Capstone Project (Insight Module)

- We'll find out which factors are responsible for price & by how much %.

```
import numpy as np
import pandas as pd

from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LinearRegression,Ridge
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler, OrdinalE
ncoder
from sklearn.compose import ColumnTransformer

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

from sklearn.decomposition import PCA
```

```
df = pd.read_csv('gurgaon_properties_post_feature_selection_v2.csv').drop(c
olumns=['store room','floor_category','balcony'])

df.head()
```

	property_type	sector	price	bedRoom	bathroom	agePossession	built_up_area	servant room	furnishing_type	luxury_category
0	flat	sector 36	0.82	3.0	2.0	New Property	850.0	0.0	0.0	Low
1	flat	sector 89	0.95	2.0	2.0	New Property	1226.0	1.0	0.0	Low
2	flat	sohna road	0.32	2.0	2.0	New Property	1000.0	0.0	0.0	Low
3	flat	sector 92	1.60	3.0	4.0	Relatively New	1615.0	1.0	1.0	High
4	flat	sector 102	0.48	2.0	2.0	Relatively New	582.0	0.0	0.0	High

```
# 0 → unfurnished
# 1 → semifurnished
# 2 → furnished

# Numerical = bedRoom, bathroom, built_up_area, servant room
# Ordinal = property_type, furnishing_type, luxury_category
# OHE = sector, agePossession
```

```
df['agePossession'].value_counts()
```

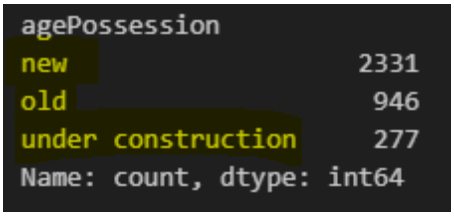
```
agePossession
Relatively New        1732
Moderately Old         619
New Property           599
Old Property           327
Under Construction     277
Name: count, dtype: int64
```

- ***Let's reduce the no. of categories to 3***

```
df['agePossession'] = df['agePossession'].replace(
   {
      'Relatively New':'new',
      'Moderately Old':'old',
      'New Property' : 'new',
      'Old Property' : 'old',
```

```
    'Under Construction' : 'under construction'
  }
)
```

```
df['agePossession'].value_counts()
```



- Now convert
  - House → 1
  - Flat → 0

```
df['property_type'] = df['property_type'].replace({'flat':0,'house':1}).astype('int64')
```

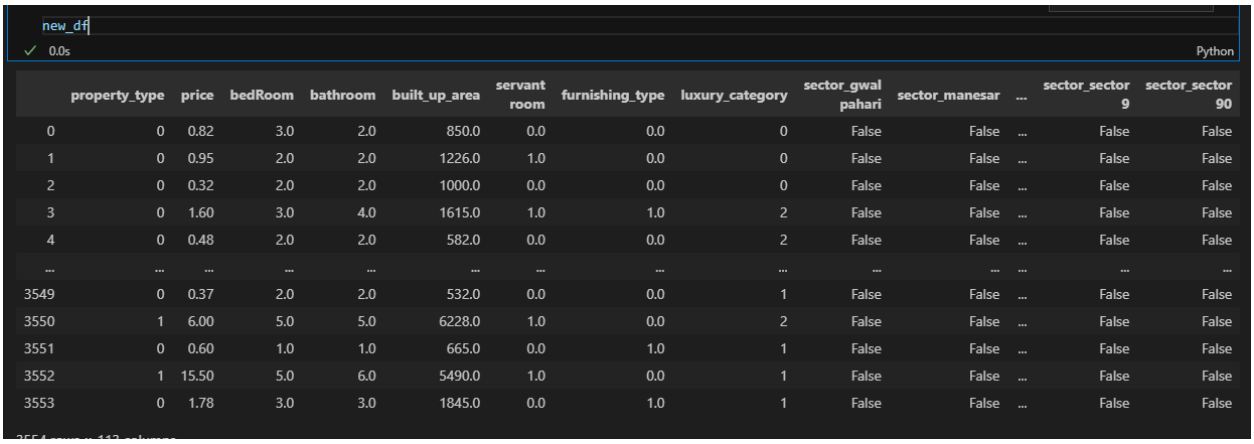
- **Deprecated Behavior:** Starting with pandas 2.1 (released in November 2023), this automatic downcasting in replace is deprecated. In a future version (likely pandas 3.0), pandas will stop doing this automatically to make type changes more explicit and avoid unexpected behavior.
- **Future Behavior:** In the future, replace will keep the original dtype (e.g., object) unless you explicitly tell pandas to change it. So, after replacing 'flat' with 0, the column might still be object with values like 0 and 1 as objects, not integers.

`.astype('int64')` → To make the code futureproof

```
df['luxury_category'] = df['luxury_category'].replace({'Low':0,'Medium':1,'High':2}).astype('int64')
```

Apply One Hot Encoding (OHE)

```
new_df = pd.get_dummies(df,columns=['sector','agePossession'],drop_first=True)
```



```
new_df
```

0.0s

Python

	property_type	price	bedRoom	bathroom	built_up_area	servant room	furnishing_type	luxury_category	sector_gwal pahari	sector_manesar	...	sector_sector 9	sector_sector 90
0	0	0.82	3.0	2.0	850.0	0.0	0.0	0	False	False	...	False	False
1	0	0.95	2.0	2.0	1226.0	1.0	0.0	0	False	False	...	False	False
2	0	0.32	2.0	2.0	1000.0	0.0	0.0	0	False	False	...	False	False
3	0	1.60	3.0	4.0	1615.0	1.0	1.0	2	False	False	...	False	False
4	0	0.48	2.0	2.0	582.0	0.0	0.0	2	False	False	...	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...
3549	0	0.37	2.0	2.0	532.0	0.0	0.0	1	False	False	...	False	False
3550	1	6.00	5.0	5.0	6228.0	1.0	0.0	2	False	False	...	False	False
3551	0	0.60	1.0	1.0	665.0	0.0	1.0	1	False	False	...	False	False
3552	1	15.50	5.0	6.0	5490.0	1.0	0.0	1	False	False	...	False	False
3553	0	1.78	3.0	3.0	1845.0	0.0	1.0	1	False	False	...	False	False

```
X = new_df.drop(columns=['price'])
y = new_df['price']
```

Log of Y:

```
y_log = np.log1p(y)
```

```
y_log
✓ 0.0s

0       0.598837
1       0.667829
2       0.277632
3       0.955511
4       0.392042
          ...
3549    0.314811
3550    1.945910
3551    0.470004
3552    2.803360
3553    1.022451
Name: price, Length: 3554, dtype: float64
```

StandardScaler

```
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

X_scaled = pd.DataFrame(X_scaled,columns=X.columns)
```

```
X_scaled
✓ 0.0s
```

	property_type	bedRoom	bathroom	built_up_area	servant room	furnishing_type	luxury_category	sector_gwal pahari	sector_manesar	sector_sector 1	...	sector_sector 9
0	-0.517180	-0.074329	-0.874300	-0.831662	-0.747968	-0.668281	-0.984642	-0.071348	-0.093805	-0.041123	...	-0.078923
1	-0.517180	-0.877269	-0.874300	-0.522517	1.336956	-0.668281	-0.984642	-0.071348	-0.093805	-0.041123	...	-0.078923
2	-0.517180	-0.877269	-0.874300	-0.708333	-0.747968	-0.668281	-0.984642	-0.071348	-0.093805	-0.041123	...	-0.078923
3	-0.517180	-0.074329	0.505173	-0.202684	1.336956	1.037949	1.866207	-0.071348	-0.093805	-0.041123	...	-0.078923
4	-0.517180	-0.877269	-0.874300	-1.052010	-0.747968	-0.668281	1.866207	-0.071348	-0.093805	-0.041123	...	-0.078923
...	...	...	...	...	...	...	...	...	...	...	...	...
3549	-0.517180	-0.877269	-0.874300	-1.093119	-0.747968	-0.668281	0.440783	-0.071348	-0.093805	-0.041123	...	-0.078923
3550	1.933563	1.531549	1.194909	3.590095	1.336956	-0.668281	1.866207	-0.071348	-0.093805	-0.041123	...	-0.078923
3551	-0.517180	-1.680208	-1.564036	-0.983768	-0.747968	1.037949	0.440783	-0.071348	-0.093805	-0.041123	...	-0.078923
3552	1.933563	1.531549	1.884645	2.983317	1.336956	-0.668281	0.440783	-0.071348	-0.093805	-0.041123	...	-0.078923
3553	-0.517180	-0.074329	-0.184564	-0.013579	-0.747968	1.037949	0.440783	-0.071348	-0.093805	-0.041123	...	-0.078923

3554 rows × 112 columns

Cross validation

```
kfold = KFold(n_splits=10, shuffle=True, random_state=42)
scores = cross_val_score(LinearRegression(), X_scaled, y_log, cv=kfold, scori
ng='r2')

scores.mean(),scores.std()
```

```
(0.8512613057405425, 0.016992929105286225)
```

Train Linear Regression Model

```
lr = LinearRegression()
lr.fit(X_scaled,y_log)
```

```
lr.coef_
```

```
array([ 1.20165036e-01,  5.40015778e-02,  6.51193988e-02,  2.10637561e-01,
        5.09461847e-02,  8.28171481e-03,  6.18569187e-03,  9.81917509e-03,
       -2.27136751e-02, -5.19263751e-03,  5.24027544e-03,  2.69999569e-02,
       -2.90628642e-03,  1.96385005e-03, -1.93922433e-02,  5.72258238e-04,
       -1.29872300e-02,  1.68170547e-02,  2.61496408e-02, -1.49660399e-02,
        8.73379008e-03,  1.62286944e-02,  3.08257676e-02,  3.21709327e-02,
       -1.94575986e-02, -1.22503903e-02,  2.87410970e-02,  3.29139011e-03,
        1.39170889e-02,  6.72176140e-03, -9.41789321e-03,  3.54737020e-02,
        2.19599278e-03,  1.77019789e-02,  5.75604298e-02,  7.38834785e-02,
        6.74158385e-03,  4.18540805e-02, -1.27147058e-02,  6.27615381e-03,
        2.24917111e-02,  2.59569793e-02,  2.49451572e-03, -1.14658162e-02,
        1.14313378e-03,  1.21374481e-02,  2.57701788e-03, -2.07745570e-02,
        7.65349283e-03,  1.67856368e-02,  6.03752164e-02,  2.69570318e-02,
        1.58617969e-02,  1.28598727e-02,  5.41626440e-02,  3.17430014e-02,
       -1.72297943e-02,  7.18216283e-02,  2.10443533e-04,  4.84749150e-03,
        4.77635759e-02,  4.56437329e-02,  1.26161821e-02,  1.18773664e-02,
        2.52303521e-02,  2.36031705e-02,  3.18204019e-02, -1.57485005e-02,
        2.39413014e-02,  3.67191438e-02,  4.29101570e-02,  3.48405339e-02,
        2.35614842e-02,  7.10460874e-02,  5.28297760e-02,  4.09188005e-02,
        8.67820177e-03,  1.09346552e-02,  1.84268932e-02, -7.30098670e-03,
        5.52018115e-03,  1.71630841e-02, -3.27102156e-03,  3.02311567e-02,
       -1.43929417e-03,  2.21769328e-02,  2.57118679e-04,  7.78019604e-04,
       -4.29412787e-03,  1.31668013e-02, -1.34625877e-02,  1.98887698e-04,
        1.56836833e-02,  5.83181285e-04,  9.75235369e-03, -1.08282478e-03,
       -1.02957577e-02,  8.74730811e-03, -3.67080666e-03, -5.96770574e-05,
        8.60108585e-03,  2.02197856e-03, -1.32886614e-02, -1.29774114e-02,
       -1.20407225e-04, -2.85559210e-02, -3.97137146e-03, -2.52221082e-02,
       -1.03122117e-02, -2.95146415e-02, -7.89967670e-03,  1.53813156e-02])
```

- 112 coeff for 112 features (columns)

- We'll convert this into dataframe

```
pd.DataFrame(lr.coef_.reshape(1,112),columns=X.columns)
```

	property_type	bedRoom	bathroom	built_up_area	servant room	furnishing_type	luxury_category	sector_gwal pahari	sector_manesar	sector_sector 1	...	sector_sector 9	sector_se
0	0.120165	0.054002	0.065119	0.210638	0.050946	0.008282	0.006186	0.009819	-0.022714	-0.005193	...	-0.013289	-0.01

- `stack()` to convert the columns → rows

```
pd.DataFrame(lr.coef_.reshape(1,112),columns=X.columns).stack()
```

```
0  property_type                        0.120165
   bedRoom                              0.054002
   bathroom                             0.065119
   built_up_area                        0.210638
   servant room                         0.050946
                                           ...
   sector_sector 95                    -0.025222
   sector_sector 99                    -0.010312
   sector_sohna road                   -0.029515
   agePossession_old                   -0.007900
   agePossession_under construction     0.015381
Length: 112, dtype: float64
```

```
pd.DataFrame(lr.coef_.reshape(1,112),columns=X.columns).stack().reset_index()
```

	level_0	level_1	0
0	0	property_type	0.120165
1	0	bedRoom	0.054002
2	0	bathroom	0.065119
3	0	built_up_area	0.210638
4	0	servant room	0.050946
...	...	...	...
107	0	sector_sector 95	-0.025222
108	0	sector_sector 99	-0.010312
109	0	sector_sohna road	-0.029515
110	0	agePossession_old	-0.007900
111	0	agePossession_under construction	0.015381

- drop `level_0` **& rename**

```
coef_df = pd.DataFrame(lr.coef_.reshape(1,112),columns=X.columns).stack().reset_index().drop('level_0',axis=1).rename(columns={'level_1':'feature',0:'coef'})
```

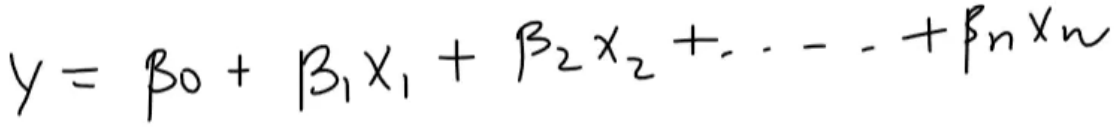
	feature	coef
0	property_type	0.120165
1	bedRoom	0.054002
2	bathroom	0.065119
3	built_up_area	0.210638
4	servant room	0.050946
...	...	...
107	sector_sector 95	-0.025222
108	sector_sector 99	-0.010312
109	sector_sohna road	-0.029515
110	agePossession_old	-0.007900
111	agePossession_under construction	0.015381

112 rows × 2 columns

Insights

If you change the value of bedroom from 1 → 2, how much difference does it make in price?

- ***Coeff*** will help answer this question



- If you increase X_2 by 1 Unit → y will increase by $ß_2$

What Happens When x_2 Increases by 1 Unit?

- Since β_2 is the coefficient of x_2, it tells you the change in (y) for a 1-unit increase in x_2, assuming all other variables ($x_1, x_3, \ldots, x_n$) stay the same.
- So, if x_2 increases by 1 (e.g., from 5 to 6), (y) changes by:

$$\text{Change in } y = \beta_2 \times 1 = \beta_2$$

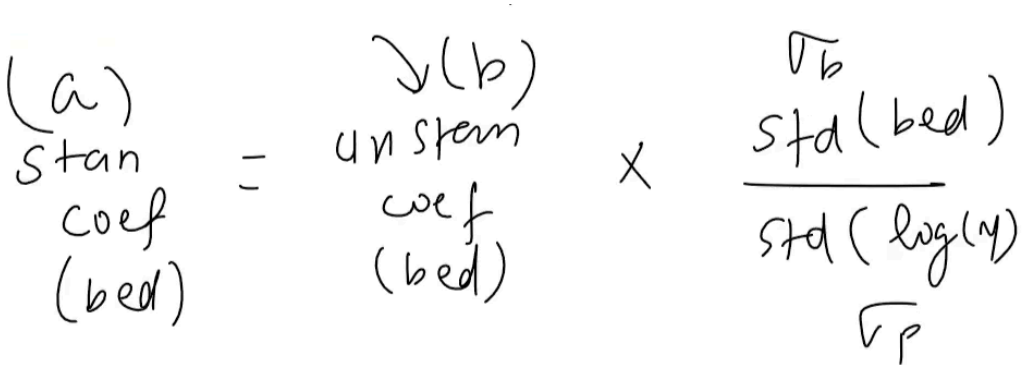
- If β_2 is known, you can calculate the exact new (y). For example:
  - If $\beta_2 = 3$, new $y = 1 + 3 = 4$.
  - If $\beta_2 = -2$, new $y = 1 + (-2) = -1$.
- Without knowing β_2, the new (y) is simply $1 + \beta_2$.

- But we have scaled X & applied log transformation on y

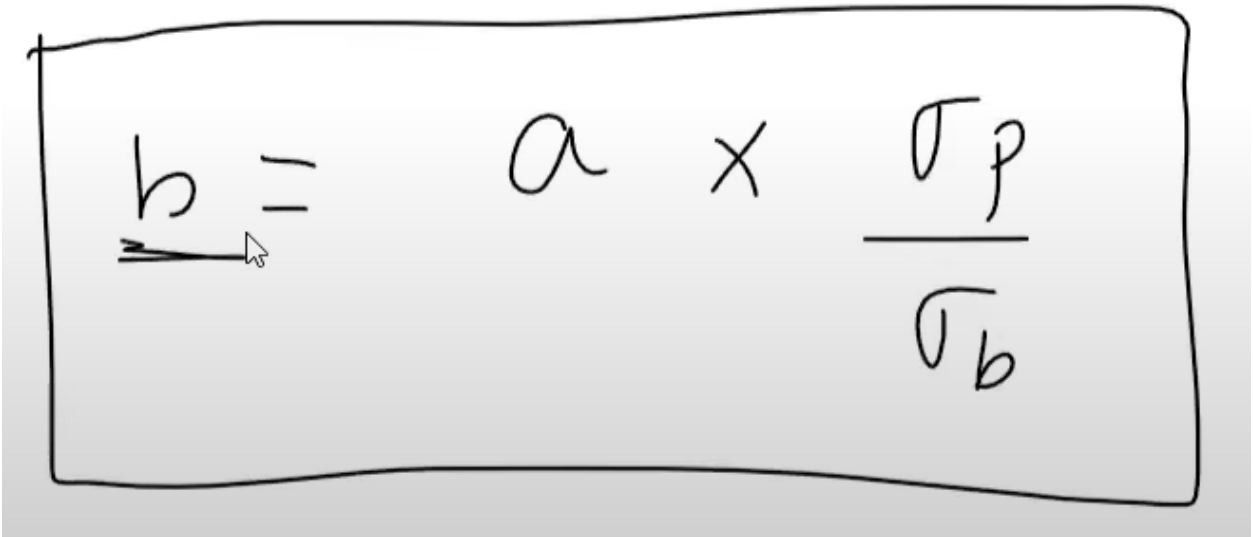
Calculate unstandardized coeff:

Standardized Coefficient for Linear Regression Formula

Standardized Coefficient (X1)	= Unstandardized Coefficient (X1)	* (Standard Deviation of x1 / Standard deviation of y)



We want unstandardized coeff:




```
y_log.std()

0.5579613263072812
```

Std of bedroom (before scaling)

```
X['bedRoom'].std()
```

1.245599503811857

```
0.054 * (0.557/1.245)
0.024159036144578313
```

- But we have taken log of y
- So, we have to calculate exponent of the above value

```
np.expm1(0.0241)
0.024392752044032903
```