Introduction

Amy Amador, Domeda Duncan, Randeep Singh, Shruti Singh

Prediction of Housing Prices Using Regression Modeling

Provides valued benefits to:
• homeowners
• realtors
• developers

to maximize profits and minimize costs [using a data set containing home sales between]:

May 2014 and May 2015 in King County, Washington

-multiple predictive models were developed to predict housing prices [based on the characteristics of the houses sold].
Predictive Modeling

What we used:
- Linear Regression
- Ridge Regression
- Decision Tree
- Extra-Trees
- and Random Forest models were compared

The random forest model provided the best Root Mean Square Error (RMSE)

Although a black box model, random forest modeling also provided a ranking of the most important features used to increase housing prices.
Project Overview

- **Real estate** professionals, **developers** and **homeowners** seek to predict housing prices to maximize profits and minimize cost for acquisition.

- **Corporate** and **municipal organizations** seek to predict property values when planning new housing developments.

Understanding the relative **importance** of **housing characteristics** on **purchase price** can guide:
- Renovation Decisions
- New Home Purchase Decisions
- Selling Price Decisions
What We Used
What We Used

Kaggle - House Sales in King County

Our dataset [is multivariate in nature]:

- **21,613** observations
- **19** features
  - i.e. size, age, condition, location, etc.

Of the **19** features, there are **13** quantitative variables [of which] are:
- 5 discrete
- 8 continuous
What we Used \{x\}

Kaggle

The remaining 6 variables are categorical

- 1 binary
- 3 ordinal
- 1 times series
- 1 standard categorical

Range of response variable (housing prices) = $75,000 to $7,700,000
The Challenge

• to analyze and transform the dataset to regress relevant features on price

• to build a model for housing sales price prediction

Using a non-time-series regression, the **primary goal** of the model is to predict housing sales price.

The **secondary goal** is to obtain inferential insights on the relative relationship of housing characteristics on sales price.
Methodology

Data transformation
Data then fit using different models and the performance of the models evaluated.

Selected model with the lowest RMSE selected and tuned to reduce the test MSE

Steps
- Data Preparation and Pre Processing
- Data Visualization
- Feature Engineering/Extraction
- Training and Test Split
- Model Fitting
- Performance Validation
- Hyper Parameter Tuning
- Pipeline Fitting
The Features

<table>
<thead>
<tr>
<th>Fields</th>
<th>Description</th>
<th>Data Type</th>
<th>Variable Type</th>
<th>Variable SubType</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>A notation for the house</td>
<td>Numeric</td>
<td>categorical</td>
<td></td>
</tr>
<tr>
<td>date</td>
<td>Date house was sold</td>
<td>String</td>
<td>quantitative</td>
<td>discrete</td>
</tr>
<tr>
<td>price</td>
<td>Price is prediction target</td>
<td>Numeric</td>
<td>quantitative</td>
<td>discrete</td>
</tr>
<tr>
<td>bedrooms</td>
<td>Number of Bedrooms/House</td>
<td>Numeric</td>
<td>quantitative</td>
<td>discrete</td>
</tr>
<tr>
<td>bathrooms</td>
<td>Number of bathrooms</td>
<td>Numeric</td>
<td>quantitative</td>
<td>discrete</td>
</tr>
<tr>
<td>sqft_living</td>
<td>Square footage of the home</td>
<td>Numeric</td>
<td>quantitative</td>
<td>continuous</td>
</tr>
<tr>
<td>sqft_lot</td>
<td>Square footage of the lot</td>
<td>Numeric</td>
<td>quantitative</td>
<td>continuous</td>
</tr>
<tr>
<td>floors</td>
<td>Total floors (levels) in house</td>
<td>Numeric</td>
<td>quantitative</td>
<td>discrete</td>
</tr>
<tr>
<td>waterfront</td>
<td>House which has a view to a waterfront</td>
<td>Numeric</td>
<td>categorical</td>
<td>binary</td>
</tr>
<tr>
<td>view</td>
<td>Has been viewed</td>
<td>Numeric</td>
<td>categorical</td>
<td>ordinal</td>
</tr>
<tr>
<td>condition</td>
<td>How good the condition is (Overall)</td>
<td>Numeric</td>
<td>categorical</td>
<td>ordinal</td>
</tr>
<tr>
<td>grade</td>
<td>Overall grade given to the housing unit, based on King County grading system</td>
<td>Numeric</td>
<td>categorical</td>
<td>ordinal</td>
</tr>
<tr>
<td>sqft_above</td>
<td>Square footage of house apart from basement</td>
<td>Numeric</td>
<td>quantitative</td>
<td>continuous</td>
</tr>
<tr>
<td>sqft_basement</td>
<td>Square footage of the basement</td>
<td>Numeric</td>
<td>quantitative</td>
<td>continuous</td>
</tr>
<tr>
<td>yr_built</td>
<td>Built Year</td>
<td>Numeric</td>
<td>quantitative</td>
<td>discrete</td>
</tr>
<tr>
<td>yr_renovated</td>
<td>Year when house was renovated</td>
<td>Numeric</td>
<td>quantitative</td>
<td>discrete</td>
</tr>
<tr>
<td>zipcode</td>
<td>Zip code</td>
<td>Numeric</td>
<td>categorical</td>
<td></td>
</tr>
<tr>
<td>lat</td>
<td>Latitude coordinate</td>
<td>Numeric</td>
<td>quantitative</td>
<td>continuous</td>
</tr>
<tr>
<td>long</td>
<td>Longitude coordinate</td>
<td>Numeric</td>
<td>quantitative</td>
<td>continuous</td>
</tr>
<tr>
<td>sqft_living15</td>
<td>Living room area in 2015 (implies--some renovations). This might or might not have affected the lot size area.</td>
<td>Numeric</td>
<td>quantitative</td>
<td>continuous</td>
</tr>
<tr>
<td>sqft_lot15</td>
<td>Lot size area in 2015 (implies--some renovations)</td>
<td>Numeric</td>
<td>quantitative</td>
<td>continuous</td>
</tr>
</tbody>
</table>
Training and Test Split

The data was transformed into training and testing sets using train test split from the Sklearn library:

- **90%** - Training Set (randomly designated)
- **10%** - Test Set

K-fold cross validation would be used to approximate the RMSE on the test data.
SEATTLE HOUSING
A DEEP ANALYSIS
PROJECT OVERVIEW
ABOUT
DATA VISUALIZATION
CONCLUSION & QUESTIONS
THANK YOU!
Geographic Distribution of Housing Prices In King County
Heat maps were used to gauge the strength of correlation between the variables.
Pairplot

The relationships between predictor variables were analyzed using scatterplots and a correlation matrix.
Box plots were used to explore the distribution and density of categorical variables.
Feature Engineering/Feature Extraction

Two new binary categorical variables were generated from existing continuous variables and added to the dataset to measure the effect of these attributes on sales price. The two variables are:

- **basement** – designating whether the house has a basement
- **renovated** – designating whether the house had been renovated in 2015

Three columns were dropped from the data frame:
- ID - unique identifier
- Date - data falls within May 2014 - May 2015
- Zipcode - too large (70 unique zip codes) to encode for linear regression
Feature Engineering using custom transformers.

```python
# Feature Engineering
# Custom transformer to add new features (basement and renovated) and create dummy variables for categorical columns

from sklearn.base import BaseEstimator, TransformerMixin

class CustomAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self, sqft_basement, yr_renovated, cat_attrs):
        self.sqft_basement = sqft_basement
        self.yr_renovated = yr_renovated
        self.cat_attrs = cat_attrs

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        X['basement'] = X[self.sqft_basement].apply(lambda val: 1 if val > 0 else 0)
        X['renovated'] = X[self.yr_renovated].apply(lambda val: 1 if val > 0 else 0)
        for cols in self.cat_attrs:
            value = pd.get_dummies(X[cols])
            value = value.add_prefix('{}_'.format(cols))
            X = X.join(value)
        return X

# Custom transformer to select relevant columns

class DataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X[self.attribute_names]
```
Pipeline

```python
# Full pipeline for feature engineering
from sklearn.pipeline import Pipeline

pipe = Pipeline([
    ('AttributeAdder', CustomAttributesAdder('sqft_basement', 'yr_renovated', ['waterfront', 'view', 'condition', 'basement', 'renovated', 'grade'])),
    ('selector', DataFrameSelector(attribs)),
])

# Prep the training set to feed into the model
housing_prepared = pipe.fit_transform(housing)
```
Model Fitting / Performance Validation

```python
In [114]:
# 5 FOLD CROSS VALIDATION ON ALL THE MODELS

lin_scores = cross_val_score(lin_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
lin_mse_scores = -np.sqrt(lin_scores)

tree_scores = cross_val_score(tree_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
tree_mse_scores = -np.sqrt(-tree_scores)

forest_scores = cross_val_score(forest_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
forest_mse_scores = -np.sqrt(-forest_scores)

extra_scores = cross_val_score(extra_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
extra_mse_scores = -np.sqrt(-extra_scores)

Results for Linear Regression model :
scores : [ 195299.60692607 172945.13146024 107569.8059208 220300.84791126
         192189.19444793]
min error : 172945.131460
max error : 220300.847911
mean error : 193784.265853
standard deviation in error : 15538.7372318

Results for Decision Tree model :
scores : [ 207084.28473941 164564.56206662 191638.26590036 218976.19241985
         187041.36484715]
min error : 164564.562067
max error : 218976.19242
mean error : 193868.917999
standard deviation in error : 18521.177403

Results for Random Forests model :
scores : [ 149588.79453086 124974.57854582 135865.6999865 157360.49987134
         149250.8102955]
min error : 124974.578546
max error : 157360.499871
mean error : 141602.076394
standard deviation in error : 11153.9163618

Results for Extra Trees model :
scores : [ 142154.312630 112787.16600015 135430.4351845 166319.70132689
         135159.6192767]
min error : 112787.166009
max error : 166319.701327
mean error : 140270.289564
standard deviation in error : 14650.858972
```
In [114]:

    # 5 Fold cross validation on all the models
    lin_scores = cross_val_score(lin_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
    lin_rmse_scores = np.sqrt(-lin_scores)

    tree_scores = cross_val_score(tree_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
    tree_rmse_scores = np.sqrt(-tree_scores)

    forest_scores = cross_val_score(forest_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
    forest_rmse_scores = np.sqrt(-forest_scores)

    extra_scores = cross_val_score(extra_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
    extra_rmse_scores = np.sqrt(-extra_scores)


Results for Linear Regression model:
scores: [195291.86989067 172945.13164624 187694.8859288 220800.04731226
  192189.39448793]
min error: 172945.131646
max error: 220800.047312
mean error: 193784.265853
standard deviation in error: 15530.7372318

Results for Decision Tree model:
scores: [207084.20473941 164564.56208662 191638.26590036 218976.19241985
  187041.36484715]
min error: 164564.562087
max error: 218976.19242
mean error: 193860.917999
standard deviation in error: 18522.177403

Results for Random Forests model:
Model Fitting / Performance Validation

```python
# 5 FOLD CROSS VALIDATION ON ALL THE MODELS

lin_scores = cross_val_score(lin_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
lin_mse_scores = np.sqrt(lin_scores)

tree_scores = cross_val_score(tree_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
tree_mse_scores = np.sqrt(tree_scores)

forest_scores = cross_val_score(forest_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=6)
forest_mse_scores = np.sqrt(forest_scores)

extra_scores = cross_val_score(extra_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
extra_mse_scores = np.sqrt(extra_scores)
```

Results for Linear Regression model :
scores : [ 195201.06099067 172945.13164624 187099.0.8059288 208000.04731126 192128.194443793]
min error : 172945.131646
max error : 208000.04731126
mean error : 193784.265853
standard deviation in error : 15530.7372318

Results for Decision Tree model :
scores : [ 207884.20473941 164564.56288662 191638.26590036 218976.1321985 187041.3648715]
min error : 164564.56288662
max error : 218976.1321985
mean error : 193860.917999
standard deviation in error : 18521.177403

Results for Random Forests model :
scores : [ 1495804.7543568 124974.57854512 135635.6909865 157360.49987134 149250.8120595]
min error : 112374.57854512
max error : 157360.49987134
mean error : 141602.076394
standard deviation in error : 11153.9163618

Results for Extra Trees model :
scores : [ 142154.312363 121767.1660915 135436.43551845 166219.70132689 135159.81932767]
min error : 111787.1660915
max error : 166219.70132689
mean error : 146730.285964
standard deviation in error : 14650.858971
Results for Linear Regression model:
scores: [ 195291.86989067  172945.13164624  187694.8859288  220800.04731226
         192189.39448793]
min error: 172945.131646
dmax error: 220800.047312
dmean error: 193784.265853
standard deviation in error: 15530.7372318

Results for Decision Tree model:
scores: [ 207084.20473941  164564.56208662  191630.26590036  218976.19241905
         187041.36484715]
min error: 164564.562087
dmax error: 218976.19242
dmean error: 193860.917999
standard deviation in error: 18522.177403

Results for Random Forests model:
scores: [ 149508.79453506  124974.57854582  135865.6969865  157360.49987134
         140250.81202959]
min error: 124974.578546
dmax error: 157360.499871
mean error: 141592.076394
standard deviation in error: 11153.9163618

Results for Extra Trees model:
scores: [ 142154.3126363  121787.16600915  135430.43551845  166319.70132689
         135159.81932767]
min error: 121787.166009
dmax error: 166319.701327
dmean error: 148170.286964
standard deviation in error: 14650.858972
26.

Model Fitting / Performance Validation

```python
In [114]: # 5 FOLD CROSS VALIDATION ON ALL THE MODELS

lin_scores = cross_val_score(lin_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
lin_mse_scores = np.sqrt(lin_scores)

mse_scores = cross_val_score(tree_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
tree_mse_scores = np.sqrt(mse_scores)

forest_scores = cross_val_score(forest_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
forest_mse_scores = np.sqrt(forest_scores)

extra_scores = cross_val_score(extra_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=5)
extra_mse_scores = np.sqrt(extra_scores)

Results for Linear Regression model :
scores : [ 195201.86069067 172945.13164624 187094.80592885 208000.04731125 192159.19448793]

min error : 172945.11646
max error : 208000.047311
deviation in error : 15530.7372318

Results for Decision Tree model :
scores : [ 207084.28473941 164564.56520042 191638.26590036 218976.19241985 187044.36484715]

min error : 164564.56520042
max error : 218976.19241985
deviation in error : 10521.177403

Results for Random Forests model :
scores : [ 195201.79453586 124974.57854382 135865.6995865 157360.49987134 149250.81202939]

min error : 124974.57854382
max error : 157360.49987134
deviation in error : 11151.9163618

Results for Extra Trees model :
scores : [ 142154.3128363 121767.16600815 135430.43551845 166319.70112689 135159.81327677]

min error : 111787.16600815
max error : 166319.70112689
deviation in error : 14658.858972
```
Hyper Parameter Tuning

In [129]: from sklearn.model_selection import GridSearchCV

    param_grid = [
        {'n_estimators': [100, 120, 140], 'max_features': [22, 27, 32]},
    ]

    forest_reg = RandomForestRegressor()
    grid_search = GridSearchCV(forest_reg, param_grid, cv=10, scoring='neg_mean_squared_error')
    grid_search.fit(housing_prepared, housing_labels)

Best combination of parameters of Random Forests: {'max_features': 32, 'n_estimators': 100}
Best Estimator: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                                 max_features=32, max_leaf_nodes=None, min_impurity_split=1e-07,
                                 min_samples_leaf=1, min_samples_split=2,
                                 min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
                                 oob_score=False, random_state=None, verbose=0, warm_start=False)
Results

Starting RMSE: 141592.076
After Tuning RMSE: 1302381.72

Result: 7.9% Improvement in model Performance
Future Considerations

- Decrease the number of levels in the categorical variables
- Add median household income using a second data source based on zip codes
- Add data to account for environmental/economic changes and time series data
Questions
Thank You!
Customize the Design
Create your own layout and change colors
Customize the Design
Create your own layout and change colors
Customize the Design