Universidad Autónoma del Estado de México

Facultad de Ciencias

Diplomado en Machine Learning

Proyecto Final Módulo 1

Ejercicio 6

Autor: José de Jesús Rodríguez Barreto

Introducción

En este proyecto se ha puesto en práctica la regresión con k-vecinos, los datos se trataron de manera similar al primer proyecto en cuanto a las normalizaciones, la diferencia radico en que en lugar de datos faltantes teníamos factores dados por cadenas de texto, los cuales se convirtieron a factores dados por números, la variable objetivo por su parte estaba dada por valores numéricos en lugar de factores lo que nos llevó a ocupar una regresión en vez de una clasificación ordinaria.

Análisis de las variables

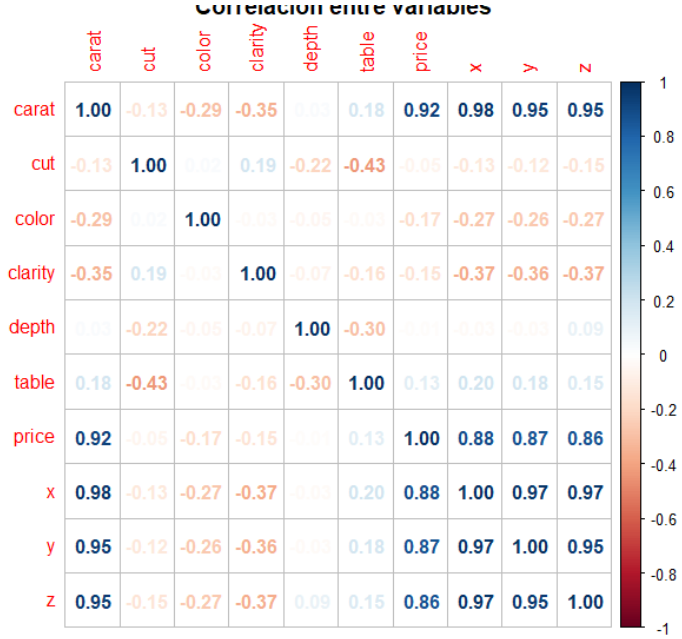
La información usada en este proyecto proviene de la página web kaggle la cual es una base de datos con atributos de cerca de 54,000 diamantes. Esta base de datos está conformada por 10 variables, 9 de las cuales son independientes entre sí y una dependiente que es la que se busca predecir, estas variables son:

- Price: Nuestra variable objetivo, son los precios de los diamantes en dólares estadounidenses.
- Carat: variable que contiene los kilates de cada diamante.
- Cut: calidad del corte del diamante.
- Color: color del diamante.
- Clarity: medida de la claridad del diamante.
- x, y, z: medidas del diamante
- depth: porcentaje de profundidad total.
- table: ancho de la parte superior del diamante en relación con el punto más ancho de este.

Preprocesamiento de datos.

Iniciamos el preprocesamiento tratando los datos que están conformados por cadenas de caracteres como es el caso de las variables cut, clarity y color, para esto vemos que la variable cut tiene 5 niveles los cuales son: {Fair, Good, Very Good, Premiun, Ideal}, como podemos notar en el orden es una medida del peor al mejor corte por lo que podemos asignarle a cada uno un número quedando los niveles en {1, 2, 3, 4, 5}. La siguiente variable es la de claridad, esta está conformada por los códigos de la claridad del diamante que son {I1, SI2, SI1, VS2, VS1, VVS2, VVS1, IF} los cuales igualmente están ordenados de peor a mejor claridad por lo que igualmente lo podemos recodificar a un número dejando {1, 2, 3, 4, 5, 6, 7, 8}, por último el color está clasificado con las letras {D, E, F, G, H ,I ,J} siendo la D el mejor color y la J el peor por lo que se consideró hacerlo en forma descendente para esta sección quedando {7, 6, 5, 4, 3, 2, 1}.

Después de esto se graficó la correlación de los datos para ver cuales estaban relacionados de manera que pudiéramos encontrar que datos podíamos omitir y cuales no, dándonos la siguiente gráfica:



La cual nos permite ver que los datos de las medidas del diamante están muy relacionados entre sí por lo que podríamos descartar 2 de ellos y quedarnos solamente con uno, en nuestro caso nos quedamos con la z.

Descripción del Algoritmo

El método por emplear fue el de regresión con k-vecinos más cercanos ponderados el cual consiste en buscar mediante la proximidad entre datos la asociación con los demás. Este método es más efectivo que una clasificación de k vecinos más cercanos por que los precios que es nuestra variable objetivo son continuos y no factores como podría darse para una clasificación. Es un método en el que una nueva observación se coloca en la clase de las observaciones del conjunto de entrenamiento más cercanas respecto a las variables empleadas, para esto se emplea el concepto de distancia. A parte del concepto de distancias se contempla que las observaciones más cercanas tengan un mayor peso en la decisión que aquellos vecinos más alejados de la nueva observación, para esto se usa una función llamada kernel que evaluara la distancia, la cual puede ser rectangular, triangular, gaussiano, inverso entre otros. Para este proyecto usaremos los mencionados antes.

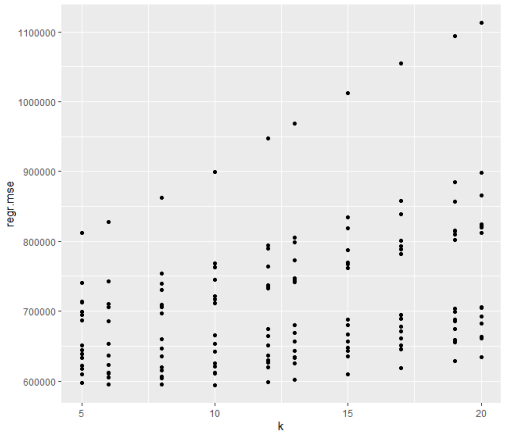
Construcción del modelo.

El modelo fue construido siguiendo la regresión de k-vecinos ponderados auto calibrados, para esto se le pidió que buscara entre 5 y 20 vecinos más cercanos el número optimo, el kernel que daría los pesos a buscar fue dado a elegir entre rectangular, triangular, gaussiano e inverso, y distancias entre 1 y 3, se usó una validación cruzada simple con 10 divisiones que al ver las gráficas se puede ver que

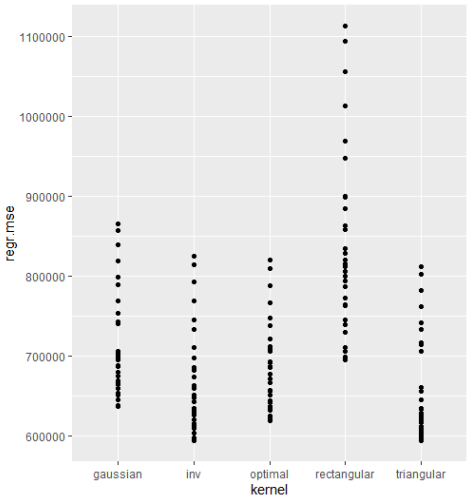
el mejor modelo esta dado por un k de 10 y un kernel triangular, como se apreciara en la siguiente imagen.

```
> instancia_busqueda$archive$best()
    k distance     kernel regr.mse runtime_learners
1: 10        2 triangular 594111.8           77.89
```

Sin embargo, en las siguientes graficas se pueden ver de una manera más clara estos resultados.



En la gráfica de la izquierda se puede apreciar como en un k de 10 se obtiene un error cuadrático medio mucho menor que con otros k.



En esta grafica se puede apreciar como el kernel triangular consiguió el error medio más bajo siendo mejor que otros kernel.

A continuación se buscó que mediante una validación cruzada anidada conseguir una estimación más confiable del desempeño predictivo del modelo para esto en la validación externa se usó una de retención con un ratio de 0.7 y para la interna se usó una validación cruzada simple con 10 divisiones, ya con esto hecho se hizo un objeto que sirviera para llevar a cabo el proceso de selección del modelo y se combinaron las dos validaciones con la tarea, para con el modelo optimo poder entrenar utilizando todo el conjunto de datos, antes de esto se intentó una validación

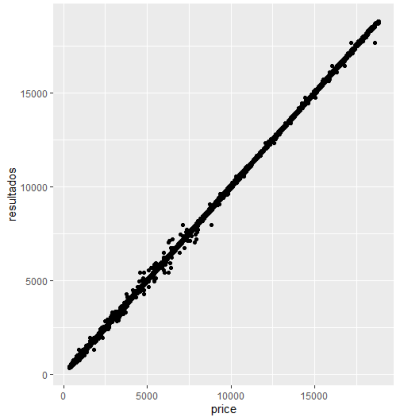
cruzada repetida pero debido a la gran cantidad de datos el hardware no pudo realizar la tarea por lo que al final se optó por validaciones simples.

Análisis de resultados.

Al final del entrenamiento, se hizo una predicción sobre todo el conjunto de datos la cual nos arrojó los siguientes resultados:

```
> pred
<PredictionRegr> for 53920 observations:
    row_ids truth  response
          1   326  326.0009
          2   326  326.0016
          3   327  327.0026
---
      53918  2757 2756.9991
      53919  2757 2757.0001
      53920  2757 2757.0016
> head(diamantes)
  carat cut color clarity depth table price    z resultados
1  0.23   5     6       2  61.5    55   326 2.43   326.0009
2  0.21   4     6       3  59.8    61   326 2.31   326.0016
3  0.23   2     6       5  56.9    65   327 2.31   327.0026
4  0.29   4     2       4  62.4    58   334 2.63   381.5005
5  0.31   2     1       2  63.3    58   335 2.75   335.0013
6  0.24   3     1       6  62.8    57   336 2.48   336.0025
```

Donde podemos ver que las predicciones tienen unos datos que varían del dato real por un orden ínfimo por lo que la predicción es bastante acertada. Además, al graficar las respuestas contra los precios podemos ver que esta es prácticamente una recta de 45° dándonos a entender que la gran mayoría de los precios están alineados con los resultados dados por el modelo.



Conclusiones

En conclusión, para esta práctica el tratamiento de datos fue bastante más sencillo el problema principal radico en la programación de la regresión pues usa elemento un poco diferentes a los de la clasificación por lo que la interpretación se siente diferente, además la matriz de confusión en la clasificación ayuda mucho a la interpretación de los datos de una manera más sencilla y aunque es tan simple como ver las predicciones y darte cuenta de que tan bueno es el modelo a partir de las predicciones. Este tipo de aprendizaje tiene aplicaciones para todos aquellos problemas que ocupen predecir datos continuos y es muy importante practicar en

conjunto este método junto con el de clasificación para obtener la habilidad de distinguir desde el principio cual problema requiere que método.

Anexo: código

```
library(tidyverse)
library(skimr)
library(paradox)
library(mlr3learners)
library(mlr3measures)
library(mlr3tuning)
library(kknn)
library(corrplot)
diamantes <- read.csv("diamonds.csv")
x <- cor(diamantes, method = "pearson")
corrplot(x, method = "number", title = "Correlacion entre variables")

diamantes <- mutate_at(diamantes,c("price"), as.numeric)
str(diamantes)
diamantes <- mutate_at(diamantes, c("cut", "color", "clarity"), as.factor)
diamantes <- subset(diamantes, select = -c(x, y))
diamantes <- diamantes[!(diamantes$z == 0),]
tarea<-TaskRegr$new(id="regresion", backend=diamantes, target="price")
regr.knn<-lrn("regr.kknn")
regr.knn$param_set
espacio_soluciones<-ParamSet$new(list(ParamInt$new("k",5,20),
                                      ParamInt$new("distance", 1, 3),
                                      ParamFct$new("kernel",c("rectangular","triangular","gaussian","inv","optimal"))))
tipo_busqueda<-mlr3tuning::tnr("grid_search")
val_cruzada<-rsmp("cv",folds=10)
instancia_busqueda=TuningInstanceSingleCrit$new(task=tarea,
                                                learner = regr.knn,
                                                resampling=val_cruzada,
                                                measure = msr("regr.mse"),
                                                search_space = espacio_soluciones,
                                                terminator=trm("none"))
future::plan("multisession")
tipo_busqueda$optimize(instancia_busqueda)
instancia_busqueda$archive$best()
ggplot(instancia_busqueda$archive$data, aes(x=k,y=regr.mse)) + geom_point()
ggplot(instancia_busqueda$archive$data, aes(x=kernel,y=regr.mse)) + geom_point()
vc_externa<-rsmp("holdout", ratio=0.7)
vc_interna<-rsmp("cv", folds=10)
optim.learner<-AutoTuner$new(learner=regr.knn,
                             vc_interna,msr("regr.mse"),
                             espacio_soluciones,
                             terminator = trm("none"),
                             tuner=tipo_busqueda)
future::plan("multisession")
resultados<-resample(tarea,optim.learner,vc_externa)
resultados$aggregate(measures=msr("regr.mse"))
modelo_final<-optim.learner$train(tarea)
pred<-modelo_final$predict_newdata(diamantes)
resultados <- pred$response
diamantes <- cbind(diamantes, resultados)

ggplot(diamantes, aes (x=price,y=resultados)) + geom_point()

```