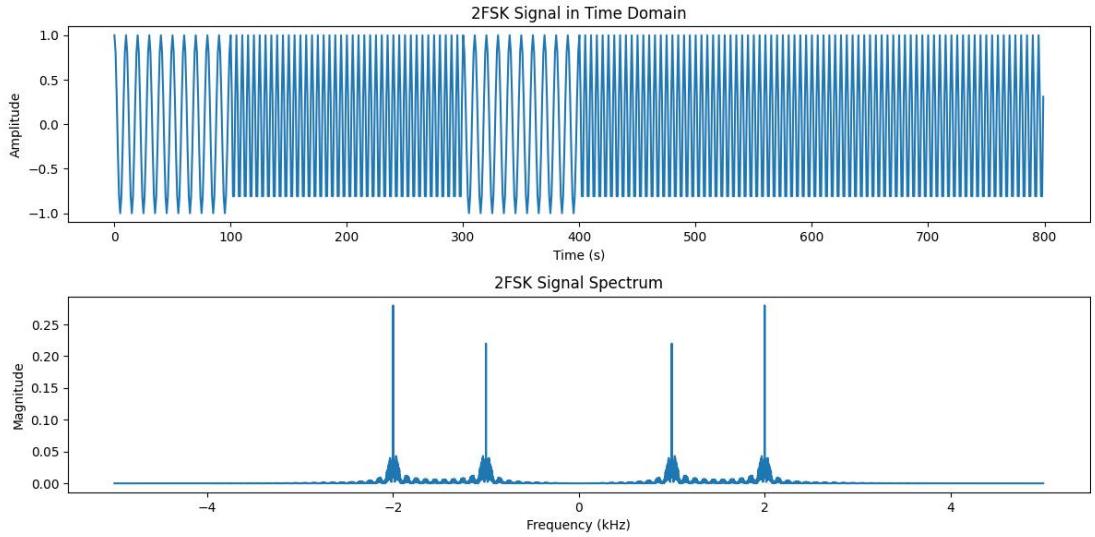
Matplotlib 绘图

一、2FSK（Frequency Shift Keying）是一种数字调制方式，它使用两种不同频率的信号来表示二进制数据的 0 和 1。接下来，我们将使用 Python 生成一个 2FSK 信号，并绘制其时域波形和频谱图。

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import fft, fftshift

# 参数设置
f0 = 1e3            # 低频率，对应比特 0，1 kHz
f1 = 2e3            # 高频率，对应比特 1，2 kHz
bit_rate = 100   # 比特率，100 bps
T = 1                # 信号持续时间，1 秒
fs = 10e3           # 采样率，10 kHz

# 生成随机二进制数据
np.random.seed(0)   # 设置随机种子以确保可重复性
data = np.random.randint(0, 2, int(T * bit_rate))

# 生成时间向量
t = np.arange(0, T, 1/fs)

# 生成 2FSK 信号
fsk_signal = np.zeros_like(t)
bit_duration = int(fs / bit_rate)

for i, bit in enumerate(data):
    f = f1 if bit else f0
    fsk_signal[i*bit_duration:(i+1)*bit_duration] = np.cos(2 * np.pi * f * t[i*bit_duration:(i+1)*bit_duration])

# 计算频谱
N = len(fsk_signal)
f = np.linspace(-fs/2, fs/2, N)
fsk_spectrum = fftshift(fft(fsk_signal))

# 绘制时域 2FSK 信号
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.plot(fsk_signal[:800])
```

```
plt.title('2FSK Signal in Time Domain')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')

# 绘制频域 2FSK 信号
plt.subplot(2, 1, 2)
plt.plot(f/1e3, np.abs(fsk_spectrum)/N)
plt.title('2FSK Signal Spectrum')
plt.xlabel('Frequency (kHz)')
plt.ylabel('Magnitude')
plt.tight_layout()
plt.show()
```

绘图结果：

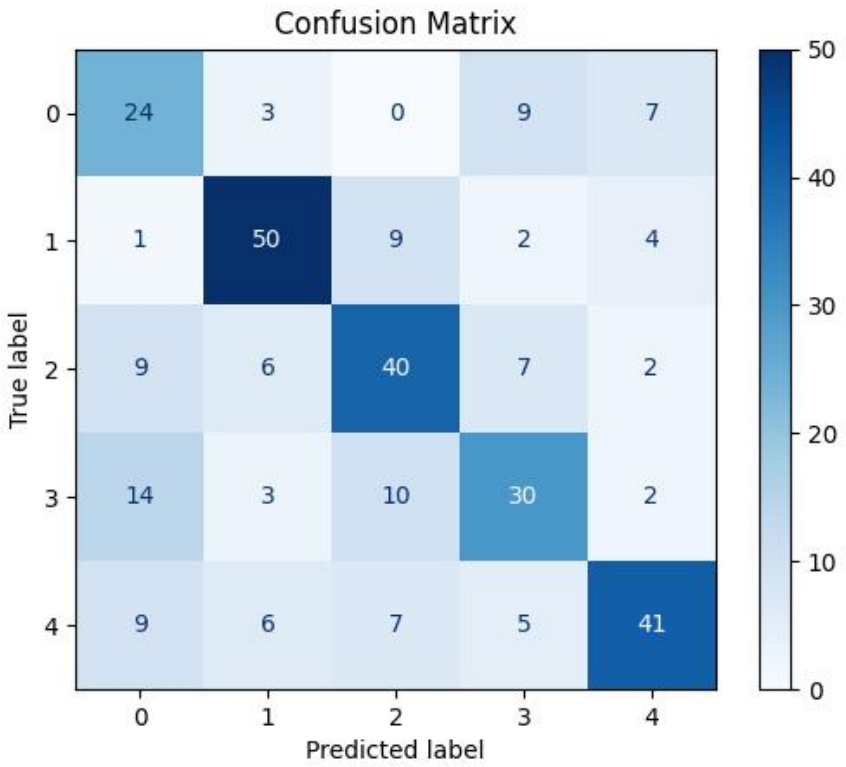


二、绘制混淆矩阵是评估分类模型性能的一个重要步骤。混淆矩阵可以帮助我们直观地看到模型预测的结果与真实标签之间的关系。以下是一个使用 Python 绘制混淆矩阵的示例，我们将使用一个简单的分类问题来演示。

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# 生成一个包含 5 个类别的分类数据集
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5,
                           n_classes=5, random_state=42)
```

```
# 划分数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 创建并训练分类模型
model = LogisticRegression(max_iter=1000, multi_class='multinomial')
model.fit(X_train, y_train)

# 在测试集上做预测
y_pred = model.predict(X_test)

# 计算混淆矩阵
cm = confusion_matrix(y_test, y_pred)

# 绘制混淆矩阵
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.arange(5))
disp.plot(cmap=plt.cm.Blues, values_format='d')
plt.title('Confusion Matrix')
plt.show()
```

绘图结果：



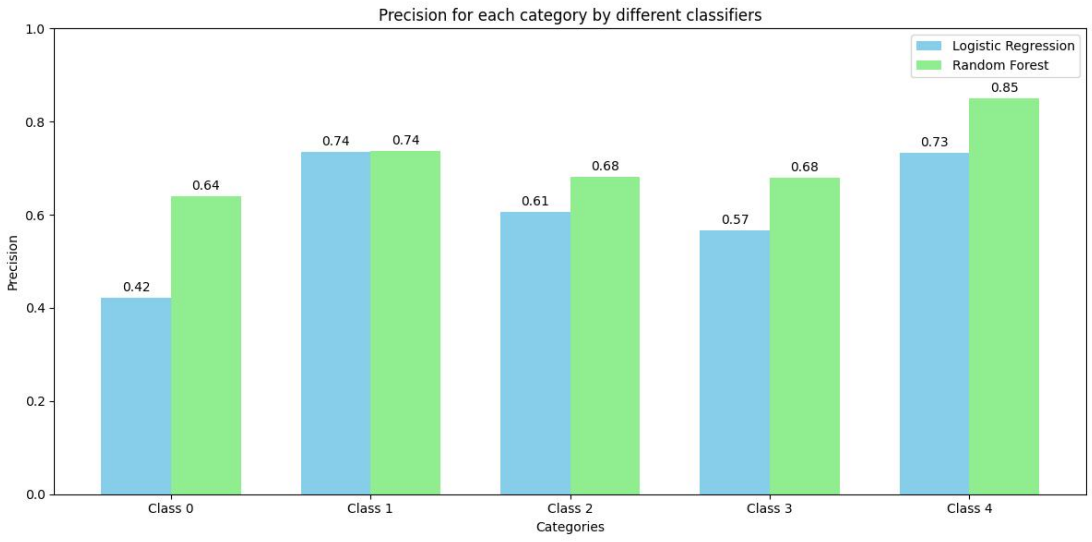
三、柱状图也是科研论文里常见的体现性能指标的图，我们通过使用两种不同的分类算法进行对比，并将它们的分类精度绘制在同一个柱状图中。我们选择逻辑回归（Logistic

Regression）和随机森林（Random Forest）作为对比的分类算法。

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# 生成一个包含 5 个类别的分类数据集
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5,
                           n_classes=5, random_state=42)

# 划分数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 创建并训练逻辑回归模型
lr_model = LogisticRegression(max_iter=1000, multi_class='multinomial')
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)

# 创建并训练随机森林模型
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

# 计算分类报告（包括每个类别的精度）
lr_report = classification_report(y_test, lr_pred, output_dict=True)
rf_report = classification_report(y_test, rf_pred, output_dict=True)

# 提取每个类别的精度
lr_accuracy_per_class = [lr_report[str(i)]['precision'] for i in range(5)]
rf_accuracy_per_class = [rf_report[str(i)]['precision'] for i in range(5)]

# 绘制柱状图
categories = [f'Class {i}' for i in range(5)]
x = np.arange(len(categories))  # 类别的标签位置
width = 0.35  # 柱状图的宽度

fig, ax = plt.subplots(figsize=(12, 6))
rects1 = ax.bar(x - width/2, lr_accuracy_per_class, width, label='Logistic Regression',
color='skyblue')
rects2 = ax.bar(x + width/2, rf_accuracy_per_class, width, label='Random Forest',
```

```
color='lightgreen')

# 添加一些文本标签
ax.set_xlabel('Categories')
ax.set_ylabel('Precision')
ax.set_title('Precision for each category by different classifiers')
ax.set_xticks(x)
ax.set_xticklabels(categories)
ax.legend()

# 添加精度标签
def add_labels(rects):
    for rect in rects:
        height = rect.get_height()
        ax.annotate(f'{height:.2f}',
                    xy=(rect.get_x() + rect.get_width() / 2, height),
                    xytext=(0, 3),    # 3 points vertical offset
                    textcoords='offset points',
                    ha='center', va='bottom')

add_labels(rects1)
add_labels(rects2)

plt.ylim(0, 1)   # 精度在 0 到 1 之间
plt.tight_layout()
plt.show()
```

绘图结果：

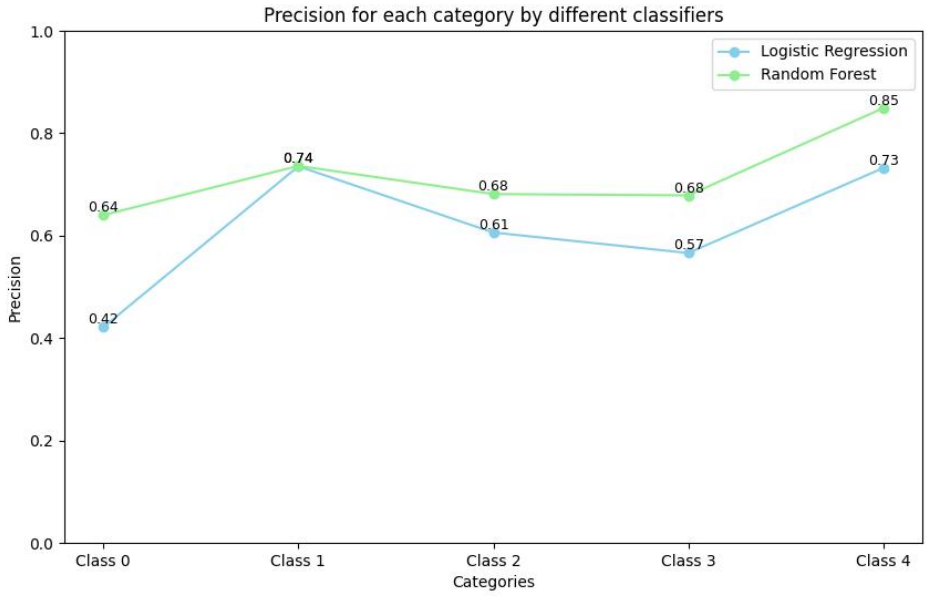


四、绘制用于体现分类进度的折线图。

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# 生成一个包含 5 个类别的分类数据集
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5,
                           n_classes=5, random_state=42)

# 划分数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 创建并训练逻辑回归模型
lr_model = LogisticRegression(max_iter=1000, multi_class='multinomial')
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)

# 创建并训练随机森林模型
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

# 计算分类报告（包括每个类别的精度）
lr_report = classification_report(y_test, lr_pred, output_dict=True)
rf_report = classification_report(y_test, rf_pred, output_dict=True)

# 提取每个类别的精度
lr_accuracy_per_class = [lr_report[str(i)]['precision'] for i in range(5)]
rf_accuracy_per_class = [rf_report[str(i)]['precision'] for i in range(5)]

# 绘制折线图
categories = [f'Class {i}' for i in range(5)]
x = np.arange(len(categories))  # 类别的标签位置

plt.figure(figsize=(10, 6))
plt.plot(x, lr_accuracy_per_class, marker='o', linestyle='-', label='Logistic Regression',
color='skyblue')
plt.plot(x, rf_accuracy_per_class, marker='o', linestyle='-', label='Random Forest',
```

```
color='lightgreen')

# 添加一些文本标签
plt.xlabel('Categories')
plt.ylabel('Precision')
plt.title('Precision for each category by different classifiers')
plt.xticks(x, categories)
plt.ylim(0, 1)  # 精度在 0 到 1 之间
plt.legend()

# 在每个数据点添加精度标签
for i in range(len(categories)):
    plt.text(x[i], lr_accuracy_per_class[i], f'{lr_accuracy_per_class[i]:.2f}', ha='center', va='bottom', fontsize=9)
    plt.text(x[i], rf_accuracy_per_class[i], f'{rf_accuracy_per_class[i]:.2f}', ha='center', va='bottom', fontsize=9)

plt.show()
```

绘图结果：



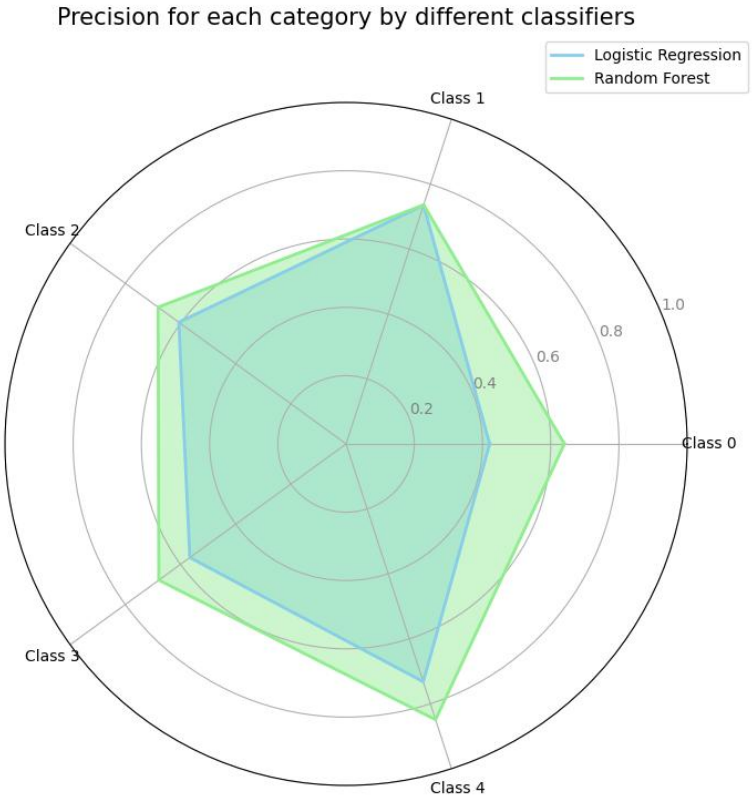
四、极坐标图（Radar Chart 或 Spider Chart）是一种在极坐标系上绘制的图表，常用于多变量数据的可视化，我们用 python 绘制一个简单的极坐标图。

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

# 生成一个包含 5 个类别的分类数据集
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5,
                           n_classes=5, random_state=42)

# 划分数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 创建并训练逻辑回归模型
lr_model = LogisticRegression(max_iter=1000, multi_class='multinomial')
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)

# 创建并训练随机森林模型
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

# 计算分类报告（包括每个类别的精度）
lr_report = classification_report(y_test, lr_pred, output_dict=True)
rf_report = classification_report(y_test, rf_pred, output_dict=True)

# 提取每个类别的精度
lr_accuracy_per_class = [lr_report[str(i)]['precision'] for i in range(5)]
rf_accuracy_per_class = [rf_report[str(i)]['precision'] for i in range(5)]

# 数据整理
labels = [f'Class {i}' for i in range(5)]
num_vars = len(labels)

# 绘制雷达图
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
angles += angles[:1]  # 完成闭合

lr_accuracy_per_class += lr_accuracy_per_class[:1]
rf_accuracy_per_class += rf_accuracy_per_class[:1]

fig, ax = plt.subplots(figsize=(8, 8), subplot_kw=dict(polar=True))

ax.fill(angles, lr_accuracy_per_class, color='skyblue', alpha=0.25)
ax.fill(angles, rf_accuracy_per_class, color='lightgreen', alpha=0.25)
```

```
ax.plot(angles, lr_accuracy_per_class, color='skyblue', linewidth=2, linestyle='solid', label='Logistic Regression')
ax.plot(angles, rf_accuracy_per_class, color='lightgreen', linewidth=2, linestyle='solid', label='Random Forest')

ax.set_yticks([0.2, 0.4, 0.6, 0.8, 1.0])
ax.set_yticklabels(['0.2', '0.4', '0.6', '0.8', '1.0'], color="grey", size=10)
ax.set_xticks(angles[:-1])
ax.set_xticklabels(labels)


plt.title('Precision for each category by different classifiers', size=15, color='black', y=1.1)
ax.legend(loc='upper right', bbox_to_anchor=(1.1, 1.1))

plt.show()
```

绘图结果：



五、绘制盒线图，盒线图（Box Plot）是数据可视化中非常常见的一种方法，用于显示数据集的分布情况。盒线图能够清晰地展示数据的五个统计量：最小值、第一四分位数（Q1）、中位数（Q2）、第三四分位数（Q3）和最大值，还可以显示异常值（outliers）。

下面是一个简单的 Python 代码示例，展示如何使用 Matplotlib 和 Seaborn 库来绘制盒线图，并比较两种分类算法在多个类别上的精度分布。

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
import pandas as pd

# 生成一个包含 5 个类别的分类数据集
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundant=5,
                           n_classes=5, random_state=42)

# 划分数据集为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 创建并训练逻辑回归模型
lr_model = LogisticRegression(max_iter=1000, multi_class='multinomial')
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)

# 创建并训练随机森林模型
rf_model = RandomForestClassifier(random_state=42)
rf_model.fit(X_train, y_train)
rf_pred = rf_model.predict(X_test)

# 计算分类报告（包括每个类别的精度）
lr_report = classification_report(y_test, lr_pred, output_dict=True)
rf_report = classification_report(y_test, rf_pred, output_dict=True)

# 提取每个类别的精度
lr_accuracy_per_class = [lr_report[str(i)]['precision'] for i in range(5)]
rf_accuracy_per_class = [rf_report[str(i)]['precision'] for i in range(5)]

# 为每个类别生成多个重复测量结果来构建足够的数据点
lr_accuracy_per_class_repeated = np.repeat(lr_accuracy_per_class, 10)
rf_accuracy_per_class_repeated = np.repeat(rf_accuracy_per_class, 10)

# 创建 DataFrame 以便于绘图
data = {
    'Class': [f'Class {i}' for i in range(5)] * 20,
```

```
    'Precision':                    np.concatenate((lr_accuracy_per_class_repeated,
rf_accuracy_per_class_repeated)),
    'Algorithm': ['Logistic Regression'] * 50 + ['Random Forest'] * 50
}
df = pd.DataFrame(data)

# 绘制盒线图
plt.figure(figsize=(10, 6))
sns.boxplot(x='Class', y='Precision', hue='Algorithm', data=df)

# 设置图表标题和标签
plt.title('Precision for each category by different classifiers')
plt.xlabel('Category')
plt.ylabel('Precision')
plt.ylim(0, 1)
plt.legend(loc='upper right')

plt.show()
```

绘图结果：