In [ ]:
```
##pip install tensorflow-text
```

In [ ]:
```
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text
```

Import the dataset (Dataset is taken from kaggle)

In [ ]:
```
import pandas as pd

df = pd.read_csv("spam.csv", encoding='latin1')
df.head(5)
```

Out[ ]:

	label	text	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

In [ ]:
```
df = df.iloc[:,:2]
```

In [ ]:
```
df.columns = ["Category", "Message"]
df
```

Out[ ]:

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

In [ ]:
```
```

In [ ]:
```
df.groupby('Category').describe()
```

Out[ ]:

	Message				
	count	unique	top	freq
Category				
ham	4825	4516	Sorry, I'll call later	30
spam	747	653	Please call our customer service representativ...	4

In [ ]:
```
747/4825
```

Out[ ]:
```
0.15481865284974095
```

15% spam emails, 85% ham emails: This indicates class imbalance

In [ ]:
```
df_spam = df[df['Category']=='spam']
df_spam.shape
```

Out[ ]:
```
(653, 2)
```

In [ ]:
```
df_ham = df[df['Category']=='ham']
df_ham.shape
```

Out[ ]:
```
(4516, 2)
```

In [ ]:
```
df_ham_downsampled = df_ham.sample(df_spam.shape[0])
df_ham_downsampled.shape
```

Out[ ]:
```
(653, 2)
```

In [ ]:
```
df_balanced = pd.concat([df_ham_downsampled, df_spam])
df_balanced.shape
```

Out[ ]:
```
(1306, 2)
```

In [ ]:
```
df_balanced['Category'].value_counts()
```

Out[ ]:
```
ham     653
spam    653
Name: Category, dtype: int64
```

In [ ]:
```
df_balanced['spam']=df_balanced['Category'].apply(lambda x: 1 if x=='spam' else 0)
df_balanced.sample(5)
```

Out[ ]:

	Category	Message	spam
1033	ham	OH MR SHEFFIELD! You wanna play THAT game, oka...	0
2630	ham	No way I'm going back there!	0
1317	spam	Win the newest □ÛÏHarry Potter and the Order o...	1
1483	ham	Purity of friendship between two is not about ...	0
4367	ham	1 I don't have her number and 2 its gonna be a...	0

Split it into training and test data set

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df_balanced['Message'],df_balanced['spam'],
```

In [ ]:
```
X_train.head(4)
```

Out[ ]:
```
4887    You have to pls make a note of all she.s expos...
2076             Trust me. Even if isn't there, its there.
1727                              I went to project centre
464                          Ok i am on the way to railway
Name: Message, dtype: object
```

Now lets import BERT model and get embeding vectors for few sample statements

In [ ]:
```
bert_preprocess = hub.KerasLayer("https://www.kaggle.com/models/tensorflow/bert/frameworks/Tenso
bert_encoder = hub.KerasLayer("https://www.kaggle.com/models/tensorflow/bert/frameworks/TensorFl
```

In [ ]:
```
def get_sentence_embeding(sentences):
    preprocessed_text = bert_preprocess(sentences)
    return bert_encoder(preprocessed_text)['pooled_output']

get_sentence_embeding([
    "500$ discount. hurry up",
    "Bhavin, are you up for a volleybal game tomorrow?"]
)
```

Out[ ]:
```
<tf.Tensor: shape=(2, 768), dtype=float32, numpy=
array([[-0.843517  , -0.5132727 , -0.8884572 , ..., -0.7474886 ,
        -0.75314724,  0.91964495],
       [-0.8720834 , -0.50543964, -0.94446665, ..., -0.8584749 ,
        -0.7174534 ,  0.88082975]], dtype=float32)>
```

Get embeding vectors for few sample words. Compare them using cosine similarity

In [ ]:
```
import tensorflow as tf
from transformers import BertTokenizer, TFBertModel
```

```
text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
preprocessed_text = bert_preprocess(text_input)
outputs = bert_encoder(preprocessed_text)

# Neural network layers
l = tf.keras.layers.Dropout(0.1, name="dropout")(outputs['pooled_output'])
l = tf.keras.layers.Dense(128, activation='gelu', name="dense_layer")(l)
l = tf.keras.layers.Dropout(0.1, name="dropout2")(l)
l = tf.keras.layers.Dense(64, activation='relu', name="dense_layer2")(l)

output_layer = tf.keras.layers.Dense(1, activation='sigmoid', name="output")(l)

# Model
model = tf.keras.Model(inputs=[text_input], outputs=[output_layer])
```

In [ ]:
```
model.summary()
```

```
Model: "model_7"
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 text (InputLayer)           [(None,)]                    0         []

 keras_layer_6 (KerasLayer)  {'input_type_ids': (None,    0         ['text[0][0]']
                             128),
                              'input_mask': (None, 128)
                             , 'input_word_ids': (None,
                              128)}

 keras_layer_7 (KerasLayer)  {'encoder_outputs': [(None   109482241   ['keras_layer_6[1][0]',
                             , 128, 768),                             'keras_layer_6[1][1]',
                              (None, 128, 768),                       'keras_layer_6[1][2]']
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768),
                              (None, 128, 768)],
                              'sequence_output': (None,
                              128, 768),
                              'pooled_output': (None, 7
                             68),
                              'default': (None, 768)}

 dropout (Dropout)           (None, 768)                  0         ['keras_layer_7[1][13]']

 dense_layer (Dense)         (None, 128)                  98432     ['dropout[0][0]']

 dropout2 (Dropout)          (None, 128)                  0         ['dense_layer[0][0]']

 dense_layer2 (Dense)        (None, 64)                   8256      ['dropout2[0][0]']

 output (Dense)              (None, 1)                    65        ['dense_layer2[0][0]']

==================================================================================================
Total params: 109588994 (418.05 MB)
Trainable params: 106753 (417.00 KB)
Non-trainable params: 109482241 (417.64 MB)
__________________________________________________________________________________________________
```

In [ ]:
```
len(X_train)
```

Out[ ]:
```
979
```

In [ ]:
```
METRICS = [
      tf.keras.metrics.BinaryAccuracy(name='accuracy'),
      tf.keras.metrics.Precision(name='precision'),
      tf.keras.metrics.Recall(name='recall')
]

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=METRICS)
```

Train the model

In [ ]:
```
model.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
31/31 [==============================] - 20s 339ms/step - loss: 0.5044 - accuracy: 0.7477 - prec
ision: 0.7580 - recall: 0.7286
Epoch 2/10
31/31 [==============================] - 11s 342ms/step - loss: 0.3170 - accuracy: 0.8498 - prec
ision: 0.8356 - recall: 0.8714
Epoch 3/10
31/31 [==============================] - 11s 343ms/step - loss: 0.2393 - accuracy: 0.9019 - prec
ision: 0.9037 - recall: 0.9000
Epoch 4/10
31/31 [==============================] - 11s 361ms/step - loss: 0.2238 - accuracy: 0.9101 - prec
ision: 0.9102 - recall: 0.9102
Epoch 5/10
31/31 [==============================] - 11s 343ms/step - loss: 0.1949 - accuracy: 0.9244 - prec
ision: 0.9194 - recall: 0.9306
Epoch 6/10
31/31 [==============================] - 11s 339ms/step - loss: 0.1745 - accuracy: 0.9356 - prec
ision: 0.9384 - recall: 0.9327
Epoch 7/10
31/31 [==============================] - 11s 343ms/step - loss: 0.1546 - accuracy: 0.9418 - prec
ision: 0.9464 - recall: 0.9367
Epoch 8/10
31/31 [==============================] - 10s 330ms/step - loss: 0.1427 - accuracy: 0.9510 - prec
ision: 0.9547 - recall: 0.9469
Epoch 9/10
31/31 [==============================] - 10s 331ms/step - loss: 0.1476 - accuracy: 0.9428 - prec
ision: 0.9447 - recall: 0.9408
Epoch 10/10
31/31 [==============================] - 11s 355ms/step - loss: 0.1423 - accuracy: 0.9459 - prec
ision: 0.9468 - recall: 0.9449
```

Out[ ]:
```
<keras.src.callbacks.History at 0x7e5e6aa27910>
```

In [ ]:
```
model.evaluate(X_test, y_test)
```

```
11/11 [==============================] - 5s 309ms/step - loss: 0.2040 - accuracy: 0.9419 - preci
sion: 0.9390 - recall: 0.9448
```

Out[ ]:
```
[0.2040223777294159,
 0.9418960213661194,
 0.9390243887901306,
 0.9447852969169617]
```

In [ ]:
```
y_predicted = model.predict(X_test)
y_predicted = y_predicted.flatten()
```

```
12/12 [==============================] - 5s 324ms/step
```

In [ ]:
```
import numpy as np

y_predicted = np.where(y_predicted > 0.5, 1, 0)
y_predicted
```

Out[ ]:
```
array([1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1,
       1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0])
```

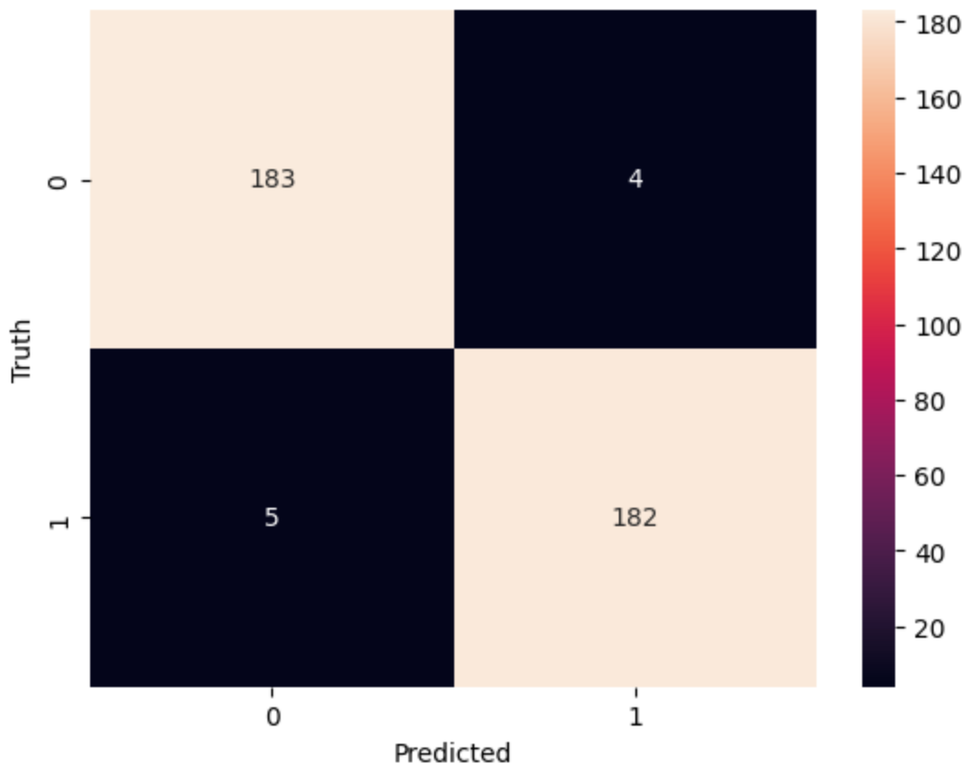
In [ ]:
```
from sklearn.metrics import confusion_matrix, classification_report

cm = confusion_matrix(y_test, y_predicted)
cm
```

Out[ ]:
```
array([[183,   4],
       [  5, 182]])
```

In [ ]:
```
from matplotlib import pyplot as plt
import seaborn as sn
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[ ]:
```
Text(50.722222222222214, 0.5, 'Truth')
```



In [ ]:
```
print(classification_report(y_test, y_predicted))
```

```
              precision    recall  f1-score   support

           0       0.94      0.87      0.90       187
           1       0.88      0.95      0.91       187

    accuracy                           0.91       374
   macro avg       0.91      0.91      0.91       374
weighted avg       0.91      0.91      0.91       374
```

Inference

In [ ]:
```
reviews = [
    'Enter a chance to win $5000, hurry up, offer valid until march 31, 2021',
    'You are awarded a SiPix Digital Camera! call 09061221061 from landline. Delivery within 28d
    'it to 80488. Your 500 free text messages are valid until 31 December 2005.',
    'Hey Sam, Are you coming for a cricket game tomorrow',
    "Why don't you wait 'til at least wednesday to see if you get your ."
]
model.predict(reviews)
```

Out[ ]:
```
array([[0.8734353 ],
       [0.92858446],
       [0.8960864 ],
       [0.29311982],
       [0.13262196]], dtype=float32)
```