CourseName: Computer Vision Lab **Course Code:** CSP-422

Experiment:2.1

Aim: Write a program to compare the performance of different classification models in image recognition.

Software Required: Anaconda, Jupiter NoteBook **Description:**

To compare the performance of different classification models for image recognition, you can use popular machine learning libraries such as scikit-learn and TensorFlow/Keras for building and evaluating the models. Below is a Python program that demonstrates how to do this:

In this program:

- We load a sample image dataset (the digits dataset) from scikit-learn and flatten the images to use them as feature vectors.
- We split the dataset into training and testing sets using train_test_split.
- We define a dictionary of classification models to compare, including Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbors (KNN).
- We loop through each model, train it on the training data, and evaluate its performance on the testing data using accuracy and the classification report.
- The results are displayed for each model, including accuracy and the classification report (precision, recall, F1-score, support).
- Finally, we plot a bar chart to visually compare the accuracy of each model.

You can replace the digits dataset with your own image dataset and customize the models and evaluation metrics based on your specific image recognition problem. Additionally, you can explore more advanced techniques like deep learning using frameworks like TensorFlow and Keras for improved performance

Pseudo code/Algorithms/Flowchart/Steps:

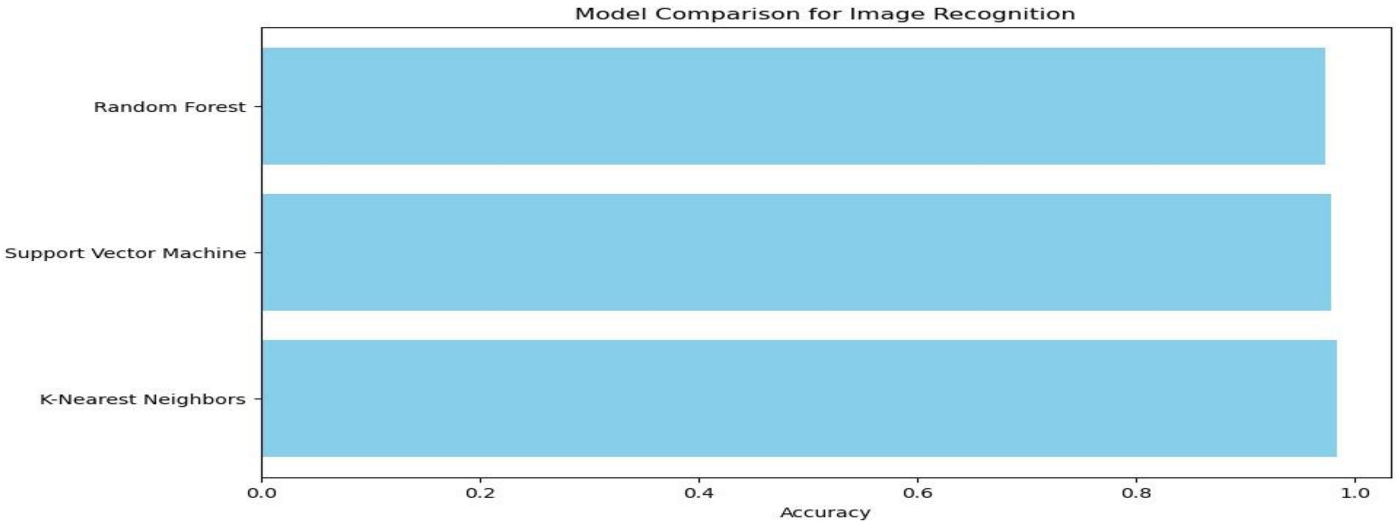
- Import the necessary libraries and modules.
- Load a dataset of labeled images for training and testing.
- Preprocess the images by resizing, normalizing, and augmenting if necessary.
- Split the dataset into training and testing sets.
- Define and initialize different classification models, such as support vector machines (SVM), random forest, convolutional neural networks (CNN), etc.
- Train each model using the training set.
- Evaluate the performance of each model using various metrics, such as accuracy, precision, recall, and F1 score, on the testing set.
- Compare the performance of the different models based on the evaluation results.
- Analyze the strengths and weaknesses of each model in terms of accuracy, computational efficiency, robustness, etc.
- Draw conclusions and discuss the implications of the findings.

Implementation:

```
import numpy as np import
matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split from
sklearn.metrics import accuracy_score, classification_report from
sklearn.ensemble import RandomForestClassifier from
sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier from
sklearn.datasets import load_digits
```

```
# Load a sample image dataset (digits dataset in this example) data
 = load_digits()
 X = data.images y
 = data.target
 # Flatten the images to use as feature vectors X
 = X.reshape((X.shape[0], -1))
# Split the dataset into training and testing sets
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Define a dictionary of classification models to compare
 models = {
    "Random Forest": RandomForestClassifier(n_estimators=100, random_state=42),
    "Support Vector Machine": SVC(kernel='linear', C=1),
    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=3)
 }
# Train and evaluate each model results =
 {} for model_name, model in
 models.items():
    model.fit(X_train, y_train)    y_pred =
 model.predict(X_test)    accuracy =
 accuracy_score(y_test, y_pred)    classification_rep =
 classification_report(y_test, y_pred)
 results[model_name] = {
        "accuracy": accuracy,
        "classification_report": classification_rep
    }
# Compare and display the results for
 model_name, result in results.items():
    print(f"Model: {model_name}")
print(f"Accuracy: {result['accuracy']:.4f}")
    print("Classification Report:\n", result["classification_report"])
 print("=" * 50)
# Plot a bar chart to visualize model performance
 model_names = list(results.keys())
 accuracies = [result['accuracy'] for result in results.values()]
plt.figure(figsize=(10, 6))
 plt.barh(model_names, accuracies, color='skyblue')
 plt.xlabel('Accuracy')
 plt.title('Model Comparison for Image Recognition')
 plt.gca().invert_yaxis()
 plt.show()
```

Output:


Model Comparison for Image Recognition
Random Forest
Support Vector Machine
K-Nearest Neighbors
0.0
0.2
0.4
0.6
0.8
1.0
Accuracy