РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра математического моделирования и искусственного интеллекта

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

Дисциплина: Интеллектуальный анализ данных

Студент: Легиньких Галина

Группа: НФИбд-02-21

Москва 2024

Вариант № 6 (Wine Data Set)

Название файла: wine.data

Класс: cultivar (столбец No 1)

Метод обработки пропущенных значений – медиана признака

Метод нормализации признаков – стандартизация

Алгоритм снижения размерности данных – рекурсивное исключение признаков (RFE)

Дополнительные базовые классификаторы:

классификатор логистической регрессии с полиномиальной зависимостью (degree=2)

классификатор метода опорных векторов

Комбинированный классификатор: VotingClassifier

Ансамблевые классификаторы: RandomForestClassifier, GradientBoostingClassifier

Показатель качества модели – доля верных ответов (accuracy)

Выполнение

1. Считайте заданный набор данных из репозитария UCI, включая указанный в индивидуальном задании столбец с метками классов.

Считала данные.

In [23]:
```
import pandas as pd

url = \
    "https://archive.ics.uci.edu/ml/"+\
    "machine-learning-databases/wine/wine.data"
```

```
wine = pd.read_csv(url, header=None)
wine.columns = ['Cultivar', 'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash',  'Magnesium'
                                'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',  'Co
                                'OD280/OD315 of diluted wines', 'Proline']
wine
```

Out[23]:

	Cultivar	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proant
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	...	
173	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
174	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
175	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
176	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
177	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

178 rows × 14 columns

2. Если среди меток класса имеются пропущенные значения, то удалите записи с пропущенными метками класса. Преобразуйте категориальные признаки в числовые при помощи кодирования меток (label encoding). Если в признаках имеются пропущенные значения, то замените пропущенные значения, используя метод, указанный в индивидуальном задании. Если в признаках пропущенных значений нет, то удалите из набора данных записи, идентифицированные как выбросы при помощи метода кластеризации DBSCAN.

Среди меток класса нет пропущенных значений. Среди признаков тоже.

In [24]:
```
import numpy as np

wine = wine.replace('?', np.NaN) # заменим '?' на np.NaN

print('Число пропущенных значений:')
for col in wine.columns:
    print("NaN in", col,"=", wine[col].isna().sum())
```

```
Число пропущенных значений:
NaN in Cultivar = 0
NaN in Alcohol = 0
NaN in Malic acid = 0
NaN in Ash = 0
NaN in Alcalinity of ash = 0
NaN in Magnesium = 0
NaN in Total phenols = 0
NaN in Flavanoids = 0
NaN in Nonflavanoid phenols = 0
NaN in Proanthocyanins = 0
NaN in Color intensity = 0
NaN in Hue = 0
NaN in OD280/OD315 of diluted wines = 0
NaN in Proline = 0
```

Все признаки числовые.

In [25]:
```
wine.dtypes
```

Out[25]:
```
Cultivar                        int64
Alcohol                       float64
Malic acid                    float64
Ash                           float64
Alcalinity of ash             float64
Magnesium                       int64
Total phenols                 float64
Flavanoids                    float64
Nonflavanoid phenols          float64
Proanthocyanins               float64
Color intensity               float64
Hue                           float64
OD280/OD315 of diluted wines  float64
Proline                         int64
dtype: object
```

Удалила из набора данных записи, идентифицированные как выбросы при помощи метода кластеризации DBSCAN. Решила настроить параметры так: eps = 3, min_samples=3. В принципе, можно было подобрать другие. Так же для этого метода нужно было нормализовать данные, так как у всех параметров разный диапазон (метод стандартизации)

In [26]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

features = wine.drop('Cultivar', axis=1)

