```python
import numpy as np
import pandas as pd
import graphviz
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.model_selection import GridSearchCV
from IPython.core.display import HTML
from sklearn.tree import export_text
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set(style="ticks")
```

```python
from sklearn import datasets
wine = datasets.load_wine()
wine_df = pd.DataFrame(wine.data, columns=wine.feature_names)
wine_df.head()
```


```
{"summary":"{\n  \"name\": \"wine_df\",\n  \"rows\": 178,\n
\"fields\": [\n    {\n      \"column\": \"alcohol\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.8118265380058575,\n        \"min\": 11.03,\n        \"max\": 14.83,\
n        \"num_unique_values\": 126,\n        \"samples\": [\n
11.62,\n          13.64,\n          13.69\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"malic_acid\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
1.1171460976144627,\n        \"min\": 0.74,\n        \"max\": 5.8,\n
\"num_unique_values\": 133,\n        \"samples\": [\n          1.21,\n
2.83,\n          1.8\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"ash\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0.27434400906081485,\n        \"min\": 1.36,\n
\"max\": 3.23,\n        \"num_unique_values\": 79,\n
\"samples\": [\n          2.31,\n          2.43,\n          2.52\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"alcalinity_of_ash\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
3.339563767173505,\n        \"min\": 10.6,\n        \"max\": 30.0,\n
\"num_unique_values\": 63,\n        \"samples\": [\n          25.5,\n
28.5,\n          15.6\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"magnesium\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 14.282483515295665,\n        \"min\":
```

70.0,\n   \"max\": 162.0,\n   \"num_unique_values\": 53,\n
\"samples\": [\n   126.0,\n   85.0,\n   162.0\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"total_phenols\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.6258510488339893,\n   \"min\": 0.98,\n   \"max\": 3.88,\n
\"num_unique_values\": 97,\n   \"samples\": [\n   1.68,\n
2.11,\n   1.35\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"flavanoids\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.9988586850169467,\n   \"min\":
0.34,\n   \"max\": 5.08,\n   \"num_unique_values\": 132,\n
\"samples\": [\n   3.18,\n   2.5,\n   3.17\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"nonflavanoid_phenols\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
0.12445334029667937,\n   \"min\": 0.13,\n   \"max\": 0.66,\n
\"num_unique_values\": 39,\n   \"samples\": [\n   0.58,\n
0.41,\n   0.39\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"proanthocyanins\",\n   \"properties\": {\n   \"dtype\":
\"number\",\n   \"std\": 0.5723588626747613,\n   \"min\":
0.41,\n   \"max\": 3.58,\n   \"num_unique_values\": 101,\n
\"samples\": [\n   0.75,\n   1.77,\n   1.42\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"color_intensity\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
2.318285871822413,\n   \"min\": 1.28,\n   \"max\": 13.0,\n
\"num_unique_values\": 132,\n   \"samples\": [\n   2.95,\n
3.3,\n   5.1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"hue\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.22857156582982338,\n   \"min\": 0.48,\n
\"max\": 1.71,\n   \"num_unique_values\": 78,\n
\"samples\": [\n   1.22,\n   1.04,\n   1.45\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"od280/od315_of_diluted_wines\",\
n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0.7099904287650504,\n   \"min\": 1.27,\n   \"max\":
4.0,\n   \"num_unique_values\": 122,\n   \"samples\": [\n
4.0,\n   1.82,\n   1.59\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"proline\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\":
314.9074742768491,\n   \"min\": 278.0,\n   \"max\": 1680.0,\
n   \"num_unique_values\": 121,\n   \"samples\": [\n
1375.0,\n   1270.0,\n   735.0\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"wine_df"}

```
wine_df.isna().sum()
```

```
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
dtype: int64
```

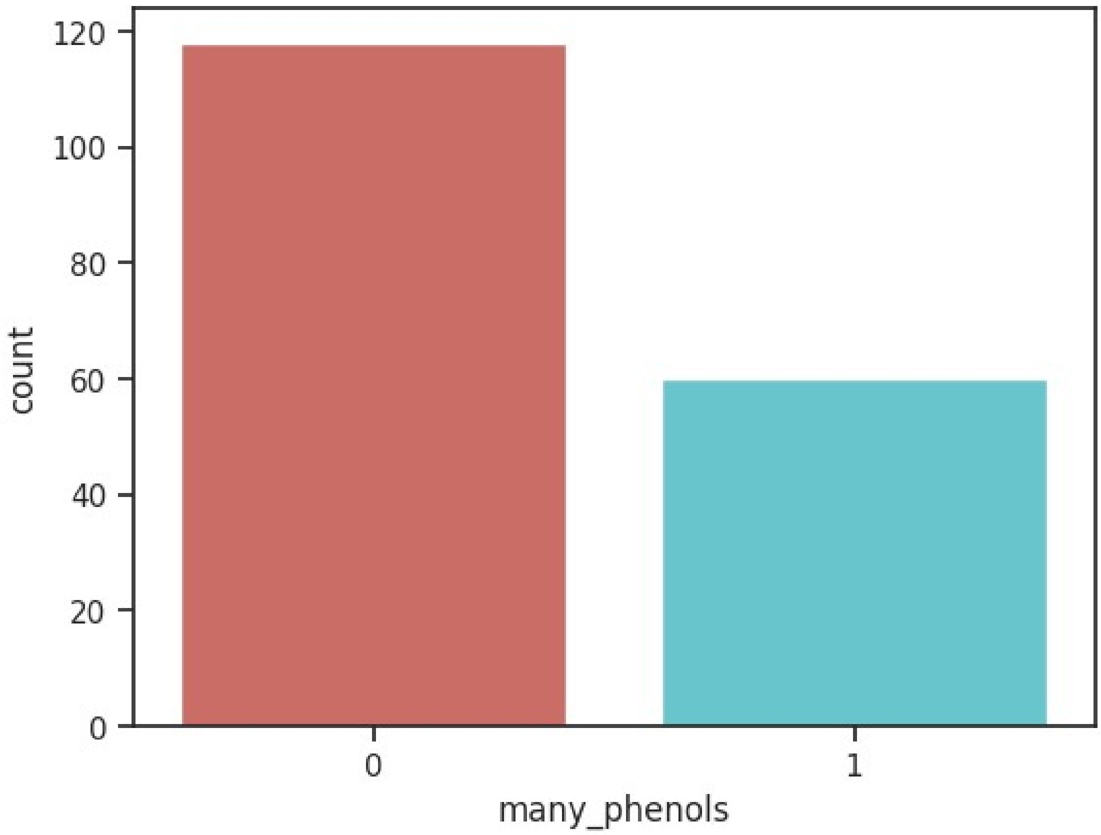
```
wine_df['many_phenols'] = np.where(wine_df['total_phenols'] >
wine_df['total_phenols'].quantile(0.65), 1, 0)
wine_df.drop(['total_phenols'], axis=1, inplace=True)

sns.countplot(x='many_phenols', data=wine_df, palette='hls')
plt.show()
```

```
<ipython-input-14-42956e5e8f46>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `x` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(x='many_phenols', data=wine_df, palette='hls')
```




```
last_column = wine_df.pop('many_phenols')
wine_df.insert(len(wine_df.columns), 'many_phenols', last_column)
X = wine_df.iloc[:, :-1].values
y = wine_df.iloc[:, -1].values

# Формирование обучающей и тестовой выборки
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 1)

logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred_test_logreg = logreg.predict(X_test)
y_pred_train_logreg = logreg.predict(X_train)
ac1 = accuracy_score(y_train, y_pred_train_logreg),
accuracy_score(y_test, y_pred_test_logreg)
ac1
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/
_logistic.py:458: ConvergenceWarning: lbfgs failed to converge
(status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
```

