```
#Name:Yash Bhupesh Aware
#RolL no. :02
#Sub :AIML
#section:3C
#Date:23/01/2025

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")

import os

os.getcwd()
```

```
'C:\\Users\\PC-6'
```

```
os.chdir("C:\\Users\\PC-6\\Desktop")

df=pd.read_csv("Housing.csv")

df.head()
```

```
      price  area  bedrooms  bathrooms  stories mainroad guestroom
basement  \
0  13300000  7420         4          2        3      yes        no
no
1  12250000  8960         4          4        4      yes        no
no
2  12250000  9960         3          2        2      yes        no
yes
3  12215000  7500         4          2        2      yes        no
yes
4  11410000  7420         4          1        2      yes       yes
yes

  hotwaterheating airconditioning  parking prefarea furnishingstatus
0              no              yes        2      yes        furnished
1              no              yes        3       no        furnished
2              no               no        2      yes   semi-furnished
3              no              yes        3      yes        furnished
4              no              yes        2       no        furnished
```

```
df.isna().sum()
```

```
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
```

```
df['furnishingstatus'].nunique()
```

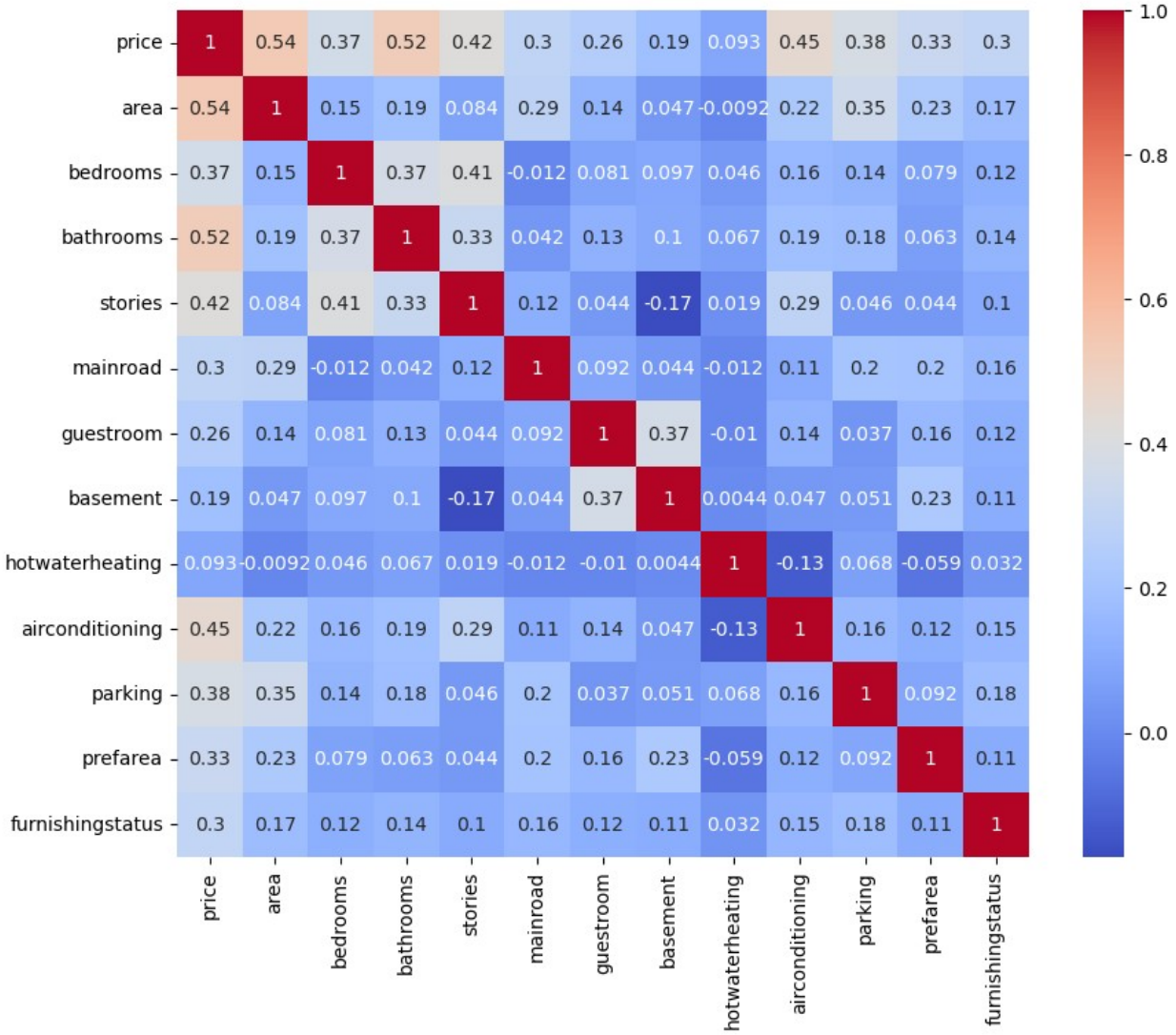
```
3
```

```
columns_to_transform = ['mainroad', 'guestroom',
'basement','hotwaterheating','airconditioning','prefarea']
df[columns_to_transform] = df[columns_to_transform].replace({'yes': 1,
'no': 0})

df['furnishingstatus'] =
df['furnishingstatus'].replace({'unfurnished': 0, 'semi-furnished': 1,
'furnished': 2})

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
sc = ['price', 'area']
df[sc] = scaler.fit_transform(df[sc])

df
```

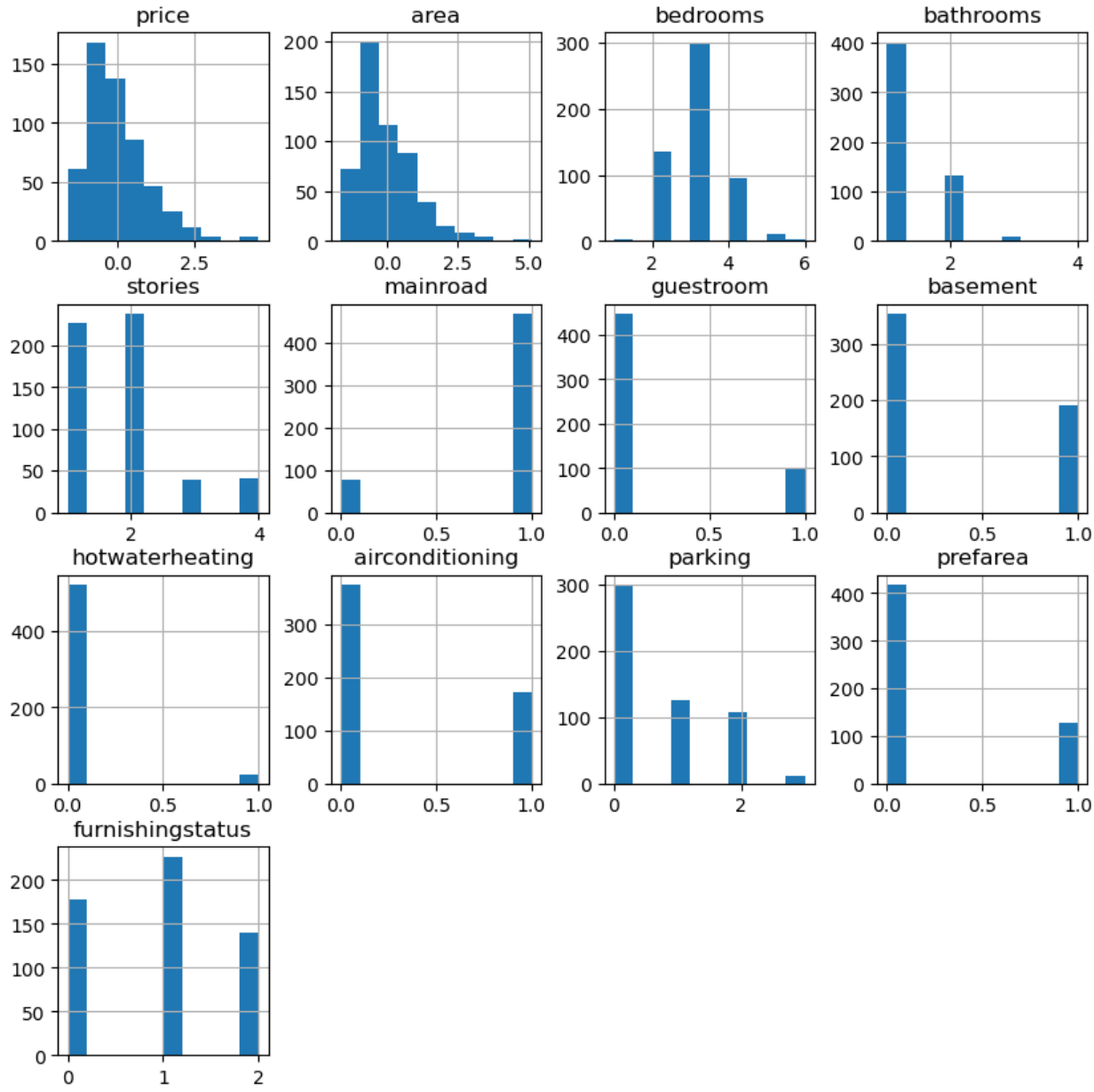
```
        price      area  bedrooms  bathrooms  stories  mainroad
guestroom  \
0    4.566365  1.046726         4          2        3         1
0
1    4.004484  1.757010         4          4        4         1
0
2    4.004484  2.218232         3          2        2         1
0
3    3.985755  1.083624         4          2        2         1
0
4    3.554979  1.046726         4          1        2         1
1
..        ...       ...       ...        ...      ...       ...
...
540 -1.576868 -0.991879         2          1        1         1
0
541 -1.605149 -1.268613         3          1        1         0
0
542 -1.614327 -0.705921         2          1        1         1
0
```

```
543 -1.614327 -1.033389         3           1        1         0
0
544 -1.614327 -0.599839         3           1        2         1
0

     basement  hotwaterheating  airconditioning  parking  prefarea  \
0           0                0                1        2         1
1           0                0                1        3         0
2           1                0                0        2         1
3           1                0                1        3         1
4           1                0                1        2         0
..        ...              ...              ...      ...       ...
540         1                0                0        2         0
541         0                0                0        0         0
542         0                0                0        0         0
543         0                0                0        0         0
544         0                0                0        0         0

     furnishingstatus
0                   2
1                   2
2                   1
3                   2
4                   2
..                ...
540                 0
541                 1
542                 0
543                 2
544                 0

[545 rows x 13 columns]

df.dtypes

price               float64
area                float64
bedrooms              int64
bathrooms             int64
stories               int64
mainroad              int64
guestroom             int64
basement              int64
hotwaterheating       int64
airconditioning       int64
parking               int64
prefarea              int64
furnishingstatus      int64
dtype: object
```

```
corr_matrix = df.corr()

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm")
plt.show()
```

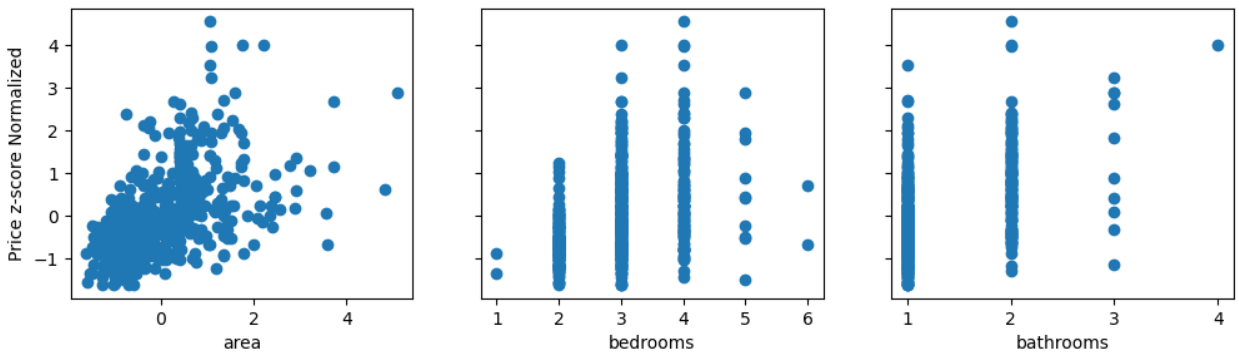



```
df.hist(figsize=(10, 10), bins=10)
plt.suptitle("Histograms for All Columns", fontsize=16)
plt.show()
```




```
X = df.drop('price', axis=1)
y = df['price']

X_features = ['area', 'bedrooms','bathrooms']

fig, ax = plt.subplots(1, 3, figsize=(12, 3), sharey=True)
for i in range(3):  # Assuming there are 4 features
    ax[i].scatter(X.iloc[:, i], y)
    ax[i].set_xlabel(X_features[i])
```

```
ax[0].set_ylabel("Price z-score Normalized")
plt.show()
```

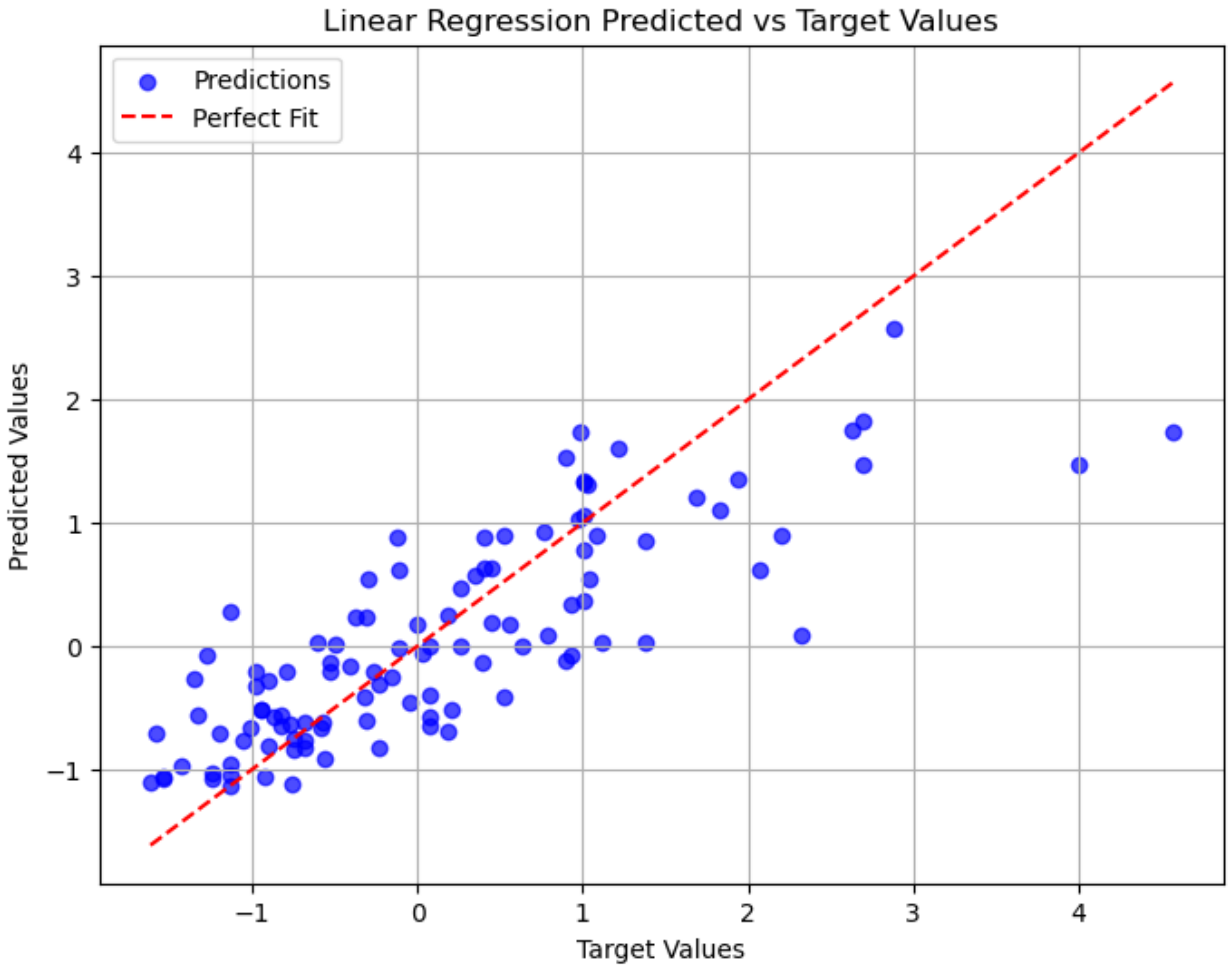



```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

from sklearn.linear_model import LinearRegression

lr_model = LinearRegression()
lr_model.fit(X_train, y_train)

y_pred = lr_model.predict(X_test)

from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import GridSearchCV

dt_model = DecisionTreeRegressor(random_state=42)

param_grid = {
    'max_depth': [5, 10, 15, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 5, 10],
    'max_features': ['auto', 'sqrt', 'log2'],
    'criterion': ['squared_error', 'friedman_mse', 'poisson',
'absolute_error']
}

grid_search = GridSearchCV(estimator=dt_model, param_grid=param_grid,
cv=5, scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)

print("Best Parameters:", grid_search.best_params_)

best_dt_model = grid_search.best_estimator_

Best Parameters: {'criterion': 'absolute_error', 'max_depth': 15,
'max_features': 'sqrt', 'min_samples_leaf': 5, 'min_samples_split': 2}
```

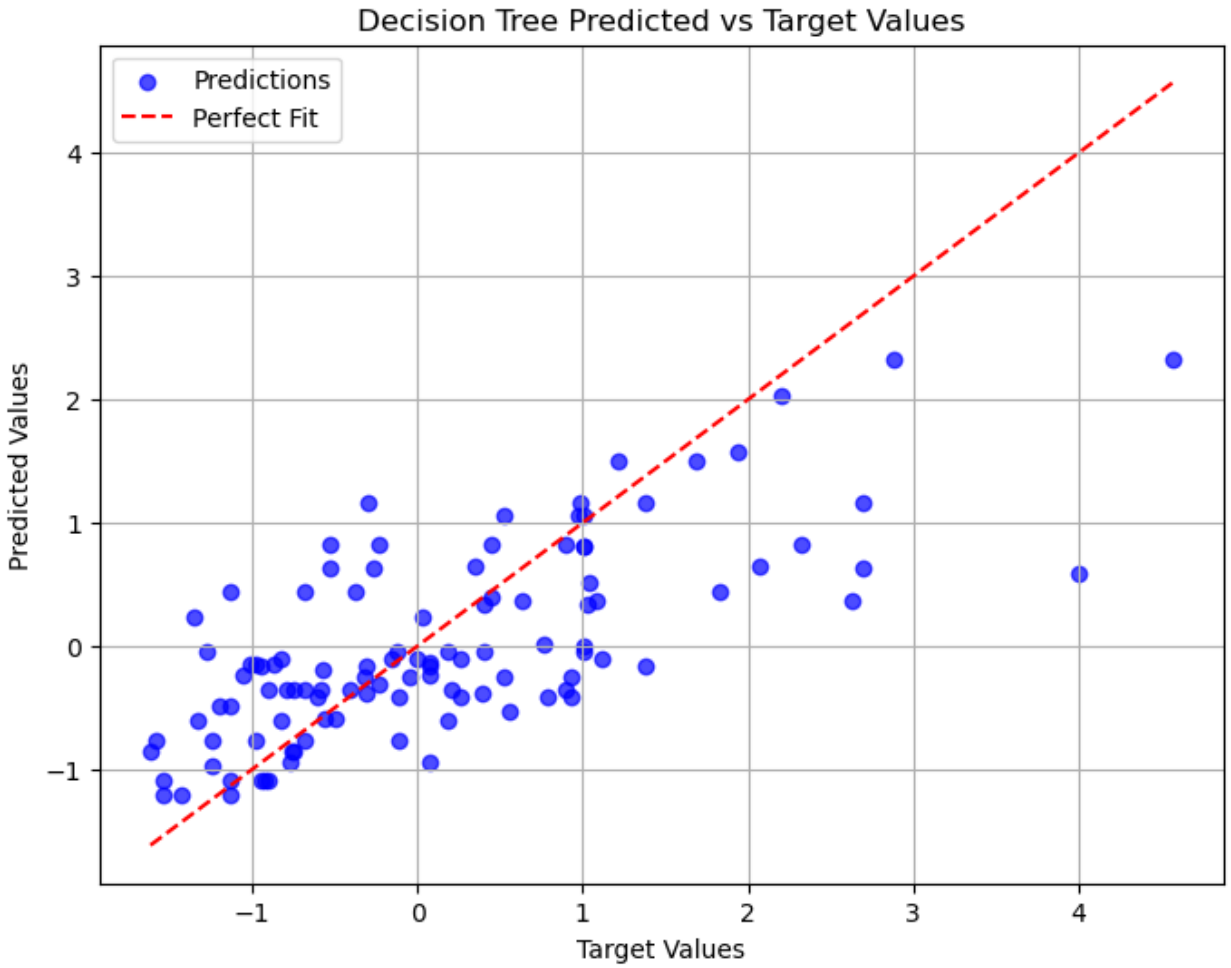
```
lr_y_pred = lr_model.predict(X_test)
dt_y_pred = best_dt_model.predict(X_test)

from sklearn.metrics import r2_score
lr_acc = r2_score(y_test, lr_y_pred)
dt_acc = r2_score(y_test, dt_y_pred)

print(f"Linear Regression Accuracy: {lr_acc}")
print(f"Decision Tree Accuracy: {dt_acc}")
```

```
Linear Regression Accuracy: 0.6494754192267798
Decision Tree Accuracy: 0.4928485257199178
```

```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, lr_y_pred, color='blue', alpha=0.7,
label='Predictions')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', linestyle='--', label='Perfect Fit')

plt.xlabel('Target Values')
plt.ylabel('Predicted Values')
plt.title('Linear Regression Predicted vs Target Values')
plt.legend()
plt.grid(True)

plt.show()
```




```
plt.figure(figsize=(8, 6))
plt.scatter(y_test, dt_y_pred, color='blue', alpha=0.7,
label='Predictions')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', linestyle='--', label='Perfect Fit')

plt.xlabel('Target Values')
plt.ylabel('Predicted Values')
plt.title('Decision Tree Predicted vs Target Values')
plt.legend()
plt.grid(True)

plt.show()
```




```
from sklearn.linear_model import SGDRegressor

sgdr = SGDRegressor(max_iter=1000)
sgdr.fit(X_train, y_train)
print(sgdr)
print(f"number of iterations completed: {sgdr.n_iter_}, number of
weight updates: {sgdr.t_}")
```

```
SGDRegressor()
number of iterations completed: 34, number of weight updates: 14825.0
```

```
b_norm = sgdr.intercept_
w_norm = sgdr.coef_
print(f"model parameters:                   w: {w_norm}, b:{b_norm}")
```

```
model parameters:                   w: [ 0.33710478 -0.14391223
0.44655225  0.22453196 -0.14067121  0.13448173
  0.20822998  0.18997849  0.43576092  0.12443931  0.32087466
0.09175696], b:[-0.98702889]
```

```
y_pred_sgd = sgdr.predict(X_train)
y_pred = np.dot(X_train, w_norm) + b_norm
print(f"prediction using np.dot() and sgdr.predict match: {(y_pred ==
y_pred_sgd).all()}")

print(f"Prediction on training set:\n{y_pred[:4]}" )
print(f"Target values \n{y_train[:4]}")
```

```
prediction using np.dot() and sgdr.predict match: True
Prediction on training set:
[ 1.1075839   0.98591619  0.12444803 -0.52972045]
Target values
46     1.476019
93     0.820491
335   -0.453106
412   -0.715317
Name: price, dtype: float64
```

```
fig,ax=plt.subplots(1,3,figsize=(12,3),sharey=True)
for i in range(len(ax)):
    ax[i].scatter(X_train.iloc[:,i],y_train, label = 'target')
    ax[i].set_xlabel(X_features[i])
    ax[i].scatter(X_train.iloc[:,i],y_pred,color="orange", label =
'predict')
ax[0].set_ylabel("Price"); ax[0].legend();
fig.suptitle("target versus prediction using z-score normalized
model")
plt.show()
```