```
import numpy as np
import pandas as pd
```

```
#!pip install feature-engine
```

```
df=pd.read_csv("/content/housing.csv")
```

```
df.head()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

```
df.info(memory_usage="deep")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object 
dtypes: float64(9), object(1)
memory usage: 2.7 MB
```

```
df.describe()
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000

scaling is required

ALso total rooms and bedrooms are way high for block.......it is given a block in range 600,3000

```
df.isnull().mean()
```

```
longitude             0.000000
latitude              0.000000
housing_median_age    0.000000
total_rooms           0.000000
total_bedrooms        0.010029
population            0.000000
households            0.000000
median_income         0.000000
median_house_value    0.000000
ocean_proximity       0.000000
dtype: float64
```

Since number of null is near 1% and entries are 20640 we can drop null values

```
df["ocean_proximity"].value_counts()
```

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

Double-click (or enter) to edit

```
df.corr()
```

- **We can see median_house_value depends highly on median_income which make sense**
- **Among total_rooms,total_bedrooms,population and households I have choosen *total_rooms as it has higher corr with price*** also took latitude

housing_median_age	-0.108197	0.011173	1.000000	-0.361262	-0.3

```
df=df.dropna()
df=df[df['population']>=600]
df=df[df['population']<=3000]
```

population	0.099773	0.108785	0.296244	0.857126	0.8

```
df.shape
```

```
(16271, 10)
```

```
from sklearn.model_selection import train_test_split

y = df['median_house_value']
X = df[['latitude','total_rooms','median_income']]

# For the larger the dataset, the smaller the test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

```
X.describe()
```

	latitude	total_rooms	median_income
count	16271.000000	16271.000000	16271.000000
mean	35.569522	2518.339623	3.867861
std	2.113046	1237.362697	1.841316
min	32.540000	121.000000	0.499900
25%	33.930000	1661.500000	2.587500
50%	34.220000	2237.000000	3.550600
75%	37.680000	3087.000000	4.728300
max	41.950000	19107.000000	15.000100

Double-click (or enter) to edit

```
from feature_engine.outliers import Winsorizer
capper = Winsorizer(capping_method='iqr', tail='both')
capper.fit(X_train)
X_train=capper.transform(X_train)
X_test=capper.transform(X_test)
```

```
from sklearn.preprocessing import RobustScaler
from feature_engine.wrappers import SklearnTransformerWrapper
scaler = SklearnTransformerWrapper(transformer = RobustScaler())
scaler.fit(X_train)
X_train=scaler.transform(X_train)
```

```
X_test=scaler.transform(X_test)
```

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
model.fit(X_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
pred=model.predict(X_test)
```

Double-click (or enter) to edit

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,pred)
```

```
6071014206.6652975
```

```
from sklearn.metrics import r2_score
r2_score(y_test,pred)
```

```
0.5243683694358883
```