CODE:

In [1]:
```
from sklearn.datasets import load_wine
from sklearn import datasets
from sklearn.tree import DecisionTreeClassifier
import pandas as pd
wine = load_wine()
```

In [2]:
```
def sklearn_to_df(sklearn_dataset):
    df = pd.DataFrame(sklearn_dataset.data, columns=sklearn_dataset.feature_names)
    df['target'] = pd.Series(sklearn_dataset.target)
    return df
```

In [3]:
```
df = sklearn_to_df(datasets.load_wine())
df.head()
```

Out[3]:

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	

In [4]:
```
wine.target[[10, 80, 140]]
list(wine.target_names)
```

Out[4]:
```
['class_0', 'class_1', 'class_2']
```

In [5]:
```
X, y = wine.data[:, 11:13], wine.target
clf = DecisionTreeClassifier()
clf.fit(X,y)
clf.score(X,y)
```

Out[5]:
```
1.0
```

In [6]:
```
from sklearn.model_selection import train_test_split as tts
X_train,X_test,y_train,y_test = tts(X,y,test_size=0.2)
```

In [7]:
```
clf.score(X_test, y_test)
```

Out[7]:
```
1.0
```

In [8]:
```
from sklearn.metrics import accuracy_score
preds = clf.predict(X_test)
accuracy_score(y_test, preds)
```

Out[8]:
```
1.0
```

In [9]:
```
from sklearn.ensemble import BaggingClassifier
bg = BaggingClassifier(base_estimator = clf , max_samples=0.1 ,
                       max_features = 0.1, n_estimators= 10)
bg.fit(X,y)
```

