Lab 08 Machine Learning

RE 519 Data Analytics and Visualization | Autumn 2025

In Lab 8, we will briefly introduce decision tree (supervised) and k-means clustering (unsupervised), two classical machine learning methods. The due day for each lab can be found on the course wesbite. The submission should include Rmd, html, and any other files required to rerun the codes.

From Lab 4, the use of any generative AI tool (ChatGPT, Copilot, etc.) is **allowed** for coding assignments, but **not allowed** for write-up assignment. Meanwhile, I still encourage you to write codes by yourself and use AI tools as a way to debug and explore new knowledge. More information about Academic Integrityand the Use of AI.

Lab 08-A: Decision Trees

For the part A, we still use the **King County sales** data from maintained by Andy Krause at Zillow. You can find the datasets and Readme file in this repository. We are going to use decision trees to predict sale prices.

We will use `tidymodels`, which is a R package for modeling and machine learning in R using tidyverse principles. It provides us a framework.

```r
#install.packages("tidymodels") # you only need to install once
#install.packages("devtools") # if you are using a Windows computer
#devtools::install_github('andykrause/kingCoData') # you only need to install once
#install.packages("rpart.plot") # you only need to install once
library(rpart.plot) # the package for visualizing trees
library(tidymodels)
```

```
## Warning: package 'ggplot2' was built under R version 4.4.3
```

```r
library(kingCoData) # load the data package
```

```r
data(kingco_sales) # load the sale data
sales <- kingco_sales %>% # only select the sales in 2023
  filter(sale_date >= as.Date("2023-01-01") & sale_date <= as.Date("2023-12-31"))
```

Data Preparation

```r
sales_pred <- sales %>%
  mutate(view_mountains = view_rainier + view_olympics + view_cascades + view_territorial,
         view_water = view_sound + view_lakewash + view_lakesamm + view_otherwater)
# select some features (subjective here)
sales_pred <- sales_pred %>%
  dplyr::select(sale_price, area, city, year_built, sqft_lot, sqft, beds, view_skyline, view_mountains, view_water)
```

Before train the model, we need to split the dataset into a training set and a testing set. The training data will be used to train the models, while the testing data will be reserved to evaluate how well the model generalizes to unseen observations.

```r
set.seed(123)
split <- initial_split(sales_pred, prop = 0.8) # 80% training data
train_data <- training(split)
test_data  <- testing(split)
```

We are going to build a recipe for data preprocessing. A recipe is the steps to do data preparation under `tidymodels` framework. For Lasso and ridge, we need to normalize features. Please note that The object `rec` is a recipe that has **not been trained** on data yet (for example, does not normalize).

```r
rec <- recipe(sale_price ~ ., data = train_data) %>% # model and dataset
  step_zv(all_numeric())  # remove variables that contain only a single value
```

Model Specification

We start from specify the tree. In `tidymodels`, the model type (`decision_tree`) and the computational engine (`rpart`) are separated.

```r
tree_spec <- decision_tree(
  cost_complexity = tune(),  # CCP alpha
  tree_depth = tune(),       # max depth
  min_n = tune()             # min samples split
) %>%
  set_mode("regression") %>%
  set_engine("rpart")
```

A workflow is a container that bundles together a recipe (data preprocessing) and a model specification. use `workflow()` to put receipt and models together. We have not train yet!

```
tree_workflow <- workflow() %>%
  add_recipe(rec) %>%
  add_model(tree_spec)
```

