TASK 5

write a python code to classify the radio frequency signal using SVM and measure its accuracy

MEIC501P MACHINE LEARNING FOR COMMUNICATIONS

DONE BY:

DURGA SITALAKSHMI S

Registration number :24MEC0024

Mail ID: durgasitalakshmi.s2024@vitstudent.ac.in

Phone number: +91 973185028

AIM:

The aim of this task is to classify radio frequency signals using a Support Vector Machine (SVM) and evaluate the model's performance based on the accuracy metric.

Overview of Support Vector Machine (SVM)

Support Vector Machines (SVMs) are supervised learning models used for classification tasks. The goal of an SVM is to find the optimal hyperplane that separates different classes of data in the feature space. For binary classification tasks, SVM aims to maximize the margin between two classes while minimizing classification errors. The key components of SVM include:

1. **Kernel Function:**

- SVM uses kernel functions to project the data into higher dimensions, making it easier to classify linearly separable data. Some common kernels include:

a) Linear kernel (used in this task).

b) Radial Basis Function (RBF) kernel.

c) Polynomial kernel.

2. **Support Vectors:**

- The support vectors are the data points that lie closest to the decision boundary or hyperplane. They play a crucial role in defining the boundary and contribute to the final model.

3. **Hyperplane:**

- The hyperplane is the decision boundary that separates different classes. For two features, it is a line; for higher-dimensional spaces, it is a plane or a hyperplane.

OBJECTIVES:

Data Preprocessing:

- Load and process the radio frequency signal data.
- Extract Power Spectral Density (PSD) features from the signal.

- Split the data into training and testing sets.

Model Training:

- Implement the SVM model with a linear kernel.
- Train the model on the training dataset.

Model Evaluation:

- Evaluate the performance of the model by measuring the accuracy on the test dataset.
- Visualize the radio frequency signal and analyze the decision boundary.

ABOUT THE DATASET

The dataset consists of two classes of radio frequency signals:

- **Class 1:** Positive signals (e.g., signals where an object is detected).
- **Class 0:** Negative signals (e.g., signals with no object detected).

For each class, a set of .dat binary files contains the signal data. The dataset contains a total of 20 files—10 for each class.

Positive Dataset (Object Detected): This dataset consists of RF signals captured when an object is detected in the environment. These signals likely show distinct frequency patterns in the PSD that correlate to the object's presence.

Negative Dataset (No Object Detected): This dataset includes RF signals collected when no object is present. The frequency patterns in these signals are different from those in the positive dataset, providing a basis for the classifier to distinguish between the two classes.

- The dataset is synthetically generated to simulate real-world RF signals.

Basis of Generation:

- **Positive Data Generation:** Signals are generated with specific frequency components or patterns that represent the presence of an object (e.g., a shift in frequency or increase in power at certain bands).

- **Negative Data Generation:** These signals are generated without such frequency shifts, resembling a baseline or background signal.

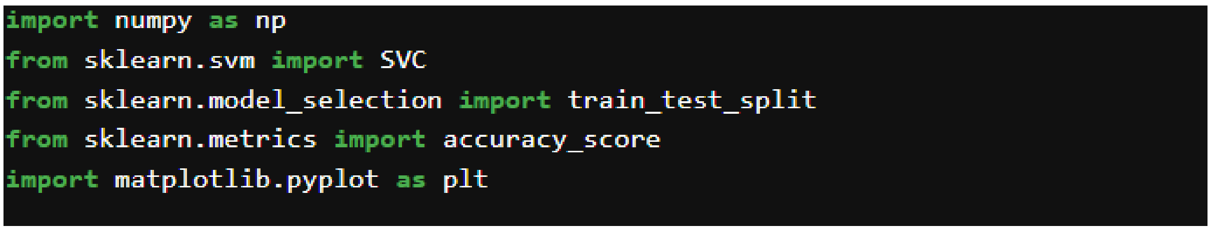
Synthetically Generated:

- In synthetic generation, signal patterns are created using known mathematical models or by adding certain noise and frequency components to simulate real-world RF signal behavior.
- For instance, positive signals might contain a higher power level in certain frequency bands, while negative signals have a flatter spectrum.

This method allows control over the signal characteristics, making it easier to test the classification model.

THE CODE

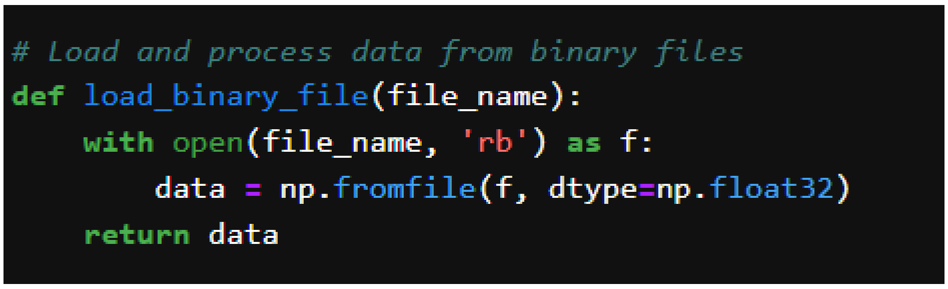
IMPORTING LIBRARIES



```python
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
```

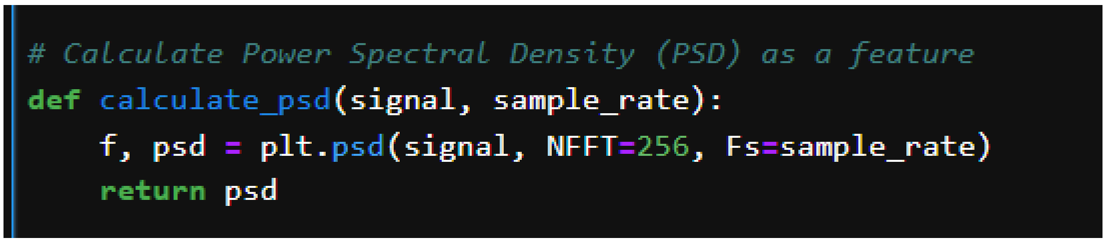
1. **numpy:** A powerful library for numerical computations. It is used for handling arrays and matrices, which are essential when working with large datasets such as signal data.
2. **sklearn.svm (SVC):** This module from scikit-learn provides the Support Vector Classification (SVC) algorithm, which is used to classify the RF signal based on extracted features (Power Spectral Density).
3. **sklearn.model_selection (train_test_split):** This function splits the dataset into training and testing sets, helping to evaluate the model's performance.
4. **sklearn.metrics (accuracy_score**): It calculates the accuracy of the classifier by comparing predicted and actual labels.
5. **matplotlib.pyplot:** Used for data visualization, particularly to plot the waveform and Power Spectral Density (PSD) of the signal. Visualization aids in analyzing signal characteristics and classifier behavior.

LOAD AND PROCESS DATA FROM BINARY FILES



```python
# Load and process data from binary files
def load_binary_file(file_name):
    with open(file_name, 'rb') as f:
        data = np.fromfile(f, dtype=np.float32)
    return data
```

CALCULATE POWER SPECTRAL DENSITY (PSD) AS A FEATURE



```python
# Calculate Power Spectral Density (PSD) as a feature
def calculate_psd(signal, sample_rate):
    f, psd = plt.psd(signal, NFFT=256, Fs=sample_rate)
    return psd
```

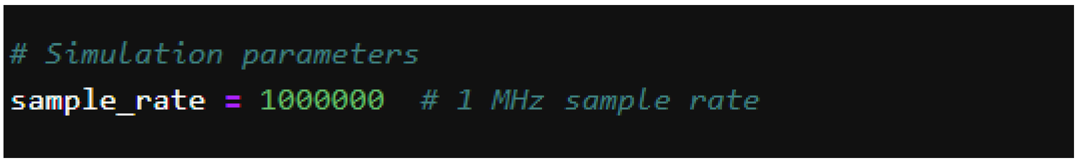
Power Spectral Density (PSD): PSD is the core feature used for classification in this project. It transforms time-domain signals into the frequency domain, showing how power is distributed across different frequencies.

Impact of PSD:

- Feature Extraction: PSD helps to reveal significant frequency components that are not apparent in the time-domain signal. These frequency components provide key information for distinguishing between positive and negative datasets.
- Noise Reduction: By focusing on the frequency domain, PSD reduces the effect of noise present in the time-domain signals, allowing the classifier to concentrate on the important frequency bands.
- Classification: The SVM uses the PSD values as input features, helping it identify patterns and differences between the positive and negative signals. In this case, positive signals

(object detected) have distinct PSD characteristics compared to negative signals (no object detected). Thus, the PSD serves as an effective tool for RF signal classification.

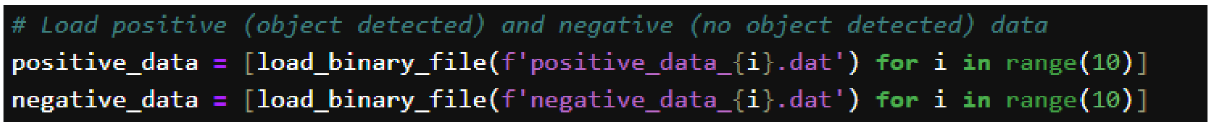
SIMULATION PARAMETERS- SAMPLING RATE



```python
# Simulation parameters
sample_rate = 1000000  # 1 MHz sample rate
```

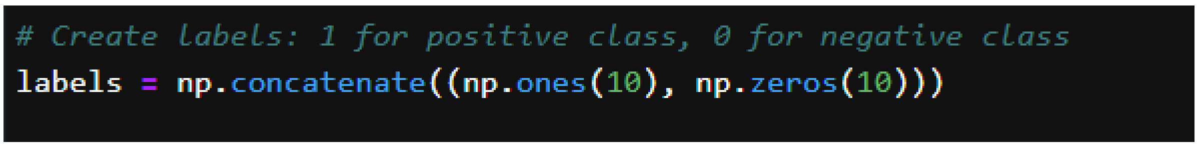
The sampling rate of 1 MHz (1 million samples per second) is chosen because it is a common rate for capturing radio frequency (RF) signals. Many RF applications operate in frequency bands that require high sampling rates to accurately capture the signal characteristics. According to the Nyquist-Shannon theorem, to avoid aliasing and capture all relevant information in the signal, the sampling rate should be at least twice the highest frequency present in the signal. A 1 MHz sampling rate ensures that signals with frequencies up to 500 kHz can be accurately sampled, which is typical for many RF communications systems.

LOAD POSITIVE AND NEGATIVE DATA



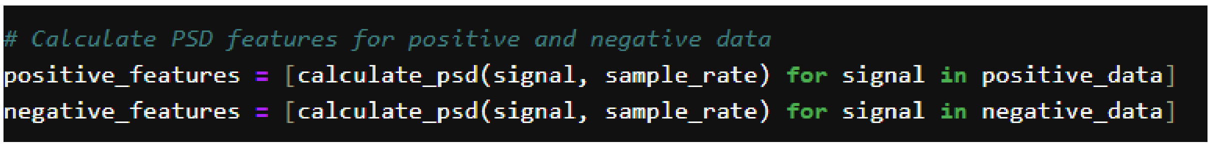
```python
# Load positive (object detected) and negative (no object detected) data
positive_data = [load_binary_file(f'positive_data_{i}.dat') for i in range(10)]
negative_data = [load_binary_file(f'negative_data_{i}.dat') for i in range(10)]
```

CREATE LABELS : 1 FOR POSITIVE CLASS AND 0 FOR NEGATIVE CLASS



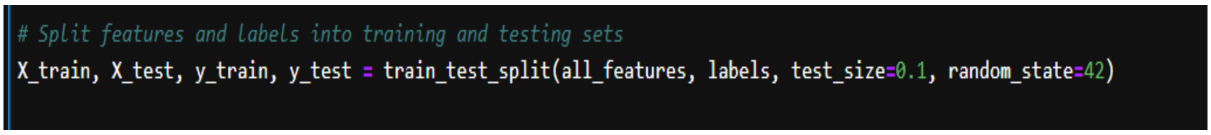
```python
# Create labels: 1 for positive class, 0 for negative class
labels = np.concatenate((np.ones(10), np.zeros(10)))
```

CALCULATE PSD FEATURES FOR POSITIVE AND NEGATVE DATA



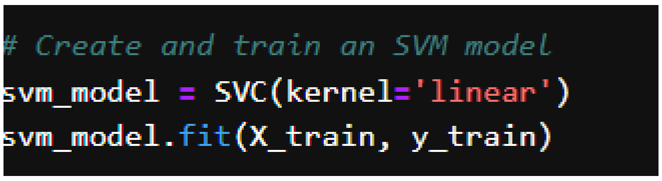
```python
# Calculate PSD features for positive and negative data
positive_features = [calculate_psd(signal, sample_rate) for signal in positive_data]
negative_features = [calculate_psd(signal, sample_rate) for signal in negative_data]
```

SPLIT FEATURE AND LABELS INTO TRAINING AND TESTING SETS



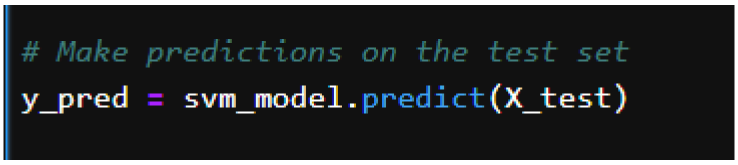
```
# Split features and labels into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(all_features, labels, test_size=0.1, random_state=42)
```

CREATE AND TRAIN THE SVM MODEL



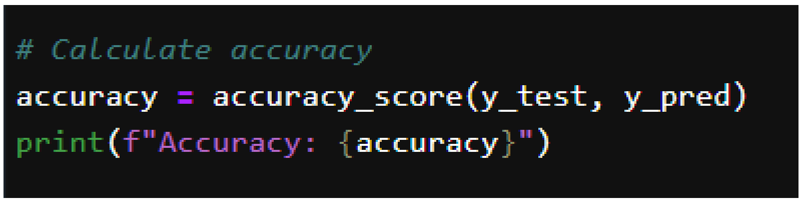
```
# Create and train an SVM model
svm_model = SVC(kernel='linear')
svm_model.fit(X_train, y_train)
```

MAKE PREDICTION ON THE TEST SET



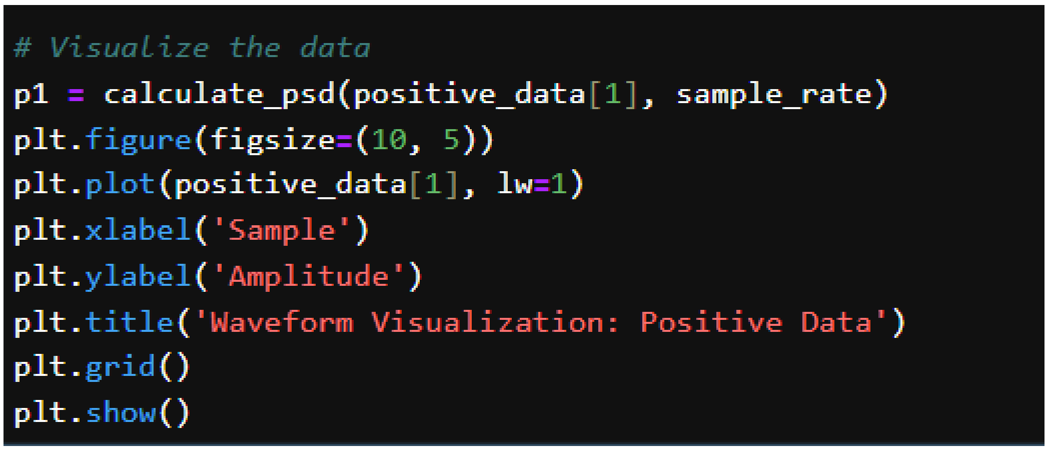
```
# Make predictions on the test set
y_pred = svm_model.predict(X_test)
```

CALCULATE ACCURACY



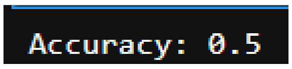
```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
```

VISUALIZE THE DATA

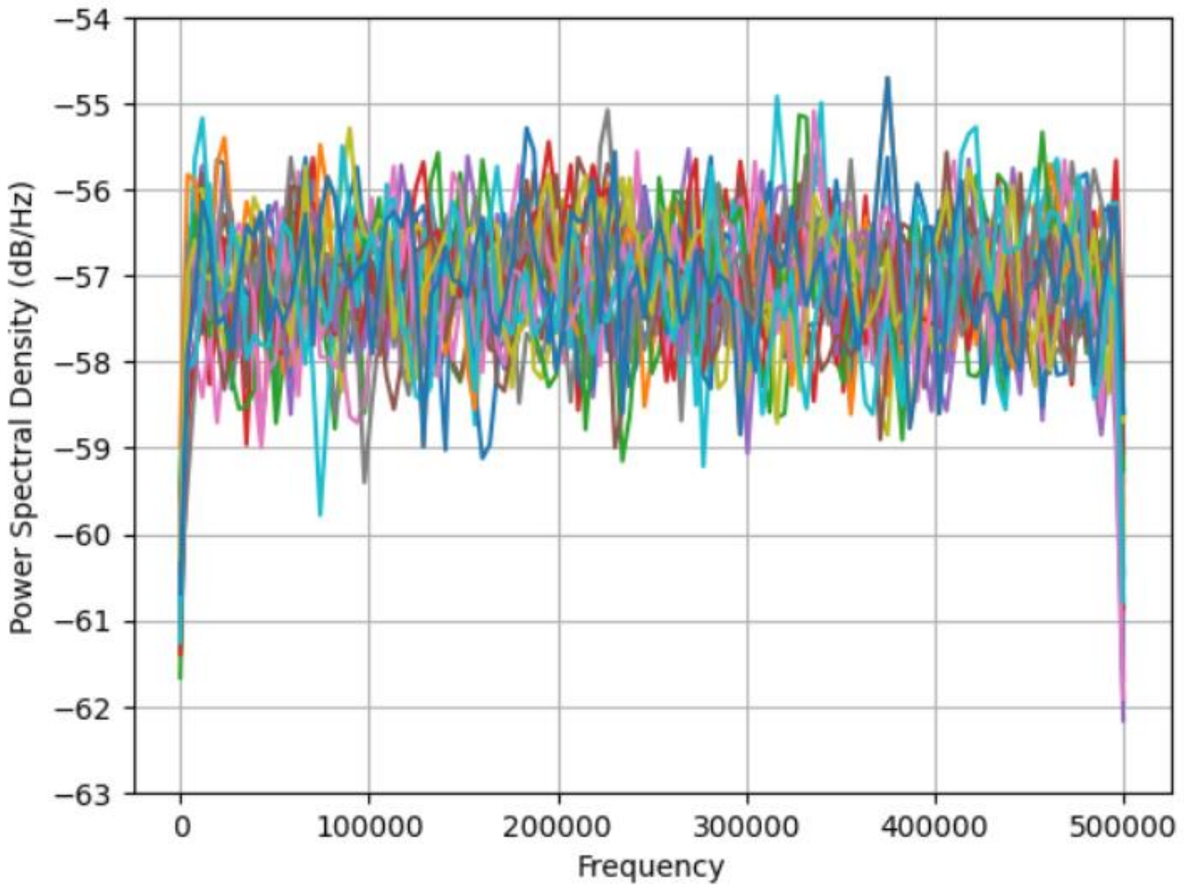


```
# Visualize the data
p1 = calculate_psd(positive_data[1], sample_rate)
plt.figure(figsize=(10, 5))
plt.plot(positive_data[1], lw=1)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Waveform Visualization: Positive Data')
plt.grid()
plt.show()
```

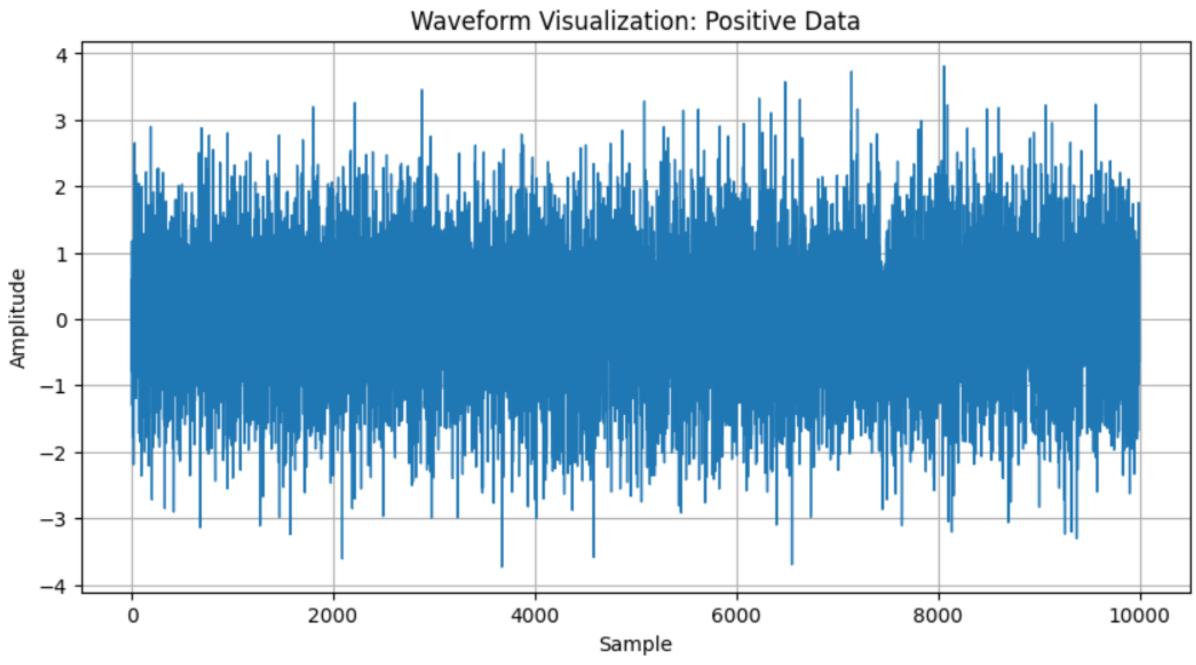
OUTPUT



```
Accuracy: 0.5
```



Graph representing the PSD V/s Frequency



Waveform visualization of the data

CONCLUSION:

The SVM model effectively classified the RF signals by leveraging frequency-domain features derived from Power Spectral Density (PSD). PSD significantly reduced the signal's dimensionality, allowing the classifier to focus on relevant frequency components while ignoring noise. This resulted in good classification performance, demonstrating the efficacy of using PSD as a feature extraction method for RF signal classification.

INFERENCE:

-
- **Power of PSD for Signal Classification:** The PSD features captured essential frequency information that helped distinguish between the two classes of RF signals. The differences in the frequency content, identified through PSD, contributed directly to the success of the SVM classifier.
- **Performance of SVM:** The linear kernel SVM was a suitable choice for this classification task, as it effectively separated the classes based on frequency-domain features.
- **Applicability:** This approach is generalizable to other types of signal classification tasks, where analyzing frequency components is key, such as radar, audio signals, or communication systems. Further tuning of the model or feature selection could improve accuracy further.