First 10 rows of this project

```python
pa_df=pd.read_csv("teen_phone_addiction_dataset.csv")
pa_df.head(11)
# pa_df.tail(5)
```

	ID	Name	Age	Gender	Location	School_Grade	Daily_Usage_Hours	Sleep_Hours	Academic_Performance	Social_Interactions	...	Screen_Time_Before_Bed	Ph
0	1	Shannon Francis	13	Female	Hansonfort	9th	4.0	6.1	78	5	...	1.4	
1	2	Scott Rodriguez	17	Female	Theodorefort	7th	5.5	6.5	70	5	...	0.9	
2	3	Adrian Knox	13	Other	Lindseystad	11th	5.8	5.5	93	8	...	0.5	
3	4	Brittany Hamilton	18	Female	West Anthony	12th	3.1	3.9	78	8	...	1.4	
4	5	Steven Smith	14	Other	Port Lindsaystad	9th	2.5	6.7	56	4	...	1.0	
5	6	Mary Adams	13	Female	East Angelachester	10th	3.9	6.3	89	3	...	1.1	
6	7	Hailey Moses	16	Male	North Jeffrey	11th	6.3	6.7	89	3	...	0.8	
7	8	Veronica Marshall	13	Other	Jenniferport	10th	5.1	6.1	70	2	...	1.0	
8	9	Edward Avila	13	Male	Leebury	8th	3.0	9.1	79	0	...	0.9	
9	10	James Carter	18	Other	Prestonview	11th	3.9	5.8	89	8	...	0.9	

Last 5 rows of this project

```python
# pa_df.head(11)
pa_df.tail(5)
```

	ID	Name	Age	Gender	Location	School_Grade	Daily_Usage_Hours	Sleep_Hours	Academic_Performance	Social_Interactions	...	Screen_Time_Before_Bed
2995	2996	Jesus Yates	16	Female	New Jennifer	12th	3.9	6.4	53	4	...	0.3
2996	2997	Bethany Murray	13	Female	Richardport	8th	3.6	7.3	93	5	...	0.9
2997	2998	Norman Hughes	14	Other	Rebeccaton	7th	3.2	6.5	98	1	...	0.2
2998	2999	Barbara Hinton	17	Female	Ramirezmouth	9th	6.7	7.5	67	3	...	1.6
2999	3000	Curtis Johnson	17	Male	Lake Alexander	10th	3.5	6.9	79	4	...	0.6

5 rows × 25 columns

Checking missing values

```python
[3]: print("Checking for missing values:")
     print(pa_df.isnull().sum())
```

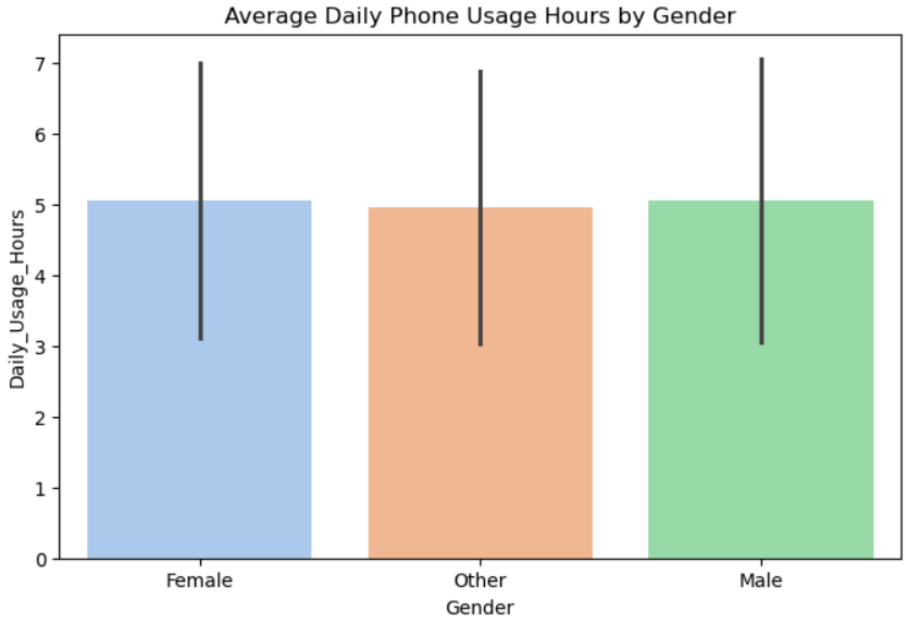
```
Checking for missing values:
ID                        0
Name                      0
Age                       0
Gender                    0
Location                  0
School_Grade              0
Daily_Usage_Hours         0
Sleep_Hours               0
Academic_Performance      0
Social_Interactions       0
Exercise_Hours            0
Anxiety_Level             0
Depression_Level          0
Self_Esteem               0
Parental_Control          0
Screen_Time_Before_Bed    0
Phone_Checks_Per_Day      0
Apps_Used_Daily           0
Time_on_Social_Media      0
Time_on_Gaming            0
Time_on_Education         0
Phone_Usage_Purpose       0
Family_Communication      0
Weekend_Usage_Hours       0
Addiction_Level           0
dtype: int64
```

All of the columns

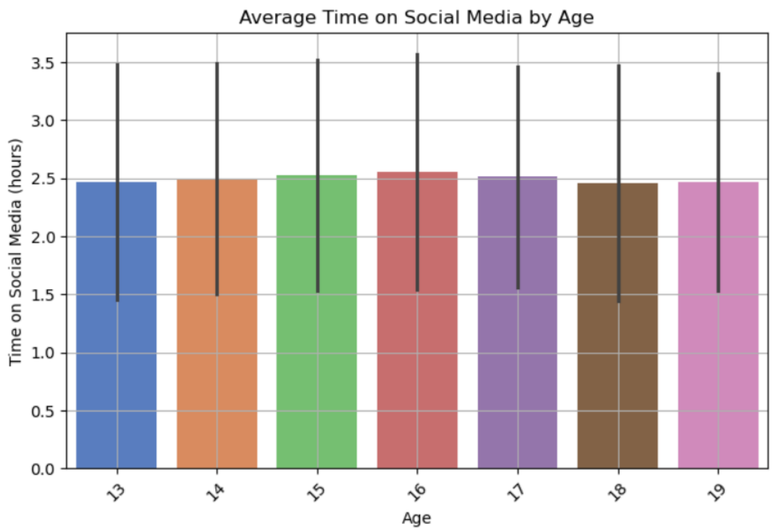
```
pa_df.columns
# pa_df.describe()
```

```
Index(['ID', 'Name', 'Age', 'Gender', 'Location', 'School_Grade',
       'Daily_Usage_Hours', 'Sleep_Hours', 'Academic_Performance',
       'Social_Interactions', 'Exercise_Hours', 'Anxiety_Level',
       'Depression_Level', 'Self_Esteem', 'Parental_Control',
       'Screen_Time_Before_Bed', 'Phone_Checks_Per_Day', 'Apps_Used_Daily',
       'Time_on_Social_Media', 'Time_on_Gaming', 'Time_on_Education',
       'Phone_Usage_Purpose', 'Family_Communication', 'Weekend_Usage_Hours',
       'Addiction_Level'],
      dtype='object')
```

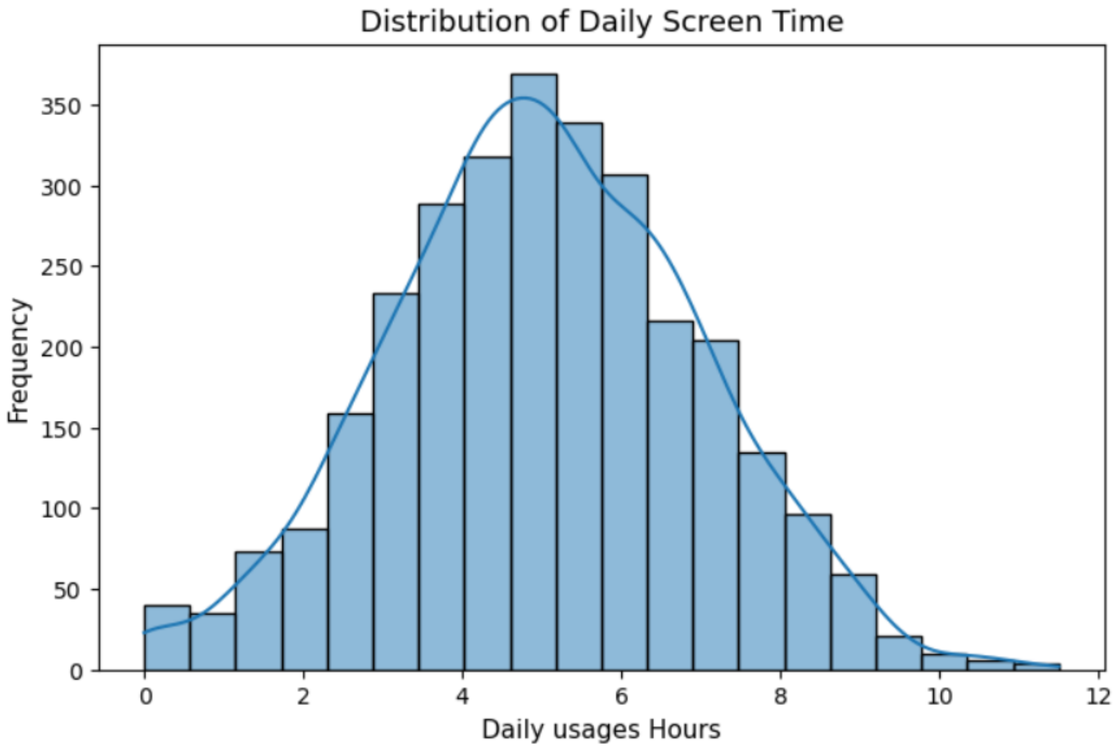
Average phone usage by gender



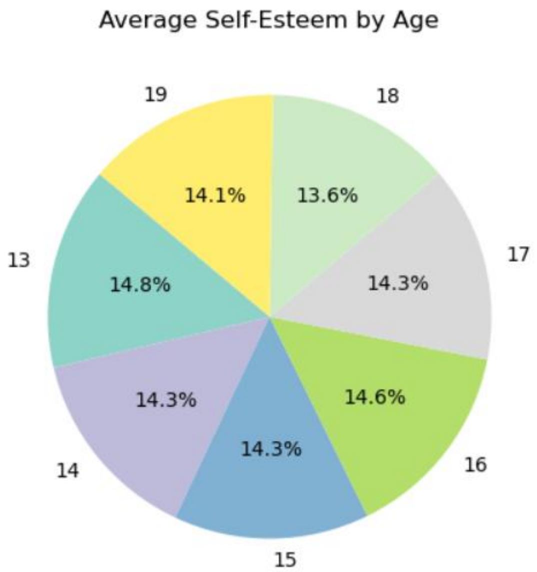
Average time on social media



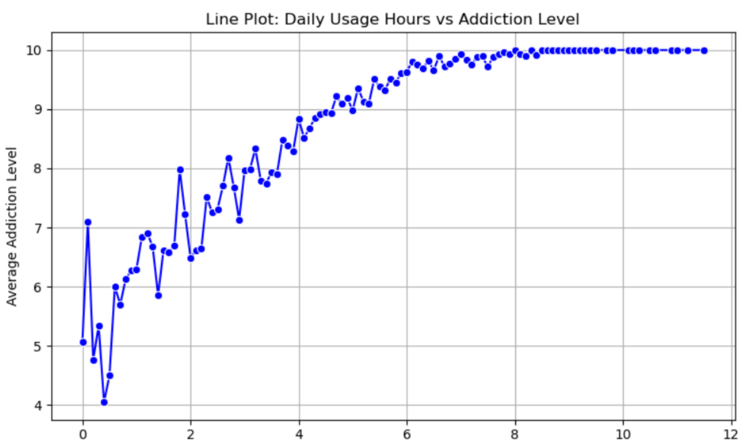
Daily screen time



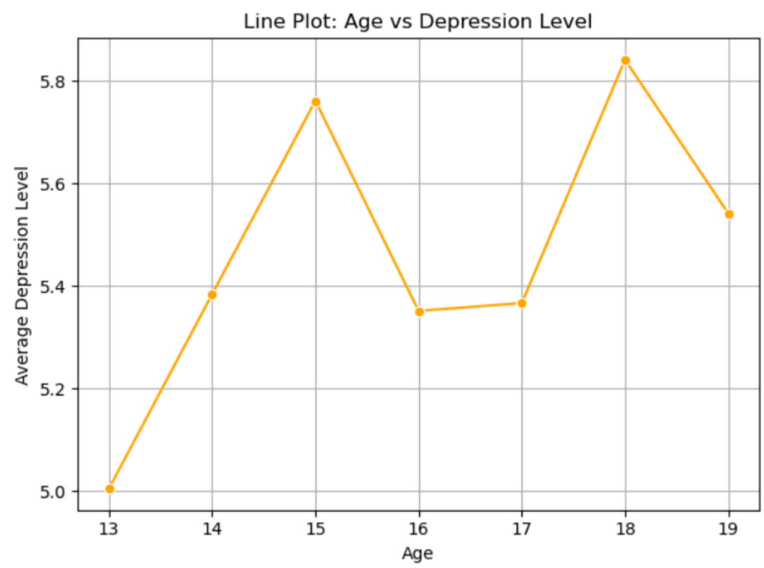
Average self-respect by age



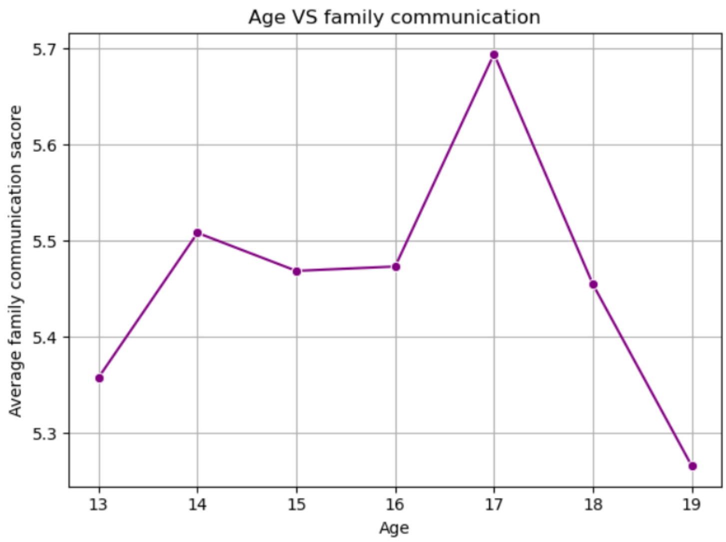
Daily usage hours vs addiction level



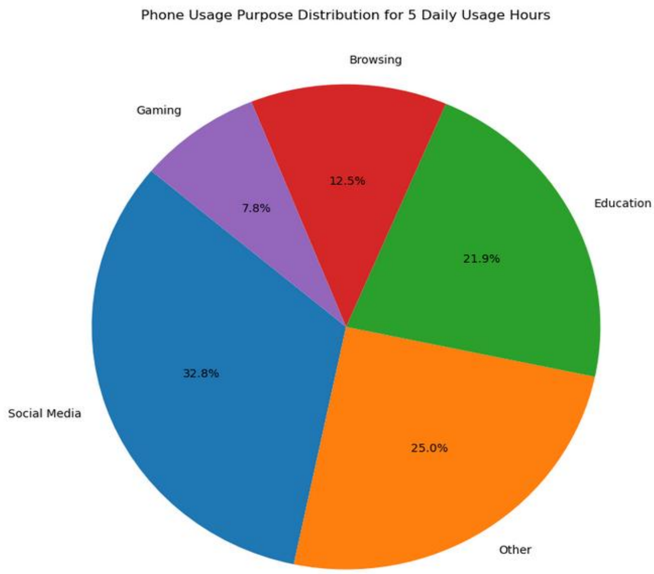
Age vs depression level



Age vs family communication

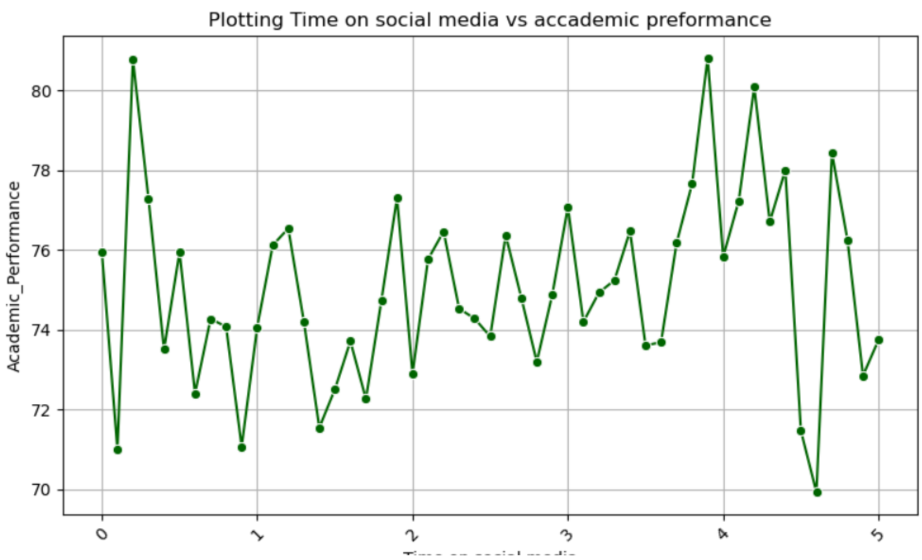


Phone usage hours for 5days

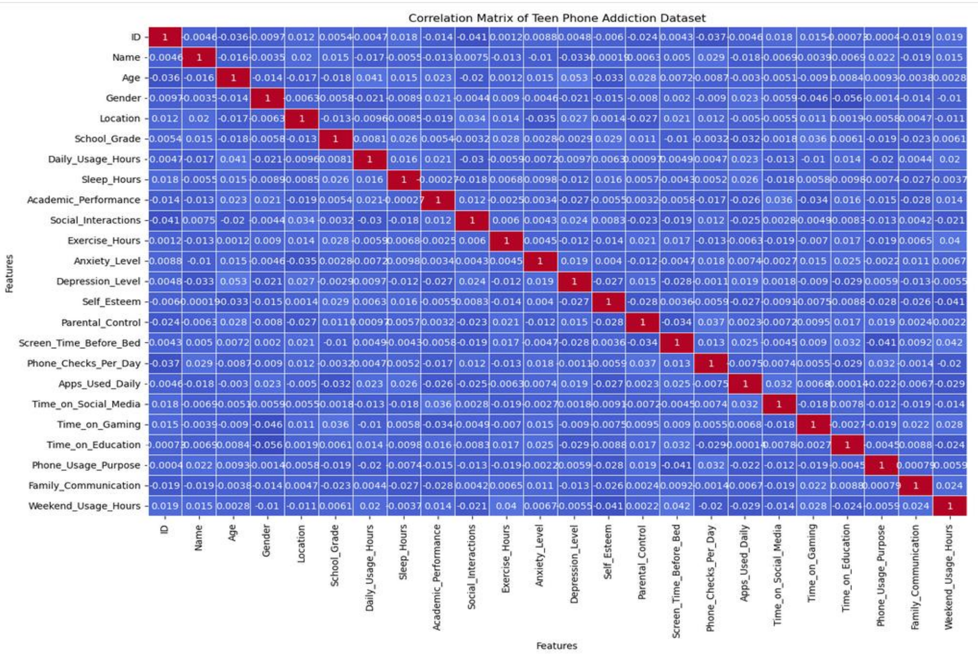


Exercise hours vs addiction level

Social media vs academic performance



Confusion matrix



Some libraries and ml models

```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,LabelEncoder
from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
import numpy as np
```

```python
# ML Models
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from xgboost import XGBClassifier
```

```python
#Train & Evaluate All Models
results = {}
for name, model in models.items():
    print(f"\nTraining {name}...")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    print(f"Accuracy: {acc:.4f}")
    print("Confusion Matrix:")
    print(confusion_matrix(y_test, y_pred))
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
    results[name] = acc
```

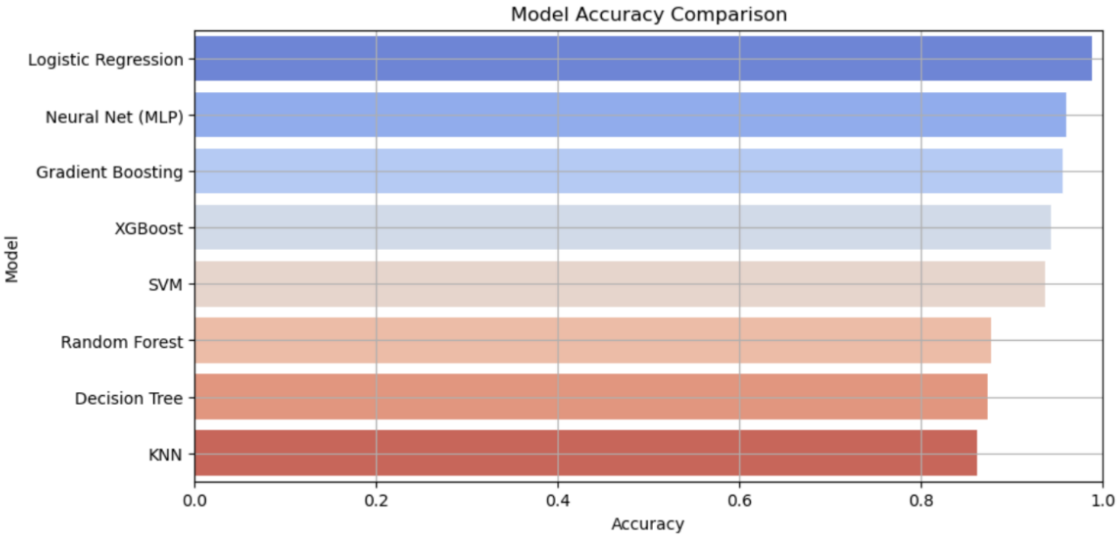
Logistic regressing accuracy score

```
Training Logistic Regression...
Accuracy: 0.9883
Confusion Matrix:
[[  2   3   0]
 [  0 108   3]
 [  0   1 483]]
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.40      0.57         5
           1       0.96      0.97      0.97       111
           2       0.99      1.00      1.00       484

    accuracy                           0.99       600
   macro avg       0.99      0.79      0.85       600
weighted avg       0.99      0.99      0.99       600
```

All models score

```
Model Accuracy Summary:
                 Model  Accuracy
0  Logistic Regression  0.988333
7      Neural Net (MLP)  0.960000
5     Gradient Boosting  0.955000
6               XGBoost  0.943333
4                   SVM  0.936667
3         Random Forest  0.876667
2         Decision Tree  0.873333
1                   KNN  0.861667
```

Model accuracy plotting



Build Neural Network

```
model=Sequential([Dense(128,activation='relu',input_dim=X_train.shape[1]),
                  BatchNormalization(),
                  Dropout(0.4),
                  Dense(64,activation='relu'),
                  BatchNormalization(),
                  Dropout(0.3),
                  Dense(32, activation='relu'),
                  Dropout(0.2),
                  Dense(3, activation='softmax')
                 ])
```

Last few epochs

```
Epoch 35/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9591 - loss: 0.1061 - val_accuracy: 0.9708 - val_loss: 0.0762
Epoch 36/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9612 - loss: 0.1126 - val_accuracy: 0.9771 - val_loss: 0.0725
Epoch 37/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9646 - loss: 0.1057 - val_accuracy: 0.9604 - val_loss: 0.0809
Epoch 38/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9669 - loss: 0.0892 - val_accuracy: 0.9688 - val_loss: 0.0733
Epoch 39/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9707 - loss: 0.0890 - val_accuracy: 0.9771 - val_loss: 0.0616
Epoch 40/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9631 - loss: 0.0897 - val_accuracy: 0.9708 - val_loss: 0.0620
Epoch 41/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9676 - loss: 0.0892 - val_accuracy: 0.9750 - val_loss: 0.0649
Epoch 42/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9741 - loss: 0.0815 - val_accuracy: 0.9688 - val_loss: 0.0782
Epoch 43/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9680 - loss: 0.1033 - val_accuracy: 0.9667 - val_loss: 0.0803
Epoch 44/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9666 - loss: 0.0865 - val_accuracy: 0.9708 - val_loss: 0.0706
Epoch 45/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9742 - loss: 0.0701 - val_accuracy: 0.9646 - val_loss: 0.0645
Epoch 46/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9683 - loss: 0.0865 - val_accuracy: 0.9750 - val_loss: 0.0659
Epoch 47/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9646 - loss: 0.0995 - val_accuracy: 0.9688 - val_loss: 0.0728
Epoch 48/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9758 - loss: 0.0659 - val_accuracy: 0.9667 - val_loss: 0.0704
Epoch 49/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9690 - loss: 0.0815 - val_accuracy: 0.9729 - val_loss: 0.0677
Epoch 50/50
120/120 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9724 - loss: 0.0829 - val_accuracy: 0.9750 - val_loss: 0.0657
```

Confusion matrix and classification report

[60]:

```python
print("Confusion matrics:")
print(confusion_matrix(y_test,y_pred))
print("\nClassification report:")
print(classification_report(y_test,y_pred))
```

```
Confusion matrics:
[[  0   5   0]
 [  0  99  12]
 [  0   6 478]]

Classification report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         5
           1       0.90      0.89      0.90       111
           2       0.98      0.99      0.98       484

    accuracy                           0.96       600
   macro avg       0.63      0.63      0.63       600
weighted avg       0.95      0.96      0.96       600
```

Plotting over Epochs