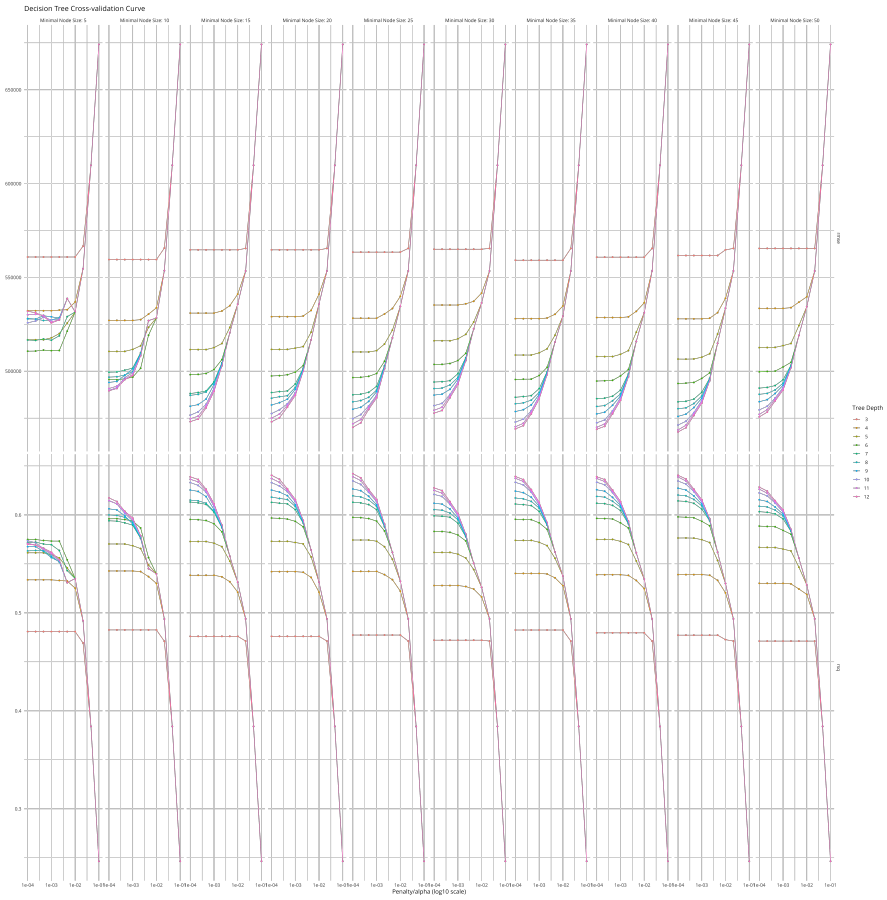
Tuning Grid for Penalty

We can define the search space for hyperparameters. Note: `range = c(a, b)` specifies the log10 scale from 10^a - 10^b.

```
tree_grid <- grid_regular(
  cost_complexity(range = c(-10, -3)),  # log10
  tree_depth(range = c(5, 25)),
  min_n(range = c(15, 40)),
  levels = 6 # try 6 values between the previous ranges
)
```

In this step, we have tried different ranges for hyperparameters. For example, we will see the following results when:

- cost_complexity(range = c(-4, -1))
- tree_depth(range = c(3, 12))
- min_n(range = c(5, 50))



Decision Tree Cross-validation Curve

Cross-validation Setup

We make 5 folds to separate the `train_data` into five equally sized subsets for cross-validation.

```
folds <- vfold_cv(train_data, v = 5)
```

Model Training via Grid Search

We will have 216 (6 * 6 * 6) different combinations of hyperparameters, so the training process will take a while.

```
tune_results <- tune_grid(
  tree_workflow,
  resamples = folds,
  grid = tree_grid,
  metrics = metric_set(rmse, rsq) # we evaluate two metrics: RMSE (error) and R² (fit)
)
```

