Ejemplo de regresión con k-NN

K-Nearest Neighbor (k-NN) es un algoritmo de aprendizaje automático supervisado que se puede utilizar para problemas de clasificación y regresión. En este algoritmo, k es una constante definida por el usuario y el vector de distancias de los vecinos más cercanos se calcula usándolo.

El paquete 'caret' proporciona la función 'knnreg' para aplicar KNN para problemas de regresión.

En esta práctica, aprenderemos brevemente cómo ajustar y predecir datos de regresión usando la función 'knnreg' en R. Realizaremos lo siguiente:

1. Preparación de los datos
2. Ajuste del modelo y predicción
3. Comprobación de precisión

Comenzaremos cargando las bibliotecas requeridas.

```
> library(caret)
```

Preparación de los datos

Utilizamos el conjunto de datos de precios de la vivienda de Boston como datos de regresión objetivo en esta práctica. Después de cargar el conjunto de datos, primero, los dividiremos en las partes de entrenamiento y prueba, y extraeremos las partes de entrada *x* y etiqueta *y*. Aquí, extraeremos el 15 por ciento del conjunto de datos como datos de prueba. Es mejor escalar *x* parte de los datos para mejorar la precisión.

```
> boston = MASS::Boston
> str(boston)
> set.seed(12)
>
> indexes = createDataPartition(boston$medv, p = .85, list = F)
> train = boston[indexes, ]
> test = boston[-indexes, ]
>
> train_x = train[, -14]
> train_x = scale(train_x)[,]
> train_y = train[,14]
> test_x = test[, -14]
```

```
> test_x = scale(test[,-14])[,]
> test_y = test[,14]
```

```
'data.frame':   506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Ajuste del modelo y predicción

Definiremos el modelo usando la función knnreg() del paquete 'caret' y ajustaremos los datos del entrenamiento. La llamada a la función es suficiente para entrenar el modelo con los datos incluidos.

```
> knnmodel = knnreg(train_x, train_y)
> str(knnmodel)
```

```
List of 3
 $ learn  :List of 2
  ..$ y: num [1:432] 24 21.6 34.7 36.2 28.7 22.9 16.5 18.9 15 18.9 ...
  ..$ X: num [1:432, 1:13] -0.412 -0.41 -0.41 -0.405 -0.41 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:432] "1" "2" "3" "5" ...
  .. .. ..$ : chr [1:13] "crim" "zn" "indus" "chas" ...
 $ k      : num 5
 $ theDots: list()
 - attr(*, "class")= chr "knnreg"
```

Ahora, podemos predecir los datos de prueba *x* con el modelo entrenado.

```
> pred_y = predict(knnmodel, data.frame(test_x))
```

Comprobación de precisión

A continuación, comprobaremos la precisión de la predicción con las métricas MSE, MAE y RMSE.

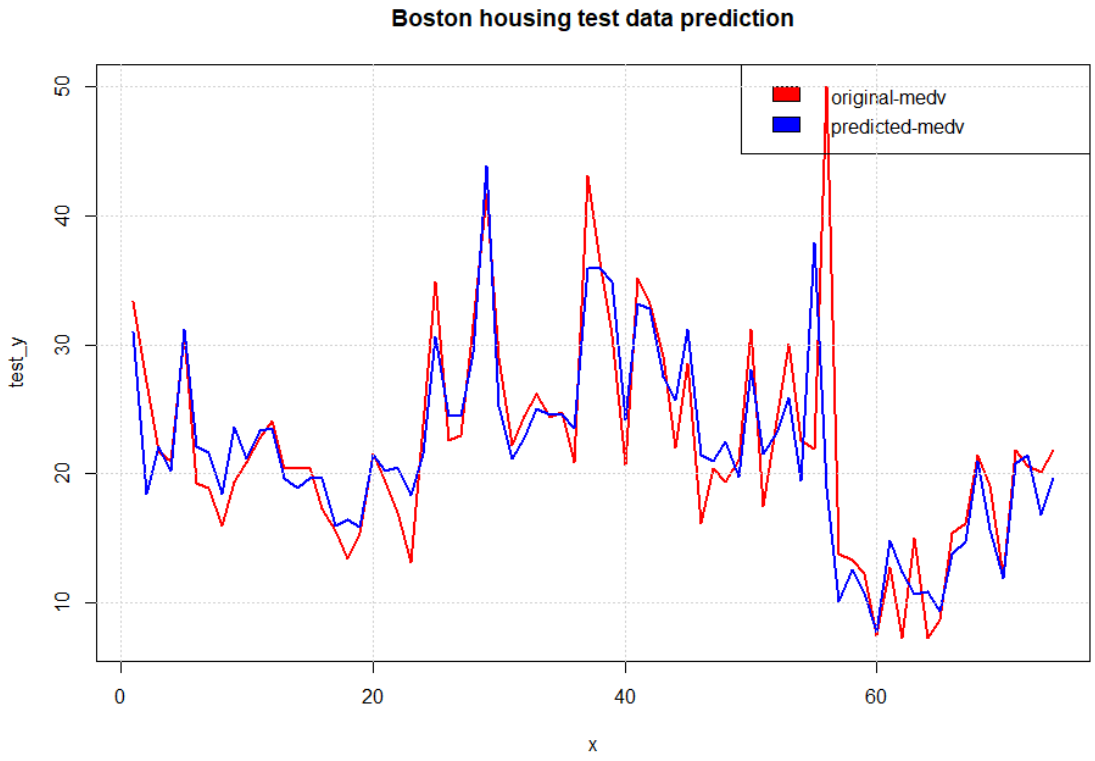
```
> print(data.frame(test_y, pred_y))

> mse = mean((test_y - pred_y)^2)
> mae = caret::MAE(test_y, pred_y)
> rmse = caret::RMSE(test_y, pred_y)
> cat("MSE: ", mse, "MAE: ", mae, " RMSE: ", rmse)
MSE:  24.13971 MAE:  2.775946  RMSE:  4.913218
```

	test_y	pred_y
1	33.4	30.98
2	27.1	18.46
3	21.7	22.14
4	21.0	20.26
5	30.8	31.20
6	19.3	22.12
7	18.9	21.60
8	16.0	18.38
9	19.4	23.58
10	20.9	21.14
11	22.8	23.38
12	24.1	23.48
13	20.4	19.62
14	20.4	18.88
15	20.5	19.64
16	17.3	19.64
17	15.6	15.92
18	13.4	16.40
19	15.4	15.88
20	21.5	21.40
21	19.4	20.18
22	17.0	20.54
23	13.1	18.28
24	23.8	21.52
25	34.9	30.66
26	22.6	24.50
27	23.0	24.50
28	31.6	29.36
29	41.7	43.88
30	29.0	25.22
31	22.2	21.12
32	24.5	22.80
33	26.2	25.08
34	24.4	24.54
35	24.8	24.54
36	20.9	23.56
37	43.1	35.94

Finalmente, visualizaremos la prueba original y los datos predichos en un gráfico.

```
> x = 1:length(test_y)
```

```
>
> plot(x, test_y, col = "red", type = "l", lwd=2,
+ main = "Boston housing test data prediction")
> lines(x, pred_y, col = "blue", lwd=2)
> legend("topright",  legend = c("original-medv", "predicted-medv"),
+ fill = c("red", "blue"), col = 2:3,  adj = c(0, 0.6))
> grid()
```



En este práctica, hemos aprendido cómo ajustar y predecir datos de regresión con la función 'knnreg' del paquete 'caret' en R.