**Introduction**

This report delves into applying LSTM recurrent neural networks for time series prediction, multi-class text classification, and text generation using three distinct datasets. First, the background section will explain LSTMs, their functionality, and their advantages. Next, the Methods/Analysis section will elaborate on applying LSTMs to the three datasets. Finally, the conclusion section will summarize each model's results.

**Background**

LSTM, or Long Short-Term Memory, is a type of Recurrent Neural Network (RNN) that is capable of learning long-term dependencies. Traditional RNNs have a limitation in that they can only retain information for a short period of time, which can lead to difficulties in processing long sequences of data. LSTMs were designed to overcome this limitation and are now widely used in various applications, such as speech recognition, language translation, and video analysis.

The architecture of an LSTM network consists of four main components: the input gate, the forget gate, the memory cell, and the output gate. These components work together to allow LSTMs to selectively remember or forget information, depending on its relevance to the task at hand.

The input gate determines how much new information is allowed into the memory cell, while the forget gate determines how much old information should be discarded. The memory cell is responsible for storing information over time, and the output gate controls how much information is output from the cell to the next layer of the network.

The key difference between LSTMs and traditional RNNs is the use of a memory cell, which allows LSTMs to remember information over longer periods of time. The memory cell is able to selectively remember or forget information using the input and forget gates, which are controlled by activation functions such as the sigmoid and hyperbolic tangent functions.

Overall, LSTMs are a powerful tool for processing time series and other sequential data, and their ability to learn long-term dependencies has made them a popular choice for many applications in the field of machine learning.

**Methods/Analysis**

**Text Classification**

Our first classification model used BBC News articles for text classification, predicting the article's category: politics, entertainment, business, sports, or technology.

The code trains a text classification model using a Bidirectional LSTM on the BBC news dataset. The goal is to classify news articles into five categories: sports, business, politics, tech, and entertainment.

The code starts by importing the necessary libraries and downloading the dataset. It then preprocesses the data by removing stopwords and tokenizing the articles using the Keras Tokenizer. It also splits the data into training and validation sets, maintaining an 80-20 split.

The model is constructed using an Embedding layer followed by a Dropout layer, a Bidirectional LSTM layer, and a Dense layer with a softmax activation function for output. The model is then compiled using the sparse categorical cross-entropy loss function and the Adam optimizer with a learning rate of 0.001.

The model is trained for ten epochs using the padded training and validation data. After training, the model predicts the categories of two sample news articles. Finally, the model's prediction for each article and its corresponding category label are displayed.

**Time Series**

The second model involved time-series prediction on airline passenger data, aiming to predict the number of international airline passengers based on historical data. It uses the airline passengers dataset, which contains the monthly number of passengers from 1949 to 1960.

The code starts by importing necessary libraries and loading the dataset. It then normalizes the data using the MinMaxScaler to bring the values into the range of 0 to 1. Next, the dataset is split into training and test sets, with 67% of the data used for training and the remaining 33% for testing.

A helper function, create\_dataset(), is defined to convert the dataset into a format suitable for training an LSTM model. This function takes a dataset and a look-back value as input and creates input-output pairs by taking the current value as input (X) and the next value as output (Y).

A simple LSTM model is created with one LSTM layer containing four units, followed by a Dense layer for output. Finally, the model is compiled using mean squared error as the loss function and the Adam optimizer.

The model is trained for 100 epochs with a batch size of 1 using the training data. After training, predictions are made on both the training and test sets. The predictions are then inverted back to the original scale using the scaler.

The root means squared error (RMSE) is calculated for the training and test predictions to evaluate the model's performance.

**Nursery Rhyme LSTM**

The final model involved training an LSTM model on nursery rhymes to generate a new one. The process begins with preprocessing the text data, involving reading from a 'nursery\_rhymes.txt' file, replacing newline characters with spaces, converting text to lowercase, and removing unwanted characters. The preprocessed text is then tokenized, splitting it into individual words, which form the basis for the word-level mappings. Finally, two dictionaries are created, one for converting words to indices and another for converting indices back to words.

The dataset is prepared by creating sequences of fixed length (20 words) and their corresponding next words. Then, these sequences are encoded using word indices.

The LSTM model is constructed as a sequential model, starting with an Embedding layer with an input dimension equal to the number of unique words, an output dimension of 128, and an input length equal to the sequence length. The model also includes an LSTM layer with 128 units and a Dense layer with a softmax activation function for output.

The model is compiled using the categorical cross-entropy loss function and the RMSprop optimizer with a learning rate of 0.01. Two utility functions, 'sample()' and 'on\_epoch\_end(),' are defined for sampling indices from a probability array and generating text after each epoch during training, respectively. A callback is configured to generate text samples after every five epochs using the 'on\_epoch\_end()' function.

The model is trained for 20 epochs on the word-level dataset, with a batch size of 128 and the previously defined callback for displaying generated text during training. After training, the code generates a new nursery rhyme with 600 words (30 lines \* 20 words per line) using the trained model, starting from a randomly selected sequence from the original text. Then, the generated words are divided into 30 lines with 20 words each and presented as the final nursery rhyme. The generated text is displayed every five epochs, providing insights into the progress of the training process.

**Results**

**BBC Text Classification**

In the first model, our goal was to perform multi-class text classification on BBC News articles, categorizing them into one of five groups. Starting with a loss of 1.59 and an accuracy of 0.27, the model improved with each epoch, ultimately achieving a loss of 0.018 and an accuracy of 100% on the training data and a loss of 0.21 and an accuracy of 95% on the validation data. Overall our model performed well, accurately predicting a given text phrase's category.

**Airline Passenger Time Series**

Our second model focused on predicting the number of airline passengers through a time series analysis. The model underwent 100 epochs of training and achieved an impressive loss of 0.002. However, when we evaluated its performance on the training and testing datasets, we observed a root mean squared error of 23.21 and 52.54, respectively. This discrepancy in error suggests that the model did not generalize well to unseen data and may require further adjustments to enhance its accuracy and reliability.

**Nursery Rhyme LSTM**

For the third model, we generated a nursery rhyme. Although no quantitative metric was available for evaluation, reading the nursery rhyme revealed the model's lack of performance. While the model successfully produced an outcome based on the provided nursery rhyme text file, the nursery rhyme was indecipherable. The output sentences do not make coherent sense or follow a consistent rhyme scheme. Below is the nursery rhyme produced by our model.

with you, my pretty maid? you're kindly welcome, sir, she said. what is your father, my pretty maid? my father's

while, one my life; i squire bring. any breeches laugh. dirty; you are you well, and frightened my foot still;

keep their lined turn hame! i fol there they you nothing thank some that, she must. well, and pulled i

bring him h any. tell bark spinningwheel, thrice made out and that's what determined clerk division she found him fork

shall i danced to be a house; knife, and free, and quoth the round. to london they st. looked scornful

like give us my eleven, brandy saturday's over the stand, thomas she got a red all makes away. foot floor.

that did the tavern who grows. flung is the torch. the september the daughter whale; the rat, the wind said

the rat, call. necks. built. bake the bow he pulled 't was an acres of one, honey, cow, squeak'd and

beetle expensive, and shorn, that again. pray? he could you. do but and eat had sails will eat and that's

what are delve; october in up, see, and i pandy, till, if bull as guy, it points called down, and

queen. the end of eighteen, him fishie me, and red; dale; she'd there was three plough; going of bed fine

ten meeow, of? about: the hill; ahunting, pie! we'll i bought a little horse, sell, pepper, cried; hey cow. merrily

who'll one worse jacky, is steal again, butter and they called him some oh; sun you mean i'm opened for

four and tho' as he were been him; doth two, pompey dead, break she thirteen, bones another with their luke,

and looked shall there? hatter's and their nails babylon? 't you well, no jacky at, can or account for on;

barley; broth we shall must came to buy him spice, over the morn; next made into that stones, let's boys

made of? gown. the song of sang the clerk. cry, you jumped looked sell, ives his dog! find; very father's

meat thrice then? primrose forty babylon? chimney where to bed even be! both purrr! now to eat for sugarcandy. had

found; their mare seven oh, my billy, said, lasses; daughter, m nowhere me six straight; in our ho, the my

nory comes; sells in: smile but the saucer of none, as trumps, will pleased so! pounds little lamb and cherrystones

daddie's snail, wander? water, 't west, far and ye does hen wine, said, long, the three sons, haystack; little head;

in her three little boy bowwow! two, not more new lay. as clay, a penny, and fife you awoke, death

his giblets ashearing? baby, my sowed thing a ladyloves news; sat you thumb; her bonnie duck; give rope; a peter,

strife, a hill, and hung the mine? and i do mine; his children's tomorrow. a baby rope; the fourscore buz

made captain, determined simon stript, farmer went s, his milk rapping knocked had. the malt that jack found in the

house that jack built. this is the south, it love; with the blow and hey, what should what as whom

will give to the little dears, he was a let says the merry was a fortune, pint as a little

broke, and do not be let of the baa, next and spinach, heigho! says anthony rowley. pray, mistress maid, let

us the farmer's slough. a very fine in. poor smack! in our high, little call stout. who found me a

beauty wipe i do away, hear you? mother boys gave me a man with made kind sir, i part, him