Capstone Project (Feature Selection)

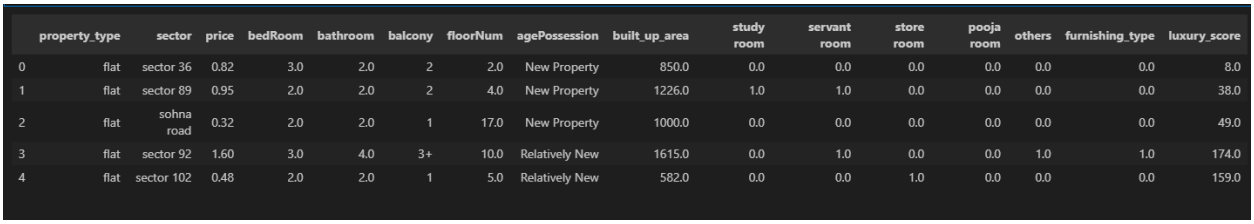
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df = pd.read_csv('gurgaon_properties_missing_value_imputation.csv')
```

- We'll drop the columns:
  - **price_per_sqft** → Highly correlated with price
  - **society** → Not useful in price prediction

```
train_df = df.drop(columns=['society','price_per_sqft'])
```

```
train_df.head()
```



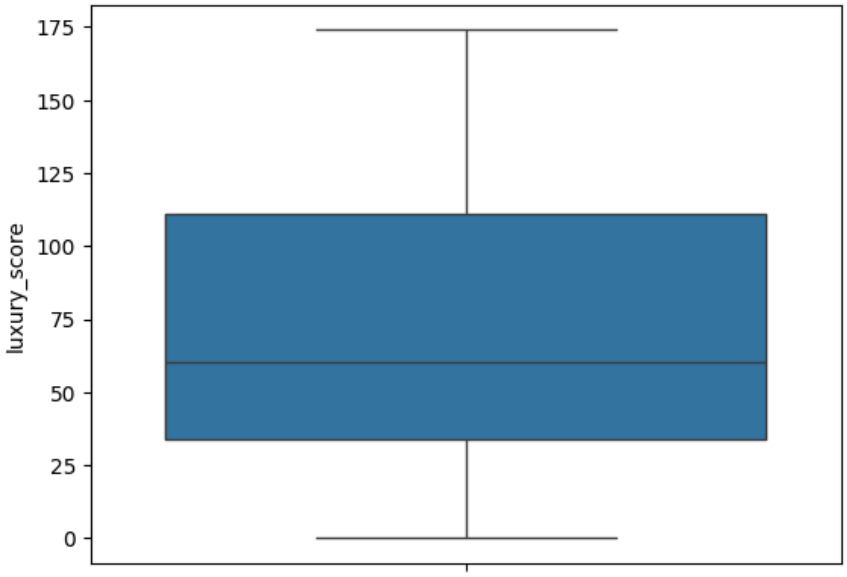
	property_type	sector	price	bedRoom	bathroom	balcony	floorNum	agePossession	built_up_area	study room	servant room	store room	pooja room	others	furnishing_type	luxury_score
0	flat	sector 36	0.82	3.0	2.0	2	2.0	New Property	850.0	0.0	0.0	0.0	0.0	0.0	0.0	8.0
1	flat	sector 89	0.95	2.0	2.0	2	4.0	New Property	1226.0	1.0	1.0	0.0	0.0	0.0	0.0	38.0
2	flat	sohna road	0.32	2.0	2.0	1	17.0	New Property	1000.0	0.0	0.0	0.0	0.0	0.0	0.0	49.0
3	flat	sector 92	1.60	3.0	4.0	3+	10.0	Relatively New	1615.0	0.0	1.0	0.0	0.0	1.0	1.0	174.0
4	flat	sector 102	0.48	2.0	2.0	1	5.0	Relatively New	582.0	0.0	0.0	1.0	0.0	0.0	0.0	159.0

luxury_category

- Convert `luxury_score` to categorical column:
  - Budget

- Semi-luxury
- Luxury

```
sns.boxplot(df['luxury_score'])
```




```
def categorize_luxury(score):
    if 0 <= score < 50:
        return "Low"
    elif 50 <= score < 150:
        return "Medium"
    elif 150 <= score <= 175:
        return "High"
    else:
        return None  # or "Undefined" or any other label for scores outside the d
efined bins
```

```
train_df['luxury_category'] = train_df['luxury_score'].apply(categorize_luxury)

train_df.sample(5)
```

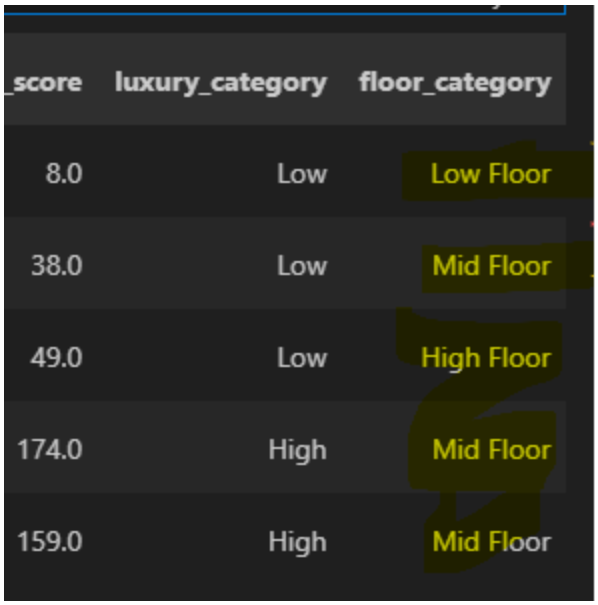
up_area	study room	servant room	store room	pooja room	others	furnishing_type	luxury_score	luxury_category
4950.0	1.0	1.0	1.0	1.0	0.0	1.0	126.0	Medium
2600.0	1.0	1.0	0.0	1.0	0.0	0.0	49.0	Low
2000.0	0.0	1.0	0.0	0.0	0.0	1.0	49.0	Low
2172.0	0.0	1.0	0.0	0.0	0.0	1.0	158.0	High
2700.0	1.0	1.0	1.0	1.0	0.0	0.0	131.0	Medium

floorNum

```
def categorize_floor(floor):
    if 0 <= floor <= 2:
        return "Low Floor"
    elif 3 <= floor <= 10:
        return "Mid Floor"
    elif 11 <= floor <= 51:
        return "High Floor"
    else:
        return None  # or "Undefined" or any other label for floors outside the def
ined bins
```

```
train_df['floor_category'] = train_df['floorNum'].apply(categorize_floor)

train_df.head()
```



_score	luxury_category	floor_category
8.0	Low	Low Floor
38.0	Low	Mid Floor
49.0	Low	High Floor
174.0	High	Mid Floor
159.0	High	Mid Floor

Drop `floorNum` & `luxury_score`

```
train_df.drop(columns=['floorNum','luxury_score'],inplace=True)
```

Convert CAT columns → NUM

`from sklearn.preprocessing import OrdinalEncoder`

What is does?

FLAT → 0

HOUSE → 1

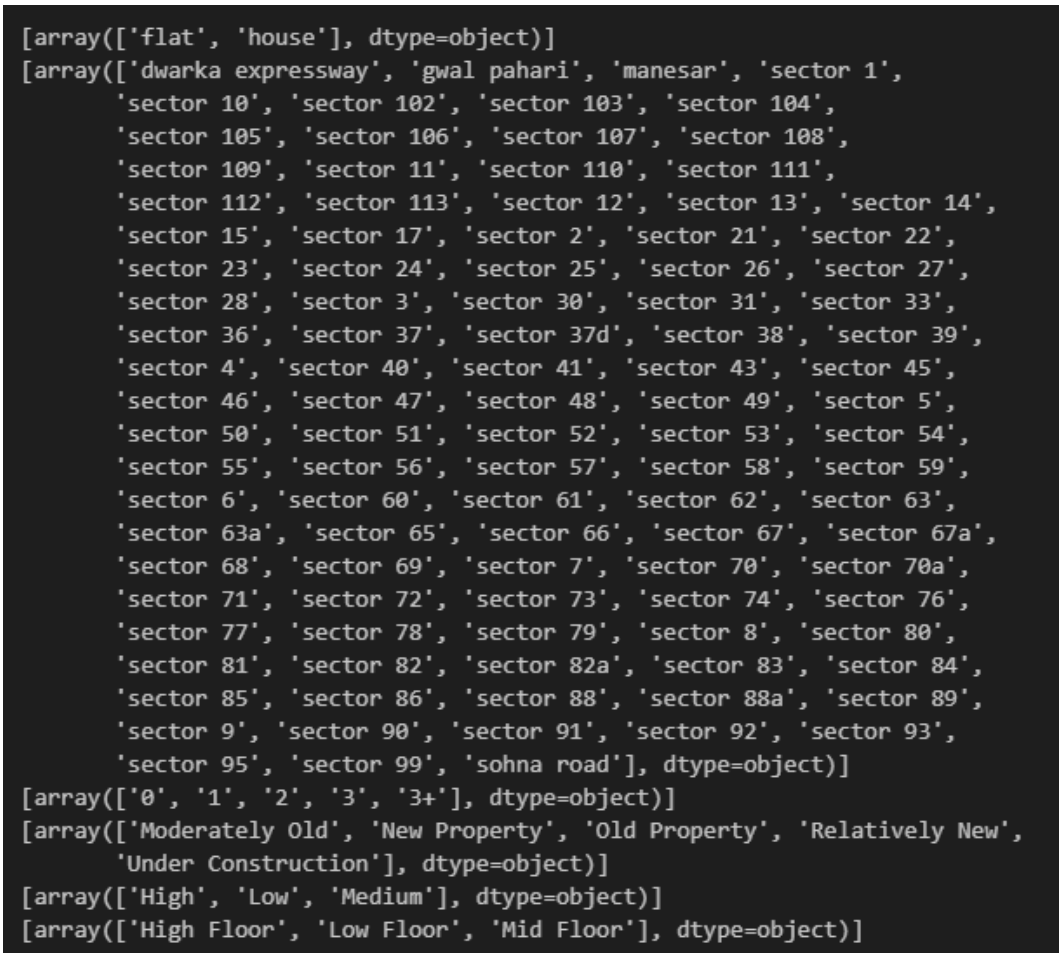
```
from sklearn.preprocessing import OrdinalEncoder

# Create a copy of the original data for label encoding
data_label_encoded = train_df.copy()

categorical_cols = train_df.select_dtypes(include=['object']).columns
```

```
# Apply label encoding to categorical columns
for col in categorical_cols:
    oe = OrdinalEncoder()
    data_label_encoded[col] = oe.fit_transform(data_label_encoded[[col]])
    print(oe.categories_)

# Splitting the dataset into training and testing sets
X_label = data_label_encoded.drop('price', axis=1)
y_label = data_label_encoded['price']
```



```
[array(['flat', 'house'], dtype=object)]
[array(['dwarka expressway', 'gwal pahari', 'manesar', 'sector 1',
       'sector 10', 'sector 102', 'sector 103', 'sector 104',
       'sector 105', 'sector 106', 'sector 107', 'sector 108',
       'sector 109', 'sector 11', 'sector 110', 'sector 111',
       'sector 112', 'sector 113', 'sector 12', 'sector 13', 'sector 14',
       'sector 15', 'sector 17', 'sector 2', 'sector 21', 'sector 22',
       'sector 23', 'sector 24', 'sector 25', 'sector 26', 'sector 27',
       'sector 28', 'sector 3', 'sector 30', 'sector 31', 'sector 33',
       'sector 36', 'sector 37', 'sector 37d', 'sector 38', 'sector 39',
       'sector 4', 'sector 40', 'sector 41', 'sector 43', 'sector 45',
       'sector 46', 'sector 47', 'sector 48', 'sector 49', 'sector 5',
       'sector 50', 'sector 51', 'sector 52', 'sector 53', 'sector 54',
       'sector 55', 'sector 56', 'sector 57', 'sector 58', 'sector 59',
       'sector 6', 'sector 60', 'sector 61', 'sector 62', 'sector 63',
       'sector 63a', 'sector 65', 'sector 66', 'sector 67', 'sector 67a',
       'sector 68', 'sector 69', 'sector 7', 'sector 70', 'sector 70a',
       'sector 71', 'sector 72', 'sector 73', 'sector 74', 'sector 76',
       'sector 77', 'sector 78', 'sector 79', 'sector 8', 'sector 80',
       'sector 81', 'sector 82', 'sector 82a', 'sector 83', 'sector 84',
       'sector 85', 'sector 86', 'sector 88', 'sector 88a', 'sector 89',
       'sector 9', 'sector 90', 'sector 91', 'sector 92', 'sector 93',
       'sector 95', 'sector 99', 'sohna road'], dtype=object)]
[array(['0', '1', '2', '3', '3+'], dtype=object)]
[array(['Moderately Old', 'New Property', 'Old Property', 'Relatively New',
       'Under Construction'], dtype=object)]
[array(['High', 'Low', 'Medium'], dtype=object)]
[array(['High Floor', 'Low Floor', 'Mid Floor'], dtype=object)]
```

```
train_df.select_dtypes(include=['object']).columns
```

```
Index(['property_type', 'sector', 'balcony', 'agePossession',
       'luxury_category', 'floor_category'],
      dtype='object')
```

X_label

	property_type	sector	bedRoom	bathroom	balcony	agePossession	built_up_area	study room	servant room	store room	pooja room	others	furnishing_type	luxury_category	floor_category
0	0.0	36.0	3.0	2.0	2.0	1.0	850.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0
1	0.0	95.0	2.0	2.0	2.0	1.0	1226.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	2.0
2	0.0	103.0	2.0	2.0	1.0	1.0	1000.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
3	0.0	99.0	3.0	4.0	4.0	3.0	1615.0	0.0	1.0	0.0	0.0	1.0	1.0	0.0	2.0
4	0.0	5.0	2.0	2.0	1.0	3.0	582.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	2.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3549	0.0	90.0	2.0	2.0	1.0	3.0	532.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0
3550	1.0	12.0	5.0	5.0	4.0	3.0	6228.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
3551	0.0	23.0	1.0	1.0	1.0	0.0	665.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	2.0
3552	1.0	44.0	5.0	6.0	3.0	0.0	5490.0	1.0	1.0	1.0	1.0	0.0	0.0	2.0	2.0
3553	0.0	71.0	3.0	3.0	4.0	3.0	1845.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	0.0

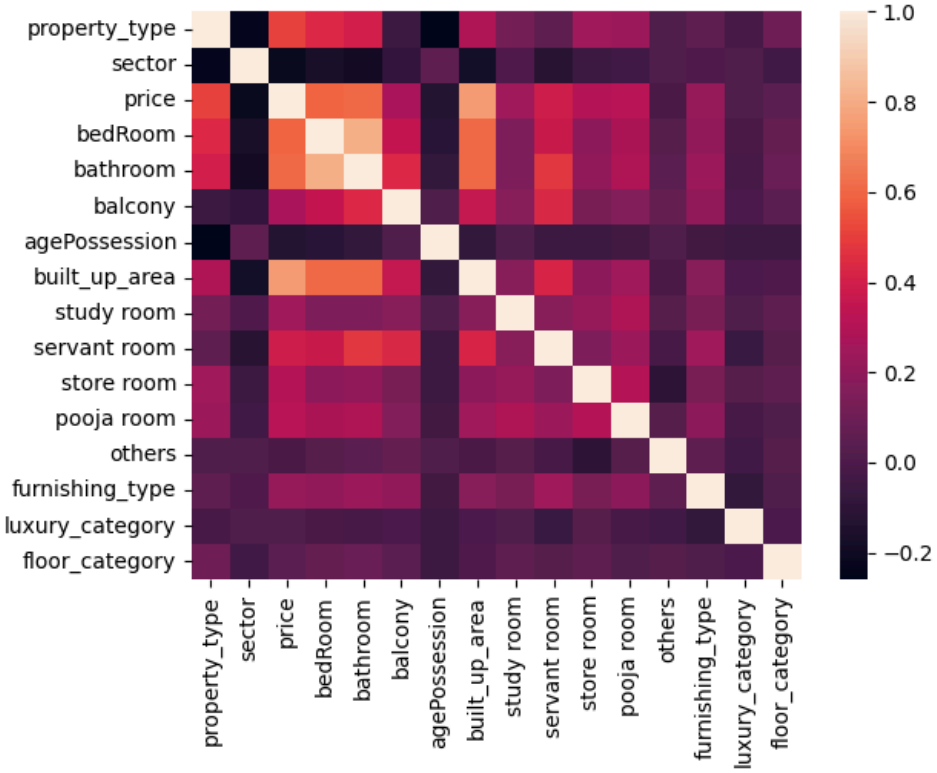
y_label

```
0         0.82
1         0.95
2         0.32
3         1.60
4         0.48
         ...
3549      0.37
3550      6.00
3551      0.60
3552     15.50
3553      1.78
Name: price, Length: 3554, dtype: float64
```

- We will apply 8 techniques
- In each technique, we'll get a feature score
- At the end, the features with highest score will be considered to be most important.

Technique 1: Correlation Analysis

```
sns.heatmap(data_label_encoded.corr())
```




```
fi_df1 = data_label_encoded.corr()['price'].iloc[1:].to_frame().reset_index().ren
ame(columns={'index':'feature','price':'corr_coeff'})
fi_df1
```

	feature	corr_coeff
0	sector	-0.212084
1	price	1.000000
2	bedRoom	0.591289
3	bathroom	0.609777
4	balcony	0.269637
5	agePossession	-0.134171
6	built_up_area	0.748574
7	study room	0.242955
8	servant room	0.391930
9	store room	0.305677
10	pooja room	0.319852
11	others	-0.013064
12	furnishing_type	0.225625
13	luxury_category	0.009788
14	floor_category	0.042745

Technique 2 - Random Forest Feature Importance

- You can get feature importance with Random Forest

```
from sklearn.ensemble import RandomForestRegressor

# Train a Random Forest regressor on label encoded data
rf_label = RandomForestRegressor(n_estimators=100, random_state=42)
rf_label.fit(X_label, y_label)

# Extract feature importance scores for label encoded data
fi_df2 = pd.DataFrame({
    'feature': X_label.columns,
    'rf_importance': rf_label.feature_importances_
```

```
}).sort_values(by='rf_importance', ascending=False)

fi_df2
```

	feature	rf_importance
6	built_up_area	0.650541
1	sector	0.102670
0	property_type	0.100079
3	bathroom	0.025882
2	bedRoom	0.024041
8	servant room	0.019320
5	agePossession	0.014519
4	balcony	0.012576
12	furnishing_type	0.010604
7	study room	0.008383
9	store room	0.008329
13	luxury_category	0.007605
14	floor_category	0.006552
10	pooja room	0.006126
11	others	0.002772

Technique 3 - Gradient Boosting Feature importances

```
from sklearn.ensemble import GradientBoostingRegressor

# Train a Random Forest regressor on label encoded data
gb_label = GradientBoostingRegressor()
gb_label.fit(X_label, y_label)

# Extract feature importance scores for label encoded data
```

```
fi_df3 = pd.DataFrame({
    'feature': X_label.columns,
    'gb_importance': gb_label.feature_importances_
}).sort_values(by='gb_importance', ascending=False)

fi_df3
```

	feature	gb_importance
6	built_up_area	0.677775
1	sector	0.102810
0	property_type	0.098374
2	bedRoom	0.038355
3	bathroom	0.035746
8	servant room	0.023195
9	store room	0.010356
5	agePossession	0.004421
7	study room	0.003087
12	furnishing_type	0.002727
4	balcony	0.001899
14	floor_category	0.000421
10	pooja room	0.000357
13	luxury_category	0.000265
11	others	0.000211

Technique 4 - Permutation Importance

- In this method, you jumble 1 column keeping others constant.
- Train the second model on this data & calculate the score
- If the score reduces → The feature is important
- Repeat this with all the columns 1 by 1

```
from sklearn.inspection import permutation_importance
from sklearn.model_selection import train_test_split

X_train_label, X_test_label, y_train_label, y_test_label = train_test_split(X_label, y_label, test_size=0.2, random_state=42)

# Train a Random Forest regressor on label encoded data
rf_label = RandomForestRegressor(n_estimators=100, random_state=42)
rf_label.fit(X_train_label, y_train_label)

# Calculate Permutation Importance
perm_importance = permutation_importance(rf_label, X_test_label, y_test_label, n_repeats=30, random_state=42)

# Organize results into a DataFrame
fi_df4 = pd.DataFrame({
    'feature': X_label.columns,
    'permutation_importance': perm_importance.importances_mean
}).sort_values(by='permutation_importance', ascending=False)

fi_df4
```

	feature	permutation_importance
6	built_up_area	0.736026
0	property_type	0.200636
1	sector	0.178881
8	servant room	0.021018
3	bathroom	0.019884
2	bedRoom	0.018006
5	agePossession	0.004970
9	store room	0.002150
14	floor_category	0.000907
11	others	0.000792
13	luxury_category	0.000525
4	balcony	-0.000432
10	pooja room	-0.001078
12	furnishing_type	-0.010475
7	study room	-0.018027

- **Negative values signify that when you jumbled the columns, the score increased.**

Technique 5 - LASSO

```
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler #Scaling is needed for line
ar models

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_label)

# Train a LASSO regression model
# We'll use a relatively small value for alpha (the regularization strength) for d
```

```
emonstration purposes
lasso = Lasso(alpha=0.01, random_state=42)
lasso.fit(X_scaled, y_label)

# Extract coefficients
fi_df5 = pd.DataFrame({
    'feature': X_label.columns,
    'lasso_coeff': lasso.coef_
}).sort_values(by='lasso_coeff', ascending=False)

fi_df5
```

	feature	lasso_coeff
6	built_up_area	1.510173
0	property_type	0.713829
3	bathroom	0.275042
9	store room	0.199655
7	study room	0.171846
12	furnishing_type	0.164113
8	servant room	0.160601
10	pooja room	0.073845
13	luxury_category	0.055268
2	bedRoom	0.014170
5	agePossession	-0.000000
14	floor_category	-0.002610
11	others	-0.017163
4	balcony	-0.043562
1	sector	-0.069634

Technique 6 - RFE

- ✨**RFE is a reliable feature selection technique.**

```
from sklearn.feature_selection import RFE

# Initialize the base estimator
estimator = RandomForestRegressor()

# Apply RFE on the label-encoded and standardized training data
selector_label = RFE(estimator, n_features_to_select=X_label.shape[1], step=1)
selector_label = selector_label.fit(X_label, y_label)

# Get the selected features based on RFE
selected_features = X_label.columns[selector_label.support_]

# Extract the coefficients for the selected features from the underlying linear r
egression model
selected_coefficients = selector_label.estimator_.feature_importances_

# Organize the results into a DataFrame
fi_df6 = pd.DataFrame({
    'feature': selected_features,
    'rfe_score': selected_coefficients
}).sort_values(by='rfe_score', ascending=False)

fi_df6
```

	feature	rfe_score
6	built_up_area	0.650789
1	sector	0.103429
0	property_type	0.101777
2	bedRoom	0.025895
3	bathroom	0.022667
8	servant room	0.021817
5	agePossession	0.013739
4	balcony	0.012081
12	furnishing_type	0.010056
13	luxury_category	0.007939
9	store room	0.006924
7	study room	0.006798
10	pooja room	0.006685
14	floor_category	0.006067
11	others	0.003338

- `step=1` → one feature is removed each time

- **`n_features_to_select`**: The number of features to select. Here, it is set to the total number of features in `X_label` (`X_label.shape[1]`), meaning no features will be eliminated.
- **`selector_label.support_`**:
  - A boolean array indicating which features were selected (`True` for selected features, `False` for eliminated features).
  - **`support_`** →Boolean mask indicating selected features (e.g., `[True, False, True]`).
- **`X_label.columns[selector_label.support_]`**:
  - Extracts the names of the selected features from the original feature set (`X_label`).

- `selector_label.estimator_`:
  - The underlying Random Forest model trained during the RFE process.
- `feature_importances_`:
  - An array of feature importance scores provided by the Random Forest model.

Technique 7 - Linear Regression Weights

```
# Train a linear regression model on the label-encoded and standardized traini
ng data
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(X_scaled, y_label)

# Extract coefficients
fi_df7 = pd.DataFrame({
    'feature': X_label.columns,
    'reg_coeffs': lin_reg.coef_
}).sort_values(by='reg_coeffs', ascending=False)

fi_df7
```

	feature	reg_coeffs
6	built_up_area	1.512629
0	property_type	0.712890
3	bathroom	0.281976
9	store room	0.204159
7	study room	0.180048
12	furnishing_type	0.173192
8	servant room	0.169605
10	pooja room	0.076893
13	luxury_category	0.066472
2	bedRoom	0.016790
5	agePossession	-0.002041
14	floor_category	-0.013482
11	others	-0.025155
4	balcony	-0.066353
1	sector	-0.078657

- `lin_reg.coef_`:
  - The coefficients of the linear regression model. Each coefficient represents the change in the target variable (`y_label`) for a one-unit change in the corresponding feature, assuming all other features are held constant.

Merge all the dataframes

```
final_fi_df = fi_df1.merge(fi_df2,on='feature').merge(fi_df3,on='feature').merge(fi_df4,on='feature').merge(fi_df5,on='feature').merge(fi_df6,on='feature').merge(fi_df7,on='feature').merge(fi_df8,on='feature').set_index('feature')
```

```
final_fi_df
```

feature	corr_coeff	rf_importance	gb_importance	permutation_importance	lasso_coeff	rfe_score	reg_coeffs	SHAP_score
sector	-0.212084	0.102670	0.102810	0.178881	-0.069634	0.103429	-0.078657	0.383640
bedRoom	0.591289	0.024041	0.038355	0.018006	0.014170	0.025895	0.016790	0.049731
bathroom	0.609777	0.025882	0.035746	0.019884	0.275042	0.022667	0.281976	0.112811
balcony	0.269637	0.012576	0.001899	-0.000432	-0.043562	0.012081	-0.066353	0.040327
agePossession	-0.134171	0.014519	0.004421	0.004970	-0.000000	0.013739	-0.002041	0.027312
built_up_area	0.748574	0.650541	0.677775	0.736026	1.510173	0.650789	1.512629	1.255552
study room	0.242955	0.008383	0.003087	-0.018027	0.171846	0.006798	0.180048	0.019533
servant room	0.391930	0.019320	0.023195	0.021018	0.160601	0.021817	0.169605	0.095577
store room	0.305677	0.008329	0.010356	0.002150	0.199655	0.006924	0.204159	0.017123
pooja room	0.319852	0.006126	0.000357	-0.001078	0.073845	0.006685	0.076893	0.011505
others	-0.013064	0.002772	0.000211	0.000792	-0.017163	0.003338	-0.025155	0.007257
furnishing_type	0.225625	0.010604	0.002727	-0.010475	0.164113	0.010056	0.173192	0.027405
luxury_category	0.009788	0.007605	0.000265	0.000525	0.055268	0.007939	0.066472	0.015824
floor_category	0.042745	0.006552	0.000421	0.000907	-0.002610	0.006067	-0.013482	0.024505

- We have to normalize this

```
# normalize the score
final_fi_df = final_fi_df.divide(final_fi_df.sum(axis=0), axis=1)

final_fi_df
```

feature	corr_coeff	rf_importance	gb_importance	permutation_importance	lasso_coeff	rfe_score	reg_coeffs	SHAP_score
sector	-0.062405	0.114088	0.114028	0.187674	-0.027946	0.115148	-0.031512	0.183727
bedRoom	0.173984	0.026715	0.042540	0.018891	0.005687	0.028829	0.006727	0.023817
bathroom	0.179424	0.028760	0.039646	0.020861	0.110381	0.025236	0.112968	0.054026
balcony	0.079339	0.013975	0.002106	-0.000453	-0.017483	0.013449	-0.026583	0.019313
agePossession	-0.039479	0.016133	0.004904	0.005215	-0.000000	0.015296	-0.000818	0.013080
built_up_area	0.220264	0.722887	0.751725	0.772206	0.606071	0.724529	0.606003	0.601289
study room	0.071488	0.009315	0.003424	-0.018914	0.068966	0.007568	0.072132	0.009354
servant room	0.115324	0.021468	0.025726	0.022052	0.064453	0.024289	0.067949	0.045772
store room	0.089944	0.009255	0.011486	0.002255	0.080126	0.007709	0.081792	0.008200
pooja room	0.094115	0.006808	0.000396	-0.001131	0.029636	0.007442	0.030806	0.005510
others	-0.003844	0.003080	0.000234	0.000831	-0.006888	0.003716	-0.010078	0.003475
furnishing_type	0.066389	0.011784	0.003025	-0.010989	0.065863	0.011195	0.069386	0.013124
luxury_category	0.002880	0.008451	0.000294	0.000551	0.022181	0.008838	0.026631	0.007578
floor_category	0.012578	0.007281	0.000467	0.000952	-0.001048	0.006755	-0.005401	0.011736

```
final_fi_df[['rf_importance','gb_importance','permutation_importance','rfe_score','SHAP_score']].mean(axis=1).sort_values(ascending=False)
```

```
feature
built_up_area       0.714527
sector              0.142933
bathroom            0.033706
bedRoom             0.028158
servant room        0.027861
agePossession       0.010926
balcony             0.009678
store room          0.007781
furnishing_type     0.005628
floor_category      0.005438
luxury_category     0.005142
pooja room          0.003805
others              0.002267
study room          0.002150
dtype: float64
```

`axis=1`: Calculates the mean **row-wise**

Drop pooja room, study room, others??

- We will calculate R2 score with and without these features & compare them

```
# with all the cols
from sklearn.model_selection import cross_val_score

rf = RandomForestRegressor(n_estimators=100, random_state=42)

scores = cross_val_score(rf, X_label, y_label, cv=5, scoring='r2')
```

```
scores.mean()

Output: 0.8193190510339333
```

- ☝️Score with these 3 columns

- Now let’s drop these columns & calculate R2 score

```
rf = RandomForestRegressor(n_estimators=100, random_state=42)

scores = cross_val_score(rf, X_label.drop(columns=['pooja room', 'study roo
m', 'others']), y_label, cv=5, scoring='r2')
```

```
scores.mean()

Output: 0.8196500940616491
```

- **The R2 score is same in both the cases.**

- Therefore, we can drop these columns

```
export_df = X_label.drop(columns=['pooja room', 'study room', 'others'])
export_df['price'] = y_label
```

Export this data as CSV file:

```
export_df.to_csv('gurgaon_properties_post_feature_selection.csv', index=Fals
e)
```