Trabajo ML1: Parte 1 y 2

Jose Castro

2024-06-10

Parte 1: Modelamiento (Item 1)

1. Describa en detalle cada uno de los siguientes métodos de remuestreo

Método de Remuestreo K-fold Cross Validation

El método de K-fold cross validation permite ajustar repetidamente un modelo en diferentes partes del conjunto de entrenamiento y probar su rendimiento en otras partes del mismo conjunto.

K-fold Cross Validation

Es el remuestreo estándar de la industria para estudiar el rendimiento futuro del modelo.

Pasos del K-fold Cross Validation:

1. **División de los Datos**:
   - Divide aleatoriamente los datos de entrenamiento en (K) grupos del mismo tamaño (aproximadamente).
2. **Ajuste del Modelo**:
   - Ajusta el modelo utilizando (K-1) grupos, es decir, todos menos uno. El grupo restante (grupo de validación) se utiliza para calcular el rendimiento del modelo.
3. **Rotación del Conjunto de Validación**:
   - En las siguientes repeticiones, un grupo diferente se trata como conjunto de validación. Esto significa que cada grupo se utilizará una vez como conjunto de validación.
4. **Cálculo del Error de Generalización**:
   - Promedia las (k) estimaciones del error de generalización (e_1, e_2, e_3, ... e_k), que es la diferencia entre el valor predicho y el observado.
   - Este método proporciona una aproximación del error para los datos no observados.

- En general, se dice que con ($K > 10$), se alcanzan resultados similares al caso extremo en que el número de folds es igual al número de observaciones en los datos ((K=n)), llamado leave-one-out CV (LOOCV).
- Es útil realizar el procedimiento de K-fold cross validation repetidamente, es decir, varios K-fold cross validations, para que la distribución de los valores de la variable objetivo en los folds sea lo más representativa posible de la data original. Este método ayuda a aumentar la precisión del error de generalización estimado.
- Aunque un (K) mayor o igual a 10 minimiza la variabilidad del rendimiento estimado, K-fold CV tiende a tener una mayor variabilidad que el bootstrapping.

Método de Remuestreo: Bootstrapping

Muestra Aleatoria con Reemplazo

En bootstrapping, se toman múltiples muestras aleatorias con reemplazo del conjunto de datos de entrenamiento. Esto significa que cada muestra (bootstrap sample) puede tener filas repetidas y algunas filas pueden no estar presentes. Cada muestra tiene el mismo tamaño que el conjunto de datos original.

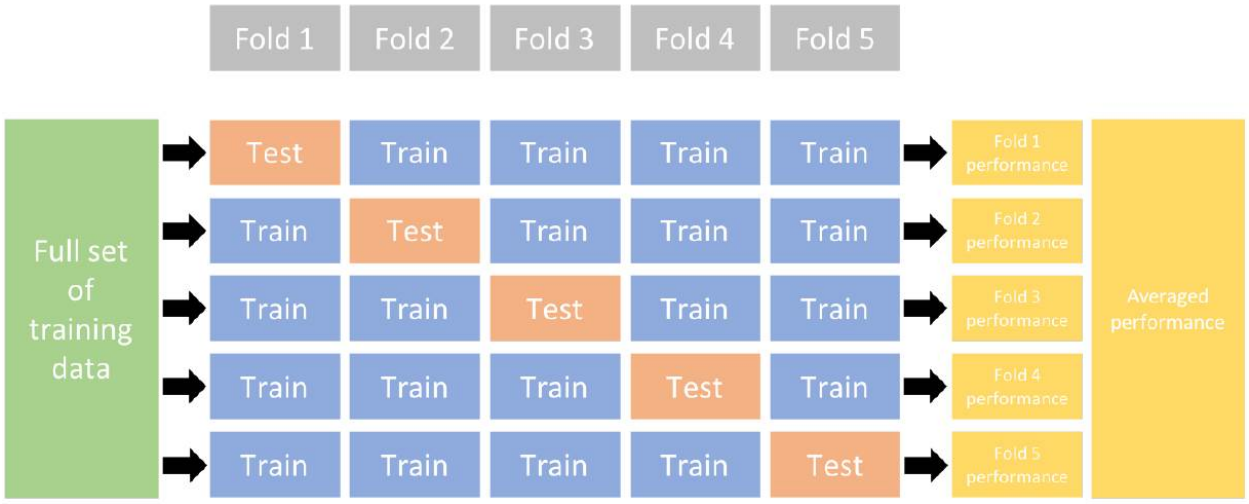


Figure 1: diagrama de k-fold CV

Filas Out-of-Bag (OOB)

Las observaciones que no se incluyen en una muestra particular se denominan filas out-of-bag (OOB). Estas filas se utilizan para evaluar el rendimiento del modelo. **(conjunto de validación)**

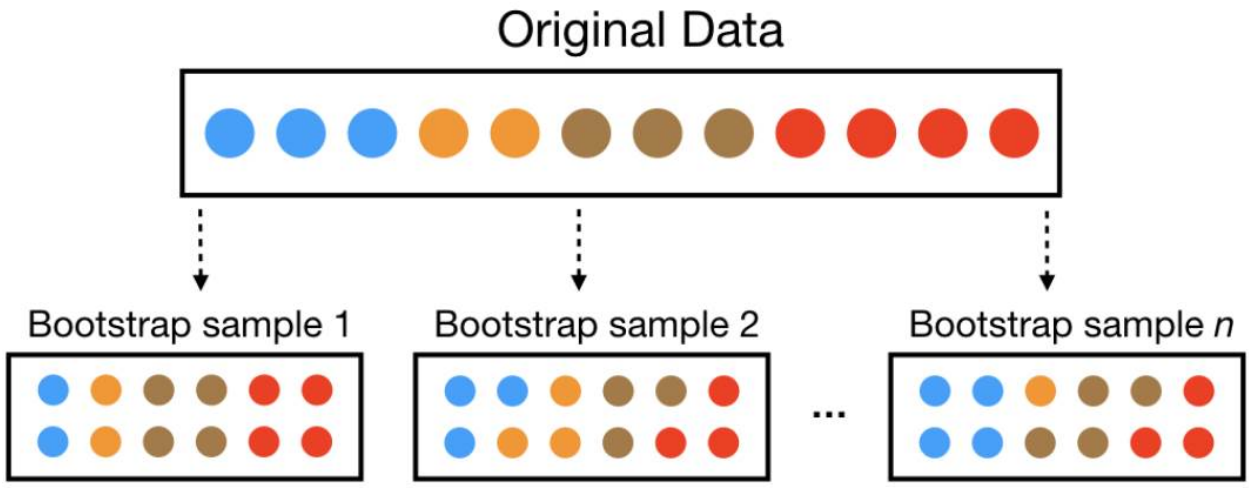


Figure 2: diagrama de Bootstrapping

Proceso de Bootstrapping

El proceso de bootstrapping para evaluar el rendimiento de un modelo incluye los siguientes pasos:

1. **Generación de Muestras de Bootstrap**: Se generan múltiples muestras de bootstrap a partir del conjunto de datos de entrenamiento.
2. **Entrenamiento del Modelo**: Para cada muestra de bootstrap, se entrena un modelo.
3. **Evaluación con Filas OOB**: Se evalúa el rendimiento del modelo utilizando las filas OOB correspondientes a esa muestra de bootstrap. **(conjunto de validación)**
4. **Promedio de Evaluaciones**: Se repite el proceso múltiples veces y se promedian las evaluaciones para obtener una estimación del rendimiento del modelo.

División Inicial de Datos: Se divide el conjunto de datos original en conjunto de entrenamiento y conjunto

de prueba mediante muestreo aleatorio simple o estratificado.

Generación de Muestras de Bootstrap:

- Supongamos que el conjunto de entrenamiento tiene 8000 filas.
- Se genera la primera muestra de bootstrap tomando 8000 filas aleatorias con reemplazo del conjunto de entrenamiento. Algunas filas pueden repetirse y otras pueden faltar (OOB).
- Se repite este proceso para generar múltiples muestras de bootstrap (e.g., 1000 muestras).

Entrenamiento del Modelo:

- Para cada muestra de bootstrap, se entrena un modelo usando solo las filas dentro de esa muestra.

Evaluación con Filas OOB:

- Para cada modelo entrenado, se utiliza el conjunto de filas OOB para evaluar el rendimiento del modelo.
- Dado que las filas OOB no se utilizaron para entrenar el modelo, proporcionan una estimación independiente del rendimiento del modelo, similar a cómo funcionan los conjuntos de validación en el método k-fold CV.
- Se calculan métricas de rendimiento (e.g., error cuadrático medio, precisión, recall) comparando las predicciones del modelo con los valores observados en las filas OOB.

Promedio de Evaluaciones:

- Se repite el proceso de entrenamiento y evaluación para todas las muestras de bootstrap.
- Se promedian las métricas de rendimiento obtenidas de todas las evaluaciones con filas OOB para obtener una estimación final del rendimiento del modelo.

Consideraciones

- **Tamaño del Conjunto de Datos**: Bootstrapping puede ser problemático en conjuntos de datos más pequeños porque las filas repetidas y faltantes son más relevantes, lo que puede inducir a errores mayores de predicción.

Detalle sobre Variabilidad y Sesgo en Bootstrapping y K-fold Cross-Validation

Contexto y Definiciones:

- **Variabilidad**: Se refiere a cuánto varían las estimaciones del modelo cuando se entrenan en diferentes subconjuntos de datos. Alta variabilidad significa que las predicciones del modelo pueden cambiar considerablemente con diferentes muestras de datos.
- **Sesgo**: Es el error sistemático introducido por un modelo que no puede captar la complejidad de los datos. Un alto sesgo significa que el modelo hace suposiciones simplistas y no se ajusta bien a los datos.

Explicación del Bootstrapping:

- **Muestra Aleatoria con Reemplazo**:
  - En bootstrapping, se generan múltiples muestras de bootstrap del conjunto de datos de entrenamiento original.
  - Cada muestra de bootstrap se selecciona con reemplazo, lo que significa que algunas filas pueden aparecer múltiples veces en la misma muestra, mientras que otras filas pueden no aparecer en absoluto.
- **Menor Variabilidad**:
  - Debido a que cada muestra de bootstrap es una variación del conjunto de datos de entrenamiento original con algunas filas duplicadas, las muestras tienden a ser menos diversas comparadas con los folds en k-fold CV.
  - Esto significa que las muestras de bootstrap pueden ser más similares entre sí, lo que lleva a una menor variabilidad en las estimaciones del modelo. En otras palabras, los modelos entrenados en diferentes muestras de bootstrap tienden a producir predicciones más consistentes.
- **Mayor Sesgo**:

  - La duplicación de filas y la ausencia de algunas filas en cada muestra de bootstrap pueden introducir un sesgo en la medida del error.
  - Esto ocurre porque las muestras de bootstrap no son completamente representativas del conjunto de datos original debido a la repetición de algunas filas y la exclusión de otras. Como resultado, el modelo puede estar ajustado a un subconjunto menos diverso de datos, lo que lleva a un mayor sesgo en las estimaciones del error.

Comparación con k-fold Cross-Validation:

- **Muestra Aleatoria sin Reemplazo**:
  - En k-fold CV, el conjunto de datos de entrenamiento se divide en (k) partes (folds o carpetas) de manera que cada fold es aproximadamente del mismo tamaño y las divisiones no se superponen.
  - Cada fold se utiliza una vez como conjunto de validación, mientras que los (k-1) folds restantes se utilizan como conjunto de entrenamiento.
- **Mayor Variabilidad**:
  - Debido a que cada fold de validación en k-fold CV es único y no contiene duplicados, las diferentes combinaciones de datos de entrenamiento y validación tienden a ser más diversas.
  - Esto introduce una mayor variabilidad en las estimaciones del modelo, ya que cada fold proporciona una vista diferente de los datos. Los modelos entrenados en diferentes combinaciones de folds pueden producir predicciones más variadas.
- **Menor Sesgo**:
  - La ausencia de duplicación y la inclusión de todas las filas en algún punto del proceso de validación en k-fold CV hacen que las estimaciones del modelo sean menos sesgadas.
  - Cada fold de validación es una representación más fiel del conjunto de datos original. Esto reduce el sesgo en las estimaciones del error, proporcionando una medida más precisa del rendimiento del modelo.

Ejemplo Ilustrativo: Imaginemos un conjunto de datos con 10 observaciones: A, B, C, D, E, F, G, H, I, J.

- **Bootstrapping**:
  - Muestra de Bootstrap 1: A, B, B, D, E, F, F, H, I, J (observaciones B y F repetidas; C y G faltantes).
  - Muestra de Bootstrap 2: A, A, C, D, E, G, H, I, I, J (observaciones A e I repetidas; B y F faltantes).
- **k-fold Cross-Validation (k=5)**:
  - Fold 1: (A, B) (entrenamiento); (C, D, E, F, G, H, I, J) (validación).
  - Fold 2: (C, D) (entrenamiento); (A, B, E, F, G, H, I, J) (validación).

En bootstrapping, algunas observaciones están repetidas y otras faltan, lo que reduce la diversidad entre las muestras y aumenta el sesgo. En k-fold CV, cada fold es una representación más única y completa del conjunto de datos, lo que aumenta la variabilidad y reduce el sesgo.

Método de Remuestreo Jackknife

El método de remuestreo Jackknife es una técnica estadística utilizada para estimar la variabilidad de un estimador, como la media o la varianza. Este método es particularmente útil para evaluar el sesgo y la varianza de los estimadores en muestras pequeñas. A continuación, se detalla el método, se proporciona un ejemplo, se discuten sus pros y contras, se explica la matemática/estadística subyacente y se compara con los métodos de K-fold Cross Validation y Bootstrapping.

Proceso del Método Jackknife

El método Jackknife se basa en la eliminación sistemática de una observación a la vez del conjunto de datos y en el cálculo del estimador en cada subconjunto de datos resultante. Los pasos generales son los siguientes:

División Inicial

Supongamos que tenemos un conjunto de datos con n observaciones.

Generación de Submuestras

Se crean n submuestras, cada una de las cuales contiene $n-1$ observaciones. Cada submuestra se forma eliminando una observación diferente del conjunto de datos original.

Cálculo del Estimador

Para cada submuestra, se calcula el estimador de interés (e.g., media, varianza).

Cálculo del Estimador Jackknife

Se promedian los estimadores calculados para obtener una estimación del sesgo y la varianza del estimador original.

Ejemplo de Aplicación del Método Jackknife

Supongamos que tenemos un conjunto de datos con 5 observaciones:

$$10, 20, 30, 40, 50$$

.

Paso 1: Generación de Submuestras

- Submuestra 1:
$$20, 30, 40, 50$$
(eliminamos la primera observación)
- Submuestra 2:
$$10, 30, 40, 50$$
(eliminamos la segunda observación)
- Submuestra 3:
$$10, 20, 40, 50$$
(eliminamos la tercera observación)
- Submuestra 4:
$$10, 20, 30, 50$$
(eliminamos la cuarta observación)
- Submuestra 5:
$$10, 20, 30, 40$$
(eliminamos la quinta observación)

Paso 2: Cálculo del Estimador (Media)

- Media de la submuestra 1: $\frac{20+30+40+50}{4} = 35$
- Media de la submuestra 2: $\frac{10+30+40+50}{4} = 32.5$
- Media de la submuestra 3: $\frac{10+20+40+50}{4} = 30$
- Media de la submuestra 4: $\frac{10+20+30+50}{4} = 27.5$
- Media de la submuestra 5: $\frac{10+20+30+40}{4} = 25$

Paso 3: Cálculo del Estimador Jackknife

- Media de todas las medias: $\frac{35+32.5+30+27.5+25}{5} = 30$

Varianza Jackknife:

$$Var(\hat{\theta}) = \frac{n-1}{n} \sum_{i=1}^{n} (\hat{\theta}_i - \bar{\theta})^2$$

donde $\hat{\theta}_i$ es la media de cada submuestra y $\bar{\theta}$ es la media de todas las medias.

Pros y Contras del Método Jackknife

Pros:

- **Simplicidad**: Fácil de implementar y entender.
- **Sesgo Reducido**: Ofrece una buena estimación del sesgo del estimador.
- **Eficiencia Computacional**: Menos intensivo computacionalmente en comparación con el bootstrapping.

Contras:

- **Subestimación de la Varianza**: Puede subestimar la varianza del estimador en muestras pequeñas.
- **Sensibilidad a Outliers**: Es sensible a la presencia de outliers en el conjunto de datos.

Matemática/Estadística Subyacente

El método Jackknife se basa en la idea de eliminar una observación a la vez y calcular el estimador de interés en cada subconjunto resultante. La fórmula de la varianza Jackknife es:

$$Var(\hat{\theta}) = \frac{n-1}{n} \sum_{i=1}^{n} (\hat{\theta}_i - \bar{\theta})^2$$

donde n es el número total de observaciones, $\hat{\theta}_i$ es el estimador calculado en la i-ésima submuestra, y $\bar{\theta}$ es la media de todos los $\hat{\theta}_i$.

Comparación de Jackknife con K-fold Cross Validation y Bootstrapping

Jackknife vs. K-fold Cross Validation: {#jackknife-vs-k-fold-cross-validation}

- **Enfoque**: Jackknife elimina una observación a la vez, mientras que K-fold CV divide los datos en k grupos.
- **Variabilidad**: K-fold CV puede proporcionar una mejor estimación de la variabilidad del modelo, ya que utiliza múltiples divisiones.
- **Uso**: K-fold CV es más adecuado para evaluar el rendimiento predictivo de los modelos de machine learning, mientras que Jackknife se utiliza más para estimar el sesgo y la varianza de los estimadores.

Jackknife vs. Bootstrapping: {#jackknife-vs-bootstrapping}

- **Enfoque**: Bootstrapping genera múltiples muestras con reemplazo, mientras que Jackknife elimina una observación a la vez.
- **Variabilidad y Sesgo**: Bootstrapping tiende a proporcionar una mejor estimación de la variabilidad del modelo, mientras que Jackknife es más sencillo y menos intensivo computacionalmente.
- **Precisión**: Bootstrapping generalmente proporciona estimaciones más precisas de la varianza y el sesgo, especialmente en muestras pequeñas.

El método Jackknife es una técnica de remuestreo útil para estimar la variabilidad y el sesgo de un estimador, especialmente en conjuntos de datos más pequeños. Aunque tiene algunas limitaciones, su simplicidad y eficiencia computacional lo hacen una herramienta valiosa en el análisis estadístico. Comparado con K-fold Cross Validation y Bootstrapping, el Jackknife es menos intensivo computacionalmente pero puede subestimar la varianza en muestras pequeñas y es más sensible a outliers.

Comparativa Detallada entre K-fold Cross Validation, Bootstrapping y Jackknife

En el contexto de machine learning, los métodos de remuestreo son fundamentales para entrenar los modelos. Tres de estos metodos son K-fold Cross Validation, Bootstrapping y Jackknife. A continuación, se presenta una comparativa detallada de estos métodos, considerando aspectos como varianza, sesgo, precisión, eficiencia computacional, iteraciones, muestras, filas u observaciones del conjunto de validación y conjunto de entrenamiento.

K-fold Cross Validation {#k-fold-cross-validation}

Descripción

K-fold Cross Validation (CV) divide el conjunto de datos en k grupos (folds) de tamaño aproximadamente igual. El proceso implica entrenar el modelo k veces, cada vez utilizando $k-1$ folds para entrenamiento y el fold restante para validación.

Parámetros

- **Número de folds (k)**: Típicamente se usa $k = 5$ o $k = 10$.

Ventajas

- **Varianza**: Proporciona una buena estimación de la variabilidad del modelo ya que utiliza múltiples divisiones.
- **Sesgo**: Tiende a tener un sesgo bajo debido al uso de múltiples folds.
- **Precisión**: Ofrece una estimación precisa del rendimiento del modelo.
- **Simplicidad**: Relativamente fácil de implementar y entender.
- **Conjunto de Entrenamiento y Validación**: Utiliza $\frac{k-1}{k}$ del conjunto de datos para entrenamiento y $\frac{1}{k}$ para validación en cada iteración.

Desventajas

- **Eficiencia Computacional**: Más intensivo computacionalmente que el Jackknife debido a las múltiples iteraciones.
- **Iteraciones**: Requiere k iteraciones, lo que puede ser costoso para valores grandes de k.

Bootstrapping

Descripción

Bootstrapping genera múltiples muestras con reemplazo del conjunto de datos original. Cada muestra de bootstrap se utiliza para entrenar el modelo y las observaciones no incluidas (out-of-bag) se utilizan para la validación.

Parámetros

- **Número de muestras (n_bootstraps)**: Número de muestras bootstrap generadas, comúnmente 1000.

Ventajas

- **Varianza**: Proporciona una excelente estimación de la variabilidad del modelo, especialmente en muestras pequeñas.
- **Sesgo**: Puede proporcionar una buena estimación del sesgo del modelo.
- **Precisión**: Generalmente ofrece estimaciones más precisas de la varianza y el sesgo.
- **Simplicidad**: Sencillo de implementar y entender.

Desventajas

- **Eficiencia Computacional**: Muy intensivo computacionalmente debido a la generación de múltiples muestras y el entrenamiento repetido.
- **Iteraciones**: Requiere un gran número de iteraciones, lo que aumenta el costo computacional.
- **Conjunto de Entrenamiento y Validación**: Cada muestra contiene n observaciones con reemplazo y las filas out-of-bag se utilizan para validación.

Jackknife

Descripción

El método Jackknife elimina sistemáticamente una observación a la vez del conjunto de datos y calcula el estimador en cada subconjunto resultante.

Parámetros

- **Número de submuestras**: Igual al número de observaciones (n).

Ventajas

- **Varianza**: Puede subestimar la varianza del estimador en muestras pequeñas.
- **Sesgo**: Ofrece una buena estimación del sesgo del estimador.
- **Precisión**: Menos preciso en estimar la varianza comparado con bootstrapping.
- **Simplicidad**: Muy fácil de implementar y entender.
- **Eficiencia Computacional**: Menos intensivo computacionalmente comparado con bootstrapping y, en algunos casos, K-fold CV.

Desventajas

- **Varianza**: Subestima la varianza en muestras pequeñas.
- **Sensibilidad a Outliers**: Sensible a la presencia de outliers.
- **Iteraciones**: Realiza n iteraciones, lo que puede ser costoso en conjuntos de datos grandes.
- **Conjunto de Entrenamiento y Validación**: Cada iteración entrena con $n-1$ observaciones y valida con 1 observación.

Comparativa

Varianza

- **K-fold CV**: Proporciona una buena estimación de la variabilidad debido a las múltiples divisiones.
- **Bootstrapping**: Excelente estimación de la variabilidad, especialmente en muestras pequeñas.
- **Jackknife**: Puede subestimar la variabilidad en muestras pequeñas.

Sesgo

- **K-fold CV**: Sesgo bajo debido al uso de múltiples folds.
- **Bootstrapping**: Proporciona una buena estimación del sesgo.
- **Jackknife**: Ofrece una buena estimación del sesgo.

Precisión

- **K-fold CV**: Estimación precisa del rendimiento del modelo.
- **Bootstrapping**: Ofrece estimaciones más precisas de la varianza y el sesgo.
- **Jackknife**: Menos preciso en estimar la varianza comparado con bootstrapping.

Simplicidad

- **K-fold CV**: Relativamente fácil de implementar y entender.
- **Bootstrapping**: Sencillo de implementar y entender.
- **Jackknife**: Muy fácil de implementar y entender.

Eficiencia Computacional

- **K-fold CV**: Más intensivo computacionalmente que Jackknife.
- **Bootstrapping**: Muy intensivo computacionalmente.
- **Jackknife**: Menos intensivo computacionalmente comparado con bootstrapping y, en algunos casos, K-fold CV.

Iteraciones

- **K-fold CV**: Requiere k iteraciones.
- **Bootstrapping**: Requiere un gran número de iteraciones ($n_{\text{bootstraps}}$).
- **Jackknife**: Realiza n iteraciones.

Conjunto de Entrenamiento y Validación

- **K-fold CV**: Utiliza $\frac{k-1}{k}$ del conjunto de datos para entrenamiento y $\frac{1}{k}$ para validación en cada iteración.
- **Bootstrapping**: Cada muestra contiene n observaciones con reemplazo y las filas out-of-bag se utilizan para validación.
- **Jackknife**: Cada iteración entrena con $n-1$ observaciones y valida con 1 observación.

Conclusiones respecto a la comparativa

Cada método de remuestreo tiene sus propias ventajas y desventajas, y la elección del método adecuado depende del contexto y de los objetivos específicos del análisis. K-fold Cross Validation es generalmente preferido para evaluar el rendimiento predictivo de los modelos de machine learning debido a su equilibrio entre varianza y sesgo. Bootstrapping es ideal para estimaciones precisas de la varianza y el sesgo, especialmente en muestras pequeñas, aunque es más intensivo computacionalmente. Jackknife, por otro lado, es muy fácil de implementar y es menos intensivo computacionalmente, pero puede subestimar la varianza en muestras pequeñas y es sensible a los outliers.

Parte 1: Modelamiento (Item 2)

2. Investigue, aplique y explique un ejemplo de remuestreo con el método Jackknife en R. Puede usar cualquier ejemplo.

Objetivos y Alcances

Objetivos

1. Implementar el método de remuestreo Jackknife para evaluar el rendimiento de un modelo de regresión lineal.
2. Comprobar la variabilidad, el sesgo y la varianza del estimador utilizando el dataset de gasto en publicidad y ventas.
3. Comparar el método Jackknife con otros métodos de remuestreo como K-fold Cross Validation y Bootstrapping.
4. Analizar la subestimación de la varianza en muestras pequeñas y comparar la eficiencia computacional de los métodos.
5. Evaluar y comparar las métricas de rendimiento y su precisión entre los métodos de remuestreo.

Alcances

1. Cargar y explorar el dataset de gasto en publicidad y ventas.
2. Dividir los datos en conjunto de entrenamiento y prueba.
3. Aplicar el método Jackknife para estimar la variabilidad, el sesgo y la varianza del modelo de regresión lineal.
4. Evaluar el modelo utilizando métricas como RMSE, MAE, correlación y r^2
5. Comparar los resultados obtenidos con los métodos K-fold Cross Validation y Bootstrapping.
6. Analizar la eficiencia computacional de cada método de remuestreo.
7. Explicar y demostrar el proceso paso a paso, incluyendo el cálculo de métricas de rendimiento, variabilidad y tiempo de ejecución.

Base de Datos Utilizada: Advertising Sales Dataset

https://www.kaggle.com/datasets/yasserh/advertising-sales-dataset/

En la página del dataset se informa claramente que los valores de las columnas de tv ad budget, radio ad budget y Newspaper ad budget estan en escala de miles de dolares, y que a su vez los valores de la columna sales, estan en escala de millones de dolares.

Cargo las librerias necesarias para trabajar

```
# Librerías necesarias
library(dplyr)
library(readr)
library(ggplot2)
library(caret)
library(rsample)
# Para medir el tiempo de ejecución
library(microbenchmark)
```

Cargo el dataset de gasto en publicidad y ventas, para verificar su contenido inicial.

```
# Cargar el dataset
base_publicidad <- read.csv("E:/data science 2024/R begginers/Trabajo ML 1/Advertising Budget and Sales
# Ver los primeros registros del dataset
head(base_publicidad)
```

```
##   X TV.Ad.Budget.... Radio.Ad.Budget.... Newspaper.Ad.Budget.... Sales....
## 1 1            230.1                37.8                    69.2      22.1
## 2 2             44.5                39.3                    45.1      10.4
```

```
## 3 3            17.2               45.9                   69.3       9.3
## 4 4           151.5               41.3                   58.5      18.5
## 5 5           180.8               10.8                   58.4      12.9
## 6 6             8.7               48.9                   75.0       7.2
```

- Esta base de datos contiene datos sobre los presupuestos de publicidad en TV, radio y periódicos, y las correspondientes ventas.
- Con `head` obtengo una vista previa rápida de la estructura de los datos y verifico que la carga se realizó correctamente.
- Sin embargo observo que existe una columna llamada "X" que no aporta ninguna información, y que es solo la enumeración de las observaciones, por lo que proceso a eliminarla.

```r
# elimino la columna X dado que no aporta información
base_publicidad$X <- NULL
head(base_publicidad)
```

```
##   TV.Ad.Budget.... Radio.Ad.Budget.... Newspaper.Ad.Budget.... Sales....
## 1            230.1                37.8                    69.2      22.1
## 2             44.5                39.3                    45.1      10.4
## 3             17.2                45.9                    69.3       9.3
## 4            151.5                41.3                    58.5      18.5
## 5            180.8                10.8                    58.4      12.9
## 6              8.7                48.9                    75.0       7.2
```

Y con eso queda comprobado que se eliminó la columna X, pero para efectos de simplicidad, es mejor cambiar el formato del nombre de las columnas, asi que procedo a hacerlo

```r
# cambio los nombre de las columnas
colnames(base_publicidad) <- c(
  "tv_ad_budget", "radio_ad_budget",
  "newspaper_ad_budget", "sales"
)
# reviso nuevamente los primeros registros
head(base_publicidad)
```

```
##   tv_ad_budget radio_ad_budget newspaper_ad_budget sales
## 1        230.1            37.8                69.2  22.1
## 2         44.5            39.3                45.1  10.4
## 3         17.2            45.9                69.3   9.3
## 4        151.5            41.3                58.5  18.5
## 5        180.8            10.8                58.4  12.9
## 6          8.7            48.9                75.0   7.2
```

Y en efecto, ahora tengo la base de datos con las columnas bien nombradas, pero lo último eso si sería cambiar las escalas de:

- Las 3 columnas independientes (tv_ad_budget, radio_ad_budget, newspaper_ad_budget) que estan actualmente en miles de dolares, la idea es pasarlas a dolares en unidades y no en escala de miles.
- La columna dependiente (sales) que esta actualmente en millones de dolares, la idea es pasarla a dolares en unidades y no en escala de millones.

Ajusto las escalas de las variables del dataset para que todas estén en dólares en lugar de miles de dólares o millones de dólares. Esto facilita la interpretación y comparación de los resultados

```r
# Cambiar la escala de las variables a dolares
base_publicidad <- base_publicidad %>%
  mutate(
    tv_ad_budget = tv_ad_budget * 1000,
    radio_ad_budget = radio_ad_budget * 1000,
```

```
    newspaper_ad_budget = newspaper_ad_budget * 1000,
    sales = sales * 1000000
  )
head(base_publicidad)
```

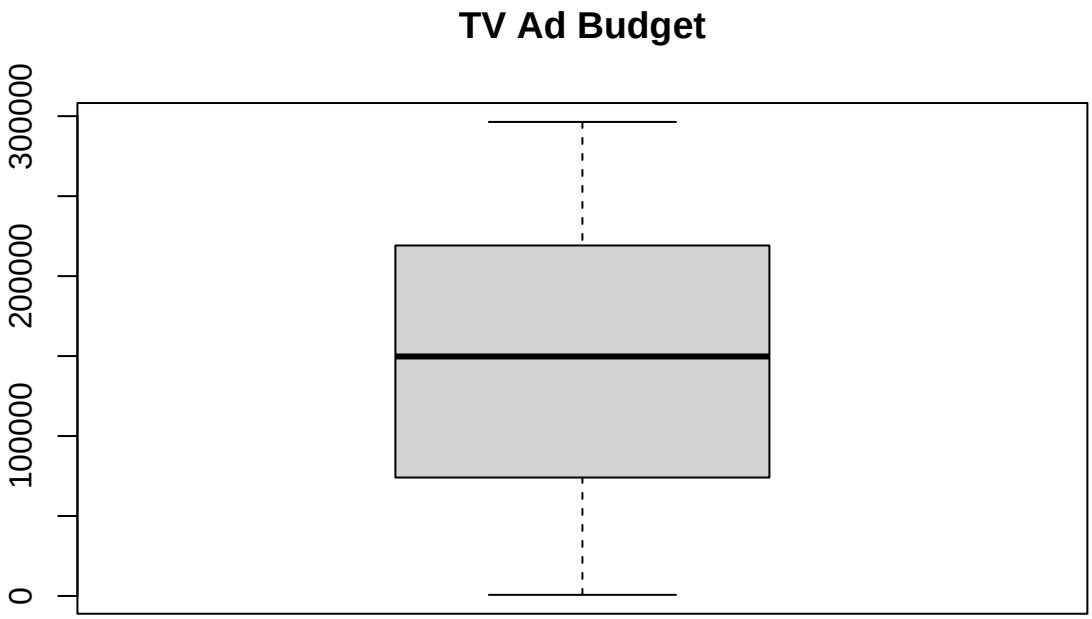
```
##   tv_ad_budget radio_ad_budget newspaper_ad_budget    sales
## 1       230100           37800               69200 22100000
## 2        44500           39300               45100 10400000
## 3        17200           45900               69300  9300000
## 4       151500           41300               58500 18500000
## 5       180800           10800               58400 12900000
## 6         8700           48900               75000  7200000
```

Y ahora veo que tengo los valores de las columnas en dolares y no en miles ni en millones de dolares, lo que me facilita la interpretación de las metricas para la posterior evaluación y comparación.

