EXPERIMENT NO. 1

TITLE : Perform EDA and Visualise the relationships for a given dataset.

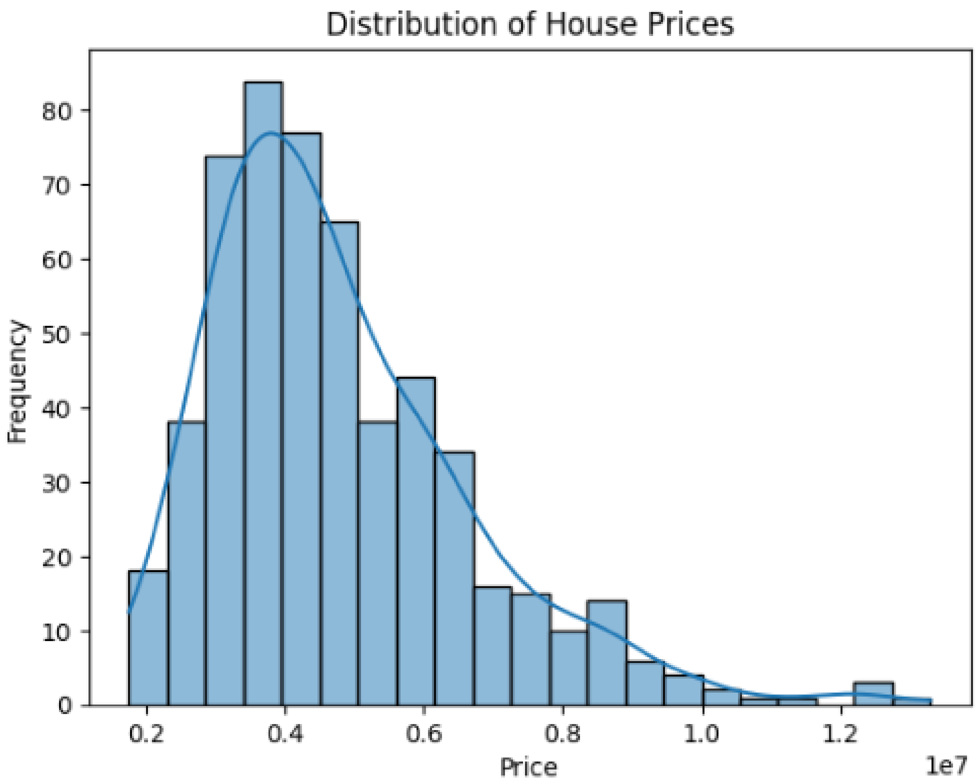
CODE :

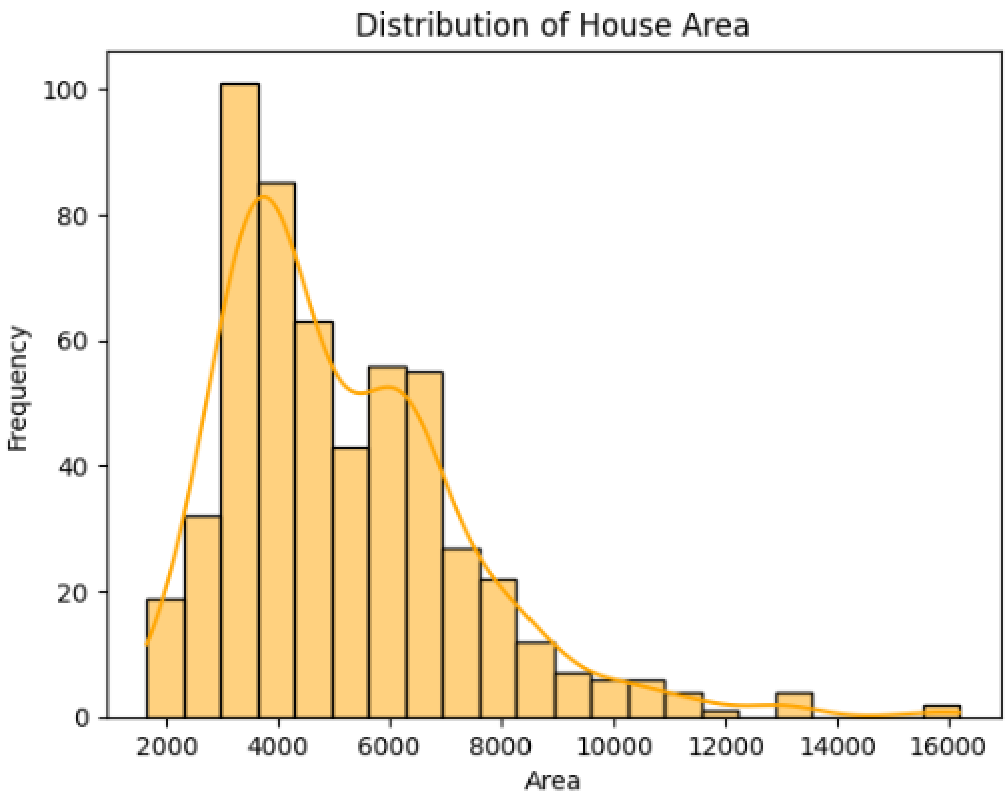
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the Dataset
data = pd.read_csv("Housing.csv")
print(data.info())
print(data.describe())

# Check for Missing Values
print("Missing Values:\n", data.isnull().sum())

# Univariate Analysis: Distribution of Prices
sns.histplot(data['price'], kde=True)
plt.title("Distribution of House Prices")
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()

# Distribution of Area
sns.histplot(data['area'], kde=True, color='orange')
plt.title("Distribution of House Area")
plt.xlabel("Area")
plt.ylabel("Frequency")
plt.show()

# Bar Plot: Furnishing Status
sns.countplot(x="furnishingstatus", data=data, palette="Set2")
```

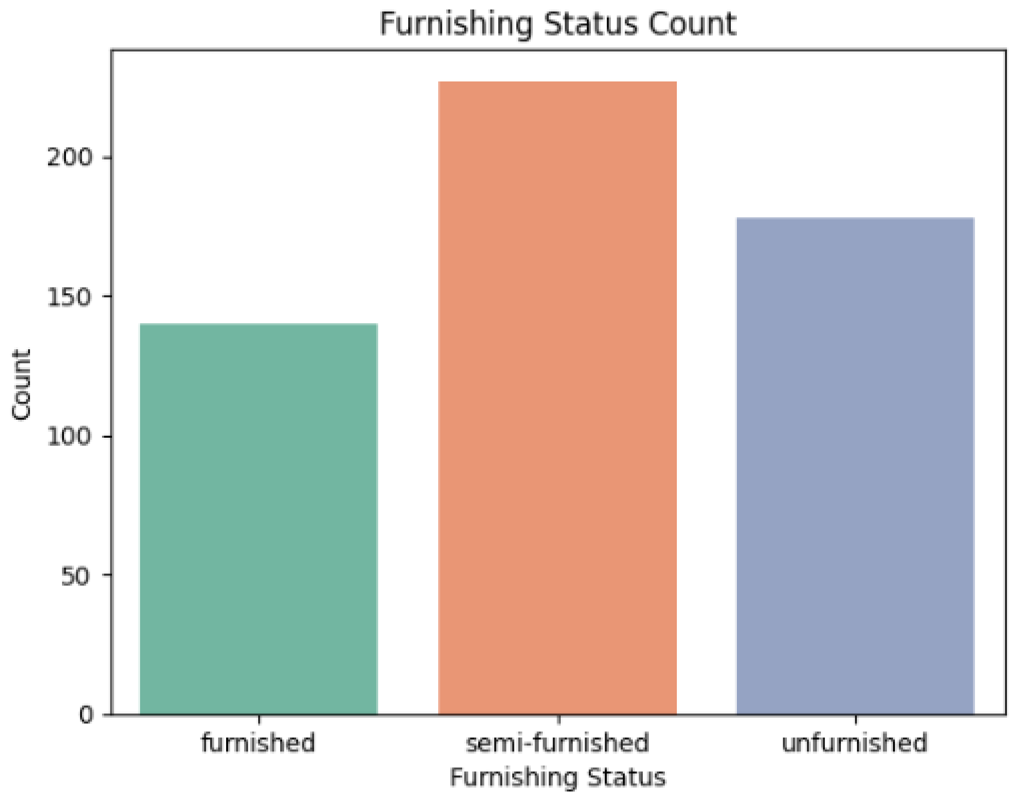
```python
plt.title("Furnishing Status Count")
plt.xlabel("Furnishing Status")
plt.ylabel("Count")
plt.show()

# Multivariate Analysis: Price vs Area
sns.scatterplot(x="area", y="price", hue="furnishingstatus", data=data, palette="cool")
plt.title("Price vs Area")
plt.xlabel("Area")
plt.ylabel("Price")
plt.show()