# Стандартизация данных
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)

# Применяем DBSCAN
dbscan = DBSCAN(eps=3, min_samples=3)
clusters = dbscan.fit_predict(features_scaled)

wine['Cluster'] = clusters
wine_new = wine[wine['Cluster'] != -1].drop(columns='Cluster')
wine_new = wine_new.reset_index(drop=True)
wine_new
```

Out[26]:

	Cultivar	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	Proant
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	
...	...	...	...	...	...	...	...	...	...	
166	3	13.71	5.65	2.45	20.5	95	1.68	0.61	0.52	
167	3	13.40	3.91	2.48	23.0	102	1.80	0.75	0.43	
168	3	13.27	4.28	2.26	20.0	120	1.59	0.69	0.43	
169	3	13.17	2.59	2.37	20.0	120	1.65	0.68	0.53	
170	3	14.13	4.10	2.74	24.5	96	2.05	0.76	0.56	

171 rows × 14 columns

Подобрала такие значения для метода, чтобы выбросов было не так много. 7 строчек удалилось.

3. Используя метод снижения размерности данных, указанный в индивидуальном задании, определите и оставьте в наборе данных не более четырех признаков.

Алгоритм снижения размерности данных – рекурсивное исключение признаков (RFE)

In [27]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

X = wine_new.drop('Cultivar', axis=1)
y = wine_new['Cultivar']

model = LogisticRegression(max_iter=5000)
rfe = RFE(estimator=model, n_features_to_select=4)
rfe.fit(X, y)
selected_features = X.columns[rfe.support_]
wine_new = wine_new[selected_features.tolist() + ['Cultivar']]
wine_new.head()
```

Out[27]:

	Alcohol	Ash	Flavanoids	OD280/OD315 of diluted wines	Cultivar
0	14.23	2.43	3.06	3.92	1
1	13.20	2.14	2.76	3.40	1
2	13.16	2.67	3.24	3.17	1
3	14.37	2.50	3.49	3.45	1
4	13.24	2.87	2.69	2.93	1

In [28]:
```
# Получаем важность признаков
importance_df = pd.DataFrame({'Feature': X.columns, 'Ranking': rfe.ranking_})
importance_df = importance_df.sort_values(by='Ranking')
importance_df
```

Out[28]:

	Feature	Ranking
0	Alcohol	1
2	Ash	1
6	Flavanoids	1
11	OD280/OD315 of diluted wines	1
9	Color intensity	2
5	Total phenols	3
10	Hue	4
8	Proanthocyanins	5
1	Malic acid	6
3	Alcalinity of ash	7
7	Nonflavanoid phenols	8
4	Magnesium	9
12	Proline	10

4. Нормализуйте оставшиеся признаки набора данных методом, указанным в индивидуальном задании.

Использовала метод стандартизации.

In [29]:
```
columns_to_standardize = wine_new.drop('Cultivar', axis=1).columns
scaler = StandardScaler().fit(wine_new[columns_to_standardize])
wine_new.loc[:, columns_to_standardize] = scaler.transform(wine_new[columns_to_standardize])

wine_new.head()
```

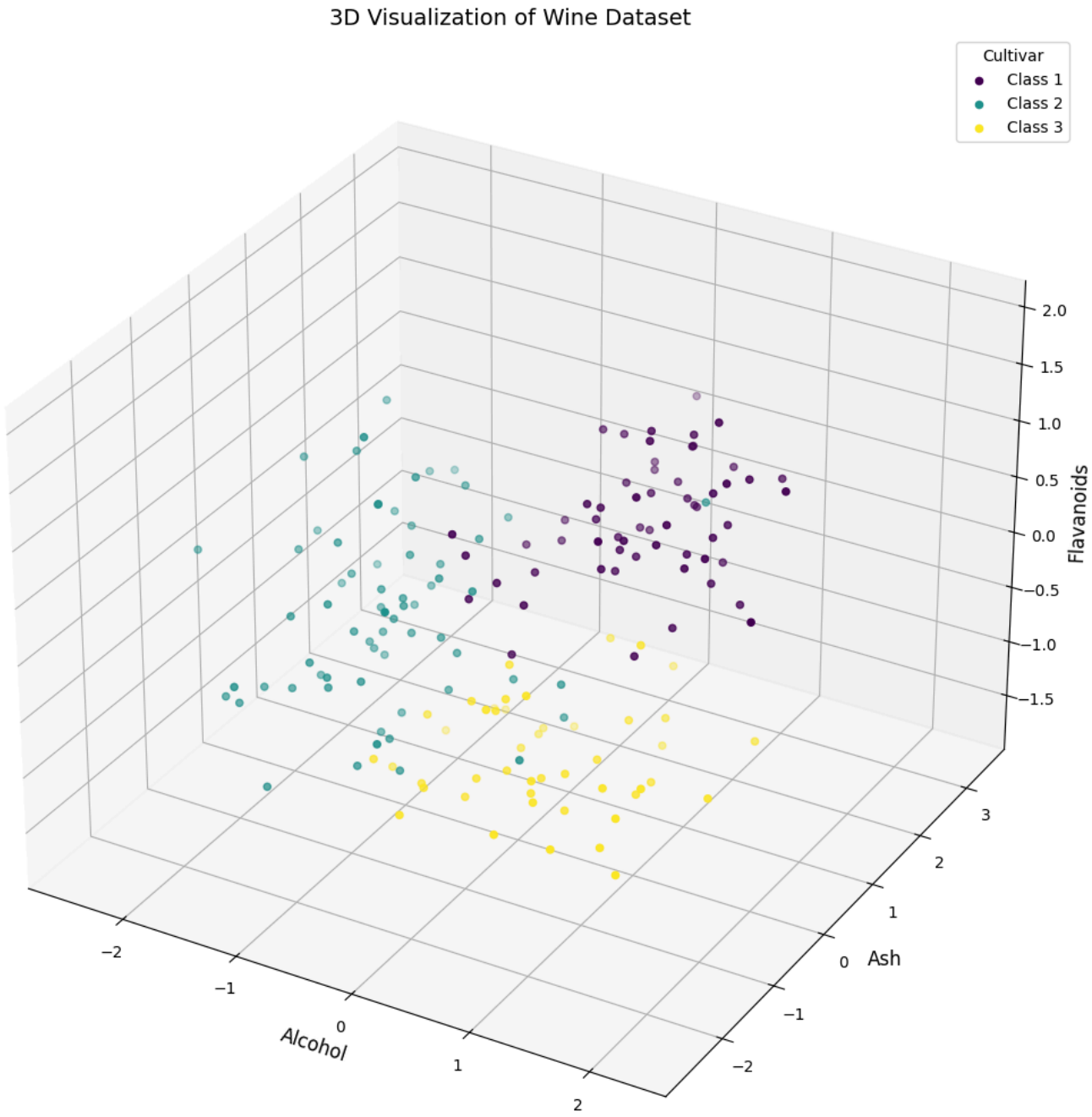
Out[29]:

	Alcohol	Ash	Flavanoids	OD280/OD315 of diluted wines	Cultivar
0	1.522696	0.263799	1.062003	1.847887	1
1	0.228518	-0.885742	0.753452	1.111911	1
2	0.178259	1.215143	1.247133	0.786383	1
3	1.698603	0.541274	1.504258	1.182678	1
4	0.278778	2.007930	0.681457	0.446701	1

5. Визуализируйте набор данных в виде точек в трехмерном пространстве, отображая точки разных классов разными цветами. При визуализации набора данных используйте три признака с наиболее высокой оценкой важности. В качестве подписей осей используйте названия признаков. В подписи рисунка укажите название набора данных. Создайте легенду набора данных.

In [30]:
```python
import matplotlib.pyplot as plt

top_features = selected_features[:3]
X_top = wine_new[top_features]
y = wine_new['Cultivar']

fig = plt.figure(figsize=(13, 13))
ax = fig.add_subplot(111, projection='3d')

unique_classes = y.unique()
colors = plt.cm.get_cmap('viridis', len(unique_classes))

for i, class_label in enumerate(unique_classes):
    ax.scatter(X_top[y == class_label].iloc[:, 0],
               X_top[y == class_label].iloc[:, 1],
               X_top[y == class_label].iloc[:, 2],
               label=f'Class {class_label}',
               color=colors(i))

ax.set_xlabel(top_features[0], fontsize=12)
ax.set_ylabel(top_features[1], fontsize=12)
ax.set_zlabel(top_features[2], fontsize=12)
ax.set_title('3D Visualization of Wine Dataset', fontsize=14)
ax.legend(title='Cultivar')
plt.show()
```

```
C:\Users\galin\AppData\Local\Temp\ipykernel_9248\519871044.py:11: MatplotlibDeprecationWarnin
g: The get_cmap function was deprecated in Matplotlib 3.7 and will be removed two minor releas
es later. Use ``matplotlib.colormaps[name]`` or ``matplotlib.colormaps.get_cmap(obj)`` instea
d.
  colors = plt.cm.get_cmap('viridis', len(unique_classes))
```



6. Разбейте набор данных на обучающую и тестовую выборки. Создайте и обучите классификатор на основе деревьев решений с глубиной дерева не более 4, определите долю верных ответов на тестовой выборке и визуализируйте границу принятия решений и построенное дерево решений. При визуализации границы принятия решений используйте два признака с наиболее высокой оценкой важности.

Разделила данные на обучающую и тестовую выборки.

In [31]:
```
from sklearn.model_selection import train_test_split

X = wine_new[top_features[:2]]
y = wine_new['Cultivar']

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

Out[31]:
```
((136, 2), (136,), (35, 2), (35,))
```

Создала и перешла к обучению классификатора на основе деревьев решений. Оценила точности на тестовой выборке.

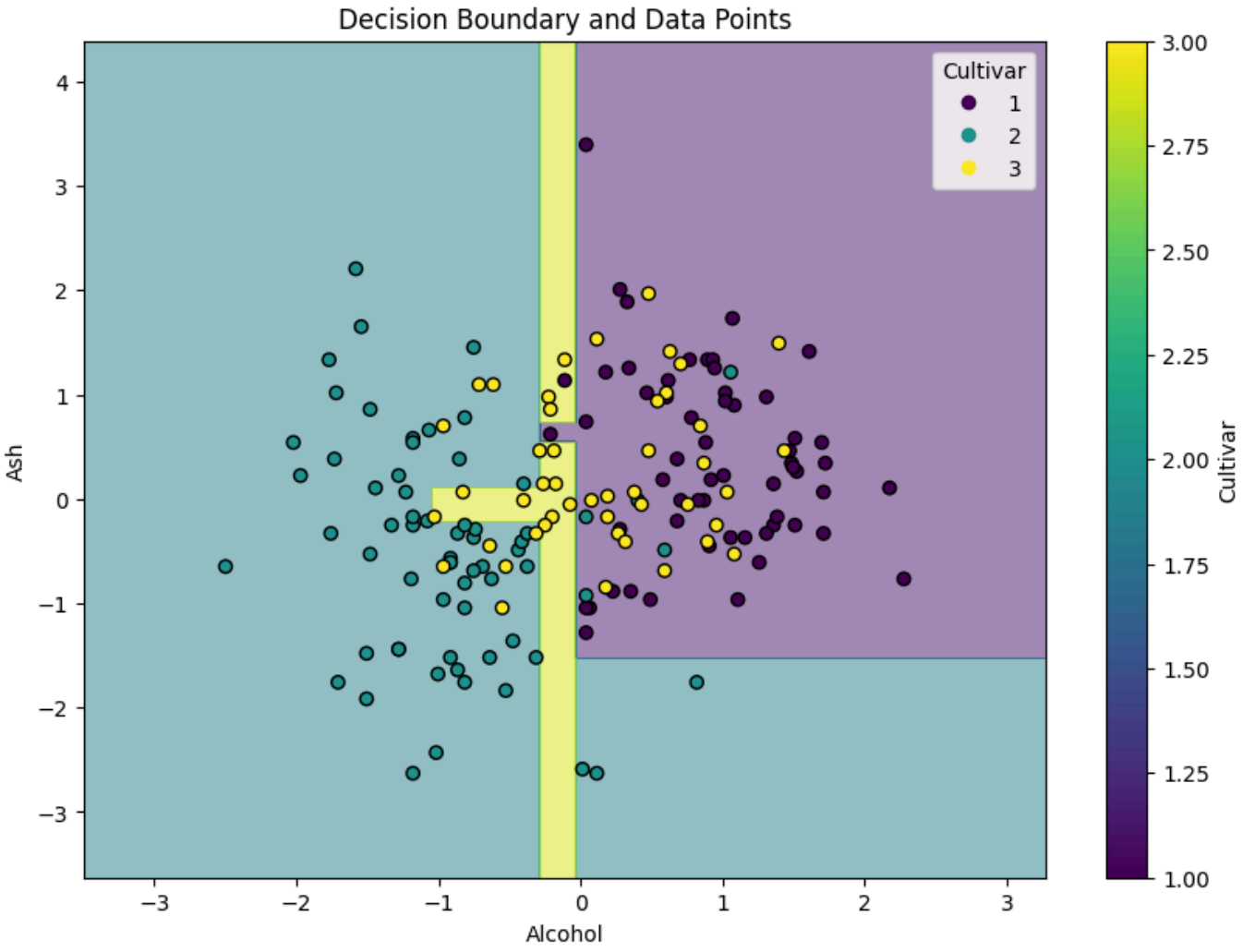
In [32]:
```
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier, plot_tree

dt_classifier = DecisionTreeClassifier(max_depth=4, random_state=42)
dt_classifier.fit(X_train, y_train)

# Оценка точности на тестовой выборке
y_pred = dt_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy on test set: {accuracy:.2f}")
```

```
Accuracy on test set: 0.77
```

Визуализировала границы принятия решений.

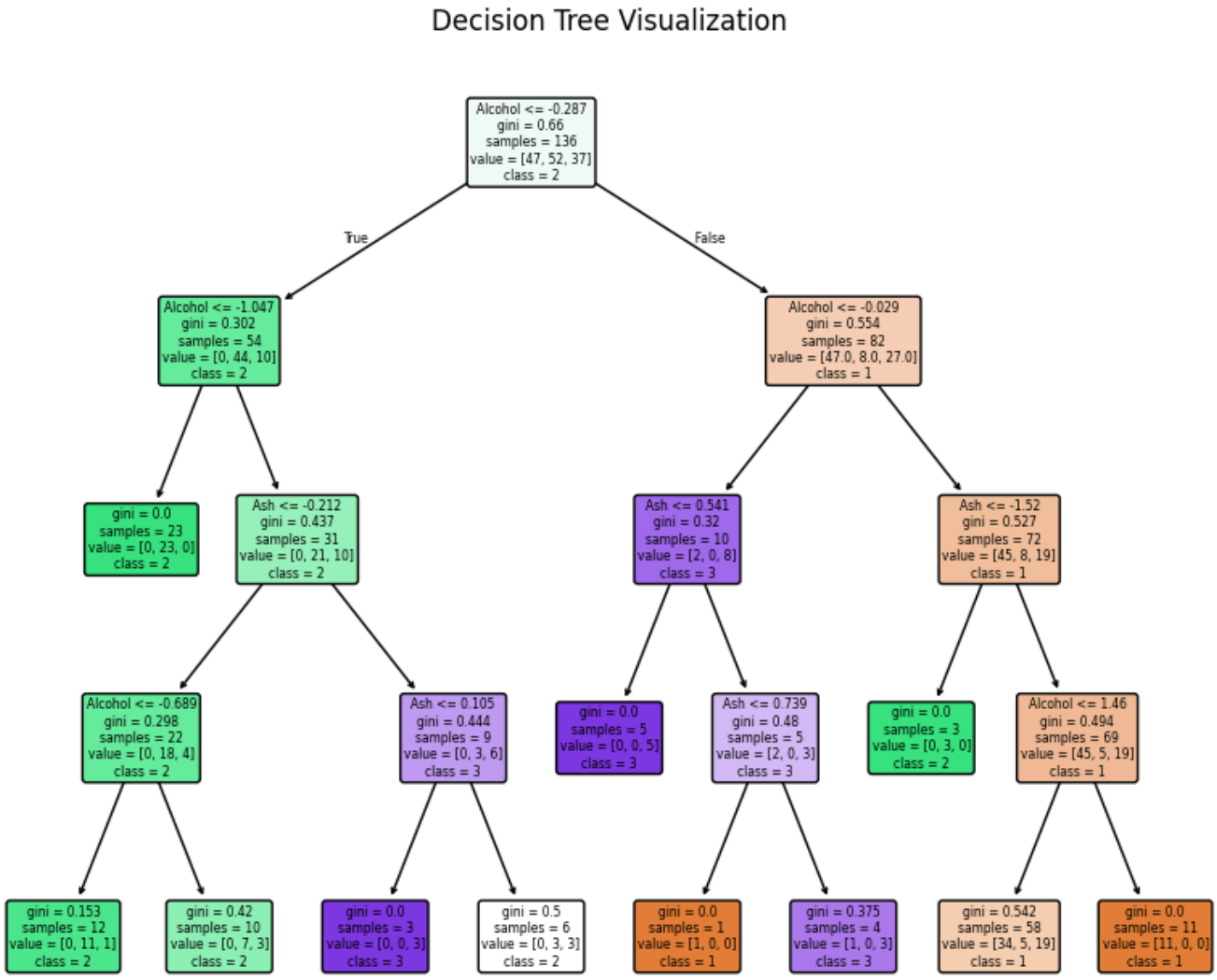
In [33]:
```
x_min, x_max = X[selected_features[0]].min() - 1, X[selected_features[0]].max() + 1
y_min, y_max = X[selected_features[1]].min() - 1, X[selected_features[1]].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))

Z = dt_classifier.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.figure(figsize=(10, 7))
plt.contourf(xx, yy, Z, alpha=0.5, cmap='viridis')

scatter = plt.scatter(X[selected_features[0]], X[selected_features[1]], c=y, edgecolors='k',

plt.xlabel(selected_features[0])
plt.ylabel(selected_features[1])
plt.title('Decision Boundary and Data Points')
plt.colorbar(scatter, label='Cultivar')
plt.legend(*scatter.legend_elements(), title='Cultivar')
plt.show()
```

```
C:\Users\galin\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py:493: UserWarnin
g: X does not have valid feature names, but DecisionTreeClassifier was fitted with feature nam
es
  warnings.warn(
```



Визуализировала дерево решений.

In [34]:
```
plt.figure(figsize=(10, 8))
plot_tree(dt_classifier, filled=True, feature_names=top_features[:2], class_names=y.unique().
plt.title('Decision Tree Visualization')
plt.show()
```



7.

Дополнительные базовые классификаторы:

классификатор логистической регрессии с полиномиальной зависимостью (degree=2)

классификатор метода опорных векторов

Комбинированный классификатор: VotingClassifier

Ансамблевые классификаторы: RandomForestClassifier, GradientBoostingClassifier

Показатель качества модели – доля верных ответов (accuracy)

In [35]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.svm import SVC

# Классификатор логистической регрессии с полиномиальной зависимостью
poly_classifier = make_pipeline(PolynomialFeatures(degree=2), LogisticRegression(max_iter=100
poly_classifier.fit(X_train, y_train)

# Классификатор метода опорных векторов
svm_classifier = SVC(probability=True)
svm_classifier.fit(X_train, y_train)
```

```
classifiers = [poly_classifier, svm_classifier, dt_classifier]
classifier_names = ['Logistic Regression (Poly)', 'SVM', 'Decision Tree']

for clf, name in zip(classifiers, classifier_names):
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f'Accuracy of {name}: {accuracy:.2f}')
```

```
Accuracy of Logistic Regression (Poly): 0.86
Accuracy of SVM: 0.80
Accuracy of Decision Tree: 0.77
```

In [36]:
```
from sklearn.ensemble import VotingClassifier, RandomForestClassifier, GradientBoostingClassi

# Создаем VotingClassifier
voting_classifier = VotingClassifier(
    estimators=[
        ('log_reg_poly', poly_classifier),
        ('svm', svm_classifier),
        ('dt', dt_classifier),
    ],
    voting='soft'
)

# Обучение комбинированного классификатора
voting_classifier.fit(X_train, y_train)
y_pred_voting = voting_classifier.predict(X_test)
accuracy_voting = accuracy_score(y_test, y_pred_voting)
print(f'Accuracy of Voting Classifier: {accuracy_voting:.2f}')
```

```
Accuracy of Voting Classifier: 0.83
```

8. Постройте и обучите пару ансамблевых классификаторов, указанных в индивидуальном задании, и сравните их производительность по показателю, указанному в индивидуальном задании.

Сравнила производительность.

In [37]:
```
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)
y_pred_rf = rf_classifier.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f'Accuracy of Random Forest Classifier: {accuracy_rf:.2f}')

gb_classifier = GradientBoostingClassifier(random_state=42)
gb_classifier.fit(X_train, y_train)
y_pred_gb = gb_classifier.predict(X_test)
accuracy_gb = accuracy_score(y_test, y_pred_gb)
print(f'Accuracy of Gradient Boosting Classifier: {accuracy_gb:.2f}')
```

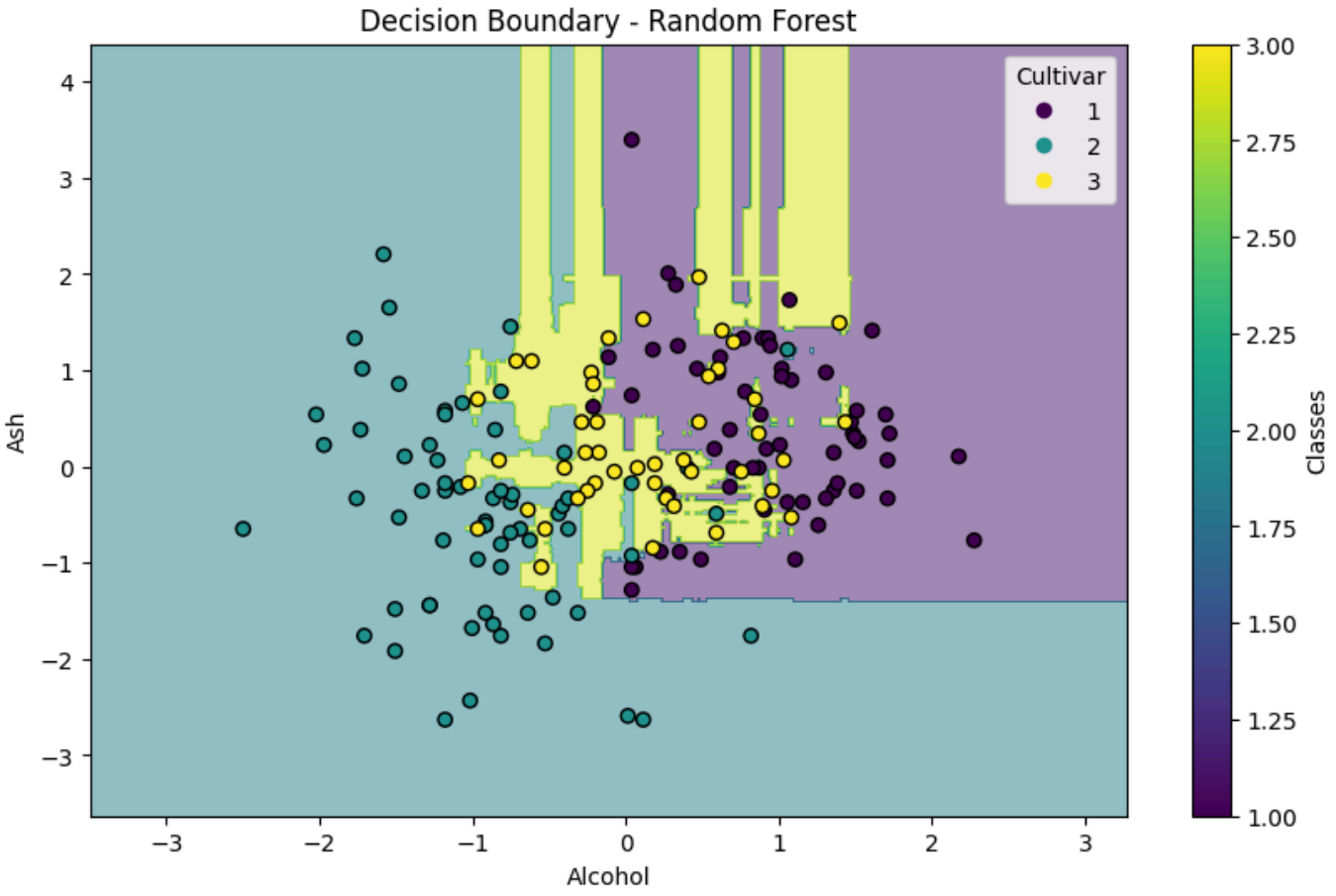
```
Accuracy of Random Forest Classifier: 0.74
Accuracy of Gradient Boosting Classifier: 0.69
```

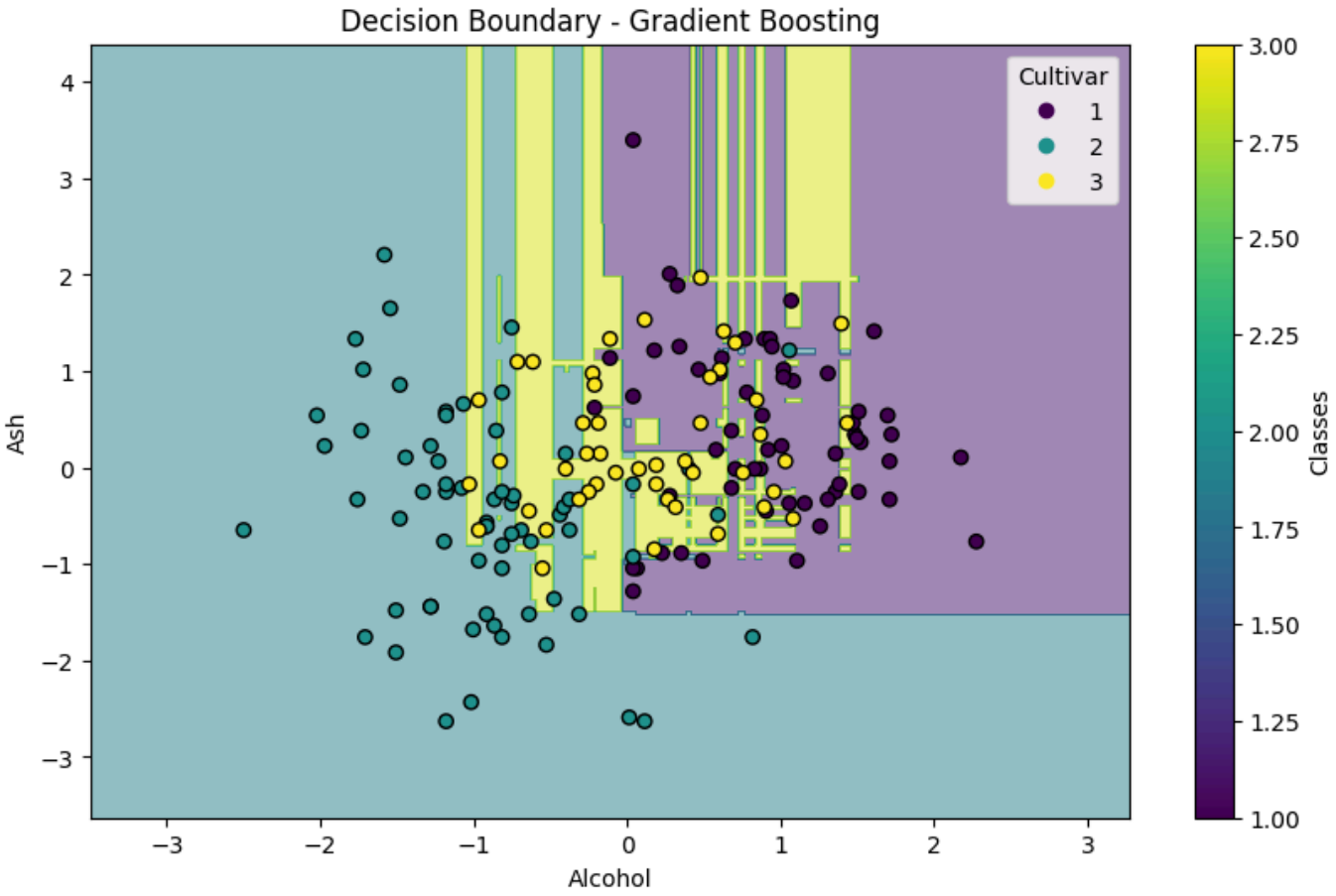
9. Постройте границы принятия решений ансамблевых классификаторов с визуализацией точек набора данных разных классов разными цветами. Подпишите оси и рисунок.

In [38]:
```
x_min, x_max = X[top_features[0]].min() - 1, X[top_features[0]].max() + 1
y_min, y_max = X[top_features[1]].min() - 1, X[top_features[1]].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))

Z_rf = rf_classifier.predict(np.c_[xx.ravel(), yy.ravel()])
```

```
Z_rf = Z_rf.reshape(xx.shape)

Z_gb = gb_classifier.predict(np.c_[xx.ravel(), yy.ravel()])
Z_gb = Z_gb.reshape(xx.shape)

plt.figure(figsize=(10, 6))
plt.contourf(xx, yy, Z_rf, alpha=0.5, cmap='viridis')

scatter_rf = plt.scatter(X[top_features[0]], X[top_features[1]], c=y, edgecolors='k', cmap='v
plt.xlabel(top_features[0])
plt.ylabel(top_features[1])
plt.title('Decision Boundary - Random Forest')
plt.colorbar(scatter_rf, label='Classes')
plt.legend(*scatter_rf.legend_elements(), title='Cultivar')
plt.show()

plt.figure(figsize=(10, 6))
plt.contourf(xx, yy, Z_gb, alpha=0.5, cmap='viridis')

scatter_gb = plt.scatter(X[top_features[0]], X[top_features[1]], c=y, edgecolors='k', cmap='v
plt.xlabel(top_features[0])
plt.ylabel(top_features[1])
plt.title('Decision Boundary - Gradient Boosting')
plt.colorbar(scatter_gb, label='Classes')
plt.legend(*scatter_gb.legend_elements(), title='Cultivar')
plt.show()
```

```
C:\Users\galin\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py:493: UserWarnin
g: X does not have valid feature names, but RandomForestClassifier was fitted with feature nam
es
  warnings.warn(
C:\Users\galin\AppData\Roaming\Python\Python312\site-packages\sklearn\base.py:493: UserWarnin
g: X does not have valid feature names, but GradientBoostingClassifier was fitted with feature
names
  warnings.warn(
```




Decision Boundary - Gradient Boosting
Cultivar
1
2
3
Ash
Alcohol
Classes
4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00