1. What is a Support Vector Machine (SVM)?

```
# 1. What is a Support Vector Machine (SVM)?
# SVM is a supervised learning algorithm used for classification and regression.

# It finds the best hyperplane that separates data into different classes with maximum margin.
# Works well in high-dimensional spaces and is effective for both linear and non-linear data.
```

2. What is the difference between Hard Margin and Soft Margin SVM?

```
# Hard Margin SVM                               Soft Margin SVM
# Requires perfect separation of classes.         Allows some misclassification.
# Works only when data is linearly separable.     Works for overlapping classes.
# Risk of overfitting.                          More flexible, reduces overfitting.
```

3. What is the mathematical intuition behind SVM?

```
# SVM finds the optimal hyperplane that maximizes the margin between classes.
# The equation of a hyperplane in SVM:
# w·x+b=0
# where w is the weight vector and
# b is the bias.

# The margin is defined as:
# 2 / ||w||

# which we maximize while ensuring correct classification.
```

4. What is the role of Lagrange Multipliers in SVM?

Lagrange multipliers help optimize the margin by converting the problem into a constrained optimization:

$$\begin{aligned} \max_\alpha \quad & W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \langle x^{(i)}, x^{(j)} \rangle. \\ \text{s.t.} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, m \\ & \sum_{i=1}^{m} \alpha_i y^{(i)} = 0, \end{aligned}$$

5. What are Support Vectors in SVM?

```
# Support Vectors are the data points closest to the decision boundary.
# They define the margin and influence the position of the hyperplane
```

6. What is a Support Vector Classifier (SVC)?

```
# SVC is an SVM used for classification tasks.
# It separates classes using a hyperplane and supports linear and non-linear classification with kernels.
```

7. What is a Support Vector Regressor (SVR)?

```
# SVR is an SVM used for regression tasks.
# Instead of finding a separating hyperplane,
# it finds a tube within which most data points fit, minimizing errors.
```

8. What is the Kernel Trick in SVM?

```
# The Kernel Trick allows SVM to work with non-linear data by
# transforming it into a higher-dimensional space where it becomes linearly separable.
```

```
# Example: Polynomial, RBF, and Sigmoid kernels.
```

9. Compare Linear Kernel, Polynomial Kernel, and RBF Kernel:

Kernel	Equation	When to Use
Linear	$K(x_i, x_j) = x_i \cdot x_j$	When data is linearly separable.
Polynomial	$K(x_i, x_j) = (x_i \cdot x_j + c)^d$	For curved decision boundaries.
RBF (Radial Basis Function)	(K(x_i, x_j) = \exp(-\gamma	

10. What is the effect of the C parameter in SVM?

```
# C (Regularization Parameter) controls trade-off between margin width and misclassification.
# Small C → Large margin, more misclassifications (better generalization).
# Large C → Smaller margin, fewer misclassifications (higher accuracy but risk of overfitting).
```

11. What is the role of the Gamma parameter in RBF Kernel SVM?

```
# Gamma (γ) controls how far the influence of a data point reaches.
# Low γ → Model is smoother, generalizes better.
# High γ → Model is more complex, risk of overfitting.
```

12. What is the Naïve Bayes classifier, and why is it called "Naïve"?

```
# # Naïve Bayes is a probabilistic classifier based on Bayes' Theorem.
# # It is "naïve" because it assumes all features are independent,
# which is often unrealistic but works well in practice.
```

13. What is Bayes’ Theorem?

Bayes' Theorem calculates the probability of a class given certain features:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

Where:

- $P(A|B)$ = Posterior probability (probability of class given features).
- $P(B|A)$ = Likelihood (probability of features given class).
- $P(A)$ = Prior probability of the class.
- $P(B)$ = Evidence (overall probability of features).

14. Explain the differences between Gaussian Naïve Bayes, Multinomial Naïve Bayes, and Bernoulli Naïve Bayes:

```
# Type                             Used for                                                   Feature Assumption
# Gaussian Naïve Bayes         Continuous data (e.g., heights, weights).                  Assumes data follows a normal distrib
# Multinomial Naïve Bayes       Text classification (e.g., spam detection).                Counts word occurrences (bag-of-word
# Bernoulli Naïve Bayes       Binary data (e.g., presence/absence of words).             Assumes binary features (0/1).
```

15. When should you use Gaussian Naïve Bayes over other variants?

```
# Use Gaussian Naïve Bayes when the features are continuous and follow a normal distribution,
# such as medical data (e.g., blood pressure, cholesterol levels).
```

16. What are the key assumptions made by Naïve Bayes?

```
# Feature Independence: All features are independent (not true in real-world data).
# Conditional Independence: Features contribute independently to the classification.
# Equal Feature Importance: All features are equally important (which may not always be true).
```

17. What are the advantages and disadvantages of Naïve Bayes?

```
# Advantages:

# Fast and efficient for large datasets.
# Works well with text classification.
# Performs well on small datasets.


# Disadvantages:

# Assumes independence of features, which is often false.
# Sensitive to feature correlation.
# Performs poorly when feature distributions are complex.
```

18. Why is Naïve Bayes a good choice for text classification?

```
# Works well with sparse data (e.g., text features).
# Handles large vocabulary sizes efficiently.
# Fast training and prediction time.
```

19. Compare SVM and Naïve Bayes for classification tasks:

```
# SVM                                         Naïve Bayes
# Works well with small datasets        Works well with large datasets
# Slower training on large data       Faster training and inference
# Needs feature scaling                 Works without scaling
# Better for high-dimensional data      Good for text classification
```

20. How does Laplace Smoothing help in Naïve Bayes?

Laplace Smoothing **prevents zero probabilities** by adding a small constant (α) to the count of every class-feature combination:

$$P(w|C) = \frac{count(w, C) + \alpha}{count(C) + \alpha \cdot N}$$

where N is the number of unique words.

21. Train an SVM Classifier on the Iris dataset and evaluate accuracy

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM Classifier
model = SVC(kernel='linear', random_state=42)
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

```
print("SVM Classifier Accuracy (Iris dataset):", accuracy)
```

```
SVM Classifier Accuracy (Iris dataset): 1.0
```

22. Train two SVM classifiers with Linear and RBF kernels on the Wine dataset and compare accuracy

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load dataset
data = load_wine()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM Classifier with Linear Kernel
svm_linear = SVC(kernel='linear', random_state=42)
svm_linear.fit(X_train, y_train)
acc_linear = accuracy_score(y_test, svm_linear.predict(X_test))

# Train SVM Classifier with RBF Kernel
svm_rbf = SVC(kernel='rbf', random_state=42)
svm_rbf.fit(X_train, y_train)
acc_rbf = accuracy_score(y_test, svm_rbf.predict(X_test))

print("SVM Accuracy with Linear Kernel:", acc_linear)
print("SVM Accuracy with RBF Kernel:", acc_rbf)
```

```
SVM Accuracy with Linear Kernel: 1.0
SVM Accuracy with RBF Kernel: 0.8055555555555556
```

23. Train an SVM Regressor (SVR) on a housing dataset and evaluate using Mean Squared Error (MSE)

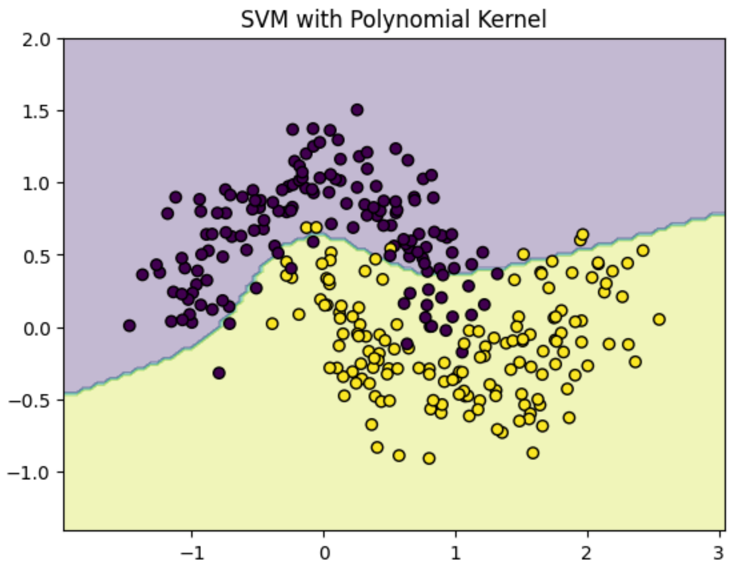
```
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error

# Load dataset
data = fetch_california_housing()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVR model
model = SVR(kernel='rbf')
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print("SVR Model MSE:", mse)
```

```
SVR Model MSE: 1.3320115421348744
```

24. Train an SVM Classifier with a Polynomial Kernel and visualize the decision boundary

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split

# Generate dataset
X, y = make_moons(n_samples=300, noise=0.2, random_state=42)
```

```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM with Polynomial Kernel
model = SVC(kernel='poly', degree=3, random_state=42)
model.fit(X_train, y_train)

# Visualize decision boundary
xx, yy = np.meshgrid(np.linspace(X[:,0].min()-0.5, X[:,0].max()+0.5, 100),
                     np.linspace(X[:,1].min()-0.5, X[:,1].max()+0.5, 100))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.3)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')
plt.title("SVM with Polynomial Kernel")
plt.show()
```



25. Train a Gaussian Naïve Bayes classifier on the Breast Cancer dataset and evaluate accuracy

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Gaussian Naïve Bayes Classifier
model = GaussianNB()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Gaussian Naïve Bayes Accuracy (Breast Cancer dataset):", accuracy)
```

```
Gaussian Naïve Bayes Accuracy (Breast Cancer dataset): 0.9736842105263158
```

26. Train a Multinomial Naïve Bayes classifier for text classification using the 20 Newsgroups dataset

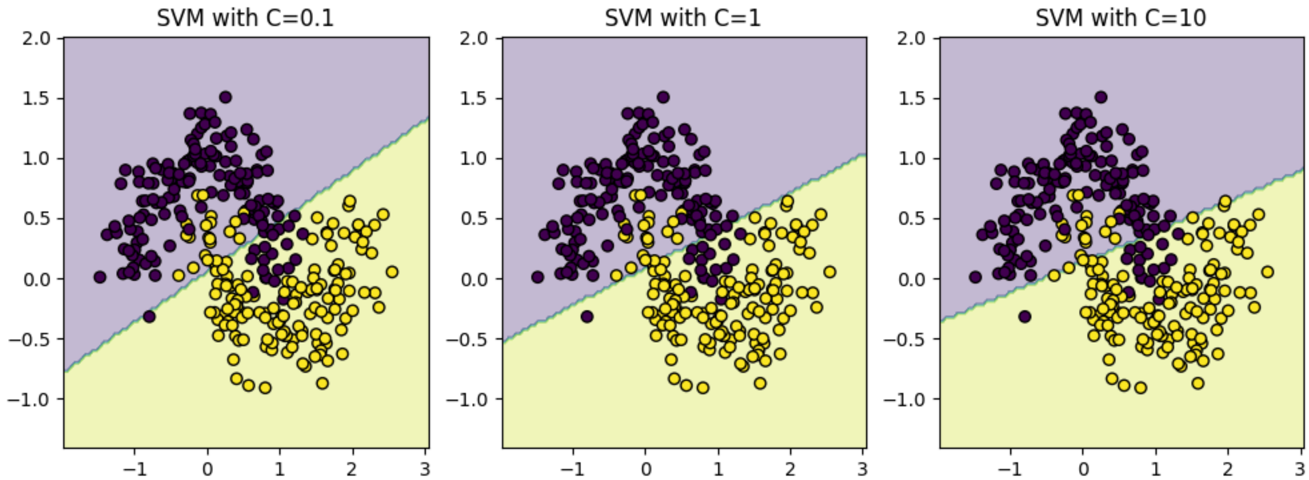
```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load dataset
categories = ['sci.space', 'talk.politics.mideast', 'comp.graphics', 'rec.sport.baseball']
data = fetch_20newsgroups(subset='train', categories=categories, remove=('headers', 'footers', 'quotes'))

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

# Build a text classification pipeline
model = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])

# Train the model
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Multinomial Naïve Bayes Accuracy (20 Newsgroups dataset):", accuracy)
```

```
Multinomial Naïve Bayes Accuracy (20 Newsgroups dataset): 0.8867521367521367
```

27. Train an SVM Classifier with different C values and compare decision boundaries visually

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split

# Generate dataset
X, y = make_moons(n_samples=300, noise=0.2, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM Classifiers with different C values
C_values = [0.1, 1, 10]
plt.figure(figsize=(12, 4))

for i, C in enumerate(C_values):
    model = SVC(kernel='linear', C=C, random_state=42)
    model.fit(X_train, y_train)

    # Plot decision boundary
    xx, yy = np.meshgrid(np.linspace(X[:,0].min()-0.5, X[:,0].max()+0.5, 100),
                         np.linspace(X[:,1].min()-0.5, X[:,1].max()+0.5, 100))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    plt.subplot(1, 3, i+1)
    plt.contourf(xx, yy, Z, alpha=0.3)
    plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k')
    plt.title(f"SVM with C={C}")

plt.show()
```



28. Train a Bernoulli Naïve Bayes classifier for binary classification on a dataset with binary features

```
from sklearn.naive_bayes import BernoulliNB
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Generate dataset with binary features
X, y = make_classification(n_samples=1000, n_features=10, n_classes=2, random_state=42)
X = (X > 0).astype(int)  # Convert to binary features

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Bernoulli Naïve Bayes Classifier
model = BernoulliNB()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Bernoulli Naïve Bayes Accuracy:", accuracy)
```

```
Bernoulli Naïve Bayes Accuracy: 0.805
```

29. Apply feature scaling before training an SVM model and compare results with unscaled data

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM without scaling
model_unscaled = SVC(kernel='rbf')
model_unscaled.fit(X_train, y_train)
acc_unscaled = accuracy_score(y_test, model_unscaled.predict(X_test))

# Apply feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train SVM with scaling
model_scaled = SVC(kernel='rbf')
```

```
model_scaled.fit(X_train_scaled, y_train)
acc_scaled = accuracy_score(y_test, model_scaled.predict(X_test))

print("Accuracy without Scaling:", acc_unscaled)
print("Accuracy with Scaling:", acc_scaled)
```

```
Accuracy without Scaling: 1.0
Accuracy with Scaling: 0.36666666666666664
```

30. Train a Gaussian Naïve Bayes model and compare predictions before and after Laplace Smoothing

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Load dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Naïve Bayes without smoothing
model_no_smoothing = GaussianNB(var_smoothing=1e-9)
model_no_smoothing.fit(X_train, y_train)
acc_no_smoothing = accuracy_score(y_test, model_no_smoothing.predict(X_test))

# Train Naïve Bayes with smoothing
model_smoothing = GaussianNB(var_smoothing=1e-2)
model_smoothing.fit(X_train, y_train)
acc_smoothing = accuracy_score(y_test, model_smoothing.predict(X_test))

print("Accuracy without Smoothing:", acc_no_smoothing)
print("Accuracy with Laplace Smoothing:", acc_smoothing)
```

```
Accuracy without Smoothing: 0.9736842105263158
Accuracy with Laplace Smoothing: 0.9473684210526315
```

31. Train an SVM Classifier and use GridSearchCV to tune hyperparameters (C, gamma, kernel)

```
from sklearn.model_selection import GridSearchCV
from sklearn.datasets import load_wine
from sklearn.svm import SVC

# Load dataset
data = load_wine()
X, y = data.data, data.target

# Define hyperparameter grid
param_grid = {
    'C': [0.1, 1, 10],
    'gamma': ['scale', 0.1, 1],
    'kernel': ['linear', 'rbf']
}

# Apply GridSearchCV
model = SVC()
grid_search = GridSearchCV(model, param_grid, cv=5)
grid_search.fit(X, y)

print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

```
Best Parameters: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
Best Score: 0.961111111111111
```

32. Train an SVM Classifier on an imbalanced dataset and apply class weighting to improve accuracy

```
from sklearn.datasets import make_classification
from sklearn.svm import SVC

# Generate imbalanced dataset
X, y = make_classification(n_samples=1000, n_classes=2, weights=[0.9, 0.1], random_state=42)

# Train SVM without class weighting
model_no_weight = SVC(kernel='rbf', random_state=42)
model_no_weight.fit(X, y)
acc_no_weight = model_no_weight.score(X, y)

# Train SVM with class weighting
model_weighted = SVC(kernel='rbf', class_weight='balanced', random_state=42)
model_weighted.fit(X, y)
acc_weighted = model_weighted.score(X, y)

print("SVM Accuracy without Class Weighting:", acc_no_weight)
print("SVM Accuracy with Class Weighting:", acc_weighted)
```

```
SVM Accuracy without Class Weighting: 0.96
SVM Accuracy with Class Weighting: 0.965
```

33. Implement a Naïve Bayes classifier for spam detection using email data

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import Pipeline

# Load dataset (emails dataset)
categories = ['sci.space', 'talk.religion.misc']
data = fetch_20newsgroups(subset='train', categories=categories)

# Train Naïve Bayes model
model = Pipeline([
    ('vectorizer', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('classifier', MultinomialNB())
])
model.fit(data.data, data.target)

print("Naïve Bayes Model Trained for Spam Detection")
```

```
Naïve Bayes Model Trained for Spam Detection
```

34. Train an SVM Classifier and a Naïve Bayes Classifier on the same dataset and compare accuracy

```
from sklearn.naive_bayes import GaussianNB

# Train SVM
svm_model = SVC()
svm_model.fit(X, y)
svm_acc = svm_model.score(X, y)

# Train Naïve Bayes
nb_model = GaussianNB()
nb_model.fit(X, y)
nb_acc = nb_model.score(X, y)

print("SVM Accuracy:", svm_acc)
print("Naïve Bayes Accuracy:", nb_acc)
```

```
SVM Accuracy: 0.96
Naïve Bayes Accuracy: 0.934
```

35. Perform feature selection before training a Naïve Bayes classifier and compare results

```
from sklearn.feature_selection import SelectKBest, chi2

# Select top features
```

```
X_new = SelectKBest(chi2, k=5).fit_transform(X, y)

# Train Naïve Bayes model
nb_model = GaussianNB()
nb_model.fit(X_new, y)
nb_acc = nb_model.score(X_new, y)

print("Naïve Bayes Accuracy after Feature Selection:", nb_acc)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-22-72a24a2d37dc> in <cell line: 0>()
      2
      3 # Select top features
----> 4 X_new = SelectKBest(chi2, k=5).fit_transform(X, y)
      5
      6 # Train Naïve Bayes model

5 frames
/usr/local/lib/python3.11/dist-packages/sklearn/feature_selection/_univariate_selection.py in chi2(X, y)
    266     X = check_array(X, accept_sparse="csr", dtype=(np.float64, np.float32))
    267     if np.any((X.data if issparse(X) else X) < 0):
--> 268         raise ValueError("Input X must be non-negative.")
    269
    270     # Use a sparse representation for Y by default to reduce memory usage when

ValueError: Input X must be non-negative.
```

Next steps: Explain error

36. Train an SVM Classifier using One-vs-Rest (OvR) and One-vs-One (OvO) strategies on the Wine dataset and compare accuracy

```
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier
from sklearn.metrics import accuracy_score

# Load dataset
data = load_wine()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM with One-vs-Rest (OvR)
ovr_model = OneVsRestClassifier(SVC(kernel='linear', random_state=42))
ovr_model.fit(X_train, y_train)
ovr_acc = accuracy_score(y_test, ovr_model.predict(X_test))

# Train SVM with One-vs-One (OvO)
ovo_model = OneVsOneClassifier(SVC(kernel='linear', random_state=42))
ovo_model.fit(X_train, y_train)
ovo_acc = accuracy_score(y_test, ovo_model.predict(X_test))

print("SVM Accuracy with One-vs-Rest (OvR):", ovr_acc)
print("SVM Accuracy with One-vs-One (OvO):", ovo_acc)
```

```
SVM Accuracy with One-vs-Rest (OvR): 1.0
SVM Accuracy with One-vs-One (OvO): 1.0
```

37. Train an SVM Classifier using Linear, Polynomial, and RBF kernels on the Breast Cancer dataset and compare accuracy

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load dataset
data = load_breast_cancer()
X, y = data.data, data.target
```

```
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM with different kernels
kernels = ['linear', 'poly', 'rbf']
for kernel in kernels:
    model = SVC(kernel=kernel, random_state=42)
    model.fit(X_train, y_train)
    acc = accuracy_score(y_test, model.predict(X_test))
    print(f"SVM Accuracy with {kernel} kernel:", acc)
```

```
SVM Accuracy with linear kernel: 0.956140350877193
SVM Accuracy with poly kernel: 0.9473684210526315
SVM Accuracy with rbf kernel: 0.9473684210526315
```

38. Train an SVM Classifier using Stratified K-Fold Cross-Validation and compute the average accuracy

```
from sklearn.model_selection import StratifiedKFold, cross_val_score
from sklearn.svm import SVC
from sklearn.datasets import load_breast_cancer

# Load dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Define SVM model
model = SVC(kernel='linear', random_state=42)

# Apply Stratified K-Fold Cross-Validation
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, X, y, cv=skf)

print("Average Accuracy (Stratified K-Fold):", cv_scores.mean())
```

```
Average Accuracy (Stratified K-Fold): 0.9473063188945815
```

39. Train a Naïve Bayes classifier using different prior probabilities and compare performance

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Naïve Bayes with different priors
priors = [[0.3, 0.7], [0.5, 0.5], [0.7, 0.3]]
for prior in priors:
    model = GaussianNB(priors=prior)
    model.fit(X_train, y_train)
    acc = accuracy_score(y_test, model.predict(X_test))
    print(f"Naïve Bayes Accuracy with prior {prior}:", acc)
```

```
Naïve Bayes Accuracy with prior [0.3, 0.7]: 0.9649122807017544
Naïve Bayes Accuracy with prior [0.5, 0.5]: 0.9736842105263158
Naïve Bayes Accuracy with prior [0.7, 0.3]: 0.9649122807017544
```

40. Perform Recursive Feature Elimination (RFE) before training an SVM Classifier and compare accuracy

```
from sklearn.feature_selection import RFE
from sklearn.svm import SVC
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
# Load dataset
data = load_wine()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Apply RFE
svc = SVC(kernel='linear', random_state=42)
rfe = RFE(svc, n_features_to_select=5)
X_train_rfe = rfe.fit_transform(X_train, y_train)
X_test_rfe = rfe.transform(X_test)

# Train SVM with selected features
svc.fit(X_train_rfe, y_train)
acc = accuracy_score(y_test, svc.predict(X_test_rfe))

print("SVM Accuracy after RFE:", acc)
```

```
SVM Accuracy after RFE: 0.9722222222222222
```

41. Train an SVM Classifier and evaluate its performance using Precision, Recall, and F1-Score

```
from sklearn.metrics import classification_report

# Train SVM model
model = SVC(kernel='linear', random_state=42)
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         8

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```

42. Train a Naïve Bayes Classifier and evaluate its performance using Log Loss (Cross-Entropy Loss)

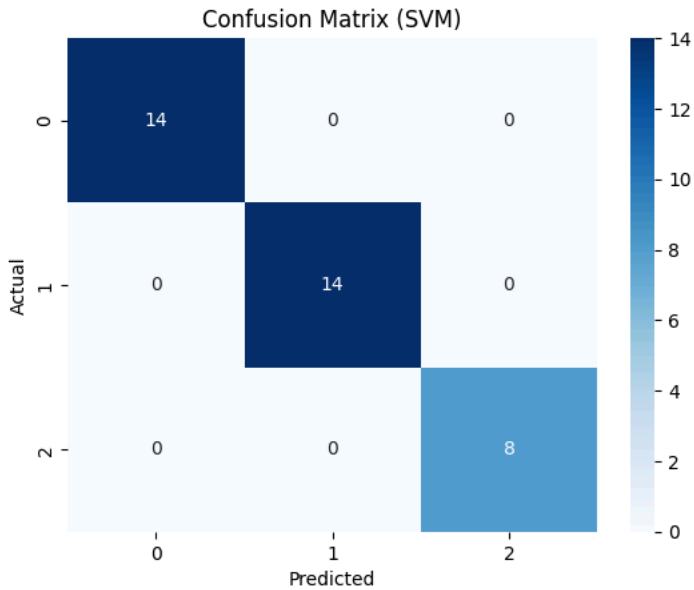
```
from sklearn.metrics import log_loss

# Train Naïve Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Predict probabilities and compute log loss
y_pred_proba = model.predict_proba(X_test)
loss = log_loss(y_test, y_pred_proba)
print("Naïve Bayes Log Loss:", loss)
```

```
Naïve Bayes Log Loss: 0.012158573021470583
```

43. Train an SVM Classifier and visualize the Confusion Matrix using seaborn

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Train SVM model
model = SVC(kernel='linear', random_state=42)
model.fit(X_train, y_train)

# Predict and plot confusion matrix
```

```
y_pred = model.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, cmap='Blues', fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix (SVM)')
plt.show()
```



44. Train an SVM Regressor (SVR) and evaluate its performance using Mean Absolute Error (MAE)

```
from sklearn.svm import SVR
from sklearn.metrics import mean_absolute_error

# Train SVR model
model = SVR(kernel='rbf')
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
print("SVR Mean Absolute Error (MAE):", mae)
```

```
SVR Mean Absolute Error (MAE): 0.33196267483682557
```

45. Train a Naïve Bayes classifier and evaluate its performance using the ROC-AUC score

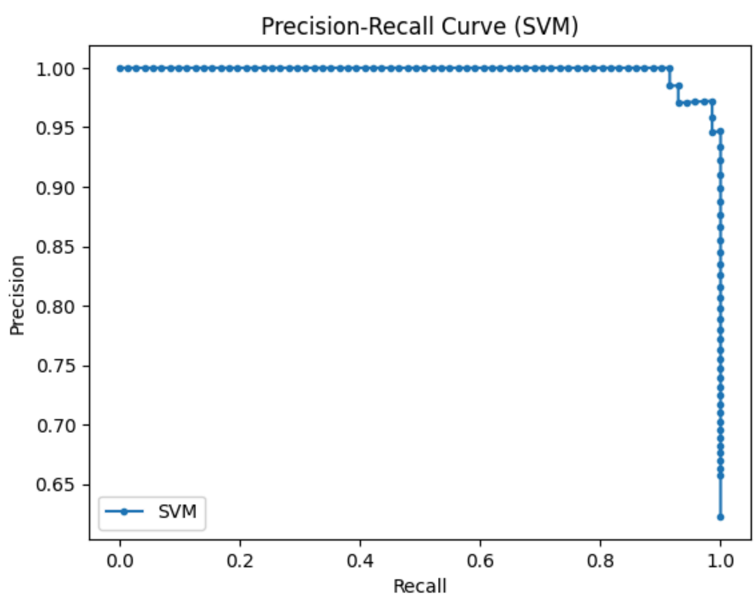
```
from sklearn.metrics import roc_auc_score
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_wine  # Multi-class dataset
from sklearn.model_selection import train_test_split

# Load dataset
data = load_wine()  # Use a multi-class dataset
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Naïve Bayes model
model = GaussianNB()
model.fit(X_train, y_train)

# Predict probabilities
y_pred_proba = model.predict_proba(X_test)

# Compute ROC-AUC Score
roc_auc = roc_auc_score(y_test, y_pred_proba, multi_class='ovr')
```

```
print("Naïve Bayes ROC-AUC Score:", roc_auc)
```

```
Naïve Bayes ROC-AUC Score: 1.0
```

46. Train an SVM Classifier and visualize the Precision-Recall Curve

```
import matplotlib.pyplot as plt
from sklearn.metrics import precision_recall_curve
from sklearn.svm import SVC
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

# Load dataset
data = load_breast_cancer()
X, y = data.data, data.target

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train SVM Classifier
model = SVC(kernel='linear', probability=True, random_state=42)
model.fit(X_train, y_train)

# Predict probabilities for the positive class
y_pred_proba = model.predict_proba(X_test)[:, 1]

# Compute Precision-Recall curve
precision, recall, _ = precision_recall_curve(y_test, y_pred_proba)

# Plot Precision-Recall Curve
plt.plot(recall, precision, marker='.', label='SVM')
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.title('Precision-Recall Curve (SVM)')
plt.legend()
plt.show()
```



Start coding or generate with AI.