LTSM Neural Network

By: Parker Christenson

This is going to the the attempt to get the Long short term memory neural network to work on the data set. I will be using the Keras library to build the model.

I will be attempting to optimize the model by using a variety of different hyperparameters, and tunning the model to get the best results possible. By doing so, I hope that I will be able to achevive the best results possible.

Importing the needed Libraries for Model development

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import time
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import tensorflow as tf
```

```
WARNING:tensorflow:From c:\Users\tehwh\anaconda3\Lib\site-packages\keras\src\losses.
py:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use t
f.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [ ]:
```
df = pd.read_csv('Occupancy.csv')
df.head()
```

Out[ ]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
0	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1
1	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1
2	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1
3	2015-02-02 14:22:00	23.7225	26.125	493.750000	774.750000	0.004744	1
4	2015-02-02 14:23:00	23.7540	26.200	488.600000	779.000000	0.004767	1

In [ ]:
```
# getting the data types of the columns
df.dtypes
```

Out[ ]:
```
date              object
Temperature      float64
Humidity         float64
Light            float64
CO2              float64
HumidityRatio    float64
Occupancy          int64
dtype: object
```

In [ ]:
```
# Checking for missing values
df.isnull().sum()
```

Out[ ]:
```
date             0
Temperature      0
Humidity         0
Light            0
CO2              0
HumidityRatio    0
Occupancy        0
dtype: int64
```

In [ ]:
```
# Checking for duplicates
df.duplicated().sum()
```

Out[ ]:
```
0
```

We are now going to do some basic feature extraction and data preprocessing. Our goal here is to pull out as much

relevant information as possible from the data set, which in theory should help our model perform better

In [ ]:
```
# making a new column called 'time' from the 'date' column
df['time'] = pd.to_datetime(df['date']).dt.time
```

In [ ]:
```
# Getting the 'Temperature' and 'Humidity' ratio and making a new column called 'Te
df['Temp_Hum_Ratio'] = df['Temperature'] / df['Humidity']
df.head()
```

Out[ ]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	
0	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1	14
1	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1	14
2	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1	14
3	2015-02-02 14:22:00	23.7225	26.125	493.750000	774.750000	0.004744	1	14
4	2015-02-02 14:23:00	23.7540	26.200	488.600000	779.000000	0.004767	1	14

In [ ]:
```
# getting the weekday from the 'date' column
df['DOTW'] = pd.to_datetime(df['date']).dt.day_name()
df.head()
```

Out[ ]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	
0	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1	14
1	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1	14
2	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1	14
3	2015-02-02 14:22:00	23.7225	26.125	493.750000	774.750000	0.004744	1	14
4	2015-02-02 14:23:00	23.7540	26.200	488.600000	779.000000	0.004767	1	14

We can not go extremely heavy on the Feature extraction

Why can't you go heavy on the feature extraction?

This data set is going to train our IoT device to predict the next value in the sequence. This means that we need to be able to extract the most relevant information from the data set, and then use that information to train the model. If you start having lots and lots of features, then we are going to have to add more CPU and memory to the model, which is going to make it more expensive to run, and build.

In [ ]:
```
# making the column 'DOTW' to be catergorical and will now be changed to numerical
day_mapping = {
    'Monday': 1,
    'Tuesday': 2,
    'Wednesday': 3,
    'Thursday': 4,
    'Friday': 5,
    'Saturday': 6,
    'Sunday': 7
}

# now mapping the new values to the 'DOTW' column
df['DOTW_encoded'] = df['DOTW'].map(day_mapping)
```

In [ ]:
```
# setting our target variable to be the 'Occupancy' column
y = df['Occupancy']
```

```
# setting our features to be the 'Temperature', 'Humidity', 'Light', 'CO2', 'Humidi
X = df[['Temperature', 'Humidity', 'Light', 'CO2', 'HumidityRatio', 'Temp_Hum_Ratio
```

In [ ]:
```
# Splitting the data into training and testing sets to keep it linear when we are o

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

We are going to make the models and only change one of the features at a time, and then see how well the model performs. We are going to start by making changes to the layers in the model. We are looking to achevive a model thats accuracy is around 93% or higher, for us to switch and move onto the next feature.

Model 1: One input layer, one hidden layer with 50 Neurons, and one output layer.

In [ ]:
```
# Setting the LTSM model to be one input layer, one hidden layer and one output lay
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(50, input_dim=7, activation='relu'),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
WARNING:tensorflow:From c:\Users\tehwh\anaconda3\Lib\site-packages\keras\src\backen
d.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_d
efault_graph instead.
```

In [ ]:
```
# model 2 with 50 neurons in the first, 100 in the second and then 50 in the third

model_2 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

In [ ]:
```
# model 3 with 50 neurons in the first, 100 in the second and then 100 in the third

model_3 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

Model 1 Creation and Evaluation

In [ ]:
```
# model 1
model.compile(optimizer='adam',
```

```
              loss='binary_crossentropy',
              metrics=['accuracy'])

# training
start_time = time.time()
history = model.fit(X_train, y_train, epochs=50, validation_split=0.2, batch_size=1
end_time = time.time()
```

```
WARNING:tensorflow:From c:\Users\tehwh\anaconda3\Lib\site-packages\keras\src\optimiz
ers\__init__.py:309: The name tf.train.Optimizer is deprecated. Please use tf.compa
t.v1.train.Optimizer instead.

Epoch 1/50
WARNING:tensorflow:From c:\Users\tehwh\anaconda3\Lib\site-packages\keras\src\utils\t
f_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.co
mpat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From c:\Users\tehwh\anaconda3\Lib\site-packages\keras\src\engine
\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is depreca
ted. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

1316/1316 [==============================] - 3s 1ms/step - loss: 0.2616 - accuracy:
0.9559 - val_loss: 0.1471 - val_accuracy: 0.9723
Epoch 2/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.1349 - accuracy:
0.9695 - val_loss: 0.0929 - val_accuracy: 0.9790
Epoch 3/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0974 - accuracy:
0.9777 - val_loss: 0.0598 - val_accuracy: 0.9863
Epoch 4/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0836 - accuracy:
0.9804 - val_loss: 0.2113 - val_accuracy: 0.9644
Epoch 5/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0859 - accuracy:
0.9812 - val_loss: 0.2794 - val_accuracy: 0.9574
Epoch 6/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0715 - accuracy:
0.9837 - val_loss: 0.0735 - val_accuracy: 0.9845
Epoch 7/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0657 - accuracy:
0.9853 - val_loss: 0.0601 - val_accuracy: 0.9860
Epoch 8/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0614 - accuracy:
0.9868 - val_loss: 0.1071 - val_accuracy: 0.9742
Epoch 9/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0619 - accuracy:
0.9854 - val_loss: 0.0569 - val_accuracy: 0.9854
Epoch 10/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0518 - accuracy:
0.9892 - val_loss: 0.0550 - val_accuracy: 0.9860
Epoch 11/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0507 - accuracy:
0.9887 - val_loss: 0.0515 - val_accuracy: 0.9860
Epoch 12/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0480 - accuracy:
0.9889 - val_loss: 0.0492 - val_accuracy: 0.9857
Epoch 13/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0457 - accuracy:
0.9894 - val_loss: 0.0499 - val_accuracy: 0.9860
Epoch 14/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0467 - accuracy:
0.9887 - val_loss: 0.0503 - val_accuracy: 0.9866
Epoch 15/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0448 - accuracy:
```

```
0.9894 - val_loss: 0.0522 - val_accuracy: 0.9857
Epoch 16/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0461 - accuracy:
0.9892 - val_loss: 0.0490 - val_accuracy: 0.9866
Epoch 17/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0457 - accuracy:
0.9885 - val_loss: 0.0537 - val_accuracy: 0.9851
Epoch 18/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0449 - accuracy:
0.9890 - val_loss: 0.0497 - val_accuracy: 0.9860
Epoch 19/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0438 - accuracy:
0.9895 - val_loss: 0.0473 - val_accuracy: 0.9866
Epoch 20/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0450 - accuracy:
0.9891 - val_loss: 0.0538 - val_accuracy: 0.9860
Epoch 21/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0432 - accuracy:
0.9895 - val_loss: 0.0488 - val_accuracy: 0.9860
Epoch 22/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0444 - accuracy:
0.9891 - val_loss: 0.0587 - val_accuracy: 0.9805
Epoch 23/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0440 - accuracy:
0.9894 - val_loss: 0.0599 - val_accuracy: 0.9830
Epoch 24/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0447 - accuracy:
0.9894 - val_loss: 0.0508 - val_accuracy: 0.9857
Epoch 25/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0445 - accuracy:
0.9890 - val_loss: 0.0498 - val_accuracy: 0.9860
Epoch 26/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0434 - accuracy:
0.9898 - val_loss: 0.0640 - val_accuracy: 0.9793
Epoch 27/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0455 - accuracy:
0.9890 - val_loss: 0.0481 - val_accuracy: 0.9863
Epoch 28/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0446 - accuracy:
0.9887 - val_loss: 0.0486 - val_accuracy: 0.9866
Epoch 29/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0434 - accuracy:
0.9897 - val_loss: 0.0539 - val_accuracy: 0.9860
Epoch 30/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0440 - accuracy:
0.9892 - val_loss: 0.0556 - val_accuracy: 0.9860
Epoch 31/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0501 - accuracy:
0.9891 - val_loss: 0.0483 - val_accuracy: 0.9866
Epoch 32/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0438 - accuracy:
0.9894 - val_loss: 0.0492 - val_accuracy: 0.9860
Epoch 33/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0434 - accuracy:
0.9896 - val_loss: 0.0479 - val_accuracy: 0.9866
Epoch 34/50
```

```
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0426 - accuracy:
0.9897 - val_loss: 0.0518 - val_accuracy: 0.9860
Epoch 35/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0434 - accuracy:
0.9893 - val_loss: 0.0481 - val_accuracy: 0.9866
Epoch 36/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0423 - accuracy:
0.9896 - val_loss: 0.0524 - val_accuracy: 0.9866
Epoch 37/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0436 - accuracy:
0.9894 - val_loss: 0.0541 - val_accuracy: 0.9866
Epoch 38/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0427 - accuracy:
0.9894 - val_loss: 0.0538 - val_accuracy: 0.9842
Epoch 39/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0427 - accuracy:
0.9896 - val_loss: 0.0478 - val_accuracy: 0.9860
Epoch 40/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0430 - accuracy:
0.9897 - val_loss: 0.0508 - val_accuracy: 0.9860
Epoch 41/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0431 - accuracy:
0.9894 - val_loss: 0.0472 - val_accuracy: 0.9866
Epoch 42/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0419 - accuracy:
0.9901 - val_loss: 0.0487 - val_accuracy: 0.9860
Epoch 43/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0421 - accuracy:
0.9897 - val_loss: 0.0537 - val_accuracy: 0.9869
Epoch 44/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0427 - accuracy:
0.9891 - val_loss: 0.0599 - val_accuracy: 0.9848
Epoch 45/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0438 - accuracy:
0.9894 - val_loss: 0.0488 - val_accuracy: 0.9860
Epoch 46/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0422 - accuracy:
0.9894 - val_loss: 0.0476 - val_accuracy: 0.9860
Epoch 47/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0456 - accuracy:
0.9896 - val_loss: 0.0497 - val_accuracy: 0.9866
Epoch 48/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0425 - accuracy:
0.9897 - val_loss: 0.0692 - val_accuracy: 0.9769
Epoch 49/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.0431 - accuracy:
0.9894 - val_loss: 0.0481 - val_accuracy: 0.9863
Epoch 50/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0425 - accuracy:
0.9895 - val_loss: 0.0495 - val_accuracy: 0.9869
```

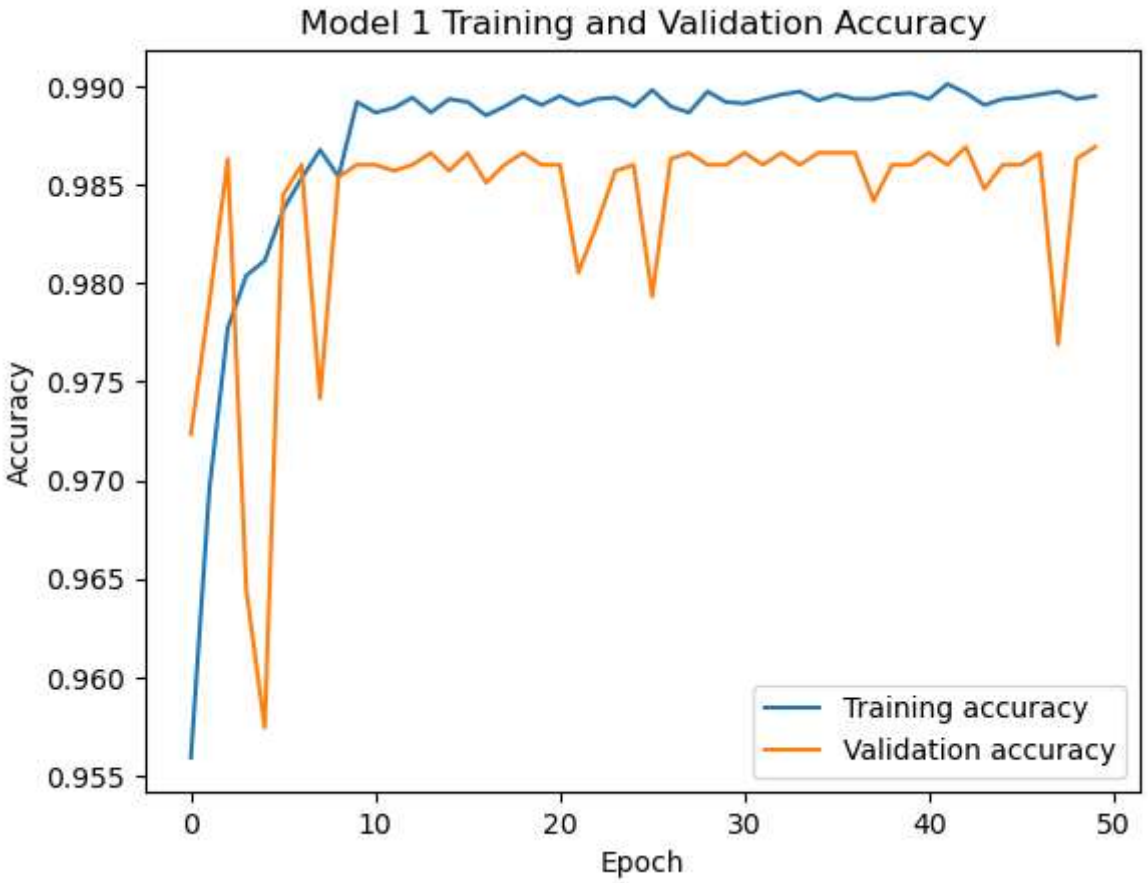
In [ ]:
```python
# calculating the training time
training_time = end_time - start_time
print(f"Model 1 Training Time: {training_time:.2f} seconds")
```

```
Model 1 Training Time: 75.78 seconds
```

In [ ]:
```
# eval model 1
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=1)
print(f"Model 1 Test Accuracy: {test_accuracy*100:.2f}%")
```

```
129/129 [==============================] - 0s 1ms/step - loss: 0.0338 - accuracy: 0.
9929
Model 1 Test Accuracy: 99.29%
```

In [ ]:
```
# plotting
plt.plot(history.history['accuracy'], label='Training accuracy')
plt.plot(history.history['val_accuracy'], label='Validation accuracy')
plt.title('Model 1 Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```



Model 2 Creation and Evaluation

In [ ]:
```
# making model 2
model_2.compile(optimizer='adam',
                loss='binary_crossentropy',
                metrics=['accuracy'])

# training model 2
start_time = time.time()  # Start time for training
```

```
history_2 = model_2.fit(X_train, y_train, epochs=50, validation_split=0.2, batch_si
end_time = time.time()  # End time for training
```

```
Epoch 1/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.4022 - accuracy:
0.9476 - val_loss: 0.1011 - val_accuracy: 0.9778
Epoch 2/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.1261 - accuracy:
0.9732 - val_loss: 0.0707 - val_accuracy: 0.9857
Epoch 3/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0879 - accuracy:
0.9793 - val_loss: 0.0639 - val_accuracy: 0.9863
Epoch 4/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0754 - accuracy:
0.9807 - val_loss: 0.0569 - val_accuracy: 0.9857
Epoch 5/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0621 - accuracy:
0.9863 - val_loss: 0.0872 - val_accuracy: 0.9720
Epoch 6/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0564 - accuracy:
0.9869 - val_loss: 0.0529 - val_accuracy: 0.9866
Epoch 7/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0624 - accuracy:
0.9852 - val_loss: 0.0541 - val_accuracy: 0.9860
Epoch 8/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0555 - accuracy:
0.9882 - val_loss: 0.0560 - val_accuracy: 0.9866
Epoch 9/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0499 - accuracy:
0.9884 - val_loss: 0.0606 - val_accuracy: 0.9857
Epoch 10/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0484 - accuracy:
0.9890 - val_loss: 0.0637 - val_accuracy: 0.9769
Epoch 11/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0496 - accuracy:
0.9882 - val_loss: 0.0555 - val_accuracy: 0.9866
Epoch 12/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0475 - accuracy:
0.9891 - val_loss: 0.0600 - val_accuracy: 0.9799
Epoch 13/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0485 - accuracy:
0.9882 - val_loss: 0.0512 - val_accuracy: 0.9860
Epoch 14/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0462 - accuracy:
0.9892 - val_loss: 0.0575 - val_accuracy: 0.9860
Epoch 15/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0472 - accuracy:
0.9888 - val_loss: 0.0522 - val_accuracy: 0.9863
Epoch 16/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0482 - accuracy:
0.9884 - val_loss: 0.0527 - val_accuracy: 0.9866
Epoch 17/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0476 - accuracy:
0.9888 - val_loss: 0.0501 - val_accuracy: 0.9860
Epoch 18/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0462 - accuracy:
0.9891 - val_loss: 0.0488 - val_accuracy: 0.9872
Epoch 19/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0448 - accuracy:
```

```
0.9894 - val_loss: 0.0622 - val_accuracy: 0.9809
Epoch 20/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0449 - accuracy:
0.9894 - val_loss: 0.0522 - val_accuracy: 0.9860
Epoch 21/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0462 - accuracy:
0.9889 - val_loss: 0.0503 - val_accuracy: 0.9860
Epoch 22/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0490 - accuracy:
0.9875 - val_loss: 0.0565 - val_accuracy: 0.9827
Epoch 23/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0463 - accuracy:
0.9899 - val_loss: 0.0498 - val_accuracy: 0.9866
Epoch 24/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0459 - accuracy:
0.9896 - val_loss: 0.0569 - val_accuracy: 0.9857
Epoch 25/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0447 - accuracy:
0.9894 - val_loss: 0.0495 - val_accuracy: 0.9857
Epoch 26/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0462 - accuracy:
0.9895 - val_loss: 0.0499 - val_accuracy: 0.9860
Epoch 27/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0443 - accuracy:
0.9897 - val_loss: 0.0485 - val_accuracy: 0.9857
Epoch 28/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0443 - accuracy:
0.9896 - val_loss: 0.0483 - val_accuracy: 0.9866
Epoch 29/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0488 - accuracy:
0.9887 - val_loss: 0.0547 - val_accuracy: 0.9839
Epoch 30/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0439 - accuracy:
0.9894 - val_loss: 0.0491 - val_accuracy: 0.9860
Epoch 31/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0442 - accuracy:
0.9893 - val_loss: 0.0529 - val_accuracy: 0.9860
Epoch 32/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0442 - accuracy:
0.9897 - val_loss: 0.0523 - val_accuracy: 0.9866
Epoch 33/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0437 - accuracy:
0.9898 - val_loss: 0.0588 - val_accuracy: 0.9857
Epoch 34/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0441 - accuracy:
0.9897 - val_loss: 0.0538 - val_accuracy: 0.9836
Epoch 35/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0426 - accuracy:
0.9900 - val_loss: 0.0638 - val_accuracy: 0.9860
Epoch 36/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0454 - accuracy:
0.9887 - val_loss: 0.0495 - val_accuracy: 0.9866
Epoch 37/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0437 - accuracy:
0.9897 - val_loss: 0.0537 - val_accuracy: 0.9860
Epoch 38/50
```

```
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0424 - accuracy:
0.9894 - val_loss: 0.0514 - val_accuracy: 0.9860
Epoch 39/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0431 - accuracy:
0.9899 - val_loss: 0.0560 - val_accuracy: 0.9839
Epoch 40/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0434 - accuracy:
0.9894 - val_loss: 0.0511 - val_accuracy: 0.9866
Epoch 41/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0439 - accuracy:
0.9894 - val_loss: 0.0494 - val_accuracy: 0.9857
Epoch 42/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0434 - accuracy:
0.9899 - val_loss: 0.0506 - val_accuracy: 0.9860
Epoch 43/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0454 - accuracy:
0.9893 - val_loss: 0.0481 - val_accuracy: 0.9860
Epoch 44/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0429 - accuracy:
0.9897 - val_loss: 0.0494 - val_accuracy: 0.9860
Epoch 45/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0424 - accuracy:
0.9897 - val_loss: 0.0492 - val_accuracy: 0.9860
Epoch 46/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0428 - accuracy:
0.9899 - val_loss: 0.0527 - val_accuracy: 0.9860
Epoch 47/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0440 - accuracy:
0.9893 - val_loss: 0.0499 - val_accuracy: 0.9866
Epoch 48/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0426 - accuracy:
0.9897 - val_loss: 0.0499 - val_accuracy: 0.9857
Epoch 49/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0421 - accuracy:
0.9897 - val_loss: 0.0535 - val_accuracy: 0.9866
Epoch 50/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0430 - accuracy:
0.9898 - val_loss: 0.0535 - val_accuracy: 0.9860
```

In [ ]:
```python
# calc and print the training time
training_time_2 = end_time - start_time
print(f"Model 2 Training Time: {training_time_2:.2f} seconds")
```

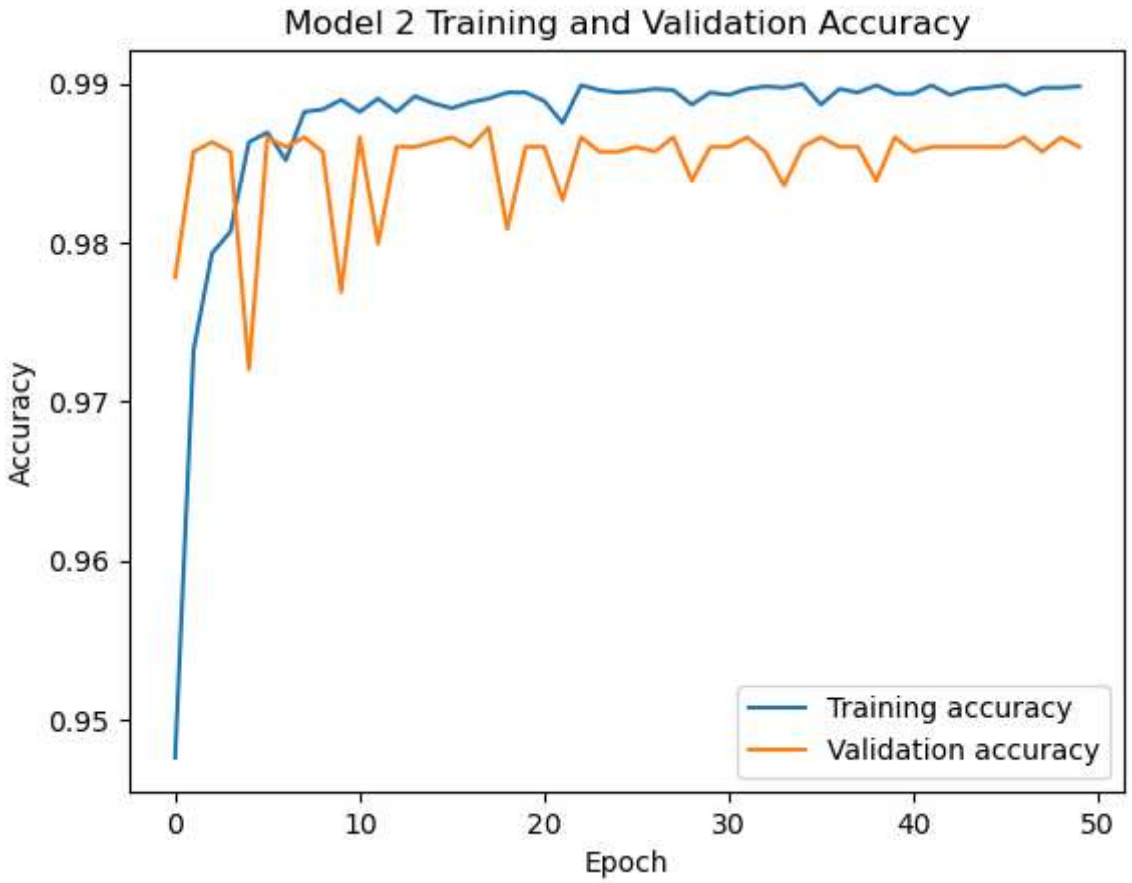
```
Model 2 Training Time: 83.60 seconds
```

In [ ]:
```python
# eval model 2
test_loss_2, test_accuracy_2 = model_2.evaluate(X_test, y_test, verbose=1)
print(f"Model 2 Test Accuracy: {test_accuracy_2*100:.2f}%")
```

```
129/129 [==============================] - 0s 1ms/step - loss: 0.0327 - accuracy: 0.
9925
Model 2 Test Accuracy: 99.25%
```

In [ ]:
```python
# plotting ltsm 2
plt.plot(history_2.history['accuracy'], label='Training accuracy')
plt.plot(history_2.history['val_accuracy'], label='Validation accuracy')
plt.title('Model 2 Training and Validation Accuracy')
```

```
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```



Model 3 Creation and Evaluation

In [ ]:
```
# making the model
model_3.compile(optimizer='adam',
                loss='binary_crossentropy',
                metrics=['accuracy'])

# traininig
start_time_m3a = time.time()  # Start time for training
history_3_adam = model_3.fit(X_train, y_train, epochs=50, validation_split=0.2, bat
end_time_m3a = time.time()  # End time for training
```

```
Epoch 1/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.6693 - accuracy:
0.9419 - val_loss: 0.0921 - val_accuracy: 0.9766
Epoch 2/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.1208 - accuracy:
0.9707 - val_loss: 0.0749 - val_accuracy: 0.9781
Epoch 3/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0759 - accuracy:
0.9786 - val_loss: 0.0687 - val_accuracy: 0.9763
Epoch 4/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0709 - accuracy:
0.9819 - val_loss: 0.0704 - val_accuracy: 0.9863
Epoch 5/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0737 - accuracy:
0.9816 - val_loss: 0.0630 - val_accuracy: 0.9857
Epoch 6/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0602 - accuracy:
0.9879 - val_loss: 0.0727 - val_accuracy: 0.9824
Epoch 7/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0570 - accuracy:
0.9867 - val_loss: 0.0659 - val_accuracy: 0.9860
Epoch 8/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0562 - accuracy:
0.9879 - val_loss: 0.0565 - val_accuracy: 0.9866
Epoch 9/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0538 - accuracy:
0.9878 - val_loss: 0.0550 - val_accuracy: 0.9857
Epoch 10/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0552 - accuracy:
0.9875 - val_loss: 0.0705 - val_accuracy: 0.9821
Epoch 11/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0519 - accuracy:
0.9887 - val_loss: 0.0567 - val_accuracy: 0.9857
Epoch 12/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0513 - accuracy:
0.9886 - val_loss: 0.0641 - val_accuracy: 0.9845
Epoch 13/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0507 - accuracy:
0.9882 - val_loss: 0.0533 - val_accuracy: 0.9857
Epoch 14/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0508 - accuracy:
0.9879 - val_loss: 0.0528 - val_accuracy: 0.9866
Epoch 15/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0487 - accuracy:
0.9888 - val_loss: 0.0544 - val_accuracy: 0.9860
Epoch 16/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0489 - accuracy:
0.9890 - val_loss: 0.0523 - val_accuracy: 0.9857
Epoch 17/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0476 - accuracy:
0.9895 - val_loss: 0.0555 - val_accuracy: 0.9860
Epoch 18/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0511 - accuracy:
0.9873 - val_loss: 0.0546 - val_accuracy: 0.9857
Epoch 19/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0465 - accuracy:
```

```
0.9891 - val_loss: 0.0585 - val_accuracy: 0.9860
Epoch 20/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0477 - accuracy:
0.9888 - val_loss: 0.0613 - val_accuracy: 0.9842
Epoch 21/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0485 - accuracy:
0.9886 - val_loss: 0.0526 - val_accuracy: 0.9860
Epoch 22/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0491 - accuracy:
0.9888 - val_loss: 0.0533 - val_accuracy: 0.9857
Epoch 23/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0462 - accuracy:
0.9888 - val_loss: 0.0570 - val_accuracy: 0.9857
Epoch 24/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0474 - accuracy:
0.9894 - val_loss: 0.0621 - val_accuracy: 0.9802
Epoch 25/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0483 - accuracy:
0.9891 - val_loss: 0.0638 - val_accuracy: 0.9848
Epoch 26/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0481 - accuracy:
0.9887 - val_loss: 0.0534 - val_accuracy: 0.9857
Epoch 27/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0469 - accuracy:
0.9891 - val_loss: 0.0704 - val_accuracy: 0.9812
Epoch 28/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0458 - accuracy:
0.9894 - val_loss: 0.0541 - val_accuracy: 0.9857
Epoch 29/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0471 - accuracy:
0.9894 - val_loss: 0.0536 - val_accuracy: 0.9860
Epoch 30/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0520 - accuracy:
0.9880 - val_loss: 0.0514 - val_accuracy: 0.9866
Epoch 31/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0471 - accuracy:
0.9895 - val_loss: 0.0535 - val_accuracy: 0.9863
Epoch 32/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0487 - accuracy:
0.9890 - val_loss: 0.0669 - val_accuracy: 0.9815
Epoch 33/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0531 - accuracy:
0.9868 - val_loss: 0.0543 - val_accuracy: 0.9866
Epoch 34/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0471 - accuracy:
0.9891 - val_loss: 0.0612 - val_accuracy: 0.9848
Epoch 35/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0459 - accuracy:
0.9891 - val_loss: 0.0530 - val_accuracy: 0.9857
Epoch 36/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0463 - accuracy:
0.9888 - val_loss: 0.0559 - val_accuracy: 0.9860
Epoch 37/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0456 - accuracy:
0.9895 - val_loss: 0.0569 - val_accuracy: 0.9857
Epoch 38/50
```

```
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0465 - accuracy:
0.9891 - val_loss: 0.0602 - val_accuracy: 0.9857
Epoch 39/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0505 - accuracy:
0.9888 - val_loss: 0.0531 - val_accuracy: 0.9866
Epoch 40/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0451 - accuracy:
0.9892 - val_loss: 0.0564 - val_accuracy: 0.9860
Epoch 41/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0474 - accuracy:
0.9889 - val_loss: 0.0523 - val_accuracy: 0.9857
Epoch 42/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0479 - accuracy:
0.9895 - val_loss: 0.0535 - val_accuracy: 0.9857
Epoch 43/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0464 - accuracy:
0.9891 - val_loss: 0.0568 - val_accuracy: 0.9860
Epoch 44/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0442 - accuracy:
0.9897 - val_loss: 0.0534 - val_accuracy: 0.9857
Epoch 45/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0462 - accuracy:
0.9896 - val_loss: 0.0525 - val_accuracy: 0.9863
Epoch 46/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0481 - accuracy:
0.9888 - val_loss: 0.0540 - val_accuracy: 0.9866
Epoch 47/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0469 - accuracy:
0.9891 - val_loss: 0.0535 - val_accuracy: 0.9857
Epoch 48/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0460 - accuracy:
0.9891 - val_loss: 0.0524 - val_accuracy: 0.9866
Epoch 49/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0462 - accuracy:
0.9888 - val_loss: 0.0536 - val_accuracy: 0.9860
Epoch 50/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.0451 - accuracy:
0.9894 - val_loss: 0.0537 - val_accuracy: 0.9857
```

In [ ]:
```python
# calc and print the training time
training_time_3 = end_time_m3a - start_time_m3a
print(f"Model 3 Training Time: {training_time_3:.2f} seconds")
```

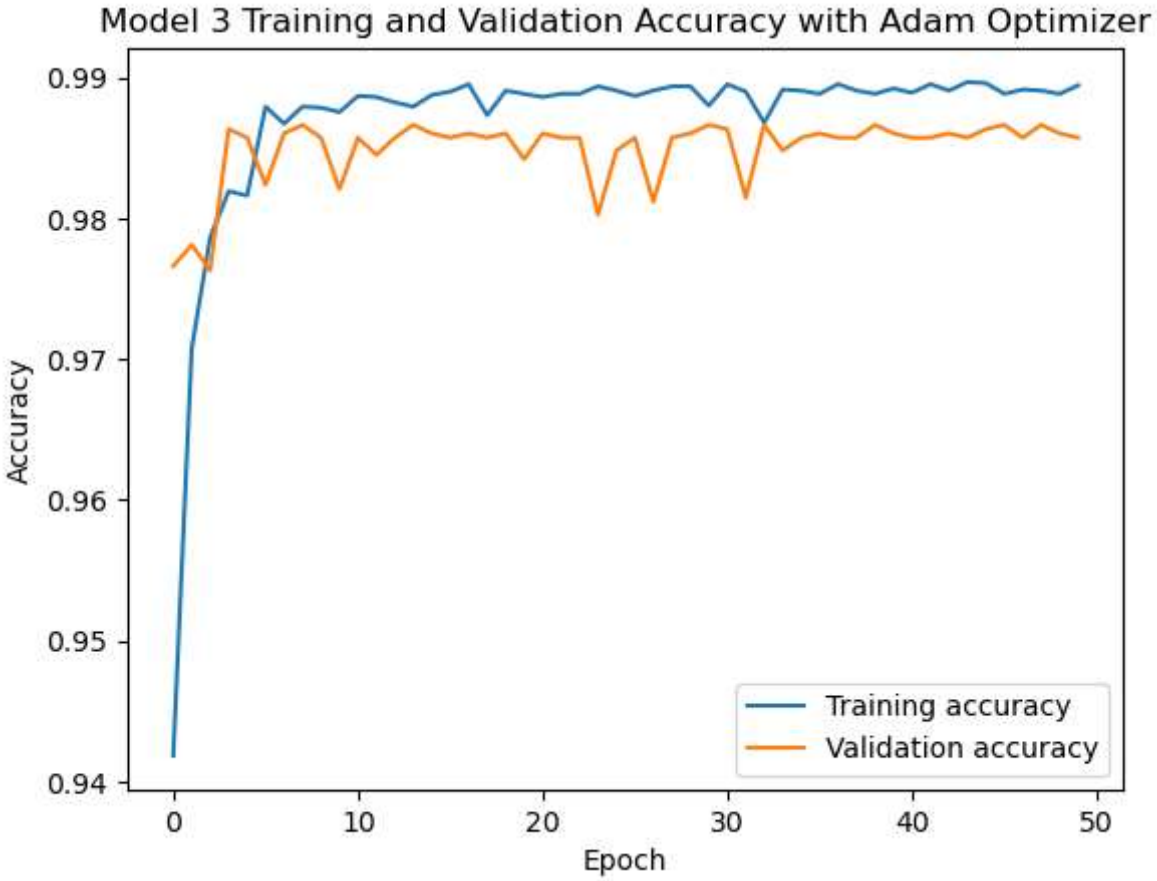
```
Model 3 Training Time: 80.06 seconds
```

In [ ]:
```python
# eval model 3
test_loss_3, test_accuracy_3 = model_3.evaluate(X_test, y_test, verbose=1)
print(f"Model 3 Test Accuracy: {test_accuracy_3*100:.2f}%")
```

```
129/129 [==============================] - 0s 1ms/step - loss: 0.0374 - accuracy: 0.
9917
Model 3 Test Accuracy: 99.17%
```

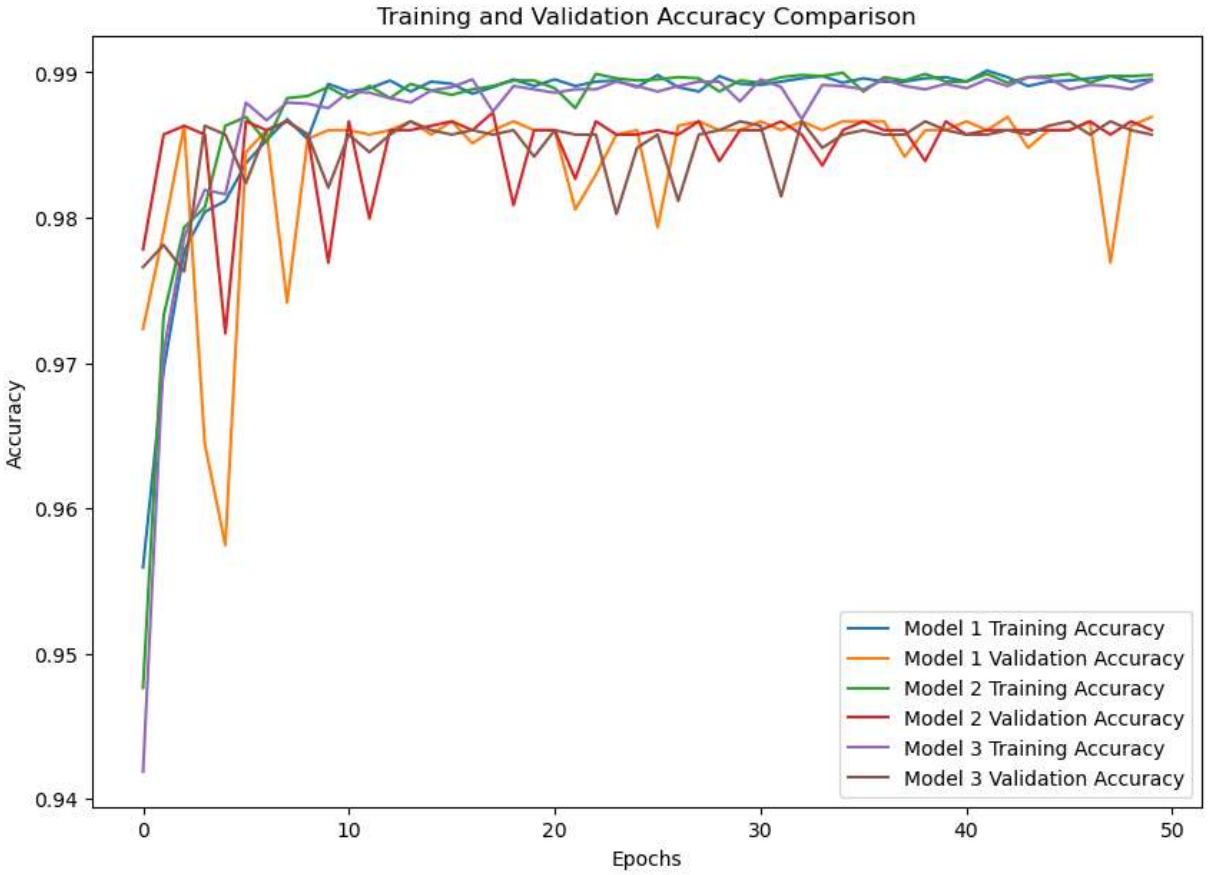
In [ ]:
```python
# model 3 plot
plt.plot(history_3_adam.history['accuracy'], label='Training accuracy')
plt.plot(history_3_adam.history['val_accuracy'], label='Validation accuracy')
plt.title('Model 3 Training and Validation Accuracy with Adam Optimizer')
```

```
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()
plt.show()
```



Comparing all 3 of the Models against eachother

In [ ]:
```
# all 3 models plotted
plt.figure(figsize=(10, 7))
plt.plot(history.history['accuracy'], label='Model 1 Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Model 1 Validation Accuracy')
plt.plot(history_2.history['accuracy'], label='Model 2 Training Accuracy')
plt.plot(history_2.history['val_accuracy'], label='Model 2 Validation Accuracy')
plt.plot(history_3_adam.history['accuracy'], label='Model 3 Training Accuracy')
plt.plot(history_3_adam.history['val_accuracy'], label='Model 3 Validation Accuracy
plt.title('Training and Validation Accuracy Comparison')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



Model 3 had the best Training accuracy, and had the Validation. We are going to use model 3 to optimize and make perform better. Our best output was, 99.29% accurate. We are going to see if we can get that score even higher, just for the sake of optimization reasons.

Training with the SDG optimizer

In [ ]:
```
# re-doing model 3 with SDG optimizer
model_3_sgd = tf.keras.models.Sequential([
    tf.keras.layers.Dense(50, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

# Compile Model 3
model_3_sgd.compile(optimizer='sgd',
                    loss='binary_crossentropy',
                    metrics=['accuracy'])
```

In [ ]:
```
# training model 3 with SDG optimizer
history_3_sgd = model_3_sgd.fit(X_train, y_train, epochs=50, validation_split=0.2,
```

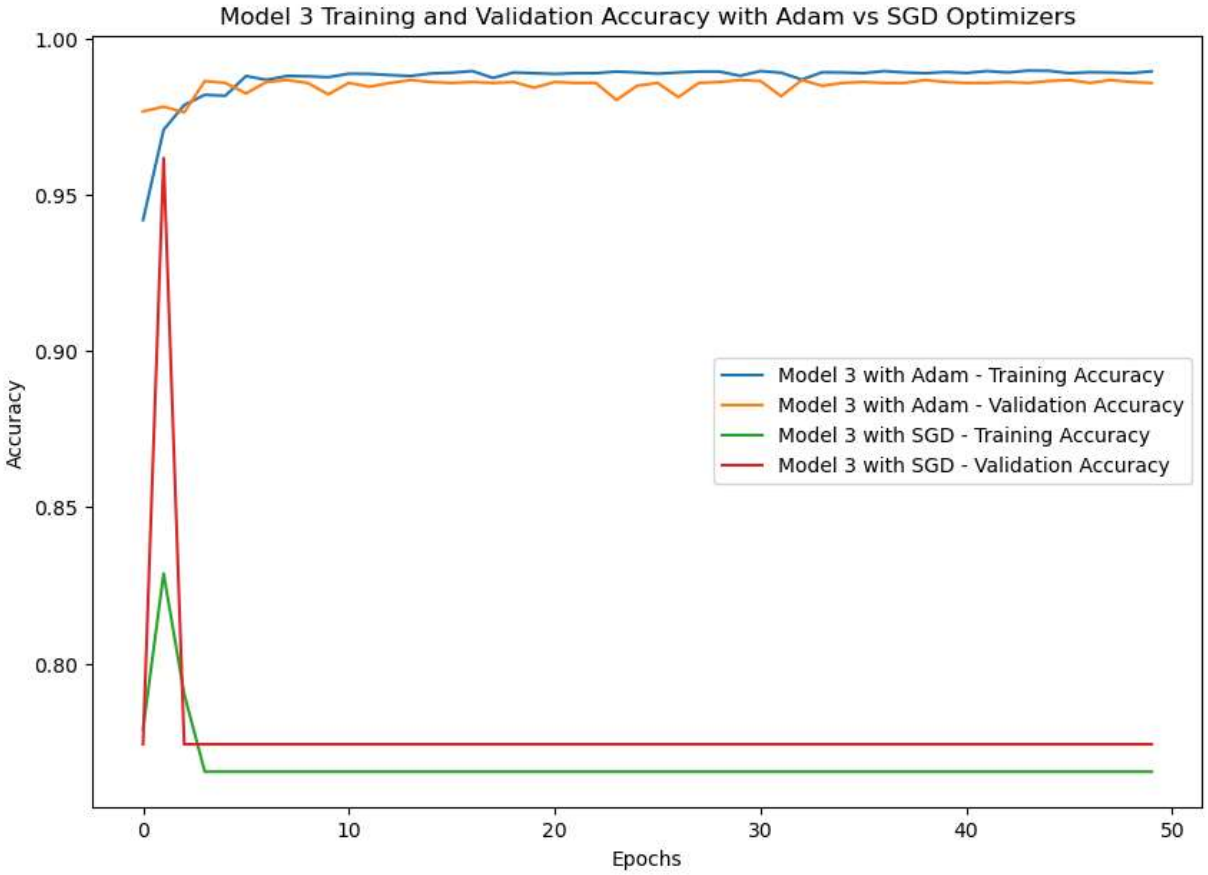
```
Epoch 1/50
1316/1316 [==============================] - 2s 1ms/step - loss: 1.2688 - accuracy:
0.7790 - val_loss: 0.5348 - val_accuracy: 0.7742
Epoch 2/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.4350 - accuracy:
0.8288 - val_loss: 0.1590 - val_accuracy: 0.9617
Epoch 3/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5018 - accuracy:
0.7902 - val_loss: 0.5346 - val_accuracy: 0.7742
Epoch 4/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5457 - accuracy:
0.7654 - val_loss: 0.5361 - val_accuracy: 0.7742
Epoch 5/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5460 - accuracy:
0.7654 - val_loss: 0.5348 - val_accuracy: 0.7742
Epoch 6/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5459 - accuracy:
0.7654 - val_loss: 0.5348 - val_accuracy: 0.7742
Epoch 7/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5458 - accuracy:
0.7654 - val_loss: 0.5347 - val_accuracy: 0.7742
Epoch 8/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5359 - val_accuracy: 0.7742
Epoch 9/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5457 - accuracy:
0.7654 - val_loss: 0.5344 - val_accuracy: 0.7742
Epoch 10/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5453 - accuracy:
0.7654 - val_loss: 0.5345 - val_accuracy: 0.7742
Epoch 11/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5344 - val_accuracy: 0.7742
Epoch 12/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5458 - accuracy:
0.7654 - val_loss: 0.5345 - val_accuracy: 0.7742
Epoch 13/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5344 - val_accuracy: 0.7742
Epoch 14/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5347 - val_accuracy: 0.7742
Epoch 15/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5457 - accuracy:
0.7654 - val_loss: 0.5342 - val_accuracy: 0.7742
Epoch 16/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5459 - accuracy:
0.7654 - val_loss: 0.5342 - val_accuracy: 0.7742
Epoch 17/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5380 - val_accuracy: 0.7742
Epoch 18/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5458 - accuracy:
0.7654 - val_loss: 0.5344 - val_accuracy: 0.7742
Epoch 19/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5456 - accuracy:
```

```
0.7654 - val_loss: 0.5349 - val_accuracy: 0.7742
Epoch 20/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5345 - val_accuracy: 0.7742
Epoch 21/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5457 - accuracy:
0.7654 - val_loss: 0.5343 - val_accuracy: 0.7742
Epoch 22/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5457 - accuracy:
0.7654 - val_loss: 0.5354 - val_accuracy: 0.7742
Epoch 23/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5345 - val_accuracy: 0.7742
Epoch 24/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5342 - val_accuracy: 0.7742
Epoch 25/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5450 - accuracy:
0.7654 - val_loss: 0.5356 - val_accuracy: 0.7742
Epoch 26/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5457 - accuracy:
0.7654 - val_loss: 0.5347 - val_accuracy: 0.7742
Epoch 27/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5344 - val_accuracy: 0.7742
Epoch 28/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5345 - val_accuracy: 0.7742
Epoch 29/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5342 - val_accuracy: 0.7742
Epoch 30/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5345 - val_accuracy: 0.7742
Epoch 31/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5362 - val_accuracy: 0.7742
Epoch 32/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5353 - val_accuracy: 0.7742
Epoch 33/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5352 - val_accuracy: 0.7742
Epoch 34/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5358 - val_accuracy: 0.7742
Epoch 35/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5457 - accuracy:
0.7654 - val_loss: 0.5364 - val_accuracy: 0.7742
Epoch 36/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5457 - accuracy:
0.7654 - val_loss: 0.5361 - val_accuracy: 0.7742
Epoch 37/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5342 - val_accuracy: 0.7742
Epoch 38/50
```

```
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5351 - val_accuracy: 0.7742
Epoch 39/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5353 - val_accuracy: 0.7742
Epoch 40/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5352 - val_accuracy: 0.7742
Epoch 41/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5362 - val_accuracy: 0.7742
Epoch 42/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5343 - val_accuracy: 0.7742
Epoch 43/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5352 - val_accuracy: 0.7742
Epoch 44/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5343 - val_accuracy: 0.7742
Epoch 45/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5455 - accuracy:
0.7654 - val_loss: 0.5344 - val_accuracy: 0.7742
Epoch 46/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5452 - accuracy:
0.7654 - val_loss: 0.5347 - val_accuracy: 0.7742
Epoch 47/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5342 - val_accuracy: 0.7742
Epoch 48/50
1316/1316 [==============================] - 2s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5344 - val_accuracy: 0.7742
Epoch 49/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5454 - accuracy:
0.7654 - val_loss: 0.5374 - val_accuracy: 0.7742
Epoch 50/50
1316/1316 [==============================] - 1s 1ms/step - loss: 0.5456 - accuracy:
0.7654 - val_loss: 0.5347 - val_accuracy: 0.7742
```

In [ ]:

```
# plotting the comparison
plt.figure(figsize=(10, 7))
plt.plot(history_3_adam.history['accuracy'], label='Model 3 with Adam - Training Ac
plt.plot(history_3_adam.history['val_accuracy'], label='Model 3 with Adam - Validat
plt.plot(history_3_sgd.history['accuracy'], label='Model 3 with SGD - Training Accu
plt.plot(history_3_sgd.history['val_accuracy'], label='Model 3 with SGD - Validatio
plt.title('Model 3 Training and Validation Accuracy with Adam vs SGD Optimizers')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```



In [ ]:
```
# creating a prediction column for the original data frame
df['Prediction'] = model_3.predict(X)
df.head()
```

```
643/643 [==============================] - 1s 721us/step
```

Out[ ]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	
0	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1	1
1	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1	1
2	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1	1
3	2015-02-02 14:22:00	23.7225	26.125	493.750000	774.750000	0.004744	1	1
4	2015-02-02 14:23:00	23.7540	26.200	488.600000	779.000000	0.004767	1	1

In [ ]:
```
# rounding the prediction column to 0 or 1
df['Prediction'] = df['Prediction'].round()
```

In [ ]:
```
# prinitng the coun tof the prediction column
df['Prediction'].value_counts()
```

Out[ ]:
```
Prediction
0.0    15616
1.0     4944
Name: count, dtype: int64
```

In [ ]:
```
# counting the actual values of the 'Occupancy' column
df['Occupancy'].value_counts()
```

Out[ ]:
```
Occupancy
0    15810
1     4750
Name: count, dtype: int64
```

In [ ]:
```
# saving the df back down to a csv file
df.to_csv('Occupancy_Predictions_LTSM_Model_3.csv', index=False)
```

In [ ]:
```
df.head()
```

Out[ ]:

	date	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy	
0	2015-02-02 14:19:00	23.7000	26.272	585.200000	749.200000	0.004764	1	14
1	2015-02-02 14:19:59	23.7180	26.290	578.400000	760.400000	0.004773	1	14
2	2015-02-02 14:21:00	23.7300	26.230	572.666667	769.666667	0.004765	1	14
3	2015-02-02 14:22:00	23.7225	26.125	493.750000	774.750000	0.004744	1	14
4	2015-02-02 14:23:00	23.7540	26.200	488.600000	779.000000	0.004767	1	14

In [ ]: