San Francisco Airbnb Listings

ISyE 7406 Data Mining & Statistical Learning
Project Presentation

Course Project Team 12
Pei-Fan Hsieh
Sachin Shetty
Yogesh Vazirani
Winnie Zhang
Jiazhen Zhu

Project Introduction and Objectives

Project introduction:

- Airbnb provides a platform for hosts to accommodate guests with short-term lodging and tourism-related activities. Guests can search for lodging using filters such as lodging type, location, price, review scores, etc. Hosts provide prices and other details for their rental or event listings, such as the home type, rules, and amenities. Airbnb also provides a platform for hosts and guests to leave and share reviews about their experience.
- For this project, we used the event listings dataset collected by Airbnb and focused our studies on the city of San Francisco.

Project objectives:

- The objective of our project is to **implement data mining and statistical learning methods to effectively predict the most appropriate listing price for a typical vacation rental in San Francisco**.
  - Suggest best model(s)
  - Suggest listing feature(s) to be included to gain competitive listing price
  - Predict potential listing price

Data Source and Variable Description

Data source:

The original dataset is from Kaggle.com. https://www.kaggle.com/stevezhenghp/airbnb-price-prediction

Variable description:

	Variable	Description	Data type
Response variable	log_price	Rental price	num
Predictors	accommodates	The number of guest accommodations	int
	bathrooms	The number of bathrooms	num
	cleaning_fee	Whether guest needs to pay cleaning fee	Int
	host_identity_verified	Whether the host identity is verified	Int
	instant_bookable	Can the property be booked instantly or not	int
	latitude	The latitude location of the property	num
	longitude	The longitude location of the property	num
	number_of_reviews	The number of reviews given by guests	int
	review_scores_rating	Satisfaction rating given by guests	num
	bedrooms	The number of bedrooms	num
	count_amenities	Count of amenities	int
	property_type_Condominium	Whether the property type is Condominium	int
	property_type_House	Whether the property type is House	int
	property_type_Loft	Whether the property type is Loft	int
	room_type_Private.room	Whether the room is private	int
	room_type_Shared.room	Whether the room is shared	int
	cancellation_policy_moderate	Whether the cancellation policy is moderate	int
	cancellation_policy_strict	Whether the cancellation policy is strict	int

Data Wrangling and Data Splitting

❖ Data wrangling and pre-processing:

Steps performed to prepare the data for analysis:

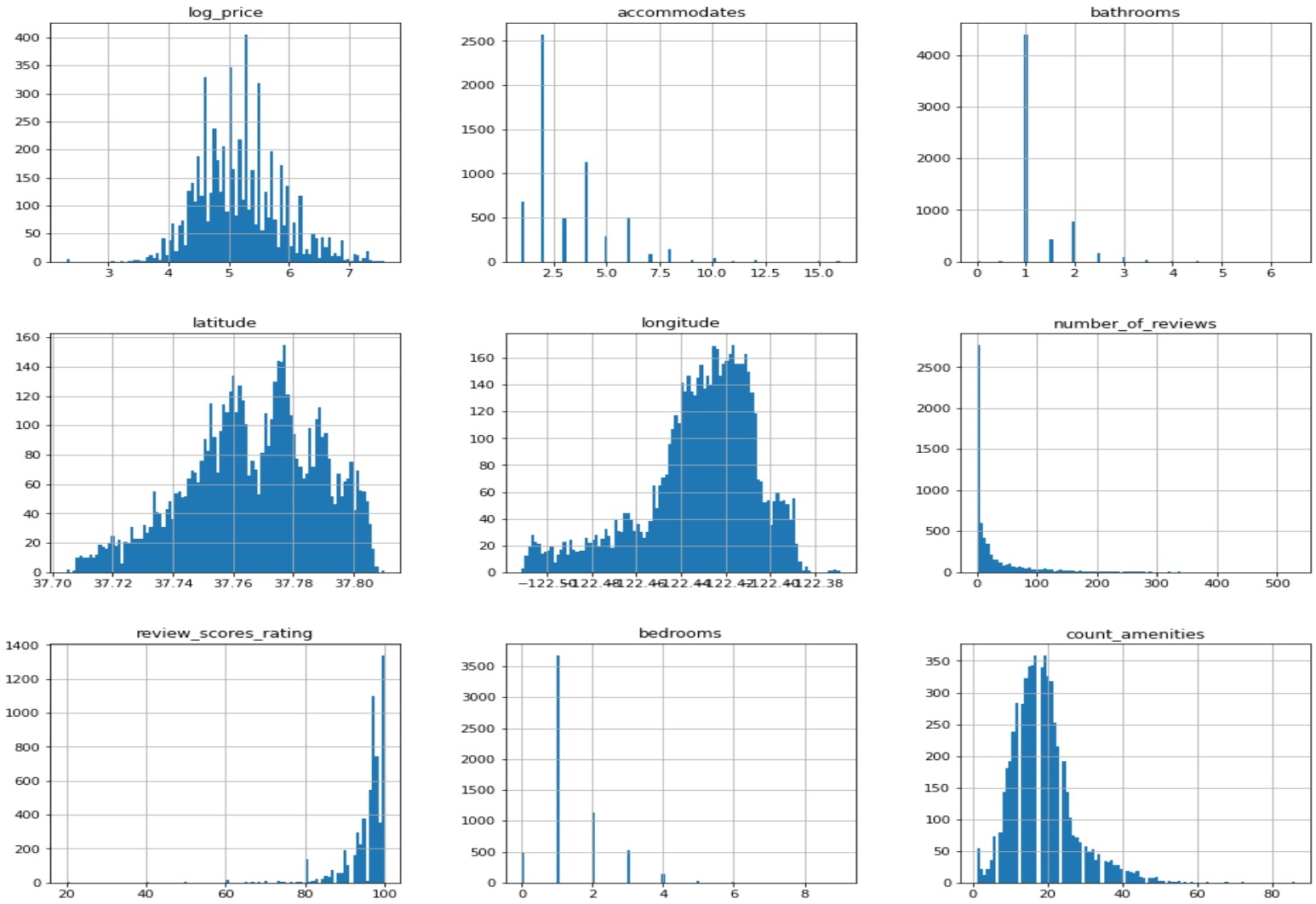
- Filtered the data to **focus on San Francisco**
- Identified and extracted **irrelevant features** from the dataset
- Identified columns with **missing values**, performed imputation, and filled in the missing data
- Identified categorical columns with binary values and transformed using scikit-learn's "LabelEncoder"
- Identified multivalued categorical columns and transformed using pandas "get_dummies" into indicator variables
- Cleaned up and transformed the individual list of amenities for each property to a "count of amenities" column

❖ Data splitting:

We performed a **"80:20" random split**

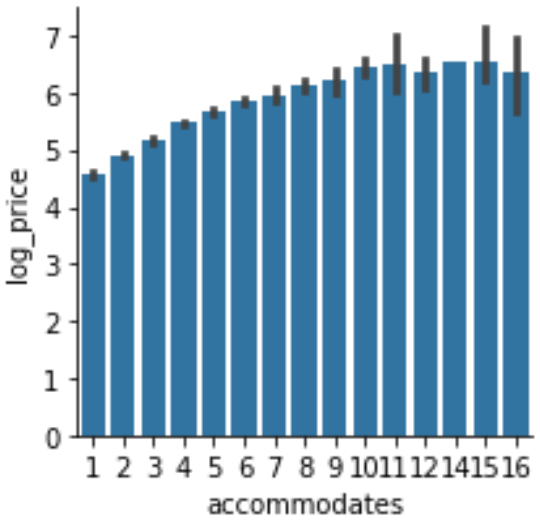
- 80% of the data used for training set
- The rest of the 20% used for testing set

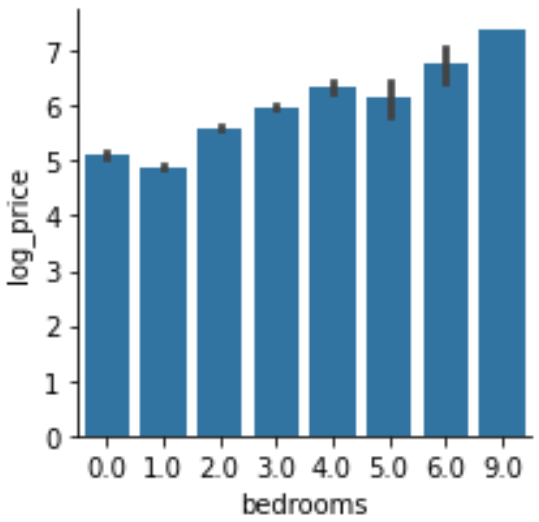
Exploratory Data Analysis: Histogram



- We used the **Matplotlib** library to plot histograms for exploring the data distribution.
  - Log_price is normally distributed.
  - **Skewness:**
    - Review score rating is left-skewed.
    - Number of accommodates, number of reviews, and count of amenities are right-skewed.
    - Thus, we **performed normalization** to build models.
  - We have many listings with Bathrooms=1 and Bedrooms=1.

