

Use K-means clustering to segment a dataset into different groups

Table of Contents

- [Description](#)
 - [Problem Statement](#)
 - [Prerequisites](#)
 - [Software Requirements](#)
 - [Hardware Requirements](#)
 - [Setup Instructions](#)
 - [Downloading the Dataset](#)
 - [Installing Required Libraries](#)
 - [Setting Up the Environment](#)
 - [Install Python](#)
 - [Install Visual Studio Code \(VSCode\)](#)
 - [Using K-means Clustering](#)
 - [Data Preprocessing](#)
 - [Applying K-means Clustering](#)
 - [Evaluating Clustering Performance](#)
 - [Visualizing Results](#)
 - [References](#)
-

Description

K-means clustering is an unsupervised machine learning algorithm that partitions data into **K** distinct groups based on feature similarities. In this lab, we will apply K-means clustering to a house prices dataset to uncover different price segments and analyze their characteristics.

Problem Statement

The goal of this lab is to segment the house prices dataset into different clusters to identify patterns and characteristics of properties in various price ranges. Understanding these segments can assist stakeholders in making informed decisions regarding pricing, marketing, and investment strategies.

Prerequisites

Completion of all previous lab guides (up to Lab Guide-05) is required before proceeding with Lab Guide-06.

Software Requirements

- **Python** : Python version 3.11.9
- **Visual Studio Code (VSCode)**: A lightweight code editor that provides powerful features for Python development, including extensions for linting, debugging, and version control.

Hardware Requirements

- A computer with at least **4 GB of RAM**.
- At least 1GB of free disk space.
- A GPU (optional, but recommended for faster training).

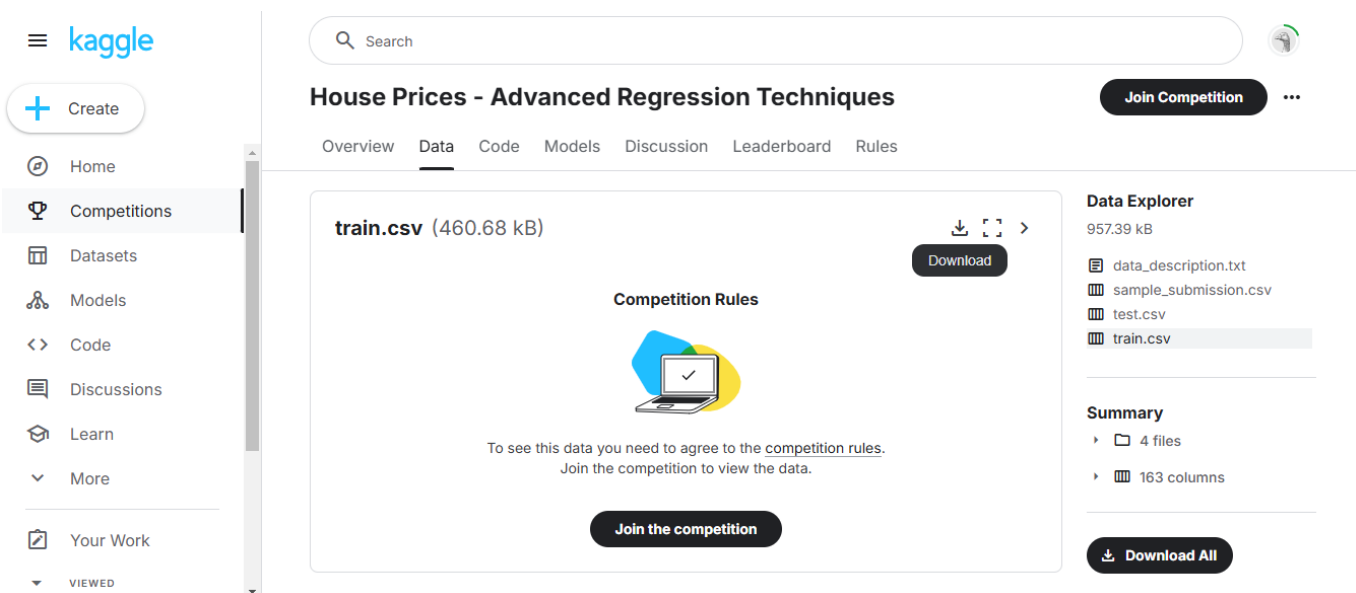
Setup Instructions

Downloading the Dataset

1. Sign in to Kaggle: Go to the [Kaggle website](#) and sign in to your account. If you don't have an account, create one.

