Classificiação de vinhos brancos usando undersampling, hot-encoding, redimensionando qualidades para alta média e baixa

In [3]:
```
#Desabilita logs e mantém apenas logs críticos (para evitar o libcuda ficar me avisando qu
%config Completer.use_jedi = False
import logging
logger = logging.getLogger()
logger.setLevel(logging.CRITICAL)
```

In [4]:
```
import pandas as pd
import numpy as np
import seaborn as sb
import matplotlib.pyplot as plt
import scipy as spy
import keras
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras import layers
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix
df = pd.read_csv('datasets/winequality-white.csv')
```

```
2021-08-18 07:02:39.473033: I tensorflow/stream_executor/platform/default/dso_loader.cc:5
3] Successfully opened dynamic library libcudart.so.11.0
```

Esse dataset já está tratado e não tem valores N/A:

In [5]:
```
pd.isna(df).sum()
```

Out[5]:
```
fixed acidity           0
volatile acidity        0
citric acid             0
residual sugar          0
chlorides               0
free sulfur dioxide     0
total sulfur dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

E todos os valores já estão como float, fora a qualidade que está como valor inteiro:

In [6]:
```
df.dtypes
```

Out[6]:
```
fixed acidity           float64
volatile acidity        float64
citric acid             float64
residual sugar          float64
chlorides               float64
free sulfur dioxide     float64
total sulfur dioxide    float64
density                 float64
pH                      float64
sulphates               float64
```

```
alcohol                 float64
quality                   int64
dtype: object
```

Podemos ver abaixo que existem muitos outliers, não só nessa coluna, mas em várias outras.Vou tentar usar um método para remover os outliers dessa, depois tento remover os das restantes com o mesmo método. **Neste método, os outliers são definidos como os parâmetros que estão 3 desvios padrões acima da média.**

Este método seria um método de under-sampling, como é colocado no artigo do qual retiramos o dataset.

In [7]:

```python
df['quality'].value_counts()
```

Out[7]:

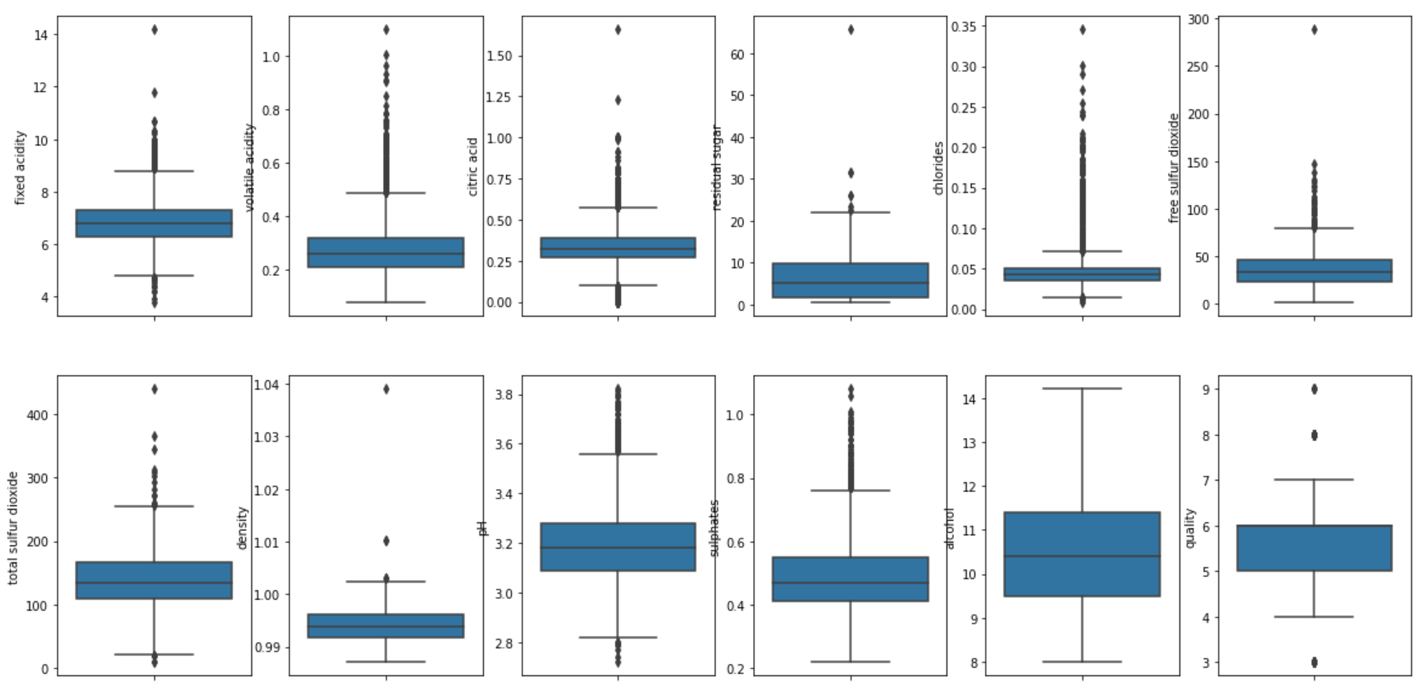
```
6    2198
5    1457
7     880
8     175
4     163
3      20
9       5
Name: quality, dtype: int64
```

Nos boxplots fica mais fácil de visualizar esses outliers:

In [8]:

```python
fig,ax = plt.subplots(ncols=6,nrows=2,figsize=(20,10))
index=0
ax=ax.flatten()

for col, value in df.items():
    if col!='type':
        sb.boxplot(y=col,data=df,ax=ax[index])
        index+=1
```



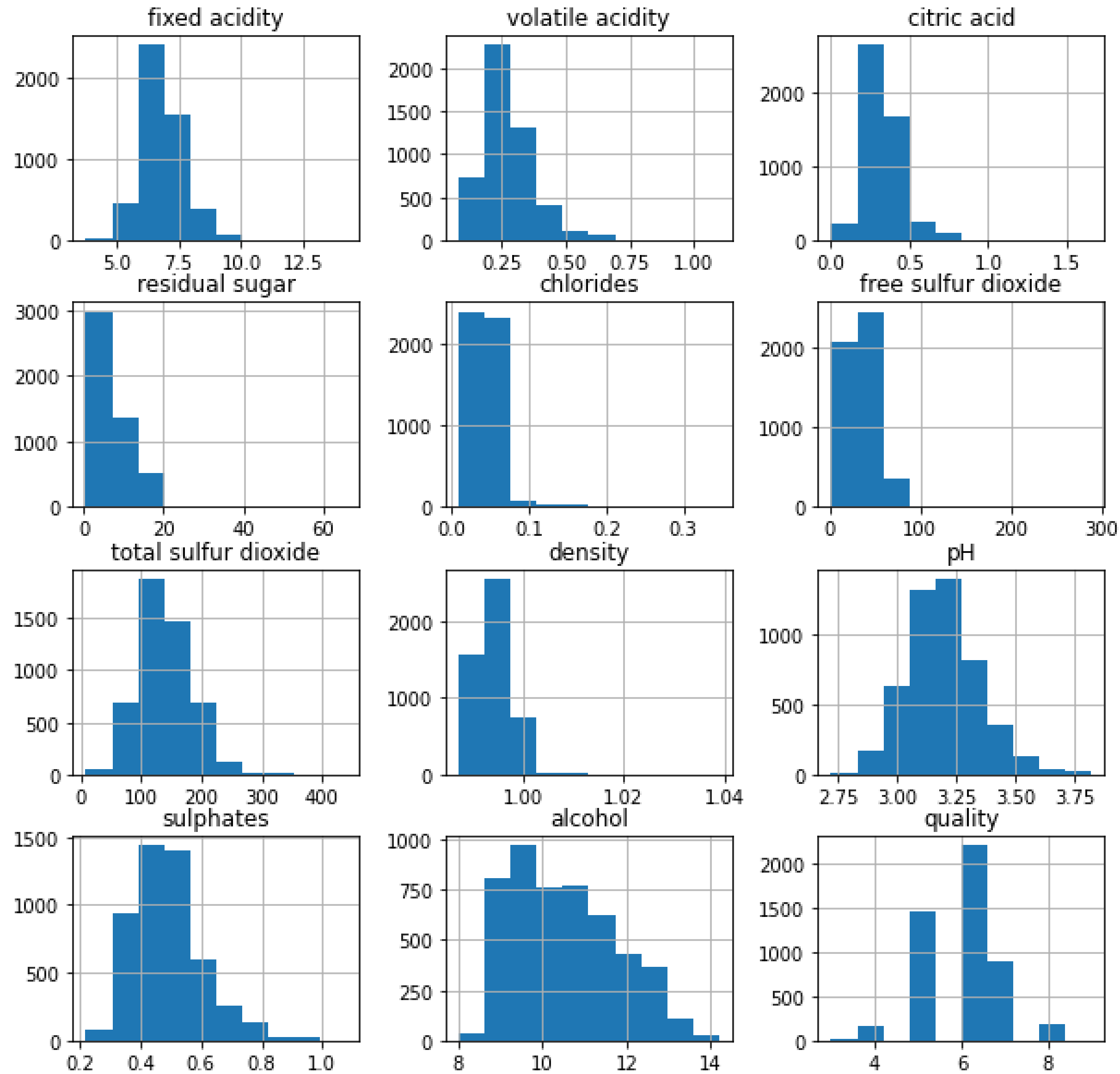
In [9]:

```python
df.hist(figsize = (10, 10))
```

Out[9]:

```
array([[<AxesSubplot:title={'center':'fixed acidity'}>,
        <AxesSubplot:title={'center':'volatile acidity'}>,
        <AxesSubplot:title={'center':'citric acid'}>],
       [<AxesSubplot:title={'center':'residual sugar'}>,
        <AxesSubplot:title={'center':'chlorides'}>,
```

```
 <AxesSubplot:title={'center':'free sulfur dioxide'}>],
[<AxesSubplot:title={'center':'total sulfur dioxide'}>,
 <AxesSubplot:title={'center':'density'}>,
 <AxesSubplot:title={'center':'pH'}>],
[<AxesSubplot:title={'center':'sulphates'}>,
 <AxesSubplot:title={'center':'alcohol'}>,
 <AxesSubplot:title={'center':'quality'}>]], dtype=object)
```



Abaixo, vamos remover todos os valores que estão acima de 3 desvios padrões da média.

In [10]:

```
cols = list(df.columns)
zscores = spy.stats.zscore(df[cols], nan_policy='omit')
abs_zscores = np.abs(zscores)
filtered_entries = (abs_zscores < 3).all(axis=1)
new_df = df[filtered_entries]
df = new_df
```

Feito esse método de under-sampling, vemos que no dataset as distribuições estão muito mais próximas de distribuições normais. E temos muito menos outliers nos boxplots abaixo:

In [11]:

```
df['quality'].value_counts()
```

Out[11]:

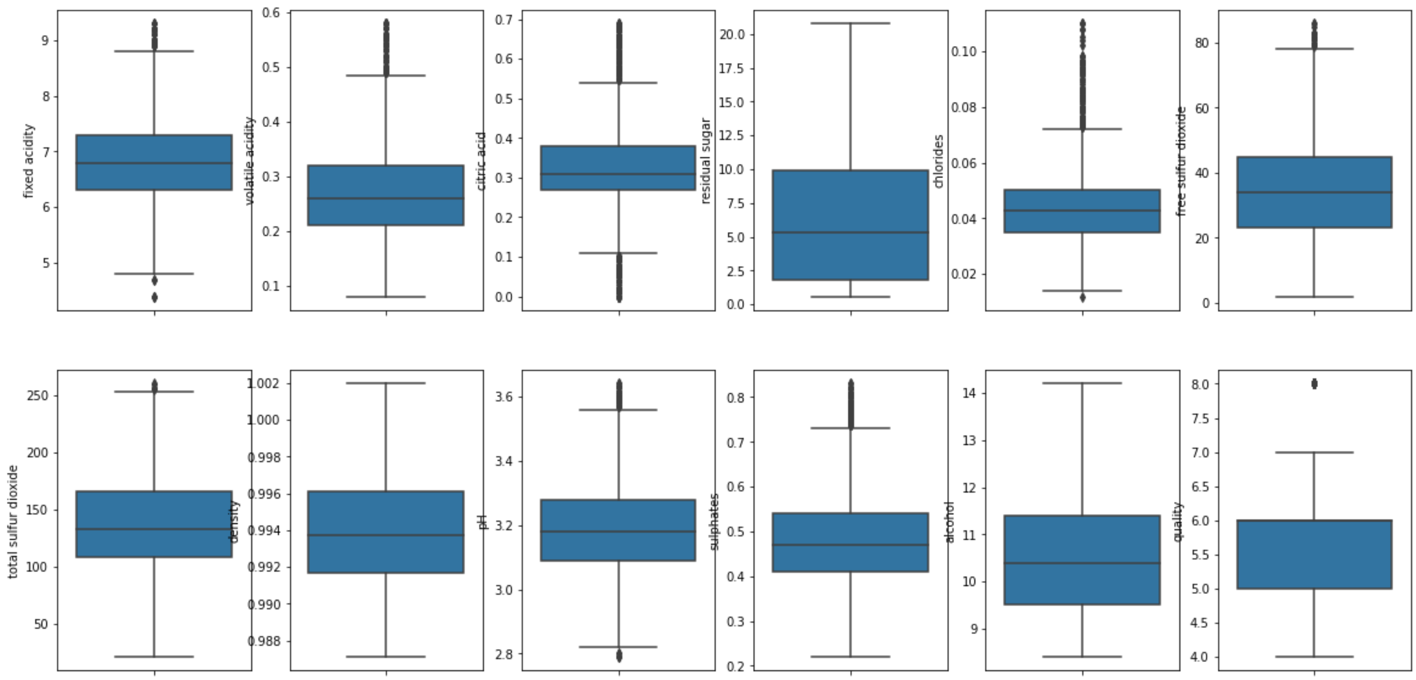
```
6    2038
5    1309
7     855
8     161
```

```
4     124
Name: quality, dtype: int64
```

In [12]:

```python
fig,ax = plt.subplots(ncols=6,nrows=2,figsize=(20,10))
index=0
ax=ax.flatten()

for col, value in df.items():
    if col!='type':
        sb.boxplot(y=col,data=df,ax=ax[index])
        index+=1
```

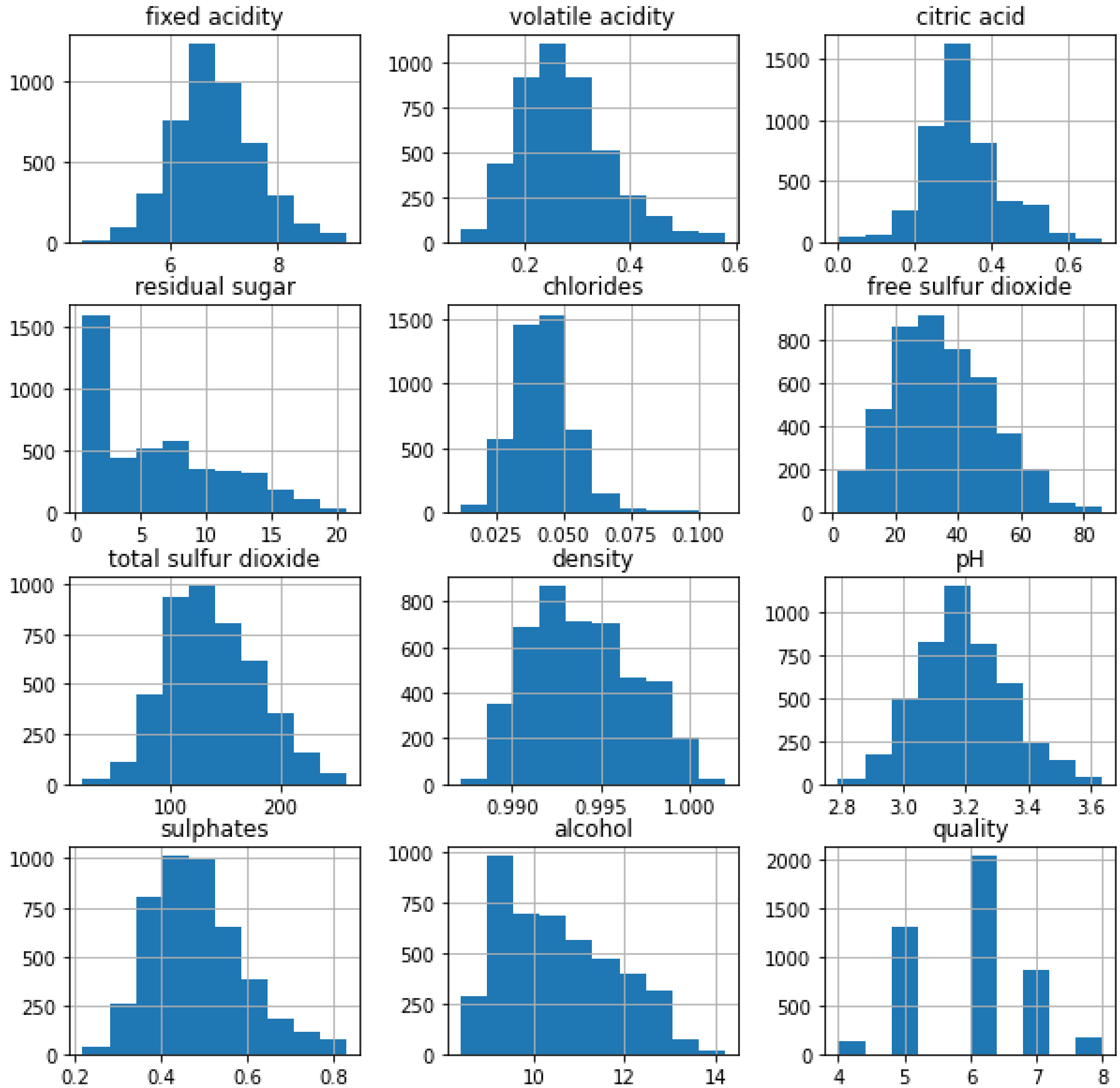


In [13]:

```python
df.hist(figsize = (10, 10))
```

Out[13]:

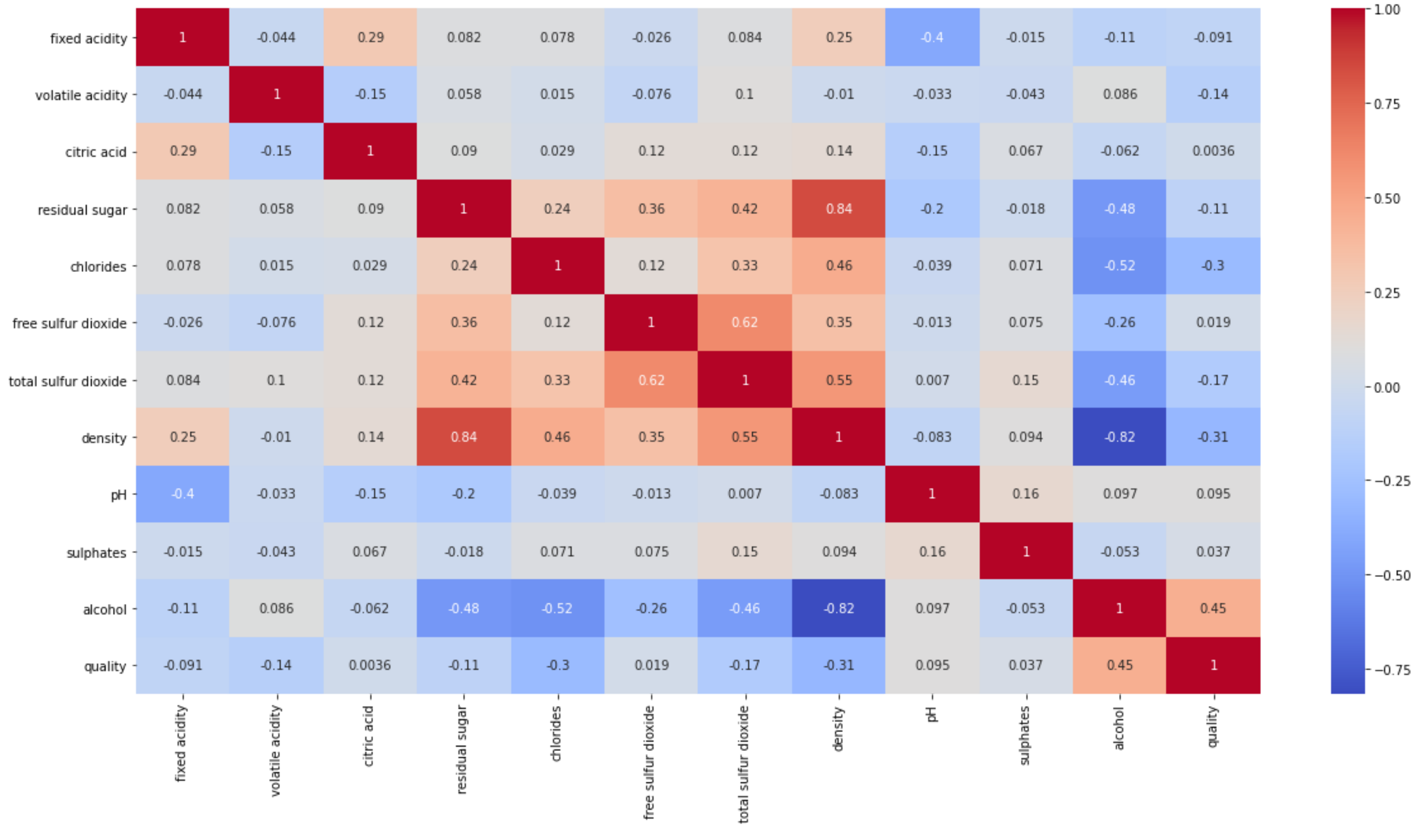
```
array([[<AxesSubplot:title={'center':'fixed acidity'}>,
        <AxesSubplot:title={'center':'volatile acidity'}>,
        <AxesSubplot:title={'center':'citric acid'}>],
       [<AxesSubplot:title={'center':'residual sugar'}>,
        <AxesSubplot:title={'center':'chlorides'}>,
        <AxesSubplot:title={'center':'free sulfur dioxide'}>],
       [<AxesSubplot:title={'center':'total sulfur dioxide'}>,
        <AxesSubplot:title={'center':'density'}>,
        <AxesSubplot:title={'center':'pH'}>],
       [<AxesSubplot:title={'center':'sulphates'}>,
        <AxesSubplot:title={'center':'alcohol'}>,
        <AxesSubplot:title={'center':'quality'}>]], dtype=object)
```



Mapa de calor

In [14]:
```
corr=df.corr()
plt.figure(figsize=(20,10))
sb.heatmap(corr,annot=True, cmap='coolwarm')
```

Out[14]: <AxesSubplot:>



Ainda vou tratar os dados de qualidade para serem entre 0, 1 e 2:

In [15]:

```
df.loc[df['quality'] ==3, 'quality'] = 0
df.loc[df['quality'] ==4, 'quality'] = 0
df.loc[df['quality'] ==5, 'quality'] = 1
df.loc[df['quality'] ==6, 'quality'] = 1
df.loc[df['quality'] ==7, 'quality'] = 1
df.loc[df['quality'] ==8, 'quality'] = 2
df.loc[df['quality'] ==9, 'quality'] = 2
df['quality'].value_counts()
df.head()
```

Out[15]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	1
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	1
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	1
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	1

Ainda temos que fazer o one-hot encoding para que a rede neural consiga prever adequadamente

In [16]:

```
one_hot_encoded_data = pd.get_dummies(df, columns = ['quality'])
df = one_hot_encoded_data.rename(columns={'quality_0': 'baixa', 'quality_1': 'media', 'qua
df.head()
```

Out[16]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	baixa	media
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	0	1

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	baixa	media
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	0	1
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	0	1
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	0	1
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	0	1

Separando o dataset em dados de treinamento e teste

In [17]:

```
X =  df.drop(columns=["baixa", "media", "alta"], axis=1)
y=  df[['baixa','media','alta']]

X_train, X_test, y_train, y_test = train_test_split(X, y,test_size = 0.2, shuffle=True)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(3589, 11)
(898, 11)
(3589, 3)
(898, 3)
```

Treinamento do dataset

In [18]:

```
dimension = X_train.shape[1]
from keras import backend as K
def create_model():
    model = Sequential()
    model.add(layers.Dense(10, input_dim=dimension, activation='relu'))
    model.add(layers.Dense(60, input_dim=dimension, activation='relu'))
    model.add(layers.Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model

model = create_model()
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 10)                120       
_________________________________________________________________
dense_1 (Dense)              (None, 60)                660       
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 183       
=================================================================
Total params: 963
Trainable params: 963
Non-trainable params: 0
_________________________________________________________________
2021-08-18 07:02:44.717554: I tensorflow/stream_executor/platform/default/dso_loader.cc:5
3] Successfully opened dynamic library libcuda.so.1
2021-08-18 07:02:44.775708: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:937] su
ccessful NUMA node read from SysFS had negative value (-1), but there must be at least one
NUMA node, so returning NUMA node zero
```

```
2021-08-18 07:02:44.775963: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1733] Found
device 0 with properties:
pciBusID: 0000:0c:00.0 name: NVIDIA GeForce GTX 660 Ti computeCapability: 3.0
coreClock: 1.0715GHz coreCount: 7 deviceMemorySize: 1.95GiB deviceMemoryBandwidth: 134.29G
iB/s
2021-08-18 07:02:44.775989: I tensorflow/stream_executor/platform/default/dso_loader.cc:5
3] Successfully opened dynamic library libcudart.so.11.0
2021-08-18 07:02:44.783050: I tensorflow/stream_executor/platform/default/dso_loader.cc:5
3] Successfully opened dynamic library libcublas.so.11
2021-08-18 07:02:44.783097: I tensorflow/stream_executor/platform/default/dso_loader.cc:5
3] Successfully opened dynamic library libcublasLt.so.11
2021-08-18 07:02:44.787484: I tensorflow/stream_executor/platform/default/dso_loader.cc:5
3] Successfully opened dynamic library libcufft.so.10
2021-08-18 07:02:44.789315: I tensorflow/stream_executor/platform/default/dso_loader.cc:5
3] Successfully opened dynamic library libcurand.so.10
2021-08-18 07:02:44.789373: W tensorflow/stream_executor/platform/default/dso_loader.cc:6
4] Could not load dynamic library 'libcusolver.so.11'; dlerror: libcusolver.so.11: cannot
open shared object file: No such file or directory
2021-08-18 07:02:44.791139: I tensorflow/stream_executor/platform/default/dso_loader.cc:5
3] Successfully opened dynamic library libcusparse.so.11
2021-08-18 07:02:44.791184: W tensorflow/stream_executor/platform/default/dso_loader.cc:6
4] Could not load dynamic library 'libcudnn.so.8'; dlerror: libcudnn.so.8: cannot open sha
red object file: No such file or directory
2021-08-18 07:02:44.791192: W tensorflow/core/common_runtime/gpu/gpu_device.cc:1766] Canno
t dlopen some GPU libraries. Please make sure the missing libraries mentioned above are in
stalled properly if you would like to use GPU. Follow the guide at https://www.tensorflow.
org/install/gpu for how to download and setup the required libraries for your platform.
Skipping registering GPU devices...
2021-08-18 07:02:44.791670: I tensorflow/core/platform/cpu_feature_guard.cc:142] This Tens
orFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the fol
lowing CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flag
s.
2021-08-18 07:02:44.791889: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1258] Devic
e interconnect StreamExecutor with strength 1 edge matrix:
2021-08-18 07:02:44.791895: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1264]
```

In [19]:

```python
history=model.fit(X_train, y_train, validation_data=(X_test, y_test),epochs=20, batch_size
```

```
Epoch 1/20
2021-08-18 07:02:44.875889: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:17
6] None of the MLIR Optimization Passes are enabled (registered 2)
2021-08-18 07:02:44.877018: I tensorflow/core/platform/profile_utils/cpu_utils.cc:114] CPU
Frequency: 3593380000 Hz
359/359 [==============================] - 1s 869us/step - loss: 0.4363 - accuracy: 0.9320
- val_loss: 0.3226 - val_accuracy: 0.9499
Epoch 2/20
359/359 [==============================] - 0s 588us/step - loss: 0.3285 - accuracy: 0.9329
- val_loss: 0.2927 - val_accuracy: 0.9499
Epoch 3/20
359/359 [==============================] - 0s 558us/step - loss: 0.3238 - accuracy: 0.9312
- val_loss: 0.2546 - val_accuracy: 0.9499
Epoch 4/20
359/359 [==============================] - 0s 541us/step - loss: 0.3260 - accuracy: 0.9323
- val_loss: 0.3048 - val_accuracy: 0.9499
Epoch 5/20
359/359 [==============================] - 0s 549us/step - loss: 0.3201 - accuracy: 0.9320
- val_loss: 0.2466 - val_accuracy: 0.9499
Epoch 6/20
359/359 [==============================] - 0s 610us/step - loss: 0.3119 - accuracy: 0.9323
- val_loss: 0.2629 - val_accuracy: 0.9499
Epoch 7/20
359/359 [==============================] - 0s 584us/step - loss: 0.3048 - accuracy: 0.9331
- val_loss: 0.2567 - val_accuracy: 0.9499
Epoch 8/20
```

```
359/359 [==============================] - 0s 555us/step - loss: 0.3171 - accuracy: 0.9323
- val_loss: 0.2446 - val_accuracy: 0.9499
Epoch 9/20
359/359 [==============================] - 0s 599us/step - loss: 0.3042 - accuracy: 0.9331
- val_loss: 0.2600 - val_accuracy: 0.9499
Epoch 10/20
359/359 [==============================] - 0s 566us/step - loss: 0.3042 - accuracy: 0.9323
- val_loss: 0.3498 - val_accuracy: 0.9499
Epoch 11/20
359/359 [==============================] - 0s 563us/step - loss: 0.3097 - accuracy: 0.9320
- val_loss: 0.2355 - val_accuracy: 0.9499
Epoch 12/20
359/359 [==============================] - 0s 675us/step - loss: 0.3031 - accuracy: 0.9331
- val_loss: 0.2349 - val_accuracy: 0.9499
Epoch 13/20
359/359 [==============================] - 0s 606us/step - loss: 0.3002 - accuracy: 0.9326
- val_loss: 0.2774 - val_accuracy: 0.9499
Epoch 14/20
359/359 [==============================] - 0s 557us/step - loss: 0.3014 - accuracy: 0.9323
- val_loss: 0.2395 - val_accuracy: 0.9499
Epoch 15/20
359/359 [==============================] - 0s 566us/step - loss: 0.2993 - accuracy: 0.9329
- val_loss: 0.2315 - val_accuracy: 0.9499
Epoch 16/20
359/359 [==============================] - 0s 549us/step - loss: 0.2919 - accuracy: 0.9326
- val_loss: 0.2656 - val_accuracy: 0.9499
Epoch 17/20
359/359 [==============================] - 0s 565us/step - loss: 0.2936 - accuracy: 0.9315
- val_loss: 0.2316 - val_accuracy: 0.9499
Epoch 18/20
359/359 [==============================] - 0s 568us/step - loss: 0.2850 - accuracy: 0.9326
- val_loss: 0.2316 - val_accuracy: 0.9499
Epoch 19/20
359/359 [==============================] - 0s 544us/step - loss: 0.2909 - accuracy: 0.9329
- val_loss: 0.2551 - val_accuracy: 0.9499
Epoch 20/20
359/359 [==============================] - 0s 622us/step - loss: 0.2838 - accuracy: 0.9331
- val_loss: 0.2289 - val_accuracy: 0.9499
```

Avaliação do resultado

Convertemos as probabilidades que foram retornadas para o máximo do array:

In [20]:

```python
y_pred = model.predict(X_test)
def max_probs(array):
    parsed_pred = np.empty((0,3))
    for idx, x in enumerate(array):
        idx_max = x.argmax()
        x = np.zeros((3,))
        x[idx_max] = 1
        array[idx] = x
max_probs(y_pred)
y_pred
```

Out[20]:

```
array([[0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.],
       ...,
       [0., 1., 0.],
       [0., 1., 0.],
       [0., 1., 0.]], dtype=float32)
```

Para criar a matriz de confusão vamos converter novamente para as categorias 0 1 e 2 após a predição:

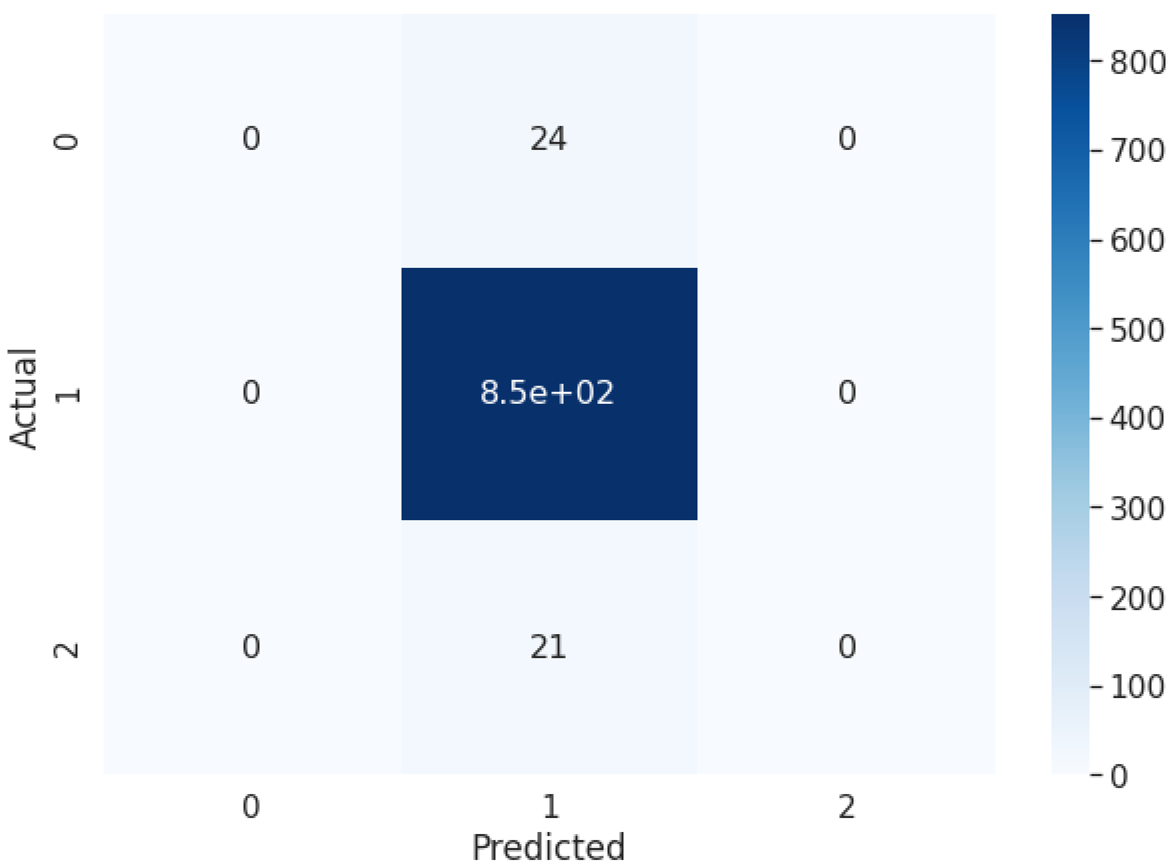
In [21]:
```
def to_category(array):
    categories = []
    for idx, x in enumerate(array):
        idx_max = x.argmax()
        x = 0
        if idx_max == 0: x = 0
        if idx_max == 1: x = 1
        if idx_max == 2: x = 2
        categories.append(x)
    return categories


categorical_y_pred = to_category(y_pred)
categorical_y_test = to_category(y_test.to_numpy())
```

In [22]:
```
data = confusion_matrix(categorical_y_test, categorical_y_pred)
```

In [23]:
```
df_cm = pd.DataFrame(data, columns=np.unique(categorical_y_test), index = np.unique(catego
df_cm.index.name = 'Actual'
df_cm.columns.name = 'Predicted'
plt.figure(figsize = (10,7))
sb.set(font_scale=1.4)#for label size
sb.heatmap(df_cm, cmap="Blues", annot=True,annot_kws={"size": 16})# font size
```

Out[23]:
```
<AxesSubplot:xlabel='Predicted', ylabel='Actual'>
```



verificação da acurácia geral:

In [25]:
```
correct = 0
total = 0
for i in range(len(categorical_y_test)):
    if(categorical_y_test[i] == categorical_y_pred[i]):
        correct += 1
```

```
    total += 1
accuracy = (correct/total)
```

In [26]:
```
accuracy
```

Out[26]:
```
0.9498886414253898
```

Conclusão

Usando a divisão em 3 categorias com undersampling, fica difícil prever quando um vinho não é de média qualidade, ou seja, temos poucas predições nas categorias altas e baixas. O mesmo problema quando tentamos prever no arquivo `redes_do_artigo/relu_white` . Exceto que com undersampling o modelo não conseguiu prever nenhum vinho de baixa qualidade e nenhum de alta qualidade.