Capstone Project (Baseline Model)

- We will apply linear regression to our data
- We will **apply One-Hot-Encoding** on the categorical columns
- Sector column has 104 categories.
    - It will create 104 columns
- **One Hot Encoding is necessary for linear models.**
- After that, we need **scaling**
- Apply **log transformation on `price` column** to make it normally distributed.

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

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

- Import all the libraries:

```
from sklearn.model_selection import KFold, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.svm import SVR
```

CAT columns: 👇

```
columns_to_encode = ['sector', 'balcony', 'agePossession', 'furnishing_type', 'luxury_category', 'floor_category']
```

Log transformation on `y` :

```
# Applying the log1p transformation to the target variable
y_transformed = np.log1p(y)
```

ColumnTransformer

`from sklearn.compose import ColumnTransformer`

- It applies different transformations on columns at the same time.

```
transformer = ColumnTransformer(transformers=[
    ('tnf1',SimpleImputer(),['fever']),
    ('tnf2',OrdinalEncoder(categories=[['Mild','Strong']]),['cough']),
    ('tnf3',OneHotEncoder(sparse=False,drop='first'),['gender','city'])
],remainder='passthrough')
```

```
transformer.fit_transform(X_train)
```

`remainder='passthrough'` keeps other columns as it is.

`remainder='drop'` will drop the columns

```
# Creating a column transformer for preprocessing
preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), ['property_type', 'bedRoom', 'bathroom', 'built_u
```

```
p_area', 'servant room', 'store room']),
        ('cat', OneHotEncoder(drop='first'), columns_to_encode)
    ],
    remainder='passthrough'
)
```

- `drop='first'` tells the `OneHotEncoder` to drop the first category in each feature.
- This is a common practice to prevent issues with multicollinearity in machine learning models

Create a Pipeline

```
# Creating a pipeline
pipeline = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', SVR(kernel='rbf'))
])
```

1. Transformation
2. Linear Apply model

Apply k-Fold Cross-Validation

1. Split the data into k equal parts (folds).
2. Train the model on $k-1$ folds and validate on the remaining fold.
3. Repeat this process k times, each time using a different fold as the validation set.
4. Average the performance across all k folds.

```
# K-fold cross-validation
kfold = KFold(n_splits=10, shuffle=True, random_state=42)
scores = cross_val_score(pipeline, X, y_transformed, cv=kfold, scoring='r2')
```

```
scores.mean()

Output: 0.8845360715052786
```

Calculate mean_absolute_error

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X,y_transformed,test_size=0.2,random_state=42)

pipeline.fit(X_train,y_train)

y_pred = pipeline.predict(X_test)
```

- Convert the log values to normal ones

```
y_pred = np.expm1(y_pred)
```

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(np.expm1(y_test),y_pred)

Output: 0.5324591082613233
```

- Meaning → Our model is making mistake of 53.24 Lac while predicting the price