IRIS flower prediction

The Iris dataset is a popular resource in machine learning and data science. It contains 150 records of three types of Iris flowers: Iris setosa, Iris virginica, and Iris versicolor. Each record has four measurements: sepal length, sepal width, petal length, and petal width. This dataset is ideal for learning classification techniques. I will examine the Iris dataset, split it into training and testing sets, and use the Support Vector Machine (SVM) algorithm to identify the species based on these measurements.

Import the necessary library:

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

df = pd.read_csv('Iris.csv')
print(df) # Displays the first few rows and gives us a glimps of the data
```

```
      Id  SepalLengthCm  SepalWidthCm  PetalLengthCm  PetalWidthCm  \
0      1            5.1           3.5            1.4           0.2
1      2            4.9           3.0            1.4           0.2
2      3            4.7           3.2            1.3           0.2
3      4            4.6           3.1            1.5           0.2
4      5            5.0           3.6            1.4           0.2
..   ...            ...           ...            ...           ...
145  146            6.7           3.0            5.2           2.3
146  147            6.3           2.5            5.0           1.9
147  148            6.5           3.0            5.2           2.0
148  149            6.2           3.4            5.4           2.3
149  150            5.9           3.0            5.1           1.8

            Species
0       Iris-setosa
1       Iris-setosa
2       Iris-setosa
3       Iris-setosa
4       Iris-setosa
..              ...
145  Iris-virginica
146  Iris-virginica
147  Iris-virginica
148  Iris-virginica
149  Iris-virginica

[150 rows x 6 columns]
```

In [4]:
```
df.describe() # Generates summmary statistics
```

Out[4]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.054000	3.758667	1.198667
std	43.445368	0.828066	0.433594	1.764420	0.763161
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

Data Exploration

In [8]:
```
data = df.values
X = data[:, 1:4]
Y = data[:, 5]
```

In [10]:
```
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)
```

In [12]:
```
from sklearn.svm import SVC
svm = SVC()
svm.fit(X_train, y_train)
```

Out[12]:

▾ SVC

SVC()

In [14]:
```
predictions = svm.predict(X_test)
```

In [16]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, predictions) * 100
print('Accuracy: %.2f' % accuracy)
```

```
Accuracy: 93.33
```

Employment Analysis in India

Unemployment data is vital for understanding the job market and aiding policymakers, economists, and researchers in making informed decisions. In this post, we will analyze Indian unemployment statistics using Python. The data was loaded, cleaned, and visualized to gain meaningful insights.

We will use the dataset 'Unemployment_Rate_upto_11_2020.csv,' which provides details on unemployment rates, employment numbers, and labor participation rates across different Indian regions.

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

data = pd.read_csv('Unemployment_Rate_upto_11_2020.csv')
print(data.head())
```

```
           Region         Date  Frequency   Estimated Unemployment Rate (%)  \
0  Andhra Pradesh   31-01-2020          M                              5.48
1  Andhra Pradesh   29-02-2020          M                              5.83
2  Andhra Pradesh   31-03-2020          M                              5.79
3  Andhra Pradesh   30-04-2020          M                             20.51
4  Andhra Pradesh   31-05-2020          M                             17.43

    Estimated Employed   Estimated Labour Participation Rate (%) Region.1  \
0             16635535                                    41.02    South
1             16545652                                    40.90    South
2             15881197                                    39.18    South
3             11336911                                    33.10    South
4             12988845                                    36.46    South

