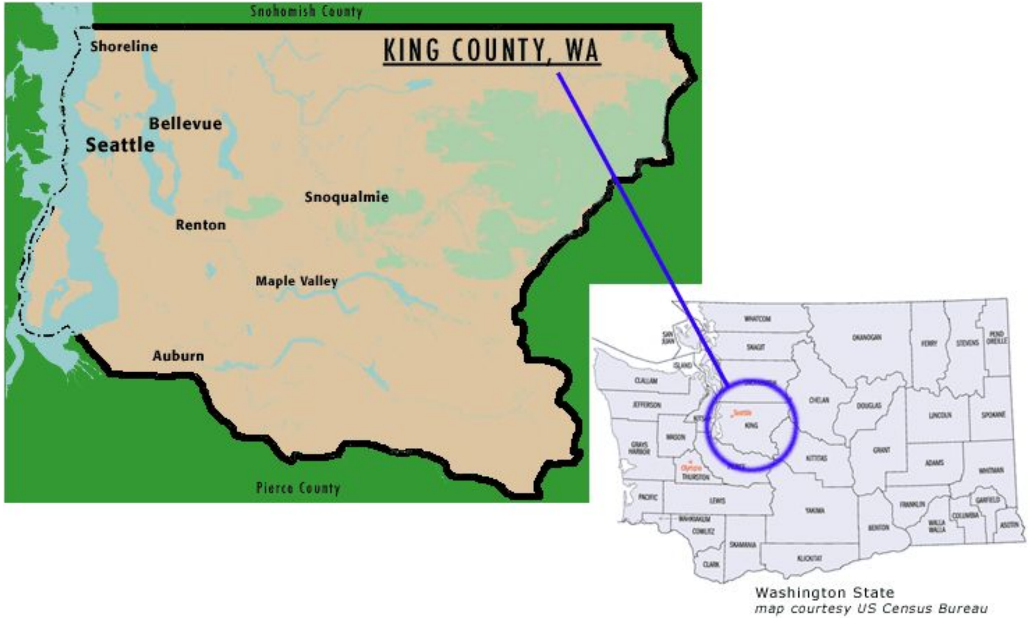
House Pricing Prediction

Machine Learning – Supervised Learning

Hayley
Mohamad
Mara

Dataset House sale prices in King County (May 2014–May 2015)



- Our goal is to predict house price
- Why interesting:
    - real-world dataset
    - interpretable features
    - mix of categorical/numerical
    - 18 features: size, location, the number of rooms etc

Project roadmap

Step 01

Data Cleaning
Feature understanding

Step 02

Feature selection
Feature engineering
Train baseline model

Step 03

Model evaluation
Model improvement
Hyperparameter tuning

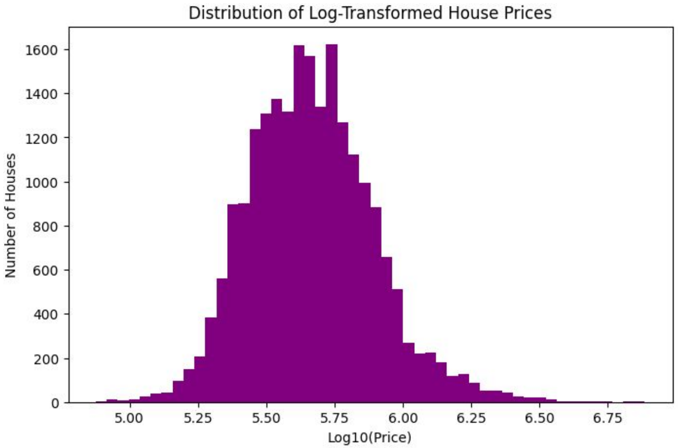
Step 04

Best-performing model

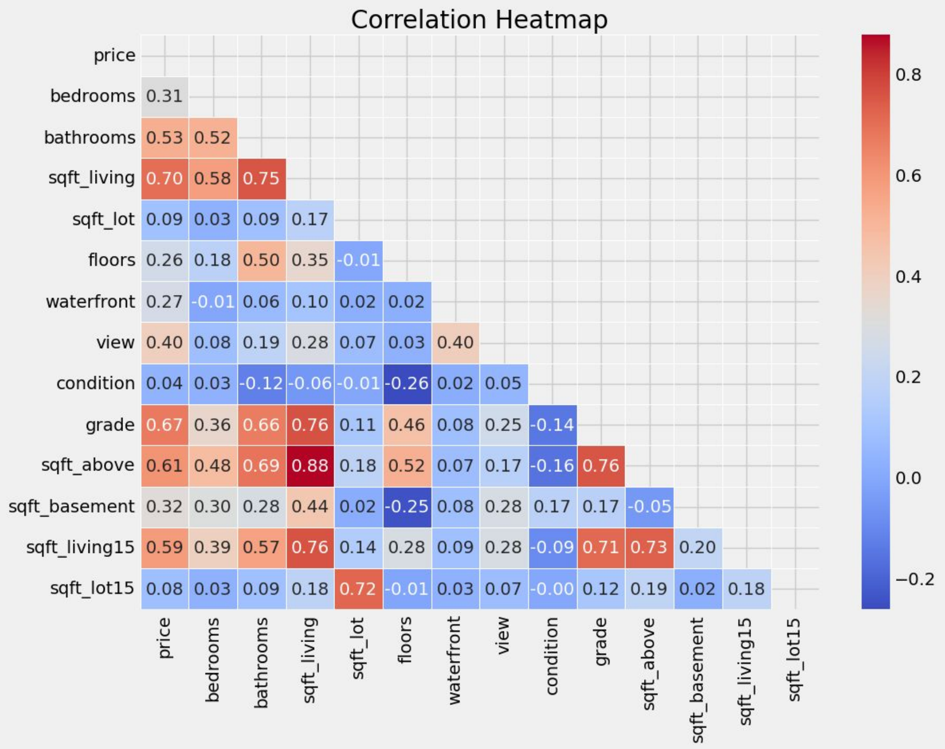
Data Understanding & Cleaning

Data Cleaning Steps:

1. no missing values
2. date -> datetime format
3. removal impossible values
   a. bed/bath = 0
   b. sale_year < yr_built



Feature Understanding



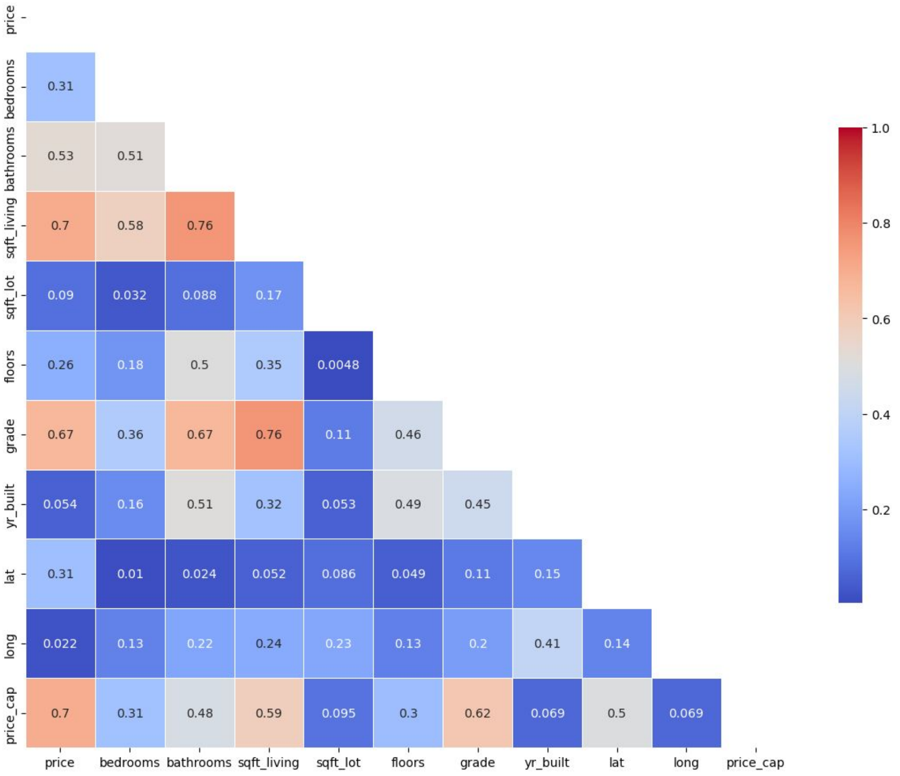
Correlation Heatmap of all features

- Highest correlation (0.88) between *sqft_living* and *sqft_above*
- Strong correlation between *sqft_living/grade* and *price*

Feature Selection

Feature	Reason to drop
ID	unique values
waterfront	only very small dataset
view	only very few unique values
ZIP	lat & long as area features
yr_renovated	we have the grade
condition	grade & yr_build
sqft_living15 & sqfft_lot15	sqft_living & sqft_lot recorded in 2015
sqft_above & sqft_basement	we have sqft_living
date	only one year of data

Correlation matrix after feature selection



correlation of selected features with target (price)

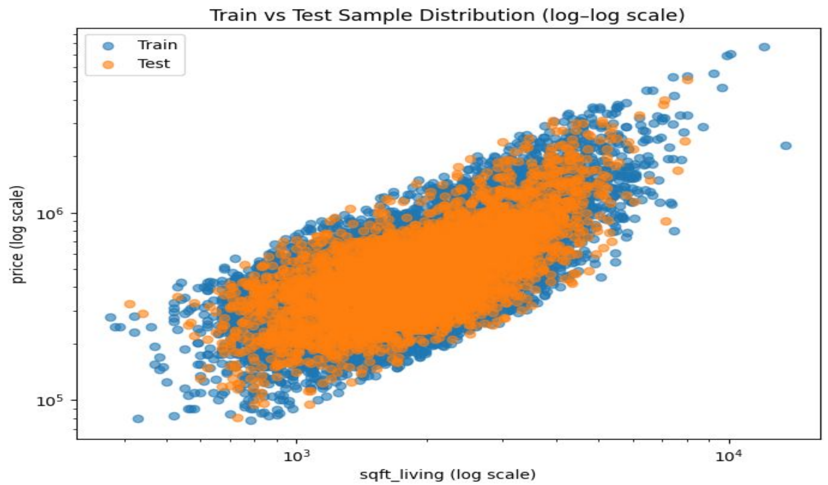
- Relatively strong correlation with sqft_living(0.7), grade (0.67)

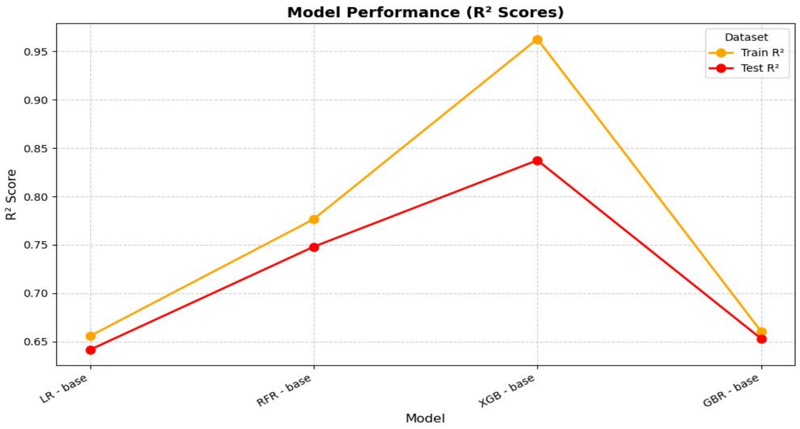
- Moderate with structural features (bathrooms(0.53), bedrooms(0.3), floors)

- Moderate or weak correlation location(lat(0.31), long(0.022))

Baseline Model + Comparison

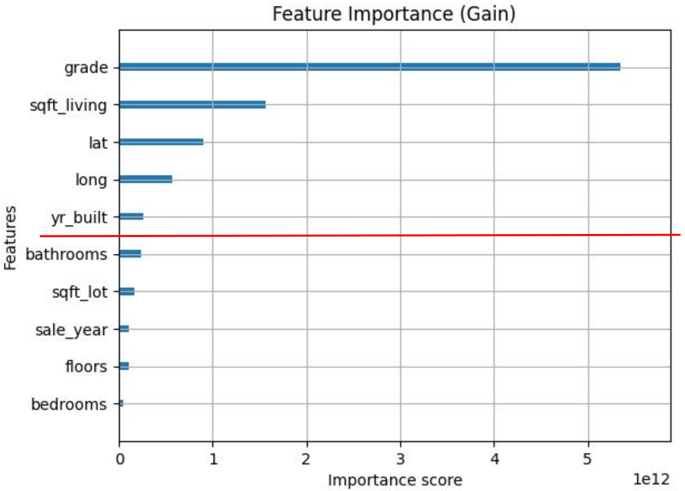
- Train/Test split 80/20, target: price
- 4 regressors:
    - Linear Regression
    - Random Forest
    - XGBoost
    - Gradient Boost
- Models assessed
- Assessment: R-Squared value.





