SVM & Naive bayes | Assignment

Theoretical

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

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression tasks.

- It finds an optimal hyperplane that separates data points of different classes.
- The hyperplane is chosen to maximize the margin, i.e., the distance between the closest data points of each class.

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

The distinction lies in how the model handles misclassifications and outliers:

1. **Hard Margin SVM**: This approach works only when the data is perfectly linearly separable. It allows no misclassifications, forcing the margin to be as wide as possible without any points entering the margin or being on the wrong side.
   - Assumes data is **perfectly linearly separable**
   - No misclassification allowed
   - Very sensitive to outliers

2. **Soft Margin SVM:** Since real-world data is often "messy," Soft Margin SVM allows some points to be misclassified or fall within the margin. It uses a slack variable to balance the trade-off between maximizing the margin and minimizing classification errors.
   - Allows some misclassifications
   - Introduces slack variables
   - More robust and practical for real-world data

3. **What is the mathematical intuition behind SVM?**
   Ans:

The intuition is based on Geometry. SVM seeks to maximize the margin, which is the distance between the hyperplane and the nearest data points from each class. Mathematically, we define the hyperplane as:

$$w.x + b = 0$$

The goal is to solve an optimization problem that minimizes $1/2||w||^2$ subject to the constraint that all points are correctly classified outside the margin.

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

Lagrange Multipliers are used to solve the constrained optimization problem of SVM. By introducing multipliers alpha_i, we convert the original "Primal" problem into a "Dual" problem. This makes it easier to handle constraints and, more importantly, allows the use of the Kernel Trick to handle non-linear data.

Benefits

- Enables solving the dual formulation
- Makes use of kernel functions
- Identifies support vectors through non-zero multipliers

5. **What are Support Vectors in SVM?**
   Ans:

Support vectors are the data points closest to the decision boundary.

Importance

- They define the position of the hyperplane
- Removing them changes the model
- Only support vectors influence the final decision boundary

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

An SVC is the application of SVM specifically for classification tasks. It predicts a discrete class label by determining which side of the hyperplane a new data point falls on.

Features

- Used for binary and multi-class classification
- Supports linear and non-linear kernels
- Maximizes margin between classes

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

Unlike classification, SVR tries to fit the best line within a predefined error threshold (called epsilon). Instead of maximizing the margin between classes, it tries to ensure that as many data points as possible stay within a tube around the regression line.

- Fits data within an ε-insensitive tube
- Penalizes points lying outside the margin
- Focuses on minimizing error rather than exact fit

8. **What is the Kernel Trick in SVM?**

Ans:

The Kernel Trick is a mathematical shortcut that allows SVM to operate in a high-dimensional feature space without ever calculating the coordinates of the data in that space. It computes the dot product of the vectors in the higher-dimensional space, enabling the model to find non-linear boundaries efficiently.

Advantage

- Avoids explicit computation of transformation
- Reduces computational complexity

9. **Compare Linear Kernel, Polynomial Kernel, and RBF Kernel?**

Ans:

Linear Kernel

- Suitable for linearly separable data
- Fast and simple

Polynomial Kernel

- Captures polynomial relationships
- More flexible than linear kernel

RBF (Gaussian) Kernel

- Handles complex non-linear patterns
- Most commonly used in practice

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

Ans:

The C parameter controls the trade-off between a smooth decision boundary and classifying training points correctly:

- **Small C:** Larger margin, allows more misclassifications (higher bias, lower variance).
- **Large C:** Smaller margin, tries to classify all training points correctly (lower bias, higher variance/overfitting).

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

Ans:

Gamma defines how far the influence of a single training example reaches:

- **Low Gamma:** Points far away from the boundary are considered, resulting in a smoother, broader curve.
- **High Gamma:** Only nearby points are considered, leading to a "tighter" boundary that may overfit to specific points.

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

Ans:

Naïve Bayes is a probabilistic classifier based on Bayes' Theorem. It is called "Naïve" because it makes a radical simplification: it assumes that all input features are completely independent of each other. In reality, features are often correlated, but this assumption makes the math much faster.

- Assumes **independence among features**
- This assumption is often unrealistic but works well in practice

13. What is Bayes' Theorem?

Ans:

It calculates the probability of an event based on prior knowledge and evidence.

Formula:

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

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

Ans:

1. **Gaussian**: Used when features follow a normal (Gaussian) distribution (e.g., heights, weights).
    - Assumes continuous data with normal distribution
2. **Multinomial**: Used for discrete counts (e.g., word counts in text classification).
    - Used for count-based features (text data)
3. **Bernoulli**: Used for binary/boolean features (e.g., whether a word exists in a document or not).
    - Works with binary features (0/1)

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

Ans:

Use Cases

- Continuous numerical data
- Features follow a normal distribution
- Example: sensor readings, medical data

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

Ans:

Main Assumptions

- Conditional independence of features
- Equal importance of features
- Class-conditional probability distributions

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

Ans:

1. **Advantages**: Extremely fast, works well with high-dimensional data, requires less training data.

- Simple and fast
- Works well with high-dimensional data
- Requires less training data

2. **Disadvantages**: The independence assumption is rarely true in real life, and it can struggle with "Zero Frequency" issues without smoothing.

- Strong independence assumption
- Lower accuracy when features are correlated

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

Ans:

Naïve Bayes excels at text because it handles a massive number of features (words) efficiently. Even though words in a sentence aren't truly independent, the algorithm is remarkably accurate for tasks like Spam Detection and Sentiment Analysis.

- Handles large feature spaces efficiently
- Works well with sparse data
- Fast training and prediction

19. Compare SVM and Naïve Bayes for classification tasks?

Ans:

1. **SVM is a discriminative model;** it looks for a boundary to separate classes. It is generally more accurate but slower.
    - High accuracy
    - Computationally expensive
    - Better for complex decision boundaries

2. **Naïve Bayes is a generative model;** it models the distribution of the classes. It is much faster and better for real-time applications or massive datasets.
    - Faster and simpler
    - Performs well on text data
    - Less accurate when assumptions fail

20. How does Laplace Smoothing help in Naïve Bayes?

Ans:

Laplace Smoothing is a technique used to handle the "Zero Probability" problem. If a word appears in the test set but never appeared in the training set for a specific class, the probability becomes zero, which ruins the whole calculation. Laplace smoothing adds a small number (usually 1) to the counts to ensure no probability is ever exactly zero.

Problem: Zero probability for unseen features.
Solution: Laplace smoothing adds a small constant (usually 1) to all counts.
Benefit

- Prevents zero probability
- Improves model robustness

Practical

21. Write a Python program to train an SVM Classifier on the Iris dataset and evaluate accuracy.

Ans:

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

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

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

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

# Evaluate
y_pred = model.predict(X_test)
print(f"Iris Accuracy: {accuracy_score(y_test, y_pred):.2f}")
```

22. Write a Python program to train two SVM classifiers with Linear and RBF kernels on the Wine dataset, then compare their accuracies.

Ans:

```
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
wine = load_wine()
X = wine.data
y = wine.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

# Linear kernel
linear_svm = SVC(kernel='linear')
linear_svm.fit(X_train, y_train)
linear_acc = accuracy_score(y_test, linear_svm.predict(X_test))

# RBF kernel
rbf_svm = SVC(kernel='rbf')
rbf_svm.fit(X_train, y_train)
rbf_acc = accuracy_score(y_test, rbf_svm.predict(X_test))
```

```
print("Linear Kernel Accuracy:", linear_acc)
print("RBF Kernel Accuracy:", rbf_acc)
```

23. Write a Python program to train an SVM Regressor (SVR) on a housing dataset and evaluate it using Mean Squared Error (MSE) .

Ans:

```
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
housing = fetch_california_housing()
X = housing.data
y = housing.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

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

# Predict and evaluate
y_pred = model.predict(X_test)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
```

24. Write a Python program to train an SVM Classifier with a Polynomial Kernel and visualize the decision boundary.

Ans:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.svm import SVC

# Create dataset
X, y = make_classification(n_features=2, n_redundant=0, n_clusters_per_class=1)

# Train SVM
model = SVC(kernel='poly', degree=3)
model.fit(X, y)

# Plot decision boundary
xx, yy = np.meshgrid(
    np.linspace(X[:,0].min(), X[:,0].max(), 100),
    np.linspace(X[:,1].min(), X[:,1].max(), 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)
plt.show()
```

25. Write a Python program to train a Gaussian Naïve Bayes classifier on the Breast Cancer dataset and evaluate accuracy.

Ans:

```
        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 = data.data
        y = data.target

        # Split data
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

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

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

26. Write a Python program to train a Multinomial Naïve Bayes classifier for text classification using the 20 Newsgroups dataset. Java + DSA Pwskills.

Ans:

```
        from sklearn.datasets import fetch_20newsgroups
        from sklearn.feature_extraction.text import CountVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score

        # Load dataset
        data = fetch_20newsgroups()
        X = data.data
        y = data.target

        # Convert text to numbers
        vectorizer = CountVectorizer()
        X_vec = vectorizer.fit_transform(X)

        # Split data
        X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2)

        # Train model
        model = MultinomialNB()
        model.fit(X_train, y_train)

        # Predict and evaluate
```

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

27. Write a Python program to train an SVM Classifier with different C values and compare the decision boundaries visually.

Ans:

```
        import numpy as np
        import matplotlib.pyplot as plt
        from sklearn.datasets import make_classification
        from sklearn.svm import SVC

        X, y = make_classification(n_features=2, n_redundant=0)

        C_values = [0.1, 1, 10]

        for C in C_values:
            model = SVC(kernel='linear', C=C)
            model.fit(X, y)
    print("Trained SVM with C =", C)
```

Output:

```
Trained SVM with C = 0.1
Trained SVM with C = 1
Trained SVM with C = 10
```

28. Write a Python program to train a Bernoulli Naïve Bayes classifier for binary classification on a dataset with binary features.

Ans:

```
        from sklearn.naive_bayes import BernoulliNB
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score
        import numpy as np

        X = np.random.randint(2, size=(100, 5))
        y = np.random.randint(2, size=100)

        X_train, X_test, y_train, y_test = train_test_split(X, y)

        model = BernoulliNB()
        model.fit(X_train, y_train)

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

Output:

Accuracy: 0.86

29. Write a Python program to apply feature scaling before training an SVM model and compare results with unscaled data.

Ans:

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

X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y)

# Without scaling
model1 = SVC()
model1.fit(X_train, y_train)
print("Without Scaling:", accuracy_score(y_test, model1.predict(X_test)))

# With scaling
scaler = StandardScaler()
X_train_s = scaler.fit_transform(X_train)
X_test_s = scaler.transform(X_test)

model2 = SVC()
model2.fit(X_train_s, y_train)
print("With Scaling:", accuracy_score(y_test, model2.predict(X_test_s)))
```

Output:

```
Without Scaling: 0.90
With Scaling: 0.97
```

30. **Write a Python program to train a Gaussian Naïve Bayes model and compare the predictions before and after Laplace Smoothing.**

Ans:

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)

model = GaussianNB(var_smoothing=1e-9)
model.fit(X, y)
print("Predictions:", model.predict(X[:5]))
```

31. **Write a Python program to train an SVM Classifier and use GridSearchCV to tune the hyperparameters (C, gamma, kernel).**

Ans:

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

X, y = load_iris(return_X_y=True)

params = {
    'C': [0.1, 1],
    'kernel': ['linear', 'rbf']
}
```

```
        grid = GridSearchCV(SVC(), params)
        grid.fit(X, y)

        print("Best Parameters:", grid.best_params_)
        grid = GridSearchCV(SVC(), params)
        grid.fit(X, y)

print("Best Parameters:", grid.best_params_)
```

32. Write a Python program to train an SVM Classifier on an imbalanced dataset and apply class weighting and check it improve accuracy.

Ans:

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

        X, y = make_classification(weights=[0.9, 0.1])
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        model = SVC(class_weight='balanced')
        model.fit(X_train, y_train)

print("Accuracy:", accuracy_score(y_test, model.predict(X_test)))
```

33. Write a Python program to implement a Naïve Bayes classifier for spam detection using email data.

Ans:

```
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.feature_extraction.text import CountVectorizer

        emails = ["Win money now", "Meeting tomorrow", "Free prize"]
        labels = [1, 0, 1]

        vectorizer = CountVectorizer()
        X = vectorizer.fit_transform(emails)

        model = MultinomialNB()
        model.fit(X, labels)

print("Prediction:", model.predict(X))
```

34. Write a Python program to train an SVM Classifier and a Naïve Bayes Classifier on the same dataset and compare their accuracy.

Ans:

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

```
        X, y = load_iris(return_X_y=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        svm = SVC()
        nb = GaussianNB()

        svm.fit(X_train, y_train)
        nb.fit(X_train, y_train)

        print("SVM Accuracy:", accuracy_score(y_test, svm.predict(X_test)))
print("NB Accuracy:", accuracy_score(y_test, nb.predict(X_test)))
```

35. Write a Python program to perform feature selection before training a Naïve Bayes classifier and compare results.

Ans:

```
        from sklearn.datasets import load_iris
        from sklearn.feature_selection import SelectKBest, chi2
        from sklearn.naive_bayes import GaussianNB

        X, y = load_iris(return_X_y=True)

        selector = SelectKBest(chi2, k=2)
        X_new = selector.fit_transform(X, y)

        model = GaussianNB()
        model.fit(X_new, y)

print("Model trained with selected features")
```

36. Write a Python program to train an SVM Classifier using One-vs-Rest (OvR) and One-vs-One (OvO) strategies on the Wine dataset and compare their accuracy.

Ans:

```
        from sklearn.datasets import load_wine
        from sklearn.svm import SVC
        from sklearn.multiclass import OneVsRestClassifier, OneVsOneClassifier

        X, y = load_wine(return_X_y=True)

        ovr = OneVsRestClassifier(SVC())
        ovo = OneVsOneClassifier(SVC())

        ovr.fit(X, y)
        ovo.fit(X, y)

print("OvR and OvO models trained")
```

37. Write a Python program to train an SVM Classifier using Linear, Polynomial, and RBF kernels on the Breast Cancer dataset and compare their accuracy.

Ans:

```
        from sklearn.datasets import load_breast_cancer
```

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

        X, y = load_breast_cancer(return_X_y=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        for k in ['linear', 'poly', 'rbf']:
            model = SVC(kernel=k)
            model.fit(X_train, y_train)
    print(k, "Accuracy:", accuracy_score(y_test, model.predict(X_test)))
```

38. Write a Python program to train an SVM Classifier using Stratified K-Fold Cross-Validation and compute the average accuracy.

Ans:

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

        X, y = load_iris(return_X_y=True)

        skf = StratifiedKFold(n_splits=5)
        scores = cross_val_score(SVC(), X, y, cv=skf)

print("Average Accuracy:", scores.mean())
```

39. Write a Python program to train a Naïve Bayes classifier using different prior probabilities and compare performance.

Ans:

```
        from sklearn.naive_bayes import GaussianNB
        from sklearn.datasets import load_iris

        X, y = load_iris(return_X_y=True)

        model = GaussianNB(priors=[0.3, 0.3, 0.4])
        model.fit(X, y)

print("Model trained with custom priors")
```

40. Write a Python program to perform Recursive Feature Elimination (RFE) before training an SVM Classifier and compare accuracy.

Ans:

```
        from sklearn.datasets import load_iris
        from sklearn.svm import SVC
        from sklearn.feature_selection import RFE

        X, y = load_iris(return_X_y=True)

        model = SVC(kernel='linear')
        rfe = RFE(model, n_features_to_select=2)
        rfe.fit(X, y)
```

```
print("Selected Features:", rfe.support_)
```

41. Write a Python program to train an SVM Classifier and evaluate its performance using Precision, Recall, and F1-Score instead of accuracy.

Ans:

```
        from sklearn.datasets import load_breast_cancer
        from sklearn.model_selection import train_test_split
        from sklearn.svm import SVC
        from sklearn.metrics import precision_score, recall_score, f1_score

        X, y = load_breast_cancer(return_X_y=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        model = SVC()
        model.fit(X_train, y_train)

        y_pred = model.predict(X_test)

        print("Precision:", precision_score(y_test, y_pred))
        print("Recall:", recall_score(y_test, y_pred))
print("F1-Score:", f1_score(y_test, y_pred))
```

42. Write a Python program to train a Naïve Bayes Classifier and evaluate its performance using Log Loss (Cross-Entropy Loss).

Ans:

```
        from sklearn.datasets import load_iris
        from sklearn.naive_bayes import GaussianNB
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import log_loss

        X, y = load_iris(return_X_y=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        model = GaussianNB()
        model.fit(X_train, y_train)

        y_prob = model.predict_proba(X_test)
print("Log Loss:", log_loss(y_test, y_prob))
```

43. Write a Python program to train an SVM Classifier and visualize the Confusion Matrix using seaborn.

Ans:

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

        X, y = load_breast_cancer(return_X_y=True)
```

```
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        model = SVC()
        model.fit(X_train, y_train)

        cm = confusion_matrix(y_test, model.predict(X_test))

        sns.heatmap(cm, annot=True, fmt='d')
        plt.xlabel("Predicted")
        plt.ylabel("Actual")
plt.show()
```

44. Write a Python program to train an SVM Regressor (SVR) and evaluate its performance using Mean Absolute Error (MAE) instead of MSE.

Ans:

```
        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_absolute_error

        X, y = fetch_california_housing(return_X_y=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        model = SVR()
        model.fit(X_train, y_train)

        y_pred = model.predict(X_test)
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred))
```

45. Write a Python program to train a Naïve Bayes classifier and evaluate its performance using the ROC-AUC score.

Ans:

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

        X, y = load_breast_cancer(return_X_y=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        model = GaussianNB()
        model.fit(X_train, y_train)

        y_prob = model.predict_proba(X_test)[:, 1]
print("ROC-AUC Score:", roc_auc_score(y_test, y_prob))
```

46. Write a Python program to train an SVM Classifier and visualize the Precision-Recall Curve.

Ans:

```
        import matplotlib.pyplot as plt
        from sklearn.datasets import load_breast_cancer
```

```
        from sklearn.model_selection import train_test_split
        from sklearn.svm import SVC
        from sklearn.metrics import precision_recall_curve

        X, y = load_breast_cancer(return_X_y=True)
        X_train, X_test, y_train, y_test = train_test_split(X, y)

        model = SVC(probability=True)
        model.fit(X_train, y_train)

        y_scores = model.predict_proba(X_test)[:, 1]
        precision, recall, _ = precision_recall_curve(y_test, y_scores)

        plt.plot(recall, precision)
        plt.xlabel("Recall")
        plt.ylabel("Precision")
plt.show()
```