Lab 8: Define and Solve an ML Problem of Your Choosing

In [1]:
```
import pandas as pd
import numpy as np
import os
import matplotlib.pyplot as plt
import seaborn as sns
```

In this lab assignment, you will follow the machine learning life cycle and implement a model to solve a machine learning problem of your choosing. You will select a data set and choose a predictive problem that the data set supports. You will then inspect the data with your problem in mind and begin to formulate a project plan. You will then implement the machine learning project plan.

You will complete the following tasks:

1. Build Your DataFrame
2. Define Your ML Problem
3. Perform exploratory data analysis to understand your data.
4. Define Your Project Plan
5. Implement Your Project Plan:
    - Prepare your data for your model.
    - Fit your model to the training data and evaluate your model.
    - Improve your model's performance.

Part 1: Build Your DataFrame

You will have the option to choose one of four data sets that you have worked with in this program:

- The "census" data set that contains Census information from 1994: `censusData.csv`
- Airbnb NYC "listings" data set: `airbnbListingsData.csv`
- World Happiness Report (WHR) data set: `WHR2018Chapter2OnlineData.csv`
- Book Review data set: `bookReviewsData.csv`

Note that these are variations of the data sets that you have worked with in this program. For example, some do not include some of the preprocessing necessary for specific models.

Load a Data Set and Save it as a Pandas DataFrame

The code cell below contains filenames (path + filename) for each of the four data sets available to you.

Task: In the code cell below, use the same method you have been using to load the data using `pd.read_csv()` and save it to DataFrame `df`.

You can load each file as a new DataFrame to inspect the data before choosing your data set.

In [2]:
```
# File names of the four data sets
adultDataSet_filename = os.path.join(os.getcwd(), "data", "censusData.csv")
airbnbDataSet_filename = os.path.join(os.getcwd(), "data", "airbnbListingsDa
WHRDataSet_filename = os.path.join(os.getcwd(), "data", "WHR2018Chapter2Onli
bookReviewDataSet_filename = os.path.join(os.getcwd(), "data", "bookReviewsD


df = pd.read_csv(airbnbDataSet_filename)

df.head()
```

Out[2]:

	name	description	neighborhood_overview	host_name	host_location	host
0	Skylit Midtown Castle	Beautiful, spacious skylit studio in the heart...	Centrally located in the heart of Manhattan ju...	Jennifer	New York, New York, United States	A Nev sinc My pa c
1	Whole flr w/private bdrm, bath & kitchen(pls r...	Enjoy 500 s.f. top floor in 1899 brownstone, w...	Just the right mix of urban center and local n...	LisaRoxanne	New York, New York, United States	La Nat (forn
2	Spacious Brooklyn Duplex, Patio + Garden	We welcome you to stay in our lovely 2 br dupl...	NaN	Rebecca	Brooklyn, New York, United States	Rebec artist/d and H
3	Large Furnished Room Near B'way	Please don’t expect the luxury here just a bas...	Theater district, many restaurants around here.	Shunichi	New York, New York, United States	I used for a indu
4	Cozy Clean Guest Room - Family Apt	Our best guests are seeking a safe, clean, spa...	Our neighborhood is full of restaurants and ca...	MaryEllen	New York, New York, United States	Wel family my old

5 rows × 50 columns

Part 2: Define Your ML Problem

Next you will formulate your ML Problem. In the markdown cell below, answer the following questions:

1. List the data set you have chosen.
2. What will you be predicting? What is the label?
3. Is this a supervised or unsupervised learning problem? Is this a clustering, classification or regression problem? Is it a binary classificaiton or multi-class classifiction problem?
4. What are your features? (note: this list may change after your explore your data)
5. Explain why this is an important problem. In other words, how would a company create value with a model that predicts this label?

For this project, I'm using the Airbnb NYC Listings dataset. My goal is to predict the price of a listing, which is a number, so this is a supervised learning problem—specifically a regression one. I'm using features like neighbourhood_group, room_type, minimum_nights, number_of_reviews, reviews_per_month, and availability_365. I think this kind of prediction could help Airbnb hosts figure out better pricing and make it easier for users to find fair listings.

