```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
```

```
url = "https://raw.githubusercontent.com/anbarasang2024aids-dev/foml/refs/heads/main/data.csv"
df = pd.read_csv(url)
print("✅ Dataset Loaded Successfully!")
print("Shape:", df.shape)
print(df.head())
```

```
✅ Dataset Loaded Successfully!
Shape: (511, 14)
      CRIM    ZN  INDUS  CHAS    NOX     RM   AGE     DIS  RAD  TAX  PTRATIO  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296     15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242     17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242     17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222     18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222     18.7

        B  LSTAT  MEDV
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

```
df = df.dropna()
print("\nAfter removing missing values:", df.shape)
```

```
After removing missing values: (506, 14)
```

```
df.columns = [col.strip().lower() for col in df.columns]
```

```
print("\nData Types:\n", df.dtypes)
```

```
Data Types:
 crim       float64
zn         float64
indus      float64
chas         int64
nox        float64
rm         float64
age        float64
dis        float64
rad          int64
tax          int64
ptratio    float64
b          float64
lstat      float64
medv       float64
dtype: object
```

```
if 'location' in df.columns:
      df = pd.get_dummies(df, columns=['location'], drop_first=True)
```

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

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)
```

```
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

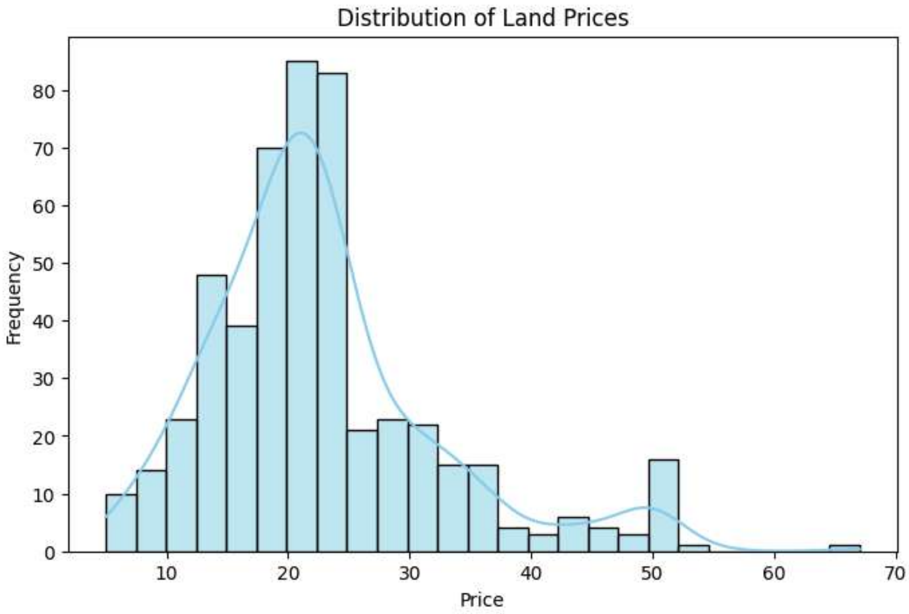
```
▾     RandomForestRegressor     ⓘ ?
RandomForestRegressor(random_state=42)
```

```
y_pred = model.predict(X_test)
```

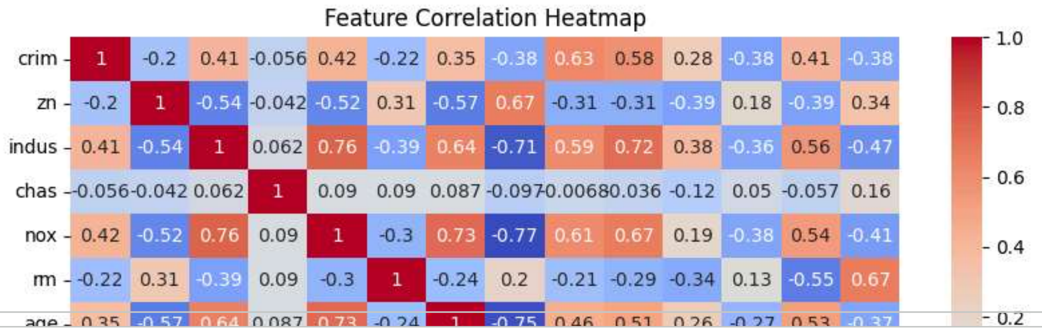
```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

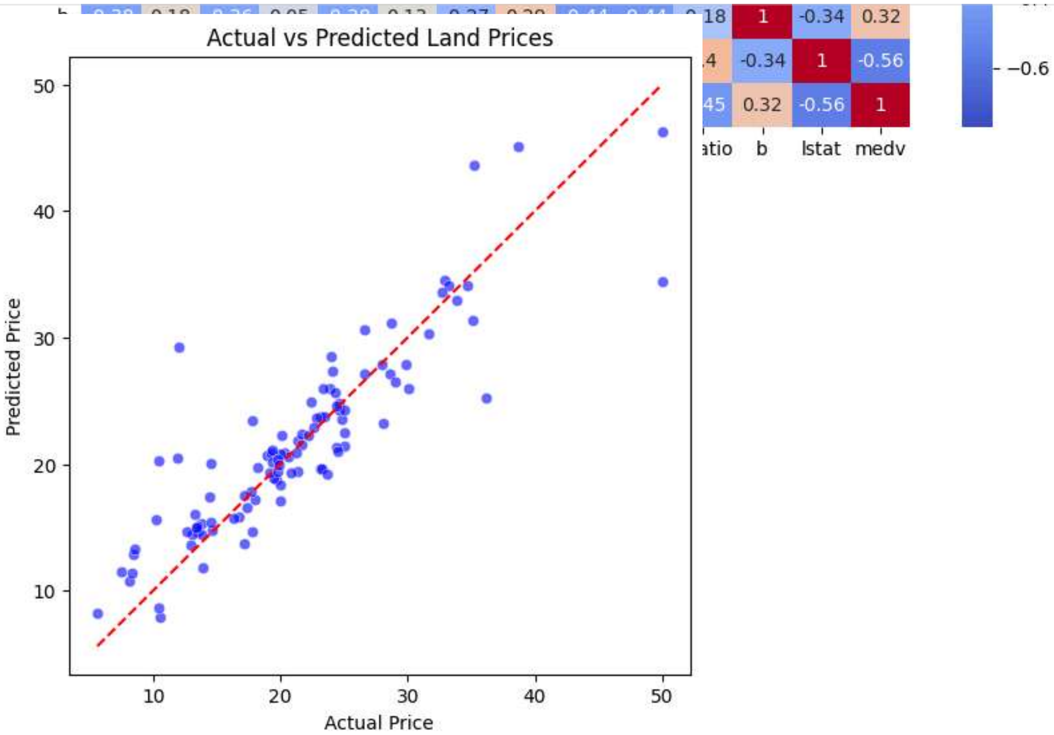
```
plt.figure(figsize=(8,5))
sns.histplot(df['medv'], kde=True, bins=25, color='skyblue')
plt.title("Distribution of Land Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()
```



```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()
```

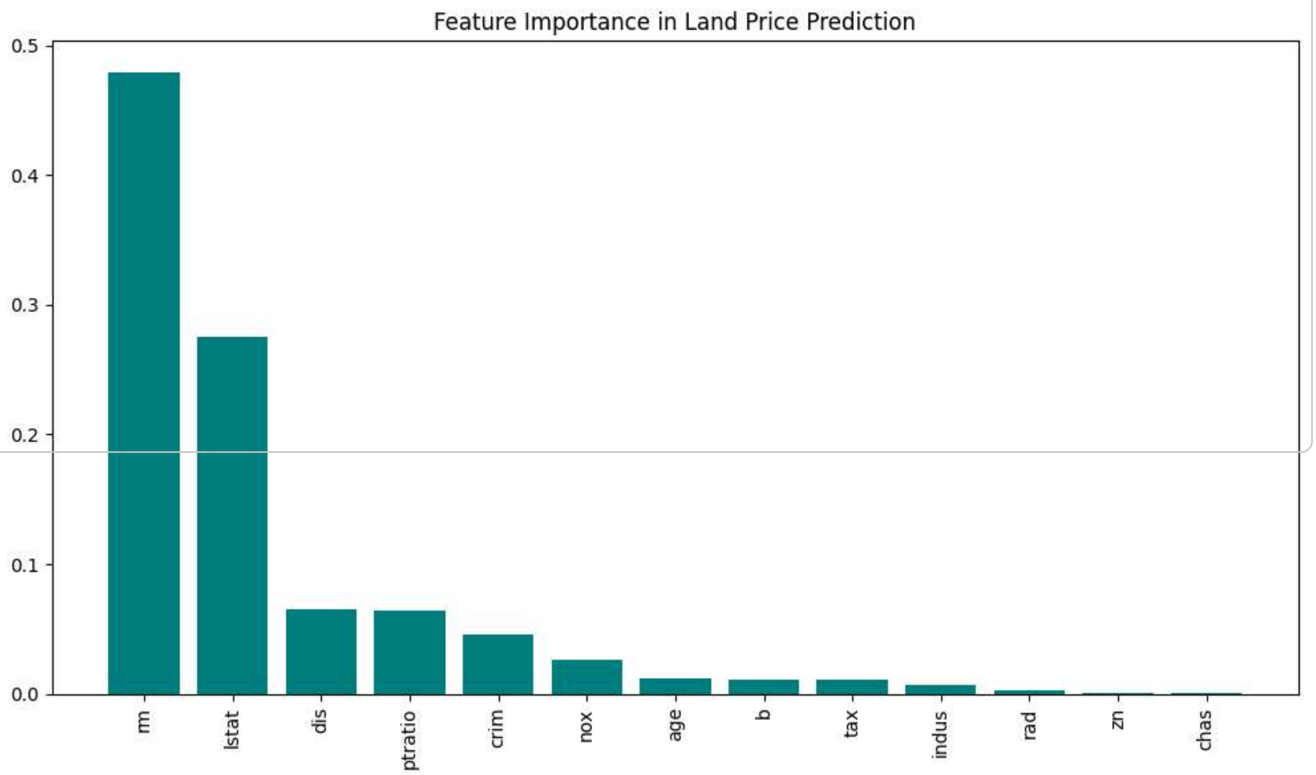


```
plt.figure(figsize=(6,6))
sns.scatterplot(x=y_test, y=y_pred, color='blue', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')  # Diagonal line
plt.xlabel("Actual Price")
plt.ylabel("Predicted Price")
plt.title("Actual vs Predicted Land Prices")
plt.show()
```



```
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]
```

```
plt.figure(figsize=(10,6))
plt.title("Feature Importance in Land Price Prediction")
plt.bar(range(len(indices)), importances[indices], color='teal', align='center')
plt.xticks(range(len(indices)), np.array(X.columns)[indices], rotation=90)
plt.tight_layout()
plt.show()
```



Feature Importance in Land Price Prediction