Iteration Features & Hyperparameters – XGB

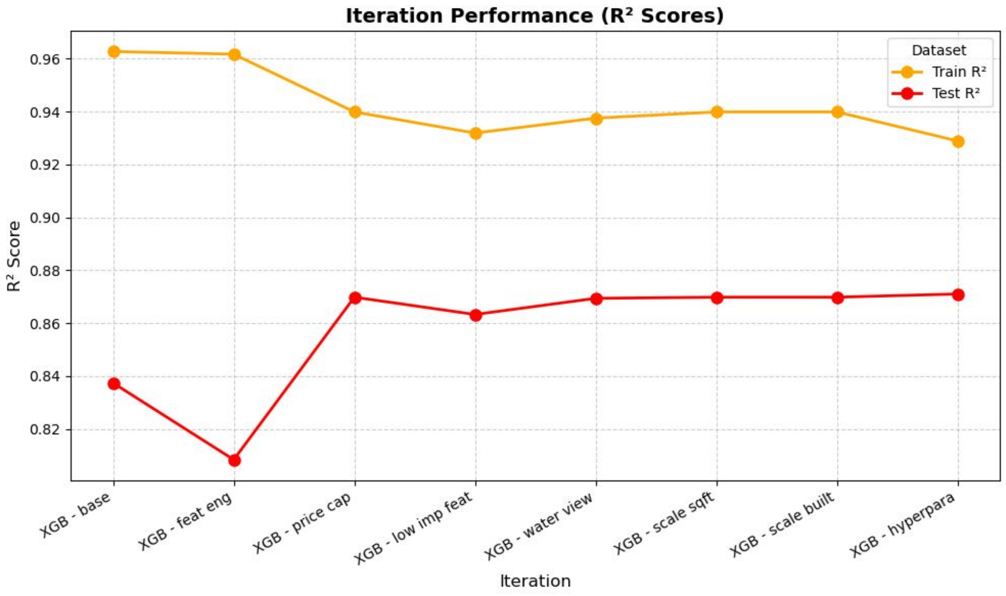
- Capping price (upper Q3)
- Feature engineering (age = sale_year - yr_built)
- Dropping low-impact features
- Adding view/waterfront back
- Standardization (sqft_living, year)
- Randomized Search CV for Hyperparameters



Feature Importance:

- Gain Importance: measures contribution of feature to model's accuracy.
- Quantifies average reduction in loss when feature is used in split.

Iteration Performance Features – XGB



- Trade-off between training and test performance.
- Possible overfitting for the base model.
- Adding price cap, i.e removing outliers ⟹ significant improvement.
- Model based on top features according to gain had worse training performance.
- Hyperparameter testing brought train & test closer.

Final Model Performance

```
Train MAE: 26625.98
Test MAE: 35764.96
Train R²: 0.94
Test R²: 0.88
```

Best Parameters Found (XGBoost)

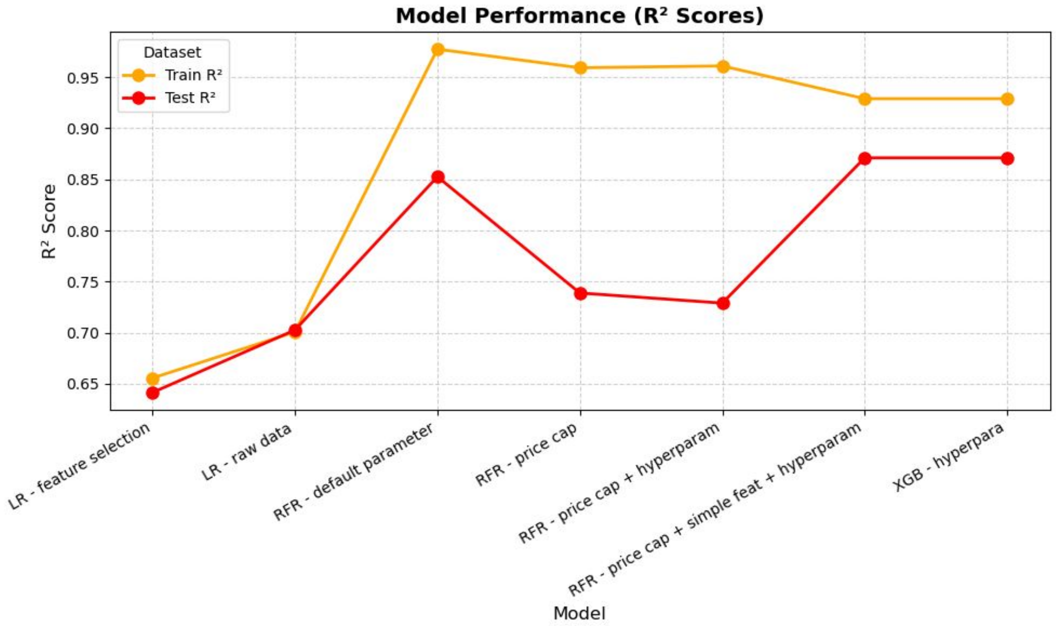
Parameter	Value
subsample	0.6
reg_lambda	1
reg_alpha	0
n_estimators	300
max_depth	7
learning_rate	0.05
gamma	0.1
colsample_bytree	0.6

final reduced dataset + price cap:

sqft_living	grade	yr_built	lat	long
2380	8	2010	47.7170	-122.020
3190	10	1999	47.5115	-122.246
1730	8	1994	47.2621	-122.308
1870	7	1977	47.1985	-122.001
2850	7	1959	47.4859	-122.205

Learnings

- Early aggressive feature dropping *hurt* performance — some columns that seemed redundant added signal.
- Model selection (try early, tune later) is more efficient than heavy pre-filtering.
- XGBoost slightly best overall, but Random Forest nearly matched it — showing simplicity can win.
- Probably better to use log(price) than capQ3(price).



Thank You!