In [ ]:
```
%%shell
jupyter nbconvert --to html /content/Student_performance_profiles.ipynb
```

```
[NbConvertApp] Converting notebook /content/Student_performance_profiles.ipynb to htm
l
[NbConvertApp] Writing 884817 bytes to /content/Student_performance_profiles.html
```

Out[ ]:

Problem Statement

Continuing with the same scenario, now that you have been able to successfuly predict each student GPA, now you will classify each Student based on they probability to have a successful GPA score.

The different classes are:

- Low : Students where final GPA is predicted to be between: 0 and 2
- Medium : Students where final GPA is predicted to be between: 2 and 3.5
- High : Students where final GPA is predicted to be between: 3.5 and 5

1) Import Libraries

First let's import the following libraries, if there is any library that you need and is not in the list bellow feel free to include it

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten
from tensorflow.keras.regularizers import l2
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
import seaborn as sns
```

2) Load Data

- You will use the same file from the previous activity (Student Performance Data)

In [ ]:
```
data = pd.read_csv("Student_performance_data _.csv")
data
```

Out[ ]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tutoring
0	1001	17	1	0	2	19.833723	7	1
1	1002	18	0	0	1	15.408756	0	0
2	1003	15	0	2	3	4.210570	26	0
3	1004	17	1	0	3	10.028829	14	0
4	1005	17	1	0	2	4.672495	17	1
...	...	...	...	...	...	...	...	...
2387	3388	18	1	0	3	10.680555	2	0
2388	3389	17	0	0	1	7.583217	4	1
2389	3390	16	1	0	2	6.805500	20	0
2390	3391	16	1	1	0	12.416653	17	0
2391	3392	16	1	0	2	17.819907	13	0

2392 rows × 15 columns

In [ ]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2392 entries, 0 to 2391
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   StudentID          2392 non-null   int64  
 1   Age                2392 non-null   int64  
 2   Gender             2392 non-null   int64  
 3   Ethnicity          2392 non-null   int64  
 4   ParentalEducation  2392 non-null   int64  
 5   StudyTimeWeekly    2392 non-null   float64
 6   Absences           2392 non-null   int64  
 7   Tutoring           2392 non-null   int64  
 8   ParentalSupport    2392 non-null   int64  
 9   Extracurricular    2392 non-null   int64  
 10  Sports             2392 non-null   int64  
 11  Music              2392 non-null   int64  
 12  Volunteering       2392 non-null   int64  
 13  GPA                2392 non-null   float64
 14  GradeClass         2392 non-null   float64
dtypes: float64(3), int64(12)
memory usage: 280.4 KB
```

3) Add a new column called 'Profile' this column will have the following information

Based on the value of GPA for each student:

- If GPA values between 0 and 2 will be labeled 'Low',

- Values between 2 and 3.5 will be 'Medium',
- And values between 3.5 and 5 will be 'High'.

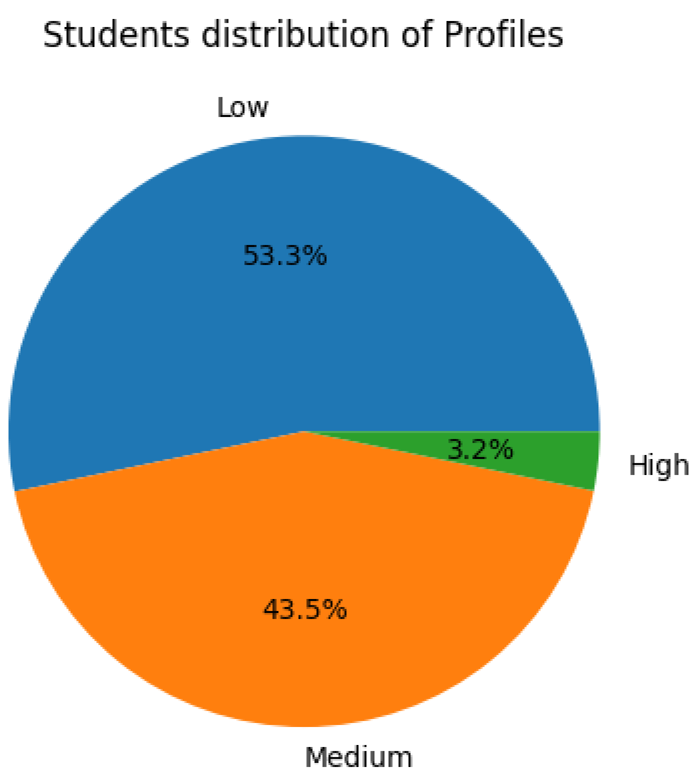
In [ ]:
```
def classify_gpa(gpa):
    if gpa <= 2:
        return 'Low'
    elif gpa <= 3.5:
        return 'Medium'
    else:
        return 'High'

data['Profile'] = data['GPA'].apply(classify_gpa)
```

4) Use Matplotlib to show a Pie chart to show the percentage of students in each profile.

- Title: Students distribution of Profiles
- Graph Type: pie

In [ ]:
```
profile_counts = data['Profile'].value_counts()
plt.pie(profile_counts, labels=profile_counts.index, autopct='%1.1f%%')
plt.title('Students distribution of Profiles')
plt.show()
```



5) Convert the Profile column into a Categorical Int

You have already created a column with three different values: 'Low', 'Medium', 'High'. These are Categorical values. But, it is important to notice that Neural Networks works better with numbers, since we apply mathematical operations to them.

Next you need to convert Profile values from Low, Medium and High, to 0, 1 and 2. IMPORTANT, the order does not matter, but make sure you always assign the same number to Low, same number to Medium and same number to High.

Make sure to use the fit_transform method from LabelEncoder.

In [ ]:
```
le = LabelEncoder()
data['Profile'] = le.fit_transform(data['Profile'])
```

6) Select the columns for your model.

Same as the last excersice we need a dataset for features and a dataset for label.

- Create the following dataset:
  - A dataset with the columns for the model.
  - From that data set generate the 'X' dataset. This dataset will have all the features (make sure Profile is NOT in this dataset)
  - Generate a second 'y' dataset, This dataset will only have our label column, which is 'Profile'.
  - Generate the Train and Test datasets for each X and y:
    - X_train with 80% of the data
    - X_test with 20% of the data
    - y_train with 80% of the data
    - y_test with 20% of the data

In [ ]:
```
X = data.drop(columns=['Profile', 'GPA'])
y = data['Profile']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

7) All Feature datasets in the same scale.

Use StandardScaler to make sure all features in the X_train and X_test datasets are on the same scale.

Standardization transforms your data so that it has a mean of 0 and a standard deviation of 1. This is important because many machine learning algorithms perform better when the input features are on a similar scale.

Reason for Using StandardScaler:

- Consistent Scale: Features with different scales (e.g., age in years, income in dollars) can bias the model. StandardScaler ensures all features contribute equally.
- Improved Convergence: Algorithms like gradient descent converge faster with standardized data.

- Regularization: Helps in achieving better performance in regularization methods like Ridge and Lasso regression.

In [ ]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

8. Define your Deep Neural Network.

- This will be a Sequential Neural Network.
- With a Dense input layer with 64 units, and input dimention based on the X_train size and Relu as the activation function.
- A Dense hidden layer with 32 units, and Relu as the activation function.
- And a Dense output layer with the number of different values in the y dataset, activation function = to sofmax

This last part of the output layer is super important, since we want to do a classification and not a regression, we will use activation functions that fits better a classification scenario.

In [ ]:
```
model = Sequential([
    Dense(64, input_dim=X_train.shape[1], activation='relu'),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarnin
g: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequentia
l models, prefer using an `Input(shape)` object as the first layer in the model inste
ad.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

9. Compile your Neural Network

- Choose Adam as the optimizer
- And sparse_categorical_crossentropy as the Loss function
- Also add the following metrics: accuracy

In [ ]:
```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accu
```

10. Fit (or train) your model

- Use the X_train and y_train datasets for the training
- Do 50 data iterations
- Choose the batch size = 10
- Also select a validation_split of 0.2
- Save the result of the fit function in a variable called 'history'

In [ ]:
```
history = model.fit(X_train, y_train, epochs=50, batch_size=10, validation_split=0.2)
```

```
Epoch 1/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 4s 9ms/step - accuracy: 0.6317 - loss: 0.8930 - val_accu
racy: 0.9112 - val_loss: 0.3364
Epoch 2/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 2s 6ms/step - accuracy: 0.9024 - loss: 0.2976 - val_accu
racy: 0.9347 - val_loss: 0.2196
Epoch 3/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9206 - loss: 0.2081 - val_accu
racy: 0.9373 - val_loss: 0.1751
Epoch 4/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9571 - loss: 0.1350 - val_accu
racy: 0.9399 - val_loss: 0.1532
Epoch 5/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9645 - loss: 0.1128 - val_accu
racy: 0.9478 - val_loss: 0.1322
Epoch 6/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9724 - loss: 0.0948 - val_accu
racy: 0.9582 - val_loss: 0.1231
Epoch 7/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9835 - loss: 0.0699 - val_accu
racy: 0.9556 - val_loss: 0.1119
Epoch 8/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9807 - loss: 0.0597 - val_accu
racy: 0.9582 - val_loss: 0.1018
Epoch 9/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9881 - loss: 0.0631 - val_accu
racy: 0.9582 - val_loss: 0.0963
Epoch 10/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9903 - loss: 0.0486 - val_accu
racy: 0.9634 - val_loss: 0.1006
Epoch 11/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9927 - loss: 0.0422 - val_accu
racy: 0.9634 - val_loss: 0.1076
Epoch 12/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9868 - loss: 0.0366 - val_accu
racy: 0.9661 - val_loss: 0.0987
Epoch 13/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9963 - loss: 0.0233 - val_accu
racy: 0.9582 - val_loss: 0.0974
Epoch 14/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9949 - loss: 0.0230 - val_accu
racy: 0.9687 - val_loss: 0.0868
Epoch 15/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9959 - loss: 0.0185 - val_accu
racy: 0.9634 - val_loss: 0.0926
Epoch 16/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9970 - loss: 0.0182 - val_accu
racy: 0.9687 - val_loss: 0.0972
Epoch 17/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9960 - loss: 0.0184 - val_accu
racy: 0.9661 - val_loss: 0.0968
Epoch 18/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9982 - loss: 0.0130 - val_accu
racy: 0.9713 - val_loss: 0.0874
Epoch 19/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9973 - loss: 0.0135 - val_accu
racy: 0.9687 - val_loss: 0.1000
Epoch 20/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9969 - loss: 0.0138 - val_accu
racy: 0.9687 - val_loss: 0.0936
```

```
Epoch 21/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9990 - loss: 0.0067 - val_accu
racy: 0.9713 - val_loss: 0.0957
Epoch 22/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9961 - loss: 0.0113 - val_accu
racy: 0.9687 - val_loss: 0.0991
Epoch 23/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9986 - loss: 0.0089 - val_accu
racy: 0.9687 - val_loss: 0.1106
Epoch 24/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9978 - loss: 0.0099 - val_accu
racy: 0.9661 - val_loss: 0.1083
Epoch 25/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9996 - loss: 0.0055 - val_accu
racy: 0.9713 - val_loss: 0.1086
Epoch 26/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9992 - loss: 0.0062 - val_accu
racy: 0.9713 - val_loss: 0.0984
Epoch 27/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9989 - loss: 0.0044 - val_accu
racy: 0.9713 - val_loss: 0.1058
Epoch 28/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9997 - loss: 0.0039 - val_accu
racy: 0.9713 - val_loss: 0.1018
Epoch 29/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9999 - loss: 0.0026 - val_accu
racy: 0.9713 - val_loss: 0.1188
Epoch 30/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9995 - loss: 0.0035 - val_accu
racy: 0.9687 - val_loss: 0.1049
Epoch 31/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9986 - loss: 0.0038 - val_accu
racy: 0.9687 - val_loss: 0.1180
Epoch 32/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9995 - loss: 0.0023 - val_accu
racy: 0.9739 - val_loss: 0.1019
Epoch 33/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0025 - val_accu
racy: 0.9739 - val_loss: 0.1128
Epoch 34/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9997 - loss: 0.0023 - val_accu
racy: 0.9765 - val_loss: 0.1030
Epoch 35/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 1.0000 - loss: 0.0010 - val_accu
racy: 0.9713 - val_loss: 0.1124
Epoch 36/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9999 - loss: 0.0012 - val_accu
racy: 0.9687 - val_loss: 0.1141
Epoch 37/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0013 - val_accu
racy: 0.9687 - val_loss: 0.1297
Epoch 38/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9998 - loss: 0.0017 - val_accu
racy: 0.9687 - val_loss: 0.1217
Epoch 39/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 1.0000 - loss: 0.0015 - val_accu
racy: 0.9661 - val_loss: 0.1229
Epoch 40/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 1.0000 - loss: 8.6694e-04 - val_
accuracy: 0.9765 - val_loss: 0.1147
```

```
Epoch 41/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 1.0000 - loss: 0.0011 - val_accu
racy: 0.9713 - val_loss: 0.1195
Epoch 42/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 1.0000 - loss: 7.3628e-04 - val_
accuracy: 0.9765 - val_loss: 0.1241
Epoch 43/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 5.7316e-04 - val_
accuracy: 0.9765 - val_loss: 0.1255
Epoch 44/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 8.3411e-04 - val_
accuracy: 0.9713 - val_loss: 0.1274
Epoch 45/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 3.9093e-04 - val_
accuracy: 0.9765 - val_loss: 0.1168
Epoch 46/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 1.0000 - loss: 5.4251e-04 - val_
accuracy: 0.9739 - val_loss: 0.1258
Epoch 47/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 1.0000 - loss: 3.3764e-04 - val_
accuracy: 0.9739 - val_loss: 0.1284
Epoch 48/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 1.0000 - loss: 3.3805e-04 - val_
accuracy: 0.9765 - val_loss: 0.1222
Epoch 49/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 3.3833e-04 - val_
accuracy: 0.9765 - val_loss: 0.1333
Epoch 50/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 2.4656e-04 - val_
accuracy: 0.9765 - val_loss: 0.1299
```

11. View your history variable:

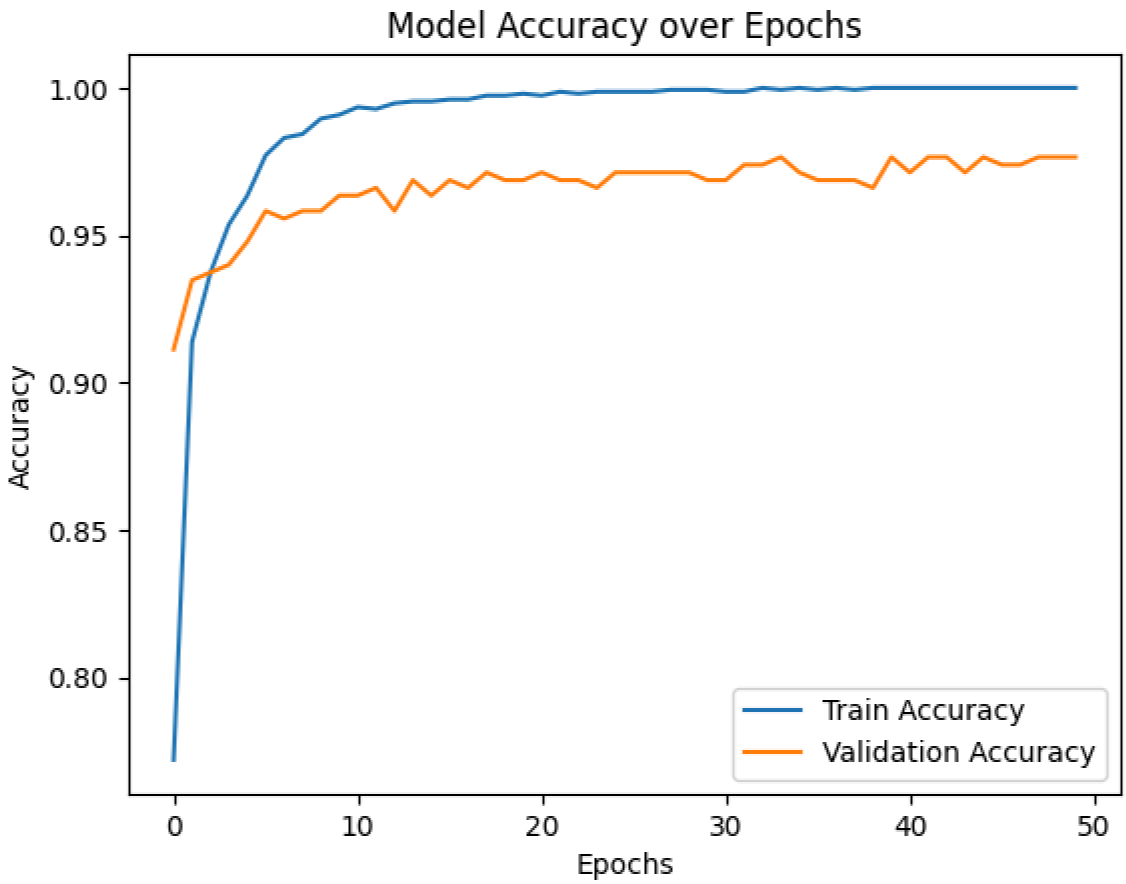
- Use Matplotlib.pyplot to show graphs of your model traning history
- In one graph:
  - Plot the Training Accuracy and the Validation Accuracy
  - X Label = Epochs
  - Y Label = Accuracy
  - Title = Model Accuracy over Epochs
- In a second graph:
  - Plot the Training Loss and the Validation Loss
  - X Label = Epochs
  - Y Label = Loss
  - Title = Model Loss over Epochs

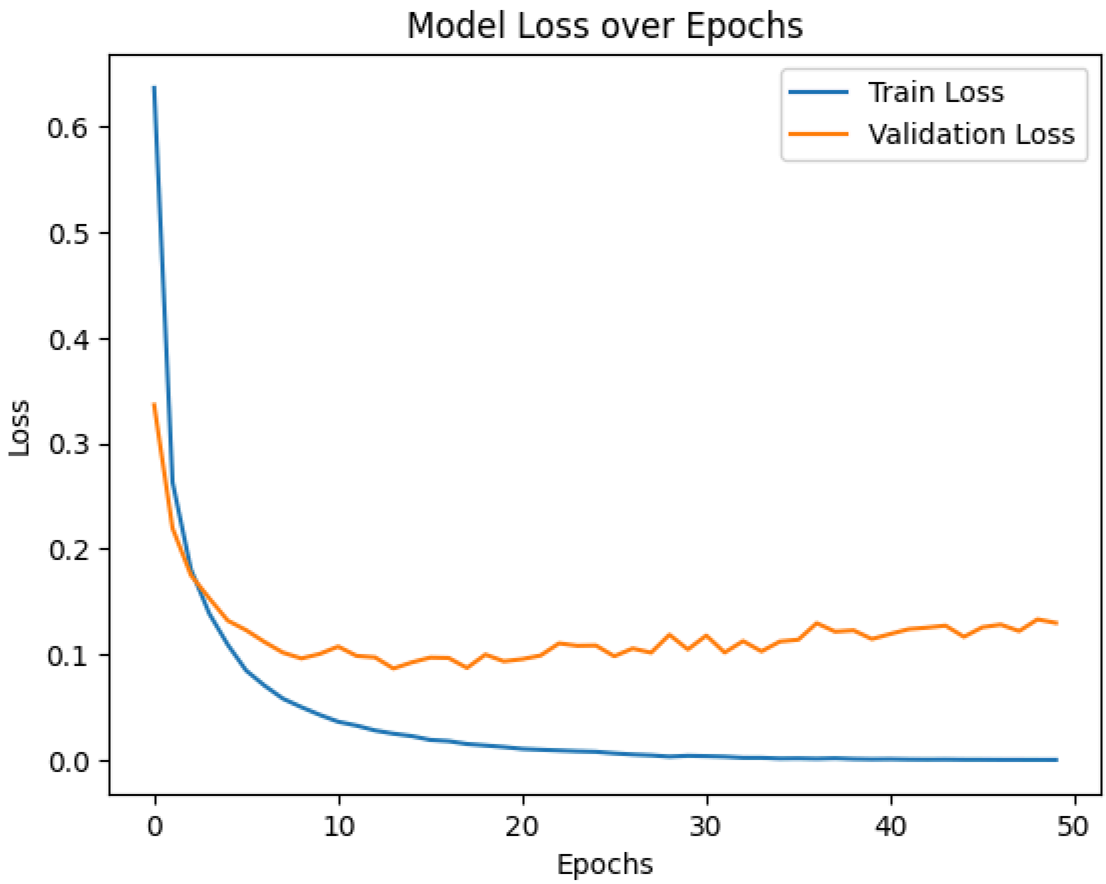
In [ ]:

```python
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy over Epochs')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
```

```
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss over Epochs')
plt.legend()
plt.show()
```





12. Evaluate your model:

- See the result of your loss function.
- What can you deduct from there?

In [ ]:
```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss}")
print(f"Test Accuracy: {accuracy}")
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9781 - loss: 0.0886
Test Loss: 0.09110262244939804
Test Accuracy: 0.9707724452018738
```

13. Use your model to make some predictions:

- Make predictions of your X_test dataset
- Print the each of the predictions and the actual value (which is in y_test)
- Replace the 'Low', 'Medium' and 'High' to your actual and predicted values.
- How good was your model?

In [ ]:
```
predictions = model.predict(X_test)
predicted_classes = np.argmax(predictions, axis=1)

for i in range(len(predicted_classes)):
    print(f"Predicted: {le.inverse_transform([predicted_classes[i]])[0]}, Actual: {le.
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: High, Actual: High
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: High, Actual: High
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
```

```
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: High, Actual: High
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: High, Actual: High
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: High
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
```

```
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: High
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: High, Actual: High
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: High, Actual: High
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: High, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
```

```
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: High, Actual: High
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
```

```
Predicted: Low, Actual: Low
Predicted: High, Actual: High
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: High, Actual: High
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
```

```
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
```

```
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: High, Actual: High
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: High
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: High
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: High
Predicted: Medium, Actual: Medium
```

```
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: High, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: High, Actual: High
Predicted: Medium, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Low, Actual: Low
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Medium, Actual: Medium
Predicted: Low, Actual: Low
```

14. Compete against this model:

- Create two more different models to compete with this model
- Here are a few ideas of things you can change:
  - During Dataset data engineering:
    - You can remove features that you think do not help in the training and prediction
    - Feature Scaling: Ensure all features are on a similar scale (as you already did with StandardScaler)
  - During Model Definition:
    - You can change the Model Architecture (change the type or number of layers or the number of units)
    - You can add dropout layers to prevent overfitting
  - During Model Compile:
    - You can try other optimizer when compiling your model, here some optimizer samples: Adam, RMSprop, or Adagrad.
    - Try another Loss Function
  - During Model Training:
    - Encrease the number of Epochs
    - Adjust the size of your batch
- Explain in a Markdown cell which changes are you implementing
- Show the comparison of your model versus the original model

Model 2:

- Changes:
  - Dataset Data Engineering
  - Model Definition
  - Model Compile
  - Model Training

In [ ]:
```
# Model 2 Cambio en Model Definition
model2 = Sequential([
    Dense(128, input_dim=X_train.shape[1], activation='relu'),  # Mas Unidades
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
])

model2.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['acc

history2 = model2.fit(X_train, y_train, epochs=50, batch_size=10, validation_split=0.2
```

```
Epoch 1/50
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarnin
g: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequentia
l models, prefer using an `Input(shape)` object as the first layer in the model inste
ad.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
153/153 ━━━━━━━━━━━━━━━━━━━━ 3s 5ms/step - accuracy: 0.6104 - loss: 0.7710 - val_accu
racy: 0.9086 - val_loss: 0.2600
Epoch 2/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9050 - loss: 0.2612 - val_accu
racy: 0.9321 - val_loss: 0.2022
Epoch 3/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9338 - loss: 0.1856 - val_accu
racy: 0.9399 - val_loss: 0.1806
Epoch 4/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9454 - loss: 0.1577 - val_accu
racy: 0.9504 - val_loss: 0.1408
Epoch 5/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9505 - loss: 0.1377 - val_accu
racy: 0.9530 - val_loss: 0.1326
Epoch 6/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9459 - loss: 0.1352 - val_accu
racy: 0.9713 - val_loss: 0.1240
Epoch 7/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9554 - loss: 0.1127 - val_accu
racy: 0.9687 - val_loss: 0.1108
Epoch 8/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9470 - loss: 0.1215 - val_accu
racy: 0.9582 - val_loss: 0.1086
Epoch 9/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9410 - loss: 0.1213 - val_accu
racy: 0.9687 - val_loss: 0.1033
Epoch 10/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9580 - loss: 0.1095 - val_accu
racy: 0.9661 - val_loss: 0.0877
Epoch 11/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9632 - loss: 0.0915 - val_accu
racy: 0.9556 - val_loss: 0.0931
Epoch 12/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.9685 - loss: 0.0806 - val_accu
racy: 0.9713 - val_loss: 0.0892
Epoch 13/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9642 - loss: 0.0793 - val_accu
racy: 0.9478 - val_loss: 0.0929
Epoch 14/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9716 - loss: 0.0669 - val_accu
racy: 0.9347 - val_loss: 0.1279
Epoch 15/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9718 - loss: 0.0747 - val_accu
racy: 0.9713 - val_loss: 0.0775
Epoch 16/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9555 - loss: 0.0920 - val_accu
racy: 0.9713 - val_loss: 0.0867
Epoch 17/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9748 - loss: 0.0538 - val_accu
racy: 0.9765 - val_loss: 0.0762
Epoch 18/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9761 - loss: 0.0594 - val_accu
racy: 0.9791 - val_loss: 0.0857
Epoch 19/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9698 - loss: 0.0729 - val_accu
racy: 0.9713 - val_loss: 0.0681
Epoch 20/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9841 - loss: 0.0531 - val_accu
racy: 0.9791 - val_loss: 0.0680
Epoch 21/50
```

```
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9783 - loss: 0.0614 - val_accu
racy: 0.9791 - val_loss: 0.0662
Epoch 22/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9768 - loss: 0.0818 - val_accu
racy: 0.9817 - val_loss: 0.0656
Epoch 23/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9786 - loss: 0.0492 - val_accu
racy: 0.9765 - val_loss: 0.0704
Epoch 24/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9737 - loss: 0.0665 - val_accu
racy: 0.9765 - val_loss: 0.0893
Epoch 25/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9745 - loss: 0.0594 - val_accu
racy: 0.9739 - val_loss: 0.0670
Epoch 26/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9843 - loss: 0.0563 - val_accu
racy: 0.9817 - val_loss: 0.0734
Epoch 27/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9790 - loss: 0.0427 - val_accu
racy: 0.9765 - val_loss: 0.0744
Epoch 28/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9814 - loss: 0.0470 - val_accu
racy: 0.9713 - val_loss: 0.0794
Epoch 29/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9900 - loss: 0.0335 - val_accu
racy: 0.9791 - val_loss: 0.0723
Epoch 30/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9773 - loss: 0.0476 - val_accu
racy: 0.9817 - val_loss: 0.0668
Epoch 31/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9785 - loss: 0.0430 - val_accu
racy: 0.9817 - val_loss: 0.0676
Epoch 32/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9841 - loss: 0.0403 - val_accu
racy: 0.9765 - val_loss: 0.0679
Epoch 33/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9851 - loss: 0.0346 - val_accu
racy: 0.9791 - val_loss: 0.0645
Epoch 34/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9904 - loss: 0.0292 - val_accu
racy: 0.9765 - val_loss: 0.0706
Epoch 35/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9833 - loss: 0.0436 - val_accu
racy: 0.9765 - val_loss: 0.0585
Epoch 36/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9863 - loss: 0.0353 - val_accu
racy: 0.9817 - val_loss: 0.0773
Epoch 37/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9874 - loss: 0.0390 - val_accu
racy: 0.9817 - val_loss: 0.0650
Epoch 38/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9871 - loss: 0.0361 - val_accu
racy: 0.9843 - val_loss: 0.0678
Epoch 39/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9835 - loss: 0.0477 - val_accu
racy: 0.9843 - val_loss: 0.0693
Epoch 40/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9884 - loss: 0.0368 - val_accu
racy: 0.9791 - val_loss: 0.0799
Epoch 41/50
```

```
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9840 - loss: 0.0394 - val_accu
racy: 0.9817 - val_loss: 0.0764
Epoch 42/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9915 - loss: 0.0256 - val_accu
racy: 0.9817 - val_loss: 0.0866
Epoch 43/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9873 - loss: 0.0315 - val_accu
racy: 0.9791 - val_loss: 0.0883
Epoch 44/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9845 - loss: 0.0387 - val_accu
racy: 0.9713 - val_loss: 0.0785
Epoch 45/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9901 - loss: 0.0262 - val_accu
racy: 0.9739 - val_loss: 0.0684
Epoch 46/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9905 - loss: 0.0246 - val_accu
racy: 0.9765 - val_loss: 0.0771
Epoch 47/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9929 - loss: 0.0217 - val_accu
racy: 0.9817 - val_loss: 0.0629
Epoch 48/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9933 - loss: 0.0240 - val_accu
racy: 0.9791 - val_loss: 0.0678
Epoch 49/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9868 - loss: 0.0289 - val_accu
racy: 0.9791 - val_loss: 0.0656
Epoch 50/50
153/153 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9943 - loss: 0.0163 - val_accu
racy: 0.9896 - val_loss: 0.0655
```

Model 3:

- Changes:
  - Dataset Data Engineering
  - Model Definition
  - Model Compile
  - Model Training

In [ ]:

```
# Model 3 Cambio en Model Training
model3 = Sequential([
    Dense(64, input_dim=X_train.shape[1], activation='relu'),
    Dense(32, activation='relu'),
    Dense(3, activation='softmax')
])

# RMSprop en vez de Adam
model3.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy', metrics=['

# Mas Epochs y Batch Size
history3 = model3.fit(X_train, y_train, epochs=100, batch_size=20, validation_split=0.
```

```
Epoch 1/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.6579 - loss: 0.8403 - val_accura
cy: 0.9086 - val_loss: 0.3621
Epoch 2/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9090 - loss: 0.3079 - val_accura
cy: 0.9321 - val_loss: 0.2097
Epoch 3/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9365 - loss: 0.1890 - val_accura
cy: 0.9295 - val_loss: 0.1659
Epoch 4/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9511 - loss: 0.1354 - val_accura
cy: 0.9504 - val_loss: 0.1354
Epoch 5/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9644 - loss: 0.1172 - val_accura
cy: 0.9608 - val_loss: 0.1212
Epoch 6/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9752 - loss: 0.0820 - val_accura
cy: 0.9530 - val_loss: 0.1244
Epoch 7/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9778 - loss: 0.0882 - val_accura
cy: 0.9687 - val_loss: 0.1102
Epoch 8/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9814 - loss: 0.0664 - val_accura
cy: 0.9713 - val_loss: 0.0995
Epoch 9/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9835 - loss: 0.0583 - val_accura
cy: 0.9713 - val_loss: 0.0973
Epoch 10/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9906 - loss: 0.0441 - val_accura
cy: 0.9713 - val_loss: 0.0927
Epoch 11/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9870 - loss: 0.0481 - val_accura
cy: 0.9713 - val_loss: 0.0919
Epoch 12/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9913 - loss: 0.0357 - val_accura
cy: 0.9739 - val_loss: 0.0860
Epoch 13/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9921 - loss: 0.0359 - val_accura
cy: 0.9765 - val_loss: 0.0857
Epoch 14/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9901 - loss: 0.0425 - val_accura
cy: 0.9765 - val_loss: 0.0836
Epoch 15/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9903 - loss: 0.0281 - val_accura
cy: 0.9791 - val_loss: 0.0817
Epoch 16/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9890 - loss: 0.0367 - val_accura
cy: 0.9713 - val_loss: 0.0842
Epoch 17/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9959 - loss: 0.0244 - val_accura
cy: 0.9687 - val_loss: 0.0862
Epoch 18/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9950 - loss: 0.0242 - val_accura
cy: 0.9713 - val_loss: 0.0851
Epoch 19/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9958 - loss: 0.0211 - val_accura
cy: 0.9713 - val_loss: 0.0905
Epoch 20/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9913 - loss: 0.0214 - val_accura
cy: 0.9634 - val_loss: 0.0860
```

```
Epoch 21/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9968 - loss: 0.0167 - val_accura
cy: 0.9687 - val_loss: 0.0858
Epoch 22/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9919 - loss: 0.0198 - val_accura
cy: 0.9739 - val_loss: 0.0838
Epoch 23/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9942 - loss: 0.0161 - val_accura
cy: 0.9713 - val_loss: 0.0891
Epoch 24/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9942 - loss: 0.0171 - val_accura
cy: 0.9634 - val_loss: 0.0947
Epoch 25/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9958 - loss: 0.0158 - val_accura
cy: 0.9739 - val_loss: 0.0909
Epoch 26/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9986 - loss: 0.0092 - val_accura
cy: 0.9713 - val_loss: 0.0852
Epoch 27/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9973 - loss: 0.0093 - val_accura
cy: 0.9713 - val_loss: 0.0942
Epoch 28/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9968 - loss: 0.0106 - val_accura
cy: 0.9634 - val_loss: 0.0931
Epoch 29/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9984 - loss: 0.0070 - val_accura
cy: 0.9634 - val_loss: 0.1040
Epoch 30/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9984 - loss: 0.0087 - val_accura
cy: 0.9634 - val_loss: 0.0986
Epoch 31/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9962 - loss: 0.0092 - val_accura
cy: 0.9634 - val_loss: 0.1086
Epoch 32/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9992 - loss: 0.0062 - val_accura
cy: 0.9739 - val_loss: 0.1003
Epoch 33/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9999 - loss: 0.0072 - val_accura
cy: 0.9739 - val_loss: 0.1035
Epoch 34/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9977 - loss: 0.0072 - val_accura
cy: 0.9634 - val_loss: 0.1091
Epoch 35/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9999 - loss: 0.0037 - val_accura
cy: 0.9739 - val_loss: 0.1051
Epoch 36/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9968 - loss: 0.0071 - val_accura
cy: 0.9608 - val_loss: 0.1116
Epoch 37/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9998 - loss: 0.0040 - val_accura
cy: 0.9687 - val_loss: 0.1080
Epoch 38/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9973 - loss: 0.0064 - val_accura
cy: 0.9687 - val_loss: 0.1140
Epoch 39/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9981 - loss: 0.0053 - val_accura
cy: 0.9634 - val_loss: 0.1116
Epoch 40/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9988 - loss: 0.0038 - val_accura
cy: 0.9661 - val_loss: 0.1055
```

```
Epoch 41/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9999 - loss: 0.0021 - val_accura
cy: 0.9687 - val_loss: 0.1045
Epoch 42/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9988 - loss: 0.0031 - val_accura
cy: 0.9687 - val_loss: 0.1110
Epoch 43/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9996 - loss: 0.0033 - val_accura
cy: 0.9687 - val_loss: 0.1077
Epoch 44/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9974 - loss: 0.0043 - val_accura
cy: 0.9634 - val_loss: 0.1283
Epoch 45/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9970 - loss: 0.0041 - val_accura
cy: 0.9661 - val_loss: 0.1243
Epoch 46/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9996 - loss: 0.0025 - val_accura
cy: 0.9634 - val_loss: 0.1436
Epoch 47/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 7.6825e-04 - val_ac
curacy: 0.9687 - val_loss: 0.1379
Epoch 48/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9989 - loss: 0.0019 - val_accura
cy: 0.9687 - val_loss: 0.1342
Epoch 49/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9997 - loss: 0.0016 - val_accura
cy: 0.9661 - val_loss: 0.1277
Epoch 50/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9994 - loss: 0.0020 - val_accura
cy: 0.9661 - val_loss: 0.1322
Epoch 51/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9995 - loss: 0.0017 - val_accura
cy: 0.9713 - val_loss: 0.1176
Epoch 52/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 5.5579e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1340
Epoch 53/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9998 - loss: 0.0011 - val_accura
cy: 0.9687 - val_loss: 0.1270
Epoch 54/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9998 - loss: 9.2166e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1326
Epoch 55/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0010 - val_accura
cy: 0.9739 - val_loss: 0.1469
Epoch 56/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9992 - loss: 0.0018 - val_accura
cy: 0.9765 - val_loss: 0.1235
Epoch 57/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9989 - loss: 0.0026 - val_accura
cy: 0.9739 - val_loss: 0.1268
Epoch 58/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9996 - loss: 0.0014 - val_accura
cy: 0.9634 - val_loss: 0.1416
Epoch 59/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9999 - loss: 6.1642e-04 - val_ac
curacy: 0.9765 - val_loss: 0.1272
Epoch 60/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9994 - loss: 8.1679e-04 - val_ac
curacy: 0.9634 - val_loss: 0.1480
```

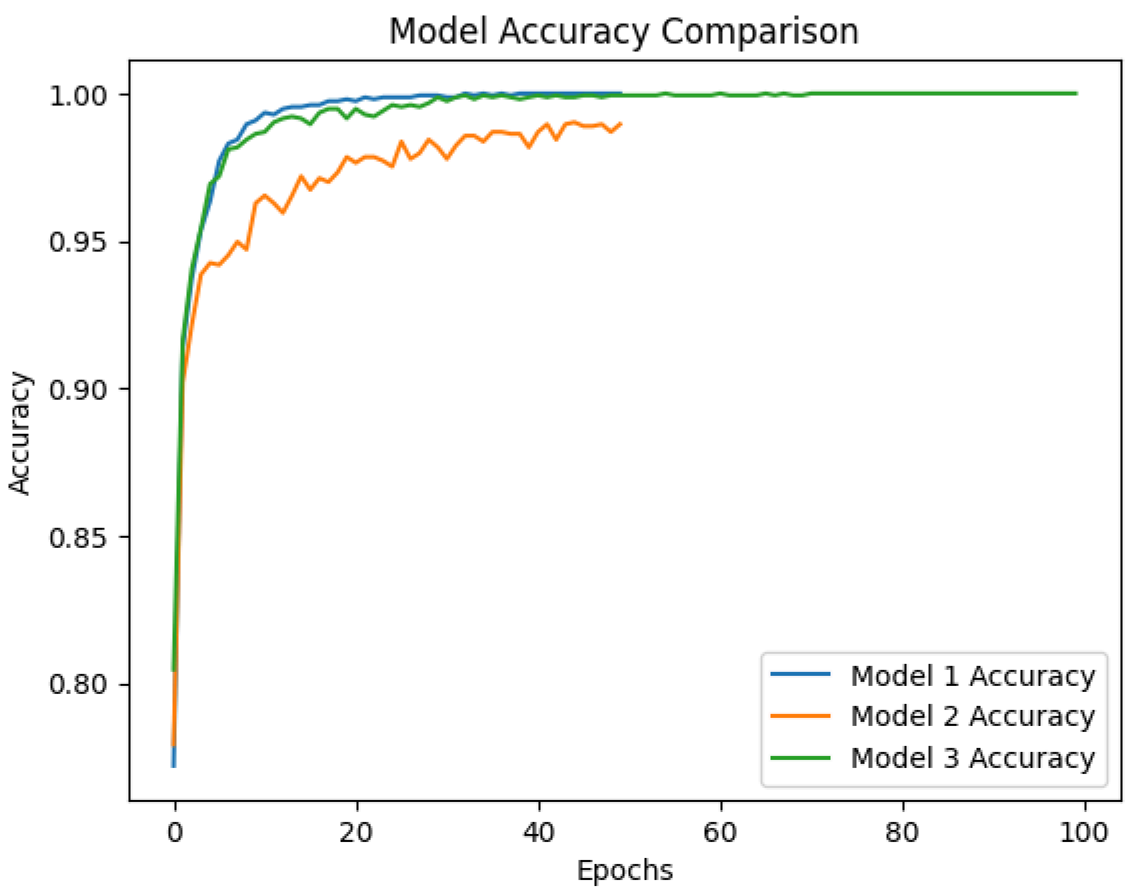
```
Epoch 61/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0011 - val_accura
cy: 0.9739 - val_loss: 0.1268
Epoch 62/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9999 - loss: 3.0610e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1278
Epoch 63/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9996 - loss: 8.5050e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1326
Epoch 64/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9995 - loss: 9.4879e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1339
Epoch 65/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9995 - loss: 6.2927e-04 - val_ac
curacy: 0.9739 - val_loss: 0.1314
Epoch 66/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 2.7222e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1396
Epoch 67/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9993 - loss: 0.0013 - val_accura
cy: 0.9739 - val_loss: 0.1299
Epoch 68/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 2.5861e-04 - val_ac
curacy: 0.9739 - val_loss: 0.1294
Epoch 69/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9998 - loss: 4.7616e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1387
Epoch 70/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9999 - loss: 1.8516e-04 - val_ac
curacy: 0.9739 - val_loss: 0.1353
Epoch 71/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 1.7161e-04 - val_ac
curacy: 0.9765 - val_loss: 0.1461
Epoch 72/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 5.1342e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1568
Epoch 73/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 5.5257e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1442
Epoch 74/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 5.0231e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1525
Epoch 75/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 1.2833e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1377
Epoch 76/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 1.9629e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1364
Epoch 77/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 2.0262e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1510
Epoch 78/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 1.5332e-04 - val_ac
curacy: 0.9765 - val_loss: 0.1510
Epoch 79/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 8.1069e-05 - val_ac
curacy: 0.9713 - val_loss: 0.1349
Epoch 80/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 8.8884e-05 - val_ac
curacy: 0.9713 - val_loss: 0.1639
```

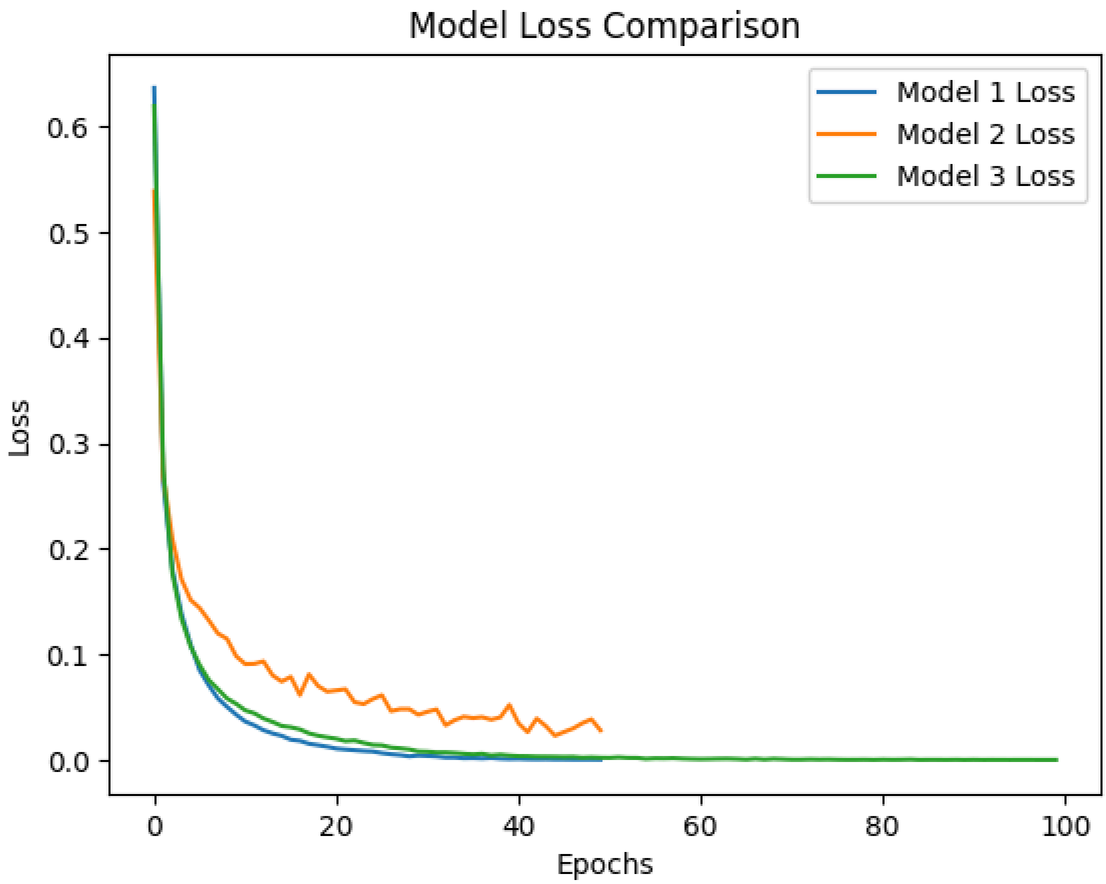
```
Epoch 81/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 3.2305e-04 - val_ac
curacy: 0.9739 - val_loss: 0.1500
Epoch 82/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 1.0000 - loss: 1.3132e-04 - val_ac
curacy: 0.9713 - val_loss: 0.1452
Epoch 83/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 1.9888e-04 - val_ac
curacy: 0.9739 - val_loss: 0.1361
Epoch 84/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 9.7344e-05 - val_ac
curacy: 0.9713 - val_loss: 0.1389
Epoch 85/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 6.6692e-05 - val_ac
curacy: 0.9713 - val_loss: 0.1561
Epoch 86/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 1.0000 - loss: 3.9041e-04 - val_ac
curacy: 0.9739 - val_loss: 0.1444
Epoch 87/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 1.0000 - loss: 6.2525e-05 - val_ac
curacy: 0.9739 - val_loss: 0.1452
Epoch 88/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 7.2012e-05 - val_ac
curacy: 0.9739 - val_loss: 0.1537
Epoch 89/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 3.4939e-05 - val_ac
curacy: 0.9739 - val_loss: 0.1371
Epoch 90/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 4.3376e-05 - val_ac
curacy: 0.9765 - val_loss: 0.1516
Epoch 91/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 8.8117e-05 - val_ac
curacy: 0.9739 - val_loss: 0.1380
Epoch 92/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 9.6849e-05 - val_ac
curacy: 0.9765 - val_loss: 0.1482
Epoch 93/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 7.9653e-05 - val_ac
curacy: 0.9765 - val_loss: 0.1423
Epoch 94/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 3.4995e-05 - val_ac
curacy: 0.9739 - val_loss: 0.1375
Epoch 95/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 6.5556e-05 - val_ac
curacy: 0.9765 - val_loss: 0.1445
Epoch 96/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 6.5782e-05 - val_ac
curacy: 0.9739 - val_loss: 0.1381
Epoch 97/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 9.0471e-05 - val_ac
curacy: 0.9739 - val_loss: 0.1428
Epoch 98/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 8.1186e-05 - val_ac
curacy: 0.9765 - val_loss: 0.1463
Epoch 99/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 6.5596e-05 - val_ac
curacy: 0.9765 - val_loss: 0.1399
Epoch 100/100
77/77 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 1.3968e-04 - val_ac
curacy: 0.9765 - val_loss: 0.1500
```

In [ ]:
```
# Comparacion Modelo 1 2 y 3 Accuracy y Loss

plt.plot(history.history['accuracy'], label='Model 1 Accuracy')
plt.plot(history2.history['accuracy'], label='Model 2 Accuracy')
plt.plot(history3.history['accuracy'], label='Model 3 Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Model Accuracy Comparison')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Model 1 Loss')
plt.plot(history2.history['loss'], label='Model 2 Loss')
plt.plot(history3.history['loss'], label='Model 3 Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Loss Comparison')
plt.legend()
plt.show()
```





In [21]:
```python
model1_loss, model1_acc = model.evaluate(X_test, y_test)
model2_loss, model2_acc = model2.evaluate(X_test, y_test)
model3_loss, model3_acc = model3.evaluate(X_test, y_test)

results = pd.DataFrame({
    'Model': ['Model 1', 'Model 2', 'Model 3'],
    'Loss': [model1_loss, model2_loss, model3_loss],
    'Accuracy': [model1_acc, model2_acc, model3_acc]
})

print("Performance of each model:")
print(results)
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9781 - loss: 0.0886
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9845 - loss: 0.0732
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9660 - loss: 0.2095
Performance of each model:
     Model      Loss  Accuracy
0  Model 1  0.091103  0.970772
1  Model 2  0.102304  0.977035
2  Model 3  0.213286  0.966597
```

In [22]:
```python
five_students = X_test[:5]
five_students_true = y_test[:5]

model1_predictions = model.predict(five_students)
model2_predictions = model2.predict(five_students)
model3_predictions = model3.predict(five_students)

model1_predicted_classes = np.argmax(model1_predictions, axis=1)
model2_predicted_classes = np.argmax(model2_predictions, axis=1)
```

```python
model3_predicted_classes = np.argmax(model3_predictions, axis=1)

predictions_df = pd.DataFrame({
    'True Label': five_students_true,
    'Model 1 Prediction': model1_predicted_classes,
    'Model 2 Prediction': model2_predicted_classes,
    'Model 3 Prediction': model3_predicted_classes
})

print("Predictions for 5 students:")
print(predictions_df)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 108ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 101ms/step
Predictions for 5 students:
      True Label  Model 1 Prediction  Model 2 Prediction  Model 3 Prediction
1004           1                   1                   1                   1
196            2                   2                   2                   2
2342           2                   2                   2                   2
1708           0                   0                   0                   0
435            1                   1                   1                   1
```