

Group Members

Name:- Chirag Ghuge (222)

Sarthak Jagdale (226)

code of K means:

```
import pandas as pd
from sklearn.cluster import KMeans

# Read the CSV file into a DataFrame
df = pd.read_csv("/content/sample_data/Cardekho1.csv")

# Select the features you want to use for clustering
selected_features = ['selling_price', 'km_driven']

# Extract the selected features from the DataFrame
X = df[selected_features]

# Create a KMeans object with the desired number of clusters
kmeans = KMeans(n_clusters=3)

# Fit the KMeans model to the data
kmeans.fit(X)

# Get the cluster labels assigned to each data point
labels = kmeans.labels_

# Get the cluster centers
cluster_centers = kmeans.cluster_centers_

# Add the cluster labels to the DataFrame
df['cluster'] = labels

# Print the updated DataFrame
print(df)

# Print the cluster centers
print("Cluster Centers:")
for center in cluster_centers:
    print(center)
```

Output:

	name	year	selling_price	km_driven	fuel \
0	Maruti 800 AC	2007	60000	70000	Petrol
1	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol
2	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel
3	Datsun RediGO T Option	2017	250000	46000	Petrol
4	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel
5	Maruti Alto LX BSIII	2007	140000	125000	Petrol
6	Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol
7	Tata Indigo Grand	Petrol 2014	240000	60000	Petrol

8	Hyundai Creta 1.6 VTVT S	2015	850000	25000	Petrol
9	Maruti Celerio Green VXI	2017	365000	78000	CNG
10	Chevrolet Sail 1.2 Base	2015	260000	35000	Petrol
11	Tata Indigo Grand	2014	250000	100000	Petrol
12	Toyota Corolla Altis 1.8 VL CVT	2018	1650000	25000	Petrol
13	Maruti 800 AC	2007	60000	70000	Petrol
14	Maruti Wagon R LXI Minor	2007	135000	50000	Petrol
15	Hyundai Verna 1.6 SX	2012	600000	100000	Diesel
16	Datsun RediGO T Option	2017	250000	46000	Petrol
17	Honda Amaze VX i-DTEC	2014	450000	141000	Diesel
18	Maruti Alto LX BSIII	2007	140000	125000	Petrol

	seller_type	transmission	owner	cluster
0	Individual	Manual	First Owner	0
1	Individual	Manual	First Owner	0
2	Individual	Manual	First Owner	1
3	Individual	Manual	First Owner	0
4	Individual	Manual	Second Owner	1
5	Individual	Manual	First Owner	0
6	Individual	Manual	First Owner	1
7	Individual	Manual	Second Owner	0
8	Individual	Manual	First Owner	1
9	Individual	Manual	First Owner	0
10	Individual	Manual	First Owner	0
11	Individual	Manual	First Owner	0
12	Dealer	Automatic	First Owner	2
13	Individual	Manual	First Owner	0
14	Individual	Manual	First Owner	0
15	Individual	Manual	First Owner	1
16	Individual	Manual	First Owner	0
17	Individual	Manual	Second Owner	1
18	Individual	Manual	First Owner	0

Cluster Centers:

[190416.66666667 71250.]

[583333.33333333 88666.66666667]

[1650000. 25000.]

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of 'n_init' will change from 10 to 'auto' in 1.4. Set the value of 'n_init' explicitly to suppress the warning
warnings.warn(

Code of KNN:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
```

Read the CSV file into a DataFrame

```
df = pd.read_csv("/content/sample_data/car dekkho.csv")
```

Select the features and target variable

```
selected_features = ['year', 'selling_price', 'km_driven', 'fuel', 'seller_type', 'transmission', 'owner']
```

```
target_variable = 'name'
```

```
# Extract the selected features and target variable from the DataFrame
```

```
X = df[selected_features]
```

```
y = df[target_variable]
```

```
# Encode categorical variables
```

```
encoder = LabelEncoder()
```

```
X['fuel'] = encoder.fit_transform(X['fuel'])
```

```
X['seller_type'] = encoder.fit_transform(X['seller_type'])
```

```
X['transmission'] = encoder.fit_transform(X['transmission'])
```

```
X['owner'] = encoder.fit_transform(X['owner'])
```

```
# Split the dataset into training and testing sets
```

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

```
# Create a KNN classifier with k=3
```

```
knn = KNeighborsClassifier(n_neighbors=3)
```

```
# Fit the classifier to the training data
```

```
knn.fit(X_train, y_train)
```

```
# Predict the target variable for the test set
```

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

```
# Calculate the accuracy of the model
```

```
accuracy = accuracy_score(y_test, y_pred)
```

```
print("Accuracy:", accuracy)
```

Output

Accuracy: 0.14285714285714285

<ipython-input-11-09911d8e3e6f>:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
X['fuel'] = encoder.fit_transform(X['fuel'])
```

<ipython-input-11-09911d8e3e6f>:21: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
X['seller_type'] = encoder.fit_transform(X['seller_type'])
```

<ipython-input-11-09911d8e3e6f>:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
X['transmission'] = encoder.fit_transform(X['transmission'])
```

<ipython-input-11-09911d8e3e6f>:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.

Try using `.loc[row_indexer,col_indexer] = value` instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

```
X['owner'] = encoder.fit_transform(X['owner'])
```

Code of Linear Regression:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Read the CSV file into a DataFrame
df = pd.read_csv("/content/sample_data/car dekkho.csv")

# Select the features and target variable
selected_features = ['year', 'km_driven']
target_variable = 'selling_price'

# Extract the selected features and target variable from the DataFrame
X = df[selected_features]
y = df[target_variable]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create a LinearRegression model
model = LinearRegression()

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

# Make predictions on the test data
y_pred = model.predict(X_test)

# Evaluate the model using mean squared error
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

Output:

Mean Squared Error: 255707328588.57532