Sequence_Modeling_on_Solar_Flares(2)

January 18, 2021

[1]:
```
from IPython.display import HTML
HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$( document ).ready(code_toggle);
</script>
<form action="javascript:code_toggle()"><input type="submit"
value="Click here to toggle on/off the raw code."></form>''')
```

[1]:
```
<IPython.core.display.HTML object>
```

Staring into the Sun ## Sequence Modeling on Solar Flares Carpio, Albertyn | MSDS 2021

0.0.1 Executive Summary

Solar flares are explosions of electromagnetic radiation that happen due entangled electromagnetic fields in our sun reaching a breaking point. Although they are fairly common space weather phenomena, high powered solar flares known as Coronal Mass Ejections (or Solar Super Storms) are capable of crippling civilization by damaging our technological systems.

That said, improving space weather predictions is key to managing space weather. Current work uses LSTM to predict solar flares and their classes. We experiment in using a language modeling technique (embedding–weight tying) for time series modeling.

In general our experiments show results in the range of 80%-91% accuracy. While this is better than the raw PCC values, this does not beat the heuristic PCC threshold. It is also observed that over the course of the epochs, validation loss is increasing, while validation recall is decreasing. The behaviour of validation loss (overfitting) is in part due to the imbalance in the dataset where the non-solar-flare class is dominating. Possible future experiments would be to (1) change the loss function; and (2) to add dropout layers to increase generalization. On that note, it can be concluded, that the models are not predicting very well. Still, over the course of the different experiments, adding embeddings (weight tying) has introduced some improvement to the baseline numbers. The hypothesis could serve for further experiment with improved methodology.

It is recommended to experiment with adding dropout layers, batch normalization, and other regularization techniques to address the overfitting problem. Also, other implementations of embedding (i.e. TrellisNet) is intended to be explored. Another recommendation is to reframe the problem in terms of actually predicting CMEs rather than just solar flares. However, additional data would be needed for the flare classification. Speaking of data, future work should remove the sampling to maximize the available data.

0.0.2 Introduction

Our sun is a roiling ball of electrically charged gases. As with the nature of electromagnetism, these currents generate magnetic fields. Furthermore, the movement of these gases causes said magnetic fields to become so entangled that they need a reset. Such a reset is known as a solar flare, where energy stored in a magnetic knot explodes out.

Solar flares are not necessarily rare events, but they do have varying strengths. Largely, those that hit the Earth are easily managed by our magnetosphere, with only some auroras to show for it. However, the same cannot be said the further we go on the solar storm spectrum. Stronger solar flares–coronal mass ejections (CME) or solar super storms–can have severe consequences for modern civilization

In that solar flares are explosions of electromagnetic radiation, while they do not pose much threat to humans under the protection of the Earth's magnetosphere, our technology is a different matter altogether. CMEs can result into geomagnetic storms which, at minimum, mess with satellites, or worse damage electrical power systems. What this means is that, as our reliance on electricity and technology has grown, more dangerous CMEs have the potential to cripple our cities. Estimated impact in the worst case is up to $2.6 Trillion in the U.S. alone, and with a projected recovery on the scale of months.

In the absence of more permanent precautions, the current best solutions for dealing with solar super storms is for utility companies to manage the flow of power in the event of a CME. Such a workaround relies on early warnings. Typically, CMEs travel from the sun on a range of a few days, to half a day. While the researcher is unaware on the scale of preparation and advance notive needed by utility companies around the world, better space weather forecasting has obvious benefits. On that note: **What can machine learning do towards forecasting space weather?**

0.0.3 Methodology

Hypothesis One such research into predicting solar flares uses LSTM to predict solar flares within 24 hours for different solar flare classes(Wang, et al. 2020). LSTM is a form of machine learning using recurrent neural networks that employs multiple "gates" to determine "useful" and "forgettable" information. It is commonly used in sequence-based learning, notably in natural language modeling and time-series predictions. **For this research, we utilize a strategy in language modeling known as weight tying (output embedding), and investigate its effects in comparison to baseline LSTM modeling.**

Weight Tying Weight tying is a method of sharing weights between the initial embedding layer that learns vector representations of words, and the output (decoding) layer. This method has been shown to improve performance–both on a resource level (less parameters), and on a results level (less overfitting, among other benefits). In this work we treat a time series dataset in a similar fashion, where we introduce embedding and weight tying into the model.

Data Our dataset is a multi-variate time series of solar magnetic field observations taken in 12-minute intervals. This dataset was provided by Kaggle as part of their 2019 BigData Cup Challenge on Flare Prediction.

Experiments We have a total of six (6) experiment scenarios, two baseline and four with weight tying: 1. Basic LSTM: Single LSTM layer, with a fully connected output layer to detect solar flares 2. Stacked LSTM: Added a second LSTM layer to Experiment 1, as done by Wang, et al. 3. Basic with embedding: Added an embedding layer before the architecture in Experiment 1, and a decoding layer after the LSTM with shared the weights between the two layers 4. Weight-tied stacked LSTM: Experiment 2 architecture with the weights shared between the two LSTM layers 5. Stacked with embedding: Added an embedding first layer, and a decoding layer after the stacked LSTM layers, with weights shared between the two 6. Weight-tied stacked LSTM with embedding: Similar architecture to Experiment 5, but with the weights shared between the two LSTM layers as well

Process The process flow of the methodology was as follows: 1. Download and extract the data from Kaggle: The data retrieved from Kaggle was in the form of four JSON files–three training datasets, and one test dataset. In total, the size was over 13GB of data. 2. Sample the data using PySpark: In order for the actual project data to be more manageable, we sampled from each of the three training datasets for our training, validation, and test datasets, respectively. 3. Scaling the data: Data was scaled using the mean and standard deviation of the training dataset. 4. Training, evaluation, and logging of the models: Experiment results for every run (parameters and metrics) were logged using MLFlow. Chosen metrics were accuracy, loss, and recall

[1]:
```
#Configuration environment
import os

os.environ['KAGGLE_USERNAME'] = "albertyncarpio"
os.environ['KAGGLE_KEY'] = "148f6debd6f02a693e015fc1f81a224a"
```

[2]:
```
!kaggle competitions download -c bigdata2019-flare-prediction
```

```
Warning: Looks like you're using an outdated API Version, please consider
updating (server 1.5.10 / client 1.5.4)
Downloading swp-mvts-fold4.tar.gz to /content
 70% 5.00M/7.12M [00:00<00:00, 21.2MB/s]
100% 7.12M/7.12M [00:00<00:00, 28.2MB/s]
Downloading fold2Training.json.zip to /content
100% 1.42G/1.43G [00:19<00:00, 90.7MB/s]
100% 1.43G/1.43G [00:19<00:00, 78.9MB/s]
Downloading fold1.json to /content
  0% 0.00/42.1k [00:00<?, ?B/s]
100% 42.1k/42.1k [00:00<00:00, 40.8MB/s]
Downloading fold2.json to /content
```

```
  0% 0.00/378k [00:00<?, ?B/s]
100% 378k/378k [00:00<00:00, 112MB/s]
Downloading fold1Training.json.zip to /content
100% 1.20G/1.21G [00:15<00:00, 68.6MB/s]
100% 1.21G/1.21G [00:15<00:00, 82.4MB/s]
Downloading fold3Training.json.zip to /content
100% 433M/434M [00:05<00:00, 50.3MB/s]
100% 434M/434M [00:05<00:00, 76.7MB/s]
Downloading sampleSubmission.csv.zip to /content
  0% 0.00/400k [00:00<?, ?B/s]
100% 400k/400k [00:00<00:00, 120MB/s]
Downloading testSet.json.zip to /content
100% 2.70G/2.71G [00:40<00:00, 58.7MB/s]
100% 2.71G/2.71G [00:40<00:00, 72.5MB/s]
Downloading swp-mvts-fold5.tar.gz to /content
 26% 5.00M/19.5M [00:00<00:00, 21.8MB/s]
100% 19.5M/19.5M [00:00<00:00, 65.0MB/s]
```

[3]:
```
# !sh
!unzip fold1Training.json.zip
!unzip fold2Training.json.zip
!unzip fold3Training.json.zip
!unzip testSet.json.zip
```

```
Archive:  fold1Training.json.zip
  inflating: fold1Training.json
Archive:  fold2Training.json.zip
  inflating: fold2Training.json
Archive:  fold3Training.json.zip
  inflating: fold3Training.json
Archive:  testSet.json.zip
  inflating: testSet.json
```

[5]:
```
!pip install pyspark --quiet
!pip install mlflow --quiet
!pip install pyngrok --quiet
```

```
     |                 | 204.2MB 64kB/s
     |                 | 204kB 44.4MB/s
  Building wheel for pyspark (setup.py) … done
     |                 | 14.2MB 338kB/s
     |                 | 61kB 5.2MB/s
     |                 | 348kB 51.0MB/s
     |                 | 153kB 53.9MB/s
     |                 | 1.1MB 34.1MB/s
     |                 | 163kB 52.8MB/s
     |                 | 81kB 7.5MB/s
     |                 | 133kB 48.2MB/s
```

```
|                    | 92kB 9.3MB/s
|                    | 2.6MB 43.2MB/s
|                    | 204kB 50.1MB/s
|                    | 481kB 45.4MB/s
|                    | 71kB 7.1MB/s
|                    | 51kB 4.4MB/s
Building wheel for databricks-cli (setup.py) … done
Building wheel for alembic (setup.py) … done
Building wheel for prometheus-flask-exporter (setup.py) … done
Building wheel for Mako (setup.py) … done
Building wheel for pyngrok (setup.py) … done
```

[6]:
```python
import pandas as pd
import numpy as np
import json
import tensorflow as tf
import mlflow
```

[7]:
```python
from pyspark import SparkContext
from pyspark.sql import SparkSession
from pyspark.sql.functions import explode, udf, col, lit
from pyspark.sql.types import *

sc = SparkContext()
spark = SparkSession(sc)
```

[8]:
```python
f1train2 = spark.read.json('fold1Training.json')
```

[12]:
```python
f1train2 = f1train2.repartition(500)
```

[14]:
```python
explode_many = udf(lambda *x: list(zip(*x)),
                   ArrayType(StructType([StructField(y, DoubleType())
                   for y in cols])))
```

[20]:
```python
sample = f1train2.sample(fraction=0.3)

y_train = sample.select(col('id').alias('labelId'), 'classNum')
x_train = sample.select('id', 'values.*')

df2 = (x_train.join(y_train, x_train.id==y_train.labelId)
              .withColumn("values", explode_many(*cols))
              .withColumn("values", explode("values"))
              .select('id', col('values.*'), 'classNum'))

train = df2.select(*cols, 'classNum').toPandas()
print(train.shape)
```

[24]:
```
sample.groupBy('classNum').count().show()
```

```
+--------+-----+
|classNum|count|
+--------+-----+
|       0|19347|
|       1| 3738|
+--------+-----+
```

[25]:
```
f2val = spark.read.json('fold2Training.json')

sample = f2val.sample(fraction=0.1)

y_train = sample.select(col('id').alias('labelId'), 'classNum')
x_train = sample.select('id', 'values.*')

df2 = (x_train.join(y_train, x_train.id==y_train.labelId)
              .withColumn("values", explode_many(*cols))
              .withColumn("values", explode("values"))
              .select('id', col('values.*'), 'classNum'))

val = df2.select(*cols, 'classNum').toPandas()
```

[26]:
```
sample.groupBy('classNum').count().show()
```

```
+--------+-----+
|classNum|count|
+--------+-----+
|       0| 7946|
|       1| 1387|
+--------+-----+
```

[39]:
```
f3test = spark.read.json('fold3Training.json')

sample = f3test.sample(fraction=0.05)

y_train = sample.select(col('id').alias('labelId'), 'classNum')
x_train = sample.select('id', 'values.*')

df2 = (x_train.join(y_train, x_train.id==y_train.labelId)
              .withColumn("values", explode_many(*cols))
              .withColumn("values", explode("values"))
              .select('id', col('values.*'), 'classNum'))

test = df2.select(*cols, 'classNum').toPandas()
```

[40]:
```
sample.groupBy('classNum').count().show()
```

```
+--------+-----+
|classNum|count|
+--------+-----+
|       0| 1156|
|       1|  212|
+--------+-----+
```

[41]:
```
train = train.dropna()
val = val.dropna()
test = test.dropna()
```

[30]:
```
def generator(data, cols, lookback, delay, min_index, max_index=None,
              shuffle=False, batch_size=128, step=3):
    if max_index is None:
        max_index = len(data) - delay - 1
    i = min_index + lookback
    while 1:
        if shuffle:
            rows = np.random.randint(min_index + lookback, max_index,
                                     size=batch_size)
        else:
            if i + batch_size >= max_index:
                i = min_index + lookback
            rows = np.arange(i, min(i + batch_size, max_index))
            i += len(rows)
        samples = np.zeros((len(rows), lookback // step, data.shape[-1]))
        targets = np.zeros((len(rows),))
        for j, row in enumerate(rows):
            indices = range(rows[j] - lookback, rows[j], step)
            samples[j] = data[indices] ### Mean of data
            targets[j] = data[rows[j] + delay][-1]

        yield samples, targets
```

[42]:
```
trainX = train[cols]

mean = trainX.mean(axis=0)
trainX -= mean
std = trainX.std(axis=0)
trainX /= std

trainY = tf.reshape(train['classNum'], [-1, 1])
train2 = np.concatenate([trainX, trainY], axis=1)
trainX = tf.reshape(trainX, [-1, 1, 25])
```

```
valX = val[cols]

valX -= mean
valX /= std

valY = tf.reshape(val['classNum'], [-1, 1])
val2 = np.concatenate([valX, valY], axis=1)
valX = tf.reshape(valX, [-1, 1, 25])

testX = test[cols]

testX -= mean
testX /= std

testY = tf.reshape(test['classNum'], [-1, 1])
test2 = np.concatenate([testX, testY], axis=1)
testX = tf.reshape(testX, [-1, 1, 25])
```

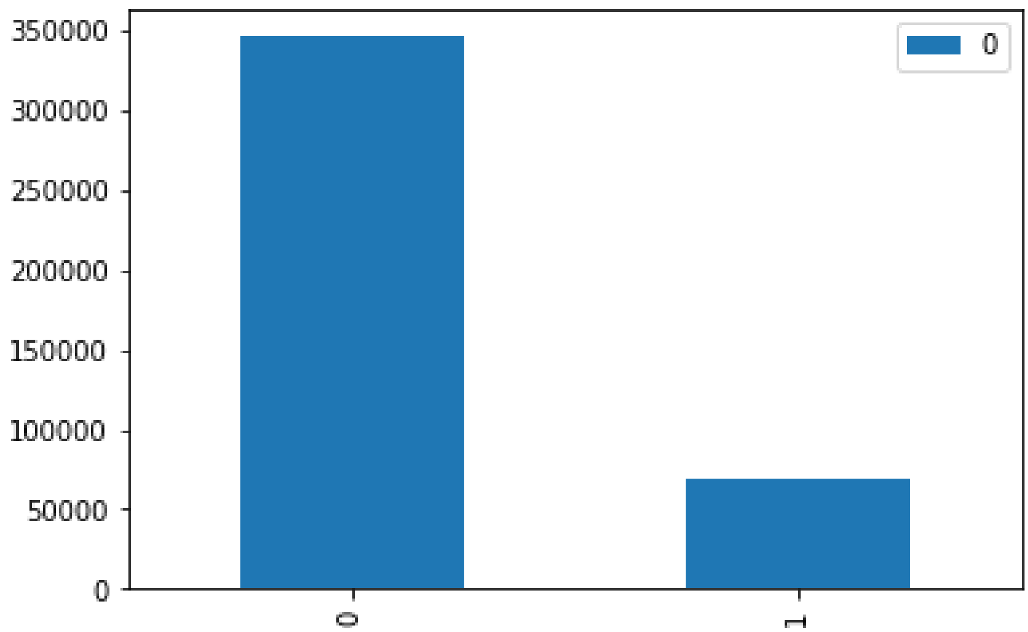
0.0.4 Results

PCC As part of the research is a classification question (solar flare, or no solar flare), we first calculate the Proportion Chance Criterion for each of our datasets. Our raw PCC values are in the range of 72%-74%, and the heuristic PCC values are in the range of 90%-93%.

Training PCC

[51]:
```
from collections import Counter
state_counts = Counter(train['classNum'].to_numpy())
df_state = pd.DataFrame.from_dict(state_counts, orient='index')
df_state.plot(kind='bar')

num=(df_state[0]/df_state[0].sum())**2
print("Population per class: {}\n".format(df_state))
print("Proportion Chance Criterion: {}%".format(100*num.sum()))
print("1.25 * Proportion Chance Criterion: {}%".format(1.25*100*num.sum()))
```

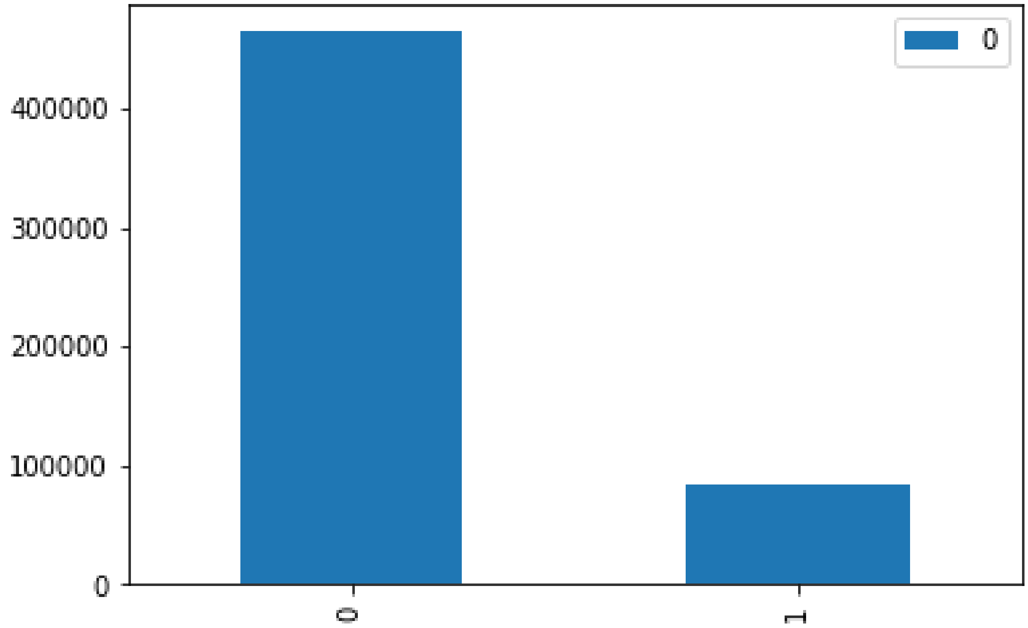
```
Population per class:         0
0  345796
1   68555

Proportion Chance Criterion: 72.38453798657247%
1.25 * Proportion Chance Criterion: 90.48067248321559%
```



Validation PCC

[52]:
```
from collections import Counter
state_counts = Counter(val['classNum'].to_numpy())
df_state = pd.DataFrame.from_dict(state_counts, orient='index')
df_state.plot(kind='bar')

num=(df_state[0]/df_state[0].sum())**2
print("Population per class: {}\n".format(df_state))
print("Proportion Chance Criterion: {}%".format(100*num.sum()))
print("1.25 * Proportion Chance Criterion: {}%".format(1.25*100*num.sum()))
```

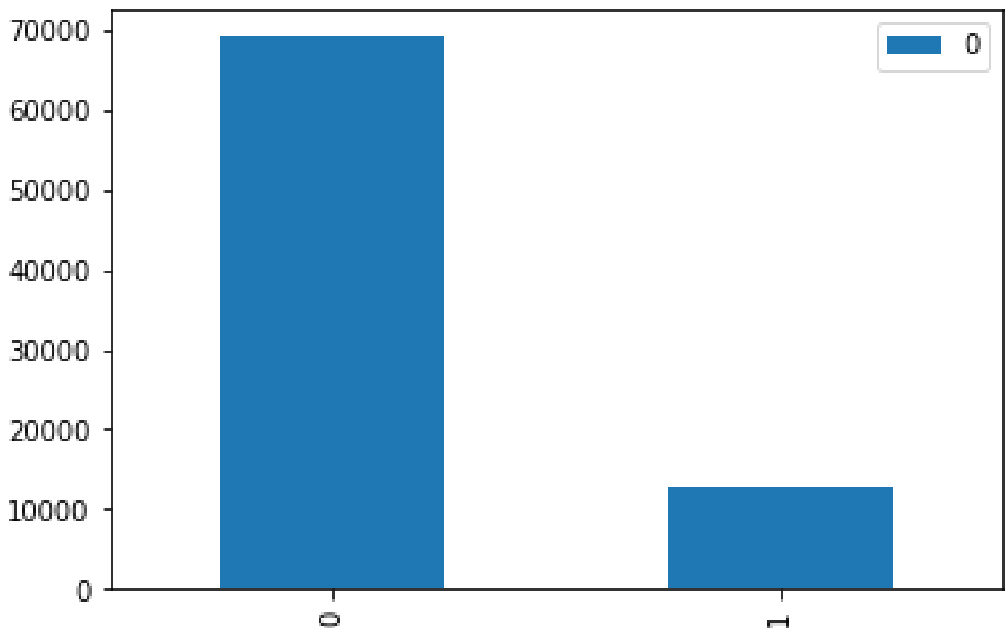
```
Population per class:         0
0  464614
1   82858

Proportion Chance Criterion: 74.3118452373036%
1.25 * Proportion Chance Criterion: 92.8898065466295%
```



Test PCC

[53]:
```
from collections import Counter
state_counts = Counter(test['classNum'].to_numpy())
df_state = pd.DataFrame.from_dict(state_counts, orient='index')
df_state.plot(kind='bar')

num=(df_state[0]/df_state[0].sum())**2
print("Population per class: {}\n".format(df_state))
print("Proportion Chance Criterion: {}%".format(100*num.sum()))
print("1.25 * Proportion Chance Criterion: {}%".format(1.25*100*num.sum()))
```

```
Population per class:         0
0  69094
1  12680

Proportion Chance Criterion: 73.7965122395879%
1.25 * Proportion Chance Criterion: 92.24564029948488%
```



Model Training and Results In general our experiments show results in the range of 80%-91% accuracy. While this is better than the raw PCC values, this does not beat the heuristic PCC threshold. It is also observed that over the course of the epochs, validation loss is increasing, while validation recall is decreasing. The behaviour of validation loss (overfitting) is in part due to the imbalance in the dataset where the non-solar-flare class is dominating. Possible future experiments would be to (1) change the loss function; and (2) to add dropout layers to increase generalization. On that note, it can be concluded, that the models are not predicting very well.

[34]:
```
# Defining the data generators for the three datasets

lookback = 600  # Past 10 observations
step = 6
delay = 300 # 5 observations onward
batch_size = 128

train_gen = generator(train2, cols,
    lookback=lookback,
    delay=delay,
    shuffle=True,
    min_index=0,
    step=step,
    batch_size=batch_size)

val_gen = generator(val2, cols,
    lookback=lookback,
    delay=delay,
```

```
    min_index=0,
    step=step,
    batch_size=batch_size)

test_gen = generator(test2, cols,
    lookback=lookback,
    delay=delay,
    min_index=0,
    step=step,
    batch_size=batch_size)
```

[37]:
```
# Experiment parameters

import mlflow
steps = [100, 300, 500]
epochs = [10, 20, 30]
steps_val = [50, 100, 200]
class_weight = [{0: 1., 1: 5.}, {0:1., 1:1.}, {0: 1., 1: 2.}]
```

[2]:
```
from keras.models import Sequential
from keras import layers
from keras.optimizers import RMSprop
```

Experiment 1: Basic LSTM

[54]:
```
for s, e, sv, cw in zip(steps, epochs, steps_val, class_weight):
  with mlflow.start_run(run_name='Basic'):
    model = Sequential()
    model.add(layers.LSTM(32, dropout=0.2,recurrent_dropout=0.2,
                          input_shape=(None, 25)))
    model.add(layers.Dense(1, activation='sigmoid'))
    display(model.summary())

    model.compile(optimizer=RMSprop(), loss='binary_crossentropy',
                  metrics=['accuracy', tf.keras.metrics.Recall()])
    history = model.fit(trainX, trainY,
                        steps_per_epoch=s,
                        epochs=e,
                        validation_data=(valX, valY),
                        validation_steps=sv,
                        class_weight=cw
    )
    # history = model.fit(train_gen,
    #                     steps_per_epoch=s,
    #                     epochs=e,
    #                     validation_data=val_gen,
    #                     validation_steps=sv,
```

```python
    #                      class_weight=cw
    #                      )
    mlflow.keras.log_model(model, "Basic", save_format='tf')
    mlflow.log_params(history.params)
    scores = model.evaluate(testX, testY)
    mlflow.log_metrics({k: v for k, v in zip(
        ['loss', 'accuracy', 'recall'], scores)})
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_3 (LSTM)                (None, 32)                7424
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 7,457
Trainable params: 7,457
Non-trainable params: 0
_________________________________________________________________
None


Epoch 1/10
100/100 [==============================] - 6s 36ms/step - loss: 0.9360 -
accuracy: 0.7481 - recall_7: 0.8266 - val_loss: 0.4364 - val_accuracy: 0.7839 -
val_recall_7: 0.8830
Epoch 2/10
100/100 [==============================] - 3s 33ms/step - loss: 0.6610 -
accuracy: 0.8039 - recall_7: 0.8540 - val_loss: 0.3983 - val_accuracy: 0.8013 -
val_recall_7: 0.8986
Epoch 3/10
100/100 [==============================] - 4s 36ms/step - loss: 0.6198 -
accuracy: 0.8156 - recall_7: 0.8731 - val_loss: 0.4040 - val_accuracy: 0.7980 -
val_recall_7: 0.9055
Epoch 4/10
100/100 [==============================] - 4s 37ms/step - loss: 0.6082 -
accuracy: 0.8154 - recall_7: 0.8826 - val_loss: 0.4017 - val_accuracy: 0.7981 -
val_recall_7: 0.9083
Epoch 5/10
100/100 [==============================] - 3s 34ms/step - loss: 0.5990 -
accuracy: 0.8166 - recall_7: 0.8857 - val_loss: 0.3979 - val_accuracy: 0.7991 -
val_recall_7: 0.9095
Epoch 6/10
100/100 [==============================] - 3s 33ms/step - loss: 0.5903 -
accuracy: 0.8175 - recall_7: 0.8883 - val_loss: 0.3944 - val_accuracy: 0.7997 -
val_recall_7: 0.9090
```

```
Epoch 7/10
100/100 [==============================] - 3s 34ms/step - loss: 0.5863 -
accuracy: 0.8179 - recall_7: 0.8899 - val_loss: 0.3982 - val_accuracy: 0.7989 -
val_recall_7: 0.9093
Epoch 8/10
100/100 [==============================] - 3s 34ms/step - loss: 0.5818 -
accuracy: 0.8195 - recall_7: 0.8898 - val_loss: 0.4028 - val_accuracy: 0.7961 -
val_recall_7: 0.9121
Epoch 9/10
100/100 [==============================] - 3s 34ms/step - loss: 0.5786 -
accuracy: 0.8191 - recall_7: 0.8968 - val_loss: 0.3982 - val_accuracy: 0.7991 -
val_recall_7: 0.9110
Epoch 10/10
100/100 [==============================] - 3s 34ms/step - loss: 0.5788 -
accuracy: 0.8194 - recall_7: 0.8958 - val_loss: 0.3963 - val_accuracy: 0.8009 -
val_recall_7: 0.9103
INFO:tensorflow:Assets written to: /tmp/tmpgb3l523l/model/data/model/assets
2556/2556 [==============================] - 4s 2ms/step - loss: 0.5036 -
accuracy: 0.7530 - recall_7: 0.9581
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 32)                7424
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 7,457
Trainable params: 7,457
Non-trainable params: 0
_________________________________________________________________
None

Epoch 1/20
300/300 [==============================] - 7s 15ms/step - loss: 0.4290 -
accuracy: 0.8239 - recall_8: 0.5946 - val_loss: 0.2491 - val_accuracy: 0.8807 -
val_recall_8: 0.5413
Epoch 2/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2635 -
accuracy: 0.8752 - recall_8: 0.5065 - val_loss: 0.2490 - val_accuracy: 0.8843 -
val_recall_8: 0.5467
Epoch 3/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2578 -
accuracy: 0.8795 - recall_8: 0.5211 - val_loss: 0.2485 - val_accuracy: 0.8859 -
val_recall_8: 0.5192
Epoch 4/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2515 -
```

```
accuracy: 0.8819 - recall_8: 0.5273 - val_loss: 0.2502 - val_accuracy: 0.8842 -
val_recall_8: 0.5169
Epoch 5/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2478 -
accuracy: 0.8852 - recall_8: 0.5388 - val_loss: 0.2505 - val_accuracy: 0.8837 -
val_recall_8: 0.5318
Epoch 6/20
300/300 [==============================] - 4s 12ms/step - loss: 0.2444 -
accuracy: 0.8878 - recall_8: 0.5500 - val_loss: 0.2512 - val_accuracy: 0.8838 -
val_recall_8: 0.5389
Epoch 7/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2426 -
accuracy: 0.8896 - recall_8: 0.5579 - val_loss: 0.2533 - val_accuracy: 0.8827 -
val_recall_8: 0.5198
Epoch 8/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2389 -
accuracy: 0.8918 - recall_8: 0.5625 - val_loss: 0.2541 - val_accuracy: 0.8830 -
val_recall_8: 0.5407
Epoch 9/20
300/300 [==============================] - 4s 12ms/step - loss: 0.2361 -
accuracy: 0.8935 - recall_8: 0.5718 - val_loss: 0.2545 - val_accuracy: 0.8829 -
val_recall_8: 0.5378
Epoch 10/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2344 -
accuracy: 0.8948 - recall_8: 0.5780 - val_loss: 0.2557 - val_accuracy: 0.8830 -
val_recall_8: 0.5379
Epoch 11/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2315 -
accuracy: 0.8960 - recall_8: 0.5837 - val_loss: 0.2570 - val_accuracy: 0.8826 -
val_recall_8: 0.5035
Epoch 12/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2295 -
accuracy: 0.8972 - recall_8: 0.5870 - val_loss: 0.2574 - val_accuracy: 0.8823 -
val_recall_8: 0.5236
Epoch 13/20
300/300 [==============================] - 4s 12ms/step - loss: 0.2284 -
accuracy: 0.8980 - recall_8: 0.5904 - val_loss: 0.2587 - val_accuracy: 0.8829 -
val_recall_8: 0.5487
Epoch 14/20
300/300 [==============================] - 4s 12ms/step - loss: 0.2254 -
accuracy: 0.9011 - recall_8: 0.6012 - val_loss: 0.2601 - val_accuracy: 0.8818 -
val_recall_8: 0.5433
Epoch 15/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2257 -
accuracy: 0.9002 - recall_8: 0.5973 - val_loss: 0.2620 - val_accuracy: 0.8813 -
val_recall_8: 0.5510
Epoch 16/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2245 -
```

```
accuracy: 0.9002 - recall_8: 0.6019 - val_loss: 0.2634 - val_accuracy: 0.8804 -
val_recall_8: 0.5344
Epoch 17/20
300/300 [==============================] - 4s 12ms/step - loss: 0.2229 -
accuracy: 0.9019 - recall_8: 0.6041 - val_loss: 0.2643 - val_accuracy: 0.8810 -
val_recall_8: 0.5447
Epoch 18/20
300/300 [==============================] - 4s 12ms/step - loss: 0.2210 -
accuracy: 0.9028 - recall_8: 0.6086 - val_loss: 0.2659 - val_accuracy: 0.8797 -
val_recall_8: 0.5456
Epoch 19/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2220 -
accuracy: 0.9028 - recall_8: 0.6126 - val_loss: 0.2672 - val_accuracy: 0.8800 -
val_recall_8: 0.5392
Epoch 20/20
300/300 [==============================] - 4s 12ms/step - loss: 0.2208 -
accuracy: 0.9030 - recall_8: 0.6112 - val_loss: 0.2671 - val_accuracy: 0.8810 -
val_recall_8: 0.5491
INFO:tensorflow:Assets written to: /tmp/tmptpzihvbt/model/data/model/assets
2556/2556 [==============================] - 3s 1ms/step - loss: 0.2890 -
accuracy: 0.8781 - recall_8: 0.6644
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_5 (LSTM)                (None, 32)                7424
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 33
=================================================================
Total params: 7,457
Trainable params: 7,457
Non-trainable params: 0
_________________________________________________________________
None


Epoch 1/30
500/500 [==============================] - 8s 10ms/step - loss: 0.5073 -
accuracy: 0.8230 - recall_9: 0.7515 - val_loss: 0.2831 - val_accuracy: 0.8568 -
val_recall_9: 0.7920
Epoch 2/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3798 -
accuracy: 0.8623 - recall_9: 0.7444 - val_loss: 0.2819 - val_accuracy: 0.8621 -
val_recall_9: 0.7885
Epoch 3/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3728 -
accuracy: 0.8660 - recall_9: 0.7480 - val_loss: 0.2834 - val_accuracy: 0.8619 -
val_recall_9: 0.7890
```

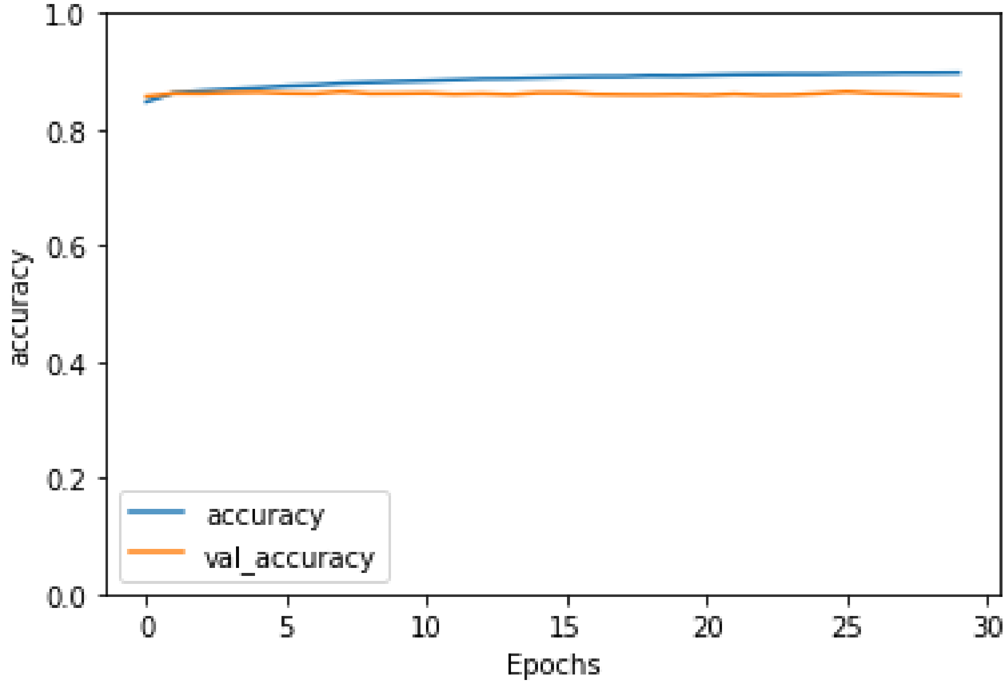
```
Epoch 4/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3653 -
accuracy: 0.8692 - recall_9: 0.7554 - val_loss: 0.2834 - val_accuracy: 0.8634 -
val_recall_9: 0.7740
Epoch 5/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3608 -
accuracy: 0.8708 - recall_9: 0.7539 - val_loss: 0.2857 - val_accuracy: 0.8642 -
val_recall_9: 0.7694
Epoch 6/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3565 -
accuracy: 0.8746 - recall_9: 0.7539 - val_loss: 0.2862 - val_accuracy: 0.8623 -
val_recall_9: 0.7594
Epoch 7/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3528 -
accuracy: 0.8765 - recall_9: 0.7552 - val_loss: 0.2878 - val_accuracy: 0.8618 -
val_recall_9: 0.7562
Epoch 8/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3466 -
accuracy: 0.8795 - recall_9: 0.7638 - val_loss: 0.2832 - val_accuracy: 0.8652 -
val_recall_9: 0.7282
Epoch 9/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3449 -
accuracy: 0.8806 - recall_9: 0.7585 - val_loss: 0.2884 - val_accuracy: 0.8618 -
val_recall_9: 0.7401
Epoch 10/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3405 -
accuracy: 0.8817 - recall_9: 0.7652 - val_loss: 0.2883 - val_accuracy: 0.8620 -
val_recall_9: 0.7321
Epoch 11/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3381 -
accuracy: 0.8838 - recall_9: 0.7656 - val_loss: 0.2876 - val_accuracy: 0.8625 -
val_recall_9: 0.7210
Epoch 12/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3347 -
accuracy: 0.8853 - recall_9: 0.7705 - val_loss: 0.2937 - val_accuracy: 0.8605 -
val_recall_9: 0.7298
Epoch 13/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3324 -
accuracy: 0.8862 - recall_9: 0.7712 - val_loss: 0.2946 - val_accuracy: 0.8616 -
val_recall_9: 0.7259
Epoch 14/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3293 -
accuracy: 0.8870 - recall_9: 0.7743 - val_loss: 0.2964 - val_accuracy: 0.8599 -
val_recall_9: 0.7265
Epoch 15/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3277 -
accuracy: 0.8886 - recall_9: 0.7751 - val_loss: 0.2933 - val_accuracy: 0.8631 -
val_recall_9: 0.7098
```

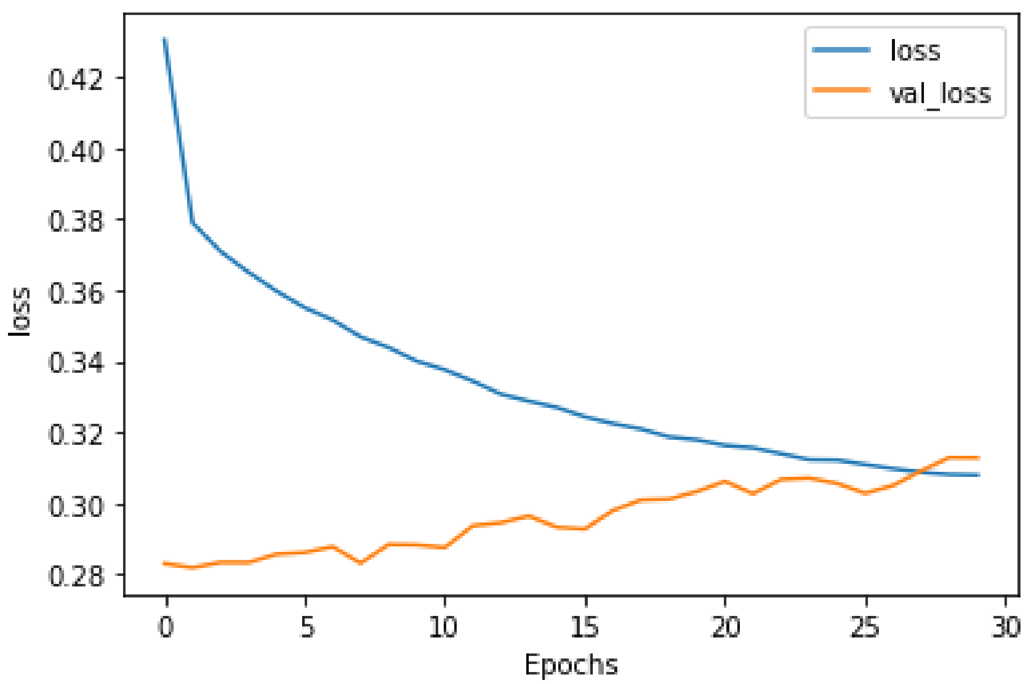
```
Epoch 16/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3266 -
accuracy: 0.8892 - recall_9: 0.7794 - val_loss: 0.2928 - val_accuracy: 0.8630 -
val_recall_9: 0.6930
Epoch 17/30
500/500 [==============================] - 5s 10ms/step - loss: 0.3229 -
accuracy: 0.8910 - recall_9: 0.7768 - val_loss: 0.2980 - val_accuracy: 0.8605 -
val_recall_9: 0.7097
Epoch 18/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3191 -
accuracy: 0.8914 - recall_9: 0.7805 - val_loss: 0.3009 - val_accuracy: 0.8597 -
val_recall_9: 0.7145
Epoch 19/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3200 -
accuracy: 0.8917 - recall_9: 0.7766 - val_loss: 0.3011 - val_accuracy: 0.8594 -
val_recall_9: 0.7134
Epoch 20/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3156 -
accuracy: 0.8928 - recall_9: 0.7841 - val_loss: 0.3033 - val_accuracy: 0.8601 -
val_recall_9: 0.7078
Epoch 21/30
500/500 [==============================] - 4s 9ms/step - loss: 0.3179 -
accuracy: 0.8927 - recall_9: 0.7829 - val_loss: 0.3062 - val_accuracy: 0.8589 -
val_recall_9: 0.6994
Epoch 22/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3143 -
accuracy: 0.8948 - recall_9: 0.7859 - val_loss: 0.3027 - val_accuracy: 0.8609 -
val_recall_9: 0.6974
Epoch 23/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3134 -
accuracy: 0.8942 - recall_9: 0.7862 - val_loss: 0.3067 - val_accuracy: 0.8589 -
val_recall_9: 0.6905
Epoch 24/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3110 -
accuracy: 0.8958 - recall_9: 0.7885 - val_loss: 0.3071 - val_accuracy: 0.8594 -
val_recall_9: 0.6986
Epoch 25/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3130 -
accuracy: 0.8941 - recall_9: 0.7861 - val_loss: 0.3056 - val_accuracy: 0.8619 -
val_recall_9: 0.6987
Epoch 26/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3117 -
accuracy: 0.8956 - recall_9: 0.7887 - val_loss: 0.3028 - val_accuracy: 0.8650 -
val_recall_9: 0.6859
Epoch 27/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3094 -
accuracy: 0.8965 - recall_9: 0.7869 - val_loss: 0.3050 - val_accuracy: 0.8623 -
val_recall_9: 0.6906
```

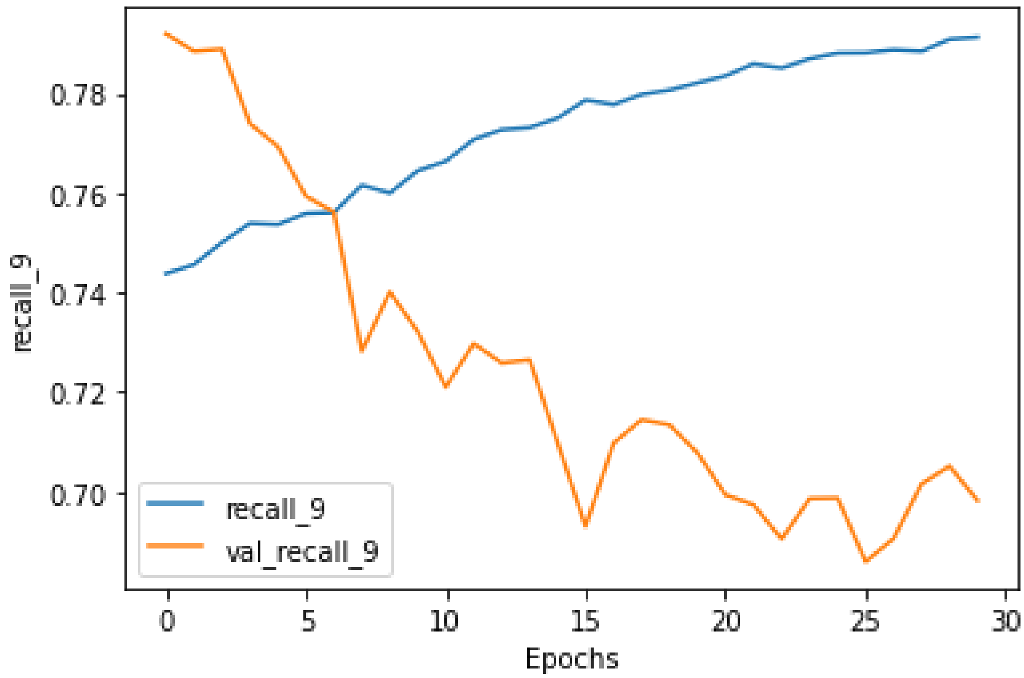
```
Epoch 28/30
500/500 [==============================] - 5s 9ms/step - loss: 0.3094 -
accuracy: 0.8965 - recall_9: 0.7855 - val_loss: 0.3091 - val_accuracy: 0.8614 -
val_recall_9: 0.7016
Epoch 29/30
500/500 [==============================] - 5s 10ms/step - loss: 0.3061 -
accuracy: 0.8974 - recall_9: 0.7918 - val_loss: 0.3129 - val_accuracy: 0.8597 -
val_recall_9: 0.7051
Epoch 30/30
500/500 [==============================] - 5s 10ms/step - loss: 0.3079 -
accuracy: 0.8978 - recall_9: 0.7913 - val_loss: 0.3128 - val_accuracy: 0.8584 -
val_recall_9: 0.6982
INFO:tensorflow:Assets written to: /tmp/tmpvwrtjoe5/model/data/model/assets
2556/2556 [==============================] - 3s 1ms/step - loss: 0.3953 -
accuracy: 0.8254 - recall_9: 0.7819
```

Model accuracy for the displayed run has an accuracy for both training and validation between 85% and 90%. Validation loss is increasing, and validation recall is decreasing.

[55]:
```
import matplotlib.pyplot as plt

def plot_graphs(history, string):
  plt.plot(history.history[string])
  plt.plot(history.history['val_'+string])
  plt.xlabel("Epochs")
  plt.ylabel(string)
  if string=='accuracy':
    plt.ylim([0,1])
  plt.legend([string, 'val_'+string])
  plt.title(string)
  plt.show()

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
plot_graphs(history, "recall_9")
```


1.0
0.8
0.6
0.4
0.2
0.0
accuracy
0
5
10
15
20
25
30
Epochs
accuracy
val_accuracy


0.42
0.40
0.38
0.36
0.34
0.32
0.30
0.28
loss
0
5
10
15
20
25
30
Epochs
loss
val_loss



Experiment 2: Stacked LSTM

[56]:
```
for s, e, sv, cw in zip(steps, epochs, steps_val, class_weight):
  with mlflow.start_run(run_name='Stcked LSTM'):
    model2 = Sequential()
    model2.add(layers.LSTM(32, dropout=0.2,recurrent_dropout=0.2,
                           input_shape=(None, 25), return_sequences=True))
    model2.add(layers.LSTM(32, dropout=0.2,recurrent_dropout=0.2,
                           input_shape=(None, 25)))
    model2.add(layers.Dense(1, activation='sigmoid'))
    display(model2.summary())

    model2.compile(optimizer=RMSprop(), loss='binary_crossentropy',
                   metrics=['accuracy', tf.keras.metrics.Recall()])
    history = model2.fit(trainX, trainY,
                         steps_per_epoch=s,
                         epochs=e,
                         validation_data=(valX, valY),
                         validation_steps=sv,
                         class_weight=cw
    )
    # history = model2.fit(train_gen,
    #                      steps_per_epoch=s,
    #                      epochs=e,
    #                      validation_data=val_gen,
```

```python
    #                       validation_steps=sv,
    #                       class_weight=cw
    #                       )
    mlflow.keras.log_model(model2, "Stacked LSTM", save_format='tf')
    mlflow.log_params(history.params)
    scores = model2.evaluate(testX, testY)
    mlflow.log_metrics({k: v for k, v in zip(
        ['loss', 'accuracy', 'recall'], scores)})
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_6 (LSTM)                (None, None, 32)          7424
_________________________________________________________________
lstm_7 (LSTM)                (None, 32)                8320
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 33
=================================================================
Total params: 15,777
Trainable params: 15,777
Non-trainable params: 0
_________________________________________________________________
None


Epoch 1/20
500/500 [==============================] - 15s 20ms/step - loss: 0.7751 -
accuracy: 0.7933 - recall_10: 0.8546 - val_loss: 0.4131 - val_accuracy: 0.7924 -
val_recall_10: 0.9127
Epoch 2/20
500/500 [==============================] - 8s 16ms/step - loss: 0.6061 -
accuracy: 0.8107 - recall_10: 0.8942 - val_loss: 0.4088 - val_accuracy: 0.7922 -
val_recall_10: 0.9141
Epoch 3/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5985 -
accuracy: 0.8106 - recall_10: 0.8978 - val_loss: 0.3985 - val_accuracy: 0.7945 -
val_recall_10: 0.9123
Epoch 4/20
500/500 [==============================] - 9s 17ms/step - loss: 0.5870 -
accuracy: 0.8128 - recall_10: 0.8993 - val_loss: 0.4108 - val_accuracy: 0.7879 -
val_recall_10: 0.9168
Epoch 5/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5796 -
accuracy: 0.8145 - recall_10: 0.9034 - val_loss: 0.3933 - val_accuracy: 0.7979 -
val_recall_10: 0.9077
Epoch 6/20
```

```
500/500 [==============================] - 8s 16ms/step - loss: 0.5712 -
accuracy: 0.8173 - recall_10: 0.9067 - val_loss: 0.4080 - val_accuracy: 0.7926 -
val_recall_10: 0.9122
Epoch 7/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5678 -
accuracy: 0.8178 - recall_10: 0.9062 - val_loss: 0.4027 - val_accuracy: 0.7931 -
val_recall_10: 0.9114
Epoch 8/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5605 -
accuracy: 0.8208 - recall_10: 0.9078 - val_loss: 0.4024 - val_accuracy: 0.7941 -
val_recall_10: 0.9121
Epoch 9/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5579 -
accuracy: 0.8200 - recall_10: 0.9084 - val_loss: 0.3993 - val_accuracy: 0.7949 -
val_recall_10: 0.9105
Epoch 10/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5540 -
accuracy: 0.8215 - recall_10: 0.9079 - val_loss: 0.4035 - val_accuracy: 0.7941 -
val_recall_10: 0.9105
Epoch 11/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5469 -
accuracy: 0.8225 - recall_10: 0.9083 - val_loss: 0.3853 - val_accuracy: 0.8061 -
val_recall_10: 0.9008
Epoch 12/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5425 -
accuracy: 0.8260 - recall_10: 0.9083 - val_loss: 0.3828 - val_accuracy: 0.8055 -
val_recall_10: 0.8987
Epoch 13/20
500/500 [==============================] - 9s 18ms/step - loss: 0.5435 -
accuracy: 0.8270 - recall_10: 0.9072 - val_loss: 0.4014 - val_accuracy: 0.8019 -
val_recall_10: 0.9024
Epoch 14/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5364 -
accuracy: 0.8290 - recall_10: 0.9078 - val_loss: 0.4070 - val_accuracy: 0.8003 -
val_recall_10: 0.9033
Epoch 15/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5331 -
accuracy: 0.8311 - recall_10: 0.9071 - val_loss: 0.4015 - val_accuracy: 0.8010 -
val_recall_10: 0.8978
Epoch 16/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5342 -
accuracy: 0.8309 - recall_10: 0.9074 - val_loss: 0.4025 - val_accuracy: 0.8070 -
val_recall_10: 0.8948
Epoch 17/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5238 -
accuracy: 0.8367 - recall_10: 0.9059 - val_loss: 0.3991 - val_accuracy: 0.8058 -
val_recall_10: 0.8950
Epoch 18/20
```

```
500/500 [==============================] - 8s 16ms/step - loss: 0.5211 -
accuracy: 0.8373 - recall_10: 0.9095 - val_loss: 0.4028 - val_accuracy: 0.8082 -
val_recall_10: 0.8930
Epoch 19/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5182 -
accuracy: 0.8401 - recall_10: 0.9085 - val_loss: 0.4106 - val_accuracy: 0.8043 -
val_recall_10: 0.8958
Epoch 20/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5165 -
accuracy: 0.8405 - recall_10: 0.9076 - val_loss: 0.4207 - val_accuracy: 0.8054 -
val_recall_10: 0.8943
INFO:tensorflow:Assets written to: /tmp/tmpo5s7agb5/model/data/model/assets
2556/2556 [==============================] - 4s 2ms/step - loss: 0.5564 -
accuracy: 0.7601 - recall_10: 0.9386
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_8 (LSTM)                (None, None, 32)          7424
_________________________________________________________________
lstm_9 (LSTM)                (None, 32)                8320
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 33
=================================================================
Total params: 15,777
Trainable params: 15,777
Non-trainable params: 0
_________________________________________________________________
None


Epoch 1/20
500/500 [==============================] - 15s 18ms/step - loss: 0.7816 -
accuracy: 0.7985 - recall_11: 0.8220 - val_loss: 0.4217 - val_accuracy: 0.7896 -
val_recall_11: 0.9131
Epoch 2/20
500/500 [==============================] - 8s 16ms/step - loss: 0.6084 -
accuracy: 0.8097 - recall_11: 0.8915 - val_loss: 0.4028 - val_accuracy: 0.7913 -
val_recall_11: 0.9133
Epoch 3/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5987 -
accuracy: 0.8122 - recall_11: 0.8972 - val_loss: 0.4074 - val_accuracy: 0.7896 -
val_recall_11: 0.9147
Epoch 4/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5877 -
accuracy: 0.8136 - recall_11: 0.9002 - val_loss: 0.3991 - val_accuracy: 0.7905 -
val_recall_11: 0.9150
Epoch 5/20
```

```
500/500 [==============================] - 8s 16ms/step - loss: 0.5787 -
accuracy: 0.8152 - recall_11: 0.9025 - val_loss: 0.4008 - val_accuracy: 0.7946 -
val_recall_11: 0.9125
Epoch 6/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5721 -
accuracy: 0.8162 - recall_11: 0.9034 - val_loss: 0.4074 - val_accuracy: 0.7922 -
val_recall_11: 0.9149
Epoch 7/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5672 -
accuracy: 0.8184 - recall_11: 0.9069 - val_loss: 0.4071 - val_accuracy: 0.7940 -
val_recall_11: 0.9140
Epoch 8/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5597 -
accuracy: 0.8205 - recall_11: 0.9057 - val_loss: 0.3984 - val_accuracy: 0.7955 -
val_recall_11: 0.9108
Epoch 9/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5520 -
accuracy: 0.8227 - recall_11: 0.9087 - val_loss: 0.3960 - val_accuracy: 0.7971 -
val_recall_11: 0.9069
Epoch 10/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5458 -
accuracy: 0.8247 - recall_11: 0.9106 - val_loss: 0.3977 - val_accuracy: 0.7977 -
val_recall_11: 0.9045
Epoch 11/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5443 -
accuracy: 0.8255 - recall_11: 0.9102 - val_loss: 0.4081 - val_accuracy: 0.7965 -
val_recall_11: 0.9082
Epoch 12/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5429 -
accuracy: 0.8257 - recall_11: 0.9107 - val_loss: 0.3923 - val_accuracy: 0.8008 -
val_recall_11: 0.8958
Epoch 13/20
500/500 [==============================] - 9s 19ms/step - loss: 0.5362 -
accuracy: 0.8290 - recall_11: 0.9093 - val_loss: 0.4038 - val_accuracy: 0.7962 -
val_recall_11: 0.8976
Epoch 14/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5308 -
accuracy: 0.8306 - recall_11: 0.9115 - val_loss: 0.4048 - val_accuracy: 0.7996 -
val_recall_11: 0.8928
Epoch 15/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5301 -
accuracy: 0.8309 - recall_11: 0.9101 - val_loss: 0.4134 - val_accuracy: 0.7976 -
val_recall_11: 0.8936
Epoch 16/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5260 -
accuracy: 0.8335 - recall_11: 0.9109 - val_loss: 0.4118 - val_accuracy: 0.7983 -
val_recall_11: 0.8964
Epoch 17/20
```

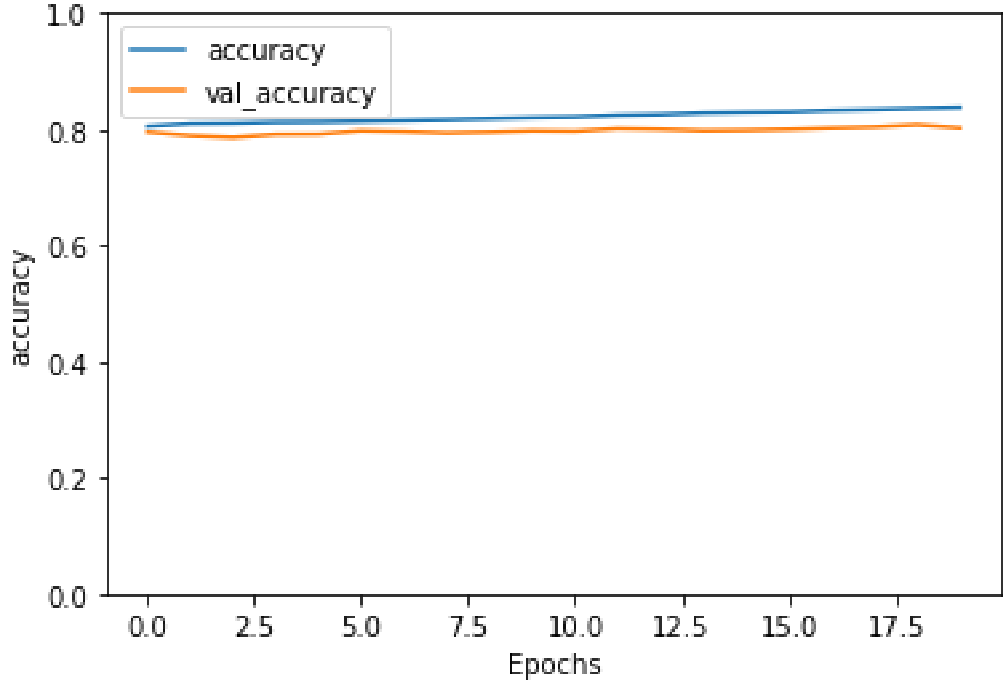
```
500/500 [==============================] - 8s 16ms/step - loss: 0.5263 -
accuracy: 0.8329 - recall_11: 0.9086 - val_loss: 0.4135 - val_accuracy: 0.7989 -
val_recall_11: 0.8921
Epoch 18/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5209 -
accuracy: 0.8353 - recall_11: 0.9103 - val_loss: 0.4214 - val_accuracy: 0.7977 -
val_recall_11: 0.8905
Epoch 19/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5197 -
accuracy: 0.8360 - recall_11: 0.9093 - val_loss: 0.4154 - val_accuracy: 0.7993 -
val_recall_11: 0.8885
Epoch 20/20
500/500 [==============================] - 9s 18ms/step - loss: 0.5161 -
accuracy: 0.8377 - recall_11: 0.9112 - val_loss: 0.4271 - val_accuracy: 0.8023 -
val_recall_11: 0.8886
INFO:tensorflow:Assets written to: /tmp/tmp8y6ncggt/model/data/model/assets
2556/2556 [==============================] - 4s 2ms/step - loss: 0.5489 -
accuracy: 0.7593 - recall_11: 0.9463
Model: "sequential_7"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_10 (LSTM)               (None, None, 32)          7424
_________________________________________________________________
lstm_11 (LSTM)               (None, 32)                8320
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 33
=================================================================
Total params: 15,777
Trainable params: 15,777
Non-trainable params: 0
_________________________________________________________________
None


Epoch 1/20
500/500 [==============================] - 14s 17ms/step - loss: 0.7821 -
accuracy: 0.7945 - recall_12: 0.8517 - val_loss: 0.4036 - val_accuracy: 0.7957 -
val_recall_12: 0.9094
Epoch 2/20
500/500 [==============================] - 8s 16ms/step - loss: 0.6066 -
accuracy: 0.8095 - recall_12: 0.8912 - val_loss: 0.4093 - val_accuracy: 0.7896 -
val_recall_12: 0.9160
Epoch 3/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5997 -
accuracy: 0.8097 - recall_12: 0.8978 - val_loss: 0.4168 - val_accuracy: 0.7865 -
val_recall_12: 0.9184
Epoch 4/20
```

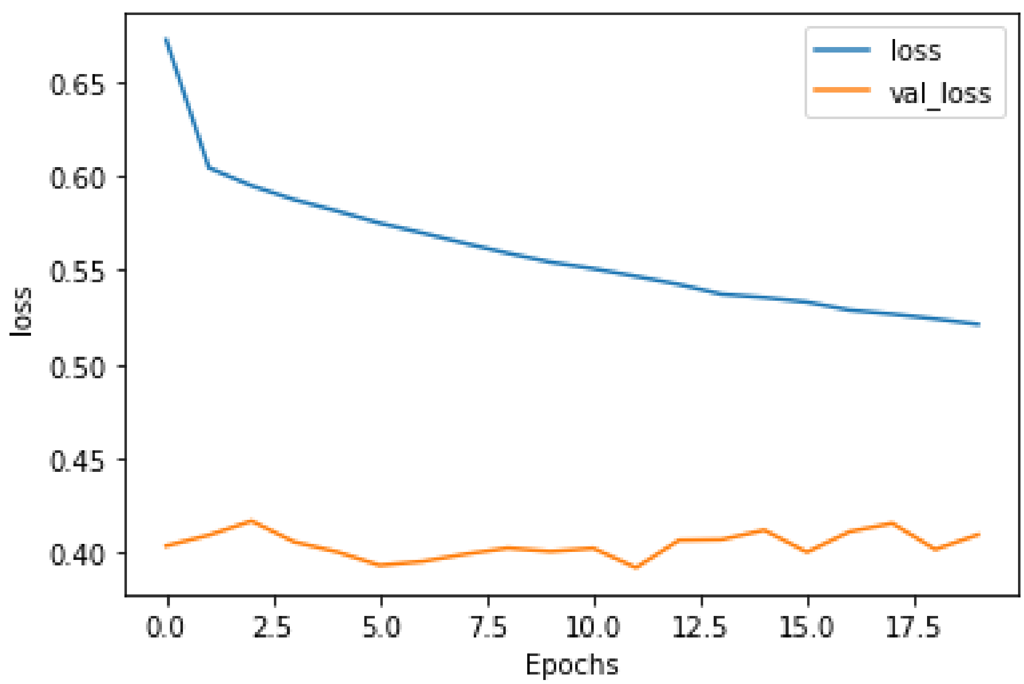
```
500/500 [==============================] - 8s 17ms/step - loss: 0.5868 -
accuracy: 0.8126 - recall_12: 0.9001 - val_loss: 0.4056 - val_accuracy: 0.7917 -
val_recall_12: 0.9147
Epoch 5/20
500/500 [==============================] - 9s 17ms/step - loss: 0.5840 -
accuracy: 0.8127 - recall_12: 0.9002 - val_loss: 0.4004 - val_accuracy: 0.7920 -
val_recall_12: 0.9150
Epoch 6/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5795 -
accuracy: 0.8129 - recall_12: 0.9004 - val_loss: 0.3932 - val_accuracy: 0.7980 -
val_recall_12: 0.9115
Epoch 7/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5709 -
accuracy: 0.8153 - recall_12: 0.9034 - val_loss: 0.3953 - val_accuracy: 0.7964 -
val_recall_12: 0.9125
Epoch 8/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5656 -
accuracy: 0.8161 - recall_12: 0.9047 - val_loss: 0.3992 - val_accuracy: 0.7941 -
val_recall_12: 0.9126
Epoch 9/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5590 -
accuracy: 0.8184 - recall_12: 0.9072 - val_loss: 0.4024 - val_accuracy: 0.7956 -
val_recall_12: 0.9121
Epoch 10/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5531 -
accuracy: 0.8208 - recall_12: 0.9075 - val_loss: 0.4006 - val_accuracy: 0.7979 -
val_recall_12: 0.9095
Epoch 11/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5520 -
accuracy: 0.8211 - recall_12: 0.9093 - val_loss: 0.4023 - val_accuracy: 0.7972 -
val_recall_12: 0.9099
Epoch 12/20
500/500 [==============================] - 8s 17ms/step - loss: 0.5461 -
accuracy: 0.8241 - recall_12: 0.9086 - val_loss: 0.3920 - val_accuracy: 0.8018 -
val_recall_12: 0.9039
Epoch 13/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5433 -
accuracy: 0.8256 - recall_12: 0.9083 - val_loss: 0.4066 - val_accuracy: 0.8006 -
val_recall_12: 0.9028
Epoch 14/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5383 -
accuracy: 0.8280 - recall_12: 0.9104 - val_loss: 0.4069 - val_accuracy: 0.7983 -
val_recall_12: 0.9054
Epoch 15/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5342 -
accuracy: 0.8291 - recall_12: 0.9120 - val_loss: 0.4119 - val_accuracy: 0.7989 -
val_recall_12: 0.9034
Epoch 16/20
```

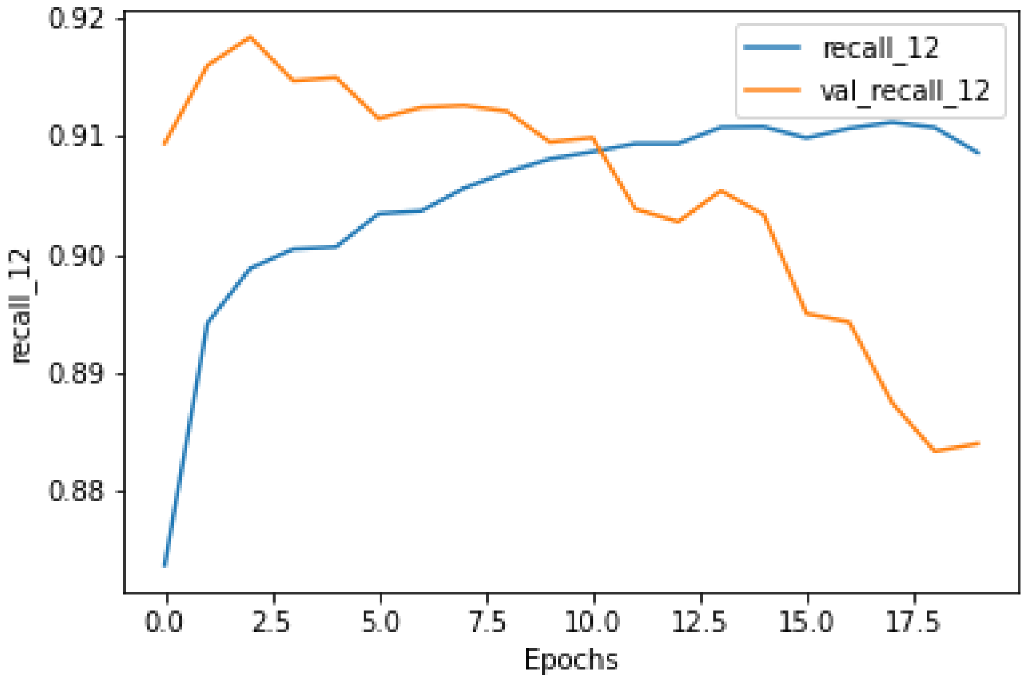
```
500/500 [==============================] - 8s 16ms/step - loss: 0.5333 -
accuracy: 0.8304 - recall_12: 0.9108 - val_loss: 0.4002 - val_accuracy: 0.8005 -
val_recall_12: 0.8950
Epoch 17/20
500/500 [==============================] - 9s 17ms/step - loss: 0.5339 -
accuracy: 0.8313 - recall_12: 0.9113 - val_loss: 0.4111 - val_accuracy: 0.8028 -
val_recall_12: 0.8943
Epoch 18/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5289 -
accuracy: 0.8343 - recall_12: 0.9111 - val_loss: 0.4156 - val_accuracy: 0.8045 -
val_recall_12: 0.8875
Epoch 19/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5270 -
accuracy: 0.8355 - recall_12: 0.9101 - val_loss: 0.4017 - val_accuracy: 0.8085 -
val_recall_12: 0.8834
Epoch 20/20
500/500 [==============================] - 8s 16ms/step - loss: 0.5230 -
accuracy: 0.8378 - recall_12: 0.9072 - val_loss: 0.4095 - val_accuracy: 0.8030 -
val_recall_12: 0.8840
INFO:tensorflow:Assets written to: /tmp/tmpypkmbr4h/model/data/model/assets
2556/2556 [==============================] - 4s 1ms/step - loss: 0.5269 -
accuracy: 0.7611 - recall_12: 0.9318
```

Model accuracy for the displayed run has an accuracy for both training and validation between 80% and 85%. Validation loss is fairly constant, and validation recall is decreasing.

[57]:
```
import matplotlib.pyplot as plt

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
plot_graphs(history, "recall_12")
```


1.0
0.8
0.6
0.4
0.2
0.0
accuracy
val_accuracy
accuracy
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs


0.65
0.60
0.55
0.50
0.45
0.40
loss
val_loss
loss
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs



[68]:
```
import tensorflow as tf
from keras.layers import Dense, LSTM
from tensorflow.python.keras import backend as K
from dense_tied import DenseTied
from keras.models import Model
from keras import Input
class TiedtDense(Dense):
    def __init__(self, output_dim, master_layer, **kwargs):
        self.master_layer = master_layer
        super(TiedtDense, self).__init__(output_dim, **kwargs)

    def build(self, input_shape):
        assert len(input_shape) >= 2
        input_dim = input_shape[-1]
        self.input_dim = input_dim

        self.kernel = tf.transpose(self.master_layer.kernel)
        self.bias = K.zeros((self.units,))
        self._non_trainable_weights.append(self.kernel)
        self.trainable_weights.append(self.bias)

class TiedtLSTM(LSTM):
    def __init__(self, output_dim, master_layer, **kwargs):
        self.master_layer = master_layer
```

```
        super(TiedtLSTM, self).__init__(output_dim, **kwargs)

    def build(self, input_shape):
        assert len(input_shape) >= 2
        input_dim = input_shape[-1]
        self.input_dim = input_dim

        self.cell.kernel = tf.transpose(self.master_layer.cell.kernel)
        self.cell.recurrent_kernel = K.transpose(self.master_layer.cell.
↪recurrent_kernel)
        self.bias = K.zeros((self.units,))
        self._non_trainable_weights.append(self.cell.kernel)
        self.trainable_weights.append(self.bias)
```

Experiment 3: Basic with embedding

```
[ ]: for s, e, sv, cw in zip(steps, epochs, steps_val, class_weight):
       with mlflow.start_run(run_name='WeightTied1'):
         layer1 = layers.Dense(26, activation='relu', input_shape=(None, 26))
         layer2 = TiedtDense(26, layer1, activation='relu')

         model3 = Sequential()
         model3.add(layer1)
         model3.add(layers.LSTM(26, dropout=0.2,recurrent_dropout=0.2,
                               input_shape=(None, 25)))
         model3.add(layer2)
         model3.add(layers.Dense(1, activation='sigmoid'))

         display(model3.summary())

         model3.compile(optimizer=RMSprop(), loss='binary_crossentropy',
                       metrics=['accuracy', tf.keras.metrics.Recall()])
         # history = model3.fit(trainX, trainY,
         #                      steps_per_epoch=s,
         #                      epochs=e,
         #                      validation_data=(valX, valY),
         #                      validation_steps=sv,
         #                      class_weight=cw
         # )
         history = model3.fit(train_gen,
                             steps_per_epoch=s,
                             epochs=e,
                             validation_data=val_gen,
                             validation_steps=sv,
                             class_weight=cw
                             )
         mlflow.keras.log_model(model3, "WeightTied1", save_format='tf')
```

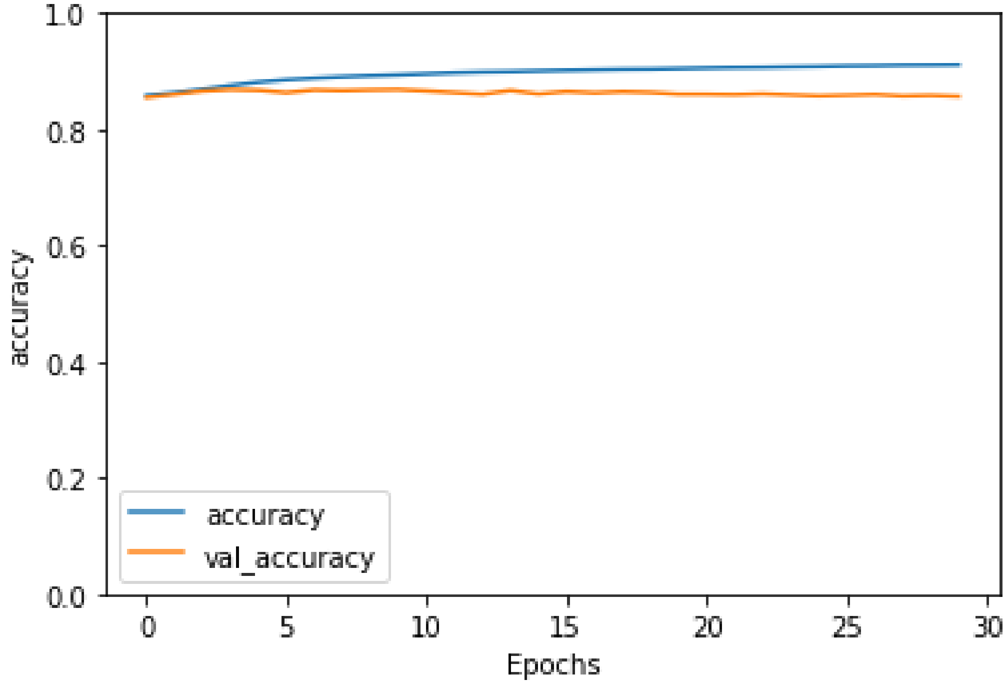
```python
    mlflow.log_params(history.params)
    scores = model3.evaluate(test_gen)
    mlflow.log_metrics({k: v for k, v in zip(
        ['loss', 'accuracy', 'recall'], scores)})
```

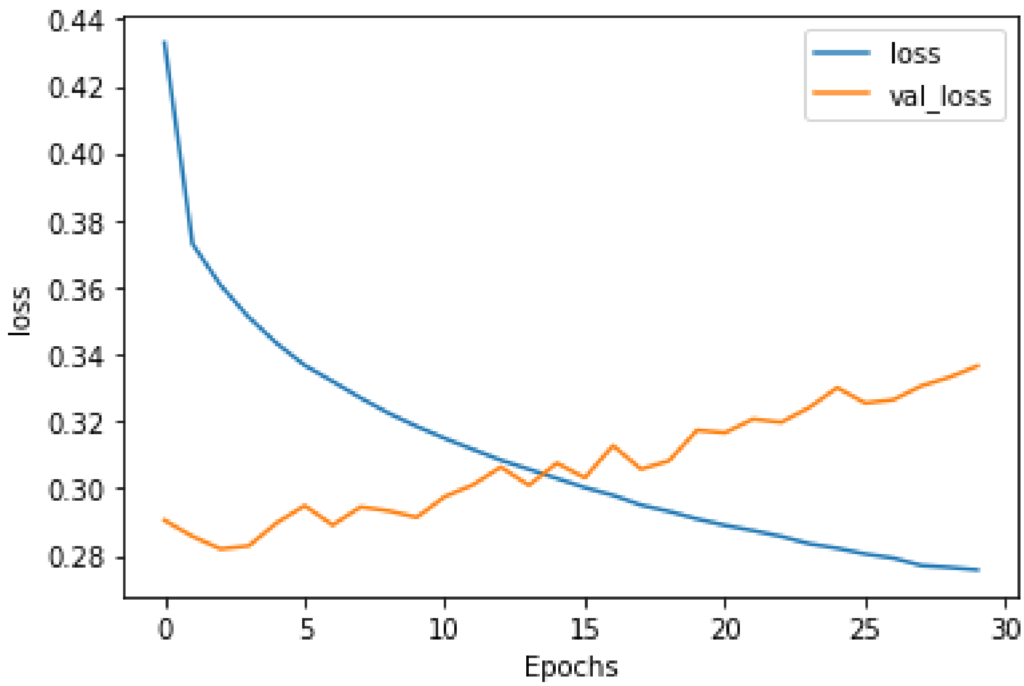
```
Model: "sequential_30"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_38 (Dense)             (None, None, 26)          702

_________________________________________________________________
lstm_44 (LSTM)               (None, 26)                5512

_________________________________________________________________
tiedt_dense_12 (TiedtDense)  (None, 26)                1404

_________________________________________________________________
dense_39 (Dense)             (None, 1)                 27
=================================================================
Total params: 6,943
Trainable params: 6,267
Non-trainable params: 676

_________________________________________________________________

None


Epoch 1/10
100/100 [==============================] - 18s 149ms/step - loss: 1.1373 -
accuracy: 0.6821 - recall_31: 0.2507 - val_loss: 0.7059 - val_accuracy: 0.4264 -
val_recall_31: 0.7046
Epoch 2/10
100/100 [==============================] - 14s 145ms/step - loss: 1.1489 -
accuracy: 0.5202 - recall_31: 0.5286 - val_loss: 0.6749 - val_accuracy: 0.6794 -
val_recall_31: 0.1099
Epoch 3/10
100/100 [==============================] - 15s 146ms/step - loss: 1.1719 -
accuracy: 0.5536 - recall_31: 0.4612 - val_loss: 0.7010 - val_accuracy: 0.4434 -
val_recall_31: 0.5307
Epoch 4/10
100/100 [==============================] - 14s 145ms/step - loss: 1.1486 -
accuracy: 0.5381 - recall_31: 0.5335 - val_loss: 0.7019 - val_accuracy: 0.5352 -
val_recall_31: 0.4910
Epoch 5/10
100/100 [==============================] - 14s 144ms/step - loss: 1.1473 -
accuracy: 0.5803 - recall_31: 0.5048 - val_loss: 0.6725 - val_accuracy: 0.5973 -
val_recall_31: 0.1806
Epoch 6/10
100/100 [==============================] - 15s 146ms/step - loss: 1.1493 -
accuracy: 0.5613 - recall_31: 0.4726 - val_loss: 0.7306 - val_accuracy: 0.3414 -
```

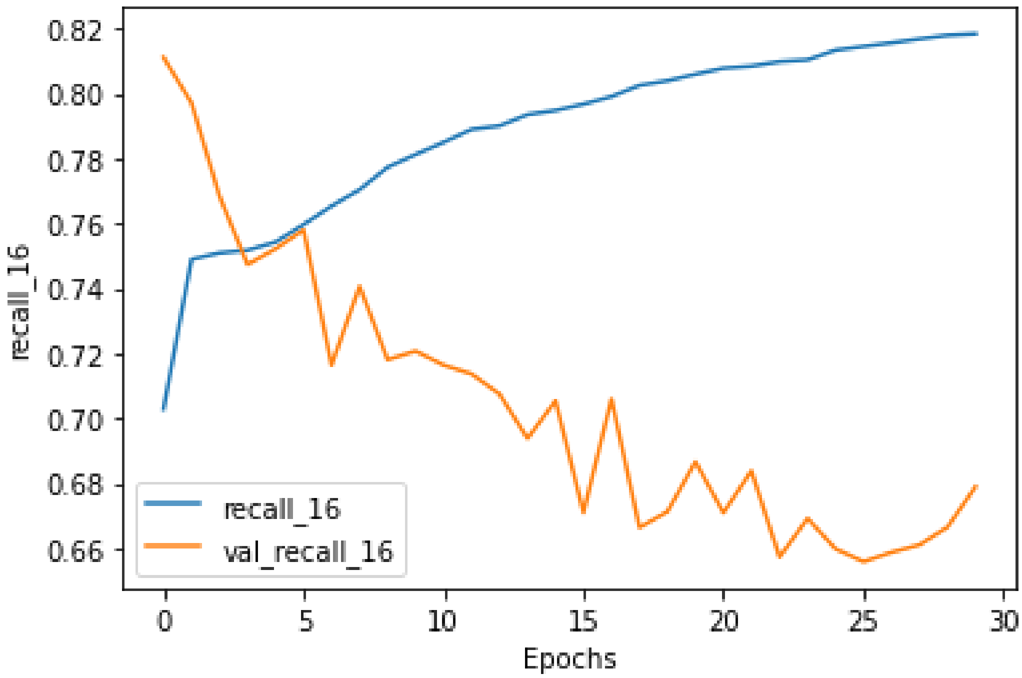
```
val_recall_31: 0.4363
Epoch 7/10
100/100 [==============================] - 14s 144ms/step - loss: 1.1155 -
accuracy: 0.5724 - recall_31: 0.5419 - val_loss: 0.7166 - val_accuracy: 0.4208 -
val_recall_31: 0.6307
Epoch 8/10
100/100 [==============================] - 14s 145ms/step - loss: 1.1148 -
accuracy: 0.6006 - recall_31: 0.5092 - val_loss: 0.7004 - val_accuracy: 0.4487 -
val_recall_31: 0.6963
Epoch 9/10
100/100 [==============================] - 14s 144ms/step - loss: 1.1300 -
accuracy: 0.5574 - recall_31: 0.5816 - val_loss: 0.7312 - val_accuracy: 0.4928 -
val_recall_31: 0.7024
Epoch 10/10
100/100 [==============================] - 15s 155ms/step - loss: 1.1232 -
accuracy: 0.5663 - recall_31: 0.5899 - val_loss: 0.7443 - val_accuracy: 0.3631 -
val_recall_31: 0.5528
INFO:tensorflow:Assets written to: /tmp/tmpjph3k67r/model/data/model/assets
 946165/Unknown - 21854s 23ms/step - loss: 0.7331 - accuracy: 0.4271 -
recall_31: 0.6017
```

Model accuracy for the displayed run has an accuracy for both training and validation between 85% and 90%. Validation loss is increasing, and validation recall is decreasing.

[63]:
```python
import matplotlib.pyplot as plt

plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
plot_graphs(history, "recall_16")
```


1.0
0.8
0.6
0.4
0.2
0.0
accuracy
accuracy
val_accuracy
0
5
10
15
20
25
30
Epochs


0.44
0.42
0.40
0.38
0.36
0.34
0.32
0.30
0.28
loss
loss
val_loss
0
5
10
15
20
25
30
Epochs




```
[65]: for s, e, sv, cw in zip(steps, epochs, steps_val, class_weight):
        with mlflow.start_run(run_name='WeightTied2'):
          layer1 = layers.Dense(25, activation='relu', name='layer1',
                                input_shape=(None, 25))
          layer2 = DenseTied(25, activation='relu', name='tied1', tied_to = layer1)

          model4 = Sequential()
          model4.add(layer1)
          model4.add(layers.LSTM(25, dropout=0.2,recurrent_dropout=0.2,
                                  input_shape=(None, 25)))
          model4.add(layer2)
          model4.add(layers.Dense(1, activation='sigmoid'))

          display(model4.summary())

          model4.compile(optimizer=RMSprop(), loss='binary_crossentropy',
                         metrics=['accuracy', tf.keras.metrics.Recall()])
          history = model4.fit(trainX, trainY,
                                steps_per_epoch=s,
                                epochs=e,
                                validation_data=(valX, valY),
                                validation_steps=sv,
                                # class_weight=cw
          )
          # history = model.fit(train_gen,
```

```python
    #                     steps_per_epoch=s,
    #                     epochs=e,
    #                     validation_data=val_gen,
    #                     validation_steps=sv,
    #                     class_weight=cw
    #                     )
    mlflow.keras.log_model(model4, "WeightTied2", save_format='tf')
    mlflow.log_params(history.params)
    scores = model4.evaluate(testX, testY)
    mlflow.log_metrics({k: v for k, v in zip(
        ['loss', 'accuracy', 'recall'], scores)})
```

```
Model: "sequential_14"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, None, 25)          650
_________________________________________________________________
lstm_18 (LSTM)               (None, 25)                5100
_________________________________________________________________
tied1 (DenseTied)            (None, 25)                1300
_________________________________________________________________
dense_18 (Dense)             (None, 1)                 26
=================================================================
Total params: 6,426
Trainable params: 5,801
Non-trainable params: 625
_________________________________________________________________
None


Epoch 1/10
100/100 [==============================] - 8s 40ms/step - loss: 0.6020 -
accuracy: 0.7318 - recall_18: 0.3841 - val_loss: 0.3017 - val_accuracy: 0.8827 -
val_recall_18: 0.5451
Epoch 2/10
100/100 [==============================] - 3s 35ms/step - loss: 0.2989 -
accuracy: 0.8727 - recall_18: 0.4852 - val_loss: 0.2527 - val_accuracy: 0.8817 -
val_recall_18: 0.5835
Epoch 3/10
100/100 [==============================] - 3s 35ms/step - loss: 0.2664 -
accuracy: 0.8753 - recall_18: 0.5037 - val_loss: 0.2494 - val_accuracy: 0.8821 -
val_recall_18: 0.5679
Epoch 4/10
100/100 [==============================] - 3s 34ms/step - loss: 0.2605 -
accuracy: 0.8782 - recall_18: 0.5087 - val_loss: 0.2481 - val_accuracy: 0.8840 -
val_recall_18: 0.5537
```

```
Epoch 5/10
100/100 [==============================] - 3s 34ms/step - loss: 0.2562 -
accuracy: 0.8794 - recall_18: 0.5084 - val_loss: 0.2470 - val_accuracy: 0.8854 -
val_recall_18: 0.5380
Epoch 6/10
100/100 [==============================] - 4s 39ms/step - loss: 0.2525 -
accuracy: 0.8822 - recall_18: 0.5183 - val_loss: 0.2487 - val_accuracy: 0.8848 -
val_recall_18: 0.5251
Epoch 7/10
100/100 [==============================] - 3s 35ms/step - loss: 0.2492 -
accuracy: 0.8840 - recall_18: 0.5228 - val_loss: 0.2528 - val_accuracy: 0.8828 -
val_recall_18: 0.5424
Epoch 8/10
100/100 [==============================] - 3s 34ms/step - loss: 0.2455 -
accuracy: 0.8870 - recall_18: 0.5371 - val_loss: 0.2521 - val_accuracy: 0.8844 -
val_recall_18: 0.5194
Epoch 9/10
100/100 [==============================] - 3s 34ms/step - loss: 0.2425 -
accuracy: 0.8887 - recall_18: 0.5445 - val_loss: 0.2541 - val_accuracy: 0.8833 -
val_recall_18: 0.5140
Epoch 10/10
100/100 [==============================] - 3s 34ms/step - loss: 0.2426 -
accuracy: 0.8892 - recall_18: 0.5544 - val_loss: 0.2556 - val_accuracy: 0.8829 -
val_recall_18: 0.5329
INFO:tensorflow:Assets written to: /tmp/tmpdn9q1htv/model/data/model/assets
2556/2556 [==============================] - 4s 1ms/step - loss: 0.2742 -
accuracy: 0.8762 - recall_18: 0.7106
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, None, 25)          650
_________________________________________________________________
lstm_19 (LSTM)               (None, 25)                5100
_________________________________________________________________
tied1 (DenseTied)            (None, 25)                1300
_________________________________________________________________
dense_19 (Dense)             (None, 1)                 26
=================================================================
Total params: 6,426
Trainable params: 5,801
Non-trainable params: 625
_________________________________________________________________
None


Epoch 1/20
300/300 [==============================] - 8s 15ms/step - loss: 0.4966 -
```

```
accuracy: 0.7852 - recall_19: 0.4684 - val_loss: 0.2499 - val_accuracy: 0.8842 -
val_recall_19: 0.5401
Epoch 2/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2600 -
accuracy: 0.8790 - recall_19: 0.4940 - val_loss: 0.2477 - val_accuracy: 0.8824 -
val_recall_19: 0.4749
Epoch 3/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2508 -
accuracy: 0.8847 - recall_19: 0.5120 - val_loss: 0.2497 - val_accuracy: 0.8853 -
val_recall_19: 0.5210
Epoch 4/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2443 -
accuracy: 0.8890 - recall_19: 0.5311 - val_loss: 0.2525 - val_accuracy: 0.8825 -
val_recall_19: 0.5201
Epoch 5/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2401 -
accuracy: 0.8921 - recall_19: 0.5512 - val_loss: 0.2540 - val_accuracy: 0.8831 -
val_recall_19: 0.5028
Epoch 6/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2370 -
accuracy: 0.8937 - recall_19: 0.5601 - val_loss: 0.2552 - val_accuracy: 0.8824 -
val_recall_19: 0.4933
Epoch 7/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2317 -
accuracy: 0.8969 - recall_19: 0.5665 - val_loss: 0.2583 - val_accuracy: 0.8812 -
val_recall_19: 0.4856
Epoch 8/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2275 -
accuracy: 0.8981 - recall_19: 0.5724 - val_loss: 0.2593 - val_accuracy: 0.8814 -
val_recall_19: 0.5028
Epoch 9/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2247 -
accuracy: 0.9000 - recall_19: 0.5881 - val_loss: 0.2576 - val_accuracy: 0.8827 -
val_recall_19: 0.4993
Epoch 10/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2221 -
accuracy: 0.9023 - recall_19: 0.5928 - val_loss: 0.2632 - val_accuracy: 0.8795 -
val_recall_19: 0.4555
Epoch 11/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2210 -
accuracy: 0.9028 - recall_19: 0.6000 - val_loss: 0.2645 - val_accuracy: 0.8796 -
val_recall_19: 0.4682
Epoch 12/20
300/300 [==============================] - 4s 15ms/step - loss: 0.2206 -
accuracy: 0.9033 - recall_19: 0.6022 - val_loss: 0.2693 - val_accuracy: 0.8783 -
val_recall_19: 0.4559
Epoch 13/20
300/300 [==============================] - 4s 14ms/step - loss: 0.2160 -
```

```
accuracy: 0.9059 - recall_19: 0.6124 - val_loss: 0.2692 - val_accuracy: 0.8786 -
val_recall_19: 0.4788
Epoch 14/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2144 -
accuracy: 0.9060 - recall_19: 0.6129 - val_loss: 0.2703 - val_accuracy: 0.8793 -
val_recall_19: 0.4912
Epoch 15/20
300/300 [==============================] - 4s 13ms/step - loss: 0.2129 -
accuracy: 0.9075 - recall_19: 0.6217 - val_loss: 0.2749 - val_accuracy: 0.8761 -
val_recall_19: 0.4639
Epoch 16/20
300/300 [==============================] - 4s 14ms/step - loss: 0.2111 -
accuracy: 0.9082 - recall_19: 0.6281 - val_loss: 0.2769 - val_accuracy: 0.8774 -
val_recall_19: 0.5023
Epoch 17/20
300/300 [==============================] - 4s 14ms/step - loss: 0.2095 -
accuracy: 0.9086 - recall_19: 0.6305 - val_loss: 0.2783 - val_accuracy: 0.8775 -
val_recall_19: 0.4724
Epoch 18/20
300/300 [==============================] - 4s 14ms/step - loss: 0.2077 -
accuracy: 0.9102 - recall_19: 0.6355 - val_loss: 0.2801 - val_accuracy: 0.8772 -
val_recall_19: 0.4772
Epoch 19/20
300/300 [==============================] - 4s 14ms/step - loss: 0.2070 -
accuracy: 0.9101 - recall_19: 0.6361 - val_loss: 0.2797 - val_accuracy: 0.8784 -
val_recall_19: 0.4515
Epoch 20/20
300/300 [==============================] - 4s 14ms/step - loss: 0.2040 -
accuracy: 0.9114 - recall_19: 0.6428 - val_loss: 0.2818 - val_accuracy: 0.8772 -
val_recall_19: 0.4858
INFO:tensorflow:Assets written to: /tmp/tmpyznwqtxt/model/data/model/assets
2556/2556 [==============================] - 4s 1ms/step - loss: 0.2985 -
accuracy: 0.8721 - recall_19: 0.5547
Model: "sequential_16"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1 (Dense)               (None, None, 25)          650
_________________________________________________________________
lstm_20 (LSTM)               (None, 25)                5100
_________________________________________________________________
tied1 (DenseTied)            (None, 25)                1300
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 26
=================================================================
Total params: 6,426
Trainable params: 5,801
Non-trainable params: 625
```

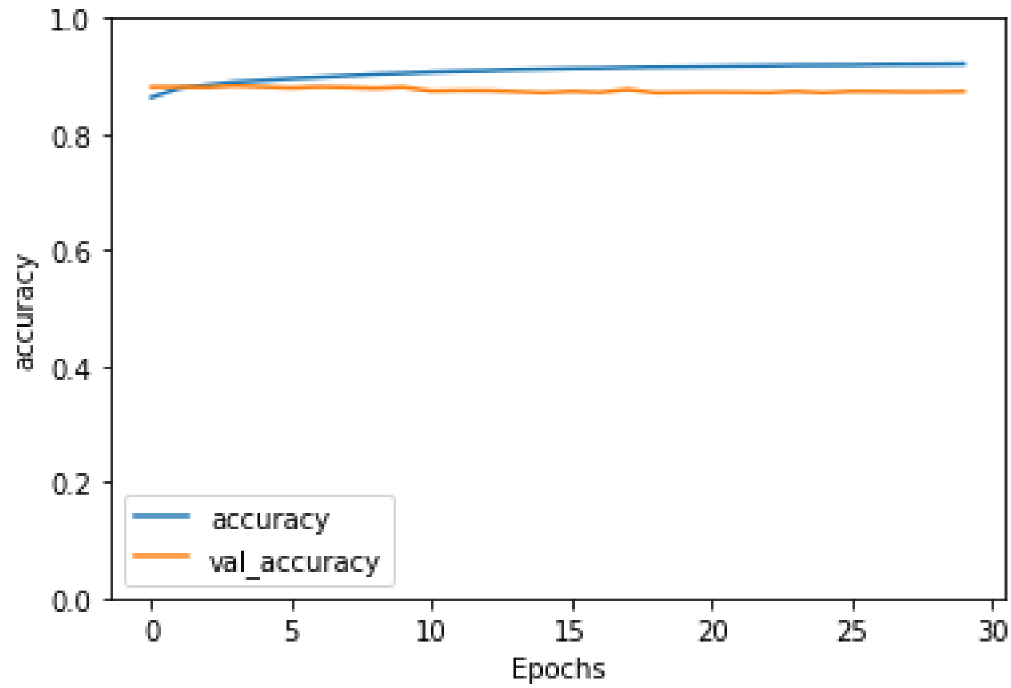
```
-----------------------------------------------------------------
None


Epoch 1/30
500/500 [==============================] - 9s 11ms/step - loss: 0.4035 -
accuracy: 0.8278 - recall_20: 0.3406 - val_loss: 0.2506 - val_accuracy: 0.8813 -
val_recall_20: 0.5643
Epoch 2/30
500/500 [==============================] - 5s 9ms/step - loss: 0.2584 -
accuracy: 0.8786 - recall_20: 0.5133 - val_loss: 0.2510 - val_accuracy: 0.8817 -
val_recall_20: 0.5520
Epoch 3/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2474 -
accuracy: 0.8832 - recall_20: 0.5247 - val_loss: 0.2573 - val_accuracy: 0.8808 -
val_recall_20: 0.5885
Epoch 4/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2402 -
accuracy: 0.8892 - recall_20: 0.5348 - val_loss: 0.2538 - val_accuracy: 0.8835 -
val_recall_20: 0.5415
Epoch 5/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2359 -
accuracy: 0.8914 - recall_20: 0.5381 - val_loss: 0.2580 - val_accuracy: 0.8816 -
val_recall_20: 0.5460
Epoch 6/30
500/500 [==============================] - 5s 9ms/step - loss: 0.2303 -
accuracy: 0.8946 - recall_20: 0.5462 - val_loss: 0.2618 - val_accuracy: 0.8798 -
val_recall_20: 0.5494
Epoch 7/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2262 -
accuracy: 0.8973 - recall_20: 0.5584 - val_loss: 0.2583 - val_accuracy: 0.8817 -
val_recall_20: 0.5216
Epoch 8/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2204 -
accuracy: 0.9006 - recall_20: 0.5703 - val_loss: 0.2635 - val_accuracy: 0.8806 -
val_recall_20: 0.5358
Epoch 9/30
500/500 [==============================] - 5s 11ms/step - loss: 0.2197 -
accuracy: 0.9022 - recall_20: 0.5774 - val_loss: 0.2666 - val_accuracy: 0.8792 -
val_recall_20: 0.5393
Epoch 10/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2149 -
accuracy: 0.9042 - recall_20: 0.5928 - val_loss: 0.2658 - val_accuracy: 0.8811 -
val_recall_20: 0.5203
Epoch 11/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2117 -
accuracy: 0.9072 - recall_20: 0.6042 - val_loss: 0.2775 - val_accuracy: 0.8745 -
val_recall_20: 0.5539
```

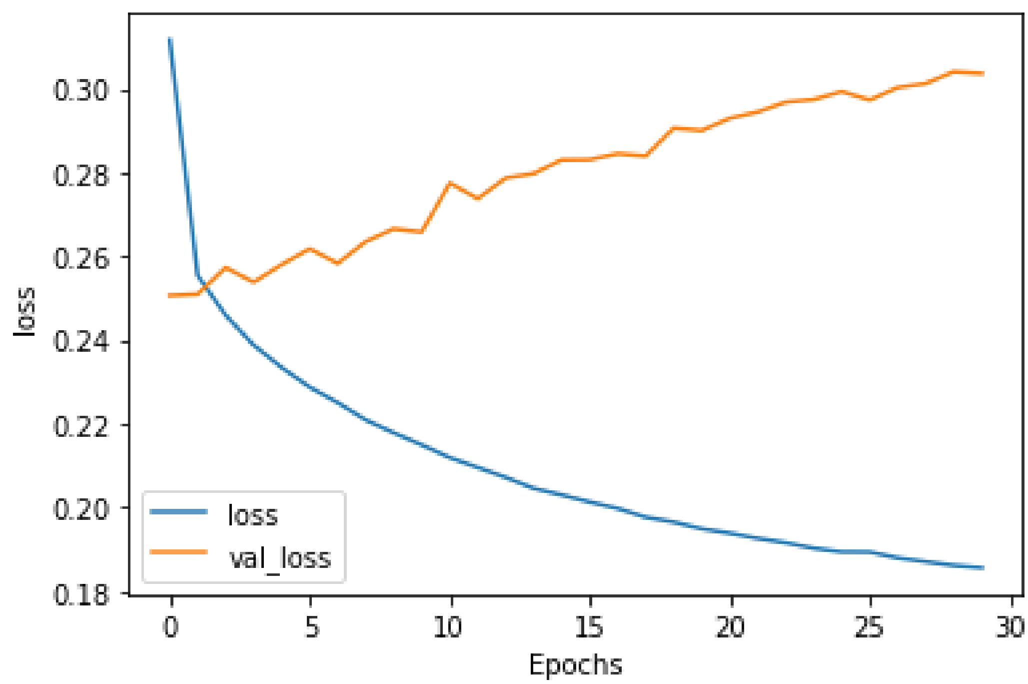
```
Epoch 12/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2106 -
accuracy: 0.9073 - recall_20: 0.6067 - val_loss: 0.2738 - val_accuracy: 0.8753 -
val_recall_20: 0.5130
Epoch 13/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2072 -
accuracy: 0.9096 - recall_20: 0.6122 - val_loss: 0.2787 - val_accuracy: 0.8751 -
val_recall_20: 0.5158
Epoch 14/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2058 -
accuracy: 0.9106 - recall_20: 0.6180 - val_loss: 0.2797 - val_accuracy: 0.8735 -
val_recall_20: 0.5068
Epoch 15/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2050 -
accuracy: 0.9112 - recall_20: 0.6240 - val_loss: 0.2830 - val_accuracy: 0.8715 -
val_recall_20: 0.5097
Epoch 16/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2024 -
accuracy: 0.9124 - recall_20: 0.6282 - val_loss: 0.2830 - val_accuracy: 0.8738 -
val_recall_20: 0.5064
Epoch 17/30
500/500 [==============================] - 5s 10ms/step - loss: 0.2001 -
accuracy: 0.9130 - recall_20: 0.6314 - val_loss: 0.2844 - val_accuracy: 0.8721 -
val_recall_20: 0.5091
Epoch 18/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1968 -
accuracy: 0.9150 - recall_20: 0.6407 - val_loss: 0.2839 - val_accuracy: 0.8771 -
val_recall_20: 0.4710
Epoch 19/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1975 -
accuracy: 0.9145 - recall_20: 0.6355 - val_loss: 0.2906 - val_accuracy: 0.8713 -
val_recall_20: 0.5274
Epoch 20/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1955 -
accuracy: 0.9152 - recall_20: 0.6447 - val_loss: 0.2900 - val_accuracy: 0.8720 -
val_recall_20: 0.5138
Epoch 21/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1945 -
accuracy: 0.9158 - recall_20: 0.6486 - val_loss: 0.2929 - val_accuracy: 0.8724 -
val_recall_20: 0.5127
Epoch 22/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1928 -
accuracy: 0.9175 - recall_20: 0.6499 - val_loss: 0.2944 - val_accuracy: 0.8721 -
val_recall_20: 0.5474
Epoch 23/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1915 -
accuracy: 0.9175 - recall_20: 0.6590 - val_loss: 0.2968 - val_accuracy: 0.8715 -
val_recall_20: 0.5291
```

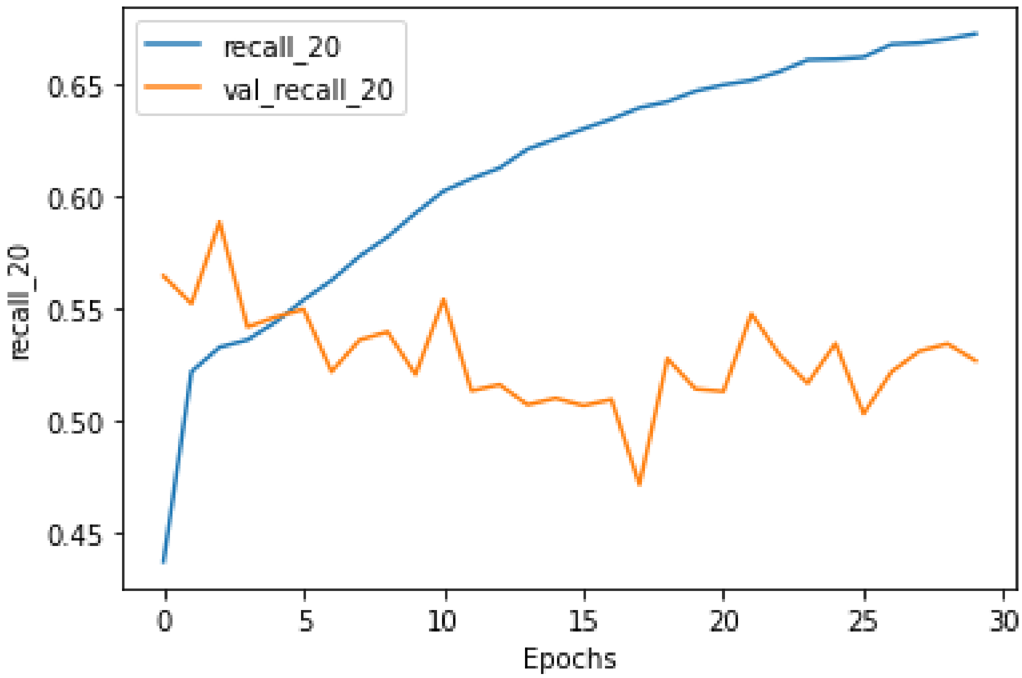
```
Epoch 24/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1887 -
accuracy: 0.9190 - recall_20: 0.6617 - val_loss: 0.2974 - val_accuracy: 0.8737 -
val_recall_20: 0.5164
Epoch 25/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1890 -
accuracy: 0.9193 - recall_20: 0.6604 - val_loss: 0.2993 - val_accuracy: 0.8715 -
val_recall_20: 0.5342
Epoch 26/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1898 -
accuracy: 0.9187 - recall_20: 0.6591 - val_loss: 0.2973 - val_accuracy: 0.8738 -
val_recall_20: 0.5028
Epoch 27/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1896 -
accuracy: 0.9193 - recall_20: 0.6666 - val_loss: 0.3003 - val_accuracy: 0.8735 -
val_recall_20: 0.5215
Epoch 28/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1874 -
accuracy: 0.9205 - recall_20: 0.6692 - val_loss: 0.3012 - val_accuracy: 0.8729 -
val_recall_20: 0.5308
Epoch 29/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1865 -
accuracy: 0.9206 - recall_20: 0.6703 - val_loss: 0.3040 - val_accuracy: 0.8730 -
val_recall_20: 0.5341
Epoch 30/30
500/500 [==============================] - 5s 10ms/step - loss: 0.1858 -
accuracy: 0.9205 - recall_20: 0.6700 - val_loss: 0.3036 - val_accuracy: 0.8738 -
val_recall_20: 0.5265
INFO:tensorflow:Assets written to: /tmp/tmpvld3_73l/model/data/model/assets
2556/2556 [==============================] - 4s 1ms/step - loss: 0.3353 -
accuracy: 0.8710 - recall_20: 0.6282
```

Model accuracy for the displayed run has an accuracy for both training and validation between 85% and 90%. Validation loss is increasing, and validation recall is decreasing.

[66]:
```
plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
plot_graphs(history, "recall_20")
```


1.0
0.8
0.6
0.4
0.2
0.0
accuracy
0
5
10
15
20
25
30
Epochs
accuracy
val_accuracy


0.30
0.28
0.26
0.24
0.22
0.20
0.18
loss
0
5
10
15
20
25
30
Epochs
loss
val_loss



Experiment 4: Weight-tied stacked LSTM

[69]:
```
for s, e, sv, cw in zip(steps, epochs, steps_val, class_weight):
  with mlflow.start_run(run_name='WeightTiedLSTM'):
    layer1 = layers.LSTM(25, dropout=0.2,recurrent_dropout=0.2,
                     input_shape=(None, 25), return_sequences=True)
    layer2 = TiedtLSTM(25, layer1, dropout=0.2,recurrent_dropout=0.2,
                        input_shape=(None, 25), return_sequences=True)

    model5 = Sequential()
    model5.add(layer1)
    model5.add(layer2)
    model5.add(layer2)
    model5.add(layers.Dense(1, activation='sigmoid'))
    display(model5.summary())

    model5.compile(optimizer=RMSprop(), loss='binary_crossentropy',
                  metrics=['accuracy', tf.keras.metrics.Recall()])
    history = model5.fit(trainX, trainY,
                        steps_per_epoch=s,
                        epochs=e,
                        validation_data=(valX, valY),
                        validation_steps=sv,
                        # class_weight=cw
    )
```

```python
    # history = model.fit(train_gen,
    #                     steps_per_epoch=s,
    #                     epochs=e,
    #                     validation_data=val_gen,
    #                     validation_steps=sv,
    #                     class_weight=cw
    #                     )
    mlflow.keras.log_model(model5, "WeightTiedLSTM", save_format='tf')
    mlflow.log_params(history.params)
    scores = model5.evaluate(testX, testY)
    mlflow.log_metrics({k: v for k, v in zip(
        ['loss', 'accuracy', 'recall'], scores)})
```

```
Model: "sequential_18"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_22 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
tiedt_lstm_1 (TiedtLSTM)     (None, None, 25)          12725
_________________________________________________________________
dense_21 (Dense)             (None, None, 1)           26
=================================================================
Total params: 12,751
Trainable params: 10,251
Non-trainable params: 2,500
_________________________________________________________________
None


Epoch 1/10
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_1/Variable:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_1/Variable:0'] when minimizing the loss.
100/100 [==============================] - 18s 91ms/step - loss: 0.6063 -
accuracy: 0.8177 - recall_21: 0.0233 - val_loss: 0.3054 - val_accuracy: 0.8487 -
val_recall_21: 0.0000e+00
Epoch 2/10
100/100 [==============================] - 8s 83ms/step - loss: 0.3046 -
accuracy: 0.8398 - recall_21: 0.0742 - val_loss: 0.2553 - val_accuracy: 0.8757 -
val_recall_21: 0.6218
Epoch 3/10
100/100 [==============================] - 8s 80ms/step - loss: 0.2742 -
accuracy: 0.8703 - recall_21: 0.5486 - val_loss: 0.2510 - val_accuracy: 0.8762 -
val_recall_21: 0.5804
Epoch 4/10
```

```
100/100 [==============================] - 8s 79ms/step - loss: 0.2696 -
accuracy: 0.8743 - recall_21: 0.5180 - val_loss: 0.2503 - val_accuracy: 0.8772 -
val_recall_21: 0.5601
Epoch 5/10
100/100 [==============================] - 8s 79ms/step - loss: 0.2658 -
accuracy: 0.8756 - recall_21: 0.5021 - val_loss: 0.2493 - val_accuracy: 0.8793 -
val_recall_21: 0.5539
Epoch 6/10
100/100 [==============================] - 8s 80ms/step - loss: 0.2624 -
accuracy: 0.8767 - recall_21: 0.4946 - val_loss: 0.2501 - val_accuracy: 0.8796 -
val_recall_21: 0.5651
Epoch 7/10
100/100 [==============================] - 8s 80ms/step - loss: 0.2604 -
accuracy: 0.8772 - recall_21: 0.4965 - val_loss: 0.2487 - val_accuracy: 0.8815 -
val_recall_21: 0.5532
Epoch 8/10
100/100 [==============================] - 8s 82ms/step - loss: 0.2568 -
accuracy: 0.8790 - recall_21: 0.5030 - val_loss: 0.2490 - val_accuracy: 0.8826 -
val_recall_21: 0.5514
Epoch 9/10
100/100 [==============================] - 9s 89ms/step - loss: 0.2556 -
accuracy: 0.8797 - recall_21: 0.5057 - val_loss: 0.2492 - val_accuracy: 0.8837 -
val_recall_21: 0.5444
Epoch 10/10
100/100 [==============================] - 9s 89ms/step - loss: 0.2545 -
accuracy: 0.8812 - recall_21: 0.5037 - val_loss: 0.2507 - val_accuracy: 0.8853 -
val_recall_21: 0.5591
INFO:tensorflow:Assets written to: /tmp/tmp2vmakvad/model/data/model/assets
2556/2556 [==============================] - 4s 2ms/step - loss: 0.2783 -
accuracy: 0.8747 - recall_21: 0.7698
Model: "sequential_19"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_23 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
tiedt_lstm_2 (TiedtLSTM)     (None, None, 25)          12725
_________________________________________________________________
dense_22 (Dense)             (None, None, 1)           26
=================================================================
Total params: 12,751
Trainable params: 10,251
Non-trainable params: 2,500
_________________________________________________________________
None

Epoch 1/20
```

```
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_2/Variable:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_2/Variable:0'] when minimizing the loss.
300/300 [==============================] - 18s 32ms/step - loss: 0.4770 -
accuracy: 0.8364 - recall_22: 0.1022 - val_loss: 0.2522 - val_accuracy: 0.8765 -
val_recall_22: 0.5868
Epoch 2/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2699 -
accuracy: 0.8735 - recall_22: 0.5156 - val_loss: 0.2474 - val_accuracy: 0.8821 -
val_recall_22: 0.5521
Epoch 3/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2621 -
accuracy: 0.8771 - recall_22: 0.4964 - val_loss: 0.2465 - val_accuracy: 0.8837 -
val_recall_22: 0.5277
Epoch 4/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2581 -
accuracy: 0.8796 - recall_22: 0.5038 - val_loss: 0.2487 - val_accuracy: 0.8848 -
val_recall_22: 0.5477
Epoch 5/20
300/300 [==============================] - 9s 28ms/step - loss: 0.2545 -
accuracy: 0.8810 - recall_22: 0.5141 - val_loss: 0.2507 - val_accuracy: 0.8824 -
val_recall_22: 0.4921
Epoch 6/20
300/300 [==============================] - 8s 28ms/step - loss: 0.2518 -
accuracy: 0.8819 - recall_22: 0.5185 - val_loss: 0.2514 - val_accuracy: 0.8841 -
val_recall_22: 0.5431
Epoch 7/20
300/300 [==============================] - 8s 28ms/step - loss: 0.2494 -
accuracy: 0.8830 - recall_22: 0.5295 - val_loss: 0.2509 - val_accuracy: 0.8837 -
val_recall_22: 0.5260
Epoch 8/20
300/300 [==============================] - 9s 31ms/step - loss: 0.2479 -
accuracy: 0.8848 - recall_22: 0.5327 - val_loss: 0.2518 - val_accuracy: 0.8840 -
val_recall_22: 0.5158
Epoch 9/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2477 -
accuracy: 0.8848 - recall_22: 0.5261 - val_loss: 0.2532 - val_accuracy: 0.8834 -
val_recall_22: 0.5186
Epoch 10/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2438 -
accuracy: 0.8874 - recall_22: 0.5358 - val_loss: 0.2531 - val_accuracy: 0.8831 -
val_recall_22: 0.4959
Epoch 11/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2443 -
accuracy: 0.8867 - recall_22: 0.5346 - val_loss: 0.2556 - val_accuracy: 0.8825 -
val_recall_22: 0.5195
Epoch 12/20
```

```
300/300 [==============================] - 9s 29ms/step - loss: 0.2390 -
accuracy: 0.8897 - recall_22: 0.5470 - val_loss: 0.2565 - val_accuracy: 0.8816 -
val_recall_22: 0.5060
Epoch 13/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2391 -
accuracy: 0.8901 - recall_22: 0.5488 - val_loss: 0.2578 - val_accuracy: 0.8821 -
val_recall_22: 0.5387
Epoch 14/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2385 -
accuracy: 0.8903 - recall_22: 0.5539 - val_loss: 0.2583 - val_accuracy: 0.8818 -
val_recall_22: 0.5045
Epoch 15/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2374 -
accuracy: 0.8907 - recall_22: 0.5519 - val_loss: 0.2586 - val_accuracy: 0.8822 -
val_recall_22: 0.5100
Epoch 16/20
300/300 [==============================] - 9s 30ms/step - loss: 0.2368 -
accuracy: 0.8911 - recall_22: 0.5587 - val_loss: 0.2601 - val_accuracy: 0.8823 -
val_recall_22: 0.5165
Epoch 17/20
300/300 [==============================] - 9s 30ms/step - loss: 0.2343 -
accuracy: 0.8927 - recall_22: 0.5603 - val_loss: 0.2623 - val_accuracy: 0.8810 -
val_recall_22: 0.5194
Epoch 18/20
300/300 [==============================] - 9s 30ms/step - loss: 0.2350 -
accuracy: 0.8932 - recall_22: 0.5645 - val_loss: 0.2617 - val_accuracy: 0.8805 -
val_recall_22: 0.5189
Epoch 19/20
300/300 [==============================] - 9s 29ms/step - loss: 0.2342 -
accuracy: 0.8935 - recall_22: 0.5660 - val_loss: 0.2629 - val_accuracy: 0.8787 -
val_recall_22: 0.4943
Epoch 20/20
300/300 [==============================] - 9s 28ms/step - loss: 0.2321 -
accuracy: 0.8945 - recall_22: 0.5646 - val_loss: 0.2647 - val_accuracy: 0.8796 -
val_recall_22: 0.5266
INFO:tensorflow:Assets written to: /tmp/tmp9oyvtout/model/data/model/assets
2556/2556 [==============================] - 5s 2ms/step - loss: 0.2846 -
accuracy: 0.8782 - recall_22: 0.6881
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_24 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
tiedt_lstm_3 (TiedtLSTM)     (None, None, 25)          12725
_________________________________________________________________
dense_23 (Dense)             (None, None, 1)           26
=================================================================
```

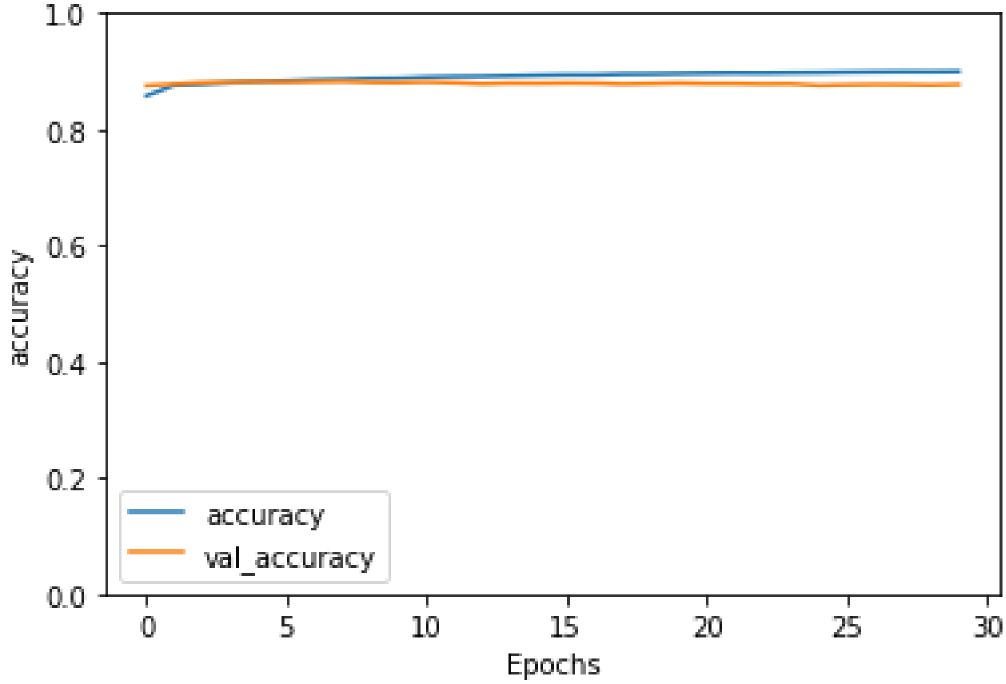
```
Total params: 12,751
Trainable params: 10,251
Non-trainable params: 2,500
_________________________________________________________________
None


Epoch 1/30
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_3/Variable:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_3/Variable:0'] when minimizing the loss.
500/500 [==============================] - 21s 23ms/step - loss: 0.4238 -
accuracy: 0.8398 - recall_23: 0.1516 - val_loss: 0.2522 - val_accuracy: 0.8754 -
val_recall_23: 0.5627
Epoch 2/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2665 -
accuracy: 0.8754 - recall_23: 0.5076 - val_loss: 0.2532 - val_accuracy: 0.8781 -
val_recall_23: 0.5560
Epoch 3/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2603 -
accuracy: 0.8769 - recall_23: 0.4926 - val_loss: 0.2519 - val_accuracy: 0.8806 -
val_recall_23: 0.4908
Epoch 4/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2562 -
accuracy: 0.8796 - recall_23: 0.4942 - val_loss: 0.2540 - val_accuracy: 0.8815 -
val_recall_23: 0.5277
Epoch 5/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2526 -
accuracy: 0.8819 - recall_23: 0.5047 - val_loss: 0.2533 - val_accuracy: 0.8823 -
val_recall_23: 0.5178
Epoch 6/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2507 -
accuracy: 0.8832 - recall_23: 0.5097 - val_loss: 0.2540 - val_accuracy: 0.8817 -
val_recall_23: 0.4958
Epoch 7/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2487 -
accuracy: 0.8848 - recall_23: 0.5142 - val_loss: 0.2547 - val_accuracy: 0.8822 -
val_recall_23: 0.4932
Epoch 8/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2462 -
accuracy: 0.8856 - recall_23: 0.5129 - val_loss: 0.2577 - val_accuracy: 0.8826 -
val_recall_23: 0.5356
Epoch 9/30
500/500 [==============================] - 11s 21ms/step - loss: 0.2450 -
accuracy: 0.8872 - recall_23: 0.5228 - val_loss: 0.2578 - val_accuracy: 0.8812 -
val_recall_23: 0.5145
Epoch 10/30
```

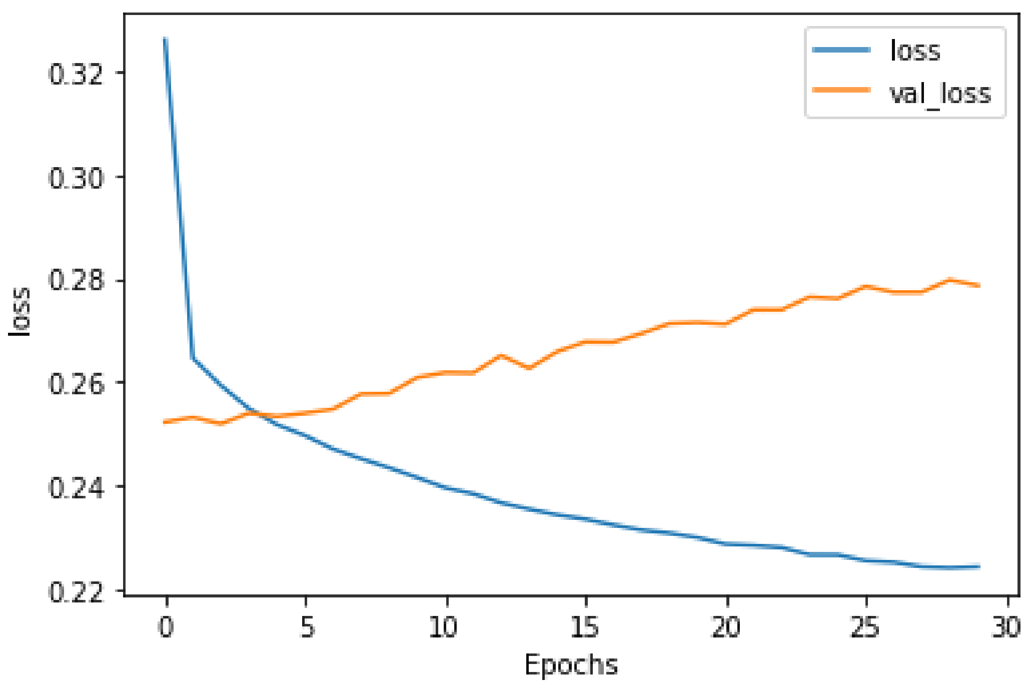
```
500/500 [==============================] - 11s 22ms/step - loss: 0.2409 -
accuracy: 0.8886 - recall_23: 0.5292 - val_loss: 0.2609 - val_accuracy: 0.8808 -
val_recall_23: 0.5441
Epoch 11/30
500/500 [==============================] - 10s 21ms/step - loss: 0.2390 -
accuracy: 0.8904 - recall_23: 0.5376 - val_loss: 0.2618 - val_accuracy: 0.8818 -
val_recall_23: 0.5369
Epoch 12/30
500/500 [==============================] - 10s 21ms/step - loss: 0.2400 -
accuracy: 0.8906 - recall_23: 0.5362 - val_loss: 0.2617 - val_accuracy: 0.8806 -
val_recall_23: 0.5363
Epoch 13/30
500/500 [==============================] - 10s 21ms/step - loss: 0.2363 -
accuracy: 0.8920 - recall_23: 0.5447 - val_loss: 0.2651 - val_accuracy: 0.8787 -
val_recall_23: 0.5161
Epoch 14/30
500/500 [==============================] - 10s 21ms/step - loss: 0.2350 -
accuracy: 0.8928 - recall_23: 0.5442 - val_loss: 0.2626 - val_accuracy: 0.8798 -
val_recall_23: 0.5217
Epoch 15/30
500/500 [==============================] - 10s 21ms/step - loss: 0.2354 -
accuracy: 0.8928 - recall_23: 0.5516 - val_loss: 0.2659 - val_accuracy: 0.8795 -
val_recall_23: 0.5304
Epoch 16/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2336 -
accuracy: 0.8934 - recall_23: 0.5504 - val_loss: 0.2677 - val_accuracy: 0.8800 -
val_recall_23: 0.5599
Epoch 17/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2316 -
accuracy: 0.8948 - recall_23: 0.5576 - val_loss: 0.2677 - val_accuracy: 0.8797 -
val_recall_23: 0.5264
Epoch 18/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2306 -
accuracy: 0.8955 - recall_23: 0.5619 - val_loss: 0.2694 - val_accuracy: 0.8783 -
val_recall_23: 0.5205
Epoch 19/30
500/500 [==============================] - 10s 21ms/step - loss: 0.2314 -
accuracy: 0.8954 - recall_23: 0.5614 - val_loss: 0.2713 - val_accuracy: 0.8790 -
val_recall_23: 0.5442
Epoch 20/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2307 -
accuracy: 0.8961 - recall_23: 0.5620 - val_loss: 0.2715 - val_accuracy: 0.8799 -
val_recall_23: 0.5638
Epoch 21/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2298 -
accuracy: 0.8966 - recall_23: 0.5662 - val_loss: 0.2712 - val_accuracy: 0.8790 -
val_recall_23: 0.4982
Epoch 22/30
```

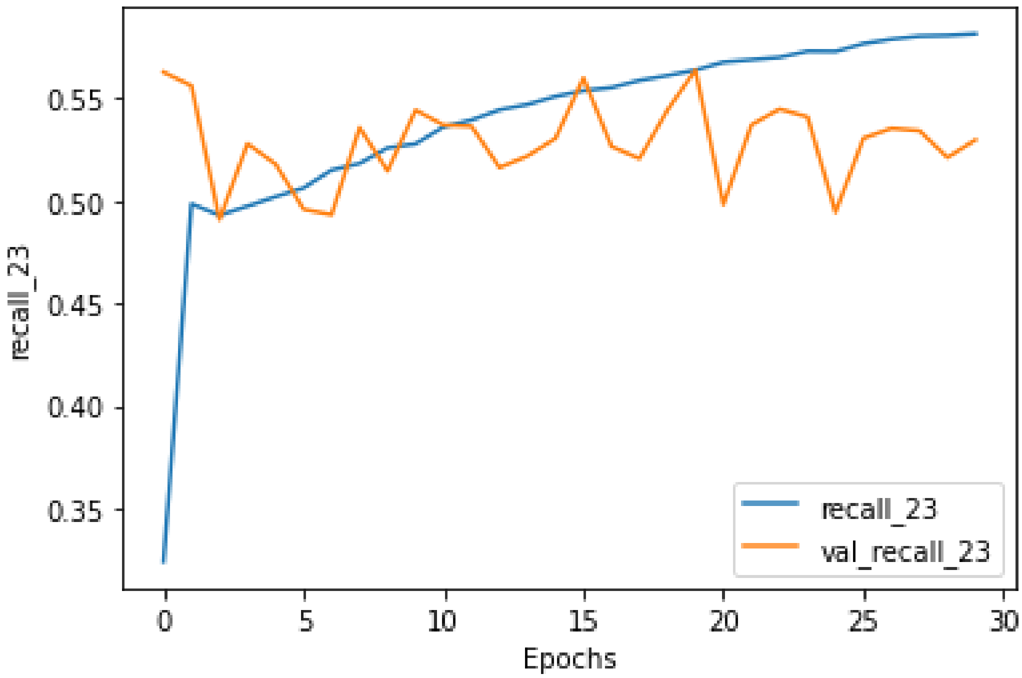
```
500/500 [==============================] - 10s 20ms/step - loss: 0.2290 -
accuracy: 0.8960 - recall_23: 0.5631 - val_loss: 0.2740 - val_accuracy: 0.8790 -
val_recall_23: 0.5369
Epoch 23/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2276 -
accuracy: 0.8971 - recall_23: 0.5719 - val_loss: 0.2740 - val_accuracy: 0.8782 -
val_recall_23: 0.5447
Epoch 24/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2244 -
accuracy: 0.8981 - recall_23: 0.5753 - val_loss: 0.2765 - val_accuracy: 0.8785 -
val_recall_23: 0.5409
Epoch 25/30
500/500 [==============================] - 11s 22ms/step - loss: 0.2277 -
accuracy: 0.8973 - recall_23: 0.5744 - val_loss: 0.2761 - val_accuracy: 0.8757 -
val_recall_23: 0.4943
Epoch 26/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2256 -
accuracy: 0.8986 - recall_23: 0.5727 - val_loss: 0.2785 - val_accuracy: 0.8766 -
val_recall_23: 0.5305
Epoch 27/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2256 -
accuracy: 0.8989 - recall_23: 0.5819 - val_loss: 0.2774 - val_accuracy: 0.8772 -
val_recall_23: 0.5352
Epoch 28/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2239 -
accuracy: 0.8994 - recall_23: 0.5812 - val_loss: 0.2774 - val_accuracy: 0.8771 -
val_recall_23: 0.5341
Epoch 29/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2243 -
accuracy: 0.8985 - recall_23: 0.5755 - val_loss: 0.2797 - val_accuracy: 0.8763 -
val_recall_23: 0.5210
Epoch 30/30
500/500 [==============================] - 10s 20ms/step - loss: 0.2246 -
accuracy: 0.8996 - recall_23: 0.5809 - val_loss: 0.2787 - val_accuracy: 0.8774 -
val_recall_23: 0.5296
INFO:tensorflow:Assets written to: /tmp/tmphm_t86a0/model/data/model/assets
2556/2556 [==============================] - 4s 2ms/step - loss: 0.3183 -
accuracy: 0.8795 - recall_23: 0.6563
```

Model accuracy for the displayed run has an accuracy for both training and validation between 85% and 90%. Validation loss is increasing, and validation recall appears to be oscillating around 0.53 (range of 0.50-0.55).

[70]:
```
plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
plot_graphs(history, "recall_23")
```


1.0
0.8
0.6
0.4
0.2
0.0
accuracy
0
5
10
15
20
25
30
Epochs
accuracy
val_accuracy


0.32
0.30
0.28
0.26
0.24
0.22
loss
0
5
10
15
20
25
30
Epochs
loss
val_loss



Experiment 5: Stacked LSTM with embedding

[72]:
```
for s, e, sv, cw in zip(steps, epochs, steps_val, class_weight):
  with mlflow.start_run(run_name='WeightTied+StackedLSTM'):
    layer1 = layers.Dense(25, activation='relu', input_shape=(None, 25))
    layer2 = TiedtDense(25, layer1, activation='relu')

    model6 = Sequential()
    model6.add(layer1)
    model6.add(layers.LSTM(25, dropout=0.2,recurrent_dropout=0.2,
                           input_shape=(None, 25), return_sequences=True))
    model6.add(layers.LSTM(25, dropout=0.2,recurrent_dropout=0.2,
                         input_shape=(None, 25)))
    model6.add(layer2)
    model6.add(layers.Dense(1, activation='sigmoid'))

    display(model6.summary())

    model6.compile(optimizer=RMSprop(), loss='binary_crossentropy',
                   metrics=['accuracy', tf.keras.metrics.Recall()])
    history = model6.fit(trainX, trainY,
                         steps_per_epoch=s,
                         epochs=e,
                         validation_data=(valX, valY),
                         validation_steps=sv,
```

```
                        # class_weight=cw
    )
    # history = model6.fit(train_gen,
    #                      steps_per_epoch=s,
    #                      epochs=e,
    #                      validation_data=val_gen,
    #                      validation_steps=sv,
    #                      class_weight=cw
    #                      )
    mlflow.keras.log_model(model6, "WeightTied+StackedLSTM", save_format='tf')
    mlflow.log_params(history.params)
    scores = model6.evaluate(testX, testY)
    mlflow.log_metrics({k: v for k, v in zip(
        ['loss', 'accuracy', 'recall'], scores)})
```

```
Model: "sequential_22"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_26 (Dense)             (None, None, 25)          650
_________________________________________________________________
lstm_27 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
lstm_28 (LSTM)               (None, 25)                5100
_________________________________________________________________
tiedt_dense_6 (TiedtDense)   (None, 25)                1300
_________________________________________________________________
dense_27 (Dense)             (None, 1)                 26
=================================================================
Total params: 11,526
Trainable params: 10,901
Non-trainable params: 625
_________________________________________________________________
None


Epoch 1/10
100/100 [==============================] - 14s 71ms/step - loss: 0.5244 -
accuracy: 0.8319 - recall_25: 0.1086 - val_loss: 0.2619 - val_accuracy: 0.8747 -
val_recall_25: 0.6044
Epoch 2/10
100/100 [==============================] - 6s 63ms/step - loss: 0.2763 -
accuracy: 0.8714 - recall_25: 0.5561 - val_loss: 0.2503 - val_accuracy: 0.8760 -
val_recall_25: 0.5508
Epoch 3/10
100/100 [==============================] - 6s 62ms/step - loss: 0.2668 -
accuracy: 0.8745 - recall_25: 0.5035 - val_loss: 0.2487 - val_accuracy: 0.8800 -
```

```
val_recall_25: 0.5349
Epoch 4/10
100/100 [==============================] - 6s 63ms/step - loss: 0.2611 -
accuracy: 0.8784 - recall_25: 0.4997 - val_loss: 0.2495 - val_accuracy: 0.8808 -
val_recall_25: 0.5581
Epoch 5/10
100/100 [==============================] - 6s 64ms/step - loss: 0.2585 -
accuracy: 0.8802 - recall_25: 0.5159 - val_loss: 0.2475 - val_accuracy: 0.8824 -
val_recall_25: 0.5235
Epoch 6/10
100/100 [==============================] - 6s 64ms/step - loss: 0.2542 -
accuracy: 0.8823 - recall_25: 0.5306 - val_loss: 0.2480 - val_accuracy: 0.8829 -
val_recall_25: 0.5223
Epoch 7/10
100/100 [==============================] - 6s 65ms/step - loss: 0.2524 -
accuracy: 0.8828 - recall_25: 0.5285 - val_loss: 0.2497 - val_accuracy: 0.8817 -
val_recall_25: 0.5582
Epoch 8/10
100/100 [==============================] - 7s 66ms/step - loss: 0.2480 -
accuracy: 0.8837 - recall_25: 0.5380 - val_loss: 0.2489 - val_accuracy: 0.8822 -
val_recall_25: 0.5231
Epoch 9/10
100/100 [==============================] - 7s 72ms/step - loss: 0.2465 -
accuracy: 0.8854 - recall_25: 0.5371 - val_loss: 0.2474 - val_accuracy: 0.8832 -
val_recall_25: 0.5073
Epoch 10/10
100/100 [==============================] - 7s 67ms/step - loss: 0.2429 -
accuracy: 0.8874 - recall_25: 0.5503 - val_loss: 0.2484 - val_accuracy: 0.8836 -
val_recall_25: 0.5443
INFO:tensorflow:Assets written to: /tmp/tmp0slumh47/model/data/model/assets
2556/2556 [==============================] - 4s 2ms/step - loss: 0.2765 -
accuracy: 0.8761 - recall_25: 0.7461
Model: "sequential_23"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_28 (Dense)             (None, None, 25)          650
_________________________________________________________________
lstm_29 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
lstm_30 (LSTM)               (None, 25)                5100
_________________________________________________________________
tiedt_dense_7 (TiedtDense)   (None, 25)                1300
_________________________________________________________________
dense_29 (Dense)             (None, 1)                 26
=================================================================
Total params: 11,526
Trainable params: 10,901
```

```
Non-trainable params: 625
_________________________________________________________________
None


Epoch 1/20
300/300 [==============================] - 14s 25ms/step - loss: 0.4089 -
accuracy: 0.8450 - recall_26: 0.1892 - val_loss: 0.2493 - val_accuracy: 0.8820 -
val_recall_26: 0.5431
Epoch 2/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2609 -
accuracy: 0.8780 - recall_26: 0.4998 - val_loss: 0.2465 - val_accuracy: 0.8858 -
val_recall_26: 0.5378
Epoch 3/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2526 -
accuracy: 0.8823 - recall_26: 0.5133 - val_loss: 0.2496 - val_accuracy: 0.8849 -
val_recall_26: 0.5373
Epoch 4/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2460 -
accuracy: 0.8867 - recall_26: 0.5396 - val_loss: 0.2574 - val_accuracy: 0.8840 -
val_recall_26: 0.4890
Epoch 5/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2411 -
accuracy: 0.8906 - recall_26: 0.5565 - val_loss: 0.2581 - val_accuracy: 0.8830 -
val_recall_26: 0.5245
Epoch 6/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2369 -
accuracy: 0.8926 - recall_26: 0.5721 - val_loss: 0.2603 - val_accuracy: 0.8834 -
val_recall_26: 0.5175
Epoch 7/20
300/300 [==============================] - 7s 23ms/step - loss: 0.2343 -
accuracy: 0.8944 - recall_26: 0.5763 - val_loss: 0.2629 - val_accuracy: 0.8819 -
val_recall_26: 0.4905
Epoch 8/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2315 -
accuracy: 0.8956 - recall_26: 0.5855 - val_loss: 0.2642 - val_accuracy: 0.8808 -
val_recall_26: 0.5045
Epoch 9/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2284 -
accuracy: 0.8979 - recall_26: 0.5940 - val_loss: 0.2657 - val_accuracy: 0.8817 -
val_recall_26: 0.5108
Epoch 10/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2249 -
accuracy: 0.8992 - recall_26: 0.5998 - val_loss: 0.2680 - val_accuracy: 0.8813 -
val_recall_26: 0.5130
Epoch 11/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2233 -
accuracy: 0.9008 - recall_26: 0.6108 - val_loss: 0.2678 - val_accuracy: 0.8797 -
```

```
val_recall_26: 0.5296
Epoch 12/20
300/300 [==============================] - 7s 23ms/step - loss: 0.2210 -
accuracy: 0.9019 - recall_26: 0.6183 - val_loss: 0.2716 - val_accuracy: 0.8786 -
val_recall_26: 0.4781
Epoch 13/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2192 -
accuracy: 0.9029 - recall_26: 0.6215 - val_loss: 0.2722 - val_accuracy: 0.8770 -
val_recall_26: 0.5160
Epoch 14/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2179 -
accuracy: 0.9036 - recall_26: 0.6327 - val_loss: 0.2745 - val_accuracy: 0.8760 -
val_recall_26: 0.4917
Epoch 15/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2152 -
accuracy: 0.9055 - recall_26: 0.6350 - val_loss: 0.2775 - val_accuracy: 0.8771 -
val_recall_26: 0.5153
Epoch 16/20
300/300 [==============================] - 7s 24ms/step - loss: 0.2143 -
accuracy: 0.9055 - recall_26: 0.6404 - val_loss: 0.2793 - val_accuracy: 0.8744 -
val_recall_26: 0.5565
Epoch 17/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2108 -
accuracy: 0.9074 - recall_26: 0.6487 - val_loss: 0.2815 - val_accuracy: 0.8757 -
val_recall_26: 0.4849
Epoch 18/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2117 -
accuracy: 0.9071 - recall_26: 0.6493 - val_loss: 0.2840 - val_accuracy: 0.8742 -
val_recall_26: 0.4963
Epoch 19/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2095 -
accuracy: 0.9074 - recall_26: 0.6538 - val_loss: 0.2861 - val_accuracy: 0.8748 -
val_recall_26: 0.5027
Epoch 20/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2079 -
accuracy: 0.9090 - recall_26: 0.6580 - val_loss: 0.2886 - val_accuracy: 0.8733 -
val_recall_26: 0.5143
INFO:tensorflow:Assets written to: /tmp/tmpka5yannl/model/data/model/assets
2556/2556 [==============================] - 4s 2ms/step - loss: 0.3082 -
accuracy: 0.8653 - recall_26: 0.6483
Model: "sequential_24"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_30 (Dense)             (None, None, 25)          650
_________________________________________________________________
lstm_31 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
```

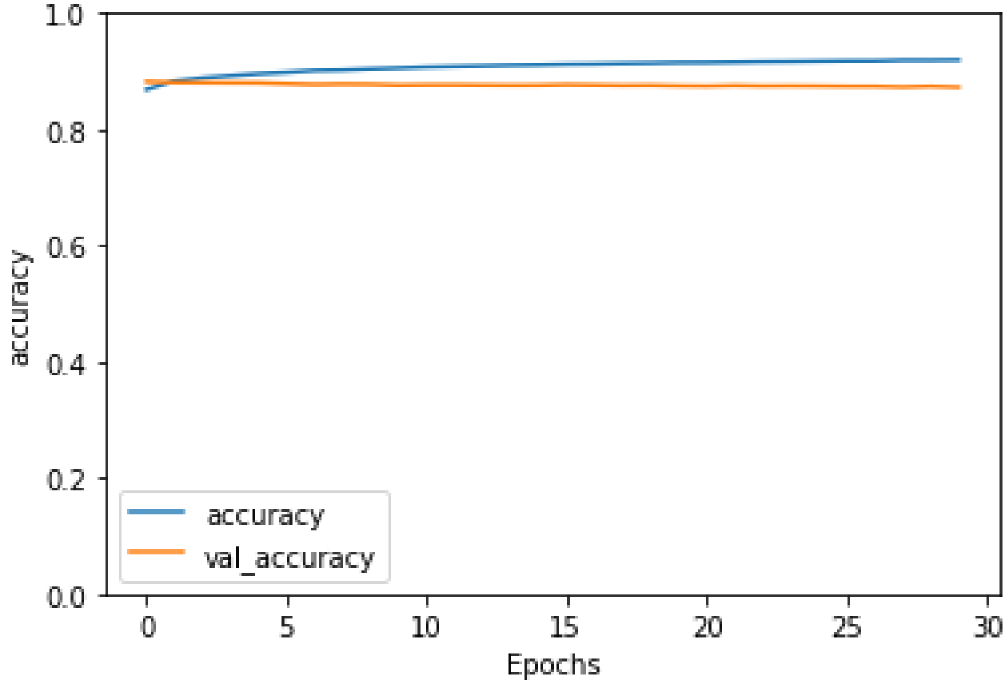
```
lstm_32 (LSTM)               (None, 25)                5100
_________________________________________________________________
tiedt_dense_8 (TiedtDense)   (None, 25)                1300
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 26
=================================================================
Total params: 11,526
Trainable params: 10,901
Non-trainable params: 625
_________________________________________________________________
None


Epoch 1/30
500/500 [==============================] - 16s 17ms/step - loss: 0.4144 -
accuracy: 0.8492 - recall_27: 0.3404 - val_loss: 0.2477 - val_accuracy: 0.8824 -
val_recall_27: 0.4738
Epoch 2/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2565 -
accuracy: 0.8816 - recall_27: 0.4991 - val_loss: 0.2551 - val_accuracy: 0.8809 -
val_recall_27: 0.5461
Epoch 3/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2442 -
accuracy: 0.8889 - recall_27: 0.5312 - val_loss: 0.2602 - val_accuracy: 0.8806 -
val_recall_27: 0.4764
Epoch 4/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2409 -
accuracy: 0.8918 - recall_27: 0.5548 - val_loss: 0.2646 - val_accuracy: 0.8804 -
val_recall_27: 0.4775
Epoch 5/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2352 -
accuracy: 0.8947 - recall_27: 0.5682 - val_loss: 0.2640 - val_accuracy: 0.8802 -
val_recall_27: 0.4660
Epoch 6/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2313 -
accuracy: 0.8978 - recall_27: 0.5787 - val_loss: 0.2674 - val_accuracy: 0.8787 -
val_recall_27: 0.5392
Epoch 7/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2269 -
accuracy: 0.9001 - recall_27: 0.5947 - val_loss: 0.2683 - val_accuracy: 0.8774 -
val_recall_27: 0.5030
Epoch 8/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2235 -
accuracy: 0.9019 - recall_27: 0.6000 - val_loss: 0.2673 - val_accuracy: 0.8779 -
val_recall_27: 0.4339
Epoch 9/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2215 -
accuracy: 0.9035 - recall_27: 0.6012 - val_loss: 0.2698 - val_accuracy: 0.8776 -
```

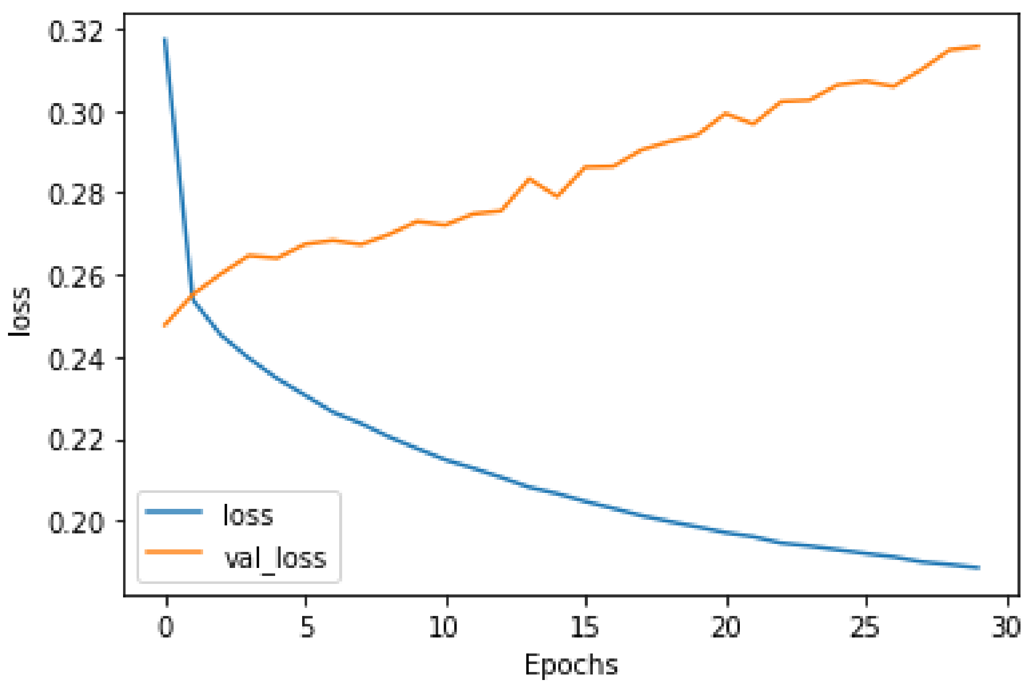
```
val_recall_27: 0.4666
Epoch 10/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2192 -
accuracy: 0.9050 - recall_27: 0.6093 - val_loss: 0.2729 - val_accuracy: 0.8763 -
val_recall_27: 0.4790
Epoch 11/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2160 -
accuracy: 0.9062 - recall_27: 0.6165 - val_loss: 0.2721 - val_accuracy: 0.8766 -
val_recall_27: 0.4850
Epoch 12/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2124 -
accuracy: 0.9078 - recall_27: 0.6227 - val_loss: 0.2748 - val_accuracy: 0.8766 -
val_recall_27: 0.4574
Epoch 13/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2117 -
accuracy: 0.9086 - recall_27: 0.6240 - val_loss: 0.2755 - val_accuracy: 0.8762 -
val_recall_27: 0.4503
Epoch 14/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2087 -
accuracy: 0.9100 - recall_27: 0.6304 - val_loss: 0.2833 - val_accuracy: 0.8760 -
val_recall_27: 0.4861
Epoch 15/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2062 -
accuracy: 0.9111 - recall_27: 0.6370 - val_loss: 0.2790 - val_accuracy: 0.8761 -
val_recall_27: 0.4938
Epoch 16/30
500/500 [==============================] - 8s 15ms/step - loss: 0.2062 -
accuracy: 0.9110 - recall_27: 0.6384 - val_loss: 0.2862 - val_accuracy: 0.8770 -
val_recall_27: 0.5134
Epoch 17/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2031 -
accuracy: 0.9120 - recall_27: 0.6426 - val_loss: 0.2864 - val_accuracy: 0.8764 -
val_recall_27: 0.4876
Epoch 18/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2010 -
accuracy: 0.9128 - recall_27: 0.6425 - val_loss: 0.2903 - val_accuracy: 0.8757 -
val_recall_27: 0.4833
Epoch 19/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1992 -
accuracy: 0.9145 - recall_27: 0.6500 - val_loss: 0.2924 - val_accuracy: 0.8760 -
val_recall_27: 0.5059
Epoch 20/30
500/500 [==============================] - 8s 17ms/step - loss: 0.1991 -
accuracy: 0.9141 - recall_27: 0.6540 - val_loss: 0.2940 - val_accuracy: 0.8751 -
val_recall_27: 0.4684
Epoch 21/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1966 -
accuracy: 0.9149 - recall_27: 0.6484 - val_loss: 0.2992 - val_accuracy: 0.8747 -
```

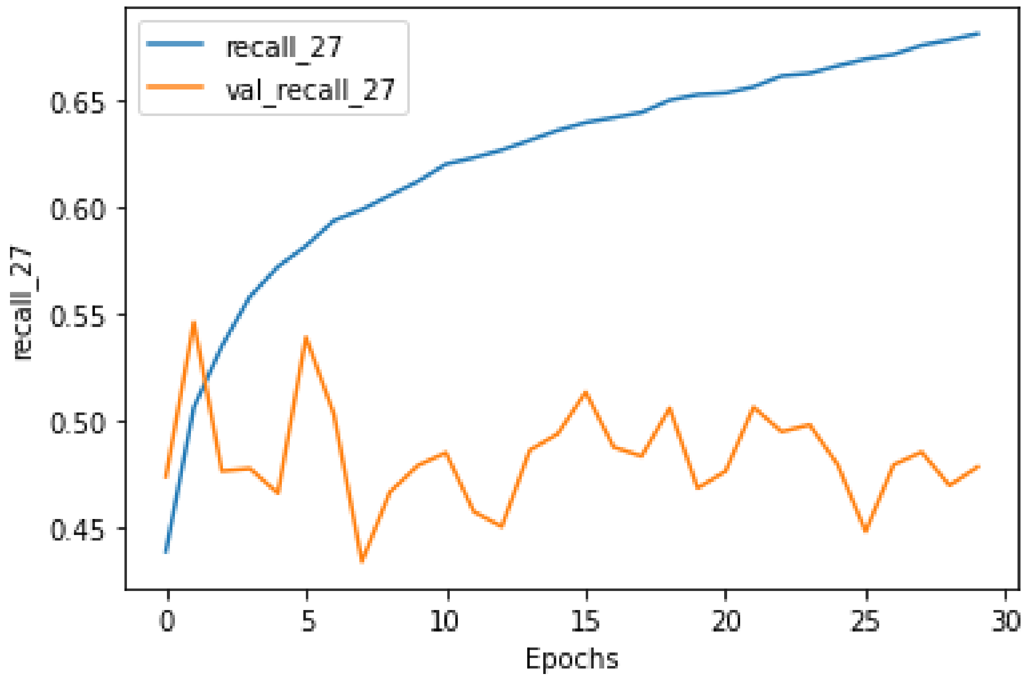
```
val_recall_27: 0.4763
Epoch 22/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1951 -
accuracy: 0.9161 - recall_27: 0.6598 - val_loss: 0.2967 - val_accuracy: 0.8756 -
val_recall_27: 0.5064
Epoch 23/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1937 -
accuracy: 0.9166 - recall_27: 0.6634 - val_loss: 0.3022 - val_accuracy: 0.8748 -
val_recall_27: 0.4948
Epoch 24/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1932 -
accuracy: 0.9170 - recall_27: 0.6654 - val_loss: 0.3025 - val_accuracy: 0.8750 -
val_recall_27: 0.4981
Epoch 25/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1924 -
accuracy: 0.9175 - recall_27: 0.6688 - val_loss: 0.3062 - val_accuracy: 0.8747 -
val_recall_27: 0.4794
Epoch 26/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1918 -
accuracy: 0.9173 - recall_27: 0.6674 - val_loss: 0.3071 - val_accuracy: 0.8743 -
val_recall_27: 0.4481
Epoch 27/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1924 -
accuracy: 0.9174 - recall_27: 0.6686 - val_loss: 0.3059 - val_accuracy: 0.8743 -
val_recall_27: 0.4793
Epoch 28/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1903 -
accuracy: 0.9184 - recall_27: 0.6761 - val_loss: 0.3100 - val_accuracy: 0.8731 -
val_recall_27: 0.4853
Epoch 29/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1903 -
accuracy: 0.9187 - recall_27: 0.6768 - val_loss: 0.3147 - val_accuracy: 0.8742 -
val_recall_27: 0.4697
Epoch 30/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1902 -
accuracy: 0.9189 - recall_27: 0.6792 - val_loss: 0.3156 - val_accuracy: 0.8728 -
val_recall_27: 0.4782
INFO:tensorflow:Assets written to: /tmp/tmp2pemkeka/model/data/model/assets
2556/2556 [==============================] - 4s 2ms/step - loss: 0.3769 -
accuracy: 0.8614 - recall_27: 0.5873
```

Model accuracy for the displayed run has an accuracy for both training and validation between 85% and 90%. Validation loss is increasing, and validation recall is appears to be oscillating around 0.48.

[73]:
```
plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
plot_graphs(history, "recall_27")
```


1.0
0.8
0.6
0.4
0.2
0.0
accuracy
accuracy
val_accuracy
0
5
10
15
20
25
30
Epochs


0.32
0.30
0.28
0.26
0.24
0.22
0.20
loss
loss
val_loss
0
5
10
15
20
25
30
Epochs



Experiment 6: Weight-tied stacked LSTM with embedding

[75]:
```
for s, e, sv, cw in zip(steps, epochs, steps_val, class_weight):
  with mlflow.start_run(run_name='WeightTied+StackedLSTM'):
    layer1 = layers.Dense(26, activation='relu', input_shape=(None, 26))
    layer2 = TiedtDense(26, layer1, activation='relu')
    layer3 = layers.LSTM(26, dropout=0.2,recurrent_dropout=0.2,
                     input_shape=(None, 26), return_sequences=True)
    layer4 = TiedtLSTM(26, layer3, dropout=0.2,recurrent_dropout=0.2,
                       input_shape=(None, 26), return_sequences=True)

    model7 = Sequential()
    model7.add(layer1)
    model7.add(layer3)
    model7.add(layer4)
    model7.add(layer2)
    model7.add(layers.Dense(1, activation='sigmoid'))

    display(model7.summary())

    model7.compile(optimizer=RMSprop(), loss='binary_crossentropy',
                  metrics=['accuracy', tf.keras.metrics.Recall()])
    # history = model7.fit(trainX, trainY,
    #                       steps_per_epoch=s,
    #                       epochs=e,
```

```
    #                       validation_data=(valX, valY),
    #                       validation_steps=sv,
    #                       # class_weight=cw
    # )
    history = model7.fit(train_gen,
                         steps_per_epoch=s,
                         epochs=e,
                         validation_data=val_gen,
                         validation_steps=sv,
                         class_weight=cw
                         )
    mlflow.keras.log_model(model7, "WeightTiedAll", save_format='tf')
    mlflow.log_params(history.params)
    scores = model7.evaluate(test_gen)
    mlflow.log_metrics({k: v for k, v in zip(
        ['loss', 'accuracy', 'recall'], scores)})
```

```
Model: "sequential_25"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_32 (Dense)             (None, None, 25)          650
_________________________________________________________________
lstm_33 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
tiedt_lstm_4 (TiedtLSTM)     (None, None, 25)          12725
_________________________________________________________________
tiedt_dense_9 (TiedtDense)   (None, None, 25)          1300
_________________________________________________________________
dense_33 (Dense)             (None, None, 1)           26
=================================================================
Total params: 14,051
Trainable params: 10,926
Non-trainable params: 3,125
_________________________________________________________________
None


Epoch 1/10
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_4/Variable:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_4/Variable:0'] when minimizing the loss.
100/100 [==============================] - 13s 70ms/step - loss: 0.5574 -
accuracy: 0.8172 - recall_28: 0.0258 - val_loss: 0.2670 - val_accuracy: 0.8855 -
val_recall_28: 0.4674
Epoch 2/10
```

```
100/100 [==============================] - 6s 61ms/step - loss: 0.2816 -
accuracy: 0.8694 - recall_28: 0.4986 - val_loss: 0.2512 - val_accuracy: 0.8833 -
val_recall_28: 0.5976
Epoch 3/10
100/100 [==============================] - 6s 61ms/step - loss: 0.2673 -
accuracy: 0.8734 - recall_28: 0.5399 - val_loss: 0.2478 - val_accuracy: 0.8852 -
val_recall_28: 0.5742
Epoch 4/10
100/100 [==============================] - 6s 61ms/step - loss: 0.2609 -
accuracy: 0.8765 - recall_28: 0.5198 - val_loss: 0.2457 - val_accuracy: 0.8862 -
val_recall_28: 0.5312
Epoch 5/10
100/100 [==============================] - 6s 61ms/step - loss: 0.2563 -
accuracy: 0.8803 - recall_28: 0.5269 - val_loss: 0.2479 - val_accuracy: 0.8843 -
val_recall_28: 0.5414
Epoch 6/10
100/100 [==============================] - 6s 61ms/step - loss: 0.2532 -
accuracy: 0.8836 - recall_28: 0.5422 - val_loss: 0.2485 - val_accuracy: 0.8838 -
val_recall_28: 0.5612
Epoch 7/10
100/100 [==============================] - 6s 62ms/step - loss: 0.2508 -
accuracy: 0.8842 - recall_28: 0.5595 - val_loss: 0.2490 - val_accuracy: 0.8830 -
val_recall_28: 0.5184
Epoch 8/10
100/100 [==============================] - 6s 61ms/step - loss: 0.2467 -
accuracy: 0.8864 - recall_28: 0.5683 - val_loss: 0.2503 - val_accuracy: 0.8832 -
val_recall_28: 0.5270
Epoch 9/10
100/100 [==============================] - 6s 63ms/step - loss: 0.2449 -
accuracy: 0.8876 - recall_28: 0.5655 - val_loss: 0.2501 - val_accuracy: 0.8829 -
val_recall_28: 0.4846
Epoch 10/10
100/100 [==============================] - 6s 64ms/step - loss: 0.2423 -
accuracy: 0.8885 - recall_28: 0.5677 - val_loss: 0.2524 - val_accuracy: 0.8819 -
val_recall_28: 0.5217
INFO:tensorflow:Assets written to: /tmp/tmppn46shzw/model/data/model/assets
2556/2556 [==============================] - 5s 2ms/step - loss: 0.2729 -
accuracy: 0.8731 - recall_28: 0.6924
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_34 (Dense)             (None, None, 25)          650
_________________________________________________________________
lstm_34 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
tiedt_lstm_5 (TiedtLSTM)     (None, None, 25)          12725
_________________________________________________________________
```

```
tiedt_dense_10 (TiedtDense)  (None, None, 25)          1300
_________________________________________________________________
dense_35 (Dense)             (None, None, 1)           26
=================================================================
Total params: 14,051
Trainable params: 10,926
Non-trainable params: 3,125
_________________________________________________________________
None


Epoch 1/20
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_5/Variable:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_5/Variable:0'] when minimizing the loss.
300/300 [==============================] - 14s 25ms/step - loss: 0.4052 -
accuracy: 0.8474 - recall_29: 0.1494 - val_loss: 0.2533 - val_accuracy: 0.8773 -
val_recall_29: 0.5679
Epoch 2/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2631 -
accuracy: 0.8777 - recall_29: 0.5211 - val_loss: 0.2457 - val_accuracy: 0.8858 -
val_recall_29: 0.5420
Epoch 3/20
300/300 [==============================] - 6s 22ms/step - loss: 0.2529 -
accuracy: 0.8850 - recall_29: 0.5397 - val_loss: 0.2493 - val_accuracy: 0.8840 -
val_recall_29: 0.5199
Epoch 4/20
300/300 [==============================] - 6s 22ms/step - loss: 0.2473 -
accuracy: 0.8887 - recall_29: 0.5564 - val_loss: 0.2523 - val_accuracy: 0.8808 -
val_recall_29: 0.4568
Epoch 5/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2441 -
accuracy: 0.8894 - recall_29: 0.5573 - val_loss: 0.2542 - val_accuracy: 0.8812 -
val_recall_29: 0.4607
Epoch 6/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2402 -
accuracy: 0.8918 - recall_29: 0.5735 - val_loss: 0.2553 - val_accuracy: 0.8814 -
val_recall_29: 0.4863
Epoch 7/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2378 -
accuracy: 0.8934 - recall_29: 0.5825 - val_loss: 0.2573 - val_accuracy: 0.8825 -
val_recall_29: 0.5084
Epoch 8/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2317 -
accuracy: 0.8962 - recall_29: 0.5950 - val_loss: 0.2604 - val_accuracy: 0.8801 -
val_recall_29: 0.4498
Epoch 9/20
```

```
300/300 [==============================] - 7s 22ms/step - loss: 0.2322 -
accuracy: 0.8962 - recall_29: 0.5953 - val_loss: 0.2616 - val_accuracy: 0.8802 -
val_recall_29: 0.4849
Epoch 10/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2283 -
accuracy: 0.8989 - recall_29: 0.6064 - val_loss: 0.2623 - val_accuracy: 0.8792 -
val_recall_29: 0.4868
Epoch 11/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2246 -
accuracy: 0.9007 - recall_29: 0.6170 - val_loss: 0.2646 - val_accuracy: 0.8795 -
val_recall_29: 0.4830
Epoch 12/20
300/300 [==============================] - 7s 23ms/step - loss: 0.2225 -
accuracy: 0.9023 - recall_29: 0.6195 - val_loss: 0.2672 - val_accuracy: 0.8791 -
val_recall_29: 0.4612
Epoch 13/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2230 -
accuracy: 0.9012 - recall_29: 0.6212 - val_loss: 0.2697 - val_accuracy: 0.8791 -
val_recall_29: 0.4502
Epoch 14/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2199 -
accuracy: 0.9040 - recall_29: 0.6288 - val_loss: 0.2698 - val_accuracy: 0.8791 -
val_recall_29: 0.4879
Epoch 15/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2178 -
accuracy: 0.9050 - recall_29: 0.6356 - val_loss: 0.2700 - val_accuracy: 0.8798 -
val_recall_29: 0.5063
Epoch 16/20
300/300 [==============================] - 7s 23ms/step - loss: 0.2155 -
accuracy: 0.9070 - recall_29: 0.6417 - val_loss: 0.2788 - val_accuracy: 0.8767 -
val_recall_29: 0.4387
Epoch 17/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2143 -
accuracy: 0.9078 - recall_29: 0.6425 - val_loss: 0.2752 - val_accuracy: 0.8796 -
val_recall_29: 0.4693
Epoch 18/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2127 -
accuracy: 0.9086 - recall_29: 0.6494 - val_loss: 0.2745 - val_accuracy: 0.8785 -
val_recall_29: 0.4879
Epoch 19/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2109 -
accuracy: 0.9093 - recall_29: 0.6500 - val_loss: 0.2763 - val_accuracy: 0.8800 -
val_recall_29: 0.4735
Epoch 20/20
300/300 [==============================] - 7s 22ms/step - loss: 0.2091 -
accuracy: 0.9093 - recall_29: 0.6554 - val_loss: 0.2791 - val_accuracy: 0.8782 -
val_recall_29: 0.4529
INFO:tensorflow:Assets written to: /tmp/tmp_nje7pds/model/data/model/assets
```

```
2556/2556 [==============================] - 4s 2ms/step - loss: 0.2784 -
accuracy: 0.8860 - recall_29: 0.6677
Model: "sequential_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_36 (Dense)             (None, None, 25)          650
_________________________________________________________________
lstm_35 (LSTM)               (None, None, 25)          5100
_________________________________________________________________
tiedt_lstm_6 (TiedtLSTM)     (None, None, 25)          12725
_________________________________________________________________
tiedt_dense_11 (TiedtDense)  (None, None, 25)          1300
_________________________________________________________________
dense_37 (Dense)             (None, None, 1)           26
=================================================================
Total params: 14,051
Trainable params: 10,926
Non-trainable params: 3,125
_________________________________________________________________
None


Epoch 1/30
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_6/Variable:0'] when minimizing the loss.
WARNING:tensorflow:Gradients do not exist for variables
['tiedt_lstm_6/Variable:0'] when minimizing the loss.
500/500 [==============================] - 15s 17ms/step - loss: 0.3842 -
accuracy: 0.8560 - recall_30: 0.3150 - val_loss: 0.2507 - val_accuracy: 0.8813 -
val_recall_30: 0.5514
Epoch 2/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2578 -
accuracy: 0.8794 - recall_30: 0.5012 - val_loss: 0.2494 - val_accuracy: 0.8860 -
val_recall_30: 0.5224
Epoch 3/30
500/500 [==============================] - 8s 15ms/step - loss: 0.2487 -
accuracy: 0.8845 - recall_30: 0.5436 - val_loss: 0.2544 - val_accuracy: 0.8834 -
val_recall_30: 0.5241
Epoch 4/30
500/500 [==============================] - 8s 15ms/step - loss: 0.2405 -
accuracy: 0.8886 - recall_30: 0.5759 - val_loss: 0.2559 - val_accuracy: 0.8822 -
val_recall_30: 0.4494
Epoch 5/30
500/500 [==============================] - 8s 15ms/step - loss: 0.2372 -
accuracy: 0.8918 - recall_30: 0.5885 - val_loss: 0.2583 - val_accuracy: 0.8797 -
val_recall_30: 0.5329
Epoch 6/30
```

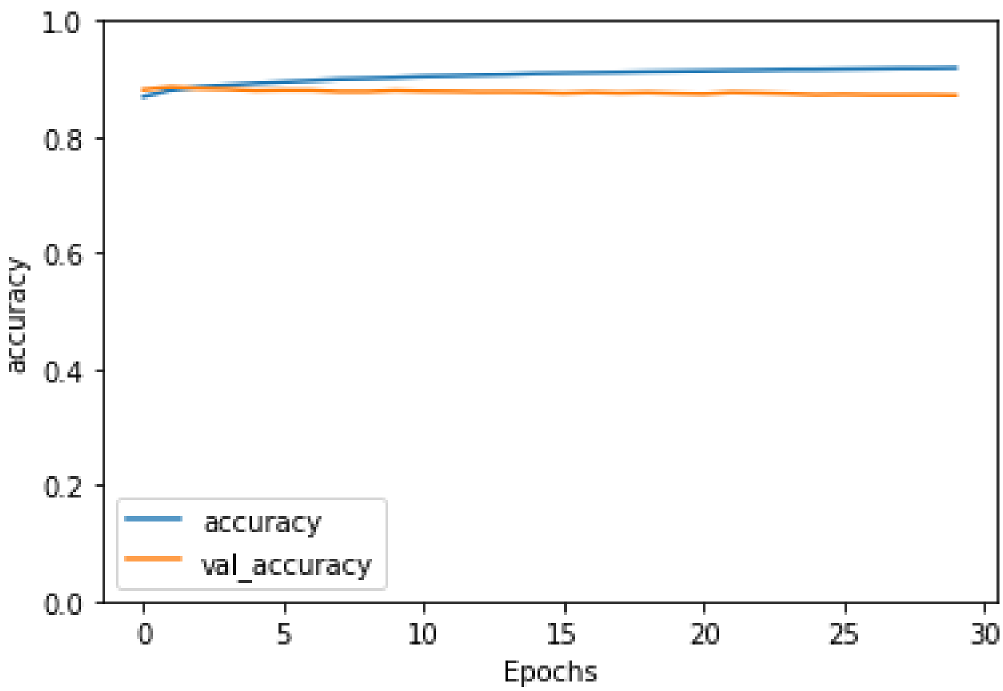
```
500/500 [==============================] - 8s 16ms/step - loss: 0.2326 -
accuracy: 0.8939 - recall_30: 0.6030 - val_loss: 0.2594 - val_accuracy: 0.8804 -
val_recall_30: 0.5411
Epoch 7/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2287 -
accuracy: 0.8959 - recall_30: 0.6113 - val_loss: 0.2597 - val_accuracy: 0.8802 -
val_recall_30: 0.5073
Epoch 8/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2246 -
accuracy: 0.8989 - recall_30: 0.6209 - val_loss: 0.2646 - val_accuracy: 0.8777 -
val_recall_30: 0.5054
Epoch 9/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2227 -
accuracy: 0.8997 - recall_30: 0.6215 - val_loss: 0.2671 - val_accuracy: 0.8777 -
val_recall_30: 0.4829
Epoch 10/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2183 -
accuracy: 0.9017 - recall_30: 0.6284 - val_loss: 0.2705 - val_accuracy: 0.8798 -
val_recall_30: 0.5062
Epoch 11/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2170 -
accuracy: 0.9033 - recall_30: 0.6321 - val_loss: 0.2701 - val_accuracy: 0.8785 -
val_recall_30: 0.5319
Epoch 12/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2144 -
accuracy: 0.9048 - recall_30: 0.6401 - val_loss: 0.2726 - val_accuracy: 0.8780 -
val_recall_30: 0.4978
Epoch 13/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2117 -
accuracy: 0.9062 - recall_30: 0.6422 - val_loss: 0.2797 - val_accuracy: 0.8771 -
val_recall_30: 0.4726
Epoch 14/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2102 -
accuracy: 0.9069 - recall_30: 0.6442 - val_loss: 0.2770 - val_accuracy: 0.8772 -
val_recall_30: 0.5442
Epoch 15/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2082 -
accuracy: 0.9080 - recall_30: 0.6545 - val_loss: 0.2798 - val_accuracy: 0.8766 -
val_recall_30: 0.4844
Epoch 16/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2055 -
accuracy: 0.9096 - recall_30: 0.6570 - val_loss: 0.2838 - val_accuracy: 0.8751 -
val_recall_30: 0.4842
Epoch 17/30
500/500 [==============================] - 8s 16ms/step - loss: 0.2050 -
accuracy: 0.9100 - recall_30: 0.6617 - val_loss: 0.2862 - val_accuracy: 0.8767 -
val_recall_30: 0.4918
Epoch 18/30
```

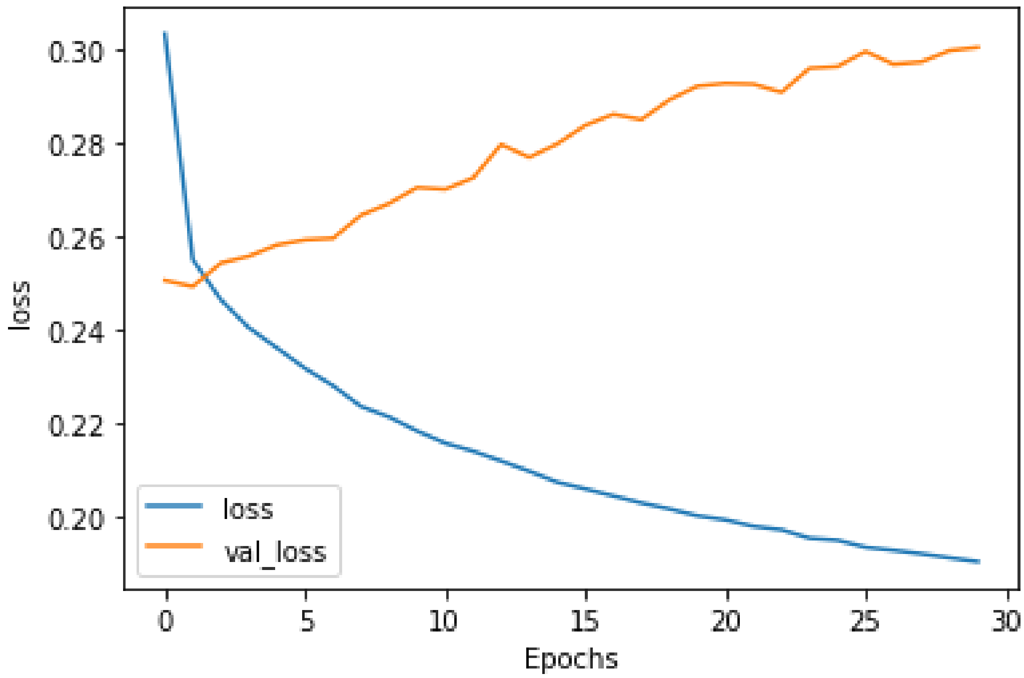
```
500/500 [==============================] - 8s 16ms/step - loss: 0.2036 -
accuracy: 0.9108 - recall_30: 0.6635 - val_loss: 0.2850 - val_accuracy: 0.8754 -
val_recall_30: 0.4842
Epoch 19/30
500/500 [==============================] - 8s 17ms/step - loss: 0.2026 -
accuracy: 0.9117 - recall_30: 0.6663 - val_loss: 0.2893 - val_accuracy: 0.8762 -
val_recall_30: 0.4985
Epoch 20/30
500/500 [==============================] - 8s 17ms/step - loss: 0.2006 -
accuracy: 0.9127 - recall_30: 0.6742 - val_loss: 0.2922 - val_accuracy: 0.8749 -
val_recall_30: 0.4734
Epoch 21/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1993 -
accuracy: 0.9128 - recall_30: 0.6734 - val_loss: 0.2928 - val_accuracy: 0.8738 -
val_recall_30: 0.4834
Epoch 22/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1980 -
accuracy: 0.9133 - recall_30: 0.6764 - val_loss: 0.2926 - val_accuracy: 0.8765 -
val_recall_30: 0.4558
Epoch 23/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1976 -
accuracy: 0.9143 - recall_30: 0.6818 - val_loss: 0.2909 - val_accuracy: 0.8756 -
val_recall_30: 0.4938
Epoch 24/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1966 -
accuracy: 0.9157 - recall_30: 0.6865 - val_loss: 0.2960 - val_accuracy: 0.8743 -
val_recall_30: 0.5000
Epoch 25/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1971 -
accuracy: 0.9148 - recall_30: 0.6877 - val_loss: 0.2964 - val_accuracy: 0.8725 -
val_recall_30: 0.4791
Epoch 26/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1951 -
accuracy: 0.9164 - recall_30: 0.6902 - val_loss: 0.2996 - val_accuracy: 0.8732 -
val_recall_30: 0.4741
Epoch 27/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1926 -
accuracy: 0.9175 - recall_30: 0.6946 - val_loss: 0.2969 - val_accuracy: 0.8724 -
val_recall_30: 0.4875
Epoch 28/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1915 -
accuracy: 0.9175 - recall_30: 0.6949 - val_loss: 0.2974 - val_accuracy: 0.8723 -
val_recall_30: 0.4869
Epoch 29/30
500/500 [==============================] - 8s 16ms/step - loss: 0.1906 -
accuracy: 0.9178 - recall_30: 0.6992 - val_loss: 0.2998 - val_accuracy: 0.8724 -
val_recall_30: 0.4753
Epoch 30/30
```

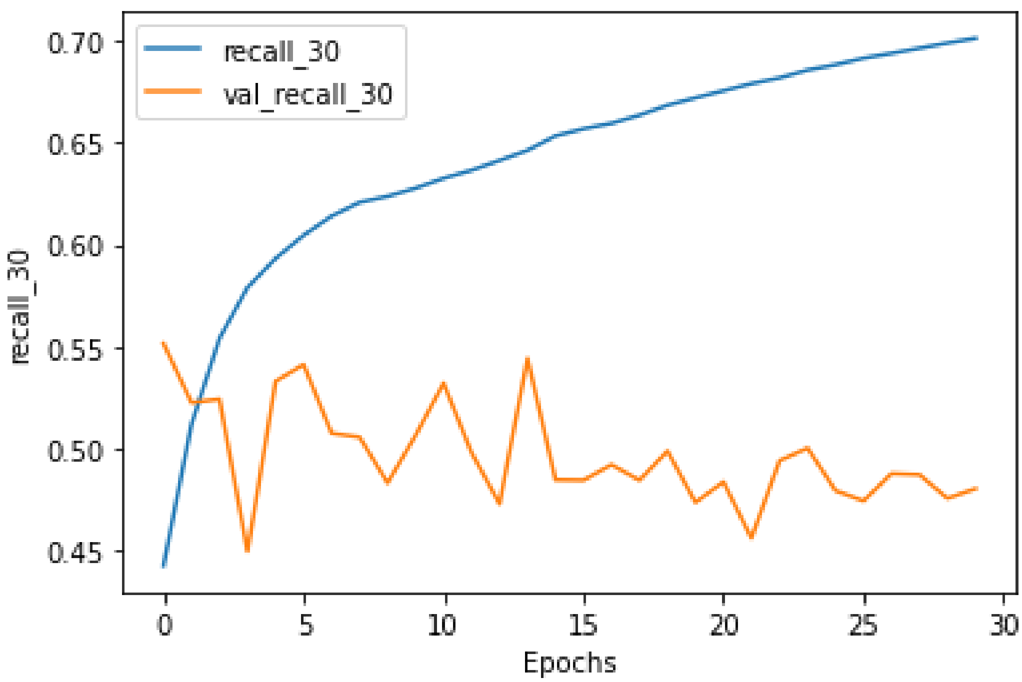
```
500/500 [==============================] - 8s 17ms/step - loss: 0.1902 -
accuracy: 0.9190 - recall_30: 0.7011 - val_loss: 0.3005 - val_accuracy: 0.8716 -
val_recall_30: 0.4801
INFO:tensorflow:Assets written to: /tmp/tmppmdnp52n/model/data/model/assets
2556/2556 [==============================] - 5s 2ms/step - loss: 0.3062 -
accuracy: 0.8824 - recall_30: 0.6906
```

Model accuracy for the displayed run has an accuracy for both training and validation between 85% and 90%. Validation loss is increasing, and validation recall appears to be converging to 0.48.

[76]:
```python
plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
plot_graphs(history, "recall_30")
```







MLFlow Logging To better keep track of each experiment's results we use the MLFlow library which provides a platform for managing the machine learning lifecycle. From the dashboard we can easily see that–if test accuracy is our primary metric–experiment scenarios (4), (5) and (6)

perform the best.

[74]:
```
get_ipython().system_raw("mlflow ui --port 5000 &")

from pyngrok import ngrok

# Terminate open tunnels if exist
ngrok.kill()

# Setting the authtoken (optional)
# Get your authtoken from https://dashboard.ngrok.com/auth
NGROK_AUTH_TOKEN = ""
ngrok.set_auth_token(NGROK_AUTH_TOKEN)

# Open an HTTPs tunnel on port 5000 for http://localhost:5000
ngrok_tunnel = ngrok.connect(addr="5000", proto="http", bind_tls=True)
print("MLflow Tracking UI:", ngrok_tunnel.public_url)
```

```
t=2021-01-15T15:52:40+0000 lvl=warn msg="can't bind default web address, trying
alternatives" obj=web addr=127.0.0.1:4040
```

```
MLflow Tracking UI: https://be330106ff85.ngrok.io
```

0.0.5 Conclusion

While it appears that our models give decent results in terms of accuracy, it is not good enough to beat a baseline heuristic. Furthermore, when analysing our other metrics, we are suffering from some overfitting. Still, over the course of the different experiments, adding embeddings (weight tying) has introduced some improvement. The hypothesis could serve for further experiment with improved methodology.

Recommendation As was mentioned in the methodology section, adding dropout layers, batch normalization, and other regularization should be explored to address the overfitting problem. Also, other implementations of embedding (i.e. TrellisNet) is intended to be explored. Another recommendation is to reframe the problem in terms of actually predicting CMEs rather than just solar flares. However, additional data would be needed for the flare classification. Speaking of data, future work should remove the sampling to maximize the available data.

0.0.6 References

Abdullah, Y., Wang, JTL., Nie, Y., Liu, C., Wang, H. (2020). DeepSun: Machine-Learning-as-a-Service for SolarFlare Prediction.

Kaggle (2019). BigData Cup Challenge 2019: Flare Prediction. Retrieved from https://www.kaggle.com/c/bigdata2019-flare-prediction/data

Kurzgesagt (2020). Could Solar Storms Destroy Civilization? Solar Flares & Coronal Mass Ejections. Retrieved from https://www.youtube.com/watch?v=oHHSSJDJ4oo

MLFlow. https://www.mlflow.org/docs/latest/index.html#

NASA. Sunspots and Solar Flares. Retrieved from https://spaceplace.nasa.gov/solar-activity/en/

Press, Ofir & Wolf, Lior. (2017). Using the Output Embedding to Improve Language Models. 157-163. 10.18653/v1/E17-2025.

Wang, Xiantong & Chen, Yang & Toth, Gabor & Manchester, Ward & Gombosi, T. & Hero, Alfred & Jiao, Zhenbang & Sun, Hu & Jin, Meng & Liu, Yang. (2019). Predicting solar flares with machine learning: investigating solar cycle dependence.

World Science Festival (2014). What's The Real Danger From Solar Flares? Retrieved from https://www.worldsciencefestival.com/2014/10/whats-real-danger-solar-flares/

[ ]:

```

```