<u>Linear Regression</u>

<u>Aim</u>: To use linear regression to predict salary of an employee using years of experience and house price using a combination of features

<u>Linear Regression</u>:
Linear regression is a supervised machine learning algorithm that uses labeled datasets to model the relationship between the dependent variable and one or more independent features. It finds an optimal linear function to map data points, allowing for predictions on new datasets. By fitting a linear equation to observed data, it predicts continuous output values based on input variables.

The primary goal of linear regression is to find the best-fit line, minimizing the error between predicted and actual values. The line with the least error is considered the best fit. When there is only one independent feature, it is called Simple Linear Regression or Univariate Regression, while multiple features lead to Multiple Linear Regression or Multivariate Regression.

<u>Simple Linear Regression</u>

<u>Algorithm</u>:

1. Import necessary libraries
2. Load and clean the dataset
3. Scale features using StandardScaler
4. Split the data into features (YearsExperience) and target (Salary)
5. Split the data into training and testing sets
6. Train a linear regression model
7. Evaluate and predict using the testing set
8. Visualize predictions vs. actual values using a scatter plot

<u>Code</u>:

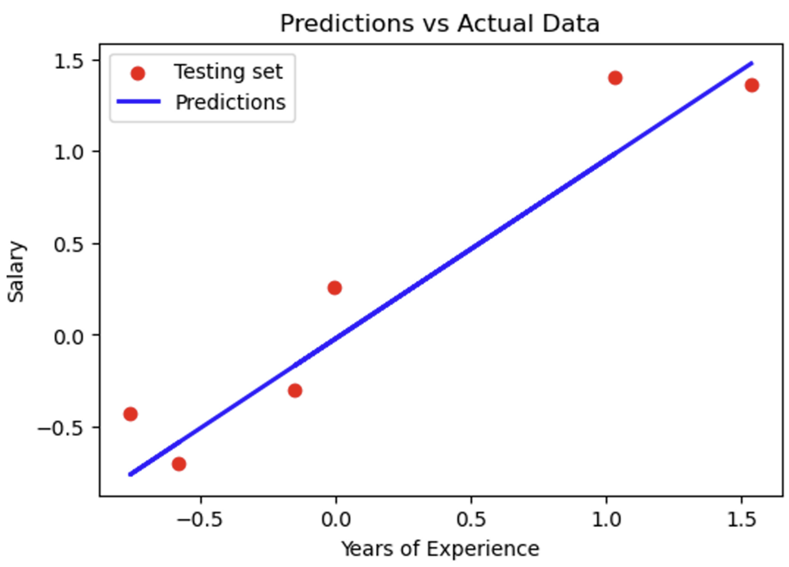
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import seaborn as sns

# Load the dataset
df = pd.read_csv("Salary_dataset.csv")
```

```
# Data cleaning
df.drop(columns=["Unnamed: 0"], inplace=True)

# Display data info
df.info()
df.shape

# Scale the features
scaler = StandardScaler()
df_scaled = scaler.fit_transform(df.to_numpy())
df_scaled = pd.DataFrame(df_scaled,
columns=['YearsExperience', 'Salary'])

# Scatter plot of Salary vs. Years of Experience
plt.figure(figsize=(6, 4))
sns.scatterplot(x='YearsExperience', y='Salary',
data=df_scaled)
plt.title('Salary vs. Years of Experience')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()

# Split data into features and target
x = df_scaled.iloc[:, :-1]
y = df_scaled.iloc[:, -1]

# 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)

# Train a linear regression model
reg = LinearRegression()
reg.fit(x_train, y_train)

# Evaluate the model
train_acc = reg.score(x_train, y_train)
test_acc = reg.score(x_test, y_test)
print("Training accuracy:", train_acc)
print("Testing accuracy:", test_acc)

# Predict the salary using the test set
y_pred = reg.predict(x_test)

# Visualize predictions vs actual data
plt.figure(figsize=(6, 4))
# Plot testing set
```

```
plt.scatter(x_test, y_test, color='red', label='Testing
set')
# Plot predictions
plt.plot(x_test, y_pred, color='blue', linewidth=2,
label='Predictions')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.title('Predictions vs Actual Data')
plt.legend()
plt.show()
```

Output:

```
Training accuracy: 0.9645401573418146
Testing accuracy: 0.9024461774180499
```



Multiple Linear Regression

Algorithm:

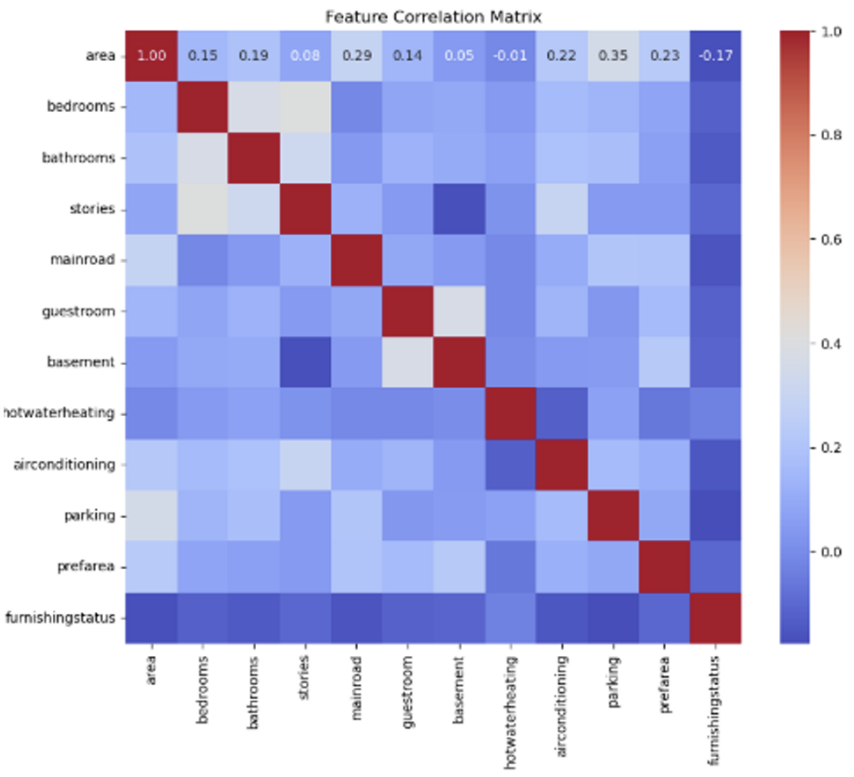
1. Import necessary libraries.
2. Load the dataset and separate features (x) and target (y).
3. Apply Label Encoding to categorical columns.
4. Scale the features using StandardScaler.
5. Split the data into training and testing sets.
6. Train a Linear Regression model and evaluate its accuracy.
7. Print the training and testing accuracy.

Code:

```
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.preprocessing import LabelEncoder,
StandardScaler

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

# Display dataset information
df.head()
df.shape
df.info()

# Separate features (x) and target variable (y)
x = df.drop(columns=["price"])
y = df["price"]

# Identify categorical columns and apply Label Encoding
categorical_cols =
x.select_dtypes(include=["object"]).columns
print("Categorical Features:", categorical_cols.tolist())

encoder = LabelEncoder()
for col in categorical_cols:
    x[col] = encoder.fit_transform(x[col])

# Scale the features
scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)

# Compute and plot the correlation matrix
correlation_matrix = pd.DataFrame(x_scaled,
columns=x.columns).corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True,
cmap="coolwarm", fmt=".2f")
plt.title("Feature Correlation Matrix")
plt.show()

# Split the data 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 the Linear Regression model
```

```
lr = LinearRegression()
lr.fit(x_train, y_train)

# Evaluate the model
train_acc = lr.score(x_train, y_train)
test_acc = lr.score(x_test, y_test)

# Print training and testing accuracy
print("Training accuracy:", train_acc)
print("Testing accuracy:", test_acc)
```

Output:



```
Training accuracy: 0.6854429472843789
Testing accuracy: 0.6494754192267795
```