Phase 2 Project: Technical Presentation of Price Predictor

Overview

- The goal of this project is to develop a predictive model for house pricing in King County.
- The model will estimate how the features of a house will affect its price.
- The price estimation tool may be benefitial for Real Estates Agencies and Developers, as well as individual sellers and buyers.

Objective

- Determining how multiple features work individually and together
- Quantifying joined features effect
- Building a predictive model
- Building a front end for a customer

Solution

- Analyzing 2014-2015 dataset with past sales
- Identifying individual and joined factors.
- Prepairing features for the model
- Calculate all the features coefficients
- Testing the results

Data

King County house sales dataset contains:

- details for 22,000 sold houses
- final sales prices

All the data is from 2014-2015

Features Identified

Main Features:

- House Sq footage
- Grade of design and materials quality
- Zipcode
- Waterfront
- View

Additional Features:

- Lot size
- Basement
- House Age

Only marginal effect from:

- Renovation, number of bedrooms, bathrooms, and floors

more on feature analysis - see "analysis_and_regression/Investigation of Features.ipynb"

Initial Data Load and Cleaning

- **Loaded the "kc_house_data.csv" using "initial_data_prep.py"**
- **filled or removed rows with missing properties**
- **Construction Grade 3-5 (below the acceptable code) were removed**
- **Out of 22,000 rows 20,880 were used in the model**

```
## importing Libraries
%run code/import_libs.py

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
%matplotlib inline

## Importing Functions
%run code/functions_v1.4.py

## Loading and Initial preparation of the data (fillnulls, new features, filtering)
%run code/initial_data_prep.py

#FILTER grade
df=df[~df["grade"].isin([3,4,5])].copy()

# SPLITTING DATA IN PREDICTORS(X) and price (Y)

initial_pred = df.drop(columns=["price"]).copy()
initial_price = df[["price"]]

mean_price_2014_2015=initial_price.mean()[0]
df.shape
```

```
(20880, 27)
```

Data Modeling

An iterative approach to data modeling

- Calculating Efficiency for basic features
- Prepairing model features
- Training multiple models
- Chosing the most efficient model
- Testing against different subset of data

Steps:

- Prepared data for modeling using custom "transform_data" function (see functions_v1.4.py)
- Created/trained model using statsmodels.OLS
- Made sure r-square is higher than 80%

```
# Create OLS linear model
pred_fin, price_fin = transform_data(initial_pred, initial_price)

pred_int = sm.add_constant(pred_fin)
model = sm.OLS(price_fin,pred_int).fit()

print(model.rsquared)
coef_df=model.params.reset_index()
coef_df.columns=["Column","Value"]
```

```
0.8812894359143344
```

In addition to automatic -sklearn- methods, custom functions were created to manually get all the coefficients from statsmodels OLS and calculate the linear slopes formula

- used custom function "calcuate_price" and "get_coeff" to get coefficients from ols model (see functions_v1.4.py)

```
##############################################################################
def calculate_price (sqft_living, decade, basement, zipcode, grade, waterfront, view , sqft_lot,
                     mean_price, coef_df=coef_df, output='yes'):

    if waterfront == 'NO' or not waterfront:
        waterfront = 0
    else:
        waterfront = 1

    b0,b1,b2,b3,b4,b5,b6,b7,b8 = get_coeff( decade, zipcode, grade, waterfront, view, coef_df)
    y=round( np.exp(b0 + b1*np.log(sqft_living) + b2*np.log(sqft_lot) + b3*basement + b4*waterfront + b5*grade + b6 + b7 + b8) )

    if output == 'yes': y=y*(mean_price/mean_price_2014_2015)
    print('{:,.0f}'.format(y))
    return y,b0,b1,b2,b3,b4,b5,b6,b7,b8
#####################################################################################
```

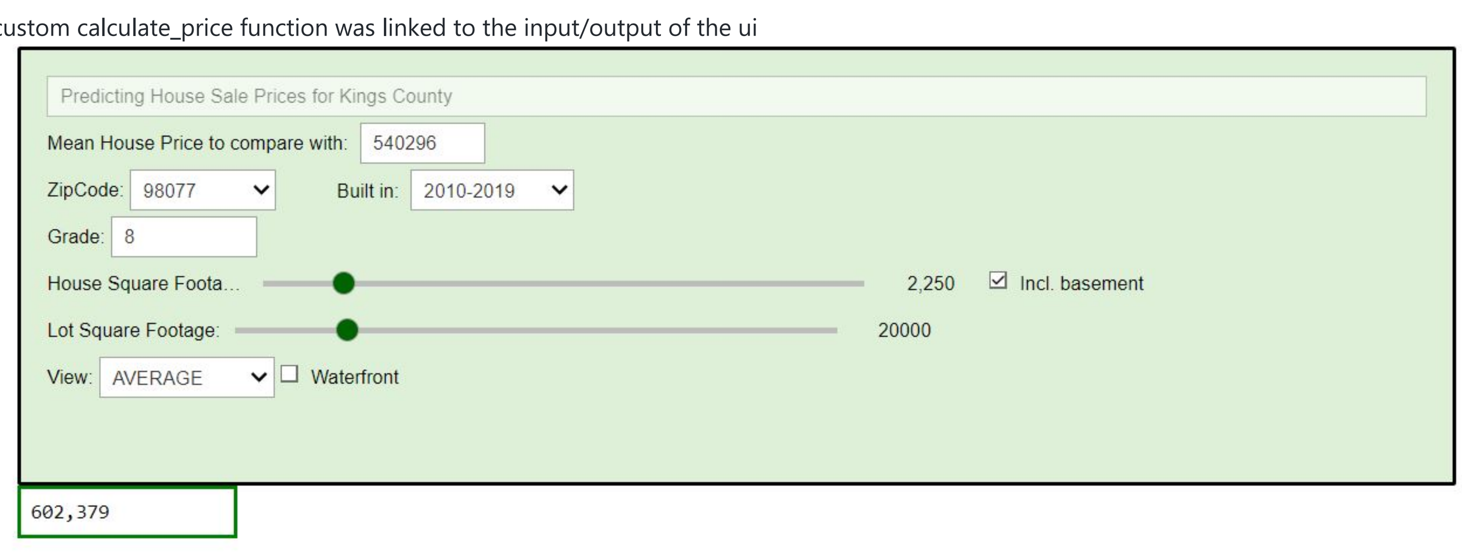
Creating UI forms

```
## Importing Widgets Forms
%run code/Build_Forms_v1.4.py

inp={ 'view':viewW,'waterfront':waterW,
    'zipcode': zipW, 'decade':decadeW, 'grade':gradeW, 'basement':basementW, "mean_price":meanW,
    'sqft_living':livingW,'sqft_lot':lotW }

output = widgets.interactive_output(calculate_price, inp )
output.layout={'border': '3px solid green', 'width':'150px'}

ui = widgets.VBox([form, output])
```

Building a Front End Tool:

- ipywidgets were used to create custom ui forms (Build_Forms_v1.4.py)

* custom calculate_price function was linked to the input/output of the ui



Testing

We made sure the tool works as expected:

- Multiple comparissons of predicted data against the actual data
- Predicted price is within 90-110% of actual price (houses newer than 1980)
- Predicted price is within 87-113% of actual price (houses older than 1980)

More deails about regression testing in "analysis_and_regression/Regression Tests.ipynb"

Conclusions

Considerations and Limitations:

- The tool can be effective to estimate base price for known features
- In the future a model should be re-trained with more up-to-date data
- The presented prototype will be greatly improved by more advanced modeling