Predicting Housing Prices Using Housing Features as Predictors

The dataset used in this part includes 3000 training samples and 999 test samples, and it examines the relationship between 11 characteristics and a single label variable. The characteristics include continuous and categorical (binary) elements like the quantity of rooms, baths, and kitchens, as well as extras like eco-friendly paint, a backyard, solar electricity, wood floors, QLM security, and club access. The categorical variables are displayed as either "Yes (1)" or "No (0)". To explore the effect of these housing attributes on home prices, the main goal is to create a linear regression model and a random forest regression model. The project objectives include determining the factors that have the most influence and assessing how well the models perform overall.

In [10]:
```
train.describe()
```

Out[10]:

	room	bathroom	kitchen	french_door	price
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	2.990000	1.489000	1.522000	1.998333	8606.600000
std	1.424281	0.499962	0.499599	0.813153	2216.248563
min	1.000000	1.000000	1.000000	1.000000	2235.000000
25%	2.000000	1.000000	1.000000	1.000000	7005.000000
50%	3.000000	1.000000	2.000000	2.000000	8615.000000
75%	4.000000	2.000000	2.000000	3.000000	10215.000000
max	5.000000	2.000000	2.000000	3.000000	15035.000000

Figure 1: Data Description of the Continuous Variables

The summary of the training dataset in figure 1 shows that the average house has almost 3 rooms, 1.5 bathrooms, 1.5 kitchens, and 2 French doors. The mean price of a house in the dataset is around 8606.6, with a standard deviation of 2216.2. The minimum price is 2235, while the maximum price is 15035. The quartile information indicates that 25% of the houses have a price less than or equal to 7005, 50% of the houses have a price less than or equal to 8615, and 75% of the houses have a price less than or equal to 10215.

```
In [11]: ##Frequency Distribution
         columns = ['backyard','furnished', 'green_paint', 'solar_power', 'woodfloor','qlm_security','club_access']
         for column in columns:
             print(train[column].value_counts())

0    1529
1    1471
Name: backyard, dtype: int64
0    1534
1    1466
Name: furnished, dtype: int64
0    1545
1    1455
Name: green_paint, dtype: int64
0    1513
1    1487
Name: solar_power, dtype: int64
1    1537
0    1463
Name: woodfloor, dtype: int64
0    1558
1    1442
Name: qlm_security, dtype: int64
0    1501
1    1499
Name: club_access, dtype: int64
```

Figure 2: Data Description of the Categorical Variables

Figure 2 displays the distribution of the presence or absence of certain amenities in the dataset of houses, which may potentially impact their prices. The variables "backyard," "furnished," "green paint," "solar power," and "qlm security" are binary categorical variables. For the "backyard" variable, 1529 houses do not have a backyard while 1471 houses have one. Similarly, for the "furnished" variable, 1534 houses are not furnished while 1466 houses are. In the case of the "green paint" variable, 1545 houses do not have green paint while 1455 houses do. For the "solar power" variable, 1513 houses do not have solar power while 1487 houses do. For the "wood floor" variable, 1537 houses have wood floors while 1463 do not. For the "qlm security" variable, 1558 houses do not have QLM security while 1442 do. Finally, for the "club access" variable, 1501 houses do not have club access while 1499 do.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.508e+31
Date:                Thu, 04 May 2023   Prob (F-statistic):               0.00
Time:                        17:31:25   Log-Likelihood:                 71913.
No. Observations:                3000   AIC:                        -1.438e+05
Df Residuals:                    2988   BIC:                        -1.437e+05
Df Model:                          11
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const            195.0000   1.05e-12   1.87e+14      0.000     195.000     195.000
room            1000.0000   1.21e-13   8.27e+15      0.000    1000.000    1000.000
bathroom         300.0000   3.45e-13    8.7e+14      0.000     300.000     300.000
kitchen          500.0000   3.45e-13   1.45e+15      0.000     500.000     500.000
french_door      240.0000   2.12e-13   1.13e+15      0.000     240.000     240.000
backyard_1       560.0000   3.44e-13   1.63e+15      0.000     560.000     560.000
furnished_1     2000.0000   3.45e-13    5.8e+15      0.000    2000.000    2000.000
green_paint_1    370.0000   3.45e-13   1.07e+15      0.000     370.000     370.000
solar_power_1   1530.0000   3.44e-13   4.44e+15      0.000    1530.000    1530.000
woodfloor_1     1890.0000   3.44e-13   5.49e+15      0.000    1890.000    1890.000
qlm_security_1   440.0000   3.45e-13   1.28e+15      0.000     440.000     440.000
club_access_1    730.0000   3.45e-13   2.12e+15      0.000     730.000     730.000
==============================================================================
Omnibus:                       27.814   Durbin-Watson:                   0.689
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               18.291
Skew:                           0.007   Prob(JB):                     0.000107
Kurtosis:                       2.618   Cond. No.                         29.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Figure 3: Linear Regression Output for predicting house prices.

The OLS regression results in figure 3.1 provide estimates for the parameters of the linear model that relate the predictor variables (independent variables) to the response variable (dependent variable). For the continuous variables, the coefficients indicate the anticipated price change for every one-unit increase in the corresponding variable, with all other variables being held constant. Specifically, an increase of one unit in the "room" variable is expected to result in a £1000 increase in the price of the house. Similarly, a one-unit increase in the "bathroom" variable is expected to increase the price of the house by £300, and an increase of one unit in the "kitchen" variable is expected to increase the price of the house by £500. Moreover, an increase of one unit in the "French door" variable is expected to increase the price of the house by £240 while holding all other variables constant.

The coefficients for the categorical variables are as follows: a house with a backyard has a coefficient of 560, meaning it costs £560 more than one without; a furnished house has a coefficient of 2000, meaning it costs £2000 more than an unfurnished one; a house with green paint has a coefficient of 370, meaning it costs £370 more than one without; a house with solar power has a coefficient of 1530, meaning it costs £1530 more than one without; a house with wood floors has a coefficient of 1890, meaning it costs £1890 more than one without; a house with QLM security has a coefficient of 440, meaning it costs £440 more than one without; and a house with club access has a coefficient of 730, meaning it costs £730 more than one without.

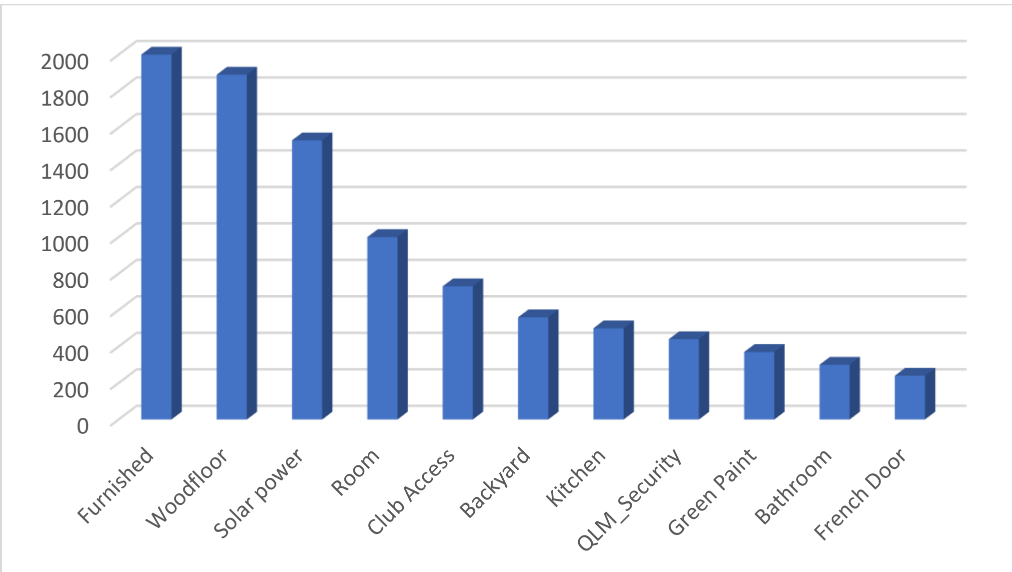

Figure 4: Feature Importance using Linear Regression for predicting house prices.

The feature importance in figure 4 shows the impact of each feature on the prediction output of the linear regression model. The values represent how much each feature contributes to the prediction, with higher values indicating more significant impact. According to the figure, the most important feature is "furnished_1" with a value of 1999.999, followed by "woodfloor_1" with a value of 1890, and "solar_power_1" with a value of 1529.999. These features have the highest impact on the predicted house price. On the other hand, "French door" has the lowest impact with a value of 240.

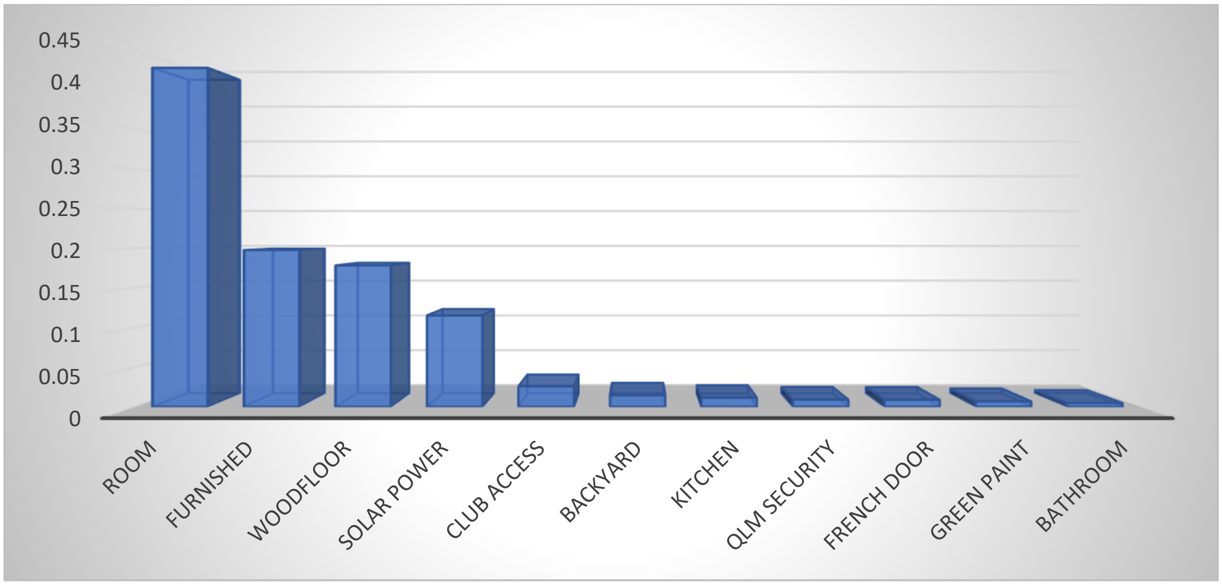

Figure 5: Feature Importance using Random Forest Regression for predicting house prices.

The Random Forest regression model in figure 5 suggested that for the Random Forest model, the most important feature is "room", followed by "furnished_1" and "woodfloor_1". The least important features are "bathroom", "green_paint_1", and "French door".

Table 1: Comparison of the Predictive Performance of the Models

Metrics	Linear Regression	Random Forest
RMSE	12.9999	224.8898
R-Squared	0.9999	0.9897

Table 1 presents a comparison of the predictive performance of two different models: linear regression and random forest. Two metrics are used to evaluate the models: Root Mean Squared

Error (RMSE) and R-Squared. Based on these metrics, the linear regression model has a better predictive performance compared to the random forest model for this particular dataset.

Conclusions

Based on the feature importance analysis, we found that the most important features for predicting the house price are furnished, wood floor, solar power, room, and club access for both the Linear Regression. The Random Forest model also indicates that the room feature is the most important feature, followed by furnished and wood floor. Both models provide useful insights into predicting the house price, but the Linear Regression model has better predictive performance, while the Random Forest model provides better feature importance analysis.

Appendix

Python Code

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from scipy import stats
```

In [2]:
```
train = pd.read_csv (r"C:\...\msc_training_dataset.csv")
train.head(6)
```

Out[2]:

	room	bathroom	kitchen	french_door	backyard	furnished	green_paint	solar_power	woodfloor	qlm_security	club_access	price
0	3	1	2	1	1	0	1	0	0	1	1	6835
1	5	2	2	2	1	0	0	0	0	1	1	9005
2	5	2	2	2	1	0	0	0	0	1	1	9005
3	1	2	1	2	0	0	0	0	1	1	0	5105
4	2	1	2	3	1	1	0	0	1	1	0	9105
5	5	1	2	1	0	0	1	0	1	0	0	8995

In [3]:
```
test = pd.read_csv (r"C:\...\msc_testing_dataset .csv")
test.head(6)
```

Out[3]:

	room	bathroom	kitchen	french_door	backyard	furnished	green_paint	solar_power	woodfloor	qlm_security	club_access	price
0	1	1	1	3	0	0	1	1	0	1	0	5068
1	5	1	1	2	0	0	0	0	0	1	1	7658
2	5	1	1	3	0	0	0	1	1	1	1	11318
3	4	2	2	1	0	1	1	0	0	1	0	8858
4	5	2	1	1	0	1	1	1	0	0	1	11178
5	5	1	1	2	1	1	1	1	0	1	0	11388

In [4]:
```
train.isna().sum().sort_values(ascending=False)
test.isna().sum().sort_values(ascending=False)
```

Out[4]:
```
room            0
bathroom        0
kitchen         0
french_door     0
backyard        0
furnished       0
green_paint     0
solar_power     0
woodfloor       0
qlm_security    0
club_access     0
price           0
dtype: int64
```

In [5]:
```
train = train.astype({'backyard': 'category',
                      'furnished': 'category',
                      'green_paint': 'category',
                      'solar_power': 'category',
                      'woodfloor': 'category',
                      'qlm_security': 'category',
                      'club_access': 'category'})
```

In [6]:
```
train['backyard'] = pd.Categorical(train['backyard'])
train['furnished'] = pd.Categorical(train['furnished'])
train['green_paint'] = pd.Categorical(train['green_paint'])
train['solar_power'] = pd.Categorical(train['solar_power'])
train['woodfloor'] = pd.Categorical(train['woodfloor'])
train['qlm_security'] = pd.Categorical(train['qlm_security'])
train['club_access'] = pd.Categorical(train['club_access'])
```

In [7]:
```
test = test.astype({'backyard': 'category',
                    'furnished': 'category',
                    'green_paint': 'category',
                    'solar_power': 'category',
                    'woodfloor': 'category',
                    'qlm_security': 'category',
                    'club_access': 'category'})
```

In [8]:
```
test['backyard'] = pd.Categorical(test['backyard'])
test['furnished'] = pd.Categorical(test['furnished'])
test['green_paint'] = pd.Categorical(test['green_paint'])
test['solar_power'] = pd.Categorical(test['solar_power'])
test['woodfloor'] = pd.Categorical(test['woodfloor'])
test['qlm_security'] = pd.Categorical(test['qlm_security'])
test['club_access'] = pd.Categorical(test['club_access'])
```

In [9]:
```
train.dtypes
```

Out[9]:
```
room               int64
bathroom           int64
kitchen            int64
french_door        int64
backyard        category
furnished       category
green_paint     category
solar_power     category
woodfloor       category
qlm_security    category
club_access     category
price              int64
dtype: object
```

In [10]:
```
train.describe()
```

Out[10]:

	room	bathroom	kitchen	french_door	price
count	3000.000000	3000.000000	3000.000000	3000.000000	3000.000000
mean	2.990000	1.489000	1.522000	1.998333	8606.600000
std	1.424281	0.499962	0.499599	0.813153	2216.248563
min	1.000000	1.000000	1.000000	1.000000	2235.000000
25%	2.000000	1.000000	1.000000	1.000000	7005.000000
50%	3.000000	1.000000	2.000000	2.000000	8615.000000
75%	4.000000	2.000000	2.000000	3.000000	10215.000000
max	5.000000	2.000000	2.000000	3.000000	15035.000000

In [11]:
```
##Frequency Distribution
columns = ['backyard','furnished', 'green_paint', 'solar_power', 'woodfloor','qlm_security','club_access']
for column in columns:
    print(train[column].value_counts())
```

```
0    1529
1    1471
Name: backyard, dtype: int64
0    1534
1    1466
Name: furnished, dtype: int64
0    1545
1    1455
Name: green_paint, dtype: int64
0    1513
1    1487
Name: solar_power, dtype: int64
1    1537
0    1463
Name: woodfloor, dtype: int64
0    1558
1    1442
Name: qlm_security, dtype: int64
```

In [12]:
```
###Creating a dummy variable for training set
train1 = pd.get_dummies(train, columns=['backyard', 'furnished', 'green_paint', 'solar_power', 'woodfloor', 'qlm_security', 'club
train1.head(4)
```

Out[12]:

	room	bathroom	kitchen	french_door	price	backyard_1	furnished_1	green_paint_1	solar_power_1	woodfloor_1	qlm_security_1	club_access_1
0	3	1	2	1	6835	1	0	1	0	0	1	1
1	5	2	2	2	9005	1	0	0	0	0	1	1
2	5	2	2	2	9005	1	0	0	0	0	1	1
3	1	2	1	2	5105	0	0	0	0	1	1	0

In [13]:
```
###Creating a dummy variable for testing set
test1 = pd.get_dummies(test, columns=['backyard', 'furnished', 'green_paint', 'solar_power', 'woodfloor', 'qlm_security', 'club_a
test1.head(4)
```

Out[13]:

	room	bathroom	kitchen	french_door	price	backyard_1	furnished_1	green_paint_1	solar_power_1	woodfloor_1	qlm_security_1	club_access_1
0	1	1	1	3	5068	0	0	1	1	0	1	0
1	5	1	1	2	7658	0	0	0	0	0	1	1
2	5	1	1	3	11318	0	0	0	1	1	1	1
3	4	2	2	1	8858	0	1	1	0	0	1	0

In [14]:
```
# Declaring and spartitioning the dataset
###Training set
X_train = train1.drop("price", axis=1)
y_train = train1["price"]

### Testing set
X_test = test1.drop("price", axis=1)
y_test = test1["price"]
```

In [15]:
```
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [16]:
```
# Fit linear regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
```

Out[16]: LinearRegression()

In [17]:
```python
import statsmodels.api as sm
######Linear Regression Analysis
Xtrain = sm.add_constant(X_train)
reg = sm.OLS(y_train, Xtrain).fit()

# print the summary of the model
print(reg.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 1.508e+31
Date:                Thu, 04 May 2023   Prob (F-statistic):               0.00
Time:                        17:31:25   Log-Likelihood:                 71913.
No. Observations:                3000   AIC:                        -1.438e+05
Df Residuals:                    2988   BIC:                        -1.437e+05
Df Model:                          11
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
const            195.0000   1.05e-12   1.87e+14      0.000     195.000     195.000
room            1000.0000   1.21e-13   8.27e+15      0.000    1000.000    1000.000
bathroom         300.0000   3.45e-13    8.7e+14      0.000     300.000     300.000
kitchen          500.0000   3.45e-13   1.45e+15      0.000     500.000     500.000
french_door      240.0000   2.12e-13   1.13e+15      0.000     240.000     240.000
backyard_1       560.0000   3.44e-13   1.63e+15      0.000     560.000     560.000
furnished_1     2000.0000   3.45e-13    5.8e+15      0.000    2000.000    2000.000
green_paint_1    370.0000   3.45e-13   1.07e+15      0.000     370.000     370.000
solar_power_1   1530.0000   3.44e-13   4.44e+15      0.000    1530.000    1530.000
woodfloor_1     1890.0000   3.44e-13   5.49e+15      0.000    1890.000    1890.000
qlm_security_1   440.0000   3.45e-13   1.28e+15      0.000     440.000     440.000
club_access_1    730.0000   3.45e-13   2.12e+15      0.000     730.000     730.000
==============================================================================
Omnibus:                       27.814   Durbin-Watson:                   0.689
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               18.291
Skew:                           0.007   Prob(JB):                     0.000107
Kurtosis:                       2.618   Cond. No.                         29.2
==============================================================================
```

In [18]:
```python
# Get coefficients and sort by absolute value
coefficients = lin_reg.coef_
feature_names = X_train.columns
indices = np.argsort(np.abs(coefficients))[::-1]

print('Feature Importance (Linear Regression):')
for f in range(X_train.shape[1]):
    print(feature_names[indices[f]], ':', coefficients[indices[f]])
```

```
Feature Importance (Linear Regression):
furnished_1 : 1999.9999999999975
woodfloor_1 : 1890.0000000000002
solar_power_1 : 1529.9999999999993
room : 1000.0000000000002
club_access_1 : 729.9999999999999
backyard_1 : 560.0000000000013
kitchen : 500.0000000000017
qlm_security_1 : 440.0000000000003
green_paint_1 : 370.00000000000045
bathroom : 300.0000000000005
french_door : 240.00000000000023
```

```
In [19]: # Fit random forest regression model
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)
rf_reg.fit(X_train, y_train)

Out[19]: RandomForestRegressor(random_state=42)
```

```
In [20]: # Calculate feature importance for random forest model
importances = rf_reg.feature_importances_
feature_names = X_train.columns
indices = np.argsort(importances)[::-1]
print('Feature Importance (Random Forest):')
for f in range(X_train.shape[1]):
    print(feature_names[indices[f]], ':', importances[indices[f]])

Feature Importance (Random Forest):
room : 0.42751860545238657
furnished_1 : 0.19767503879356174
woodfloor_1 : 0.1782731259197991
solar_power_1 : 0.11517045330099454
club_access_1 : 0.02591107755049794
backyard_1 : 0.014622057069054077
kitchen : 0.011361951753313705
qlm_security_1 : 0.008978174668507427
french_door : 0.008891939653965673
green_paint_1 : 0.006909489489235856
bathroom : 0.0046880863486834165
```

```
In [21]: # Predict on testing set and evaluate performance
y_pred_lin = lin_reg.predict(X_test)
rmse_lin = np.sqrt(mean_squared_error(y_test, y_pred_lin))
r2_lin = r2_score(y_test, y_pred_lin)
print('Linear Regression Performance:')
print('RMSE:', rmse_lin)
print('R^2:', r2_lin)

Linear Regression Performance:
RMSE: 12.999999999999572
R^2: 0.9999656095212318
```

```
In [22]: # Predict on testing set and evaluate performance
y_pred_rf = rf_reg.predict(X_test)
rmse_rf = np.sqrt(mean_squared_error(y_test, y_pred_rf))
r2_rf = r2_score(y_test, y_pred_rf)
print('Random Forest Regression Performance:')
print('RMSE:', rmse_rf)
print('R^2:', r2_rf)

Random Forest Regression Performance:
RMSE: 224.88978101195173
R^2: 0.9897082089483727
```

Data links

httpsdrive.google.comfiled1f-VKXrqc_Hj8QY1TC5uU8AFNuSc7_Dt6viewusp=sharing
httpsdrive.google.comfiled1TKrP4mE6rsq3effIecEhHV4BuYT-zbskviewusp=sharing