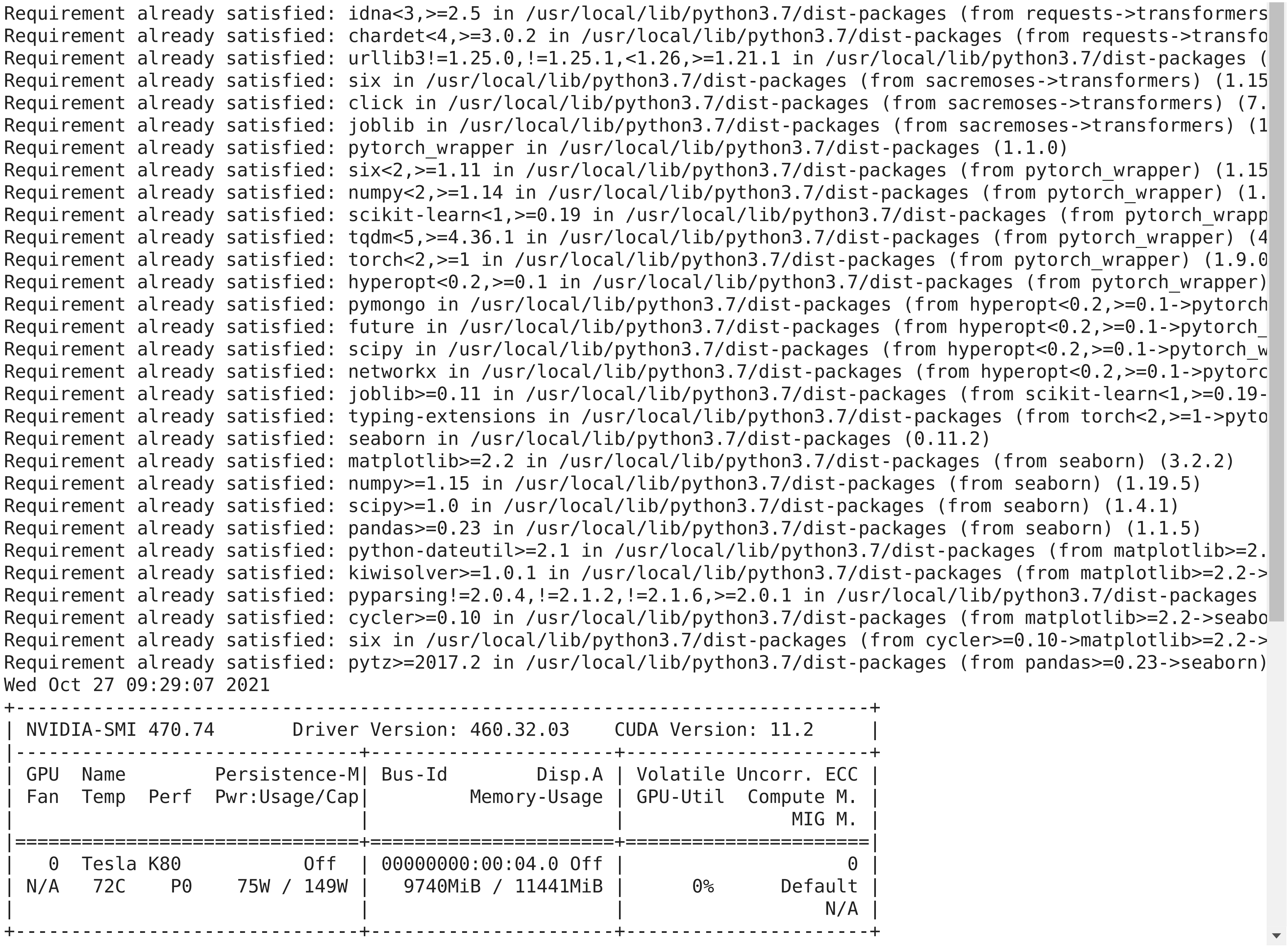
Part 1 : Load Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_re
```

```
!pip install transformers
!pip install pytorch_wrapper
!pip install seaborn
!nvidia-smi

from google.colab import drive
drive.mount('/content/drive')
!unrar x -Y /content/drive/MyDrive/dataset.rar dataset.csv
!grep -v is_fake dataset.csv > clean_dataset.csv
!git clone https://github.com/nlpaueb/greek-bert.git
```

```
    Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-packages (4.11.3)
    Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from transformers) (6.0)
    Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.62.3)
    Requirement already satisfied: sacremoses in /usr/local/lib/python3.7/dist-packages (from transformers) (0.0.46)
    Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (20
    Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (21.0
    Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transformers) (4
    Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.19.5)
    Requirement already satisfied: tokenizers<0.11,>=0.10.1 in /usr/local/lib/python3.7/dist-packages (from transforme
    Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.3.0)
    Requirement already satisfied: huggingface-hub>=0.0.17 in /usr/local/lib/python3.7/dist-packages (from transformer
    Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.0)
    Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from huggingface-hub>=
    Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=20.0->t
    Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->trans
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->transf
```