2. Download the Dataset:

- Navigate to the [House Prices: Advanced Regression Techniques](#) competition page.



- Click on the "Data" tab and download the `train.csv` file (the dataset used for training).
- Move the downloaded `train.csv` file into your project directory.

Installing Required Libraries

You can install the required libraries using pip. Run the following command in your terminal or command prompt:

```
pip install pandas numpy matplotlib seaborn scikit-learn
```

```

PS C:\Users\Administrator\Desktop\AIML> pip install pandas numpy matplotlib seaborn scikit-learn
Collecting pandas
  Using cached pandas-2.2.3-cp311-cp311-win_amd64.whl.metadata (19 kB)
Collecting numpy
  Using cached numpy-2.1.2-cp311-cp311-win_amd64.whl.metadata (59 kB)
Collecting matplotlib
  Using cached matplotlib-3.9.2-cp311-cp311-win_amd64.whl.metadata (11 kB)
Collecting seaborn
  Using cached seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Collecting scikit-learn
  Using cached scikit_learn-1.5.2-cp311-cp311-win_amd64.whl.metadata (13 kB)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\program files\python311\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\program files\python311\lib\site-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in c:\program files\python311\lib\site-packages (from pandas) (2024.2)
Requirement already satisfied: contourpy>=1.0.1 in c:\program files\python311\lib\site-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\program files\python311\lib\site-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\program files\python311\lib\site-packages (from matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\program files\python311\lib\site-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in c:\program files\python311\lib\site-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=8 in c:\program files\python311\lib\site-packages (from matplotlib) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\program files\python311\lib\site-packages (from matplotlib) (3.2.0)
Requirement already satisfied: scipy>=1.6.0 in c:\program files\python311\lib\site-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in c:\program files\python311\lib\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\program files\python311\lib\site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\program files\python311\lib\site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Using cached pandas-2.2.3-cp311-cp311-win_amd64.whl (11.6 MB)
Using cached numpy-2.1.2-cp311-cp311-win_amd64.whl (12.9 MB)
Using cached matplotlib-3.9.2-cp311-cp311-win_amd64.whl (7.8 MB)
Using cached seaborn-0.13.2-py3-none-any.whl (294 kB)
Using cached scikit_learn-1.5.2-cp311-cp311-win_amd64.whl (11.0 MB)
Installing collected packages: numpy, pandas, scikit-learn, matplotlib, seaborn
Successfully installed matplotlib-3.9.2 numpy-2.1.2 pandas-2.2.3 scikit-learn-1.5.2 seaborn-0.13.2

```

Activate Windows
Go to Settings to activate Windows.

Setting Up the Environment

1. Install Python:

You can download and install Python 3.11.9 from the official Python website:

- Visit the [official Python website](https://www.python.org/).
- Locate a reliable version of Python 3, "**Download Python 3.11.9**".
- Choose the correct link for your device from the options provided: either Windows installer (64-bit) or Windows installer (32-bit) and proceed to download the executable file.

← → ↻ 🌐 <https://www.python.org/downloads/windows/>

■ Python 3.11.9 - April 2, 2024

Note that Python 3.11.9 cannot be used on Windows 7 or earlier.

- ➔ ■ Download [Windows installer \(64-bit\)](#)
- ➔ ■ Download [Windows installer \(32-bit\)](#)
- Download [Windows installer \(ARM64\)](#)
- Download [Windows embeddable package \(64-bit\)](#)
- Download [Windows embeddable package \(32-bit\)](#)
- Download [Windows embeddable package \(ARM64\)](#)

2. Install Visual Studio Code (VSCode):

Download and install VSCode from the official Visual Studio Code website:

[Download Visual Studio Code](#)

Using K-means Clustering

K-means Clustering K-means clustering is an unsupervised machine learning algorithm used to partition a dataset into K distinct groups (clusters) based on feature similarities. The algorithm works by randomly initializing K cluster centroids and iteratively refining them to minimize the distance between data points and their assigned centroids.

- **Create a new python file**
 - Create a Python file named `kmeans_clustering.py` and write following code in it.

Data Preprocessing

- [Import Libraries](#)
 - [Load the Dataset](#)
 - [Explore the Dataset](#)
 - [Check for Non-Numeric Entries](#)
 - [Convert Columns to Numeric](#)
 - [Fill Missing Values](#)
-

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

Load the Dataset

```
df = pd.read_csv('./train.csv')
```

Explore the Dataset

```
print("Head of the dataset:")
print(df.head())
print("\nData types:")
print(df.dtypes)
```

Run the Python file

- Use the command below in your terminal to run the Python file:

```
python Kmeans_clustering.py
```

Output

```
PS C:\Users\Administrator\Desktop\AIML> python Kmeans_clustering.py
Head of the dataset:
   Id  MSSubClass  MSZoning  LotFrontage  LotArea  Street  ...  MiscVal  MoSold  YrSold  SaleType  SaleCondition  SalePrice
0    1         60        RL         65.0     8450   Pave  ...         0         2    2008         WD         Normal     208500
1    2         20        RL         80.0     9600   Pave  ...         0         5    2007         WD         Normal     181500
2    3         60        RL         68.0    11250   Pave  ...         0         9    2008         WD         Normal     223500
3    4         70        RL         60.0     9550   Pave  ...         0         2    2006         WD      Abnorml     140000
4    5         60        RL         84.0    14260   Pave  ...         0        12    2008         WD         Normal     250000

[5 rows x 81 columns]

Data types:
Id                int64
MSSubClass        int64
MSZoning          object
LotFrontage       float64
LotArea           int64
...
MoSold            int64
YrSold            int64
SaleType          object
SaleCondition     object
SalePrice         int64
Length: 81, dtype: object
```

Check for Non-Numeric Entries

```
numeric_columns = ['LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
                   'SalePrice']
print("\nChecking for non-numeric entries:")
non_numeric_found = False # Flag to check if any non-numeric entries were found

for column in numeric_columns:
    non_numeric_entries = df[~df[column].apply(lambda x: isinstance(x, (int,
float)))][column]
    if not non_numeric_entries.empty:
        print(f"Non-numeric entries found in {column}:")
        print(non_numeric_entries)
        non_numeric_found = True

if not non_numeric_found:
```

```
print("No non-numeric entries found in the specified columns.")
```

Output

```
PS C:\Users\Administrator\Desktop\AIML> python Kmeans_clustering.py

Checking for non-numeric entries:
No non-numeric entries found in the specified columns.
```

Convert Columns to Numeric

```
for column in numeric_columns:
    df[column] = pd.to_numeric(df[column], errors='coerce')

print("\nChecking for NaN values after conversion:")
print(df[numeric_columns].isnull().sum())
```

Output

```
PS C:\Users\Administrator\Desktop\AIML> python Kmeans_clustering.py

Checking for non-numeric entries:

Checking for NaN values after conversion:
LotArea      0
OverallQual  0
OverallCond  0
YearBuilt    0
SalePrice    0
dtype: int64
```

Fill Missing Values

```
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].mean())
```

Applying K-means Clustering

- [Feature Selection](#)
- [Standardize the Data](#)
- [Determine Optimal Number of Clusters \(Elbow Method\)](#)
- [Fit K-means Model with Optimal k](#)

Feature Selection

```
features = df[numeric_columns]
print("\nFeatures DataFrame shape:")
print(features.shape)
```

Run the Python file

- Use the command below in your terminal to run the Python file:

```
python data_preprocessing.py
```

Output

```
PS C:\Users\Administrator\Desktop\AIML> python Kmeans_clustering.py
Features DataFrame shape:
(1460, 5)
```

Standardize the Data

- **Standardization:** A preprocessing step to scale features to have a mean of zero and a standard deviation of one.

```
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

Determine Optimal Number of Clusters (Elbow Method)

- **Elbow Method:** A technique to determine the optimal number of clusters by plotting the inertia against the number of clusters.

