In [1]:
```
#Importamos librerías
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")

# Scaler
from sklearn.preprocessing import RobustScaler

# Train Test Split
from sklearn.model_selection import train_test_split

#Models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
import xgboost as xgb

#Metrics
from sklearn.metrics import accuracy_score, classification_report

# Cross Validation
from sklearn.model_selection import GridSearchCV
```

In [2]:
```
#Carga de los datos
df = pd.read_csv('winequality-red.csv')
```

In [3]:
```
#Primer visualización de los datos
df.head()
```

Out[3]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4

In [4]:
```
#Cantidad de registros y columnas
df.shape
```

Out[4]:
```
(1599, 12)
```

In [5]:
```
#Cantidad de nulos
df.info()
```

```
<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
```

In [6]:
```
#Caracteristicas de la variable target
pd.DataFrame( { 'quality': df["quality"].value_counts().index , 'counts': df["qua
```

Out[6]:

	quality	counts
0	3	10
1	4	53
2	5	681
3	6	638
4	7	199
5	8	18

In [7]:

```
#Definición de bins para el target (4.5, 6.5)
def helper(row):
    if row.quality < 4.5:
        return 0
    elif row.quality < 6.5:
        return 1
    else:
        return 2

df["quality"] = df.apply(helper,axis=1)
```

In [8]:

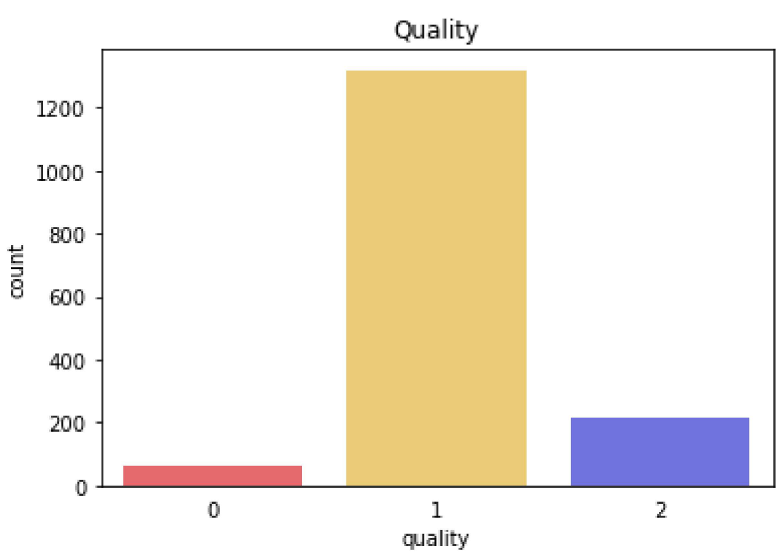
```
#Caracteristicas de la variable target
pd.DataFrame( { 'quality': df["quality"].value_counts().index , 'counts': df["qua
```

Out[8]:

	quality	counts
0	0	63
1	1	1319
2	2	217

In [9]:

```
#Caracteristicas de la variable target
ax = sns.countplot(data=df, x='quality', palette=['#FA5458','#FDD563','#5F63F1'])
ax.set(xticklabels=['0','1','2'], title="Quality")
ax.tick_params(bottom=False)
```



In [10]:
```
#Separamos los datos entre features y target
X = df.drop('quality',axis=1)
y = df[['quality']]

print(X.columns)
print(y.columns)
```

```
Index(['fixed acidity', 'volatile acidity', 'citric acid', 'residual sugar',
       'chlorides', 'free sulfur dioxide', 'total sulfur dioxide', 'density',
       'pH', 'sulphates', 'alcohol'],
      dtype='object')
Index(['quality'], dtype='object')
```

In [11]:
```
#Caracteristicas de los features
X.describe().transpose()
```

Out[11]:

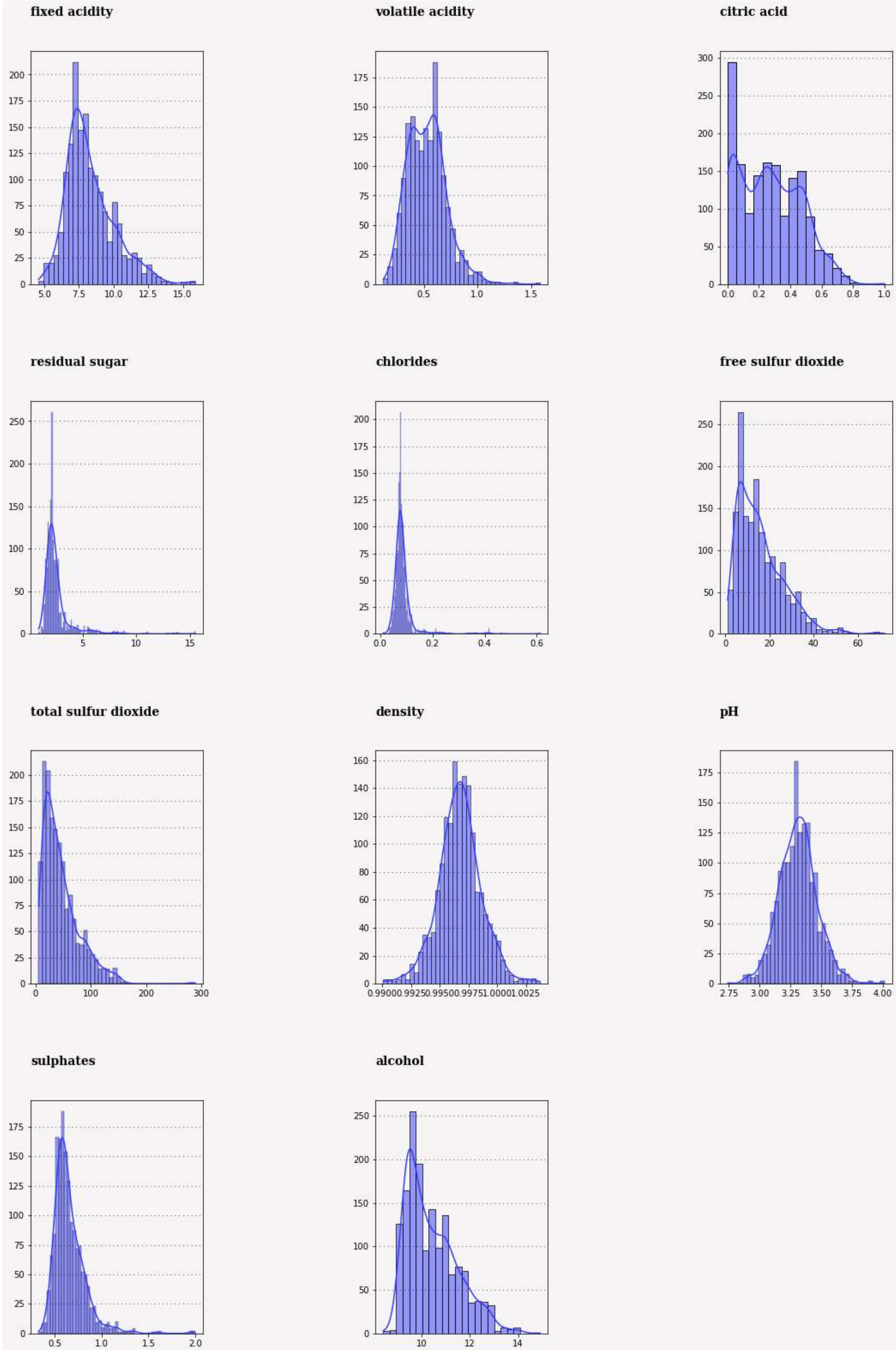
	count	mean	std	min	25%	50%	75%	max
fixed acidity	1599.0	8.319637	1.741096	4.60000	7.1000	7.90000	9.200000	15.90000
volatile acidity	1599.0	0.527821	0.179060	0.12000	0.3900	0.52000	0.640000	1.58000
citric acid	1599.0	0.270976	0.194801	0.00000	0.0900	0.26000	0.420000	1.00000
residual sugar	1599.0	2.538806	1.409928	0.90000	1.9000	2.20000	2.600000	15.50000
chlorides	1599.0	0.087467	0.047065	0.01200	0.0700	0.07900	0.090000	0.61100
free sulfur dioxide	1599.0	15.874922	10.460157	1.00000	7.0000	14.00000	21.000000	72.00000
total sulfur dioxide	1599.0	46.467792	32.895324	6.00000	22.0000	38.00000	62.000000	289.00000
density	1599.0	0.996747	0.001887	0.99007	0.9956	0.99675	0.997835	1.00369
pH	1599.0	3.311113	0.154386	2.74000	3.2100	3.31000	3.400000	4.01000
sulphates	1599.0	0.658149	0.169507	0.33000	0.5500	0.62000	0.730000	2.00000
alcohol	1599.0	10.422983	1.065668	8.40000	9.5000	10.20000	11.100000	14.90000

In [12]:
```
#Visualizamos la distribución de los features
fig = plt.figure(figsize=(18,35))
gs = fig.add_gridspec(5,3)
gs.update(wspace=1, hspace=0.5)

ax1 = fig.add_subplot(gs[0,0])
ax2 = fig.add_subplot(gs[0,1])
ax3 = fig.add_subplot(gs[0,2])
ax4 = fig.add_subplot(gs[1,0])
ax5 = fig.add_subplot(gs[1,1])
ax6 = fig.add_subplot(gs[1,2])
ax7 = fig.add_subplot(gs[2,0])
ax8 = fig.add_subplot(gs[2,1])
ax9 = fig.add_subplot(gs[2,2])
ax10 = fig.add_subplot(gs[3,0])
ax11 = fig.add_subplot(gs[3,1])

background_color = "#f6f5f5"
color_palette = ["#FA5458","#FDD563","#5F63F1"]

fig.patch.set_facecolor(background_color)
ax1.set_facecolor(background_color)
ax2.set_facecolor(background_color)
ax3.set_facecolor(background_color)
ax4.set_facecolor(background_color)
ax5.set_facecolor(background_color)
ax6.set_facecolor(background_color)
ax7.set_facecolor(background_color)
ax8.set_facecolor(background_color)
ax9.set_facecolor(background_color)
ax10.set_facecolor(background_color)
ax11.set_facecolor(background_color)

ax1.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax1,x=df['fixed acidity'],color= "#3339FF", kde=True)
Xstart, Xend = ax1.get_xlim()
Ystart, Yend = ax1.get_ylim()
ax1.text(Xstart, Yend+(Yend*0.15), 'fixed acidity', fontsize=14, fontweight='bolc
ax1.set_xlabel("")
ax1.set_ylabel("")

ax2.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax2,x=df['volatile acidity'],color= "#3339FF", kde=True)
Xstart, Xend = ax2.get_xlim()
Ystart, Yend = ax2.get_ylim()
ax2.text(Xstart, Yend+(Yend*0.15), 'volatile acidity', fontsize=14, fontweight='b
ax2.set_xlabel("")
ax2.set_ylabel("")

ax3.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax3,x=df['citric acid'],color= "#3339FF", kde=True)
Xstart, Xend = ax3.get_xlim()
Ystart, Yend = ax3.get_ylim()
ax3.text(Xstart, Yend+(Yend*0.15), 'citric acid', fontsize=14, fontweight='bold',
ax3.set_xlabel("")
ax3.set_ylabel("")
```

```
ax4.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax4,x=df['residual sugar'],color= "#3339FF", kde=True)
Xstart, Xend = ax4.get_xlim()
Ystart, Yend = ax4.get_ylim()
ax4.text(Xstart, Yend+(Yend*0.15), 'residual sugar', fontsize=14, fontweight='bol
ax4.set_xlabel("")
ax4.set_ylabel("")

ax5.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax5,x=df['chlorides'],color= "#3339FF", kde=True)
Xstart, Xend = ax5.get_xlim()
Ystart, Yend = ax5.get_ylim()
ax5.text(Xstart, Yend+(Yend*0.15), 'chlorides', fontsize=14, fontweight='bold', f
ax5.set_xlabel("")
ax5.set_ylabel("")

ax6.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax6,x=df['free sulfur dioxide'],color= "#3339FF", kde=True)
Xstart, Xend = ax6.get_xlim()
Ystart, Yend = ax6.get_ylim()
ax6.text(Xstart, Yend+(Yend*0.15), 'free sulfur dioxide', fontsize=14, fontweight
ax6.set_xlabel("")
ax6.set_ylabel("")

ax7.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax7,x=df['total sulfur dioxide'],color= "#3339FF", kde=True)
Xstart, Xend = ax7.get_xlim()
Ystart, Yend = ax7.get_ylim()
ax7.text(Xstart, Yend+(Yend*0.15), 'total sulfur dioxide', fontsize=14, fontweigh
ax7.set_xlabel("")
ax7.set_ylabel("")

ax8.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax8,x=df['density'],color= "#3339FF", kde=True)
Xstart, Xend = ax8.get_xlim()
Ystart, Yend = ax8.get_ylim()
ax8.text(Xstart, Yend+(Yend*0.15), 'density', fontsize=14, fontweight='bold', fon
ax8.set_xlabel("")
ax8.set_ylabel("")

ax9.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax9,x=df['pH'],color= "#3339FF", kde=True)
Xstart, Xend = ax9.get_xlim()
Ystart, Yend = ax9.get_ylim()
ax9.text(Xstart, Yend+(Yend*0.15), 'pH', fontsize=14, fontweight='bold', fontfami
ax9.set_xlabel("")
ax9.set_ylabel("")

ax10.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax10,x=df['sulphates'],color= "#3339FF", kde=True)
Xstart, Xend = ax10.get_xlim()
Ystart, Yend = ax10.get_ylim()
ax10.text(Xstart, Yend+(Yend*0.15), 'sulphates', fontsize=14, fontweight='bold',
ax10.set_xlabel("")
ax10.set_ylabel("")
```

```
ax11.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.histplot(ax=ax11,x=df['alcohol'],color= "#3339FF", kde=True)
Xstart, Xend = ax11.get_xlim()
Ystart, Yend = ax11.get_ylim()
ax11.text(Xstart, Yend+(Yend*0.15), 'alcohol', fontsize=14, fontweight='bold', fo
ax11.set_xlabel("")
ax11.set_ylabel("")
```

Out[12]: Text(0, 0.5, '')

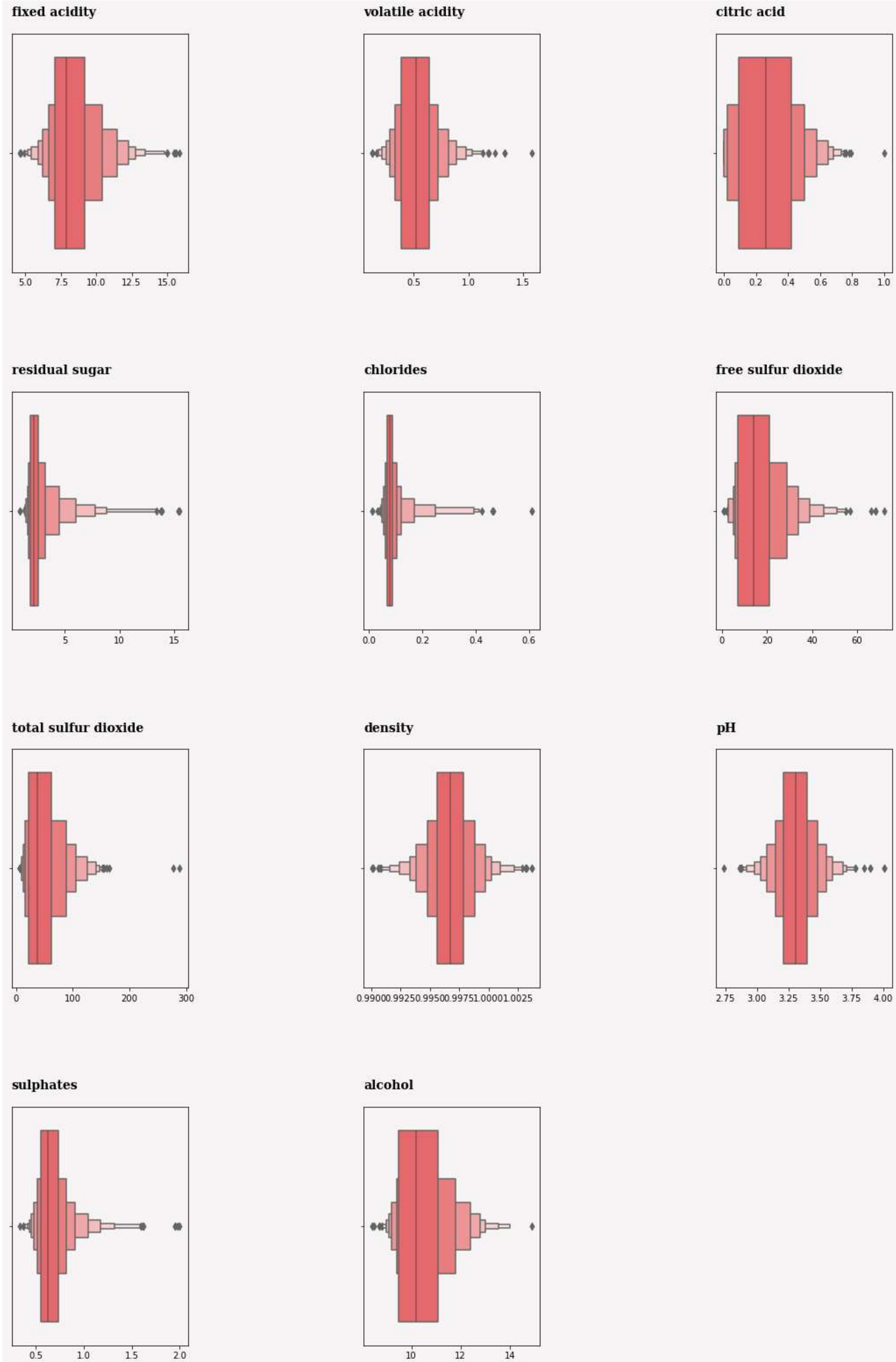

fixed acidity
0
25
50
75
100
125
150
175
200
5.0
7.5
10.0
12.5
15.0
volatile acidity
0
25
50
75
100
125
150
175
0.5
1.0
1.5
citric acid
0
50
100
150
200
250
300
0.0
0.2
0.4
0.6
0.8
1.0
residual sugar
0
50
100
150
200
250
5
10
15
chlorides
0
25
50
75
100
125
150
175
200
0.0
0.2
0.4
0.6
free sulfur dioxide
0
50
100
150
200
250
0
20
40
60
total sulfur dioxide
0
25
50
75
100
125
150
175
200
0
100
200
300
density
0
20
40
60
80
100
120
140
160
0.99000.99250.99500.99751.00001.0025
pH
0
25
50
75
100
125
150
175
2.75
3.00
3.25
3.50
3.75
4.00
sulphates
0
25
50
75
100
125
150
175
0.5
1.0
1.5
2.0
alcohol
0
50
100
150
200
250
10
12
14

In [13]:
```
#Visualizamos la distribución de los features
fig = plt.figure(figsize=(18,35))
gs = fig.add_gridspec(5,3)
gs.update(wspace=1, hspace=0.5)
ax1 = fig.add_subplot(gs[0,0])
ax2 = fig.add_subplot(gs[0,1])
ax3 = fig.add_subplot(gs[0,2])
ax4 = fig.add_subplot(gs[1,0])
ax5 = fig.add_subplot(gs[1,1])
ax6 = fig.add_subplot(gs[1,2])
ax7 = fig.add_subplot(gs[2,0])
ax8 = fig.add_subplot(gs[2,1])
ax9 = fig.add_subplot(gs[2,2])
ax10 = fig.add_subplot(gs[3,0])
ax11 = fig.add_subplot(gs[3,1])

background_color = "#f6f5f5"
color_palette = ["#FA5458","#FDD563","#5F63F1"]

fig.patch.set_facecolor(background_color)
ax1.set_facecolor(background_color)
ax2.set_facecolor(background_color)
ax3.set_facecolor(background_color)
ax4.set_facecolor(background_color)
ax5.set_facecolor(background_color)
ax6.set_facecolor(background_color)
ax7.set_facecolor(background_color)
ax8.set_facecolor(background_color)
ax9.set_facecolor(background_color)
ax10.set_facecolor(background_color)
ax11.set_facecolor(background_color)

ax1.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax1,x=df['fixed acidity'],color= "#FA5458")
Xstart, Xend = ax1.get_xlim()
Ystart, Yend = ax1.get_ylim()
ax1.text(Xstart, Yend+(Yend*0.15), 'fixed acidity', fontsize=14, fontweight='bol
ax1.set_xlabel("")
ax1.set_ylabel("")

ax2.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax2,x=df['volatile acidity'],color= "#FA5458")
Xstart, Xend = ax2.get_xlim()
Ystart, Yend = ax2.get_ylim()
ax2.text(Xstart, Yend+(Yend*0.15), 'volatile acidity', fontsize=14, fontweight='b
ax2.set_xlabel("")
ax2.set_ylabel("")

ax3.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax3,x=df['citric acid'],color= "#FA5458")
Xstart, Xend = ax3.get_xlim()
Ystart, Yend = ax3.get_ylim()
ax3.text(Xstart, Yend+(Yend*0.15), 'citric acid', fontsize=14, fontweight='bold',
ax3.set_xlabel("")
ax3.set_ylabel("")
```

```
ax4.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax4,x=df['residual sugar'],color= "#FA5458")
Xstart, Xend = ax4.get_xlim()
Ystart, Yend = ax4.get_ylim()
ax4.text(Xstart, Yend+(Yend*0.15), 'residual sugar', fontsize=14, fontweight='bol
ax4.set_xlabel("")
ax4.set_ylabel("")

ax5.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax5,x=df['chlorides'],color= "#FA5458")
Xstart, Xend = ax5.get_xlim()
Ystart, Yend = ax5.get_ylim()
ax5.text(Xstart, Yend+(Yend*0.15), 'chlorides', fontsize=14, fontweight='bold', f
ax5.set_xlabel("")
ax5.set_ylabel("")

ax6.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax6,x=df['free sulfur dioxide'],color= "#FA5458")
Xstart, Xend = ax6.get_xlim()
Ystart, Yend = ax6.get_ylim()
ax6.text(Xstart, Yend+(Yend*0.15), 'free sulfur dioxide', fontsize=14, fontweight
ax6.set_xlabel("")
ax6.set_ylabel("")

ax7.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax7,x=df['total sulfur dioxide'],color= "#FA5458")
Xstart, Xend = ax7.get_xlim()
Ystart, Yend = ax7.get_ylim()
ax7.text(Xstart, Yend+(Yend*0.15), 'total sulfur dioxide', fontsize=14, fontweigh
ax7.set_xlabel("")
ax7.set_ylabel("")

ax8.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax8,x=df['density'],color= "#FA5458")
Xstart, Xend = ax8.get_xlim()
Ystart, Yend = ax8.get_ylim()
ax8.text(Xstart, Yend+(Yend*0.15), 'density', fontsize=14, fontweight='bold', fon
ax8.set_xlabel("")
ax8.set_ylabel("")

ax9.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax9,x=df['pH'],color= "#FA5458")
Xstart, Xend = ax9.get_xlim()
Ystart, Yend = ax9.get_ylim()
ax9.text(Xstart, Yend+(Yend*0.15), 'pH', fontsize=14, fontweight='bold', fontfami
ax9.set_xlabel("")
ax9.set_ylabel("")

ax10.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
sns.boxenplot(ax=ax10,x=df['sulphates'],color= "#FA5458")
Xstart, Xend = ax10.get_xlim()
Ystart, Yend = ax10.get_ylim()
ax10.text(Xstart, Yend+(Yend*0.15), 'sulphates', fontsize=14, fontweight='bold',
ax10.set_xlabel("")
ax10.set_ylabel("")

ax11.grid(color='#000000', linestyle=':', axis='y', zorder=0,  dashes=(1,5))
```

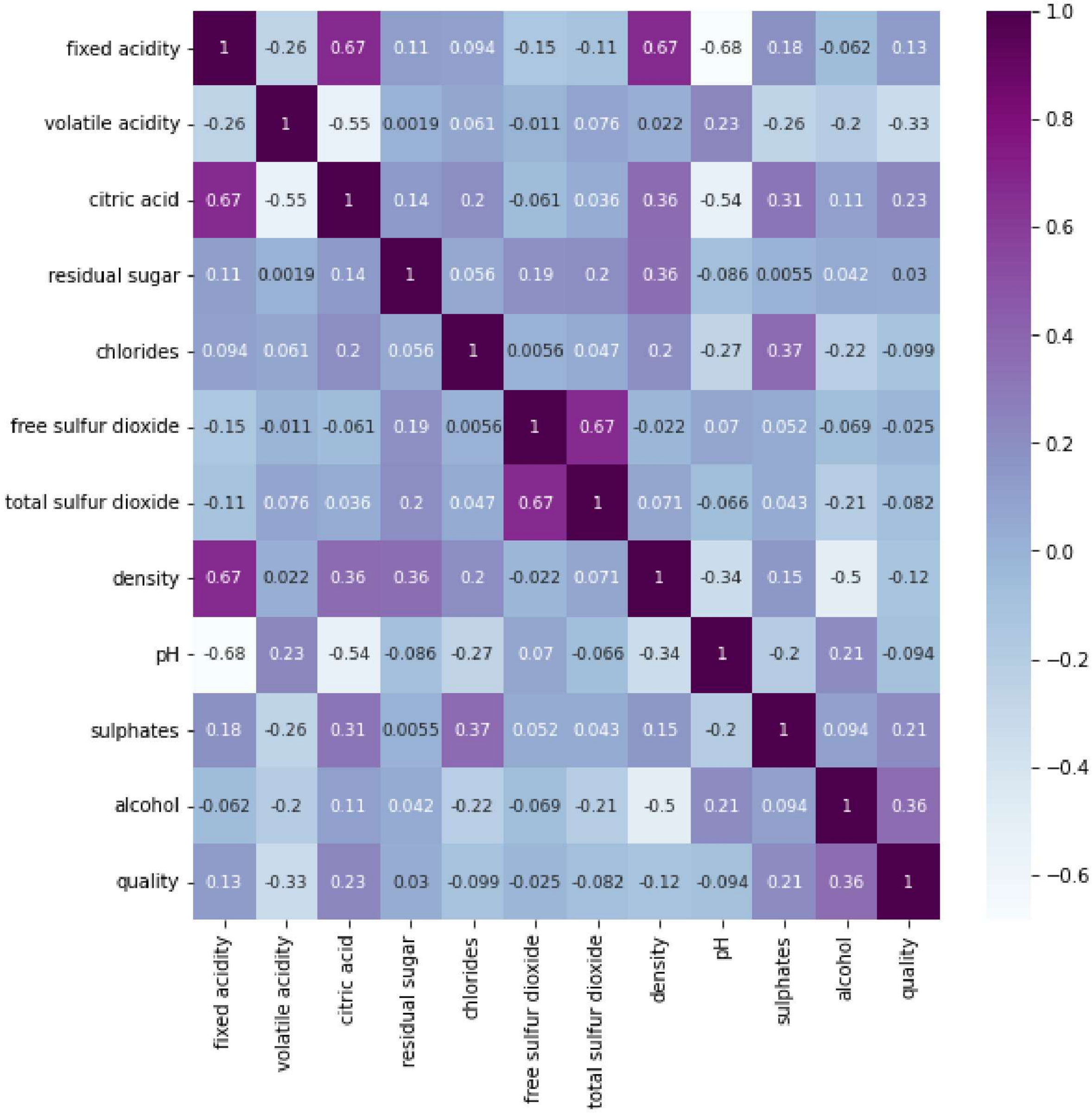
```
sns.boxenplot(ax=ax11,x=df['alcohol'],color= "#FA5458")
Xstart, Xend = ax11.get_xlim()
Ystart, Yend = ax11.get_ylim()
ax11.text(Xstart, Yend+(Yend*0.15), 'alcohol', fontsize=14, fontweight='bold', fo
ax11.set_xlabel("")
ax11.set_ylabel("")
```

Out[13]: Text(0, 0.5, '')


fixed acidity
5.0 7.5 10.0 12.5 15.0
volatile acidity
0.5 1.0 1.5
citric acid
0.0 0.2 0.4 0.6 0.8 1.0
residual sugar
5 10 15
chlorides
0.0 0.2 0.4 0.6
free sulfur dioxide
0 20 40 60
total sulfur dioxide
0 100 200 300
density
0.99000.99250.99500.99751.00001.0025
pH
2.75 3.00 3.25 3.50 3.75 4.00
sulphates
0.5 1.0 1.5 2.0
alcohol
10 12 14

In [14]:
```python
#Visualizamos la correlación entre las variables
plt.figure(figsize=(9, 9))
sns.heatmap(df.corr(), vmax = 1, annot=True, annot_kws={"size": 9}, cmap="BuPu")
```

Out[14]: <AxesSubplot:>



In [15]:
```
#Visualizamos la correlación entre las variables
df.corr().transpose().loc[:, ["quality"]].sort_values(by="quality",ascending=Fals
```

Out[15]:

	quality
quality	1.000000
alcohol	0.361363
citric acid	0.228930
sulphates	0.205409
fixed acidity	0.125886
residual sugar	0.030153
free sulfur dioxide	-0.025075
total sulfur dioxide	-0.081960
pH	-0.093946
chlorides	-0.098829
density	-0.123566
volatile acidity	-0.333816

In [16]:
```
#Escalamos los datos
scaler = RobustScaler()
X = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.1, random_
```

In [17]:
```
#Definimos un diccionario donde vamos a registrar la medición del accuracy
models_accuracy = dict()
```

In [18]:
```
#Logistic Regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_proba = logreg.predict_proba(X_test)
y_pred = np.argmax(y_pred_proba,axis=1)
models_accuracy["Logistic Regression"] = accuracy_score(y_pred,y_test)
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.98      0.86      0.92       152
           2       0.23      0.62      0.33         8

    accuracy                           0.85       160
   macro avg       0.40      0.50      0.42       160
weighted avg       0.94      0.85      0.89       160
```

In [19]:
```
#KNN
param_grid = {'n_neighbors':np.arange(1,50), 'weights':['uniform','distance'], 'l
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn,param_grid,cv=5)
knn_cv.fit(X_train, y_train)
y_pred = knn_cv.predict(X_test)
print(knn_cv.best_params_)
print(knn_cv.best_score_)
models_accuracy["KNN"] = accuracy_score(y_pred,y_test)
print(classification_report(y_pred,y_test))
```

```
{'leaf_size': 1, 'n_neighbors': 12, 'weights': 'distance'}
0.8693500774293458
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.97      0.91      0.94       143
           2       0.59      0.76      0.67        17

    accuracy                           0.89       160
   macro avg       0.52      0.56      0.54       160
weighted avg       0.93      0.89      0.91       160
```

In [20]:
```
#Decision Tree
param_grid  = {"max_depth":np.arange(2,10), "min_samples_leaf":np.arange(0.02, 0.
dt = DecisionTreeClassifier()
grid_dt = GridSearchCV(estimator = dt,
                      param_grid = param_grid,
                      scoring="accuracy",
                      cv=10,
                      n_jobs=-1)
grid_dt.fit(X_train, y_train)
y_pred = grid_dt.predict(X_test)
print(grid_dt.best_params_)
print(grid_dt.best_score_)
models_accuracy["Decision Trees"] = accuracy_score(y_pred,y_test)
print(classification_report(y_pred,y_test))
```

```
{'max_depth': 8, 'max_features': 0.8, 'min_samples_leaf': 0.02}
0.8436674436674437
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.96      0.89      0.92       145
           2       0.45      0.67      0.54        15

    accuracy                           0.87       160
   macro avg       0.47      0.52      0.49       160
weighted avg       0.92      0.87      0.89       160
```

In [21]:
```
#Random Forest
params_rf = {'n_estimators':[100,200,300,400,500],
             'max_depth':[4,6,8,10,12,14],
             'max_features':['log2','sqrt']}
rf = RandomForestClassifier()
grid_rf = GridSearchCV(estimator = rf,
                       param_grid = params_rf,
                       cv=3,
                       scoring = 'neg_mean_squared_error',
                       verbose = 1,
                       n_jobs = -1)
grid_rf.fit(X_train, y_train)
y_pred = grid_rf.predict(X_test)
print(grid_rf.best_params_)
print(grid_rf.best_score_)
models_accuracy["Random Forest"] = accuracy_score(y_pred,y_test)
print(classification_report(y_pred,y_test))
```

```
Fitting 3 folds for each of 60 candidates, totalling 180 fits
{'max_depth': 12, 'max_features': 'log2', 'n_estimators': 100}
-0.13550075388540941
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.96      0.89      0.92       145
           2       0.45      0.67      0.54        15

    accuracy                           0.87       160
   macro avg       0.47      0.52      0.49       160
weighted avg       0.92      0.87      0.89       160
```

In [22]:
```
#Definimos la performance para cada modelo (diccionario)
classifiers = [('Logistic Regression',logreg),
              ('K Nearest Neighbors', knn),
              ('Classification Tree', dt)]
for clf_name,clf in classifiers:
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(clf_name, accuracy_score(y_test,y_pred))
vc = VotingClassifier(estimators = classifiers)
vc.fit(X_train, y_train)
y_pred = vc.predict(X_test)
models_accuracy["Voting Classifier"] = accuracy_score(y_pred,y_test)
print(accuracy_score(y_test, y_pred))
```

```
Logistic Regression 0.85
K Nearest Neighbors 0.85
Classification Tree 0.85625
0.86875
```

In [23]:
```
#Definimos la performance para cada modelo (lista)
model = []
accuracy = []
for index,col in enumerate(models_accuracy):
    model.append(col)
    accuracy.append(models_accuracy[col])
print(model)
print(accuracy)
```

```
['Logistic Regression', 'KNN', 'Decision Trees', 'Random Forest', 'Voting Class
ifier']
[0.85, 0.89375, 0.86875, 0.86875, 0.86875]
```

In [24]:
```
#Visualización de la performance para cada modelo
acc = pd.DataFrame({"Models":model, "Accuracy":accuracy})
plt.figure(figsize=(8,6))
sns.scatterplot(x = 'Models', y='Accuracy', data=acc, color='#3339FF',cmap=True)
plt.show()
```