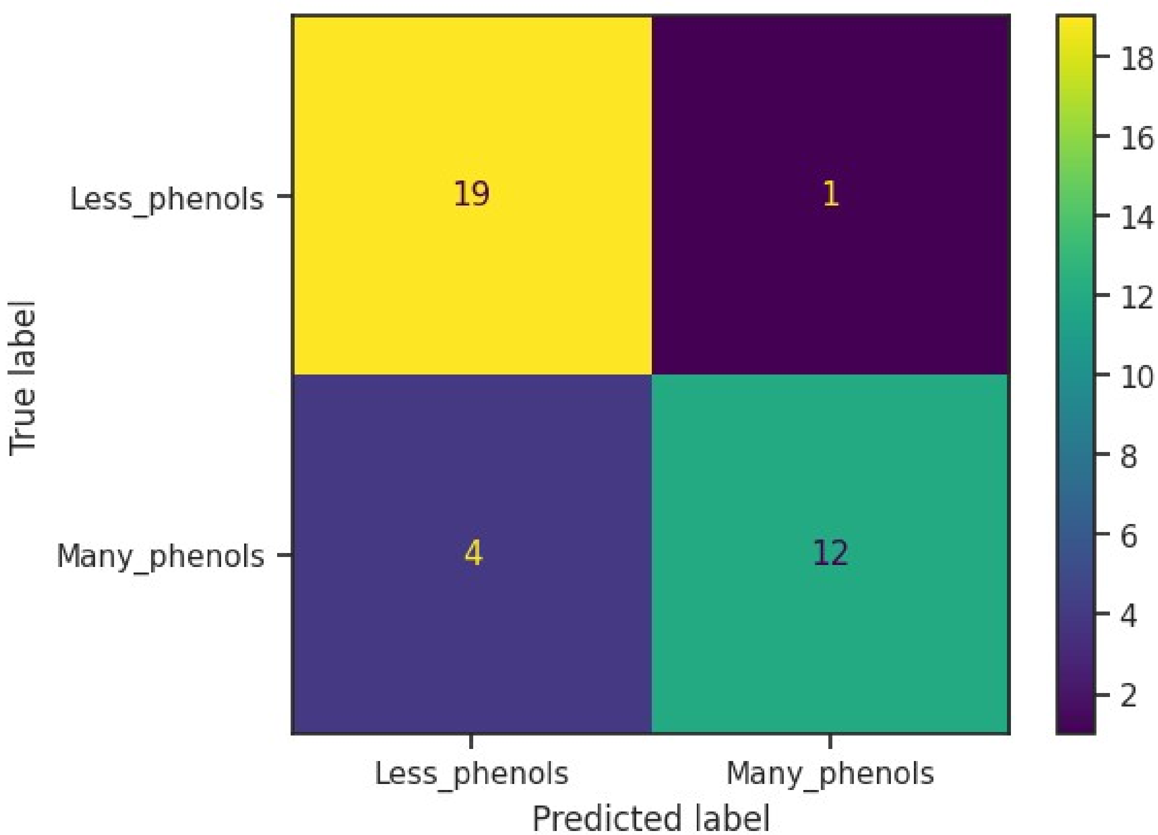
```
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:

https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

```
(0.9084507042253521, 0.8611111111111112)
```

```
cm1 = confusion_matrix(y_test, y_pred_test_logreg, labels =
logreg.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm1,
display_labels=['Less_phenols', 'Many_phenols'])
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x79bfef0a1ea0>
```




```
svc = SVC(kernel='poly')
svc.fit(X_train, y_train)
y_pred_test_svc = svc.predict(X_test)
y_pred_train_svc = svc.predict(X_train)
ac2 = accuracy_score(y_train, y_pred_train_svc),
```

```
accuracy_score(y_test, y_pred_test_svc)
ac2
```

```
(0.7887323943661971, 0.6666666666666666)
```

```
param_grid = {'degree': [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15],
'kernel':['poly']}
grid = GridSearchCV(SVC(), param_grid, verbose=2, scoring='accuracy')
grid.fit(X_train, y_train)
grid.best_params_
```

```
Fitting 5 folds for each of 15 candidates, totalling 75 fits
[CV] END ..............................degree=1, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=1, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=1, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=1, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=1, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=2, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=3, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=4, kernel=poly; total
time=   0.0s
```

```
[CV] END ..............................degree=4, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=5, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=6, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=7, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=8, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=9, kernel=poly; total
```

```
time=   0.1s
[CV] END ...............................degree=9, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=10, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=10, kernel=poly; total
time=   0.0s
[CV] END ..............................degree=10, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=10, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=10, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=11, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=11, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=11, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=11, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=11, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=12, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=12, kernel=poly; total
time=   0.3s
[CV] END ..............................degree=12, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=12, kernel=poly; total
time=   0.2s
[CV] END ..............................degree=12, kernel=poly; total
time=   0.2s
[CV] END ..............................degree=13, kernel=poly; total
time=   0.3s
[CV] END ..............................degree=13, kernel=poly; total
time=   0.2s
[CV] END ..............................degree=13, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=13, kernel=poly; total
time=   0.1s
[CV] END ..............................degree=13, kernel=poly; total
time=   0.3s
[CV] END ..............................degree=14, kernel=poly; total
time=   0.4s
[CV] END ..............................degree=14, kernel=poly; total
time=   1.3s
[CV] END ..............................degree=14, kernel=poly; total
time=   0.5s
```