```
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->transformers
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->transfo
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (1.15
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (7.
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from sacremoses->transformers) (1
Requirement already satisfied: pytorch_wrapper in /usr/local/lib/python3.7/dist-packages (1.1.0)
Requirement already satisfied: six<2,>=1.11 in /usr/local/lib/python3.7/dist-packages (from pytorch_wrapper) (1.15
Requirement already satisfied: numpy<2,>=1.14 in /usr/local/lib/python3.7/dist-packages (from pytorch_wrapper) (1.
Requirement already satisfied: scikit-learn<1,>=0.19 in /usr/local/lib/python3.7/dist-packages (from pytorch_wrapp
Requirement already satisfied: tqdm<5,>=4.36.1 in /usr/local/lib/python3.7/dist-packages (from pytorch_wrapper) (4
Requirement already satisfied: torch<2,>=1 in /usr/local/lib/python3.7/dist-packages (from pytorch_wrapper) (1.9.0
Requirement already satisfied: hyperopt<0.2,>=0.1 in /usr/local/lib/python3.7/dist-packages (from pytorch_wrapper)
Requirement already satisfied: pymongo in /usr/local/lib/python3.7/dist-packages (from hyperopt<0.2,>=0.1->pytorch
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages (from hyperopt<0.2,>=0.1->pytorch_
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from hyperopt<0.2,>=0.1->pytorch_w
Requirement already satisfied: networkx in /usr/local/lib/python3.7/dist-packages (from hyperopt<0.2,>=0.1->pytorc
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-learn<1,>=0.19-
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from torch<2,>=1->pyto
Requirement already satisfied: seaborn in /usr/local/lib/python3.7/dist-packages (0.11.2)
Requirement already satisfied: matplotlib>=2.2 in /usr/local/lib/python3.7/dist-packages (from seaborn) (3.2.2)
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-packages (from seaborn) (1.19.5)
Requirement already satisfied: scipy>=1.0 in /usr/local/lib/python3.7/dist-packages (from seaborn) (1.4.1)
Requirement already satisfied: pandas>=0.23 in /usr/local/lib/python3.7/dist-packages (from seaborn) (1.1.5)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=2.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=2.2->
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages 
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib>=2.2->seabo
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from cycler>=0.10->matplotlib>=2.2->
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from pandas>=0.23->seaborn)
Wed Oct 27 09:29:07 2021       
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.74       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   72C    P0    75W / 149W |   9740MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
import datetime
import os
import random
import time
import unicodedata

import numpy as np
import pandas as pd
import torch
import matplotlib
import matplotlib.pyplot as plt
import transformers
from transformers import AutoTokenizer, AutoModel, get_linear_schedule_with_warmup
import pytorch_wrapper.functional as pwF
import sklearn
import seaborn


plt.rcParams['figure.figsize'] = [20, 5]

print("Seaborn version: %s" % (seaborn.__version__))
print("Sklearn version: %s" % (sklearn.__version__))
print("Transformers version: %s" % (transformers.__version__))
print("PyTorch version: %s" % (torch.__version__))
print("Matplotlib version: %s" % (matplotlib.__version__))
print("NumPy version: %s" % (np.__version__))
print("Pandas version: %s" % (pd.__version__))
```

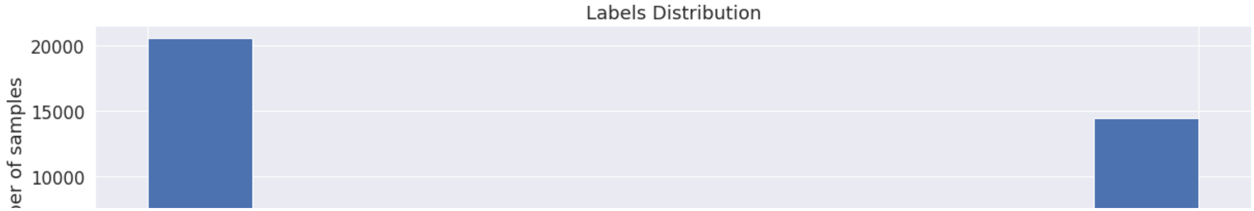
```
Seaborn version: 0.11.2
Sklearn version: 0.22.2.post1
Transformers version: 4.11.3
PyTorch version: 1.9.0+cu111
Matplotlib version: 3.2.2
NumPy version: 1.19.5
Pandas version: 1.1.5
```

```
df = pd.read_csv("clean_dataset.csv")
df.info()
```

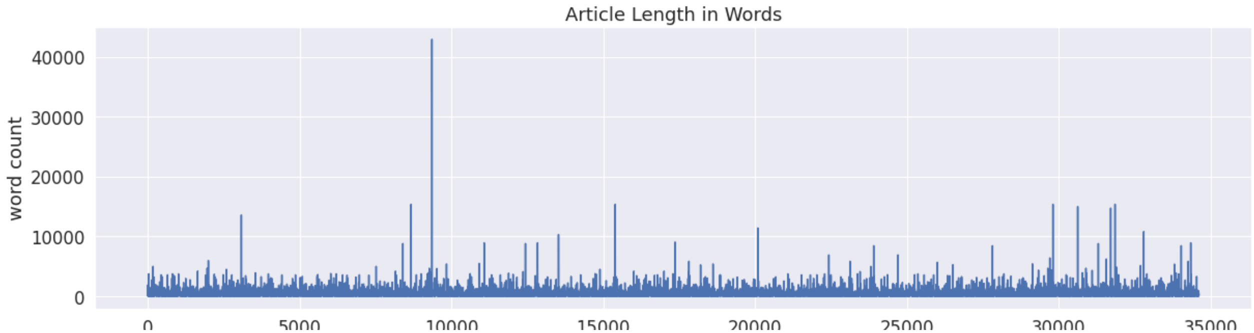
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 34976 entries, 0 to 34975
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   id      34976 non-null  int64  
 1   title   34970 non-null  object 
 2   author  0 non-null      float64
 3   text    34611 non-null  object 
 4   label   34976 non-null  int64  
dtypes: float64(1), int64(2), object(2)
memory usage: 1.3+ MB
```

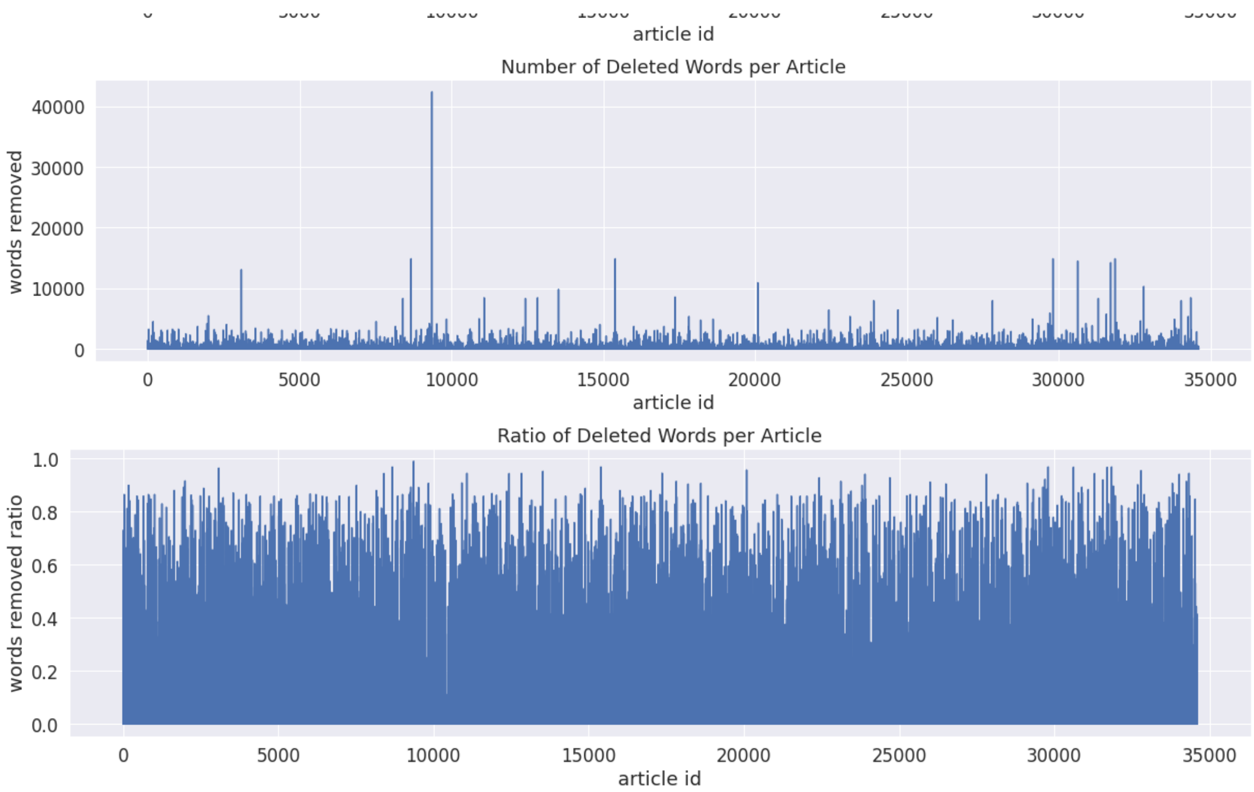
```
#plot labels
labels = df.label
plt.hist(labels)
plt.xlabel("label")
plt.ylabel("number of samples")
plt.title("Labels Distribution")
plt.xticks(np.arange(len(np.unique(labels))))
```

```
([<matplotlib.axis.XTick at 0x7fd14a4229d0>,
  <matplotlib.axis.XTick at 0x7fd14a422990>],
 <a list of 2 Text major ticklabel objects>)
```



```
# plot article lengths and discarded data length
lengths = pd.DataFrame([len(x.split()) for x in df.title + df.text if isinstance(x, str)])
lost_data = pd.DataFrame([max(len(x.split())-512,0) for x in df.title + df.text if isinstance(x, str)])
lost_data_ratios = pd.DataFrame([max(len(x.split())-512,0)/float(len(x.split())) for x in df.title + df.text if isinstanc

print("Article lengths description: %s\n" % (lengths.describe()))
lengths.plot(xlabel="article id", ylabel="word count", title="Article Length in Words", legend=None)
print("Article discarded data description: %s\n" % (lost_data.describe()))
lost_data.plot(xlabel="article id", ylabel="words removed", title="Number of Deleted Words per Article", legend=None)
print("Article discarded data ratios description: %s\n" % (lost_data_ratios.describe()))
lost_data_ratios.plot(xlabel="article id", ylabel="words removed ratio", title="Ratio of Deleted Words per Article", lege

s1 = lost_data.sum()
s2 = lengths.sum()
ratio = float(s1) / float(s2)
s3 = (lengths <= 512).astype(int).sum(axis=0)
ratio2 = float(s3) / float(len(lengths))
print("Articles exceeding 512 word limit: %d out of %d (%.2lf%%)" % (s3, len(lengths), ratio2*100))
print("Total data loss: %d out of %d words (%.2lf%% total, %.2lf%% average)" % (s1, s2, ratio*100, lost_data_ratios.mean(
Article lengths description:               0
count  34605.00
mean     347.27
std      551.12
min        5.00
25%      128.00
50%      227.00
75%      385.00
max    42891.00

Article discarded data description:               0
count  34605.00
mean      89.87
std      482.38
min        0.00
25%        0.00
50%        0.00
75%        0.00
max    42379.00

Article discarded data ratios description:               0
count  34605.00
mean       0.06
std        0.17
min        0.00
25%        0.00
50%        0.00
75%        0.00
max        0.99

Articles exceeding 512 word limit: 28942 out of 34605 (83.64%)
Total data loss: 3109972 out of 12017277 words (25.88% total, 6.04% average
```




0
5000
10000
15000
20000
25000
30000
35000
article id
Number of Deleted Words per Article
words removed
40000
30000
20000
10000
0
0
5000
10000
15000
20000
25000
30000
35000
article id
Ratio of Deleted Words per Article
words removed ratio
1.0
0.8
0.6
0.4
0.2
0.0
0
5000
10000
15000
20000
25000
30000
35000
article id

Part 2: Tokenization and Input Formatting

```
start = time.time()

def strip_accents_and_lowercase(s):
   return ''.join(c for c in unicodedata.normalize('NFD', s)
                  if unicodedata.category(c) != 'Mn').lower()
################################################################################
class BinaryGreekBERTModel(torch.nn.Module):

    def __init__(self, bert_model, dp):
        super(BinaryGreekBERTModel, self).__init__()
        self._bert_model = bert_model
        self._dp = torch.nn.Dropout(dp)
        self._output_linear = torch.nn.Linear(768, 1)

    def forward(self, text, text_len):
        attention_mask = pwF.create_mask_from_length(text_len, text.shape[1])
        return self._output_linear(self._dp(self._bert_model(text, attention_mask=attention_mask)[0][:, 0, :]))
################################################################################

#from transformers import BertTokenizer
#tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
tokenizer = AutoTokenizer.from_pretrained("nlpaueb/bert-base-greek-uncased-v1")
greekbert_model = AutoModel.from_pretrained('nlpaueb/bert-base-greek-uncased-v1')
binary_greekbert_model = BinaryGreekBERTModel(greekbert_model, dp=0)

working_set_df = df.head(2000)

text_data = working_set_df.title + working_set_df.text
text_data.fillna('', inplace=True)
```

```
input_ids = []
attention_masks = []
for item in text_data:
    unaccented_string = strip_accents_and_lowercase(item)
    encoded_dict = tokenizer.encode_plus(unaccented_string,
                                         add_special_tokens=True,
                                         max_length=512,
                                         truncation=True,
                                         padding='max_length',
                                         #pad_to_max_length=True,
                                         return_attention_mask=True,
                                         return_tensors="pt")
    input_ids.append(encoded_dict["input_ids"])
    attention_masks.append(encoded_dict["attention_mask"])

input_ids = torch.cat(input_ids, dim=0)
attention_masks = torch.cat(attention_masks, dim=0)
labels = torch.tensor(working_set_df.label.values)

print("Time elapsed: %.1lf seconds." % (time.time() - start))
```

```
Some weights of the model checkpoint at nlpaueb/bert-base-greek-uncased-v1 were not used when initializing BertModel
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactl
Time elapsed: 9.6 seconds.
```

```
# split training and validation sets
from torch.utils.data import TensorDataset, random_split, DataLoader, RandomSampler, SequentialSampler

dataset = TensorDataset(input_ids, attention_masks, labels)
train_size = int(0.8 * len(dataset))
validation_size = int(0.1 * len(dataset))
test_size = int(0.1 * len(dataset))
train_dataset, validation_dataset, test_dataset = random_split(dataset, [train_size, validation_size, test_size])
print("Training size: %s, Validation size: %s Test size: %s" % (train_size, validation_size, test_size))
```

```
batch_size = 8
train_dataloader = DataLoader(train_dataset, sampler=RandomSampler(train_dataset), batch_size=batch_size)
validation_dataloader = DataLoader(validation_dataset, sampler=SequentialSampler(validation_dataset), batch_size=batch_si
test_dataloader = DataLoader(test_dataset, sampler=SequentialSampler(test_dataset), batch_size=batch_size)
```

```
    Training size: 1600, Validation size: 200 Test size: 200
```