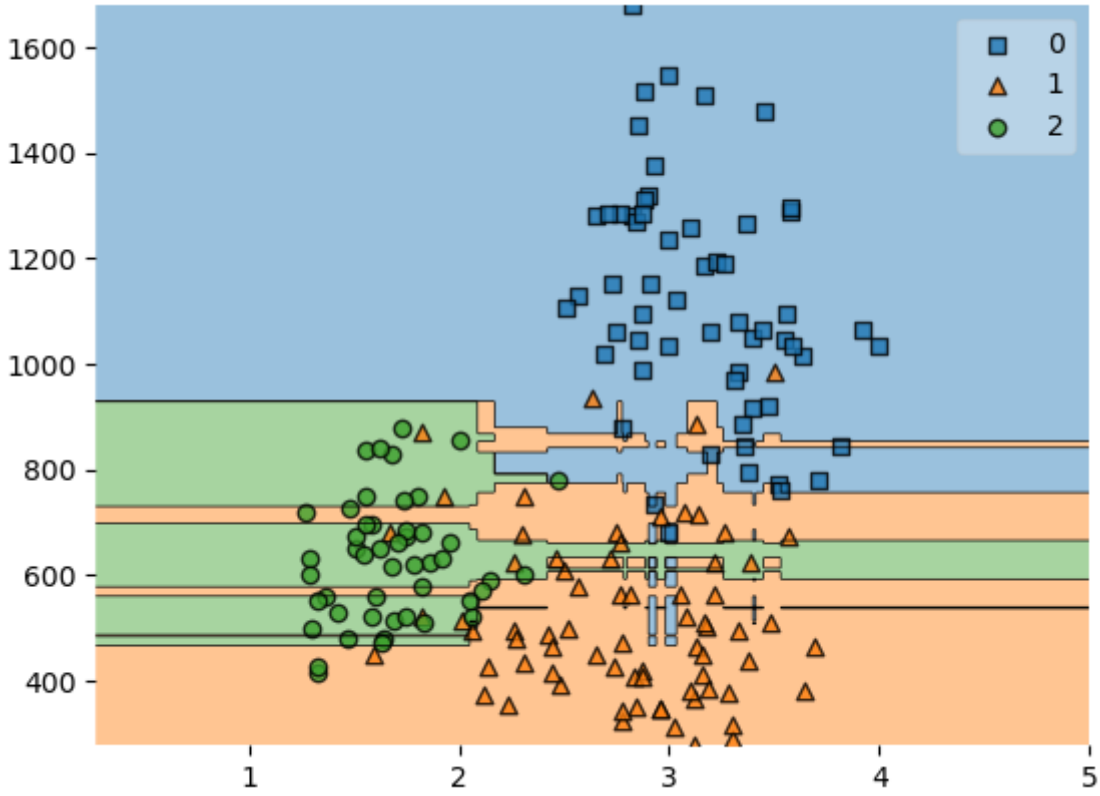
Out[9]:
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(), max_features=0.1,
                  max_samples=0.1)
```

In [10]:
```
bg.score(X,y)
```

Out[10]:
```
0.8764044943820225
```

In [11]:
```
from mlxtend.plotting import  plot_decision_regions
plot_decision_regions(X,y,bg)
```

Out[11]:
```
<AxesSubplot:>
```



In [12]:
```
from sklearn.ensemble import  RandomForestClassifier
rf = RandomForestClassifier(n_estimators=10)
rf.fit(X,y)
```

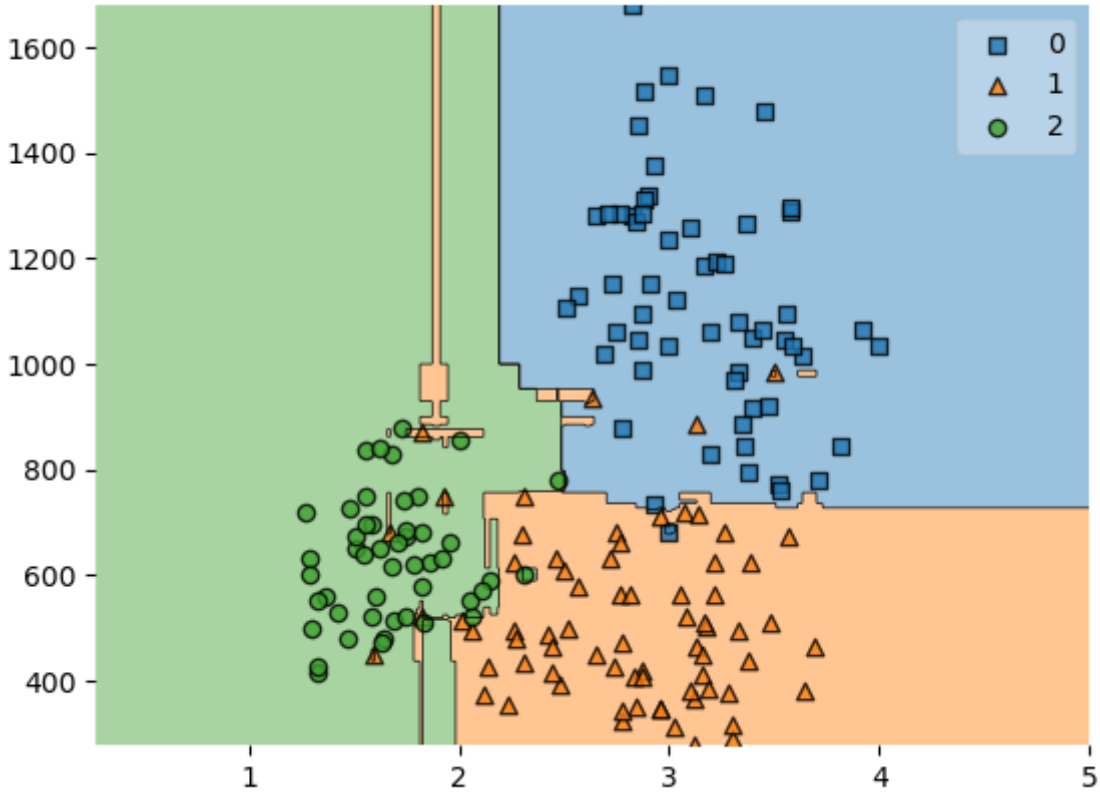
Out[12]:
```
RandomForestClassifier(n_estimators=10)
```

In [13]:
```
rf.score(X,y)
```

Out[13]:
```
0.9943820224719101
```

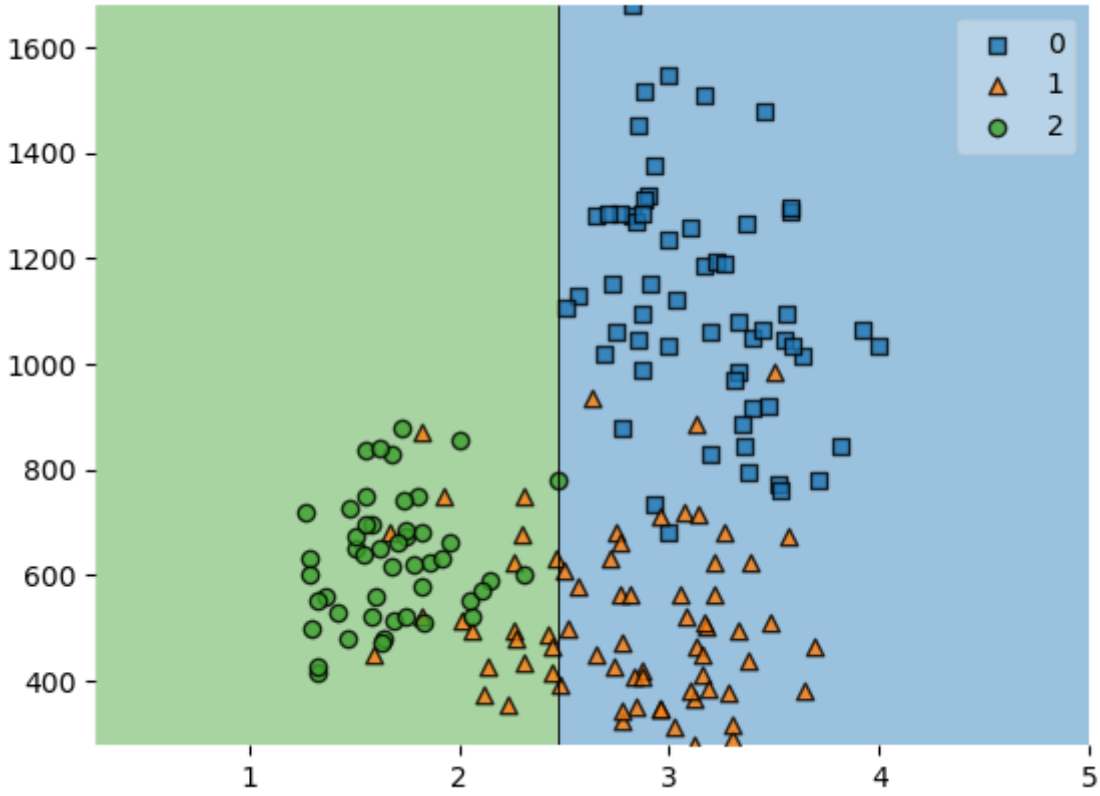
In [14]:
```
from mlxtend.plotting import  plot_decision_regions
plot_decision_regions(X,y,rf)
```

Out[14]:
```
<AxesSubplot:>
```



In [15]:
```
from sklearn.ensemble import AdaBoostClassifier
clf = DecisionTreeClassifier(criterion='entropy', max_depth= 1)
clf_boost = AdaBoostClassifier(clf, n_estimators = 1)
clf_boost.fit(X,y)
plot_decision_regions(X,y,clf_boost)
```

Out[15]: <AxesSubplot:>



In [16]:
```
clf_boost.score(X,y)
```

Out[16]:
```
0.601123595505618
```

In [17]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from mlxtend.classifier import StackingClassifier
```

In [18]:
```
clfk = KNeighborsClassifier(n_neighbors=1)
clfg = GaussianNB()
clfr = RandomForestClassifier()
lr=LogisticRegression()
```

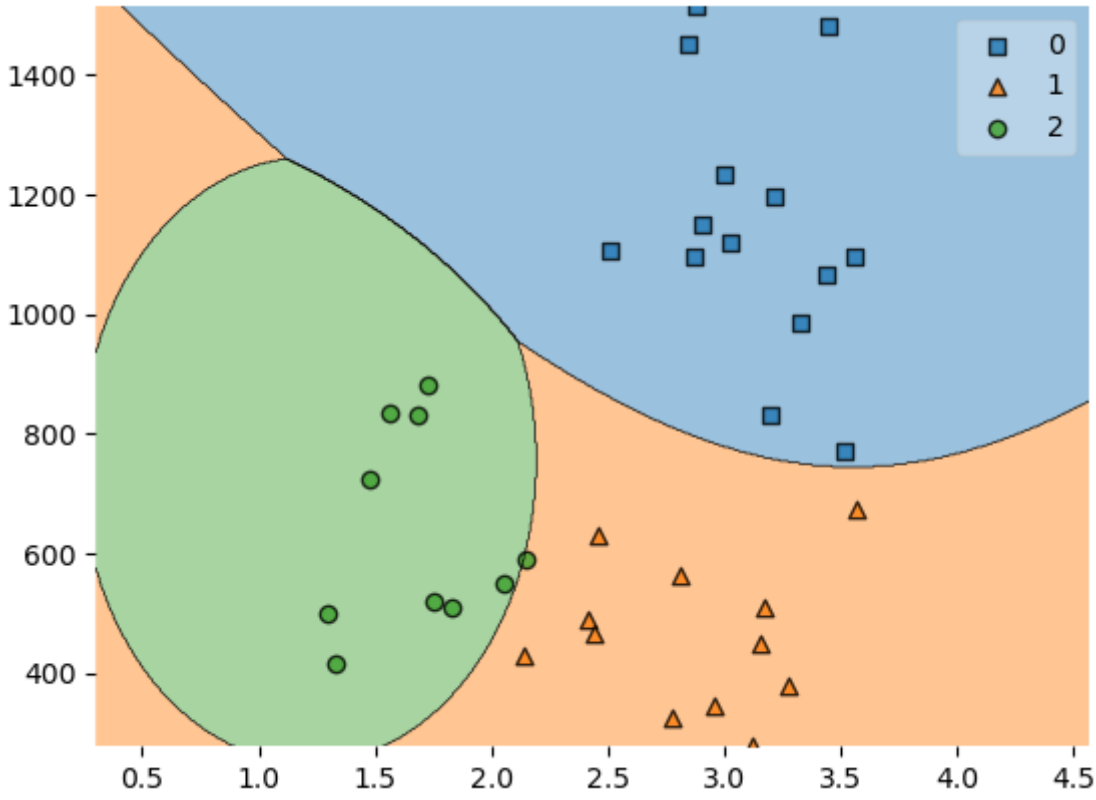
In [19]:
```
sclf = StackingClassifier(classifiers=[clfk, clfg, clfr], meta_classifier= lr)
clfs = [clfk, clfg, clfr,sclf]
clfs
```

Out[19]:
```
[KNeighborsClassifier(n_neighbors=1),
 GaussianNB(),
 RandomForestClassifier(),
 StackingClassifier(classifiers=[KNeighborsClassifier(n_neighbors=1),
                                 GaussianNB(), RandomForestClassifier()],
                    meta_classifier=LogisticRegression())]
```

In [20]:
```
clfg.fit(X,y)
clfg.score(X_test,y_test)
```

Out[20]:
```
0.9722222222222222
```

In [21]:
```
import matplotlib.pyplot as plt
plot_decision_regions(X_test, y_test,clfg)
plt.show()
```



In [22]:
```
print("DecisionTreeClassifier: ",accuracy_score(y_test, preds)*100)
print("BaggingClassifier: ",bg.score(X,y)*100)
print("RandomForestClassifier: ",rf.score(X,y)*100)
```

```
print("AdaBoostClassifier: ",clf_boost.score(X,y)*100)
print("GaussianNB: ",clfg.score(X_test,y_test)*100)
```

```
DecisionTreeClassifier:  100.0
BaggingClassifier:  87.64044943820225
RandomForestClassifier:  99.43820224719101
AdaBoostClassifier:  60.1123595505618
GaussianNB:  97.22222222222221
```

In [ ]:

RESULT:

Hence, we successfully implemented and executed the Ensemble learners with different classifiers on given dataset with best accuracy decision tree.