# Correlation Heatmap
sns.heatmap(data.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()

# Price vs Number of Bedrooms
sns.boxplot(x="bedrooms", y="price", data=data, palette="Set3")
plt.title("Price vs Number of Bedrooms")
plt.xlabel("Number of Bedrooms")
plt.ylabel("Price")
plt.show()
```

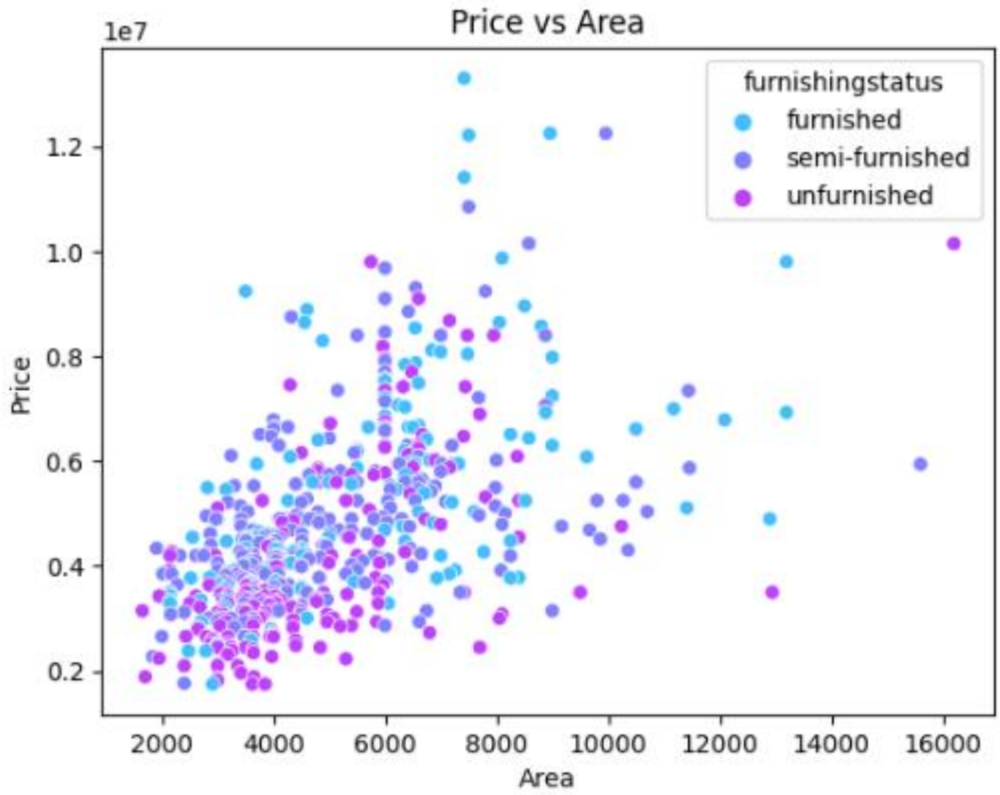
OUTPUT :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   price             545 non-null    int64 
 1   area              545 non-null    int64 
 2   bedrooms          545 non-null    int64 
 3   bathrooms         545 non-null    int64 
 4   stories           545 non-null    int64 
 5   mainroad          545 non-null    object
 6   guestroom         545 non-null    object
 7   basement          545 non-null    object
 8   hotwaterheating   545 non-null    object
 9   airconditioning   545 non-null    object
 10  parking           545 non-null    int64 
 11  prefarea          545 non-null    object
 12  furnishingstatus  545 non-null    object
dtypes: int64(6), object(7)
memory usage: 55.5+ KB
None
              price          area    bedrooms   bathrooms     stories  \
count  5.450000e+02    545.000000  545.000000  545.000000  545.000000   
mean   4.766729e+06   5150.541284    2.965138    1.286239    1.805505   
std    1.870440e+06   2170.141023    0.738064    0.502470    0.867492   
min    1.750000e+06   1650.000000    1.000000    1.000000    1.000000   
25%    3.430000e+06   3600.000000    2.000000    1.000000    1.000000   
50%    4.340000e+06   4600.000000    3.000000    1.000000    2.000000   
75%    5.740000e+06   6360.000000    3.000000    2.000000    2.000000   
max    1.330000e+07  16200.000000    6.000000    4.000000    4.000000   

          parking  
count  545.000000  
mean     0.693578  
std      0.861586  
min      0.000000  
25%      0.000000  
50%      0.000000  
75%      1.000000  
max      3.000000  
Missing Values:
 price               0
area                0
bedrooms            0
bathrooms           0
stories             0
mainroad            0
guestroom           0
basement            0
hotwaterheating     0
airconditioning     0
parking             0
prefarea            0
furnishingstatus    0
dtype: int64
```


Distribution of House Prices
Frequency
0
10
20
30
40
50
60
70
80
0.2
0.4
0.6
0.8
1.0
1.2
Price
1e7


Distribution of House Area
Frequency
0
20
40
60
80
100
2000
4000
6000
8000
10000
12000
14000
16000
Area


Furnishing Status Count
Count
0
50
100
150
200
furnished
semi-furnished
unfurnished
Furnishing Status


Price vs Area
1e7
Price
0.2
0.4
0.6
0.8
1.0
1.2
2000
4000
6000
8000
10000
12000
14000
16000
Area
furnishingstatus
furnished
semi-furnished
unfurnished

EXPERIMENT NO. 2

TITLE : Perform EDA and Visualise the relationships and also train a Linear regression model and evaluate model's performance for Housing Price dataset.

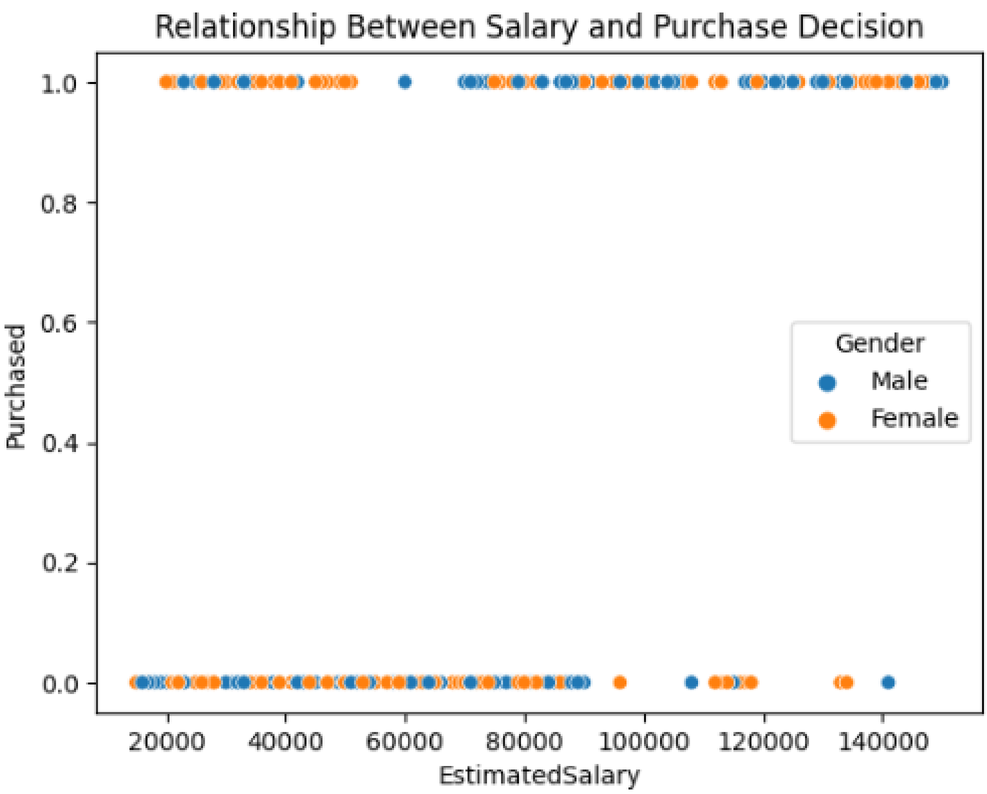
CODE :

```
import pandas as pd
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
# Load Dataset
data = pd.read_csv("User_Data.csv")
print(data.info())

# EDA: Visualize the relationships
sns.scatterplot(x="EstimatedSalary", y="Purchased", hue="Gender", data=data)
plt.title("Relationship Between Salary and Purchase Decision")
plt.show()

# Preprocessing
data = pd.get_dummies(data, columns=["Gender"], drop_first=True)
X = data[["Age", "EstimatedSalary", "Gender_Male"]]
y = data["Purchased"]

# Split 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)
# Train Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)
# Make Predictions
```

```
y_pred = model.predict(X_test)

# Evaluate Model

rmse = mean_squared_error(y_test, y_pred, squared=False)

r2 = r2_score(y_test, y_pred)

print("RMSE:", rmse)

print("R² Score:", r2)
```