Part 3: Understand Your Data

The next step is to perform exploratory data analysis. Inspect and analyze your data set with your machine learning problem in mind. Consider the following as you inspect your data:

1. What data preparation techniques would you like to use? These data preparation techniques may include:
    - addressing missingness, such as replacing missing values with means
    - finding and replacing outliers
    - renaming features and labels
    - finding and replacing outliers
    - performing feature engineering techniques such as one-hot encoding on categorical features
    - selecting appropriate features and removing irrelevant features
    - performing specific data cleaning and preprocessing techniques for an NLP problem
    - addressing class imbalance in your data sample to promote fair AI
2. What machine learning model (or models) you would like to use that is suitable for your predictive problem and data?

- Are there other data preparation techniques that you will need to apply to build a balanced modeling data set for your problem and model? For example, will you need to scale your data?

3. How will you evaluate and improve the model's performance?

    - Are there specific evaluation metrics and methods that are appropriate for your model?

Think of the different techniques you have used to inspect and analyze your data in this course. These include using Pandas to apply data filters, using the Pandas `describe()` method to get insight into key statistics for each column, using the Pandas `dtypes` property to inspect the data type of each column, and using Matplotlib and Seaborn to detect outliers and visualize relationships between features and labels. If you are working on a classification problem, use techniques you have learned to determine if there is class imbalance.

Task: Use the techniques you have learned in this course to inspect and analyze your data. You can import additional packages that you have used in this course that you will need to perform this task.

Note: You can add code cells if needed by going to the **Insert** menu and clicking on **Insert Cell Below** in the drop-drown menu.

In [3]:
```
# YOUR CODE HERE
df.info()
df.describe()
df.isnull().sum()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28022 entries, 0 to 28021
Data columns (total 50 columns):
 #   Column                                        Non-Null Count  Dtype  
---  ------                                        --------------  -----  
 0   name                                          28017 non-null  object 
 1   description                                   27452 non-null  object 
 2   neighborhood_overview                         18206 non-null  object 
 3   host_name                                     28022 non-null  object 
 4   host_location                                 27962 non-null  object 
 5   host_about                                    17077 non-null  object 
 6   host_response_rate                            16179 non-null  float64
 7   host_acceptance_rate                          16909 non-null  float64
 8   host_is_superhost                             28022 non-null  bool   
 9   host_listings_count                           28022 non-null  float64
 10  host_total_listings_count                     28022 non-null  float64
 11  host_has_profile_pic                          28022 non-null  bool   
 12  host_identity_verified                        28022 non-null  bool   
 13  neighbourhood_group_cleansed                  28022 non-null  object 
 14  room_type                                     28022 non-null  object 
 15  accommodates                                  28022 non-null  int64  
 16  bathrooms                                     28022 non-null  float64
 17  bedrooms                                      25104 non-null  float64
 18  beds                                          26668 non-null  float64
 19  amenities                                     28022 non-null  object 
 20  price                                         28022 non-null  float64
 21  minimum_nights                                28022 non-null  int64  
 22  maximum_nights                                28022 non-null  int64  
 23  minimum_minimum_nights                        28022 non-null  float64
 24  maximum_minimum_nights                        28022 non-null  float64
 25  minimum_maximum_nights                        28022 non-null  float64
 26  maximum_maximum_nights                        28022 non-null  float64
 27  minimum_nights_avg_ntm                        28022 non-null  float64
 28  maximum_nights_avg_ntm                        28022 non-null  float64
 29  has_availability                              28022 non-null  bool   
 30  availability_30                               28022 non-null  int64  
 31  availability_60                               28022 non-null  int64  
 32  availability_90                               28022 non-null  int64  
 33  availability_365                              28022 non-null  int64  
 34  number_of_reviews                             28022 non-null  int64  
 35  number_of_reviews_ltm                         28022 non-null  int64  
 36  number_of_reviews_l30d                        28022 non-null  int64  
 37  review_scores_rating                          28022 non-null  float64
 38  review_scores_cleanliness                     28022 non-null  float64
 39  review_scores_checkin                         28022 non-null  float64
 40  review_scores_communication                   28022 non-null  float64
 41  review_scores_location                        28022 non-null  float64
 42  review_scores_value                           28022 non-null  float64
 43  instant_bookable                              28022 non-null  bool   
 44  calculated_host_listings_count                28022 non-null  int64  
 45  calculated_host_listings_count_entire_homes   28022 non-null  int64  
 46  calculated_host_listings_count_private_rooms  28022 non-null  int64  
 47  calculated_host_listings_count_shared_rooms   28022 non-null  int64  
 48  reviews_per_month                             28022 non-null  float64
 49  n_host_verifications                          28022 non-null  int64  
```

