CE9010 PROJECT

Predicting House Rent in Delhi

Aarushi Jain

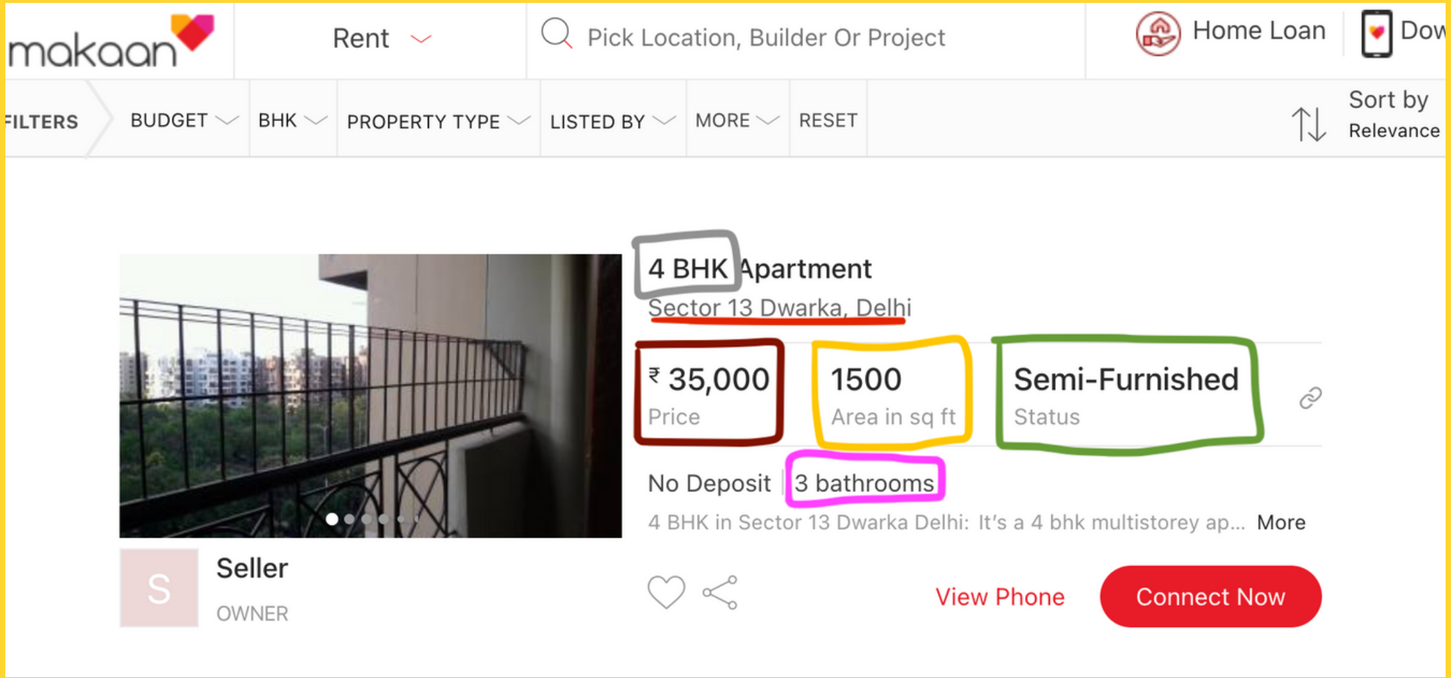
Introduction

MOTIVATION

- Housing Market in Delhi
- Complicated nature of prediction
- Personal Reasons

DATA ACQUISITION

- Data scraped from Makaan.com
- Important house information, and a new variable



	Location	House Price	Area	Status	No. of Bedrooms	No. of Bathrooms	Apartment Type	Latitude	Longitude	Distance
0	Malviya Nagar	17000	600	Unfurnished	2	1	BHK	28.5339	77.2124	6.44217
1	Saket	27000	1680	Semi-Furnished	3	3	BHK	28.5244	77.2137	7.34554
2	South Extension 2	75000	1800	Semi-Furnished	3	3	BHK	28.5668	77.2201	5.10805
3	Saket	23000	1650	Semi-Furnished	3	3	BHK	28.5244	77.2137	7.34554

DATA CLEANING

CONVERTING LAKH UNIT

1.45 Lakh rupees=
1.45 x 100000 INR=
145000 INR

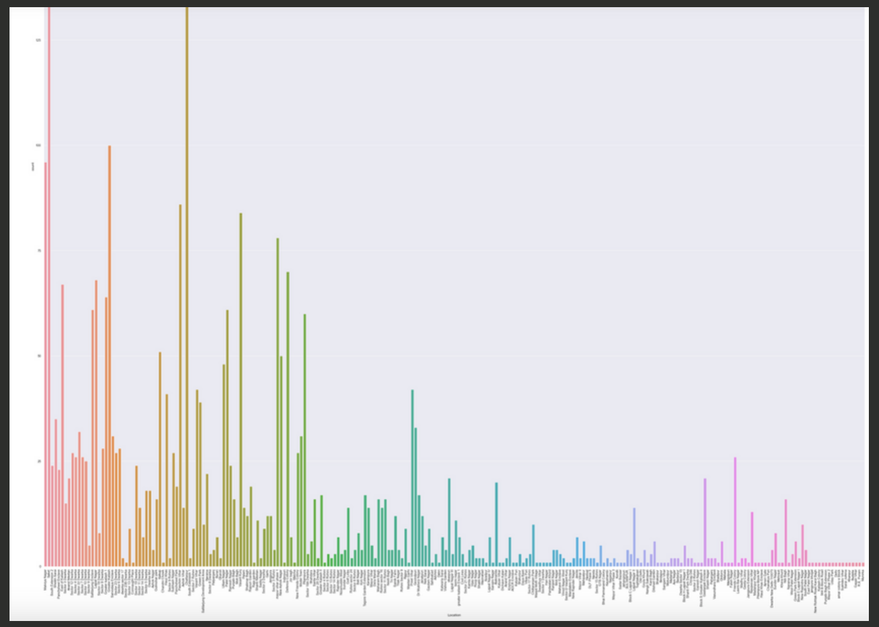
REMOVING COMMAS, CHANGING TO FLOAT

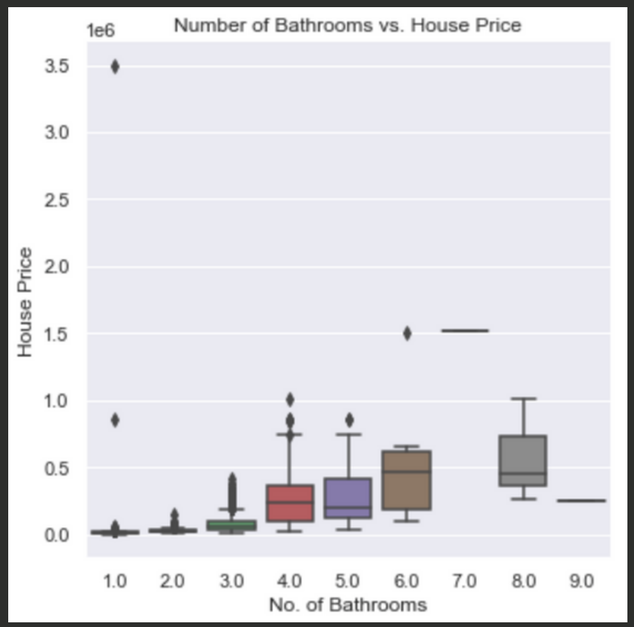
data.replace()
pd.to_numeric

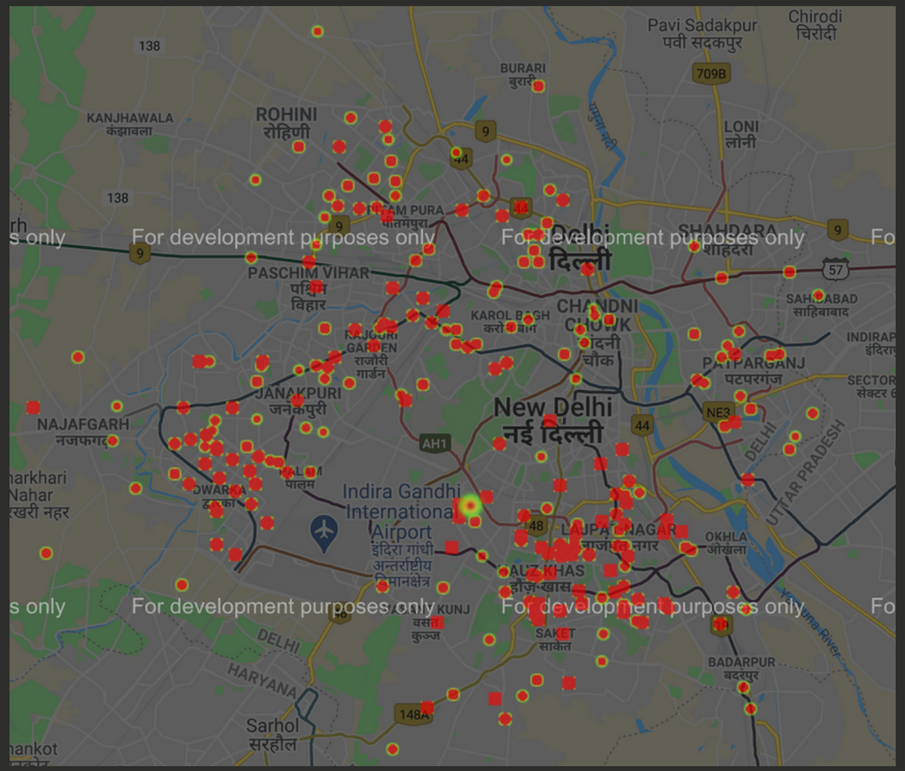
DROPPING "NA" VALUES

data.dropna()
lost 22% of data

```
Data type :  <class 'pandas.core.frame.DataFrame'>
Data dims :  (3084, 10)
Location              object
House Price          float32
Area                 float32
Status                object
No. of Bedrooms      float32
No. of Bathrooms     float32
Apartment Type        object
Latitude              object
Longitude             object
Distance             float32
```







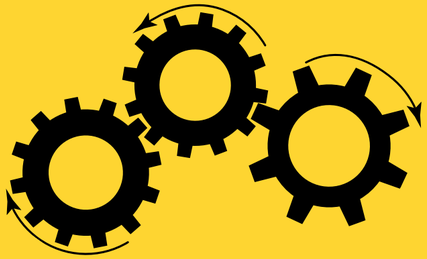
DATA EXPLORATION

Google Maps, Boxplots, Countplots, Heatmaps and more

```
meanx=data_x.mean()
stdevx=data_x.std()
data_x_n=(data_x-meanx)/stdevx
```

	Area	No. of Bedrooms	No. of Bathrooms	Distance
0	-0.728394	-0.421963	-1.168563	-0.771544
1	-0.012647	0.515598	0.618617	-0.517135
2	0.066880	0.515598	0.618617	-1.147260
3	-0.032529	0.515598	0.618617	-0.517135
4	-0.794667	-1.359525	-1.168563	-0.647236

Data Pre-processing

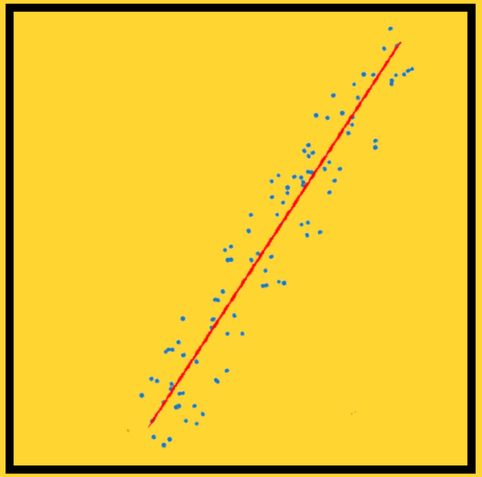


- Data normalization using Z-scoring
- One hot encoding to make sense of categorical data
- Splitting data into X and Y

```
statdum = pd.get_dummies(datafinal['Status'], prefix='Status')
locdum = pd.get_dummies(data['Location'],prefix='Location')
data_ohe = pd.concat([data_x, statdum,locdum], axis=1)
data_ohe_n = pd.concat([data_x_n, statdum,locdum], axis=1)
print(data_ohe_n.shape)
data_ohe_n
```

```
(3084, 251)
```

DATA ANALYSIS



LINEAR REGRESSION

LinearRegression() from sklearn

RANDOM FOREST

RandomForestRegressor() from sklearn

NEURAL NETWORKS

MLPRegressor() from sklearn

```
def neural(xdata,ydata,test=None,cross=1):
    X_train, X_test, y_train, y_test = train_test_split(xdata, ydata, test_size = 0.25)

    # Check the sample sizes
    print("Train Set Shape :", y_train.shape, X_train.shape)
    print("Test Set Shape :", y_test.shape, X_test.shape)

    neural_model = MLPRegressor()
    neural_model.fit(X_train, y_train)

    y_train_pred = neural_model.predict(X_train)
    y_test_pred = neural_model.predict(X_test)
```

Important parts of the code

RANDOM FOREST N_ESTIMATORS, K FOLD CROSS VALIDATION AND MORE

The 'test' are new data points for which the model predicts the house price

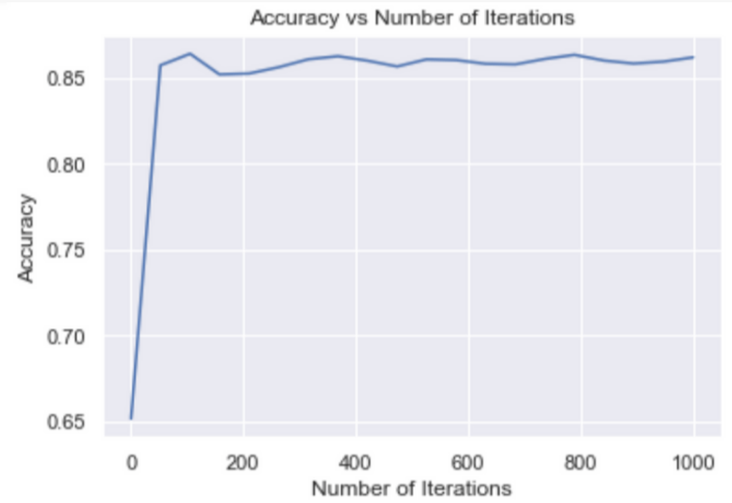
```
#Non-compulsory: To predict house price for new data points
if test is not None:
    test_pred=linreg_all.predict(test)
    print("Predicted House Price is:",test_pred)
else:
    print("no test")

#Non-compulsory: K-fold cross validation
if cross==1:
    cv = KFold(n_splits=10, random_state=1, shuffle=True)
    score= cross_val_score(linreg_all,xdata, ydata, cv=cv,scoring='r2')
    print('Cross Validation:')
    print('R_squared values:', score)
    print('R_squared mean:', score.mean())
    print("\n")
else:
    print("No cross validation")
```

Estimates the skill of the model by shuffling the dataset, splitting into k=10 sets, predicting the model

```
while count<n:
    variable = RandomForestRegressor(n_estimators=x[count])
    number= variable.fit(X_train, y_train)
    score= variable.score(X_test,y_test)
    y.append(score)
    if score==max(y):
        count1=count
    count+=1
plt.plot(x,y)
plt.title('Accuracy vs Number of Iterations')
plt.xlabel('Number of Iterations')
plt.ylabel('Accuracy')
plt.show()
```

In order to decrease time complexity of the regressor

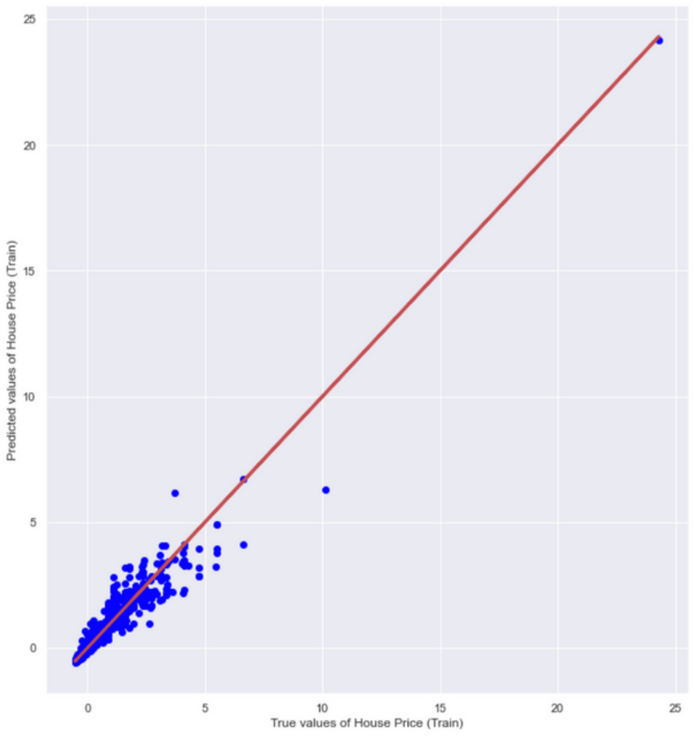


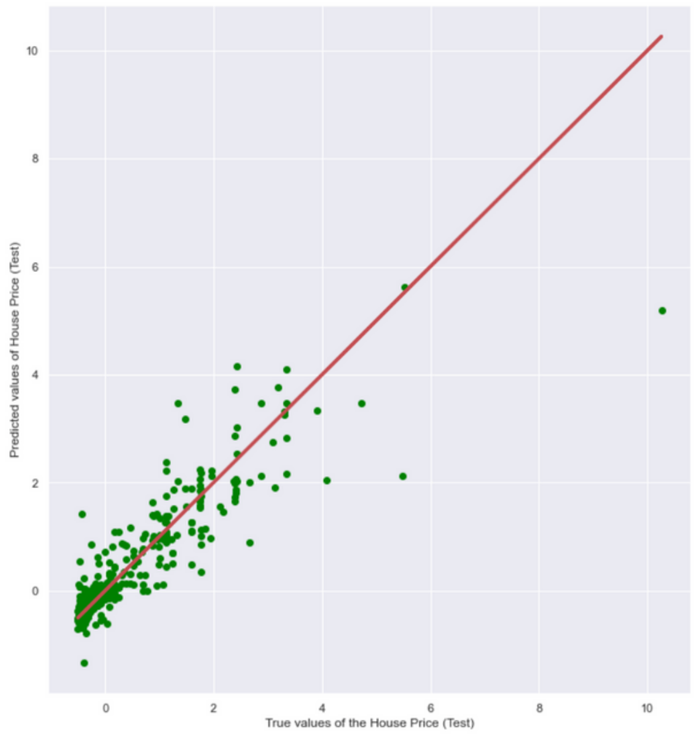

```
Random tree with number of iterations: 106
Accuracy: 0.852883367175693
```

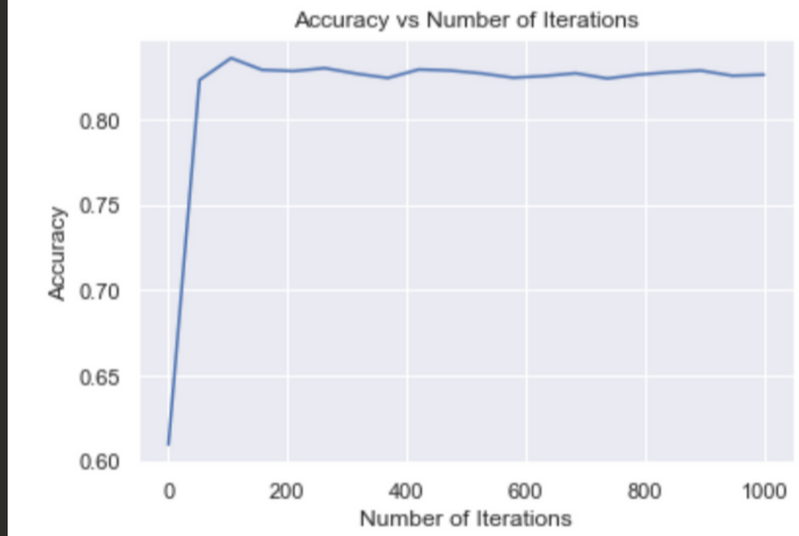
Findings

ACCURACY OF PREDICTIVE MODELS BASED ON MEAN R SQUARED VALUE FROM K FOLD VALIDATION

Predictive Model	Not Normalized		Normalized	
	w/out OHE	w/ OHE	w/out OHE	w/ OHE
Linear Regression	0.704	0.731	0.704	-106987
Random Forest	0.765	**0.776**	0.755	**0.785**
Neural Network	0.638	0.639	0.751	**0.774**








```
Random tree with number of iterations: 106
Accuracy: 0.8238656647895012
No test specified
Cross Validation:
R_squared values: [0.82880622 0.84927257 0.91907513
 0.79537304 0.8116912  0.6190757  0.82923002]
R_squared mean: 0.781360657707249
```

Testing real data

TWO NEW DATA POINTS

	Area	No. of Bedrooms	No. of Bathrooms	Distance
0	1000.0	3.0	2.0	14.015196
1	927.0	2.0	1.0	10.173493