OUTPUT :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
None
```




```
RMSE: 0.315190846926146
R² Score: 0.5633174945669398
```

EXPERIMENT NO. 3

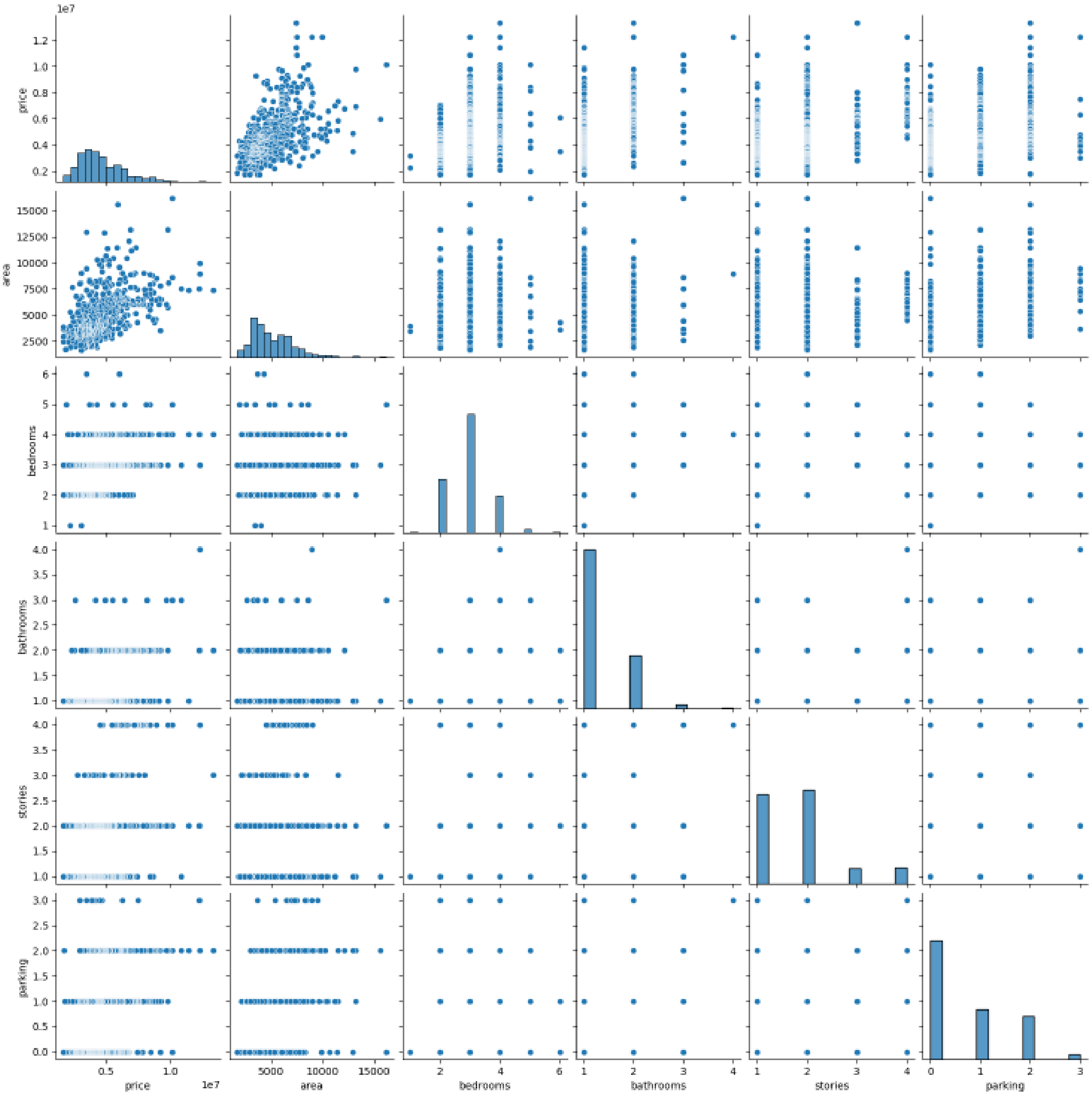
TITLE : Perform EDA and Visualise the relationships and also train a Ridge regression model and evaluate model's performance for given dataset.

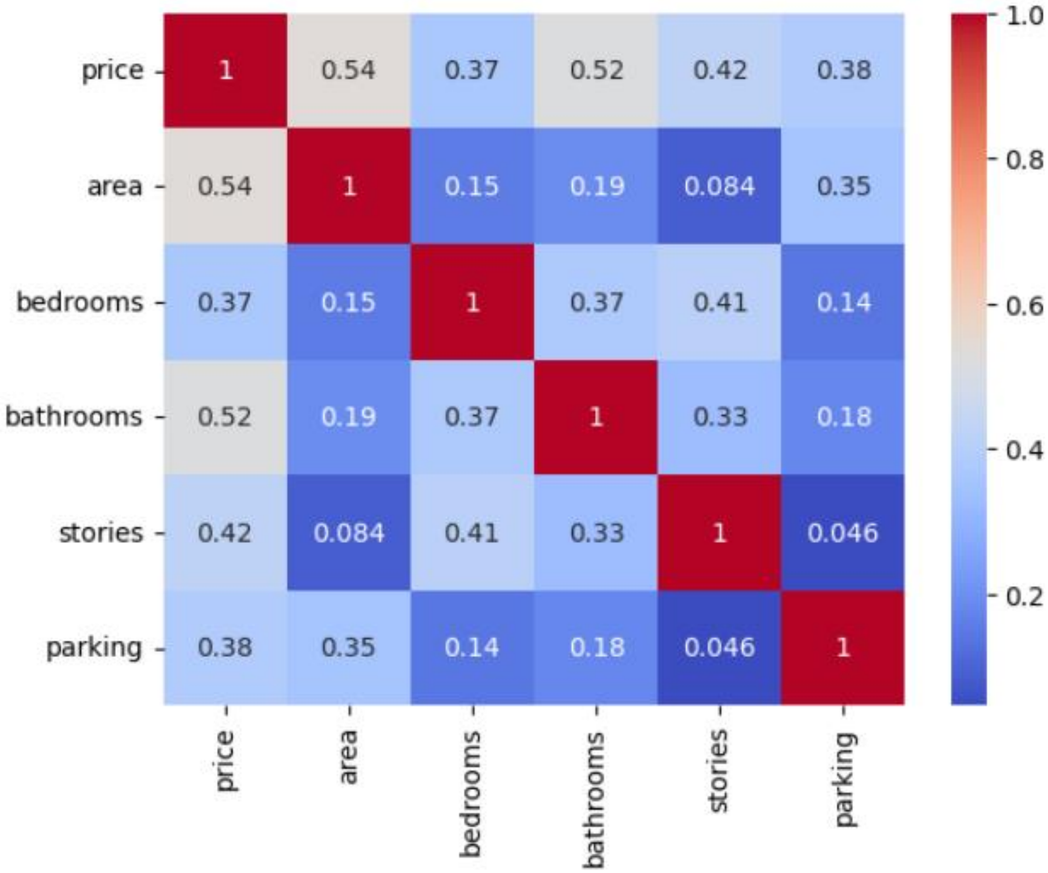
CODE :

```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv("Housing.csv")
print(df.shape)

# EDA
sns.pairplot(df)
plt.show()
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.show()

# Preprocessing
df = pd.get_dummies(df, drop_first=True)
X = df.drop("price", axis=1)
y = df["price"]

# Standardization
```

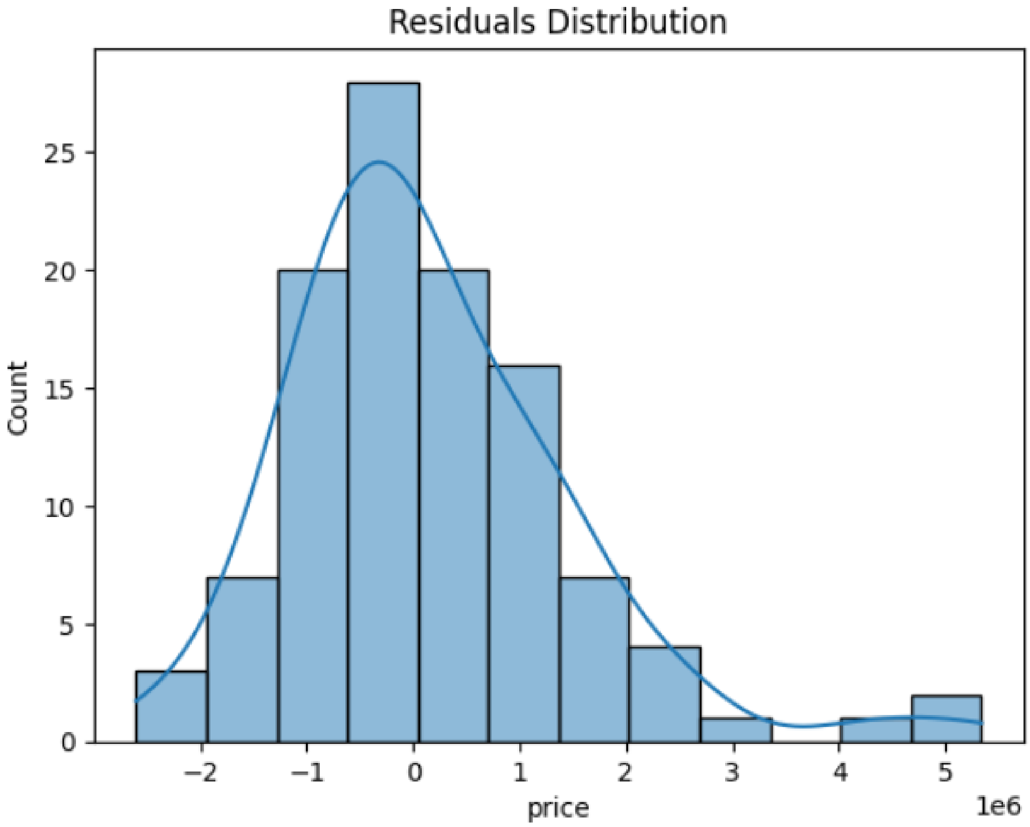
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Model Training and Evaluation
ridge = Ridge(alpha=1.0)  # Try different alpha (λ) values
ridge.fit(X_train, y_train)
y_pred = ridge.predict(X_test)

# Performance Metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Residuals Plot
residuals = y_test - y_pred
sns.histplot(residuals, kde=True)
plt.title("Residuals Distribution")
plt.show()
```