Part 3: Training

```
start = time.time()

from transformers import BertForSequenceClassification, AdamW, BertConfig

model = BertForSequenceClassification.from_pretrained(
    "nlpaueb/bert-base-greek-uncased-v1",
    num_labels = 2,
    output_attentions = False,
    output_hidden_states = False,
)

device = None
if torch.cuda.is_available():
    device = torch.device("cuda:0")
    model.cuda()
else:
    device = torch.device("cpu")

############################################################
# Batch sizes: 16, 32
# Learning rate (Adam): 5e-5, 3e-5, 2e-5
# Epochs: 2, 3, 4
############################################################

optimizer = AdamW(model.parameters(), lr=2e-5, eps=1e-8)
```

```
epochs = 4
total_steps = len(train_dataloader) * epochs
scheduler = get_linear_schedule_with_warmup(optimizer,
                                            num_warmup_steps = 0, # Default value in run_glue.py
                                            num_training_steps = total_steps)

print("Time elapsed: %.2lf" % (time.time() - start))
!nvidia-smi
```

```
Some weights of the model checkpoint at nlpaueb/bert-base-greek-uncased-v1 were not used when initializing BertForSe
- This IS expected if you are initializing BertForSequenceClassification from the checkpoint of a model trained on a
- This IS NOT expected if you are initializing BertForSequenceClassification from the checkpoint of a model that you
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at nlpaueb/bert-base-gr
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Time elapsed: 1.81
Wed Oct 27 09:00:36 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.74       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   70C    P0    76W / 149W |   9740MiB / 11441MiB |     36%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Training Loop

```
start = time.time()

def flat_accuracy(preds, labels):
    """Function to calculate the accuracy of our predictions vs labels"""
    pred_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()
    return np.sum(pred_flat == labels_flat) / len(labels_flat)
###############################################################################

seed_val = 21
random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)

training_stats = []
total_t0 = time.time()

for epoch_i in range(0, epochs):
    print('Epoch %s out of %s' % (epoch_i + 1, epochs))
    print('Training...')

    t0 = time.time()
    total_train_loss = 0
    model.train()
    for step, batch in enumerate(train_dataloader):
        if step % 40 == 0 and not step == 0:
            print('  Batch %s out  of  %s, Elapsed %.2lf seconds' % (step, len(train_dataloader), time.time() - t0))

        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)
        model.zero_grad()
        loss, logits = model(
            b_input_ids,
            token_type_ids=None,
            attention_mask=b_input_mask,
            labels=b_labels,
            return_dict=False
        )

        total_train_loss += loss.item()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()
    avg_train_loss = total_train_loss / len(train_dataloader)
    training_time = time.time() - t0

    print("  Average training loss: %.2lf" % (avg_train_loss))
    print("  Epoch %s training time %.2lf seconds" % (epoch_i, training_time))
    print("Running Validation...")

    t0 = time.time()
    model.eval()
    total_eval_accuracy = 0
    total_eval_loss = 0
    nb_eval_steps = 0
    for batch in validation_dataloader:
        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)
        with torch.no_grad():
            loss, logits = model(
                b_input_ids,
                token_type_ids=None,
                attention_mask=b_input_mask,
                labels=b_labels,
                return_dict=False
            )

        total_eval_loss += loss.item()
        logits = logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()
    total_eval_accuracy += flat_accuracy(logits, label_ids)
avg_val_accuracy = total_eval_accuracy / len(validation_dataloader)
avg_val_loss = total_eval_loss / len(validation_dataloader)
validation_time = time.time() - t0

print("  Accuracy: %.2lf" % (avg_val_accuracy))
print("  Validation Loss: %2lf" % (avg_val_loss))
print("  Validation time %.2lf seconds" % (validation_time))

training_stats.append(
    {
        'epoch': epoch_i + 1,
        'Training Loss': avg_train_loss,
        'Valid. Loss': avg_val_loss,
        'Valid. Accur.': avg_val_accuracy,
        'Training Time': training_time,
        'Validation Time': validation_time
    }
)
print("Training complete!")
print("Time elapsed: %.2lf seconds" % (time.time() - start))
```

```
Epoch 1 out of 4
Training...
  Batch 40 out  of  200, Elapsed 62.23 seconds
  Batch 80 out  of  200, Elapsed 124.20 seconds
  Batch 120 out  of  200, Elapsed 186.11 seconds
  Batch 160 out  of  200, Elapsed 248.16 seconds
  Average training loss: 0.37
  Epoch 0 training time 310.11 seconds
Running Validation...
  Accuracy: 0.91
  Validation Loss: 0.308900
  Validation time 13.87 seconds
Epoch 2 out of 4
Training...
  Batch 40 out  of  200, Elapsed 61.84 seconds
  Batch 80 out  of  200, Elapsed 123.63 seconds
```

```
      Batch 120 out  of  200, Elapsed 185.44 seconds
      Batch 160 out  of  200, Elapsed 247.28 seconds
      Average training loss: 0.15
      Epoch 1 training time 309.16 seconds
    Running Validation...
      Accuracy: 0.91
      Validation Loss: 0.349148
      Validation time 13.87 seconds
    Epoch 3 out of 4
    Training...
      Batch 40 out  of  200, Elapsed 61.68 seconds
      Batch 80 out  of  200, Elapsed 123.35 seconds
      Batch 120 out  of  200, Elapsed 185.04 seconds
      Batch 160 out  of  200, Elapsed 246.73 seconds
      Average training loss: 0.06
      Epoch 2 training time 308.48 seconds
    Running Validation...
      Accuracy: 0.92
      Validation Loss: 0.479133
      Validation time 13.86 seconds
    Epoch 4 out of 4
    Training...
      Batch 40 out  of  200, Elapsed 61.71 seconds
      Batch 80 out  of  200, Elapsed 123.35 seconds
      Batch 120 out  of  200, Elapsed 184.84 seconds
      Batch 160 out  of  200, Elapsed 246.53 seconds
      Average training loss: 0.02
      Epoch 3 training time 308.13 seconds
    Running Validation...
      Accuracy: 0.91
      Validation Loss: 0.500599
      Validation time 13.88 seconds
    Training complete!
    Time elapsed: 1291.37 seconds
```

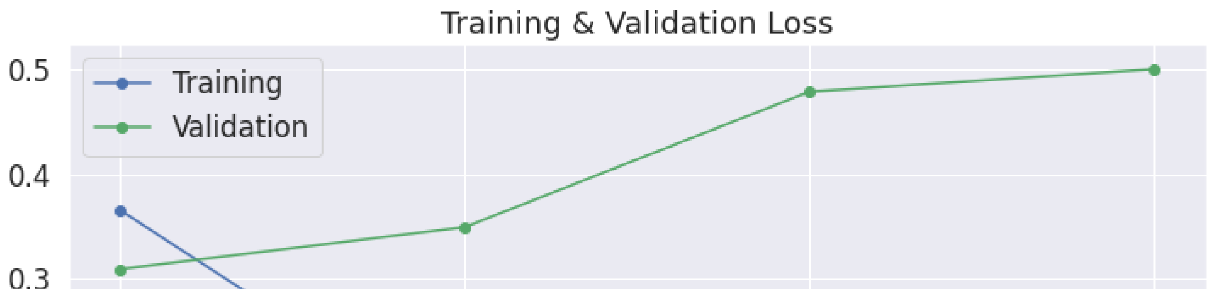
```
!nvidia-smi
```

```
    Wed Oct 27 09:25:45 2021
    +-----------------------------------------------------------------------------+
    | NVIDIA-SMI 470.74       Driver Version: 460.32.03    CUDA Version: 11.2     |
    |-------------------------------+----------------------+----------------------+
    | GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
    | Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
    |                               |                      |               MIG M. |
    |===============================+======================+======================|
    |   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
    | N/A   74C    P0    76W / 149W |   9740MiB / 11441MiB |      0%      Default |
    |                               |                      |                  N/A |
    +-------------------------------+----------------------+----------------------+

    +-----------------------------------------------------------------------------+
    | Processes:                                                                  |
    |  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
    |        ID   ID                                                   Usage      |
    |=============================================================================|
    |  No running processes found                                                 |
    +-----------------------------------------------------------------------------+
```

```
##############################
# View training summary
##############################

pd.set_option('precision', 2)
df_stats = pd.DataFrame(data=training_stats)
df_stats = df_stats.set_index('epoch')
df_stats
```

epoch	Training Loss	Valid. Loss	Valid. Accur.	Training Time	Validation Time
1	0.37	0.31	0.91	310.11	13.87
2	0.15	0.35	0.91	309.16	13.87
3	0.06	0.48	0.92	308.48	13.86
4	0.02	0.50	0.91	308.13	13.88

```
# If validation loss is increasing, it means we are overfitting!
# Plot training and validation loss
import seaborn as sns

# Use plot styling from seaborn.
sns.set(style='darkgrid')

# Increase the plot size and font size.
sns.set(font_scale=1.5)
plt.rcParams["figure.figsize"] = (12,6)

# Plot the learning curve.
plt.plot(df_stats['Training Loss'], 'b-o', label="Training")
plt.plot(df_stats['Valid. Loss'], 'g-o', label="Validation")

# Label the plot.
plt.title("Training & Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.xticks([1, 2, 3, 4])

plt.show()
```



```
####################
# Save model to disk
####################
tstamp = datetime.datetime.now().strftime("%Y-%m-%d_%H%M%S")
output_dir = './greekbert_model_save/%s/' % (tstamp)
if not os.path.exists(output_dir):
    os.makedirs(output_dir)
print("Saving model to %s" % output_dir)
model_to_save = model.module if hasattr(model, 'module') else model  # Take care of distributed/parallel training
model_to_save.save_pretrained(output_dir)
tokenizer.save_pretrained(output_dir)
torch.save(optimizer.state_dict(), os.path.join(output_dir, 'training_args.bin'))
```

```
    Saving model to ./greekbert_model_save/2021-10-27_092610/
```

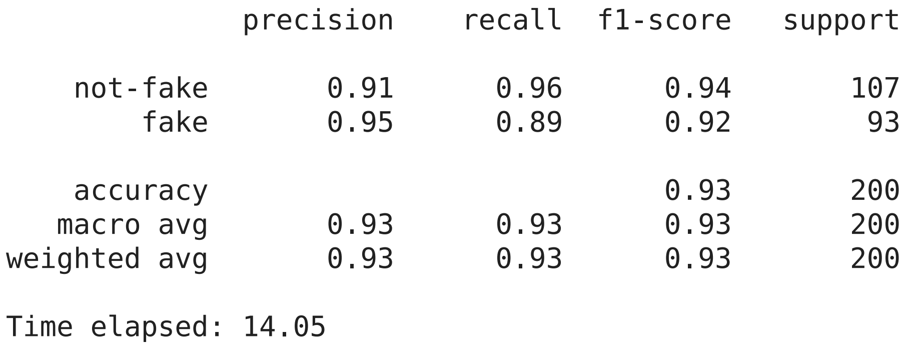
```
!ls -lhatrR greekbert_model_save/
```

```
    greekbert_model_save/:
    total 16K
    drwxr-xr-x 1 root root 4.0K Oct 27 08:39 ..
    drwxr-xr-x 2 root root 4.0K Oct 27 08:39 2021-10-27_083917
    drwxr-xr-x 4 root root 4.0K Oct 27 09:26 .
    drwxr-xr-x 2 root root 4.0K Oct 27 09:26 2021-10-27_092610

    greekbert_model_save/2021-10-27_083917:
    total 1.3G
    -rw-r--r-- 1 root root  732 Oct 27 08:39 config.json
    -rw-r--r-- 1 root root  526 Oct 27 08:39 tokenizer_config.json
    -rw-r--r-- 1 root root  112 Oct 27 08:39 special_tokens_map.json
    -rw-r--r-- 1 root root 431M Oct 27 08:39 pytorch_model.bin
```

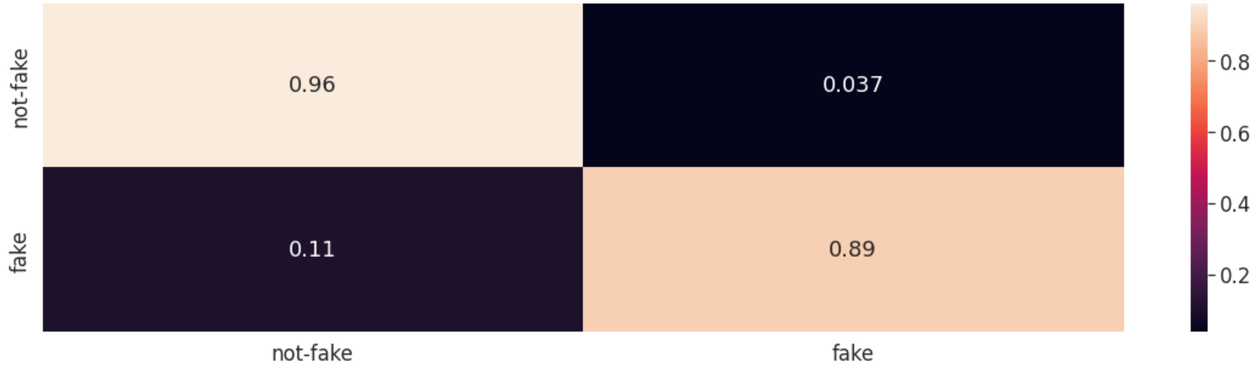
```
-rw-r--r-- 1 root root 518K Oct 27 08:39 vocab.txt
-rw-r--r-- 1 root root 782K Oct 27 08:39 tokenizer.json
drwxr-xr-x 2 root root 4.0K Oct 27 08:39 .
-rw-r--r-- 1 root root 862M Oct 27 08:39 training_args.bin
drwxr-xr-x 4 root root 4.0K Oct 27 09:26 ..

