dmccrary-mod8-lab1

May 21, 2025

0.1 Assignment Module 8 Lab 1: Predicting House Prices in California using Linear Regression

0.1.1 Background

Accurately predicting house prices is essential in the real estate market. In this assignment, you will use linear regression to predict house prices based on various features such as the number of rooms, population density, and median income in different neighborhoods in California.

0.1.2 Dataset

The dataset you will be using is the California Housing dataset, which contains information about houses in different locations in California. The dataset includes 20,640 observations and 8 variables, including:

- `MedInc`: Median income in the block
- `HouseAge`: Median house age in the block
- `AveRooms`: Average number of rooms per dwelling
- `AveBedrms`: Average number of bedrooms per dwelling
- `Population`: Block population
- `AveOccup`: Average house occupancy
- `Latitude`: Latitude coordinate of the block
- `Longitude`: Longitude coordinate of the block

You can load the dataset using the following code:

```python
from sklearn.datasets import fetch_california_housing

# Load the California Housing dataset
california = fetch_california_housing(as_frame=True)

# Access the data and target variables
X = california.data
y = california.target
```

0.1.3 Task

Your task is to predict house prices in California using linear regression. You will need to perform the following steps:

1. Load the California Housing dataset into a Pandas DataFrame.
2. Split the dataset into training and testing sets, with a 70/30 split.

3. Fit a `LinearRegression` model to the training data.
4. Predict house prices for the testing set.
5. Evaluate the performance of the model using the following error checking metrics:
    - R-squared (R2)
    - Mean Squared Error (MSE)
    - Mean Absolute Error (MAE)
    - Akaike Information Criterion (AIC)
6. Analyze the performance of the model and provide recommendations for improvement.

You can use the `OLS` (ordinary least squares) method from the `statsmodels.regression.linear_model` module to calculate the Akaike Information Criterion (AIC).

0.1.4 Submission Instructions

Please submit a Jupyter notebook containing your code, output, and analysis. Make sure to include the following sections in your notebook:

1. Introduction
2. Dataset Description
3. Data Preprocessing
4. Model Training
5. Model Evaluation
6. Performance Analysis and Recommendations
7. Conclusion

Your notebook should be well-documented and easy to follow, with clear explanations of the steps you took and the results you obtained. Make sure to comment your code and include markdown cells explaining your thought process and conclusions.

1 CS82B – Principles of Data Science

1.1 Module 8 – Lab 1

Student Name: Derek McCrary
Due Date: May 25, 2025

[5]:
```python
from sklearn.datasets import fetch_california_housing
import pandas as pd
import numpy as np


california = fetch_california_housing(as_frame=True)
X = california.data
y = california.target
```

[6]:
```python
from sklearn.model_selection import train_test_split

# Split dataset: 70% training, 30% testing
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Confirm shapes
X_train.shape, X_test.shape
```

[6]:
```
((14448, 8), (6192, 8))
```

[7]:
```
from sklearn.preprocessing import StandardScaler
import pandas as pd

# Standardize and preserve column names
scaler = StandardScaler()
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), columns=X.columns,␣
 ↪index=X_train.index)
X_test_scaled = pd.DataFrame(scaler.transform(X_test), columns=X.columns,␣
 ↪index=X_test.index)
```

[8]:
```
# Sanity check: drop rows with invalid data from scaled sets
X_train_scaled = X_train_scaled.replace([np.inf, -np.inf], np.nan).dropna()
X_test_scaled = X_test_scaled.replace([np.inf, -np.inf], np.nan).dropna()

# Also filter y_train and y_test to match the new index
y_train = y_train.loc[X_train_scaled.index]
y_test = y_test.loc[X_test_scaled.index]
```

[10]:
```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

# Evaluate model
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"R² Score: {r2:.3f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
```

```
R² Score: 0.596
Mean Squared Error (MSE): 0.53
Mean Absolute Error (MAE): 0.53
```

[11]:
```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

# Make predictions
#   Predict using scaled test set
y_pred = model.predict(X_test_scaled)

# Calculate metrics
```

```python
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"R² Score: {r2:.3f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
```

```
R² Score: 0.596
Mean Squared Error (MSE): 0.53
Mean Absolute Error (MAE): 0.53
```

```
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: divide by zero
encountered in matmul
  return X @ coef_ + self.intercept_
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: overflow encountered
in matmul
  return X @ coef_ + self.intercept_
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: invalid value
encountered in matmul
  return X @ coef_ + self.intercept_
```