OUTPUT :






```
Mean Squared Error: 1754768938809.1035
R-squared: 0.6528351861223265
```


Residuals Distribution
Count
25
20
15
10
5
0
−2
−1
0
1
2
3
4
5
price
1e6

EXPERIMENT NO. 4

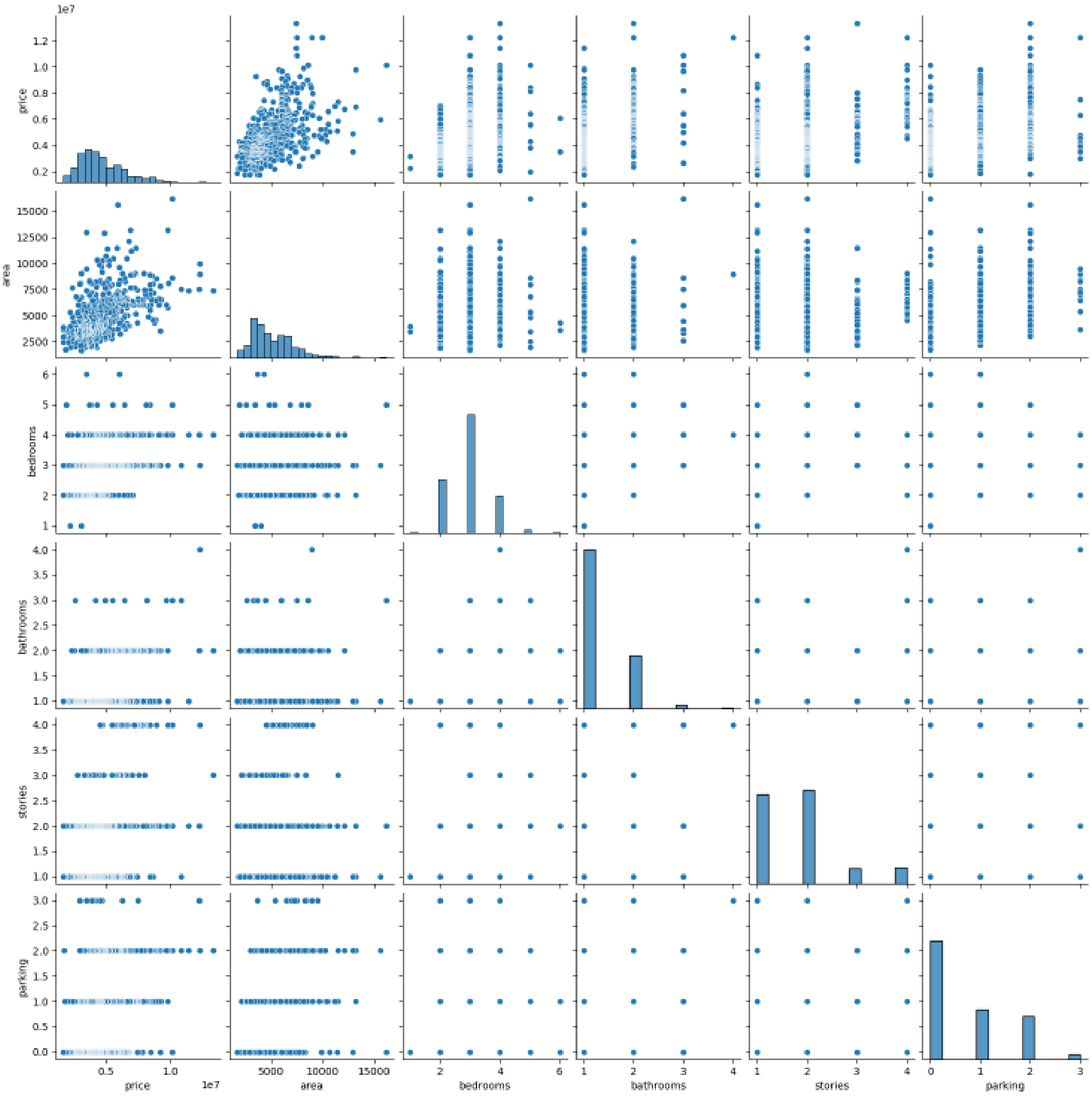
TITLE : Perform EDA and Visualise the relationships and also train a Lasso regression model and evaluate model's performance for given dataset.

CODE :

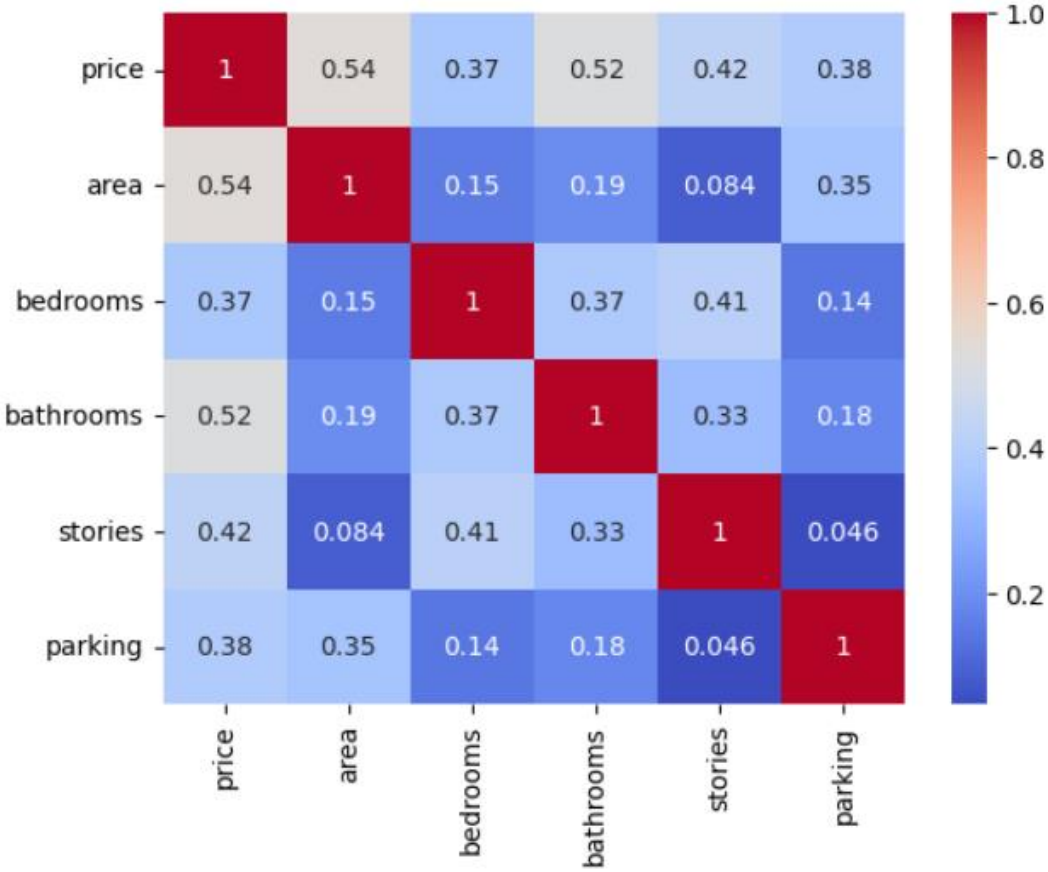
```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
df = pd.read_csv("Housing.csv")
print(f"Dataset Shape: {df.shape}")

# EDA
sns.pairplot(df)
plt.show()
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.show()

# Preprocessing
df = pd.get_dummies(df, drop_first=True)
X = df.drop("price", axis=1)
y = df["price"]

# Standardization
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# Model Training and Evaluation
lasso = Lasso(alpha=1.0)  # Experiment with different alpha (λ) values
lasso.fit(X_train, y_train)
y_pred = lasso.predict(X_test)

# Performance Metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")

# Residual Plot
residuals = y_test - y_pred
sns.histplot(residuals, kde=True)
plt.title("Residuals Distribution")
plt.show()
```

