# 针对三维moons data的分类

## 作业要求

以下的程序生成了一个3D的数据集。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。

比较利用Logistic Regression, SVM与XGBoost的分类性能。并讨论对于这个问题，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.

```
# Generating 3D make-moons data

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import FancyArrowPatch
from mpl_toolkits.mplot3d.art3d import Poly3DCollection
from matplotlib.widgets import Slider

def make_moons_3d(n_samples=500, noise=0.1):
    # Generate the original 2D make_moons data
    t = np.linspace(0, 2 * np.pi, n_samples)
    x = 1.5 * np.cos(t)
    y = np.sin(t)
    z = np.sin(2 * t)  # Adding a sinusoidal variation in the third dimension

    # Concatenating the positive and negative moons with an offset and noise
    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)])

    # Adding Gaussian noise
    X += np.random.normal(scale=noise, size=X.shape)

    return X, y
```

## 作业内容

### 模型选择

本次作业选择了Logistic Regression, SVM（Linear），SVM（poly），SVM（sigmoid），SVM（RBF）与XGBoost共**6**种模型进行训练

### 数据准备

导入所需包，对数据集进行数量和噪声的设置，1000个训练数据，500个测试数据

```
from xgboost import XGBClassifier
from sklearn.model_selection import train_test_split
from data import make_moons_3d
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

# 划分数据集
x_test, y_test = make_moons_3d(n_samples=250, noise=0.2)
x_train, y_train = make_moons_3d(n_samples=500, noise=0.2)
```

## 模型构建以及训练

对六类实验所需模型进行初始化以及训练，通过`models`和`model_names`对模型进行集成，方便后续模型训练以及结果计算和输出

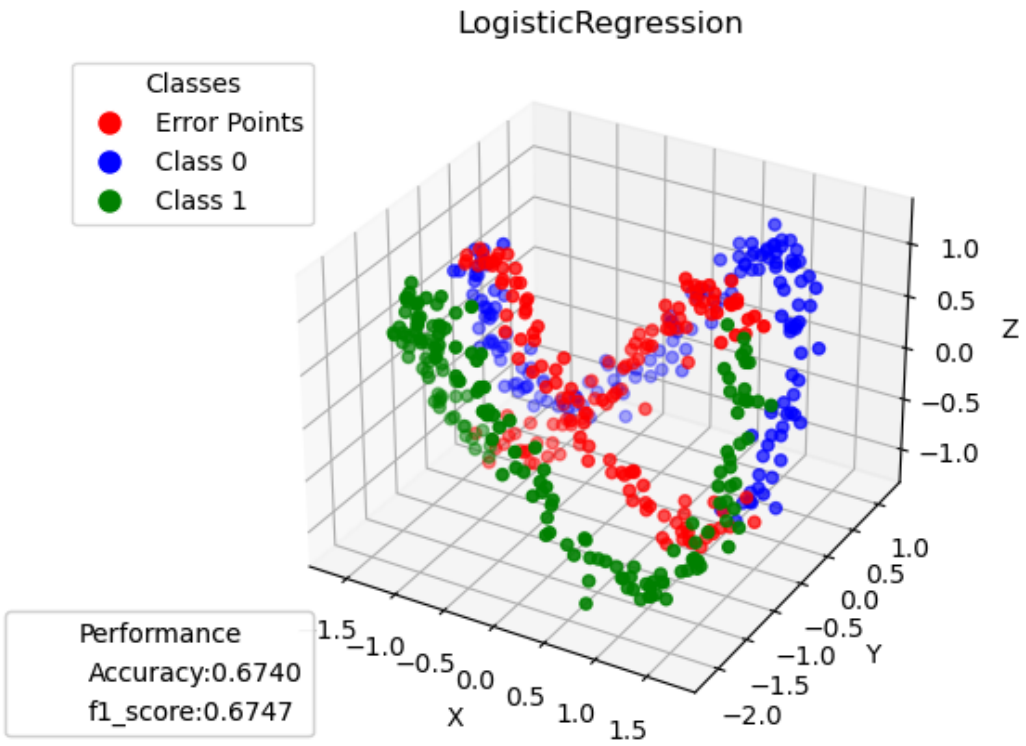
```
# 初始化模型
# 逻辑回归
lr = LogisticRegression()
# 支持向量机
svm_linear = SVC(kernel='linear')
svm_poly = SVC(kernel='poly')
svm_sigmoid = SVC(kernel='sigmoid')
svm_rbf = SVC(kernel='rbf')
# xgboost
xgb = XGBClassifier()

# 训练模型
models = [lr, svm_linear, svm_poly, svm_sigmoid, svm_rbf, xgb]
model_names = ['LogisticRegression', 'SVM (Linear)', 'SVM (poly)', 'SVM
(sigmoid)', 'SVM (RBF)', 'XGBoost']
for model, name in zip(models, model_names):
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)

    # 评估模型性能
    accuracy = accuracy_score(y_test, y_pred)
    precision = precision_score(y_test, y_pred)
    recall = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)

    print(f"{name} performance: Accuracy: {accuracy:.4f}, Precision:
{precision:.4f}, Recall: {recall:.4f}, F1-score: {f1:.4f}")
```

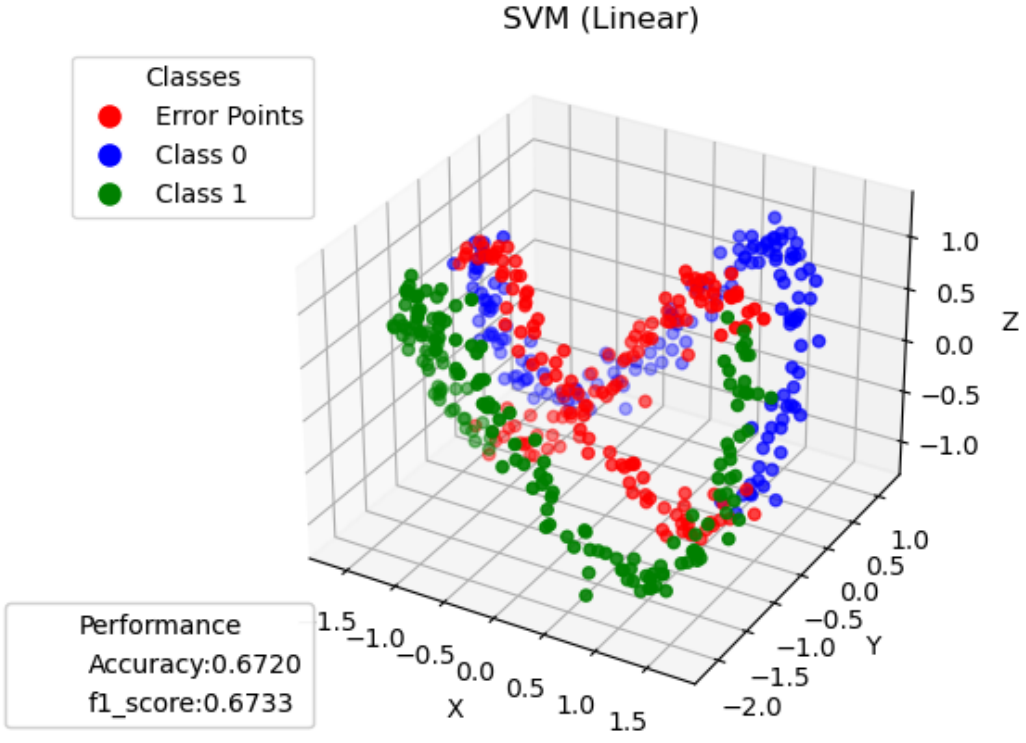
## 运行结果

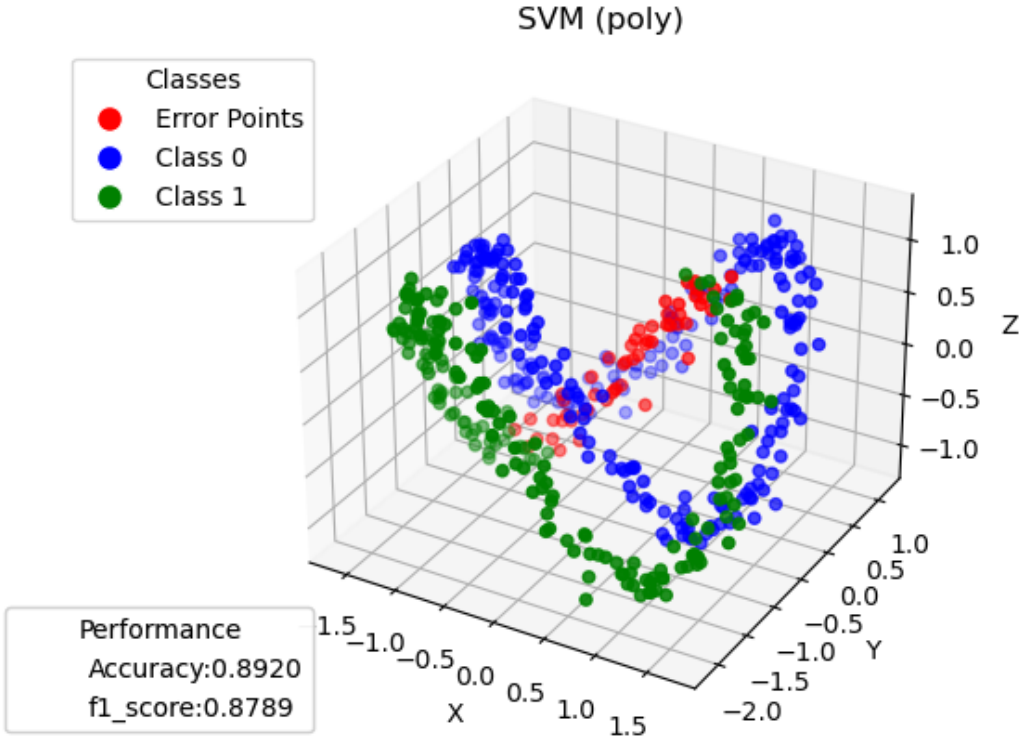
对不同模型在不同噪声情况下的模型性能进行测试，当噪声值为**0.1**时各模型性能如下：

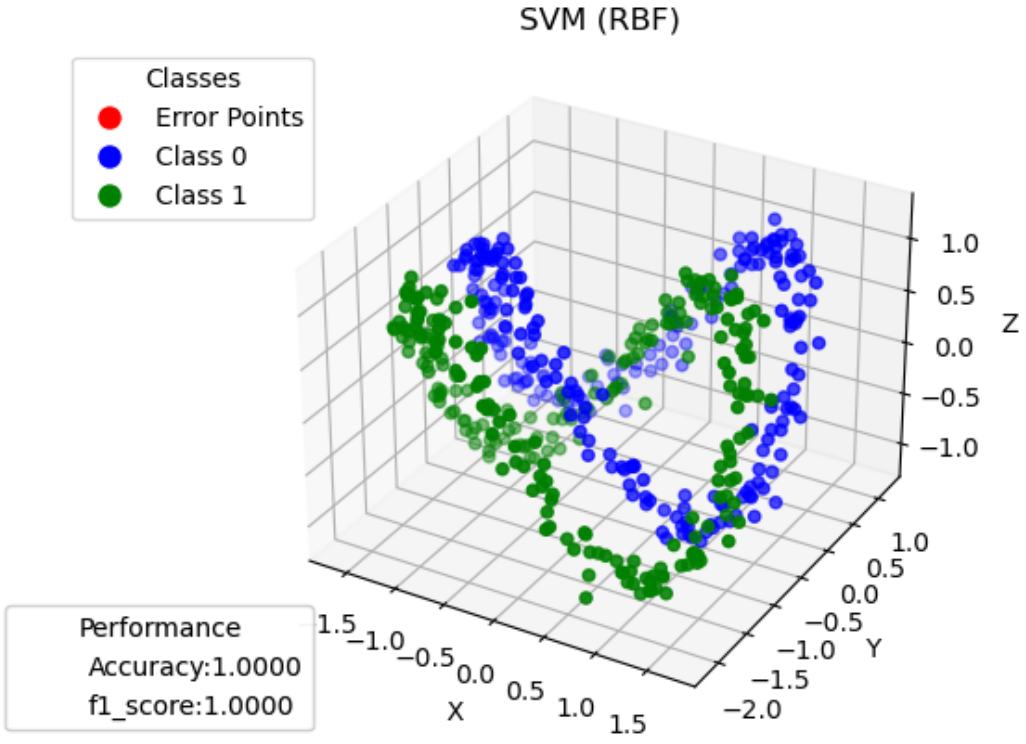
```
LogisticRegression performance: Accuracy: 0.6740, Precision: 0.6733, Recall: 0.6760, F1-score: 0.6747
SVM (Linear) performance: Accuracy: 0.6720, Precision: 0.6706, Recall: 0.6760, F1-score: 0.6733
SVM (poly) performance: Accuracy: 0.8920, Precision: 1.0000, Recall: 0.7840, F1-score: 0.8789
SVM (sigmoid) performance: Accuracy: 0.4600, Precision: 0.4573, Recall: 0.4280, F1-score: 0.4421
SVM (RBF) performance: Accuracy: 1.0000, Precision: 1.0000, Recall: 1.0000, F1-score: 1.0000
XGBoost performance: Accuracy: 0.9960, Precision: 0.9960, Recall: 0.9960, F1-score: 0.9960
```

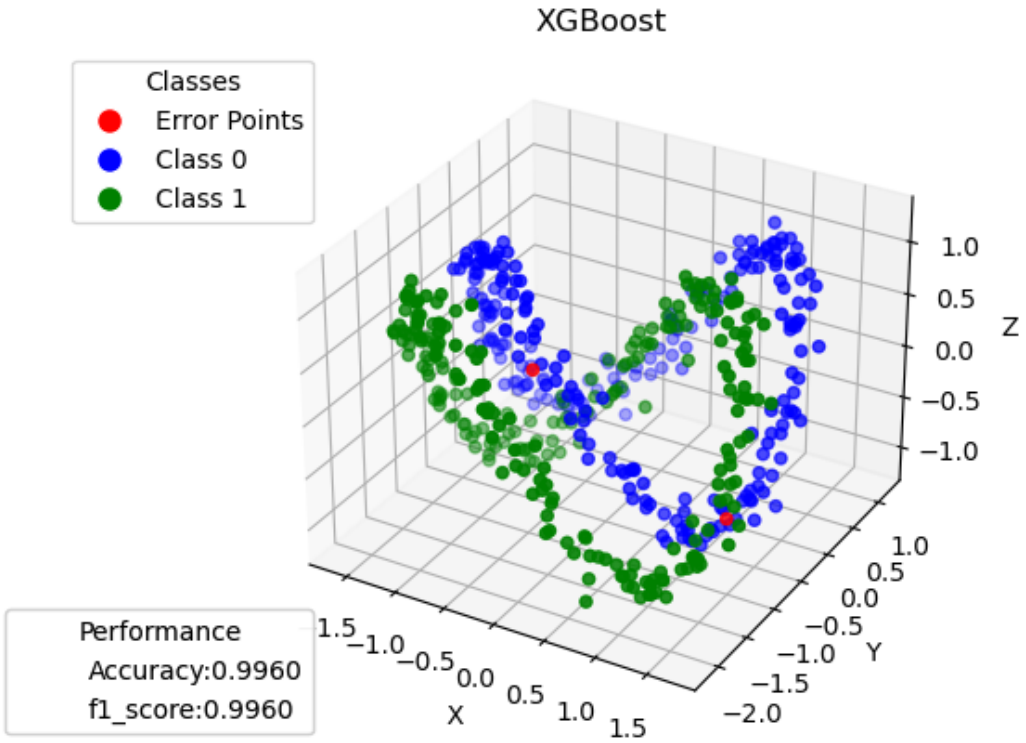
对散点图进行处理，红色为错误分类，蓝色是class = 0，绿色是class = 1，得到六种模型情况如下：



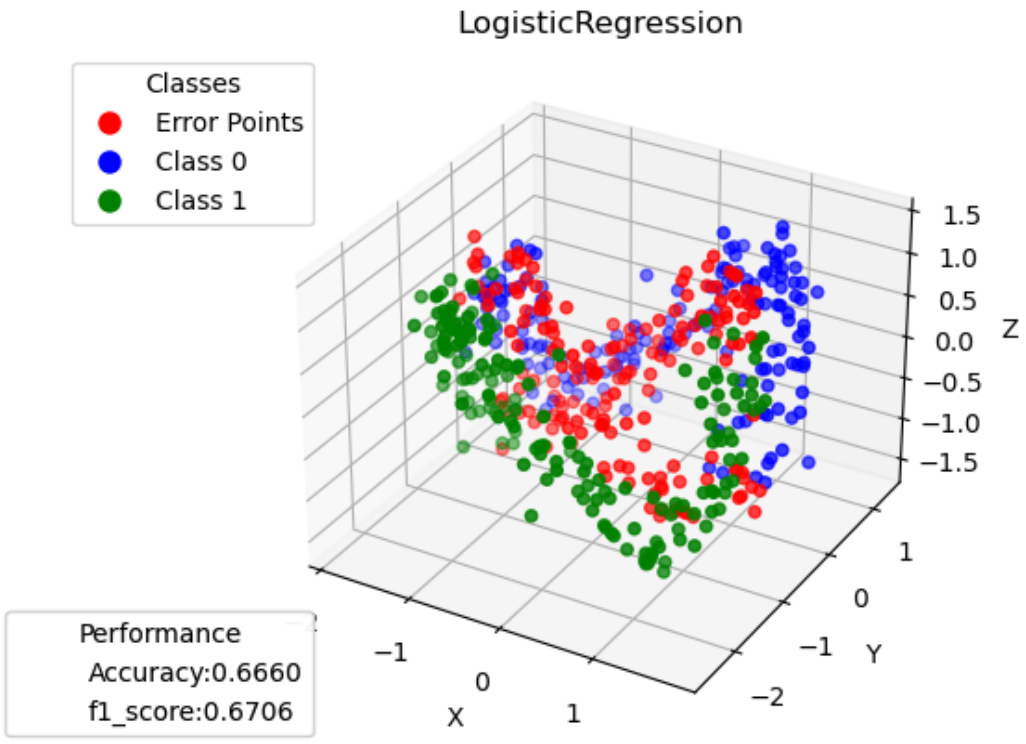

SVM (poly)
Classes
Error Points
Class 0
Class 1
Performance
Accuracy:0.8920
f1_score:0.8789
X
Y
Z

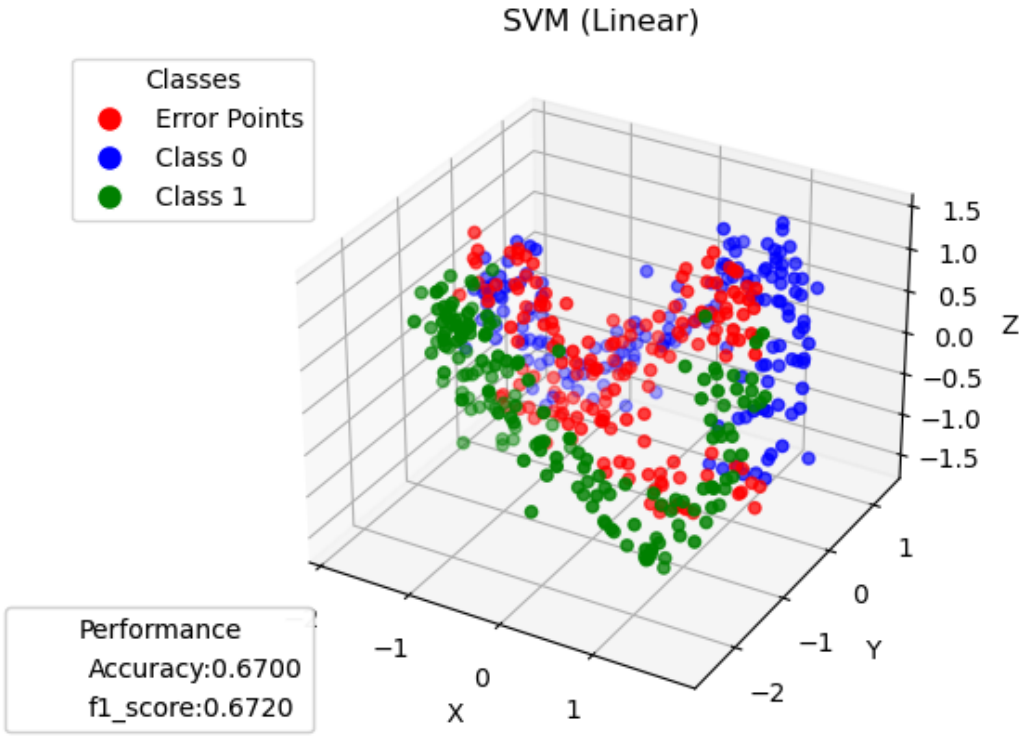

SVM (RBF)
Classes
Error Points
Class 0
Class 1
Performance
Accuracy:1.0000
f1_score:1.0000
X
Y
Z

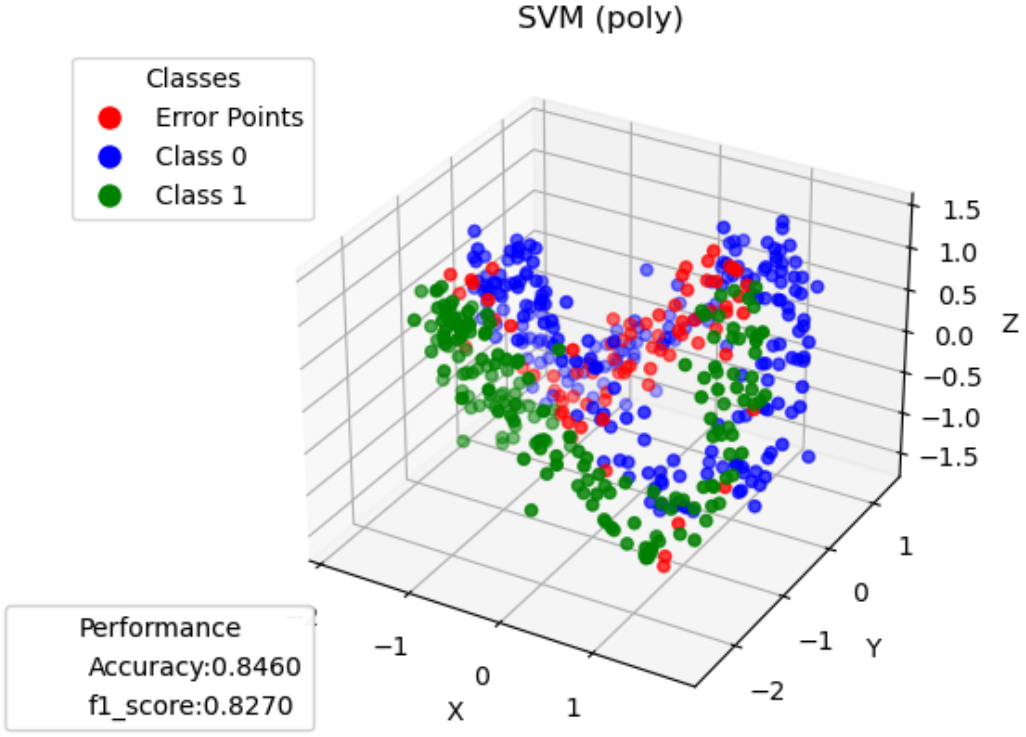


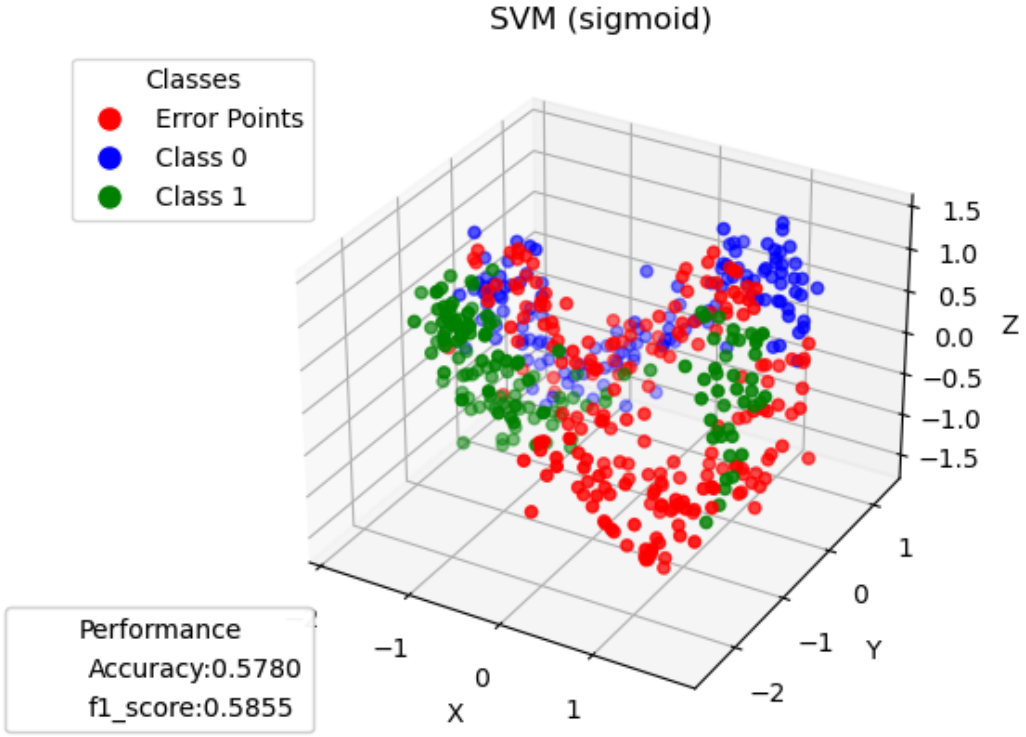

将噪声值改为**0.2**，并进行第二次测试

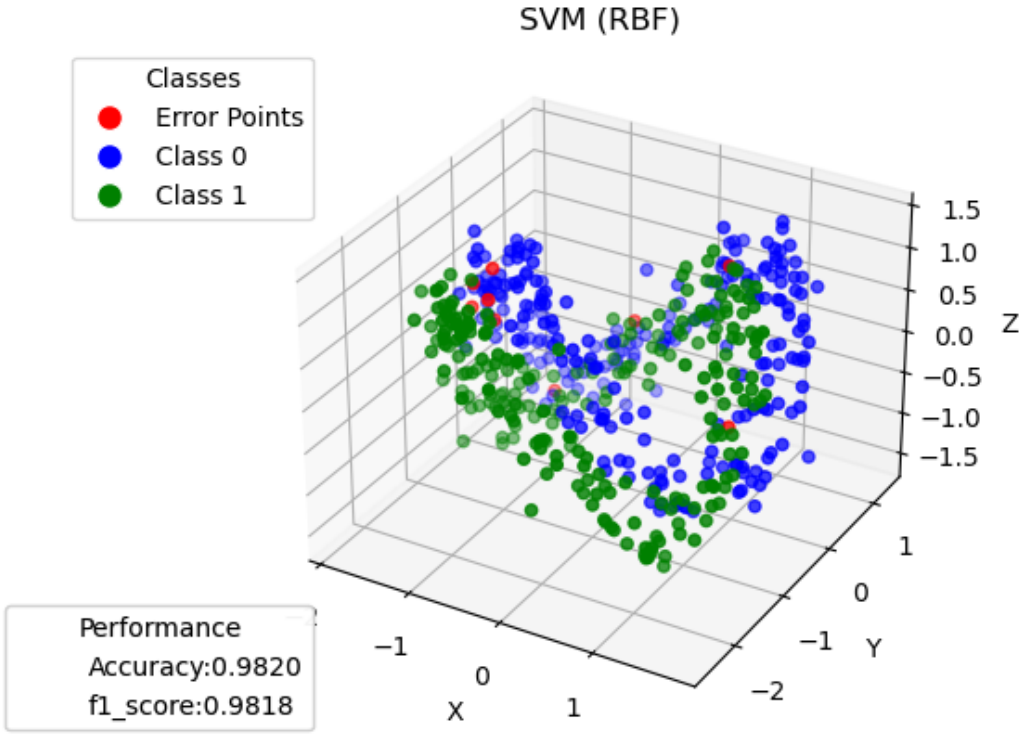
```
LogisticRegression performance: Accuracy: 0.6660, Precision: 0.6615, Recall: 0.6800, F1-score: 0.6706
SVM (Linear) performance: Accuracy: 0.6700, Precision: 0.6680, Recall: 0.6760, F1-score: 0.6720
SVM (poly) performance: Accuracy: 0.8460, Precision: 0.9436, Recall: 0.7360, F1-score: 0.8270
SVM (sigmoid) performance: Accuracy: 0.5780, Precision: 0.5753, Recall: 0.5960, F1-score: 0.5855
SVM (RBF) performance: Accuracy: 0.9820, Precision: 0.9918, Recall: 0.9720, F1-score: 0.9818
XGBoost performance: Accuracy: 0.9620, Precision: 0.9753, Recall: 0.9480, F1-score: 0.9615
```

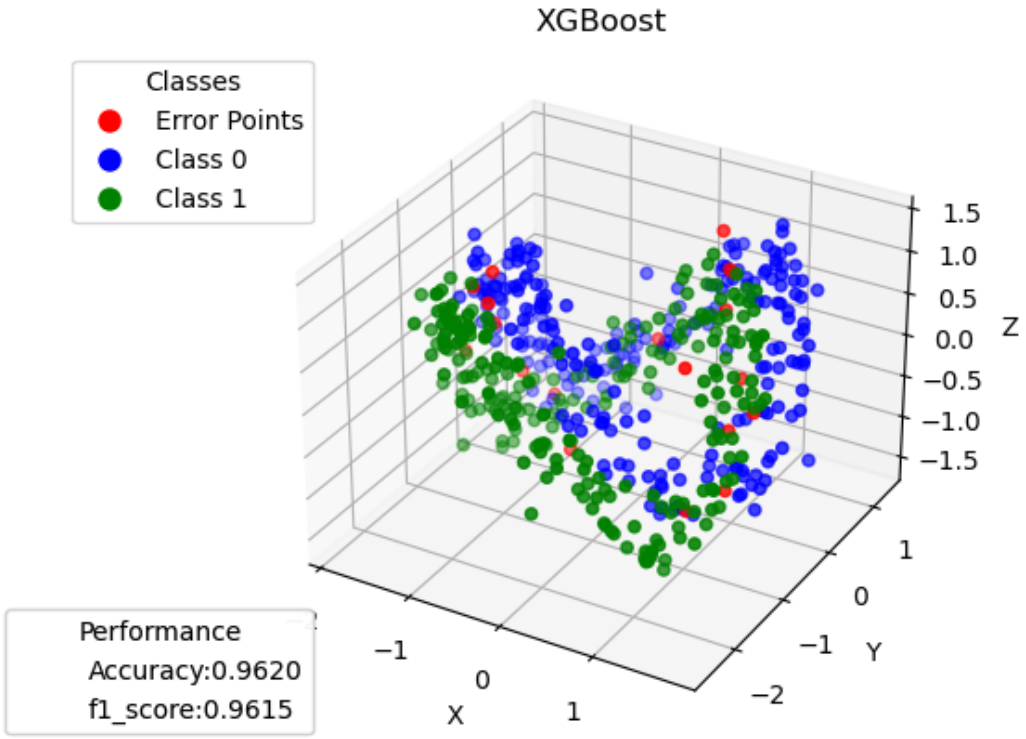



SVM (Linear)
Classes
Error Points
Class 0
Class 1
Performance
Accuracy:0.6700
f1_score:0.6720
X
Y
Z


SVM (poly)
Classes
Error Points
Class 0
Class 1
Performance
Accuracy:0.8460
f1_score:0.8270
X
Y
Z


SVM (sigmoid)
Classes
Error Points
Class 0
Class 1
Performance
Accuracy:0.5780
f1_score:0.5855
X
Y
Z


SVM (RBF)
Classes
Error Points
Class 0
Class 1
Performance
Accuracy:0.9820
f1_score:0.9818
X
Y
Z




相较于噪声低的数据，噪声高的数据对于poly、rbf和xgboost这种性能好的模型在F1-score上带来了负梯度的影响，而对于sigmoid这种不能很好完成当前分类任务的模型，更高的噪声带来更大的分布范围，使其分类性能正梯度影响，但由于其值在0.5左右徘徊，实际上并不具备分类功能。

## 模型分析

在此次分类任务上，模型性能排序为：***SVM(RBF) > XGBoost >SVM(poly)> LR =SVM(Linear) > SVM(sigmoid)***

对于各个模型的优缺点首先罗列在下面

1. **SVM with RBF Kernel**
    - 优点：RBF核函数可以处理**非线性问题**，因此在数据具有复杂结构或难以用线性方法分割的情况下表现良好。它也**不容易过拟合。**
    - 缺点：对于大规模数据集或特征维度较高的数据，训练时间可能较长。此外，对于超参数的调优要求较高，需要仔细选择合适的参数。
2. **XGBoost**
    - 优点：XGBoost也是一种强大的集成学习方法，能够处理**大规模数据集和高维特征。**它在处理结构化数据和特征工程方面表现优秀，具有较高的预测准确性。
    - 缺点：对**异常值敏感**（噪声等），需要在数据预处理阶段进行处理。此外，模型的解释性相对较弱。
3. **SVM with Polynomial Kernel**
    - 优点：多项式核函数可以处理**一定程度**的非线性关系，比线性模型更灵活。在数据具有一定非线性特征但不是非常复杂时，多项式核函数效果较好。
    - 缺点：对于高次多项式核函数，容易导致**过拟合。**同时，在选择合适的多项式次数和正则化参数方面需要谨慎。
4. **Logistic Regression (LR)**
    - 优点：LR是一种简单且高效的**线性分类器**，对于线性可分的数据集表现良好。它的模型参数易于解释，适用于需要理解特征权重的场景。

- 缺点：对于**非线性问题表现较弱**，无法处理复杂的数据结构。对**异常值敏感**，需要进行数据预处理和异常值处理。

5. **SVM with Linear Kernel**
   - 优点：线性核函数适用于**线性可分或近似线性可分**的数据集，训练速度较快且对于高维稀疏数据表现良好。
   - 缺点：在数据具有复杂非线性结构时表现较差，无法处理**非线性**问题。对于特征维度较高的数据集，模型的复杂度可能较高。
6. **SVM with Sigmoid Kernel**
   - 优点：Sigmoid核函数可以处理部分非线性问题，并且相对于其他核函数，计算复杂度较低。
   - 缺点：在实际应用中，Sigmoid核函数的**效果常常不如**其他核函数，特别是对于复杂数据结构的处理能力较弱。需要谨慎调整参数以避免过拟合或欠拟合。

在此次任务中，数据具有复杂的非线性结构，使用SVM的RBF核函数或者XGBoost这样的集成学习方法效果最好。多项式核SVM适用于一定程度的非线性问题，所以效果相较前两者略差。而LR或者线性核SVM对于简单的线性问题是很好的选择，但在复杂非线性问题上效果并不好。而Sigmoid核SVM在实际应用中使用较少，通常效果不如其他核函数。

分析的结果和实验得出的结果相同，理论与实践相符，是个成功的实验！