linear-regg-fl

March 30, 2025

Develop a program to demonstrate the working of linear regression and polynomial regression.Use Boston housing dataset for linear regression and auto MPG dataset for Polynomial regression

Linear Regression using Boston housing dataset

[45]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')
```

[46]:
```
# Load dataset
data = pd.read_csv('./Boston housing dataset.csv')
```

[47]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   CRIM     486 non-null    float64
 1   ZN       486 non-null    float64
 2   INDUS    486 non-null    float64
 3   CHAS     486 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      486 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    int64  
 9   TAX      506 non-null    int64  
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
```

```
 12  LSTAT    486 non-null    float64
 13  MEDV     506 non-null    float64
dtypes: float64(12), int64(2)
memory usage: 55.5 KB
```

CRIM: Per capita crime rate by town. ZN: Proportion of residential land zoned for lots over 25,000 square feet. INDUS: Proportion of non-retail business acres per town. CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise). NOX: Nitric oxide concentration (parts per 10 million). RM: Average number of rooms per dwelling. AGE: Proportion of owner-occupied units built before 1940. DIS: Weighted distances to five Boston employment centers. RAD: Index of accessibility to radial highways. TAX: Full-value property-tax rate

[48]:
```
# Data Cleaning(For all attributes)
data.fillna(0, inplace=True)  # Replace null values with 0
data.drop_duplicates(inplace=True)  # Drop duplicate rows
```

[49]:
```
# Selecting features and target
X = data.drop(columns=['MEDV'])  # All columns except 'MEDV'
y = data['MEDV']  # Target variable (median house price)
```

[50]:
```
# Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

[51]:
```
# Split the data into training (80%) and testing (20%) sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,␣
 ↪random_state=42)
```

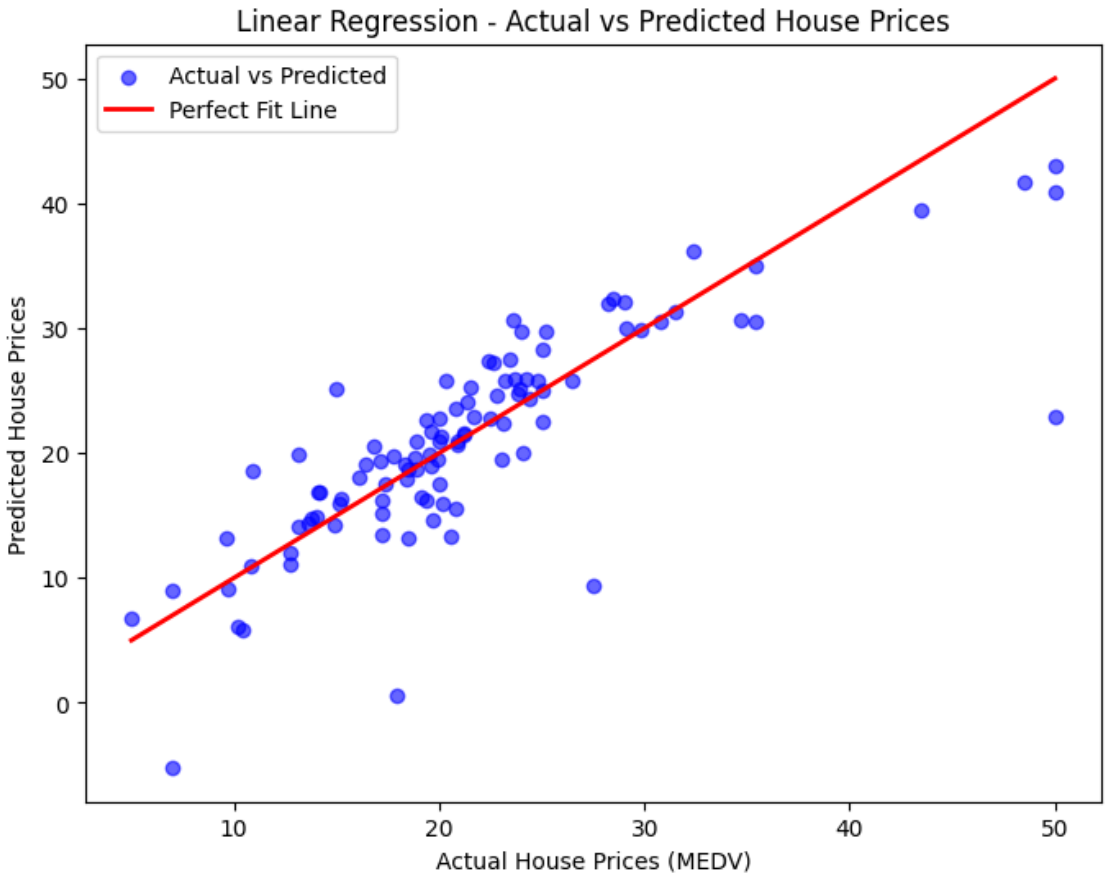
[52]:
```
# Initialize the linear regression model
model = LinearRegression()

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

# Predict on the test set
y_pred = model.predict(X_test)
```

[53]:
```
# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Print metrics
print(f'Mean Squared Error: {mse}')
print(f'Root Mean Squared Error: {rmse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 25.929696549627074
```

```
Root Mean Squared Error: 5.092121026608369
R-squared: 0.646415397758229
```

[54]:
```
# Plot Actual vs Predicted Values
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.6, label='Actual vs␣
 ↪Predicted')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',␣
 ↪linewidth=2, label='Perfect Fit Line')

plt.xlabel("Actual House Prices (MEDV)")
plt.ylabel("Predicted House Prices")
plt.title("Linear Regression - Actual vs Predicted House Prices")
plt.legend()
plt.show()
```



[ ]: