

# linear\_regression

August 5, 2024

## 0.0.1 ML-A1 Implementation of Linear regression

- Instructions
  - Prepare a report to present your findings
  - Write a python code to implement stochastic gradient descent from scratch for the given house price prediction dataset.
  - Write a python code to implement stochastic gradient descent using scikit-learn for the given data and compare the output.
  - Write a python code to implement batch gradient descent from scratch and also using scikit-learn for the given house price prediction.
  - Compare the output of all the implementations and write conclusion.

Dataset: [House Price Prediction Challenge \(kaggle.com\)](https://www.kaggle.com/c/house-prices-advanced-regression-techniques)

- Submission Intruction:
  - Submission should include python notebook file for all the implementations.
  - There must be a report pdf file to illustrate your data science lifecycle implementation and present your finds. Report must not exceed 10 page or 1500 words.

## Table of Content

1. Problem Statement
2. Data Understanding
3. Data Preparation
4. Modeling
  - 4.1. SGD using scikit-learn
  - 4.2 Batch Gradient Descent using scikit-learn
  - 4.3 SGD Scratch Implementation
  - 4.4 Batch Gradient Descent Scratch Implementation

## Data Ingestion/Loading

- Load the necessary modules
- Load the data
- process the data

```
[ ]: # loading the library
import numpy as np
import pandas as pd
```

```

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split

plt.rcParams['figure.figsize'] = (18,6)

np.random.seed(32)
# data path
train_data = "/home/suman/Applied-Machine-Learning/Linear Regression/train.csv"
test_data = "/home/suman/Applied-Machine-Learning/Linear Regression/test.csv"

# load the train and test data
df_train = pd.read_csv(train_data)
df_test = pd.read_csv(test_data)

df_train.head()

```

```

[ ]:  POSTED_BY  UNDER_CONSTRUCTION  RERA  BHK_NO.  BHK_OR_RK    SQUARE_FT  \
0      Owner                0      0        2      BHK    1300.236407
1      Dealer                0      0        2      BHK    1275.000000
2      Owner                0      0        2      BHK     933.159722
3      Owner                0      1        2      BHK     929.921143
4      Dealer                1      0        2      BHK     999.009247

      READY_TO_MOVE  RESALE                ADDRESS  LONGITUDE  LATITUDE  \
0                1      1      Ksfc Layout,Bangalore  12.969910  77.597960
1                1      1  Vishweshwara Nagar,Mysore  12.274538  76.644605
2                1      1      Jigani,Bangalore  12.778033  77.632191
3                1      1  Sector-1 Vaishali,Ghaziabad  28.642300  77.344500
4                0      1      New Town,Kolkata  22.592200  88.484911

      TARGET(PRICE_IN_LACS)
0                55.0
1                51.0
2                43.0
3                62.5
4                60.5

```

**Data Understanding and Exploration** The dataset used for this assignmetn is from Kaggle Dataset: [House Price Prediction Challenge \(kaggle.com\)](https://www.kaggle.com/c/house-price-prediction-challenge)

- **Training Splits:** 29451 rows x 12 columns
- **Testing Splits:** 68720 x 11 columns

- since we are using competition data testing data do not contain the labels, they are evaluated based on this splits.

- **Attributes of the Dataset**

Column	Description
POSTED_BY	Category marking who has listed the property
UNDER_CONSTRUCTION	Under Construction or Not
RERA	Rera approved or Not
BHK_NO	Number of Rooms
BHK_OR_RK	Type of property
SQUARE_FT	Total area of the house in square feet
READY_TO_MOVE	Category marking Ready to move or Not
RESALE	Category marking Resale or not
ADDRESS	Address of the property
LONGITUDE	Longitude of the property
LATITUDE	Latitude of the property

RERA stands for Real Estate (Regulation and Development) Act, which was enacted by the Indian government in 2016. It aims to protect home buyers and ensure transparency in the real estate sector. RERA establishes regulatory authorities at the state level to oversee real estate transactions and address grievances.

```
[ ]: print(f'The train dataset contains {df_train.shape[0]} rows and {df_train.
      ↪shape[0]} columns.')
print(f'The train dataset contains {df_test.shape[0]} rows and {df_test.
      ↪shape[0]} columns.')
```

The train dataset contains 29451 rows and 29451 columns.

The train dataset contains 68720 rows and 68720 columns.

```
[ ]: # info of the dataset
df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 29451 entries, 0 to 29450
Data columns (total 12 columns):
#   Column              Non-Null Count  Dtype
---  -
0   POSTED_BY           29451 non-null  object
1   UNDER_CONSTRUCTION  29451 non-null  int64
2   RERA                 29451 non-null  int64
3   BHK_NO              29451 non-null  int64
4   BHK_OR_RK           29451 non-null  object
5   SQUARE_FT           29451 non-null  float64
6   READY_TO_MOVE       29451 non-null  int64
7   RESALE              29451 non-null  int64
8   ADDRESS              29451 non-null  object
```

```

9    LONGITUDE                29451 non-null float64
10   LATITUDE                 29451 non-null float64
11   TARGET(PRICE_IN_LACS)    29451 non-null float64
dtypes: float64(4), int64(5), object(3)
memory usage: 2.7+ MB

```

- since we are solving linear regression problem, the target or dependent variable must be continous and here we can see that it is continous
- rest we can see that there are total of three dtypes
- there are two categorical variable which are useful: BHK\_OR\_RK and POSTED\_BY

## Exploration and Descriptive Statistics

```
[ ]: df_train.describe()
```

```
[ ]:
      UNDER_CONSTRUCTION      RERA      BHK_NO.      SQUARE_FT \
count      29451.000000    29451.000000    29451.000000    2.945100e+04
mean           0.179756      0.317918      2.392279    1.980217e+04
std           0.383991      0.465675      0.879091    1.901335e+06
min           0.000000      0.000000      1.000000    3.000000e+00
25%           0.000000      0.000000      2.000000    9.000211e+02
50%           0.000000      0.000000      2.000000    1.175057e+03
75%           0.000000      1.000000      3.000000    1.550688e+03
max           1.000000      1.000000     20.000000    2.545455e+08

```

```

      READY_TO_MOVE      RESALE      LONGITUDE      LATITUDE \
count      29451.000000    29451.000000    29451.000000    29451.000000
mean           0.820244      0.929578      21.300255      76.837695
std           0.383991      0.255861      6.205306      10.557747
min           0.000000      0.000000     -37.713008     -121.761248
25%           1.000000      1.000000      18.452663      73.798100
50%           1.000000      1.000000      20.750000      77.324137
75%           1.000000      1.000000      26.900926      77.828740
max           1.000000      1.000000      59.912884     152.962676

```

```

      TARGET(PRICE_IN_LACS)
count      29451.000000
mean           142.898746
std           656.880713
min           0.250000
25%           38.000000
50%           62.000000
75%          100.000000
max          30000.000000

```

```
[ ]: # categorical data
```

```
df_train.describe(exclude=["float", "int"])
```

```
[ ]:      POSTED_BY BHK_OR_RK      ADDRESS
count      29451      29451      29451
unique         3         2      6899
top      Dealer      BHK  Zirakpur,Chandigarh
freq      18291      29427      509
```

```
[ ]: # check for null values
df_train.isnull().sum()
```

```
[ ]: POSTED_BY      0
UNDER_CONSTRUCTION  0
RERA                0
BHK_NO.            0
BHK_OR_RK          0
SQUARE_FT          0
READY_TO_MOVE      0
RESALE             0
ADDRESS            0
LONGITUDE          0
LATITUDE           0
TARGET(PRICE_IN_LACS)  0
dtype: int64
```

### Interpretations:

- there are numerical and categorical variables
- the dataset have no missing records (since this is competition data it is already been curated)
- target/dependent variable is continous (as float dtype)

### Basic EDA

```
[ ]: df_train['POSTED_BY'].value_counts()
```

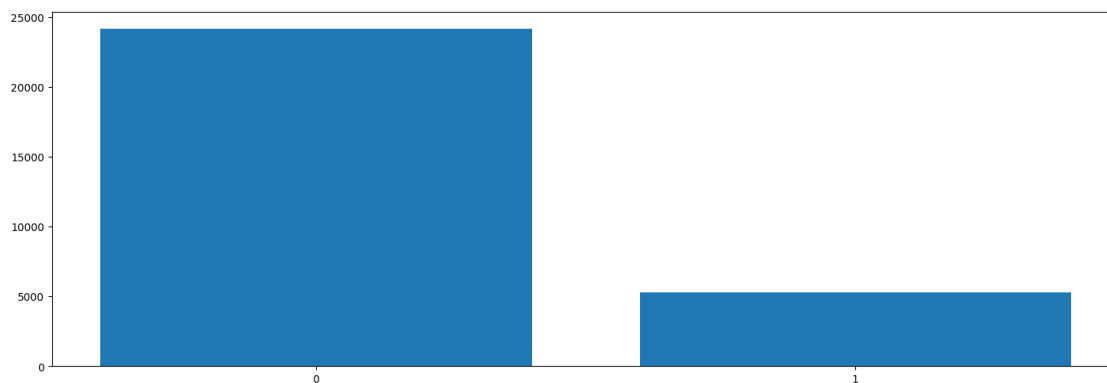
```
[ ]: POSTED_BY
Dealer      18291
Owner       10538
Builder      622
Name: count, dtype: int64
```

```
[ ]: df_train['UNDER_CONSTRUCTION'].value_counts()
```

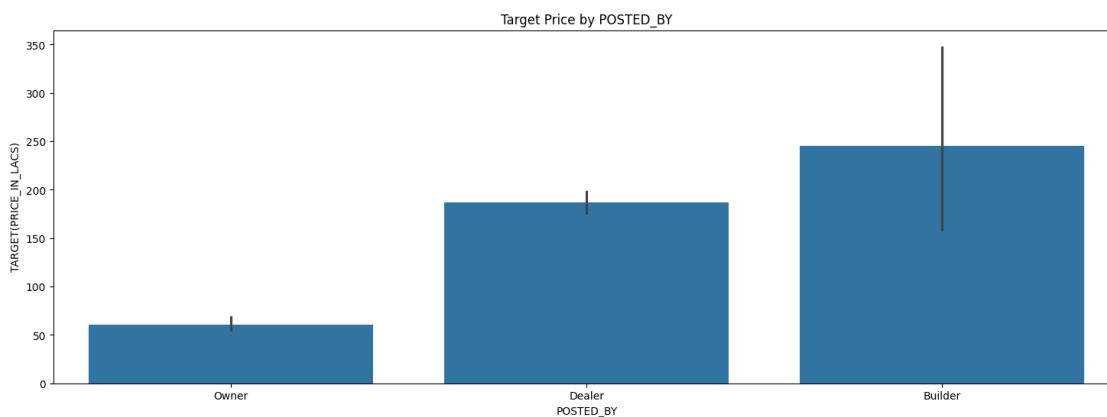
```
[ ]: UNDER_CONSTRUCTION
0      24157
1       5294
Name: count, dtype: int64
```

```
[ ]: plt.bar(["0","1"],df_train["UNDER_CONSTRUCTION"].value_counts())
```

```
[ ]: <BarContainer object of 2 artists>
```

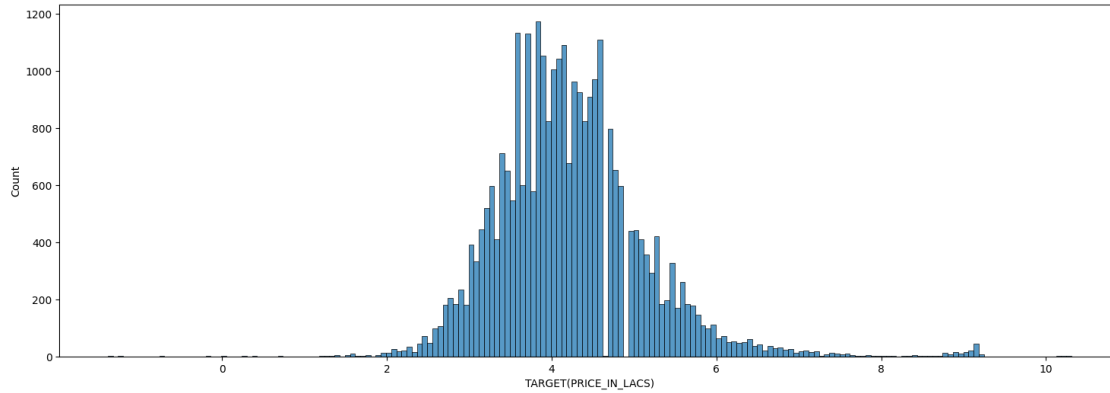


```
[ ]: sns.barplot(x='POSTED_BY', y='TARGET(PRICE_IN_LACS)', data=df_train)
plt.title('Target Price by POSTED_BY')
plt.show()
```



```
[ ]: sns.histplot(np.log(df_train["TARGET(PRICE_IN_LACS)"]))
```

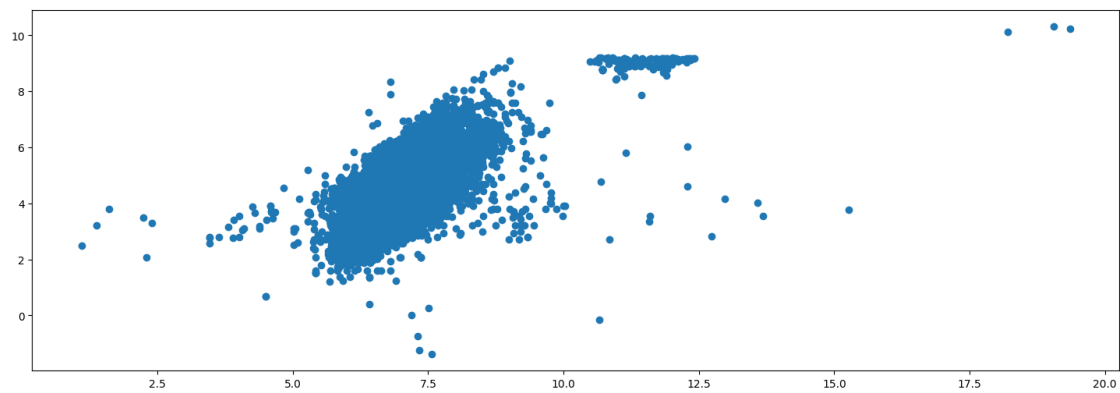
```
[ ]: <Axes: xlabel='TARGET(PRICE_IN_LACS)', ylabel='Count'>
```



```
[ ]: # check the relationship between square ft and the price
# does square ft influences price?
plt.scatter(x=np.log(df_train["SQUARE_FT"]), y=np.
    ↳log(df_train["TARGET(PRICE_IN_LACS)"]))

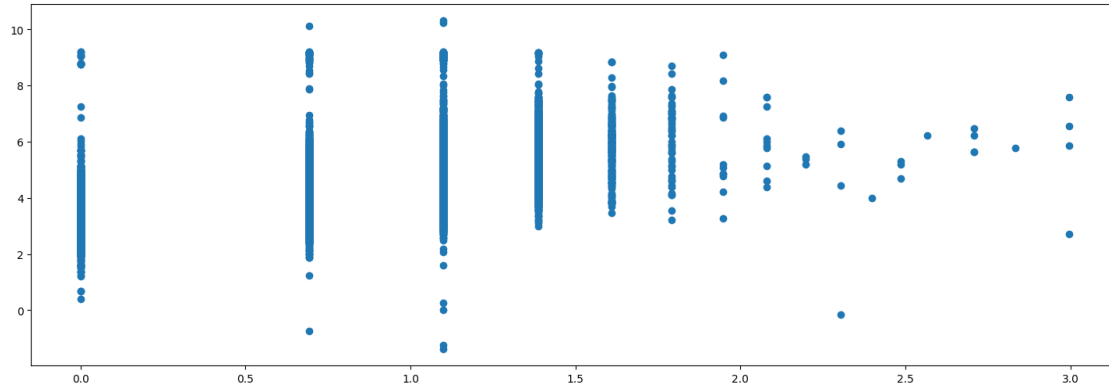
# looks like it does
```

```
[ ]: <matplotlib.collections.PathCollection at 0x7f1f5cd06f90>
```



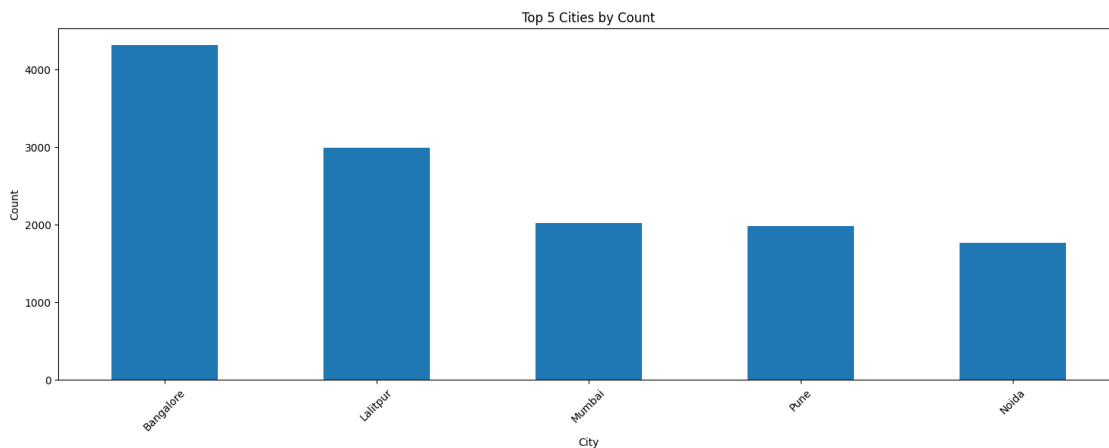
```
[ ]: plt.scatter(x=np.log(df_train["BHK_NO."]), y=np.
    ↳log(df_train["TARGET(PRICE_IN_LACS)"]))
```

```
[ ]: <matplotlib.collections.PathCollection at 0x7f1f5cae22d0>
```



```
[ ]: # which city is most popular?
new_df = df_train['ADDRESS'].str.split(',').str.get(1)
city_counts = new_df.value_counts().head(5)
```

```
[ ]: city_counts.plot(kind='bar')
plt.title('Top 5 Cities by Count')
plt.xlabel('City')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



```
[ ]: df_train_copy = df_train.copy()
```

```
[ ]: # remove the unwanted columns
df_train = df_train.drop(['ADDRESS', 'LONGITUDE', 'LATITUDE'], axis=1)
df_train
```



```

-----
KeyError                                Traceback (most recent call last)
Cell In[19], line 2
      1 # remove the unwanted columns
----> 2 df_train = df_train.drop(['ADDRESS', 'LONGITUDE', 'LATITUDE'], axis=1)
      3 df_train

File ~/conda/envs/documentai/lib/python3.12/site-packages/pandas/core/frame.py:
  5581, in DataFrame.drop(self, labels, axis, index, columns, level, inplace,
  errors)
    5433 def drop(
    5434     self,
    5435     labels: IndexLabel | None = None,
    (...)
    5442     errors: IgnoreRaise = "raise",
    5443 ) -> DataFrame | None:
    5444     """
    5445     Drop specified labels from rows or columns.
    5446
    (...)
    5579         weight 1.0      0.8
    5580     """
-> 5581     return super().drop(
    5582         labels=labels,
    5583         axis=axis,
    5584         index=index,
    5585         columns=columns,
    5586         level=level,
    5587         inplace=inplace,
    5588         errors=errors,
    5589     )

File ~/conda/envs/documentai/lib/python3.12/site-packages/pandas/core/generic.py:
  4788, in NDFrame.drop(self, labels, axis, index, columns, level, inplace,
  errors)
    4786 for axis, labels in axes.items():
    4787     if labels is not None:
-> 4788         obj = obj._drop_axis(labels, axis, level=level, errors=errors)
    4790 if inplace:
    4791     self._update_inplace(obj)

File ~/conda/envs/documentai/lib/python3.12/site-packages/pandas/core/generic.py:
  4830, in NDFrame._drop_axis(self, labels, axis, level, errors, only_slice)
    4828     new_axis = axis.drop(labels, level=level, errors=errors)
    4829     else:
-> 4830         new_axis = axis.drop(labels, errors=errors)
    4831     indexer = axis.get_indexer(new_axis)

```

```
4833 # Case for non-unique axis
4834 else:
```

```
File ~/.conda/envs/documentai/lib/python3.12/site-packages/pandas/core/indexes/
↳ base.py:7070, in Index.drop(self, labels, errors)
```

```
7068 if mask.any():
7069     if errors != "ignore":
-> 7070         raise KeyError(f"{labels[mask].tolist()} not found in axis")
7071     indexer = indexer[~mask]
7072 return self.delete(indexer)
```

```
KeyError: "['ADDRESS', 'LONGITUDE', 'LATITUDE'] not found in axis"
```

```
[ ]:
```