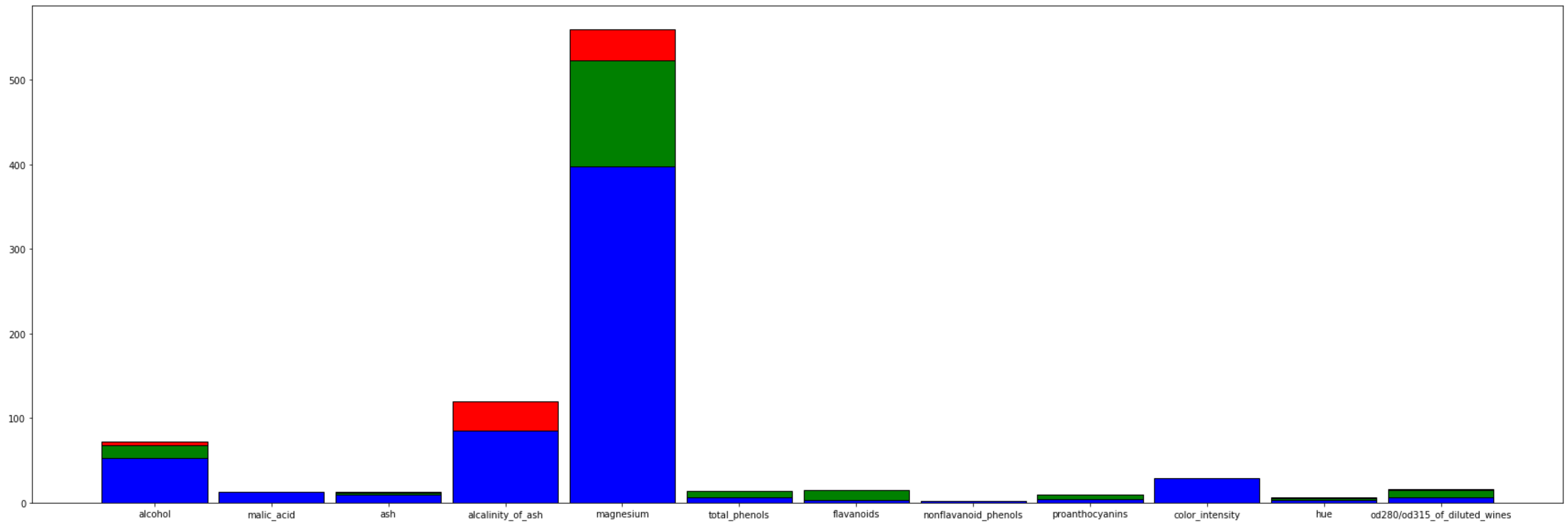
In [9]:
```
from sklearn.datasets import load_wine
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix,ConfusionMatrixDisplay
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
from sklearn.metrics import classification_report

wine_data = load_wine()
df = pd.DataFrame(data=wine_data['data'],columns=wine_data['feature_names'])
df['class_type']=wine_data['target']

print(wine_data.DESCR)
```

```
.. _wine_dataset:

Wine recognition dataset
------------------------

**Data Set Characteristics:**

:Number of Instances: 178
:Number of Attributes: 13 numeric, predictive attributes and the class
:Attribute Information:
    - Alcohol
    - Malic acid
    - Ash
    - Alcalinity of ash
    - Magnesium
    - Total phenols
    - Flavanoids
    - Nonflavanoid phenols
    - Proanthocyanins
    - Color intensity
```

```
    - Hue
    - OD280/OD315 of diluted wines
    - Proline
    - class:
        - class_0
        - class_1
        - class_2

:Summary Statistics:

============================= ==== ===== ======= =====
                                Min   Max   Mean     SD
============================= ==== ===== ======= =====
Alcohol:                      11.0  14.8    13.0   0.8
Malic Acid:                   0.74  5.80    2.34  1.12
Ash:                          1.36  3.23    2.36  0.27
Alcalinity of Ash:            10.6  30.0    19.5   3.3
Magnesium:                    70.0 162.0    99.7  14.3
Total Phenols:                0.98  3.88    2.29  0.63
Flavanoids:                   0.34  5.08    2.03  1.00
Nonflavanoid Phenols:         0.13  0.66    0.36  0.12
Proanthocyanins:              0.41  3.58    1.59  0.57
Colour Intensity:              1.3  13.0     5.1   2.3
Hue:                          0.48  1.71    0.96  0.23
OD280/OD315 of diluted wines: 1.27  4.00    2.61  0.71
Proline:                       278  1680     746   315
============================= ==== ===== ======= =====

:Missing Attribute Values: None
:Class Distribution: class_0 (59), class_1 (71), class_2 (48)
:Creator: R.A. Fisher
:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
:Date: July, 1988

This is a copy of UCI ML Wine recognition datasets.
https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data (https://archive.ics.uci.edu/ml/m
achine-learning-databases/wine/wine.data)

The data is the results of a chemical analysis of wines grown in the same
region in Italy by three different cultivators. There are thirteen different
measurements taken for different constituents found in the three types of
```

```
wine.

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies,
Via Brigata Salerno, 16147 Genoa, Italy.

Citation:

Lichman, M. (2013). UCI Machine Learning Repository
[https://archive.ics.uci.edu/ml]. Irvine, CA: University of California,
School of Information and Computer Science.

|details-start|
**References**
|details-split|

(1) S. Aeberhard, D. Coomans and O. de Vel,
Comparison of Classifiers in High Dimensional Settings,
Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
Mathematics and Statistics, James Cook University of North Queensland.
(Also submitted to Technometrics).

The data was used with many others for comparing various
classifiers. The classes are separable, though only RDA
has achieved 100% correct classification.
(RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
(All results using the leave-one-out technique)

(2) S. Aeberhard, D. Coomans and O. de Vel,
"THE CLASSIFICATION PERFORMANCE OF RDA"
Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
Mathematics and Statistics, James Cook University of North Queensland.
(Also submitted to Journal of Chemometrics).

|details-end|
```

In [3]:
```
col_count = 12
x = df.columns[0:col_count]
data_class = [np.zeros(col_count),np.zeros(col_count),np.zeros(col_count)]
for i in range(len(df.values)):
    data_class[int(df.values[i][13])] += df.values[i][0:col_count]

plt.figure(figsize = (30, 10))
plt.bar(x,data_class[1]/len(data_class[1]), color='red', edgecolor='black',width=0.9)
plt.bar(x,data_class[0]/len(data_class[0]), color='green', edgecolor='black',width=0.9,alpha=1)
plt.bar(x,data_class[2]/len(data_class[2]), color='blue', edgecolor='black',width=0.9,alpha=1)
plt.show()

# class red highest in:
#   alchol
#   alcalinity of ash
#   magnesium
#   proathocyanins
#   hue
#   non flavonoids
#   0d280/315 of diluated vines
#   ash

# class green highest in:
#   flavanoids
#   total phenols
#   proline

# class green is highest in
#   color intensity
#   Malic Acid

# the 3 classes are grouped by the bases of chemical attributes
```



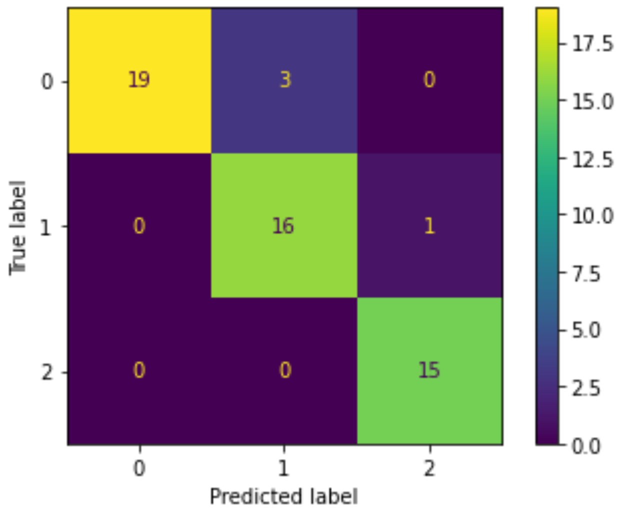
In [4]:

```
#Split arrays or matrices into random train and test subsets.
X_train, X_test, y_train, y_test = train_test_split(wine_data.data, wine_data.target, test_size=0.30)
```

In [5]:
```
#for smaller datasets solver should be liblinear
model = LogisticRegression(solver='liblinear')
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

cm = confusion_matrix(y_test,y_pred)

sensitivity = metrics.recall_score(y_test,y_pred,average=None)
specificity = metrics.recall_score(y_test,y_pred,average=None,labels=['2','1','0'])

target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_test,y_pred, target_names=target_names))
print(f'sensitivity: {sensitivity}\nspecificity: {specificity}')

disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=model.classes_)
disp.plot()
```

```
              precision    recall  f1-score   support

     class 0       1.00      0.86      0.93        22
     class 1       0.84      0.94      0.89        17
     class 2       0.94      1.00      0.97        15

    accuracy                           0.93        54
   macro avg       0.93      0.93      0.93        54
weighted avg       0.93      0.93      0.93        54

sensitivity: [0.86363636 0.94117647 1.        ]
specificity: [1.         0.94117647 0.86363636]
```

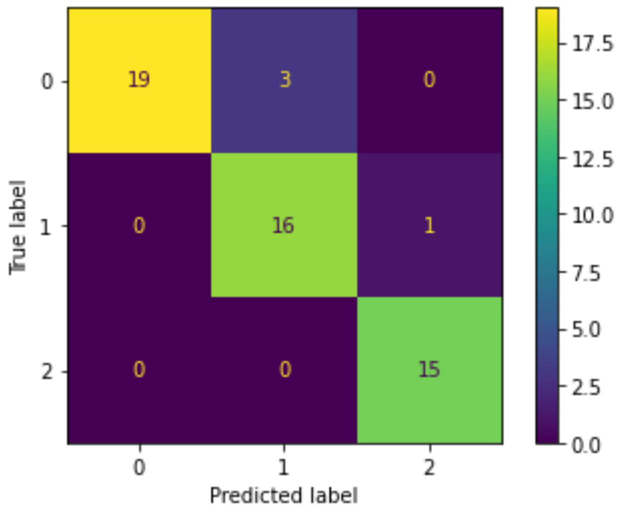
Out[5]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x719f6aca3700>
```


True label
Predicted label
0
1
2
19
3
16
15
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0

In [6]:
```
# rf = RandomForestClassifier()
model = RandomForestClassifier()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

cm = confusion_matrix(y_test,y_pred)

sensitivity = metrics.recall_score(y_test,y_pred,average=None)
specificity = metrics.recall_score(y_test,y_pred,average=None,labels=['2','1','0'])

target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_test,y_pred, target_names=target_names))
print(f'sensitivity: {sensitivity}\nspecificity: {specificity}')

disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=model.classes_)
disp.plot()
```

```
              precision    recall  f1-score   support

     class 0       1.00      0.86      0.93        22
     class 1       0.84      0.94      0.89        17
     class 2       0.94      1.00      0.97        15

    accuracy                           0.93        54
   macro avg       0.93      0.93      0.93        54
weighted avg       0.93      0.93      0.93        54

sensitivity: [0.86363636 0.94117647 1.        ]
specificity: [1.         0.94117647 0.86363636]
```

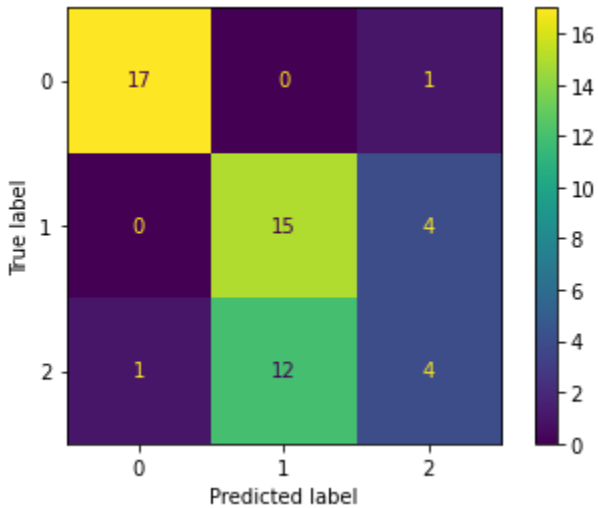
Out[6]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x719fce7d2920>
```


19
3
0
0
16
1
0
0
15
True label
Predicted label
0
1
2
0
1
2
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0

In [123]:
```
#creates KNN Classifier object and scans 50 nearest neighbours
model = KNeighborsClassifier(n_neighbors=50)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

cm = confusion_matrix(y_test,y_pred)

sensitivity = metrics.recall_score(y_test,y_pred,average=None)
specificity = metrics.recall_score(y_test,y_pred,average=None,labels=['2','1','0'])

target_names = ['class 0', 'class 1', 'class 2']
print(classification_report(y_test,y_pred, target_names=target_names))
print(f'sensitivity: {sensitivity}\nspecificity: {specificity}')

disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=model.classes_)
disp.plot()
```

```
              precision    recall  f1-score   support

     class 0       0.94      0.94      0.94        18
     class 1       0.56      0.79      0.65        19
     class 2       0.44      0.24      0.31        17

    accuracy                           0.67        54
   macro avg       0.65      0.66      0.63        54
weighted avg       0.65      0.67      0.64        54

sensitivity: [0.94444444 0.78947368 0.23529412]
specificity: [0.23529412 0.78947368 0.94444444]
```

Out[123]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x73f582475780>
```


17
0
1
0
15
4
1
12
4
True label
Predicted label
0
1
2
0
1
2
16
14
12
10
8
6
4
2
0