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


data_url = "http://lib.stat.cmu.edu/datasets/boston"
raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)

data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
target = raw_df.values[1::2, 2]


columns = [
    "CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE",
    "DIS", "RAD", "TAX", "PTRATIO", "B", "LSTAT"
]

df = pd.DataFrame(data, columns=columns)
df['MEDV'] = target



df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 506,\n  \"fields\": [\
n    {\n      \"column\": \"CRIM\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 8.601545105332487,\n
\"min\": 0.00632,\n        \"max\": 88.9762,\n
\"num_unique_values\": 504,\n        \"samples\": [\n
0.09178,\n          0.05644,\n          0.10574\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"ZN\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 23.322452994515036,\n
\"min\": 0.0,\n        \"max\": 100.0,\n        \"num_unique_values\":
26,\n        \"samples\": [\n          25.0,\n          30.0,\n
18.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"INDUS\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 6.8603529408975845,\n        \"min\": 0.46,\n        \"max\":
27.74,\n        \"num_unique_values\": 76,\n        \"samples\": [\n
8.14,\n          1.47,\n          1.22\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"CHAS\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 0.2539940413404118,\n
\"min\": 0.0,\n        \"max\": 1.0,\n        \"num_unique_values\":
2,\n        \"samples\": [\n          1.0,\n          0.0\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
```

```
}\n    },\n    {\n      \"column\": \"NOX\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\":
0.11587767566755611,\n        \"min\": 0.385,\n        \"max\":
0.871,\n        \"num_unique_values\": 81,\n        \"samples\": [\n
0.401,\n          0.538\n        ],\n        \"semantic_type\": \"\",\
n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"RM\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0.7026171434153237,\n        \"min\":
3.561,\n        \"max\": 8.78,\n        \"num_unique_values\": 446,\n
\"samples\": [\n          6.849,\n          4.88\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"AGE\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 28.148861406903638,\n
\"min\": 2.9,\n        \"max\": 100.0,\n        \"num_unique_values\":
356,\n        \"samples\": [\n          51.8,\n          33.8\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"DIS\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 2.1057101266276104,\
n        \"min\": 1.1296,\n        \"max\": 12.1265,\n
\"num_unique_values\": 412,\n        \"samples\": [\n
2.2955,\n          4.2515\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"RAD\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 8.707259384239377,\n        \"min\":
1.0,\n        \"max\": 24.0,\n        \"num_unique_values\": 9,\n
\"samples\": [\n          7.0,\n          2.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"TAX\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 168.53711605495926,\n
\"min\": 187.0,\n        \"max\": 711.0,\n
\"num_unique_values\": 66,\n        \"samples\": [\n          370.0,\n
666.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"PTRATIO\",\n      \"properties\": {\n        \"dtype\": \"number\",\
n        \"std\": 2.164945523714446,\n        \"min\": 12.6,\n
\"max\": 22.0,\n        \"num_unique_values\": 46,\n
\"samples\": [\n          19.6,\n          15.6\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"B\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 91.29486438415779,\n
\"min\": 0.32,\n        \"max\": 396.9,\n
\"num_unique_values\": 357,\n        \"samples\": [\n
396.24,\n          395.11\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"LSTAT\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 7.141061511348571,\n        \"min\":
1.73,\n        \"max\": 37.97,\n        \"num_unique_values\": 455,\n
\"samples\": [\n          6.15,\n          4.32\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
```

```
n    },\n    {\n      \"column\": \"MEDV\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 9.19710408737982,\n
\"min\": 5.0,\n        \"max\": 50.0,\n        \"num_unique_values\":
229,\n        \"samples\": [\n          14.1,\n          22.5\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