```
[CV] END .............................degree=14, kernel=poly; total
time=   0.3s
[CV] END .............................degree=14, kernel=poly; total
time=   0.3s
[CV] END .............................degree=15, kernel=poly; total
time=   1.5s
[CV] END .............................degree=15, kernel=poly; total
time=   3.3s
[CV] END .............................degree=15, kernel=poly; total
time=   1.8s
[CV] END .............................degree=15, kernel=poly; total
time=   0.3s
[CV] END .............................degree=15, kernel=poly; total
time=   1.2s
```

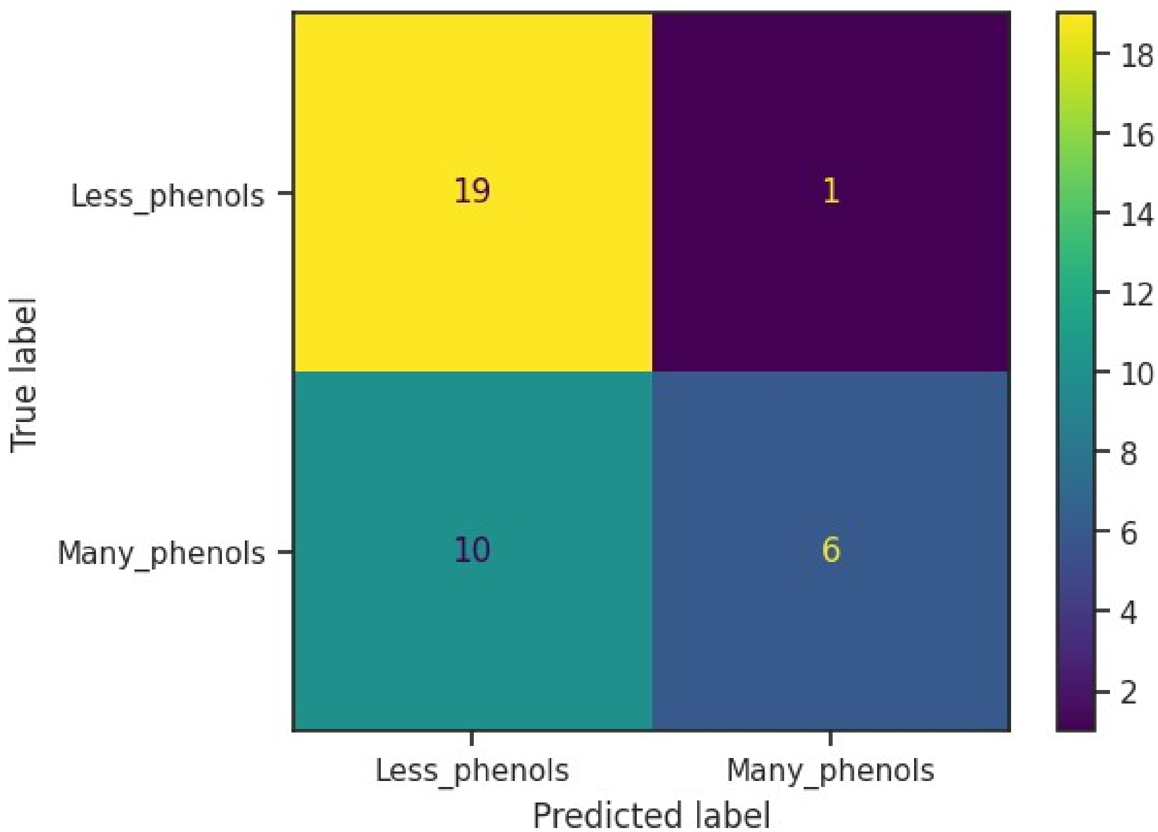
```
{'degree': 14, 'kernel': 'poly'}
```

```
svc = SVC(kernel='poly', degree=14)
svc.fit(X_train, y_train)
y_pred_test_svc = svc.predict(X_test)
y_pred_train_svc = svc.predict(X_train)
accuracy_score(y_train, y_pred_train_svc), accuracy_score(y_test,
y_pred_test_svc)
```

```
(0.8380281690140845, 0.6944444444444444)
```

```
cm2 = confusion_matrix(y_test, y_pred_test_svc, labels = svc.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm2,
display_labels=['Less_phenols', 'Many_phenols'])
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x79bfeee4f940>
```

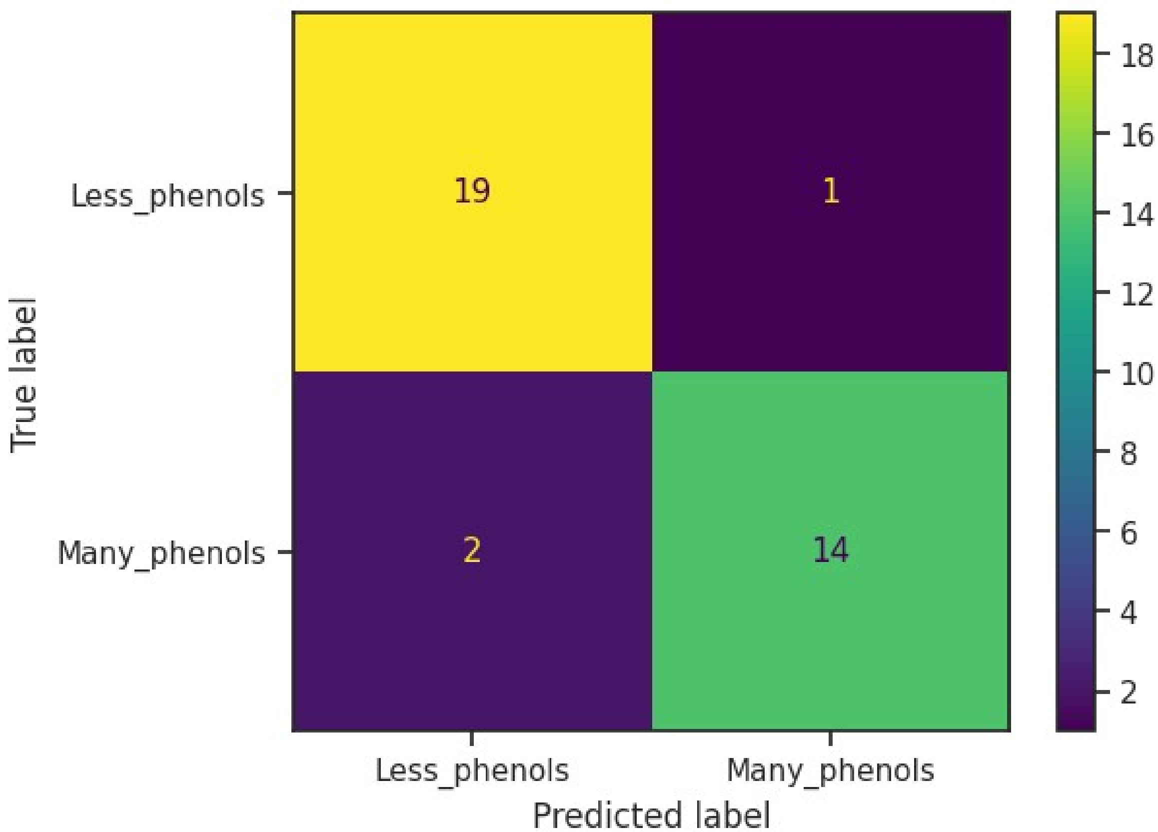



```
tree = DecisionTreeClassifier(random_state=1).fit(X_train, y_train)
y_pred_test_tree = tree.predict(X_test)
y_pred_train_tree = tree.predict(X_train)
ac3 = accuracy_score(y_train, y_pred_train_tree),
accuracy_score(y_test, y_pred_test_tree)
ac3
```

```
(1.0, 0.9166666666666666)
```

```
cm3 = confusion_matrix(y_test, y_pred_test_tree, labels =
tree.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm3,
display_labels=['Less_phenols', 'Many_phenols'])
disp.plot()
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x79bfef0d75b0>
```




```
# сравнение качества моделей по 2 метрикам
print('LogisticRegression:     ', ac1)
print('SVM:                    ', ac2)
print('DecisionTreeClassifier: ', ac3)
```

```
LogisticRegression:      (0.9084507042253521, 0.8611111111111112)
SVM:                     (0.7887323943661971, 0.6666666666666666)
DecisionTreeClassifier:  (1.0, 0.9166666666666666)
```

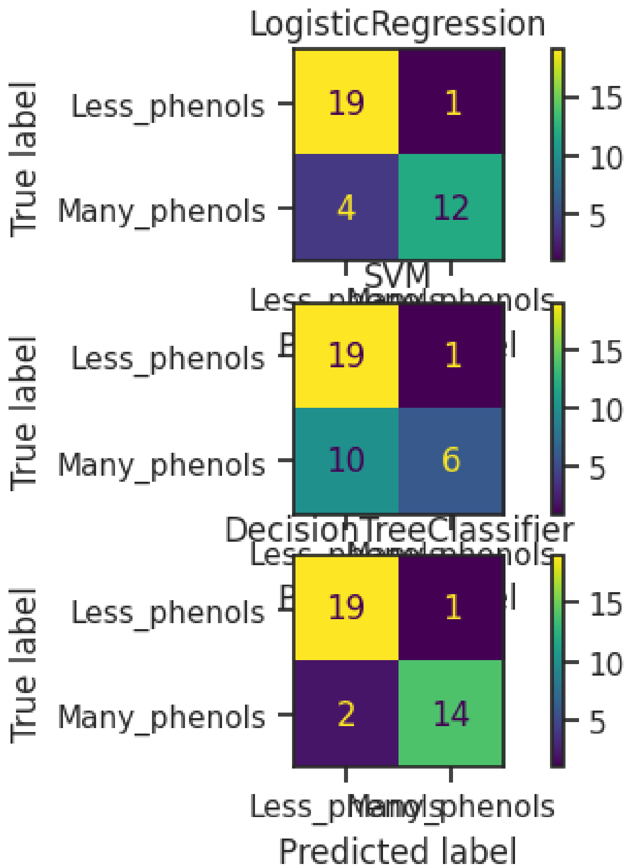
```
fig, ax = plt.subplots(3,1)
ax[0].set_title("LogisticRegression")
ax[1].set_title("SVM")
ax[2].set_title("DecisionTreeClassifier")

ConfusionMatrixDisplay(confusion_matrix=cm1,
display_labels=['Less_phenols', 'Many_phenols']).plot(ax=ax[0])

ConfusionMatrixDisplay(confusion_matrix=cm2,
display_labels=['Less_phenols', 'Many_phenols']).plot(ax=ax[1])

ConfusionMatrixDisplay(confusion_matrix=cm3,
display_labels=['Less_phenols', 'Many_phenols']).plot(ax=ax[2])
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x79bfeee10d30>
```




```
tree_rules = export_text(tree, feature_names=list(wine_df.iloc[:, :-
1].columns))
HTML('<pre>' + tree_rules + '</pre>')

<IPython.core.display.HTML object>

dot_data = export_graphviz(tree, out_file=None,
                           feature_names=wine_df.iloc[:, :-1].columns,
                           class_names=['Less_phenols',
'Many_phenols'],
                           filled=True, rounded=True,
special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

```
from operator import itemgetter

def draw_feature_importances(tree_model, X_dataset, figsize=(18,5)):
    """
    Вывод важности признаков в виде графика
    """
    # Сортировка значений важности признаков по убыванию
    list_to_sort = list(zip(X_dataset.columns.values, 
tree_model.feature_importances_))
    sorted_list = sorted(list_to_sort, key=itemgetter(1), reverse = 
True)
    # Названия признаков
    labels = [x for x,_ in sorted_list]
    # Важности признаков
    data = [x for _,x in sorted_list]
    # Вывод графика
    fig, ax = plt.subplots(figsize=figsize)
    ind = np.arange(len(labels))
    plt.bar(ind, data)
    plt.xticks(ind, labels, rotation='vertical')
    # Вывод значений
    for a,b in zip(ind, data):
        plt.text(a-0.05, b+0.01, str(round(b,3)))
    plt.show()
    return labels, data
```

```
fl, fd = draw_feature_importances(tree, wine_df.iloc[:, :-1])
```