Machine Learning - Linear Regression

April 3, 2025

[10]:
```
import pandas as pd
import numpy as np

data = pd.read_csv("/home/kali/Desktop/myenv/Datasets/
 ↪ds_bostonHousing_removingOutliers.csv")
print(data.head())
```

```
      crim    zn  indus  chas    nox  …  tax  ptratio       b  lstat  medv
0  0.00632  18.0   2.31     0  0.538  …  296     15.3  396.90   4.98  24.0
1  0.02731   0.0   7.07     0  0.469  …  242     17.8  396.90   9.14  21.6
2  0.02729   0.0   7.07     0  0.469  …  242     17.8  392.83   4.03  34.7
3  0.03237   0.0   2.18     0  0.458  …  222     18.7  394.63   2.94  33.4
4  0.06905   0.0   2.18     0  0.458  …  222     18.7  396.90   5.33  36.2

[5 rows x 14 columns]
```

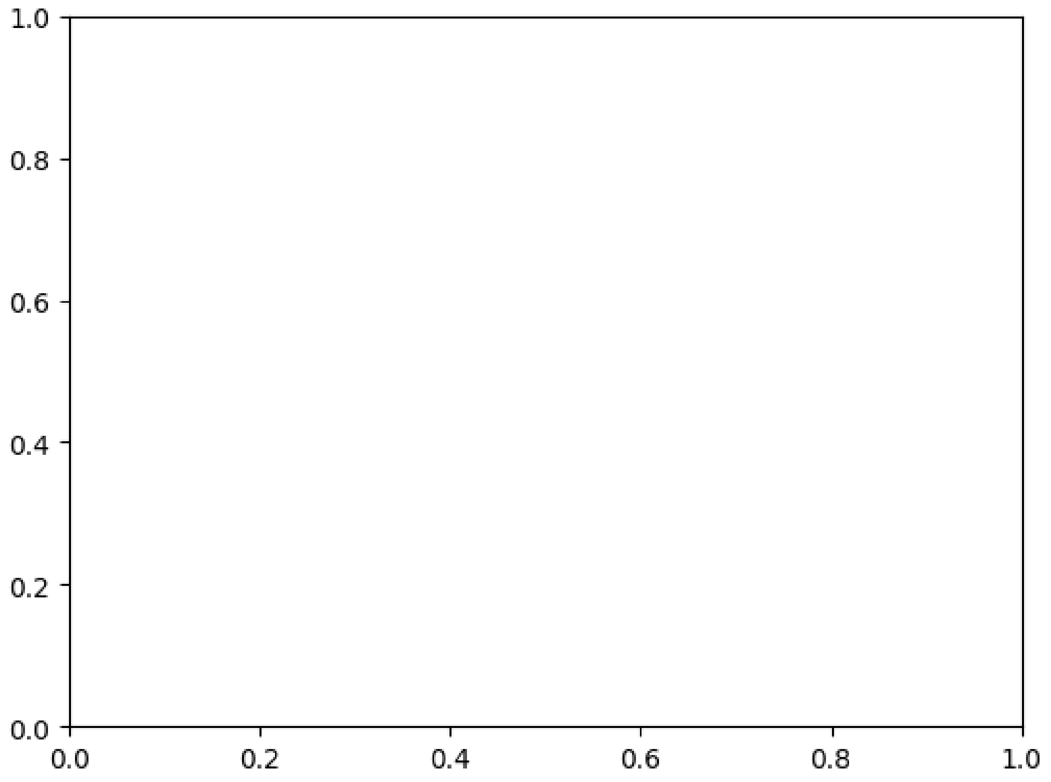
[13]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X = data[['crim', 'zn']]
Y = data['indus']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,␣
 ↪random_state=42)

model = LinearRegression()
model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)
```

[14]:
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

print("MAE:", mean_absolute_error(Y_test, Y_pred))
print("MSE:", mean_squared_error(Y_test, Y_pred))
print("R-Squared:", r2_score(Y_test, Y_pred))
```

```
MAE: 4.294177525902672
MSE: 27.502333954992025
R-Squared: 0.38542221769367513
```

```
[16]: import matplotlib.pyplot as plt

      plt.scatter(X_train, Y_train, color='blue', label='Training Data')
      plt.scatter(X_test, Y_test, color='green', label='Test Data')

      plt.plot(Y, model.predict(Y), color='red', linewidth=2, label='Regression Line')

      plt.show()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[16], line 3
      1 import matplotlib.pyplot as plt
----> 3 plt.scatter(X_train, Y_train, color=      , label=               )
      4 plt.scatter(X_test, Y_test, color='green', label='Test Data')
      6 plt.plot(Y, model.predict(Y), color='red', linewidth=2,␣
 ↪label='Regression Line')

File ~/Desktop/myenv/lib/python3.13/site-packages/matplotlib/_api/deprecation.p:
 ↪453, in make_keyword_only.<locals>.wrapper(*args, **kwargs)
    447 if len(args) > name_idx:
    448     warn_deprecated(
    449         since, message="Passing the %(name)s %(obj_type)s "
    450         "positionally is deprecated since Matplotlib %(since)s; the "
    451         "parameter will become keyword-only in %(removal)s.",
    452         name=name, obj_type=f"parameter of {func.__name__}()")
--> 453 return func(*args, **kwargs)

File ~/Desktop/myenv/lib/python3.13/site-packages/matplotlib/pyplot.py:3937, in
 ↪scatter(x, y, s, c, marker, cmap, norm, vmin, vmax, alpha, linewidths,␣
 ↪edgecolors, colorizer, plotnonfinite, data, **kwargs)
   3917 @_copy_docstring_and_deprecators(Axes.scatter)
   3918 def scatter(
   3919     x: float | ArrayLike,
   (…)   3935     **kwargs,
   3936 ) -> PathCollection:
-> 3937     __ret = gca().scatter(
   3938         x,
   3939         y,
   3940         s=s,
   3941         c=c,
   3942         marker=marker,
   3943         cmap=cmap,
   3944         norm=norm,
   3945         vmin=vmin,
   3946         vmax=vmax,
   3947         alpha=alpha,
   3948         linewidths=linewidths,
   3949         edgecolors=edgecolors,
   3950         colorizer=colorizer,
   3951         plotnonfinite=plotnonfinite,
   3952         **({       : data} if data is not None else {}),
   3953         **kwargs,
   3954     )
   3955     sci(__ret)
   3956     return __ret

File ~/Desktop/myenv/lib/python3.13/site-packages/matplotlib/_api/deprecation.py:
 ↪453, in make_keyword_only.<locals>.wrapper(*args, **kwargs)
    447 if len(args) > name_idx:
    448     warn_deprecated(
    449         since, message="Passing the %(name)s %(obj_type)s "
    450         "positionally is deprecated since Matplotlib %(since)s; the "
    451         "parameter will become keyword-only in %(removal)s.",
    452         name=name, obj_type=f"parameter of {func.__name__}()")
--> 453 return func(*args, **kwargs)

File ~/Desktop/myenv/lib/python3.13/site-packages/matplotlib/__init__.py:1521,␣
 ↪in _preprocess_data.<locals>.inner(ax, data, *args, **kwargs)
   1518 @functools.wraps(func)
   1519 def inner(ax, *args, data=None, **kwargs):
   1520     if data is None:
-> 1521         return func(
   1522             ax,
   1523             *map(cbook.sanitize_sequence, args),
   1524             **{k: cbook.sanitize_sequence(v) for k, v in kwargs.items()})
   1526     bound = new_sig.bind(ax, *args, **kwargs)
   1527     auto_label = (bound.arguments.get(label_namer)
   1528                   or bound.kwargs.get(label_namer))

File ~/Desktop/myenv/lib/python3.13/site-packages/matplotlib/axes/_axes.py:4930 ␣
 ↪in Axes.scatter(self, x, y, s, c, marker, cmap, norm, vmin, vmax, alpha,␣
 ↪linewidths, edgecolors, colorizer, plotnonfinite, **kwargs)
   4928 y = np.ma.ravel(y)
   4929 if x.size != y.size:
-> 4930     raise ValueError("x and y must be the same size")
   4932 if s is None:
   4933     s = (20 if mpl.rcParams['_internal.classic_mode'] else
   4934          mpl.rcParams['lines.markersize'] ** 2.0)

ValueError: x and y must be the same size
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

```
[ ]:
```