1、问题描述

利用 Logistic Regression, SVM 与 XGBoost 对所给数据进行分类，并比较三者的分类性能。同时讨论对于这个问题，为什么某些算法表现相对更好。（其中 SVM 至少选用三种不同的 Kernel Fucntions.）

2、数据集

通过所给程序生成的一个 3D 数据集。一共有 1000 个数据，被分成了两大类：C0 与 C1。请利用该数据做训练，同时利用程序新生成与训练数据同分布的 500 个数据（250 个为 C0 类，250 个数据为 C1 类）来做测试。

3、数据预处理

3.1、Python 库的选择：

numpy: 在拟合模型中，将数据转换为 numpy 数组进行高效的数学运算。

xgboost: 一个优化的分布式梯度增强库，基于 Gradient Boosting 算法。

matplotlib:用于数据的可视化分析。

sklearn:用于机器学习，训练回归模型并进行相应的评估。

3.2、数据的生成

```
# Generating 3D make-moons data
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


def make_moons_3d(n_samples=500, noise=0.1):
    t = np.linspace(0, 2 * np.pi, n_samples)
    x = 1.5 * np.cos(t)
    y = np.sin(t)
    z = np.sin(2 * t)
    X = np.vstack([np.column_stack([x, y, z]), np.column_stack([-x, y -
1, -z])])
    y = np.hstack([np.zeros(n_samples), np.ones(n_samples)])
    X += np.random.normal(scale=noise, size=X.shape)
    return X, y

X, labels = make_moons_3d(n_samples=1000, noise=0.2)
```

4、数据处理与模型建立

4.1 训练数据和测试数据

4.1.1 训练数据：

```
# Generate the data (1000 datapoints)
X, labels = make_moons_3d(n_samples=1000, noise=0.2)

X_train, X_test, y_train, y_test = train_test_split(
    X, labels, test_size=0.2, random_state=42
)
```

4.1.2 测试数据：

```
# Generate the data (500 datapoints)
X1, labels1 = make_moons_3d(n_samples=500, noise=0.2)
```

根据对散点图的分析，我们选取三种算法进行比较：

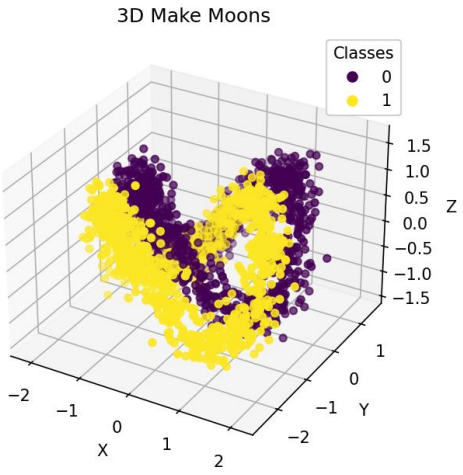

图 1 基于 3D 数据集的散点图

4.2 模型建立

4.2.1 Logistic 回归：

```
# Logistic
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

4.2.2 SVM

```
# SVM 基于不同的核函数
svm_model1 = svm.SVC(C=1.0, kernel="linear", gamma="scale")
svm_model2 = svm.SVC(C=1.0, kernel="poly", gamma="scale")
svm_model3 = svm.SVC(C=1.0, kernel="rbf", gamma="scale")

model1 = svm_model1
model1.fit(X_train, y_train)
model2 = svm_model2
```

```
model2.fit(X_train, y_train)
model3 = svm_model3
model3.fit(X_train, y_train)
```

4.2.3 XGBoost

```
# XGBoost
xgb_model = XGBClassifier()
xgb_model.fit(X_train, y_train)
```

5、模型的训练与评估

5.1、模型训练

5.1.1 Logistics：

```
y_pred = logreg.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

训练所得模型 accuracy=0.71。

5.1.2 SVM：

```
predictions1 = model1.predict(X_test)
accuracy_test1 = accuracy_score(y_test, predictions1)
print("Accuracy1:", accuracy_test1)


predictions2 = model2.predict(X_test)
accuracy_test2 = accuracy_score(y_test, predictions2)
print("Accuracy2:", accuracy_test2)

predictions3 = model3.predict(X_test)
accuracy_test3 = accuracy_score(y_test, predictions3)
print("Accuracy3:", accuracy_test3)
```

基于 linear、polynomial、rbf 为核函数，训练所得模型 accuracy 分别为 0.7175、0.8625、0.9775。

5.1.3 Xgboost：

```
xgb_pred = xgb_model.predict(X_test)
accuracy = accuracy_score(y_test, xgb_pred)
print("accuracy: ", accuracy)
```

训练所得模型 accuracy=0.97。

5.2、模型的评估

将最终的测试数据带入训练好的模型之中：

5.2.1 Logistics：

```
# Generate the data (500 datapoints)
X1, labels1 = make_moons_3d(n_samples=500, noise=0.2)
y1_pred = logreg.predict(X1)
accuracy_test = accuracy_score(labels1, y1_pred)
print("Accuracy:", accuracy_test)
```

测试所得模型 accuracy=0.71。

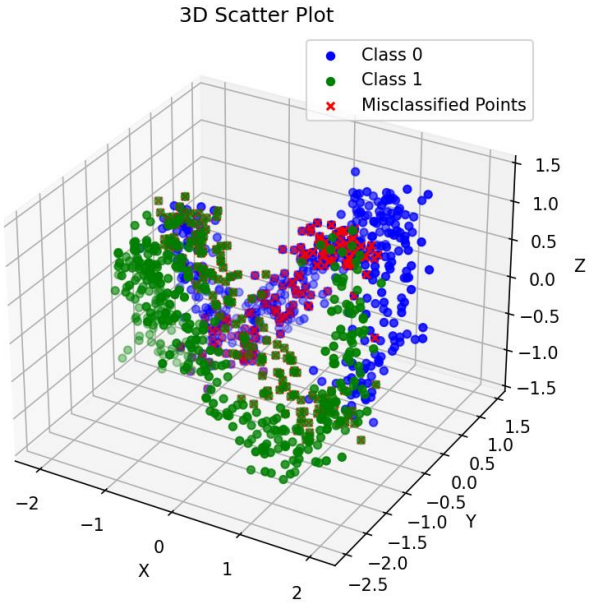

图 2 logistics 模型的测试数据散点图

5.2.2 SVM：

```
# Generate the data (500 datapoints)
X1, labels1 = make_moons_3d(n_samples=500, noise=0.2)

predictions1 = model1.predict(X1)
accuracy_test1 = accuracy_score(labels1, predictions1)
print("Accuracy1:", accuracy_test1)


predictions2 = model2.predict(X1)
accuracy_test2 = accuracy_score(labels1, predictions2)
print("Accuracy2:", accuracy_test2)

predictions3 = model3.predict(X1)
accuracy_test3 = accuracy_score(labels1, predictions3)
print("Accuracy3:", accuracy_test3)
```

基于 linear、polynomial、rbf 为核函数，测试所得模型 accuracy 分别为 0.678、0.854、0.972。

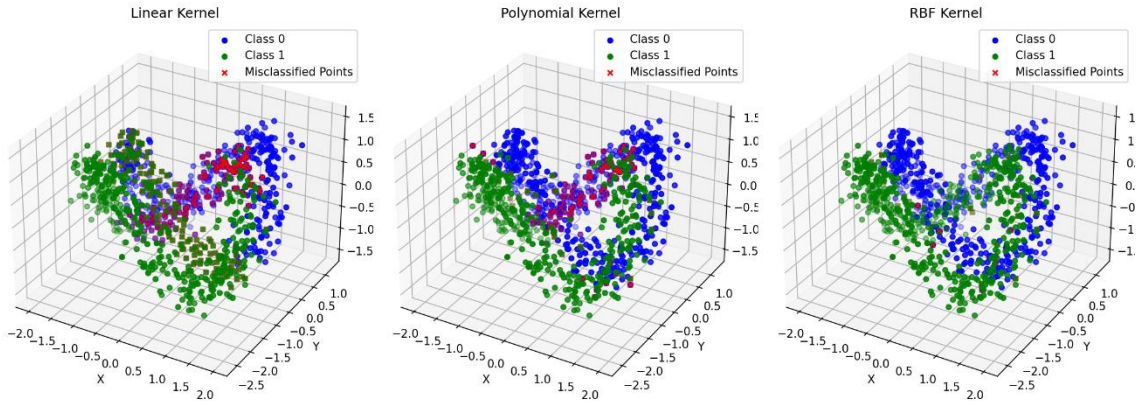

图 3 基于不同核函数的 SVM 模型测试数据散点图

5.2.3 Xgboost：

```
# Generate the data (500 datapoints)
X1, labels1 = make_moons_3d(n_samples=500, noise=0.2)

xgb_pred = xgb_model.predict(X1)
accuracy = accuracy_score(labels1, xgb_pred)
print("accuracy: ", accuracy)
```

测试所得模型 accuracy=0.98。

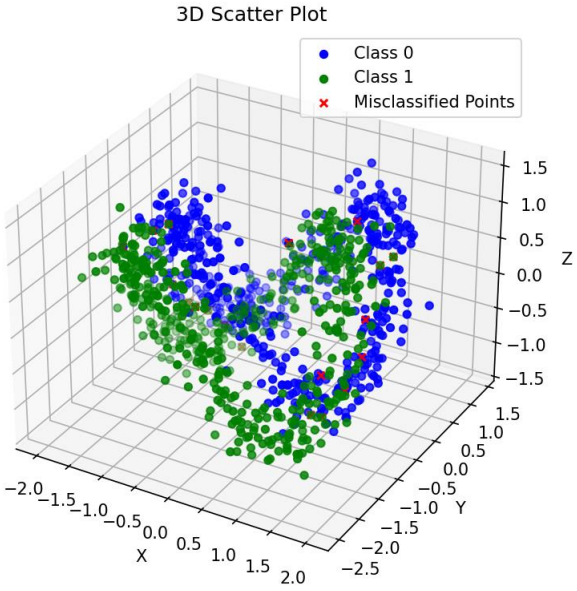

图 4 XGBoost 模型测试数据散点图

6、分析与讨论

6.1 数据集分析

本次实验的 3d 数据集从散点图中观察发现呈现明显的非线性关系。对不同算法的分类性能将产生较大影响。

6.2 分类性能比较分析

利用 Logistic Regression, SVM 与 XGBoost 对所给数据进行分类，最终得到的模型准确率分别为：0.71、0.678（Linear Kernel）、0.854（Polynomial Kernel）、0.972（RBF Kernel）、0.98。

在这个问题中，XGBoost 能够有效处理数据中的非线性关系，进行合适的参数调整，分类表现最好；

此外核函数的选择对 SVM 的分类效果影响显著。核函数的主要作用是将输入空间中的数据映射到高维空间，以更好地处理非线性问题。不同的核函数会产生不同的映射，从而影响 SVM 的分类性能。

因此，基于 RBF 核 SVM 分类表现很好，能够有效处理非线性可分问题；多项式核的 SVM 分类表现良好，能够根据数据特点，调整多项式的阶数，较好的处理数据之间的非线性关系；

而 Logistic Regression 和线性核函数的 SVM 的性能相对较差，因为本次数据并不是简单的线性关系，其无法完全捕捉数据中的非线性关系，建立的线性模型假设并不能很好的适用于本实验。

6.3 总结归纳

本次作业，利用不同的分类算法 Logistic Regression, SVM 与 XGBoost 对非线性数据的处理，可以较好的对比得出不同算法之间的优劣。XGBoost 优化梯度提升决策树算法对于本次数据，能够有效地处理之间的非线性关系和复杂的特征交互。SVM 模型需要根据不同的数据集，以选择合适的核函数进行合适的处理，其中径向核函数 RBF 能够很好的适用于本次数据，而线性核函数处理非线性问题较差。Logistic Regression 与线性核 SVM 类似，不适用于处理本次非线性问题，但在线性数据处理中仍不失为一种较好的选择。