greekbert_model_save/2021-10-27_092610:
total 1.3G
drwxr-xr-x 4 root root 4.0K Oct 27 09:26 ..
-rw-r--r-- 1 root root  732 Oct 27 09:26 config.json
-rw-r--r-- 1 root root 431M Oct 27 09:26 pytorch_model.bin
-rw-r--r-- 1 root root  526 Oct 27 09:26 tokenizer_config.json
-rw-r--r-- 1 root root  112 Oct 27 09:26 special_tokens_map.json
-rw-r--r-- 1 root root 518K Oct 27 09:26 vocab.txt
-rw-r--r-- 1 root root 782K Oct 27 09:26 tokenizer.json
drwxr-xr-x 2 root root 4.0K Oct 27 09:26 .
-rw-r--r-- 1 root root 862M Oct 27 09:26 training_args.bin
```

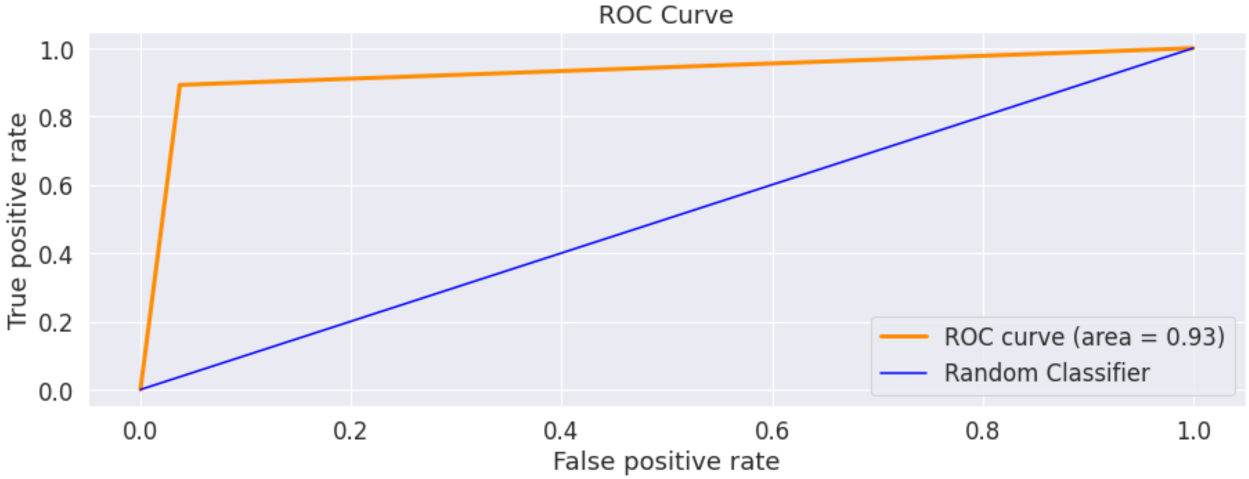
Part 4: Test

```
start = time.time()

epoch_loss = 0
epoch_acc = 0
predicted_labels = []
labels = []
model.eval()
for batch in test_dataloader:
    b_input_ids = batch[0].to(device)
    b_input_mask = batch[1].to(device)
    b_labels = batch[2].to(device)
    with torch.no_grad():
        loss, logits = model(
            b_input_ids,
            token_type_ids=None,
            attention mask=b input mask,
            labels=b_labels,
            return_dict=False
        )
    epoch_loss += loss.item()
    logits = logits.detach().cpu().numpy()
    label_ids = b_labels.to('cpu').numpy()
    epoch_acc += flat_accuracy(logits, label_ids)
    predicted_labels.extend(np.argmax(logits, axis=1).flatten())
    labels.extend(label_ids)
    test_loss = epoch_loss / len(test_dataloader)
    test_acc = epoch_acc / len(test_dataloader)
    test_labels = labels
    test_predictions = predicted_labels

print(sklearn.metrics.classification_report(test_labels, test_predictions, target_names=["not-fake", "fake"]))
cm = sklearn.metrics.confusion_matrix(test_labels, test_predictions, normalize='true')
df_cm = pd.DataFrame(cm, index=["not-fake", "fake"], columns=["not-fake", "fake"])
seaborn.heatmap(df_cm, annot=True)
fpr, tpr, thresholds = sklearn.metrics.roc_curve(test_labels, predicted_labels)
auc = sklearn.metrics.auc(fpr, tpr)
plt.figure(figsize=(15, 5))
plt.plot(fpr, tpr, lw=3, color='darkorange', label="ROC curve (area = %.2lf)" % (auc))
plt.plot([0, 1], [0, 1], color="blue", label="Random Classifier")
plt.xlabel("False positive rate")
plt.ylabel("True positive rate")
plt.title("ROC Curve")
plt.legend()

print("Time elapsed: %.2lf" % (time.time() - start))
```



```
              precision    recall  f1-score   support

    not-fake       0.91      0.96      0.94       107
        fake       0.95      0.89      0.92        93

    accuracy                           0.93       200
   macro avg       0.93      0.93      0.93       200
weighted avg       0.93      0.93      0.93       200

Time elapsed: 14.05
```





✓ 15s completed at 12:54 PM ● ✕