Question 1: What is a Support Vector Machine (SVM), and how does it work?

Ans:- A Support Vector Machine (SVM) is a supervised machine learning algorithm used primarily for classification tasks (though it can also be used for regression).

🔧 **How It Works**

Find the Hyperplane:

SVM tries to find the hyperplane that best separates the classes of data points. In 2D, this is a line; in 3D, it's a plane; in higher dimensions, it's called a hyperplane.

Maximize the Margin:

The "best" hyperplane is the one that maximizes the margin—the distance between the hyperplane and the nearest data points from each class. These nearest points are called support vectors.

Classification Decision:

Once the hyperplane is established, new data points can be classified by checking on which side of the hyperplane they lie.

Q2.Question 2: Explain the difference between Hard Margin and Soft Margin SVM.?

Ans:-

Feature	Hard Margin SVM	Soft Margin SVM
Data requirement	Perfectly linearly separable	Can handle overlapping/noisy data
Margin	Rigid, no tolerance	Flexible, allows violations
Misclassifications	Not allowed	Allowed (with penalty)
Regularization	No	Uses **C** to control trade-off

Question 3: What is the Kernel Trick in SVM? Give one example of a kernel and explain its use case ?

Ans:- The Kernel Trick is a technique used in Support Vector Machines (SVMs) to enable them to perform non-linear classification.

🔢 **Example: Radial Basis Function (RBF) Kernel**

```
K(x,x')=exp(−γ∥x−x'∥2)
```

- Here,γ is a tunable parameter that controls the spread of the kernel.
- x and *x*'x'are data points.

🧠 Use Case:

Useful when the data has a complex, non-linear boundary.

Common in image recognition, speech recognition, and bioinformatics (e.g., classifying DNA sequences).

💡 Behavior

- The RBF kernel measures similarity between two data points.
- It maps the data into infinite-dimensional space.
- The parameter γ controls how far the influence of a single training example reaches.

Question 4: What is a Naïve Bayes Classifier, and why is it called "naïve"?

Ans:- The Naïve Bayes Classifier is a supervised machine learning algorithm based on Bayes' Theorem with a strong (naïve) assumption that all features are independent of each other, given the class.

Despite this unrealistic assumption, it works surprisingly well in many practical applications, especially in text classification.

🧠 **Why is it called "Naïve"?**

It is called naïve because it assumes all features are independent given the class label — this is a strong and often unrealistic assumption, hence the term "naïve."

❗ **Example:**

If you're classifying emails as spam or not spam, and the words "free" and "money" appear together, Naïve Bayes assumes their probabilities are independent:

P("free", "money"|spam)=*P*("free"|spam)*P*("money"|spam)P("free" "money"|spam)=P("free"|spam)·P("money"|spam)

Question 5: Describe the Gaussian, Multinomial, and Bernoulli Naïve Bayes variants.When would you use each one?

Ans:- 1.🧮 Gaussian Naïve Bayes

- Assumes that features are continuous and follow a normal (Gaussian) distribution.
- Common in datasets where features are real-valued, like measurements, temperatures, or physical quantities.

🔬 How It Works:

Calculates the likelihood of features using the probability density function of a Gaussian (bell curve).

🔹 Use Case:

- Iris flower classification
- Medical data with continuous lab test results
- Any dataset with numeric features

✅ Use When:

- "You have numerical/continuous data.

2. 📊 Multinomial Naïve Bayes Designed for discrete count data, such as word counts or term frequencies in documents.

Assumes features represent frequencies or counts (not binary or continuous).

📘 Used When:

- Features are discrete counts (e.g., word counts in a document)

🔬 How It Works:

Uses a multinomial distribution to model the number of times events occur.

🔹 Use Case:

Text classification (e.g., spam detection, topic categorization)

Document classification using bag-of-words or TF-IDF

🔍 Example:

If a document has 3 occurrences of "free" and 2 of "money", Multinomial NB uses these counts to compute probabilities.

3. ✅ Bernoulli Naïve Bayes

📘 Used When:

- "Features are binary/Boolean (0 or 1, i.e., presence or absence of a feature).

🔬 How It Works:

- "Each feature is modeled using a Bernoulli distribution.
- Assumes binary feature vectors — each feature is present (1) or absent (0).
- Ideal for binary/boolean input data.

🔹 Use Case:

- Text classification with binary word indicators (e.g., whether a word appears at all)
- Short texts, or feature spaces where presence/absence is more meaningful than frequency

🔍 Example:

The feature vector might look like: [“free” = 1, “money” = 1, “click” = 0]

```
# 1. Import libraries
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# 2. Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# 3. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# 4. Train Gaussian Naïve Bayes model
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# 5. Make predictions
y_pred = gnb.predict(X_test)

# 6. Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 1.00
```

Question 6: Write a Python program to:

● Load the Iris dataset ● Train an SVM Classifier with a linear kernel. ● Print the model's accuracy and support vectors.

```
# Import required libraries
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

# Step 1: Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Step 2: Split into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 3: Create and train SVM with linear kernel
svm_linear = SVC(kernel='linear')
svm_linear.fit(X_train, y_train)

# Step 4: Make predictions
y_pred = svm_linear.predict(X_test)

# Step 5: Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")

# Step 6: Print support vectors
support_vectors = svm_linear.support_vectors_
print("Support Vectors:")
print(support_vectors)
```

```
Accuracy: 1.00
Support Vectors:
[[4.8 3.4 1.9 0.2]
 [5.1 3.3 1.7 0.5]
 [4.5 2.3 1.3 0.3]
 [5.6 3.  4.5 1.5]
 [5.4 3.  4.5 1.5]
 [6.7 3.  5.  1.7]
 [5.9 3.2 4.8 1.8]
 [5.1 2.5 3.  1.1]
 [6.  2.7 5.1 1.6]
 [6.3 2.5 4.9 1.5]
 [6.1 2.9 4.7 1.4]
 [6.5 2.8 4.6 1.5]
 [6.9 3.1 4.9 1.5]
 [6.3 2.3 4.4 1.3]
 [6.3 2.5 5.  1.9]
 [6.3 2.8 5.1 1.5]
 [6.3 2.7 4.9 1.8]
 [6.  3.  4.8 1.8]
 [6.  2.2 5.  1.5]
 [6.2 2.8 4.8 1.8]
 [6.5 3.  5.2 2. ]
 [7.2 3.  5.8 1.6]
 [5.6 2.8 4.9 2. ]
 [5.9 3.  5.1 1.8]
 [4.9 2.5 4.5 1.7]]
```

** Question 7: Write a Python program to:**

● Load the Breast Cancer dataset ● Train a Gaussian Naïve Bayes model ● Print its classification report including precision, recall, and F1-score.

```
# Step 1: Import necessary libraries
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 classification_report

# Step 2: Load the Breast Cancer dataset
data = load_breast_cancer()
X = data.data
y = data.target

# Step 3: Split data into training and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Train Gaussian Naïve Bayes classifier
gnb = GaussianNB()
gnb.fit(X_train, y_train)

# Step 5: Make predictions
y_pred = gnb.predict(X_test)

# Step 6: Print classification report
print(classification_report(y_test, y_pred))

# Step 5: Make predictions
y_pred = gnb.predict(X_test)

# Step 6: Print classification report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.93      0.96        43
           1       0.96      1.00      0.98        71

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114

              precision    recall  f1-score   support

           0       1.00      0.93      0.96        43
           1       0.96      1.00      0.98        71

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

Question 8: Write a Python program to:

● Train an SVM Classifier on the Wine dataset using GridSearchCV to find the best C and gamma. ● Print the best hyperparameters and accuracy

```
# Step 1: Import required libraries
from sklearn.datasets import load_wine
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Step 2: Load the Wine dataset
wine = load_wine()
X = wine.data
y = wine.target

# Step 3: Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Step 4: Define SVM model and parameter grid
svm = SVC()
```

```
param_grid = {'C': [0.1, 1, 10, 100], 'gamma': [1, 0.1, 0.01, 0.001]}

# Step 5: Grid Search with 5-fold cross-validation
grid_search = GridSearchCV(svm, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

# Step 6: Print best parameters and accuracy on test set
print("Best Parameters:", grid_search.best_params_)
print("Best Accuracy:", grid_search.best_score_)
```

```
Best Parameters: {'C': 100, 'gamma': 0.001}
Best Accuracy: 0.7179802955665024
```

Question 9: Write a Python program to:

● Train a Naïve Bayes Classifier on a synthetic text dataset (e.g. using sklearn.datasets.fetch_20newsgroups). ● Print the model's ROC-AUC score for its predictions.

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize

# Step 1: Load data (binary classification)
categories = ['sci.space', 'rec.sport.hockey']  # Two categories
newsgroups = fetch_20newsgroups(subset='all', categories=categories, remove=('headers', 'footers', 'quotes'))

# Step 2: Split data
X_train, X_test, y_train, y_test = train_test_split(newsgroups.data, newsgroups.target, test_size=0.2, random_state=42)

# Step 3: Text vectorization
vectorizer = TfidfVectorizer(stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)

# Step 4: Train Naïve Bayes classifier
clf = MultinomialNB()
clf.fit(X_train_tfidf, y_train)

# Step 5: Predict probabilities
y_proba = clf.predict_proba(X_test_tfidf)[:, 1]  # Probability for class 1

# Step 6: Compute ROC-AUC score
roc_auc = roc_auc_score(y_test, y_proba)
print(f"ROC-AUC Score: {roc_auc:.4f}")
```

```
ROC-AUC Score: 0.9947
```

Question 10: Imagine you're working as a data scientist for a company that handles email communications.Your task is to automatically classify emails as Spam or Not Spam. The emails may contain: ● Text with diverse vocabulary ● Potential class imbalance (far more legitimate emails than spam) ● Some incomplete or missing data

Explain the approach you would take to:

● Preprocess the data (e.g. text vectorization, handling missing data) ● Choose and justify an appropriate model (SVM vs. Naïve Bayes) ● Address class imbalance ● Evaluate the performance of your solution with suitable metrics And explain the business impact of your solution.

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# Load binary classification dataset (Spam-like: 'talk.politics.misc' vs Ham-like: 'sci.space')
data = fetch_20newsgroups(subset='all', categories=['talk.politics.misc', 'sci.space'], remove=('headers', 'footers', 'quotes'))
```

```python
# Check for class imbalance
class_counts = pd.Series(data.target).value_counts()
print(class_counts)

# Handle missing values (replace with empty string)
data.data = [text.replace('\n', ' ') if text else '' for text in data.data]

# Text preprocessing & TF-IDF vectorization
vectorizer = TfidfVectorizer(stop_words='english')
X = vectorizer.fit_transform(data.data)
y = 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 Multinomial Naive Bayes
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_predictions = nb_model.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_predictions)

# Predict labels and probabilities
y_pred_proba = nb_model.predict_proba(X_test)

# Print evaluation metrics
print(f"Naive Bayes Accuracy: {nb_accuracy:.2f}")

# ROC-AUC Score
roc_auc = roc_auc_score(y_test, y_pred_proba[:, 1])
print(f"ROC-AUC Score: {roc_auc:.4f}")

# Optional: Heatmap of confusion matrix
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
```

```
0    987
1    775
Name: count, dtype: int64
Naive Bayes Accuracy: 0.94
ROC-AUC Score: 0.9868
```