HW04

2024-03-25

```r
library(ggmap)
```

```
## Loading required package: ggplot2

## i Google's Terms of Service: <https://mapsplatform.google.com>
##   Stadia Maps' Terms of Service: <https://stadiamaps.com/terms-of-service/>
##   OpenStreetMap's Tile Usage Policy: <https://operations.osmfoundation.org/policies/tiles/>
## i Please cite ggmap if you use it! Use `citation("ggmap")` for details.
```

```r
library(osmdata)
```

```
## Data (c) OpenStreetMap contributors, ODbL 1.0. https://www.openstreetmap.org/copyright
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v stringr   1.5.0
## v lubridate 1.9.2     v tibble    3.2.1
## v purrr     1.0.1     v tidyr     1.3.0
## v readr     2.1.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```r
library(tidyr)
library(rsample)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```r
library(rpart)
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.9
## This version of gbm is no longer under development. Consider transitioning to gbm3, https://github.co
```

```r
library(ggplot2)
```

```r
# Q3
# Data preprocessing
green = read_csv('/Users/vita/Desktop/greenbuildings.csv', show_col_types = FALSE)
green$revenue = green$Rent *green$leasing_rate

green_split = initial_split(green, prop=0.8)
green_train = training(green_split)
green_test  = testing(green_split)

# LEED and EnergyStar collapse them into a single "green certified" category
green_train = green_train %>%
  mutate(green_certified = ifelse(LEED == 1 | Energystar == 1, 1, 0))

green_test = green_test %>%
  mutate(green_certified = ifelse(LEED == 1 | Energystar == 1, 1, 0))
```

```r
# Model training
# use all available variables in green_train as predictors except for Rent and leasing_rate
# a.cart model
cart_model <- rpart(revenue ~ . - Rent - leasing_rate, data = green_train, method = "anova")

# b.Gradient Boosting Model
gbm_model <- gbm(revenue ~ . - Rent - leasing_rate, data = green_train, distribution = "gaussian", n.tre

# c.Linear Model
lm_model <- lm(revenue ~ . - Rent - leasing_rate, data = green_train)
```

```r
# Model evaluation
predictions_cart <- predict(cart_model, newdata = green_test)
predictions_gbm <- predict(gbm_model, newdata = green_test, n.trees = 500)
predictions_lm <- predict(lm_model, newdata = green_test)

# Calculate RMSE for each model
```

```r
rmse_cart <- sqrt(mean((green_test$revenue - predictions_cart)^2))
rmse_gbm <- sqrt(mean((green_test$revenue - predictions_gbm)^2))
rmse_lm <- sqrt(mean((predictions_lm - green_test$revenue)^2, na.rm = TRUE))

print(paste("CART RMSE:", rmse_cart))
```

```
## [1] "CART RMSE: 1104.15241109812"
```

```r
print(paste("Gradient Boosting RMSE:", rmse_gbm))
```

```
## [1] "Gradient Boosting RMSE: 907.455516541811"
```

```r
print(paste("LM RMSE:", rmse_lm))
```

```
## [1] "LM RMSE: 1101.32423183007"
```

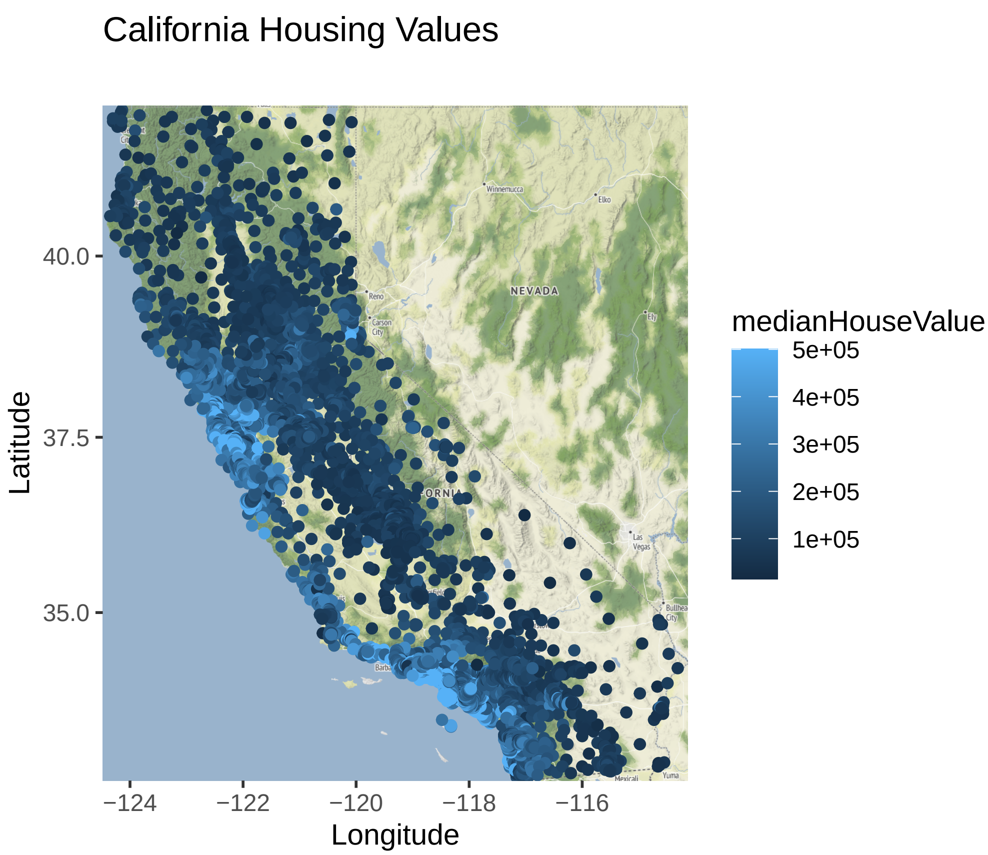
```r
# Answer:
# The initial step involved cleaning the dataset to handle missing values and create new features. Nota
# The analysis employed three distinct modeling approaches:CART: Served as a foundational model to esta
# Result: The models were evaluated based on their Root Mean Squared Error (RMSE): Gradient Boosting ha
```

```r
## Q4: Predictive model building: California housing
# get API key
register_stadiamaps(key = "ff75cbd1-a355-4aba-9135-e12bd22345f9")
CAmap = get_stadiamap( getbb('california'), source="stadia", zoom = 7)
```

```
## i © Stadia Maps © Stamen Design © OpenMapTiles © OpenStreetMap contributors.
```

```r
# (1) Original data plot
housing_data = read.csv('/Users/vita/Desktop/CAhousing.csv')

# plot
ggmap(CAmap) +
  geom_point(aes(x = longitude, y = latitude, color = medianHouseValue),
             data = housing_data) +
  labs(x = 'Longitude', y = 'Latitude', title = 'California Housing Values', subtitle = '')
```




```r
# data preprocessing
housing_data$bedroomsPerHousehold = housing_data$totalBedrooms /housing_data$households
housing_data$roomsPerHousehold = housing_data$totalRooms / housing_data$households
ca_split = initial_split(housing_data, 0.8)
ca_train = training(ca_split)
ca_test = testing(ca_split)
```

```r
# (2) Model's predictions
# Model training
# a.cart mode
cart_model = rpart(medianHouseValue ~ ., data = ca_train, method = "anova")
# b.Random Forest Model
rf_model = randomForest(medianHouseValue ~ ., data = ca_train, ntree = 500)
# c.Gradient Boosting Model
gbm_model= gbm(medianHouseValue ~ ., data = ca_train, distribution = "gaussian", n.trees = 500, interac
```

```r
# Model evaluation
# Predictions
predictions_cart = predict(cart_model, newdata = ca_test)
predictions_rf = predict(rf_model, newdata = ca_test)
predictions_gbm = predict(gbm_model, newdata = ca_test, n.trees = 500)

# Calculate RMSE
rmse_cart = sqrt(mean((ca_test$medianHouseValue - predictions_cart)^2))
rmse_rf = sqrt(mean((ca_test$medianHouseValue - predictions_rf)^2))
rmse_gbm = sqrt(mean((ca_test$medianHouseValue - predictions_gbm)^2))
```

```r
# Print RMSE values
print(paste("CART RMSE:", rmse_cart))
```

```
## [1] "CART RMSE: 84231.1316095992"
```

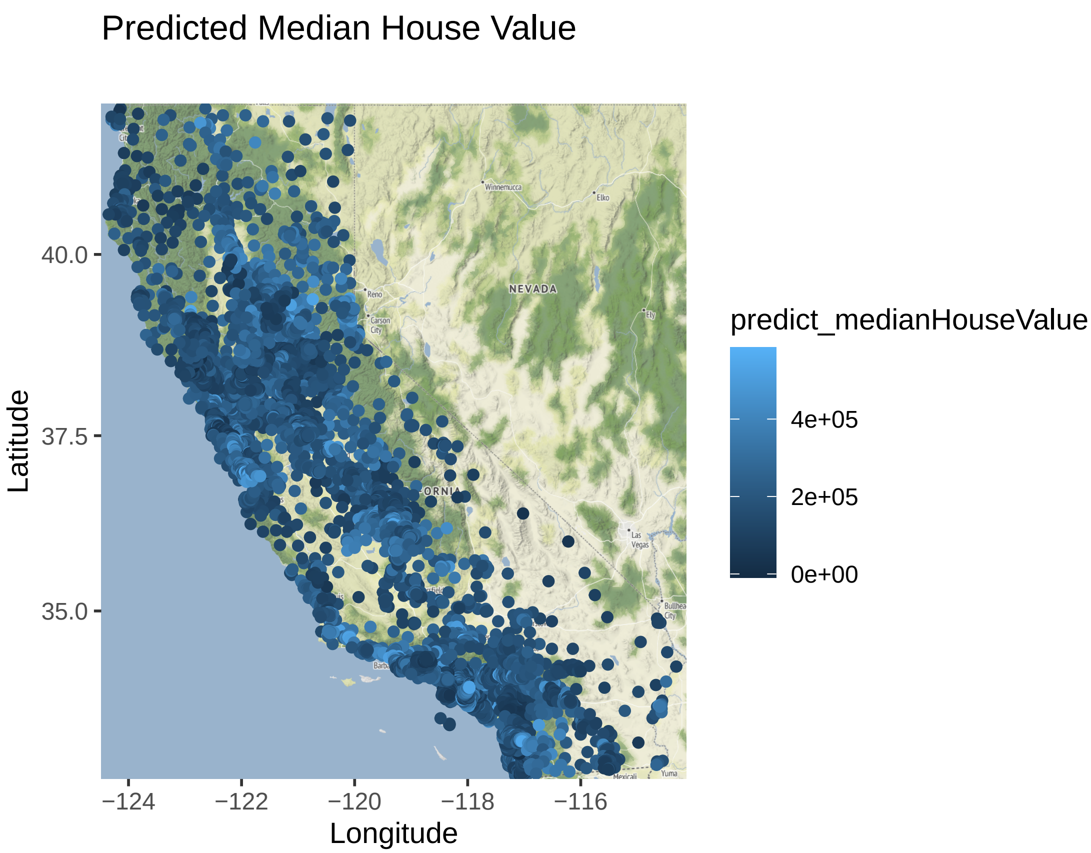
```r
print(paste("Random Forest RMSE:", rmse_rf))
```

```
## [1] "Random Forest RMSE: 51310.074303095"
```

```r
print(paste("Gradient Boosting RMSE:", rmse_gbm))
```

```
## [1] "Gradient Boosting RMSE: 50726.3173707887"
```

```r
### Answer:The best performance based on RMS is Gradient Boosting
# plot with prediction
# Add Predictions to Dataset
housing_data$predict_medianHouseValue = predictions_gbm
ggmap(CAmap) +
  geom_point(aes(x = longitude, y = latitude, color = predict_medianHouseValue),
             data = housing_data) +
  labs(x = 'Longitude', y = 'Latitude', title = "Predicted Median House Value", subtitle = '')
```




```r
# Report: At data preprocessing stage (1) Standardization: Given that total rooms and bedrooms are aggr
# Result: The GBM model emerged as the most effective, demonstrating superior predictive accuracy.
```