Exploratory Data Analysis: Histogram and Boxplot





accommodates	log_price
14	6.551080
15	6.536302
11	6.513996
10	6.458519
16	6.382590

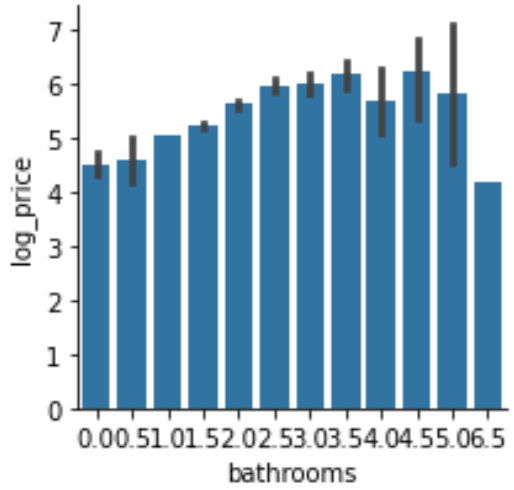
bedrooms	log_price
9.0	7.377759
6.0	6.790919
4.0	6.330405
5.0	6.141518
3.0	5.984163

- We used the **Histograms and Boxplots** to explore the correlation between response variable and predictors.

  - **Accommodates:**
    - log_price increases as accommodates increases.
    - There is a small fluctuation at accommodates=16.
    - There are more outlier when accommodates become larger, such as 11, 12, 15, and 16.

  - **Bedrooms:**
    - log_price increases as bedrooms increases.
    - There is a small fluctuation at bedrooms=5.
    - There are more outlier when bedrooms become larger, such as 5.0 and 6.0.

Exploratory Data Analysis: Histogram and Boxplot



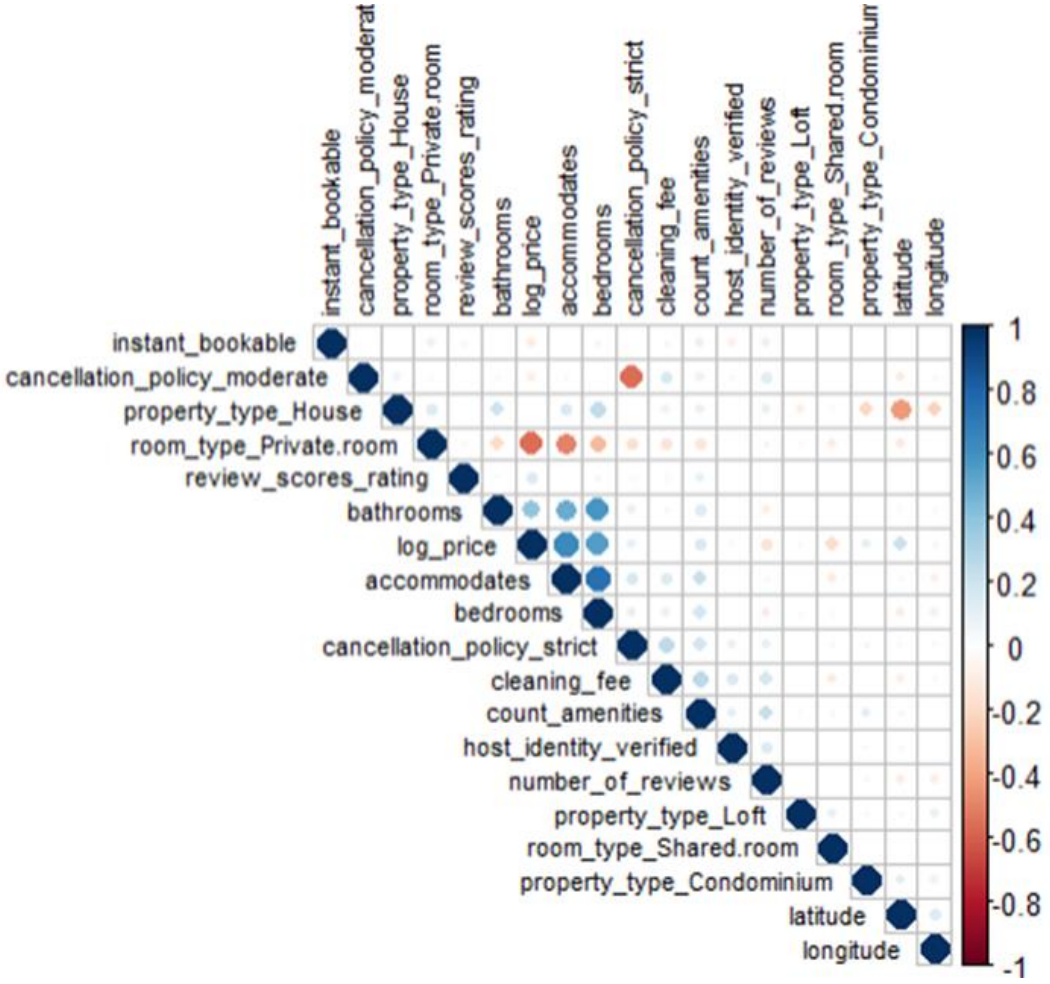
bathrooms	log_price
4.5	6.232917
3.5	6.186974
3.0	6.017105
2.5	5.977972
5.0	5.811338

bathrooms	accommodates	acco/bath	log_price
4.0	12	3.000000	7.377759
2.5	16	6.400000	7.207860
3.0	15	5.000000	7.130899
5.0	10	2.000000	7.090077
3.0	11	3.666667	7.047517

- We used the **Histograms and Boxplots** to explore the correlation between response variable and predictors.

- **Bathrooms:**
  - log_price increases as bathrooms increases.
  - **After bathrooms=4.5, the log_price decreases as bathrooms increases**.
  - There are more outlier when bathroom become larger, such as 0.5, 4, 4.5, and 5.
  - Potential Reasons for the fluctuation trend: More bathrooms mean more space and are more expensive, but **people might not need more than 4.5 bathrooms**.
  - Best price matrix: We can find the best price matrix happens when **accommodates : bathrooms = 3 : 1**.

Exploratory Data Analysis: Correlation Matrix

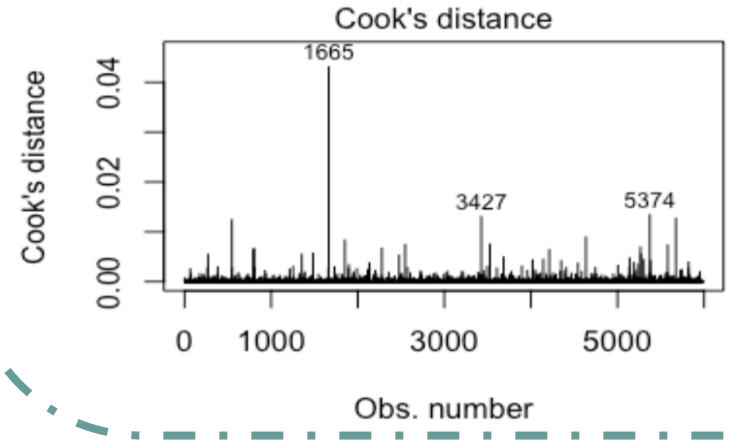


- We used the **Correlation Matrix** to explore the correlation between response variable and predictors.
  - Log_price **significantly correlates with near all predictors**, except cleaning_fee and property_type_House.
  - Log_price has **strong positive correlation with accommodates** (r=0.63).
  - Log_price has **moderate positive correlation with bedrooms** (r=0.55) and **bathrooms** (r=0.39).
  - Log_price has **moderate negative correlation with room_type_Private.room** (r=-0.55).

Exploratory Data Analysis: Cook's Distance and VIF

Outlier and Influential

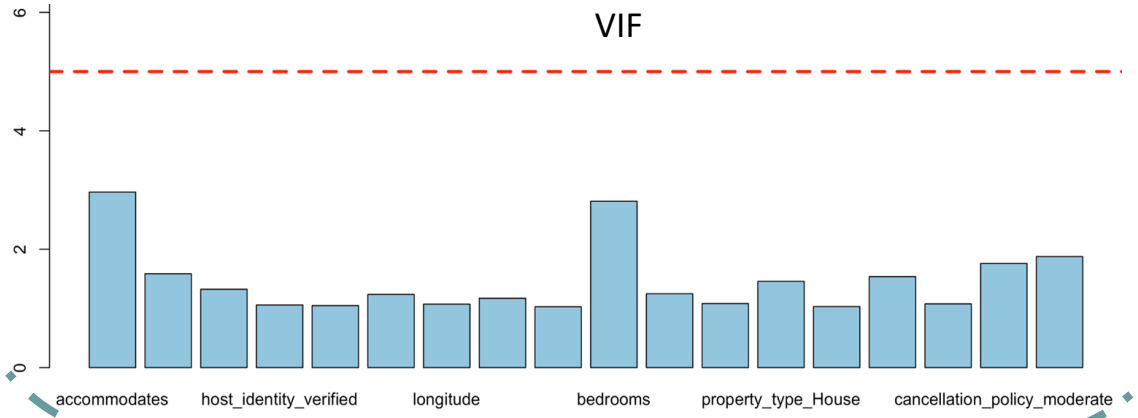
- We used the **Cook's distance (D_i's)** to identify outliers and influential points that affect our regression models.
  - **Outliers:** Observations 1665, 3427, and 5374 appear to be the outliers in our data. However, if we set Di >1 as outliers, there would be **no outliers** in our data.
  - **Influential points:** After examination, the presence of data points 1665, 3427, and 5374 did not change our modeling results. Thus, we **kept all observations**.



	CookD
1665	0.043064035
5374	0.013405998
5674	0.012673153

Multicollinearity

- We used the **Variance Inflation Factor (VIF)** to explore the collinearity among predictors.
  - **Multicollinearity:**
    - As a rule of thumb, a VIF>10 suggests high multilinearity.
    - Since none predictors have VIF>10 (VIF threshold), **the multicollinearity is negligible** in the model and should not affect our inferences.
    - Hence, we **kept all the predictors**.



Models: Multiple Linear Regression (MLR)

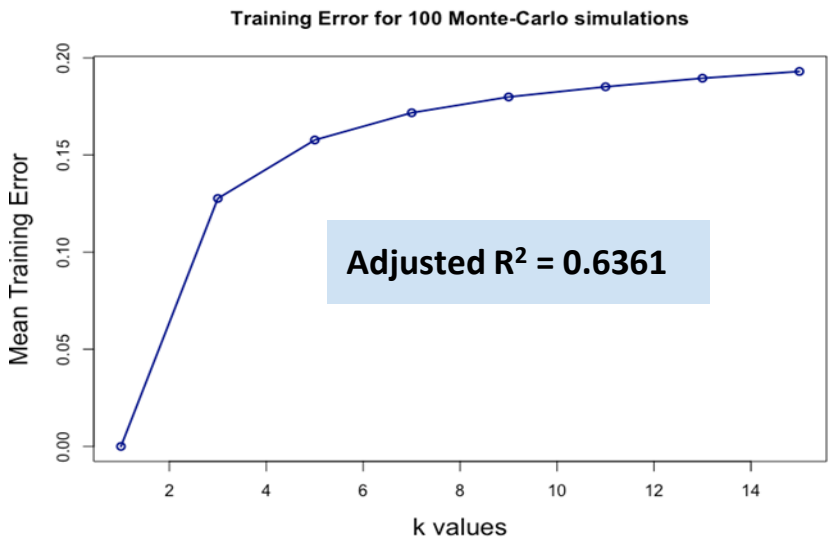
- MLR Models Comparison

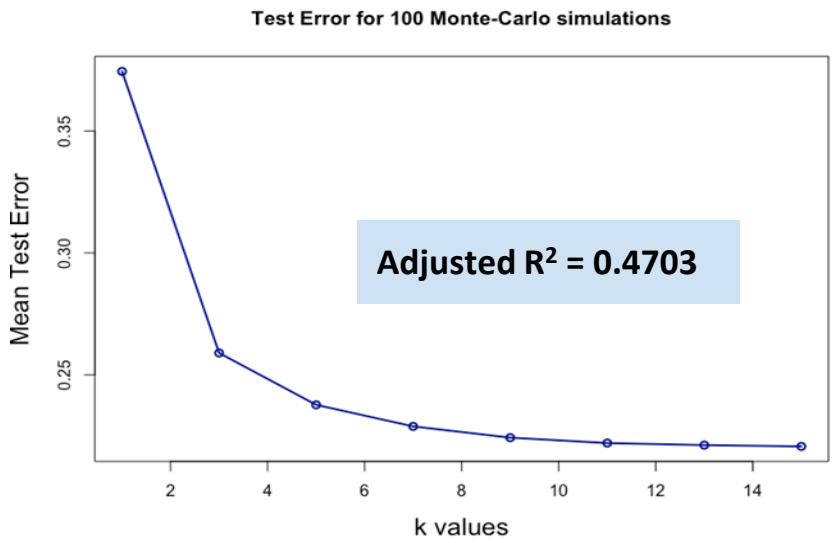
Model	Methods	Tuning parameter	Adjusted R^2	Training Error	Testing Error
1	MLR	None	0.5986	0.1856	0.1868
2	MLR with best subset k=18	Exhaustive search for best subset selection by: ▪ minimize **Residual sum of squares (RSS)**	0.5986	0.1856	0.1868
3	**MLR with best subset k=17**	Exhaustive search for best subset selection by: ▪ maximize **Adjusted R^2** ▪ minimize **Mallows' Cp** ▪ minimize **BIC**	**0.5986**	**0.1856**	**0.1867**
4	MLR with best subset k=5	Exhaustive search for best subset selection by: ▪ the maximize decreasing **Residual sum of squares (RSS)**	0.5641	0.2021	0.2028
5	MLR with Stepwise AIC	▪ Stepwise AIC	0.5986	0.1856	0.1868

- We built 5 **MLR models** to evaluate the model performance:
  - **Methodology:**
    MLR uses the methods of least squares (MLS) to find the β's that minimizes RSS
  - **Model and variable selection:**
    - **Analytics measures:**
      - model 2: built by minimizing RSS
      - model 3: built by maximizing adjusted R^2 or minimizing Mallow's Cp or BIC
      - model 4: built by maximizing the decreasing of RSS
    - **Sequential Search Algorithm:**
      - model 5: built by stepwise AIC
  - **Model performance:**
    With the largest adjusted R^2 and the smallest testing error, **MLR with best subset k=17 performs best**

Models: K-nearest Neighbors (KNN) Regression

- KNN regression is a **non-parametric method** that approximates the association between response and independent variables by averaging the observations in the same neighborhood.
- The tuning parameter is **K** (**the number of neighbors**). We first trained our model with several values of K and then ran Monte Carlo simulation to derive the average training/testing errors and determine the best K value using the **elbow method**.
- Model Results Summary:
  - **Lowest training error is 0 at K=1**. But this will **lead to model overfit with bias of 0 and high variance**.
    The optimal value of K is possibly 7. An increase in K only marginally worsens the training error.
  - **Testing error is 0.2288 at K = 7**. This can be taken as the optimal value. An increase in K above this value does not provide significant improvement in reducing the testing error.





K	Training Error	Testing Error
1	0.0000	0.3744
3	0.1276	0.2589
5	0.1577	0.2376
7	0.1717	0.2288
9	0.1798	0.2242
11	0.1851	0.2220
13	0.1895	0.2211
15	0.1930	0.2206

Models: Penalized Regression Methods

- Penalized regression (also known as shrinkage or regularization methods) is designed to **create a linear regression model that is penalized**, for having too many variables in the model, by **adding a constraint in the equation** (James et al. 2014, P. Bruce and Bruce (2017)), **to reduce (i.e., shrink) the coefficient values towards zero**.

- Three commonly used penalized regression methods are studied:
  - **Ridge Regression:** shrinks the regression coefficients with a penalty term **L2-norm** (sum of the squared coefficients).
  - **LASSO Regression:** shrinks the regression coefficients with a penalty term **L1-norm** (sum of the absolute coefficients).
  - **Elastic Net Regression:** a regression model that is penalized with both the **L1-norm and L2-norm**.

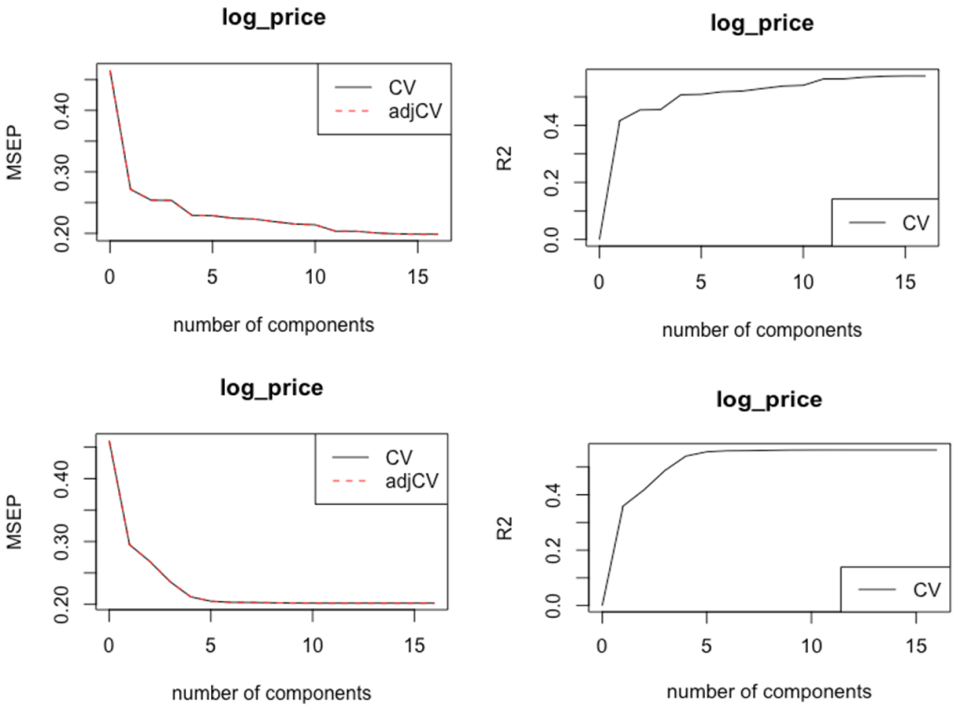
- Model Results Comparison

Methods	Tuning Parameter using 10-fold Cross Validation	Training Error /Adjusted R^2	Testing Error /Adjusted R^2
Ridge	Alpha = 0, Lambda = 0.04289114	0.1838 / 0.6019	0.1953 / 0.5709
LASSO	**Alpha = 1, Lambda = 0.0007671722**	**0.1834 / 0.6028**	**0.1951 / 0.5713**
Elastic Net	Alpha = 0.2, Lambda = 0.00244311	0.1834 / 0.6028	0.1951 / 0.5713

Models: Dimension Reduction Methods

- Dimension reduction methods work by first **summarizing the original predictors into few new variables called principal components (PCs)**, which are then used as predictors to fit the linear regression model.
- Two commonly used dimension reduction methods are studied
  - **Principal component regression (PCR):**
    - First applies **principal component analysis (PCA)** on the data set to summarize the original predictor variables into PCs.
    - **The optimal number of PCs derived from Cross Validation (CV) is 16**, which slightly reduced the original dimension of 18.
  - **Partial Least Squares (PLS) regression:**
    - Identifies PCs by not only summarizing the original predictors, but **also those are related to the outcome**.
    - Compared to PCR, PLS uses a dimension reduction strategy that is supervised by the outcome.
    - **The optimal number of PCs derived from CV is 12.**
- Model Results Comparison

Methods	Adjusted R^2	Training Error	Testing Error
PCR	0.5628	0.2011	0.2032
PLS	**0.5613**	**0.2011**	**0.2029**



Models: Regression Tree

- Decision Trees (DTs) are a **non-parametric supervised learning methods** used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.
- In our study, we trained the model by using default parameters and tuned parameters.
- For parameter tuning, we used **GridSearchCV** to get the value for some key parameters, which are listed below:
  - max_depth: 6
  - max_leaf_nodes: 5
  - min_samples_leaf: 40
  - min_samples_split: 10
- Model Results Comparison

Methods	Adjusted R^2	Training Error	Testing Error
With Parameter Tuning	**0.5701**	**0.1159**	**0.1835**
Without Parameter Tuning	0.5653	0.1443	0.1849

Models: Ensemble Methods

- Ensemble learning is the process by which **multiple models are strategically generated and combined** to solve a problem. Ensemble learning is primarily used to improve the performance of a model.
- Two commonly used ensemble methods are studied.
  - **Random Forests:**
    operate by constructing a **multitude of decision trees** at training time (combining trees with other trees) using **bootstrap sample technique**. In our study, we trained the model using default parameters and tuned parameters.
    Some key tuned parameters are listed below:

    bootstrap: False **max_depth: 8**
    max_features: log2 **min_samples_leaf: 1**
    min_samles_split: 3 **n_estimators: 50**

  - **Gradient Boosting Regressors (GBR):**
    operate by **converting weak learners to strong ones**. Decision trees are used as the weak learner in gradient boosting. After training in a gradual, additive and sequential manner, GBR is to combine those weak learners and get a stronger one.
    Some key tuned parameters are listed below:

    learning_rate: 0.01 **max_depth: 4**
    random_state: 1 **subsample: 0.75**
    n_estimators: 1000

- Model Results Comparison

Methods		Adjusted R^2	Training Error	Testing Error
Random Forests	With Parameter Tuning	0.5636	0.1353	0.1906
	Without Parameter Tuning	0.5370	0.0251	0.1999
GBR	With Parameter Tuning	0.5558	0.1230	0.1881
	Without Parameter Tuning	**0.5632**	**0.1143**	**0.1850**

Models: Support Vector Machine (SVM) Method

- SVM is **kernel-based** learning method used for pattern analysis and to find general types of relations in datasets. This method relies on **user-specified kernel,** a window function that is zero-valued outside of some chosen interval.
- In our study, we explored the application of several SVM kernels on our dataset.
  - Model tuning parameters:
    - type = "**nu-svr**" for regression where nu=**0.2**
    - **epsilon**, the insensitive-loss function = **0.1**
  - The kernels we used for training are:
    - rdfdot - Radial based kernel "Gaussian"
    - vanilladot - Linear kernel
    - laplacedot - Laplacian kernel
    - anovadot - ANOVA RBF kernel
- Model Result: We have tried several values of **C (the regularization parameter which optimizes the fit of the line to the data and penalizes the miss-classified points)**. The best results obtained were:

Kernel	Cost (C)	Training Error /Adjusted R^2	Testing Error /Adjusted R^2
laplacedot	**10**	**0.1833 / 0.8129**	**0.1899 / 0.5830**
laplacedot	100	0.0025	0.1995

Model Performance Comparison

Model	Methods	Model Performance Evaluation Metrics		
		Adjusted R^2	Training error	Testing error
1	MLR with full model	0.5986	0.1856	0.1868
2	MLR with best subset k=17	0.5986	0.1856	0.1867
3	MLR with best subset k=5	0.5641	0.2021	0.2028
4	MLR with Stepwise AIC	0.5986	0.1856	0.1868
5	KNN regression with k=7	0.6361	0.1717	0.2288
6	Ridge regression with tuning parameter	0.6019	0.1838	0.1953
7	LASSO with tuning parameter	0.6028	0.1834	0.1951
8	Elastic Net regression with tuning parameter	0.6028	0.1834	0.1951
9	PCR with PCs =16	0.5628	0.2011	0.2032
10	PLS with PCs =12	0.5613	0.2011	0.2029
11	Regression tree	0.5653	0.1443	0.1849
12	**Regression tree with tuning parameter**	0.5701	0.1159	**0.1835**
13	Random Forest	0.5370	0.0251	0.1999
14	Random Forest with tuning parameter	0.5636	0.1353	0.1906
15	Boosting	0.5632	0.1143	0.1850
16	Boosting with parameter tuning	0.5558	0.1230	0.1881
17	**SVM with laplacedot kernel and c=10**	**0.8129**	0.1833	0.1899

- We built **17 models** to evaluate the model performance:
  - **Model performance:**
    - Model selection by the **MSE**: the **Regression tree with tuning parameter performs best**; the KNN regression with k=7 performs the worst
    - Model selection by **Adjusted R^2**: the **SVM with laplacedot kernel and c=10** performs best; the **Random Forest** performs the worst

Conclusion

Model performance:

- Among all models, **Regression tree with tuning parameter** and **SVM with laplacedot kernel and c=10** derived the best performance.
  - Regression tree with tuning parameter (max_depth=6,max_leaf_nodes=5, min_samples_leaf=40, min_samples_split=10)
  - SVM with laplacedot kernel and c=10
- Among all models, **KNN regression with k=7** and **Random Forest** derived the worst performance.
  - KNN regression with k=7
  - Random Forest

Variable importance:

- Among all attributes, the following variables played significant roles on predicting the rental price.
  - **accommodates**
  - **bedrooms**
  - **room_type_Private.room**

Tuning parameter:

- When built models, we arbitrary set the tuning parameter based on various selected criteria.
- Choosing the tuning parameter is challenging and might lead to different model performance.

Further Analysis

Building other models:

- We used most of the features to predict the listing price. For future analysis we would consider exploring **Deep Learning Models**.

Including other predictors:

- From our **domain knowledge**, demographic factors (gender, age, income, and education) could impact customer price sensitivity.
- We would consider gathering **demographic factors information from external sources** by matching on the zip code.
- We could also consider using geocode data to understand the neighborhoods with the best pricing structure.

Selecting tuning parameter:

- We arbitrary set the tuning parameter based on various selected criteria to build models.
- We would consider using **other tuning parameters** to improve the predictive power.

Further Analysis

❖ Adding new model performance evaluation metrics:

- We used **Adjusted R2** and **MSE** to evaluate model performance.
- Since **MSE depends on the scale of the response data and is sensitive to outliers**, it might not be a robust measure of predicting accuracy.
- We may want to consider using the **precision measure (PM)** to evaluate prediction accuracy for the linear models because it does not depend on scale.
  - Precision measure (PM)= $\frac{\sum_{i=1}^{n}(Y_i - Y_i^*)^2}{\sum_{i=1}^{n}(Y_i - \bar{Y}_i^*)^2}$

❖ Expanding the analysis scale:

- We filtered the original data to focus on San Francisco city.
- We would consider expanding our analysis scale to **other cities or countries** and compare the results to make our models more robust.

Appendix: References

- Wiki "Airbnb”: https://en.wikipedia.org/wiki/Airbnb
- Airbnb dataset: https://www.kaggle.com/stevezhenghp/airbnb-price-prediction
- Bruce, Peter, and Andrew Bruce. 2017. *Practical Statistics for Data Scientists.* O’Reilly Media.
- James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. *An Introduction to Statistical Learning: With Applications in R.* Springer Publishing Company, Incorporated.

Thank you!