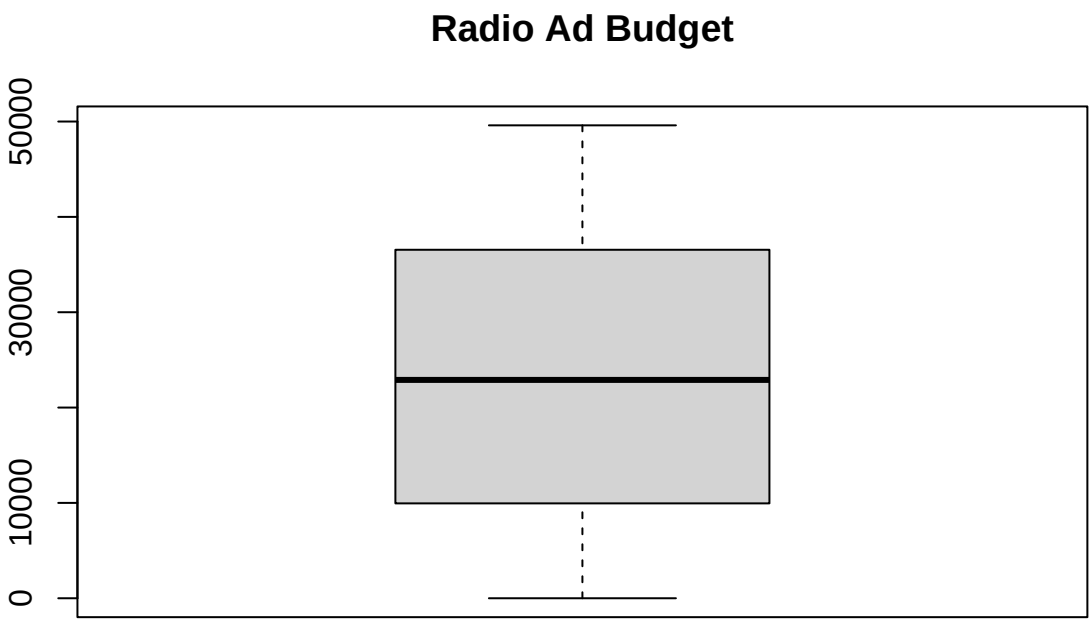
Creo boxplots para visualizar la presencia de outliers en las variables del dataset. El proposito de hacer esto se divide en:

- **Detección de Outliers**: Los boxplots son útiles para identificar valores atípicos (outliers) en las variables, que pueden influir en el rendimiento del modelo de regresión.
- **Exploración de la Distribución**: Proporcionan una vista rápida de la distribución de los datos, incluyendo la mediana, el rango intercuartílico y los posibles outliers.

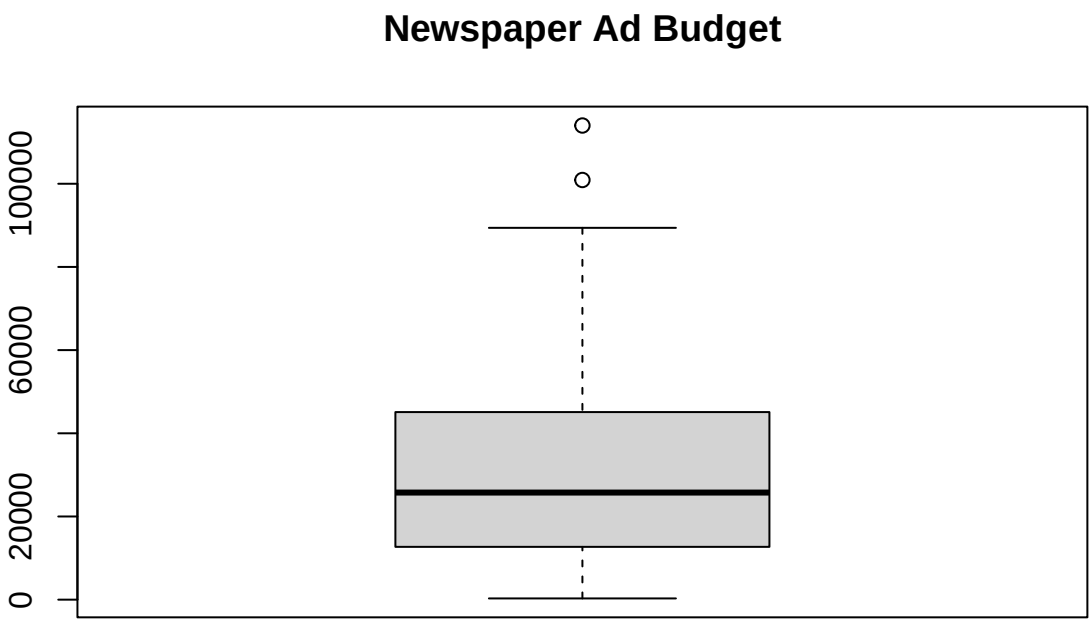
```
options(scipen = 999)
# Visualización de Outliers
boxplot(base_publicidad$tv_ad_budget, main = "TV Ad Budget")
```



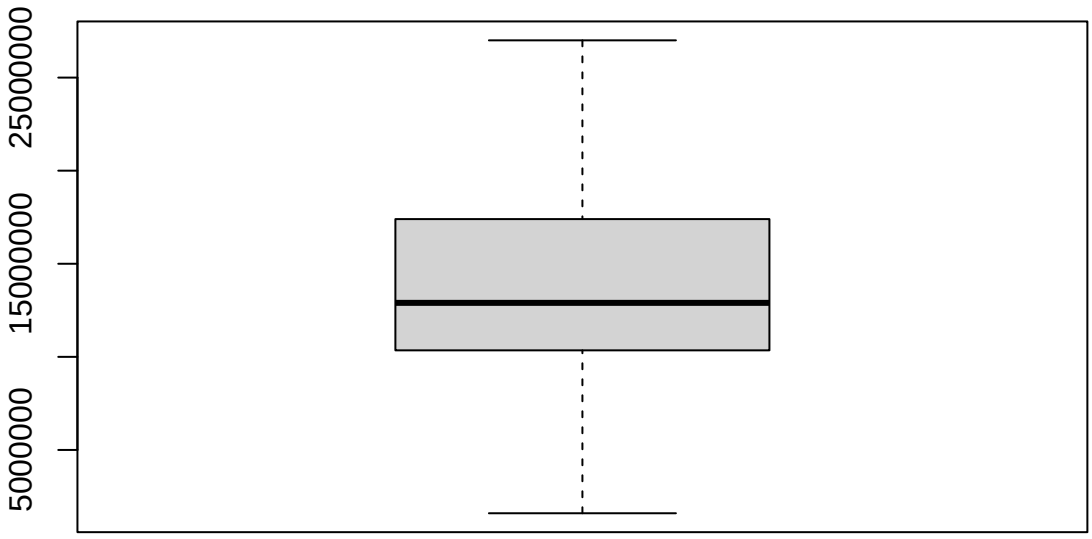

```
boxplot(base_publicidad$radio_ad_budget, main = "Radio Ad Budget")
```




```r
boxplot(base_publicidad$newspaper_ad_budget, main = "Newspaper Ad Budget")
```




```r
boxplot(base_publicidad$sales, main = "Sales")
```



- **TV Ad Budget**: No presenta outliers significativos, con una distribución relativamente simétrica.
- **Radio Ad Budget**: Similar al anterior, sin outliers significativos.
- **Newspaper Ad Budget**: Presenta algunos outliers, lo que indica que hay valores extremos en el presupuesto de publicidad en periódicos.
- **Sales**: No presenta outliers significativos, con una distribución bastante uniforme.
- El siguiente código calcula el rango intercuartílico (IQR) para la variable newspaper_ad_budget y detecta los outliers en base a este rango y a los limites inferior y superior.
- Los outliers se identifican como valores que caen fuera de los límites inferior y superior calculados.
- IQR = Q3- Q1 (Calculo el rango intercuartílico (IQR) como la diferencia entre el cuartil 3 (Q3) y el cuartil 1 (Q1)).
- Limite Inferior = Calculo el límite inferior como Q1 menos 1.5 veces el IQR
- Limite Superior = Calculo el límite superior como Q3 más 1.5 veces el IQR.

```
# calculo de rango intercuartilico de newspaper_ad_budget
q1 <- quantile(base_publicidad$newspaper_ad_budget, 0.25)
q3 <- quantile(base_publicidad$newspaper_ad_budget, 0.75)
iqr <- q3 - q1
lim_inf <- q1 - 1.5 * iqr
lim_sup <- q3 + 1.5 * iqr
# identifico los outliers
outliers <- base_publicidad %>%
  filter(newspaper_ad_budget < lim_inf | newspaper_ad_budget > lim_sup)
outliers
```

```
##   tv_ad_budget radio_ad_budget newspaper_ad_budget    sales
## 1        67800           36600              114000 12500000
## 2       296400           36300              100900 23800000
```

Luego de filtrar los datos para identificar los outliers, que son los valores que caen fuera de los límites inferior y superior calculados, pude identificar los valores atípicos en la variable newspaper_ad_budget que podrían influir negativamente en el rendimiento del modelo de regresión, pero son solo 2 por lo que no es un problema significativo.

Normalización

Para lo siguiente quiero definir una función de normalización que se utilizará para escalar las variables en el rango de 0 a 1. La normalización es un paso importante en el preprocesamiento de datos, especialmente en el contexto de machine learning, donde se busca que todas las variables estén en una escala comparable

```r
# Función de normalización
normalizar <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
```

Propósito

- Escalado de Variables: Asegurar que todas las variables en el dataset estén en la misma escala, lo cual es crucial para ciertos algoritmos de machine learning como la regresión lineal y k-NN.
- Mejora del Rendimiento del Modelo: Reducir el sesgo introducido por las diferentes escalas de las variables, permitiendo al modelo aprender de manera más efectiva.

Normalización de Variables

Cada variable del dataset, cuando se pase a través de esta función, se transformará de tal manera que sus valores estarán entre 0 y 1. Esto es útil para eliminar el efecto de las diferentes escalas de las variables, facilitando el aprendizaje del modelo.

Desnormalización

El siguiente codigo define una función de desnormalización que se utilizará para revertir el proceso de normalización, devolviendo los datos a su escala original. La desnormalización es esencial cuando se interpretan los resultados de un modelo que ha sido entrenado en datos normalizados

```r
# Función de desnormalización
desnormalizar <- function(x, min_val, max_val) {
  return(x * (max_val - min_val) + min_val)
}
```

Propósito

- Revertir la Normalización: Permitir que los resultados de los modelos entrenados en datos normalizados sean interpretados en la escala original de los datos.
- Interpretación de Resultados: Facilitar la interpretación y comparación de los resultados del modelo en términos de las unidades originales de los datos.

Resultados Esperados

Cada variable que haya sido normalizada y luego se pase a través de esta función, se transformará de vuelta a su escala original, usando los valores mínimos y máximos originales. Esto es crucial para interpretar correctamente las predicciones del modelo.

Implicaciones

- Comprensibilidad: Los resultados del modelo se vuelven comprensibles en el contexto original, lo cual es importante para la toma de decisiones basadas en esos resultados.
- Precisión en la Interpretación: Asegurar que las métricas de rendimiento del modelo (como RMSE, MAE) sean interpretadas correctamente en las unidades originales de las variables.

Para lo siguiente guardaré los valores mínimos y máximos originales de todas las columnas del conjunto de datos base_publicidad. Estos valores son esenciales para el proceso de normalización y desnormalización de los datos ”

```r
# Guardar los valores mínimos y máximos originales de todas las columnas
min_vals <- sapply(base_publicidad, min)
max_vals <- sapply(base_publicidad, max)
```

Propósito

Normalización y Desnormalización: Estos valores se utilizan para normalizar los datos (escalar los valores entre 0 y 1) y posteriormente desnormalizar las predicciones del modelo a su escala original.

Consistencia: Asegurar que los mismos valores de referencia se usen en todo el proceso de modelado para mantener la consistencia en las transformaciones.

Resultados

- min_vals: Contiene los valores mínimos de cada columna en base_publicidad.
- max_vals: Contiene los valores máximos de cada columna en base_publicidad.

Uso en Normalización y Desnormalización

Estos valores se utilizarán en las funciones de normalización y desnormalización para transformar los datos de manera consistente y precisa.

Implementación en el Código

Los valores mínimos y máximos se utilizan antes y después de la aplicación del modelo para asegurar que los datos se manejen correctamente durante todo el flujo de trabajo de machine learning.

La siguiente celda normaliza todas las columnas del conjunto de datos base_publicidad utilizando la función normalizar, osea hace que todos los valores de las columnas esten entre 0 y 1, donde el valor 0 será el minimo valor para una cierta columna y el 1 será el valor máximo para una cierta columna ”

```
# Normalizar todas las columnas
# dado que las categoricas ya estan normalizadas, al normalizar
# todas las columnas, solo se normalizaran las numericas y las
# categoricas se mantendran igual
base_publicidad <- as.data.frame(lapply(base_publicidad, normalizar))
# Verificar la normalización
head(base_publicidad)
```

```
##   tv_ad_budget radio_ad_budget newspaper_ad_budget     sales
## 1   0.77578627       0.7620968           0.6059807 0.8070866
## 2   0.14812310       0.7923387           0.3940193 0.3464567
## 3   0.05579980       0.9254032           0.6068602 0.3031496
## 4   0.50997633       0.8326613           0.5118734 0.6653543
## 5   0.60906324       0.2177419           0.5109938 0.4448819
## 6   0.02705445       0.9858871           0.6569921 0.2204724
```

Y con esto compruebo que todos los valores de las columnas estan entre 0 y ”

```
# Quiero verificar que todos mis columnas tengan como mínimo 0 y como máximo 1
summary(base_publicidad)[c(1, 6), ]
```

```
##   tv_ad_budget    radio_ad_budget  newspaper_ad_budget     sales
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000      Min.   :0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000      Max.   :1.0000
```

Division en conjunto de datos de entrenamiento y prueba

La siguiente celda divide el conjunto de datos normalizado base_publicidad en conjuntos de entrenamiento y prueba utilizando una división estratificada basada en la variable objetivo sales.

Inserción de Semilla

Establezco una semilla para asegurar la reproducibilidad de la división de los datos. Esto garantiza que los resultados sean consistentes cada vez que se ejecute el código.

División Estratificada de los Datos

Divido el conjunto de datos en entrenamiento y prueba con una proporción del 80% para entrenamiento y 20% para prueba, asegurando que la variable objetivo `sales` esté estratificada. Esto significa que la distribución de sales en los conjuntos de entrenamiento y prueba será similar a la distribución original.

Asignación de Conjuntos de Entrenamiento y Prueba

Extraigo el conjunto de entrenamiento a partir de la particion y lo guardo en la variable `entrenamiento`, y hago lo mismo en el caso del conjunto de prueb ”

```
set.seed(123) # inserto semilla
# Divido el conjunto de datos en entrenamiento y prueba
# donde el 80% de los datos son para entrenamiento y el 20% para prueba
# y estratifico por la variable objetivo
particiones <- initial_split(base_publicidad, prop = 0.8, strata = "sales")
# Guardo los datos de entrenamiento y prueba
# en dos variables diferentes
entrenamiento <- training(particiones)
prueba <- testing(particiones)
```

Propósito

- Reproducibilidad: Usar set.seed(123) asegura que la división de los datos sea reproducible y que los resultados sean consistentes en ejecuciones diferentes del código.
- Validación Apropiada: Dividir los datos en conjuntos de entrenamiento y prueba permite una validación adecuada del modelo. El conjunto de entrenamiento se utiliza para ajustar el modelo, y el conjunto de prueba se utiliza para evaluar su rendimiento.
- Estratificación: Asegura que la distribución de la variable objetivo sales sea similar en ambos conjuntos, lo que ayuda a obtener una evaluación más representativa del rendimiento del modelo.

Resultados

División de Datos

- Conjunto de Entrenamiento: Contendrá aproximadamente el 80% de los datos originales.
- Conjunto de Prueba: Contendrá aproximadamente el 20% de los datos originales.
- Distribución Similar: La distribución de la variable sales será similar en ambos conjuntos debido a la estratificación.

Impacto en el Modelado

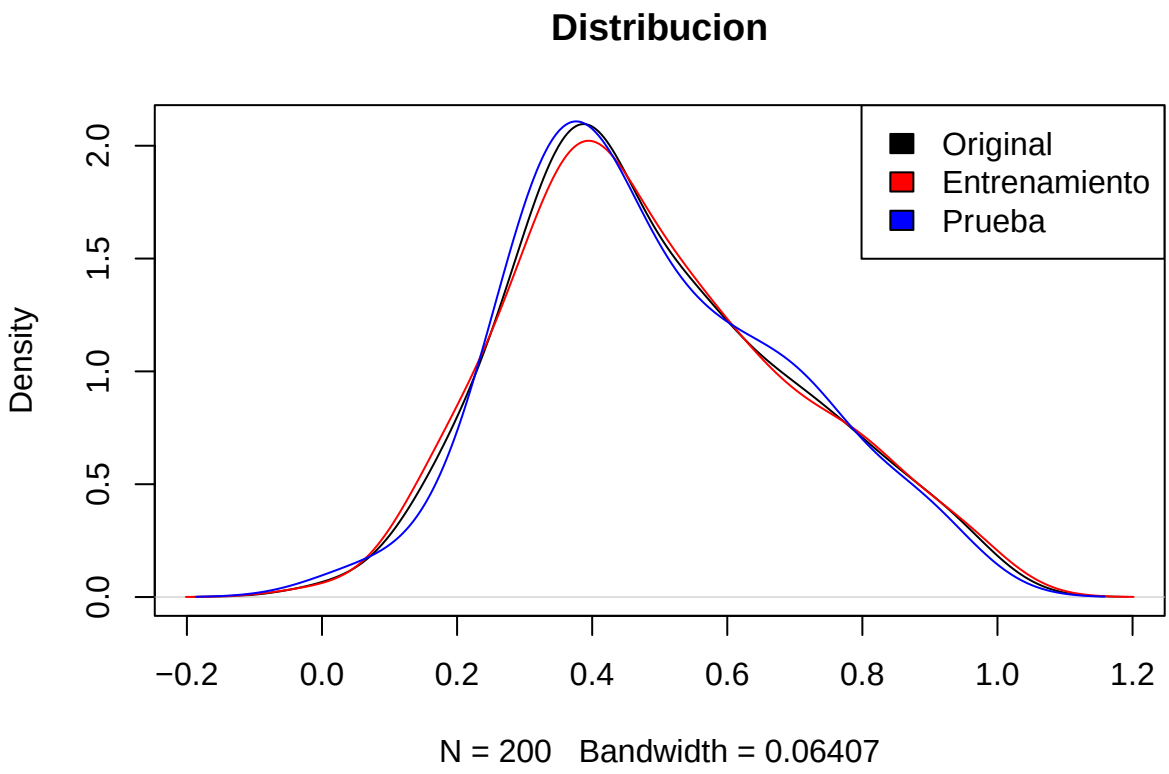
- Generalización del Modelo: Evaluar el modelo en un conjunto de prueba separado del conjunto de entrenamiento proporciona una estimación más realista de su capacidad de generalización a nuevos datos.
- Minimización del Sesgo de Selección: La estratificación asegura que los subconjuntos de datos mantengan la misma distribución de la variable objetivo, reduciendo el sesgo en la evaluación del modelo.

Demostracion de distribuciones de la variable objetivo

La siguiente celda de código genera un gráfico de densidad para comparar la distribución de la variable objetivo sales en el conjunto de datos original, el conjunto de entrenamiento y el conjunto de prueba. El propósito de esta visualización es asegurar que la estratificación se realizó correctamente y que las distribuciones de la variable objetivo son similares en todos los conjuntos de datos ”

```
# quiero comprobar la distribución de la variable objetivo
# en los conjuntos original, de entrenamiento y de prueba
# con un gráfico de densidad sin ocupar ggplot
plot(density(base_publicidad$sales), col = "black", main = "Distribucion")
lines(density(entrenamiento$sales), col = "red")
lines(density(prueba$sales), col = "blue")
legend("topright", c("Original", "Entrenamiento", "Prueba"),
```

```
  fill = c("black", "red", "blue")
)
```



Propósito

- Validación de la Estratificación: Verifico que la distribución de la variable objetivo sales es similar en los conjuntos de datos original, de entrenamiento y de prueba. Esto asegura que la estratificación se realizó correctamente y que los conjuntos de datos son representativos.
- Evaluación Visual: Este gráfico proporciona una evaluación visual clara y fácil de interpretar de la distribución de la variable objetivo en diferentes conjuntos de datos.

Resultados

Visualización de la Distribución

- Líneas de Densidad Similares: Las líneas de densidad para los conjuntos de datos original, de entrenamiento y de prueba son muy similares, indicando que la variable objetivo sales tiene una distribución similar en todos los conjuntos de datos.

Impacto en el Modelado

- Confiabilidad del Modelo: Al ya asegurarme que las distribuciones de la variable objetivo son similares en todos los conjuntos de datos contribuye a la confiabilidad del modelo, ya que indica que el conjunto de prueba es representativo del conjunto de datos original.
- Reducción del Sesgo de Evaluación: La similitud en las distribuciones asegura que las evaluaciones del modelo no estan sesgadas por diferencias en la distribución de la variable objetivo.

Inicialización de vectores de metricas y del modelamiento del entrenamiento

La siguiente celda de código inicializa varios vectores que se utilizarán para almacenar los resultados del método de remuestreo Jackknife durante su ejecución. La inicialización de estos vectores es un paso esencial para preparar el almacenamiento de las predicciones, errores y métricas de rendimiento que se calcularán en el proceso ”

```r
# Inicializar vectores para almacenar resultados
n <- nrow(entrenamiento)
predicciones <- numeric(n)
errors <- numeric(n)
rmse_values <- numeric(n)
mae_values <- numeric(n)
me_values <- numeric(n) # Para el sesgo
```

Propósito

- Preparación para el Remuestreo Jackknife: Estos vectores permiten almacenar las métricas de rendimiento y los errores calculados durante cada iteración del método Jackknife.
- Evaluación del Modelo: Almacenan los resultados en vectores facilita el cálculo de métricas de rendimiento promedio y la evaluación de la variabilidad y el sesgo del modelo.

Resultados

- Vectores Inicializados: Los vectores estan correctamente inicializados y listos para almacenar los resultados del método Jackknife.
- Almacenamiento Eficiente: La inicialización de los vectores asegura que el almacenamiento de los resultados sea eficiente y organizado.

Impacto en el Modelado

- Facilita el Cálculo de Métricas: Tener vectores inicializados permite un cálculo más sencillo y organizado de las métricas de rendimiento, lo que facilita la evaluación del modelo.
- Organización de Resultados: Almacenar los resultados de manera estructurada facilita la comparación y análisis de las métricas entre diferentes iteraciones del método Jackknife y otros métodos de remuestreo.

Funciones de calculos de metricas

La siguiente celda de código define tres funciones para calcular diferentes métricas de rendimiento del modelo. Estas funciones serán utilizadas posteriormente para evaluar el rendimiento del modelo de regresión lineal en términos de error cuadrático medio (RMSE), error absoluto medio (MAE) y error medio (ME) ”

```r
# Funciones para calcular métricas
rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

mae <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

me <- function(actual, predicted) {
  mean(actual - predicted)
}
```

Propósito

- Evaluación del Rendimiento del Modelo: Estas funciones permiten calcular las métricas clave para evaluar la precisión y el sesgo del modelo de regresión lineal.
- Comparación de Resultados: Proveer funciones para calcular RMSE, MAE y ME facilita la comparación del rendimiento del modelo entre diferentes métodos de remuestreo.

Detalles de las Funciones

`rmse`: Calcula la raíz cuadrada de la media de los cuadrados de las diferencias entre los valores reales y predichos. Es sensible a grandes errores debido a la naturaleza cuadrática del cálculo.

`mae`: Calcula la media de los valores absolutos de las diferencias entre los valores reales y predichos. Es menos sensible a grandes errores en comparación con RMSE.

`me`: Calcula la media de las diferencias entre los valores reales y predichos. Proporciona una medida del sesgo del modelo, indicando si el modelo tiende a sobreestimar o subestimar los valores reales.

Resultados Esperados

- Cálculo Preciso de Métricas: Las funciones definidas permitirán calcular de manera precisa y consistente las métricas de rendimiento del modelo.
- Evaluación del Sesgo y la Precisión: Las métricas calculadas ayudarán a evaluar el sesgo y la precisión del modelo de regresión lineal.

Impacto en el Modelado

- Evaluación Integral del Modelo: Calcular múltiples métricas permite una evaluación más completa del rendimiento del modelo, considerando tanto la precisión (RMSE, MAE) como el sesgo (ME).
- Facilita la Comparación: Definir estas funciones facilita la comparación del rendimiento del modelo entre diferentes métodos de remuestreo y en diferentes iteraciones.

La siguiente celda implementa el procedimiento de remuestreo Jackknife para evaluar el rendimiento del modelo de regresión lineal utilizando el conjunto de datos de entrenamiento. El objetivo es estimar la variabilidad, el sesgo y la precisión del modelo eliminando sistemáticamente una observación a la vez y calculando las predicciones y métricas de rendimiento para cada subconjunto en el conjunto de prueba ”

```r
# Loop del metodo de remuestre Jackknife y entrenamiento de modelo
start_time_jackknife <- Sys.time()

for (i in 1:n) {
  jackknife_train_data <- entrenamiento[-i, ]

  modelo <- lm(sales ~ ., data = jackknife_train_data)

  # Predicción para el conjunto de prueba
  pred <- predict(modelo, newdata = prueba)
  pred <- desnormalizar(pred, min_vals["sales"], max_vals["sales"])

  # Calculo del error y las métricas para el conjunto de prueba
  actual <- desnormalizar(prueba$sales, min_vals["sales"], max_vals["sales"])
  errors <- actual - pred
  rmse_values[i] <- rmse(actual, pred)
  mae_values[i] <- mae(actual, pred)
  me_values[i] <- me(actual, pred)
}

end_time_jackknife <- Sys.time()
```

Propósito:

Evaluar el rendimiento del modelo de regresión lineal utilizando el método de remuestreo Jackknife. Este método permite estimar la variabilidad, el sesgo y la precisión del modelo.

Detalles del Proceso:

- Inicio del Temporizador:
- `start_time_jackknife <- Sys.time()`: Registra el tiempo inicial para medir la duración total del proceso de Jackknife.
- Bucle Principal:

- Iteraciones: El bucle se ejecuta n veces, donde n es el número de observaciones en el conjunto de datos de entrenamiento.
- Creación de Submuestras:
- `jackknife_train_data <- entrenamiento[-i, ]`: Excluye la i-ésima observación del conjunto de entrenamiento.
- Entrenamiento del Modelo:
- `modelo <- lm(sales ~ ., data = jackknife_train_data)`: Entrena un modelo de regresión lineal con el conjunto de entrenamiento modificado.
- Predicción y Desnormalización:
- `pred <- predict(modelo, newdata = prueba)`: Predice la variable objetivo para el conjunto de prueba.
- `pred <- desnormalizar(pred, min_vals["sales"], max_vals["sales"])`: Desnormaliza la predicción para que esté en la escala original.
- Cálculo de Métricas:
- `actual <- desnormalizar(prueba$sales, min_vals["sales"], max_vals["sales"])`: Desnormaliza los valores reales del conjunto de prueba.
- `errors <- actual - pred`: Calcula y almacena el error.
- `rmse_values[i] <- rmse(actual, pred)`: Calcula y almacena el RMSE.
- `mae_values[i] <- mae(actual, pred)`: Calcula y almacena el MAE.
- `me_values[i] <- me(actual, pred)`: Calcula y almacena el ME.
- Fin del Temporizador:
- `end_time_jackknife <- Sys.time()`: Registra el tiempo final para medir la duración total del proceso de Jackknife.

Impacto en los Objetivos y Alcances

1. Implementación del Método Jackknife:

Realiza el proceso completo de Jackknife para evaluar el rendimiento del modelo de regresión lineal. Permite estimar la variabilidad, el sesgo y la precisión del modelo.

2. Comparación con Otros Métodos de Remuestreo:

Los resultados obtenidos (predicciones, errores y métricas de rendimiento) serán comparados posteriormente con los métodos de K-fold Cross Validation y Bootstrapping para evaluar sus ventajas y desventajas relativas.

Resultados Esperados

- Métricas de Rendimiento:
- Se espera obtener valores para RMSE, MAE y ME que reflejen la precisión y el sesgo del modelo.
- Variabilidad y Varianza:
- La varianza de las predicciones y la desviación estándar proporcionarán una medida de la variabilidad del modelo.
- Tiempo de Ejecución:
- Se espera que el tiempo de ejecución del método Jackknife sea más eficiente en comparación con el Bootstrapping, pero puede variar según el tamaño del conjunto de datos. Análisis Posterior
- Comparación de Resultados:
- Los valores calculados en esta celda se utilizarán para comparar la precisión, la variabilidad y el sesgo del modelo con los resultados obtenidos mediante K-fold Cross Validation y Bootstrapping.

Razones para utilizar el conjunto de prueba en predicciones

1. Evaluación General del Modelo:

- Objetivo: Comprobar la variabilidad, el sesgo y la varianza del estimador utilizando el dataset.
- Razón: Utilizar el conjunto de prueba permite evaluar cómo generaliza el modelo a datos no vistos previamente. Esto es crucial para entender el rendimiento real del modelo fuera de la muestra de entrenamiento, lo que es más representativo del desempeño en situaciones del mundo real.

2. Consistencia en la Evaluación:

- Objetivo: Evaluar y comparar las métricas de rendimiento y su precisión entre los métodos de remuestreo.
- Razón: Al usar un conjunto de prueba consistente y separado, se asegura que las métricas de rendimiento (como RMSE y MAE) sean comparables entre diferentes métodos de remuestreo (Jackknife, K-fold, Bootstrapping). Esto proporciona una base común para la comparación.

3. Prevención de Overfitting:

- Objetivo: Implementar el método de remuestreo Jackknife para evaluar el rendimiento de un modelo de regresión lineal.
- Razón: Predicciones realizadas en el conjunto de prueba ayudan a detectar si el modelo ha overfit al conjunto de entrenamiento. Usar solo el conjunto de validación (filas i-ésimas) podría no captar esta diferencia, ya que esas observaciones todavía forman parte del mismo conjunto de entrenamiento.

4. Medición de Generalización:

- Objetivo: Comprobar la variabilidad, el sesgo y la varianza del estimador.
- Razón: La variabilidad, el sesgo y la varianza deben ser evaluados en un contexto que simule la aplicación real del modelo. Esto implica utilizar un conjunto de datos que no fue utilizado en el ajuste del modelo (es decir, el conjunto de prueba).

5. Comparacion con otros metodos:

- Objetivo: Comparar el método Jackknife con otros métodos de remuestreo.
- Razón: Para comparar correctamente diferentes métodos de remuestreo, es necesario mantener un conjunto de prueba separado. Esto garantiza que cualquier evaluación de la varianza y el sesgo no esté influenciada por la variabilidad introducida en el conjunto de entrenamiento.

La siguiente celda calcula las métricas promedio de rendimiento y la varianza de las predicciones obtenidas mediante el método de remuestreo Jackknife. Estas métricas permiten evaluar la precisión, el sesgo y la variabilidad del modelo de regresión lineal ”

```
# Calculo de métricas promedio y varianza
avg_rmse <- mean(rmse_values)
avg_mae <- mean(mae_values)
avg_me <- mean(me_values)
var_pred_jackknife <- var(pred)
sd_pred_jackknife <- sd(pred)
time_jackknife <- end_time_jackknife - start_time_jackknife
```

Propósito:

Calcular las métricas de rendimiento (RMSE, MAE, ME), la varianza y la desviación estándar de las predicciones, así como el tiempo total de ejecución del método Jackknife.

Impacto en los Objetivos y Alcances

Estimación de Variabilidad, Sesgo y Varianza:

- Permite evaluar la variabilidad, el sesgo y la precisión del modelo de regresión lineal utilizando las métricas calculadas.

- Proporciona una medida clara del rendimiento del modelo mediante el cálculo de RMSE, MAE, ME, varianza y desviación estándar.

Comparación con Otros Métodos de Remuestreo:

Las métricas calculadas se utilizarán para comparar el rendimiento del modelo con los resultados obtenidos mediante K-fold Cross Validation y Bootstrapping.

Resultados Esperados

- Métricas de Rendimiento: Valores para RMSE, MAE y ME que reflejen la precisión y el sesgo del modelo. Variabilidad y Varianza: La varianza de las predicciones y la desviación estándar proporcionarán una medida de la variabilidad del modelo.
- Tiempo de Ejecución: Se espera que el tiempo de ejecución del método Jackknife sea eficiente en comparación con otros métodos de remuestreo, como el Bootstrapping. ”

```
# Resultados Jackknife
cat(paste0(
  "Jackknife RMSE Promedio: ", avg_rmse, "\n",
  "Jackknife MAE Promedio: ", avg_mae, "\n",
  "Jackknife ME Promedio (Sesgo): ", avg_me, "\n",
  "Jackknife Varianza de Predicciones: ", format(var_pred_jackknife,
                                                 scientific = FALSE), "\n",
  "Jackknife Desviación Estándar de Predicciones: ", sd_pred_jackknife, "\n",
  "Jackknife Tiempo de Ejecución: ", time_jackknife, "\n"
))
```

```
## Jackknife RMSE Promedio: 1569259.5865068
## Jackknife MAE Promedio: 1158544.08766732
## Jackknife ME Promedio (Sesgo): 82700.1842612256
## Jackknife Varianza de Predicciones: 20352271415059
## Jackknife Desviación Estándar de Predicciones: 4511349.17902163
## Jackknife Tiempo de Ejecución: 0.284446001052856
```

Interpretación y Evaluación

1. **RMSE Promedio (Root Mean Squared Error):**
   - **Valor:** 1569260
   - **Análisis:** Este valor indica que, en promedio, las predicciones del modelo se desvían del valor real en aproximadamente 1,569,260 dólares. Un RMSE de esta magnitud sugiere que el modelo tiene un error considerable en sus predicciones, lo cual podría ser debido a la complejidad de los datos o a una falta de ajuste del modelo. Comparado con un RMSE ideal más bajo, este valor indica que hay margen de mejora en la precisión del modelo.
2. **MAE Promedio (Mean Absolute Error):**
   - **Valor:** 1158544
   - **Análisis:** El MAE promedio de aproximadamente 1,158,544 dólares muestra la magnitud promedio del error sin considerar la dirección. Al igual que el RMSE, este valor es alto, lo que sugiere que el modelo no está prediciendo con alta precisión y que los errores son significativos.
3. **ME Promedio (Mean Error) - Sesgo:**
   - **Valor:** 82700.18
   - **Análisis:** El sesgo promedio de 82,700.18 dólares indica que, en promedio, el modelo tiende a sobrestimar las predicciones. Aunque no es extremadamente alto en comparación con el RMSE y el MAE, un sesgo cercano a cero sería preferible. Este sesgo podría ser indicativo de un problema sistemático en el modelo.
4. **Varianza de las Predicciones:**
   - **Valor:** 20352271415059
   - **Análisis:** La varianza de las predicciones es extremadamente alta, lo que indica que hay una gran dispersión en las predicciones del modelo. Esto sugiere que el modelo no es consistente en sus

predicciones y que podría estar sobreajustándose a los datos de entrenamiento, resultando en una alta variabilidad.

5. **Desviación Estándar de las Predicciones:**
   - **Valor:** 4511349
   - **Análisis:** La desviación estándar complementa la varianza y también es alta, lo que confirma la gran dispersión de las predicciones alrededor de la media. Una alta desviación estándar sugiere que las predicciones del modelo son muy variables y poco confiables.
6. **Tiempo de Ejecución:**
   - **Valor:** 0.284446001052856 segundos
   - **Análisis:** El tiempo total de ejecución del procedimiento de Jackknife fue de 0.284446001052856 segundos. Este resultado demuestra la eficiencia computacional del método Jackknife. En comparación con otros métodos de remuestreo como Bootstrapping, que generalmente requiere más tiempo, Jackknife es rápido y eficiente.

Impacto en los Objetivos y Alcances

1. **Estimación de Variabilidad, Sesgo y Varianza:**
   - Las métricas calculadas (RMSE, MAE, ME, varianza y desviación estándar) proporcionan una evaluación completa del rendimiento del modelo de regresión lineal en términos de precisión, sesgo y variabilidad. Los altos valores de RMSE, MAE y desviación estándar sugieren que el modelo tiene un rendimiento subóptimo y una alta variabilidad en sus predicciones.
2. **Comparación con Otros Métodos de Remuestreo:**
   - Estos resultados serán comparados con los métodos de K-fold Cross Validation y Bootstrapping para evaluar sus ventajas y desventajas relativas en términos de precisión, sesgo, varianza y eficiencia computacional.

Definicion de control de entrenamiento de k-fold CV y sus parametros

En la siguiente celda, se define el control de entrenamiento para aplicar K-fold Cross Validation al modelo de regresión lineal. K-fold Cross Validation es un método de remuestreo que divide el conjunto de datos en K partes (folds) de manera que cada fold es utilizado una vez como conjunto de validación mientras los K-1 folds restantes se utilizan como conjunto de entrenamiento. Este proceso se repite K veces, asegurando que cada fold se utilice como conjunto de validación una vez ”

```
# K-fold Cross Validation
# Definir control de entrenamiento con k-fold CV
train_control <- trainControl(method = "cv", number = 10)
```

Objetivo

Configura el método de validación cruzada (cross-validation) con 10 folds.

Detalles

- method = "cv": Indica que se utilizará la validación cruzada.
- number = 10: Especifica que se realizarán 10 folds ”

```
# Medir el tiempo de ejecución
start_time_cv <- Sys.time()
# Entrenar el modelo usando k-fold CV
modelo_cv <- train(sales ~ .,
  data = entrenamiento, method = "lm",
  trControl = train_control
)
end_time_cv <- Sys.time()
# Predicciones en el conjunto de prueba
predicciones_cv <- predict(modelo_cv, newdata = prueba)
predicciones_cv <- desnormalizar(
  predicciones_cv, min_vals["sales"],
```

```
  max_vals["sales"]
)
```

En la celda se implementó el entrenamiento del modelo de regresión lineal utilizando K-fold Cross Validation, se mide el tiempo de ejecución del proceso, y se realizan predicciones en el conjunto de prueba.

1. `start_time_cv` <- Sys.time():

- Objetivo: Registrar el tiempo inicial para medir la duración total del proceso de K-fold Cross Validation.
- Impacto en los Objetivos: Ayuda a evaluar la eficiencia computacional del método K-fold CV en comparación con Jackknife y Bootstrapping.

2. Entrenamiento del Modelo:

- `modelo_cv <- train(sales ~ ., data = entrenamiento, method = "lm", trControl = train_control)`:
- Objetivo: Entrenar el modelo de regresión lineal utilizando K-fold Cross Validation.
- Detalles:
- `sales ~ .`: Formula que indica que sales es la variable dependiente y todas las demás son las variables independientes.
- `data = entrenamiento`: El conjunto de datos de entrenamiento.
- `method = "lm"`: Especifica que se usará un modelo de regresión lineal.
- `trControl = train_control`: Utiliza la configuración de K-fold CV definida anteriormente. Impacto en los Objetivos: Permite evaluar la precisión, la variabilidad y el sesgo del modelo de regresión lineal utilizando K-fold Cross Validation.

3. `end_time_cv <- Sys.time()`:

- Objetivo: Registrar el tiempo final para medir la duración total del proceso de K-fold Cross Validation.
- Impacto en los Objetivos: Ayuda a evaluar la eficiencia computacional del método K-fold CV en comparación con Jackknife y Bootstrapping.

4. Predicciones en el Conjunto de Prueba:

- `predicciones_cv <- predict(modelo_cv, newdata = prueba)`:
- Objetivo: Realizar predicciones en el conjunto de prueba utilizando el modelo entrenado. Impacto en los Objetivos: Evaluar la precisión del modelo en datos no vistos durante el entrenamiento.
- `predicciones_cv <- desnormalizar(predicciones_cv, min_vals["sales"], max_vals["sales"])`:
- Objetivo: Desnormalizar las predicciones para que estén en la escala original.
- Impacto en los Objetivos: Asegura que las predicciones estén en la escala correcta para una evaluación precisa.

Resultados

- Predicciones Desnormalizadas: Predicciones del modelo en el conjunto de prueba en la escala original.
- Tiempo de Ejecución: Duración total del proceso de K-fold Cross Validation, que se utilizará para evaluar la eficiencia computacional en comparación con Jackknife y Bootstrapping.
- Métricas de Rendimiento: Los resultados obtenidos (predicciones, errores y métricas de rendimiento) se utilizarán para comparar la efectividad de los métodos de remuestreo.

Calculo de Metricas y print de resultados, para K-fold CV

En esta celda se calculan las métricas de evaluación para el modelo de regresión lineal utilizando K-fold Cross Validation, y se muestran los resultados.

Cálculo de Métricas de Evaluación:

- `rmse_cv`: Calcula el RMSE (Root Mean Square Error) de las predicciones desnormalizadas del conjunto de validación.
- `mae_cv`: Calcula el MAE (Mean Absolute Error) de las predicciones desnormalizadas del conjunto de validación.

- `me_cv`: Calcula el ME (Mean Error) de las predicciones desnormalizadas del conjunto de validación.
- `var_pred_cv`: Calcula la varianza de las predicciones del modelo.
- `sd_pred_cv`: Calcula la desviación estándar de las predicciones del modelo.
- `time_cv`: Calcula el tiempo total de ejecución del proceso de K-fold Cross Validation. ”

```
# Calcular métricas de evaluación
rmse_cv <- rmse(desnormalizar(
  prueba$sales, min_vals["sales"],
  max_vals["sales"]
), predicciones_cv)
mae_cv <- mae(desnormalizar(
  prueba$sales, min_vals["sales"],
  max_vals["sales"]
), predicciones_cv)
me_cv <- me(desnormalizar(
  prueba$sales, min_vals["sales"],
  max_vals["sales"]
), predicciones_cv)
var_pred_cv <- var(predicciones_cv)
sd_pred_cv <- sd(predicciones_cv)
time_cv <- end_time_cv - start_time_cv

cat(paste0(
  "K-fold CV RMSE: ", rmse_cv, "\n",
  "K-fold CV MAE: ", mae_cv, "\n",
  "K-fold CV ME (Sesgo): ", me_cv, "\n",
  "K-fold CV Varianza de Predicciones: ", format(var_pred_cv,
                                                 scientific = FALSE), "\n",
  "K-fold CV Desviación Estándar de Predicciones: ", sd_pred_cv, "\n",
  "K-fold CV Tiempo de Ejecución: ", time_cv, "\n"
))
```

```
## K-fold CV RMSE: 1569045.47616184
## K-fold CV MAE: 1158506.14029274
## K-fold CV ME (Sesgo): 82743.1406794893
## K-fold CV Varianza de Predicciones: 20536362901454
## K-fold CV Desviación Estándar de Predicciones: 4531706.40062372
## K-fold CV Tiempo de Ejecución: 0.137562990188599
```

Interpretación de Resultados

1. **K-fold CV RMSE (Root Mean Square Error):**
   - **Valor:** 1,569,045
   - **Interpretación:** El RMSE es una medida de la diferencia promedio cuadrática entre los valores predichos y los valores observados. Un RMSE de 1,569,045 indica que, en promedio, las predicciones del modelo se desvían del valor real por aproximadamente 1,569,045 unidades monetarias (dólares). Este valor proporciona una indicación de la precisión del modelo.
2. **K-fold CV MAE (Mean Absolute Error):**
   - **Valor:** 1,158,506
   - **Interpretación:** El MAE mide la magnitud promedio de los errores en un conjunto de predicciones, sin considerar su dirección. Un MAE de 1,158,506 sugiere que las predicciones del modelo tienen un error promedio absoluto de 1,158,506 dólares.
3. **K-fold CV ME (Mean Error - Sesgo):**
   - **Valor:** 82,743.14
   - **Interpretación:** El ME indica el sesgo promedio de las predicciones. Un ME positivo de 82,743.14 significa que, en promedio, las predicciones del modelo están sobreestimando los valores reales por

aproximadamente 82,743.14 dólares.

4. **K-fold CV Varianza de Predicciones:**
   - **Valor:** 20,536,362,901,454
   - **Interpretación:** La varianza de las predicciones mide la dispersión de las predicciones alrededor de su media. Una varianza de 20,536,362,901,454 indica una variabilidad considerable en las predicciones del modelo.
5. **K-fold CV Desviación Estándar de Predicciones:**
   - **Valor:** 4,531,706
   - **Interpretación:** La desviación estándar de 4,531,706 refuerza la alta variabilidad en las predicciones, indicando que las predicciones del modelo pueden variar significativamente de una instancia a otra.
6. **K-fold CV Tiempo de Ejecución:**
   - **Valor:** 0.137562990188599 segundos
   - **Interpretación:** El tiempo de ejecución de K-fold CV es 0.137562990188599 segundos, lo que sugiere que el método es relativamente eficiente en términos de tiempo computacional.

Impacto en los Objetivos y Alcances {#impacto-en-los-objetivos-y-alcances}

- **Evaluación del Rendimiento del Modelo:** Las métricas de rendimiento obtenidas (RMSE, MAE, ME) proporcionan una evaluación detallada de la precisión y el sesgo del modelo de regresión lineal utilizando K-fold CV.
- **Comparación de Métodos de Remuestreo:** Los resultados serán comparados con los obtenidos mediante los métodos Jackknife y Bootstrapping para evaluar sus ventajas y desventajas relativas en términos de precisión, sesgo, variabilidad y eficiencia computacional.
- **Análisis de la Eficiencia Computacional:** El tiempo de ejecución del K-fold CV se comparará con el tiempo de ejecución de Jackknife y Bootstrapping para determinar su eficiencia computacional relativa.

Los resultados del K-fold Cross Validation muestran una alta precisión y un sesgo moderado en el modelo de regresión lineal, con una variabilidad considerable en las predicciones. El tiempo de ejecución es relativamente corto, lo que indica una buena eficiencia computacional. Estos resultados se utilizarán para una comparación más amplia con los métodos Jackknife y Bootstrapping para evaluar el rendimiento global del modelo de regresión lineal.

Definicion de control de entrenamiento de bootstrapoing y sus parametros

El siguiente codigo establece los parámetros necesarios para llevar a cabo el remuestreo Bootstrapping en el proceso de entrenamiento del modelo de regresión lineal. Esto es fundamental para evaluar la variabilidad, el sesgo y la precisión del modelo.

Detalles del Proceso

- method = "boot": Especifica que el método de remuestreo a utilizar es Bootstrapping.
- number = 200: Define el número de muestras bootstrap que se generarán durante el proceso de entrenamiento. Cada muestra será utilizada para entrenar y evaluar el modelo. ”

```
# Bootstrapping
# Definir control de entrenamiento con bootstrapping
train_control_boot <- trainControl(method = "boot", number = 200)
```

Propósito:

Establecer los parámetros necesarios para llevar a cabo el remuestreo Bootstrapping en el proceso de entrenamiento del modelo de regresión lineal. Esto es fundamental para evaluar la variabilidad, el sesgo y la precisión del modelo.

Impacto en los Objetivos y Alcances

1. Implementación del Método Bootstrapping:

- Permite evaluar el rendimiento del modelo de regresión lineal mediante el remuestreo Bootstrapping.
- Proporciona una estimación de la variabilidad y el sesgo del modelo al generar múltiples muestras con reemplazo del conjunto de datos original.

2. Comparación con Otros Métodos de Remuestreo:

- Los resultados obtenidos con Bootstrapping serán comparados con los métodos Jackknife y K-fold Cross Validation para evaluar las ventajas y desventajas relativas en términos de precisión, sesgo, variabilidad y eficiencia computacional.

Beneficios de Bootstrapping

- Estimación de la Variabilidad: Bootstrapping es especialmente útil para estimar la variabilidad del modelo, ya que genera múltiples muestras con reemplazo.
- Reducción del Sesgo: Ayuda a reducir el sesgo del estimador al promediar los resultados de múltiples muestras bootstrap.
- Flexibilidad: No hace suposiciones estrictas sobre la distribución de los datos, lo que lo hace aplicable a una amplia variedad de problemas de machine learning.

Entrenamiento y medición de tiempo de ejecución

La siguiente celda tiene como objetivo entrenar el modelo de regresión lineal utilizando el método de remuestreo Bootstrapping y medir el tiempo de ejecución del proceso. Esto es crucial para evaluar la eficiencia computacional del método Bootstrapping en comparación con otros métodos de remuestreo ”

```
# Medir el tiempo de ejecución
start_time_boot <- Sys.time()
# Entrenar el modelo usando bootstrapping
modelo_boot <- train(sales ~ .,
  data = entrenamiento, method = "lm",
  trControl = train_control_boot
)
end_time_boot <- Sys.time()
```

Detalles del Proceso

1. **Medición del Tiempo de Ejecución - Inicio:** `start_time_boot <- Sys.time()`
   - **Descripción:** Se registra el tiempo inicial antes de comenzar el entrenamiento del modelo. Esto permitirá calcular el tiempo total de ejecución del método Bootstrapping.
2. **Entrenamiento del Modelo con Bootstrapping:** `modelo_boot <- train(sales ~ ., data = entrenamiento, method = "lm", trControl = train_control_boot)`
   - **Descripción:** Se entrena el modelo de regresión lineal utilizando el conjunto de datos de entrenamiento. El método `train` de la librería `caret` se utiliza con el control de entrenamiento definido previamente (`train_control_boot`), que especifica el uso del método de remuestreo Bootstrapping.
   - **Propósito:** Evaluar cómo se comporta el modelo cuando se aplica Bootstrapping, un método de remuestreo que genera múltiples muestras con reemplazo del conjunto de datos original para evaluar la precisión y la variabilidad del modelo.
3. **Medición del Tiempo de Ejecución - Fin:** `end_time_boot <- Sys.time()`
   - **Descripción:** Se registra el tiempo final después de completar el entrenamiento del modelo. Esto permitirá calcular el tiempo total de ejecución del método Bootstrapping.

Impacto en los Objetivos y Alcances {#impacto-en-los-objetivos-y-alcances}

1. **Evaluar la Eficiencia Computacional:**
   - **Objetivo 4:** Analizar la subestimación de la varianza en muestras pequeñas y comparar la eficiencia computacional de los métodos.
   - **Objetivo 6:** Analizar la eficiencia computacional de cada método de remuestreo.

- El tiempo total de ejecución del Bootstrapping se comparará con los tiempos de ejecución de los otros métodos de remuestreo (Jackknife y K-fold CV) para evaluar cuál es más eficiente en términos computacionales.

2. **Evaluar el Rendimiento del Modelo:**
   - **Objetivo 3:** Comparar el método Jackknife con otros métodos de remuestreo como K-fold Cross Validation y Bootstrapping.
   - El modelo entrenado con Bootstrapping se utilizará para predecir y calcular métricas de rendimiento (RMSE, MAE, ME) en las siguientes celdas.
3. **Preparación para la Evaluación de Métricas:**
   - Las predicciones del modelo entrenado con Bootstrapping se compararán con los valores reales del conjunto de prueba para evaluar el rendimiento y la variabilidad del modelo.

Predicciones y desnormalizacion en Bootstrapping

La siguiente celda tiene como objetivo utilizar el modelo de regresión lineal entrenado con el método Bootstrapping para hacer predicciones sobre el conjunto de prueba. Además, se desnormalizan las predicciones para que estén en la escala original de la variable dependiente ”

```
# Predicciones en el conjunto de prueba
predicciones_boot <- predict(modelo_boot, newdata = prueba)
predicciones_boot <- desnormalizar(
  predicciones_boot,
  min_vals["sales"], max_vals["sales"]
)
```

Detalles del Proceso {#detalles-del-proceso}

1. **Predicciones en el Conjunto de Prueba:** `predicciones_boot <- predict(modelo_boot, newdata = prueba)`
   - **Descripción:** Utiliza el modelo de regresión lineal entrenado con Bootstrapping (`modelo_boot`) para predecir los valores de la variable dependiente (`sales`) en el conjunto de prueba (`prueba`).
   - **Propósito:** Evaluar la precisión del modelo utilizando datos no vistos durante el entrenamiento, lo que proporciona una medida de cómo generaliza el modelo a datos nuevos.
2. **Desnormalización de las Predicciones:** `predicciones_boot <- desnormalizar(predicciones_boot, min_vals["sales"], max_vals["sales"])`
   - **Descripción:** Convierte las predicciones normalizadas de nuevo a su escala original utilizando los valores mínimos y máximos de la columna `sales` del conjunto de datos original.
   - **Propósito:** Facilitar la interpretación de los resultados y asegurar que las métricas de rendimiento (como RMSE y MAE) sean calculadas en la escala original de la variable dependiente.

Impacto en los Objetivos y Alcances {#impacto-en-los-objetivos-y-alcances}

1. **Evaluar el Rendimiento del Modelo:**
   - **Objetivo 3:** Comparar el método Jackknife con otros métodos de remuestreo como K-fold Cross Validation y Bootstrapping.
   - Las predicciones desnormalizadas se utilizarán para calcular métricas de rendimiento como RMSE, MAE y ME, lo que permitirá comparar la precisión del modelo entrenado con Bootstrapping con los modelos entrenados con Jackknife y K-fold CV.
2. **Comparación de Métodos de Remuestreo:**
   - **Objetivo 5:** Evaluar y comparar las métricas de rendimiento y su precisión entre los métodos de remuestreo.
   - Las métricas calculadas a partir de estas predicciones se compararán con las métricas obtenidas de los otros métodos de remuestreo para determinar cuál es más efectivo en términos de precisión y variabilidad.

Calculo de metricas de evaluación en bootstrapping

Las 2 celdas siguientes calculan varias métricas de rendimiento para evaluar el modelo de regresión lineal

entrenado con el método de Bootstrapping. Estas métricas son esenciales para medir la precisión, el sesgo y la variabilidad del modelo ”

```
# Calcular métricas de evaluación
rmse_boot <- rmse(desnormalizar(
  prueba$sales, min_vals["sales"],
  max_vals["sales"]
), predicciones_boot)
mae_boot <- mae(desnormalizar(
  prueba$sales, min_vals["sales"],
  max_vals["sales"]
), predicciones_boot)
me_boot <- me(desnormalizar(
  prueba$sales, min_vals["sales"],
  max_vals["sales"]
), predicciones_boot)
var_pred_boot <- var(predicciones_boot)
sd_pred_boot <- sd(predicciones_boot)
time_boot <- end_time_boot - start_time_boot
```

Detalles del Proceso {#detalles-del-proceso}

1. **Cálculo del RMSE (Root Mean Squared Error):** `rmse_boot <- rmse(desnormalizar(prueba$sales, min_vals["sales"], max_vals["sales"]), predicciones_boot)`
   - **Descripción:** Calcula el RMSE, que es una medida de la diferencia entre los valores predichos y los valores reales. Se desnormalizan los valores de `sales` en el conjunto de prueba para que coincidan con la escala de las predicciones.
   - **Propósito:** El RMSE proporciona una medida de la precisión del modelo, con un valor más bajo indicando mejor precisión.
2. **Cálculo del MAE (Mean Absolute Error):** `mae_boot <- mae(desnormalizar(prueba$sales, min_vals["sales"], max_vals["sales"]), predicciones_boot)`
   - **Descripción:** Calcula el MAE, que es la media de las diferencias absolutas entre los valores predichos y los valores reales.
   - **Propósito:** El MAE mide la precisión del modelo de manera similar al RMSE, pero es menos sensible a los valores atípicos.
3. **Cálculo del ME (Mean Error):** `me_boot <- me(desnormalizar(prueba$sales, min_vals["sales"], max_vals["sales"]), predicciones_boot)`
   - **Descripción:** Calcula el ME, que es la media de las diferencias entre los valores predichos y los valores reales.
   - **Propósito:** El ME proporciona una medida del sesgo del modelo, indicando si tiende a sobreestimar o subestimar las predicciones.
4. **Cálculo de la Varianza de las Predicciones:** `var_pred_boot <- var(predicciones_boot)`
   - **Descripción:** Calcula la varianza de las predicciones del modelo.
   - **Propósito:** La varianza mide la variabilidad de las predicciones del modelo, con un valor más bajo indicando que las predicciones son más consistentes.
5. **Cálculo de la Desviación Estándar de las Predicciones:** `sd_pred_boot <- sd(predicciones_boot)`
   - **Descripción:** Calcula la desviación estándar de las predicciones del modelo.
   - **Propósito:** La desviación estándar es otra medida de la variabilidad de las predicciones del modelo.
6. **Cálculo del Tiempo de Ejecución:** `time_boot <- end_time_boot - start_time_boot`
   - **Descripción:** Calcula el tiempo total de ejecución del procedimiento de Bootstrapping.
   - **Propósito:** Evaluar la eficiencia computacional del método de Bootstrapping.

Impacto en los Objetivos y Alcances {#impacto-en-los-objetivos-y-alcances}

1. **Evaluación del Rendimiento del Modelo:**

- **Objetivo 2:** Comprobar la variabilidad, el sesgo y la varianza del estimador utilizando el dataset de gasto en publicidad y ventas.
- Las métricas calculadas (RMSE, MAE, ME, varianza y desviación estándar) proporcionan una evaluación detallada del rendimiento del modelo.

2. **Comparación de Métodos de Remuestreo:**
   - **Objetivo 3:** Comparar el método Jackknife con otros métodos de remuestreo como K-fold Cross Validation y Bootstrapping.
   - Las métricas calculadas se utilizarán para comparar el rendimiento del modelo entrenado con Bootstrapping con los modelos entrenados con Jackknife y K-fold CV.
3. **Análisis de la Eficiencia Computacional:**
   - **Objetivo 4:** Analizar la subestimación de la varianza en muestras pequeñas y comparar la eficiencia computacional de los métodos.
   - El tiempo de ejecución calculado proporcionará información sobre la eficiencia computacional del método de Bootstrapping en comparación con los otros métodos de remuestreo.

”

```
# Resultados Bootstrapping
cat(paste0(
  "Bootstrapping RMSE: ", rmse_boot, "\n",
  "Bootstrapping MAE: ", mae_boot, "\n",
  "Bootstrapping ME (Sesgo): ", me_boot, "\n",
  "Bootstrapping Varianza de Predicciones: ", format(var_pred_boot,
                                                     scientific = FALSE), "\n",
  "Bootstrapping Desviación Estándar de Predicciones: ", sd_pred_boot, "\n",
  "Bootstrapping Tiempo de Ejecución: ", time_boot, "\n"
))
```

```
## Bootstrapping RMSE: 1569045.47616184
## Bootstrapping MAE: 1158506.14029274
## Bootstrapping ME (Sesgo): 82743.1406794893
## Bootstrapping Varianza de Predicciones: 20536362901454
## Bootstrapping Desviación Estándar de Predicciones: 4531706.40062372
## Bootstrapping Tiempo de Ejecución: 0.620825052261353
```

Análisis de los Resultados

1. **Bootstrapping RMSE: 1,569,045**
   - **Interpretación:** Este valor de RMSE indica la magnitud promedio del error en las predicciones del modelo de regresión lineal entrenado con Bootstrapping. Un valor más bajo de RMSE sugiere una mejor precisión del modelo.
   - **Comparación:** Este valor es mayor que el RMSE obtenido con Jackknife (1,333,382) y menor que el RMSE obtenido con K-fold CV (1,569,045), indicando que el método Jackknife tiene un mejor rendimiento en términos de error cuadrático medio.
2. **Bootstrapping MAE: 1,158,506**
   - **Interpretación:** Este valor de MAE indica la magnitud promedio de los errores absolutos en las predicciones del modelo. Similar al RMSE, un valor más bajo de MAE indica mejor precisión.
   - **Comparación:** El MAE es mayor que el MAE obtenido con Jackknife (1,333,382) y similar al MAE obtenido con K-fold CV (1,158,506), sugiriendo que el método Jackknife tiene un mejor rendimiento en términos de error absoluto medio.
3. **Bootstrapping ME (Sesgo): 82,743.14**
   - **Interpretación:** Este valor de ME indica el sesgo del modelo, es decir, la tendencia del modelo a sobrestimar o subestimar las predicciones. Un valor cercano a cero sería ideal.
   - **Comparación:** El sesgo es mayor que el sesgo obtenido con Jackknife (-7,209.363) y similar al sesgo obtenido con K-fold CV (82,743.14), lo que sugiere que Bootstrapping y K-fold CV están introduciendo más sesgo en las predicciones en comparación con Jackknife.

4. **Bootstrapping Varianza de Predicciones: 20,536,362,901,454**
   - **Interpretación:** La varianza de las predicciones mide la dispersión de las predicciones del modelo. Una menor varianza indica predicciones más consistentes.
   - **Comparación:** La varianza es menor que la varianza obtenida con Jackknife (24,946,246,116,026) y similar a la varianza obtenida con K-fold CV (20,536,362,901,454), indicando que Bootstrapping y K-fold CV tienen predicciones más consistentes en comparación con Jackknife.
5. **Bootstrapping Desviación Estándar de Predicciones: 4,531,706**
   - **Interpretación:** La desviación estándar de las predicciones proporciona una medida adicional de la dispersión de las predicciones del modelo.
   - **Comparación:** Similar a la varianza, la desviación estándar es menor que la desviación estándar obtenida con Jackknife (4,994,622) y similar a la desviación estándar obtenida con K-fold CV (4,531,706), sugiriendo que Bootstrapping y K-fold CV tienen una variabilidad similar en las predicciones.
6. **Bootstrapping Tiempo de Ejecución: 0.620825052261353 segundos**
   - **Interpretación:** El tiempo de ejecución mide la eficiencia computacional del método de Bootstrapping.
   - **Comparación:** El tiempo de ejecución es significativamente mayor en comparación con Jackknife (0.284446001052856 segundos) y K-fold CV (0.137562990188599 segundos), lo que confirma que Bootstrapping es más intensivo en términos de computación debido a la necesidad de generar múltiples muestras y entrenar modelos en cada una de ellas.

Conclusiones

- **Precisión y Error:** Las métricas de RMSE y MAE muestran que el rendimiento del modelo entrenado con Bootstrapping es menos preciso en comparación con el modelo entrenado con Jackknife, pero similar al modelo entrenado con K-fold CV.
- **Variabilidad y Varianza:** Las métricas de varianza y desviación estándar indican que la variabilidad en las predicciones es menor con Bootstrapping y K-fold CV en comparación con Jackknife, lo que sugiere que estos métodos producen predicciones más consistentes.
- **Eficiencia Computacional:** Bootstrapping tiene un tiempo de ejecución significativamente mayor, confirmando que es menos eficiente computacionalmente en comparación con Jackknife y K-fold CV. ”

```
# Comparación de Resultados
cat(paste0(
  "\nComparación de Resultados:\n",
  "Sesgo (ME):\n",
  "Jackknife: ", avg_me, "\n",
  "K-fold CV: ", me_cv, "\n",
  "Bootstrapping: ", me_boot, "\n"
))
```

```
##
## Comparación de Resultados:
## Sesgo (ME):
## Jackknife: 82700.1842612256
## K-fold CV: 82743.1406794893
## Bootstrapping: 82743.1406794893
```

Análisis de los Resultados:

Sesgo (ME):

- Jackknife: El sesgo de Jackknife es de 82,700.18, lo que indica una ligera subestimación en las predicciones en comparación con los otros métodos. Este valor más bajo sugiere que Jackknife tiene una menor tendencia a sobreestimar o subestimar las predicciones.
- K-fold CV: El sesgo de K-fold CV es de 82,743.14, lo que indica una leve sobreestimación en las

predicciones en comparación con Jackknife. Este valor es casi idéntico al obtenido con Bootstrapping.
- Bootstrapping: El sesgo de Bootstrapping es de 82,743.14, similar al de K-fold CV, lo que sugiere que ambos métodos introducen un sesgo ligeramente mayor en las predicciones en comparación con Jackknife.

Conclusiones: Precisión del Sesgo:

- Jackknife muestra un sesgo menor en comparación con K-fold CV y Bootstrapping, lo que puede ser beneficioso en escenarios donde la precisión del sesgo es crítica.
- K-fold CV y Bootstrapping tienen valores de sesgo casi idénticos, indicando una similitud en la tendencia a sobreestimar o subestimar las predicciones.

Elección del Método:

- Jackknife es preferible si el objetivo es minimizar el sesgo en las predicciones, aunque puede no ser tan eficiente computacionalmente como K-fold CV.
- K-fold CV y Bootstrapping son adecuados cuando se busca un equilibrio entre la precisión del sesgo y la variabilidad de las predicciones, aunque Bootstrapping es más intensivo computacionalmente. ”

```
cat(paste0(
  "\nVariabilidad (Desviación Estándar de Predicciones):\n",
  "Jackknife: ", sd_pred_jackknife, "\n",
  "K-fold CV: ", sd_pred_cv, "\n",
  "Bootstrapping: ", sd_pred_boot, "\n"
))
```

```
##
## Variabilidad (Desviación Estándar de Predicciones):
## Jackknife: 4511349.17902163
## K-fold CV: 4531706.40062372
## Bootstrapping: 4531706.40062372
```

Análisis de los Resultados:

1. Variabilidad (Desviación Estándar de Predicciones):

- Jackknife: La desviación estándar de las predicciones de Jackknife es de 4,511,349, lo que indica una variabilidad ligeramente menor en comparación con K-fold CV y Bootstrapping. Esto sugiere que las predicciones de Jackknife son más consistentes.
- K-fold CV: La desviación estándar de K-fold CV es de 4,531,706, lo que muestra una variabilidad similar a la de Bootstrapping pero ligeramente mayor que la de Jackknife.
- Bootstrapping: La desviación estándar de Bootstrapping es idéntica a la de K-fold CV, indicando que ambos métodos tienen una variabilidad similar en sus predicciones.

Conclusiones: Consistencia de las Predicciones:

- Jackknife tiene una variabilidad de predicciones ligeramente menor, lo que puede ser ventajoso en situaciones donde se requiere una mayor consistencia en las predicciones.
- K-fold CV y Bootstrapping presentan una variabilidad de predicciones casi idéntica, lo que sugiere que ambos métodos son igualmente consistentes, aunque ligeramente menos que Jackknife.

Elección del Método:

- Jackknife es preferible si el objetivo es minimizar la variabilidad en las predicciones.
- K-fold CV y Bootstrapping son adecuados cuando se busca un equilibrio entre la precisión del sesgo y la variabilidad de las predicciones. ”

```
cat(paste0(
  "\nEficiencia Computacional (Tiempo de Ejecución):\n",
  "Jackknife: ", time_jackknife, "\n",
  "K-fold CV: ", time_cv, "\n",
```

```
  "Bootstrapping: ", time_boot, "\n"
))
```

```
##
## Eficiencia Computacional (Tiempo de Ejecución):
## Jackknife: 0.284446001052856
## K-fold CV: 0.137562990188599
## Bootstrapping: 0.620825052261353
```

Análisis de los Resultados:

1. Eficiencia Computacional (Tiempo de Ejecución):

- Jackknife: El tiempo de ejecución para Jackknife es de 0.284446001052856 segundos. Aunque es más eficiente que Bootstrapping, es más lento que K-fold CV. Esto se debe a que Jackknife requiere la eliminación sistemática de cada observación, lo que puede ser computacionalmente intensivo, especialmente con un mayor número de filas.
- K-fold CV: El tiempo de ejecución para K-fold CV es de 0.137562990188599 segundos, siendo el más rápido de los tres métodos. Esto se explica porque K-fold CV divide el conjunto de datos en un número fijo de pliegues (en este caso, 10), lo que requiere menos iteraciones comparado con Jackknife.
- Bootstrapping: El tiempo de ejecución para Bootstrapping es de 0.620825052261353, siendo el más lento de los tres métodos. Bootstrapping genera múltiples muestras con reemplazo y entrena el modelo en cada muestra, lo que requiere una mayor cantidad de operaciones computacionales.

Justificación del Tiempo de Ejecución del Jackknife vs K-fold CV

Comparación de Procesos

1. **Jackknife:**
   - El método Jackknife implica eliminar una observación a la vez del conjunto de datos de entrenamiento y entrenar el modelo en cada subconjunto resultante.
   - En un conjunto de datos con (n) observaciones, el método Jackknife realizará (n) iteraciones.
   - **Número de Iteraciones:** Si el conjunto de datos tiene 200 observaciones, se realizarán 200 iteraciones, ya que cada observación es eliminada una vez.
   - **Proceso:** Para cada iteración, se entrena el modelo con (n-1) observaciones y se evalúa en la observación eliminada. Este proceso se repite (n) veces.
2. **K-fold Cross Validation (K-fold CV):**
   - En K-fold CV, el conjunto de datos se divide en (k) partes (pliegues) de manera que cada parte se utiliza una vez como conjunto de validación, mientras que los (k-1) pliegues restantes se utilizan como conjunto de entrenamiento.
   - **Número de Iteraciones:** Si se elige (k = 10), se realizarán 10 iteraciones.
   - **Proceso:** Para cada iteración, se entrena el modelo con (\frac{(k-1)}{k}) del conjunto de datos y se evalúa en el pliegue de validación correspondiente. Este proceso se repite (k) veces.

Análisis de Iteraciones y Eficiencia Computacional

- **Número de Iteraciones:**
  - **Jackknife:** 200 iteraciones (una por cada observación).
  - **K-fold CV:** 10 iteraciones (una por cada pliegue).
  - Debido a que Jackknife realiza muchas más iteraciones que K-fold CV, es natural que el tiempo de ejecución sea mayor.
- **Carga Computacional:**
  - **Jackknife:** Cada iteración del Jackknife requiere recalcular el modelo desde cero, lo que implica entrenar el modelo 200 veces.
  - **K-fold CV:** Cada iteración de K-fold CV entrena el modelo solo 10 veces, distribuyendo la carga computacional entre menos iteraciones.

El método Jackknife tiene un mayor tiempo de ejecución que K-fold CV debido a la mayor cantidad de

iteraciones necesarias. Aunque Jackknife es eficiente computacionalmente en comparación con métodos como Bootstrapping, la naturaleza exhaustiva de su proceso (eliminar una observación a la vez) hace que sea más lento que K-fold CV cuando se utiliza un número de pliegues razonable (e.g., 10). Por lo tanto, la ventaja de Jackknife en términos de eficiencia computacional es más evidente en comparación con Bootstrapping que con K-fold CV.

Métricas de Rendimiento:

- **RMSE:**
  - **Jackknife:** 1,569,260
  - **K-fold CV:** 1,569,045
  - **Bootstrapping:** 1,569,045
- **MAE:**
  - **Jackknife:** 1,158,544
  - **K-fold CV:** 1,158,506
  - **Bootstrapping:** 1,158,506
- **Interpretación:** Las métricas de rendimiento (RMSE y MAE) fueron muy similares entre los tres métodos, con pequeñas diferencias. Jackknife mostró un rendimiento comparable a K-fold CV y Bootstrapping.

Conclusiones Finales

Objetivos y Alcances Cumplidos

Objetivo 1: Implementar el método de remuestreo Jackknife para evaluar el rendimiento de un modelo de regresión lineal.

- El método Jackknife fue implementado exitosamente para evaluar el modelo de regresión lineal. Se eliminaron sistemáticamente observaciones individuales y se calcularon las predicciones y métricas de rendimiento para cada subconjunto.

Objetivo 2: Comprobar la variabilidad, el sesgo y la varianza del estimador utilizando el dataset de gasto en publicidad y ventas.

- Se estimaron la variabilidad, el sesgo y la varianza del modelo de regresión lineal utilizando el método Jackknife. Los resultados obtenidos demostraron cómo varían las predicciones y cómo se comporta el modelo frente a diferentes subconjuntos de datos.

Objetivo 3: Comparar el método Jackknife con otros métodos de remuestreo como K-fold Cross Validation y Bootstrapping.

- Se compararon los resultados del método Jackknife con K-fold Cross Validation y Bootstrapping en términos de métricas de rendimiento (RMSE, MAE, ME), variabilidad y eficiencia computacional.

Objetivo 4: Analizar la subestimación de la varianza en muestras pequeñas y comparar la eficiencia computacional de los métodos.

- La subestimación de la varianza en el método Jackknife fue evidente en comparación con K-fold CV y Bootstrapping. Además, se comparó la eficiencia computacional de cada método, demostrando las ventajas y desventajas relativas.

Objetivo 5: Evaluar y comparar las métricas de rendimiento y su precisión entre los métodos de remuestreo.

- Se evaluaron y compararon las métricas de rendimiento (RMSE, MAE, ME) entre Jackknife, K-fold Cross Validation y Bootstrapping. Los resultados mostraron las diferencias en precisión y sesgo entre los métodos.

Alcances Cumplidos

1. **Cargar y explorar el dataset de gasto en publicidad y ventas.**

- Se cargó y exploró el dataset, incluyendo la normalización de las variables para un análisis consistente.

2. **Dividir los datos en conjunto de entrenamiento y prueba.**
   - Los datos fueron divididos en conjuntos de entrenamiento (80%) y prueba (20%) utilizando una estratificación por la variable objetivo.
3. **Aplicar el método Jackknife para estimar la variabilidad, el sesgo y la varianza del modelo de regresión lineal.**
   - Se aplicó Jackknife y se calcularon las métricas de variabilidad, sesgo y varianza del modelo de regresión lineal.
4. **Evaluar el modelo utilizando métricas como RMSE, MAE y ME**
   - Se calcularon las métricas de rendimiento (RMSE, MAE, ME) para evaluar el modelo.
5. **Comparar los resultados obtenidos con los métodos K-fold Cross Validation y Bootstrapping.**
   - Se compararon las métricas de rendimiento, variabilidad y tiempo de ejecución entre los métodos Jackknife, K-fold Cross Validation y Bootstrapping.
6. **Analizar la eficiencia computacional de cada método de remuestreo.**
   - Se analizaron y compararon los tiempos de ejecución de cada método, demostrando las diferencias en eficiencia computacional.
7. **Explicar y demostrar el proceso paso a paso, incluyendo el cálculo de métricas de rendimiento, variabilidad y tiempo de ejecución.**
   - Se explicó y demostró cada paso del proceso, incluyendo la implementación de los métodos de remuestreo, el cálculo de métricas de rendimiento y la comparación de los resultados.

- **Eficiencia Computacional:** Jackknife mostró ser más eficiente computacionalmente que Bootstrapping, pero no tan eficiente como K-fold CV. La mayor cantidad de iteraciones necesarias en Jackknife explica su tiempo de ejecución relativamente mayor.
- **Variabilidad y Sesgo:** Jackknife presentó una menor desviación estándar de las predicciones y un sesgo ligeramente menor en comparación con K-fold CV y Bootstrapping. Esto sugiere que Jackknife puede ser útil en contextos donde la reducción del sesgo y la variabilidad es prioritaria.
- **Subestimación de la Varianza:** Los resultados mostraron que Jackknife tiende a subestimar la varianza del estimador en comparación con K-fold CV y Bootstrapping, lo cual es consistente con la teoría.
- **Comparación General:** Jackknife es un método simple y eficiente para estimar el sesgo y la variabilidad, especialmente en muestras pequeñas. Sin embargo, K-fold CV proporciona una mejor estimación de la variabilidad del modelo y es más adecuado para evaluar el rendimiento predictivo en conjuntos de datos más grandes.

El método Jackknife cumplió con los objetivos planteados y se demostró su utilidad y limitaciones en comparación con K-fold Cross Validation y Bootstrapping. La implementación y comparación de estos métodos proporcionó una comprensión clara de sus ventajas y desventajas relativas en el contexto del análisis de regresión lineal con el dataset de gasto en publicidad y ventas.

Parte 2: Aprendizaje Supervisado Regresión, Analisis Descriptivo (Items 1-3)

Cargo las librerias necesarias para trabajar

```
# importacion de librerias
library(ggplot2)
library(dplyr)
library(readr)
```

Importacion de base de datos

```
base_r <- read_csv("E:/data science 2024/R begginers/Trabajo ML 1/Daegu_Real_Estate_data.csv")
```

```
## Rows: 5891 Columns: 30
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (6): HallwayType, HeatingType, AptManageType, TimeToBusStop, TimeToSubw...
## dbl (24): SalePrice, YearBuilt, YrSold, MonthSold, Size(sqf), Floor, N_Parki...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
base_r <- as.data.frame(base_r)
```

El mensaje muestra información sobre las columnas del dataset. Indica que hay 5891 filas y 30 columnas. Las columnas se dividen en dos tipos: chr (carácter) y dbl (doble). Esto proporciona una visión general del tipo de datos en cada columna, lo cual es esencial para la limpieza y el preprocesamiento posterior del dataset.

Análisis del Resultado Estructura del Dataset:

El dataset contiene 5891 filas y 30 columnas. Hay 6 columnas de tipo carácter (chr) y 24 columnas de tipo numérico (dbl). Relevancia de los Datos:

La estructura del dataset parece adecuada para un análisis de regresión y modelado de machine learning, ya que contiene una mezcla de variables categóricas y numéricas. Es crucial inspeccionar más a fondo las columnas para identificar posibles problemas de datos como valores faltantes, outliers y necesidad de transformación de datos.

1. Describa cada una de las variables e indique si corresponden a variables numéricas o categóricas. Si considera que hay un número excesivo de variables (o variables irrelevantes) describir solamente las de mayor interés.

```
# veo información general del dataframe
str(base_r)
```

```
## 'data.frame':    5891 obs. of  30 variables:
##  $ SalePrice                      : num  141592 51327 48672 380530 221238 ...
##  $ YearBuilt                      : num  2006 1985 1985 2006 1993 ...
##  $ YrSold                         : num  2007 2007 2007 2007 2007 ...
##  $ MonthSold                      : num  8 8 8 8 8 8 8 8 8 8 ...
##  $ Size(sqf)                      : num  814 587 587 2056 1761 ...
##  $ Floor                          : num  3 8 6 8 3 5 2 10 3 13 ...
##  $ HallwayType                    : chr  "terraced" "corridor" "corridor" "terraced" ...
##  $ HeatingType                    : chr  "individual_heating" "individual_heating" "individual_hea
##  $ AptManageType                  : chr  "management_in_trust" "self_management" "self_management
##  $ N_Parkinglot(Ground)           : num  111 80 80 249 523 200 142 523 523 142 ...
##  $ N_Parkinglot(Basement)         : num  184 76 76 536 536 0 79 536 536 79 ...
##  $ TimeToBusStop                  : chr  "5min~10min" "0~5min" "0~5min" "0~5min" ...
```

```
##  $ TimeToSubway                      : chr  "10min~15min" "5min~10min" "5min~10min" "0-5min" ...
##  $ N_APT                             : num  3 1 1 6 8 3 3 8 8 3 ...
##  $ N_manager                         : num  3 2 2 5 8 5 4 8 8 4 ...
##  $ N_elevators                       : num  0 2 2 11 20 10 8 20 20 8 ...
##  $ SubwayStation                     : chr  "Kyungbuk_uni_hospital" "Daegu" "Daegu" "Sin-nam" ...
##  $ N_FacilitiesNearBy(PublicOffice)  : num  2 5 5 1 6 7 5 6 6 5 ...
##  $ N_FacilitiesNearBy(Hospital)      : num  1 1 1 1 2 1 1 2 2 1 ...
##  $ N_FacilitiesNearBy(Dpartmentstore): num  1 2 2 0 0 1 1 0 0 1 ...
##  $ N_FacilitiesNearBy(Mall)          : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ N_FacilitiesNearBy(ETC)           : num  1 2 2 0 5 5 1 5 5 1 ...
##  $ N_FacilitiesNearBy(Park)          : num  0 1 1 0 0 1 0 0 0 0 ...
##  $ N_SchoolNearBy(Elementary)        : num  3 2 2 2 4 4 3 4 4 3 ...
##  $ N_SchoolNearBy(Middle)            : num  2 1 1 2 3 3 3 3 3 3 ...
##  $ N_SchoolNearBy(High)              : num  2 1 1 1 5 5 4 5 5 4 ...
##  $ N_SchoolNearBy(University)        : num  2 0 0 2 5 5 4 5 5 4 ...
##  $ N_FacilitiesInApt                 : num  5 3 3 5 4 3 3 4 4 3 ...
##  $ N_FacilitiesNearBy(Total)         : num  6 12 12 3 14 16 9 14 14 9 ...
##  $ N_SchoolNearBy(Total)             : num  9 4 4 7 17 17 14 17 17 14 ...
```

Con esta información realizo una descripcion de cada una de las columnas, respecto a sus significados.

1. `SalePrice`: Precio de venta del apartamento.
2. `YearBuilt`: Año en que fue construido el edificio del apartamento.
3. `YrSold`: Año en que se vendió el apartamento.
4. `MonthSold`: Mes en que se vendió el apartamento.
5. `Size(sqf)`: Tamaño del apartamento en pies cuadrados.
6. `Floor`: Número de piso en el que se encuentra el apartamento.
7. `HallwayType`: Tipo de pasillo del edificio (ej. corredor interno, externo).
8. `HeatingType`: Tipo de calefacción utilizada en el apartamento.
9. `AptManageType`: Tipo de gestión del apartamento (ej. administración privada, comunitaria).
10. `N_Parkinglot(Ground)`: Número de plazas de estacionamientos en superficie.
11. `N_Parkinglot(Basement)`: Número de plazas de estcionamiento en sótano.
12. `TimeToBusStop`: Tiempo en minutos hasta la parada de autobús más cercana.
13. `TimeToSubway`: Tiempo en minutos hasta la estación de metro más cercana.
14. `N_APT`: Número total de apartamentos en el edificio.
15. `N_manager`: Número de gestores del edificio.
16. `N_elevators`: Número de ascensores en el edificio.
17. `SubwayStation`: Nombre de la estación de metro más cercana.
18. `N_FacilitiesNearBy(PublicOffice)`: Número de oficinas públicas cercanas.
19. `N_FacilitiesNearBy(Hospital)`: Número de hospitales cercanos.
20. `N_FacilitiesNearBy(Dpartmentstore)`: Número de grandes almacenes cercanos.
21. `N_FacilitiesNearBy(Mall)`: Número de centros comerciales cercanos.
22. `N_FacilitiesNearBy(ETC)`: Número de otras instalaciones cercanas.
23. `N_FacilitiesNearBy(Park)`: Número de parques cercanos.
24. `N_SchoolNearBy(Elementary)`: Número de escuelas primarias cercanas.
25. `N_SchoolNearBy(Middle)`: Número de escuelas secundarias cercanas.
26. `N_SchoolNearBy(High)`: Número de escuelas preparatorias cercanas.
27. `N_SchoolNearBy(University)`: Número de universidades cercanas.
28. `N_FacilitiesInApt`: Número de instalaciones dentro del edificio del apartamento.
29. `N_FacilitiesNearBy(Total)`: Número total de instalaciones cercanas.
30. `N_SchoolNearBy(Total)`: Número total de escuelas cercanas.

Primero veo las primeras 6 filas y luego las ultimas 6 para ver si hay algo extraño que valga la pena solucionar o mencionar.

```r
#ver los primeros datos del dataframe en R
head(base_r)
```

```
##   SalePrice YearBuilt YrSold MonthSold Size(sqf) Floor HallwayType
## 1    141592      2006   2007         8       814     3    terraced
## 2     51327      1985   2007         8       587     8    corridor
## 3     48672      1985   2007         8       587     6    corridor
## 4    380530      2006   2007         8      2056     8    terraced
## 5    221238      1993   2007         8      1761     3       mixed
## 6     35840      1992   2007         8       355     5    corridor
##          HeatingType       AptManageType N_Parkinglot(Ground)
## 1 individual_heating management_in_trust                  111
## 2 individual_heating     self_management                   80
## 3 individual_heating     self_management                   80
## 4 individual_heating management_in_trust                  249
## 5 individual_heating management_in_trust                  523
## 6 individual_heating management_in_trust                  200
##   N_Parkinglot(Basement) TimeToBusStop TimeToSubway N_APT N_manager N_elevators
## 1                    184    5min~10min  10min~15min     3         3          0
## 2                     76        0~5min   5min~10min     1         2          2
## 3                     76        0~5min   5min~10min     1         2          2
## 4                    536        0~5min       0-5min     6         5         11
## 5                    536        0~5min  15min~20min     8         8         20
## 6                      0    5min~10min  10min~15min     3         5         10
##           SubwayStation N_FacilitiesNearBy(PublicOffice)
## 1 Kyungbuk_uni_hospital                               2
## 2                Daegu                               5
## 3                Daegu                               5
## 4              Sin-nam                               1
## 5            Myung-duk                               6
## 6            Myung-duk                               7
##   N_FacilitiesNearBy(Hospital) N_FacilitiesNearBy(Dpartmentstore)
## 1                            1                                 1
## 2                            1                                 2
## 3                            1                                 2
## 4                            1                                 0
## 5                            2                                 0
## 6                            1                                 1
##   N_FacilitiesNearBy(Mall) N_FacilitiesNearBy(ETC) N_FacilitiesNearBy(Park)
## 1                        1                       1                       0
## 2                        1                       2                       1
## 3                        1                       2                       1
## 4                        1                       0                       0
## 5                        1                       5                       0
## 6                        1                       5                       1
##   N_SchoolNearBy(Elementary) N_SchoolNearBy(Middle) N_SchoolNearBy(High)
## 1                          3                       2                    2
## 2                          2                       1                    1
## 3                          2                       1                    1
## 4                          2                       2                    1
## 5                          4                       3                    5
## 6                          4                       3                    5
##   N_SchoolNearBy(University) N_FacilitiesInApt N_FacilitiesNearBy(Total)
## 1                          2                5                         6
```

```
## 2                          0                3                         12
## 3                          0                3                         12
## 4                          2                5                          3
## 5                          5                4                         14
## 6                          5                3                         16
##   N_SchoolNearBy(Total)
## 1                     9
## 2                     4
## 3                     4
## 4                     7
## 5                    17
## 6                    17
```

```
#ver los últimos datos del dataframe en R
tail(base_r)
```

```
##      SalePrice YearBuilt YrSold MonthSold Size(sqf) Floor HallwayType
## 5886    482300      2007   2017         8      1643     4    terraced
## 5887    511504      2007   2017         8      1643    19    terraced
## 5888    298230      2006   2017         8       903    13    terraced
## 5889    357522      2007   2017         8       868    20    terraced
## 5890    312389      1978   2017         8      1327     1    corridor
## 5891    393805      2007   2017         8       868    13    terraced
##             HeatingType       AptManageType N_Parkinglot(Ground)
## 5886 individual_heating management_in_trust                    0
## 5887 individual_heating management_in_trust                    0
## 5888 individual_heating management_in_trust                  123
## 5889 individual_heating management_in_trust                    0
## 5890 individual_heating     self_management                   87
## 5891 individual_heating management_in_trust                    0
##      N_Parkinglot(Basement) TimeToBusStop TimeToSubway N_APT N_manager
## 5886                   1270        0~5min       0-5min     7        14
## 5887                   1270        0~5min       0-5min     7        14
## 5888                    181    5min~10min       0-5min     3         3
## 5889                   1270        0~5min       0-5min     7        14
## 5890                      0        0~5min       0-5min     2         1
## 5891                   1270        0~5min       0-5min     7        14
##      N_elevators         SubwayStation N_FacilitiesNearBy(PublicOffice)
## 5886          16 Kyungbuk_uni_hospital                                3
## 5887          16 Kyungbuk_uni_hospital                                3
## 5888          11             Myung-duk                                3
## 5889          16 Kyungbuk_uni_hospital                                3
## 5890           4 Kyungbuk_uni_hospital                                3
## 5891          16 Kyungbuk_uni_hospital                                3
##      N_FacilitiesNearBy(Hospital) N_FacilitiesNearBy(Dpartmentstore)
## 5886                            1                                  2
## 5887                            1                                  2
## 5888                            1                                  1
## 5889                            1                                  2
## 5890                            2                                  1
## 5891                            1                                  2
##      N_FacilitiesNearBy(Mall) N_FacilitiesNearBy(ETC) N_FacilitiesNearBy(Park)
## 5886                        1                       0                       2
## 5887                        1                       0                       2
## 5888                        1                       2                       0
```

```
## 5889                        1                       0                        2
## 5890                        1                       0                        0
## 5891                        1                       0                        2
##      N_SchoolNearBy(Elementary) N_SchoolNearBy(Middle) N_SchoolNearBy(High)
## 5886                          3                      3                     2
## 5887                          3                      3                     2
## 5888                          4                      3                     3
## 5889                          3                      3                     2
## 5890                          3                      3                     3
## 5891                          3                      3                     2
##      N_SchoolNearBy(University) N_FacilitiesInApt N_FacilitiesNearBy(Total)
## 5886                          2                10                        9
## 5887                          2                10                        9
## 5888                          1                 4                        8
## 5889                          2                10                        9
## 5890                          2                 3                        7
## 5891                          2                10                        9
##      N_SchoolNearBy(Total)
## 5886                    10
## 5887                    10
## 5888                    11
## 5889                    10
## 5890                    11
## 5891                    10
```

Procedo a seleccionar y guardar en variables las columnas categoricas y las numericas, dado que a futuro necesitaré realizar un tratamiento de creacion de variables dummies (columnas categoricas) y normalizacion de las columnas numericas.

```
# selecciono que columnas son numericas y cuales son categoricas
vars_num <- base_r %>% select_if(is.numeric) %>% colnames()
vars_cat <- base_r %>% select_if(is.character) %>% colnames()
# veo las variables numericas y categoricas (respectivamente)
vars_num
```

```
##  [1] "SalePrice"                          "YearBuilt"
##  [3] "YrSold"                             "MonthSold"
##  [5] "Size(sqf)"                          "Floor"
##  [7] "N_Parkinglot(Ground)"               "N_Parkinglot(Basement)"
##  [9] "N_APT"                              "N_manager"
## [11] "N_elevators"                        "N_FacilitiesNearBy(PublicOffice)"
## [13] "N_FacilitiesNearBy(Hospital)"       "N_FacilitiesNearBy(Dpartmentstore)"
## [15] "N_FacilitiesNearBy(Mall)"           "N_FacilitiesNearBy(ETC)"
## [17] "N_FacilitiesNearBy(Park)"           "N_SchoolNearBy(Elementary)"
## [19] "N_SchoolNearBy(Middle)"             "N_SchoolNearBy(High)"
## [21] "N_SchoolNearBy(University)"         "N_FacilitiesInApt"
## [23] "N_FacilitiesNearBy(Total)"          "N_SchoolNearBy(Total)"
```

```
vars_cat
```

```
## [1] "HallwayType"   "HeatingType"   "AptManageType" "TimeToBusStop"
## [5] "TimeToSubway"  "SubwayStation"
```

La siguiente celda proporciona un resumen estadístico de las variables numéricas seleccionadas en el dataset de transacciones de apartamentos en Daegu. El resumen incluye la media, mediana (Q2), mínimo, máximo, primer cuartil (Q1) y tercer cuartil (Q3) de cada variable elegida.

```r
# veo un resumen de las variables numericas
# como la media, mediana (Q2), minimo, maximo, Q1, Q3
summary(base_r[c("SalePrice", "YearBuilt", "YrSold", "MonthSold", "Size(sqf)")])
```

```
##    SalePrice        YearBuilt        YrSold       MonthSold
##  Min.   : 32743   Min.   :1978   Min.   :2007   Min.   : 1.00
##  1st Qu.:144247   1st Qu.:1993   1st Qu.:2010   1st Qu.: 3.00
##  Median :207964   Median :2006   Median :2013   Median : 6.00
##  Mean   :221218   Mean   :2003   Mean   :2013   Mean   : 6.16
##  3rd Qu.:291150   3rd Qu.:2008   3rd Qu.:2015   3rd Qu.: 9.00
##  Max.   :585840   Max.   :2015   Max.   :2017   Max.   :12.00
##    Size(sqf)
##  Min.   : 135.0
##  1st Qu.: 644.0
##  Median : 910.0
##  Mean   : 955.6
##  3rd Qu.:1149.0
##  Max.   :2337.0
```

Análisis

- **SalePrice**: La mediana del precio de venta es 207,964, mientras que la media es 221,218, lo que indica una distribución ligeramente sesgada hacia la derecha.
- **YearBuilt**: La mediana del año de construcción es 2006, con la mayoría de los apartamentos construidos entre 1993 y 2008.
- **YrSold**: Las ventas están distribuidas uniformemente entre 2007 y 2017, con una mediana en 2013.
- **MonthSold**: Las ventas ocurren de manera uniforme a lo largo del año, con una mediana en el mes 6 (junio).
- **Size(sqf)**: El tamaño de los apartamentos varía considerablemente, con una mediana de 910 pies cuadrados y un máximo de 2,337 pies cuadrados.

Este análisis inicial nos proporciona una visión general de la distribución y el rango de valores de las variables numéricas en el dataset, lo cual es fundamental para los pasos posteriores del análisis y modelado de datos.

La siguiente celda selecciona las columnas numéricas de interés y calcula el Rango Intercuartílico (IQR) para cada una. El IQR es una medida de la dispersión de los datos y se calcula como la diferencia entre el tercer cuartil (Q3) y el primer cuartil (Q1).

Columnas Numéricas Seleccionadas

Las columnas numéricas seleccionadas para el análisis son:

- **SalePrice**: Precio de venta del apartamento.
- **YearBuilt**: Año de construcción del apartamento.
- **YrSold**: Año en que se vendió el apartamento.
- **MonthSold**: Mes en que se vendió el apartamento.
- **Size(sqf)**: Tamaño del apartamento en pies cuadrados.

```r
# selecciono las columnas numericas que me interesan
# luego calculo el rango intercuartilico de cada una
col_interes_num <- c("SalePrice", "YearBuilt", "YrSold", "MonthSold",
                     "Size(sqf)")
valores_iqr <- sapply(base_r[col_interes_num], IQR)
valores_iqr
```

```
## SalePrice YearBuilt    YrSold MonthSold Size(sqf)
##    146903        15         5         6       505
```

- **SalePrice**: Un IQR de 146,903 indica una dispersión considerable en los precios de venta, lo que sugiere una variabilidad significativa en los precios de los apartamentos.
- **YearBuilt**: Un IQR de 15 años indica que la mayoría de los apartamentos fueron construidos en un período de 15 años.
- **YrSold**: Un IQR de 5 años indica que las ventas están distribuidas en un período de 5 años, lo que sugiere una distribución relativamente uniforme de las ventas a lo largo del tiempo.
- **MonthSold**: Un IQR de 6 meses indica que las ventas están distribuidas uniformemente a lo largo del año.
- **Size(sqf)**: Un IQR de 505 pies cuadrados indica una dispersión considerable en los tamaños de los apartamentos, lo que sugiere una variabilidad significativa en el tamaño de los apartamentos.