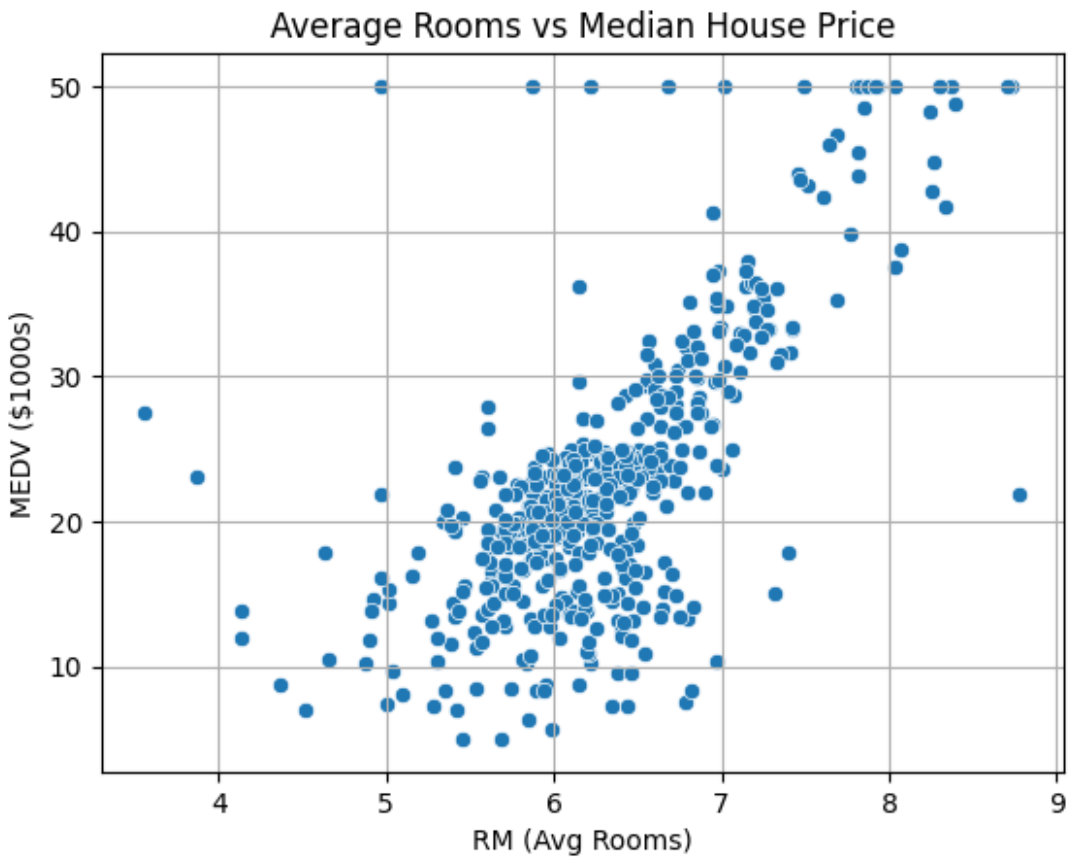
```
X = df.drop('MEDV', axis=1)
y = df['MEDV']

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

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

```
LinearRegression()
```

```
y_pred = model.predict(X_test)

print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R² Score:", r2_score(y_test, y_pred))
```

```
Mean Squared Error: 24.291119474973478
R² Score: 0.6687594935356326
```

```
sns.scatterplot(x=df['RM'], y=df['MEDV'])
plt.title("Average Rooms vs Median House Price")
plt.xlabel("RM (Avg Rooms)")
plt.ylabel("MEDV ($1000s)")
plt.grid(True)
plt.show()
```




```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


iris = load_iris()
X = iris.data
y = iris.target

df = pd.DataFrame(X, columns=iris.feature_names)
df['species'] = y




print(df.isnull().sum())
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
```

```
petal width (cm)     0
species              0
dtype: int64
```

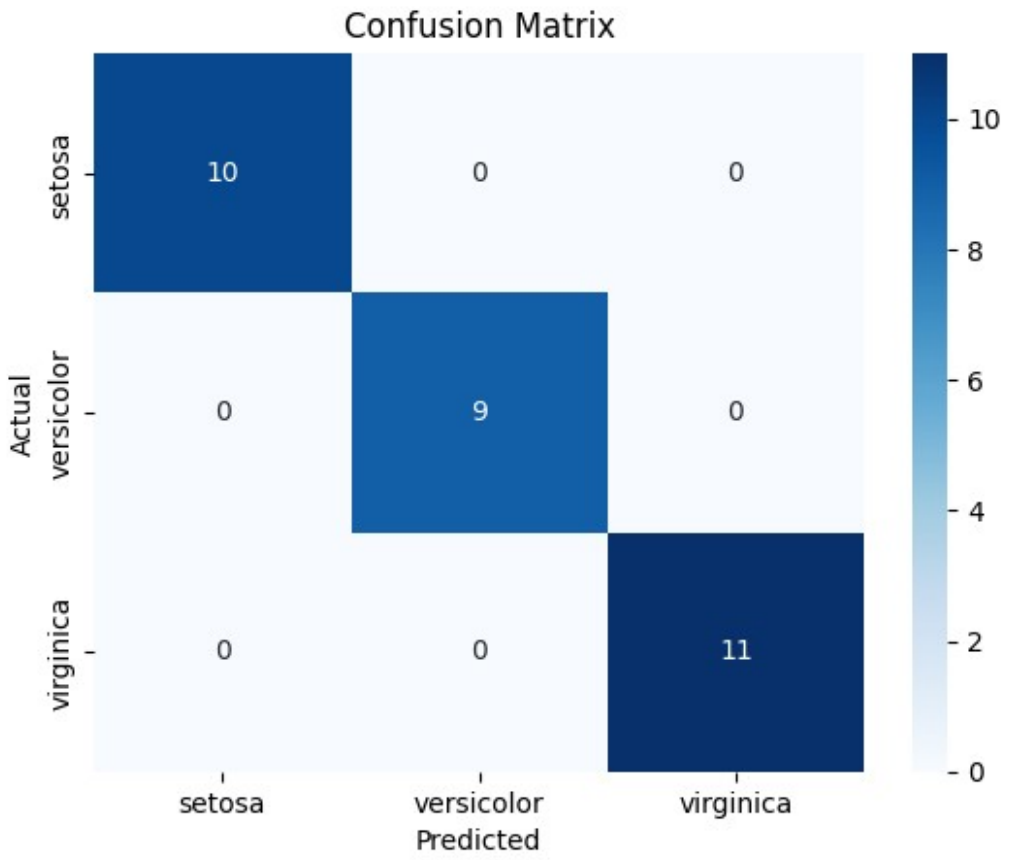
```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

model = LogisticRegression(max_iter=200)
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=200)
```

```
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

```
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap="Blues", fmt='d',
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```

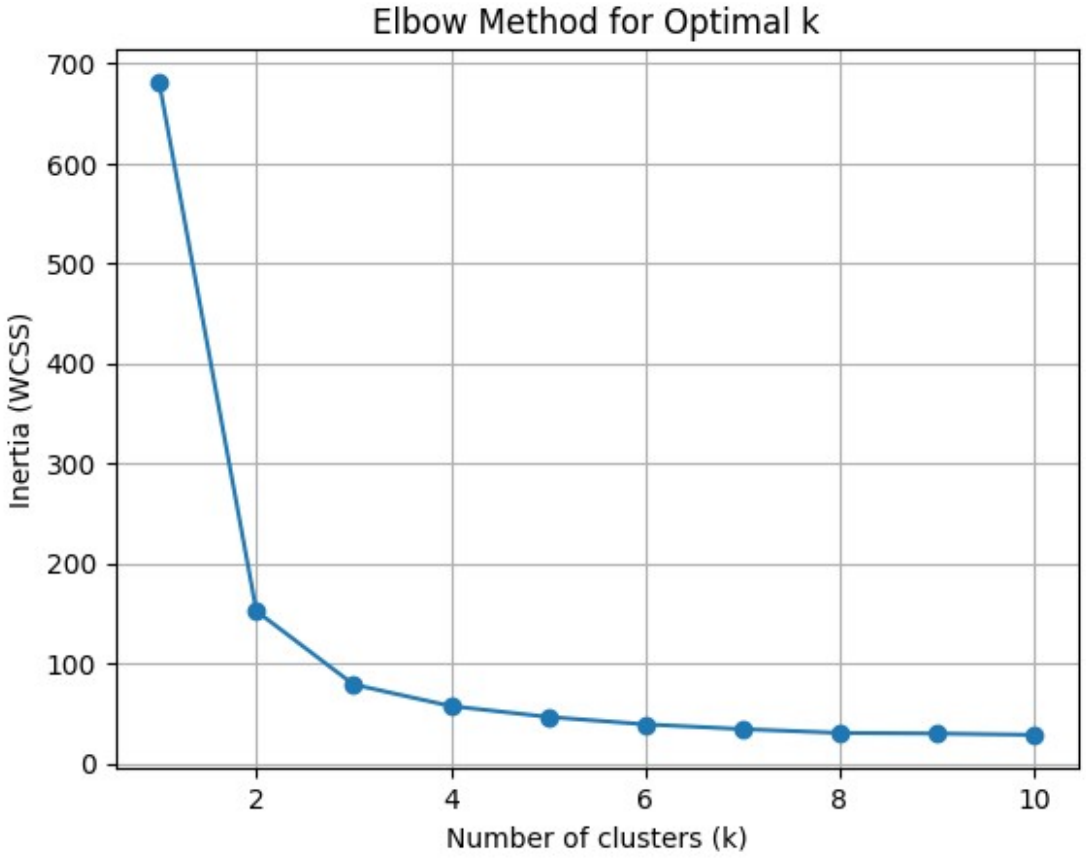



```
feature_importance = pd.Series(
    abs(model.coef_).mean(axis=0),
    index=iris.feature_names
).sort_values(ascending=False)

print("Feature Importance:\n", feature_importance)
```

```
Feature Importance:
 petal length (cm)    1.725414
petal width (cm)     1.183001
sepal width (cm)     0.641724
sepal length (cm)    0.338936
dtype: float64
```

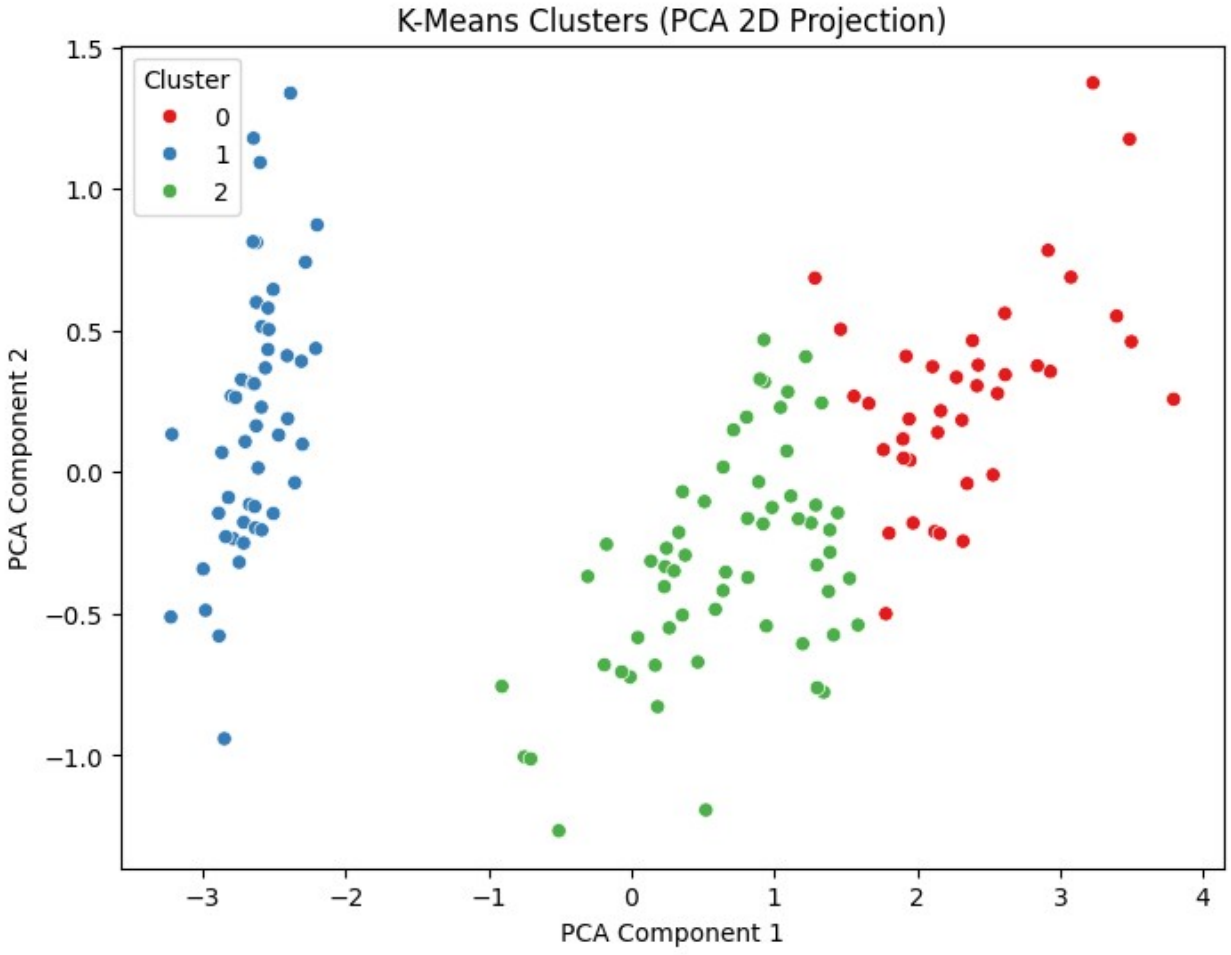
```
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

```
iris = load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = iris.target

inertia = []
k_range = range(1, 11)

for k in k_range:
    model = KMeans(n_clusters=k, random_state=42)
    model.fit(X)
    inertia.append(model.inertia_)

plt.plot(k_range, inertia, marker='o')
plt.title('Elbow Method for Optimal k')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia (WCSS)')
plt.grid(True)
plt.show()
```




```
kmeans = KMeans(n_clusters=3, random_state=42)
X['cluster'] = kmeans.fit_predict(X)
```

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(iris.data)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=X['cluster'],
palette='Set1')
plt.title("K-Means Clusters (PCA 2D Projection)")
plt.xlabel("PCA Component 1")
plt.ylabel("PCA Component 2")
plt.legend(title='Cluster')
plt.show()
```




```
from sklearn.metrics import adjusted_rand_score

ari = adjusted_rand_score(y, X['cluster'])
print("Adjusted Rand Index (ARI):", ari)
```

```
Adjusted Rand Index (ARI): 0.7163421126838476
```

```
#A: 3 clusters
#B
```