INFORME PRÁCTICA FINAL:

Calidad del vino tinto

AUTOR:

Jhony Alejandro Herrera Parrado

PRESENTADO A:

Francisco Carlos Calderón Bocanegra

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE ELECTRÓNICA
BOGOTÁ D.C. 2022

INTRODUCCIÓN

Como consumidor promedio de vino en algunos momentos es necesario tener una ayuda a la hora de seleccionarlo, es por esto por lo que surge este algoritmo que ayuda a tomar una decisión de acuerdo con las principales características de los vinos. Acá se podrá encontrar un algoritmo que ayudará a decidir cuál es el mejor de acuerdo con los gustos del consumidor. Para esto se utilizarán los métodos KNN y DecisionTreeClassifier, además de esto se podrá observar las diferentes cualidades de estos por medios.

Ilustración 1 imágenes de vinos que he probado

DESARROLLO

1. En primer lugar se debe seleccionar una base de datos, se elige esta base de datos ya que es la que mayores calificaciones presenta.

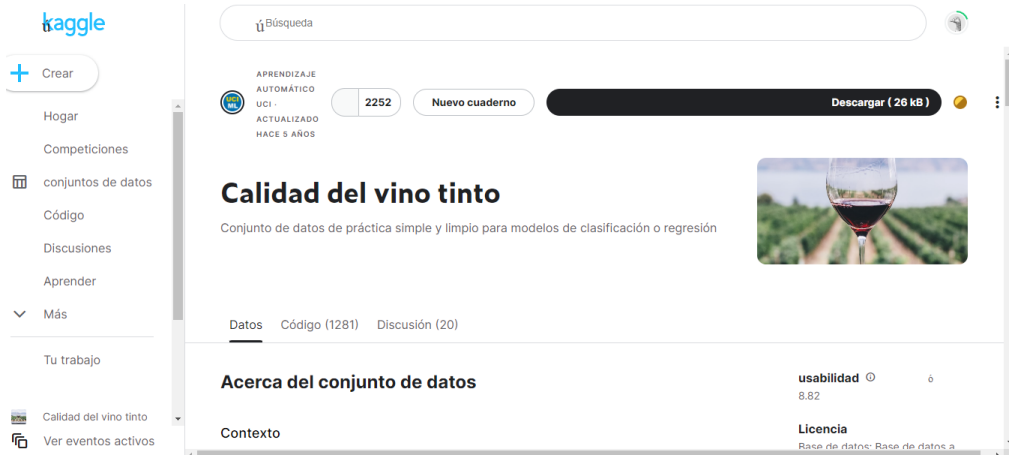


Ilustración 2 base de datos en kaggle

2. Se importa la base de datos y las librerías necesarias para el desarrollo de este.

```
[149] # jhony alejandro herrera parrado
      #En primer lugar importamos los archivos necesario para el programa y la base de datos tomada de  https://www.kaggle.com/datasets/uciml/red-wine-quality-cortez-et-al-2009 y los
      from sklearn.metrics import matthews_corrcoef
      from sklearn.metrics import recall_score
      from sklearn.metrics import confusion_matrix
      from sklearn.metrics import roc_auc_score
      from sklearn.model_selection import train_test_split
      from sklearn.linear_model import LogisticRegression
      from sklearn.svm import SVC
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn import metrics
      from sklearn.preprocessing import StandardScaler
      import pandas as pd
      import matplotlib.pyplot as plt
      import numpy as np
      from sklearn.metrics import roc_auc_score
      from sklearn.neighbors import KNeighborsClassifier
      from subprocess import check_output
      from sklearn.metrics import accuracy_score
      from sklearn.metrics import f1_score
      from sklearn.metrics import roc_curve,roc_auc_score
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.preprocessing import StandardScaler
      import seaborn as sns
      import matplotlib.pyplot as plt
      from sklearn import decomposition
      %matplotlib inline
      from sklearn.linear_model import SGDClassifier
      from sklearn.linear_model import RidgeClassifier
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.ensemble import BaggingClassifier
      from sklearn.ensemble import RandomForestClassifier
      import lightgbm as lgb
      from sklearn.metrics import confusion_matrix
```

3. Se visualizan los datos para verificar que sean los requeridos:

```
rdwineq = pd.read_csv('/content/winequality-red.csv')  # cargar dataset
rdwineq.head(-1)  # cabecera de primeros  y ultimos datos del dataset
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1593	6.8	0.620	0.08	1.9	0.068	28.0	38.0	0.99651	3.42	0.82	9.5	6
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5

1598 rows × 12 columns

4. Se procede a hacer la eliminación y ajustes de valores NAN en cada una de las características.

```
 # proceso de eliminación y ajustes de valores NAN  en  cada  cualidad
rdwineq['fixed acidity'].fillna(rdwineq['fixed acidity'].mean(),inplace=True) # acidez fija
rdwineq['volatile acidity'].fillna(rdwineq['volatile acidity'].mean(),inplace=True) #acidez volátil
rdwineq['residual sugar'].fillna(rdwineq['residual sugar'].mean(), inplace=True) #ácido cítrico
rdwineq['citric acid'].fillna(rdwineq['citric acid'].mean(),inplace=True) #azúcar residual
rdwineq['chlorides'].fillna(rdwineq['chlorides'].mean(), inplace=True) #cloruros
rdwineq['free sulfur dioxide'].fillna(rdwineq['free sulfur dioxide'].mean(),inplace=True)#dióxido de azufre libre
rdwineq['total sulfur dioxide'].fillna(rdwineq['total sulfur dioxide'].mean(),inplace=True) #dióxido de azufre total
rdwineq['density'].fillna(rdwineq['density'].mean(), inplace=True) #densidad
rdwineq['pH'].fillna(rdwineq['pH'].mean(), inplace=True) #pH
rdwineq['sulphates'].fillna(rdwineq['sulphates'].mean(),inplace=True)#sulfatos
rdwineq['alcohol'].fillna(rdwineq['alcohol'].mean(),inplace=True) # alcohol
rdwineq['quality'].fillna(rdwineq['quality'].mean(),inplace=True) #calidad
```

5. Nuevamente se visualizan los datos luego de realizar los ajustes.

```
rdwineq.info(-1) # resumen de información

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1599 entries, 0 to 1598
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 150.0 KB
```

```
rdwineq.describe()  # cabecera de primeros  y ultimos datos del datase luego de los ajustes
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
count	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000	1599.000000
mean	8.319637	0.527821	0.270976	2.538806	0.087467	15.874922	46.467792	0.996747	3.311113	0.658149	10.422983	5.636023
std	1.741096	0.179060	0.194801	1.409928	0.047065	10.460157	32.895324	0.001887	0.154386	0.169507	1.065668	0.807569
min	4.600000	0.120000	0.000000	0.900000	0.012000	1.000000	6.000000	0.990070	2.740000	0.330000	8.400000	3.000000
25%	7.100000	0.390000	0.090000	1.900000	0.070000	7.000000	22.000000	0.995600	3.210000	0.550000	9.500000	5.000000
50%	7.900000	0.520000	0.260000	2.200000	0.079000	14.000000	38.000000	0.996750	3.310000	0.620000	10.200000	6.000000
75%	9.200000	0.640000	0.420000	2.600000	0.090000	21.000000	62.000000	0.997835	3.400000	0.730000	11.100000	6.000000
max	15.900000	1.580000	1.000000	15.500000	0.611000	72.000000	289.000000	1.003690	4.010000	2.000000	14.900000	8.000000

6. Ahora se usan los datos de forma que el 70 % de estos se usan para entrenar, 20 % para probar y el 10 % para validar el método.

7. Se genera un nuevo archivo luego de realizar los ajustes de los datos en general y se le hace una estandarización de los datos.

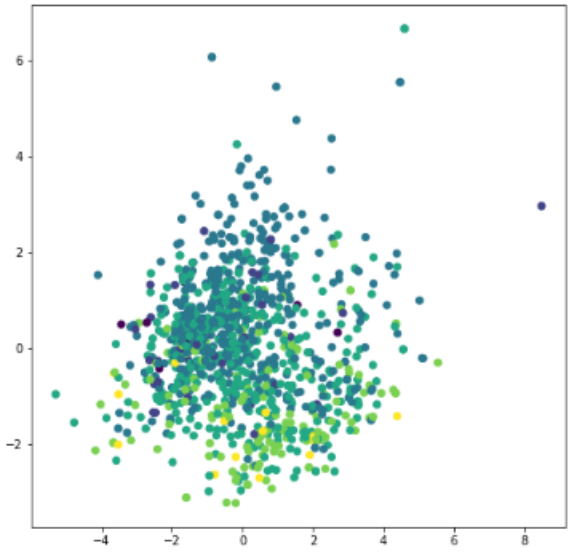
	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
1	7.8	0.880	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5
2	7.8	0.760	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5
3	11.2	0.280	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6
4	7.4	0.700	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5
...	...	...	...	...	...	...	...	...	...	...	...	...
1594	6.2	0.600	0.08	2.0	0.090	32.0	44.0	0.99490	3.45	0.58	10.5	5
1595	5.9	0.550	0.10	2.2	0.062	39.0	51.0	0.99512	3.52	0.76	11.2	6
1596	6.3	0.510	0.13	2.3	0.076	29.0	40.0	0.99574	3.42	0.75	11.0	6
1597	5.9	0.645	0.12	2.0	0.075	32.0	44.0	0.99547	3.57	0.71	10.2	5
1598	6.0	0.310	0.47	3.6	0.067	18.0	42.0	0.99549	3.39	0.66	11.0	6

1599 rows × 12 columns

```
[181] dato2.to_csv('datos2_data.csv', index=False) # se  exporta el archivo con los nuevos datos

[185] X_train1, X_validate, Y_train1, Y_validate = train_test_split(X,y, test_size=0.1, random_state=2) #  ahora se usan los datos de forma que 70 % de los datos para se usan entrenar, 20 % para probar y el 10 % para validar
      X_train, X_test, Y_train, Y_test = train_test_split(X_train1,Y_train1, test_size=0.2222, random_state=2)

scaler = StandardScaler() # se estandarizan los datos
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
X_validate = scaler.transform(X_validate)
```

8. Luego de esto se hace uso de PCA, la cual es una herramienta para reducir la dimensionalidad en los datos, que puede ser utilizada para convertir un conjunto bastante grande de variables en un conjunto más pequeño que contenga la mayor cantidad de información contenida en el conjunto grande.

```
n=6
pca=decomposition.PCA(n_components=n)
pca.fit(X_train)
X_train=pca.transform(X_train)
Y_train=np.choose(Y_train, [1,2,3,4,5,6,7,8,9,10,0]).astype(float)
plt.scatter(X_train[:, 0], X_train[:, 1],  c=Y_train,)
plt.show()
print("  grafica de pca :",pca.explained_variance_ratio_)
#por el criterio del 97 % no hay necesidad de hacer una reduccion de dimensiones#
```




```
grafica de pca : [0.27766488 0.17835655 0.14059171 0.11059867 0.09049408 0.05802737]
```

9. Proceso de entrenamiento de método KNN.

```
# proceso de entrenemanieto
modelo_knn = KNeighborsClassifier().fit(X_train, Y_train) #modelo knn
params = { 'n_neighbors': range(1,50),
           'weights' : ["uniform","distance"],
           'metric' : ["euclidean","manhattan","chebyshev","minkowski"],
           'algorithm' : ["auto", "ball_tree", "kd_tree", "brute"]}
grid = GridSearchCV(estimator= KNeighborsClassifier(), param_grid=params, cv= 10)
grid_search=grid.fit(X_train, Y_train) #grid_search
print(grid_search.best_params_) #se imprime grid_search
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 7 members, which is less than n_splits=10.
  UserWarning,
{'algorithm': 'auto', 'metric': 'chebyshev', 'n_neighbors': 31, 'weights': 'distance'}
```

10. Se hace un reciclaje con los mejores parámetros, ya que del 90% de los datos que se usan 70 % se usan entrenar y 20 % para probar

```
[190]   #ahora se hace un reciclaje con los mejores parámetros y con el 90% de los datos que son  70 % de los datos para se usan entrenar, 20 % para probar
    knn = KNeighborsClassifier(algorithm= 'auto', metric= 'chebyshev', n_neighbors= 31,metric_params=None, weights= 'distance')
    knn.fit(X_train1, Y_train1)
    y_test_hat=knn.predict(X_validate)
    test_accuracy=accuracy_score(Y_validate,y_test_hat)*100
    print("La precisión de nuestro conjunto de datos de entrenamiento con ajuste es con el total de datos : {:.2f}%".format(test_accuracy) )
    MCC = matthews_corrcoef(Y_validate, y_test_hat) # cofeciente de matthews
    print("MCC = ", MCC)
```

11. Se hace DecisionTreeClassifier.

```
[ ] paramdt = {'criterion' : ['gini','entropy'], #se hace DecisionTreeClassifier
                'max_depth': range(1,20),
                'min_samples_split': range(1,15),
                'min_samples_leaf': range(1,15)}
    griddt = GridSearchCV(estimator= DecisionTreeClassifier(),param_grid=paramdt, cv= 10)
    grid_searchdt=griddt.fit(X_train, Y_train) #grid_search
    print(grid_searchdt.best_params_) # imprimir los dados
```

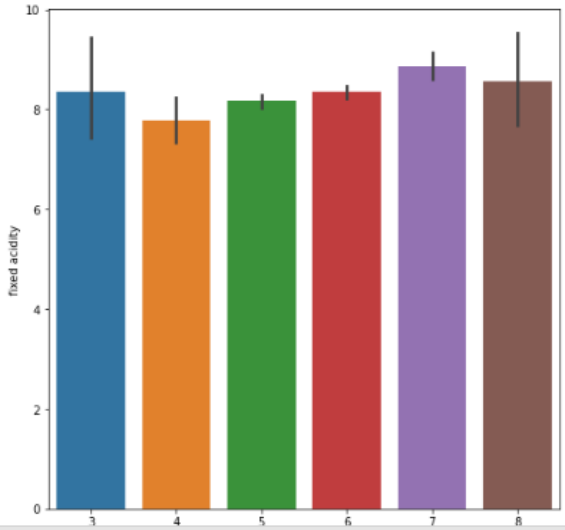
```
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_split.py:680: UserWarning: The least populated class in y has only 7 members, which is less than n_splits=10.
  UserWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py:372: FitFailedWarning:
5320 fits failed out of a total of 74480.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
5320 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_validation.py", line 680, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line 942, in fit
    X_idx_sorted=X_idx_sorted,
  File "/usr/local/lib/python3.7/dist-packages/sklearn/tree/_classes.py", line 254, in fit
    % self.min_samples_split
ValueError: min_samples_split must be an integer greater than 1 or a float in (0.0, 1.0]; got the integer 1

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/model_selection/_search.py:972: UserWarning: One or more of the test scores are non-finite: [       nan 0.51739865 0.51739865 ... (
  category=UserWarning,
```

RESULTADOS

1. Se puede observar cada característica y la variación de este.

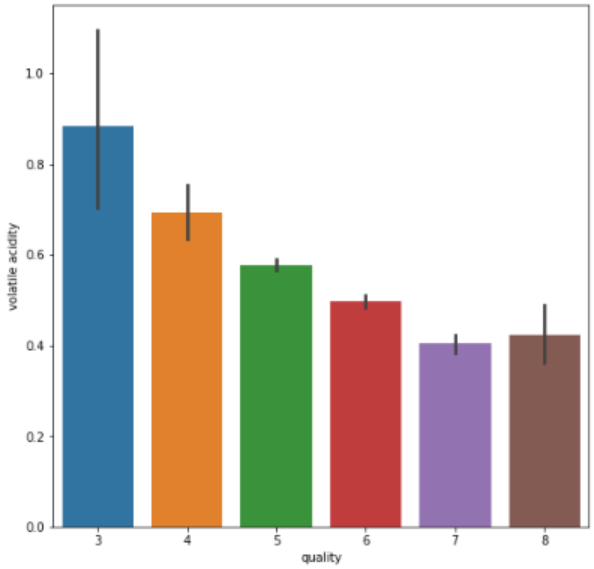
```
fig = plt.figure(figsize = (8,8))
sns.barplot(x = 'quality', y = 'fixed acidity', data =rdwineq)  # acidez fija
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20a2405e50>
```

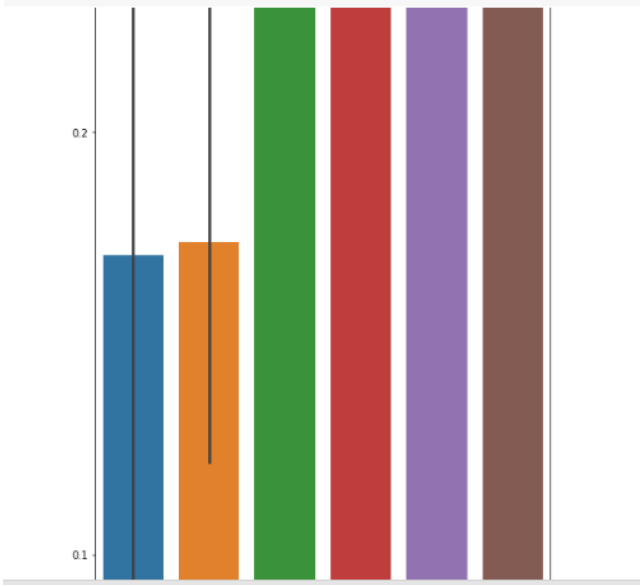



```
[155] fig = plt.figure(figsize = (8,8))
      sns.barplot(x = 'quality', y = 'volatile acidity', data =rdwineq) #acidez voláti
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20a2495890>
```

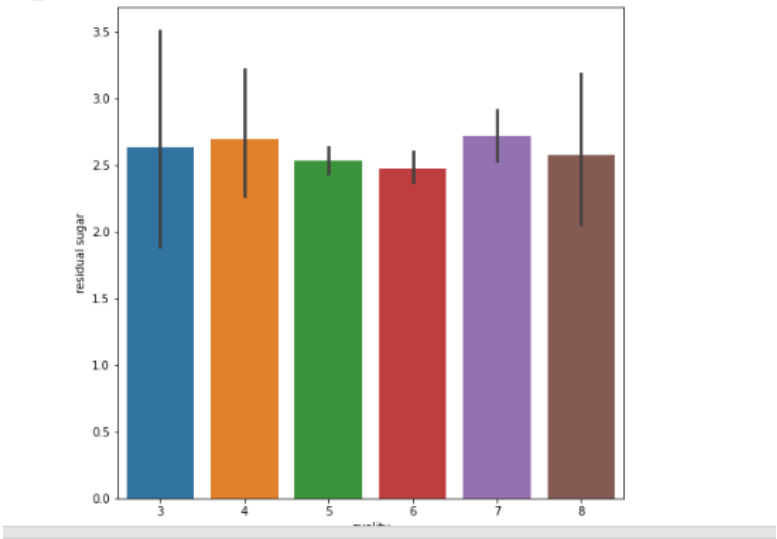


```
[156] fig = plt.figure(figsize = (8,38))
      sns.barplot(x = 'quality', y = 'citric acid', data = rdwineq) #ácido cítric
```



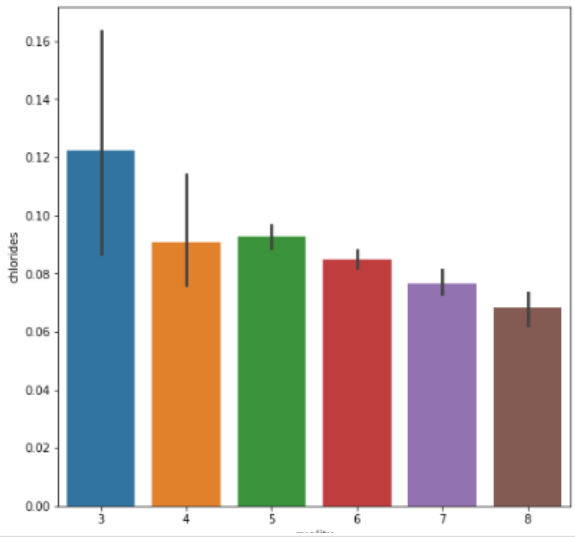

```
[ ] fig = plt.figure(figsize = (8,8))
    sns.barplot(x = 'quality', y = 'residual sugar', data = rdwineq)#azúcar residual
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f209fc6f3d0>
```



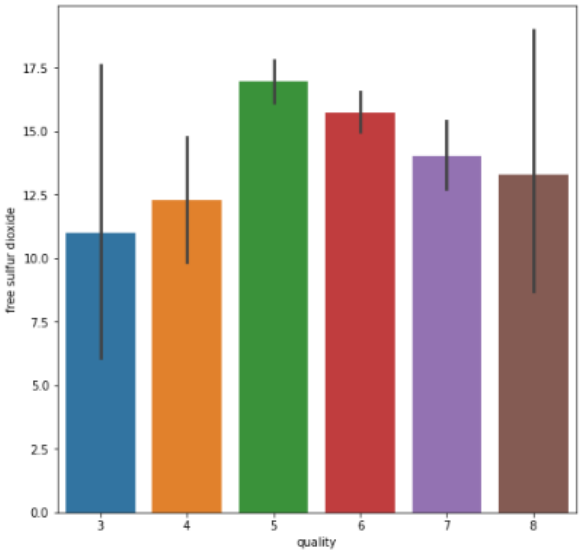
```
fig = plt.figure(figsize = (8,8))
sns.barplot(x = 'quality', y = 'chlorides', data =rdwineq)#libre de cloruros
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20a1f0b910>
```



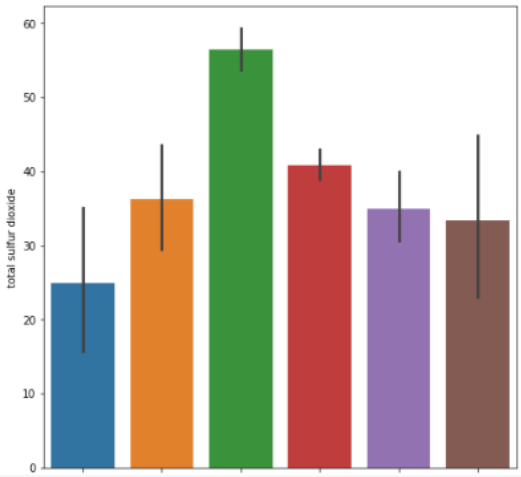

```
fig = plt.figure(figsize = (8,8))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = rdwineq)#dioxido sulfuroso
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20a1837910>
```



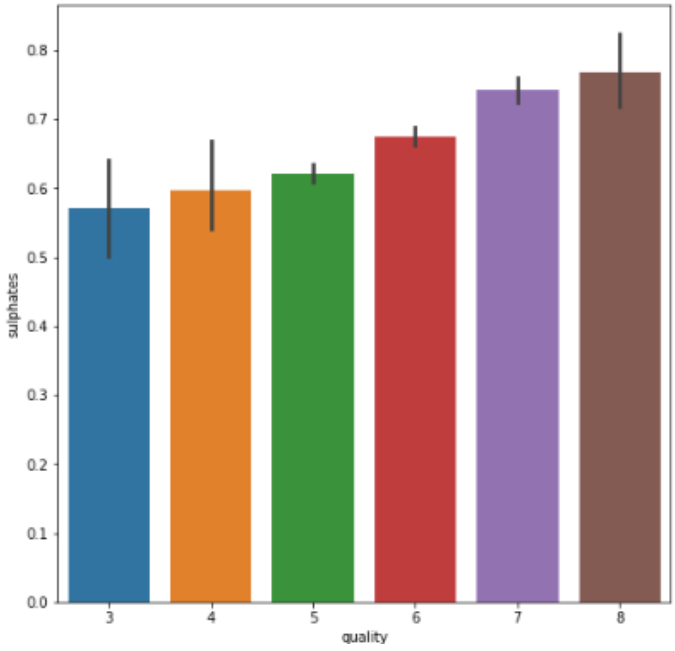
```
fig = plt.figure(figsize = (8,8))
sns.barplot(x = 'quality', y = 'total sulfur dioxide', data =rdwineq)# dióxido de azufre libre
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20a1a5db50>
```



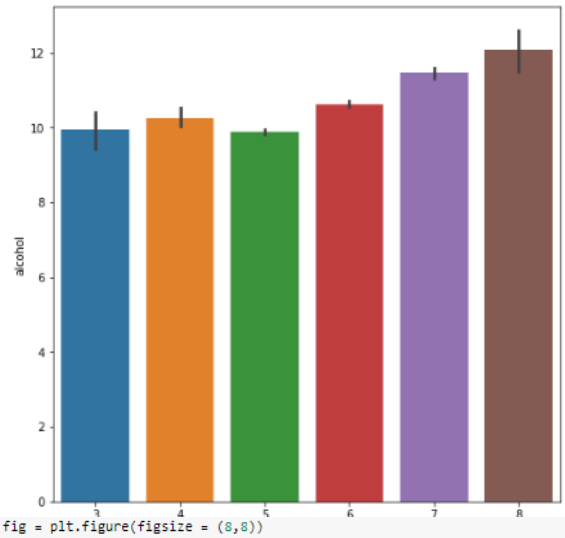

```
[161] fig = plt.figure(figsize = (8,8))
      sns.barplot(x = 'quality', y = 'sulphates', data =rdwineq) # % de sulfatos
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20a1fbdb10>
```



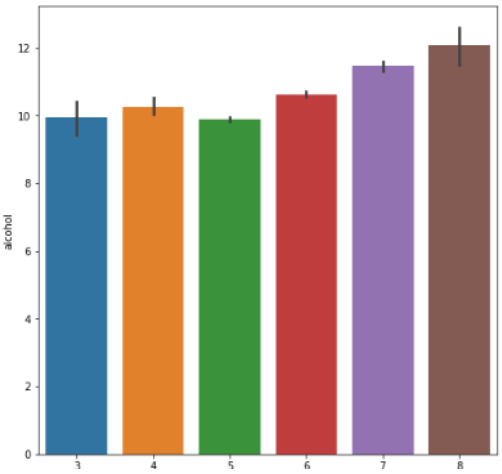
```
[162] fig = plt.figure(figsize = (8,8))
      sns.barplot(x = 'quality', y = 'alcohol', data = rdwineq) # % alcohol
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20a19f3cd0>
```




```
[162] fig = plt.figure(figsize = (8,8))
      sns.barplot(x = 'quality', y = 'alcohol', data = rdwineq) # % alcohol
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f20a19f3cd0>
```

2. La precisión de este conjunto de datos de entrenamiento con ajuste en KNN : 67.29% CON MCC = 0.2275418598167701

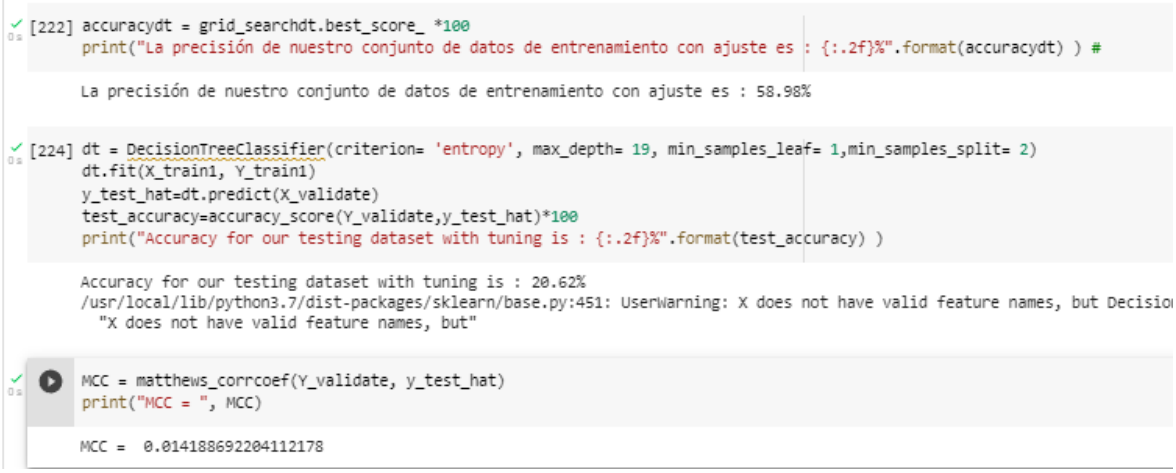
```
[191] accuracy = grid_search.best_score_ *100  #  se escoge el  mejor accuracy
      print("La precisión de nuestro conjunto de datos de entrenamiento con ajuste es: {:.2f}%".format(accuracy) )
```

```
La precisión de nuestro conjunto de datos de entrenamiento con ajuste es: 67.29%
```

```
[190]    #ahora se hace un reciclaje con los mejores parámetros y con el 90% de los datos que son  70 % de los datos para se usan entrenar, 20 % para probar
      knn = KNeighborsClassifier(algorithm= 'auto', metric= 'chebyshev', n_neighbors= 31,metric_params=None, weights= 'distance')
      knn.fit(X_train1, Y_train1)
      y_test_hat=knn.predict(X_validate)
      test_accuracy=accuracy_score(Y_validate,y_test_hat)*100
      print("La precisión de nuestro conjunto de datos de entrenamiento con ajuste es con el total de datos : {:.2f}%".format(test_accuracy) )
      MCC = matthews_corrcoef(Y_validate, y_test_hat) # cofeciente de matthews
      print("MCC = ", MCC)
```

```
La precisión de nuestro conjunto de datos de entrenamiento con ajuste es : 51.25%
MCC =  0.2275418598167701
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
  "X does not have valid feature names, but"
```

3. La precisión del conjunto de datos de entrenamiento con ajuste en DecisionTreeClassifier : 59.43% MCC = 0.014188692204112178



```
[222] accuracydt = grid_searchdt.best_score_ *100
      print("La precisión de nuestro conjunto de datos de entrenamiento con ajuste es : {:.2f}%".format(accuracydt) ) #

      La precisión de nuestro conjunto de datos de entrenamiento con ajuste es : 58.98%

[224] dt = DecisionTreeClassifier(criterion= 'entropy', max_depth= 19, min_samples_leaf= 1,min_samples_split= 2)
      dt.fit(X_train1, Y_train1)
      y_test_hat=dt.predict(X_validate)
      test_accuracy=accuracy_score(Y_validate,y_test_hat)*100
      print("Accuracy for our testing dataset with tuning is : {:.2f}%".format(test_accuracy) )

      Accuracy for our testing dataset with tuning is : 20.62%
      /usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not have valid feature names, but Decisio
        "X does not have valid feature names, but"

      MCC = matthews_corrcoef(Y_validate, y_test_hat)
      print("MCC = ", MCC)

      MCC =  0.014188692204112178
```

CONCLUSIONES

Se pudo observar que en ambos casos los valores de entrenamientos y los valores finales al realizar la validación del 10% presentan una variación, esto se puede inferir ya que muchos datos como el PH, azúcar residual, acidez fija, no presentan variaciones mayores al 0.1. Sin embargo, se puede analizar que hay características que son muy significativas, como lo son, el ácido cítrico el cual varia hasta en 0.3 , el % de sulfatos, el dióxido de azufre libre, el dióxido sulfuroso y el % de cloruros, por lo tanto estas son las principales características a la hora de analizar y obtener las bases de datos. Aunque la diferencia entre KNN y DecisionTreeClassifier en el entrenamiento es alta, una vez se pone a prueba con la validación se puede descartar el método de DecisionTreeClassifier, ya que su precisión es mucho menor a la de KNN.

Enlace de https://github.com/
https://github.com/exhanime/exhanime.git
gh repo clone exhanime/exhanime
git@github.com:exhanime/exhanime.git

Enlace presentación en video YouTube
https://youtu.be/zRDkIg3r_w4