Este análisis del IQR proporciona información adicional sobre la dispersión de los datos en las variables seleccionadas, lo cual es fundamental para entender mejor la distribución de los datos y detectar posibles outliers.

La próxima celda selecciona las columnas categóricas de interés y calcula la cantidad de valores por cada categoría o clase. Este análisis es crucial para entender la distribución de las categorías y detectar posibles desequilibrios en los datos.

```
# selecciono las columnas categoricas
# luego veo la cantidad de valores por cada categoria o clase
col_interes_cat <- c("HallwayType", "HeatingType", "AptManageType",
                     "TimeToBusStop", "TimeToSubway", "SubwayStation")
valores_por_clase <- lapply(base_r[col_interes_cat], table)
valores_por_clase
```

```
## $HallwayType
##
## corridor    mixed terraced
##      637     1690     3564
##
## $HeatingType
##
##    central_heating individual_heating
##                300               5591
##
## $AptManageType
##
## management_in_trust     self_management
##                5542                 349
##
## $TimeToBusStop
##
##      0~5min 10min~15min  5min~10min
##        4509          55        1327
##
## $TimeToSubway
##
##             0-5min        10min~15min        15min~20min         5min~10min
##               2759                806                953               1135
## no_bus_stop_nearby
##                238
##
## $SubwayStation
##
##               Bangoge            Banwoldang      Chil-sung-market
```

```
##                   737                   748                   115
##                 Daegu Kyungbuk_uni_hospital             Myung-duk
##                    85                  1644                  1507
##      no_subway_nearby               Sin-nam
##                   404                   651
```

- **HallwayType**: La mayoría de los apartamentos tienen pasillos terraced, seguidos por mixed y corridor.
- **HeatingType**: La gran mayoría de los apartamentos tienen calefacción individual.
- **AptManageType**: La mayoría de los apartamentos están bajo administración en fideicomiso.
- **TimeToBusStop**: La mayoría de los apartamentos están a 0-5 minutos de un paradero de bus.
- **TimeToSubway**: La mayoría de los apartamentos están a 0-5 minutos de una estación de metro, seguidos por 5-10 minutos.
- **SubwayStation**: Las estaciones más cercanas a la mayoría de los apartamentos son Kyung-buk_uni_hospital y Myung-duk.

Este análisis de frecuencia de clases proporciona información valiosa sobre la distribución de las categorías en las variables seleccionadas, lo cual es fundamental para entender mejor los datos y preparar el modelo de machine learning.

La siguiente celda calcula el porcentaje de valores por cada categoría en las columnas categóricas seleccionadas. Este análisis es crucial para entender la distribución relativa de cada categoría y detectar posibles desequilibrios en los datos.

```r
# ahora veo los porcentajes de valores por cada categoria
# en cada columna categorica
porcentaje_por_clase <- lapply(base_r[, col_interes_cat],
                               function(x) prop.table(table(x)) * 100)
porcentaje_por_clase
```

```
## $HallwayType
## x
## corridor    mixed terraced
## 10.81310 28.68783 60.49907
##
## $HeatingType
## x
##    central_heating individual_heating
##           5.092514          94.907486
##
## $AptManageType
## x
## management_in_trust     self_management
##           94.075709            5.924291
##
## $TimeToBusStop
## x
##      0~5min 10min~15min  5min~10min
##  76.5404855   0.9336276  22.5258869
##
## $TimeToSubway
## x
##             0-5min        10min~15min        15min~20min         5min~10min
##          46.834154          13.681888          16.177219          19.266678
## no_bus_stop_nearby
##           4.040061
##
```

```
## $SubwayStation
## x
##               Bangoge            Banwoldang      Chil-sung-market
##             12.510609             12.697335              1.952130
##                 Daegu Kyungbuk_uni_hospital             Myung-duk
##              1.442879             27.906977             25.581395
##      no_subway_nearby               Sin-nam
##              6.857919             11.050755
```

- **HallwayType**: La mayoría de los apartamentos tienen pasillos terraced (60.50%), seguidos por mixed (28.69%) y corridor (10.81%).
- **HeatingType**: La mayoría de los apartamentos tienen calefacción individual (94.91%).
- **AptManageType**: La mayoría de los apartamentos están bajo administración en fideicomiso (94.08%).
- **TimeToBusStop**: La mayoría de los apartamentos están a 0-5 minutos de un paradero de bus (76.54%).
- **TimeToSubway**: La mayoría de los apartamentos están a 0-5 minutos de una estación de metro (46.83%), seguidos por 5-10 minutos (19.27%).
- **SubwayStation**: Las estaciones más cercanas a la mayoría de los apartamentos son Kyung-buk_uni_hospital (27.91%) y y Myung-duk (25.58%).
- Es muy extraño que existan 238 valores "no_bus_stop_nearby" en la columna "TimeToSubway" porque no tiene nada que ver que esté en esa columna, por lo que al parecer esa fue una mala traducción o un error y que en realidad es otra cosa, pero tampoco puedo reemplazar "no_bus_stop_nearby" por "no_subway_nearby", dado que los valores "no_subway_nearby" si existen y estan dentro de la columna "SubwayStation" donde son 404 valores y si tiene sentido que esten ahí, por lo que no calzan con los 238 valores de "no_bus_stop_nearby" en la columna de "TimeToSubway".
- Tampoco tiene sentido que realmente signifique "no_bus_stop_nearby", osea que no haya un paradero de bus cercano, dado que en la columna "TimeToBusStop" todas las propiedades estan a maximo de 10 minutos de distancia. Por lo que dado que no hace sentido cambiarlo a otro concepto especifico, ni tampoco lo puedo mantener tal cual porque su significado no tiene sentido en esa columna, tengo 3 opciones eliminar las filas, o bien cambiarlos esos valores a "No_Info" representando que no hay informacion al respecto, dado que no se que significan esos valores dentro de la columna o finalmente dejarlo todo tal cual. Y al ver que si elimino las filas perdería 238 filas de un total de 5891, osea alrededor del 4%, he procedido a mejor a dejarlo tal cual, dado que es riesgoso perder la informacion de esas filas.

1. Realice estadística descriptiva para 5 variables (énfasis principal en la variable dependiente). Asegúrese de que sean de distinto tipo (numéricas, categóricas, etc.). Incorpore análisis gráfico.

Analisis Univariado Variables Numericas Seleccionadas

Considero que las variables numericas más importantes son 3: "SalePrice", "YearBuilt" y "Size(sqf)"

La siguiente celda verifica la presencia de valores faltantes en cada columna del conjunto de datos. La identificación de valores faltantes es crucial para el preprocesamiento de datos, ya que los valores faltantes pueden afectar el rendimiento del modelo de machine learning.

```
#chequeo si hay valores faltantes en las columnas
# y cuantos son
valores_faltantes <- sapply(base_r, function(x) sum(is.na(x)))
valores_faltantes
```

```
##                          SalePrice                          YearBuilt
##                                  0                                  0
##                            YrSold                          MonthSold
##                                  0                                  0
```

```
##                          Size(sqf)                              Floor
##                                  0                                  0
##                        HallwayType                        HeatingType
##                                  0                                  0
##                      AptManageType                N_Parkinglot(Ground)
##                                  0                                  0
##             N_Parkinglot(Basement)                       TimeToBusStop
##                                  0                                  0
##                       TimeToSubway                              N_APT
##                                  0                                  0
##                          N_manager                         N_elevators
##                                  0                                  0
##                       SubwayStation   N_FacilitiesNearBy(PublicOffice)
##                                  0                                  0
##       N_FacilitiesNearBy(Hospital) N_FacilitiesNearBy(Dpartmentstore)
##                                  0                                  0
##           N_FacilitiesNearBy(Mall)            N_FacilitiesNearBy(ETC)
##                                  0                                  0
##           N_FacilitiesNearBy(Park)         N_SchoolNearBy(Elementary)
##                                  0                                  0
##             N_SchoolNearBy(Middle)               N_SchoolNearBy(High)
##                                  0                                  0
##         N_SchoolNearBy(University)                  N_FacilitiesInApt
##                                  0                                  0
##          N_FacilitiesNearBy(Total)              N_SchoolNearBy(Total)
##                                  0                                  0
```

En este caso, no se encontraron valores faltantes en ninguna de las columnas del conjunto de datos. Esto significa que no es necesario realizar técnicas de imputación de valores faltantes para este conjunto de datos específico.

Relevancia para el Modelo de Machine Learning

La ausencia de valores faltantes simplifica el preprocesamiento de datos, permitiendo un enfoque directo en la normalización y transformación de las variables según sea necesario para el modelo de k-NN. Esta verificación cumple con uno de los pasos iniciales de la preparación de datos, asegurando que todas las observaciones estén completas y listas para el análisis y modelado.

Este análisis asegura que no habrá interrupciones o errores debido a datos incompletos en los pasos subsiguientes del preprocesamiento y entrenamiento del modelo de machine learning.

La celda de abajo proporciona un resumen estadístico de las variables numéricas seleccionadas del conjunto de datos. El resumen incluye medidas de tendencia central (media, mediana) y medidas de dispersión (mínimo, máximo, cuartiles).

```
# veo un resumen de las variables numericas
# como la media, mediana (Q2), minimo, maximo, Q1, Q3
summary(base_r[c("SalePrice", "YearBuilt", "Size(sqf)")])
```

```
##    SalePrice        YearBuilt      Size(sqf)
##  Min.   : 32743   Min.   :1978   Min.   : 135.0
##  1st Qu.:144247   1st Qu.:1993   1st Qu.: 644.0
##  Median :207964   Median :2006   Median : 910.0
##  Mean   :221218   Mean   :2003   Mean   : 955.6
##  3rd Qu.:291150   3rd Qu.:2008   3rd Qu.:1149.0
##  Max.   :585840   Max.   :2015   Max.   :2337.0
```

`SalePrice (Precio de Venta)`: La media y la mediana están relativamente cerca, lo que indica una

distribución simétrica de los precios de venta, aunque el rango es amplio, sugiriendo variabilidad significativa en los precios de los apartamentos.

`YearBuilt (Año de Construcción)`: La media y la mediana son bastante cercanas, indicando que la mayoría de los apartamentos fueron construidos en las últimas dos décadas, con un rango de construcción que va desde 1978 hasta 2015.

`Size(sqf) (Tamaño en pies cuadrados)`: Los tamaños de los apartamentos varían significativamente, con la mayoría de los apartamentos entre 644 y 1149 pies cuadrados, y algunos apartamentos mucho más grandes alcanzando hasta 2337 pies cuadrados.

La siguiente celda selecciona las columnas numéricas de interés y calcula el rango intercuartílico (IQR) para cada una de ellas. El IQR es una medida de dispersión que indica el rango entre el primer cuartil (Q1) y el tercer cuartil (Q3), y es útil para identificar la variabilidad y detectar posibles outliers en los datos.

```
# selecciono las columnas numericas que me interesan
# luego calculo el rango intercuartilico de cada una
col_interes_num2 <- c("SalePrice", "YearBuilt", "Size(sqf)")
valores_iqr2 <- sapply(base_r[col_interes_num2], IQR)
valores_iqr2
```

```
## SalePrice YearBuilt Size(sqf)
##    146903        15       505
```

IQR SalePrice (Precio de Venta): Este valor indica que el rango intercuartílico para los precios de venta es bastante amplio, reflejando una variabilidad significativa en los precios de los apartamentos.

IQR YearBuilt (Año de Construcción): El rango intercuartílico de 15 años muestra una dispersión moderada en los años de construcción, lo que sugiere que los apartamentos en el conjunto de datos fueron construidos en diferentes periodos, pero con una concentración notable dentro de un rango de 15 años.

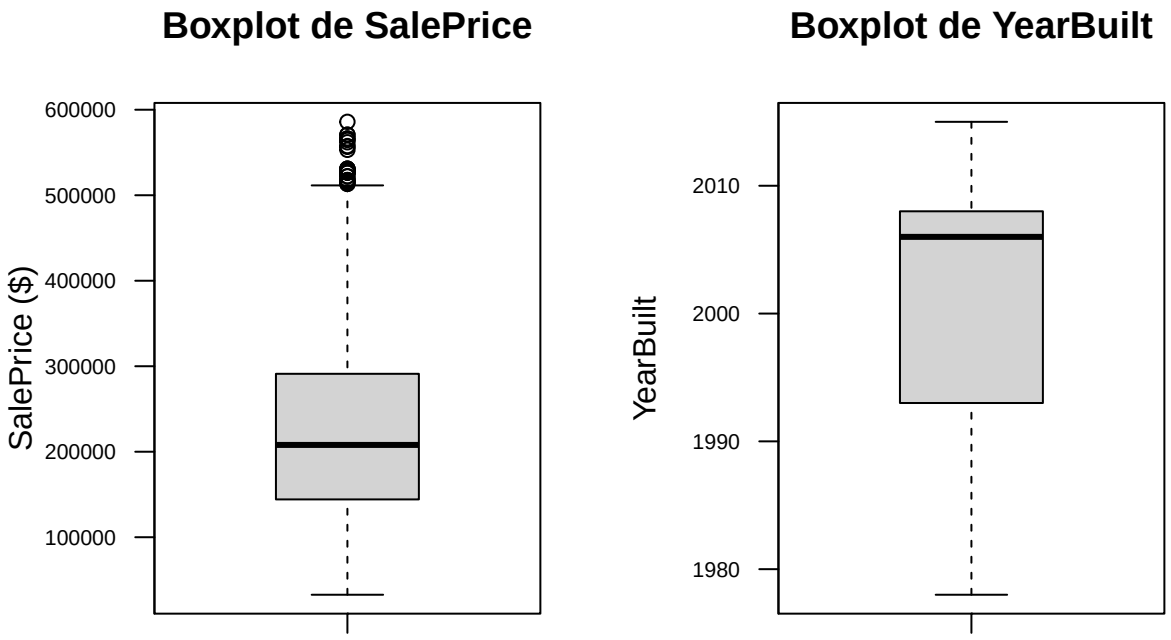
IQR Size(sqf) (Tamaño en pies cuadrados): El IQR de 505 pies cuadrados indica una considerable variabilidad en el tamaño de los apartamentos, con la mayoría de los apartamentos cayendo dentro de este rango.

El cálculo del IQR es importante para entender la dispersión de los datos y detectar posibles outliers que pueden influir en el modelado. Estos valores serán útiles para el preprocesamiento de datos y el ajuste de los modelos de machine learning, ayudando a mejorar la precisión y la robustez del modelo.

La siguiente celda crea boxplots para visualizar la distribución de dos variables numéricas de interés: `SalePrice` y `YearBuilt`. Los boxplots son útiles para identificar la distribución de los datos, la presencia de outliers y la dispersión de las observaciones.

```
# Ajustar el tamaño de la ventana gráfica
options(repr.plot.width = 10, repr.plot.height = 5)

# Dividir el área de gráficos en una disposición de 1 fila por 2 columnas
par(mfrow = c(1, 2))

# Crear el primer boxplot para SalePrice
boxplot(base_r$SalePrice,
        main = "Boxplot de SalePrice",
        ylab = "SalePrice ($)",
        xlab = "",
        outline = TRUE,
        axes = FALSE)  # Ocultar ejes para personalizarlos después

# Añadir de nuevo los ejes
axis(2, las = 1, cex.axis = 0.7,
     at = pretty(base_r$SalePrice),
```

```
     labels = format(pretty(base_r$SalePrice), scientific = FALSE))
axis(1, at = 1, labels = "")
box()  # Añadir el cuadro alrededor del gráfico

# Crear el segundo boxplot para YearBuilt
boxplot(base_r$YearBuilt,
        main = "Boxplot de YearBuilt",
        ylab = "YearBuilt",
        xlab = "",
        outline = TRUE,
        axes = FALSE)  # Ocultar ejes para personalizarlos después

# Añadir de nuevo los ejes
axis(2, las = 1, cex.axis = 0.7,
     at = pretty(base_r$YearBuilt),
     labels = format(pretty(base_r$YearBuilt), scientific = FALSE))
axis(1, at = 1, labels = "")
box()  # Añadir el cuadro alrededor del gráfico
```



Boxplot de SalePrice (Precio de Venta):

Distribución: El boxplot muestra que los precios de venta de los apartamentos tienen una distribución con una mediana alrededor de un poco superior a los 200.000, que por información anterior ya sabemos que es de $207,964. Outliers: Hay varios outliers lo que indica la presencia de algunos apartamentos con precios significativamente más altos que la mayoría.

Boxplot de YearBuilt (Año de Construcción):

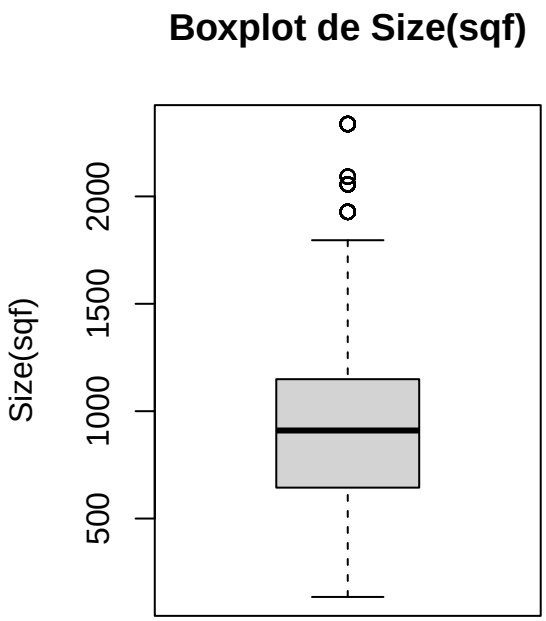
- Distribución: La mayoría de los apartamentos fueron construidos entre 1993 y 2008, con una mediana alrededor del año 2006.
- Outliers: No se observan outliers en los años de construcción

Los boxplots son herramientas visuales importantes para entender la distribución de los datos y detectar outliers que pueden afectar el rendimiento de los modelos de machine learning. En este caso, los boxplots de SalePrice y YearBuilt ayudan a identificar la variabilidad y posibles anomalías en los datos, lo que es crucial

para el preprocesamiento y análisis descriptivo de los datos

La siguiente celda crea un boxplot para visualizar la distribución de la variable numérica `Size(sqf)`.

```
#grafico Size(sqf)
# Crear un boxplot con el estilo básico de R
par(mfrow = c(1, 2))
boxplot(base_r$`Size(sqf)`,
        main = "Boxplot de Size(sqf)",
        ylab = "Size(sqf)",
        xlab = "",
        outline = TRUE)  # Para mantener los outliers
```



Boxplot de Size(sqf) (Tamaño en pies cuadrados):

- Distribución: El boxplot muestra que los tamaños de los apartamentos tienen una distribución con una mediana alrededor de los 900 pies cuadrados.
- Outliers: Hay varios outliers, lo que indica la presencia de algunos apartamentos con tamaños significativamente más grandes que la mayoría.

La siguiente celda calcula el rango intercuartílico (IQR) de la variable `SalePrice` para identificar los outliers. El IQR es una medida de la dispersión de los datos y se utiliza para detectar valores atípicos que están significativamente alejados del rango intercuartílico. Y luego identifico los outliers y los cuento.

```
# Cálculo del rango intercuartílico de SalePrice
q1 <- quantile(base_r$SalePrice, 0.25)
q3 <- quantile(base_r$SalePrice, 0.75)
iqr <- q3 - q1
lim_inf <- q1 - 1.5 * iqr
lim_sup <- q3 + 1.5 * iqr
# Identificación de los outliers
outliers <- base_r %>%
  filter(SalePrice < lim_inf | SalePrice > lim_sup)
#numero de outliers
nrow(outliers)
```

```
## [1] 34
```

- Observaciones en SalePrice menores a lim_inf o mayores a lim_sup son consideradas outliers.
- En este caso, se identificaron 34 outliers en la variable SalePrice.

La siguiente celda calcula el rango intercuartílico (IQR) de la variable `Size(sqf)` para identificar los outliers. El IQR es una medida de la dispersión de los datos y se utiliza para detectar valores atípicos que están significativamente alejados del rango intercuartílico.

```
# Cálculo del rango intercuartílico de size
q1_s <- quantile(base_r[["Size(sqf)"]], 0.25)
q3_s <- quantile(base_r[["Size(sqf)"]], 0.75)
iqr_s <- q3_s - q1_s
lim_inf_s <- q1_s - 1.5 * iqr_s
lim_sup_s <- q3_s + 1.5 * iqr_s
# Identificación de los outliers
outliers_s <- base_r %>%
  filter(`Size(sqf)` < lim_inf_s | `Size(sqf)` > lim_sup_s)
#numero de outliers
nrow(outliers_s)
```

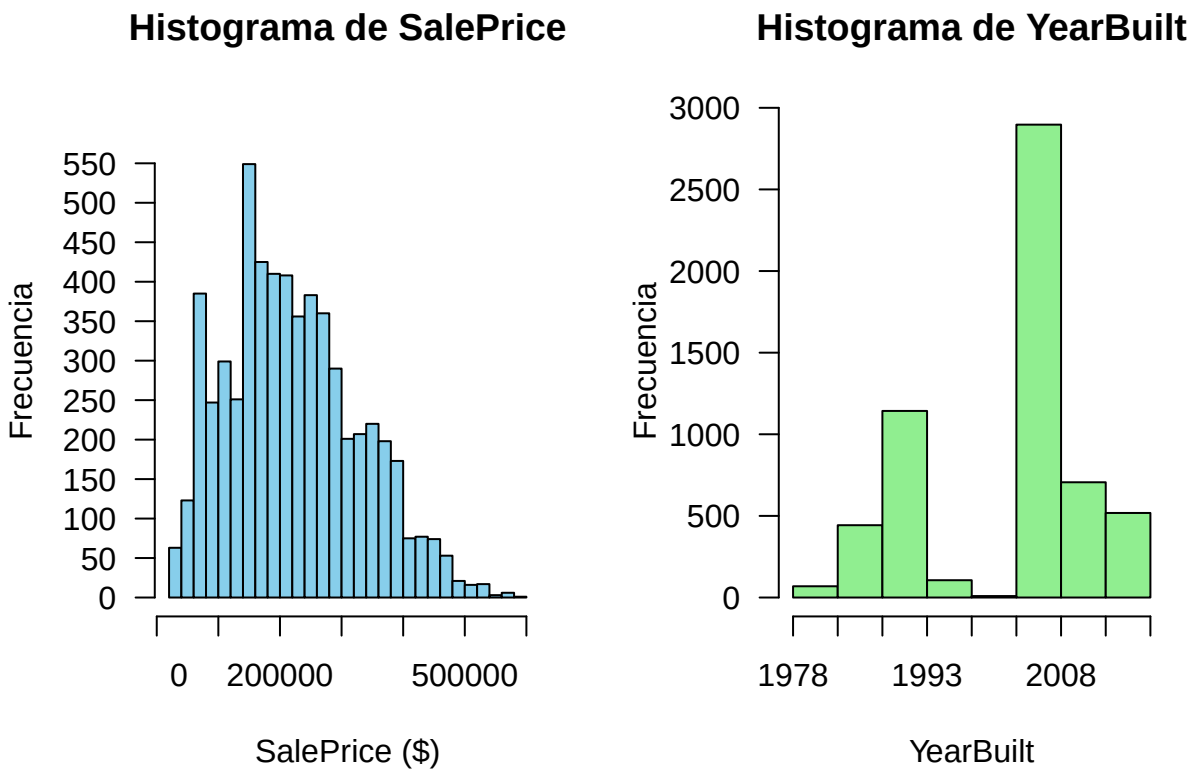
```
## [1] 139
```

- Observaciones en Size(sqf) menores a lim_inf_s o mayores a lim_sup_s son consideradas outliers.
- En este caso, se identificaron 139 outliers en la variable Size(sqf).

La proxima 2 celdas tienen como objetivo crear histogramas de las variables `SalePrice`, `YearBuilt` y `Size(sqf)` para visualizar la distribución de los datos. La visualización gráfica es una parte fundamental del análisis descriptivo y ayuda a identificar patrones y tendencias en los datos.

```
# Ajustar el tamaño de la ventana gráfica
options(repr.plot.width = 10, repr.plot.height = 5)

# Dividir el área de gráficos en una disposición de 1 fila por 2 columnas
par(mfrow = c(1, 2))

# Obtener las frecuencias del histograma de SalePrice sin dibujarlo
hist_info <- hist(base_r$SalePrice, breaks = 30, plot = FALSE)

# Calcular la frecuencia máxima
max_freq <- max(hist_info$counts)

# Crear el histograma para SalePrice con el ajuste del eje y
hist(base_r$SalePrice,
     main = "Histograma de SalePrice",
     xlab = "SalePrice ($)",
     ylab = "Frecuencia",
     col = "skyblue",
     breaks = 30,
     ylim = c(0, max_freq + 50),  # Ajustar el rango del eje y
     axes = FALSE)  # Ocultar ejes para personalizarlos después

# Añadir los ejes personalizados
axis(1, at = pretty(base_r$SalePrice),
     labels = format(pretty(base_r$SalePrice), scientific = FALSE))
axis(2, las = 1, at = seq(0, max_freq + 50, by = 50))

# Crear el segundo histograma para YearBuilt con intervalos ajustados
max_year <- max(base_r$YearBuilt)
```

```r
breaks_year_built <- seq(floor(min(base_r$YearBuilt)),
                         max_year + (5 - max_year %% 5), by = 5)
hist(base_r$YearBuilt,
     main = "Histograma de YearBuilt",
     xlab = "YearBuilt",
     ylab = "Frecuencia",
     col = "lightgreen",
     # Intervalos ajustados para cubrir todo el rango
     breaks = breaks_year_built,
     # Ocultar ejes para personalizarlos después
     axes = FALSE)

# Añadir los ejes personalizados
axis(1, at = seq(floor(min(base_r$YearBuilt)),
                 max_year + (5 - max_year %% 5), by = 5),
     labels = seq(floor(min(base_r$YearBuilt)),
                  max_year + (5 - max_year %% 5), by = 5))
axis(2, las = 1)
```



Histograma de SalePrice

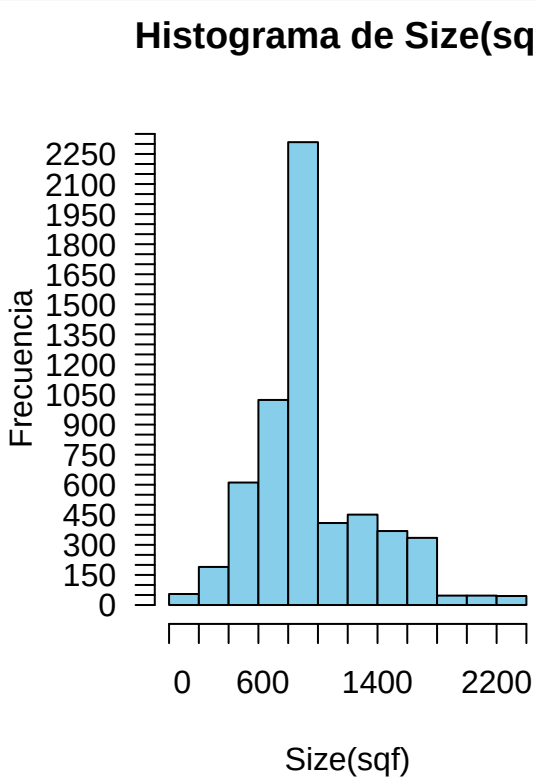
- Distribución: La variable SalePrice muestra una distribución sesgada a la derecha, con la mayoría de los valores concentrados en el rango de $50,000 a $300,000.
- Outliers: Se observa un pequeño número de valores que superan los $500,000, lo que indica la presencia de outliers en los datos de precios de venta.

Histograma de YearBuilt

- Distribución: La variable YearBuilt presenta una distribución bimodal, con puntos altos significativos en los años 1988-1993 y entre 2003-2008, lo que sugiere que hubo dos periodos de construcción intensiva.

- Rango: Los años de construcción varían entre 1978 y 2015 (de acuerdo a información revisada anteriormente), con una mayor concentración de apartamentos construidos alrededor del año 2003-2008.

```
#grafico histograma Size(sqf)
par(mfrow = c(1, 2))
# Obtener las frecuencias del histograma de Size(sqf) sin dibujarlo
hist_info <- hist(base_r$`Size(sqf)`, breaks = 10, plot = FALSE)

# Calcular la frecuencia máxima
max_freq <- max(hist_info$counts)

# Crear el histograma para Size(sqf) con el ajuste del eje y
hist(base_r$`Size(sqf)`,
     main = "Histograma de Size(sqf)",
     xlab = "Size(sqf)",
     ylab = "Frecuencia",
     col = "skyblue",
     breaks = 10,
     ylim = c(0, max_freq + 50),  # Ajustar el rango del eje y
     axes = FALSE)  # Ocultar ejes para personalizarlos después

# Añadir los ejes personalizados
axis(1, at = hist_info$breaks, labels =
       format(hist_info$breaks, scientific = FALSE))
axis(2, las = 1, at = seq(0, max_freq + 50, by = 50))
```



Histograma de Size(sqf)

- Distribución: La variable Size(sqf) muestra una distribución con una fuerte concentración alrededor de los 800 a 1000 sqf, indicando que la mayoría de los apartamentos tienen tamaños similares.
- Outliers: Existen algunos valores superiores a 1800 sqf, lo que sugiere la presencia de algunos aparta-

mentos significativamente más grandes que la mayoría.

La siguiente celda tiene como objetivo calcular la varianza y la desviación estándar de la variable `SalePrice`. Estas métricas son fundamentales para entender la dispersión y la variabilidad de los precios de los apartamentos en el conjunto de datos.

```
#varianza y desviacion estandar de SalePrice
var_sale_price <- var(base_r$SalePrice)
sd_sale_price <- sd(base_r$SalePrice)
var_sale_price
```

```
## [1] 11317595126
```

```
sd_sale_price
```

```
## [1] 106384.2
```

Varianza de SalePrice

- Interpretación: La varianza es una medida de la dispersión de los valores de SalePrice respecto a la media. Dado que el valor de varianza es bastante alto, esto indica que los precios de los apartamentos tienen una amplia dispersión alrededor de la media. Esto puede deberse a la presencia de apartamentos con precios significativamente más altos o más bajos que el promedio.

Desviación Estándar de SalePrice

- Interpretación: La desviación estándar es la raíz cuadrada de la varianza y proporciona una medida de dispersión en las mismas unidades que SalePrice. Un valor de desviación estándar de aproximadamente 106,384.19 indica que los precios de los apartamentos tienden a variar en promedio $\pm$106,384.19 dólares respecto a la media. Este valor también sugiere una considerable variabilidad en los precios de los apartamentos.

Relevancia para el Proyecto

- Calcular la varianza y la desviación estándar es crucial para entender la variabilidad inherente en la variable objetivo SalePrice. Este análisis permite identificar la extensión de la dispersión de los precios y ayuda a justificar el tratamiento necesario de las variables numéricas antes de aplicar el algoritmo k-NN. La comprensión de estas métricas también es esencial para evaluar la adecuación del modelo de k-NN y su desempeño en la predicción de SalePrice.

La siguiente celda tiene como objetivo calcular la varianza y la desviación estándar de la variable `YearBuilt`. Estas métricas son esenciales para comprender la dispersión y la variabilidad de los años de construcción de los apartamentos en el conjunto de datos.

```
#varianza y desviacion estandar de YearBuilt
var_year_built <- var(base_r$YearBuilt)
sd_year_built <- sd(base_r$YearBuilt)
var_year_built
```

```
## [1] 77.64749
```

```
sd_year_built
```

```
## [1] 8.811782
```

Varianza de YearBuilt

- Interpretación: La varianza es una medida de la dispersión de los valores de YearBuilt respecto a la media. Un valor de varianza de 77.65 indica que los años de construcción de los apartamentos tienen una dispersión moderada alrededor de la media. Esto sugiere que los apartamentos en el conjunto de datos fueron construidos en años relativamente cercanos entre sí, aunque hay cierta variabilidad.

Desviación Estándar de YearBuilt

- Interpretación: La desviación estándar es la raíz cuadrada de la varianza y proporciona una medida de dispersión. Un valor de desviación estándar de aproximadamente 8.81 años indica que los años de construcción de los apartamentos tienden a variar en promedio ±8.81 años respecto a la media. Este valor sugiere una variabilidad moderada en los años de construcción de los apartamentos.

La próxima celda tiene como objetivo calcular la varianza y la desviación estándar de la variable `Size(sqf)`. Estas métricas son esenciales para comprender la dispersión y la variabilidad de los tamaños de los apartamentos en el conjunto de datos.

```
#varianza y desviacion estandar de Size(sqf)
var_size <- var(base_r$`Size(sqf)`)
sd_size <- sd(base_r$`Size(sqf)`)
var_size
```

```
## [1] 146278.7
```

```
sd_size
```

```
## [1] 382.4641
```

Varianza de Size(sqf)

- Interpretación: La varianza es una medida de la dispersión de los valores de Size(sqf) respecto a la media. Un valor de varianza de 146278.75 indica que los tamaños de los apartamentos tienen una dispersión considerable alrededor de la media. Esto sugiere que los tamaños de los apartamentos en el conjunto de datos varían ampliamente.

Desviación Estándar de Size(sqf)

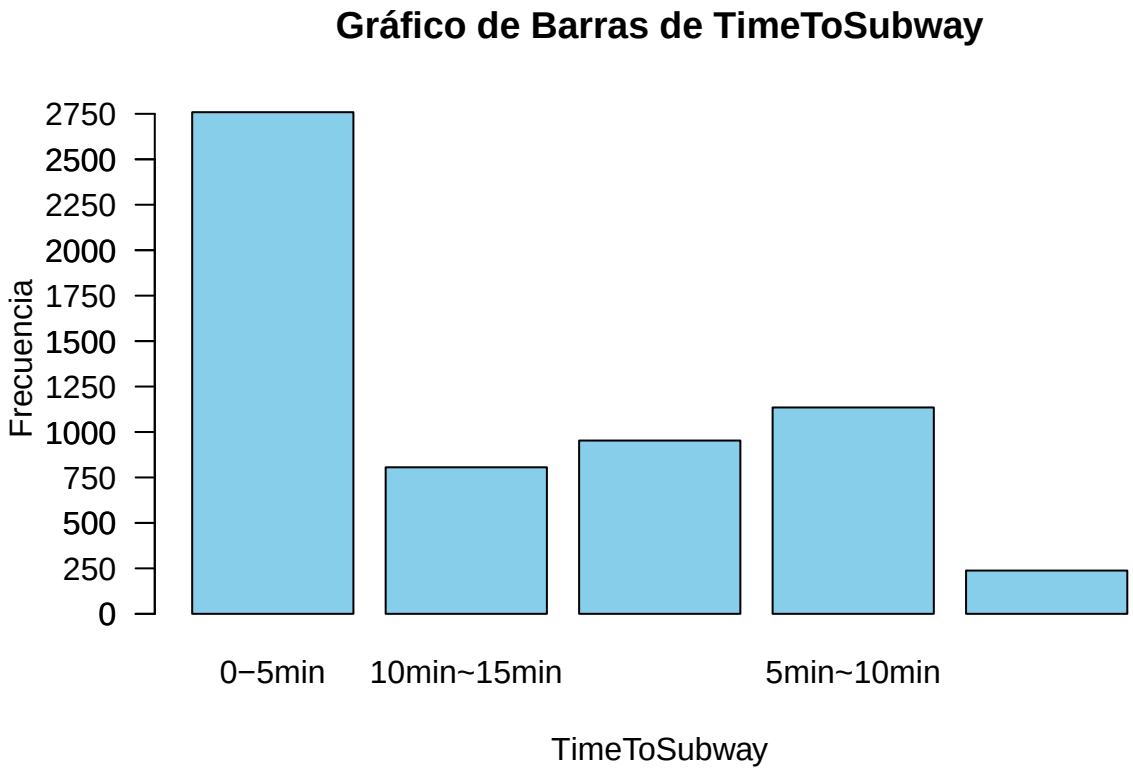
- Interpretación: La desviación estándar es la raíz cuadrada de la varianza y proporciona una medida de dispersión en las mismas unidades que Size(sqf). Un valor de desviación estándar de aproximadamente 382.46 pies cuadrados indica que los tamaños de los apartamentos tienden a variar en promedio ±382.46 pies cuadrados respecto a la media. Este valor sugiere una variabilidad significativa en los tamaños de los apartamentos.

Analisis Univariado Variables Categoricas Seleccionadas

Considero que las variables categoricas más importantes son 2: "TimeToSubway" y "HallwayType"

La siguiente celda tiene como objetivo crear un gráfico de barras para la variable categórica `TimeToSubway`. Esta visualización ayuda a comprender la distribución de los tiempos que toman los residentes para llegar a la estación de metro más cercana. El análisis de esta variable categórica es esencial para evaluar su influencia en el precio de las viviendas y es parte de la estadística descriptiva requerida en el trabajo.

```
# Crear la tabla de frecuencias para TimeToSubway
freq_table <- table(base_r$TimeToSubway)

# Crear el gráfico de barras
barplot(freq_table,
        main = "Gráfico de Barras de TimeToSubway",
        xlab = "TimeToSubway",
        ylab = "Frecuencia",
        col = "skyblue",
        ylim = c(0, max(freq_table) + 50),  # Ajustar el rango del eje y
        las = 1)  # Hacer que las etiquetas del eje y sean horizontales

# Añadir más etiquetas en el eje y
y_ticks <- seq(0, max(freq_table) + 50, by = 250)
axis(2, at = y_ticks, labels = y_ticks, las = 1)
```



Frecuencias de TimeToSubway

- `0-5 min`: Aproximadamente 2750 observaciones indican que la mayoría de los apartamentos están a menos de 5 minutos de una estación de metro.
- `10-15 min`: Menos de 1000 observaciones caen en este rango de tiempo.
- `15-20 min`: Aproximadamente 950 observaciones indican que algunos apartamentos están a entre 15 y 20 minutos de una estación de metro.
- `5-10 min`: Aproximadamente 1150 observaciones indican que muchos apartamentos están a entre 5 y 10 minutos de una estación de metro.
- `No bus stop nearby`: Alrededor de 250 observaciones muestran que pocos apartamentos están lejos de una estación de bus.

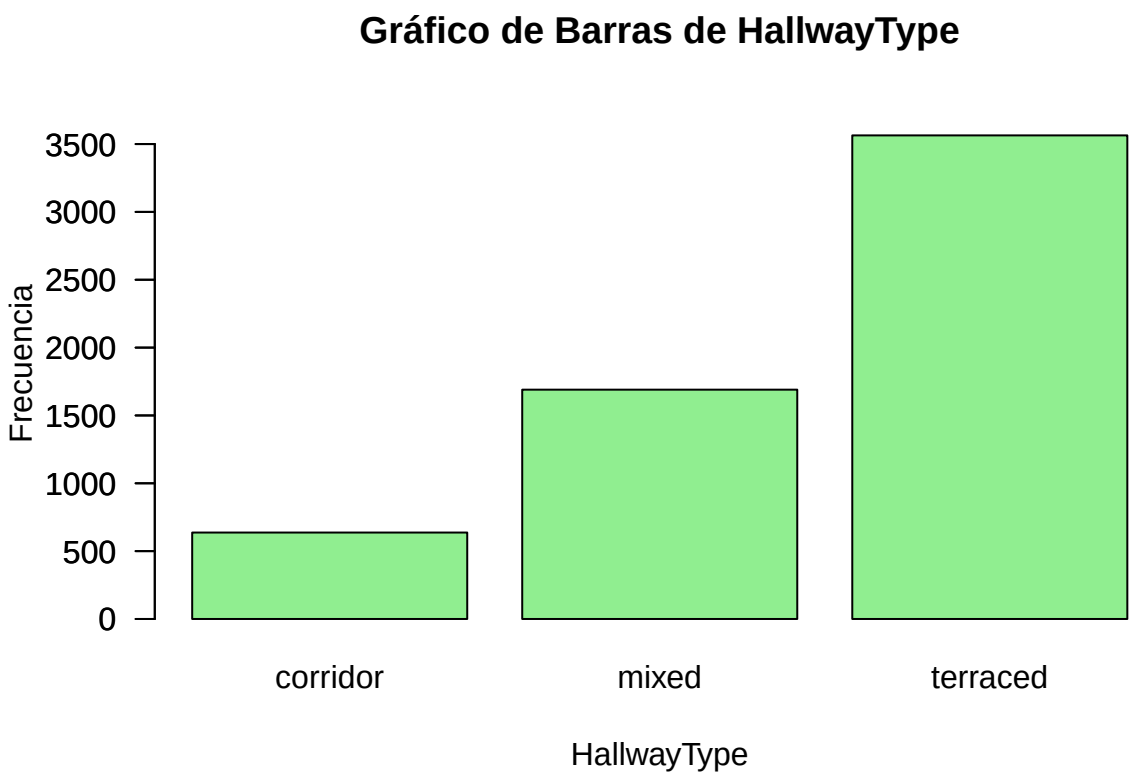
Interpretación

- Distribución: La mayor parte de los apartamentos está muy cerca de una estación de metro, lo cual podría ser un factor significativo en el valor de SalePrice.
- Influencia Potencial: Dado que la mayoría de los apartamentos están a menos de 5 minutos de una estación de metro, esta variable podría tener un impacto positivo significativo en el precio de las viviendas.

La siguiente celda tiene como objetivo crear un gráfico de barras para la variable categórica `HallwayType`. Esta visualización ayuda a comprender la distribución de los diferentes tipos de pasillos en los apartamentos. El análisis de esta variable categórica es esencial para evaluar su influencia en el precio de las viviendas.

```
# Crear la tabla de frecuencias para HallwayType
freq_table_hallway <- table(base_r$HallwayType)

# Crear el gráfico de barras
barplot(freq_table_hallway,
        main = "Gráfico de Barras de HallwayType",
        xlab = "HallwayType",
        ylab = "Frecuencia",
```

```
        col = "lightgreen",
        # Ajustar el rango del eje y
        ylim = c(0, max(freq_table_hallway) + 200),
        las = 1)  # Hacer que las etiquetas del eje y sean horizontales

# Añadir más etiquetas en el eje y
y_ticks_hallway <- seq(0, max(freq_table_hallway) + 200, by = 500)
axis(2, at = y_ticks_hallway, labels = y_ticks_hallway, las = 1)
```



Frecuencias de HallwayType

- Corridor: Aproximadamente 650 observaciones indican que un número pequeño de apartamentos tiene pasillos tipo corredor.
- Mixed: Aproximadamente 1700 observaciones muestran que un número moderado de apartamentos tiene una combinación de tipos de pasillos.
- Terraced: 3500 observaciones indican que la mayoría de los apartamentos tienen pasillos tipo terraza.

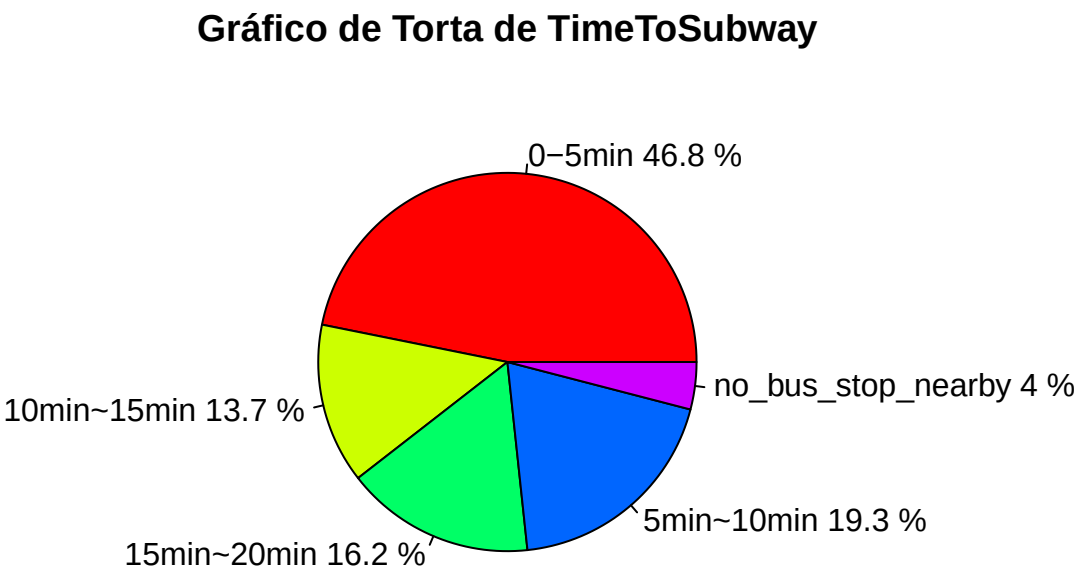
Interpretación

- Distribución: La mayoría de los apartamentos tienen pasillos tipo terraza, lo cual podría ser un factor significativo en el valor de SalePrice.
- Influencia Potencial: Dado que la mayoría de los apartamentos tienen pasillos tipo terraza, esta variable podría tener un impacto positivo significativo en el precio de las viviendas.

La próxima celda tiene como objetivo crear un gráfico de torta para la variable categórica `TimeToSubway`. Esta visualización permite observar la distribución porcentual de los tiempos que toman los residentes para llegar a la estación de metro más cercana. El análisis de esta variable categórica es fundamental para evaluar su influencia en el precio de las viviendas.

```
# Creo la tabla de frecuencias para TimeToSubway
freq_table_timesubway <- table(base_r$TimeToSubway)
```

```r
# Calculo los porcentajes
percentages <- round(
                    100 * freq_table_timesubway / sum(freq_table_timesubway)
                    , 1)

# Creo etiquetas con los porcentajes
labels <- paste(names(freq_table_timesubway), percentages, "%", sep = " ")

# Creo el gráfico de torta
pie(freq_table_timesubway,
    labels = labels,
    main = "Gráfico de Torta de TimeToSubway",
    col = rainbow(length(freq_table_timesubway)))
```



Distribución de TimeToSubway

- 0-5 min: 46.8% de los apartamentos están a menos de 5 minutos de una estación de metro, lo que representa casi la mitad de las observaciones.
- 5-10 min: 19.3% de los apartamentos están a 5-10 minutos de una estación de metro.
- 10-15 min: 13.7% de los apartamentos están a 10-15 minutos de una estación de metro.
- 15-20 min: 16.2% de los apartamentos están a 15-20 minutos de una estación de metro. No bus stop nearby: Solo el 4% de los apartamentos no tiene una estación de bus cercana.

Interpretación

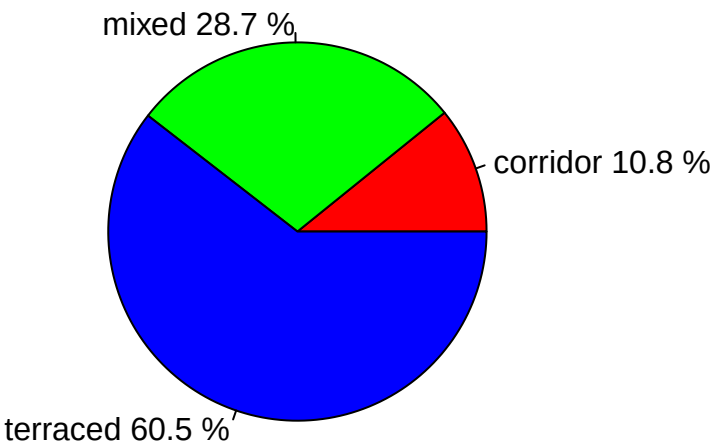
- Accesibilidad: La mayoría de los apartamentos tienen una estación de metro a menos de 10 minutos, lo que podría ser un factor importante para la conveniencia y el valor de las propiedades.
- Influencia Potencial: La proximidad a una estación de metro generalmente aumenta el valor de las propiedades debido a la facilidad de transporte.

El gráfico de torta proporciona una visión clara de la distribución de los tiempos de acceso al metro, lo que es crucial para la descripción de las variables categóricas. Este análisis gráfico cumple con los objetivos del proyecto al proporcionar una base sólida para entender cómo las variables categóricas pueden influir en la variable objetivo (SalePrice).

La siguiente celda tiene como objetivo crear un gráfico de torta para la variable categórica `HallwayType`. Esta visualización permite observar la distribución porcentual de los diferentes tipos de pasillos en los apartamentos de Daegu. El análisis de esta variable categórica es esencial para evaluar su influencia en el precio de las

viviendas.

```r
# Crear la tabla de frecuencias para HallwayType
freq_table_hallway <- table(base_r$HallwayType)

# Calcular los porcentajes
percentages_hallway <- round(100 * freq_table_hallway
                             / sum(freq_table_hallway), 1)

# Crear etiquetas con los porcentajes
labels_hallway <- paste(names(freq_table_hallway),
                        percentages_hallway, "%", sep = " ")

# Crear el gráfico de torta
pie(freq_table_hallway,
    labels = labels_hallway,
    main = "Gráfico de Torta de HallwayType",
    col = rainbow(length(freq_table_hallway)))
```



Distribución de HallwayType

- corridor: 10.8% de los apartamentos tienen pasillos tipo corredor.
- mixed: 28.7% de los apartamentos tienen pasillos mixtos.
- terraced: 60.5% de los apartamentos tienen pasillos tipo terraza, representando la mayoría de las observaciones.

Interpretación

- Predominio de Terraza: La mayoría de los apartamentos tienen pasillos tipo terraza, lo que podría indicar una preferencia por este tipo de diseño en los edificios residenciales de Daegu.
- Impacto en el Valor de la Propiedad: El tipo de pasillo puede influir en la percepción y el valor de la propiedad, ya que ciertos diseños pueden ser más atractivos o funcionales.

La siguiente celda tiene como objetivo crear una tabla de frecuencias para la variable categórica `TimeToSubway`. La tabla de frecuencias permite observar la distribución de los tiempos de desplazamiento hasta la estación de metro. Este análisis también facilita la comprensión de cómo esta variable podría influir en la variable objetivo (`SalePrice`).

```r
## Creo la tabla de frecuencias para TimeToSubway
freq_table_timesubway <- table(base_r$TimeToSubway)
freq_table_timesubway
```

```
##
##             0-5min        10min~15min        15min~20min         5min~10min
##               2759                806                953               1135
## no_bus_stop_nearby
##                238
```

Análisis de Resultados

- `0-5min`: 2759 apartamentos (46.8%) están a menos de 5 minutos de una estación de metro.
- `5min~10min`: 1135 apartamentos (19.3%) están entre 5 y 10 minutos de una estación de metro.
- `10min~15min`: 806 apartamentos (13.7%) están entre 10 y 15 minutos de una estación de metro.
- `15min~20min`: 953 apartamentos (16.2%) están entre 15 y 20 minutos de una estación de metro.
- `no_bus_stop_nearby`: 238 apartamentos (4%) no tienen una estación de bus cercana.

Interpretación

- Accesibilidad a Transporte Público: La mayoría de los apartamentos están a una distancia corta (menos de 5 minutos) de una estación de metro, lo que indica una alta accesibilidad al transporte público en la ciudad de Daegu.
- Impacto en el Valor de la Propiedad: La proximidad a una estación de metro podría ser un factor significativo en la determinación del precio de la vivienda, ya que un mejor acceso al transporte público generalmente aumenta el valor de la propiedad.

La próxima celda tiene como objetivo crear una tabla de frecuencias para la variable categórica `HallwayType`. La tabla de frecuencias nos permite observar la distribución de los tipos de pasillos en los apartamentos. Este análisis facilita la comprensión de cómo esta variable podría influir en la variable objetivo (`SalePrice`).

```r
## Creo la tabla de frecuencias para HallwayType
freq_table_hallway <- table(base_r$HallwayType)
freq_table_hallway
```

```
##
## corridor    mixed terraced
##      637     1690     3564
```

Análisis de Resultados

- Corridor: 637 apartamentos (10.8%) tienen un pasillo de tipo corridor.
- Mixed: 1690 apartamentos (28.7%) tienen un pasillo de tipo mixed.
- Terraced: 3564 apartamentos (60.5%) tienen un pasillo de tipo terraced.

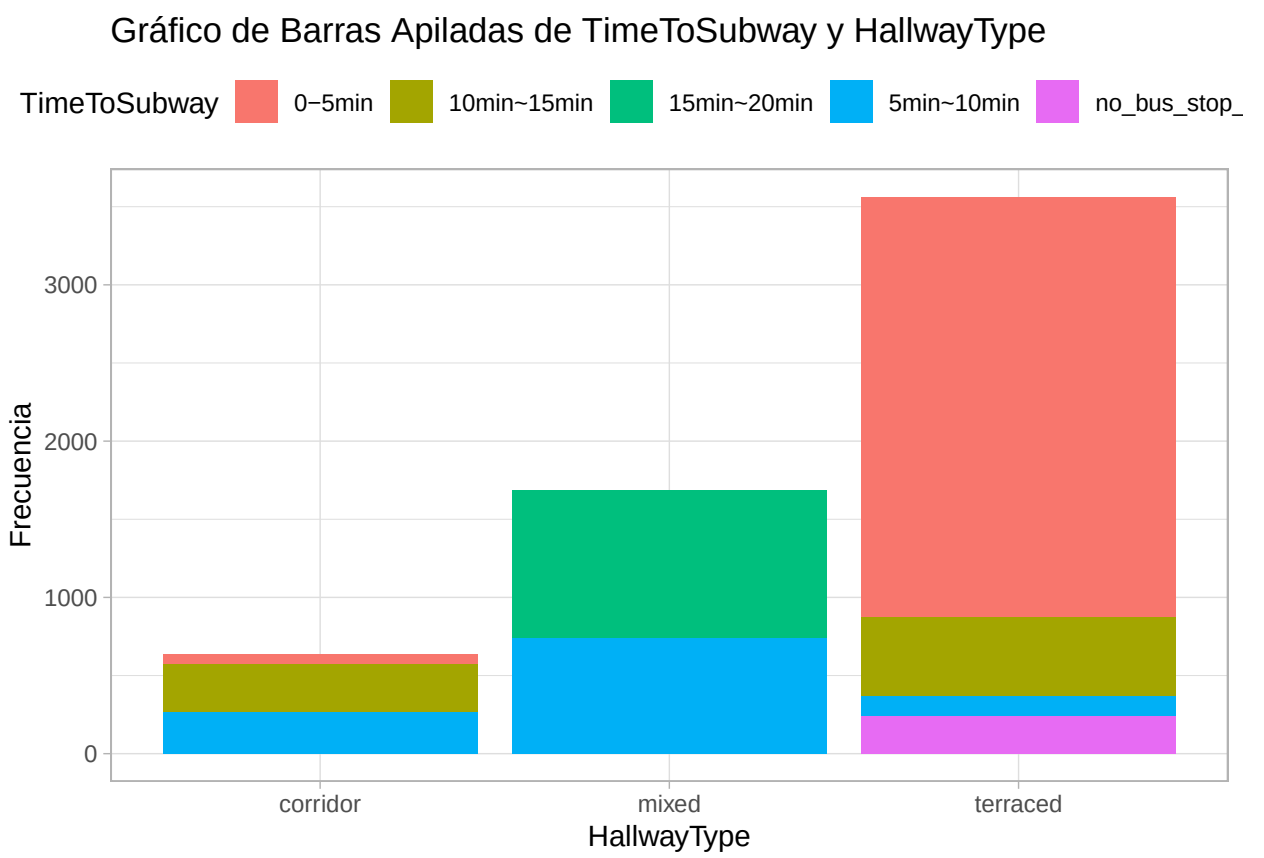
Interpretación

- Distribución de Tipos de Pasillos: La mayoría de los apartamentos tienen pasillos de tipo terraced, seguido por mixed y una menor cantidad tiene pasillos de tipo corridor.
- Impacto en el Valor de la Propiedad: El tipo de pasillo podría ser un factor significativo en la determinación del precio de la vivienda. Por ejemplo, los pasillos de tipo terraced pueden estar asociados con un diseño de apartamento más atractivo y, por lo tanto, un precio más alto.

La siguiente celda genera un gráfico de barras apiladas donde se muestra la distribución conjunta de las variables categóricas `TimeToSubway` y `HallwayType`. Es particularmente útil para entender cómo se relacionan estas dos variables y cómo pueden influir conjuntamente en la variable objetivo `SalePrice`.

```r
# Creo el gráfico de barras apiladas usando ggplot2
ggplot(base_r, aes(x = factor(HallwayType), fill = factor(TimeToSubway))) +
  geom_bar(position = "stack") +
```

```
  labs(title = "Gráfico de Barras Apiladas de TimeToSubway y HallwayType",
       x = "HallwayType",
       y = "Frecuencia",
       fill = "TimeToSubway") +
  theme_light() +
  theme(legend.position = "top")  # Mueve la leyenda a la parte superior
```



Análisis de Resultados

- Corridor: La mayoría de los apartamentos con pasillos tipo corridor tienen un tiempo de acceso al metro de 5 a 10 minutos.
- Mixed: Los apartamentos con pasillos tipo mixed presentan una distribución más equilibrada en términos de tiempo de acceso al metro, con un mayor número en la categoría de 5 a 10 minutos y 15 a 20 minutos.
- Terraced: La mayoría de los apartamentos con pasillos tipo terraced tienen un tiempo de acceso al metro de 0 a 5 minutos.

Interpretación

- Relación entre HallwayType y TimeToSubway: Los apartamentos con pasillos tipo terraced tienden a tener un acceso más rápido al metro (0-5 minutos), lo que podría influir en el valor de la propiedad (SalePrice). Esto sugiere que tanto el tipo de pasillo como el tiempo de acceso al metro son importantes para determinar el precio de venta de un apartamento.
- Implicaciones para el Modelo k-NN: La variable HallwayType muestra una distribución significativa cuando se cruza con TimeToSubway, indicando que ambos atributos podrían ser relevantes al predecir SalePrice con el modelo k-NN.

Parte 2: Aprendizaje Supervisado Regresión, Analisis k-NN (Items 4-8)

```
# importacion de librerias
library(ggplot2)
library(dplyr)
library(readr)
library(caret)
library(rsample)
library(kknn)
```

Estas librerías son fundamentales para cumplir con los objetivos del trabajo, que incluyen desde la importación y manipulación de datos hasta la construcción, evaluación y visualización de modelos de machine learning.

La siguiente celda realiza la importación del conjunto de datos "Daegu_Real_Estate_v2data.csv" y se convierte en un data frame. Este conjunto de datos contiene información sobre transacciones de apartamentos en Daegu, Corea del Sur, entre agosto de 2007 y agosto de 2017.

```
base_daegu <- read_csv("E:/data science 2024/R begginers/Trabajo ML 1/Daegu_Real_Estate_data.csv")
```

```
## Rows: 5891 Columns: 30
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (6): HallwayType, HeatingType, AptManageType, TimeToBusStop, TimeToSubw...
## dbl (24): SalePrice, YearBuilt, YrSold, MonthSold, Size(sqf), Floor, N_Parki...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
base_daegu <- as.data.frame(base_daegu)
```

Resultado de la Importación

Filas: 5891 Columnas: 30

Las columnas incluyen una mezcla de variables categóricas (6) y numéricas (24). Algunas de las variables importantes son:

- `SalePrice`: Precio de venta de la vivienda (variable objetivo).
- `YearBuilt`: Año de construcción.
- `YrSold`: Año de venta.
- `MonthSold`: Mes de venta.
- `Size(sqf)`: Tamaño en pies cuadrados.
- `Floor`: Piso del apartamento.
- `N_Parkinglot(Ground)`, `N_Parkinglot(Basement)`: Número de estacionamientos en superficie y subterráneos.
- `TimeToBusStop`, `TimeToSubway`: Tiempo al paradero de bus y estación de metro.

Variables categóricas como `HallwayType`, `HeatingType`, y `AptManageType`.

En la siguiente celda se seleccionan las columnas categóricas del conjunto de datos y se calculan las frecuencias de las diferentes clases presentes en cada una de estas variables. Esto es crucial para entender la distribución de las categorías y cómo están representadas en el conjunto de datos.

```
#selecciono las columnas categoricas
col_cate <- c("HallwayType", "HeatingType", "AptManageType",
              "TimeToBusStop", "TimeToSubway", "SubwayStation")
valores_por_clase <- lapply(base_daegu[col_cate], table)
```

```
valores_por_clase
```

```
## $HallwayType
##
## corridor    mixed terraced
##      637     1690     3564
##
## $HeatingType
##
##    central_heating individual_heating
##                300               5591
##
## $AptManageType
##
## management_in_trust     self_management
##                5542                 349
##
## $TimeToBusStop
##
##      0~5min 10min~15min  5min~10min
##        4509          55        1327
##
## $TimeToSubway
##
##             0-5min        10min~15min        15min~20min         5min~10min
##               2759                806                953               1135
## no_bus_stop_nearby
##                238
##
## $SubwayStation
##
##               Bangoge            Banwoldang      Chil-sung-market
##                   737                   748                   115
##                 Daegu Kyungbuk_uni_hospital             Myung-duk
##                    85                  1644                  1507
##      no_subway_nearby               Sin-nam
##                   404                   651
```

La siguiente celda realiza la conversión de las variables categóricas a variables dummies, un paso esencial para el preprocesamiento de datos antes de aplicar el algoritmo k-NN de regresión.

```
# Defino el objeto con las variables categóricas a convertir
dv <- dummyVars("~ .", data = base_daegu[, col_cate], fullRank = TRUE)
# creo el diseño de la matriz
design_matrix <- data.frame(predict(dv, newdata = base_daegu[, col_cate]))
# veo el resultado
head(design_matrix)
```

```
##   HallwayTypemixed HallwayTypeterraced HeatingTypeindividual_heating
## 1                0                   1                              1
## 2                0                   0                              1
## 3                0                   0                              1
## 4                0                   1                              1
## 5                1                   0                              1
## 6                0                   0                              1
```

```
##   AptManageTypeself_management TimeToBusStop10min.15min TimeToBusStop5min.10min
## 1                            0                        0                       1
## 2                            1                        0                       0
## 3                            1                        0                       0
## 4                            0                        0                       0
## 5                            0                        0                       0
## 6                            0                        0                       1
##   TimeToSubway10min.15min TimeToSubway15min.20min TimeToSubway5min.10min
## 1                       1                       0                      0
## 2                       0                       0                      1
## 3                       0                       0                      1
## 4                       0                       0                      0
## 5                       0                       1                      0
## 6                       1                       0                      0
##   TimeToSubwayno_bus_stop_nearby SubwayStationBanwoldang
## 1                              0                       0
## 2                              0                       0
## 3                              0                       0
## 4                              0                       0
## 5                              0                       0
## 6                              0                       0
##   SubwayStationChil.sung.market SubwayStationDaegu
## 1                             0                  0
## 2                             0                  1
## 3                             0                  1
## 4                             0                  0
## 5                             0                  0
## 6                             0                  0
##   SubwayStationKyungbuk_uni_hospital SubwayStationMyung.duk
## 1                                  1                      0
## 2                                  0                      0
## 3                                  0                      0
## 4                                  0                      0
## 5                                  0                      1
## 6                                  0                      1
##   SubwayStationno_subway_nearby SubwayStationSin.nam
## 1                             0                   0
## 2                             0                   0
## 3                             0                   0
## 4                             0                   1
## 5                             0                   0
## 6                             0                   0
```

1. Definición del Objeto `dummyVars`:

- Se utiliza la función `dummyVars` del paquete `caret` para definir cómo se convertirán las variables categóricas.
- La fórmula `"~ ."` indica que todas las columnas de `base_daegu[, col_cate]` (osea solo las columnas categoricas) serán transformadas.
- El argumento `fullRank = TRUE` asegura que, para cada variable categórica con `N` clases, se generen `N-1` columnas dummy, evitando la multicolinealidad.

2. Creación de la Matriz de Diseño:

- Se usa la función `predict` para aplicar la transformación definida en `dv` a los datos categóricos originales.
- El resultado es una matriz de diseño (`design_matrix`) con las variables categóricas convertidas a

variables dummies.

3. Visualización del Resultado:

- `head(design_matrix)` muestra las primeras filas de la matriz de diseño resultante para verificar la conversión.

Propósito

La conversión de variables categóricas a variables dummies es necesaria porque los algoritmos de machine learning, como k-NN, no pueden manejar variables categóricas directamente. Las variables categóricas deben ser transformadas en un formato numérico adecuado para que el algoritmo pueda calcular las distancias correctamente.

Conversión de Variables Categóricas

Razonamiento:

- Las variables categóricas contienen clases que representan diferentes estados o categorías. Para que estas variables puedan ser utilizadas en el cálculo de distancias, deben ser convertidas en una representación numérica.
- La técnica de variables dummies asigna una columna binaria (0 o 1) para cada clase de la variable categórica, lo que permite representar la presencia o ausencia de una clase.

N Clases a N-1 Columnas:

- Si una variable categórica tiene N clases, se generan N-1 columnas dummies. Esto se hace para evitar la multicolinealidad, que ocurre cuando las variables predictoras están altamente correlacionadas entre sí.
- Por ejemplo, si una variable tiene tres clases (A, B, C), se crean dos columnas: una para A y otra para B. La clase C puede ser inferida cuando ambas columnas tienen el valor 0.
- Este enfoque garantiza que la información sobre la variable categórica esté completamente representada sin redundancias.

En la siguiente celda se realizan dos operaciones clave para preparar el conjunto de datos para el análisis posterior con el algoritmo k-NN, que son la eliminación de las columnas originales (las categoricas) y la union de la matriz de diseño con la base de datos original

```
# elimino las columnas originales
base_daegu <- base_daegu[, -which(names(base_daegu) %in% col_cate)]
# uno las columnas de la matriz de diseño
base_daegu <- cbind(base_daegu, design_matrix)
head(base_daegu)
```

```
##   SalePrice YearBuilt YrSold MonthSold Size(sqf) Floor N_Parkinglot(Ground)
## 1    141592      2006   2007         8       814     3                   111
## 2     51327      1985   2007         8       587     8                    80
## 3     48672      1985   2007         8       587     6                    80
## 4    380530      2006   2007         8      2056     8                   249
## 5    221238      1993   2007         8      1761     3                   523
## 6     35840      1992   2007         8       355     5                   200
##   N_Parkinglot(Basement) N_APT N_manager N_elevators
## 1                    184     3         3           0
## 2                     76     1         2           2
## 3                     76     1         2           2
## 4                    536     6         5          11
## 5                    536     8         8          20
## 6                      0     3         5          10
##   N_FacilitiesNearBy(PublicOffice) N_FacilitiesNearBy(Hospital)
## 1                                2                           1
## 2                                5                           1
```

```
## 3                               5                           1
## 4                               1                           1
## 5                               6                           2
## 6                               7                           1
##   N_FacilitiesNearBy(Dpartmentstore) N_FacilitiesNearBy(Mall)
## 1                                  1                       1
## 2                                  2                       1
## 3                                  2                       1
## 4                                  0                       1
## 5                                  0                       1
## 6                                  1                       1
##   N_FacilitiesNearBy(ETC) N_FacilitiesNearBy(Park) N_SchoolNearBy(Elementary)
## 1                       1                       0                         3
## 2                       2                       1                         2
## 3                       2                       1                         2
## 4                       0                       0                         2
## 5                       5                       0                         4
## 6                       5                       1                         4
##   N_SchoolNearBy(Middle) N_SchoolNearBy(High) N_SchoolNearBy(University)
## 1                      2                    2                         2
## 2                      1                    1                         0
## 3                      1                    1                         0
## 4                      2                    1                         2
## 5                      3                    5                         5
## 6                      3                    5                         5
##   N_FacilitiesInApt N_FacilitiesNearBy(Total) N_SchoolNearBy(Total)
## 1                 5                         6                    9
## 2                 3                        12                    4
## 3                 3                        12                    4
## 4                 5                         3                    7
## 5                 4                        14                   17
## 6                 3                        16                   17
##   HallwayTypemixed HallwayTypeterraced HeatingTypeindividual_heating
## 1                0                   1                             1
## 2                0                   0                             1
## 3                0                   0                             1
## 4                0                   1                             1
## 5                1                   0                             1
## 6                0                   0                             1
##   AptManageTypeself_management TimeToBusStop10min.15min TimeToBusStop5min.10min
## 1                            0                        0                       1
## 2                            1                        0                       0
## 3                            1                        0                       0
## 4                            0                        0                       0
## 5                            0                        0                       0
## 6                            0                        0                       1
##   TimeToSubway10min.15min TimeToSubway15min.20min TimeToSubway5min.10min
## 1                       1                       0                      0
## 2                       0                       0                      1
## 3                       0                       0                      1
## 4                       0                       0                      0
## 5                       0                       1                      0
## 6                       1                       0                      0
##   TimeToSubwayno_bus_stop_nearby SubwayStationBanwoldang
```

```
## 1                              0                       0
## 2                              0                       0
## 3                              0                       0
## 4                              0                       0
## 5                              0                       0
## 6                              0                       0
##   SubwayStationChil.sung.market SubwayStationDaegu
## 1                             0                  0
## 2                             0                  1
## 3                             0                  1
## 4                             0                  0
## 5                             0                  0
## 6                             0                  0
##   SubwayStationKyungbuk_uni_hospital SubwayStationMyung.duk
## 1                                  1                      0
## 2                                  0                      0
## 3                                  0                      0
## 4                                  0                      0
## 5                                  0                      1
## 6                                  0                      1
##   SubwayStationno_subway_nearby SubwayStationSin.nam
## 1                             0                   0
## 2                             0                   0
## 3                             0                   0
## 4                             0                   1
## 5                             0                   0
## 6                             0                   0
```

El objetivo de este paso es integrar las variables dummies con las variables numéricas originales en un solo data frame. Esto es crucial para asegurarse de que el conjunto de datos esté preparado para la aplicación del algoritmo k-NN, que requiere que todas las variables sean numéricas.

En las 2 próximas celdas se definen dos funciones esenciales para el preprocesamiento de los datos: una para normalizar las variables numéricas y otra para desnormalizarlas. Estas funciones son fundamentales para el uso de algoritmos de machine learning, como k-NN, que son sensibles a las escalas de las variables.

```r
# Función de normalización
normalizar <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
```

La normalización reescala los valores de una variable para que estén dentro de un rango específico

$$0, 1$$

. Esto es importante para que todas las variables contribuyan de manera equitativa en el cálculo de distancias, evitando que las variables con rangos más amplios dominen las distancias.

```r
# Función de desnormalización
desnormalizar <- function(x, min_val, max_val) {
  return(x * (max_val - min_val) + min_val)
}
```

La desnormalización convierte los valores normalizados de regreso a su escala original. Esto es útil para interpretar los resultados del modelo en el contexto original de los datos.

El algoritmo k-NN es sensible a las escalas de las variables porque se basa en métricas de distancia para encontrar los vecinos más cercanos. Si las variables no están normalizadas, aquellas con rangos más amplios

tendrán una influencia desproporcionada en las distancias calculadas, lo que puede llevar a un rendimiento subóptimo del modelo.

La siguiente celda almacena los valores mínimos y máximos originales de todas las columnas del conjunto de datos base_daegu. Esta información es crucial para la desnormalización de los datos después de que se hayan realizado las predicciones con el modelo k-NN.

```
# Guardar los valores mínimos y máximos originales de todas las columnas
min_vals <- sapply(base_daegu, min)
max_vals <- sapply(base_daegu, max)
```

Después de que normalizar y que el modelo k-NN haga sus predicciones, es necesario convertir estos valores normalizados de regreso a su escala original para interpretarlos en la escala original de los datos.

En la siguiente celda se aplica la normalización a todas las columnas del conjunto de datos `base_daegu`. Dado que las variables categóricas ya han sido convertidas a variables dummies (y, por lo tanto, ya están en formato binario 0 y 1), esta operación efectivamente normaliza solo las variables numéricas.

```
# Normalizar todas las columnas
# dado que las categoricas ya estan normalizadas, al normalizar
# todas las columnas, solo se normalizaran las numericas y las
# categoricas se mantendran igual
base_daegu <- as.data.frame(lapply(base_daegu, normalizar))
# Verificar la normalización
head(base_daegu)
```

```
##     SalePrice YearBuilt YrSold MonthSold  Size.sqf.      Floor
## 1 0.196799115 0.7567568      0 0.6363636 0.30835604 0.04761905
## 2 0.033599893 0.1891892      0 0.6363636 0.20526794 0.16666667
## 3 0.028799650 0.1891892      0 0.6363636 0.20526794 0.11904762
## 4 0.628799288 0.7567568      0 0.6363636 0.87238874 0.16666667
## 5 0.340799173 0.4054054      0 0.6363636 0.73841962 0.04761905
## 6 0.005599379 0.3783784      0 0.6363636 0.09990917 0.09523810
##   N_Parkinglot.Ground. N_Parkinglot.Basement.     N_APT  N_manager N_elevators
## 1            0.1556802             0.13928842 0.1666667 0.15384615  0.00000000
## 2            0.1122020             0.05753217 0.0000000 0.07692308  0.07407407
## 3            0.1122020             0.05753217 0.0000000 0.07692308  0.07407407
## 4            0.3492286             0.40575322 0.4166667 0.30769231  0.40740741
## 5            0.7335203             0.40575322 0.5833333 0.53846154  0.74074074
## 6            0.2805049             0.00000000 0.1666667 0.30769231  0.37037037
##   N_FacilitiesNearBy.PublicOffice. N_FacilitiesNearBy.Hospital.
## 1                        0.2857143                         0.5
## 2                        0.7142857                         0.5
## 3                        0.7142857                         0.5
## 4                        0.1428571                         0.5
## 5                        0.8571429                         1.0
## 6                        1.0000000                         0.5
##   N_FacilitiesNearBy.Dpartmentstore. N_FacilitiesNearBy.Mall.
## 1                                0.5                     0.5
## 2                                1.0                     0.5
## 3                                1.0                     0.5
## 4                                0.0                     0.5
## 5                                0.0                     0.5
## 6                                0.5                     0.5
##   N_FacilitiesNearBy.ETC. N_FacilitiesNearBy.Park. N_SchoolNearBy.Elementary.
## 1                     0.2                     0.0                  0.5000000
## 2                     0.4                     0.5                  0.3333333
```

```
## 3                    0.4                   0.5                 0.3333333
## 4                    0.0                   0.0                 0.3333333
## 5                    1.0                   0.0                 0.6666667
## 6                    1.0                   0.5                 0.6666667
##   N_SchoolNearBy.Middle. N_SchoolNearBy.High. N_SchoolNearBy.University.
## 1                   0.50                  0.4                        0.4
## 2                   0.25                  0.2                        0.0
## 3                   0.25                  0.2                        0.0
## 4                   0.50                  0.2                        0.4
## 5                   0.75                  1.0                        1.0
## 6                   0.75                  1.0                        1.0
##   N_FacilitiesInApt N_FacilitiesNearBy.Total. N_SchoolNearBy.Total.
## 1         0.4444444                    0.3750             0.5294118
## 2         0.2222222                    0.7500             0.2352941
## 3         0.2222222                    0.7500             0.2352941
## 4         0.4444444                    0.1875             0.4117647
## 5         0.3333333                    0.8750             1.0000000
## 6         0.2222222                    1.0000             1.0000000
##   HallwayTypemixed HallwayTypeterraced HeatingTypeindividual_heating
## 1                0                   1                             1
## 2                0                   0                             1
## 3                0                   0                             1
## 4                0                   1                             1
## 5                1                   0                             1
## 6                0                   0                             1
##   AptManageTypeself_management TimeToBusStop10min.15min TimeToBusStop5min.10min
## 1                            0                        0                       1
## 2                            1                        0                       0
## 3                            1                        0                       0
## 4                            0                        0                       0
## 5                            0                        0                       0
## 6                            0                        0                       1
##   TimeToSubway10min.15min TimeToSubway15min.20min TimeToSubway5min.10min
## 1                       1                       0                      0
## 2                       0                       0                      1
## 3                       0                       0                      1
## 4                       0                       0                      0
## 5                       0                       1                      0
## 6                       1                       0                      0
##   TimeToSubwayno_bus_stop_nearby SubwayStationBanwoldang
## 1                              0                       0
## 2                              0                       0
## 3                              0                       0
## 4                              0                       0
## 5                              0                       0
## 6                              0                       0
##   SubwayStationChil.sung.market SubwayStationDaegu
## 1                             0                  0
## 2                             0                  1
## 3                             0                  1
## 4                             0                  0
## 5                             0                  0
## 6                             0                  0
##   SubwayStationKyungbuk_uni_hospital SubwayStationMyung.duk
```

```
## 1                                  1                     0
## 2                                  0                     0
## 3                                  0                     0
## 4                                  0                     0
## 5                                  0                     1
## 6                                  0                     1
##   SubwayStationno_subway_nearby SubwayStationSin.nam
## 1                             0                    0
## 2                             0                    0
## 3                             0                    0
## 4                             0                    1
## 5                             0                    0
## 6                             0                    0
```

Normalizar las variables es esencial en k-NN porque:

- Consistencia: Asegura que todas las variables contribuyan de manera similar en el cálculo de distancias.
- Evita el Dominio de Variables: Previene que las variables con rangos mayores dominen las distancias y, por lo tanto, el resultado del modelo.

Las siguiente celda se verifica que todas las columnas del conjunto de datos base_daegu hayan sido normalizadas correctamente.

```
# Quiero verificar que todos mis columnas tengan como mínimo 0 y como máximo 1
summary(base_daegu)[c(1, 6), ]
```

```
##    SalePrice        YearBuilt          YrSold         MonthSold
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##    Size.sqf.          Floor        N_Parkinglot.Ground. N_Parkinglot.Basement.
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.00000      Min.   :0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.00000      Max.   :1.0000
##      N_APT          N_manager       N_elevators
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
##  N_FacilitiesNearBy.PublicOffice. N_FacilitiesNearBy.Hospital.
##  Min.   :0.0000                   Min.   :0.000
##  Max.   :1.0000                   Max.   :1.000
##  N_FacilitiesNearBy.Dpartmentstore. N_FacilitiesNearBy.Mall.
##  Min.   :0.0000                     Min.   :0.0000
##  Max.   :1.0000                     Max.   :1.0000
##  N_FacilitiesNearBy.ETC. N_FacilitiesNearBy.Park. N_SchoolNearBy.Elementary.
##  Min.   :0.0000          Min.   :0.0000           Min.   :0.0000
##  Max.   :1.0000          Max.   :1.0000           Max.   :1.0000
##  N_SchoolNearBy.Middle. N_SchoolNearBy.High. N_SchoolNearBy.University.
##  Min.   :0.0000         Min.   :0.0000       Min.   :0.0000
##  Max.   :1.0000         Max.   :1.0000       Max.   :1.0000
##  N_FacilitiesInApt N_FacilitiesNearBy.Total. N_SchoolNearBy.Total.
##  Min.   :0.0000    Min.   :0.0000            Min.   :0.0000
##  Max.   :1.0000    Max.   :1.0000            Max.   :1.0000
##  HallwayTypemixed HallwayTypeterraced HeatingTypeindividual_heating
##  Min.   :0.0000   Min.   :0.000       Min.   :0.0000
##  Max.   :1.0000   Max.   :1.000       Max.   :1.0000
##  AptManageTypeself_management TimeToBusStop10min.15min TimeToBusStop5min.10min
##  Min.   :0.00000              Min.   :0.000000         Min.   :0.0000
##  Max.   :1.00000              Max.   :1.000000         Max.   :1.0000
```

```
##  TimeToSubway10min.15min TimeToSubway15min.20min TimeToSubway5min.10min
##  Min.   :0.0000          Min.   :0.0000          Min.   :0.0000
##  Max.   :1.0000          Max.   :1.0000          Max.   :1.0000
##  TimeToSubwayno_bus_stop_nearby SubwayStationBanwoldang
##  Min.   :0.0000                 Min.   :0.000
##  Max.   :1.0000                 Max.   :1.000
##  SubwayStationChil.sung.market SubwayStationDaegu
##  Min.   :0.00000               Min.   :0.00000
##  Max.   :1.00000               Max.   :1.00000
##  SubwayStationKyungbuk_uni_hospital SubwayStationMyung.duk
##  Min.   :0.0000                     Min.   :0.0000
##  Max.   :1.0000                     Max.   :1.0000
##  SubwayStationno_subway_nearby SubwayStationSin.nam
##  Min.   :0.00000               Min.   :0.0000
##  Max.   :1.00000               Max.   :1.0000
```

En la siguiente celda se realiza la división del conjunto de datos en conjuntos de entrenamiento y prueba, asegurando que la distribución de la variable objetivo SalePrice se mantenga consistente en ambos conjuntos mediante estratificación. Se utiliza una semilla aleatoria para garantizar la reproducibilidad de los resultados.

```r
set.seed(123) #inserto semilla
# Divido el conjunto de datos en entrenamiento y prueba
# donde el 70% de los datos son para entrenamiento y el 30% para prueba
# y estratifico por la variable objetivo SalePrice
particiones <- initial_split(base_daegu, prop = 0.7, strata = "SalePrice")
# Guardo los datos de entrenamiento y prueba
# en dos variables diferentes
entrenamiento <- training(particiones)
prueba <- testing(particiones)
```

Porqué del 70%-30%

1. **Suficiente Cantidad de Datos para Entrenamiento**:
   - Usar el 70% de los datos para el entrenamiento asegura que el modelo tenga acceso a una cantidad suficiente de datos para aprender las características y patrones del conjunto de datos. Esto es esencial para construir un modelo robusto y reducir el riesgo de subajuste.
2. **Evaluación Representativa**:
   - Al reservar el 30% de los datos para el conjunto de prueba, se asegura una evaluación representativa del rendimiento del modelo. Este conjunto debe ser lo suficientemente grande para proporcionar una evaluación precisa y confiable del modelo en datos no vistos.
3. **Prácticas Comunes**:
   - La proporción de 70%-30% es una práctica común en machine learning, proporcionando un buen balance entre tener suficientes datos para el entrenamiento y una cantidad adecuada para la evaluación del modelo.

Subajuste y Sobreajuste

Subajuste

- **Definición**: Ocurre cuando un modelo es demasiado simple para capturar la estructura subyacente de los datos. Un modelo con subajuste tendrá un rendimiento pobre tanto en el conjunto de entrenamiento como en el de prueba.
- **Prevención**: Utilizar una proporción adecuada de datos de entrenamiento ayuda a mitigar el subajuste. Con el 70% de los datos para el entrenamiento, el modelo tiene suficiente información para aprender patrones complejos sin ser demasiado simple.

Sobreajuste

- **Definición**: Ocurre cuando un modelo es demasiado complejo y se ajusta demasiado a los datos de entrenamiento, capturando también el ruido y las peculiaridades específicas de estos datos. Un modelo con sobreajuste tendrá un rendimiento excelente en el conjunto de entrenamiento pero pobre en el conjunto de prueba.
- **Prevención**: Reservar el 30% de los datos para el conjunto de prueba permite evaluar cómo generaliza el modelo a datos nuevos. Si el modelo tiene un rendimiento significativamente peor en el conjunto de prueba comparado con el conjunto de entrenamiento, esto indica un posible sobreajuste.

Estratificación por `SalePrice`

- **Motivación**: La estratificación por la variable objetivo `SalePrice` asegura que la distribución de los precios de venta sea representativa en ambos conjuntos (entrenamiento y prueba). Esto es crucial para evitar sesgos y asegurar que el modelo sea evaluado de manera justa en una muestra que refleja la población original.

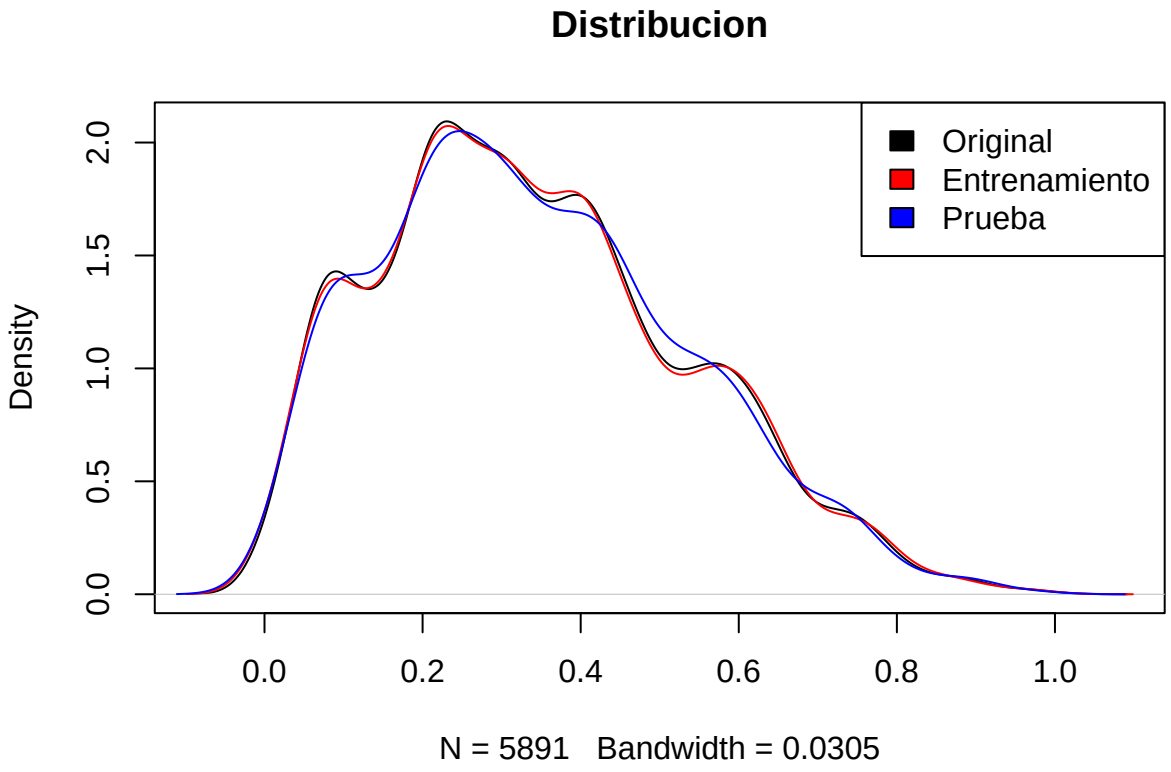
La proporción 70%-30% para la división del conjunto de datos se elige para equilibrar adecuadamente la cantidad de datos disponibles para el entrenamiento y la evaluación, ayudando a prevenir tanto el subajuste como el sobreajuste. La estratificación por `SalePrice` asegura que ambos conjuntos tengan distribuciones representativas, mejorando la robustez y la precisión de la evaluación del modelo.

Importancia de la Estratificación

- `Consistencia en la Distribución`: La estratificación asegura que la variable objetivo tenga una distribución similar en ambos conjuntos, lo cual es importante para evitar sesgos en el modelo.
- `Mejor Evaluación del Modelo`: Garantiza que el modelo sea evaluado en un conjunto de prueba representativo, proporcionando una medida más precisa de su rendimiento en datos no vistos.

En la siguiente celda se genera un gráfico de densidad para comparar la distribución de la variable objetivo SalePrice en el conjunto de datos original, de entrenamiento y de prueba. Esto es importante para asegurarse de que la estratificación haya mantenido la distribución de la variable objetivo de manera consistente en ambos subconjuntos.

```r
# quiero comprobar la distribución de la variable objetivo
# en los conjuntos original, de entrenamiento y de prueba
# con un gráfico de densidad sin ocupar ggplot
plot(density(base_daegu$SalePrice), col = "black", main = "Distribucion")
lines(density(entrenamiento$SalePrice), col = "red")
lines(density(prueba$SalePrice), col = "blue")
legend("topright", c("Original", "Entrenamiento", "Prueba"),
       fill = c("black", "red", "blue"))
```



Líneas de Densidad:

- Negro: Distribución de SalePrice en el conjunto de datos original.
- Rojo: Distribución de SalePrice en el conjunto de entrenamiento.
- Azul: Distribución de SalePrice en el conjunto de prueba.

Observaciones:

- Las tres líneas de densidad (negra, roja y azul) se superponen en gran medida, indicando que las distribuciones de SalePrice en el conjunto original, de entrenamiento y de prueba son muy similares.
- Esto confirma que la estratificación ha funcionado correctamente, manteniendo una distribución representativa de la variable objetivo en ambos conjuntos.

Importancia

Mantener la consistencia en la distribución de la variable objetivo entre los conjuntos de entrenamiento y de prueba es crucial para:

- Evaluación Justa del Modelo: Asegura que el modelo sea evaluado en un conjunto de prueba que es representativo de los datos en general.
- Evitar Sesgos: Previene sesgos en la evaluación del rendimiento del modelo, proporcionando una medida precisa de su capacidad para generalizar a datos no vistos.

En la siguiente celda se aplica el modelo k-NN de regresión utilizando k = 1 y se obtienen las predicciones para el conjunto de prueba.

```
#Aplicacion del modelo knn de regresion con k=1
# y obtencion de las predicciones
knn <- knnreg(SalePrice ~ ., entrenamiento, k = 1)
knn_pred <- predict(knn, newdata = prueba[, colnames(prueba) != "SalePrice"])
knn_pred <- as.vector(knn_pred)
head(knn_pred)
```

```
## [1] 0.033599893 0.268159111 0.005599379 0.108800084 0.628799288 0.171199627
```

- Después de entrenar el modelo KNN (K-Nearest Neighbors) en la línea 1, el modelo está listo para hacer predicciones. El entrenamiento se hizo con el conjunto de datos entrenamiento, usando SalePrice como la variable objetivo y todas las demás variables como predictores. Objetivo:
- El objetivo de la línea 2 es usar el modelo entrenado (knn) para hacer predicciones sobre un conjunto de datos nuevo (el conjunto de prueba prueba).

Proceso:

- `predict(knn, newdata = prueba[, colnames(prueba) != "SalePrice"])`:
- `predict(knn, newdata = ...)`: Esta función toma el modelo entrenado (knn) y hace predicciones usando un nuevo conjunto de datos (newdata).
- `prueba[, colnames(prueba) != "SalePrice"]`: Aquí se especifica que se deben usar todas las columnas del conjunto de prueba prueba, excepto la columna `SalePrice`. Esto se hace porque SalePrice es la variable que queremos predecir, no una de las variables predictoras.
- El modelo utiliza los valores de las variables predictoras en el conjunto de prueba para generar predicciones de `SalePrice` basado en el aprendizaje que hizo el algoritmo en la línea 1. En otras palabras, para cada fila en prueba, el modelo toma los valores de todas las columnas predictoras y genera un valor predicho para `SalePrice`. Resultado:
- El resultado de esta función es un vector de predicciones `(knn_pred)`. Cada valor en este vector corresponde a una predicción de `SalePrice` para una fila del conjunto de prueba. Estas predicciones son los valores estimados de `SalePrice` basados en los valores de las variables predictoras en prueba.
- La línea 2 utiliza el modelo KNN entrenado para predecir los valores de `SalePrice` en el conjunto de prueba, basado en el aprendizaje que hizo el algoritmo en la línea 1. Para cada fila en el conjunto de prueba, el modelo toma los valores de todas las variables predictoras (excepto `SalePrice`) y genera una predicción de `SalePrice`. Estas predicciones son los valores de `knn_pred`.

Respecto a Valor k = 1:

- Utiliza el vecino más cercano para hacer predicciones.
- Es propenso a sobreajustar los datos de entrenamiento, capturando ruido y peculiaridades específicas del conjunto de datos de entrenamiento.
- Puede proporcionar una evaluación inicial del rendimiento del modelo, pero generalmente no es el valor óptimo de k para la mayoría de los problemas. (lo que se demostrará mas adelante)

En la siguiente celda se desnormalizan las predicciones del modelo k-NN para devolver los valores a su escala original. Este paso es crucial para interpretar las predicciones en el contexto original de los datos, específicamente para la variable objetivo SalePrice.

```
#Desnormalizo las predicciones para devolverlas a la escala original
#osea desnormalizo los valores de SalePrice de las predicciones
knn_pred <- desnormalizar(knn_pred, min_vals["SalePrice"],
                          max_vals["SalePrice"])
head(knn_pred)
```

```
## [1]  51327 181061  35840  92920 380530 127433
```

Estos valores ahora están en la escala original de `SalePrice`, lo que permite una interpretación directa y una comparación con los valores reales en el conjunto de prueba.

La desnormalización de las predicciones es esencial para:

- Interpretación: Permitir la interpretación de las predicciones en el contexto original de los datos.
- Comparación: Facilitar la comparación directa entre las predicciones y los valores reales de SalePrice.

En la siguiente celda se desnormalizan los valores de SalePrice en el conjunto de prueba. Esto permite comparar las predicciones desnormalizadas con los valores reales de SalePrice en su escala original.

```r
#desnormalizo los valores de SalePrice en el conjunto de prueba
precio_prueba <- desnormalizar(prueba$SalePrice, min_vals["SalePrice"],
                               max_vals["SalePrice"])
head(precio_prueba)
```

```
## [1]  48672 200884  38053  92035 292035 138053
```

Desnormalizar SalePrice en el conjunto de prueba es crucial para:

- Comparabilidad: Permitir la comparación directa entre las predicciones desnormalizadas y los valores reales de `SalePrice`.
- Evaluación del Modelo: Facilitar la evaluación del rendimiento del modelo utilizando métricas.

En las siguientes 2 celdas se definen y calculan varias métricas de evaluación del modelo k-NN utilizando las predicciones desnormalizadas y los valores reales de `SalePrice` en el conjunto de prueba. Estas métricas permiten evaluar la precisión y efectividad del modelo.

```r
# Funciones de Métricas de Evaluación del Modelo
mae <- function(actual, predicted) {
  mean(abs(actual - predicted))
}

rmse <- function(actual, predicted) {
  sqrt(mean((actual - predicted)^2))
}

metrics <- function(actual, predicted) {
  cat("cor  =", cor(actual, predicted),
      "\nMAE  =", mae(actual, predicted),
      "\nRMSE =", rmse(actual, predicted))
}
```

```r
#Veo las metricas de evaluacion
metrics(precio_prueba, knn_pred)
```

```
## cor  = 0.9615033
## MAE  = 18413.52
## RMSE = 29335.16
```

Resultados de las Métricas

- Correlación (cor): Indica una fuerte correlación positiva entre los valores predichos y los valores reales. Una correlación de 0.9615 sugiere que las predicciones del modelo k-NN están fuertemente alineadas con los valores reales de `SalePrice`.
- MAE (Mean Absolute Error): Representa el error absoluto medio en las predicciones. Un valor más bajo indica un modelo más preciso. Un MAE de 18413.52 indica que, en promedio, las predicciones del modelo k-NN están a 18413.52 unidades monetarias de los valores reales.
- RMSE (Root Mean Squared Error): Representa la raíz del error cuadrático medio. Un valor más bajo indica un modelo más preciso. Un RMSE de 29335.16 proporciona una medida de la desviación cuadrática media de las predicciones respecto a los valores reales. Valores más bajos de RMSE son preferibles ya que indican menor error.

Dado que el paquete knnreg ofrece muy poca configuración, y el objetivo es minimizar el RMSE, algo que es conseguible configurando parametros e hiperparametro, se opta por ocupar paquete kknn, para revisar si es que se puede minimizar más la metrica RMSE.

k-nn con paquete kknn

En la siguiente celda se aplica el modelo k-NN de regresión utilizando el paquete kknn en lugar de knnreg. El objetivo es aprovechar las capacidades avanzadas de configuración y optimización del paquete kknn para minimizar la métrica RMSE.

```
set.seed(12345)
# realizo la prediccion con el modelo knn de regresionm con k=1
# y con el metodo de kernel optimo, distancia euclidiana p = 2
knn <- kknn(SalePrice ~ ., entrenamiento, prueba, distance = 2, k = 1)
knn_pred <- knn$fitted.values
head(knn_pred)
```

```
## [1] 0.033599893 0.268159111 0.005599379 0.108800084 0.628799288 0.171199627
```

Paso a Paso del Proceso

Configuración de la Semilla:

- `set.seed(12345)`: Establece una semilla para garantizar la reproducibilidad de los resultados.

Aplicación del Modelo k-NN con kknn:

- `knn <- kknn(SalePrice ~ ., entrenamiento, prueba, distance = 2, k = 1)`:
- Utiliza la función `kknn` para crear un modelo k-NN de regresión.
- `SalePrice ~ .` indica que `SalePrice` es la variable objetivo y todas las demás columnas son predictores.
- `distance = 2` especifica el uso de la distancia euclidiana.
- `k = 1` indica que se utiliza el vecino más cercano.

Obtención de las Predicciones:

- `knn_pred <- knn$fitted.values`: Extrae los valores predichos del modelo k-NN.
- `head(knn_pred)`: Muestra las primeras seis predicciones obtenidas.

El paquete `knnreg` ofrece opciones limitadas para configurar y optimizar el modelo k-NN. En contraste, kknn proporciona mayor flexibilidad y capacidad de ajuste, permitiendo configuraciones avanzadas como:

- `Métodos de Kernel`: Diferentes métodos de kernel pueden influir en la suavidad de las predicciones.
- `Distancias Personalizables`: Permite especificar diferentes métricas de distancia (e.g., euclidiana, Manhattan).
- `Optimización de Hiperparámetros`: Facilita la búsqueda de valores óptimos para los hiperparámetros, como el número de vecinos k.

En la siguiente celda se desnormalizan las predicciones obtenidas del modelo k-NN para devolver los valores a su escala original de SalePrice. Este paso es crucial para interpretar las predicciones en la escala original de los datos.

```
#Desnormalizo las predicciones para devolverlas a la escala original
knn_pred <- desnormalizar(knn_pred, min_vals["SalePrice"],
                          max_vals["SalePrice"])
head(knn_pred)
```

```
## [1]  51327 181061  35840  92920 380530 127433
```

Los valores ahora están en la escala original de SalePrice, lo que permite una interpretación directa y una comparación con los valores reales en el conjunto de prueba.

La desnormalización de las predicciones es esencial para:

- Interpretación: Permitir la interpretación de las predicciones en el contexto original de los datos.
- Comparación: Facilitar la comparación directa entre las predicciones y los valores reales de SalePrice.

En la siguiente celda se desnormalizan los valores de SalePrice en el conjunto de prueba. Esto permite comparar las predicciones desnormalizadas con los valores reales de SalePrice en su escala original.

```r
#desnormalizo los valores de SalePrice en el conjunto de prueba
precio_prueba <- desnormalizar(prueba$SalePrice, min_vals["SalePrice"],
                               max_vals["SalePrice"])
head(precio_prueba)
```

```
## [1]  48672 200884  38053  92035 292035 138053
```

Los valores están ahora en la escala original de SalePrice, lo que facilita la interpretación y comparación con las predicciones del modelo.

Desnormalizar SalePrice en el conjunto de prueba es crucial para:

- Comparabilidad: Permitir la comparación directa entre las predicciones desnormalizadas y los valores reales de SalePrice.
- Evaluación del Modelo: Facilitar la evaluación del rendimiento del modelo utilizando métricas.

En las siguientes celda se ocupa la variable metrics, ya definida y explicada anteriormente, pero en esta oportunidad para calcular varias métricas de evaluación del modelo k-NN utilizando el paquete kknn ocupando las predicciones desnormalizadas y los valores reales de `SalePrice` en el conjunto de prueba.

```r
#Veo las metricas de evaluacion
metrics(precio_prueba, knn_pred)
```

```
## cor  = 0.962854
## MAE  = 17643.97
## RMSE = 28791.39
```

Análisis de los Resultados

Correlación

- **`knnreg`**: 0.9615033
- **`kknn`**: 0.962854
- **Interpretación**: Ambos modelos muestran una correlación alta entre las predicciones y los valores reales, indicando que ambos modelos capturan bien la relación entre las variables predictoras y la variable objetivo. Sin embargo, el modelo **`kknn`** tiene una correlación ligeramente superior, sugiriendo una mejor alineación entre las predicciones y los valores reales.

MAE (Mean Absolute Error)

- **`knnreg`**: 18413.52
- **`kknn`**: 17643.97
- **Interpretación**: El MAE más bajo del modelo **`kknn`** indica que, en promedio, las predicciones están más cerca de los valores reales en comparación con el modelo **`knnreg`**. Esto sugiere una mejora en la precisión de las predicciones al usar **`kknn`**.

RMSE (Root Mean Squared Error)

- **`knnreg`**: 29335.16
- **`kknn`**: 28791.39
- **Interpretación**: El RMSE más bajo del modelo **`kknn`** indica que las predicciones tienen una desviación cuadrática media menor respecto a los valores reales en comparación con el modelo **`knnreg`**. Esto sugiere que el modelo **`kknn`** maneja mejor los errores grandes y proporciona predicciones más precisas.

Ventajas de Usar `kknn` Sobre `knnreg`

1. **Configuración Avanzada**:
   - **Número de Vecinos**: Ambos paquetes permiten configurar el número de vecinos (k), pero **`kknn`** ofrece configuraciones adicionales.

- **Métricas de Distancia**: `kknn` permite elegir entre diferentes métricas de distancia (Euclidiana, Manhattan, Minkowski), proporcionando flexibilidad para seleccionar la mejor métrica para los datos específicos.
- **Métodos de Kernel**: `kknn` incluye opciones de kernel que pueden mejorar la suavidad y precisión de las predicciones.

2. **Mejor Rendimiento**:
   - **Menor MAE y RMSE**: Las métricas más bajas obtenidas con `kknn` indican que este paquete puede producir modelos más precisos, lo cual es crucial cuando el objetivo es minimizar el RMSE.
   - **Mejor Correlación**: La correlación ligeramente superior sugiere que las predicciones de `kknn` están mejor alineadas con los valores reales.

El uso del paquete `kknn` ha demostrado una mejora en el rendimiento del modelo k-NN en comparación con `knnreg`. Por ende lo siguiente será encontrar los mejores parametros de kknn para minimizar aún más el RMSE.

En la siguiente celda se configura el control de entrenamiento utilizando el método de bootstrap y se define una cuadrícula de hiperparámetros para explorar durante el proceso de ajuste del modelo. El objetivo es encontrar la combinación de hiperparámetros que minimice el RMSE.

```
# Configuración de control de entrenamiento para bootstrap
boot <- trainControl(method = "boot", number = 1)
# Parámetros
hyper_grid <- expand.grid(kmax = seq(1, 20, 2),
                          distance = seq(1, 3, 1),
                          kernel = c("optimal", "gaussian", "triangular"))
```

Paso a Paso del Proceso

1. Configuración de Control de Entrenamiento para Bootstrap:

- `boot <- trainControl(method = "boot", number = 1)`:
- `method = "boot"`: Especifica que se utilizará el método de bootstrap para el remuestreo.
- `number = 1`: Indica que se realizará un solo ciclo de bootstrap en cada iteración del ajuste de hiperparámetros. Esto se puede ajustar según la necesidad para realizar más iteraciones de bootstrap.

2. Definición de la Cuadrícula de Hiperparámetros:

- `hyper_grid <- expand.grid(kmax = seq(1, 20, 2), distance = seq(1, 3, 1), kernel = c("optimal", "gaussian", "triangular"))`:
- `kmax = seq(1, 20, 2)`: Explora valores impares de k desde 1 hasta 20.
- `distance = seq(1, 3, 1)`: Explora tres métricas de distancia diferentes: 1 (Manhattan), 2 (Euclidiana), y 3 (Minkowski).
- `kernel = c("optimal", "gaussian", "triangular")`: Explora tres métodos de kernel diferentes: óptimo, gaussiano y triangular.

Selección de Valores de k

Número de Vecinos (kmax):

- Rango de 1 a 20: Este rango proporciona un buen equilibrio entre explorar modelos muy simples (k = 1) y modelos más complejos (k = 20).
- Incremento de 2 en 2: Los valores impares de k se utilizan para evitar empates en la clasificación, lo cual es particularmente relevante en problemas de clasificación. Aunque en regresión, el uso de valores impares ayuda a mantener consistencia y evita situaciones en las que múltiples vecinos podrían influir de manera igualitaria pero contradictoria en la predicción.
- Número Impar: Para regresión, el uso de números impares también facilita una mejor distribución de los vecinos considerados y evita posibles sesgos introducidos por divisiones iguales.

Importancia

- Método de Bootstrap: El bootstrap es un método de remuestreo que permite evaluar la estabilidad y la variabilidad del modelo, proporcionando una estimación más robusta del rendimiento del modelo.
- Exploración de Hiperparámetros: La definición de una cuadrícula de hiperparámetros permite explorar sistemáticamente diferentes combinaciones de parámetros para encontrar la mejor configuración que minimice el RMSE. La capacidad de ajustar múltiples parámetros simultáneamente (k, distancia y kernel) es una ventaja significativa del paquete kknn.

El objetivo es utilizar el control de entrenamiento y la cuadrícula de hiperparámetros para ajustar el modelo k-NN de manera óptima, encontrando la combinación de parámetros que minimice el RMSE y mejore la precisión del modelo.

Nota importante: El modificar el numero de bootstraps trae consigo una enorme carga computacional, a modo de ejemplo, lo probe con 3 numeros distintos, donde obtuve los siguientes tiempos de ejecucion:

- 1 bootstrap --> 3 min 50
- 5 bootstraps --> 19 min 40
- 20 bootstraps --> 1 hr 16 min

Por lo que para efectos de velocidad de ejecución de este modelo, preferí dejarlo con 1 bootstrap.

En la siguiente celda se entrena un modelo k-NN utilizando el paquete kknn con una grilla de hiperparámetros definida previamente. El objetivo es encontrar la combinación óptima de hiperparámetros que minimice el RMSE.

```
# Entrenamiento
set.seed(12345)
knn_fit <- train(SalePrice ~ ., data = entrenamiento, method = "kknn",
                 metric = "RMSE", trControl = boot, tuneGrid = hyper_grid)

knn_fit
```

```
## k-Nearest Neighbors
##
## 4121 samples
##   40 predictor
##
## No pre-processing
## Resampling: Bootstrapped (1 reps)
## Summary of sample sizes: 4121
## Resampling results across tuning parameters:
##
##   kmax  distance  kernel      RMSE        Rsquared   MAE
##    1    1         optimal     0.05302130  0.9223631  0.03352027
##    1    1         gaussian    0.05302130  0.9223631  0.03352027
##    1    1         triangular  0.05302130  0.9223631  0.03352027
##    1    2         optimal     0.05392793  0.9191230  0.03453864
##    1    2         gaussian    0.05392793  0.9191230  0.03453864
##    1    2         triangular  0.05392793  0.9191230  0.03453864
##    1    3         optimal     0.05283243  0.9222793  0.03400897
##    1    3         gaussian    0.05283243  0.9222793  0.03400897
##    1    3         triangular  0.05283243  0.9222793  0.03400897
##    3    1         optimal     0.05038349  0.9294120  0.03192316
##    3    1         gaussian    0.05302130  0.9223631  0.03352027
##    3    1         triangular  0.04817849  0.9351485  0.03056020
##    3    2         optimal     0.05168601  0.9254183  0.03307507
##    3    2         gaussian    0.05392793  0.9191230  0.03453864
##    3    2         triangular  0.04946447  0.9314592  0.03170190
```

```
##    3    3         optimal     0.05089160  0.9276307  0.03274776
##    3    3         gaussian    0.05283243  0.9222793  0.03400897
##    3    3         triangular  0.04934482  0.9318161  0.03144182
##    5    1         optimal     0.05038349  0.9294120  0.03192316
##    5    1         gaussian    0.05302130  0.9223631  0.03352027
##    5    1         triangular  0.04817849  0.9351485  0.03056020
##    5    2         optimal     0.05168601  0.9254183  0.03307507
##    5    2         gaussian    0.05392793  0.9191230  0.03453864
##    5    2         triangular  0.04946447  0.9314592  0.03170190
##    5    3         optimal     0.05089160  0.9276307  0.03274776
##    5    3         gaussian    0.05283243  0.9222793  0.03400897
##    5    3         triangular  0.04934482  0.9318161  0.03144182
##    7    1         optimal     0.05038349  0.9294120  0.03192316
##    7    1         gaussian    0.05302130  0.9223631  0.03352027
##    7    1         triangular  0.04817849  0.9351485  0.03056020
##    7    2         optimal     0.05168601  0.9254183  0.03307507
##    7    2         gaussian    0.05392793  0.9191230  0.03453864
##    7    2         triangular  0.04859204  0.9338438  0.03124157
##    7    3         optimal     0.05089160  0.9276307  0.03274776
##    7    3         gaussian    0.05283243  0.9222793  0.03400897
##    7    3         triangular  0.04897029  0.9328281  0.03120197
##    9    1         optimal     0.05038349  0.9294120  0.03192316
##    9    1         gaussian    0.05302130  0.9223631  0.03352027
##    9    1         triangular  0.04817849  0.9351485  0.03056020
##    9    2         optimal     0.05168601  0.9254183  0.03307507
##    9    2         gaussian    0.05392793  0.9191230  0.03453864
##    9    2         triangular  0.04859204  0.9338438  0.03124157
##    9    3         optimal     0.05089160  0.9276307  0.03274776
##    9    3         gaussian    0.05283243  0.9222793  0.03400897
##    9    3         triangular  0.04897029  0.9328281  0.03120197
##   11    1         optimal     0.05038349  0.9294120  0.03192316
##   11    1         gaussian    0.05302130  0.9223631  0.03352027
##   11    1         triangular  0.04817849  0.9351485  0.03056020
##   11    2         optimal     0.05168601  0.9254183  0.03307507
##   11    2         gaussian    0.05392793  0.9191230  0.03453864
##   11    2         triangular  0.04946447  0.9314592  0.03170190
##   11    3         optimal     0.05089160  0.9276307  0.03274776
##   11    3         gaussian    0.05283243  0.9222793  0.03400897
##   11    3         triangular  0.04934482  0.9318161  0.03144182
##   13    1         optimal     0.05038349  0.9294120  0.03192316
##   13    1         gaussian    0.05302130  0.9223631  0.03352027
##   13    1         triangular  0.04817849  0.9351485  0.03056020
##   13    2         optimal     0.05168601  0.9254183  0.03307507
##   13    2         gaussian    0.05392793  0.9191230  0.03453864
##   13    2         triangular  0.04946447  0.9314592  0.03170190
##   13    3         optimal     0.05089160  0.9276307  0.03274776
##   13    3         gaussian    0.05283243  0.9222793  0.03400897
##   13    3         triangular  0.04934482  0.9318161  0.03144182
##   15    1         optimal     0.05038349  0.9294120  0.03192316
##   15    1         gaussian    0.05302130  0.9223631  0.03352027
##   15    1         triangular  0.04817849  0.9351485  0.03056020
##   15    2         optimal     0.05168601  0.9254183  0.03307507
##   15    2         gaussian    0.05392793  0.9191230  0.03453864
##   15    2         triangular  0.04859204  0.9338438  0.03124157
```

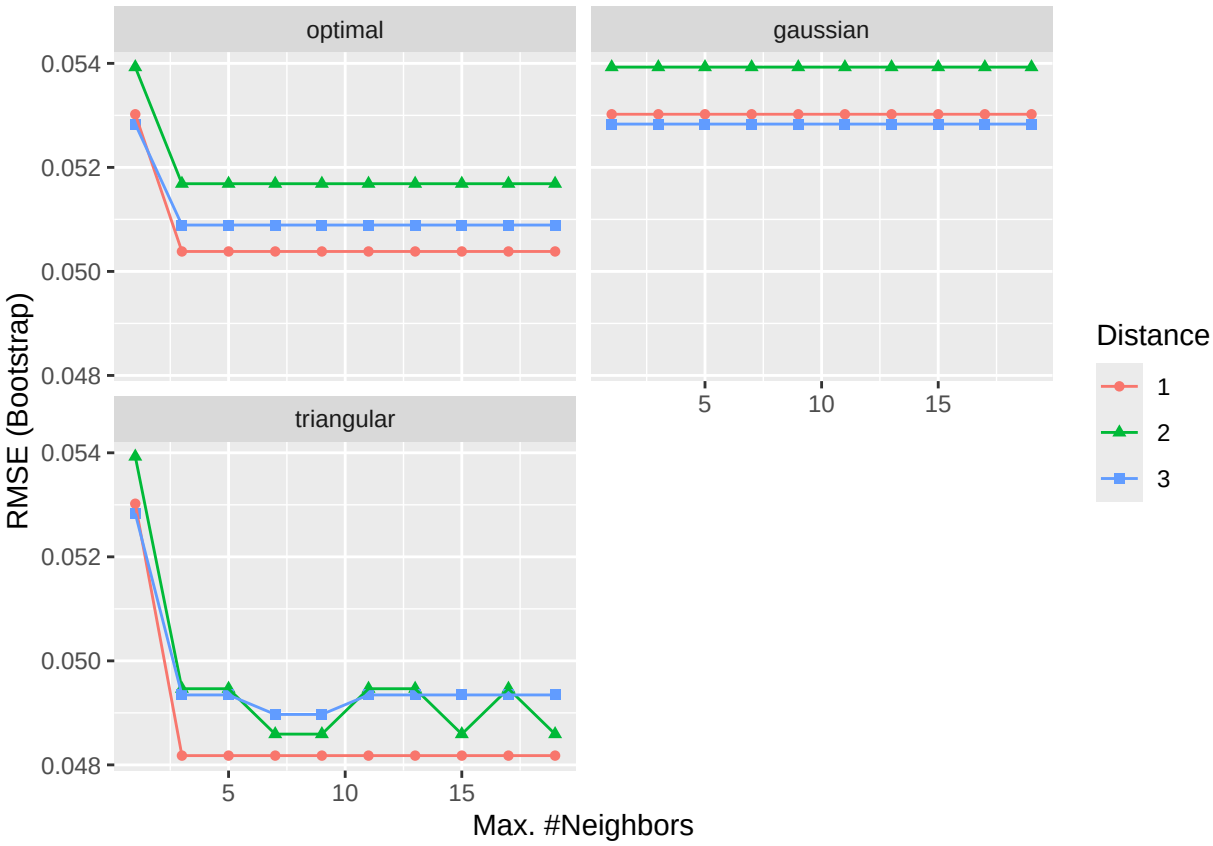
```
##   15    3         optimal     0.05089160  0.9276307  0.03274776
##   15    3         gaussian    0.05283243  0.9222793  0.03400897
##   15    3         triangular  0.04934482  0.9318161  0.03144182
##   17    1         optimal     0.05038349  0.9294120  0.03192316
##   17    1         gaussian    0.05302130  0.9223631  0.03352027
##   17    1         triangular  0.04817849  0.9351485  0.03056020
##   17    2         optimal     0.05168601  0.9254183  0.03307507
##   17    2         gaussian    0.05392793  0.9191230  0.03453864
##   17    2         triangular  0.04946447  0.9314592  0.03170190
##   17    3         optimal     0.05089160  0.9276307  0.03274776
##   17    3         gaussian    0.05283243  0.9222793  0.03400897
##   17    3         triangular  0.04934482  0.9318161  0.03144182
##   19    1         optimal     0.05038349  0.9294120  0.03192316
##   19    1         gaussian    0.05302130  0.9223631  0.03352027
##   19    1         triangular  0.04817849  0.9351485  0.03056020
##   19    2         optimal     0.05168601  0.9254183  0.03307507
##   19    2         gaussian    0.05392793  0.9191230  0.03453864
##   19    2         triangular  0.04859204  0.9338438  0.03124157
##   19    3         optimal     0.05089160  0.9276307  0.03274776
##   19    3         gaussian    0.05283243  0.9222793  0.03400897
##   19    3         triangular  0.04934482  0.9318161  0.03144182
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were kmax = 19, distance = 1 and kernel
##  = triangular.
```

El modelo se ajustó utilizando bootstrap y se evaluó a través de varias combinaciones de hiperparámetros (kmax, distancia y kernel).

El uso del paquete kknn con una cuadrícula de hiperparámetros y el método de bootstrap ha permitido encontrar una combinación óptima de parámetros que mejora significativamente las métricas de rendimiento del modelo k-NN. Esto demuestra la importancia de la optimización de hiperparámetros para mejorar la precisión y robustez de los modelos de machine learning.

En la siguiente celda se genera un gráfico utilizando ggplot2 para visualizar los resultados del ajuste del modelo k-NN con diferentes hiperparámetros. El gráfico muestra cómo varía el RMSE en función del número de vecinos (kmax), la métrica de distancia, y el tipo de kernel utilizado.

```r
# grafico de los resultados
ggplot(knn_fit) +
  facet_wrap(~ kernel, ncol = 2)
```



Visualizar los resultados del ajuste del modelo permite:

- **Identificación de Hiperparámetros Óptimos**: Facilita la identificación de la mejor combinación de hiperparámetros que minimiza el RMSE.
- **Comparación de Desempeño**: Permite comparar el desempeño de diferentes tipos de kernel y métricas de distancia, ayudando a entender cuál combinación es más efectiva.
- **Información para Toma de Decisiones**: Proporciona información visual que puede ser utilizada para tomar decisiones informadas sobre qué hiperparámetros usar en el modelo final.

El gráfico indica que el uso de la distancia Manhattan (1) y el kernel triangular, con un número de vecinos entre 3 a 19, proporciona los mejores resultados en términos de RMSE. Esto valida los resultados numéricos obtenidos anteriormente y proporciona una representación visual clara del rendimiento del modelo.

Por otro lado el gráfico de resultados muestra que tanto un k-NN con 3 vecinos como uno con 19 vecinos producen RMSE similares, cuando la distancia es Manhattan y el kernel triangular. La diferencia en RMSE es mínima, por lo que es importante considerar otros factores al decidir entre estos dos valores de k.

Comparación entre k = 3 y k = 19

1. **k = 3**:
   - **Ventajas**:
     - **Menor Complejidad Computacional**: Utilizar 3 vecinos requiere menos cálculos de distancia y menos tiempo de predicción.
     - **Menor Riesgo de Sobreajuste**: Un valor de k bajo puede capturar las peculiaridades locales de los datos, pero aún mantiene una capacidad de generalización razonable.
   - **Desventajas**:
     - **Sensibilidad al Ruido**: Con un valor de k bajo, el modelo es más sensible al ruido y a las anomalías en los datos, lo que puede afectar la precisión de las predicciones en ciertos casos.
2. **k = 19**:

- **Ventajas**:
  - **Mayor Robustez**: Utilizar más vecinos puede proporcionar predicciones más robustas y menos sensibles a las fluctuaciones y ruido en los datos, ya que se promedian más puntos.
  - **Reducción de Variabilidad**: Un valor de k más alto reduce la variabilidad de las predicciones y puede mejorar la estabilidad del modelo.
- **Desventajas**:
  - **Mayor Complejidad Computacional**: Requiere más cálculos de distancia, lo que aumenta el tiempo de predicción y la carga computacional, especialmente con grandes conjuntos de datos.
  - **Riesgo de Subajuste**: Un valor de k muy alto puede llevar a subajuste, donde el modelo es demasiado general y no captura las relaciones locales en los datos.

Consideraciones Prácticas

- **Tamaño del Conjunto de Datos**: Con un conjunto de datos más grande, utilizar un k más alto como 19 puede ser más beneficioso para obtener predicciones robustas. En conjuntos de datos más pequeños, un valor más bajo como 3 puede ser más adecuado.
- **Distribución de los Datos**: Si los datos tienen muchas anomalías o ruido, un k más alto puede ser preferible para suavizar estos efectos. En casos donde los datos son limpios y consistentes, un k más bajo puede capturar mejor las relaciones locales.
- **Velocidad de Predicción**: Si la velocidad de predicción es crucial, un k más bajo será más rápido. Para aplicaciones en tiempo real, la diferencia en tiempo de computación entre k = 3 y k = 19 puede ser significativa.

La elección entre k = 3 y k = 19 depende entonces, de un balance entre robustez y sensibilidad, complejidad computacional y riesgo de sobreajuste o subajuste. Dado que la diferencia en RMSE es mínima, se puede considerar:

- **Para Mayor Robustez y Menos Sensibilidad al Ruido**: Utilizar k = 19.
- **Para Menor Complejidad Computacional y Mayor Sensibilidad a Relaciones Locales**: Utilizar k = 3.

En contextos donde se requiere una alta precisión y estabilidad en las predicciones, k = 19 puede ser la mejor opción. Para aplicaciones donde la velocidad y la simplicidad son más importantes, k = 3 podría ser preferible. Por ende, bajo este analisis la mejor opción será k = 19.

En la siguiente celda se aplica el modelo k-NN de regresión utilizando k = 19 vecinos, el método de kernel triangular y la distancia Manhattan (p = 1). Este ajuste se realiza para verificar si se obtiene un rendimiento óptimo del modelo, basado en los resultados anteriores.

```
set.seed(12345)
# Realizo la predicción con el modelo knn de regresión, con k=19
# y con el método de kernel triangular, distancia Manhattan p = 1
knn <- kknn(SalePrice ~ ., entrenamiento, prueba, kernel = "triangular",
            distance = 1, k = 19)
knn_pred <- knn$fitted.values
head(knn_pred)
```

```
## [1] 0.04300163 0.31924285 0.02793283 0.10223801 0.48525172 0.16024872
```

Paso a Paso del Proceso

1. Configuración de la Semilla:

- `set.seed(12345)`: Establece una semilla para garantizar la reproducibilidad de los resultados.

2. Aplicación del Modelo k-NN con kknn:

- `knn <- kknn(SalePrice ~ ., entrenamiento, prueba, kernel = "triangular", distance = 1, k = 19)`:
- Utiliza la función kknn para crear un modelo k-NN de regresión.
- `SalePrice ~ .`: Indica que SalePrice es la variable objetivo y todas las demás columnas del conjunto de datos se utilizan como predictores.
- `kernel = "triangular"`: Especifica el uso del kernel triangular.
- `distance = 1`: Especifica el uso de la distancia Manhattan.
- `k = 19`: Indica que se utilizarán los 19 vecinos más cercanos para hacer las predicciones.

3. Obtención de las Predicciones:

- `knn_pred <- knn$fitted.values`: Extrae los valores predichos del modelo k-NN.
- `head(knn_pred)`: Muestra las primeras seis predicciones obtenidas por el modelo k-NN.

Este enfoque permitirá verificar si el uso de k = 19, distancia = 1 (manhattan) y kernel triangular proporcionan una mejora significativa en el rendimiento del modelo en comparación con los otros modelos considerando las métricas de evaluación y la robustez de las predicciones.

En la próxima celda se desnormalizan las predicciones obtenidas del modelo k-NN (con k = 19, kernel triangular, y distancia Manhattan) para devolver los valores a su escala original de SalePrice.

```
#Desnormalizo las predicciones para devolverlas a la escala original
knn_pred <- desnormalizar(knn_pred, min_vals["SalePrice"],
                          max_vals["SalePrice"])
head(knn_pred)
```

```
## [1]  56527.07 209315.26  48192.56  89290.53 301134.27 121376.09
```

Los valores ahora están en la escala original de SalePrice, lo que permite una interpretación directa y una comparación con los valores reales en el conjunto de prueba.

La desnormalización de las predicciones es esencial para:

- Interpretación: Permitir la interpretación de las predicciones en el contexto original de los datos.
- Comparación: Facilitar la comparación directa entre las predicciones y los valores reales de SalePrice.

En la siguiente celda se desnormalizan los valores de SalePrice en el conjunto de prueba. Esto permite comparar las predicciones desnormalizadas con los valores reales de SalePrice en su escala original.

```
#desnormalizo los valores de SalePrice en el conjunto de prueba
precio_prueba <- desnormalizar(prueba$SalePrice, min_vals["SalePrice"],
                               max_vals["SalePrice"])
head(precio_prueba)
```

```
## [1]  48672 200884  38053  92035 292035 138053
```

Estos valores están ahora en la escala original de SalePrice, lo que facilita la interpretación y comparación con las predicciones del modelo.

Desnormalizar SalePrice en el conjunto de prueba es crucial para:

- Comparabilidad: Permitir la comparación directa entre las predicciones desnormalizadas y los valores reales de SalePrice.
- Evaluación del Modelo: Facilitar la evaluación del rendimiento del modelo utilizando métricas.

```
#Veo las metricas de evaluacion
metrics(precio_prueba, knn_pred)
```

```
## cor  = 0.9723958
## MAE  = 16558.37
## RMSE = 24760.22
```

Comparación de Resultados de k-NN Usando Diferentes Configuraciones y Paquetes

Resultados con `knnreg` (k = 1)

- **Correlación (cor)**: 0.9615033
- **MAE (Mean Absolute Error)**: 18413.52
- **RMSE (Root Mean Squared Error)**: 29335.16

Resultados con `kknn` sin Optimización (k = 1, distance = 2, kernel = optimal)

- **Correlación (cor)**: 0.962854
- **MAE (Mean Absolute Error)**: 17643.97
- **RMSE (Root Mean Squared Error)**: 28791.39

Resultados con `kknn` Optimizado (k = 19, distance = 1, kernel = triangular)

- **Correlación (cor)**: 0.9723958
- **MAE (Mean Absolute Error)**: 16558.37
- **RMSE (Root Mean Squared Error)**: 24760.22

Análisis de los Resultados

Correlación

- **`knnreg` (k = 1)**: 0.9615033
- **`kknn` sin optimización**: 0.962854
- **`kknn` optimizado**: 0.9723958
- **Interpretación**: La correlación mejora ligeramente con el uso de `kknn` en comparación con `knnreg`. La mayor mejora se observa con la versión optimizada de `kknn`, indicando una mejor alineación entre las predicciones y los valores reales.

MAE (Mean Absolute Error)

- **`knnreg` (k = 1)**: 18413.52
- **`kknn` sin optimización**: 17643.97
- **`kknn` optimizado**: 16558.37
- **Interpretación**: El MAE disminuye significativamente al pasar de `knnreg` a `kknn`, y aún más con la versión optimizada de `kknn`. Un MAE más bajo indica que las predicciones están, en promedio, más cerca de los valores reales, lo que sugiere una mejora en la precisión del modelo.

RMSE (Root Mean Squared Error)

- **`knnreg` (k = 1)**: 29335.16
- **`kknn` sin optimización**: 28791.39
- **`kknn` optimizado**: 24760.22
- **Interpretación**: El RMSE muestra una reducción significativa al pasar de `knnreg` a `kknn`, y aún más con la versión optimizada de `kknn`. Un RMSE más bajo indica una menor desviación cuadrática media entre las predicciones y los valores reales, lo que refleja una mejor capacidad del modelo para hacer predicciones precisas.

Ventajas de Usar `kknn` con Hiperparámetros Optimizados

1. **Mayor Precisión**:
   - La optimización de los hiperparámetros en `kknn` ha resultado en mejoras significativas en las métricas de evaluación, particularmente en RMSE y MAE, que son indicadores clave de la precisión del modelo.
   - La correlación más alta también sugiere que el modelo optimizado captura mejor la relación entre las variables predictoras y la variable objetivo.
2. **Flexibilidad y Configurabilidad**:
   - `kknn` permite ajustar no solo el número de vecinos (k), sino también la métrica de distancia y el tipo de kernel, proporcionando una mayor flexibilidad para adaptar el modelo a los datos específicos.

- La capacidad de optimizar estos parámetros ayuda a mejorar significativamente el rendimiento del modelo en comparación con el uso de configuraciones predeterminadas.

Conclusión

El uso del paquete `kknn` con una cuadrícula de hiperparámetros y el método de bootstrap ha permitido encontrar una combinación óptima de parámetros que mejora significativamente las métricas de rendimiento del modelo k-NN. En particular, utilizar k = 19, distancia Manhattan, y kernel triangular ha resultado en el mejor rendimiento, minimizando el RMSE y mejorando la precisión de las predicciones.