MySQL

Importamos la base de datos de MySql a R.

```
library(RODBC)
```

```
## Warning: package 'RODBC' was built under R version 4.1.1
```

```
library(RMySQL)
```

```
## Warning: package 'RMySQL' was built under R version 4.1.1
```

```
## Loading required package: DBI
```

```
mysqlconnection = dbConnect(RMySQL::MySQL(),
                            dbname='exams',
                            host='localhost',
                            port=3306,
                            user='root',
                            password='turone97')

dbListTables(mysqlconnection)
```

```
## [1] "exmas"
```

```
dbSendQuery(mysqlconnection, "SET GLOBAL local_infile = true;") # <--- Added this
```

```
## <MySQLResult:0,0,1>
```

```
#dbWriteTable(mysqlconnection, name= "exams", value= df, append= TRUE, temporary= FALSE)
#dbDisconnect(mysqlconnection)
```

Verificamos la existencia de nuestra tabla de datos.

```
SELECT * FROM exmas;
```

Table 1: Displaying records 1 - 10

gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Nota_media
male	group A	high school	standard	completed	67	67	63	
female	group D	some high school	free/reduced	none	40	59	55	
male	group E	some college	free/reduced	none	59	60	50	
male	group B	high school	standard	none	77	78	68	
male	group E	associate's degree	standard	completed	78	73	68	
female	group D	high school	standard	none	63	77	76	
female	group A	bachelor's degree	standard	none	62	59	63	
male	group E	some college	standard	completed	93	88	84	
male	group D	high school	standard	none	63	56	65	
male	group C	some college	free/reduced	none	47	42	45	

Comenzamos creando una tabla denominada 'Nota_nivel_educacion' de modo a poder ver las notas de los alumnos ordenados según el grado académico que poseen.

```
create table Nota_nivel_educacion
as select `parental level of education`, avg(`math score`) as Nota_media_mates, avg(`reading score`) as
from exmas
group by `parental level of education`;
```

```
select * from Nota_nivel_educacion;
```

Table 2: 6 records

parental level of education	Nota_media_mates	Nota_media_lectura	Nota_media_escritura
high school	65.2079	67.4010	64.8465
some high school	60.7016	64.4084	62.5393
some college	65.2973	68.0450	66.7342
associate's degree	69.5369	70.9852	70.1133
bachelor's degree	71.4911	74.0089	74.4107
master's degree	71.5857	75.4286	75.8857

A continuación realizamos la Nota media

```
select `parental level of education`, sum(Nota_media_mates + Nota_media_lectura + Nota_media_escritura)/
 from Nota_nivel_educacion
 group by `parental level of education`;
```

Table 3: 6 records

parental level of education	Nota_media
high school	65.81847
some high school	62.54977
some college	66.69217
associate's degree	70.21180
bachelor's degree	73.30357
master's degree	74.30000

Observamos como la nota media aumenta conforme el diploma adquirido requiere una edad más elevada. Podríamos decir que hay una correlación entre madurez del individuo y el estudio que efectúa.

Vamos a ver si hay una correlación entre los alumnos que han realizado tests durante todo el curso y los que no lo han hecho.

```
create table eval_continua
as select `test preparation course`, avg(`math score`) as Nota_media_mates, avg(`reading score`) as Nota
from exmas
group by `test preparation course`;
```

```
select * from eval_continua;
```

Table 4: 2 records

test preparation course	Nota_media_mates	Nota_media_lectura	Nota_media_escritura
completed	69.6866	74.0896	74.6716
none	64.7383	66.4391	64.2451

Efectivamente, podemos ver a simple vista cómo los alumnos que han realizado una evaluación continua consiguen tener una nota media mayor que los que no lo han hecho.

Con el objetivo de añadir un poquito más de 'chicha' al asunto, vamos a ver qué sucede si ordenamos por géneros esta tabla.

```
create table Generos
as select gender, avg(`math score`) as Nota_media_mates, avg(`reading score`) as Nota_media_lectura, avg
from exmas
group by gender;
```

```
select * from Generos;
```

Table 5: 2 records

gender	Nota_media_mates	Nota_media_lectura	Nota_media_escritura
male	69.3849	66.3056	64.0290
female	63.1967	71.8882	71.7081

Obtenemos una nota media mayor para las mujeres que para los hombres. Veamos si hay alguna correlación con los test realizados durante todo el curso.

```
create table hombres
as select gender, `test preparation course` from exmas where gender = 'male';
```

```
select `test preparation course`, count(`test preparation course`) c
from hombres
group by `test preparation course` having c > 1;
```

Table 6: 2 records

test preparation course	c
completed	175
none	342

Hay 175 hombres que han completado la evaluación continua y 342 que no, mientras que para las mujeres:

```
create table mujeres
as select gender, `test preparation course` from exmas where gender = 'female';
```

```
select `test preparation course`, count(`test preparation course`) c
from mujeres
group by `test preparation course` having c > 1;
```

Table 7: 2 records

test preparation course	c
none	323
completed	160

En el caso de las mujeres; 160 han completado la evaluación continua y 323 no. Realizando un cácluclo rápido, el 51,1% de los hombres han realizado la evaluacion continua y el 49.5% la han realizado las mujeres.

De MySQL a Python

Para terminar con un mejor sabor de boca, usaremos lo aprendido anteriormente para predecir las notas medias de los alumnos según el grado de educación que poseen, el género, etc…

Empezamos filtrando y ordenando los datos, así como convertir las columnas con cadena de texto en números enteros. En machine learning es imprescindible convertir las cadenas de texto en datos numéricos para poder ejecutar un modelo predictivo.

Con los siguientes comandos podemos sustituir las cadenas de texto por números enteros.

```
df['gender'].replace(['male', 'female'], [0,1], inplace = True)
df['test preparation course'].replace(['completed', 'none'], [2,3], inplace = True)
df['Apreciacion'].replace(['Suspenso', 'Aprobado', 'Notable', 'Sobresaliente'], [4,5,6,7], inplace = Tru

df['parental level of education'].replace(['high school', 'some high school', 'some college', "associat

import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split



y = df['Nota_media']

X = df.drop(['race/ethnicity', 'lunch', 'math score', 'reading score',
             'writing score', 'Nota_media'], axis = 1 )


xtrain, xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.1, random_state = 5)
```

Los dos modelos que usaremos son los siguientes:

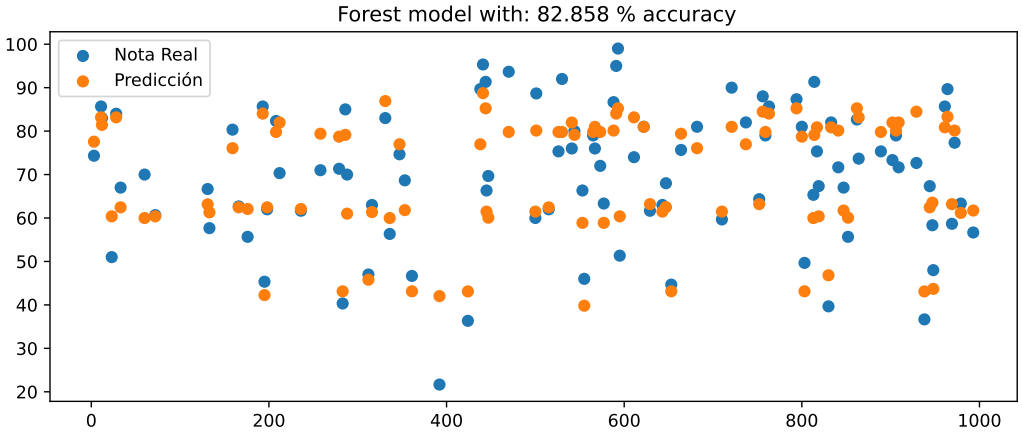
```
model_R = Ridge()
model_R.fit(xtrain, ytrain)
```

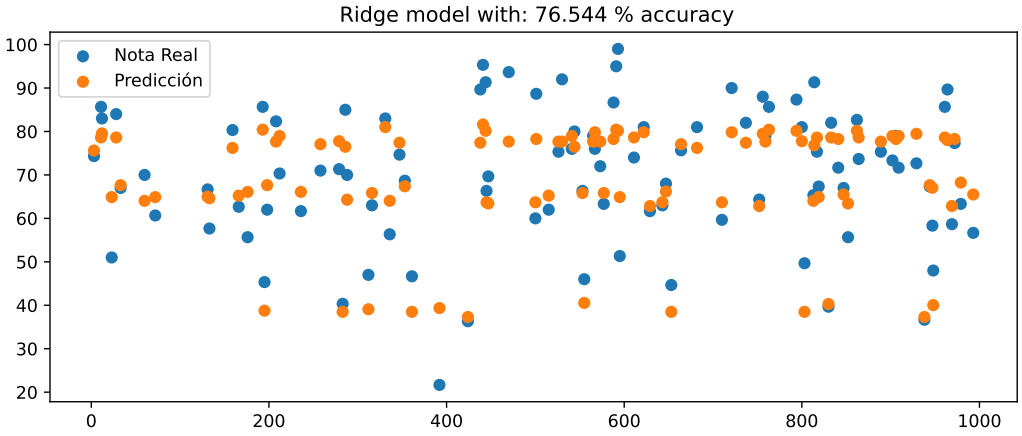
```
## Ridge()
```

```
model_F = RandomForestRegressor()
model_F.fit(xtrain, ytrain)
```

```
## RandomForestRegressor()
```

```
plt.figure(figsize=(10, 10))
plt.subplot(2,1,2)
plt.scatter(ytest.index, ytest, label = 'Nota Real')
plt.scatter(ytest.index, model_R.predict(xtest), label = 'Predicción')
plt.title(f' Ridge model with: {round(model_R.score(xtrain, ytrain)*100, 3)} % accuracy')
plt.legend()

plt.subplot(2,1,1)
plt.scatter(ytest.index, ytest, label = 'Nota Real')
plt.scatter(ytest.index, model_F.predict(xtest), label = 'Predicción')
plt.title(f' Forest model with: {round(model_F.score(xtrain, ytrain)*100, 3)} % accuracy')
plt.legend()

plt.subplots_adjust(hspace=0.6)
plt.show()
```





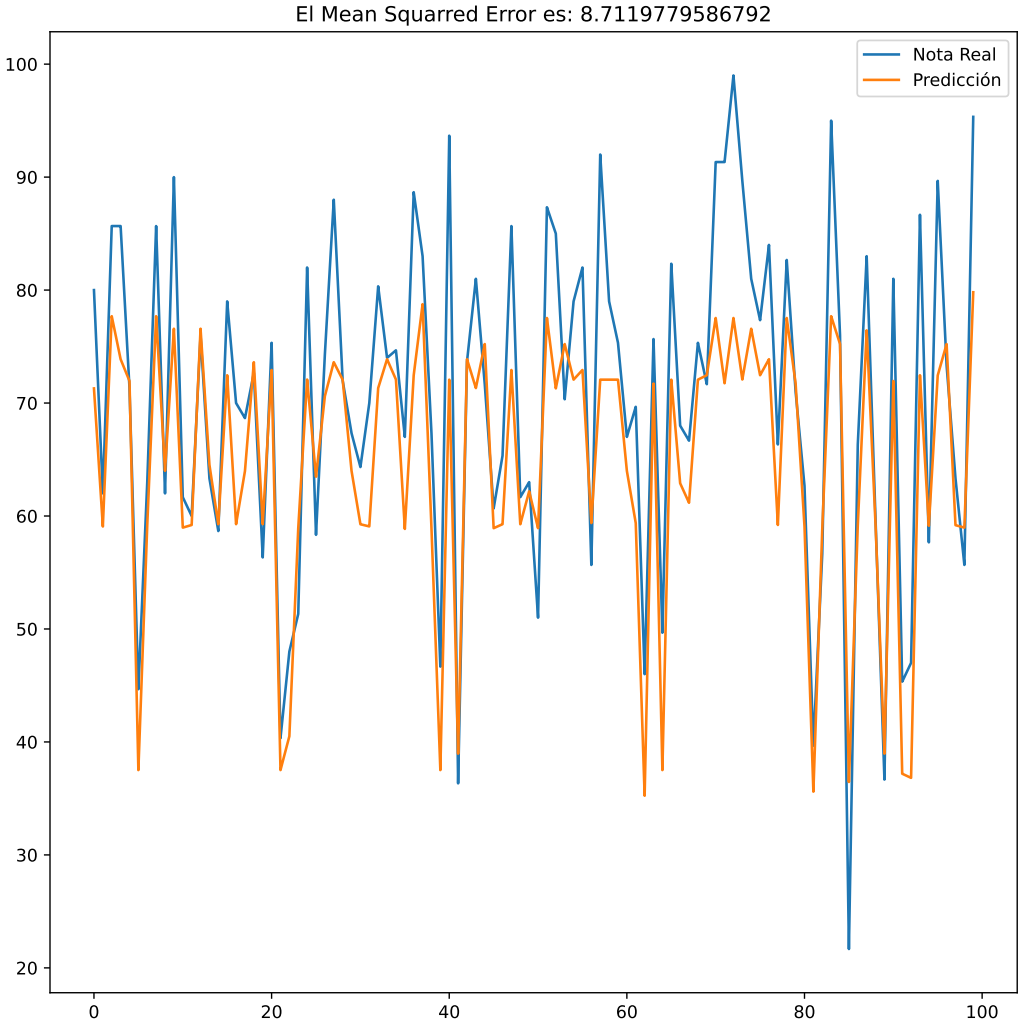
Destacamos un ajuste mejor ejecutado para Random Forest Regressor que para el modelo Ridge. Aún así obtenemos unos resultados bastante óptimos.

Si quisiérmos realizar un ajuste mediante redes neuronales obtendríamos el siguiente resultado a corde al modelo que sigue:

```
import tensorflow as tf
from tensorflow.keras.layers import Dense, Dropout, LSTM, InputLayer
from tensorflow.keras.models import Sequential


model = Sequential()

model.add(InputLayer((4,1)))
```

```
model.add(LSTM(64,'relu'))

model.add(Dense(32, activation = 'relu'))
model.add(Dropout(0.2))

model.add(Dense(64, activation = 'relu'))
model.add(Dropout(0.2))

model.add(Dense(1, activation = 'linear'))

model.compile(optimizer = 'Adam', loss = 'mse', metrics = ['RootMeanSquaredError'] )
```

```
scores = model.evaluate(data_test, data_pred_test, verbose=0)

plt.plot(data_pred_test, label = 'Nota Real')
plt.plot(model.predict(data_test), label = 'Predicción')
plt.title(f'El Mean Squarred Error es: {scores[1]}')
plt.legend()
plt.show()
```



Mediante redes neruonales observamos un ajuste, de la predicción a la realidad, notorio.

Modelo de Clasificación

Terminemos prediciendo algo un tanto más interesante. **¿Podríamos predecir qué diploma posee el alumno según las notas que haay obtenido?**

Importamos los modelos pertinentes a, esta vez, una clasificación.

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
```

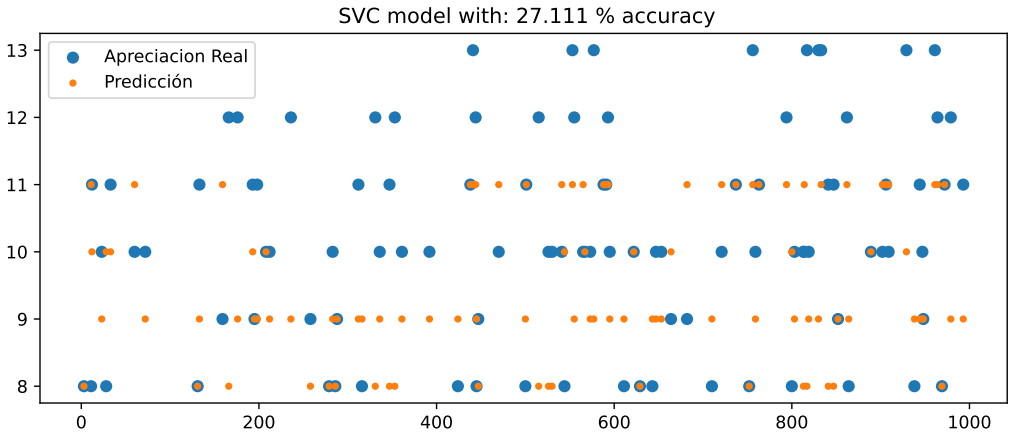
```
model_KN = KNeighborsClassifier(algorithm = 'brute')
model_KN.fit(xtrain, ytrain)
```

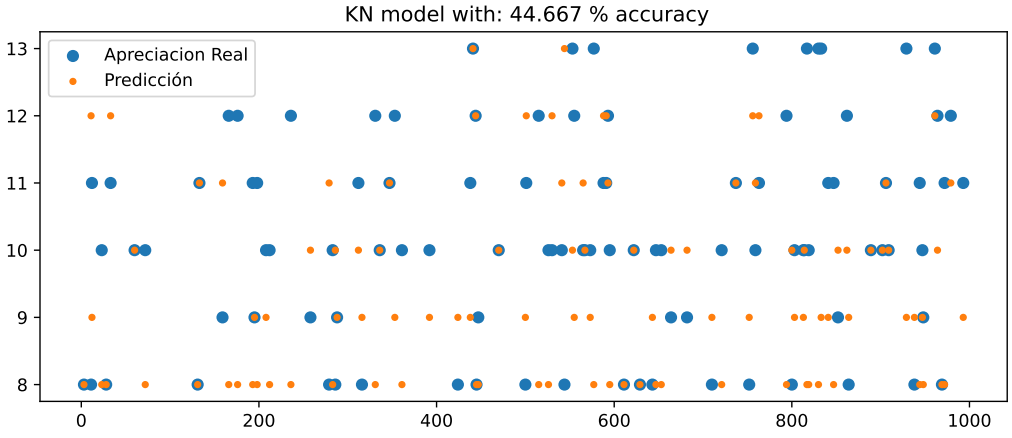
```
## KNeighborsClassifier(algorithm='brute')
```

```
model_SVC = SVC(decision_function_shape= 'ovo', kernel='poly')
model_SVC.fit(xtrain, ytrain)
```

```
## SVC(decision_function_shape='ovo', kernel='poly')
```

```
plt.show()
```





Concluimos con un resultado un tanto catastrófico de parte del modelo SVC, sin embargo para el modelo de KNeighbors parece presentar resultados no tan desastrosos. Si dispusiéramos de una tabla con una mayor cantidad de datos, estoy seguro que, obtendríamos un resultado más satisfactorio. Aún así, sigue siendo un

porcentaje acceptable para una proyecto de ‘exhibición’.