

Image Classification with Support Vector Machine (SVM)

**Tanmay
Butta**

Date

16/04/24

Course title

Task-1 & 2

(Research Internship Task)



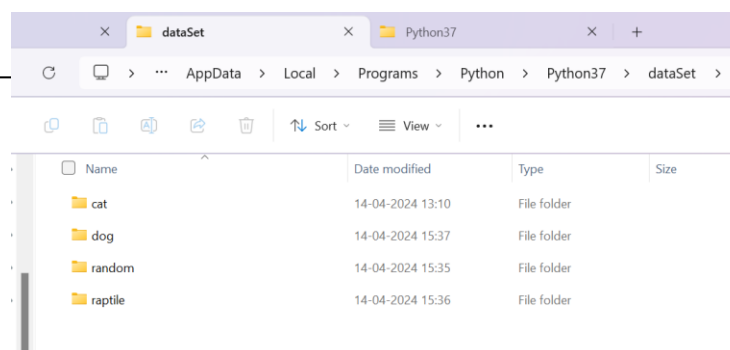
Objective

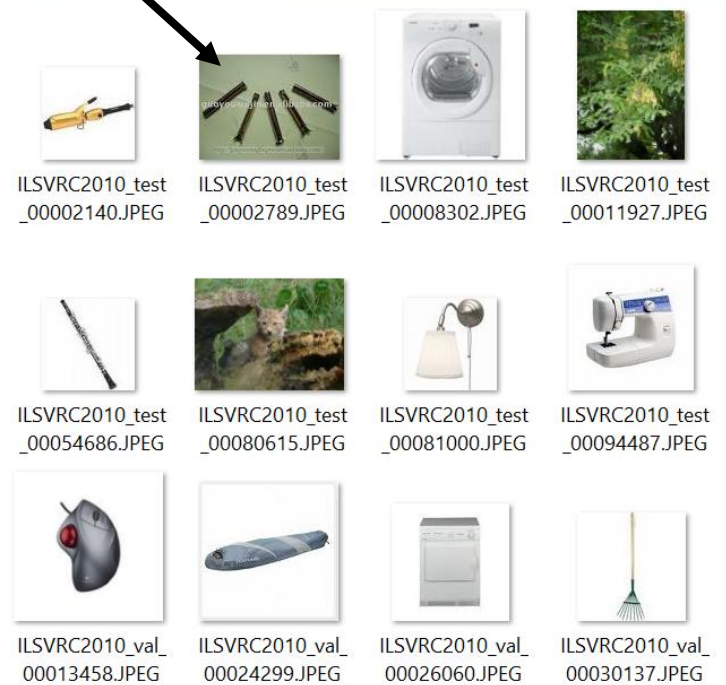
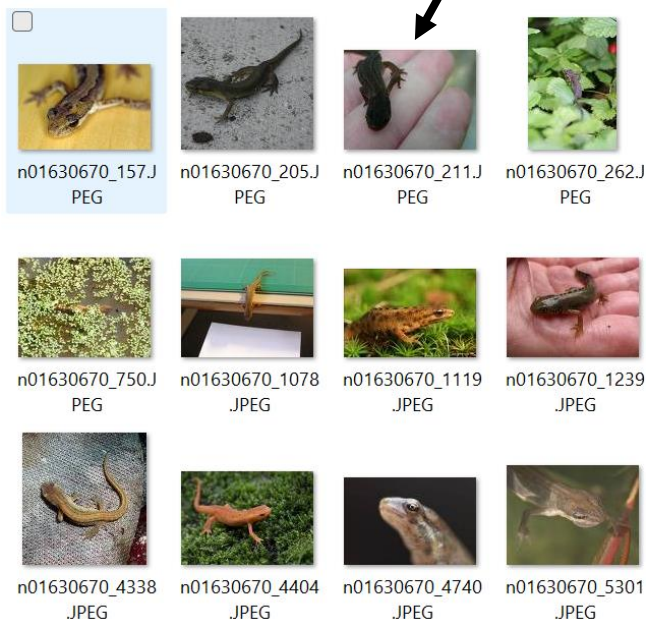
Task-1

1. Select a custom image dataset and split it into training and testing datasets.
2. Apply different data augmentation techniques, such as rotation, scaling, flipping, and cropping, on the training dataset.
3. Train a Support Vector Machine (SVM) model on the augmented training dataset.
4. Evaluate the performance of the SVM model on the testing dataset.
5. Compute the classification report metrics such as precision, recall, F1 score, and accuracy.

Select a custom image dataset and split it into training and testing datasets.

- For this experiment, a curated image dataset sourced from ImageNet was utilized.
- The dataset comprises approximately **400 images** distributed across distinct classes or categories.
- Specifically, the dataset encompasses representations of **cats, dogs, reptiles, and random items**, each category comprising **approximately 100 images**.
- This dataset selection ensures a diverse and comprehensive range of visual data for analysis and experimentation within the scope of the project.
- Out of the 400 images in the dataset, the distribution for testing and training is as follows:
- **Training: Approximately 300** images are used for training purposes.
- **Testing: Approximately 100** images are reserved for testing the trained model.
- This distribution ensures a substantial portion of the dataset is allocated for model training while also allowing for robust evaluation of the model's performance on unseen data during testing.





Apply different data augmentation techniques, such as rotation, scaling, flipping, and cropping, on the training dataset.

Data augmentation techniques, including rotation, scaling, flipping, and cropping, were applied using the “**ImageDataGenerator**” from “**tensorflow.keras.preprocessing.image**”.

- Parameters were set to rotate images randomly **within 40 degrees**, introduce **horizontal and vertical shifts up to 20%** of the total width and height
- applied a **shear range of 20%**
- **zoomed within a 20% range**,
- **horizontally flip images** with a **50%** probability
- used the '**nearest**' fill mode to handle empty pixels.
- Augmented batches of images were generated iteratively, diversifying the dataset with variations in orientation, position, and perspective.
- This augmentation process significantly increased the dataset's size and introduced variability, enhancing model robustness and generalization.
- By exposing the model to a broader range of examples, data augmentation aids in mitigating overfitting and improving performance on unseen data.

Overall, data augmentation is a crucial preprocessing step in training machine learning models, particularly for image classification tasks, as it promotes model resilience and performance in real-world scenarios.

NOTE

Create a "preview4" directory in the Python installation directory before running the code.
This directory will store augmented images, ensuring smooth execution and organized storage.

SOURCE CODE:

```
import numpy as np
import os
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import Image # Only needed for loading the image
# Load the image as a NumPy array
img =
np.array(Image.open("C:\\Users\\tanma\\AppData\\Local\\Programs\\Python\\Python37\\dataSet\\dog\\t
est_dog_3.jpg"))
# Define the data augmentation parameters
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
# Generate batches of augmented images
i = 0
for batch in datagen.flow(img[np.newaxis, ...], # Add extra dimension for batch
    batch_size=1,
    save_to_dir='preview4',
    save_prefix='val',
    save_format='jpeg'):

    i += 1
    if i > 20:
        break
```

OUTPUT:

Preview



val_0_1621.jpeg



val_0_1642.jpeg



val_0_2500.jpeg



val_0_2607.jpeg



val_0_3471.jpeg



val_0_4529.jpeg



val_0_4728.jpeg



val_0_7181.jpeg



val_0_7345.jpeg



val_0_7942.jpeg



val_0_7990.jpeg



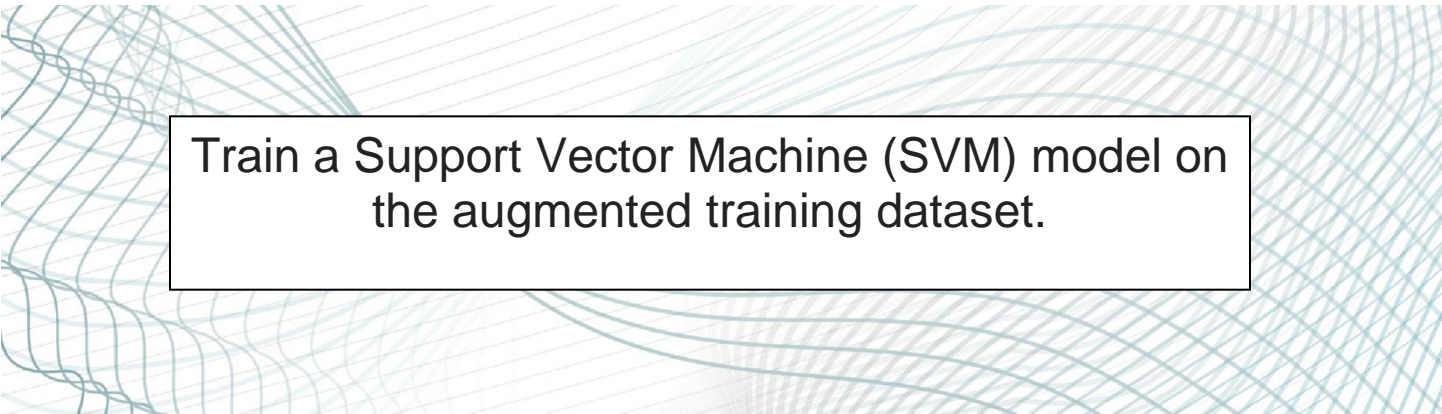
val_0_8331.jpeg



val_0_8438.jpeg



val_0_8756.jpeg



Train a Support Vector Machine (SVM) model on the augmented training dataset.

Training a Support Vector Machine (SVM) Model:

- For this experiment, a Support Vector Machine (SVM) model was trained on the augmented training dataset. The dataset consisted of images representing various classes such as **cats, random items, reptiles, and dogs**.
- Using the **scikit-learn library, specifically the SVC class**, the SVM model was initialized with appropriate hyperparameters. **A polynomial kernel was chosen with a regularization parameter (C) set to 1, and the 'auto' option was used for gamma.**
- The training dataset was then **split into training and testing sets using the “train_test_split” function from scikit-learn**. The SVM model was trained on the training dataset using the **fit method**.
- Once trained, the model was saved to a file **named "model3.sav"** using the pickle library for future use.

This process of training an SVM model forms a critical step in building a robust classification system capable of accurately identifying and classifying images from the given dataset.

RUNNING TUTORIAL

Import Libraries:

Import necessary libraries such as os, numpy, cv2 (OpenCV), matplotlib.pyplot, pickle, random, sklearn (for model selection and evaluation), and mpl_toolkits.mplot3d.

Modify the "**dir**" **variable** to reflect the path to the dataset stored on your PC. Ensure the dataset path is correctly specified to access the image data.

Run this code

```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import classification_report, accuracy_score,
precision_recall_fscore_support

from mpl_toolkits.mplot3d import Axes3D

dir = "C:\\Users\\tanma\\AppData\\Local\\Programs\\Python\\Python37\\dataSet"
data=[]
c=['cat','random','reptile','dog']

for cata in c:
    path=os.path.join(dir,cata)
    label=c.index(cata)

    for img in os.listdir(path):
        imgpath=os.path.join(path,img)
        p_imag=cv2.imread(imgpath,0)
        p_imag=cv2.resize(p_imag,(50,50))
        image=np.array(p_imag).flatten()
        data.append([image,label])

print(len(data))
pick_in=open("data1.pickle","wb")
pickle.dump(data,pick_in)
```



```
pick_in.close()
```

This code will make a data1.pickle file which have the data stored in it

After the creation of data1.pickle file

Run this code

```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import classification_report, accuracy_score,
precision_recall_fscore_support

from mpl_toolkits.mplot3d import Axes3D

pick_in=open("data1.pickle","rb")
data=pickle.load(pick_in)
pick_in.close()
random.shuffle(data)
fs=[]
labels=[]

for f, label in data:
    fs.append(f)
    labels.append(label)

xtrain,xtest,ytrain,ytest=train_test_split(fs,labels,test_size=0.80)

model = SVC(C=1, kernel="poly", gamma="auto")
model.fit(xtrain, ytrain)
pick=open("model3.sav","wb")
pickle.dump(model,pick)
pick.close()
```

This code will train the model and will save it as “model3.sav”.
Once the model is trained it is ready to predict the results.

Run This code for the Result and 3-D representation

```
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import pickle
import random
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, accuracy_score
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import classification_report, accuracy_score,
precision_recall_fscore_support

from mpl_toolkits.mplot3d import Axes3D
pick_in=open("data1.pickle","rb")
data=pickle.load(pick_in)
pick_in.close()
random.shuffle(data)
fs=[]
labels=[]

for f, label in data:
    fs.append(f)
    labels.append(label)

xtrain,xtest,ytrain,ytest=train_test_split(fs,labels,test_size=0.25)
pick=open("model3.sav","rb")
model=pickle.load(pick)
pick.close()
pre=model.predict(xtest)
ac=model.score(xtest,ytest)

c=['cat','random','reptile','dog']

print("Accuracy:",ac)
print("prediction:",c[pre[3]])
# Make predictions on the test set
y_pred = model.predict(xtest)

# Compute accuracy
accuracy = accuracy_score(ytest, y_pred)

# Generate classification report
class_report = classification_report(ytest, y_pred)
```

```

print("Accuracy:", accuracy)
print("Classification Report:")
print(class_report)

mypet = xtest[3].reshape(50, 50) # Reshape the image
mypet_rgb = cv2.cvtColor(mypet, cv2.COLOR_GRAY2RGB) # Convert grayscale to RGB
plt.imshow(mypet_rgb)
plt.show()

cm = confusion_matrix(ytest, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
ax.set_title("Confusion Matrix")
ax.set_xlabel("Predicted Label")
ax.set_ylabel("True Label")

plt.show()
precision, recall, f1_score, _ = precision_recall_fscore_support(ytest, y_pred,
average='weighted')
# Plot 3D visualization
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Scatter plot
ax.scatter(precision, recall, f1_score, c='blue', marker='o', s=100)

# Labels
ax.set_xlabel('Precision')
ax.set_ylabel('Recall')
ax.set_zlabel('F1 Score')
ax.set_title('3D Visualization of Classification Metrics')

plt.show()

```

Evaluate the performance of the SVM model on the testing dataset.

&

Compute the classification report metrics such as precision, recall, F1 score, and accuracy.

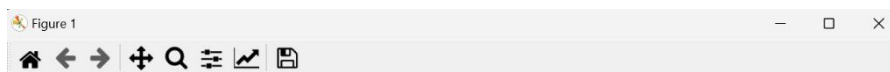
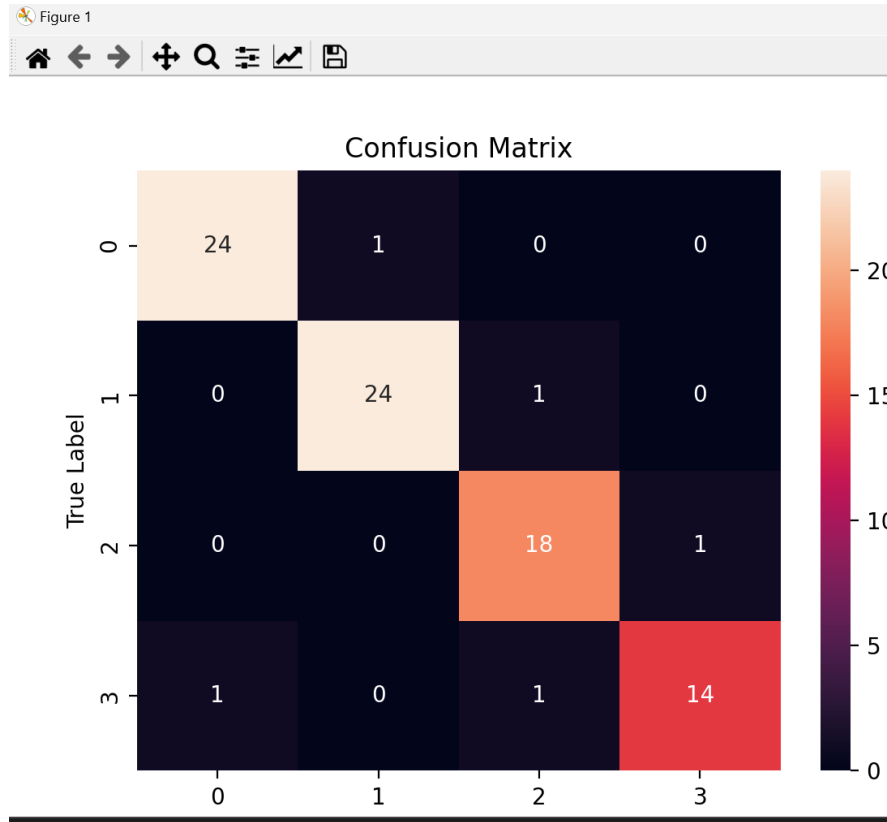
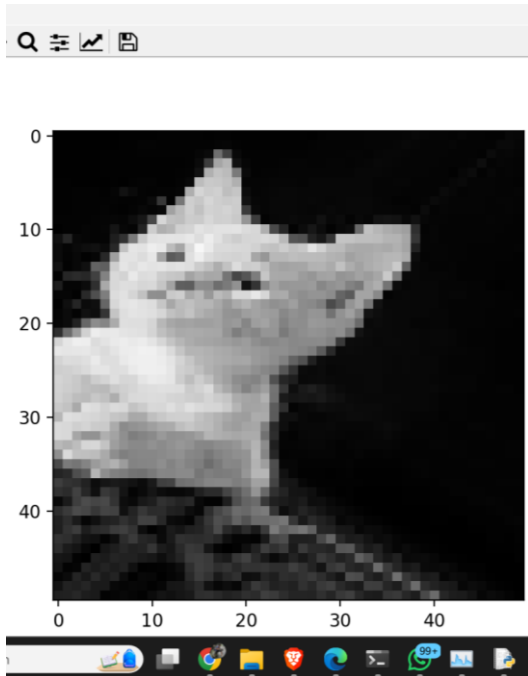
- **Accuracy Evaluation:** Assess the proportion of correctly classified instances against the total dataset, providing an **overall measure of model performance**.
- **Precision and Recall Analysis:** Calculate precision, representing the ratio of correctly **predicted positive instances** to all predicted positives, and recall, indicating the **ratio of correctly predicted positives to actual positives**, for each class in the dataset.
- **F1 Score Calculation:** Compute the F1 score, which is the **harmonic mean of precision and recall**, offering a balanced assessment of model performance.
- **Confusion Matrix Examination:** Analyze the confusion matrix to understand the model's classification patterns, **identifying true positives, true negatives, false positives, and false negatives** for each class.
- **Insight Generation:** Use these metrics to gain insights into the SVM model's effectiveness in accurately classifying images, helping identify strengths and areas for improvement.

OUTPUT 1

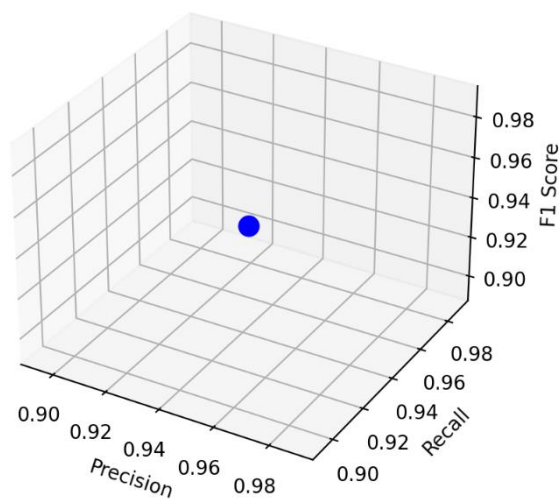
```
PS C:\Users\tanma\AppData\Local\Programs\Python\Python37> python -u "c:\Users\tanma\AppData\Local\Programs\Python\Python37\newdataset.py"
Accuracy: 0.9411764705882353
prediction: cat
Accuracy: 0.9411764705882353
Classification Report:
      precision    recall  f1-score   support

     0       0.96       0.96       0.96         25
     1       0.96       0.96       0.96         25
     2       0.90       0.95       0.92         19
     3       0.93       0.88       0.90         16

 accuracy          0.94         85
 macro avg       0.94       0.94       0.94         85
 weighted avg    0.94       0.94       0.94         85
```

3D Visualization of Classification Metrics



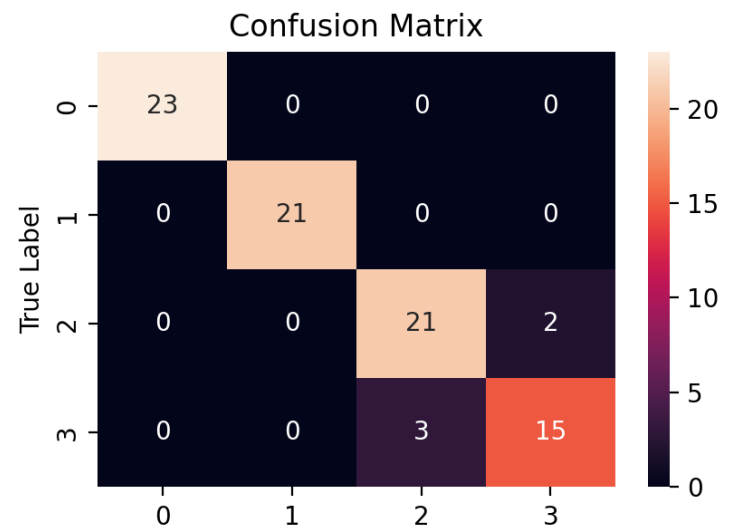
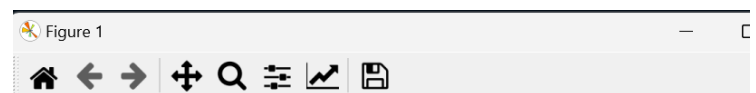
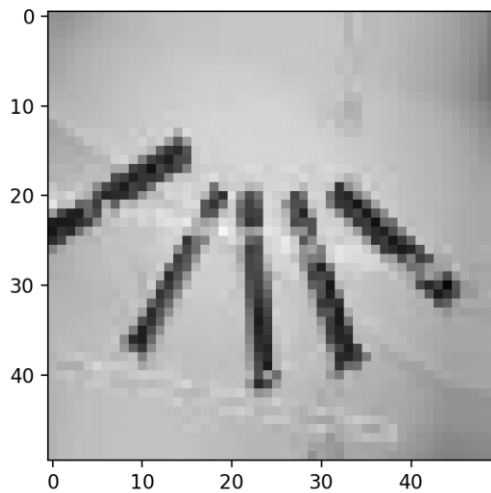
OUTPUT 2

```

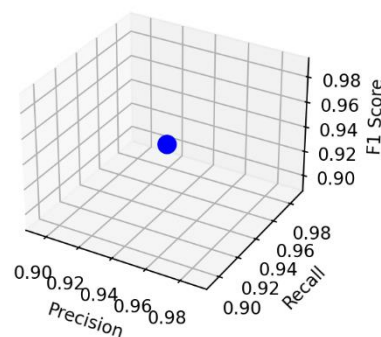
PS C:\Users\tanma\AppData\Local\Programs\Python\Python37> python -
Accuracy: 0.9411764705882353
prediction: random
Accuracy: 0.9411764705882353
Classification Report:

```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	23
1	1.00	1.00	1.00	21
2	0.88	0.91	0.89	23
3	0.88	0.83	0.86	18
accuracy			0.94	85
macro avg	0.94	0.94	0.94	85
weighted avg	0.94	0.94	0.94	85



3D Visualization of Classification Metrics



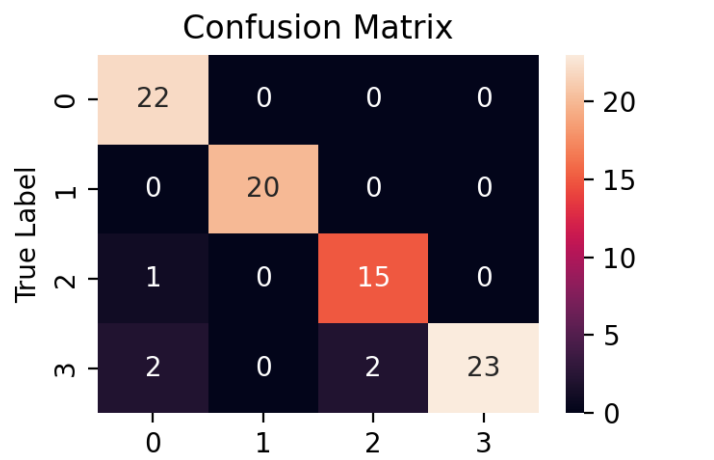
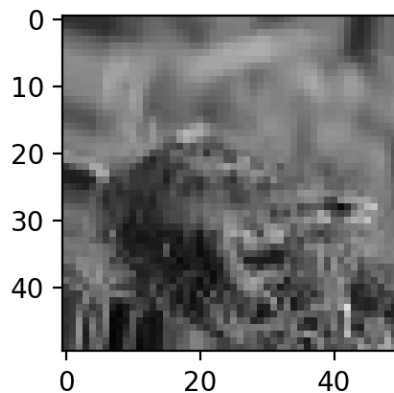
OUTPUT 3

```

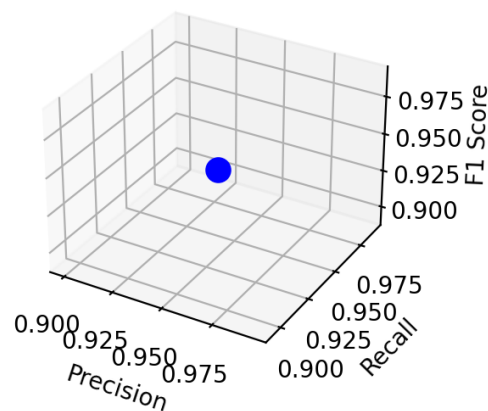
PS C:\Users\tanma\AppData\Local\Programs\Python\Python37> pyth
Accuracy: 0.9411764705882353
prediction: reptile
Accuracy: 0.9411764705882353
Classification Report:

```

	precision	recall	f1-score	support
0	0.88	1.00	0.94	22
1	1.00	1.00	1.00	20
2	0.88	0.94	0.91	16
3	1.00	0.85	0.92	27
accuracy			0.94	85
macro avg	0.94	0.95	0.94	85
weighted avg	0.95	0.94	0.94	85



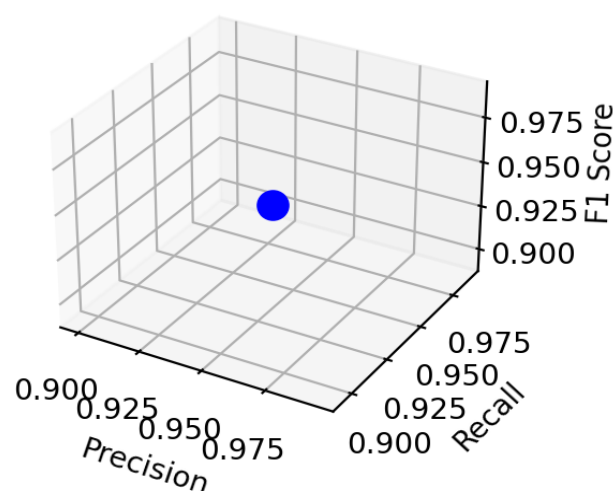
3D Visualization of Classification Metrics




Develop a 3-D illustration of the achieved results from Task-1.

- **Visualization Objective:** Develop a 3-D illustration to depict the performance metrics of the SVM model obtained from Task-1.
- **Representation of Metrics:** Precision, recall, and F1 score are plotted along different axes of the 3-D space, enabling easy comparison and analysis.
- **Scatter Plot Visualization:** Data points corresponding to precision, recall, and F1 score are depicted as individual points in the 3-D space, with each point representing a specific class or category.
- **Inclusion in Project Report:** The 3-D illustration is included in the project report to visually communicate the SVM model's performance on the augmented dataset, enhancing the clarity and comprehensibility of the findings for stakeholders.

3D Visualization of Classification Metrics





Interpret and provide insights into the performance of the SVM model on the augmented dataset in the illustration.

Interpretation and Insights into SVM Model Performance:

- **Class-Specific Analysis:** The 3-D illustration allows for a detailed examination of precision, recall, and F1 score across different classes. By analyzing these metrics individually, we can assess the model's performance on a per-class basis.
- **Identifying Class Imbalances:** Disparities in precision, recall, or F1 score between classes indicate potential class imbalances. **Classes with lower scores** may require further investigation to understand the reasons behind the discrepancies.
- **Detection of Outliers:** Outliers in the 3-D visualization, representing classes with unusually high or low performance metrics, can provide valuable insights. These outliers may highlight areas where the **model excels or struggles, warranting closer scrutiny.**
- **Pattern Recognition:** Patterns or trends in the distribution of data points across the 3-D space can reveal underlying characteristics of the dataset and the model's behavior. Understanding these patterns helps in **refining the model** and **optimizing its performance.**
- **Improvement Opportunities:** By interpreting the insights, we can identify opportunities for model improvement. **Addressing class imbalances, mitigating outliers, and refining the model parameters** are potential strategies for enhancing overall performance.
- **Enhanced Decision-Making:** Armed with a comprehensive understanding of the SVM model's performance, stakeholders can make informed decisions regarding model deployment, further experimentation, and potential areas of focus for future research and development.