[13]:
```python
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error

# Make predictions
#  Predict using scaled test set
y_pred = model.predict(X_test_scaled)

# Calculate metrics
r2 = r2_score(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print(f"R² Score: {r2:.3f}")
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
```

```
R² Score: 0.596
Mean Squared Error (MSE): 0.53
Mean Absolute Error (MAE): 0.53
```

```
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: divide by zero
encountered in matmul
  return X @ coef_ + self.intercept_
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
```

```
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: overflow encountered
in matmul
  return X @ coef_ + self.intercept_
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: invalid value
encountered in matmul
  return X @ coef_ + self.intercept_
```

```
[14]: import numpy as np

      # Check for NaN or infinite values
      print("NaNs in X_train_scaled:", np.isnan(X_train_scaled).sum().sum())
      print("Infs in X_train_scaled:", np.isinf(X_train_scaled).sum().sum())
      print("NaNs in X_test_scaled:", np.isnan(X_test_scaled).sum().sum())
      print("Infs in X_test_scaled:", np.isinf(X_test_scaled).sum().sum())
```

```
NaNs in X_train_scaled: 0
Infs in X_train_scaled: 0
NaNs in X_test_scaled: 0
Infs in X_test_scaled: 0
```

```
[15]: #  Target (y_train, y_test) range and null check
      print("y_train min:", y_train.min())
      print("y_train max:", y_train.max())
      print("y_test min:", y_test.min())
      print("y_test max:", y_test.max())
      print("Any NaNs in y_train:", y_train.isna().sum())
      print("Any NaNs in y_test:", y_test.isna().sum())
```

```
y_train min: 0.14999
y_train max: 5.00001
y_test min: 0.14999
y_test max: 5.00001
Any NaNs in y_train: 0
Any NaNs in y_test: 0
```

```
[16]: # Model training and prediction
      model = LinearRegression()
      model.fit(X_train_scaled, y_train)  # Train on scaled data

      y_pred = model.predict(X_test_scaled)  # Predict on scaled data
```

```
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: divide by zero
encountered in matmul
  return X @ coef_ + self.intercept_
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: overflow encountered
in matmul
```

```
  return X @ coef_ + self.intercept_
/Volumes/DevDereks/opt/anaconda3/envs/jupyter_env/lib/python3.11/site-
packages/sklearn/linear_model/_base.py:279: RuntimeWarning: invalid value
encountered in matmul
  return X @ coef_ + self.intercept_
```

[ ]:
```
# Final code block: AIC calculation
import statsmodels.api as sm

X_train_sm = sm.add_constant(X_train_scaled)
ols_model = sm.OLS(y_train, X_train_sm).fit()

print(f"AIC: {ols_model.aic:.2f}")
```

```
AIC: 31664.71
```

1.2 Conclusion

In this lab, I built a linear regression model to predict California housing prices using features such as income, location, and occupancy. After preprocessing and splitting the dataset, I trained the model and evaluated its performance.

1.2.1 Model Evaluation Metrics:

- **R² Score**: 0.596
- **Mean Squared Error (MSE)**: 0.53
- **Mean Absolute Error (MAE)**: 0.53
- **Akaike Information Criterion (AIC)**: 31664.71

1.2.2 Observations:

- The model demonstrated moderate predictive power, explaining nearly 60% of the variance in housing prices.
- All features were successfully scaled and validated (no NaNs or infinities).
- Standardization ensured stable training and prediction.
- AIC was used to assess model quality with respect to complexity.

1.2.3 Recommendations:

- Explore advanced models like Ridge or Lasso Regression.
- Try feature engineering (e.g., interaction terms, log-transforming skewed variables).
- Use cross-validation to further assess model generalizability.

1.3 Conclusion

In this lab, I built a linear regression model to predict California housing prices using features such as income, location, and occupancy. After preprocessing and splitting the dataset, I trained the model and evaluated its performance.

1.3.1 Model Evaluation Metrics:

- **R^2 Score**: 0.596
- **Mean Squared Error (MSE)**: 0.53
- **Mean Absolute Error (MAE)**: 0.53
- **Akaike Information Criterion (AIC)**: 31664.71

1.3.2 Observations:

- The model demonstrated moderate predictive power, explaining nearly 60% of the variance in housing prices.
- All features were successfully scaled and validated (no NaNs or infinities).
- Standardization ensured stable training and prediction.
- AIC was used to assess model quality with respect to complexity.

1.3.3 Recommendations:

- Explore advanced models like Ridge or Lasso Regression.
- Try feature engineering (e.g., interaction terms, log-transforming skewed variables).
- Use cross-validation to further assess model generalizability.