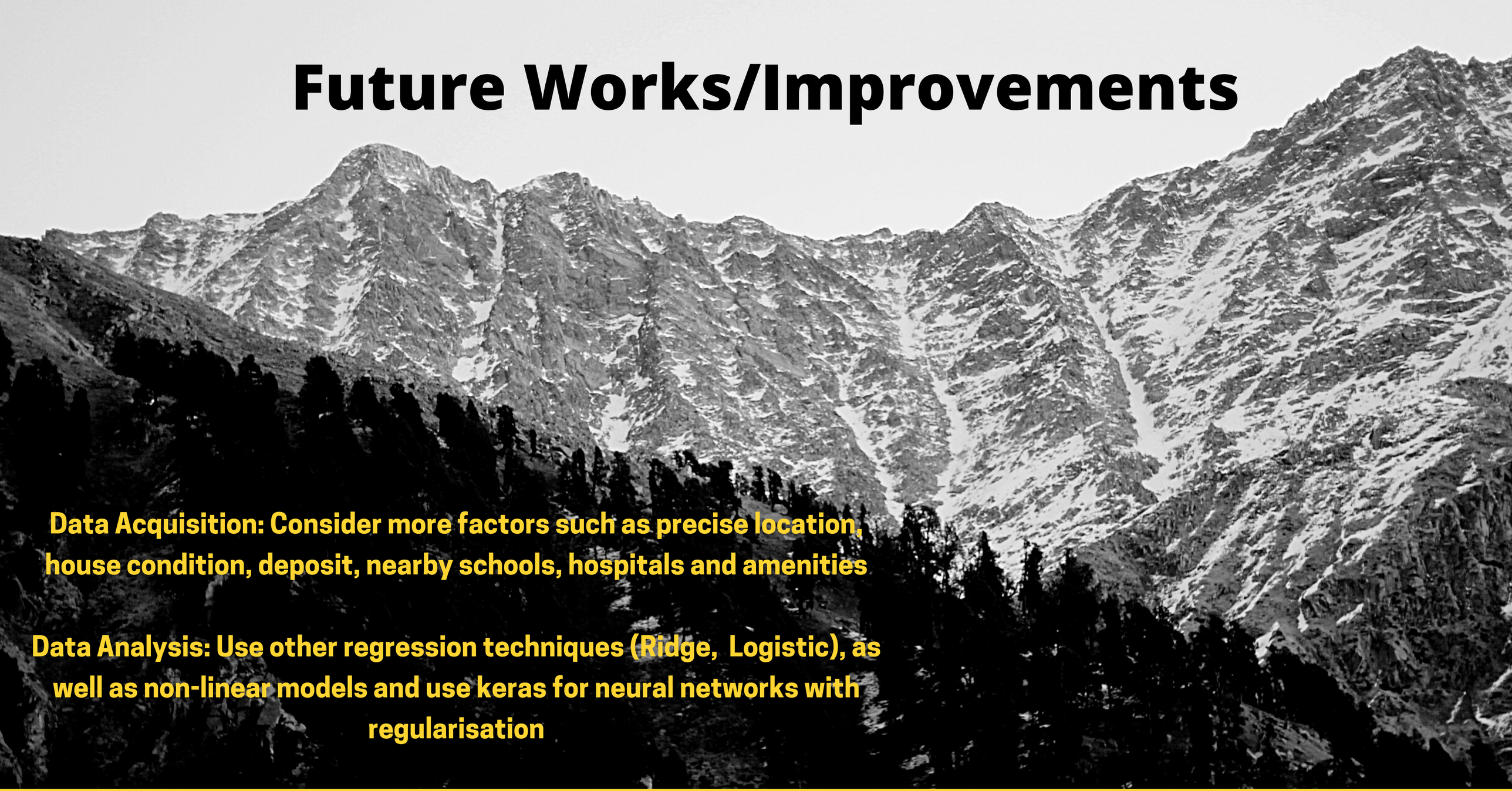
Actual Rent Prices: [27500, 20000]

Random Forest with One Hot Encoding and Normalization is the Winner!!

```
def normtoreal(var):
    return var*stdevy+meany
print("Normalized Random Forest Prediction:")
print(normtoreal(-0.35834767))
print(normtoreal(-0.40214251))
print("Non Normalized Random Forest Prediction:")
print("24443.57077888")
print("19275.42315504")
print("Normalized Neural Network Prediction:")
print(normtoreal(-0.29045409))
print(normtoreal(-0.27437021))
```

```
Normalized Random Forest Prediction:
25457.384539824372
19287.942473941876
Non Normalized Random Forest Prediction:
24443.57077888
19275.42315504
Normalized Neural Network Prediction:
35021.650751445624
37287.410047623125
```

Future Works/Improvements



Data Acquisition: Consider more factors such as precise location, house condition, deposit, nearby schools, hospitals and amenities

Data Analysis: Use other regression techniques (Ridge, Logistic), as well as non-linear models and use keras for neural networks with regularisation