人工智能基础与进阶

线性回归与分类

上海交通大学

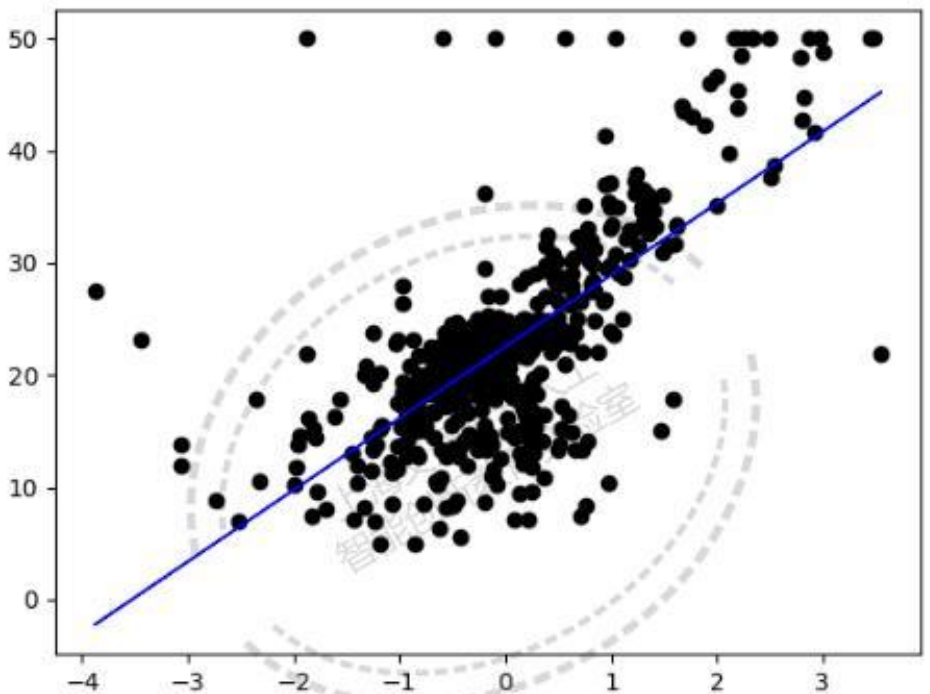
第七节 机器学习相关代码

简单线性回归

实验目的：本实验通过对波士顿郊区房价的预测来对线性回归进行联系。

数据集简介：

波士顿房价数据集来自卡内基梅隆大学StatLib库，涵盖了麻省波士顿的506个不同郊区的房屋数据，404条训练数据集，102条测试数据集 每条数据14个字段，包含13个属性和1个房价的平均值。



波士顿房价（纵轴）与房屋平均房间数（横轴，该数据已归一化）的关系图

简单线性回归

字段介绍：

列名	说明	类型
CRIM	城镇人均犯罪率	float
ZN	住宅用地超过 25000 sq.ft. 的比例	float
INDUS	城镇非零售商用土地的比例	float
CHAS	查理斯河空变量（如果边界是河流，则为1；否则为0）	int
NOX	一氧化氮浓度	float
RM	住宅平均房间数	float
AGE	1940 年之前建成的自用房屋比例	float
DIS	到波士顿五个中心区域的加权距离	float
RAD	辐射性公路的接近指数	float
TAX	每 10000 美元的全值财产税率	float
PTRATIO	城镇师生比例	float
B	1000（Bk-0.63）^ 2，其中 Bk 指代城镇中黑人的比例	float
LSTAT	人口中地位低下者的比例	float
MEDV	自住房的平均房价，以千美元计	float

简单线性回归

1. 导入数据集并分割：

```
"获取数据集并分割"
# diabetes = datasets.load_diabetes()
boston = datasets.load_boston()
X = boston.data
Y = boston.target
# X_train, X_test, Y_train, Y_test = train_test_split(X, Y, random_state= 33, test_size= 0.25)
```

2. 选取拟合变量并归一化：

```
"选取拟合自变量"
X_train = np.array([X[:, 5]]) # 数字代表所选取的字段
Y_train = Y

"数据标准化处理"
ss_X = StandardScaler()
X_train = ss_X.fit_transform(X_train.T)
```