```
inertia = []
K = range(1, 11) # Testing for k from 1 to 10

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=0)
    kmeans.fit(features_scaled)
    inertia.append(kmeans.inertia_)

plt.plot(K, inertia, 'bx-')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
```

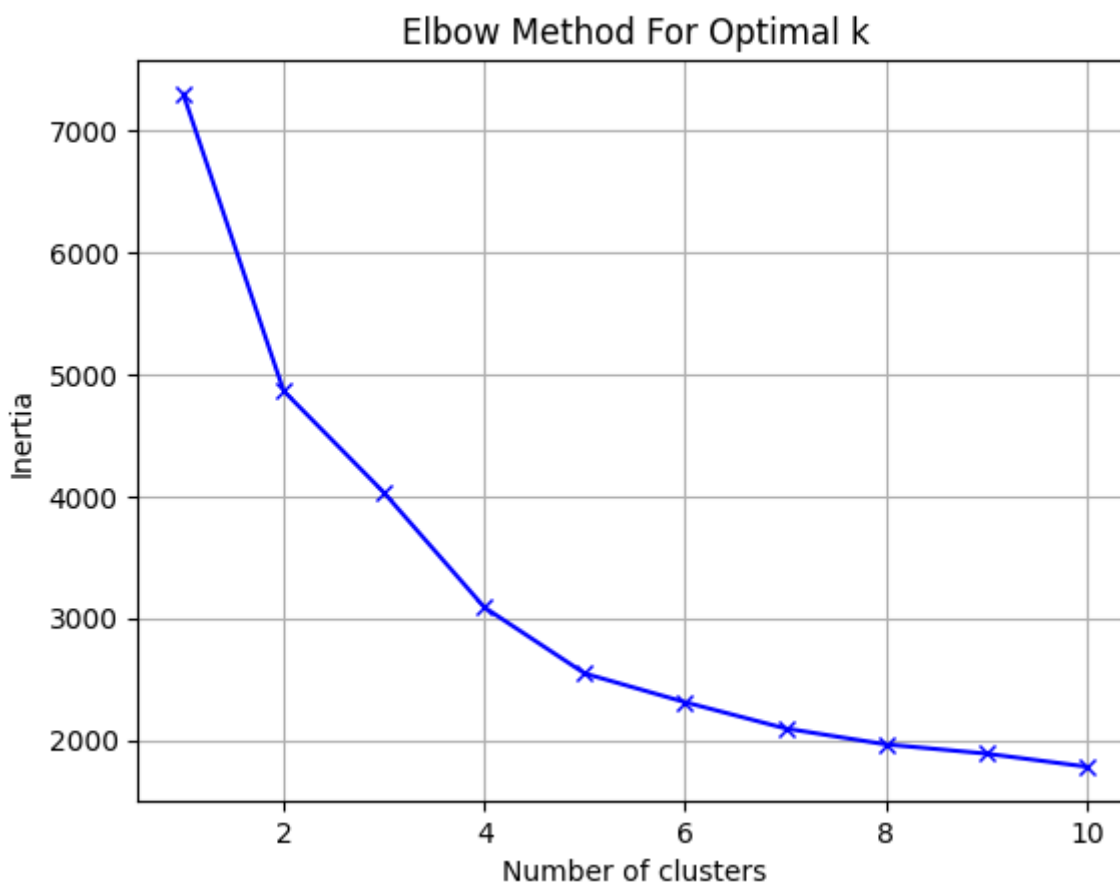
```
plt.grid()  
plt.show()
```

Run the Python file

- Use the command below in your terminal to run the Python file:

```
python data_preprocessing.py
```

Output



Fit K-means Model with Optimal k

```
optimal_k = 3 # Replace with the optimal k from the Elbow Method  
kmeans = KMeans(n_clusters=optimal_k, random_state=0)  
kmeans.fit(features_scaled)  
  
df['Cluster'] = kmeans.labels_
```

Evaluating Clustering Performance

Analyze Cluster Characteristics


```
try:
    cluster_analysis = df.groupby('Cluster')[numeric_columns].mean()
    print("\nCluster Analysis:")
    print(cluster_analysis)
except Exception as e:
    print("Error in cluster analysis:", e)
```

Run the Python file

- Use the command below in your terminal to run the Python file:

Output

