**Analysis of the Impact of Noisy Datasets on Different Models with Code Examples**

1. **How do you think increased noise in the dataset will impact model performance? Which model do you expect to handle noise best, and why?**

* Increased noise in a dataset generally reduces the performance of models by making the decision boundary less clear. Models like logistic regression, which fit a linear decision boundary, may be more resilient to noise because they don’t try to perfectly fit the training data. On the other hand, models like decision trees can overfit noisy data, leading to poor generalization on unseen data.

**Code Example**: Add noise to a synthetic dataset and fit both logistic regression and decision tree models to visualize the impact.

from sklearn.datasets import make\_moons

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

import matplotlib.pyplot as plt

import numpy as np

# Generate noisy data

X, y = make\_moons(n\_samples=250, noise=0.3, random\_state=42)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Fit logistic regression and decision tree

log\_reg = LogisticRegression()

tree = DecisionTreeClassifier()

log\_reg.fit(X\_train, y\_train)

tree.fit(X\_train, y\_train)

# Plot decision surfaces

def plot\_decision\_surface(X, y, model, title):

h = 0.02

x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max() + 1

y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.8, cmap=plt.cm.Paired)

plt.scatter(X[:, 0], X[:, 1], c=y, edgecolor='k', cmap=plt.cm.Paired)

plt.title(title)

plt.show()

plot\_decision\_surface(X\_train, y\_train, log\_reg, "Logistic Regression with Noise")

plot\_decision\_surface(X\_train, y\_train, tree, "Decision Tree with Noise")

**Explanation**: The plots show how logistic regression maintains a linear decision boundary despite noise, while the decision tree may show more erratic, step-like boundaries, indicating overfitting.

1. **How might the logistic regression model be affected by noisy data?**

* Logistic regression models are less likely to overfit to noise because they seek a global linear solution rather than fitting to local variations. However, excessive noise can reduce overall accuracy as it blurs the true boundary.

**Code Insight**

train\_accuracy = log\_reg.score(X\_train, y\_train)

test\_accuracy = log\_reg.score(X\_test, y\_test)

print(f"Logistic Regression Train Accuracy: {train\_accuracy:.2f}")

print(f"Logistic Regression Test Accuracy: {test\_accuracy:.2f}")

**Explanation**: The relatively stable accuracy across train and test sets reflects the robustness of logistic regression against noise.

1. **How do you think noise will influence the decision tree model?**

* Decision trees are prone to overfitting noise because they split data into increasingly smaller regions, capturing even random variations. This can lead to highly irregular and complex boundaries.

**Code Insight**

train\_accuracy\_tree = tree.score(X\_train, y\_train)

test\_accuracy\_tree = tree.score(X\_test, y\_test)

print(f"Decision Tree Train Accuracy: {train\_accuracy\_tree:.2f}")

print(f"Decision Tree Test Accuracy: {test\_accuracy\_tree:.2f}")

 **Explanation**: If the training accuracy is significantly higher than the testing accuracy, it indicates overfitting due to noise. The decision boundary plot will reveal intricate, step-like patterns due to noise fitting.

1. **How will the SVM's decision boundary be influenced by noisy data?**

* SVMs are sensitive to points near the margin as they rely on support vectors to form the boundary. Noise can introduce misleading support vectors, causing the SVM to create a more complex or tighter margin that may overfit the data.

**Code Example**:

from sklearn.svm import SVC

# Fit SVM model

svm = SVC(kernel='rbf', C=1)

svm.fit(X\_train, y\_train)

plot\_decision\_surface(X\_train, y\_train, svm, "SVM with Noise")

**Explanation**: The decision boundary may appear wavy or more adaptive, showing overfitting if noisy points influence the margin significantly. The SVM's inherent regularization can mitigate some noise impact, but excessive noise can still alter the boundary and affect support vector selection.

1. **What strategies can be used to mitigate the effects of noise on these models?**

* Logistic Regression: Regularization techniques (e.g., L1 or L2 regularization) help in reducing the impact of noise by penalizing overly complex models.
* Decision Trees: Pruning and setting a maximum depth can prevent trees from fitting to noise.
* SVM: Adjusting the regularization parameter C and using kernel functions can help manage the influence of noise. A higher C allows the model to fit the training set closely, while a lower C introduces more regularization to mitigate noise.

**Code Examples**:

# Logistic Regression with Regularization

log\_reg\_regularized = LogisticRegression(C=0.1) # Stronger regularization

log\_reg\_regularized.fit(X\_train, y\_train)

# Decision Tree with Pruning

tree\_pruned = DecisionTreeClassifier(max\_depth=3)

tree\_pruned.fit(X\_train, y\_train)

# SVM with Adjusted C

svm\_regularized = SVC(kernel='rbf', C=0.5)

svm\_regularized.fit(X\_train, y\_train)

# Visualize all strategies

plot\_decision\_surface(X\_train, y\_train, log\_reg\_regularized, "Regularized Logistic Regression")

plot\_decision\_surface(X\_train, y\_train, tree\_pruned, "Pruned Decision Tree")

plot\_decision\_surface(X\_train, y\_train, svm\_regularized, "Regularized SVM")

**Explanation**: These strategies show practical ways to reduce overfitting due to noise and ensure the models generalize better to unseen data. The plots for regularized models will reveal smoother and less complex boundaries.