Cross-validation Results

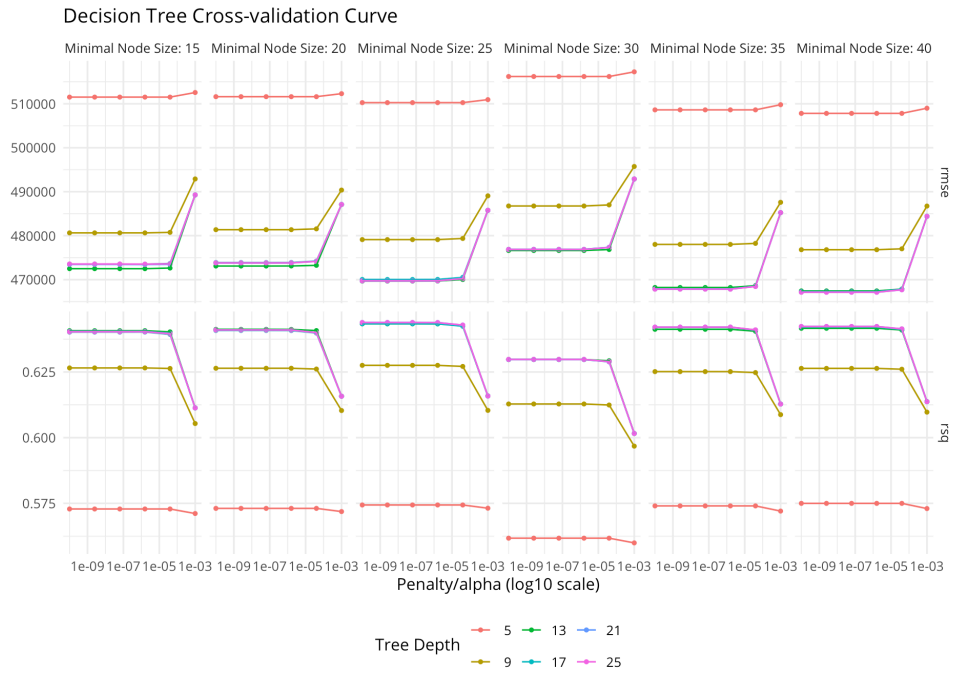
We can get the best hyperparameters in terms of RMSE.

```
best_params <- select_best(tune_results, metric = "rmse")
best_params
```

```
## # A tibble: 1 × 4
##   cost_complexity tree_depth min_n .config
##             <dbl>      <int> <int> <chr>
## 1      0.00000158         21    40 pre0_mod138_post0
```

We can show the model performance changes as hyperparameters change.

```
autoplot(tune_results)+
  labs(title = "Decision Tree Cross-validation Curve",
       x = "Penalty/alpha (log10 scale)") +
  theme_minimal(base_family = "opensans", base_size = 12) +
  theme(legend.position = "bottom")
```




```
# ggsave("results.png", dpi = 400, height=25, width = 25)
```

Fit the Final Model using the Best Hyperparameters

```
# another workflow
final_tree <- workflow() %>%
  add_model(finalize_model(tree_spec, best_params)) %>%
  # finalize_model() inserts the chosen hyperparameters into the model specification
  add_recipe(rec) %>% # the same recipe
  fit(train_data) # train the final Lasso model using the full training set
```

Let's visualize the classical tree figure for our model.

```
png("my_tree.png",
    width = 15000,
    height = 15000,
    res = 400)
rpart.plot(extract_fit_engine(final_tree),
           type = 2,
           extra = 101,
           under = TRUE,
           faclen = 0,
           roundint = FALSE,
           fallen.leaves = FALSE)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

Testing

We evaluate the performance of the final decision tree model on the held-out test set. The `predict()` function generates predicted sale prices for the test data, and `metrics()` computes evaluation measures such as RMSE and R^2.

```r
pred_tree <- predict(final_tree, test_data) %>%
  bind_cols(test_data)
metrics(pred_tree, truth = sale_price, estimate = .pred)
```