```
PS C:\Users\Administrator\Desktop\AIML> python Kmeans_clustering.py

Cluster Analysis:

```

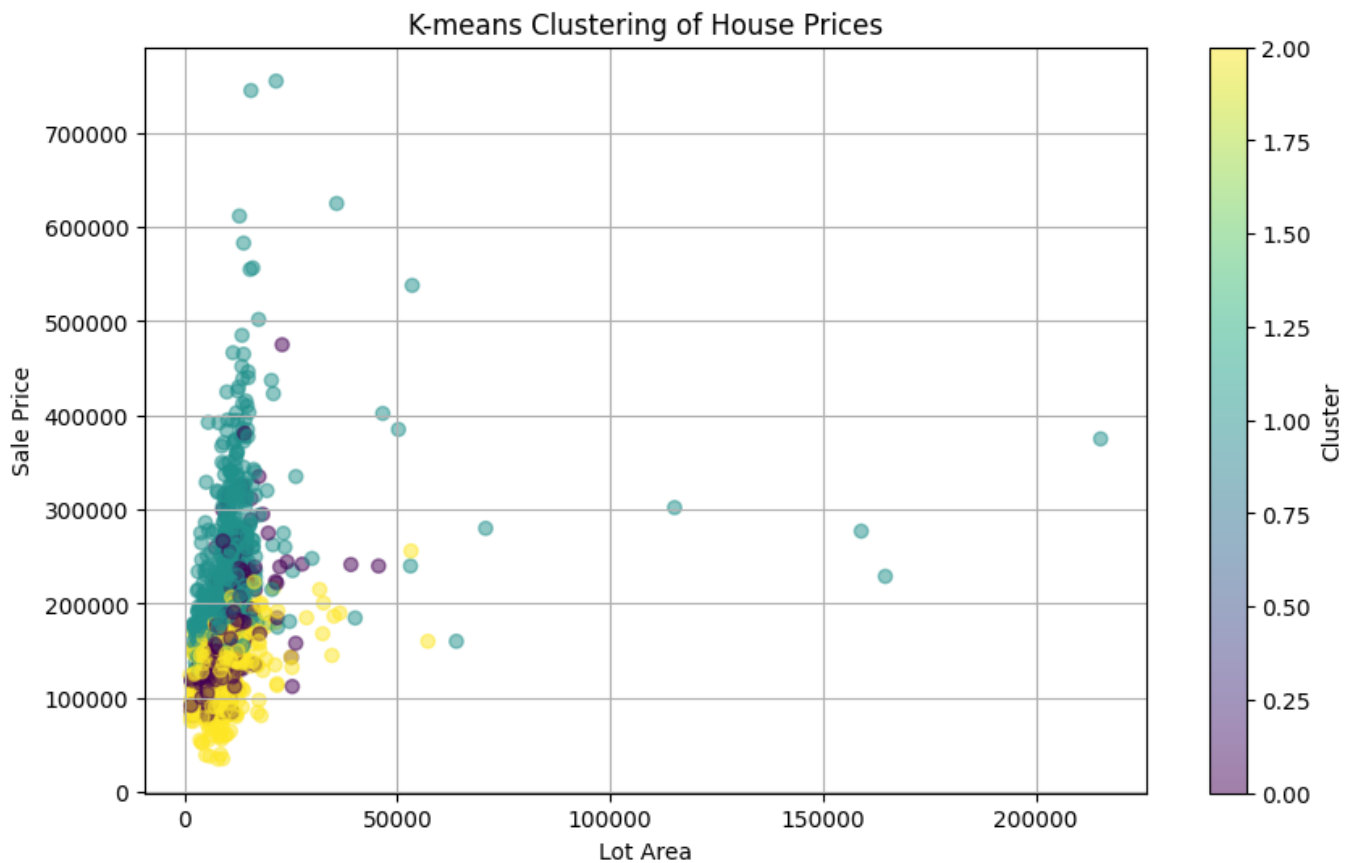
	LotArea	OverallQual	OverallCond	YearBuilt	SalePrice
Cluster					
0	9448.029851	5.582090	7.188060	1941.083582	153162.477612
1	11974.166667	7.408602	5.103943	1999.344086	246318.804659
2	9714.098765	5.116402	5.086420	1961.470899	132962.298060

Visualizing Results

Visualize Clusters

```
plt.figure(figsize=(10, 6))
plt.scatter(df['LotArea'], df['SalePrice'], c=df['Cluster'], cmap='viridis',
            alpha=0.5)
plt.xlabel('Lot Area')
plt.ylabel('Sale Price')
plt.title('K-means Clustering of House Prices')
plt.colorbar(label='Cluster')
plt.grid()
plt.show()
```

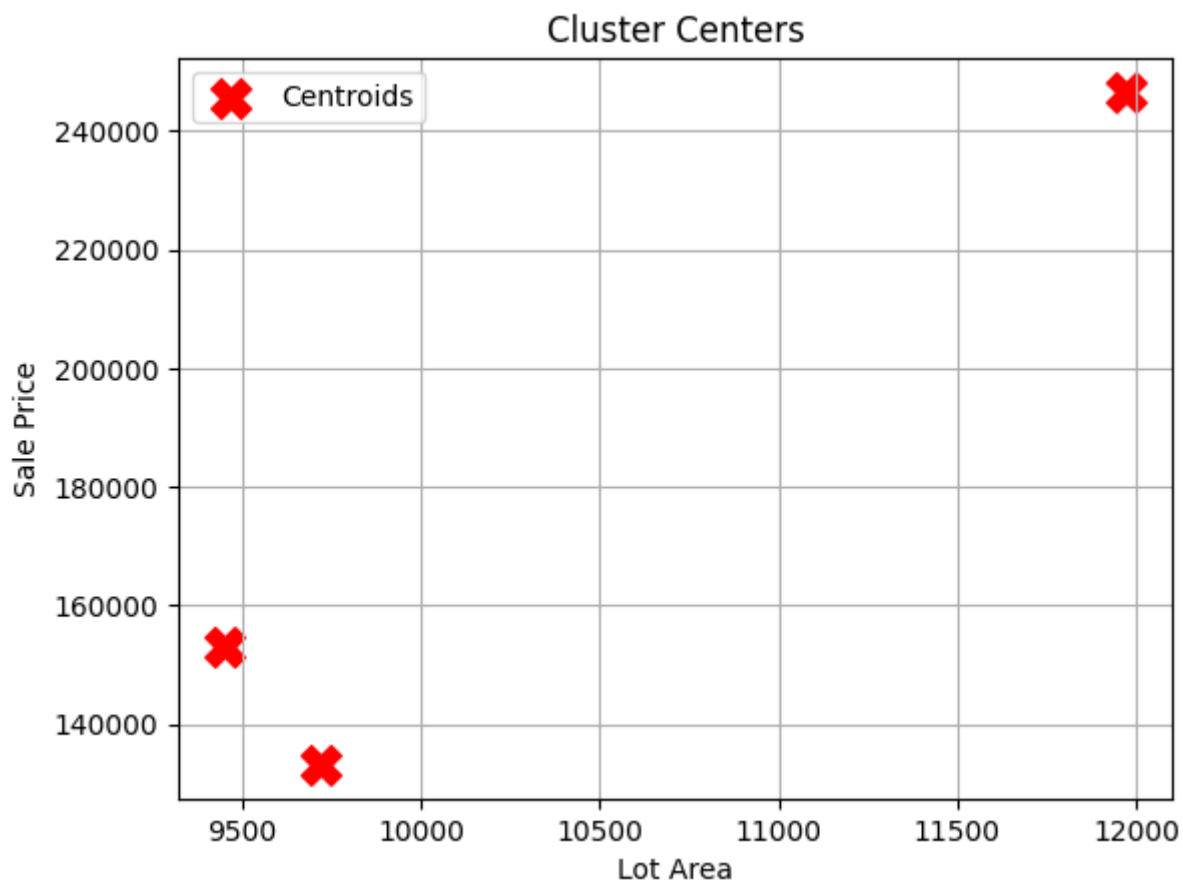
Output



Visualize Cluster Centers

```
centers = scaler.inverse_transform(kmeans.cluster_centers_)
plt.scatter(centers[:, 0], centers[:, 4], c='red', marker='X', s=200,
            label='Centroids')
plt.xlabel('Lot Area')
plt.ylabel('Sale Price')
plt.title('Cluster Centers')
plt.legend()
plt.grid()
plt.show()
```

Output



References

- [K-means Clustering Documentation - Scikit-learn](#)
- [StandardScaler Documentation - Scikit-learn](#)
- [Matplotlib Documentation](#)