Department of Applied Mathematics and Statistics

INTRODUCTION TO MACHINE LEARNING

MINI PROJECT.4
GROUP 2

Lecturer : Mr. TOCH Sopheak

01

OUR
CREATIVE TEAM(GROUP 2)

KANG PENGKHEANG
E20210527

CHAN SOPHARA
E20211081

CHHORN SOLITA
E20210537

EN SREY TOCH
E20210085

CHUM RATANAKCHENTRIA
E20211542

TABLE OF CONTENTS

01 About project

02 datasets

03 training model

04 testing model

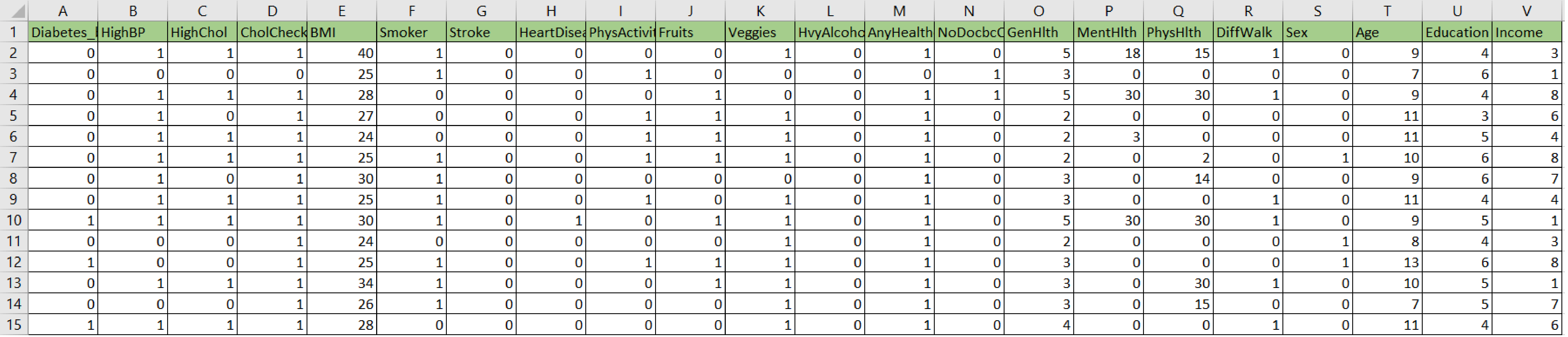
ABOUT
OUR MINI PROJECT.4

Diabetes is a chronic condition affecting millions worldwide. Early detection and intervention are crucial for managing diabetes and preventing complications. Machine learning techniques offer a promising avenue for predicting diabetes risk based on various factors such as demographics, medical history, and lifestyle choices. In this project, we aim to divide your dataset into three subsets: training set (60 %), validation set (20 %), and test set (20 %)and use the validation set to find the best possible neural network model and evaluate it on the test set.

DATASET

For our datasets we got from kaggle which related about diabetics .

diabetes_binary_health_indicators_BRFSS2015.csv



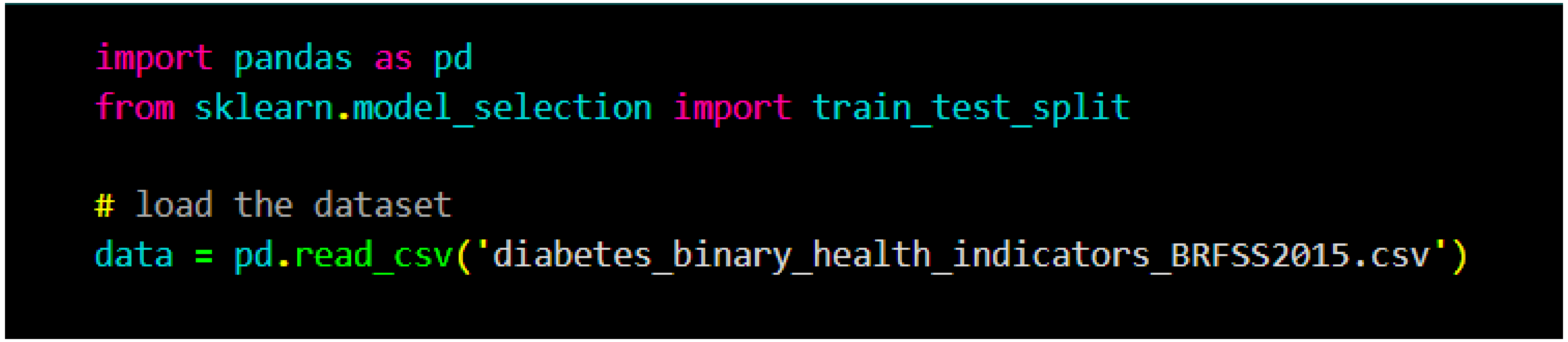
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	Diabetes_	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDisea	PhysActivit	Fruits	Veggies	HvyAlcoho	AnyHealth	NoDocbcC	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
2	0	1	1	1	40	1	0	0	0	0	1	0	1	0	5	18	15	1	0	9	4	3
3	0	0	0	0	25	1	0	0	1	0	0	0	0	1	3	0	0	0	0	7	6	1
4	0	1	1	1	28	0	0	0	0	1	0	0	1	1	5	30	30	1	0	9	4	8
5	0	1	0	1	27	0	0	0	1	1	1	0	1	0	2	0	0	0	0	11	3	6
6	0	1	1	1	24	0	0	0	1	1	1	0	1	0	2	3	0	0	0	11	5	4
7	0	1	1	1	25	1	0	0	1	1	1	0	1	0	2	0	2	0	1	10	6	8
8	0	1	0	1	30	1	0	0	0	0	0	0	1	0	3	0	14	0	0	9	6	7
9	0	1	1	1	25	1	0	0	1	0	1	0	1	0	3	0	0	1	0	11	4	4
10	1	1	1	1	30	1	0	1	0	1	1	0	1	0	5	30	30	1	0	9	5	1
11	0	0	0	1	24	0	0	0	0	0	1	0	1	0	2	0	0	0	1	8	4	3
12	1	0	0	1	25	1	0	0	1	1	1	0	1	0	3	0	0	0	1	13	6	8
13	0	1	1	1	34	1	0	0	0	1	1	0	1	0	3	0	30	1	0	10	5	1
14	0	0	0	1	26	1	0	0	0	0	1	0	1	0	3	0	15	0	0	7	5	7
15	1	1	1	1	28	0	0	0	0	0	1	0	1	0	4	0	0	1	0	11	4	6

253681 x 22

Here is the link to our dataset in kaggle:

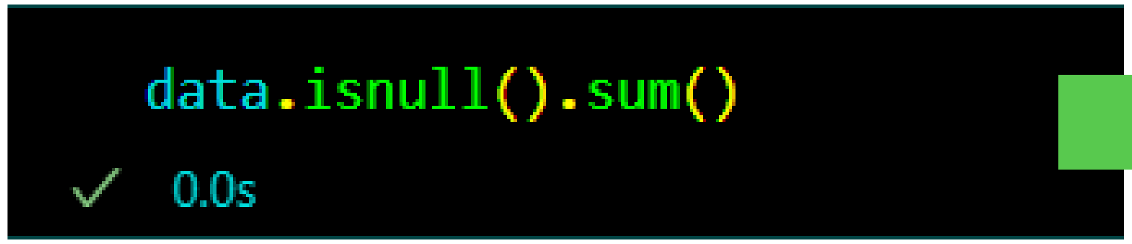
https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

IMPORT LIBRARIES

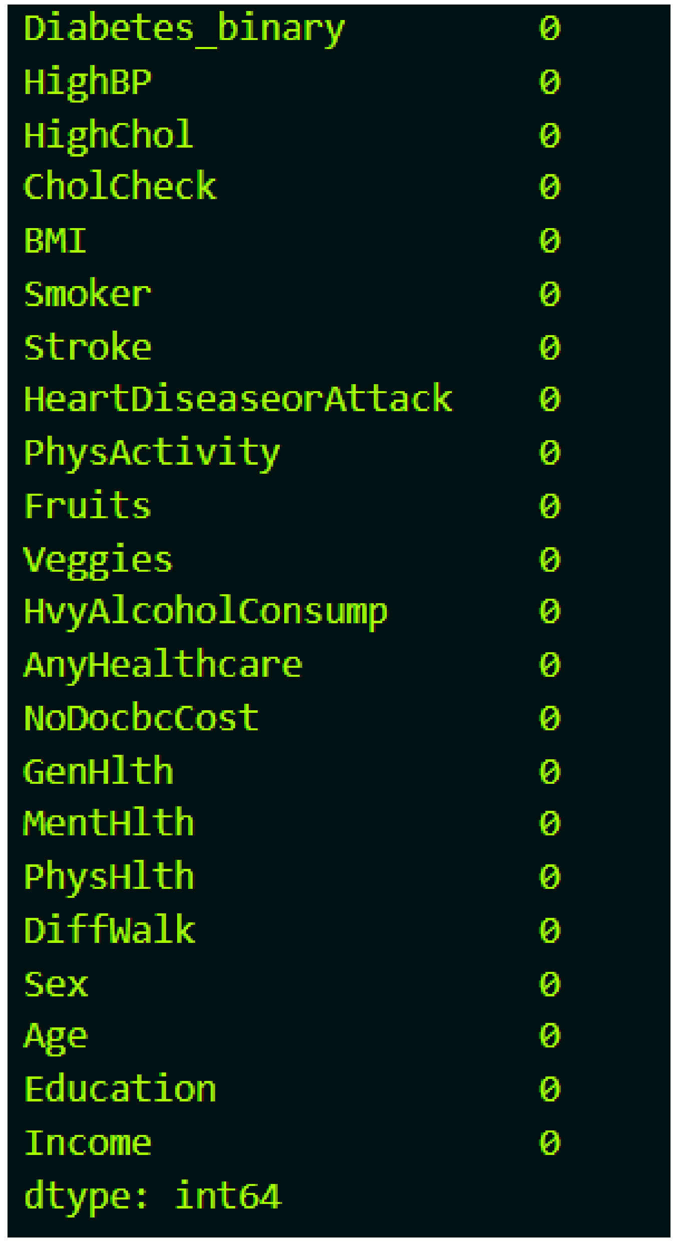


```
import pandas as pd
from sklearn.model_selection import train_test_split

# load the dataset
data = pd.read_csv('diabetes_binary_health_indicators_BRFSS2015.csv')
```

CHECK MISSING VALUE

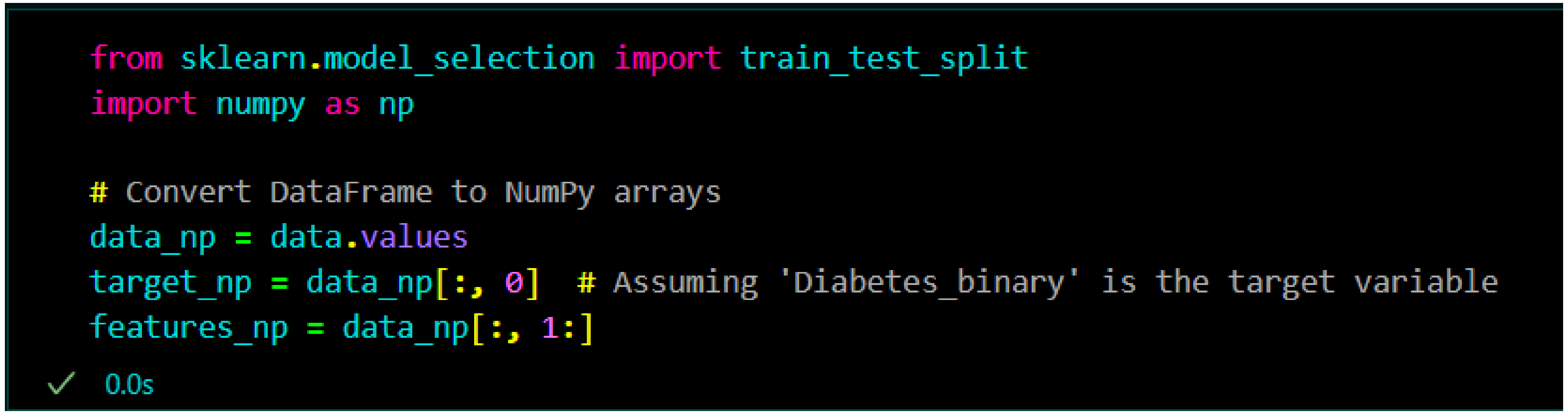


```
data.isnull().sum()
✓ 0.0s
```



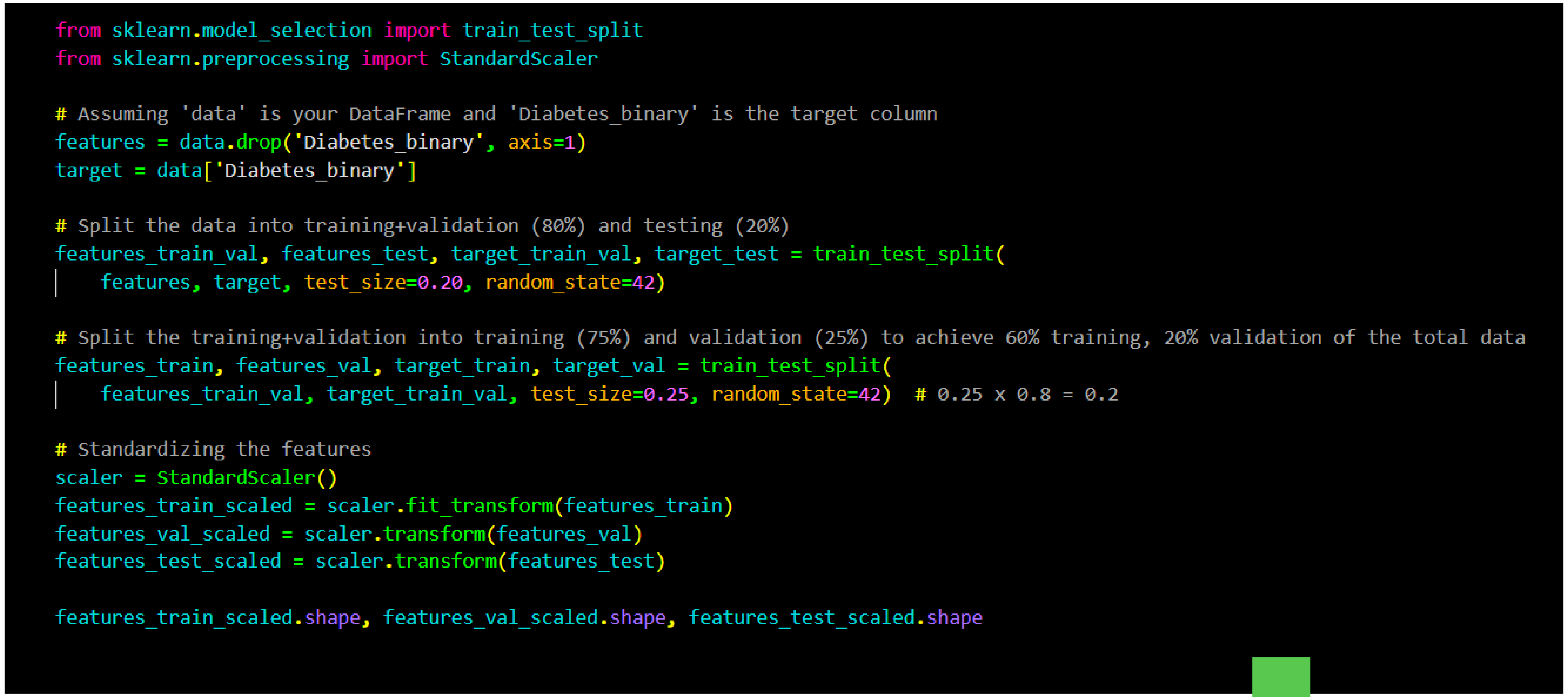
```
Diabetes_binary         0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
AnyHealthcare           0
NoDocbcCost             0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
Education               0
Income                  0
dtype: int64
```

SPLIT DATA

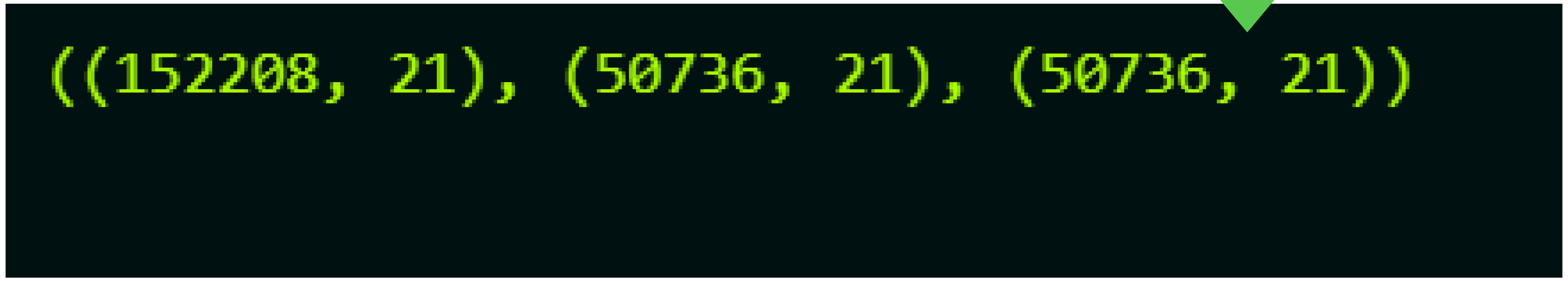


```
from sklearn.model_selection import train_test_split
import numpy as np

# Convert DataFrame to NumPy arrays
data_np = data.values
target_np = data_np[:, 0]  # Assuming 'Diabetes_binary' is the target variable
features_np = data_np[:, 1:]
```

✓ 0.0s



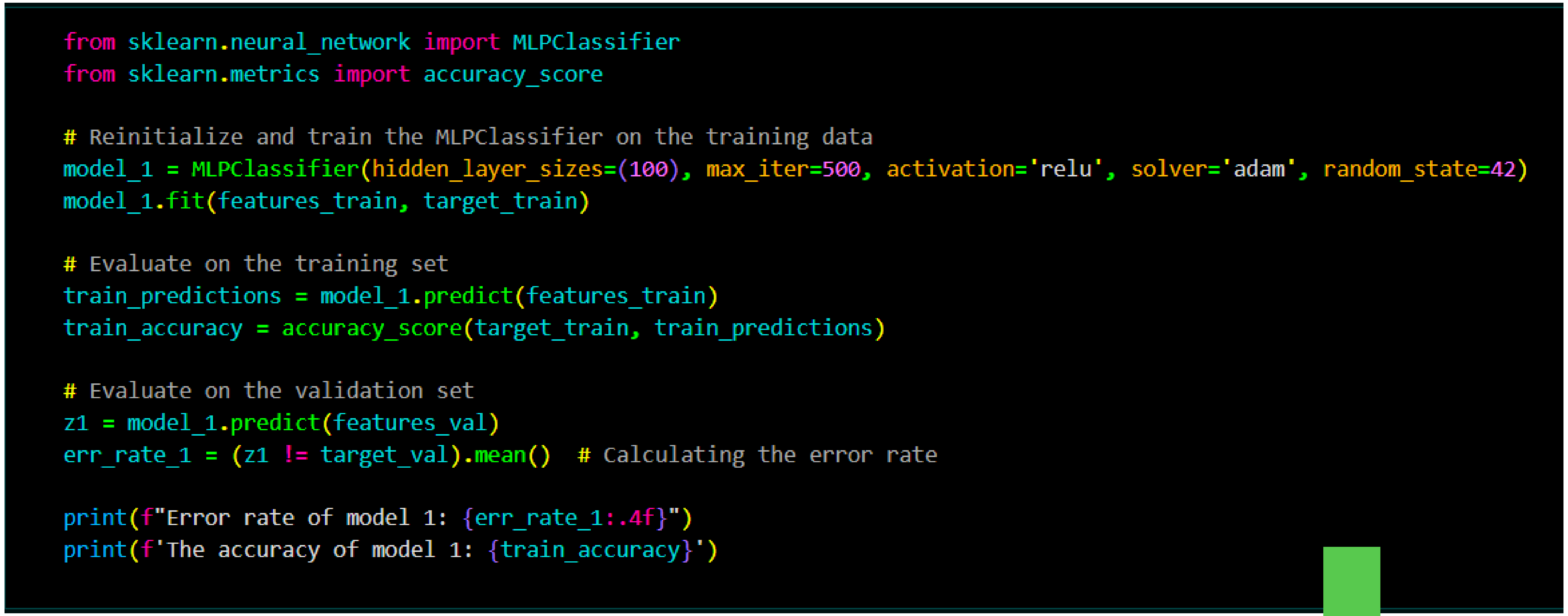
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Assuming 'data' is your DataFrame and 'Diabetes_binary' is the target column
features = data.drop('Diabetes_binary', axis=1)
target = data['Diabetes_binary']

# Split the data into training+validation (80%) and testing (20%)
features_train_val, features_test, target_train_val, target_test = train_test_split(
    features, target, test_size=0.20, random_state=42)

# Split the training+validation into training (75%) and validation (25%) to achieve 60% training, 20% validation of the total data
features_train, features_val, target_train, target_val = train_test_split(
    features_train_val, target_train_val, test_size=0.25, random_state=42)  # 0.25 x 0.8 = 0.2

# Standardizing the features
scaler = StandardScaler()
features_train_scaled = scaler.fit_transform(features_train)
features_val_scaled = scaler.transform(features_val)
features_test_scaled = scaler.transform(features_test)

features_train_scaled.shape, features_val_scaled.shape, features_test_scaled.shape
```



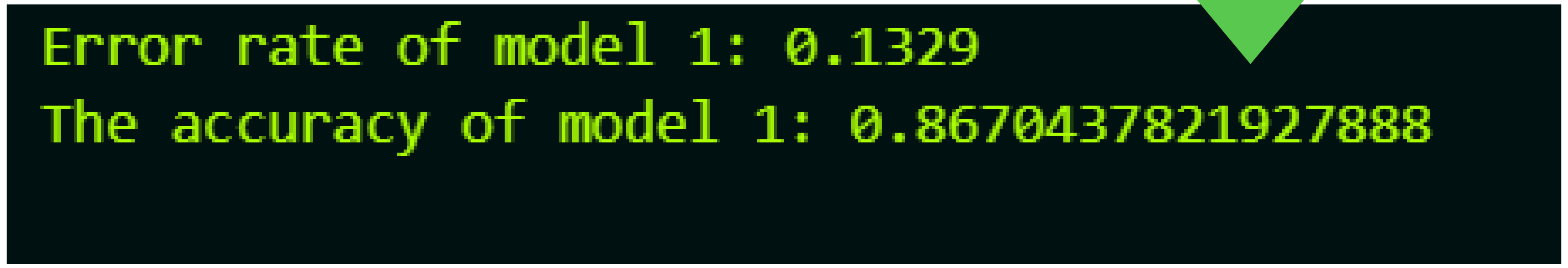
```
((152208, 21), (50736, 21), (50736, 21))
```

MODEL BUILDING

TRAINING SET
MODEL 1

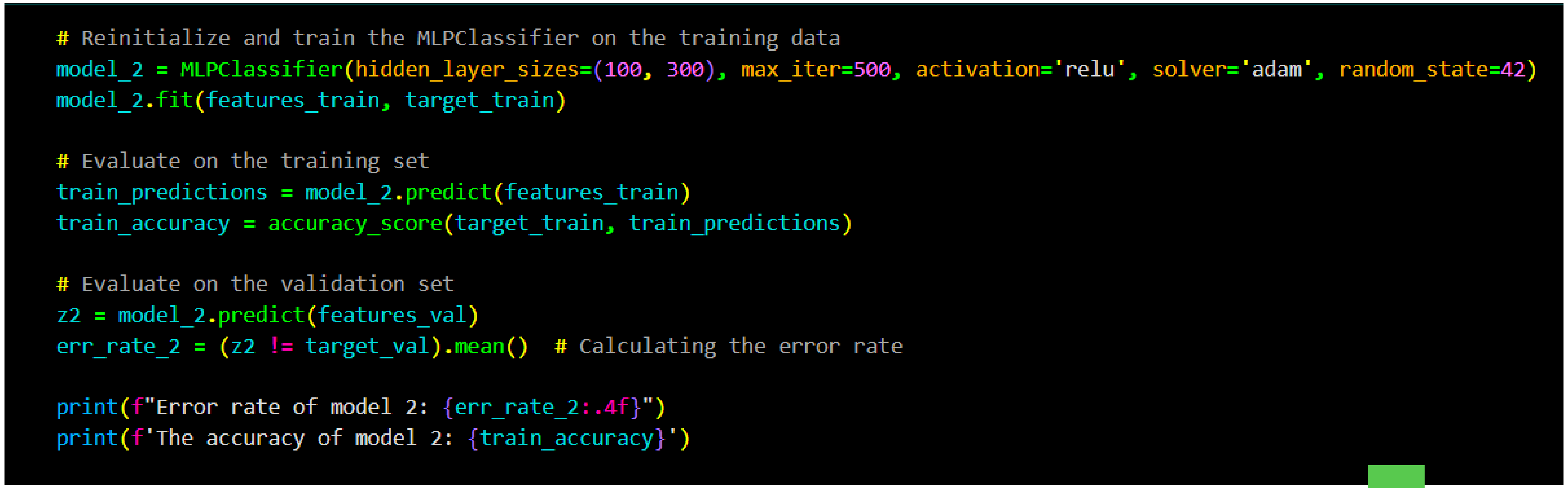


```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Reinitialize and train the MLPClassifier on the training data
model_1 = MLPClassifier(hidden_layer_sizes=(100), max_iter=500, activation='relu', solver='adam', random_state=42)
model_1.fit(features_train, target_train)

# Evaluate on the training set
train_predictions = model_1.predict(features_train)
train_accuracy = accuracy_score(target_train, train_predictions)

# Evaluate on the validation set
z1 = model_1.predict(features_val)
err_rate_1 = (z1 != target_val).mean()  # Calculating the error rate

print(f"Error rate of model 1: {err_rate_1:.4f}")
print(f'The accuracy of model 1: {train_accuracy}')
```

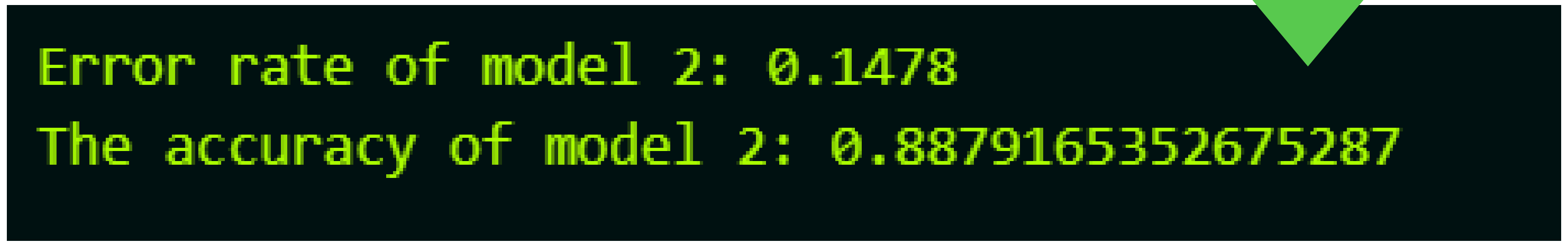


```
Error rate of model 1: 0.1329
The accuracy of model 1: 0.8670437821927888
```

MODEL 2

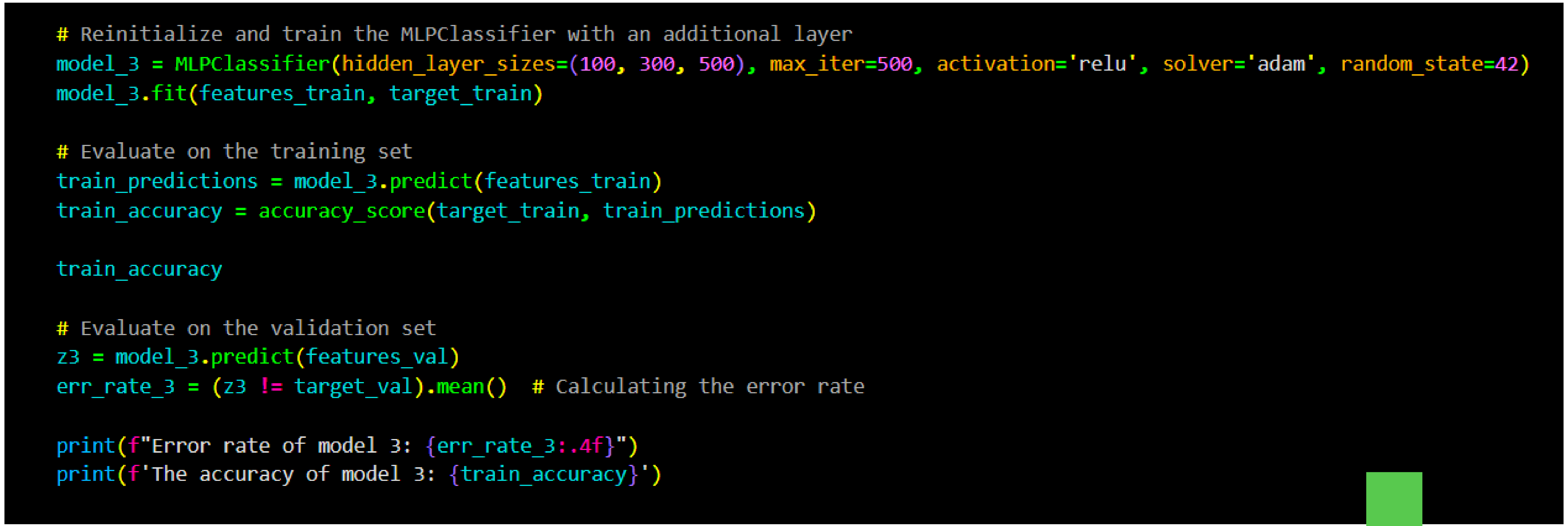


```
# Reinitialize and train the MLPClassifier on the training data
model_2 = MLPClassifier(hidden_layer_sizes=(100, 300), max_iter=500, activation='relu', solver='adam', random_state=42)
model_2.fit(features_train, target_train)

# Evaluate on the training set
train_predictions = model_2.predict(features_train)
train_accuracy = accuracy_score(target_train, train_predictions)

# Evaluate on the validation set
z2 = model_2.predict(features_val)
err_rate_2 = (z2 != target_val).mean()  # Calculating the error rate

print(f"Error rate of model 2: {err_rate_2:.4f}")
print(f'The accuracy of model 2: {train_accuracy}')
```

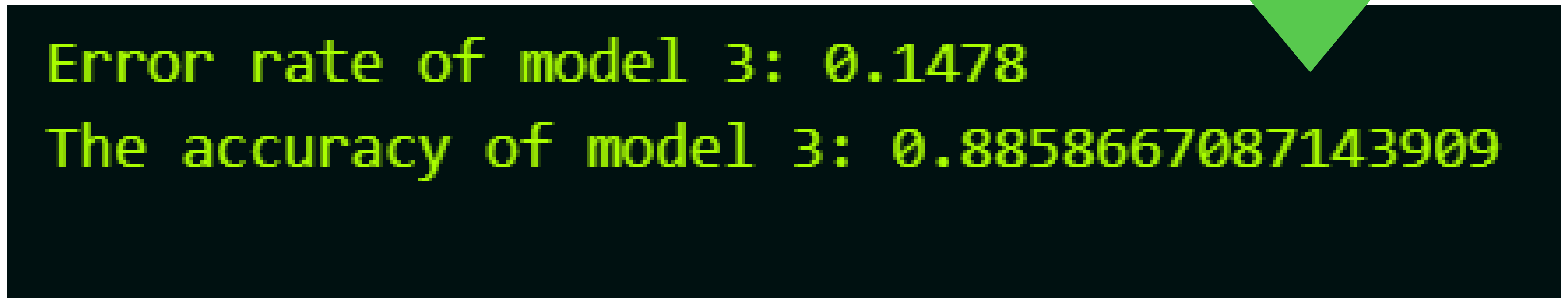


```
Error rate of model 2: 0.1478
The accuracy of model 2: 0.8879165352675287
```

MODEL 3



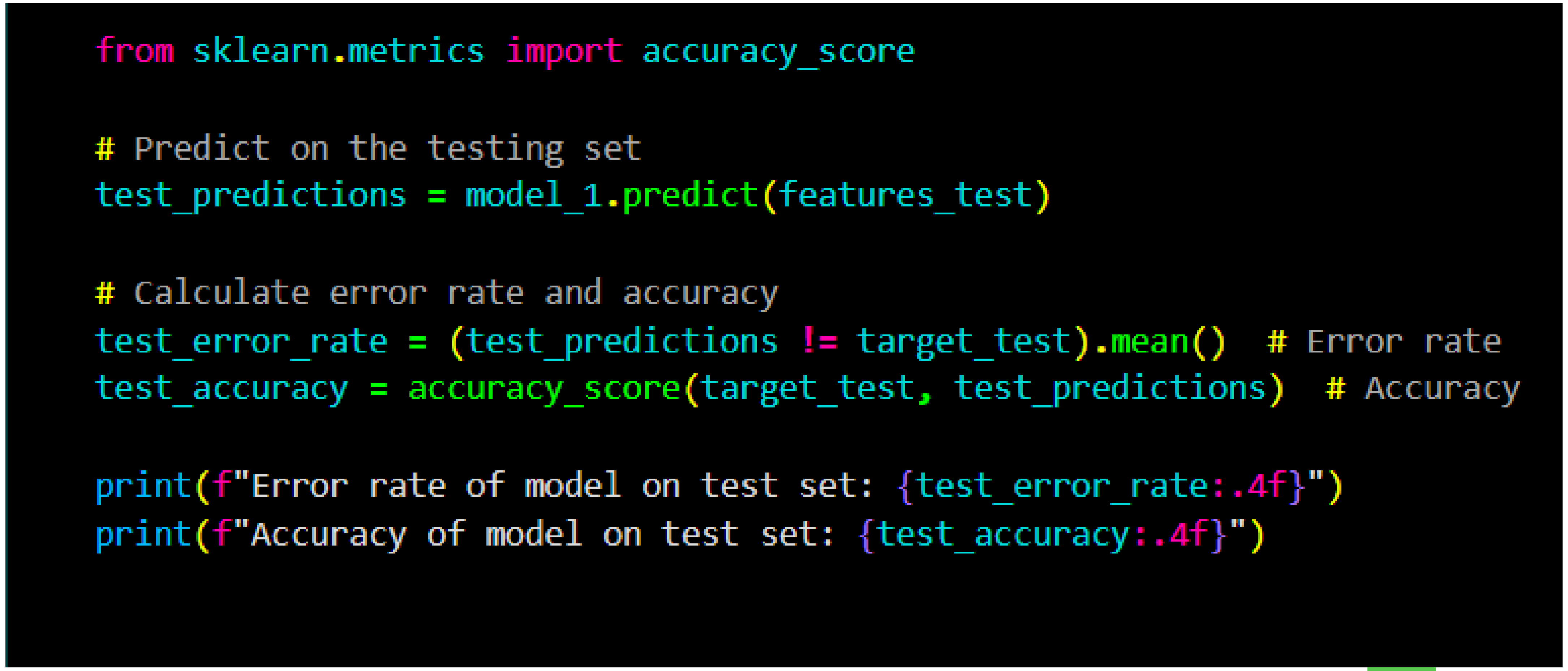
```
# Reinitialize and train the MLPClassifier with an additional layer
model_3 = MLPClassifier(hidden_layer_sizes=(100, 300, 500), max_iter=500, activation='relu', solver='adam', random_state=42)
model_3.fit(features_train, target_train)

# Evaluate on the training set
train_predictions = model_3.predict(features_train)
train_accuracy = accuracy_score(target_train, train_predictions)

train_accuracy

# Evaluate on the validation set
z3 = model_3.predict(features_val)
err_rate_3 = (z3 != target_val).mean()  # Calculating the error rate

print(f"Error rate of model 3: {err_rate_3:.4f}")
print(f'The accuracy of model 3: {train_accuracy}')
```



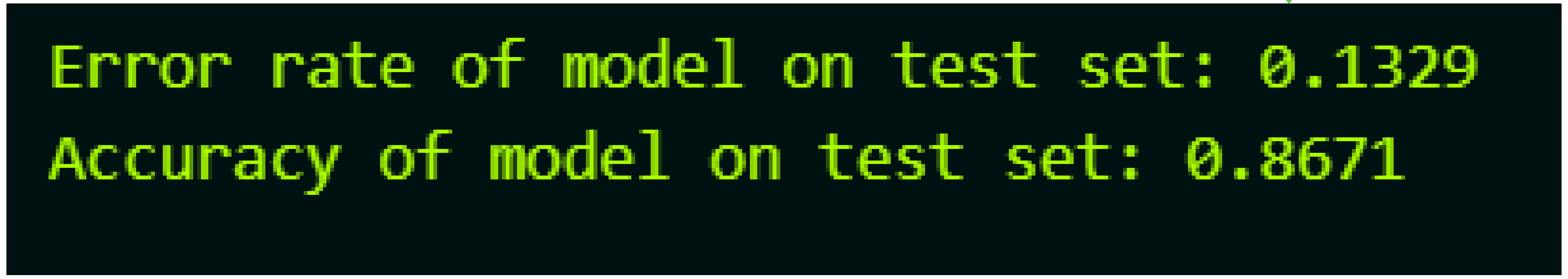
```
Error rate of model 3: 0.1478
The accuracy of model 3: 0.8858667087143909
```

TESTING SET

MODEL 1

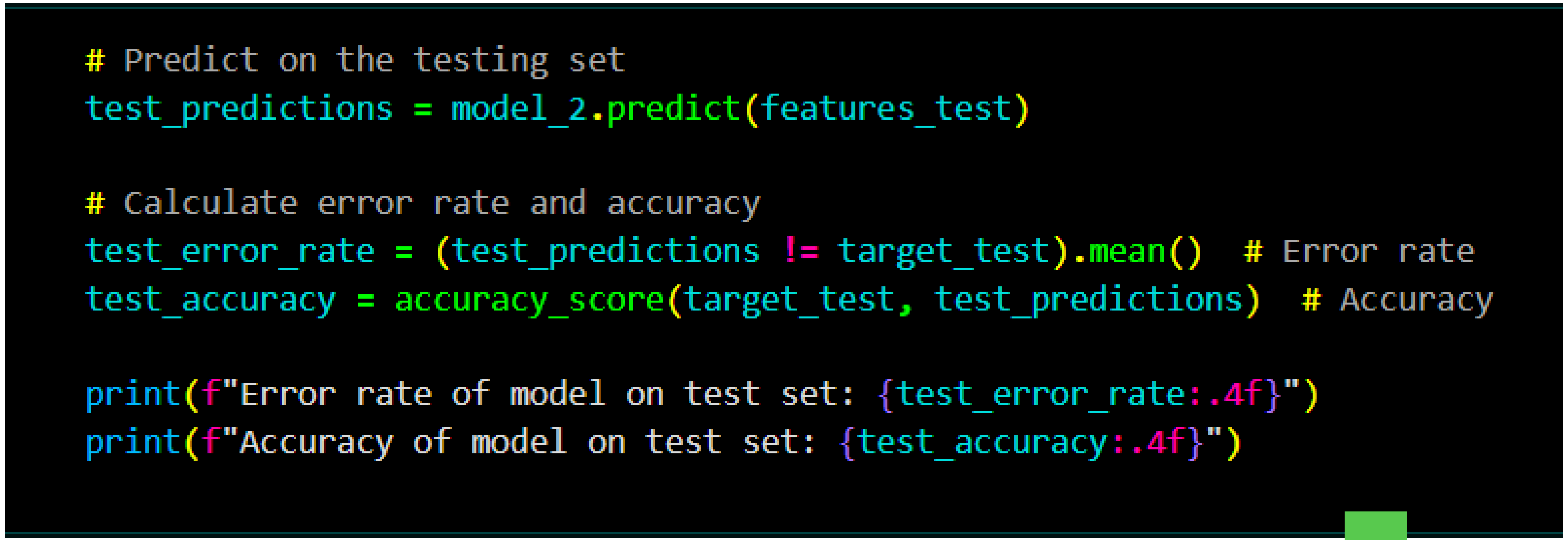


```python
from sklearn.metrics import accuracy_score

# Predict on the testing set
test_predictions = model_1.predict(features_test)

# Calculate error rate and accuracy
test_error_rate = (test_predictions != target_test).mean()  # Error rate
test_accuracy = accuracy_score(target_test, test_predictions)  # Accuracy

print(f"Error rate of model on test set: {test_error_rate:.4f}")
print(f"Accuracy of model on test set: {test_accuracy:.4f}")
```

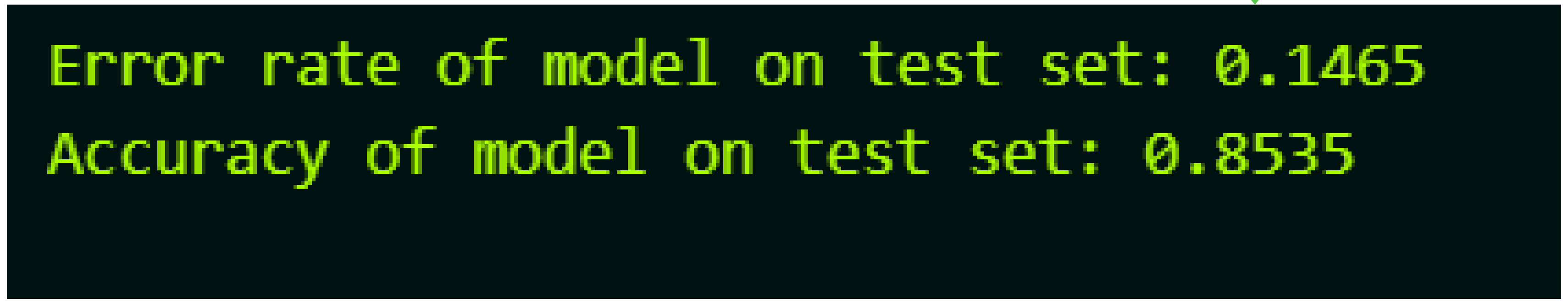


```
Error rate of model on test set: 0.1329
Accuracy of model on test set: 0.8671
```

MODEL 2

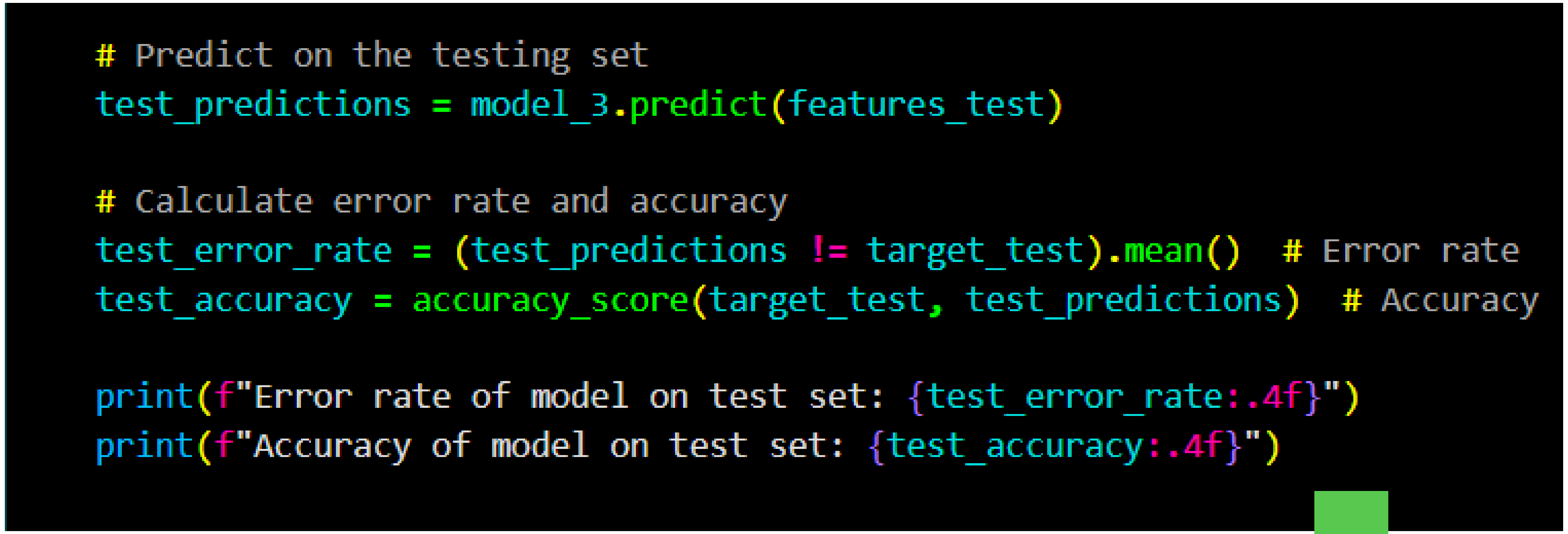


```python
# Predict on the testing set
test_predictions = model_2.predict(features_test)

# Calculate error rate and accuracy
test_error_rate = (test_predictions != target_test).mean()  # Error rate
test_accuracy = accuracy_score(target_test, test_predictions)  # Accuracy

print(f"Error rate of model on test set: {test_error_rate:.4f}")
print(f"Accuracy of model on test set: {test_accuracy:.4f}")
```

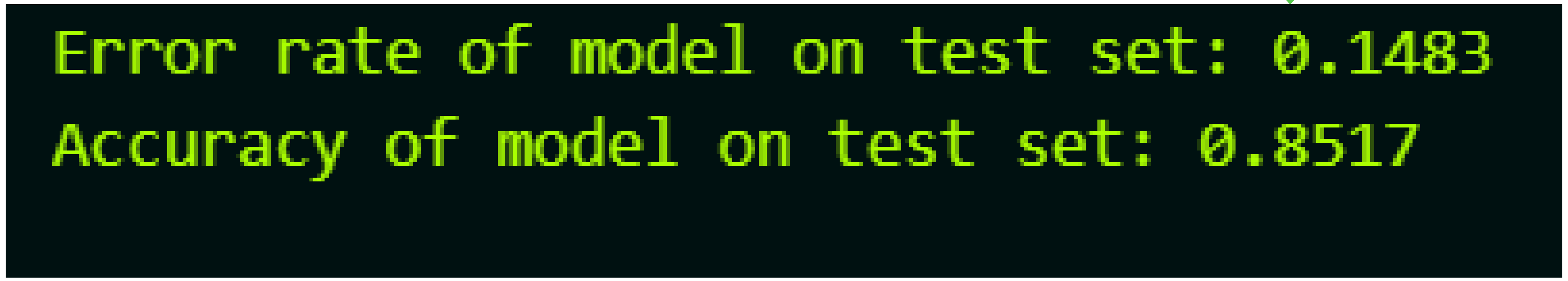


```
Error rate of model on test set: 0.1465
Accuracy of model on test set: 0.8535
```

MODEL 3

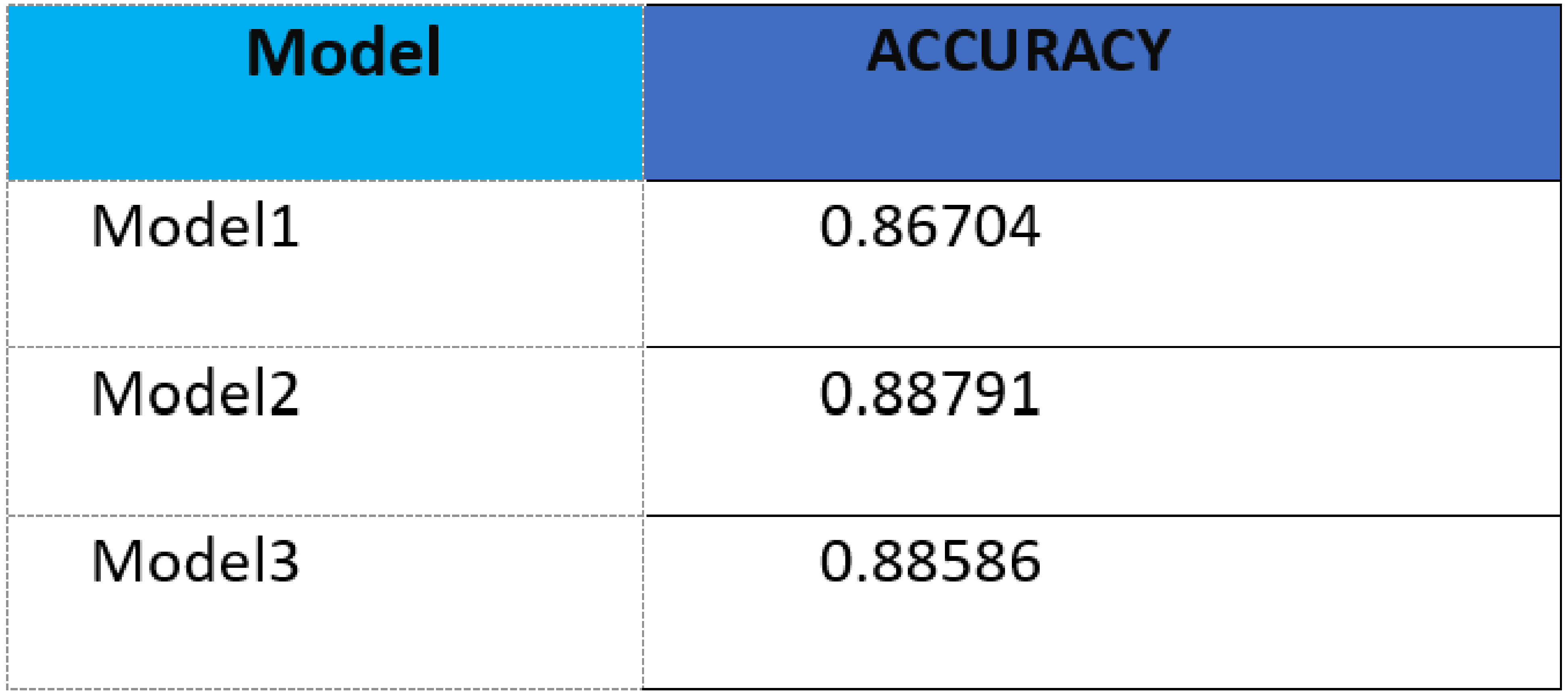


```
# Predict on the testing set
test_predictions = model_3.predict(features_test)

# Calculate error rate and accuracy
test_error_rate = (test_predictions != target_test).mean()  # Error rate
test_accuracy = accuracy_score(target_test, test_predictions)  # Accuracy

print(f"Error rate of model on test set: {test_error_rate:.4f}")
print(f"Accuracy of model on test set: {test_accuracy:.4f}")
```



```
Error rate of model on test set: 0.1483
Accuracy of model on test set: 0.8517
```

EVALUATION



Model	ACCURACY
Model1	0.86704
Model2	0.88791
Model3	0.88586

CONCLUSION

The reasons why we use Sklearn instead of Tensor flow to train our model because while we testing we”ve seen that the accuracy of Sklearn is higher than Tensor flow. And also base on the Second model in our Trained data is higher than other model. Not only this , we think our model learning is good because the accuracy of our training set is higher than testing set.

THANK YOU!!!