~~~
---
title: "California Housing Price Prediction using R"
author: "Anoop Reddy"
date: "2024-02-18"
output: html_document
---

##Installing packages

-   **CARET** - **C**lassification **A**nd **RE**gression **T**raining

-   **ggplot2** - Grammar of Graphics plot v2

-   **scales** - Used for alter the limits of a graph

-   **corrplot** - Alternative to ggplot heatmap

```{r packages, include=FALSE}
library(caret)
library(ggplot2)
library(scales)
library(corrplot)
```

-   Reading CSV
-   Exclude irrelevant features
-   Splitting target and features

```{r Reading csv, excluding irrelevant features, splitting target and features}
housing=read.csv("housing.csv")
housing = na.omit(housing)
housing=housing[, !names(housing) %in% c("ocean_proximity")]

set.seed(69696)

target=housing$median_house_value
target_column="median_house_value"
features=housing[,setdiff(names(housing),target_column)]
```

## Heatmap with corrplot

```{r corrplot, echo=FALSE}
cor_matrix = cor(housing)
corrplot(
  cor_matrix,
  method = "color",
  tl.col = "black",
  tl.srt = 45,
  number.cex = 0.7,
  addCoef.col = "black"
)
```

## Heatmap with ggplot2

```{r ggplot2, echo=FALSE}
melted_cor_matrix = melt(cor_matrix)

ggplot(melted_cor_matrix, aes(Var1, Var2, fill = value)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "green", high = "blue") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
~~~

~~~
  geom_text(aes(label = round(value, 2)), vjust = 1) +
  labs(title = "Correlation Heatmap", x = "Variables", y = "Variables")

```

## Splitting Training and Testing sets

-   Training - 75%
-   Testing - 25%

```{r training set}

trainingIndex=createDataPartition(housing[[target_column]],p=0.75,list=FALSE)
x_train=features[trainingIndex,]
y_train=housing$median_house_value[trainingIndex]
x_test=features[-trainingIndex,]
y_test=housing$median_house_value[-trainingIndex]

```

## Linear Regression Model

```{r LR}
lm_model = lm(y_train ~ ., data = cbind(x_train, y_train))
predictions = predict(lm_model, newdata = x_test)
comparison = data.frame(Actual = y_test, Predicted = predictions)

```

## Compare Actual and Predicted values

```{r compare}
print(comparison)

```

## Evaluating Metrics (MSE and RMSE)

```{r metrics}
mse = mean((predictions - y_test)^2)
rmse = sqrt(mse)
print(mse)
print(rmse)

```

## Scatter Plot with regression line

```{r scatter}
ggplot(comparison, aes(x = Actual, y = Predicted)) +
  geom_point(color = "blue", alpha = 0.7) +
  geom_smooth(method = "lm", se = FALSE, linetype = "dashed", color = "red") +
  ggtitle("Actual vs. Predicted Values") +
  xlab("Actual Values") +
  ylab("Predicted Values") +
  scale_x_continuous(labels = label_number(), breaks = seq(0, max(comparison$Actual), by =
100000)) +
  scale_y_continuous(labels = label_number(), breaks = seq(0, max(comparison$Predicted),
by = 100000))

```

## Contour Plot for density and prediction outlier mapping

-   Contour plots are used for multi-variate mapping, mostly *3 Dimensional* to determine
~~~

~~~
if a variable is influencing other variables.
-   Ignore the *legend "level"* as the function thinks we are using 3 variables.

```{r contour}
ggplot(comparison, aes(x = Actual, y = Predicted)) +
  geom_density_2d_filled(color = "blue", alpha = 0.7) +
  ggtitle("Contour Plot of Actual vs. Predicted Values") +
  xlab("Actual Values") +
  ylab("Predicted Values") + scale_x_continuous(labels = label_number(), breaks = seq(0,
max(comparison$Actual), by = 100000)) +
  scale_y_continuous(labels = label_number(), breaks = seq(0, max(comparison$Predicted),
by = 100000))

```
~~~