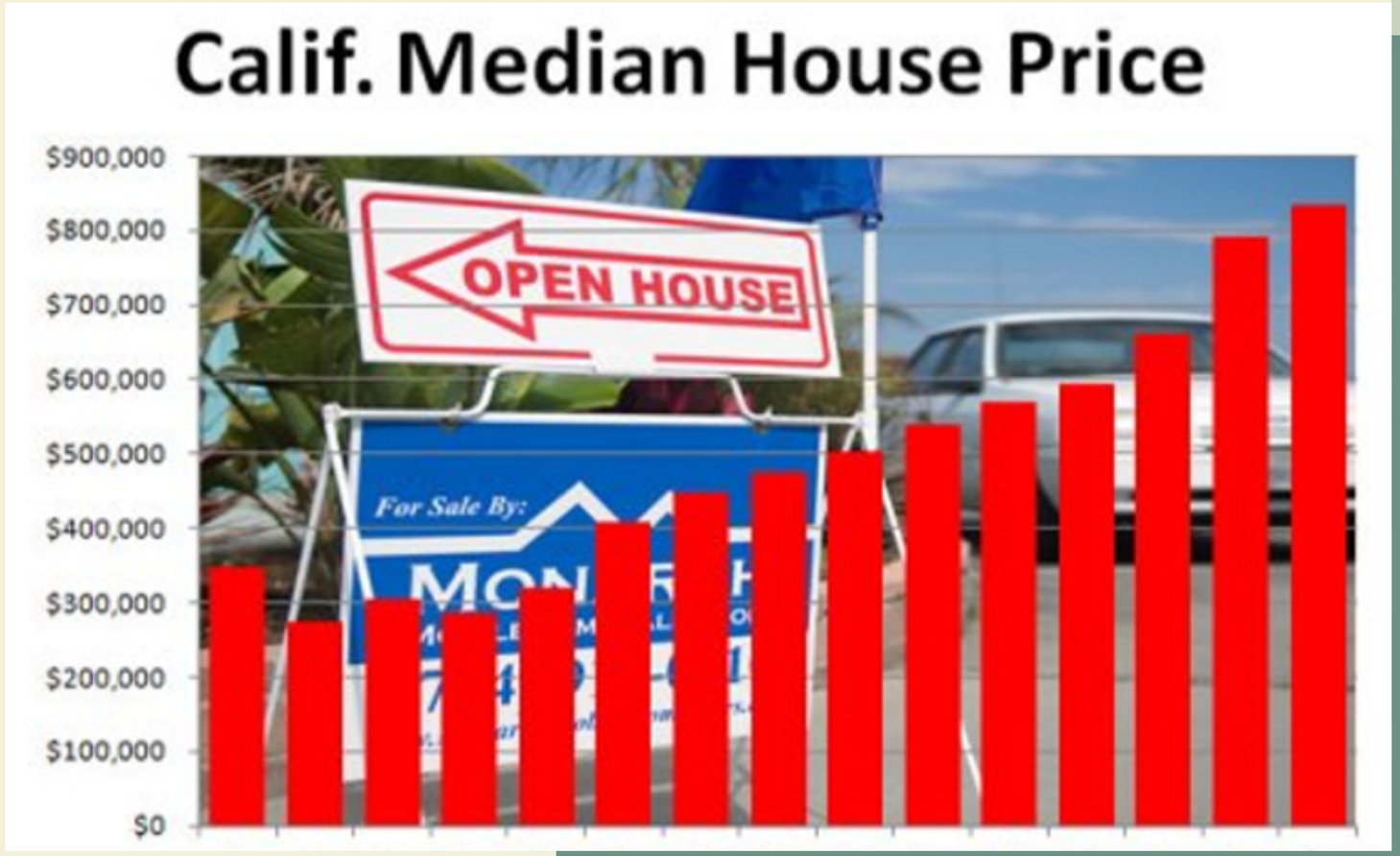
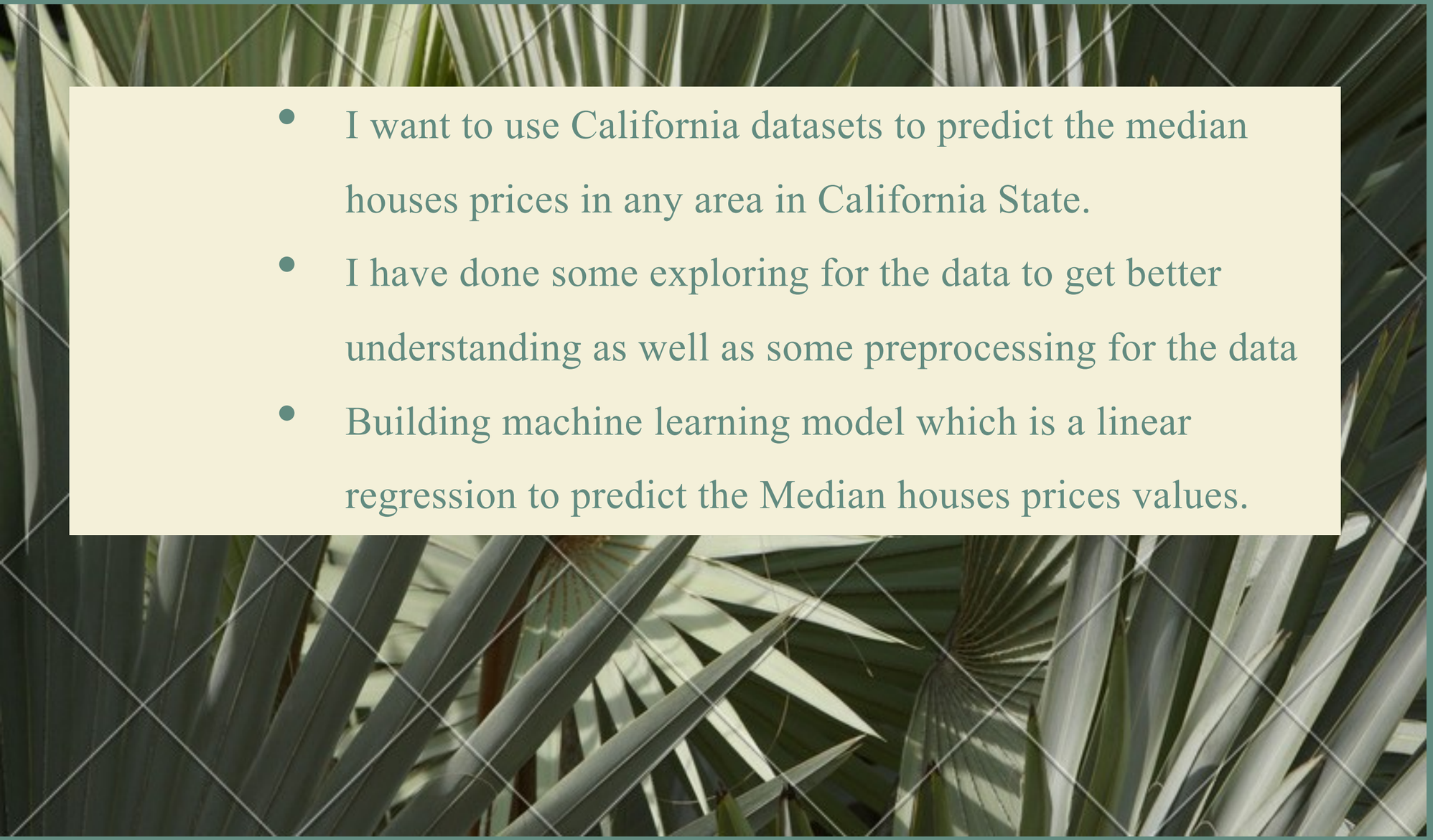


California Houses Prices Prediction

Introduction



- I want to use California datasets to predict the median houses prices in any area in California State.
- I have done some exploring for the data to get better understanding as well as some preprocessing for the data
- Building machine learning model which is a linear regression to predict the Median houses prices values.

Datasets Description

The data pertains to the houses found in a given California district and some summary stats about them based on the 1990 census data. Be warned the data aren't cleaned so there are some preprocessing steps required! The columns are as follows , their names are pretty self explanitory:

- Longitude
- Latitude
- Housingmedianage
- total_rooms
- total_bedrooms
- Population
- Households
- median_income
- Medianhousevalue
- ocean_proximity

Data Preprocessing (Analysis and Visualization)

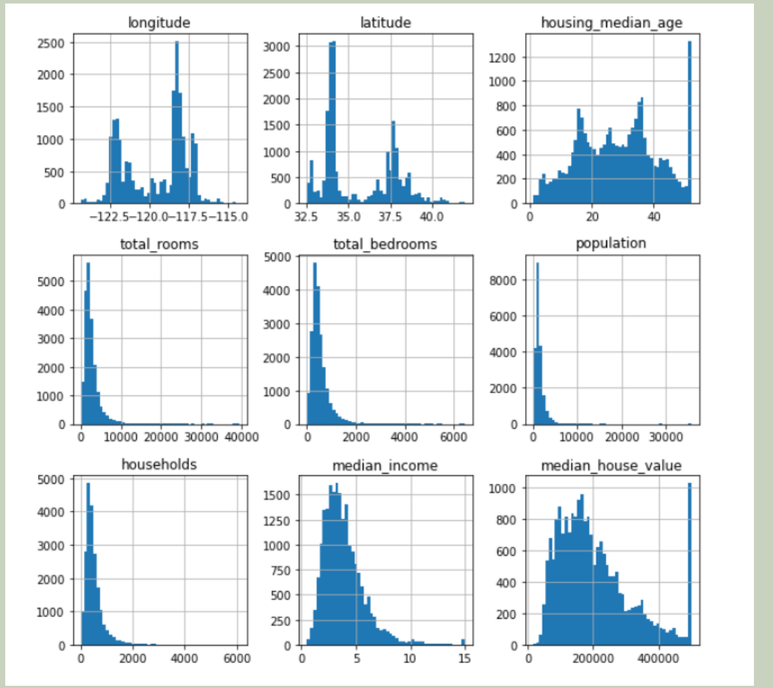
- Checking for missing values and imputing them by median

```
y = housing['median_house_value']
x = housing.drop('median_house_value', axis = 1)
x['total_bedrooms'].fillna(x.total_bedrooms.median(), inplace = True )
```

