HOUSE PRICE PREDICTION APP

Purba Chowdhury

Date: 19-08-2024

Abstract:

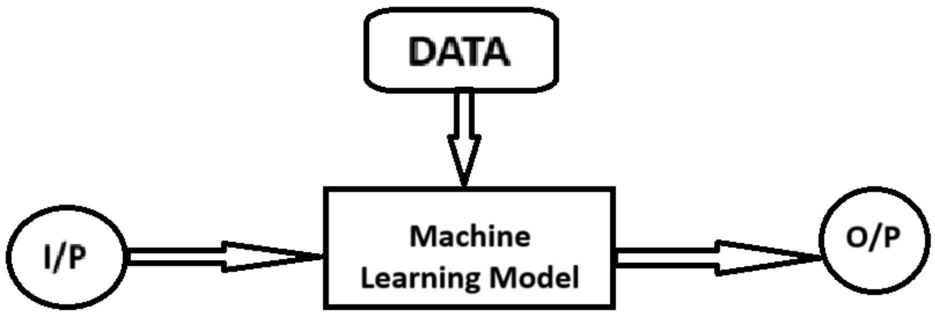
The objective of this project is to develop a house price prediction app using machine learning. So that users can obtain the house price directly for a particular location, bypassing the need for a broker. The app will utilize various parameters such as location, square feet, no of bedrooms, no of bathrooms, no of parking slot, distance from nearest railway station, distance from market, distance from highway, market conditions etc. to predict house prices accurately.

Sometimes brokers mislead users, resulting in them overpaying for houses, this app helps users avoid such pitfalls by providing accurate price estimates. This app will help users, buyers, sellers, and real estate professionals in making informed decisions.

1.Problem Statement

Target is to predict house price using an app (House Price Prediction App) using a machine learning model. In this app various features will be there and various user inputs will be there. If they put the input like location, square feet, no of bedrooms, no of bathrooms, no of parking slot, distance from nearest railway station, distance from market, distance from highway they will get to know the output which is price of that house here.

In recent years, the real estate market has experienced significant fluctuations due to a variety of economic, social, and environmental factors. Understanding this price factor is very crucial for user, sellers, buyers and also for broker. Tradition method of estimating house price often fail to provide accurate price to the user, due to various factor, like less knowledge, limited no of past data. With data science and machine learning, there
is an opportunity to create a more dynamic tools to predict housing price.



The purpose of this project is to develop a user-friendly Housing Price App. By integrating a wide range of data sources, including historical sales data, economic indicators, neighbourhood features, and more, the app aims to offer valuable insights to users looking to buy, sell, or invest in real estate.

- **Accurate Predictions:** Develop a machine learning model that provides highly accurate predictions of housing prices based on a comprehensive set of variables.
- **User Accessibility:** Design an easy-to-use interface that allows users of varying technical backgrounds to interact with the app effortlessly.
- **Real-Time Updates:** Ensure the app can update its predictions in real-time as new data becomes available, maintaining relevance in a fast-changing market.
- **Insightful Analytics:** Offer detailed analytics and visualizations to help users understand the factors influencing housing prices and make informed decisions.
- **Scalability:** Build a scalable system that can handle large volumes of data and a growing user base without compromising performance.

1.1 Initial Needs Statement

We need some previous sales data to build the model. Desire detailed insights into neighbourhood trends and factors affecting local housing prices. Require advanced market analysis to identify future market trends. Require up-to-date market trends information.

2.0 Customer Needs Assessment

In housing price prediction app generally our customers are homebuyers, real estate agents, investors, sellers. They want an accurate price prediction app which is reliable and up-to-date with respect to market. The app should be user-friendly interface and easy-to-navigate. Ability for users to input specific details and preferences to get personalized predictions. Customization Options are very important aspect. Real-time data updates to ensure predictions reflect the latest market conditions. Most importantly buyers want to avoid the broker charge to get information about various house price in various locality.

3.0 Target Specifications

Here target is to predict a particular house price accurately, so that users can get the proper information about a house in a particular location. This app will allow users to compare similar properties. This app will have various parameters as input field, so that users can put their input and get the prices of those houses as an output. In a very short time, they can compare similar type of house price and can make their decision.

4.0 Applicable Constraints

Data Accuracy and Quality: Ensure that all input data, such as historical housing prices, interest rates, and economic indicators, are accurate and up-to-date.

Scalability: The app should be able to handle an increasing number of users and large datasets efficiently.

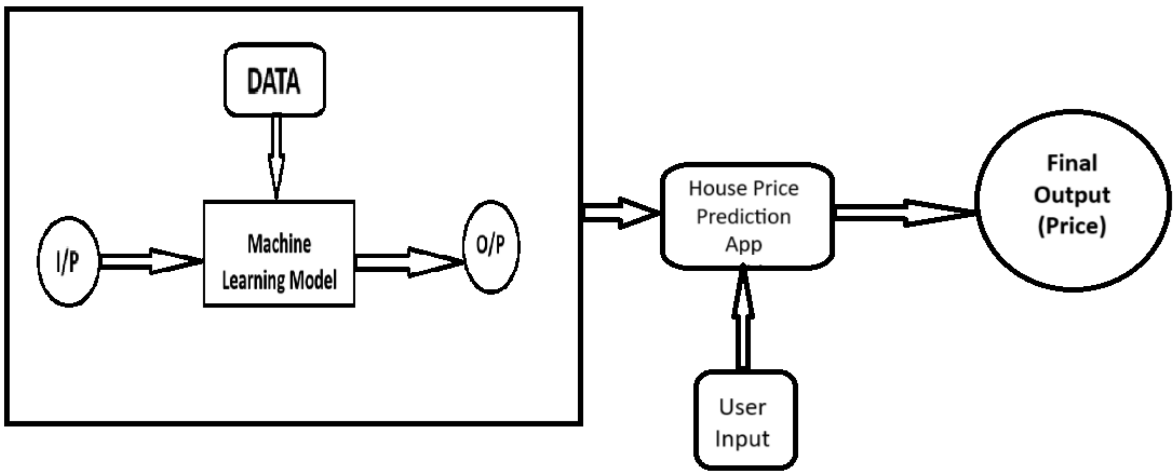
4.1 Business Opportunity

Developing a project housing price app presents a significant business opportunity. This app will provide reliable predictions on housing prices based on the machine learning model which analyses historical data, economic indicators, and market trends. Our target market is homebuyers, real estate investors, financial institutions, policy makers etc. Following revenue stream will be there.

- **Subscription Model**: There will be subscription plan. Subscribed users will get advanced analytics, personalized reports, and exclusive market insights. Using this subscription model uses can access required contact details.
- **Free Model**: Free model will provide basic features and the photos of the houses.
- **Data Sales**: We will sale aggregated and anonymized data to real estate firms, financial institutions, and market researchers.
- **Advertisements**: Generate revenue through targeted ads from real estate agencies and home improvement businesses.

5.0 Concept Generation

Here we will to develop an app so that users can predict accurate prices of house. This app will enable users to input their parameters to receive the proper result. There will be several input parameters where users can put their values, like locality, no of bedrooms, no of bathrooms, area of the house, no of parking slot, distance form nearest railway station, distance form nearest highway, distance form nearest market etc. We will collect enough past housing data to build the machine learning model. We will build the model and develop an app. Users will use this app.



Concept Of The Project

6.0 Final Product Design and Algorithm:

To develop the house price prediction app, we will need past house pricing data. First, we have to collect the data and need to do market analysis. After collecting the data, we will clean the data using several preprocessing techniques. After preprocessing the data, we fit it into our chosen machine learning model to predict house prices. Here we will use Regression model, suitable for predicting continuous values. We will implement following preprocessing techniques:

1. Imputation of missing values: If there is very less no of cells, having missing values we can drop the row. Otherwise, we will impute the missing values. There is various imputation technique, like mean, mode, median, most frequent and constant. If the data is numerical data we can use

2. Feature Scaling: Feature scaling transform all the features feature values such that all the features are on the same scale. We will apply following techniques, like Standardization, Normalization, MaxAbsScaler.

3. Handling Categorical Values: We need to convert categorical features into numerical features. We will apply following techniques, like Ordinal Encoding, One-Hot Encoding, Label Encoder.

Here We will use Column Transformer and Pipeline technique if needed.

After preprocessing we will proceed to fit a regression model to the training dataset. Here is the detail steps involved:

1. Data Splitting: The dataset will be divided into training and testing sets. 80% of the data will be used for training, and 20% will be reserved for testing.

2. Model Selection: We will select a regression algorithm suitable for predicting continuous values. Common choices are:

- Linear Regression
- Random Forest Regressor

We will select Random Forest Regressor due to its robustness and ability to handle non-linearity.

3. Model Training: We will train the model using training data.

4. Hyperparameter Tuning: To enhance the model's performance, we will perform hyperparameter tuning using Grid Search.

5. Model Evaluation: After training, we will evaluate the model's performance on the testing set using appropriate metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2).

7.0 Code of The Machine Learning Model

Meta data of the dataset:

Date and Year: User search date.

Locality: In which locality a house is located.

Estimated Value: Estimated value of a house.

Sale Price: Salling price of a house.

Property: How many families will be fit in a house.

Residential: Description of the property, like Detached House, Duplex etc.

num_rooms: Number of rooms in a house.

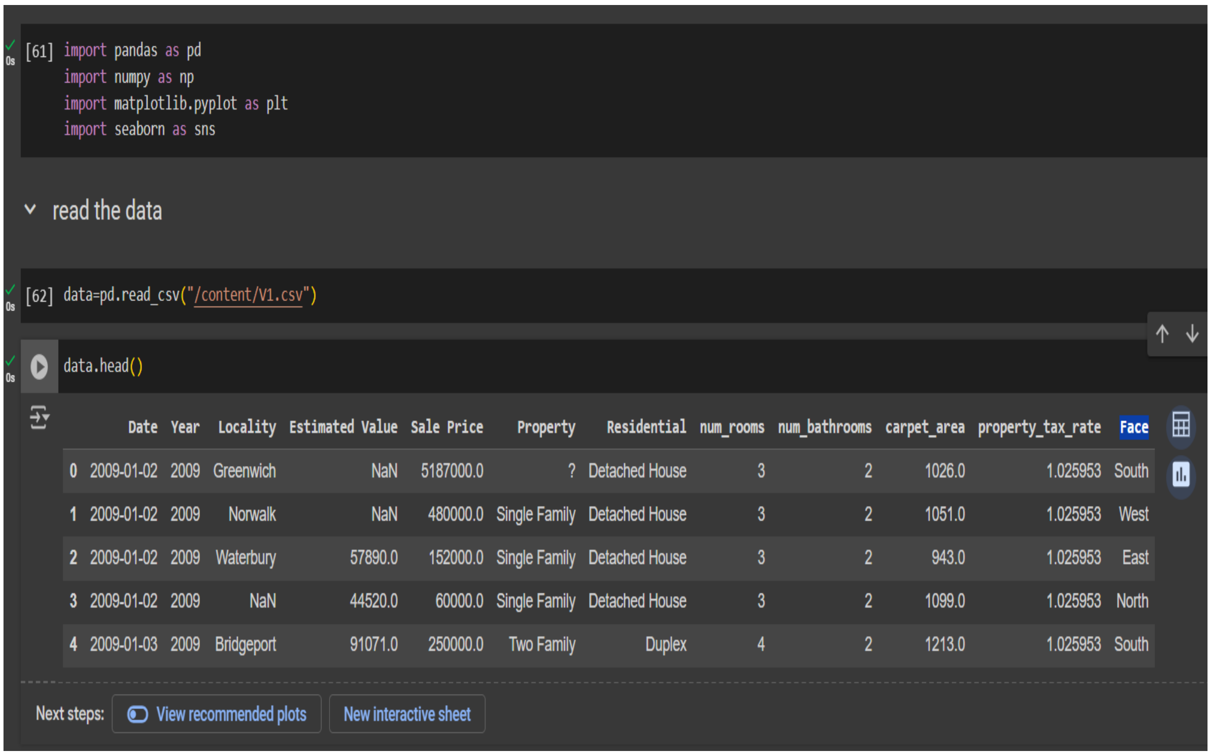
Num bathrooms: Number of bathrooms in a house.

carpet_area: area of a house.

property_tax_rate: tax rate of a house or property.

Face: Facing of a house.

Preprocessing and model fitting of a house price dataset.



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

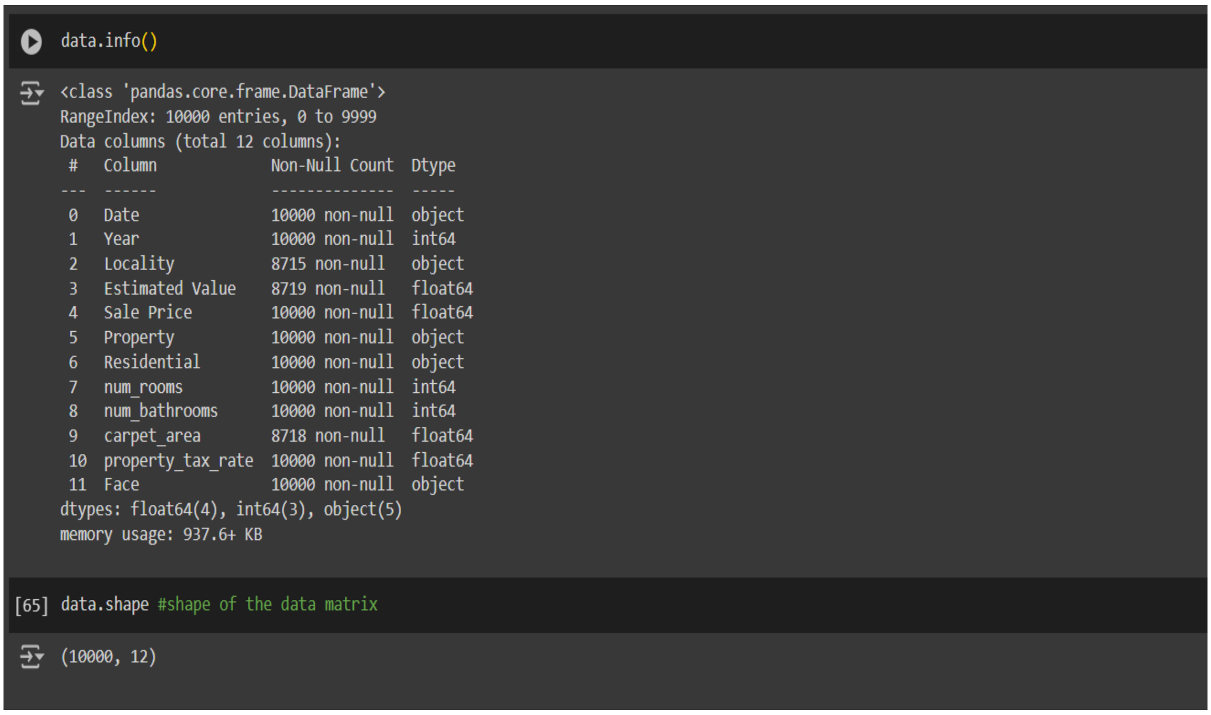
read the data

```
[62] data=pd.read_csv("/content/V1.csv")
```

```
data.head()
```

	Date	Year	Locality	Estimated Value	Sale Price	Property	Residential	num_rooms	num_bathrooms	carpet_area	property_tax_rate	Face
0	2009-01-02	2009	Greenwich	NaN	5187000.0	?	Detached House	3	2	1026.0	1.025953	South
1	2009-01-02	2009	Norwalk	NaN	480000.0	Single Family	Detached House	3	2	1051.0	1.025953	West
2	2009-01-02	2009	Waterbury	57890.0	152000.0	Single Family	Detached House	3	2	943.0	1.025953	East
3	2009-01-02	2009	NaN	44520.0	60000.0	Single Family	Detached House	3	2	1099.0	1.025953	North
4	2009-01-03	2009	Bridgeport	91071.0	250000.0	Two Family	Duplex	4	2	1213.0	1.025953	South

Next steps: View recommended plots New interactive sheet

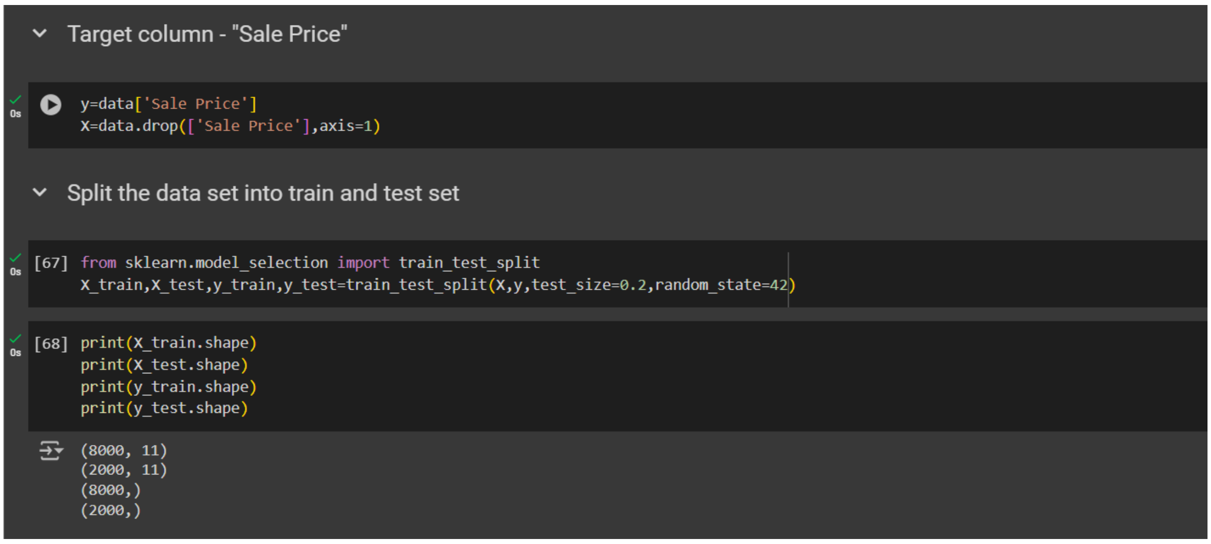


```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Date               10000 non-null  object
 1   Year               10000 non-null  int64
 2   Locality           8715 non-null   object
 3   Estimated Value    8719 non-null   float64
 4   Sale Price         10000 non-null  float64
 5   Property           10000 non-null  object
 6   Residential        10000 non-null  object
 7   num_rooms          10000 non-null  int64
 8   num_bathrooms      10000 non-null  int64
 9   carpet_area        8718 non-null   float64
 10  property_tax_rate  10000 non-null  float64
 11  Face               10000 non-null  object
dtypes: float64(4), int64(3), object(5)
memory usage: 937.6+ KB
```

```
[65] data.shape #shape of the data matrix
```

```
(10000, 12)
```



Target column - "Sale Price"

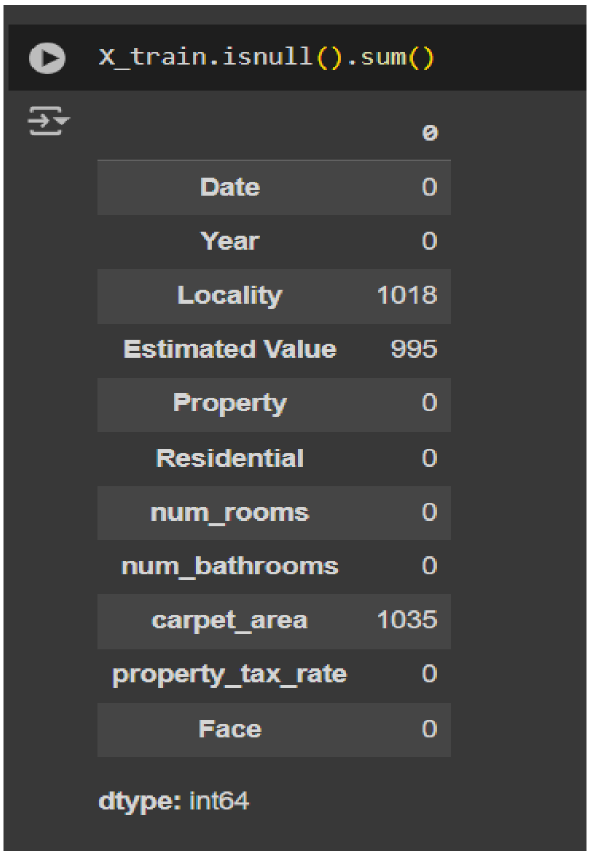
```
y=data['Sale Price']
X=data.drop(['Sale Price'],axis=1)
```

Split the data set into train and test set

```
[67] 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)
```

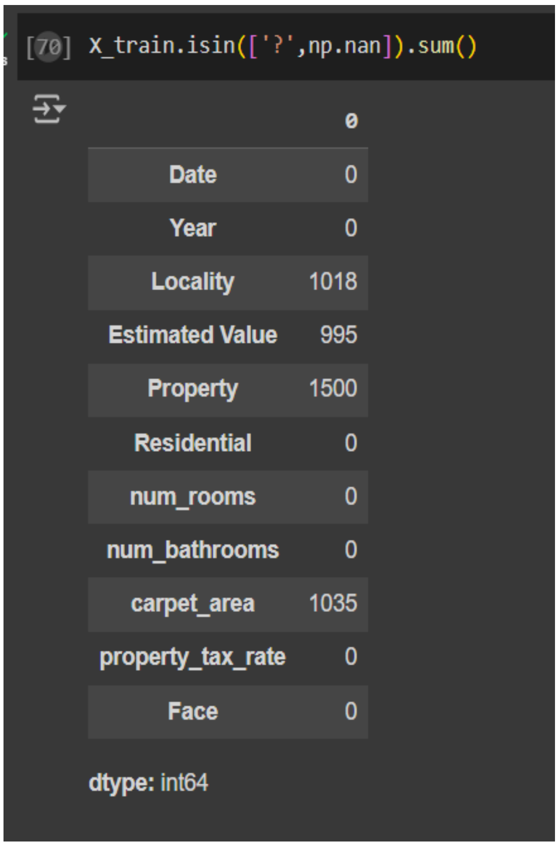
```
[68] print(X_train.shape)
     print(X_test.shape)
     print(y_train.shape)
     print(y_test.shape)
```

```
(8000, 11)
(2000, 11)
(8000,)
(2000,)
```



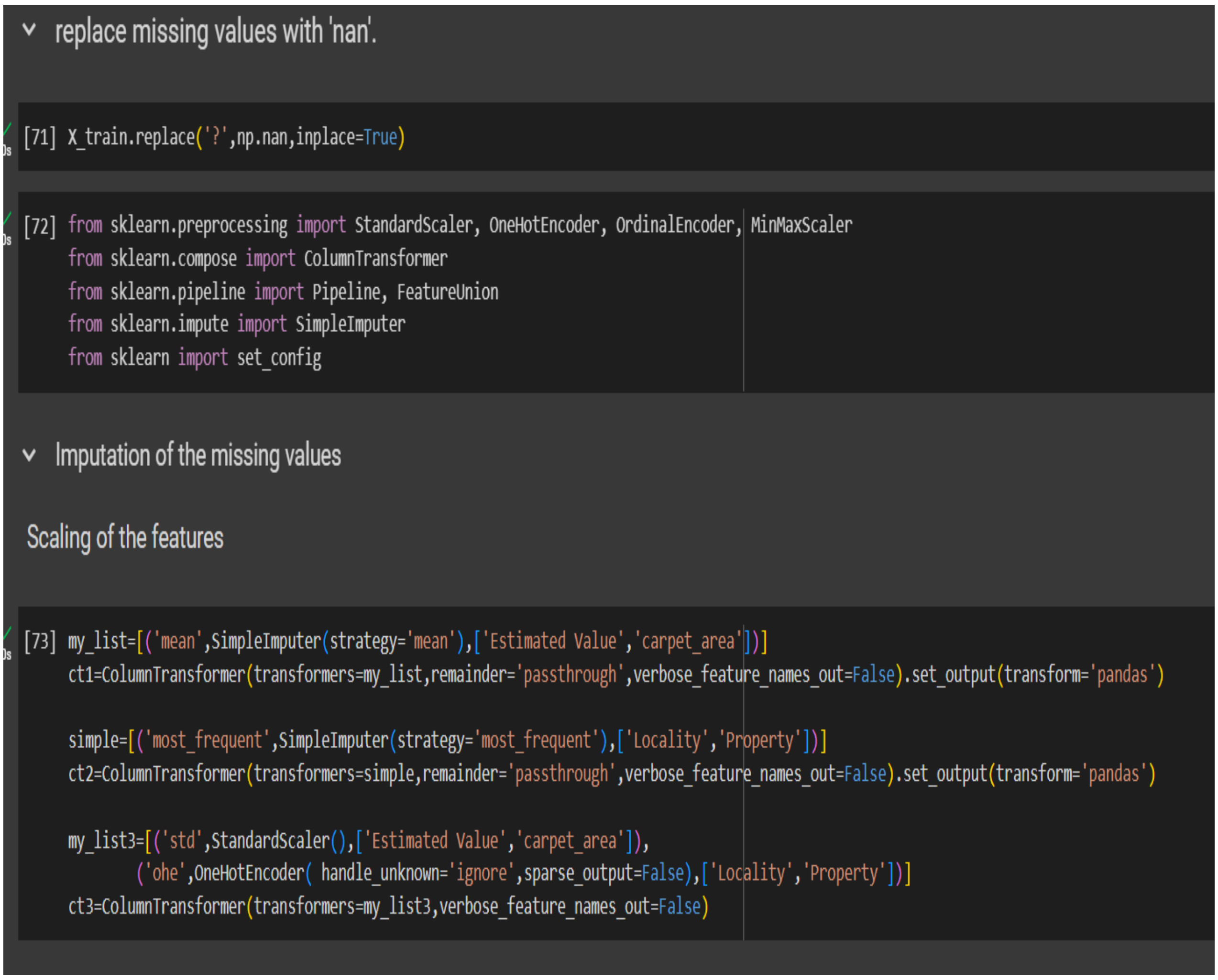
```
X_train.isnull().sum()
```

```
                      0
           Date       0
           Year       0
       Locality    1018
Estimated Value     995
       Property       0
    Residential       0
      num_rooms       0
  num_bathrooms       0
    carpet_area    1035
property_tax_rate     0
           Face       0

dtype: int64
```



```
[70] X_train.isin(['?',np.nan]).sum()
```

```
                      0
           Date       0
           Year       0
       Locality    1018
Estimated Value     995
       Property    1500
    Residential       0
      num_rooms       0
  num_bathrooms       0
    carpet_area    1035
property_tax_rate     0
           Face       0

dtype: int64
```


replace missing values with 'nan'.

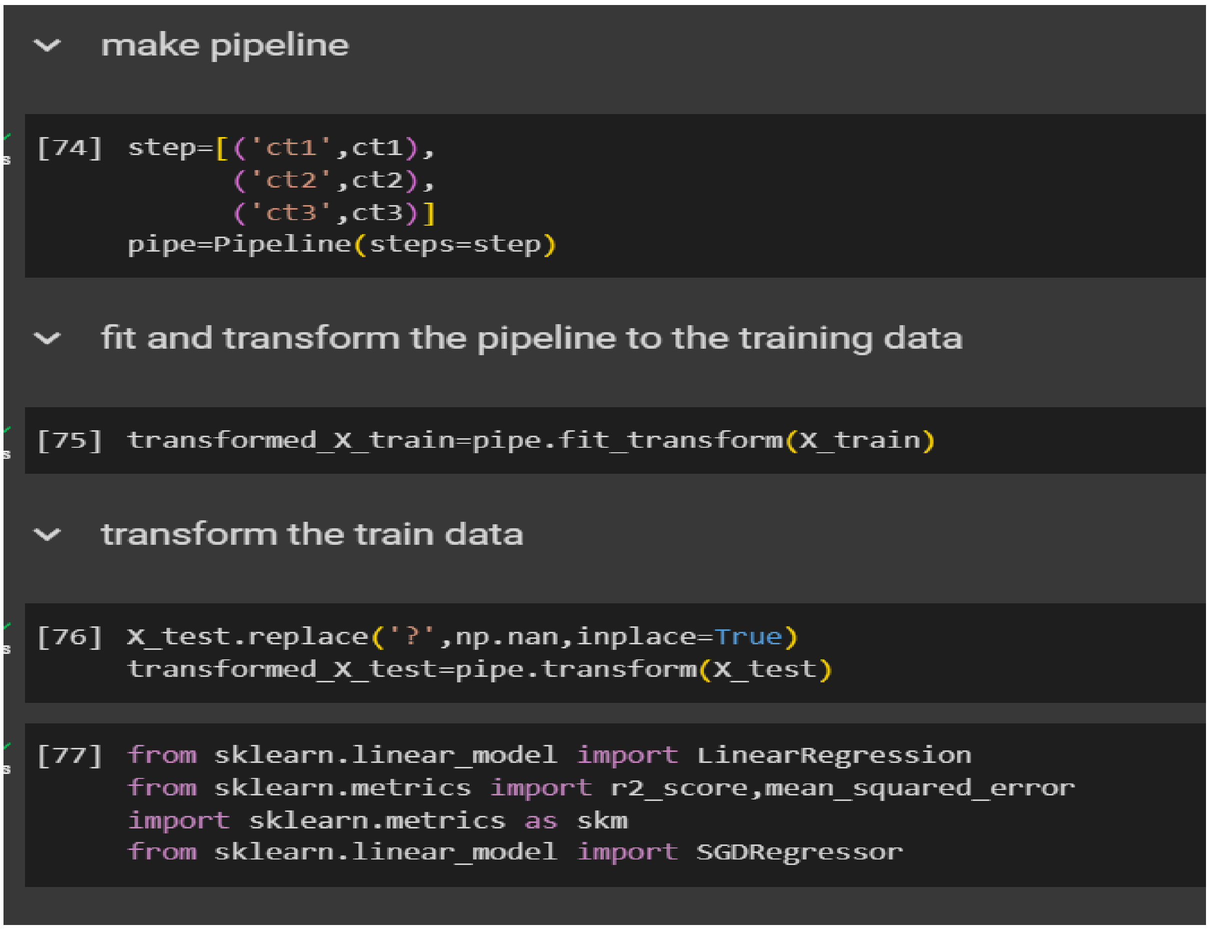
```
[71] X_train.replace('?',np.nan,inplace=True)
```

```
[72] from sklearn.preprocessing import StandardScaler, OneHotEncoder, OrdinalEncoder, MinMaxScaler
     from sklearn.compose import ColumnTransformer
     from sklearn.pipeline import Pipeline, FeatureUnion
     from sklearn.impute import SimpleImputer
     from sklearn import set_config
```

Imputation of the missing values

Scaling of the features

```
[73] my_list=[('mean',SimpleImputer(strategy='mean'),['Estimated Value','carpet_area'])]
     ct1=ColumnTransformer(transformers=my_list,remainder='passthrough',verbose_feature_names_out=False).set_output(transform='pandas')

     simple=[('most_frequent',SimpleImputer(strategy='most_frequent'),['Locality','Property'])]
     ct2=ColumnTransformer(transformers=simple,remainder='passthrough',verbose_feature_names_out=False).set_output(transform='pandas')

     my_list3=[('std',StandardScaler(),['Estimated Value','carpet_area']),
               ('ohe',OneHotEncoder( handle_unknown='ignore',sparse_output=False),['Locality','Property'])]
     ct3=ColumnTransformer(transformers=my_list3,verbose_feature_names_out=False)
```



make pipeline

```
[74] step=[('ct1',ct1),
           ('ct2',ct2),
           ('ct3',ct3)]
     pipe=Pipeline(steps=step)
```

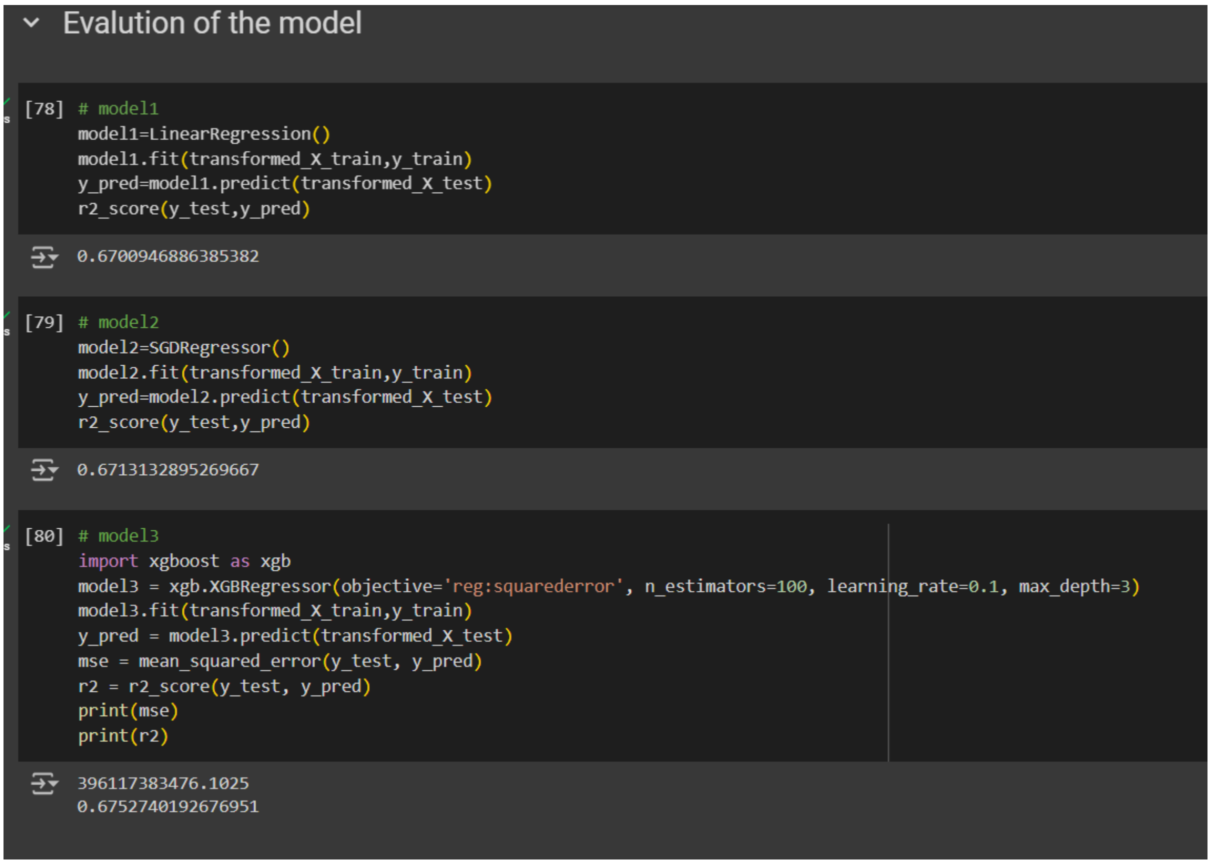
fit and transform the pipeline to the training data

```
[75] transformed_X_train=pipe.fit_transform(X_train)
```

transform the train data

```
[76] X_test.replace('?',np.nan,inplace=True)
     transformed_X_test=pipe.transform(X_test)
```

```
[77] from sklearn.linear_model import LinearRegression
     from sklearn.metrics import r2_score,mean_squared_error
     import sklearn.metrics as skm
     from sklearn.linear_model import SGDRegressor
```



Evalution of the model

```
[78] # model1
     model1=LinearRegression()
     model1.fit(transformed_X_train,y_train)
     y_pred=model1.predict(transformed_X_test)
     r2_score(y_test,y_pred)
```

```
0.6700946886385382
```

```
[79] # model2
     model2=SGDRegressor()
     model2.fit(transformed_X_train,y_train)
     y_pred=model2.predict(transformed_X_test)
     r2_score(y_test,y_pred)
```

```
0.6713132895269667
```

```
[80] # model3
     import xgboost as xgb
     model3 = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=100, learning_rate=0.1, max_depth=3)
     model3.fit(transformed_X_train,y_train)
     y_pred = model3.predict(transformed_X_test)
     mse = mean_squared_error(y_test, y_pred)
     r2 = r2_score(y_test, y_pred)
     print(mse)
     print(r2)
```

```
396117383476.1025
0.6752740192676951
```

8.0 Business Model

This machine learning model will be deployed as a business model, so that this service can reach to many users. This model will be developed as a software application. Users need to download the application. Then they have to put various input. According to their input, this application will show various house. If they want to check the price of the house they need to pay a specific amount. In this process this business model can generate revenue.

9.0 Financial Modelling

Consider per house price check user have to pay 300/-.

Total profit = y

Per house price check =m

Total rate as a function of time = x(t)

Total production and maintenance cost = c

Financial equation = >

y=mx(t)-c

m=300

c=1000

y=300x(t)-1000

10. Conclusion

In this house price prediction app project, we can successfully develop a machine learning model to estimate house prices based on various features. Product will be developed more according to market analysis.