建议使用：5-RM（平均房间数），12-LSTAT（低收入人口比例）

简单线性回归

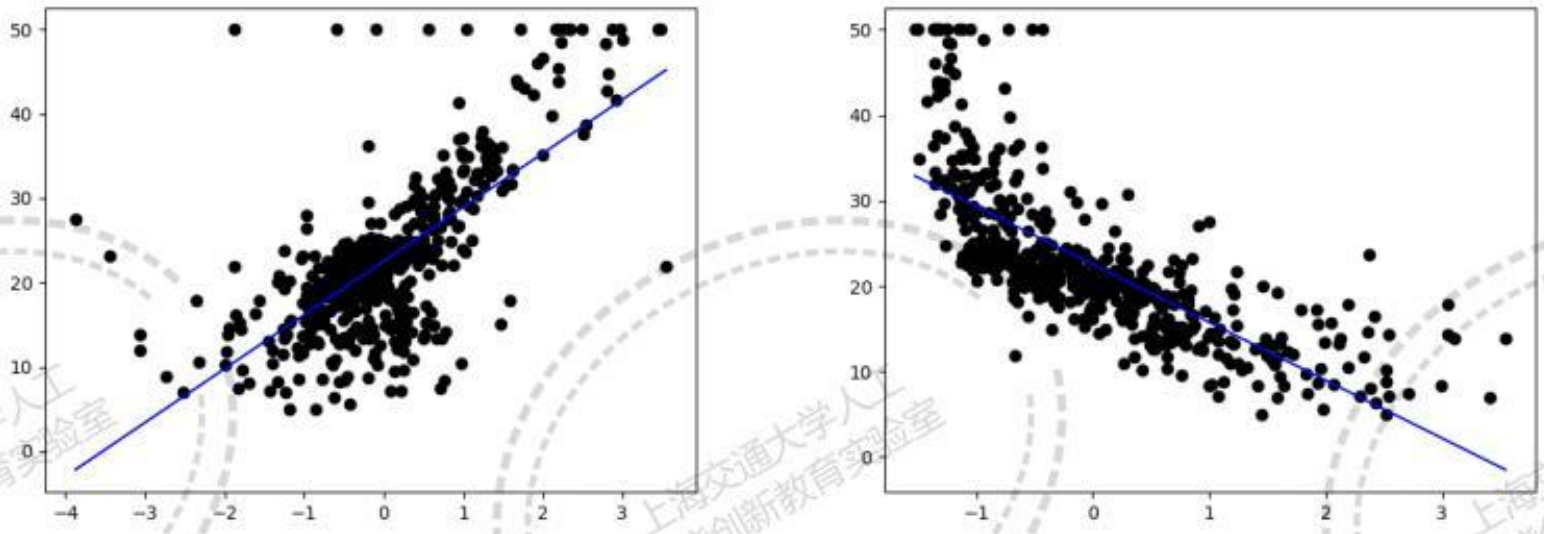
3. 建立回归模型：多种回归模型可供选择

```
"多种方法可选"
method = 'linear'
if method == 'linear':
    "创建线性回归对象"
    linear = linear_model.LinearRegression()
    "使用训练数据来训练模型"
    linear.fit(X_train, Y_train)
    "画出预测的点"
    plt.plot(X_train, linear.predict(X_train), color="blue", linewidth=1)

elif method == 'ridge':
    "岭回归-交叉验证"
    ridge = linear_model.RidgeCV()
    ridge.fit(X_train, Y_train)
    plt.plot(X_train, ridge.predict(X_train), color="blue", linewidth=1)

elif method == 'lasso':
    lasso = linear_model.LassoCV()
    lasso.fit(X_train, Y_train)
    plt.plot(X_train, lasso.predict(X_train), color="blue", linewidth=1)

else:
    print("wrong method")
```

简单线性回归

4. 可视化实验结果：

```
"画出样本点"
plt.scatter(X_train, Y_train, color="black")
plt.show()
```



左图：波士顿房价（纵轴）与房屋平均房间数（横轴，该数据已归一化）的关系图

右图：波士顿房价（纵轴）与地区低收入人口比例（横轴，该数据已归一化）的关系图

多元线性回归

实验目的：泰坦尼克号数据集中有泰坦尼克号所有乘客的信息，包括姓名、性别、年龄等。然而，其中部分乘客的年龄信息缺失。本实验即是通过对数据集中的其他信息进行回归，来预测未知的乘客年龄。

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

实验步骤：

1. 读取数据集并清除冗余数据；
2. 转换数据格式（如性别）；
3. 分离训练集和测试集；
4. 构造回归模型；
5. 利用训练好的模型对未知年龄进行预测。

多元线性回归

1. 读取数据集，并清除冗余数据：从原数据集中删除无效数据，例如姓名、船舱等。
   清除后的数据集如下图所示。

Survived	Pclass	Sex	Age	SibSp	Parch	Fare
0	3	male	22.0	1	0	7.2500
1	1	female	38.0	1	0	71.2833
1	3	female	26.0	0	0	7.9250
1	1	female	35.0	1	0	53.1000
0	3	male	35.0	0	0	8.0500

2. 转换数据格式：部分数据为离散化或字符类型，无法参与回归计算。
   转换格式后的数据集如下图所示。

Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Pclass_3	Sex_female	Sex_male
0	3	male	22.0	1	0	7.2500	0	0	1	0	1
1	1	female	38.0	1	0	71.2833	1	0	0	1	0
1	3	female	26.0	0	0	7.9250	0	0	1	1	0
1	1	female	35.0	1	0	53.1000	1	0	0	1	0
0	3	male	35.0	0	0	8.0500	0	0	1	0	1

多元线性回归

3. 分离数据集：将原数据集中已知年龄部分作为训练集，未知年龄部分为测试集。

```
"""分离训练集和测试集"""
missing = new_data.loc[origin_data.Age.isnull(), ]
no_missing = new_data.loc[~origin_data.Age.isnull(), ]
```

4. 构造回归模型：将原数据集中已知年龄部分作为训练集，未知年龄部分为测试集。

```
"""构造回归模型"""
Class = sm.formula.ols(formula= 'Age ~ Survived+SibSp+Pclass_1+Pclass_2', data = no_missing)
model = Class.fit()
print(model.summary())
```

‘Age ~ Survived+SibSp+Pclass_1+Pclass_2’

Age：年龄，作为因变量。

Survived+SibSp+Pclass_1+Pclass_2：经过检验后得出的有影响的自变量。

多元线性回归

5. 利用训练好的模型对未知年龄进行预测。

	Survived	Age	SibSp	Parch	Fare	Pclass_1	Pclass_2	Sex_female
5	0	29.432299	0	0	8.4583	0	0	0
17	1	28.139132	0	0	13.0000	0	1	0
19	1	22.423082	0	0	7.2250	0	0	1
26	0	29.432299	0	0	7.2250	0	0	0
28	1	22.423082	0	0	7.8792	0	0	1
..	...	...	...	...	...	...	...	...
859	0	29.432299	0	0	7.2292	0	0	0
863	0	-6.250881	8	2	69.5500	0	0	1
868	0	29.432299	0	0	9.5000	0	0	0
878	0	29.432299	0	0	7.8958	0	0	0
888	0	24.971901	1	2	23.4500	0	0	1

多元线性回归——整体代码

```
origin_data = pd.read_csv("./train.csv")

"""缺失数据比例"""
# c = origin_data.isnull().sum(axis= 0) / origin_data.shape[0]
# print(c)

"""剔除数据"""
origin_data.drop(labels= ['PassengerId', 'Name', 'Ticket', 'Cabin', 'Embarked'], axis= 1, inplace= True)

"""数据转换"""
origin_data.Pclass = origin_data.Pclass.astype(str)
dummies = pd.get_dummies(data = origin_data[['Pclass', 'Sex']])
new_data = pd.concat([origin_data, dummies], axis= 1)

"""剔除冗余数据"""
# print(origin_data.head())
new_data.drop(labels= ['Pclass', 'Pclass_3', 'Sex', 'Sex_male'], axis= 1, inplace= True)

"""分离训练集和测试集"""
missing = new_data.loc[origin_data.Age.isnull(), ]
no_missing = new_data.loc[~origin_data.Age.isnull(), ]

"""构造回归模型"""
Class = sm.formula.ols(formula= 'Age ~ Survived+SibSp+Pclass_1+Pclass_2', data = no_missing)
model = Class.fit()
# print(model.summary())

"""年龄预测"""
prediction = model.predict(exog = missing[['Survived', 'SibSp', 'Pclass_1', 'Pclass_2']])
missing.loc[:, 'Age'] = prediction
print(missing)
```

Iris-鸢尾花数据集简介

Iris是一类多重变量分析的数据集。其包含150个数据样本，分为3类，每类50个数据，每个数据包含4个属性。

花萼长度	花萼宽度	花瓣长度	花瓣宽度	品种
5.1	3.5	1.4	0.2	Iris-setosa
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
6.3	3.3	6.0	2.5	Iris-virginica
5.8	2.7	5.1	1.9	Iris-virginica
7.1	3.0	5.9	2.1	Iris-virginica
6.3	2.9	5.6	1.8	Iris-virginica
6.5	3.0	5.8	2.2	Iris-virginica

支持向量机

实验目的：本实验使用Python自带的鸢尾花数据集，根据花瓣的长度和宽度两个信息，利用支持向量机对鸢尾花品种进行分类。

```
"获取鸢尾花数据集"
iris = datasets.load_iris()
print(iris.target)
"选取其中两类，数据集中一共有三种鸢尾花，选出其中两种的全部数据集"
index = np.where(iris.target > 0)
"选取其中两个特征，一共有四个特征"
#print(iris.data.shape)
x = np.squeeze(iris.data[index, 2:4])
print(x)
y = iris.target[index]
"画出样本点"
plt.scatter(x[:, 0], x[:, 1], c=y, marker='o')
```

该数据集原本有三个鸢尾花品种，用iris.target记录。此处去除了品种0。花的特征有花萼的长宽、花瓣的长宽共4个数据，此处仅考虑花瓣的长宽。

支持向量机

训练SVM模型：核函数的选择。

本实验中，因鸢尾花数据集线性可分，所以我们选取线性核函数。

```
"选择核函数"
clf = svm.SVC(kernel="linear")
"训练模型"
clf.fit(x, y)
```

面临复杂问题时，可能需要选择非线性核函数，将线性不可分的数据转换为线性可分。

例如：保留品种0的鸢尾花，且加以考虑花萼的长宽。

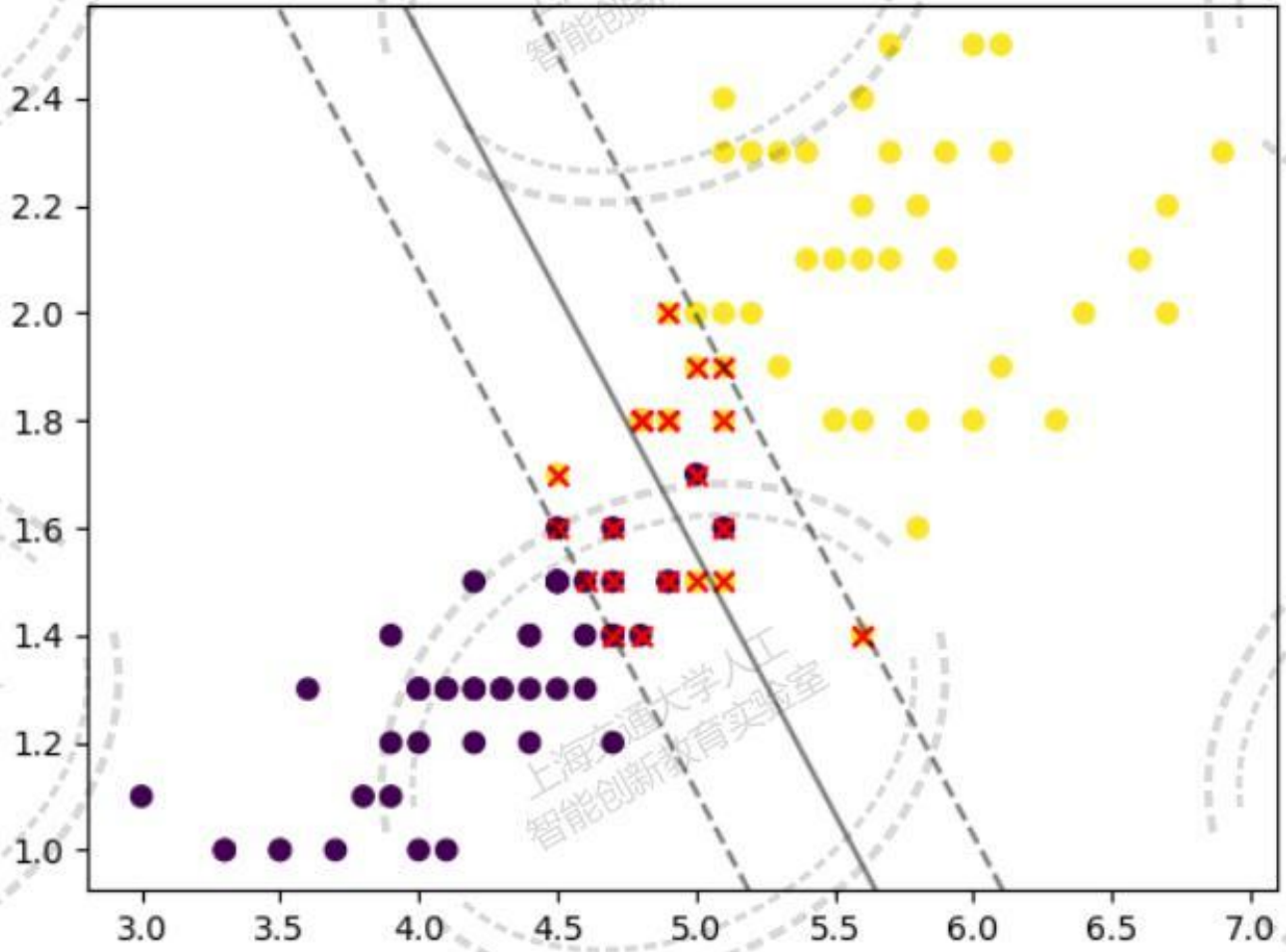
常用核函数包括：线性函数、多项式函数、高斯函数等。

支持向量机

在散点图上画出支持向量、分界面。

```
"画支持向量"
plt.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1], c='red', marker='x')
"画分界面"
ax = plt.gca()
x1 = np.linspace(plt.xlim()[0], plt.xlim()[1], 30)
y1 = np.linspace(plt.ylim()[0], plt.ylim()[1], 30)
#print(plt.ylim())
Y, X = np.meshgrid(y1, x1)
P = np.zeros_like(X)

for i, xi in enumerate(x1):
    for j, yj in enumerate(y1):
        tmp = np.array([xi, yj])
        tmp = tmp.reshape((1, -1))
        P[i, j] = clf.decision_function(tmp)
ax.contour(X, Y, P, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])
plt.show()
```

支持向量机——实验结果



决策树

实验目的：本实验同样使用鸢尾花数据集，使用决策树的方法对花的品种进行分类。

实验需求：Python matplotlib库。如果提示缺失，则通过以下指令安装。

```
pip install matplotlib
```

实验步骤：

1. 导入数据，并将其分割为训练集和测试集；
2. 构建决策树模型并训练；
3. 对测试集数据进行预测；
4. 可视化实验结果。

决策树

1. 导入数据：

从python数据库中导入数据，或者导入自己感兴趣的其他数据。（注意数据格式转换）

```
"""导入鸢尾花的数据集"""
iris = load_iris()
"""导入其他数据集，分割训练集和测试集"""
# data = pd.read_csv("./data_addr")
# df = pd.DataFrame(data.data, columns= data.feature_names)
# df['target'] = data.target
# X_train, X_test, Y_train, Y_test = train_test_split(df[data.feature_names], df['target'], random_state= 0)
```

2. 构建决策树模型并训练：

根据个性化需求，选择决策树的判别标准、本身特性等。

具体体现在函数 DecisionTreeClassifier 的参数中。

```
"""构建模型"""
clf = tree.DecisionTreeClassifier()
"""训练模型"""
clf.fit(iris.data, iris.target)
# clf.fit(X_train, Y_train)
```

决策树

函数 DecisionTreeClassifier 主要参数

criterion : gini / entropy，前者是基尼系数，后者是信息熵。

max_depth : int / None，决策树的最大深度，深度越大，越容易过拟合。

max_leaf_nodes : int / None，最大叶节点数，默认是“None”，即不限制最大的叶子节点数。

min_samples_leaf: int / None，叶节点最少样本数，如果某叶子节点数目小于样本数，则会和兄弟节点一起被剪枝。

min_samples_split: int / None，结点的最小样本数量，当样本数量可能小于此值时，结点将不会在划分。

例：

```
tree.DecisionTreeClassifier(criterion='entropy', min_samples_leaf=3)
```

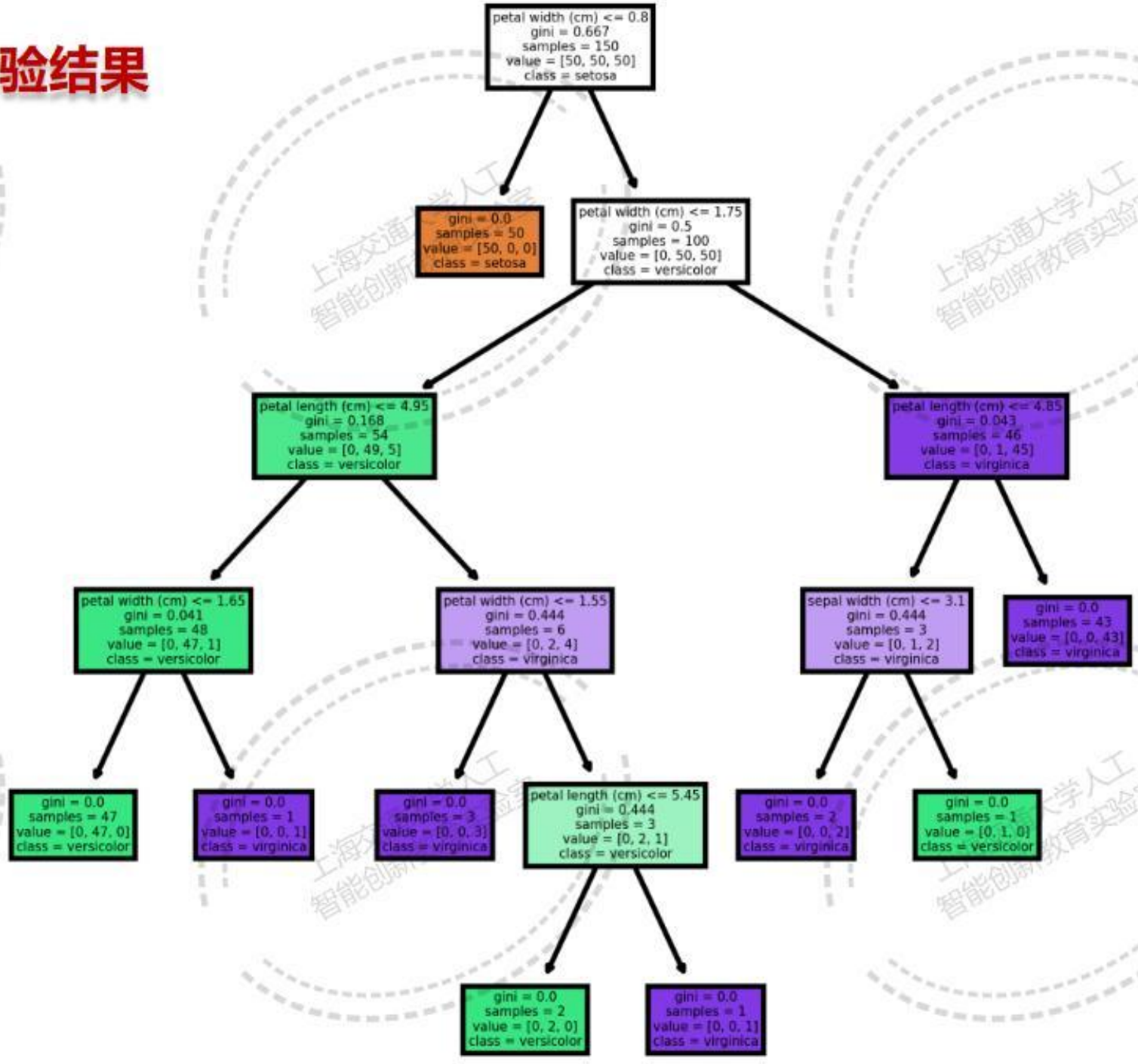
决策树

3. 对测试集数据进行预测（使用自定义的数据集）。

4. 可视化实验结果。

```
"""结果预测"""
# clf.predict(X_test)

"""使用matplotlib绘图"""
fn=['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']
cn=['setosa', 'versicolor', 'virginica']
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=600)
tree.plot_tree(clf,
               feature_names = fn,
               class_names=cn,
               filled = True)
fig.savefig('imagename.png')
```

决策树——实验结果


petal width (cm) <= 0.8
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
petal length (cm) <= 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
petal length (cm) <= 4.85
gini = 0.043
samples = 46
value = [0, 1, 45]
class = virginica
petal width (cm) <= 1.65
gini = 0.041
samples = 48
value = [0, 47, 1]
class = versicolor
petal width (cm) <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica
sepal width (cm) <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica
gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
petal length (cm) <= 5.45
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica


谢谢聆听
THANKS FOR YOUR ATTENTION