```
## # A tibble: 3 × 3
##   .metric .estimator  .estimate
##   <chr>   <chr>           <dbl>
## 1 rmse    standard   514025.
## 2 rsq     standard        0.628
## 3 mae     standard   232454.
```

📚 TODO: Decision Trees

You may discuss in small group, but you must prepare the submission by yourself. Generative AI is NOT allowed for write-up questions. You ask conceptual questions but you need to cite properly. How to cite?

Your submission should be HTML or PDF. Handwritten PDFs are acceptable as long as they are scanned and easy to read. Part B will not require any coding as well.

5 points

1. [1 pt] What is the purpose of pruning?
2. [1 pt] Why we call Cost Complexity Pruning (CCP) is post-pruning instead of pre-pruning?
3. [1 pt] A transportation agency is trying to distinguish the use of walking vs biking by residents in Seattle using the speed information obtained from each subject in a sample. Calculate GINI index for those 6 training samples. Walking (count = 2), Bike (count =4), Scooter (count = 1).
4. [2 pt] You are using a decision tree to predict whether a mortgage will be payback (a classification problem). Please think about at least **2 strategies** for each situation. Briefly explain the reason.
    1. Your model has high training error and high testing error.
    2. Your model has low training error but high testing error.

📚 TODO: Neural Network

4 points

In this small task, you will use ideas from the 3Blue1Brown video: But what is a Neural Network?, to build intuition about how neural networks works (only this video is required but you are encourage to watch more in the same list). You can also play with some neural networks on this website.

1. [1 pt] Do you think neural networks are interesting after watching this video? Why or why not?
2. [1 pt] Use the idea from the video to explain how the network breaks down the problem of recognizing a “3”, from pixels to digit.
3. [2 pt] The creator said “a neural network is just a function.” Given that many AI systems are built on neural networks, what responsibilities do developers and organizations have when deploying these “functions“ into real-world decisions? Maybe in terms of

fairness, equity, transparency, accountability or societal impact.

📚 TODO: Simulate a Simple Neural Network

2 points

Using the fully connected neural network below, what is the prediction for the data point $x = (4, -1)$? Assume every neuron (including the last one) uses the following activation function:

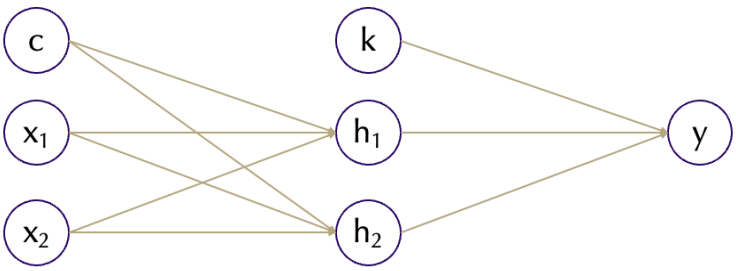
$$g(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$$

Assume the network uses the following weights into hidden neurons:

input	h[1]	h[2]
c (constant)	2	-3
x[1]	-1	2
x[2]	3	1

Assume the network uses the following weights into output neuron:

input	y
k (constant)	1
h[1]	-2
h[2]	4



Neural Network Diagram

Lab 08-B: K-means Clustering

In the part B, we will use homes located in Kirkland to conduct a simple k-means clustering. `kmeans()` is a built-in function for `R`. In addition, we use `plotly` for 3D interactive charts. You can check the examples of `plotly` here.

```
#install.packages(c('plotly')) # you only need to install once
library(plotly)
```

Data Preparation

We select only three features of house in Kirkland to cluster: `year_built`, `sqft`, and `beds`. Note: I tried to cluster for the whole King County but it requires more memory than I have (2021 MacBook Pro 16 GB).

```
data(kingco_homes)
homes <- kingco_homes %>%
  filter(city == "KIRKLAND") %>%
  select(pinx, year_built, sqft, beds) %>%
  distinct() # remove replicated rows; pinx ensures unique parcels
X <- homes %>%
  select(year_built, sqft, beds) %>%  # only some variables are kept for clustering
  scale() # normalization is required
```

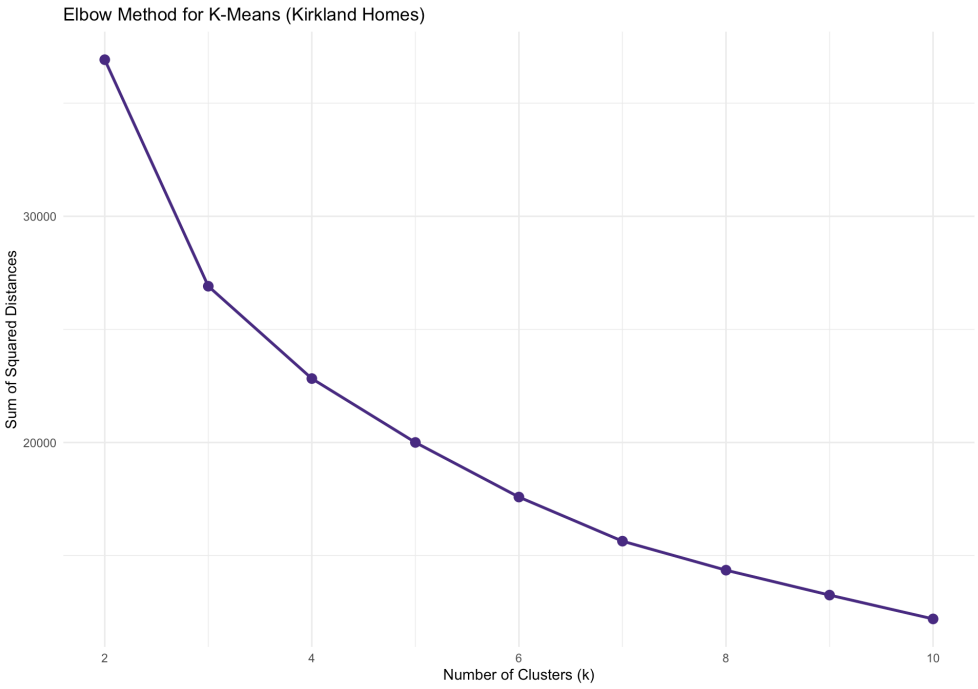
Choosing K (Elbow Method)

```
set.seed(123)
ssd <- c() # ssd = total within-cluster sum of squares
for (k in 2:10) { # a for loop: test k from 2 to 10
  km <- stats::kmeans(X, centers=k, nstart=20, iter.max = 100)
  # nstart: runs 20 random initializations to improve stability
  # iter.max: allow max 50 time iterations
  ssd <- c(ssd, km$tot.withinss) # add the individual ssd to this vector
}
```

This course does not mention for or while loops, which is a useful operations. Please check Section 7.5 Loops in An Introduction to R.

```
df_elbow <- data.frame(
  k = 2:10,
  ssd = ssd
)

ggplot(df_elbow, aes(x = k, y = ssd)) +
  geom_point(size = 3, color = "#4B2E83") +
  geom_line(linewidth = 1, color = "#4B2E83") +
  theme_minimal() +
  labs(
    title = "Elbow Method for K-Means (Kirkland Homes)",
    x = "Number of Clusters (k)",
    y = "Sum of Squared Distances"
  )
```



Look for the "elbow": the point where adding more clusters yields diminishing returns. The elbow may not very clear in some cases, but k = 3 is a reasonable choice. Also, moving from k = 3 to 4 still gives a noticeable improvement.

Fit Final K-Means Model

We will use k = 3 to do clustering.

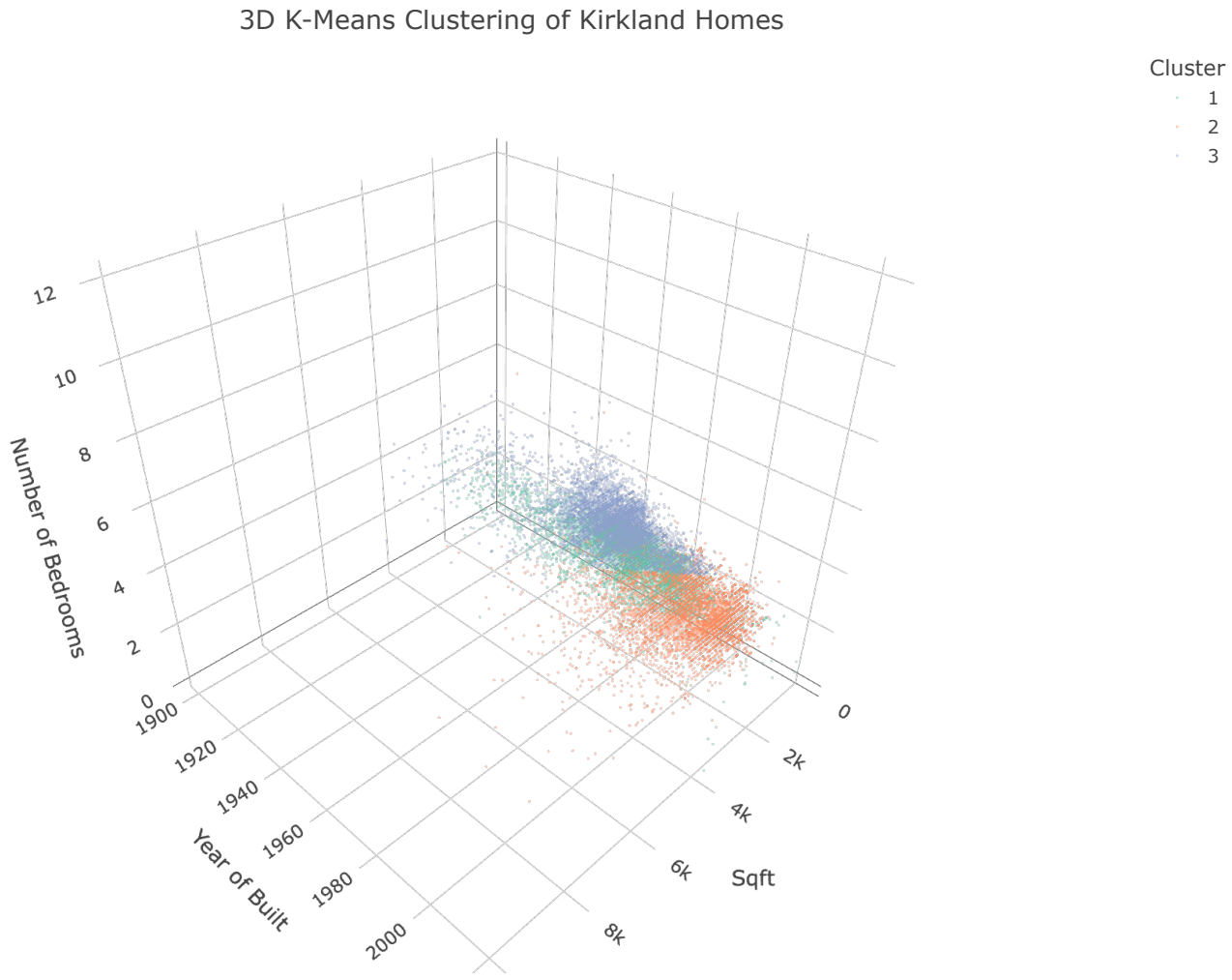
```
km <- stats::kmeans(X, centers = 3, nstart = 20) # clustering again
homes_clustered <- homes %>% # add cluster labels back to the original homes dataset
    mutate(cluster = factor(km$cluster))
```

3D Visualization

We visualize clusters along the three selected features using `plotly` package. Besides `plotly`, there are many other packages for interactive visualization in R, such as `leaflet` (for maps) and `highcharter` (for 2-D visualization).