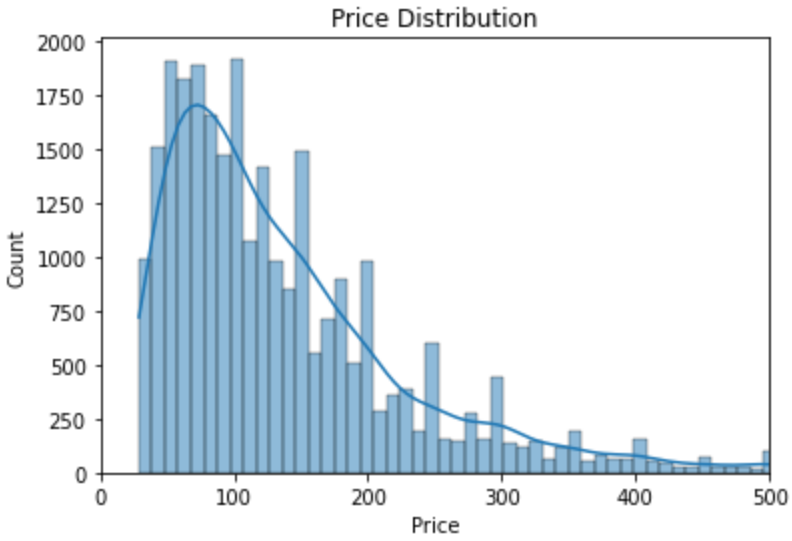
```
dtypes: bool(5), float64(21), int64(15), object(9)
memory usage: 9.8+ MB
```

Out[3]:

```
name                                            5
description                                   570
neighborhood_overview                        9816
host_name                                       0
host_location                                  60
host_about                                  10945
host_response_rate                          11843
host_acceptance_rate                        11113
host_is_superhost                               0
host_listings_count                             0
host_total_listings_count                       0
host_has_profile_pic                            0
host_identity_verified                          0
neighbourhood_group_cleansed                    0
room_type                                       0
accommodates                                    0
bathrooms                                       0
bedrooms                                     2918
beds                                         1354
amenities                                       0
price                                           0
minimum_nights                                  0
maximum_nights                                  0
minimum_minimum_nights                          0
maximum_minimum_nights                          0
minimum_maximum_nights                          0
maximum_maximum_nights                          0
minimum_nights_avg_ntm                          0
maximum_nights_avg_ntm                          0
has_availability                                0
availability_30                                 0
availability_60                                 0
availability_90                                 0
availability_365                                0
number_of_reviews                               0
number_of_reviews_ltm                           0
number_of_reviews_l30d                          0
review_scores_rating                            0
review_scores_cleanliness                       0
review_scores_checkin                           0
review_scores_communication                     0
review_scores_location                          0
review_scores_value                             0
instant_bookable                                0
calculated_host_listings_count                  0
calculated_host_listings_count_entire_homes     0
calculated_host_listings_count_private_rooms    0
calculated_host_listings_count_shared_rooms     0
reviews_per_month                               0
n_host_verifications                            0
dtype: int64
```

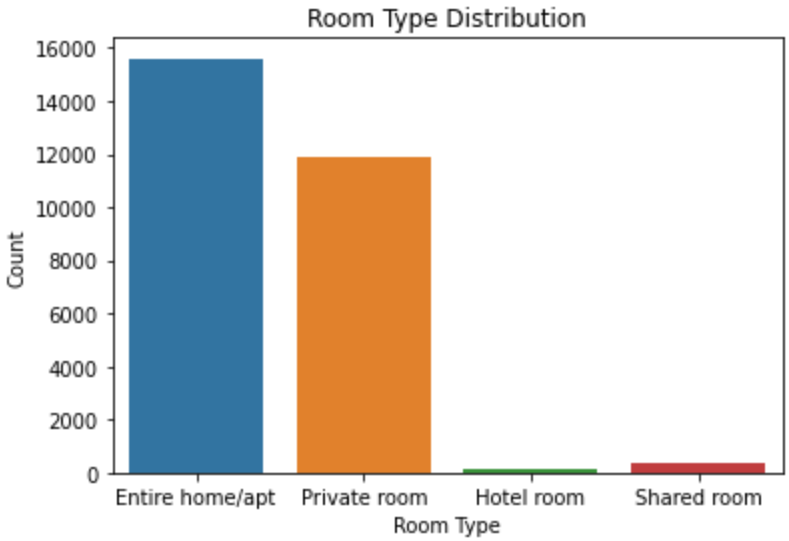
Most listings are under $500, so I'll visualize price distribution and plan to remove extreme outliers later

In [4]:
```
sns.histplot(df['price'], bins=100, kde=True)
plt.xlim(0, 500)
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Count')
plt.show()
```



This plot shows how many listings fall into each room type category.

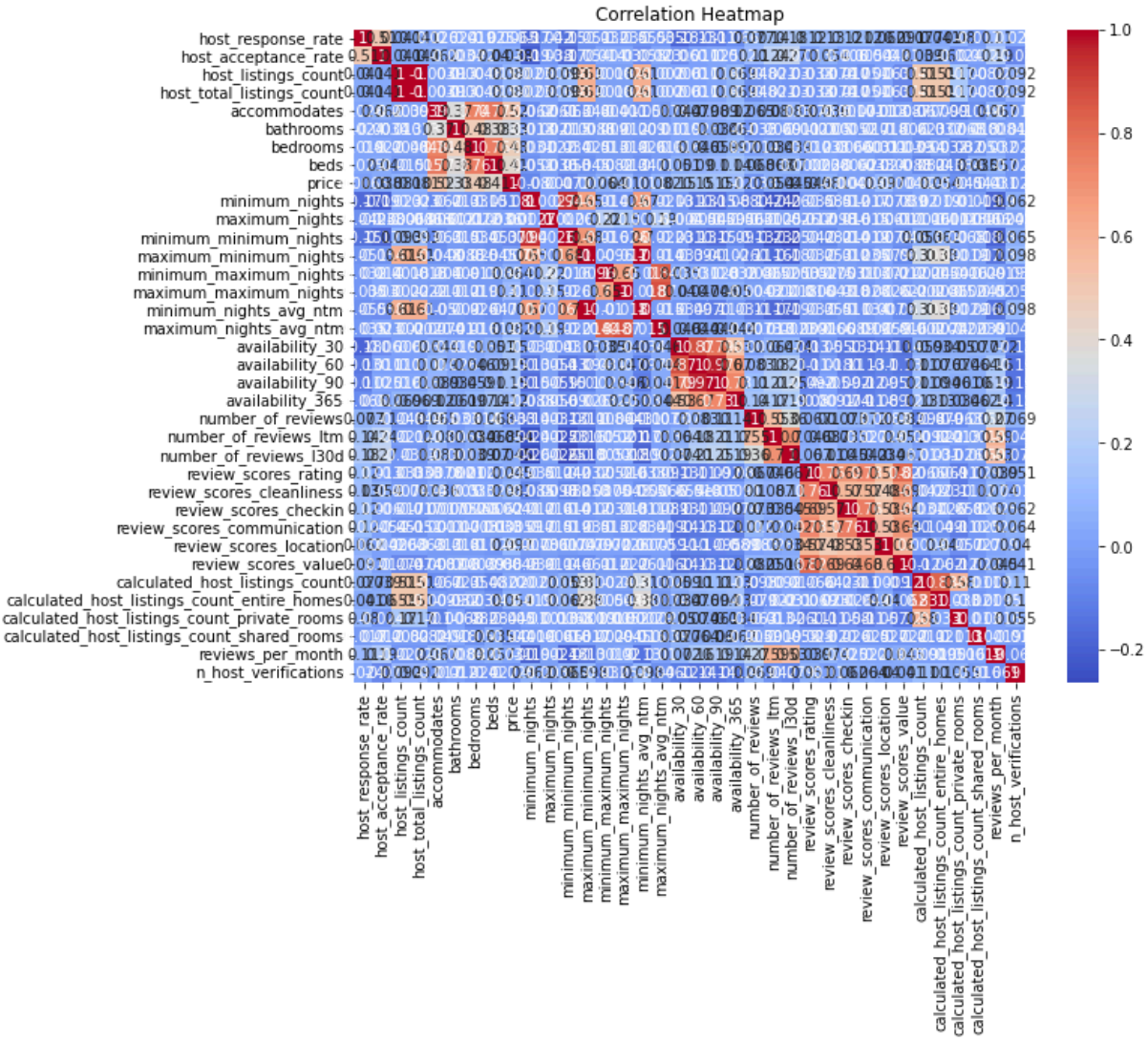
In [5]:
```
sns.countplot(data=df, x='room_type')
plt.title('Room Type Distribution')
plt.xlabel('Room Type')
plt.ylabel('Count')
plt.show()
```



I'll use a heatmap to check relationships between numeric features. This will help decide which ones to keep or ignore.

In [6]:
```
numeric_df = df.select_dtypes(include=['float64', 'int64'])

plt.figure(figsize=(10, 8))
sns.heatmap(numeric_df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



-Most prices are under $500, so I am going to remove the really high ones -There are more Entire home/apt listings than shared or private rooms - The `reviews_per_month` column has some missing values, so I'm filling them with the median

- `minimum_nights` and `availability_365` don't seem super connected to price, but I'll keep them just in case they help

Part 4: Define Your Project Plan

Now that you understand your data, in the markdown cell below, define your plan to implement the remaining phases of the machine learning life cycle (data preparation, modeling, evaluation) to solve your ML problem. Answer the following questions:

- Do you have a new feature list? If so, what are the features that you chose to keep and remove after inspecting the data?
- Explain different data preparation techniques that you will use to prepare your data for modeling.
- What is your model (or models)?
- Describe your plan to train your model, analyze its performance and then improve the model. That is, describe your model building, validation and selection plan to produce a model that generalizes well to new data.

Based on the insights from my data exploration, I will now outline the steps for preparing the data and building my machine learning model

Features to Keep and Drop

I plan to keep the following features:

- `room_type`
- `neighbourhood_group`
- `minimum_nights`
- `number_of_reviews`
- `reviews_per_month`
- `availability_365`

I will drop the following features **if they exist** in the dataset:

- `name`
- `host_name`
- `last_review`
- `id`

In [7]:
```
# Drop only the columns that exist
columns_to_drop = ['name', 'host_name', 'last_review', 'id']
existing_columns = [col for col in columns_to_drop if col in df.columns]
df.drop(existing_columns, axis=1, inplace=True)
```

Handle Missing Values and Outliers

Next, I'll fill in missing values in `reviews_per_month` using the median, and filter out listings with extremely high prices (over $500).

In [8]:
```
# Fill missing values and filter out high-priced listings
df['reviews_per_month'].fillna(df['reviews_per_month'].median(), inplace=Tru
df = df[df['price'] < 500]
```

Check Available Columns

I'll double-check what columns are left in the DataFrame before encoding categorical ones

In [9]:
```
df.columns
```

Out[9]:
```
Index(['description', 'neighborhood_overview', 'host_location', 'host_abou
t',
       'host_response_rate', 'host_acceptance_rate', 'host_is_superhost',
       'host_listings_count', 'host_total_listings_count',
       'host_has_profile_pic', 'host_identity_verified',
       'neighbourhood_group_cleansed', 'room_type', 'accommodates',
       'bathrooms', 'bedrooms', 'beds', 'amenities', 'price', 'minimum_nigh
ts',
       'maximum_nights', 'minimum_minimum_nights', 'maximum_minimum_night
s',
       'minimum_maximum_nights', 'maximum_maximum_nights',
       'minimum_nights_avg_ntm', 'maximum_nights_avg_ntm', 'has_availabilit
y',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'number_of_reviews', 'number_of_reviews_ltm',
       'number_of_reviews_l30d', 'review_scores_rating',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'instant_bookable',
       'calculated_host_listings_count',
       'calculated_host_listings_count_entire_homes',
       'calculated_host_listings_count_private_rooms',
       'calculated_host_listings_count_shared_rooms', 'reviews_per_month',
       'n_host_verifications'],
      dtype='object')
```

Encode Categorical Features

Now I'll use one-hot encoding to convert `room_type` and `neighbourhood_group` into numeric features

In [10]:
```
# Only encode columns that still exist
columns_to_encode = ['room_type', 'neighbourhood_group']
existing_encode_columns = [col for col in columns_to_encode if col in df.col

df = pd.get_dummies(df, columns=existing_encode_columns, drop_first=True)
```

Modeling & Evaluation Plan

First, I'm going to use a Linear Regression model and then try a Random Forest Regressor to see which one does better. To test how well each model works, I'll split the data 80/20 — most for training, the rest for testing. I'll look at RMSE and R^2 score to compare the results. I'll also use GridSearchCV to help tune the Random Forest and hopefully improve the performance.The goal is to figure out which model makes the best predictions based on the data I cleaned.

Part 5: Implement Your Project Plan

Task: In the code cell below, import additional packages that you have used in this course that you will need to implement your project plan.

In this section I will prepare the final dataset, split it into training and test sets, train linear regression and random forest models, evaluate each model using RMSE and R^2 score and optionally improve model performance using GridSearchcv

In [11]:
```
# YOUR CODE HERE
# Select only numeric columns and drop the label column
X = df.select_dtypes(include=[np.number]).drop(columns=['price'])
y = df['price']
```

In [12]:
```
# Remove any rows that still have missing values
X = X.dropna()
y = y[X.index]  # Keep y in sync with X
```

Train-Test Split

I will now split the data into training and testing sets using an 80/20 ratio.

In [13]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rar
```

Model 1: Linear Regression

I'll start with a basic Linear Regression model.

In [14]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

y_pred_lr = lr_model.predict(X_test)
rmse_lr = mean_squared_error(y_test, y_pred_lr, squared=False)
r2_lr = r2_score(y_test, y_pred_lr)

print("Linear Regression RMSE:", rmse_lr)
print("Linear Regression R² Score:", r2_lr)
```

```
Linear Regression RMSE: 70.18692493057605
Linear Regression R² Score: 0.40972638353705426
```

While working on Part 5, I kept getting an error when trying to fit the Linear Regression model. I didn't know what was causing it at first, but after reading the error message and

using AI to help explain it, I learned that the problem was coming from missing values (NaNs) in my features.

AI helped me figure out that even though I filled in some missing values earlier, there were still other columns with NaNs that I needed to deal with. I used `.dropna()` on `X` and made sure `y` matched by using `y = y[X.index]`. That fixed the issue and let me train both models without errors.

This helped me understand how important it is to clean the data fully before training

Task: Use the rest of this notebook to carry out your project plan.

You will:

1. Prepare your data for your model.
2. Fit your model to the training data and evaluate your model.
3. Improve your model's performance by performing model selection and/or feature selection techniques to find best model for your problem.

Add code cells below and populate the notebook with commentary, code, analyses, results, and figures as you see fit.

Model 2: Random Forest

Next, I'll try a more advanced model: Random Forest Regressor.

In [15]:
```
# YOUR CODE HERE
from sklearn.ensemble import RandomForestRegressor

rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)
rmse_rf = mean_squared_error(y_test, y_pred_rf, squared=False)
r2_rf = r2_score(y_test, y_pred_rf)

print("Random Forest RMSE:", rmse_rf)
print("Random Forest R² Score:", r2_rf)
```

```
Random Forest RMSE: 58.04166026020077
Random Forest R² Score: 0.596335465062574
```

To improve the Random Forest model, I used GridSearchCV to test different values for `n_estimators` and `max_depth`. This helped me find the best parameters and reduce prediction error.

In [16]:
```
from sklearn.model_selection import GridSearchCV

# Set the parameter grid to search
param_grid = {
    'n_estimators': [50, 100],
```

```
    'max_depth': [None, 10, 20]
}

# Set up the grid search
grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_gri

# Fit the model to the training data
grid_search.fit(X_train, y_train)

# Get the best model
best_rf_model = grid_search.best_estimator_

# Predict and evaluate the best model
y_pred_best = best_rf_model.predict(X_test)
rmse_best = mean_squared_error(y_test, y_pred_best, squared=False)
r2_best = r2_score(y_test, y_pred_best)

print("Best Random Forest RMSE:", rmse_best)
print("Best Random Forest R² Score:", r2_best)
print("Best Parameters:", grid_search.best_params_)
```

```
Best Random Forest RMSE: 58.04166026020077
Best Random Forest R² Score: 0.596335465062574
Best Parameters: {'max_depth': None, 'n_estimators': 100}
```

Grid Search Results After training my basic Random Forest model, I used GridSearchCV to see if I could improve the performance. I tested a few values for `n_estimators` and `max_depth` . The best combination it found was n_estimators: 100 max_depth: None With those parameters, the model gave me an RMSE of around 58.04 and a R² score of 0.596. It did better than my first two models, so i stuck with it. GridSearchCV helped me test different combinations, and this one gave the best results overall.