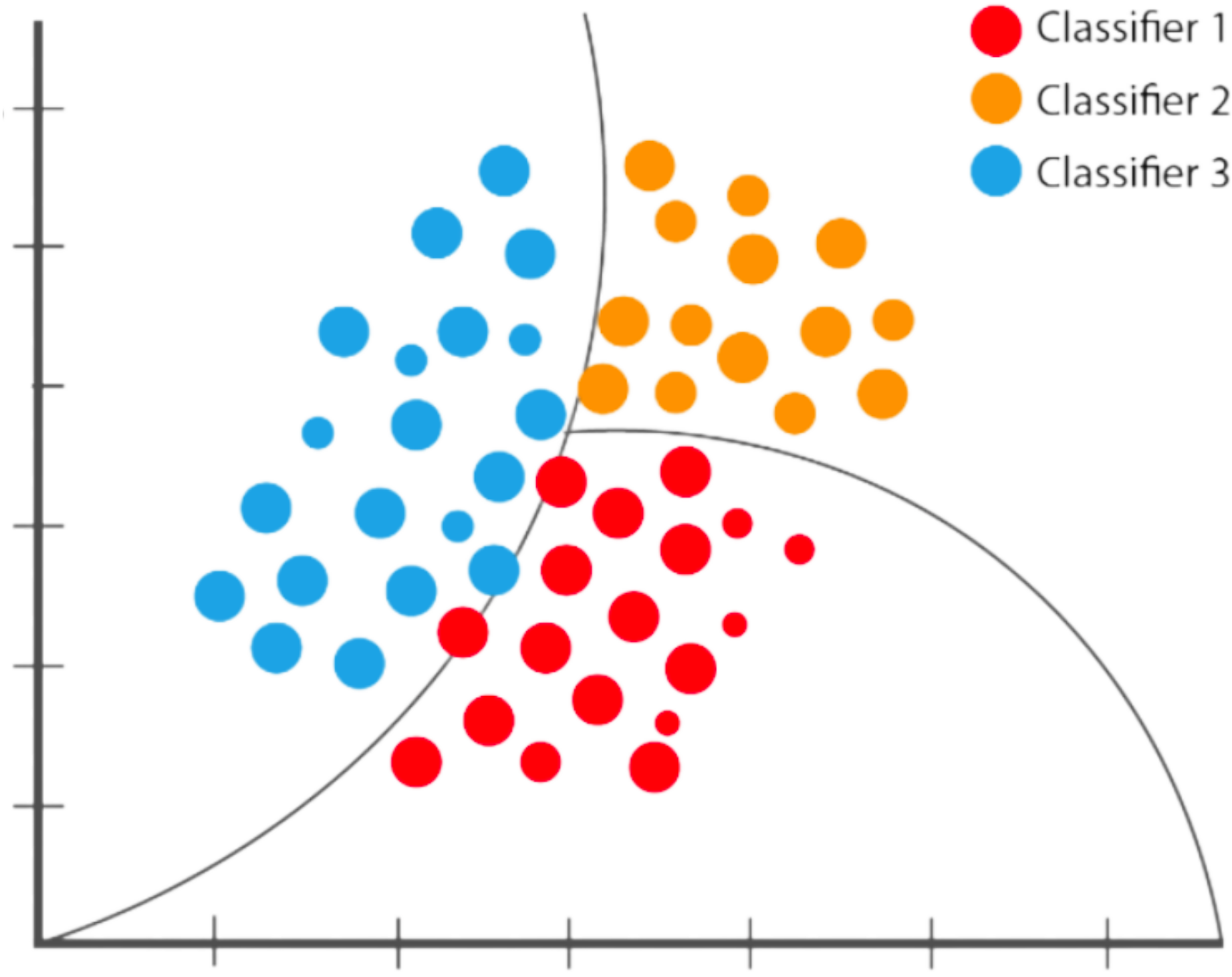
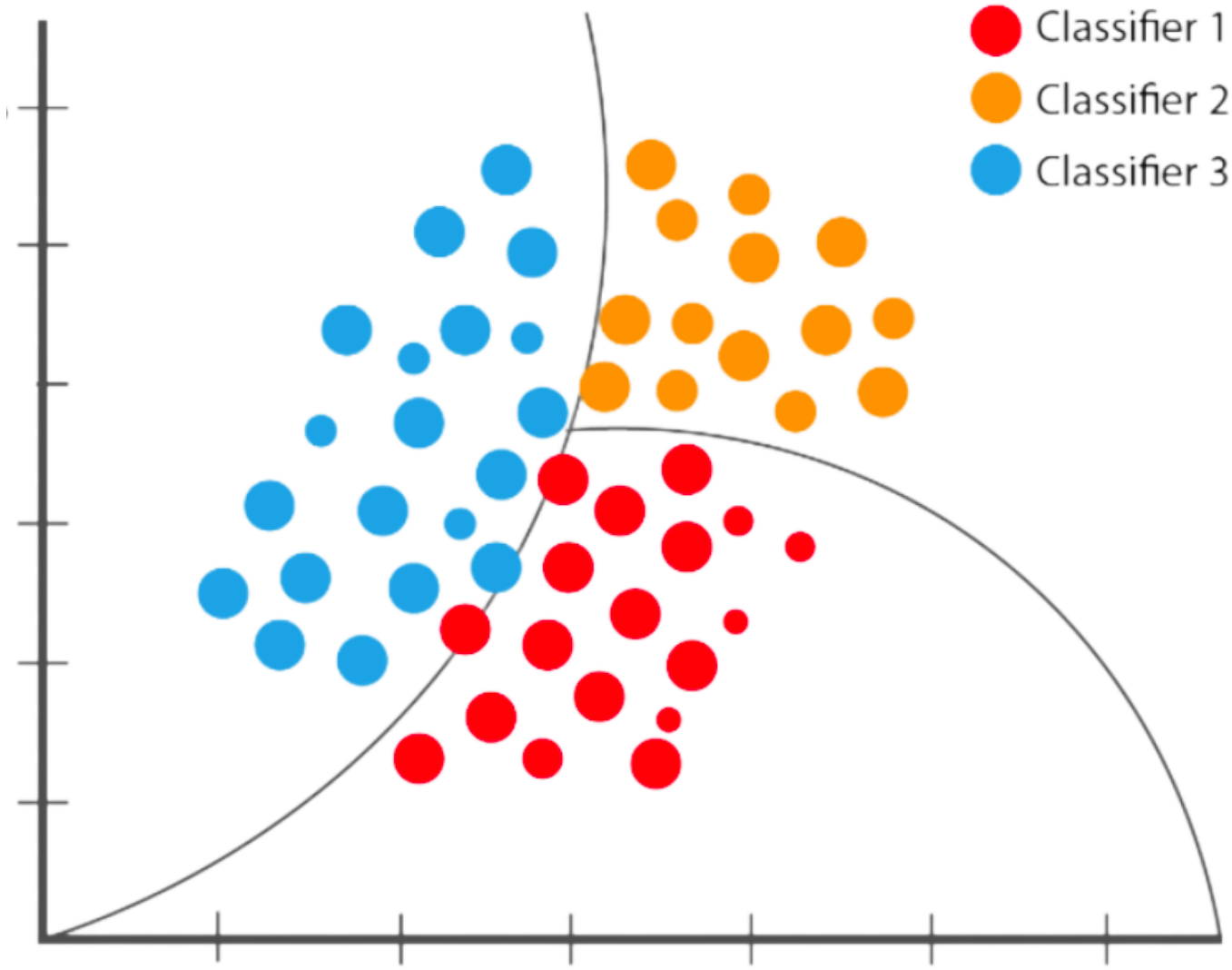
Analytics Vidhya

Naive Bayes and it's Hyperparameter



What is Naive Bayes?

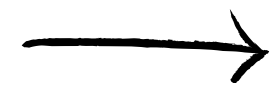
Naive Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem. It is termed 'naive' because it makes the assumption that the presence of a particular feature in a class is unrelated to the presence of any other feature, even if these features depend on each other.



Analytics Vidhya

Why to use Naive Bayes?

- **Efficiency**: Naive Bayes is fast and can be used for real-time predictions.
- **Simple**: It's easy to understand and implement.
- **Highly Scalable**: It scales linearly with the number of predictors and data points.
- **Good for Multi-class Predictions**: Can predict the probability of multiple classes of the target variable.



Advantages of Naive Bayes

- **Minimal Training Data**: Requires a small amount of training data to estimate parameters.

- **Handle Missing Data**: Can handle missing data and is not sensitive to irrelevant features.

- **Multinomial Models**: Suitable for discrete data.

Disdvantages of Naive Bayes

- **Naive Assumption**: The assumption that features are independent is rarely true in real-world scenarios.

- **Data Sufficiency**: For estimating probabilities, Naive Bayes requires a good amount of data.

- **Continuous Data**: Not ideal for continuous data.

Hyperparameters in Naive Bayes

- Different types of Naive Bayes classifiers have different hyperparameters. For Gaussian Naive Bayes, there aren't many hyperparameters to tune. However, for Multinomial and Bernoulli Naive Bayes, you might consider:

- **Alpha (Laplace/Lidstone Smoothing Parameter)**: Helps with cases where a given class and feature value did not occur together in the training data.

Analytics Vidhya

Python Code to Implement Naive Bayes

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.naive_bayes import MultinomialNB

# Load data
data = load_iris()
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target,
                                                    test_size=0.2, random_state=42)

 # Define Naive Bayes with GridSearchCV for hyperparameter tuning
param_grid = {'alpha': [0.1, 0.5, 1.0, 2.0, 10.0]}
naive_bayes_classifier = MultinomialNB()
clf = GridSearchCV(naive_bayes_classifier, param_grid, cv=5)

# Fit and predict
clf.fit(X_train, y_train)
predictions = clf.predict(X_test)

# Print best hyperparameters
print(f"Best Hyperparameters: {clf.best_params_}")
```