- Adding new features (columns) to get better understanding and to help for building ML model

```
x['rooms_per_house'] = x['total_rooms']/x['households']
x['bedrooms_per_room'] = x['total_bedrooms']/x['total_rooms'
x['population_per_houshold'] = x['population']/x['households
```

- Plotting histograms for the feature to understand the distribution



Data Preprocessing (Analysis and Visualization)

- **Splitting data for training and testing, and creating correlation matrix for training datasets**

- **Printing the correlation matrix**

- **Choosing specific columns for training and building a new matrix and then plotting histograms for them to understand the distribution**

```
x_train, x_test, y_train, y_test = train_test_split(x, y)
```

```
len(x_train)
```

```
15480
```

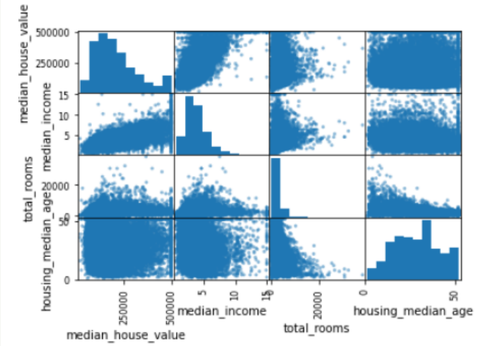
```
train = x_train.join(y_train)
corr_mat = train.corr()
```

```
corr_mat['median_house_value'].sort_values(ascending = False)
```

```
median_house_value          1.000000
median_income               0.689846
rooms_per_house             0.147699
total_rooms                 0.135577
housing_median_age          0.101256
households                  0.067721
total_bedrooms              0.049218
population                 -0.021525
population_per_houshold    -0.021730
longitude                  -0.046076
latitude                   -0.144464
bedrooms_per_room          -0.227088
Name: median_house_value, dtype: float64
```

```
housing_cols = train[['median_house_value','median_income','total_rooms','housing_median_age']]
```

```
scatter_matrix(housing_cols);
```



Machine learning Model

Building machine learning model (linear regression)

```
lm = LinearRegression()
x = train['median_income'].values.reshape(-1,1)
y = train['median_house_value']
lm.fit(x,y)
predict = lm.predict(x)
```

Predictions of the Model

- **Printing predicted values**

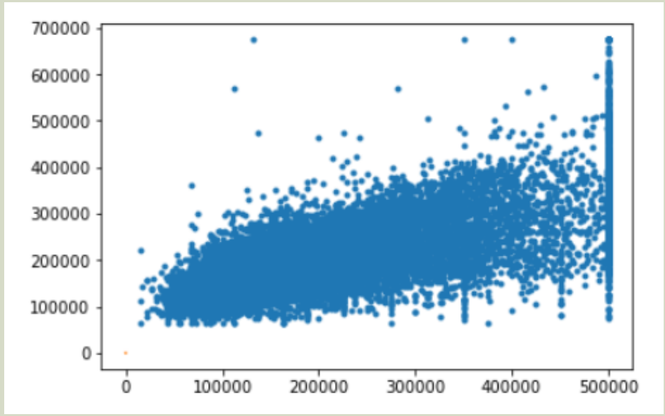
```
predict
```

```
array([270326.48920241, 498076.78368883, 215866.29098622, ...,
       440546.15415523, 253981.70002508, 319238.88069577])
```

- **Comparing between True and predicted values**

	True Value	Predictied value	Difference
2229	125300.0	270326.489202	-145026.489202
6919	500001.0	498076.783689	1924.216311
13061	109100.0	215866.290986	-106766.290986
10609	311500.0	347011.562872	-35511.562872
8506	195300.0	180686.719015	14613.280985
4050	500001.0	482787.718156	17213.281844
2840	76200.0	177877.064056	-101677.064056
4776	161500.0	152728.969746	8771.030254
4172	143800.0	157746.811611	-13946.811611
14264	98000.0	130100.311543	-32100.311543

- **Plotting predicted values**



Cross Validation

- **Printing predicted values**

```
print("Training Accuracy :", lm.score(x, y))
```

```
Training Accuracy : 0.47339685374682916
```

- **Mean Squared Error for testing**

```
mse = mean_squared_error(predict, y)
np.sqrt(mse)
```

```
83327.43137737601
```

Thank You !

Have a great day ahead.