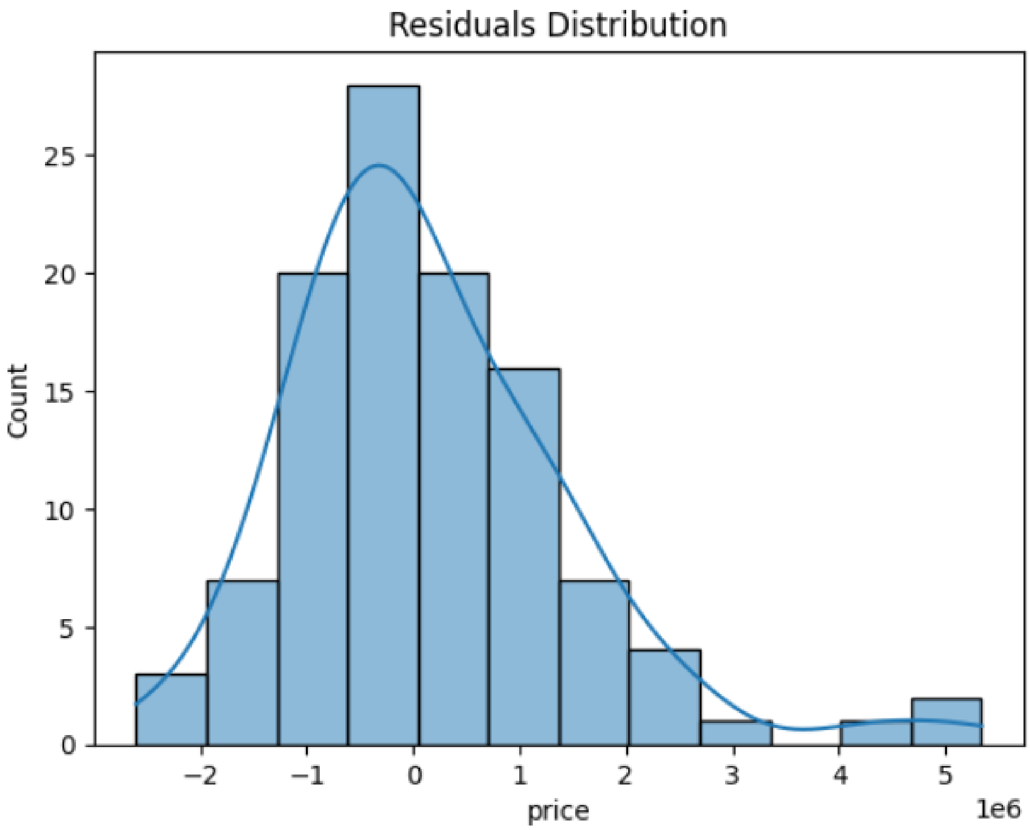
OUTPUT :



price
area
bedrooms
bathrooms
stories
parking
1e7




```
Mean Squared Error: 1754768938809.1035
R-squared: 0.6528351861223265
```



EXPERIMENT NO. 5

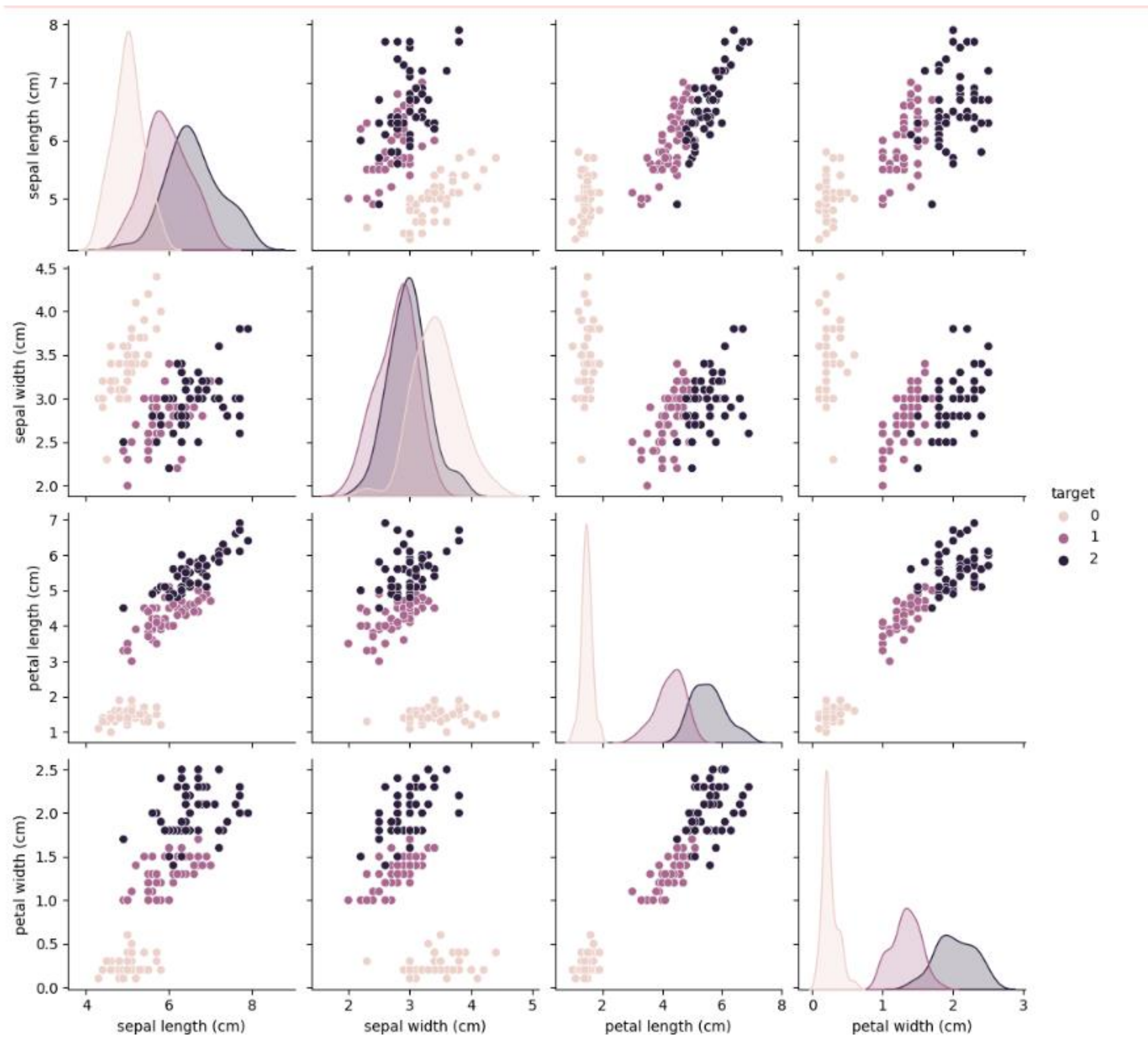
TITLE : Perform EDA and Visualise the relationships and also train a Logistic regression model and evaluate model's performance for Iris dataset.

CODE :

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


# Load Dataset

from sklearn.datasets import load_iris

data = load_iris(as_frame=True)

df = data.frame


# EDA: Pair plot

sns.pairplot(df, hue='target', vars=['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'])

plt.show()


# Split Data

X = df.drop('target', axis=1)

y = df['target']

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


# Train Logistic Regression

model = LogisticRegression()

model.fit(X_train, y_train)


# Predictions and Evaluation
```

```
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

OUTPUT :



```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

EXPERIMENT NO. 6

TITLE : Perform EDA and Visualise the relationships and also train a Logistic regression model and evaluate model's performance for User dataset.

CODE :

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


# Load Dataset

data = pd.read_csv('User_Data.csv')


# EDA: Check dataset info

print(data.info())


# Scatter Plot

sns.scatterplot(x='Age', y='EstimatedSalary', hue='Purchased', data=data)

plt.title('Age vs Estimated Salary Colored by Purchase')

plt.show()


# Data Preprocessing

data['Gender'] = data['Gender'].map({'Male': 0, 'Female': 1})  # Encode Gender

X = data[['Gender', 'Age', 'EstimatedSalary']]

y = data['Purchased']

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


# Train Logistic Regression

model = LogisticRegression()
```

model.fit(X_train, y_train)

Predictions and Evaluation

y_pred = model.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))

print("Classification Report:\n", classification_report(y_test, y_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

OUTPUT :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
None
```


```
Accuracy: 0.65
Classification Report:
              precision    recall  f1-score   support

           0       0.65      1.00      0.79        52
           1       0.00      0.00      0.00        28

    accuracy                           0.65        80
   macro avg       0.33      0.50      0.39        80
weighted avg       0.42      0.65      0.51        80

