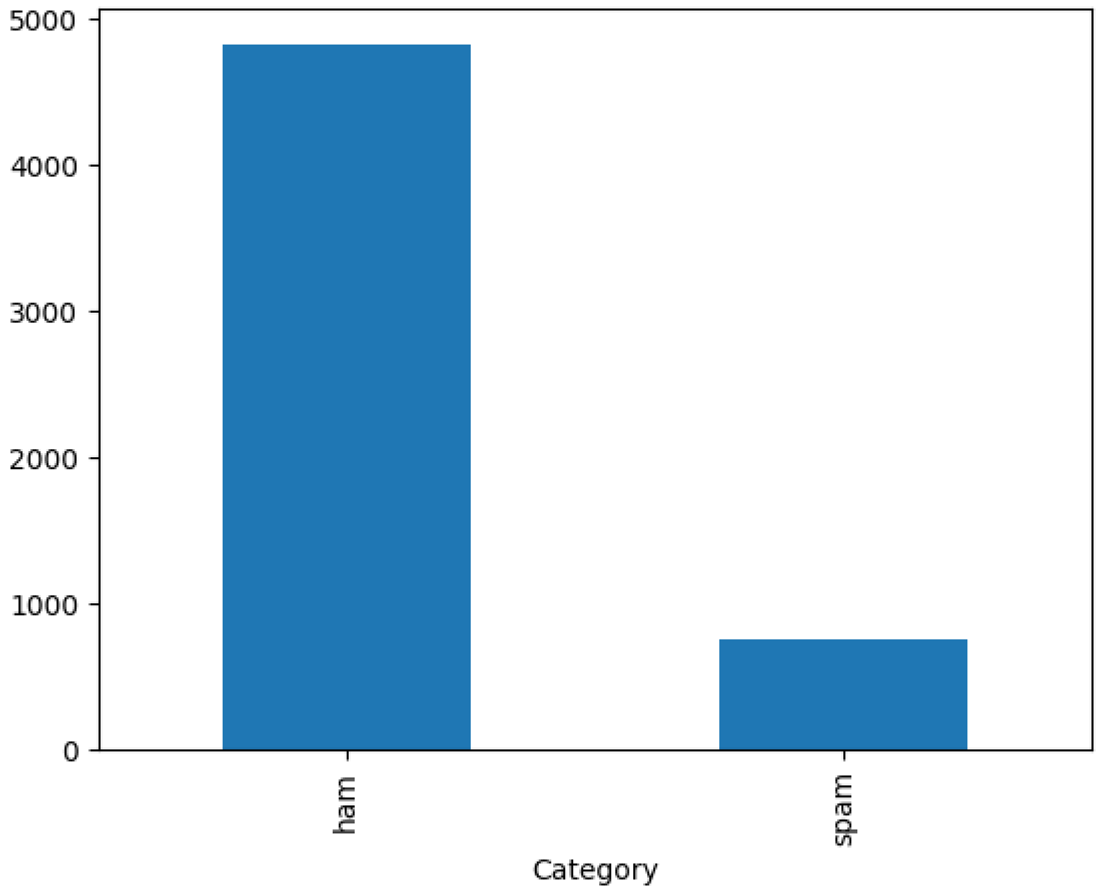
In [237…
```python
import os
# hide alert msg
os.environ['TF_CPP_MIN_LOG_LEVEL']='2'
#  main libs
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Load CSV file into a pandas dataframe

In [238…
```python
# loading spam data cols: ['Category'] ['Message']
# size for each: 5572
df        = pd.read_csv('./data/SPAM_Data.csv')
# print(df)
sentences = df['Message'].tolist()
labels    = df['Category'].tolist() # ham or spam

df['Category'].value_counts().plot(kind='bar')
```

Out[238]:
```
<Axes: xlabel='Category'>
```



Callbacks for fit process

In [239…

```
class myCallback(tf.keras.callbacks.Callback):
#     def on_epoch_end(self, epoch, logs={}):
#         if(logs.get('accuracy')> 0.99):
#             print("\nReached 99% accuracy so cancelling training!", epoch)
#             self.model.stop_training = True
    def on_train_end(self, logs=None):
        print("Training has ended.")

callbacks = myCallback()
```

Preparing data for training

In [240…

```
vocab_size = 1100
embedding_dim = 24
max_length = 100
trunc_type='post'
padding_type='post'
oov_tok = "<OOV>"

# sentences & labels
total_size = 0.88
training_size = int(len(sentences) * total_size)
# tranform 'spam' and 'ham' in 1 or 0
labels = [1 if label == 'spam' else 0 for label in labels]

training_sentences = sentences[0:training_size]
training_labels = labels[0:training_size]
# for future testing data
testing_sentences = sentences[training_size:]
testing_labels = labels[training_size:]

# The dataset is tokenized
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)
tokenizer.fit_on_texts(training_sentences)
word_index = tokenizer.word_index

#sorted into sequences
sequences = tokenizer.texts_to_sequences(training_sentences)

#  sentence must have the same length
padded = pad_sequences(
    sequences,
    maxlen=max_length,
    padding=padding_type,
    truncating=trunc_type
)
testing_sequences = tokenizer.texts_to_sequences(testing_sentences)
testing_padded = pad_sequences(
    testing_sequences,
    maxlen=max_length,
    padding=padding_type,
    truncating=trunc_type
)


prepared_training_answers = np.array(training_labels)
prepared_testing_answers = np.array(testing_labels)
```

Recurrent Neural Network (RNN)

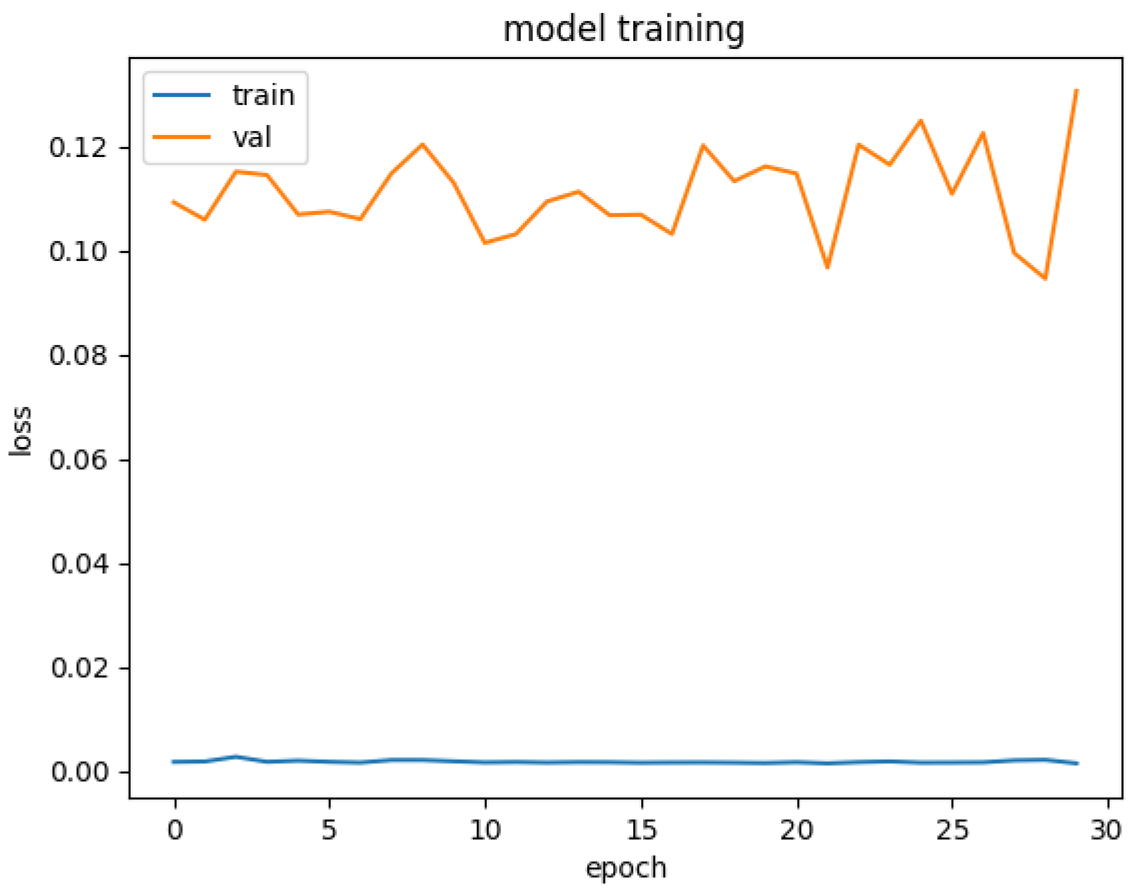
In [278…
```
model = Sequential([
    layers.Embedding(
        vocab_size,
        embedding_dim,
        input_length=max_length
    ),
    layers.Flatten(),
    layers.Dense(24, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

Train Model

In [313…
```
history = model.fit(
    padded,
    prepared_training_answers,
    epochs=30,
    verbose=0,
    validation_data=(testing_padded, prepared_testing_answers),
    callbacks=[callbacks]
)
```

```
Training has ended.
```

In [314…
```
# convert the training history to a dataframe
history_df = pd.DataFrame(history.history)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model training')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```



In [316…

```
text_messages = [
    'Hello Mr.user, to get free money, visit this website free-scamers.pro',
    'did you see this is awesome food pic?',
    'Greg, can you call me back once you get this?',
    'pinterest send you new picture',
    'You won 🏆 1 place, you a winner 100%',
    'Buy new ssd for only for 10k $',
    'C++ webinar today, dont forget Bro!',
    'there are your NASA posts about Mars',
    'free pro credit card, for pro guys as you!:)',
    'Where are you?'
]

# Create the sequences
padding_type='post'
sample_sequences = tokenizer.texts_to_sequences(text_messages)
fakes_padded = pad_sequences(sample_sequences, padding=padding_type, maxlen=

predictions = model.predict(fakes_padded)
for answer, text in zip(predictions, text_messages):
    spamPersentVal = round(answer[0] * 100, 3)
    print('[SPAM DETECTION {}%]: {}'.format(spamPersentVal, text))
```

```
1/1 [==============================] - 0s 21ms/step
[SPAM DETECTION 97.873%]: Hello Mr.user, to get free money, visit this websit
e free-scamers.pro
[SPAM DETECTION 0.0%]: did you see this is awesome food pic?
[SPAM DETECTION 0.0%]: Greg, can you call me back once you get this?
[SPAM DETECTION 10.169%]: pinterest send you new picture
[SPAM DETECTION 62.091%]: You won 🏆 1 place, you a winner 100%
[SPAM DETECTION 4.922%]: Buy new ssd for only for 10k $
[SPAM DETECTION 20.318%]: C++ webinar today, dont forget Bro!
[SPAM DETECTION 0.001%]: there are your NASA posts about Mars
[SPAM DETECTION 91.241%]: free pro credit card, for pro guys as you!:)
[SPAM DETECTION 0.0%]: Where are you?
```

```
In [ ]:
```

```
In [ ]:
```