```
plot_ly(
  homes_clustered,
  x = ~sqft,
  y = ~year_built,
  z = ~beds,
  color = ~cluster,
  colors = "Set2",
  type = "scatter3d",
  mode = "markers",
  marker = list(
    size = 1,
    opacity = 0.4, # transparency
    line = list(width = 0) # remove marker border
  ))%>%
  layout(
    title = "3D K-Means Clustering of Kirkland Homes",
    scene = list(
```

```
    xaxis = list(title = "Sqft"),
    yaxis = list(title = "Year of Built"),
    zaxis = list(title = "Number of Bedrooms")),
  legend = list(title = list(text = "Cluster"))
)
```



📚 TODO: Review the Code for Machine Learning

1 point

Carefully review and run all code chunks. You are encouraged to modify the model. Reply "yes" once you have reviewed them.

📚 TODO: K-means Clustering

4 points

1. [1 pt] You are suggesting machine learning methods/categories to your friend, please suggest at least one for each case.
    1. A planner wants to visualize patterns in a dataset with 15 community features per community. They just want to plot communities in a 2D space to better explore patterns.
    2. Predict whether a building permit should be treated as commercial, residential, or

industrial based on application text and metadata.

2. [2 pt] True or false? Explanation is optional.
    1. If you run k-means multiple times using the same initial cluster centers, the resulting clusters will be the same.
    2. K-means clustering will always converge.
    3. K-means clustering will always converge to a global optimum.
    4. K-means clustering is sensitive to feature scale.
3. [1 pt] We get the clustering of Kirkland property, what types of analysis we can do next to understand those clusters?

Acknowledgement

The materials are developed by Haoyu Yue based materials from Dr. Feiyang Sun at UC San Diego, Siman Ning and Christian Phillips at University of Washington, Dr. Charles Lanfear at University of Cambridge.