Confusion Matrix:
 [[52  0]
 [28  0]]
```

EXPERIMENT NO. 7

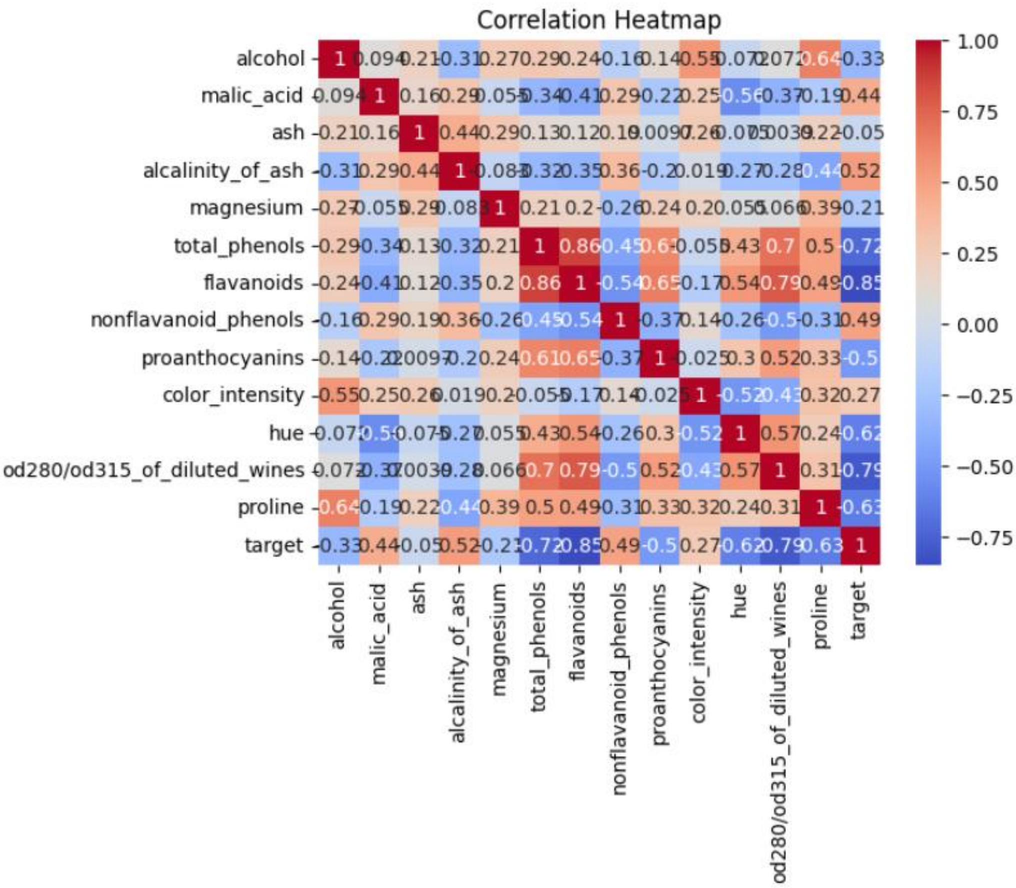
TITLE : Perform EDA and Visualise the relationships and also train a Naive Baye's Classifier model and evaluate model's performance for Wine dataset.

CODE :

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Load Dataset
from sklearn.datasets import load_wine
data = load_wine(as_frame=True)
df = data.frame

# EDA: Visualizing Correlation Heatmap
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()

# Splitting Features and Target
X = df.drop('target', axis=1)
y = df['target']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Naive Bayes Classifier
model = GaussianNB()
```

```
model.fit(X_train, y_train)


# Predictions and Evaluation
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

OUTPUT :



```
Accuracy: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         8

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36

Confusion Matrix:
 [[14  0  0]
 [ 0 14  0]
 [ 0  0  8]]
```

EXPERIMENT NO. 8

TITLE : Perform EDA and Visualise the relationships and also train a Decision Tree Classifier model and evaluate model's performance for Titanic dataset.

CODE :

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix


# Load Dataset

data = pd.read_csv('titanic.csv')


# EDA: Summary and Visualization

print(data.info())

sns.barplot(x='Sex', y='Survived', data=data)

plt.title('Survival Rate by Gender')

plt.show()


# Preprocessing: Handle missing values and encode categorical variables

data['Age'].fillna(data['Age'].median(), inplace=True)

data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

data = pd.get_dummies(data, columns=['Sex', 'Embarked'], drop_first=True)


# Feature Selection

features = ['Pclass', 'Age', 'SibSp', 'Parch', 'Fare', 'Sex_male', 'Embarked_Q', 'Embarked_S']

X = data[features]

y = data['Survived']


# Split Data
```

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

Train Decision Tree Classifier

model = DecisionTreeClassifier(max_depth=5, random_state=42)

model.fit(X_train, y_train)

Predictions and Evaluation

y_pred = model.predict(X_test)

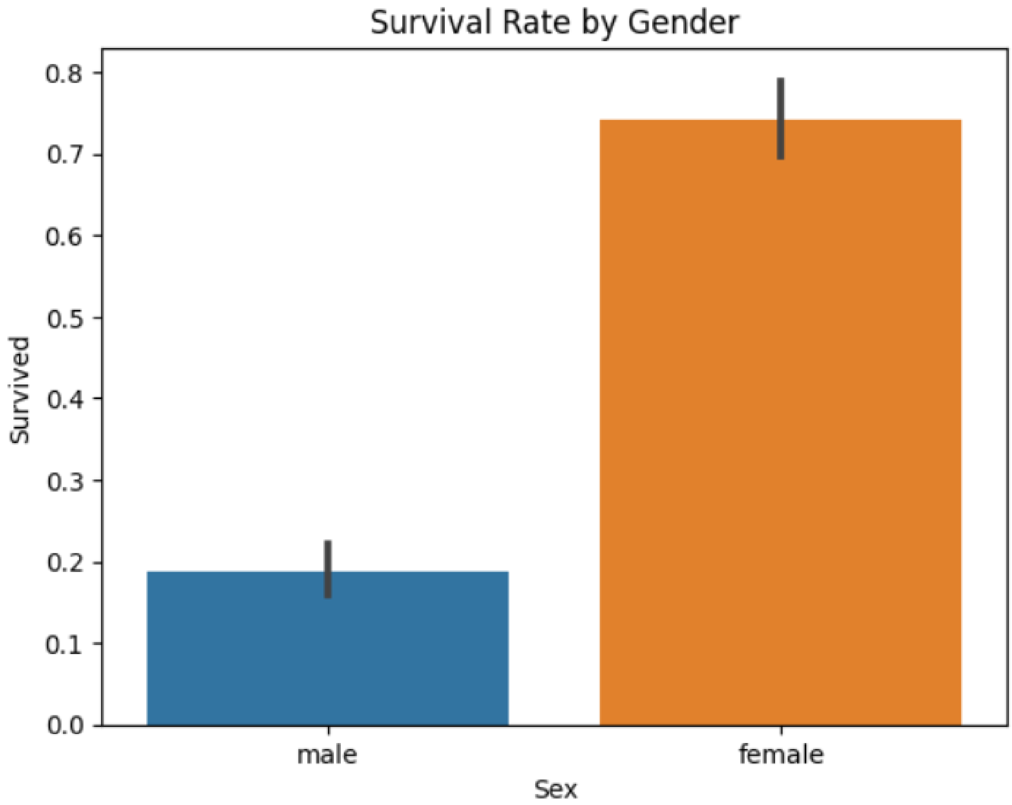
print("Accuracy:", accuracy_score(y_test, y_pred))

print("Classification Report:\n", classification_report(y_test, y_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

OUTPUT :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
None
```




```
Accuracy: 0.7988826815642458
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.90      0.84       105
           1       0.83      0.65      0.73        74

    accuracy                           0.80       179
   macro avg       0.81      0.78      0.78       179
weighted avg       0.80      0.80      0.79       179

Confusion Matrix:
 [[95 10]
 [26 48]]
```

EXPERIMENT NO. 9

TITLE : Perform EDA and Visualise the relationships and also train a KNN model and evaluate model's performance for Digit dataset.

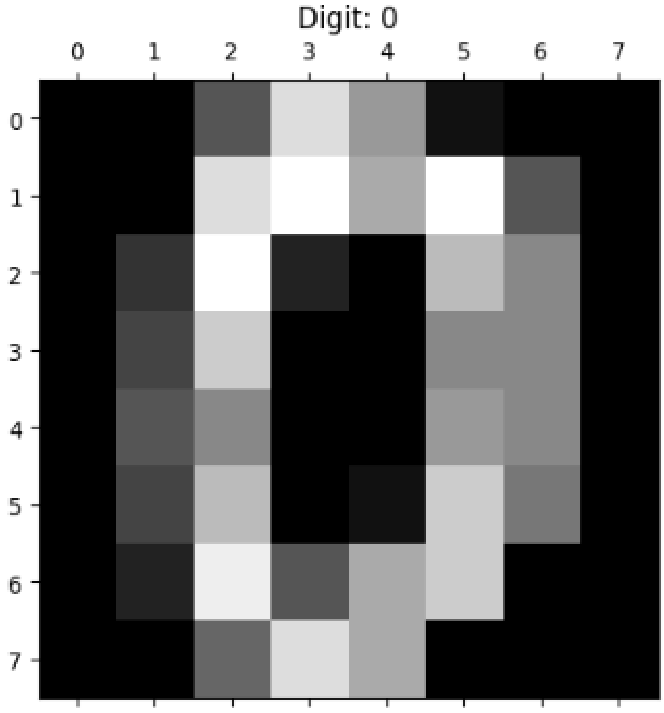
CODE :

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Load Digits Dataset
digits = load_digits()
X = digits.data
y = digits.target
print(f"Dataset Shape: {X.shape}")

# Visualize a Sample Digit
plt.gray()
plt.matshow(digits.images[0])
plt.title(f"Digit: {digits.target[0]}")
plt.show()

# Normalize the Data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split into Training and Testing Sets
```

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

# Train KNN Model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Make Predictions
y_pred = knn.predict(X_test)

# Evaluate Model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

OUTPUT :

```
Dataset Shape: (1797, 64)
<Figure size 640x480 with 0 Axes>
```




```
Accuracy: 0.975
Confusion Matrix:
 [[33  0  0  0  0  0  0  0  0  0]
 [ 0 28  0  0  0  0  0  0  0  0]
 [ 0  0 33  0  0  0  0  0  0  0]
 [ 0  0  1 33  0  0  0  0  0  0]
 [ 0  0  0  0 46  0  0  0  0  0]
 [ 0  0  0  0  0 45  1  0  0  1]
 [ 0  0  0  0  0  0 35  0  0  0]
 [ 0  0  0  0  0  1  0 32  0  1]
 [ 0  0  0  0  0  0  0  0 30  0]
 [ 0  0  0  1  1  1  0  0  1 36]]
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        33
           1       1.00      1.00      1.00        28
           2       0.97      1.00      0.99        33
           3       0.97      0.97      0.97        34
           4       0.98      1.00      0.99        46
           5       0.96      0.96      0.96        47
           6       0.97      1.00      0.99        35
           7       1.00      0.94      0.97        34
           8       0.97      1.00      0.98        30
           9       0.95      0.90      0.92        40

    accuracy                           0.97       360
   macro avg       0.98      0.98      0.98       360
weighted avg       0.98      0.97      0.97       360
```

EXPERIMENT NO. 10

TITLE : . Perform EDA and Visualise the relationships and also train a K-Means regression model and evaluate model's performance for Income dataset.

CODE :

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score


# Load Dataset

data = pd.read_csv("User_Data.csv")

print(data.info())


# EDA: Visualize EstimatedSalary and Age distributions

sns.scatterplot(x="Age", y="EstimatedSalary", hue="Purchased", data=data)

plt.title("Scatterplot: Age vs Estimated Salary Colored by Purchase")

plt.show()


# Preprocessing

data = pd.get_dummies(data, columns=["Gender"], drop_first=True)

scaler = StandardScaler()

data[["Age", "EstimatedSalary"]] = scaler.fit_transform(data[["Age", "EstimatedSalary"]])

# Apply K-Means Clustering

kmeans = KMeans(n_clusters=3, random_state=42)

data["Cluster"] = kmeans.fit_predict(data[["Age", "EstimatedSalary"]])

# Visualize Clusters

sns.scatterplot(x="Age", y="EstimatedSalary", hue="Cluster", data=data, palette="viridis")

plt.title("Clusters Visualized: Age vs Estimated Salary")
```

plt.show()

Evaluate Model

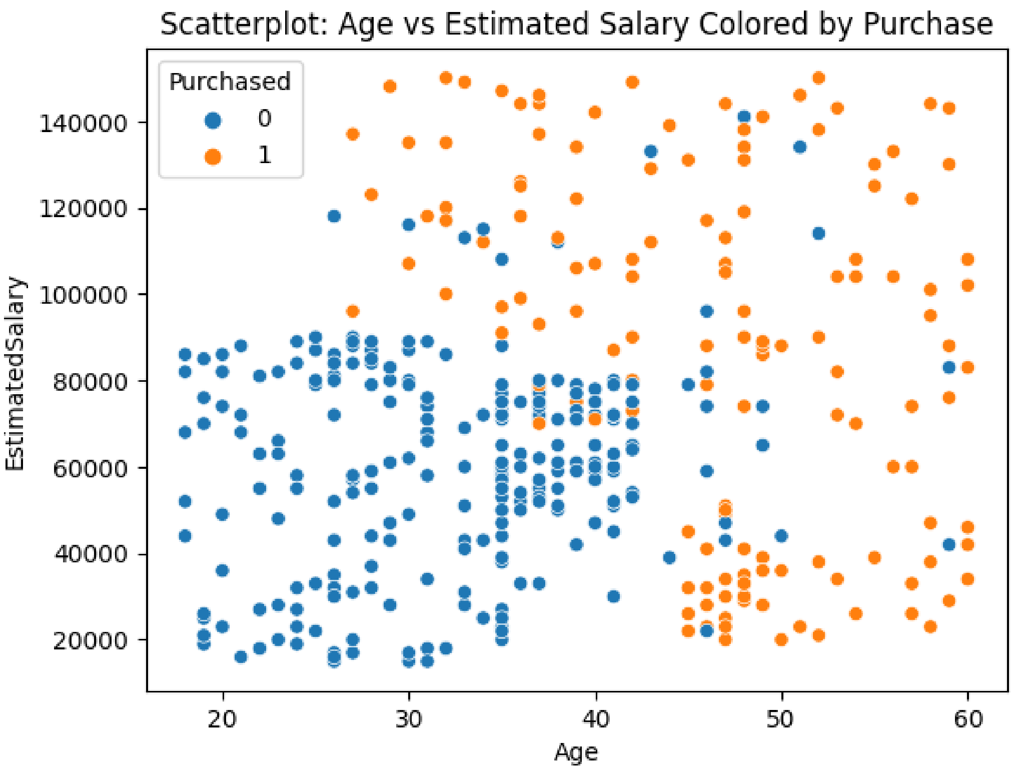
print("Cluster Centroids:\n", kmeans.cluster_centers_)

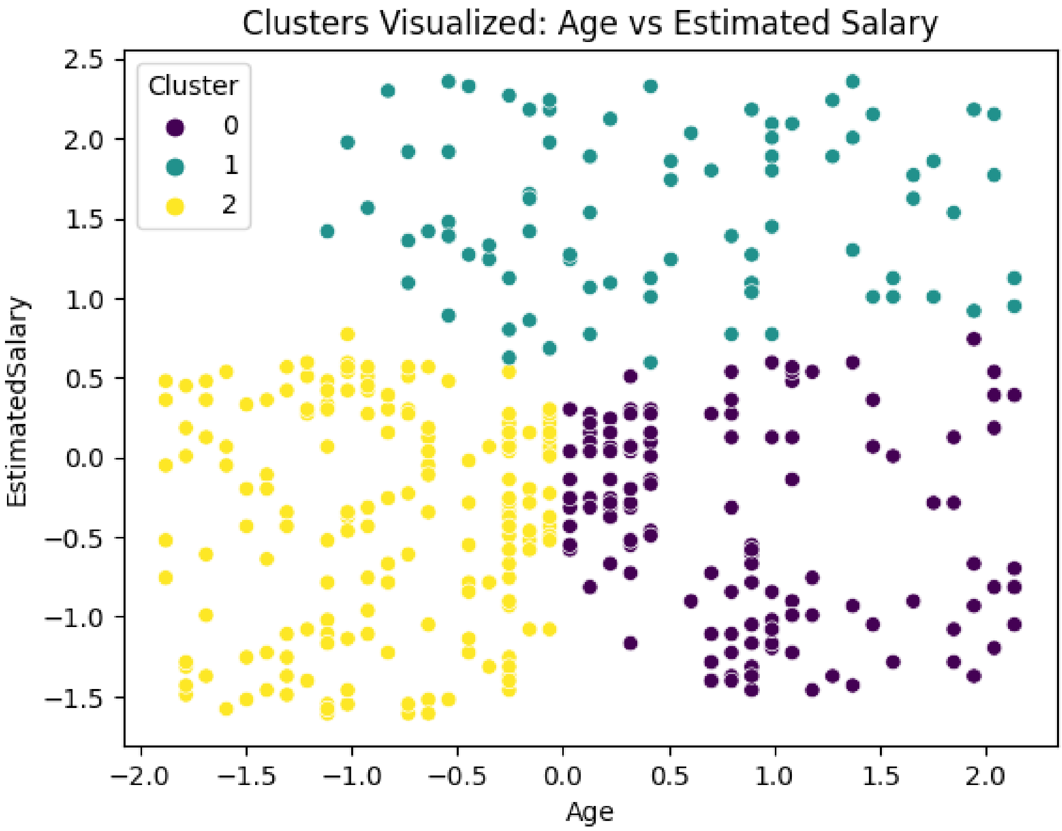
print("Intra-Cluster Variance (WCSS):", kmeans.inertia_)

print("Silhouette Score:", silhouette_score(data[["Age", "EstimatedSalary"]], data["Cluster"]))

OUTPUT :

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          400 non-null    int64
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64
 3   EstimatedSalary  400 non-null    int64
 4   Purchased        400 non-null    int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
None
```






```
Cluster Centroids:
 [[ 0.7991185  -0.40988428]
 [ 0.46159632  1.54420291]
 [-0.83200206 -0.38911019]]
Intra-Cluster Variance (WCSS): 323.8667384794876
Silhouette Score: 0.36205845294545275
```

EXPERIMENT NO. 11

TITLE : Perform EDA and Visualise the relationships and also train a K-Means regression model and evaluate model's performance for Iris dataset.

