Trabajo Práctico
Inferencia Estadística y Reconocimiento de Patrones

Docente: *María Florencia Statti*

Estudiantes: *Arias Valentín Santino*
La Grutta Dario

Ciclo Lectivo : *2025 2do Cuatrimestre*

Índice

1. Regresiones

Implementación en lenguaje R de los modelos de regresión con el dataset de vinos

- Regresión Lineal Múltiple
- Regresión Ridge
- Regresión LASSO

1.1 Regresión Lineal Múltiple

Para poder comenzar a analizar nuestro modelo y su confiabilidad, ajustamos el modelo de regresión lineal múltiple con los datos proporcionados, e imprimimos los resultados estadísticos obtenidos:

```
Residuals:
     Min       1Q   Median       3Q      Max
-2.68911 -0.36652 -0.04699  0.45202  2.02498

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           2.197e+01  2.119e+01   1.036   0.3002
fixed.acidity         2.499e-02  2.595e-02   0.963   0.3357
volatile.acidity     -1.084e+00  1.211e-01  -8.948  < 2e-16 ***
citric.acid          -1.826e-01  1.472e-01  -1.240   0.2150
residual.sugar        1.633e-02  1.500e-02   1.089   0.2765
chlorides            -1.874e+00  4.193e-01  -4.470 8.37e-06 ***
free.sulfur.dioxide   4.361e-03  2.171e-03   2.009   0.0447 *
total.sulfur.dioxide -3.265e-03  7.287e-04  -4.480 8.00e-06 ***
density              -1.788e+01  2.163e+01  -0.827   0.4086
pH                   -4.137e-01  1.916e-01  -2.159   0.0310 *
sulphates             9.163e-01  1.143e-01   8.014 2.13e-15 ***
alcohol               2.762e-01  2.648e-02  10.429  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.648 on 1587 degrees of freedom
Multiple R-squared:  0.3606,    Adjusted R-squared:  0.3561
F-statistic: 81.35 on 11 and 1587 DF,  p-value: < 2.2e-16
```

1.1.1 Significatividad Estadística del modelo y las covariables

Notamos principalmente que el modelo implementado es estadísticamente significativo, ya que logra obtener un p-valor general muy por debajo del 0,05 que se necesita como mínimo para considerar su significatividad. Por otro lado, si analizamos las variables explicativas del modelo, podemos interpretar que solamente resultaron estadísticamente significativas aquellas que fueron remarcadas, debido a que, como se observa, obtuvieron un p-valor menor a 0,05.

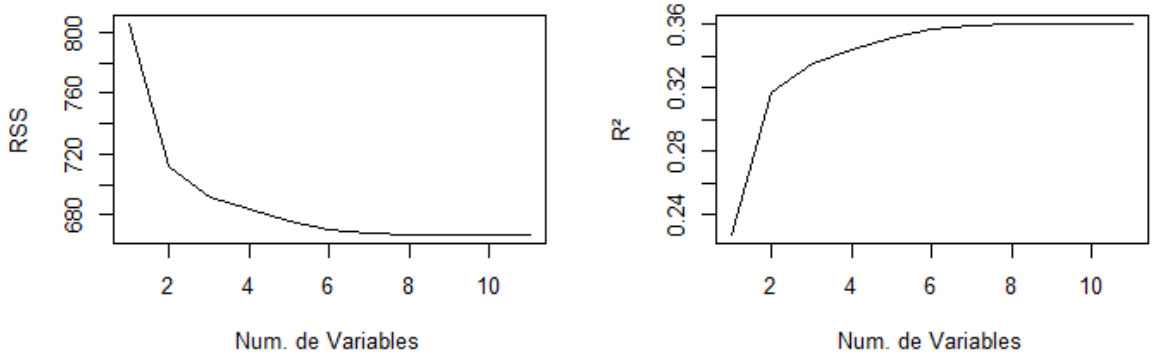
1.1.2 Métricas del modelo

Observando la anterior imagen, es posible notar que la variabilidad de la variable respuesta logra ser explicada aproximadamente en un 36%, basado en el resultado de la métrica R^2. A continuación se imprime el valor de dicha métrica a medida que se añaden cada una de las variables explicativas al modelo:

```
> bestmodelo1$rsq
 [1] 0.2267344 0.3170024 0.3358973 0.3437824 0.3514942 0.3571736 0.3594709 0.35992
65 0.3601728
[10] 0.3602764 0.3605517
```

Observando la evolución de esta métrica, podemos notar que presenta una mejora significativa con la adición de las primeras siete covariables, para después continuar mejorando, pero con diferencias cada vez más pequeñas, lo que permite inferir que no se obtienen mejoras que valgan la pena por añadir complejidad al modelo.

A continuación, se adjuntan gráficos complementarios que refuerzan la propuesta. Por un lado, veremos la curva decreciente de la suma de los residuos al cuadrado (RSS), según la cantidad de variables (nótese el gráfico izquierdo), mientras que por el otro lado, tenemos la representación de la evolución mencionada anteriormente del R^2, en función de la cantidad de variables.



1.1.3 Modelo final

Basado en los análisis expuestos sobre las métricas del modelo, se decidió ajustar nuevamente, excluyendo las covariables que no fueron consideradas estadísticamente significativas. A continuación se imprimen los resultados de dicho ajuste:

```
Call:
lm(formula = quality ~ ., data = train_set)

Residuals:
     Min       1Q   Median       3Q      Max
-2.19067 -0.36548 -0.06067  0.46638  2.06107

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           4.5208564  0.4599904   9.828  < 2e-16 ***
volatile.acidity     -1.0571807  0.1106889  -9.551  < 2e-16 ***
chlorides            -1.8491373  0.4505973  -4.104 4.32e-05 ***
free.sulfur.dioxide   0.0057709  0.0024139   2.391 0.016959 *
total.sulfur.dioxide -0.0034126  0.0007958  -4.288 1.94e-05 ***
pH                   -0.5014415  0.1322895  -3.790 0.000157 ***
sulphates             0.9078255  0.1231132   7.374 2.98e-13 ***
alcohol               0.2846247  0.0190692  14.926  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6483 on 1273 degrees of freedom
Multiple R-squared:  0.3616,    Adjusted R-squared:  0.3581
F-statistic:   103 on 7 and 1273 DF,  p-value: < 2.2e-16
```

Como es posible observar, no existe demasiada diferencia entre este modelo y el inicial, por lo tanto, decidimos quedarnos con el último, debido a que devuelve los mismos resultados y, además, aporta una mayor simplicidad.

De este punto en adelante, se va a trabajar con este mismo conjunto de covariables.

1.2 Regresión Ridge

Para la regresión ridge es necesario determinar el valor óptimo del parámetro lambda. Se optó por utilizar un método de selección mediante **validación cruzada**, específicamente, k-folds (con k=10), mediante la función en R cv.glmnet().
La búsqueda del lambda que minimiza el MSE, según la implementación antes mencionada, dió el siguiente resultado:

```
========================================
 El mejor Lambda para Ridge es:   0.037976
========================================
```

1.3 Regresión LASSO

Al igual que con el método de regresión ridge, se aplicó lo equivalente para el modelo de regresión lasso. El resultado se muestra a continuación:

```
========================================
 El mejor Lambda para Lasso es:   0.000898
========================================
```

1.4 Comparación de Modelos

Para lograr una correcta evaluación y posterior comparación de cada modelo, se aplicó el método de validación cruzada de separación train/test, realizando el ajuste de cada modelo con el conjunto train (80% de los datos) y probando sus resultados con el conjunto test (20% restante).
Las métricas resultantes de cada modelo luego del ajuste antes mencionado se muestran a continuación:

```
==================== COMPARACIÓN DE MODELOS ====================

Modelo                          MSE       RMSE        R²
------------------------------------------------------------

Regresión Lineal              0.418222   0.646701   0.361629
Ridge                         0.418396   0.646835   0.347644
Lasso                         0.418168   0.646659   0.347999
============================================================
```

Métricas utilizadas:

- MSE (Error Cuadrático Medio)
- RMSE (Raíz del Error Cuadrático Medio)
- R^2 (Coeficiente de Determinación)

1.5 Elección del mejor modelo

En base a los resultados obtenidos, el espíritu es buscar el modelo que minimice el error (tanto MSE como RMSE), pero también el que mejor explique la variabilidad de y. Con esto en mente, el modelo que mejor logra equilibrar ambas cuestiones, terminó siendo la **regresión lineal múltiple.**
Además, brinda una menor complejidad y una interpretación más sencilla de sus resultados.

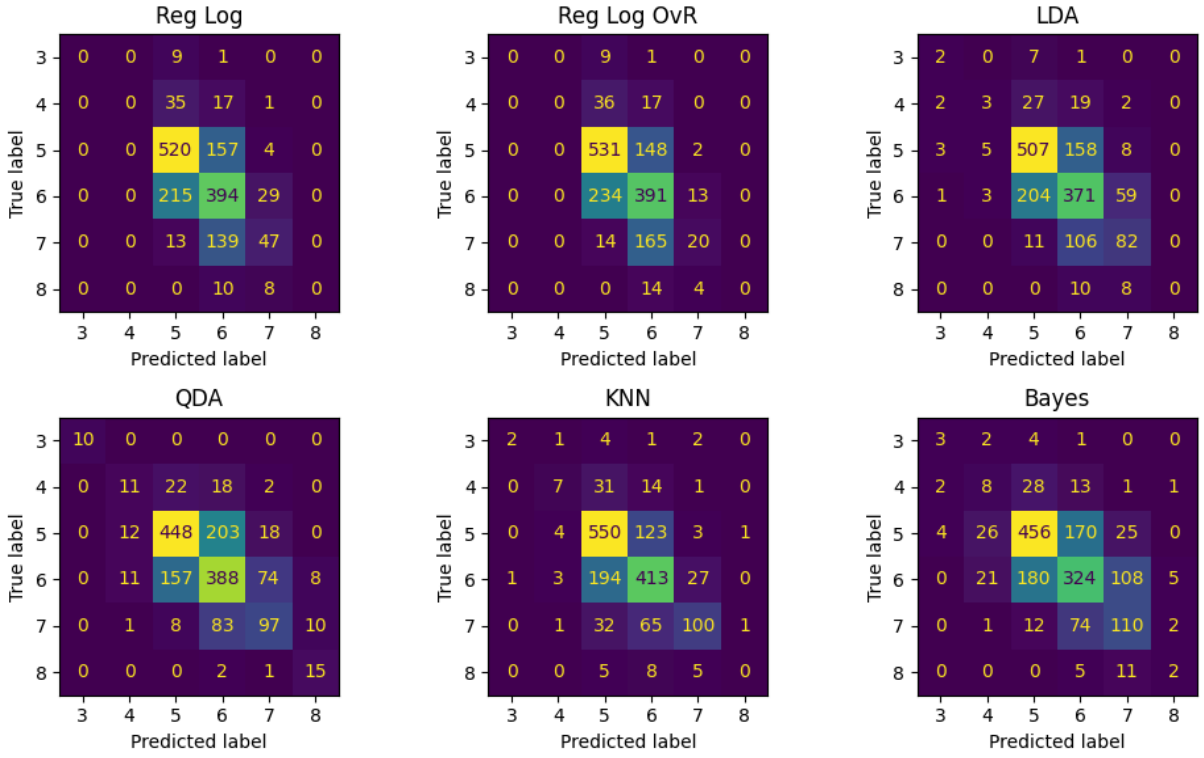
2. Métodos de Clasificación

Implementación en lenguaje Python de los modelos de clasificación:

- Regresión Logística y Regresión Logística OvR
- K Nearest Neighbors (KNN)
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- Bayes

2.1 Comparación General

Con el objetivo de brindar un vistazo general a cada modelo de clasificación implementado, cada modelo fue ajustado y testeado con todos los datos, y se contrastaron los resultados con los valores reales mediante matrices de confusión. A continuación se muestra dicha comparación, además de una tabla con la métrica accuracy (o precisión) de cada predicción realizada por los modelos.




```
Accuracy:
Reg Log: 0.6010
Reg Log OvR: 0.5891
LDA: 0.6035
QDA: 0.6060
KNN: 0.6704
Bayes: 0.5647
```

Aclaraciones:

Reg Log y Reg Log OvR utilizan la misma implementación de sklearn, pero cambiando el parámetro 'multi_class'. La primera utiliza el valor 'multinomial' y la segunda (obviamente) 'ovr'.

KNN utiliza sus parámetros definidos por defecto, es decir 'metric' = 'euclidean' y 'n_neighbors' = 5.

Bayes se refiere a la implementación de sklearn GaussianNB.

2.2 Cross Validation: K Nearest Neighbors

El único modelo de clasificación con un hiperparámetro ajustable (manteniendo lo visto en clase) es el KNN, por lo tanto, se implementó un método de validación cruzada, descrito a continuación.

Tomando como punto de partida a X con únicamente las variables estadísticamente significativas y escalando los datos en el proceso

In [12]:
```
X_es = X[["volatile acidity", "chlorides","free sulfur dioxide",
          "total sulfur dioxide", "pH", "sulphates", "alcohol"]]

X_train, X_test, y_train, y_test = train_test_split(X_es, y, test_size= 0.2, random_state= 42)
```

In [13]:
```
# Grilla de parámetros a evaluar
parametros_grid = {
       "knn__n_neighbors": list(range(1, 31)),
        "knn__metric": ["euclidean", "manhattan"]
        }

from sklearn.pipeline import Pipeline
# Implementación mediante pipelines
pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("knn", KNN())
])
```

Luego de definir los parámetros a optimizar y modelar el pipeline, se ejecuta este último y se imprime el accuracy del modelo ajustado con los parámetros que maximizan la efectividad del mismo.

```
best_knn = GridSearchCV(pipe, parametros_grid, cv = 5, scoring= "accuracy")
best_knn.fit(X_train,y_train)
y_pred = best_knn.predict(X_test)
print("Accuracy obtenido: ", accuracy_score(y_test, y_pred))
```

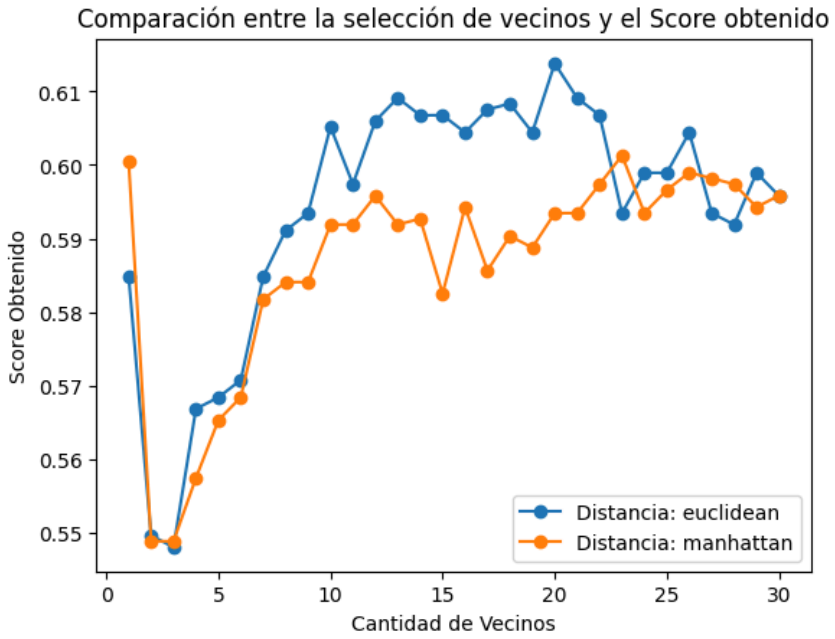
```
Accuracy obtenido:  0.59375
```

A continuación, los parámetros obtenidos:

In [15]:
```
best_knn.best_params_
```

Out[15]:
```
{'knn__metric': 'euclidean', 'knn__n_neighbors': 20}
```

Por último, se muestra un gráfico donde es posible observar la evolución del score (accuracy) para cada k definido (de 1 a 30)



2.3 Cross Validation: Comparación de Modelos

Se implementó el método de validación cruzada **k-folds estratificados**, que se basa en separar el dataset en conjunto de train y test de forma aleatoria, k cantidad de veces. Para poder ajustar y medir el score de cada modelo dentro de cada 'fold', se implementó la función 'CrossValidate' definida a continuación:

```
# Definimos la función para encontrar el mejor modelo
def CrossValidate(title, modelos, best_scores, X, y):
    # Inicializamos un DataFrame para guardar los scores
    scores = pd.DataFrame(columns=(list(modelos.keys())))
    # Inicializamos el K-Fold Estratificado
    kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
    # Iteramos sobre cada fold
    for i, (train, test) in enumerate(kf.split(X, y)):
        # Iteramos sobre cada modelo
        for model in scores.columns:
            # Entrenamos el modelo y guardamos el score
            modelos[model].fit(X.iloc[train], y.iloc[train])
            scores.loc[i, model] = modelos[model].score(X.iloc[test], y.iloc[test])

    # Calculamos y guardamos el Accuracy promedio para cada modelo
    scores.loc['Promedio'] = np.mean(scores, axis=0)
    print(scores)
    # El modelo con mejor Accuracy medio
    best_acc = scores.loc['Promedio'].max()
    best_model = scores.loc['Promedio'].idxmax()
    best_scores[title] = (best_model, best_acc)
    print(f'Mejor modelo: {best_model}. Accuracy = {best_acc:.5f}')
```

La idea es que ajuste cada modelo, calcule el score de sus predicciones y lo guarde en una tabla, para después calcular el promedio de los cinco valores (uno para cada fold) de cada uno de los seis modelos. La función fue ejecutada 4 veces, con distintos X.

- Primero, con el dataset **inalterado**:

```
# Inicializamos otro diccionario para guardar los mejores scores
best_scores = {}

CrossValidate('Todas las Covariables', modelos, best_scores, X, y)
```

	Reg Log	Reg Log OvR	LDA	QDA	KNN	Bayes
0	0.603125	0.596875	0.6125	0.58125	0.5375	0.575
1	0.578125	0.546875	0.590625	0.590625	0.490625	0.54375
2	0.6125	0.61875	0.590625	0.5875	0.51875	0.528125
3	0.6	0.578125	0.60625	0.5125	0.475	0.56875
4	0.598746	0.586207	0.592476	0.567398	0.482759	0.579937
Promedio	0.598499	0.585366	0.598495	0.567855	0.500927	0.559112

Mejor modelo: Reg Log. Accuracy = 0.59850

- Luego, con el mismo conjunto, pero **estandarizando** la escala de las covariables:

```
CrossValidate('Todas Estandarizadas', modelos, best_scores, X, y)
```

	Reg Log	Reg Log OvR	LDA	QDA	KNN	Bayes
0	0.6	0.59375	0.6125	0.58125	0.5375	0.571875
1	0.58125	0.565625	0.590625	0.590625	0.565625	0.540625
2	0.615625	0.615625	0.590625	0.5875	0.565625	0.515625
3	0.6125	0.58125	0.60625	0.5125	0.590625	0.55625
4	0.586207	0.579937	0.592476	0.567398	0.548589	0.561129
Promedio	0.599116	0.587237	0.598495	0.567855	0.561593	0.549101

Mejor modelo: Reg Log. Accuracy = 0.59912

- Luego, con las covariables **estadísticamente significativas** inalteradas(1.1.1):

```
CrossValidate('Est. Significativas', modelos, best_scores, X_es, y)
```

	Reg Log	Reg Log OvR	LDA	QDA	KNN	Bayes
0	0.615625	0.584375	0.596875	0.553125	0.565625	0.584375
1	0.559375	0.55	0.59375	0.571875	0.51875	0.590625
2	0.60625	0.60625	0.60625	0.56875	0.553125	0.58125
3	0.584375	0.56875	0.596875	0.534375	0.540625	0.571875
4	0.60815	0.592476	0.598746	0.589342	0.53605	0.595611
Promedio	0.594755	0.58037	0.598499	0.563493	0.542835	0.584747

Mejor modelo: LDA. Accuracy = 0.59850

- Por último, con el mismo conjunto, pero **estandarizando** la escala de las covariables:

```
CrossValidate('E. S. Estandarizadas', modelos, best_scores, X_s, y)
```

	Reg Log	Reg Log OvR	LDA	QDA	KNN	Bayes
0	0.6	0.5875	0.596875	0.553125	0.534375	0.584375
1	0.59375	0.553125	0.59375	0.571875	0.565625	0.590625
2	0.609375	0.60625	0.60625	0.56875	0.55625	0.58125
3	0.615625	0.575	0.596875	0.534375	0.640625	0.571875
4	0.605016	0.589342	0.598746	0.589342	0.579937	0.595611
Promedio	0.604753	0.582243	0.598499	0.563493	0.575362	0.584747

Mejor modelo: Reg Log. Accuracy = 0.60475

Para finalizar, se muestran los mejores modelos y sus scores, para cada repetición del procedimiento:

```
Para Todas las Covariables : mejor modelo: Reg Log , accuracy = 0.59850
Para Est. Significativas   : mejor modelo: LDA     , accuracy = 0.59850
Para E. S. Estandarizadas  : mejor modelo: Reg Log , accuracy = 0.60475
Para Todas Estandarizadas  : mejor modelo: Reg Log , accuracy = 0.59912
```

2.4 Elección del Mejor Modelo

Analizando los scores de cada modelo en cada instancia de la validación cruzada, es claro que existe un ganador.
La **regresión logística multinomial** resulta ser el modelo con la mejor performance de forma casi absoluta, dejando poco margen para debate.

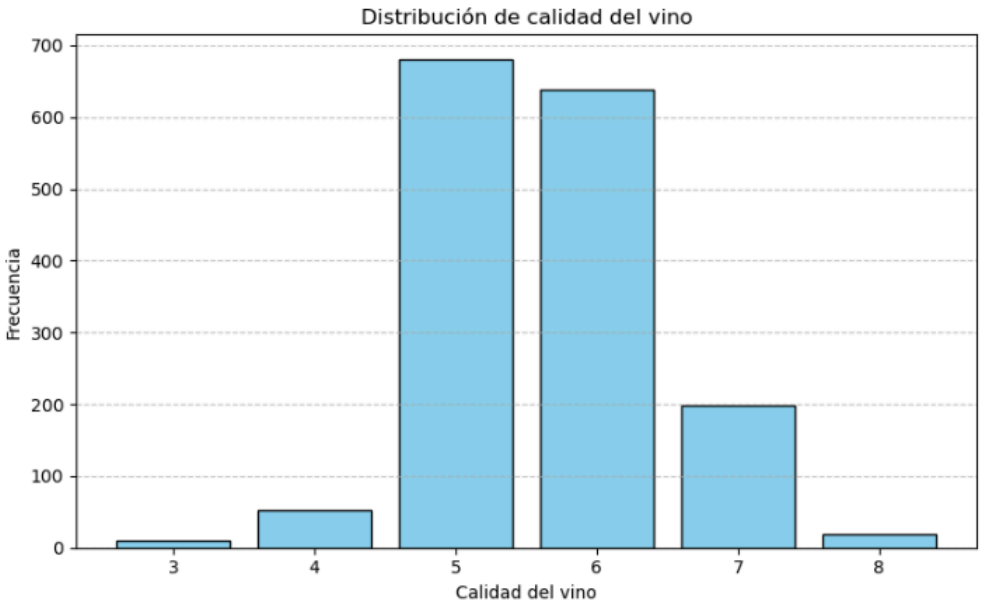
2.4.1 Explicación del Ajuste

El modelo de regresión logística multinomial es la extensión de la regresión logística (clasificación binaria), y permite realizar tareas de clasificación multiclase. El algoritmo que implementa este modelo en la librería sklearn es conocido como regresión softmax, y utiliza una función del mismo nombre para estimar la probabilidad de cada clase dado un vector de covariables X_1, …, X_{p-1}. Este modelo, al igual que su versión más simple, cuenta con fronteras de decisión lineales entre cualesquiera dos clases. Además, como todos los métodos de clasificación vistos hasta ahora, la regresión logística multinomial aplica la regla de Bayes para clasificar. Es decir, predice la clase de un determinado dato como la clase que tenga la probabilidad estimada más grande entre las clases.

Este modelo, además, presenta una mayor estabilidad y precisión frente a distintas estandarizaciones. Sin embargo, la recomendación es no contemplar ciegamente este método para cuándo debemos trabajar con más de diez variables, por eso se tuvo en cuenta el análisis de las variables estadísticamente significativas y, en contraposición, el dataset completo.

3. Conclusiones

Finalizando nuestro desarrollo, llegamos a dos caminos ante los cuales tomar una decisión. Por un lado, nos encontramos con la Regresión Lineal (1.5) la cuál logra explicar en un 36% la variabilidad de la decisión frente a la variable respuesta.
Por otro lado, tenemos una Regresión Logística que presentó una exactitud del ~60% en sus decisiones. No obstante, debemos remarcar que nos encontrábamos frente a clases desbalanceadas sobre las cuales nuestro modelo se entrenó. Se puede apreciar la distribución de nuestros datos según las clases en el siguiente gráfico:



Este es un aspecto muy importante a considerar, debido a que, al intentar clasificar con pocas observaciones en ciertas clases, los métodos de clasificación no logran interpretar correctamente los patrones de esas clases (2.1 en el gráfico se puede ver que prácticamente ningún modelo clasifica bien las clases minoritarias). Además, la variable respuesta se presenta como una variable categórica ordinal, lo que permitiría, en principio, considerar los métodos de regresión como una opción viable.

Sin embargo, consideramos que frente a nuestra variable respuesta ‘quality’, el modelo de clasificación (Regresión Logística) sería el más adecuado, debido a que la misma cuenta con un rango de valores discreto. Además, según lo que vimos en clase, lo correcto a hacer frente a una variable respuesta de estas características, es utilizar métodos de clasificación.