with value: 91011559.58907944 a
nd parameters: {'model': 'xgboost', 'learning_rate': 0.036033153750373526, 'n
_estimators': 610, 'max_depth': 20}. Best is trial 61 with value: 68663172.73
313762.
[I 2024-11-01 11:33:17,451] Trial 87 finished with value: 70053906.65758651 a
nd parameters: {'model': 'mlp', 'hidden_layer_sizes': (100, 100), 'activatio
n': 'relu', 'learning_rate_init': 0.0008913320404642503}. Best is trial 61 wi
th value: 68663172.73313762.
[I 2024-11-01 11:33:17,513] Trial 88 finished with value: 68663398.64904138 a
nd parameters: {'model': 'ridge', 'alpha': 1.5095219037751253}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:33:17,571] Trial 89 finished with value: 68663175.3619619 an
d parameters: {'model': 'ridge', 'alpha': 9.881868398303055}. Best is trial 6
1 with value: 68663172.73313762.
[I 2024-11-01 11:33:17,607] Trial 90 finished with value: 68663373.3667783 an
d parameters: {'model': 'ridge', 'alpha': 2.4497163137202813}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:33:17,656] Trial 91 finished with value: 68663242.30499963 a
nd parameters: {'model': 'ridge', 'alpha': 7.355368516355657}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:33:17,719] Trial 92 finished with value: 68663173.84165607 a
nd parameters: {'model': 'ridge', 'alpha': 9.93941182592321}. Best is trial 6
1 with value: 68663172.73313762.
[I 2024-11-01 11:33:17,766] Trial 93 finished with value: 68663323.42011787 a
nd parameters: {'model': 'ridge', 'alpha': 4.312919020257117}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:33:17,830] Trial 94 finished with value: 68663246.95398425 a
nd parameters: {'model': 'ridge', 'alpha': 7.180437815460158}. Best is trial
61 with value: 68663172.73313762.
[I 2024-11-01 11:34:09,082] Trial 95 finished with value: 74365145.90719002 a
nd parameters: {'model': 'random_forest', 'n_estimators': 318, 'max_depth': 2
4}. Best is trial 61 with value: 68663172.73313762.
[I 2024-11-01 11:34:09,145] Trial 96 finished with value: 68663438.94410203 a
nd parameters: {'model': 'ridge', 'alpha': 0.015076899643281646}. Best is tri
al 61 with value: 68663172.73313762.
[I 2024-11-01 11:35:57,883] Trial 97 finished with value: 88788938.93956111 a
nd parameters: {'model': 'gradient_boosting', 'learning_rate': 0.001943916004
3005915, 'n_estimators': 470, 'max_depth': 22}. Best is trial 61 with value:
```

```
68663172.73313762.
[I 2024-11-01 11:35:58,055] Trial 98 finished with value: 98593193.53870712 a
nd parameters: {'model': 'decision_tree', 'max_depth': 19, 'min_samples_spli
t': 15}. Best is trial 61 with value: 68663172.73313762.
[I 2024-11-01 11:35:58,113] Trial 99 finished with value: 68663246.01176262 a
nd parameters: {'model': 'ridge', 'alpha': 7.215885965570791}. Best is trial
61 with value: 68663172.73313762.
Best model parameters: {'model': 'ridge', 'alpha': 9.981373794788786}
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[77], line 101
     98 best_model.fit(X_train_scaled, y_train)
    100 # Save the best model as a .pkl file
--> 101 pickle.dump(best_model, 'best_model.pkl', 'wb')
    102 print("Best model saved as 'best_model.pkl'")

TypeError: 'str' object cannot be interpreted as an integer
```

In [79]:

```
pickle.dump(best_model,open('base_model.pkl','wb'))
model = pickle.load(open('base_model.pkl','rb'))
```

In [81]:

```
price_features.head()
```

Out[81]:

	city_from	city_to	vehicle_type	material_weight	truck_length_ft	truck_height_ft	business_cate
0	0	3	0	1.61	8	6.0	
1	0	4	0	1.98	20	8.0	
2	1	2	0	7.28	7	6.0	
3	2	0	0	8.69	19	8.0	
4	2	1	1	7.60	14	6.0	

In [83]:

```
price_features.dtypes
```

Out[83]:

```
city_from              int64
city_to                int64
vehicle_type           int64
material_weight      float64
truck_length_ft        int64
truck_height_ft      float64
business_category      int64
dtype: object
```

In [84]:

```
price_features.head()
```

Out[84]:

	city_from	city_to	vehicle_type	material_weight	truck_length_ft	truck_height_ft	business_cate
0	0	3	0	1.61	8	6.0	
1	0	4	0	1.98	20	8.0	
2	1	2	0	7.28	7	6.0	
3	2	0	0	8.69	19	8.0	
4	2	1	1	7.60	14	6.0	

In [86]:
```
y.head()
```

Out[86]:
```
0    21282.00
1    32028.75
2    20158.93
3    21944.58
4    14546.95
Name: order_price, dtype: float64
```

In [88]:
```
print('Enter Truck Details')

city_from = int(input('Enter the City From: '))
city_to = int(input('Enter the City To: '))
vehicle_type = int(input('Enter the Vehicle Type: '))
material_weight = float(input('Enter the Material Weight: '))
truck_length_ft = int(input('Enter the Truck Length ft: '))
truck_height_ft = float(input('Enter the Truck Height ft: '))
business_category = int(input('Enter the Business Category: '))

input_point = np.array([[city_from, city_to, vehicle_type, material_weight,

model.predict(input_point)
```

```
Enter Truck Details
Enter the City From: 0
Enter the City To: 3
Enter the Vehicle Type: 0
Enter the Material Weight: 1.61
Enter the Truck Length ft: 8
Enter the Truck Height ft: 6.0
Enter the Business Category: 0
```

Out[88]:
```
array([21259.9722974])
```

In [89]:
```
df.head()
```

Out[89]:

	order_id	customer_id	city_from	city_to	route_distance_km	vehicle_type	material_weight	truc
0	ORD1000	CUS9726	0	3	1144	0	1.61	
1	ORD1001	CUS7464	0	4	1912	0	1.98	
2	ORD1002	CUS4668	1	2	1145	0	7.28	
3	ORD1003	CUS2876	2	0	1247	0	8.69	
4	ORD1004	CUS4137	2	1	569	1	7.60	

In [90]:

```
price_features.head()
```

Out[90]:

	city_from	city_to	vehicle_type	material_weight	truck_length_ft	truck_height_ft	business_cate
0	0	3	0	1.61	8	6.0	
1	0	4	0	1.98	20	8.0	
2	1	2	0	7.28	7	6.0	
3	2	0	0	8.69	19	8.0	
4	2	1	1	7.60	14	6.0	

In [ ]:

```
price_features.to_csv('price_features.csv')
```

In [91]:

```
df_1 = pd.read_csv('wheelseye_data.csv')
pd.set_option('display.max_columns',20)
print(df.shape)
```

```
(10000, 18)
```

In [92]:

```
df_1.head()
```

Out[92]:

	order_id	customer_id	city_from	city_to	route_distance_km	vehicle_type	material_weight	tru
0	ORD1000	CUS9726	Chennai	Delhi	1144	Open Truck	1.61 Tons	
1	ORD1001	CUS7464	Chennai	Pune	1912	Open Truck	1.98 Tons	
2	ORD1002	CUS4668	Kanpur	Indore	1145	Open Truck	7.28 Tons	
3	ORD1003	CUS2876	Indore	Chennai	1247	Open Truck	8.69 Tons	
4	ORD1004	CUS4137	Indore	Kanpur	569	Small Pickup Truck	7.6 Tons	

In [94]:

```
price_features.columns
```

Out[94]:

```
Index(['city_from', 'city_to', 'vehicle_type', 'material_weight',
       'truck_length_ft', 'truck_height_ft', 'business_category'],
      dtype='object')
```

In [95]:
```
df_v1 = df_1[['city_from', 'city_to', 'vehicle_type', 'material_weight', 'tr
```

In [96]:
```
df_v1.head()
```

Out[96]:

	city_from	city_to	vehicle_type	material_weight	truck_length_ft	truck_height_ft	business_cate
0	Chennai	Delhi	Open Truck	1.61 Tons	8	6.0	Transpo
1	Chennai	Pune	Open Truck	1.98 Tons	20	8.0	Transpo
2	Kanpur	Indore	Open Truck	7.28 Tons	7	6.0	Manufact
3	Indore	Chennai	Open Truck	8.69 Tons	19	8.0	Truck O
4	Indore	Kanpur	Small Pickup Truck	7.6 Tons	14	6.0	Transpo

In [97]:
```
price_features.to_csv('encoded_features.csv')
df_v1.to_csv('cc_features.csv')
```

Thank You!

In [99]:
```
# More Advance Working Sonn..!
```

In [100...
```
# Notebook Project By : PRASAD JADHAV (ML-ENG)
```

In [ ]:
```
# LinkedIn: linkedin.com/in/prasadmjadhav2 Github: github.com/prasadmjadhav2
```