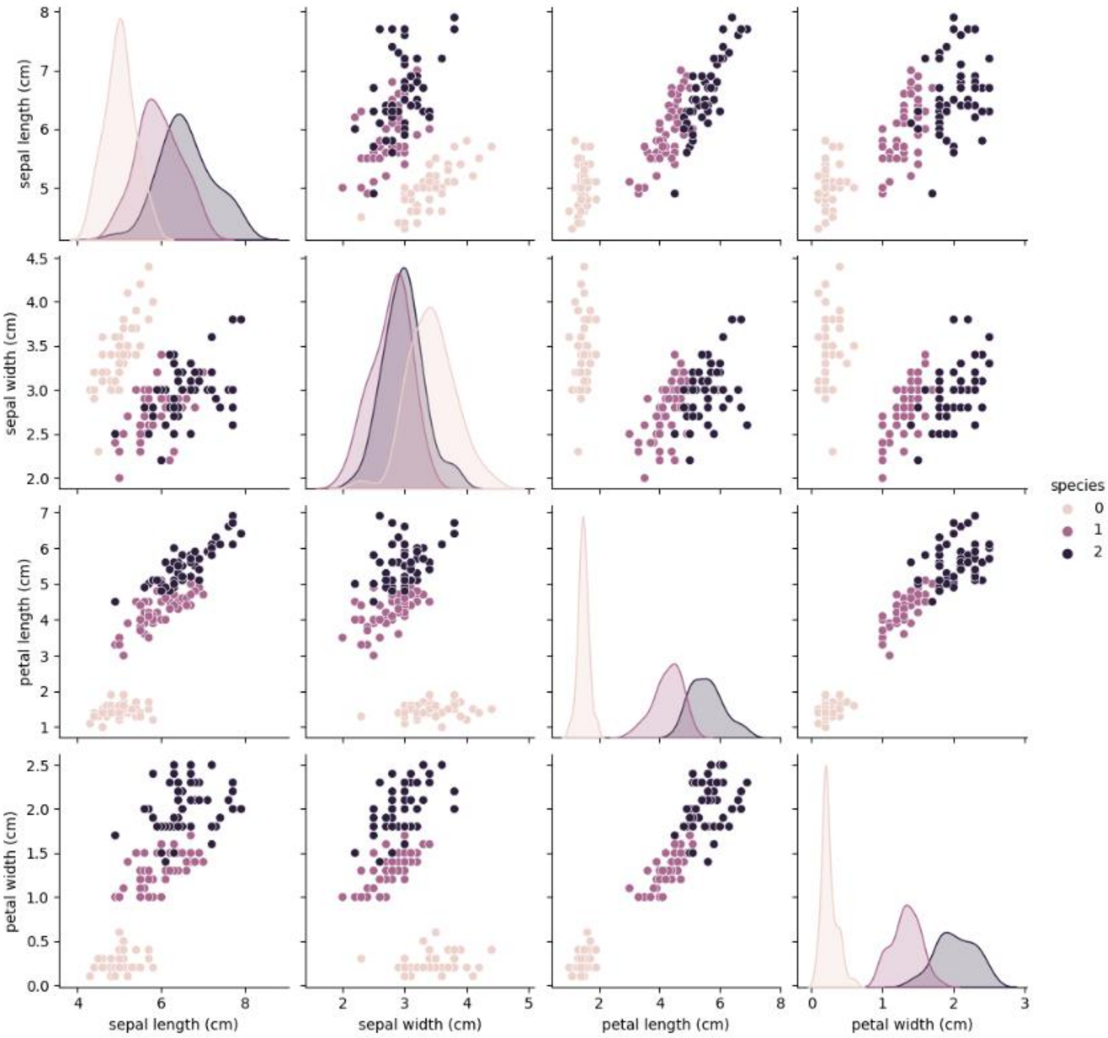
CODE :

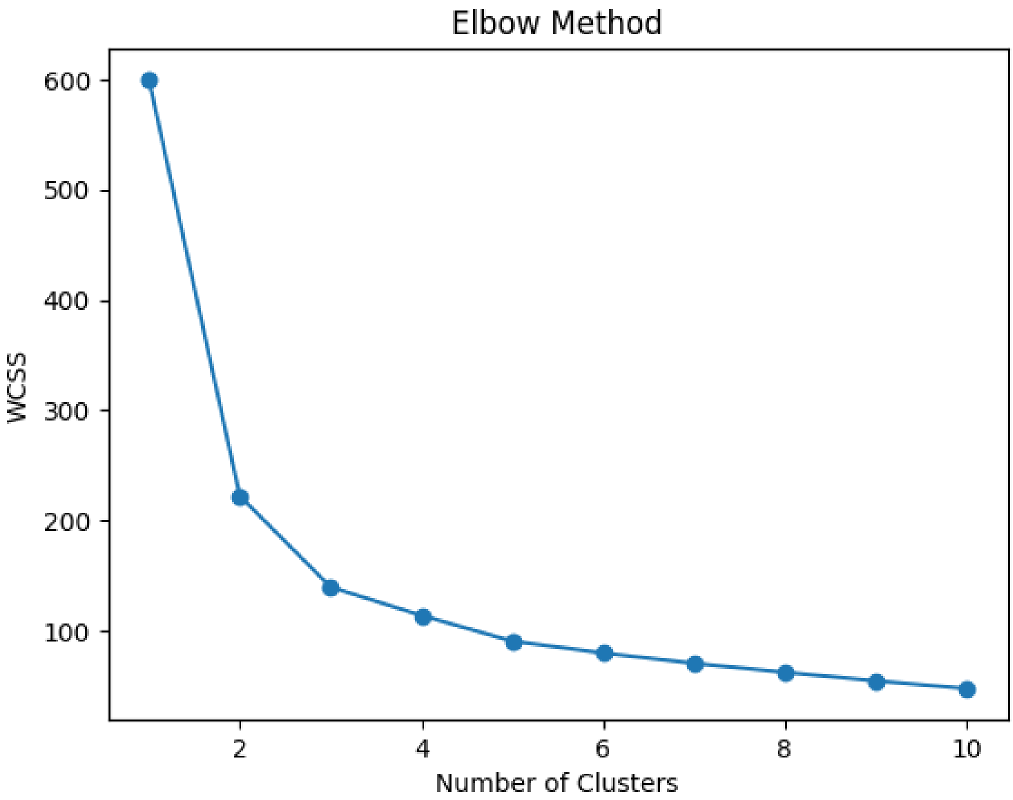
```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.datasets import load_iris

from sklearn.preprocessing import StandardScaler

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette_score

import numpy as np


# Load Dataset

iris = load_iris()

data = pd.DataFrame(iris.data, columns=iris.feature_names)

data['species'] = iris.target

print(data.head())


# EDA: Pair Plot

sns.pairplot(data, hue='species', vars=iris.feature_names)

plt.show()


# Data Preprocessing: Standardize Features

scaler = StandardScaler()

data_scaled = scaler.fit_transform(data[iris.feature_names])


# Determine Optimal k using Elbow Method

wcss = []

for k in range(1, 11):

    kmeans = KMeans(n_clusters=k, random_state=42)
```

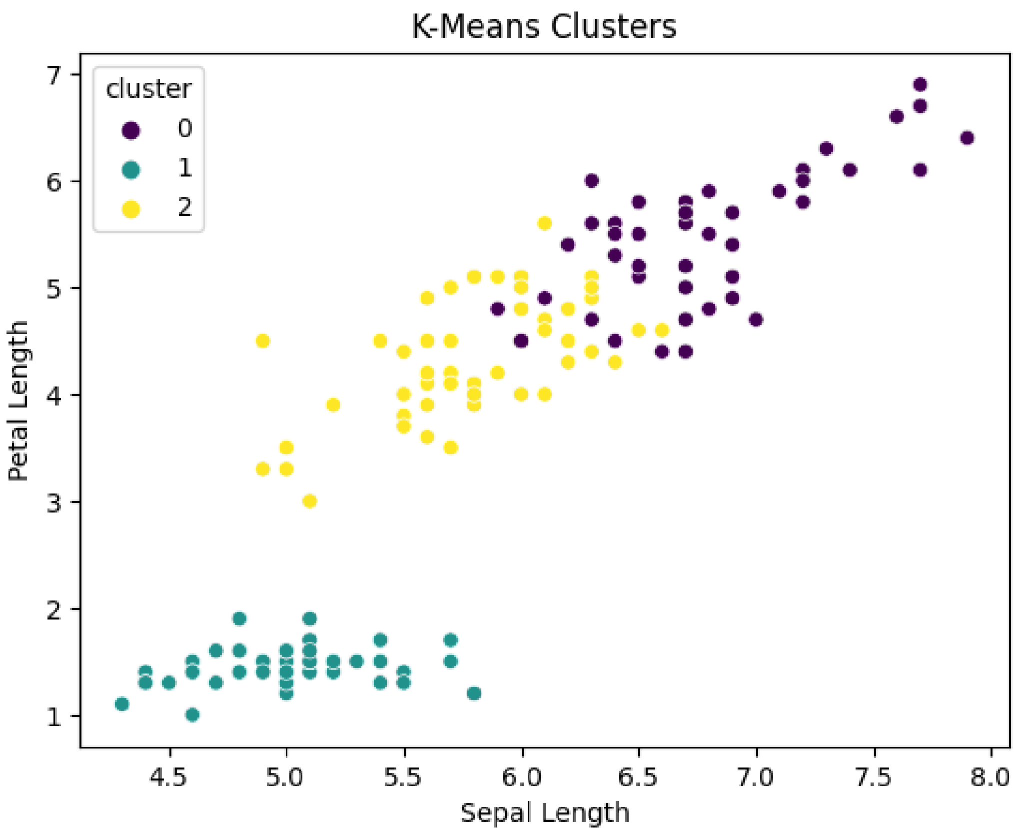
```
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()

# Train K-Means Model
kmeans = KMeans(n_clusters=3, random_state=42)
data['cluster'] = kmeans.fit_predict(data_scaled)

# Visualize Clusters
sns.scatterplot(x=data[iris.feature_names[0]], y=data[iris.feature_names[2]], hue=data['cluster'], palette='viridis')
plt.title('K-Means Clusters')
plt.xlabel('Sepal Length')
plt.ylabel('Petal Length')
plt.show()

# Evaluate Clustering
silhouette_avg = silhouette_score(data_scaled, data['cluster'])
print("Silhouette Score:", silhouette_avg)
```

OUTPUT :

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   species
0        0
1        0
2        0
3        0
4        0
```








```
Silhouette Score: 0.45994823920518635
```