Ensemble Techniques: Wine (Scikit-Learn)

1. Data Exploration

1.1 Load dataset

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
from sklearn.datasets import load_wine
wine = load_wine()
```

1.2 Exploratory data analysis

In [ ]:
```
wine.keys()
```

Out[ ]:
```
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names'])
```

In [ ]:
```
print(wine['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

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.

.. topic:: References

  (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).
```

In [ ]:
```
df = pd.DataFrame(data=wine['data'], columns=wine['feature_names'])
df['target'] = wine['target']
```

In [ ]:
```
df.head()
```

Out[ ]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	cc
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	

In [ ]:
```
df.info()
```

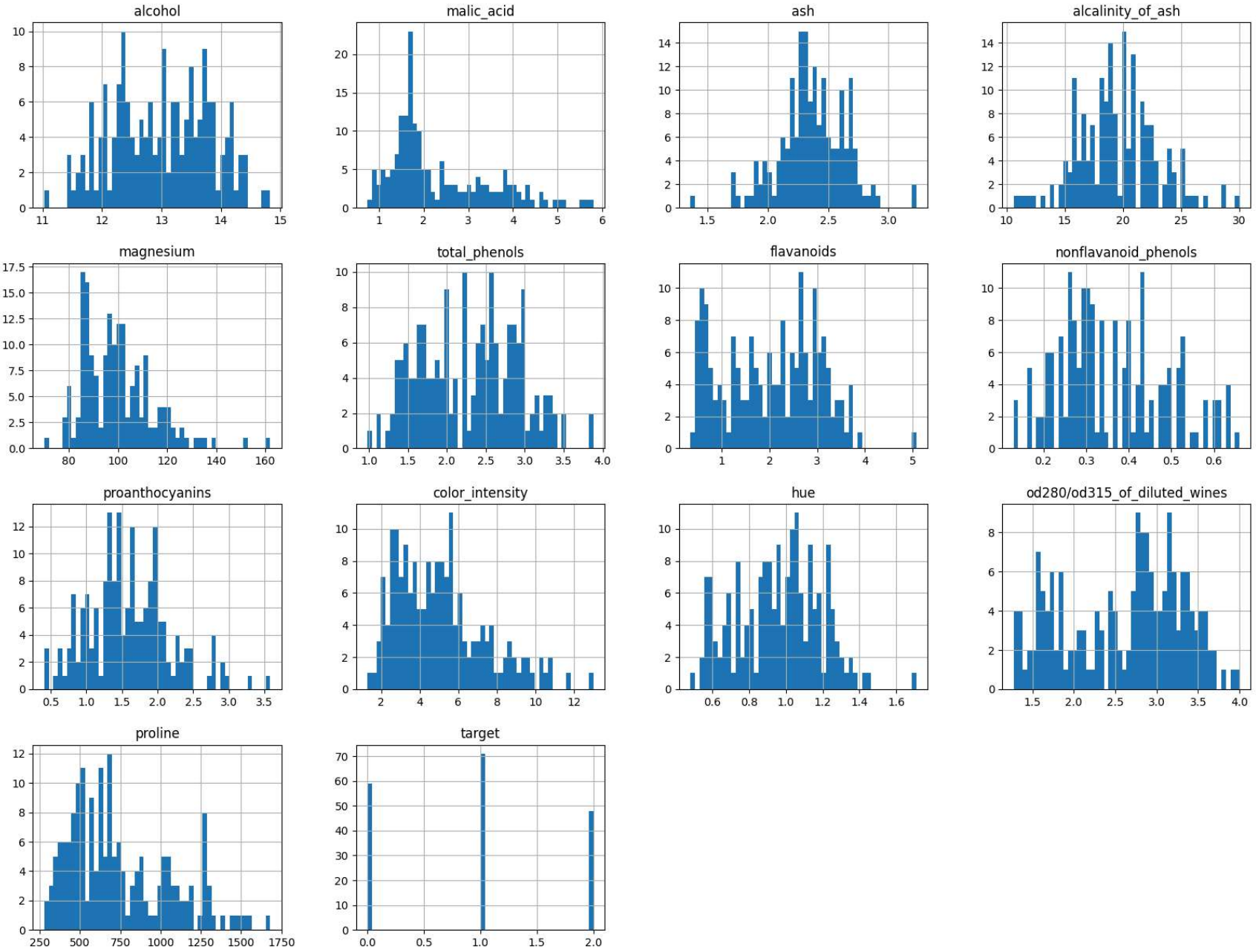
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    int32
dtypes: float64(13), int32(1)
memory usage: 18.9 KB
```

In [ ]:
```
df.describe()
```

Out[ ]:

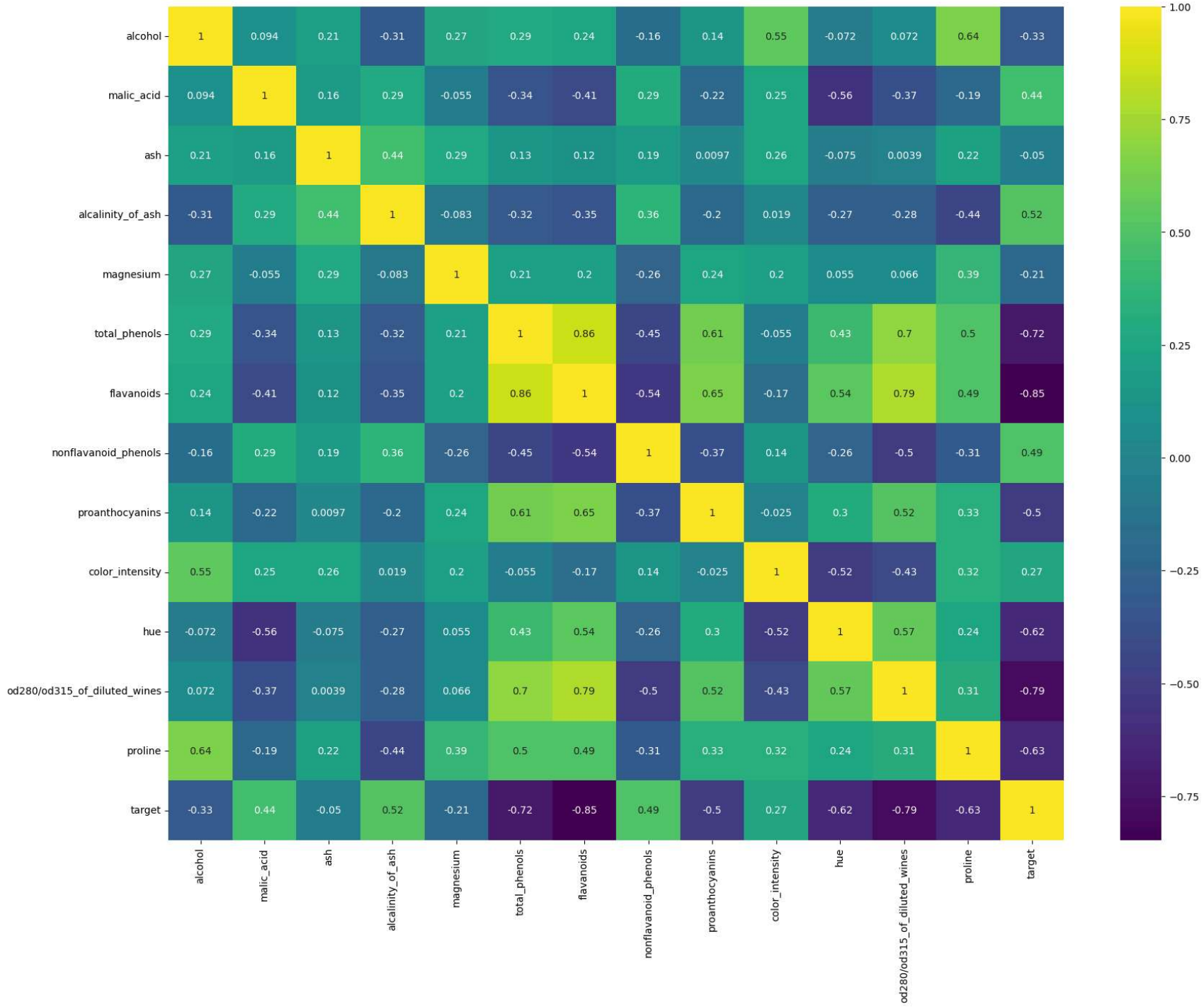
	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proa
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.029270	0.361854	
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.998859	0.124453	
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.340000	0.130000	
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.205000	0.270000	
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.135000	0.340000	
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.875000	0.437500	
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.080000	0.660000	

In [ ]:
```
df.hist(bins=50, figsize=(20,15))
plt.show()
```



1.3 Features correlation

In [ ]:
```
import seaborn as sns

corr = df.corr()
plt.figure(figsize=(20,15))
sns.heatmap(corr, annot=True, cmap='viridis')
plt.show()
```


alcohol 1 0.094 0.21 -0.31 0.27 0.29 0.24 -0.16 0.14 0.55 -0.072 0.072 0.64 -0.33
malic_acid 0.094 1 0.16 0.29 -0.055 -0.34 -0.41 0.29 -0.22 0.25 -0.56 -0.37 -0.19 0.44
ash 0.21 0.16 1 0.44 0.29 0.13 0.12 0.19 0.0097 0.26 -0.075 0.0039 0.22 -0.05
alcalinity_of_ash -0.31 0.29 0.44 1 -0.083 -0.32 -0.35 0.36 -0.2 0.019 -0.27 -0.28 -0.44 0.52
magnesium 0.27 -0.055 0.29 -0.083 1 0.21 0.2 -0.26 0.24 0.2 0.055 0.066 0.39 -0.21
total_phenols 0.29 -0.34 0.13 -0.32 0.21 1 0.86 -0.45 0.61 -0.055 0.43 0.7 0.5 -0.72
flavanoids 0.24 -0.41 0.12 -0.35 0.2 0.86 1 -0.54 0.65 -0.17 0.54 0.79 0.49 -0.85
nonflavanoid_phenols -0.16 0.29 0.19 0.36 -0.26 -0.45 -0.54 1 -0.37 0.14 -0.26 -0.5 -0.31 0.49
proanthocyanins 0.14 -0.22 0.0097 -0.2 0.24 0.61 0.65 -0.37 1 -0.025 0.3 0.52 0.33 -0.5
color_intensity 0.55 0.25 0.26 0.019 0.2 -0.055 -0.17 0.14 -0.025 1 -0.52 -0.43 0.32 0.27
hue -0.072 -0.56 -0.075 -0.27 0.055 0.43 0.54 -0.26 0.3 -0.52 1 0.57 0.24 -0.62
od280/od315_of_diluted_wines 0.072 -0.37 0.0039 -0.28 0.066 0.7 0.79 -0.5 0.52 -0.43 0.57 1 0.31 -0.79
proline 0.64 -0.19 0.22 -0.44 0.39 0.5 0.49 -0.31 0.33 0.32 0.24 0.31 1 -0.63
target -0.33 0.44 -0.05 0.52 -0.21 -0.72 -0.85 0.49 -0.5 0.27 -0.62 -0.79 -0.63 1
alcohol
malic_acid
ash
alcalinity_of_ash
magnesium
total_phenols
flavanoids
nonflavanoid_phenols
proanthocyanins
color_intensity
hue
od280/od315_of_diluted_wines
proline
target
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

2. Data Preprocessing

2.1 Split features and target

In [ ]:
```
X = df.drop('target', axis=1)
y = df['target']
X.shape, y.shape
```

Out[ ]:
```
((178, 13), (178,))
```

2.2 Split training and test datasets

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[ ]:
```
((142, 13), (36, 13), (142,), (36,))
```

2.3 Build pipeline

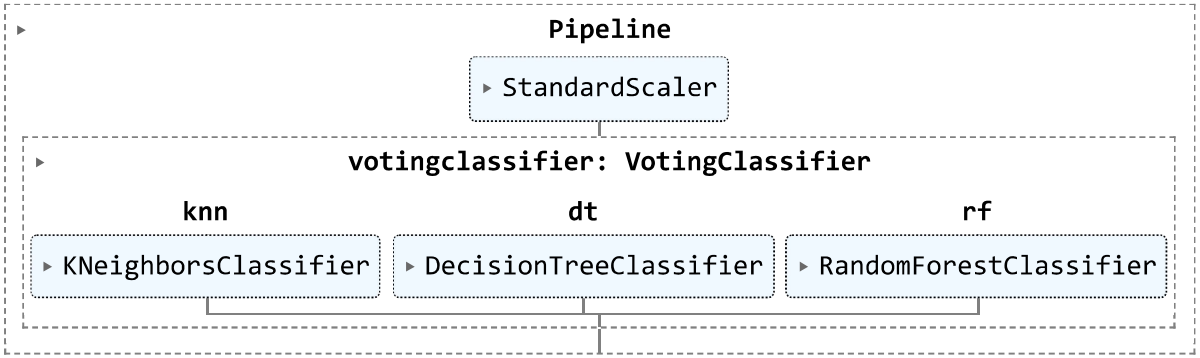
In [ ]:
```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

def std_pipeline():
    return StandardScaler()

preprocessing = std_pipeline()
```

3. Model Training

3.1 Voting Classifier

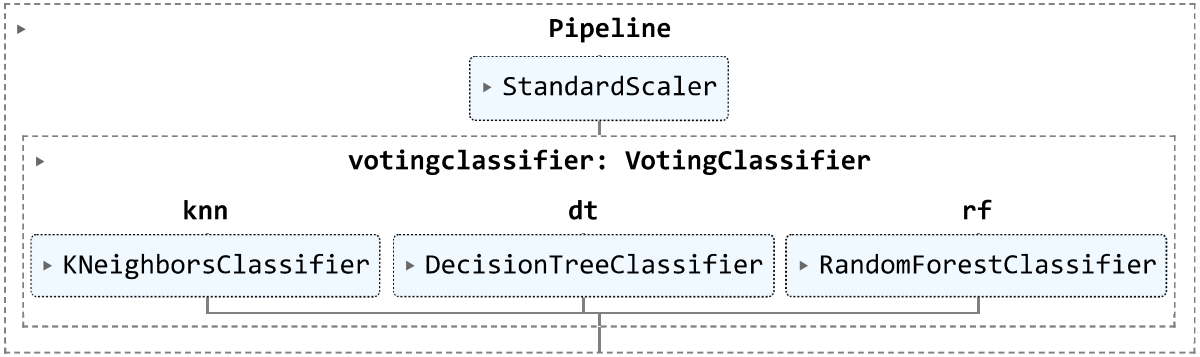
In [ ]:
```
from sklearn.ensemble import VotingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier

knn = KNeighborsClassifier()
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()

hard_voting = make_pipeline(preprocessing, VotingClassifier(
    estimators=[('knn', knn), ('dt', dt), ('rf', rf)],
    voting='hard'))

hard_voting.fit(X_train, y_train)
```

Out[ ]:



In [ ]:
```
soft_voting = make_pipeline(preprocessing, VotingClassifier(
    estimators=[('knn', knn), ('dt', dt), ('rf', rf)],
    voting='soft'))

soft_voting.fit(X_train, y_train)
```

Out[ ]:

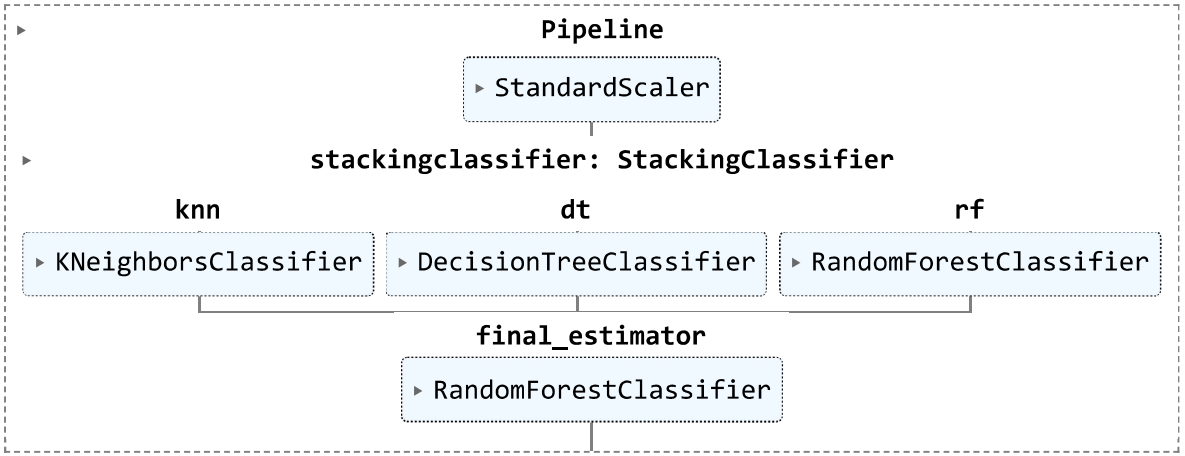


3.2 Stacking Classifier

In [ ]:

```
from sklearn.ensemble import StackingClassifier

stacking = make_pipeline(preprocessing, StackingClassifier(
    estimators=[('knn', knn), ('dt', dt), ('rf', rf)],
    final_estimator=rf,
    cv=5))

stacking.fit(X_train, y_train)
```

Out[ ]:

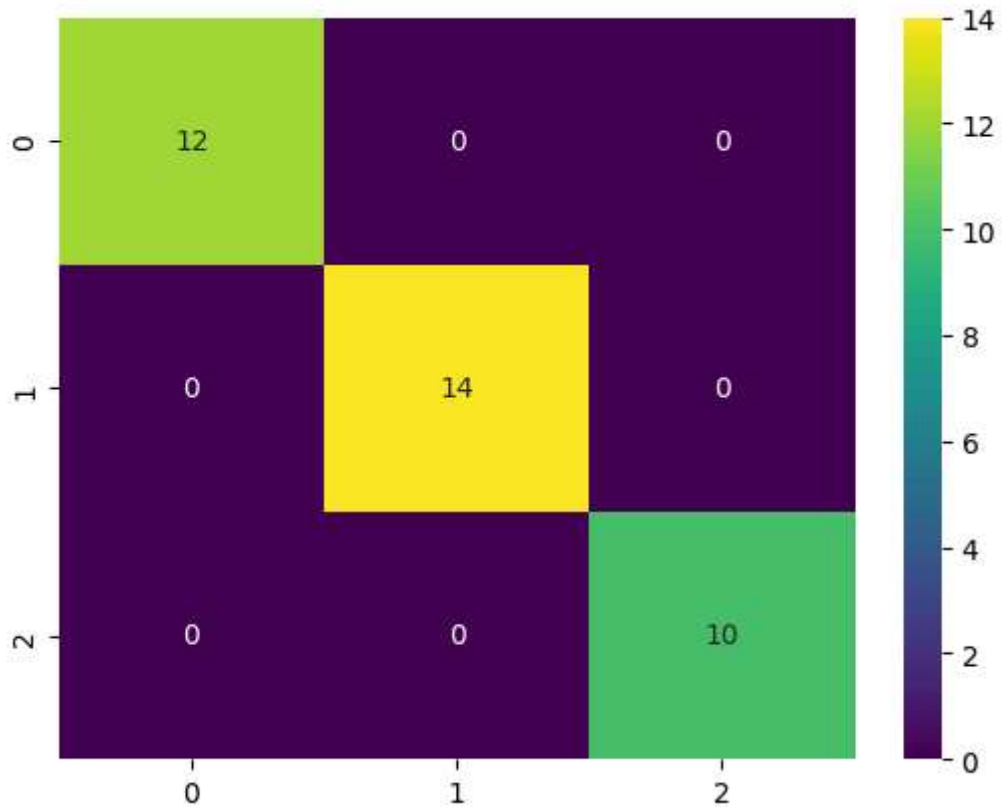


4. Model Evaluation

4.1 Voting Classifier

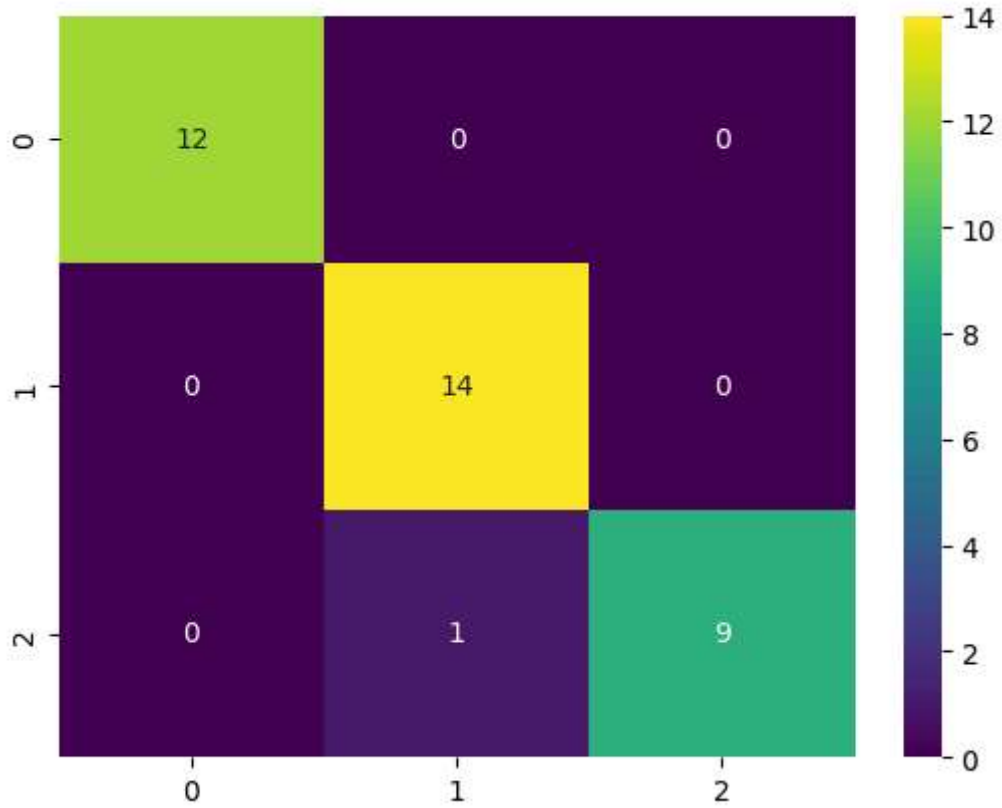
In [ ]:
```
from sklearn.metrics import classification_report, confusion_matrix

def evaluate_model(model):
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, cmap='viridis', fmt='d')

evaluate_model(hard_voting)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00        10

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```



In [ ]:
```
evaluate_model(soft_voting)
```

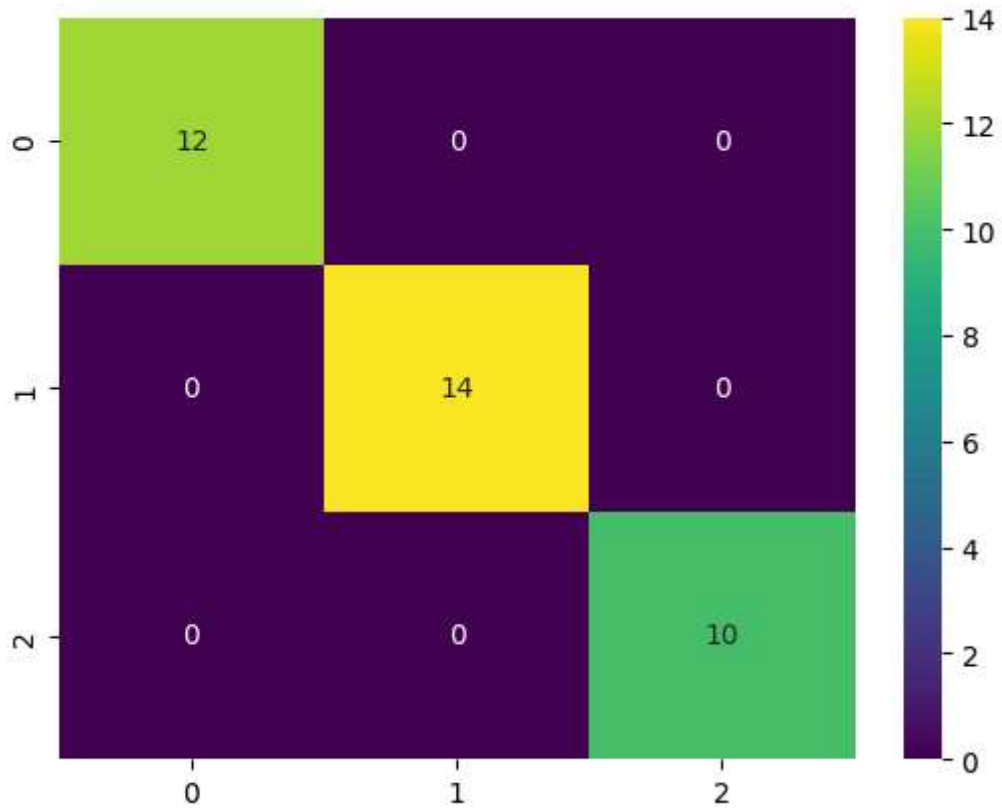
	precision	recall	f1-score	support
0	1.00	1.00	1.00	12
1	0.93	1.00	0.97	14
2	1.00	0.90	0.95	10
accuracy			0.97	36
macro avg	0.98	0.97	0.97	36
weighted avg	0.97	0.97	0.97	36



Stacking Classifier

In [ ]:
```
evaluate_model(stacking)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        12
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00        10

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```



1. Voting Classifier:

- Both hard voting and soft voting achieved perfect precision, recall, and F1-score for all classes, resulting in 100% accuracy.
- It means that all three individual classifiers unanimously agreed on the correct class for each sample.

2. Stacking Classifier:

- Also achieved high accuracy (97%).