College of Computer and Information Sciences

Text Spam Classification(English)/Positive and Negative Classification(Arabic)

CS365 Project

Member name	Academic number	Section
Abdulelah Al-Jarboa	440015920	172
Omar Al-Khuwaytim	440013412	171
Abdulrazaq Al-Thuwaini	440017330	171

Table Of Contents

Chapter 1: Text Spam Classification(English)

1.1 Project description:

As we all know, spam messages are very common nowadays, and if people are not careful they can be a target of these scammers which will cause them harm in many ways. So In this project, our aim is to build a model using deep learning to identify if the message we are receiving is spam or not.

1.2 Potential Use Cases:

The use cases are going to focus primarily on text spam. An example of that is text spam filtering, text message, and email spam filtering. This is more efficient because it will give us the ability to filter spam messages in big quantities and with high accuracy. By using machine learning we are removing the human factor which is slow and not as accurate.

1.3 Software and Services:

In this project, we used Python to build the model because of its huge libraries and the big community support.

Libraries:

Tensorflow: is an open-source machine-learning framework that allows us to build machine learning, deep learning, and neural networks.

Nltk: an open-source library for natural language processing that provides an easy-to-use interface for a wide range of NLP tasks such as tokenization, stemming, lemmatization, parsing, and much more.

Numpy: is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. We used it to perform a wide variety of mathematical operations on arrays and tuning.

Pandas: is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. We used it to read and clean the dataset.

Sklearn: is an open-source data analysis library and the gold standard for Machine Learning in the Python ecosystem. We used it to split the dataset, import the models, train, evaluate, and test the models.

Re: is a library that provides powerful regular expression features.

String: is a built-in Python library that is used to process strings in Python.

Keras: is a library that gives us the ability to create deep learning models.

Matplotlib: is a comprehensive library for creating static, animated, and interactive visualizations in Python. We used it to make the data visualization.

Seaborn: is a Python data visualization library based on Matplotlib it is used to create static, animated, and interactive visualizations. We used it to make the data visualization.

For the IDE we will use visual studio code From Microsoft because of the ease of use and some awesome add-on that make testing faster. The most important ad in this project was the jupyter notebook.

SMS Spam Collection Dataset | Kaggle

1.4 Dataset

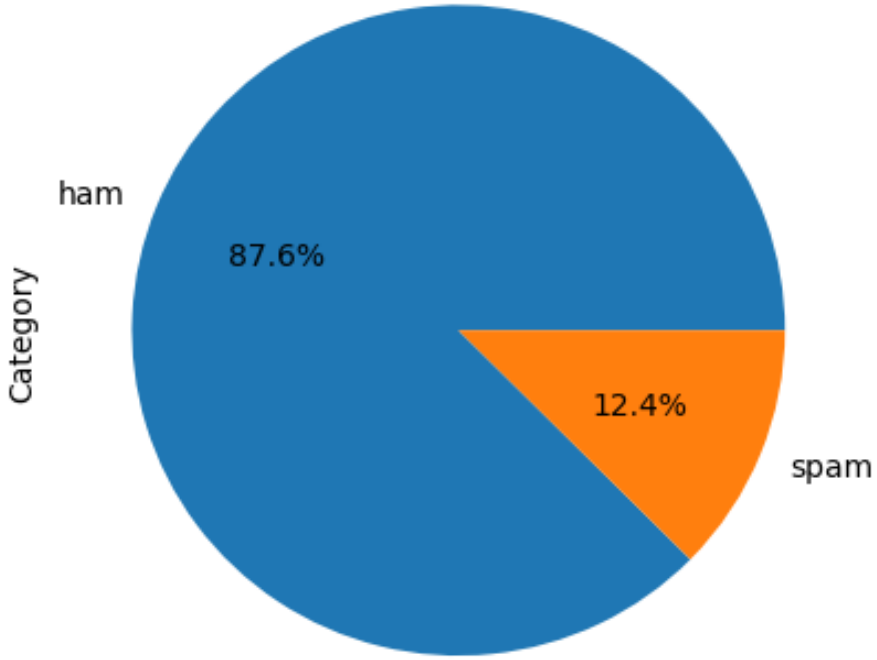
The dataset we used in this project was SPAM text messages. Which is a dataset that contains 5576 spam and ham text messages.

Attributes

The attributes in the data set are Category which indicates if the text message is spam or ham, and the other attribute Message which is the spam or ham text message.

	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...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...
10	ham	I'm gonna be home soon and i don't want to tal...
11	spam	SIX chances to win CASH! From 100 to 20,000 po...
12	spam	URGENT! You have won a 1 week FREE membership ...
13	ham	I've been searching for the right words to tha...
14	ham	I HAVE A DATE ON SUNDAY WITH WILL!!
15	spam	XXXMobileMovieClub: To use your credit, click ...
16	ham	Oh k...i'm watching here:)
17	ham	Eh u remember how 2 spell his name... Yes i di...
18	ham	Fine if that□s the way u feel. That□s the way ...
19	spam	England v Macedonia - dont miss the goals/team...

The following chart is going to show the percentage of spam and ham messages in the dataset:



1.5 The Model

We used in this project was Long Short Term Memory (LSTM) which is an artificial neural network used in the fields of artificial intelligence and deep learning that is designed to handle sequential data for applications such as text classification, Part of speech tagging and much more.

We chose this model for our project because RNNs are able to collect additional information and values and position them accurately and recognize the patterns, and other than that they can be stacked to give us an even better performance. It does that using its short and long-term memory to see what values to keep and what values to drop.

Model implementation

```
# Create a sequential model

 model = tensorflow.keras.models.Sequential()



# Add an embedding layer to the model with the specified maximum
number of words and embedding dimension

model.add(layers.Embedding(MAX_NUM_WORDS, EMBEDDING_DIM,
input_length=X.shape[1]))



# Add an LSTM layer to the model with 32 units

model.add(layers.LSTM(32))



# Add a dense output layer to the model with a sigmoid
activation function

model.add(layers.Dense(1, activation='sigmoid'))



# Compile the model with binary cross-entropy loss, the Adam
optimizer, and accuracy as a metric

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



# Print a summary of the model architecture

print(model.summary())
```

```
history = model.fit(X_train, Y_train, epochs=10,
batch_size=64,validation_split=0.2,callbacks=[EarlyStopping(moni
tor='val_loss', patience=3, min_delta=0.0001)])

#the dataset is split into 20% 80% which mean 80% for traning
and 20% for testing

#epochs is the total number of iterations of all the training
data in one cycle for training the machine learning model
```

Model Summary

```
Model: "sequential_19"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_19 (Embedding)    (None, 442, 100)          707000

 lstm_19 (LSTM)              (None, 32)                17024

 dense_19 (Dense)            (None, 1)                 33

=================================================================
Total params: 724,057
Trainable params: 724,057
Non-trainable params: 0
_________________________________________________________________
```

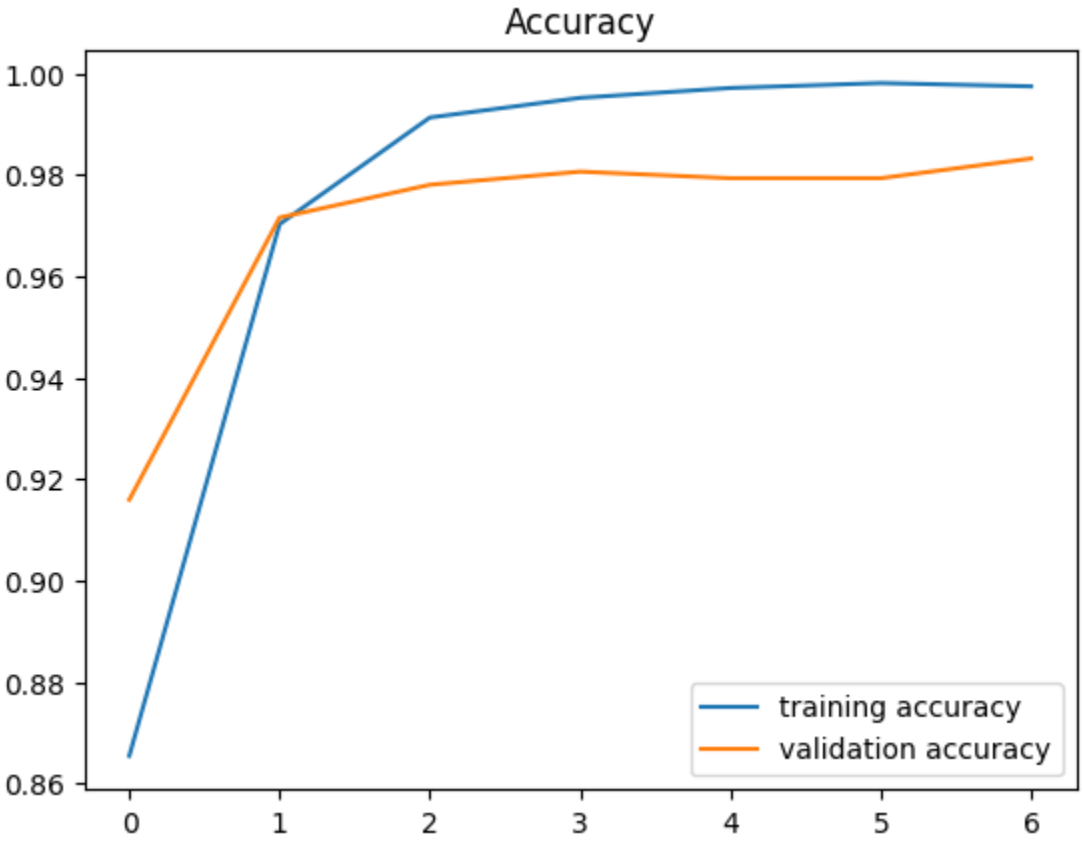
1.6 Model Evaluation

When we trained the model we used 5 epochs, because we saw that it gave us the best accuracy and loss function performance completed to the training time. We achieved more than 99% accuracy and less than 1.7% loss function.

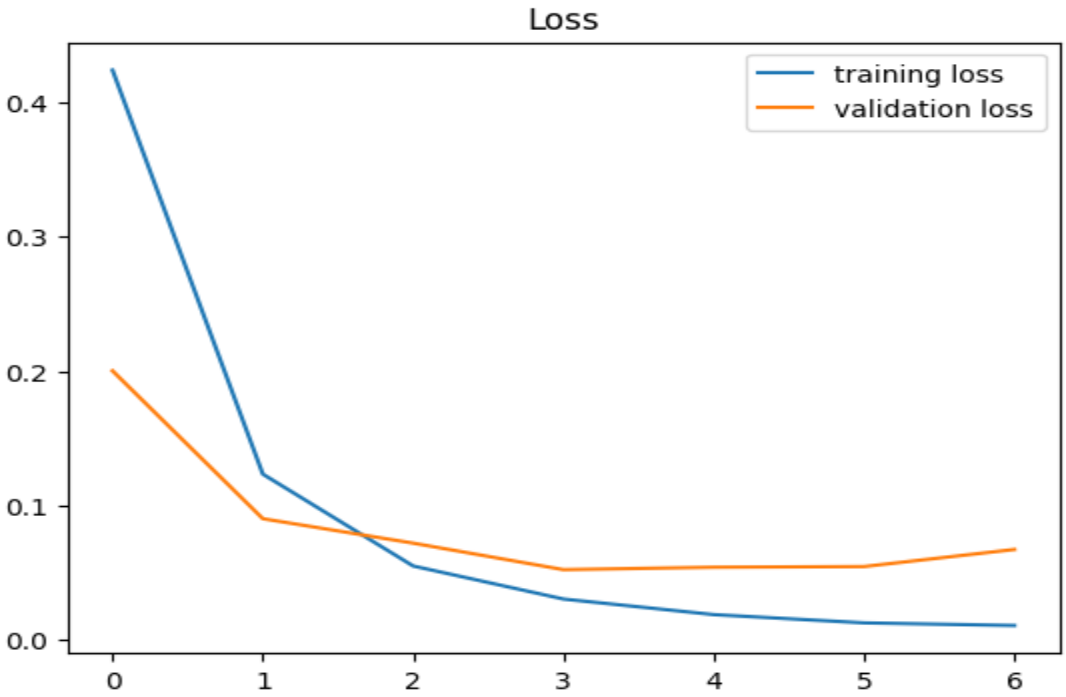
```
Epoch 1/10
49/49 [==============================] - 4s 47ms/step - loss: 0.4048 - accuracy: 0.8628 - val_loss: 0.1954 - val_accuracy: 0.9224
Epoch 2/10
49/49 [==============================] - 2s 39ms/step - loss: 0.1218 - accuracy: 0.9696 - val_loss: 0.0919 - val_accuracy: 0.9754
Epoch 3/10
49/49 [==============================] - 2s 38ms/step - loss: 0.0578 - accuracy: 0.9906 - val_loss: 0.0676 - val_accuracy: 0.9806
Epoch 4/10
49/49 [==============================] - 2s 38ms/step - loss: 0.0327 - accuracy: 0.9951 - val_loss: 0.0583 - val_accuracy: 0.9819
Epoch 5/10
...
Epoch 7/10
49/49 [==============================] - 2s 40ms/step - loss: 0.0091 - accuracy: 0.9987 - val_loss: 0.0583 - val_accuracy: 0.9819
Epoch 8/10
49/49 [==============================] - 2s 39ms/step - loss: 0.0071 - accuracy: 0.9990 - val_loss: 0.0638 - val_accuracy: 0.9780
```

Using matplotlib we can represent the data with graphs:

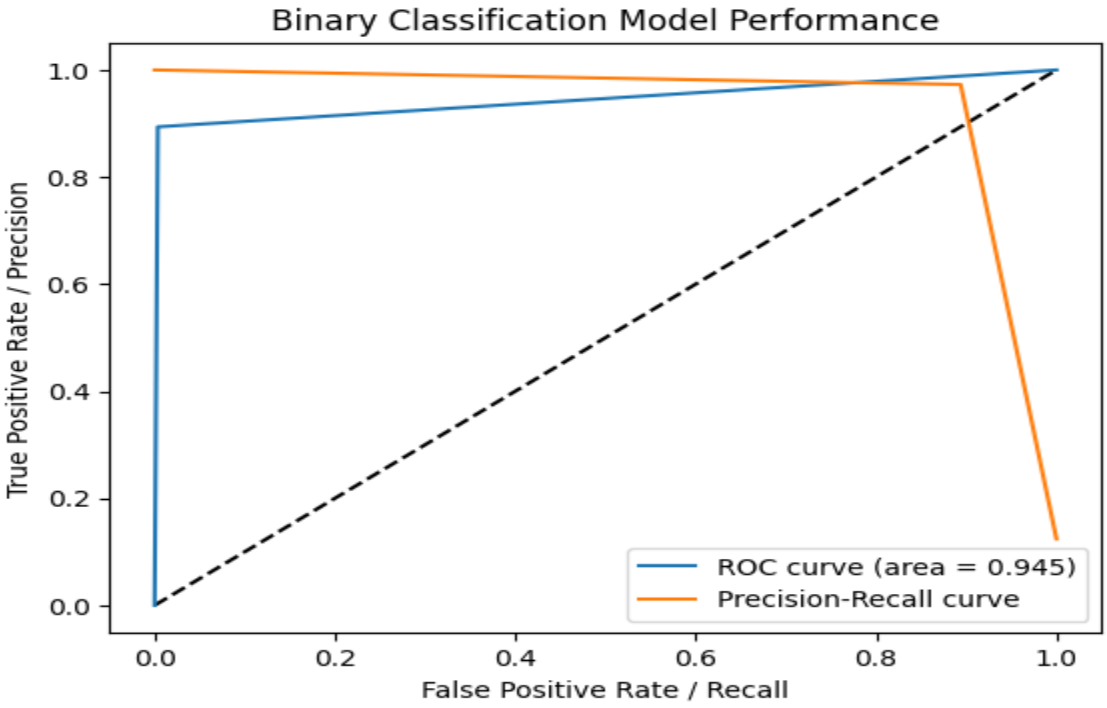
Accuracy


Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0.86
0
1
2
3
4
5
6
training accuracy
validation accuracy

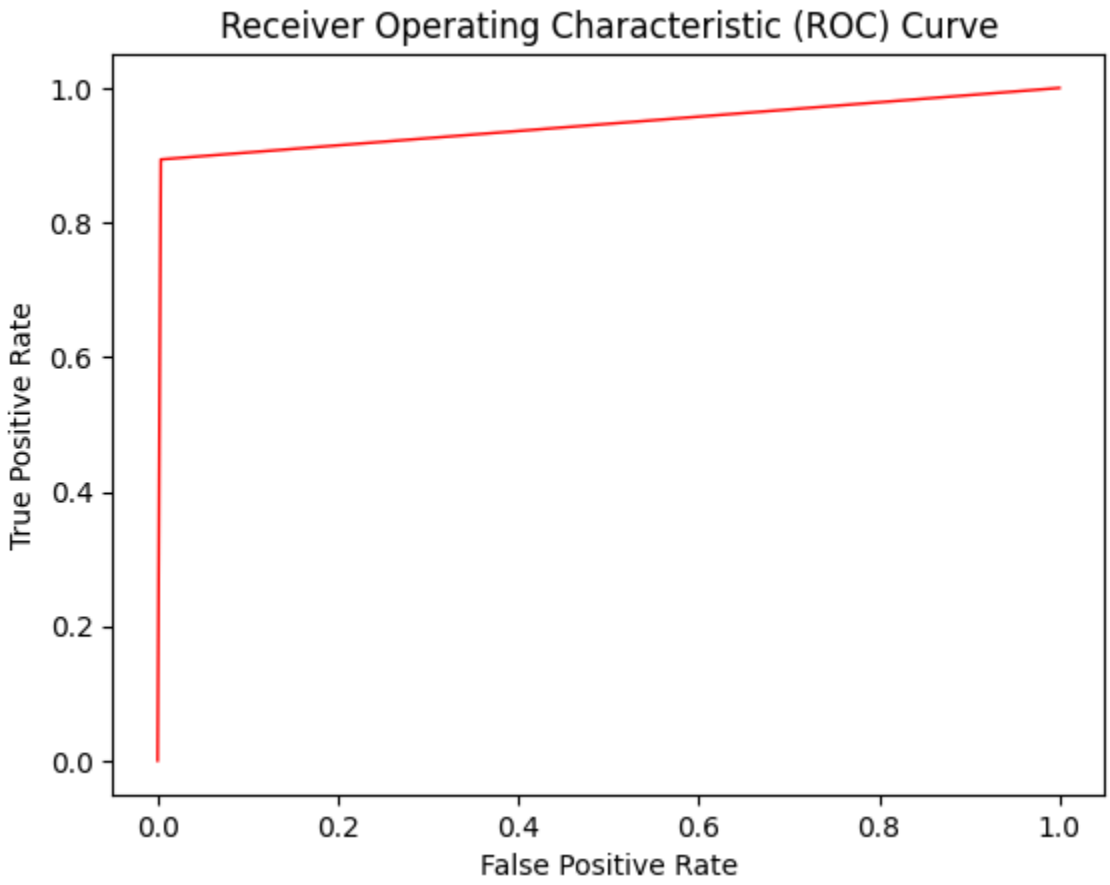
Loss function



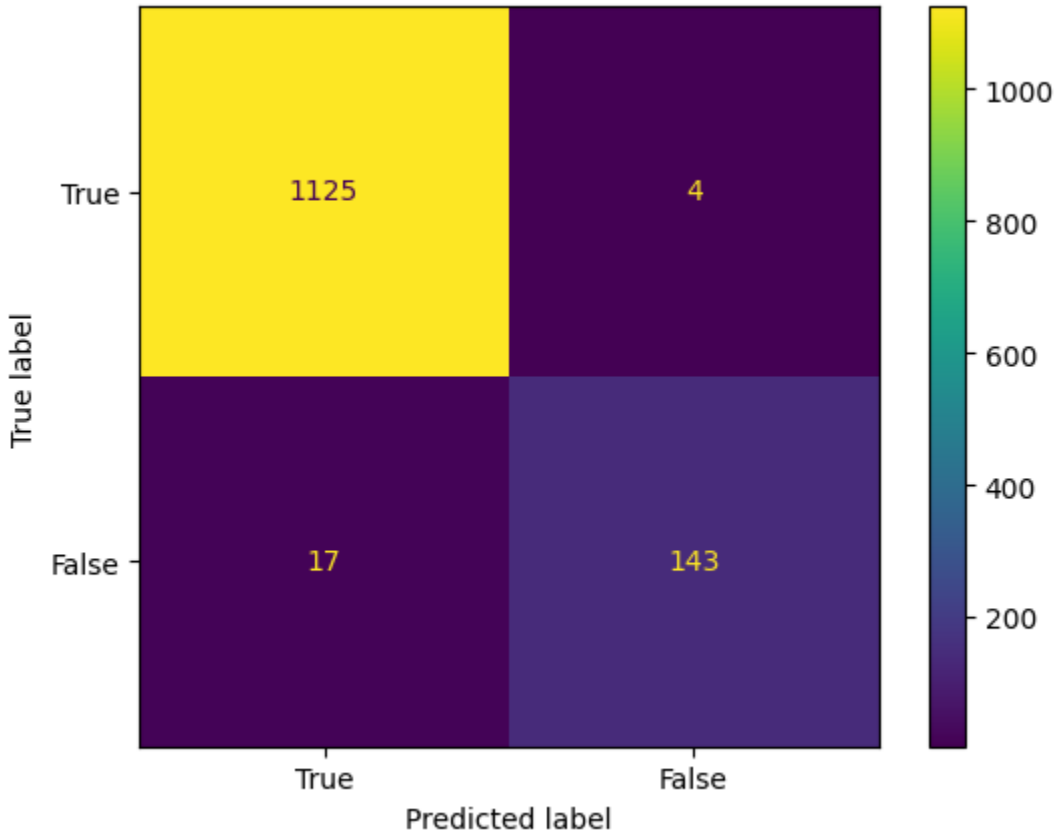
Recall vs Precision



Ture and False Positive



Confusion Matrix



Classification Report

```
···  41/41 - 1s - 949ms/epoch - 23ms/step
                precision    recall  f1-score   support

           HAM       0.99      1.00      0.99      1129
          SPAM       0.97      0.89      0.93       160

      accuracy                           0.98      1289
     macro avg       0.98      0.95      0.96      1289
  weighted avg       0.98      0.98      0.98      1289
```

1.7 Testing

After training the model we will save it as a file which can be done by the following command

```
model.save("./spamClassificationModel.h5")

#this funcaion saves the model
```

Testing the model

```
REMINDER FROM O2: To get 2.50 pounds free call credit and
details of great offers pls reply 2 this text with your valid
name, house no and postcode

1/1 [==============================] - 0s 408ms/step

0.9850654

SPAM
```

```
This is the 2nd time we have tried 2 contact u. U have won the
£750 Pound prize. 2 claim is easy

1/1 [==============================] - 0s 396ms/step

0.99109054

SPAM
```

```
Will ü b going to esplanade fr home?

1/1 [==============================] - 0s 400ms/step

0.00052707683

HAM
```

```
Pity, * was in mood for that. So...any other suggestions?

1/1 [==============================] - 0s 461ms/step

0.002344746

HAM
```

Chapter 2: Positive and Negative Classification(Arabic)

2.1 Project description:

For this project, we wanted to make a model that can classify positive and negative input text after learning from big Arabic data taken from a wide array of resources.

The dataset combines reviews from hotels, books, movies, products, and a few airlines. It has two classes (Negative and Positive). Most were mapped from reviewers' ratings, above 3 positive and below 3 negative. Each row has a label and text separated by a comma (CSV).

2.2 Potential Use Cases:

The model can be used for social media posts, other types of text data, customer evaluations and comments for goods and services, and other types of text data to assess overall sentiment. By doing this, organizations may better understand how customers perceive them and make data-driven decisions.

2.3 Libraries:

Tensorflow: is an open-source machine-learning framework that allows us to build machine learning, deep learning, and neural networks.

Nltk: an open-source library for natural language processing that provides an easy-to-use interface for a wide range of NLP tasks such as tokenization, stemming, lemmatization, parsing, and much more.

Numpy: is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. We used it to perform a wide variety of mathematical operations on arrays and tuning.

Pandas: is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. We used it to read and clean the dataset.

Sklearn: is an open-source data analysis library and the gold standard for Machine Learning in the Python ecosystem. We used it to split the dataset, import the models, train, evaluate, and test the models.

Re: is a library that provides powerful regular expression features.

String: is a built-in Python library that is used to process strings in Python.

Keras: is a library that gives us the ability to create deep learning models.

Matplotlib: is a comprehensive library for creating static, animated, and interactive visualizations in Python. We used it to make the data visualization.

Seaborn: is a Python data visualization library based on Matplotlib it is used to create static, animated, and interactive visualizations. We used it to make the data visualization.

Farasa: is a library for Arabic stemming and lemmatization and sentence segmentation and other Arabic processing

For the IDE we will use visual studio code From Microsoft because of the ease of use and some awesome add-on that make testing faster. The most important ad in this project was the jupyter notebook.

2.4 Dataset

The dataset we used in this project was Arabic 100k Reviews. Which is a dataset that contains 100k positive and negative and mixed reviews but we removed the mixed reviews to fit our use case.

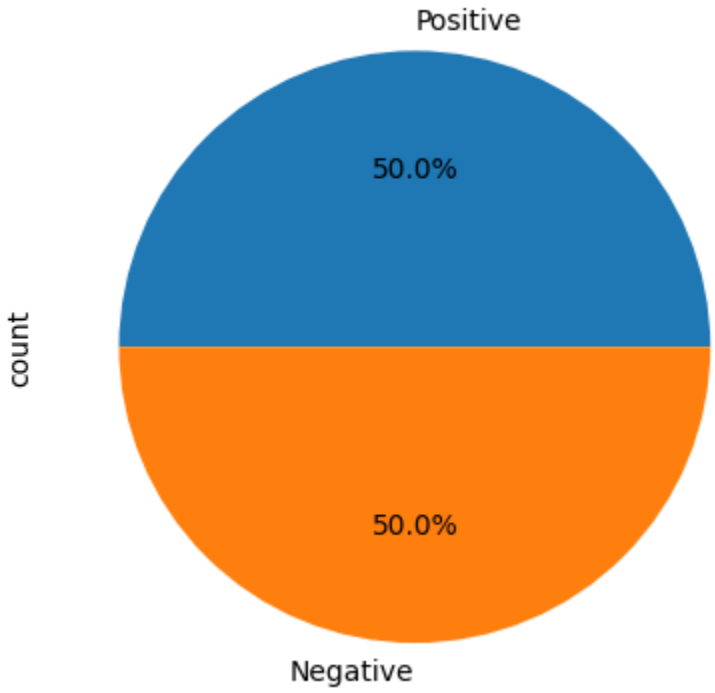
Arabic 100k Reviews | Kaggle

Attributes

The attributes in the data set are the Label which indicates if the review is positive or negative, and the other attribute Text which is the positive or negative review.

	label	text
0	Positive	...ممتاز نوعا ما . النظافة والموقع والتجهيز والشا
1	Positive	...أحد أسباب نجاح الإمارات أن كل شخص في هذه الدول
2	Positive	...هادفة .. وقوية. تنقلك من صخب شوارع القاهرة الى
3	Positive	...خلصنا .. مبدئيا اللي مستني ابهار زي الفيل الاز
4	Positive	...ياسات جلوريا جزء لا يتجزأ من دبي . فندق متكامل
5	Positive	...أسلوب الكاتب رائع جدا و عميق جدا، قرأته عدة مر
6	Positive	...استثنائي. الهدوء في الجناح مع مسبح. عدم وجود ع
7	Positive	...الكتاب هو السيرة الذاتية للحداثة في المملكة بل
8	Positive	..من أجمل ما قرأت.. رواية تستحق القراءة فعلا
9	Positive	...بشكل عام جيده .. . التجاوب جيد جدا من قبل موظف
10	Positive	... .انا بموووووووووووووت ف حاجة اسمها أدهم صبري
11	Positive	... كالعادة يدهشني د. منذر بأسلوب كتابته .. تسلسل
12	Positive	...كتاب حاله فضائيه : . الجنون بعينه. كتاب تنويري
13	Positive	استثنائي. . الواي فاي ليس مجانا
14	Positive	..الكتاب من أول صفحة فرق معايا
15	Positive	وهذا ما سيحدث معنا
16	Positive	...استثنائي. هدواء المكان وتميز الخدمات. يحتاج ال
17	Positive	...عندما نفرح ،نظن أن لا نهاية لفرحنآ . و عندمآنح
18	Positive	عمان . كل شي. لا يوجد
19	Positive	...لا أقيم الرواية بلغتها الأدبية أبدا . ولا بتم

The following chart is going to show the percentage of the positive and negative reviews in the dataset:



2.5 The Model

We used the same model in the 1.5 section

Model implementation

```
# Create a sequential model

model = tensorflow.keras.models.Sequential()



# Add an embedding layer to the model with the specified maximum
number of words and embedding dimension

model.add(layers.Embedding(MAX_NUM_WORDS, EMBEDDING_DIM,
input_length=X.shape[1]))



# Add an LSTM layer to the model with 32 units

model.add(layers.LSTM(32))



# Add a dense output layer to the model with a sigmoid
activation function

model.add(layers.Dense(1, activation='sigmoid'))



# Compile the model with binary cross-entropy loss, the Adam
optimizer, and accuracy as a metric

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

```
# Print a summary of the model architecture

print(model.summary())


history = model.fit(X_train, Y_train, epochs=10,
batch_size=64,validation_split=0.2,callbacks=[EarlyStopping(moni
tor='val_loss', patience=3, min_delta=0.0001)])

 #the dataset is split into 20% 80% which mean 80% for traning
and 20% for testing

 #epochs is the total number of iterations of all the training
data in one cycle for training the machine learning model
```

Model Summary

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, 6106, 100)         4795500

 lstm_5 (LSTM)               (None, 32)                17024

 dense_5 (Dense)             (None, 1)                 33

=================================================================
Total params: 4,812,557
Trainable params: 4,812,557
Non-trainable params: 0
```

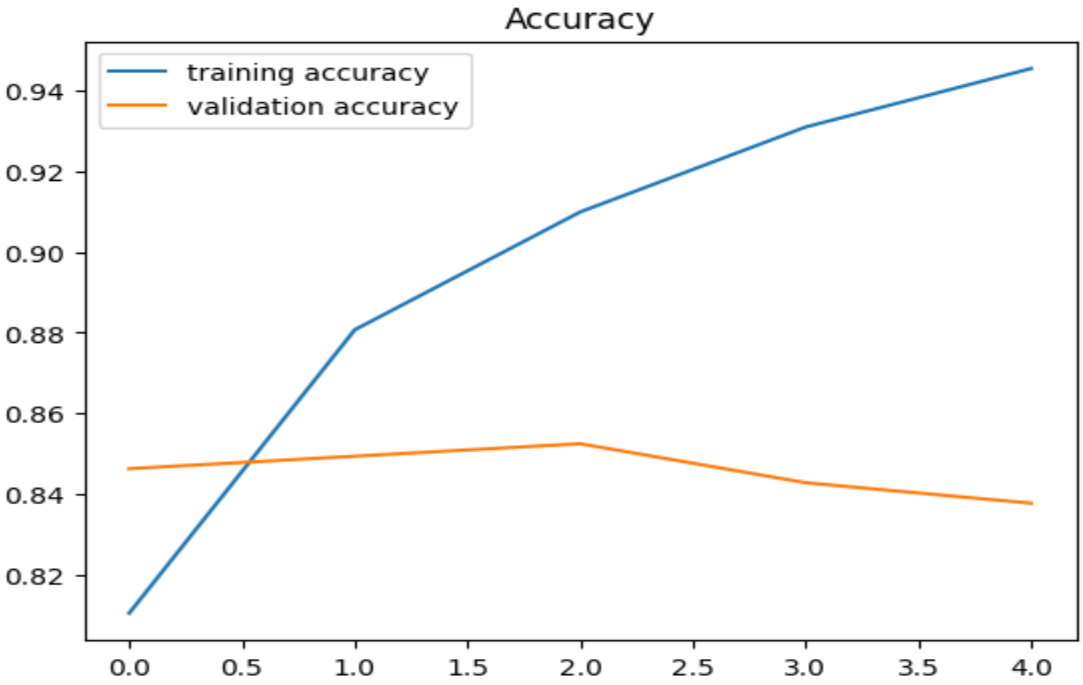
2.6 Model Evaluation

When we trained the model we used 10 epochs, because we saw that it gave us the best accuracy and loss function performance completed to the training time. We achieved 94% accuracy the training and 83% on validation and less than 0.1414 loss function.

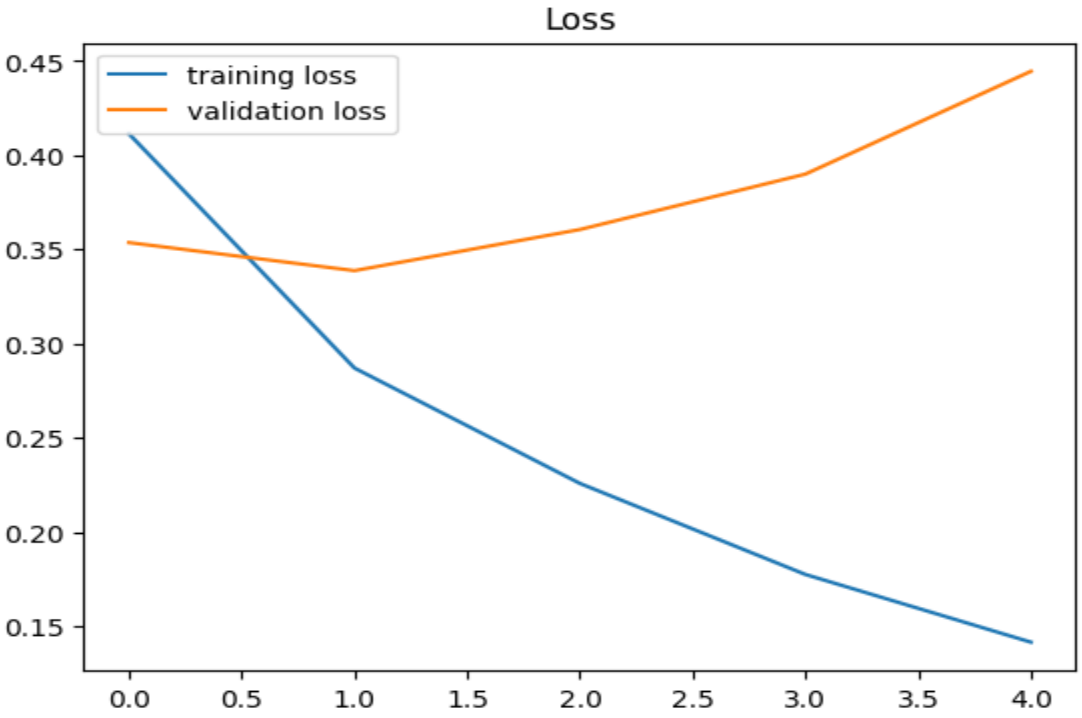
```
None
Epoch 1/10
667/667 [==============================] - 307s 449ms/step - loss: 0.4114 - accuracy: 0.8106 - val_loss: 0.3536 - val_accuracy: 0.8463
Epoch 2/10
667/667 [==============================] - 303s 455ms/step - loss: 0.2870 - accuracy: 0.8807 - val_loss: 0.3387 - val_accuracy: 0.8494
Epoch 3/10
667/667 [==============================] - 323s 485ms/step - loss: 0.2257 - accuracy: 0.9099 - val_loss: 0.3606 - val_accuracy: 0.8525
Epoch 4/10
667/667 [==============================] - 328s 492ms/step - loss: 0.1774 - accuracy: 0.9309 - val_loss: 0.3900 - val_accuracy: 0.8429
Epoch 5/10
667/667 [==============================] - 324s 486ms/step - loss: 0.1414 - accuracy: 0.9454 - val_loss: 0.4446 - val_accuracy: 0.8378
```

Using matplotlib we can represent the data with graphs:

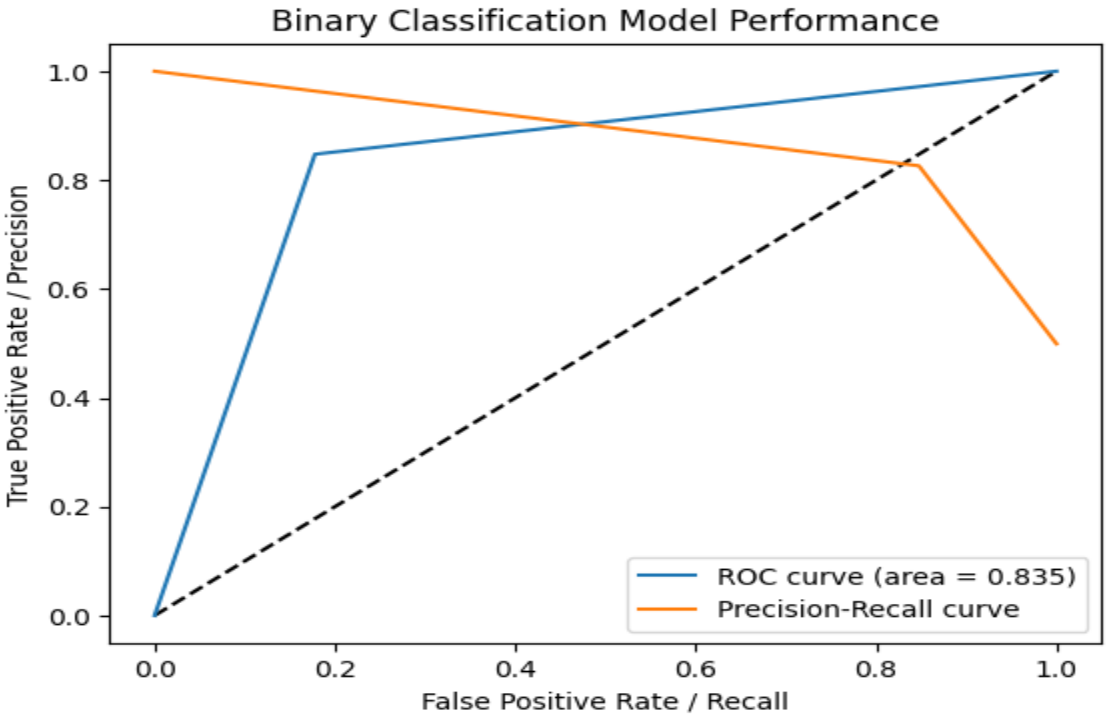
Accuracy



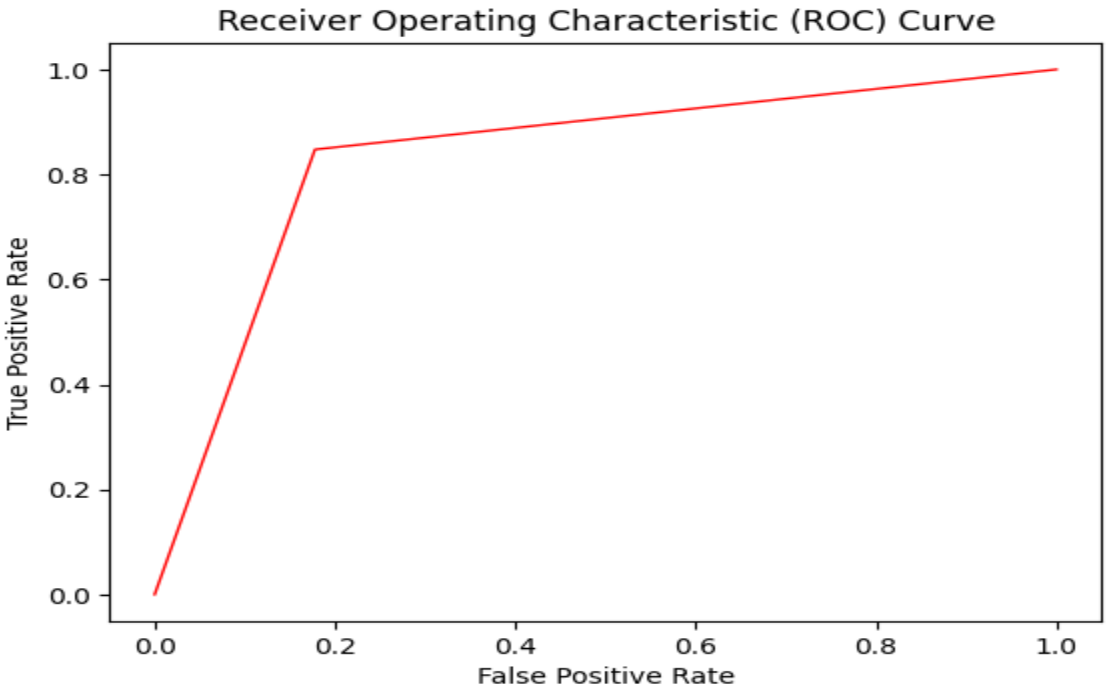
Loss function



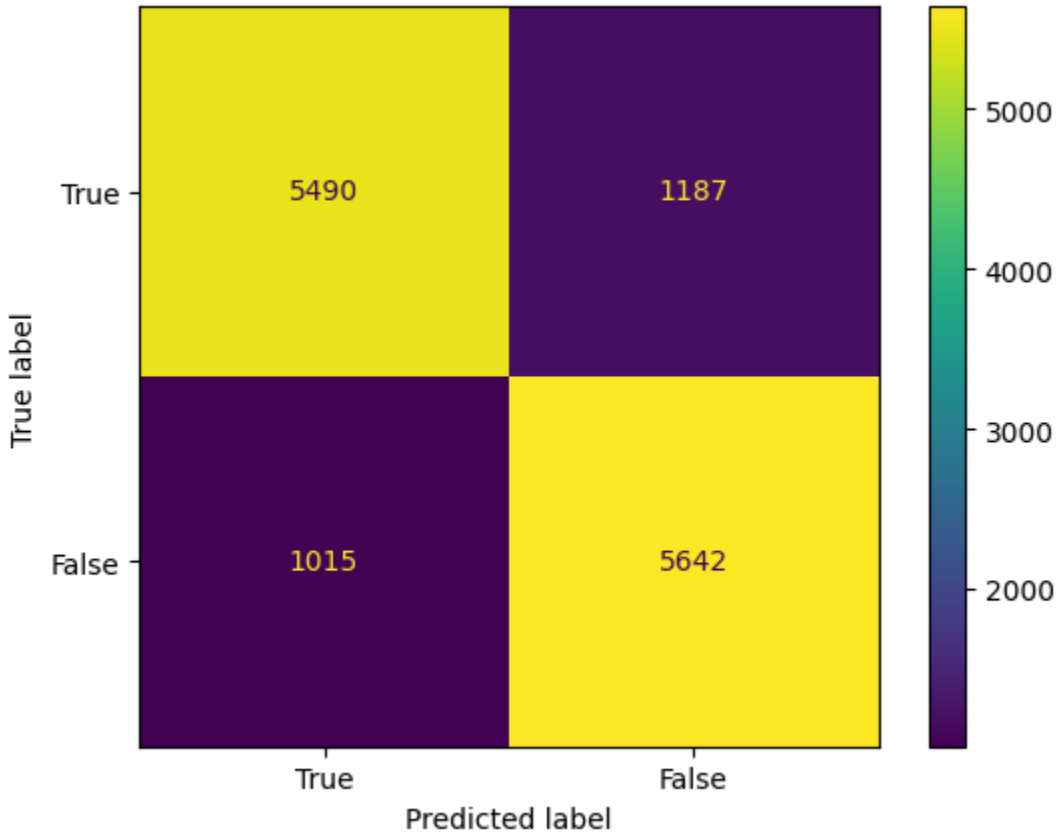
Recall vs Precision



Ture and False Positive



Confusion Matrix



Classification Report

```
···  417/417 - 75s - 75s/epoch - 180ms/step
                precision    recall  f1-score   support

      POSITIVE       0.84      0.82      0.83      6677
      NEGATIVE       0.83      0.85      0.84      6657

      accuracy                           0.83     13334
     macro avg       0.84      0.83      0.83     13334
  weighted avg       0.84      0.83      0.83     13334
```

2.7 Testing

After training the model we will save it as a file which can be done by the following command

```
model.save("./NO_STOPWORDS.h5") #this funcaion saves the model
```

Testing the model

```
جميل جدا

1/1 [==============================] - 1s 552ms/step

0.9315876

POSITIVE
```

```
احد اسوء الكتب

1/1 [==============================] - 3s 3s/step

0.021445224

NEGATIVE
```

```
اكثر شي حلو

1/1 [==============================] - 1s 583ms/step

0.84355307

POSITIVE
```

```
اكثر شي سيئ

1/1 [==============================] - 1s 537ms/step

0.028282495

NEGATIVE
```