   longitude  latitude
0    15.9129     79.74
1    15.9129     79.74
2    15.9129     79.74
3    15.9129     79.74
4    15.9129     79.74
```

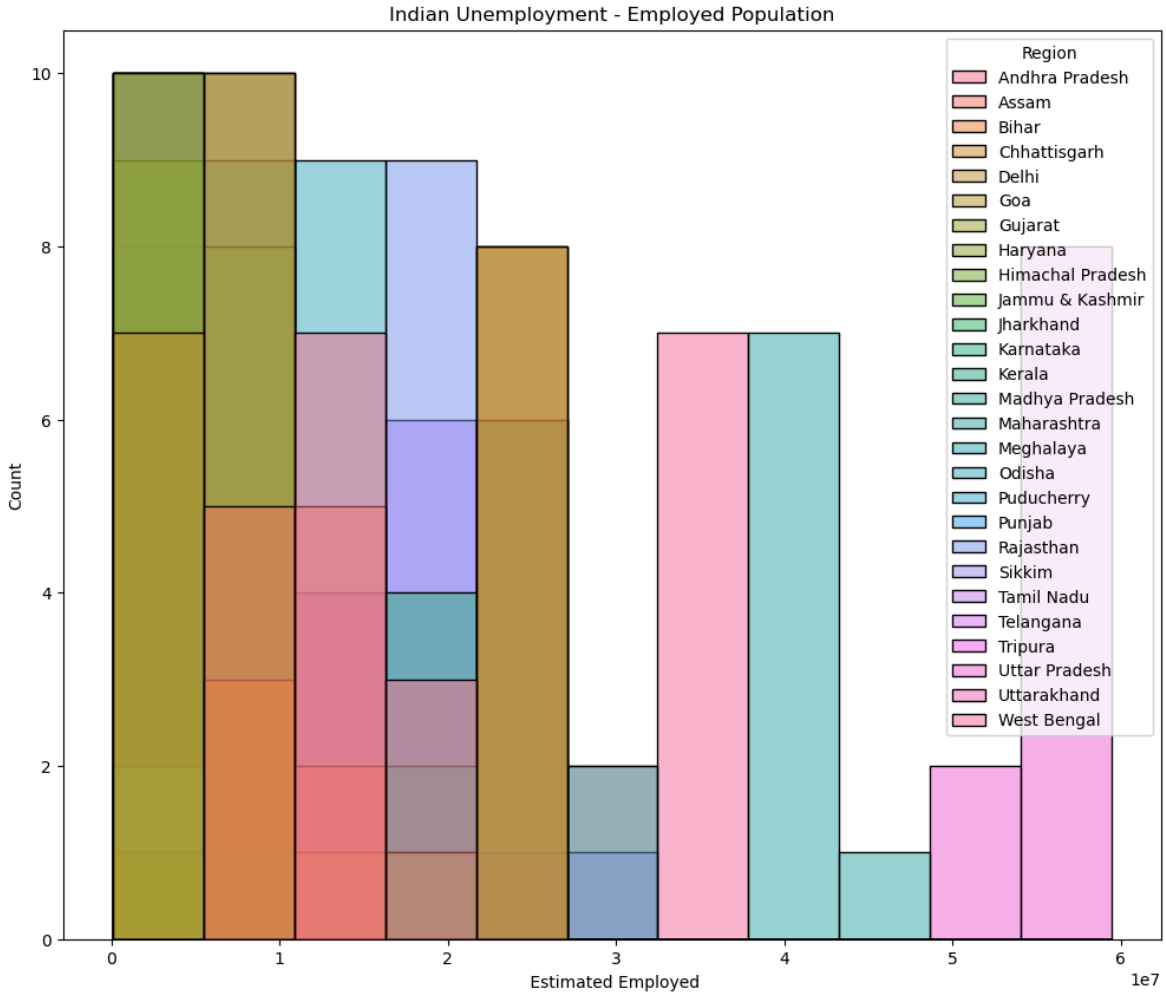
In [3]:
```
missing_values = data.isnull().sum()
print(missing_values)
```

```
Region                                    0
 Date                                     0
 Frequency                                0
 Estimated Unemployment Rate (%)          0
 Estimated Employed                       0
 Estimated Labour Participation Rate (%)  0
Region.1                                  0
longitude                                 0
latitude                                  0
dtype: int64
```

In [4]:
```
plt.figure(figsize=(14, 10))
sns.heatmap(data[[' Estimated Unemployment Rate (%)',
 ' Estimated Employed',
 ' Estimated Labour Participation Rate (%)','longitude',
 'latitude']].corr())
plt.title("Correlation Heatmap")
plt.show()
```

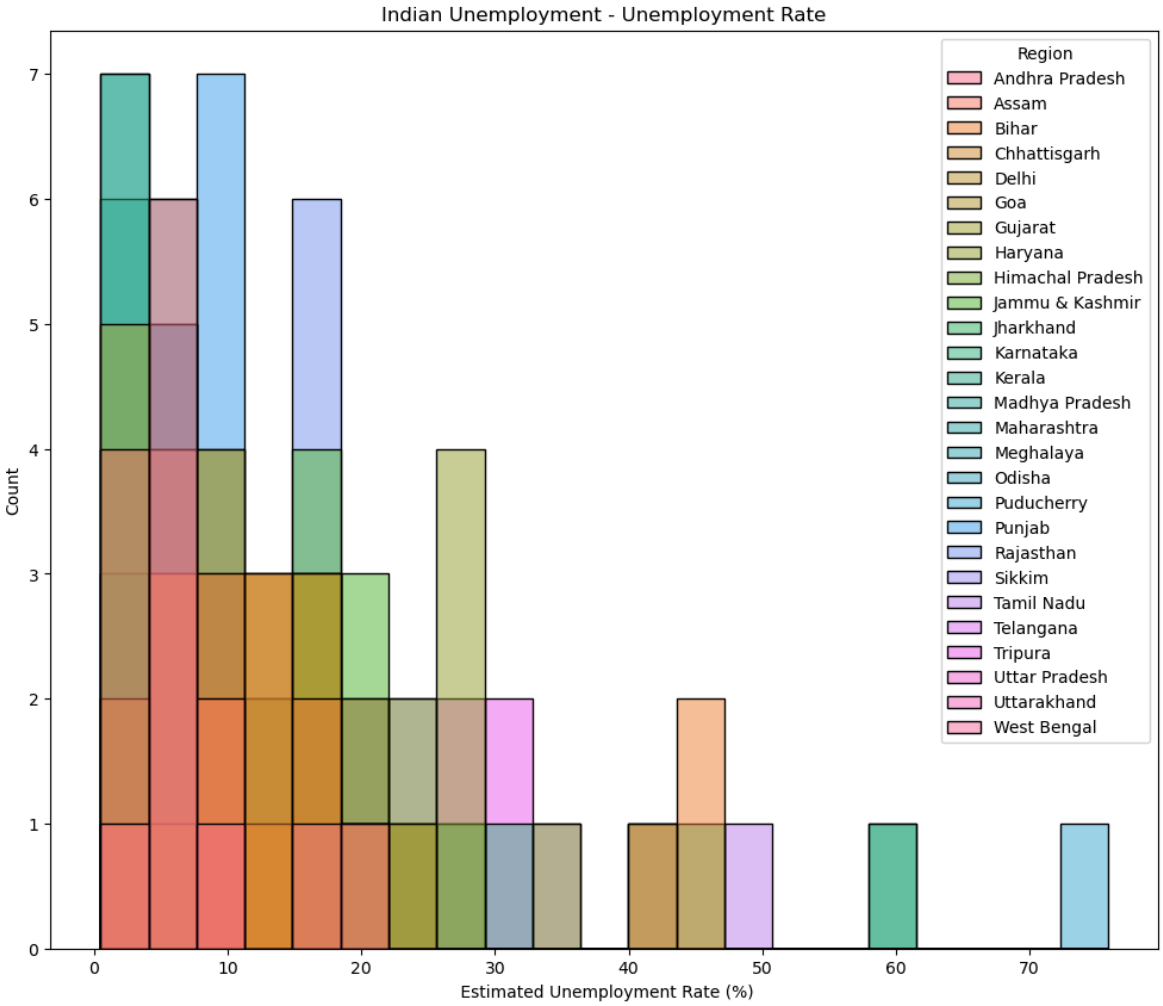
In [5]:
```
plt.figure(figsize=(12, 10))
plt.title("Indian Unemployment - Employed Population")
sns.histplot(x=" Estimated Employed", hue="Region", data=data)
plt.show()
```

```
C:\Users\pinto\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



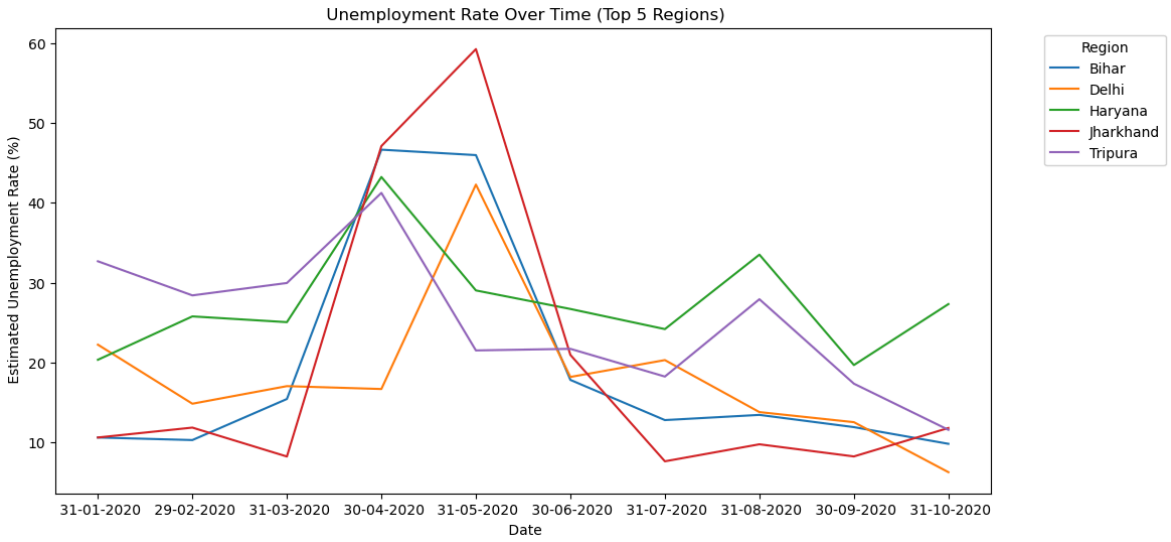
In [6]:
```
plt.figure(figsize=(12, 10))
plt.title("Indian Unemployment - Unemployment Rate")
sns.histplot(x=" Estimated Unemployment Rate (%)", hue="Region", data=data)
plt.show()
```

```
C:\Users\pinto\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

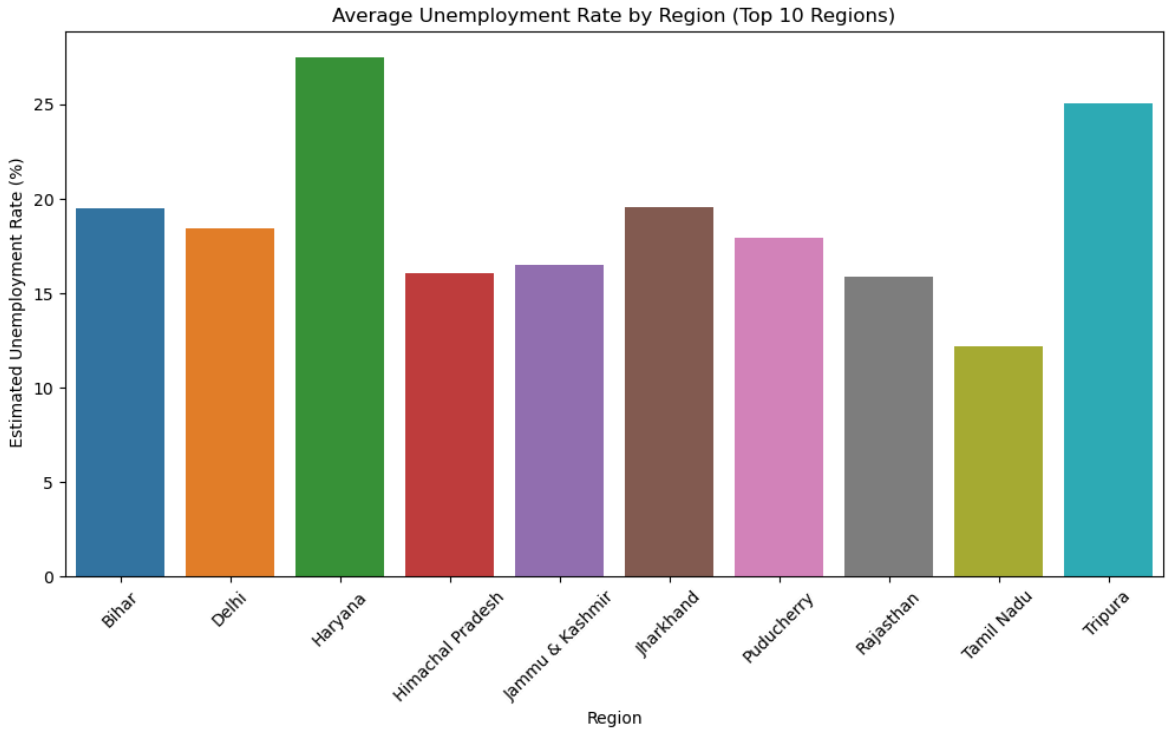


In [8]:
```
# Filter for top 5 regions with the highest average unemployment rates
top_regions = data.groupby('Region')[' Estimated Unemployment Rate (%)'].mean().
data_top_regions = data[data['Region'].isin(top_regions)]

# Unemployment Trends Over Time (Top 5 Regions)
plt.figure(figsize=(12, 6))
sns.lineplot(data=data_top_regions, x=' Date', y=' Estimated Unemployment Rate (
plt.title(' Unemployment Rate Over Time (Top 5 Regions)')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

```
C:\Users\pinto\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\pinto\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

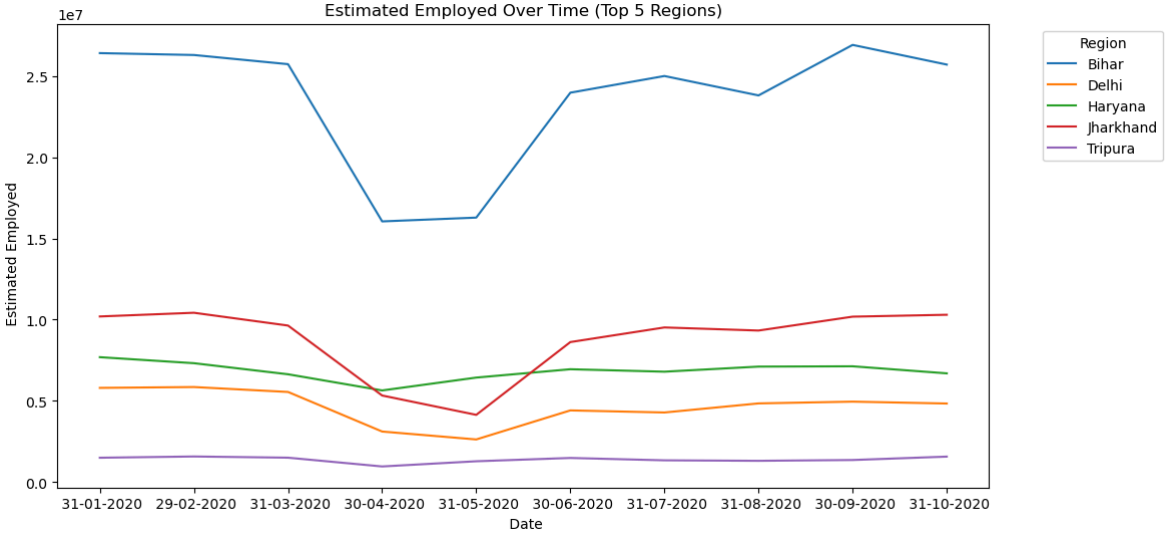


In [10]:
```
# Filter for top 10 regions with the highest average unemployment rates
top10_regions = data.groupby('Region')[' Estimated Unemployment Rate (%)'].mean(
data_top10_regions = data[data['Region'].isin(top10_regions)]

# Regional Analysis (Top 10 Regions)
regional_avg_unemployment = data_top10_regions.groupby('Region')[' Estimated Une
plt.figure(figsize=(12, 6))
sns.barplot(data=regional_avg_unemployment, x='Region', y=' Estimated Unemployme
plt.title('Average Unemployment Rate by Region (Top 10 Regions)')
plt.xticks(rotation=45)
plt.show()
```



In [11]:
```
# Employment Statistics (Top 5 Regions)
plt.figure(figsize=(12, 6))
sns.lineplot(data=data_top_regions, x=' Date', y=' Estimated Employed', hue='Reg
plt.title('Estimated Employed Over Time (Top 5 Regions)')
plt.legend(title='Region', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```

```
C:\Users\pinto\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\pinto\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarnin
g: use_inf_as_na option is deprecated and will be removed in a future version. Co
nvert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Car Price Prediction

Predicting car prices is crucial for both buyers and sellers in the automotive market, aiding buyers in making informed purchasing decisions and helping sellers set competitive prices. We will develop a car price prediction model using Python and machine learning techniques. We will begin by exploring and preprocessing a dataset containing various car features such as make, model, year, mileage, and transmission. After ensuring the data is clean and well-structured, we will apply different machine learning algorithms to train our model and evaluate its performance. The goal is to create a reliable model that accurately predicts car prices based on the provided features, offering valuable insights and techniques for predictive modeling tasks.

In [15]:
```
import pandas as pd
from sklearn.metrics import accuracy_score,confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
df = pd.read_csv('car data.csv')
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Car_Name       301 non-null    object 
 1   Year           301 non-null    int64  
 2   Selling_Price  301 non-null    float64
 3   Present_Price  301 non-null    float64
 4   Driven_kms     301 non-null    int64  
 5   Fuel_Type      301 non-null    object 
 6   Selling_type   301 non-null    object 
 7   Transmission   301 non-null    object 
 8   Owner          301 non-null    int64  
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
None
```

	Year	Selling_Price	Present_Price	Driven_kms	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.642584	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

In [16]:
```
df = df.dropna()  # Dropping missing values as an example
df = pd.get_dummies(df, columns=['Fuel_Type', 'Selling_type', 'Transmission'], d
# Encoding the 'Car_Name' column using label encoding
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Car_Name'] = label_encoder.fit_transform(df['Car_Name'])

X = df.drop(['Selling_Price'], axis=1)
y = df['Selling_Price']
```

Model Training

In [17]:
```
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, random_
from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()
model.fit(X_train, y_train)
```

Out[17]:
```
▾ RandomForestRegressor

RandomForestRegressor()
```

Model Evaluation

In [19]:
```
y_pred = model.predict(X_test)

#Evaluate the Model:
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
```

```
print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
print(f"MSE: {mean_squared_error(y_test, y_pred)}")
print(f"R-squared: {r2_score(y_test, y_pred)}")
```

```
MAE: 0.5667180327868857
MSE: 0.7860304104918038
R-squared: 0.9658775574025721
```

Deployment

In [20]:
```
#Model Deployment:
from flask import Flask, request, jsonify
import joblib

app = Flask(__name__)

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json()
    prediction = best_model.predict(pd.DataFrame([data]))
    return jsonify({'prediction': prediction[0]})

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 deployment.
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

C:\Users\pinto\anaconda3\Lib\site-packages\IPython\core\interactiveshell.py:3561:
UserWarning: To exit: use 'exit', 'quit', or Ctrl-D.
  warn("To exit: use 'exit', 'quit', or Ctrl-D.", stacklevel=1)
```

In [ ]:

In [ ]: