Chapter_4_Building_Good_Training_Sets_Data_Preprocessing

March 18, 2024

0.1 Identifying missing values in tabular data

[2]:
```
import pandas as pd
from io import StringIO
```

[3]:
```
csv_data = \
'''A,B,C,D
1.0,2.0,3.0,4.0
5.0,6.0,,8.0
10.0,11.0,12.0,'''
```

[4]:
```
csv_data
```

[4]:
```
'A,B,C,D\n1.0,2.0,3.0,4.0\n5.0,6.0,,8.0\n10.0,11.0,12.0,'
```

[5]:
```
df = pd.read_csv(StringIO(csv_data))
```

[6]:
```
df
```

[6]:
```
      A     B     C    D
0   1.0   2.0   3.0  4.0
1   5.0   6.0   NaN  8.0
2  10.0  11.0  12.0  NaN
```

[7]:
```
df.isnull().sum()
```

[7]:
```
A    0
B    0
C    1
D    1
dtype: int64
```

[8]:
```
df.values
```

[8]:
```
array([[ 1.,  2.,  3.,  4.],
       [ 5.,  6., nan,  8.],
       [10., 11., 12., nan]])
```

[9]:
```
df.dropna(axis=0)
```

[9]:
```
     A    B    C    D
0  1.0  2.0  3.0  4.0
```

[10]:
```
df.dropna(axis=1)
```

[10]:
```
      A     B
0   1.0   2.0
1   5.0   6.0
2  10.0  11.0
```

[11]:
```
# only drop rows where all columns are NaN
# (returns the whole array here since we don't
# have a row with where all values are NaN
df.dropna(how='all')
```

[11]:
```
      A     B     C    D
0   1.0   2.0   3.0  4.0
1   5.0   6.0   NaN  8.0
2  10.0  11.0  12.0  NaN
```

[12]:
```
# drop rows that have less than 4 real values
df.dropna(thresh=4)
```

[12]:
```
     A    B    C    D
0  1.0  2.0  3.0  4.0
```

[13]:
```
# only drop rows where NaN appear in specific columns (here: 'C')
df.dropna(subset=['C'])
```

[13]:
```
      A     B     C    D
0   1.0   2.0   3.0  4.0
2  10.0  11.0  12.0  NaN
```

0.2 Imputing missing values

[14]:
```
# ! pip install scikit-learn==0.20.4 --user
```

[15]:
```
from sklearn.preprocessing import Imputer
imr = Imputer(missing_values='NaN', strategy='mean', axis=0)
imr = imr.fit(df.values)
imputed_data = imr.transform(df.values)
imputed_data
```

```
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\utils\deprecation.py:58:
DeprecationWarning: Class Imputer is deprecated; Imputer was deprecated in
version 0.20 and will be removed in 0.22. Import impute.SimpleImputer from
```

```
sklearn instead.
  warnings.warn(msg, category=DeprecationWarning)
```

[15]:
```
array([[ 1. ,  2. ,  3. ,  4. ],
       [ 5. ,  6. ,  7.5,  8. ],
       [10. , 11. , 12. ,  6. ]])
```

0.3 Handling categorical data

[16]:
```python
import pandas as pd
df = pd.DataFrame([['green', 'M', 10.1, 'class1'],['red', 'L', 13.5,␣
↪'class2'],['blue', 'XL', 15.3, 'class1']])
df.columns = ['color', 'size', 'price', 'classlabel']
df
```

[16]:

	color	size	price	classlabel
0	green	M	10.1	class1
1	red	L	13.5	class2
2	blue	XL	15.3	class1

0.4 Mapping ordinal features

[17]:
```python
size_mapping = {
    'XL': 3,
    'L': 2,
    'M': 1}
df['size'] = df['size'].map(size_mapping)
df
```

[17]:

	color	size	price	classlabel
0	green	1	10.1	class1
1	red	2	13.5	class2
2	blue	3	15.3	class1

[18]:
```python
inv_size_mapping = {v: k for k, v in size_mapping.items()}
df['size'].map(inv_size_mapping)
```

[18]:
```
0     M
1     L
2    XL
Name: size, dtype: object
```

0.5 Encoding class labels

[19]:
```
import numpy as np
class_mapping = {label:idx for idx,label in enumerate(np.
 ↪unique(df['classlabel']))}
class_mapping
```

[19]:
```
{'class1': 0, 'class2': 1}
```

[20]:
```
df['classlabel'] = df['classlabel'].map(class_mapping)
df
```

[20]:

	color	size	price	classlabel
0	green	1	10.1	0
1	red	2	13.5	1
2	blue	3	15.3	0

[21]:
```
inv_class_mapping = {v: k for k, v in class_mapping.items()}
df['classlabel'] = df['classlabel'].map(inv_class_mapping)
df
```

[21]:

	color	size	price	classlabel
0	green	1	10.1	class1
1	red	2	13.5	class2
2	blue	3	15.3	class1

[22]:
```
from sklearn.preprocessing import LabelEncoder
class_le = LabelEncoder()
y = class_le.fit_transform(df['classlabel'].values)
y
```

[22]:
```
array([0, 1, 0])
```

[23]:
```
class_le.inverse_transform(y)
```

[23]:
```
array(['class1', 'class2', 'class1'], dtype=object)
```

0.6 Performing one-hot encoding on nominal features

[24]:
```
X = df[['color', 'size', 'price']].values
color_le = LabelEncoder()
X[:, 0] = color_le.fit_transform(X[:, 0])
X
```

[24]:
```
array([[1, 1, 10.1],
       [2, 2, 13.5],
       [0, 3, 15.3]], dtype=object)
```

[25]:
```
## Performing one-hot encoding on nominal features
```

[26]:
```python
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(categorical_features=[0])
ohe.fit_transform(X).toarray()
```

```
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\preprocessing\_encoders.py:371:
FutureWarning: The handling of integer data will change in version 0.22.
Currently, the categories are determined based on the range [0, max(values)],
while in the future they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the
categories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\preprocessing\_encoders.py:392:
DeprecationWarning: The 'categorical_features' keyword is deprecated in version
0.20 and will be removed in 0.22. You can use the ColumnTransformer instead.
  "use the ColumnTransformer instead.", DeprecationWarning)
```

[26]:
```
array([[ 0. ,  1. ,  0. ,  1. , 10.1],
       [ 0. ,  0. ,  1. ,  2. , 13.5],
       [ 1. ,  0. ,  0. ,  3. , 15.3]])
```

[27]:
```python
pd.get_dummies(df[['price', 'color', 'size']])
```

[27]:

	price	size	color_blue	color_green	color_red
0	10.1	1	0	1	0
1	13.5	2	0	0	1
2	15.3	3	1	0	0

[28]:
```python
pd.get_dummies(df[['price', 'color', 'size']],drop_first=True)
```

[28]:

	price	size	color_green	color_red
0	10.1	1	1	0
1	13.5	2	0	1
2	15.3	3	0	0

[29]:
```python
ohe = OneHotEncoder(categorical_features=[0])
ohe.fit_transform(X).toarray()[:, 1:]
```

```
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\preprocessing\_encoders.py:371:
FutureWarning: The handling of integer data will change in version 0.22.
Currently, the categories are determined based on the range [0, max(values)],
while in the future they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify
"categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the
```

```
categories to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\preprocessing\_encoders.py:392:
DeprecationWarning: The 'categorical_features' keyword is deprecated in version
0.20 and will be removed in 0.22. You can use the ColumnTransformer instead.
  "use the ColumnTransformer instead.", DeprecationWarning)
```

[29]:
```
array([[ 1. ,  0. ,  1. , 10.1],
       [ 0. ,  1. ,  2. , 13.5],
       [ 0. ,  0. ,  3. , 15.3]])
```

0.7 Partitioning a dataset into separate training and test sets

[30]:
```python
df_wine = pd.read_csv('https://archive.ics.uci.edu/'
'ml/machine-learning-databases/'
'wine/wine.data', header=None)
df_wine.columns = ['Class label', 'Alcohol','Malic acid', 'Ash','Alcalinity of␣
 ↪ash', 'Magnesium','Total phenols', 'Flavanoids','Nonflavanoid␣
 ↪phenols','Proanthocyanins','Color intensity', 'Hue','OD280/OD315 of diluted␣
 ↪wines','Proline']
print('Class labels', np.unique(df_wine['Class label']))
df_wine.head()
```

```
Class labels [1 2 3]
```

[30]:

	Class label	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	\
0	1	14.23	1.71	2.43	15.6	127	
1	1	13.20	1.78	2.14	11.2	100	
2	1	13.16	2.36	2.67	18.6	101	
3	1	14.37	1.95	2.50	16.8	113	
4	1	13.24	2.59	2.87	21.0	118	

	Total phenols	Flavanoids	Nonflavanoid phenols	Proanthocyanins	\
0	2.80	3.06	0.28	2.29	
1	2.65	2.76	0.26	1.28	
2	2.80	3.24	0.30	2.81	
3	3.85	3.49	0.24	2.18	
4	2.80	2.69	0.39	1.82	

	Color intensity	Hue	OD280/OD315 of diluted wines	Proline
0	5.64	1.04	3.92	1065
1	4.38	1.05	3.40	1050
2	5.68	1.03	3.17	1185
3	7.80	0.86	3.45	1480
4	4.32	1.04	2.93	735

[31]:
```
from sklearn.model_selection import train_test_split
X, y = df_wine.iloc[:, 1:].values, df_wine.iloc[:, 0].values
X_train, X_test, y_train, y_test =\
train_test_split(X, y,test_size=0.3,random_state=0,stratify=y)
```

0.8 Bringing features onto the same scale

[32]:
```
## Normalization (special case of min-max scaling)

from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler()
X_train_norm = mms.fit_transform(X_train)
X_test_norm = mms.transform(X_test)
```

[33]:
```
ex = np.array([0, 1, 2, 3, 4, 5])
print('standardized:', (ex - ex.mean()) / ex.std())
print('normalized:', (ex - ex.min()) / (ex.max() - ex.min()))
```

```
standardized: [-1.46385011 -0.87831007 -0.29277002  0.29277002  0.87831007
1.46385011]
normalized: [0.  0.2 0.4 0.6 0.8 1. ]
```

[34]:
```
from sklearn.preprocessing import StandardScaler
stdsc = StandardScaler()
X_train_std = stdsc.fit_transform(X_train)
X_test_std = stdsc.transform(X_test)
```

0.9 Selecting meaningful features

[35]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(penalty='l1', C=1.0)
lr.fit(X_train_std, y_train)
print('Training accuracy:', lr.score(X_train_std, y_train))
print('Test accuracy:', lr.score(X_test_std, y_test))
```

```
Training accuracy: 1.0
Test accuracy: 1.0
```

```
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

[36]:
```
lr.intercept_
```

[36]:
```
array([-1.26343028, -1.21590709, -2.37021461])
```

[37]:
```
lr.coef_

## sparse solution by l1 regularization
```

[37]:
```
array([[ 1.24599199,  0.18071499,  0.74188587, -1.15855841,  0.        ,
         0.        ,  1.16921084,  0.        ,  0.        ,  0.        ,
         0.        ,  0.5476912 ,  2.51040758],
       [-1.53742611, -0.38728583, -0.99524683,  0.3651676 , -0.05952479,
         0.        ,  0.66779033,  0.        ,  0.        , -1.93408289,
         1.23373083,  0.        , -2.23140403],
       [ 0.135754  ,  0.16833636,  0.35712506,  0.        ,  0.        ,
         0.        , -2.43837054,  0.        ,  0.        ,  1.56393217,
        -0.81893443, -0.49226063,  0.        ]])
```

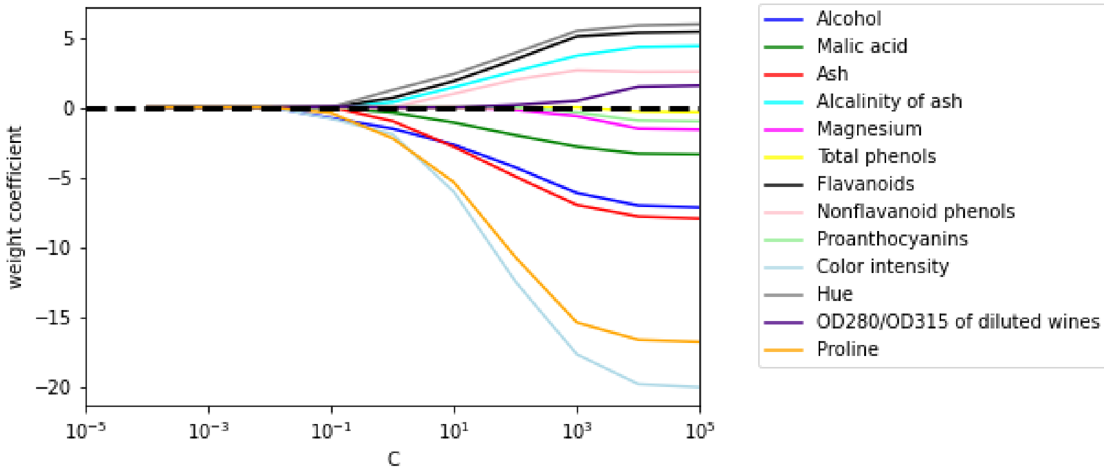
[38]:
```
import matplotlib.pyplot as plt
fig = plt.figure()
ax = plt.subplot(111)
colors = ['blue', 'green', 'red', 'cyan','magenta', 'yellow', 'black','pink',␣
 ↪'lightgreen', 'lightblue','gray', 'indigo', 'orange']
weights, params = [], []
for c in np.arange(-4., 6.):
    lr = LogisticRegression(penalty='l1',C=10.**c,random_state=0)
    lr.fit(X_train_std, y_train)
    weights.append(lr.coef_[1])
    params.append(10**c)
weights = np.array(weights)
for column, color in zip(range(weights.shape[1]), colors):
    plt.plot(params, weights[:, column],label=df_wine.columns[column +␣
 ↪1],color=color)

plt.axhline(0, color='black', linestyle='--', linewidth=3)
plt.xlim([10**(-5), 10**5])
plt.ylabel('weight coefficient')
plt.xlabel('C')
plt.xscale('log')
plt.legend(loc='upper left')
ax.legend(loc='upper center',bbox_to_anchor=(1.38, 1.03),ncol=1, fancybox=True)
plt.show()
```

```
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
```

```
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
```

```
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:433:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
C:\Users\ankit19.gupta\Desktop\Self_Projects\Python_Machine_Learning_Sebastian_R
aschka\venv_python_3.6\lib\site-packages\sklearn\linear_model\logistic.py:460:
FutureWarning: Default multi_class will be changed to 'auto' in 0.22. Specify
the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```



[39]:
```
from sklearn.base import clone
from itertools import combinations
import numpy as np
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
```

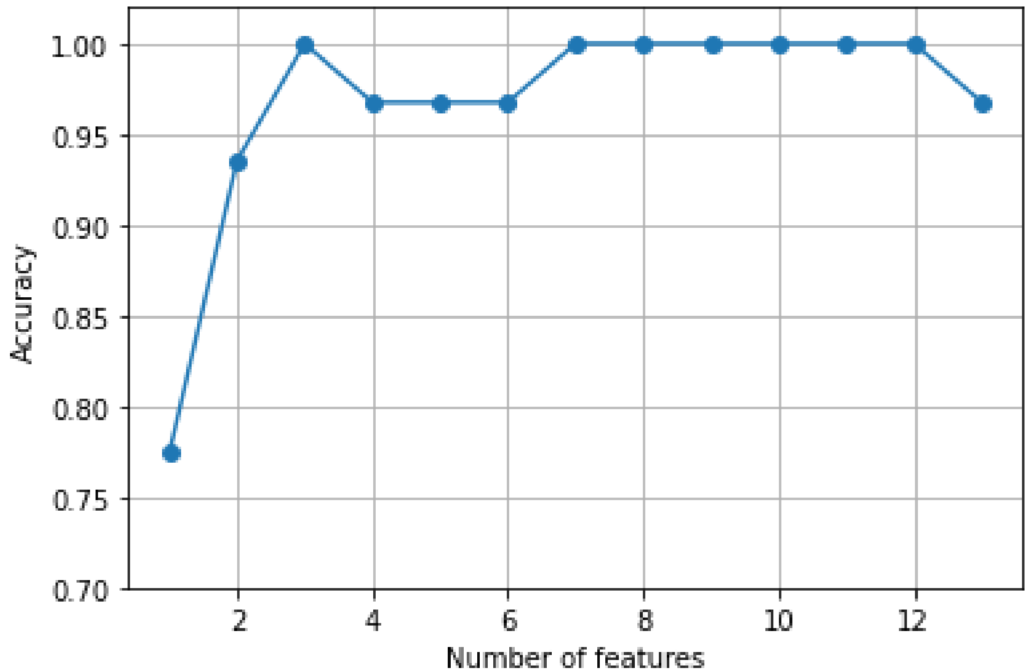
[41]:
```
class SBS():
    def __init__(self, estimator, k_features,scoring=accuracy_score,test_size=0.
↪25, random_state=1):
        self.scoring = scoring
        self.estimator = clone(estimator)
        self.k_features = k_features
        self.test_size = test_size
        self.random_state = random_state
    def fit(self, X, y):
        X_train, X_test, y_train, y_test = \
        train_test_split(X, y, test_size=self.test_size,
        random_state=self.random_state)
        dim = X_train.shape[1]
        self.indices_ = tuple(range(dim))
        self.subsets_ = [self.indices_]
        score = self._calc_score(X_train, y_train,X_test, y_test, self.indices_)
        self.scores_ = [score]
        while dim > self.k_features:
            scores = []
            subsets = []
            for p in combinations(self.indices_, r=dim - 1):
                score = self._calc_score(X_train, y_train,X_test, y_test, p)
                scores.append(score)
                subsets.append(p)
```

```
            best = np.argmax(scores)
            self.indices_ = subsets[best]
            self.subsets_.append(self.indices_)
            dim -= 1
            self.scores_.append(scores[best])
        self.k_score_ = self.scores_[-1]
        return self
    def transform(self, X):
        return X[:, self.indices_]
    def _calc_score(self, X_train, y_train, X_test, y_test,indices):
        self.estimator.fit(X_train[:, indices], y_train)
        y_pred = self.estimator.predict(X_test[:, indices])
        score = self.scoring(y_test, y_pred)
        return score
```

[42]:
```
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
sbs = SBS(knn, k_features=1)
sbs.fit(X_train_std, y_train)
```

[42]: <__main__.SBS at 0x1a17fee5e10>

[43]:
```
k_feat = [len(k) for k in sbs.subsets_]
plt.plot(k_feat, sbs.scores_, marker='o')
plt.ylim([0.7, 1.02])
plt.ylabel('Accuracy')
plt.xlabel('Number of features')
plt.grid()
plt.show()
```



[44]:
```
k3 = list(sbs.subsets_[10])
print(df_wine.columns[1:][k3])
```

```
Index(['Alcohol', 'Malic acid', 'OD280/OD315 of diluted wines'], dtype='object')
```

[45]:
```
knn.fit(X_train_std, y_train)
print('Training accuracy:', knn.score(X_train_std, y_train))
print('Test accuracy:', knn.score(X_test_std, y_test))
```

```
Training accuracy: 0.967741935483871
Test accuracy: 0.9629629629629629
```

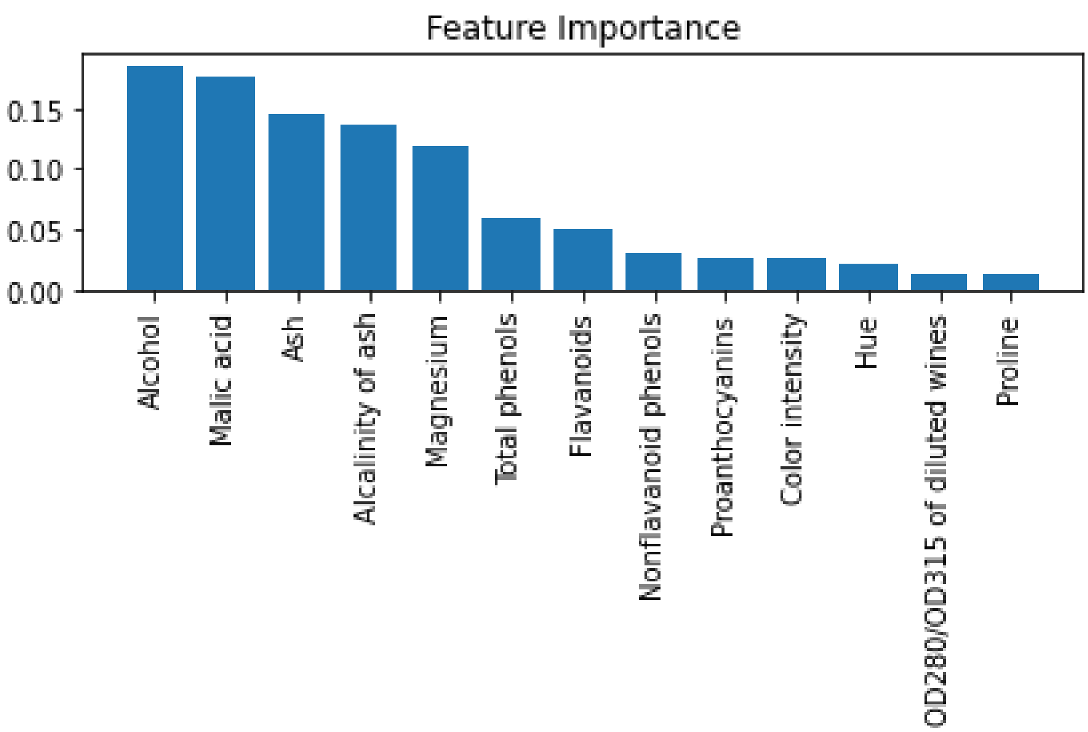
[46]:
```
knn.fit(X_train_std[:, k3], y_train)
print('Training accuracy:',knn.score(X_train_std[:, k3], y_train))
print('Test accuracy:',knn.score(X_test_std[:, k3], y_test))
```

```
Training accuracy: 0.9516129032258065
Test accuracy: 0.9259259259259259
```

[47]:
```
from sklearn.ensemble import RandomForestClassifier
feat_labels = df_wine.columns[1:]
forest = RandomForestClassifier(n_estimators=500,random_state=1)
forest.fit(X_train, y_train)
importances = forest.feature_importances_
indices = np.argsort(importances)[::-1]
for f in range(X_train.shape[1]):
```

```
    print("%2d) %-*s %f" % (f + 1,␣
↪30,feat_labels[indices[f]],importances[indices[f]]))
plt.title('Feature Importance')
plt.bar(range(X_train.shape[1]),importances[indices],align='center')
plt.xticks(range(X_train.shape[1]),feat_labels, rotation=90)
plt.xlim([-1, X_train.shape[1]])
plt.tight_layout()
plt.show()
```

```
 1) Proline                        0.185453
 2) Flavanoids                     0.174751
 3) Color intensity                0.143920
 4) OD280/OD315 of diluted wines   0.136162
 5) Alcohol                        0.118529
 6) Hue                            0.058739
 7) Total phenols                  0.050872
 8) Magnesium                      0.031357
 9) Malic acid                     0.025648
10) Proanthocyanins                0.025570
11) Alcalinity of ash              0.022366
12) Nonflavanoid phenols           0.013354
13) Ash                            0.013279
```



[48]:

```
from sklearn.feature_selection import SelectFromModel
sfm = SelectFromModel(forest, threshold=0.1, prefit=True)
X_selected = sfm.transform(X_train)
print('Number of samples that meet this criterion:',X_selected.shape[0])
for f in range(X_selected.shape[1]):
    print("%2d) %-*s %f" % (f + 1,␣
↪30,feat_labels[indices[f]],importances[indices[f]]))
```

```
Number of samples that meet this criterion: 124
 1) Proline                        0.185453
 2) Flavanoids                     0.174751
 3) Color intensity                0.143920
 4) OD280/OD315 of diluted wines   0.136162
 5) Alcohol                        0.118529
```

[ ]: