Predicting Weekdays' Airbnb Prices In Vienna

Code

Yunxuan Jiang

3/13/2023

Introduction

The purpose of this project is to develop a machine learning model to predict the Airbnb weekdays' prices in Vienna. I will be using a data set from Kaggle to do the analysis and prediction, and the details of the data set will be described later in their respective citations. Throughout this project, I will be trying four machine learning models to find the most suitable one for this regression problem.

Where Is Vienna?

Vienna, Austria's capital and largest city, lies in the country's east on the Danube River. As the most popular and primate city of Austria, Vienna is also the cultural, economic, and political center. Additionally, Known as the "City of Music" due to its musical legacy, Vienna becomes the shrine and home for many musicians including Mozart, Beethoven and Sigmund Freud. The city is also famous for its Imperial palaces, including Schönbrunn, the Habsburgs' summer residence. Therefore, Vienna is a good place to travel.

What are we going to do and why we will do it?

I am a classical music lover who learned lots of music instruments including piano and flute since I was young. Vienna is so famous for every classical music learner that I am particularly drawn to Vienna's rich history and architectural marvels. The winter quarter is going to end soon and I will have a ten day holiday, so I am excited to explore Vienna's vibrant cultural scene with my friends and immerse myself in the city's rich heritage. However, an important thing is we don't have too much money while we still want to experience native people's life with a good living equality, so an idea raised in my mind: can I analyze the past data of Vienna's Airbnb prices in advance according to factors such as "whether the room is private or

not" and "the cleanliness rating of the listing," and then design a model to predict the Airbnb price? If this is feasible, then we can know whether we need to prepare more money or we should lower our living quality. In this way, we can better enjoy our trip.

Therefore, for this project, I am going to predict the Airbnb prices in Vienna on weekdays, when the prices are comparatively lower than that on weekends. The goal of this project is not only for me but also for everyone who plans to visit Vienna. Using the model we can save much time searching Airbnb information online for the price and be more prepared for everything we need.

Outline

To solve the problem "developing a model to predict weekdays' Airbnb prices in Vienna," we first need to clean and tidy the raw data and then explore specifically for each predictor. After that, we are going to split the data and then create a recipe. Since we will predict the price, we should build five regression model: linear regression model, elastic net model, knn model, random forest model, and polynomial regression model to compare the which model has the smallest rmse and and highest R^2 with train data set. Lastly, we will evaluate the performance of the best model we choose by implementing the testing data. Then, we can use our model to predict the price!

Exploring the Data

Loading and Exploring the Data

Hide

```
# Loading all important packages
library(tidymodels)
library(ISLR)
library(ISLR2)
library(tidyverse)
library(glmnet)
library(modeldata)
library(ggthemes)
library(janitor)
library(naniar)
library(corrplot)
library(patchwork)
library(rpart.plot)
library(xgboost)
library(ranger)
library(vip)
library(dplyr)
library(doParallel)
library(yardstick)
library(forcats)
library(caret)
library(kknn)
registerDoParallel(cores = parallel::detectCores())
tidymodels_prefer()

# Set seed for future use
set.seed(3435)

# Read the csv we need, unify the variable names, and show the first six rows to be famili
ar with all variables
vienna <- read_csv("vienna_weekdays.csv")
vienna <- vienna %>%
  clean_names()
head(vienna)
```

```
## # A tibble: 6 × 20
##      x1 real_sum room_type   room_…¹ room_…² perso…³ host_…⁴ multi   biz clean…⁵
##   <dbl>    <dbl> <chr>       <lgl>   <lgl>     <dbl> <lgl>   <dbl> <dbl>   <dbl>
## 1     0     251. Entire hom… FALSE   FALSE         3 TRUE        1     0      10
## 2     1     157. Entire hom… FALSE   FALSE         3 FALSE       0     0      10
## 3     2     283. Entire hom… FALSE   FALSE         5 TRUE        0     1      10
## 4     3     302. Entire hom… FALSE   FALSE         4 TRUE        0     1      10
## 5     4     151. Entire hom… FALSE   FALSE         2 TRUE        0     1      10
## 6     5     162. Private ro… FALSE   TRUE          2 TRUE        1     0      10
## # … with 10 more variables: guest_satisfaction_overall <dbl>, bedrooms <dbl>,
## #   dist <dbl>, metro_dist <dbl>, attr_index <dbl>, attr_index_norm <dbl>,
## #   rest_index <dbl>, rest_index_norm <dbl>, lng <dbl>, lat <dbl>, and
## #   abbreviated variable names ¹room_shared, ²room_private, ³person_capacity,
## #   ⁴host_is_superhost, ⁵cleanliness_rating
```

The data is from the Kaggle data set, "Airbnb Prices in European Cities."

Variable Selection

We first look at how many observations and variables there are.

Hide

```
dim(vienna)
```

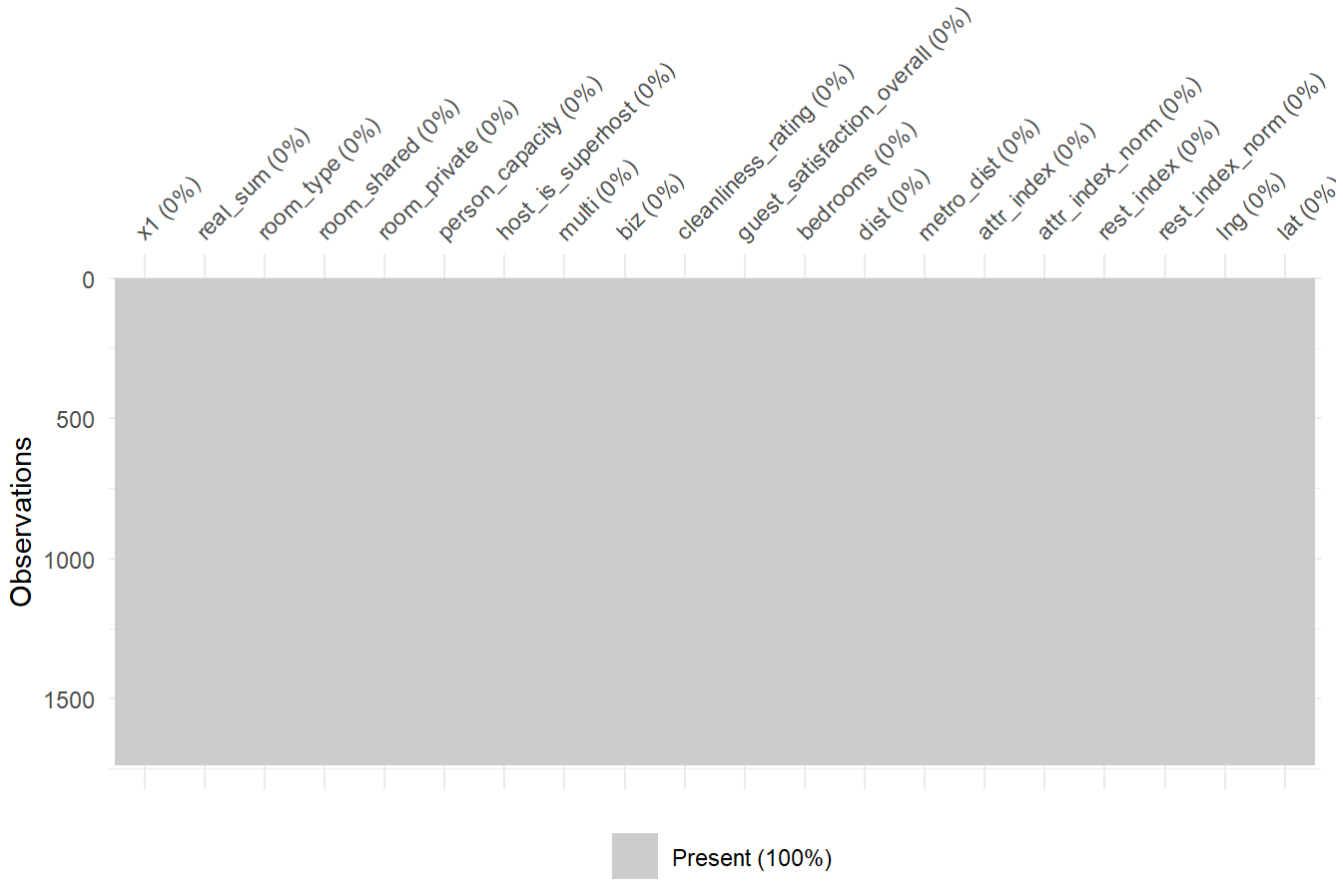
```
## [1] 1738   20
```

There are 1738 observations and 20 variables. The variable, `real_sum`, is our response, and the rest 19 variables may be the predictors. However, 19 variables are too much, so we must narrow that down.

But first we need to see if there are any missing data.

Hide

```
vis_miss(vienna)
```



We are surprised to find that there is no missing data! This will save us a lot of work.

Now we need to think about keeping what variables in the data set. We see that `x1` only represents the order of the observations, which has nothing to do with the Airbnb price, so we will drop it. What's more, according to the coding book on the Kaggle, I find that there should have been 15 useful variables, and `x1`,

`attr_index`, `attr_index_norm`, `rest_index`, `rest_index_norm` are variables without any explanations. After carefully thinking about the meaning of these five variables, my conclusion is that they are meaningless. Thus, all these five variables should not be included.

Tidying the Data

Though `cleanliness_rating` and `guest_satisfaction_overall` are all numeric, these scores actually represents the quality of living. After glancing at the data set, I find that the scores for these two columns are really high, I guess this may because all these Airbnb do provide good houses or people don't want to criticize too much when filling out the survey. I am a person who really like clean rooms, and I also believe that we should set a more strict criteria because of the high scores. So for cleanliness, I want to add a new column called `clean_rate` that has the value `0` for the score below 9 and `1` for the score greater than or equal to 9. For guest satisfaction rate I also want to add a new column called `satisfaction_rate` that has the value `0` for the score below 90 and `1` for the score greater than or equal to 90. Then we remove the column `cleanliness_rating` and `guest_satisfaction_overall`.

We first add the two new columns we want. Then, we will use `select` to form our new data set for further tidying! Note that we will note include `cleanliness_rating` and `guest_satisfaction_overall` anymore.

Hide

```
vienna <- vienna%>%
  mutate(clean_rate = if_else(cleanliness_rating < 9, 0, 1)) %>%
  mutate(satisfaction_rate = if_else(guest_satisfaction_overall< 90, 0, 1))%>%
  select(c("real_sum","room_type","room_shared","room_private","person_capacity","host_is_
superhost","multi","biz","clean_rate","satisfaction_rate","bedrooms","dist","metro_dis
t","lng","lat"))
```

Now we only have 15 variables. Though I want to include fewer variables, these 15 variables are significant factors that we usually considered when booking Airbnb.

Converting Factors and Reordering Levels

We will convert `room_type`, `room_shared`, `room_private`, `host_is_superhost`, `multi`, `biz`, `clean_rate`, `satisfaction_rate` to factors.

When changing `room_shared`, `room_private`, `host_is_superhost` to factors, we will reorder the factors so that `TRUE` is the first level for all these three predictors.

Lastly, since there are only two rows that the Airbnb prices are greater than 10000 and all other values are less than 1000, we will remove these two rows for easier analysis.

Hide

```
vienna <- vienna %>%
  mutate(room_type = factor(room_type),
         room_shared = factor(room_shared, levels = c("TRUE","FALSE")),
         room_private = factor(room_private, levels = c("TRUE","FALSE")),
         host_is_superhost = factor(host_is_superhost, levels = c("TRUE","FALSE")),
         multi = factor(multi),
         biz = factor(biz),
         clean_rate = factor(clean_rate),
         satisfaction_rate = factor(satisfaction_rate))

# Making real_sum an integer
vienna$real_sum <- as.integer(vienna$real_sum)

# Remove two rows that contain outliers
vienna <- vienna %>%
  filter(real_sum < 10000)

dim(vienna)
```

```
## [1] 1736   15
```

Now we make sure that we have thrown all the variables and observations that we don't want.

Describing the Predictors

After removing unnecessary variables and tidying, we can get a better understanding of what each variable that we will be using represents. The variables that I selected in my model recipe to predict the Airbnb prices later on are as following:

- `real_sum` : The total price of the Airbnb listing.
- `room_type` : The type of room being offered (e.g. private, shared, etc.).
- `room_shared` : Whether the room is shared or not.
- `room_private` : Whether the room is private or not.
- `person_capacity` : The maximum number of people that can stay in the room.
- `host_is_superhost` : Whether the host is a superhost or not.
- `multi` : Whether the listing is for multiple rooms or not.
- `biz` : Whether the listing is for business purposes or not.
- `clean_rate` : The cleanliness rating of the listing.
- `satisfaction_rate` : The overall guest satisfaction rating of the listing.
- `bedrooms` : The number of bedrooms in the listing.
- `dist` : The distance from the city center.
- `metro_dist` : The distance from the nearest metro station.
- `lng` : The longitude of the listing.
- `lat` : The latitude of the listing.

Visual EDA

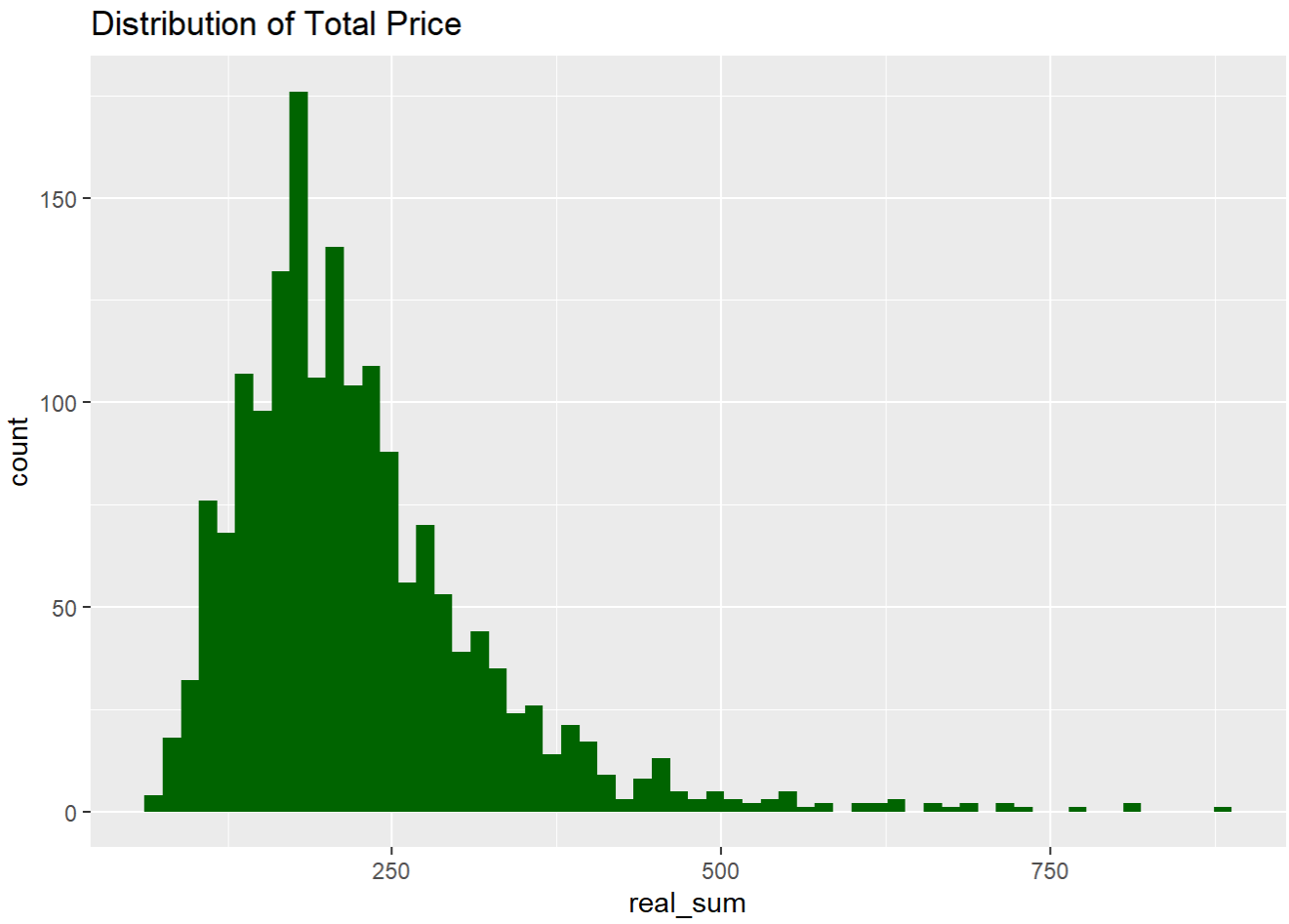
Now we will explore the relationships between selected predictors and the outcome.

The Total Price of The Airbnb Listing

I use the histogram to see the distribution of our outcome, the total price. By analyzing it we can master the information of Airbnb prices in Vienna.

Hide

```
vienna %>%
  ggplot(aes(x = real_sum)) +
  geom_histogram(bins = 60, fill = "darkgreen") +
  labs(title = "Distribution of Total Price")
```



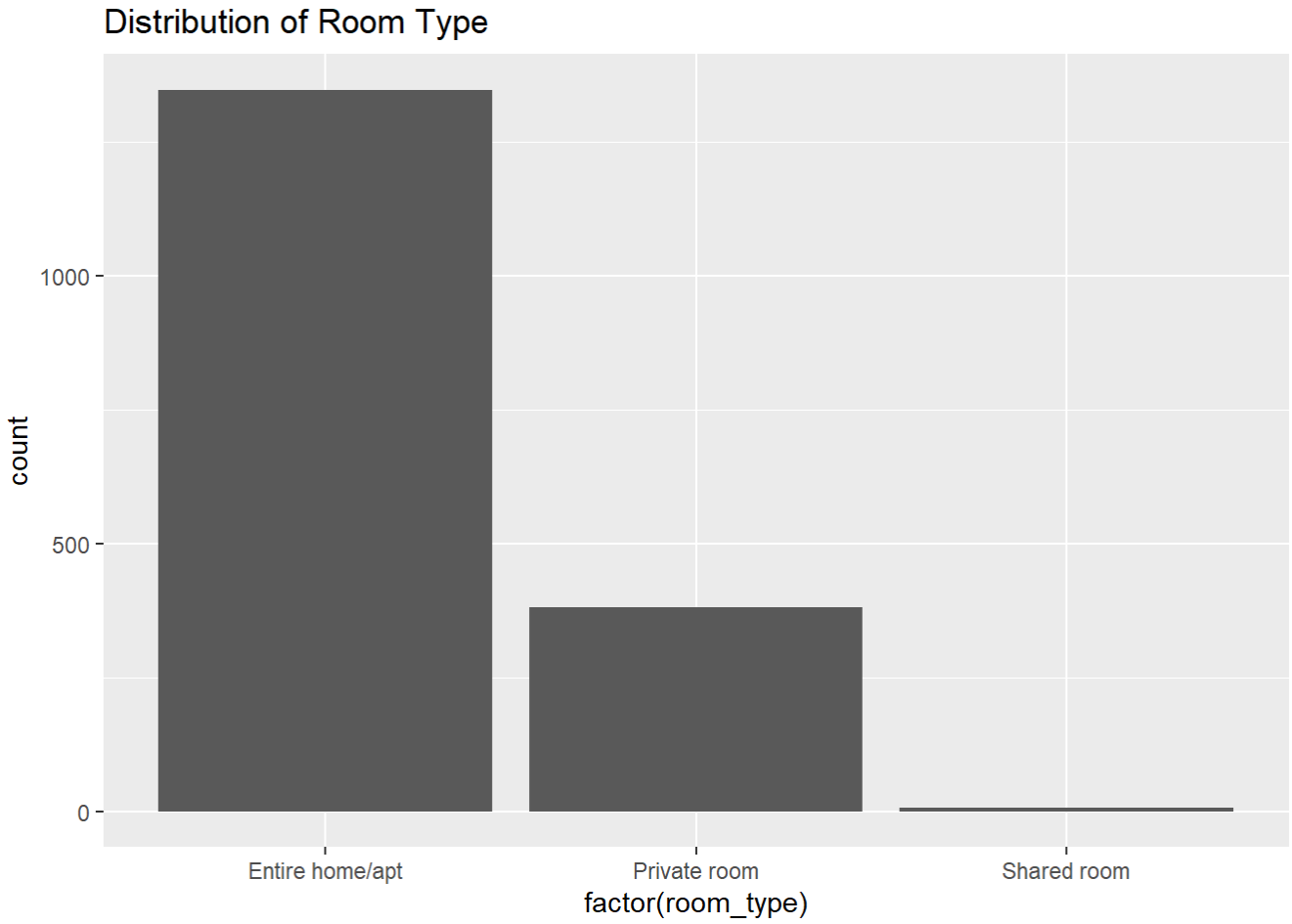
We see that the values of `real_sum` ranges from 0 to 1000. We see that there are only few houses on Airbnb have prices higher than 500. Most prices range from 50 to 375. This means the Airbnb prices are similar with the Airbnb prices in the U.S. and my friend and I don't need to worry too much about the accommodation expenditure.

Room Type

There are three kinds of room types: entire home/apt, private room, and shared room. This poses a question: How many rooms are provided for each room type? According to the distribution of room type, we can know whether it's easy to find a room type we want with given number of people. So I choose I bar chart.

Hide

```
vienna %>%
  ggplot(aes(x = factor(room_type))) +
  geom_bar()+
  labs(title = "Distribution of Room Type")
```



As we can see, most houses provided are entire home/apt, and the number is more than 1250, which accounts for more than 90% of the total number on the list. Only few Airbnb has shared room. This means that it will be hard for tourists to find shared room on Airbnb in Vienna, but if you travel there with a family, then you will have a lot of choices.

Person Capacity

We also want to check the relationship between person capacity and the price. This is significant since if we have 5 people, whether we should book a double room + triple room or just book a room which capacity is 5 people in order to spend the least money on accommodation. So we make a box plot, grouping by person capacity.

Hide

```
vienna %>%
  ggplot(aes(group = person_capacity, x = real_sum, y = person_capacity)) +
  geom_boxplot() +
  labs(title = "Box Plot of Person Capacity vs. Total Price",
       y = "Person Capacity", x = " Total Price")
```

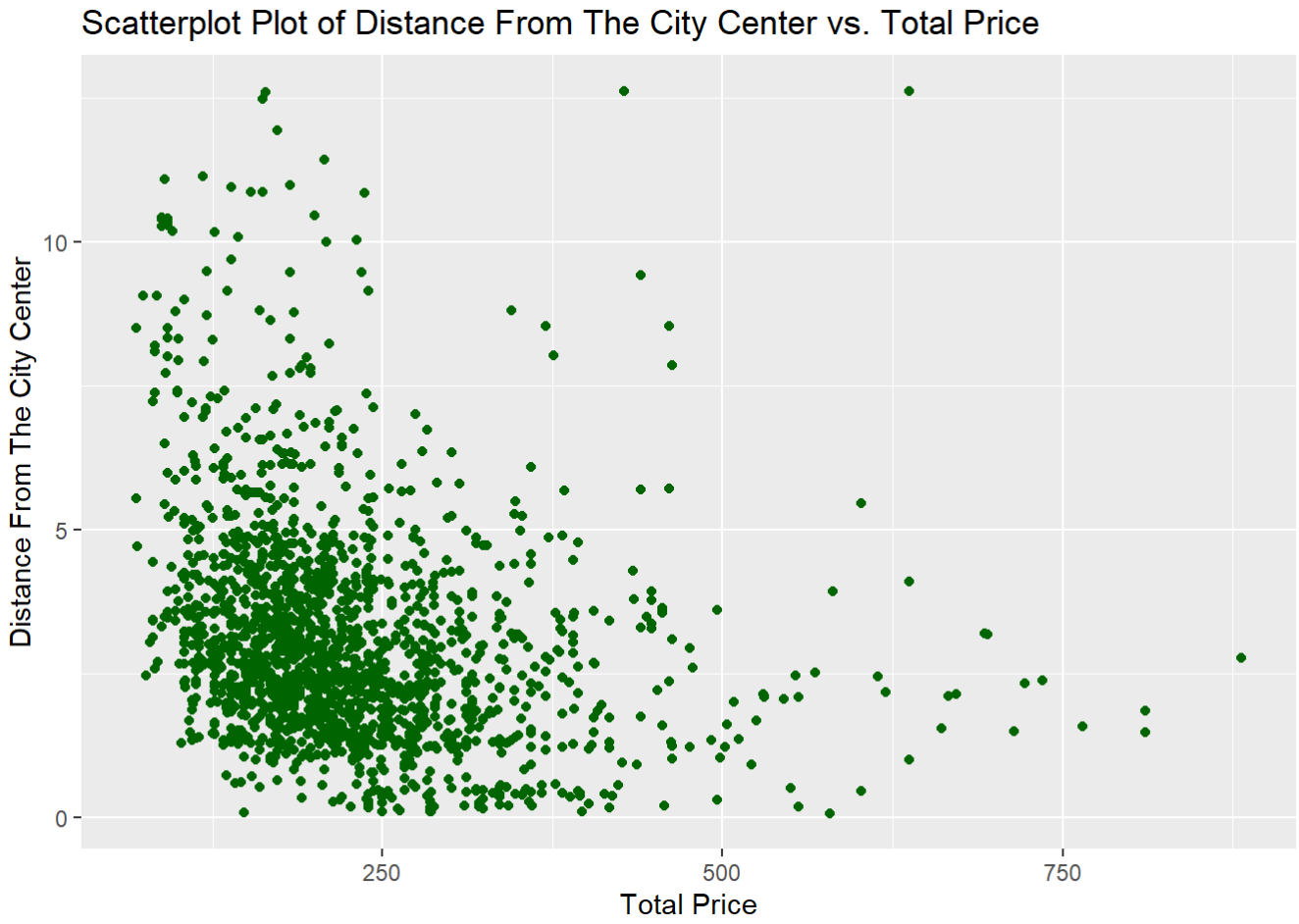
We see that the bigger the room capacity is, the higher the total price is. This actually meet our satisfaction. But an interesting thing is, we find that the number of outliers are the most when the person capacity is 2, while there is only one outlier for person capacity = 5. This means tourists have more choices for different prices of rooms when the tourist number is less than or equal to 2, while people can not find an Airbnb which price ranges from 550-765.

The Distance From The City Center

When we choose the houses, we always care about whether it it far from the center of city. If we live too far away, then we need to spend much time on commuting. Therefore, we will analyze the relationship between `dist` and `real_sum` by a scatterplot.

Hide

```
ggplot(vienna, aes(x=real_sum, y=dist))+geom_point(color="darkgreen") +
  labs(title = "Scatterplot Plot of Distance From The City Center vs. Total Price",
       y = "Distance From The City Center", x = "Total Price")
```



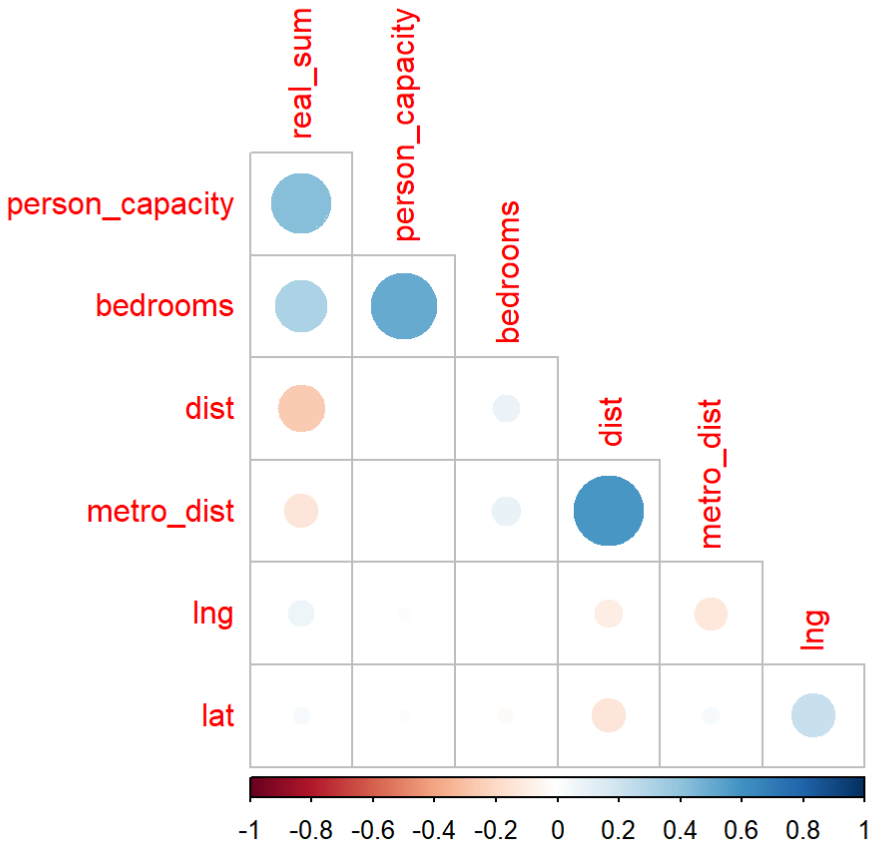
From the plot we see that most Airbnb are close to the city center as well as cheap. When the price is less than 250, we find that the distance can vary from 0 to 10, which means you can always find a cheap Airbnb no matter where you live in Vienna. When the price is higher than 500, we can see that the distance is below 5 for most data, representing that expensive houses provided on Airbnb are always close to the city center.

Correlation Plot

To get an idea of the relationships between our numeric variables, we'll make a correlation plot.

Hide

```
vienna %>%
  select(where(is.numeric)) %>%
  cor() %>%
  corrplot(type = "lower", diag = FALSE)
```



I only include numerical variables in the corrplot. Other categorical variables are not included. We see that `dist` has strong positive relation with `metro_dist`, and `bedrooms` have a strong positive relation with `real_sum` and `person_capacity`, `person_capacity` is the variable that has the strongest relation with `real_sum` among all the numerical variables. `dist` and `metro_dist` are negatively related with `real_sum`.

By this graph I also realize that lat and lng may not be included as predictors when we build the recipe since they have weak relation with real_sum.

Setting Up Models

Base on what we did in the former sections, now we have a better idea of what predictors are related with `real_sum`, so we can finally move on to building our models to see whether we really can predict the Airbnb weekdays' prices in the Vienna. In this part we will randomly split our data into training and testing sets, set up and create our recipe, and create folds for k-fold cross validation within our models.

Train/Test Split

The first step for building models is to split our data into separate data sets, a training set and a testing test. The training set is for training our models and the testing set is to evaluate our models. For the split, I decide to choose a 72/25 split, which means 75% of the data will go towards the training set and 25% of the data will go towards the testing set. Additionally, the split will be stratified on the outcome variable, `real_sum`. In this way, we can make sure that both the training and testing data have an equal proportion of observations with a given categorical variable `real_sum`.

Hide

```
vienna_split <- initial_split(vienna, prop = 0.75,
strata = real_sum)
train <- training(vienna_split)
test <- testing(vienna_split)
dim(train)
```

```
## [1] 1300   15
```

Hide

```
dim(test)
```

```
## [1] 436  15
```

Since 1300/1736 = 0.74884793, we verify that our training set has nearly 75% of the total data and testing set has nearly 25% of the total data.

Creating a Recipe

Since all the models will use the same predictors, outcome, and observations, We will now build one recipe which we will use for all the models. Every model will utilize this recipe and work with it by different four models.

We will be using 10 predictors: `room_type` , `room_shared` , `room_private` , `person_capacity` , `biz` , `clean_rate` , `satisfaction_rate` , `bedrooms` , `dist` , `metro_dist` . We exclude `lng` and `lat` because we see from the correlation plot that there is weak relationship between `lng` and `real_sum` , and `lat` and `real_sum` . We exclude `multi` since we observed that multi and biz are highly negatively related, so we must delete one of them to minimize the error. Also, we delete `host_is_superhost` since we also find that this has no relation with `real_sum` .

When we build the recipe, we will dummy all the nominal variables and normalize(center and scale) them.

Hide

```
vienna_recipe <- recipe(real_sum ~ room_type+room_shared+room_private+person_capacity+biz+
clean_rate+satisfaction_rate+bedrooms+dist+metro_dist, data = train) %>%
  step_dummy(room_type,room_shared,room_private,biz,clean_rate,satisfaction_rate)%>%
  step_interact(terms = ~ starts_with("bedrooms"):person_capacity + metro_dist:dist + metr
o_dist:bedrooms) %>%
  step_normalize(all_predictors())
```

K-Fold Cross Validation

We'll stratify our cross validation on our response variable, `real_sum` , as well as use 10 folds to conduct k-fold stratified cross validation. Among the 10 folds, each will be the testing set while other k-1(=9) folds will be the train sets. Then, we will get the average accuracy for evaluating the performance of each model.

Hide

```
vienna_folds <- vfold_cv(train, v = 10,strata="real_sum")
```

Building Prediction Models

It is now time to build our models! Since our data set is not small and my computer always runs codes slowly, I saved the results of each model as RDA files in avoid of taking too much time running the code again and again. These RDA files will be loaded later for our explorations of the results. Since our outcome is numerical, I choose Root Mean Squared Error(RMSE) as my metric. The RMSE is a commonly used measurement for evaluating the performance of regression models by showing how much error the predicted values are different from true values in total. Therefore, the lower the RMSE, the better our model is. Let's start building our models!

1. Set up models and workflows

Hide

```
# Linear Regression Model
lm_mod <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

lm_wkflow <- workflow() %>%
  add_model(lm_mod) %>%
  add_recipe(vienna_recipe)


# Elastic Net Regression Model
en_mod <- linear_reg(mixture = tune(), penalty = tune()) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

en_wkflow <- workflow() %>%
  add_model(en_mod) %>%
  add_recipe(vienna_recipe)


# KNN Model
knn_mod <- nearest_neighbor(neighbors = tune()) %>%
  set_mode("regression") %>%
  set_engine("kknn")

knn_wkflow <- workflow() %>%
  add_model(knn_mod) %>%
  add_recipe(vienna_recipe)

# Random Forest Model
rf_mod <- rand_forest(mtry = tune(),
                          trees = tune(),
                          min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

rf_wkflow <- workflow() %>%
  add_model(rf_mod) %>%
  add_recipe(vienna_recipe)

# Polynomial Regression Model
rec_poly <- vienna_recipe %>%
  step_poly(person_capacity,bedrooms,dist,metro_dist, degree = tune())

poly_mod <- linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

poly_wkflow <- workflow() %>%
  add_model(poly_mod) %>%
  add_recipe(rec_poly)
```

2. Create a tuning grid to specify the ranges of the parameters as well as the ranges of levels.

Hide

```
# Linear Regression Model
# No grid since there is no tuning parameter.


# Elastic Net Regression Model
en_grid <- grid_regular(penalty(range = c(0, 1),
                                trans = identity_trans()),
                        mixture(range = c(0, 1)),
                        levels = 10)

# KNN Model
neighbors_grid <- grid_regular(neighbors(range = c(1, 10)), levels = 10)


# Random Forest Model
rf_grid <- grid_regular(mtry(range = c(1, 8)),
                        trees(range = c(100, 200)),
                        min_n(range = c(10, 20)),
                        levels = 6)

# Polynomial Regression Model
degree_grid <- grid_regular(degree(range = c(1, 10)),
                            levels = 10)
```

3. Tune the models with specific parameters of choice, and save them to RDA files to save time and work.

4. Load back the RDA files.

Hide

```
load("en_tune.rda")
load("knn_tune.rda")
load("rf_tune.rda")
load("poly_tune.rda")
```

5. Fit linear regression model.

Hide

```
fit_lm <- lm_wkflow %>%
  fit_resamples(resamples = vienna_folds)
```

6. Collect the metrics of the tuned models, use slice to choose the lowest RMSE

Hide

```
# Linear Regression Model
lm_rmse <- collect_metrics(fit_lm)%>%
  slice(1)  # use slice to select only the lowest rmse

# Elastic Net Regression Model
en_rmse <- collect_metrics(en_tune)%>%
  arrange(mean)%>%
  slice(101)

# KNN Model
knn_rmse <- collect_metrics(knn_tune)%>%
  arrange(mean)%>%
  slice(11)

# Random Forest Model
rf_rmse <- collect_metrics(rf_tune)%>%
  arrange(mean)%>%
  slice(217)

# Polynomial Regression Model
poly_rmse <- collect_metrics(poly_tune)%>%
  arrange(mean)%>%
  slice(5)
```

Model Results

Now it's time to compare all the results of our models and select the one with the least RMSE!

Hide

```
# Creating a tibble
final_compare_tibble <- tibble(Model = c("Linear Regression","Elastic Net","K Nearest Neig
hbors","Random Forest","Polynomial Regression"),RMSE = c(lm_rmse$mean,en_rmse$mean,knn_rms
e$mean,rf_rmse$mean,poly_rmse$mean))

final_compare_tibble <- final_compare_tibble %>%
  arrange(RMSE)

final_compare_tibble
```

```
## # A tibble: 5 × 2
##   Model                 RMSE
##   <chr>                <dbl>
## 1 Polynomial Regression  74.8
## 2 Random Forest          75.7
## 3 Elastic Net            76.0
## 4 Linear Regression      76.0
## 5 K Nearest Neighbors    81.1
```

From the tibble we can see that the polynomial regression model performed the best! We can also see a trend that simpler models performed worse, which implies our data is probably nonlinear. One thing important is that our rmse is really big, which means the outcome `real_sum` is actually really difficult to predict, and this may because the raw data is not good such as the random error of my predictors are really big, and we may need to learn more complicated models to truly tackle it.

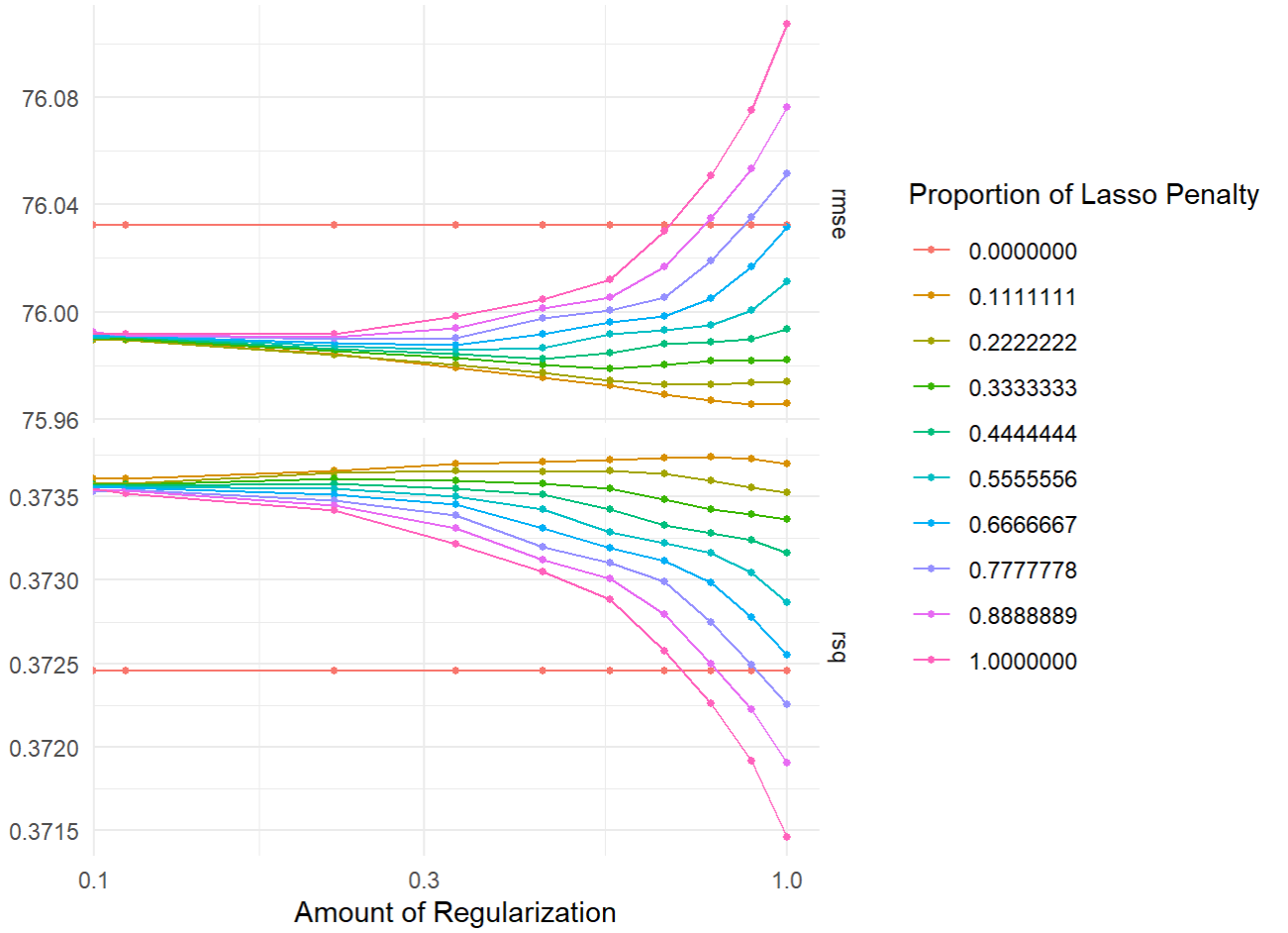
Model Autoplots

We are going to use the autoplot in R to visualize the performance of each model with different tuned parameters. Since our model is regression, the performance is evaluated by the RMSE(the lower, the better).

Elastic Net Plot

Hide

```
autoplot(en_tune) + theme_minimal()
```

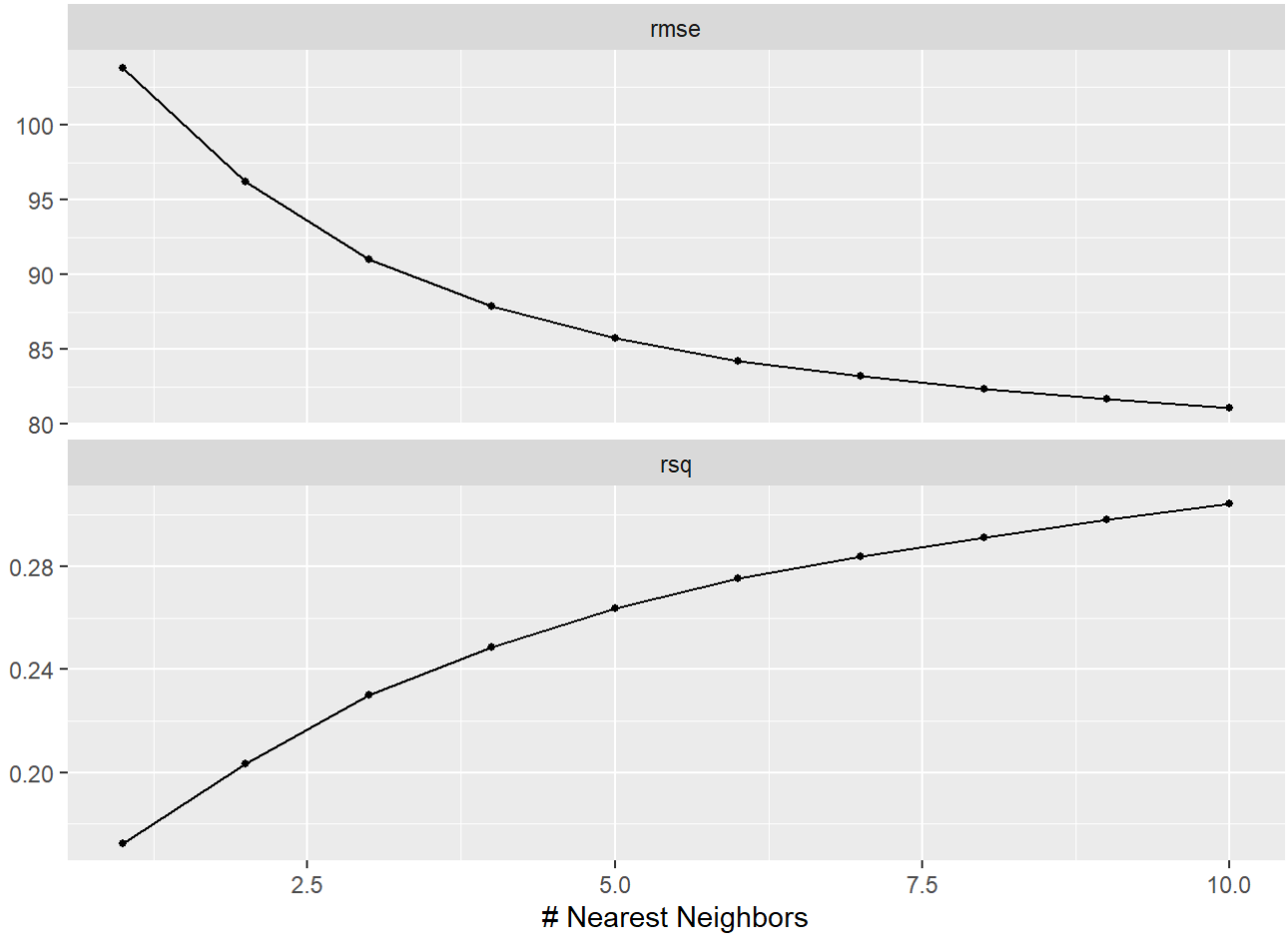


For the elastic net model, we use 10 different levels of penalties and see that when the penalty is 0, the rmse and rsq don't change, while the rmse and rsq values will change with other penalties. We also observe that the greater the penalty, the bigger the rmse and smaller rsq. Moreover, when the penalty is small(take 0.11111 for instance), the rmse decreases and rsq increases as the amount of regularization becomes bigger.

KNN Plot

Hide

```
autoplot(knn_tune)
```

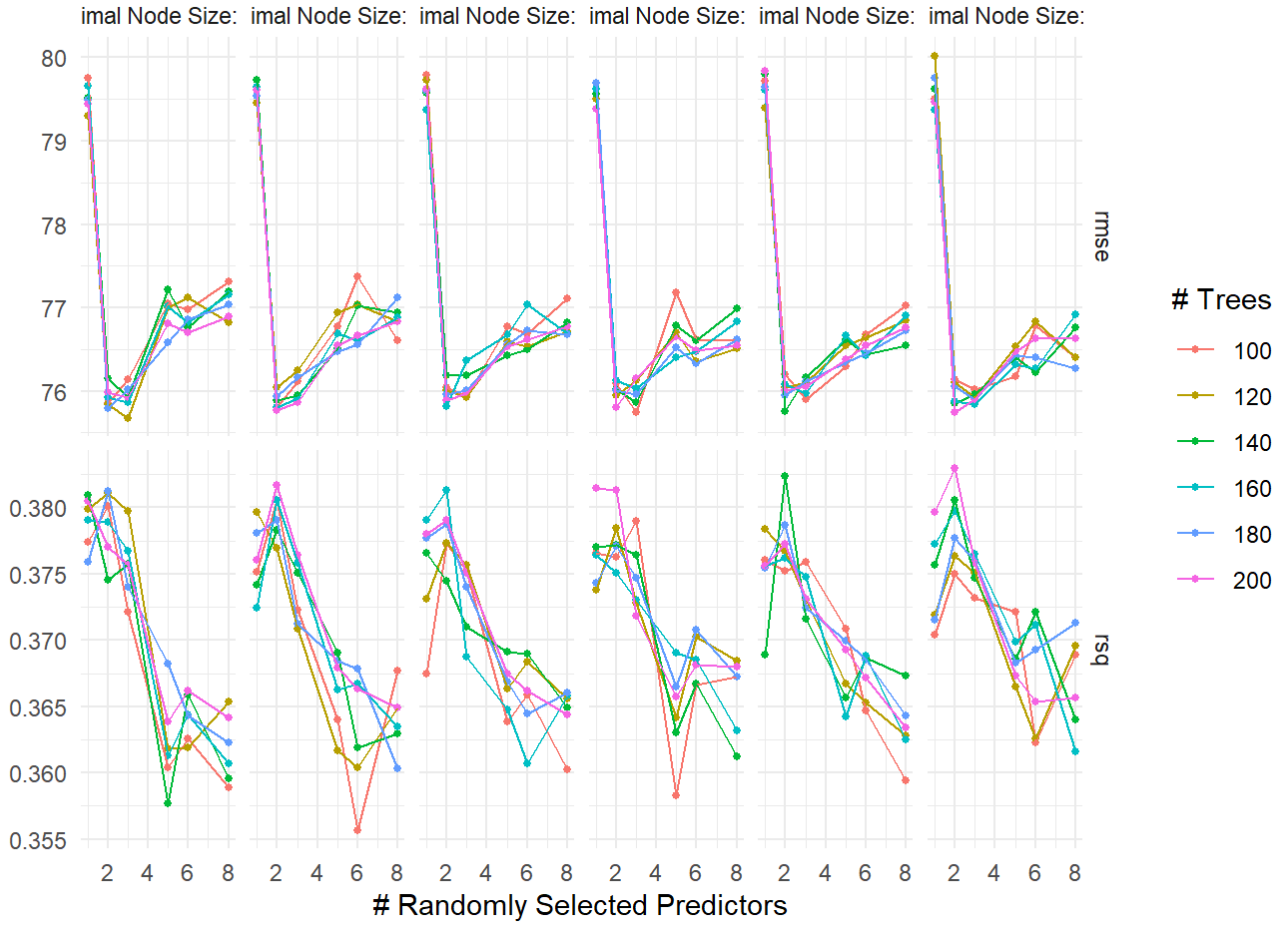


For Knn model, we see from the graphs that both rmse curves and rsq curves are smooth curves. We observe that when k=10, we get the smallest rmse and greatest rsq, which means our best data is given by k=10.

Random Forest Plot

Hide

```
autoplot(rf_tune) + theme_minimal()
```

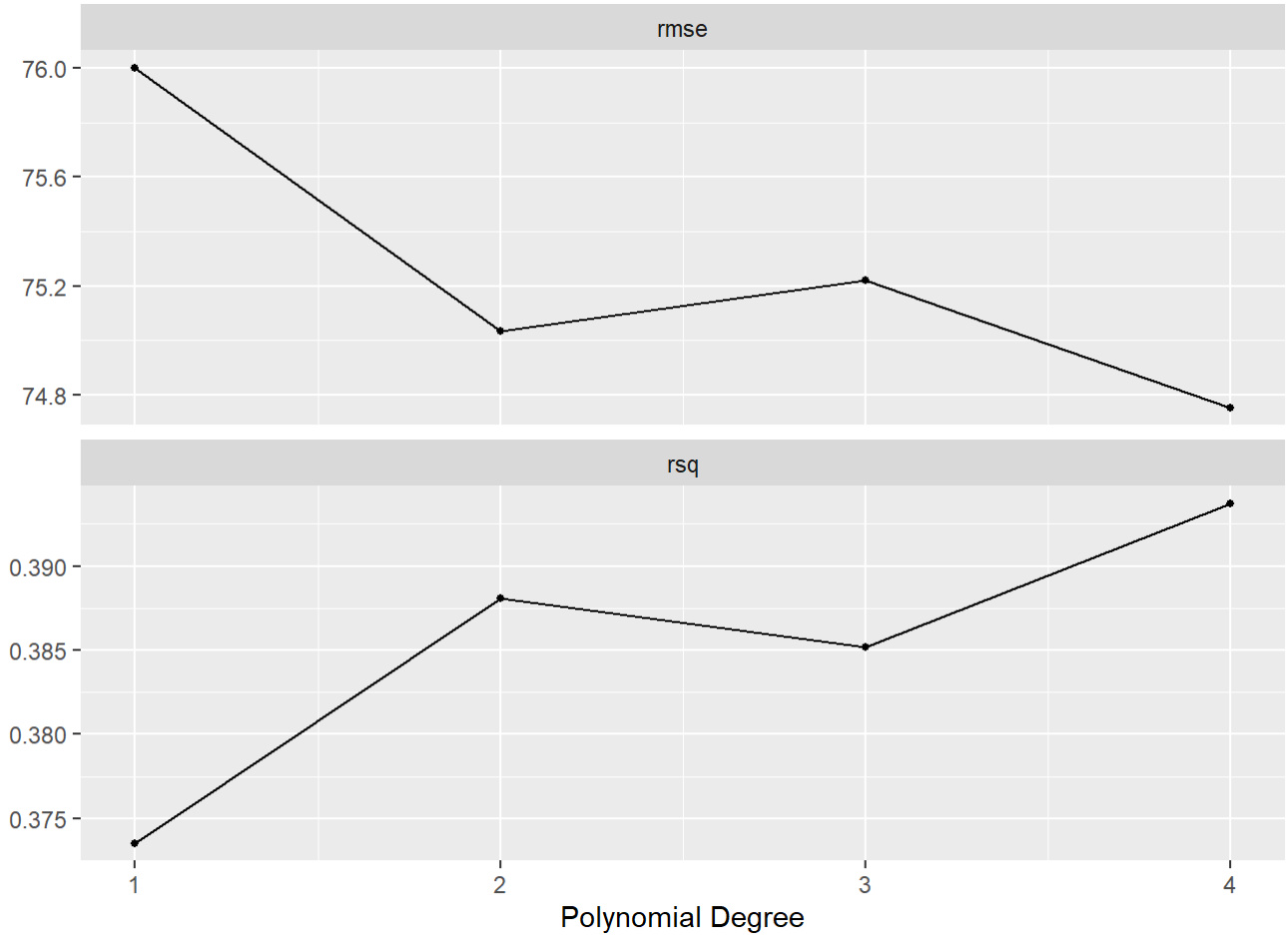


For random forest model, we see that mid range trees value have better rmse. Notice that when we set the mtry for the range(1,8) since we need to avoid creating a bagging model. According to the graph, mtry doesn't have a great effect on the performance of the model. Moreover, We observe that when the number of randomly selected predictors is 1, the rmse is really high, which means much error. But when the number of randomly selected predictors be 2, we get the smallest rmse and greatest rsq.

Polynomial Regression Plot

Hide

```
autoplot(poly_tune)
```



For polynomial regression model, we know from the graph that the model is the best when the polynomial degree is 4 since we get the smallest rmse and the greatest rsq now. On the other hand, the model is the worst when the polynomial degree is 1. According to the gradient of curve, we can say that when the polynomial degree increases, the rmse decreases and rsq increases, which means the model becomes better as the polynomial degree increases.

Results of the Best Model

Performance on the folds

According to the metrics above, we see that the best model is polunomial regression model:

Hide

```
poly_tune %>%
  collect_metrics()%>%
  arrange(mean)%>%
  slice(5)
```

```
## # A tibble: 1 × 7
##   degree .metric .estimator  mean     n std_err .config
##    <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1      4 rmse    standard    74.8    10    3.61 Preprocessor04_Model1
```

So when the polynomial degree is 4, we get the best model with RMSE 74.75131.

Fitting to Training Data

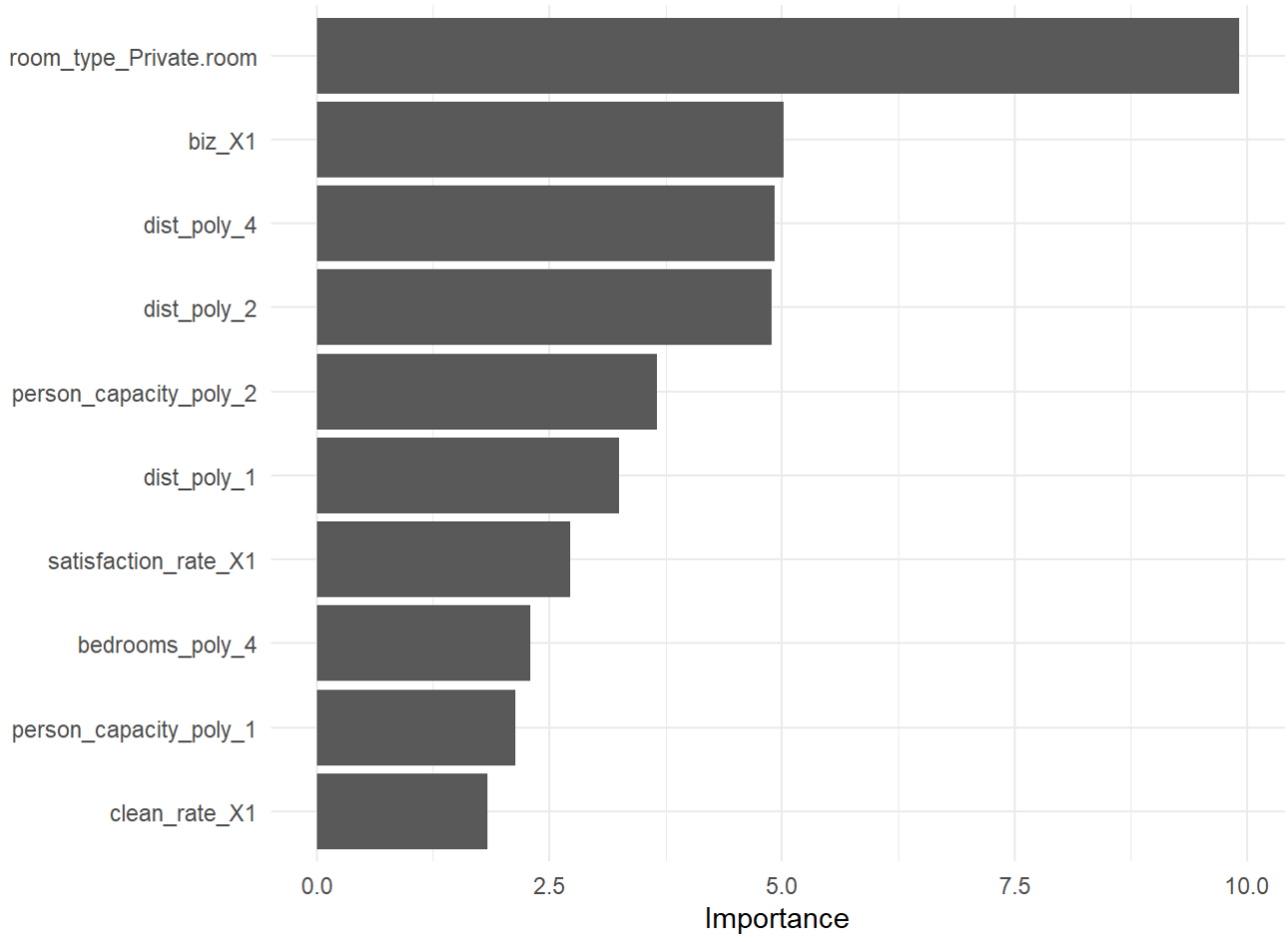
Now, we will take the best model – polynomial regression model and fit it into the entire training data.

Hide

```
best_poly <- select_best(poly_tune,metric = "rmse")
final_poly_model <- finalize_workflow(poly_wkflow, best_poly)
final_poly_model <- fit(final_poly_model, train)
```

Hide

```
final_poly_model %>% extract_fit_parsnip() %>%
  vip() +
  theme_minimal()
```



From the graph we see that the top three important predictors are whether the room type is private room, Whether the listing is for business purposes or not, and the distance from the city center.

Testing the Model

Now we will put the test data into the model to evaluate the performance.

Hide

```
final_poly_model_test <- augment(final_poly_model, test)
rmse(final_poly_model_test, truth = real_sum, .pred)
```

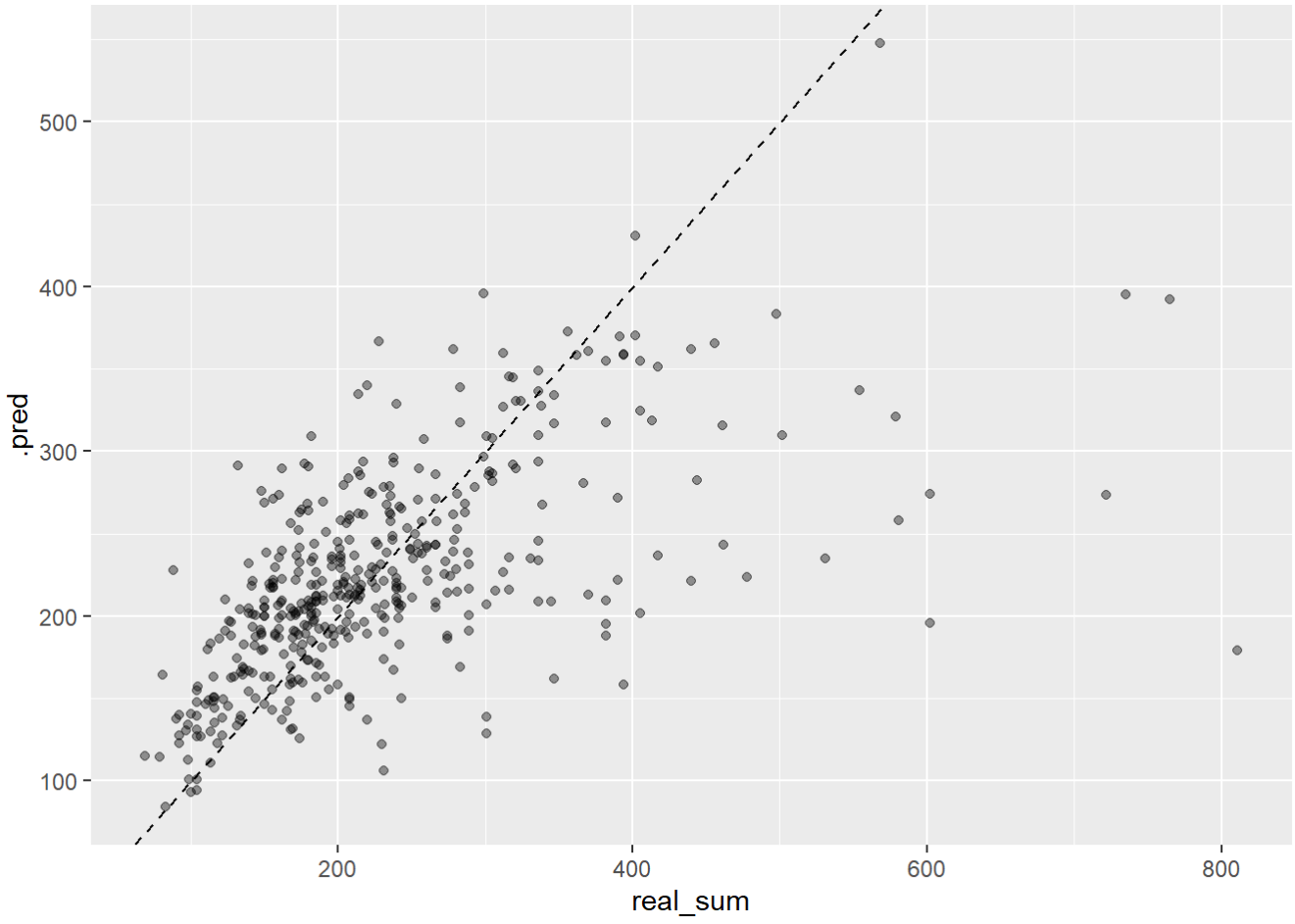
```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        84.3
```

Our polynomial regression model performed worse on the testing set than on the cross-validation folds with an RMSE of 84.32798. Though this is a huge RMSE, but our outcome, `real_sum` , ranges from 70 to 881 under the case that we have moved the two outliers, both are greater than 10000. Therefore, the RMSE is not too bad compared with the ranges of our outcome. So our model is not that bad and can still show some relationship between the outcome and our predictors.

We could also view a scatterplot of the actual price values in the testing set versus the model-predicted values:

Hide

```
final_poly_model_test %>%
  ggplot(aes(x = real_sum, y = .pred)) +
  geom_point(alpha = 0.4) +
  geom_abline(lty=2)
```



Hide

```
  labs(title = "Predicted Values vs. Actual Values")
```

```
## $title
## [1] "Predicted Values vs. Actual Values"
##
## attr(,"class")
## [1] "labels"
```

If the model predicted accurately, then every point would be at or near the straight line. From the plot we can see that our model predicts accurately when the real_sum is less than 400, but didn't perform good when real_sum is greater than 400. This means we still have a lot to do to predict the price more accurately.

Conclusion

Throughout this project, we researched and explored our data, and designed the model to fit them and then evaluate the performance of the model. After doing everything, we find that the polynomial regression model is the best since it has the smallest RMSE, and as an regression model, RMSE is the most important criterion.

When I explored the data, I found that the outcome, `real_sum` , has a huge range, and because there are many predictors which also have big ranges of the value, I knew that my model will have a great RMSE and the model may not be perfectly suitable. To minimize the RMSE of the five models I chose, I searched for the internet and asked ULA and TA for help, but no matter how I adjusted the parameters and redesigned the recipe, the RMSE didn't change too much. Therefore, I think I need to learn more complex models in the future, then I can better predict the value of the airbnb price. It seems that simple models have bigger error for this data set since this is nonlinear.

One potential question that may be lingering is whether the choice to change `cleanliness_rating` and `guest_satisfaction_overall` to categorical variables was correct. I assumed that this will make my result better, but it seems that I make the RMSE a little bit bigger. And I also found that if we persist using these two predictors as numerical variables and delete `guest_satisfaction_overall` , the RMSE will be smaller, though the improvement will only be a little bit. But I believe that changing these factors to categorical variables with new criterion is helpful to customers. I think the the more error is created by my strict criterion, but customers want to see a strict and real evaluation of the cleanliness and satisfaction rate.

As for further extensions, if we get the data of the Airbnb prices in different cities or different countries, we can make a model for each of them and then compare some similarities and differences. I believe that this is important for international enterprises because they need lots of models to analyze the outcome they want in different places. I also believe that, if we gain a lot of data, we can make a new model which combines all these data, then larger number of observations can help create a more accurate model, and two variables, lat and lng, can be utilized. We didn't use these two variables since we only analyze the Airbnb prices in one city, Vienna.

Overall, attempting to predict the Airbnb weekday prices in Vienna using this data set provided a great opportunity for me to bild my machine learning and data analysis skills. I was lost when I first learned this course, but I found myself becoming more passionate about learning data analysis when I learned more and mastered mroe skills. I am excited that I can analyze a thing I want and use the result to help someone. Though the polynomial regression model might not have been perfect, I am glad I can develop a model and analyze a data set by myself. I'm confident that I will do better when I learn further about machine learning!

Sources

This data was taken from the Kaggle data set, "Airbnb Prices in European Cities," provided by user The Devastator.

The information about Vienna was found on Wekipedia.

Definition of each variables mentioned in the project were all found on the "Airbnb Prices in European Cities".